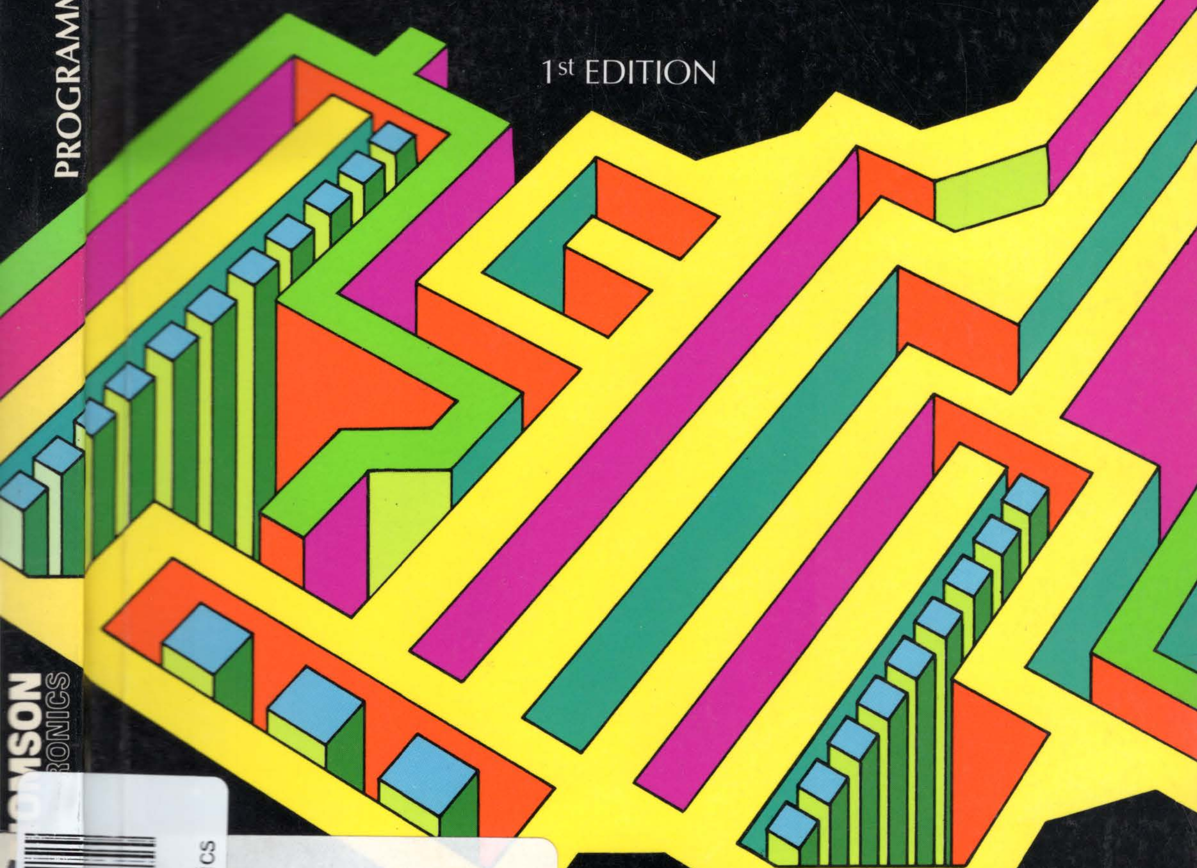
GAL PROGRAMMABLELOGIC DEVICES

SGS-THOMSON MICROELECTRONICS

GAL PROGRAMMABLE LOGIC DEVICES

DATABOOK

1st EDITION



SGS-THOMSON
MICROELECTRONICS

GAL
PROGRAMMABLE LOGIC DEVICES

DATABOOK

1st EDITION

MAY 1992

USE IN LIFE SUPPORT DEVICES OR SYSTEMS MUST BE EXPRESSLY AUTHORIZED.

SGS-THOMSON PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS OF LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF SGS-THOMSON Microelectronics.

1. Life support devices or systems are those which (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided with the product, can be reasonably expected to result in significant injury to the user.

2. A critical component is any component of a life support device or system whose failure to perform can reasonably be expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

TABLE OF CONTENTS

GENERAL INDEX

Page

INTRODUCTION

DATASHEETS

ST-CUPL DOCUMENTATION

LOGIC CONCEPT

DEVELOPMENT TOOLS

POWER CONSUMPTION

INTRODUCTION

INTRODUCTION

GAL® IN PLD SCENARIO

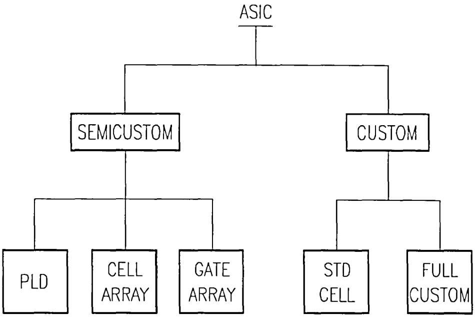
SGS-THOMSON is today committed to serving the market with high volume families including Programmable Logic Devices (PLDs), which shows one of the fastest growth rates in the semiconductor market.

PLDs IN THE ASIC SCENARIO

PLDs represent today one of the largest families in the ASIC market.

The devices in the ASIC category offer advantages over other alternatives, in that they perform a function defined by the user that is optimized for a specific application. The ASIC family can be segmented into sub-families of devices — according to the degree of physical specialization that the customer design imprints into the device — as shown in Figure 1.

Figure 1. The ASIC scenario



The standard cell approach uses pre-configured, pre-tested and pre-characterized logic blocks to construct a custom silicon chip for the designer. The chip is usually designed by hand, using a graphics terminal. The outcome is a fairly efficient logic design that may take weeks or months to complete, while incurring a hefty up-front engineering charge, or non-recurring expense (NRE).

In addition, the custom piece of silicon will take additional weeks, if not months, to manufacture. The development time and complexity of a standard cell design severely impacts the ability to incorporate changes or corrections to the design. In addition, the customer is typically obligated for some minimum production lot size to cover the manufacturer's expenses. Since most logic designs are subject to revision during the debug phase, the time and dollar penalties of the standard cell approach make this a relatively high risk one.

The gate array approach has gained extensive market recognition as a more optimum "bridge" between the standard cell (full custom) and programmable alternatives. The gate array is a pre-manufactured silicon matrix that awaits only a custom interconnected pattern to establish functionality. The designer can choose from NAND gates, flip-flops, and various types of buffers to construct the logic.

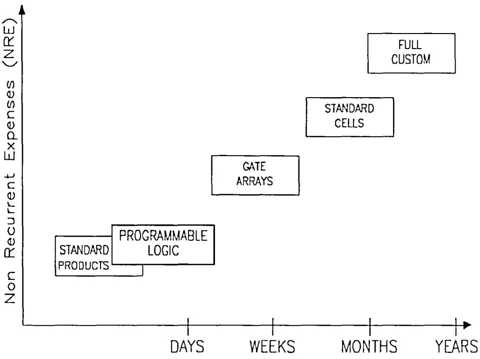
The flexibility of a gate array is less than that of a standard cell device, since the user must interconnect existing structures. However, since the device usually has many pins (68 or more) it offers greater logic functionality than a typical PLD. This increased functionality, however, comes at the penalty of lower speed performance.

Since the gate array relies on only one or two custom mask layers, the wafer fabrication can be done much more quickly than with a standard cell. The turn time from design completion to final chip for a gate array is, at best, 4 to 8 weeks. Although not as costly up front as a standard cell, there is still an up front development cost, a minimum lot size, and a long and costly cycle for logic changes. These features make this, too, a risky approach.

Advantages of PLDs

The previous alternatives are ideal choices for high volume applications where, once debugged, the design is not subject to change. The average customer, however, uses hundreds to thousands of devices of a given logic pattern, each year. He cannot afford the NRE of a gate array or standard cell, since the volume is not high enough to dilute this expense. Figure 2 shows the cost and development time relationships of the various design alternatives.

Figure 2. Costs vs Development Time



The PLD offers a solution to these woes. The simple, but powerful and affordable development tools, associated with the low unit cost, flexibility, high performance, and proven reliability of these devices result in a cost-efficient, higher-performance, lower-risk, and more timely design cycle.

Programmable logic is ideal for simplifying the design process, because the designer can implement the exact logic function whenever and wherever required. Programmable logic offers more efficient utilization, as well as reduced chip count, by simplifying the lay-out process at both conceptual and implementation stage.

As a consequence PLDs contributes to increase substantially system reliability. It has been statistically demonstrated that systems with higher levels of integration such as those designed with programmable logic, have much higher reliability than equivalent systems designed with many low density standard components.

GAL®

SGS-THOMSON has decided to enter the PLD market by producing the GAL® (Generic Array Logic) family of devices. GAL® are ideal devices among PLDs for several reasons:

- GAL® devices are fabricated using very high speed Electrically Erasable CMOS technology which offers the highest degree of testability and quality of any process technology. In fact AC, DC and functionality can be 100% tested and that guarantees 100% programming and functional yield to the customer with no further board rework. With PLDs in bipolar technologies, complete testing is not possible and manufacturers must rely on complex schemes using test rows and columns to simulate and correlate device performances, since the fuse array cannot be tested prior to programming. Due to the incomplete test of these bipolar PLDs, rejects may be found when the user programs them, with no recovery possibility since mis-programmed parts must be discarded. Instant erasability, instead, makes GAL® ideal for unforecasted design changes.
- GAL® switching speed is as fast as any other bipolar programmable logic devices except ECL, but they have the low power consumption of CMOS.
- GAL® devices utilize the Output Logic Macrocell (OLMC) which allows the user to configure outputs as needed and to replace several other programmable logic devices and low complexity gate arrays.

The main advantage of GAL® devices comes from their intrinsic "genericity" that allows the user to define the architecture and functionality of each output and also has advantages at the shop floor level: users can put in inventory one generic GAL® type instead of many different PAL® device types; this will not only save money, but also minimize the paper work, reduce manufacturing flow because the handling process is simplified, reduce the risk or running out of inventory. For example, the GAL16V8AS can replace 21 different bipolar PAL®.

GAL® Development Tools

GAL® devices have been developed to support the philosophy that users should not be required to purchase special development tools. GAL® are in fact supported by existing programmable logic development tools and device programmers.

Software packages such as ABEL® from DATA I/O and CUPL™ from Logical Devices, offer generic development support for all programmable logic devices.
They allow an almost instantaneous compilation of a description of the logic circuit that the GAL® is expected to implement.
Their output is a file (the JEDEC file) that can be fed into the hardware programming tool (the device programmer), which in turn performs the task of writing the corresponding pattern into the GAL® memory.
SGS-THOMSON, together with Logical Devices, supplies a dedicated GAL® high level software development tool, named ST-CUPL™.

GAL® are supported by several device programmers that ensure the highest quality.
An updated list of device programmers, qualified by SGS-THOMSON for its GAL®, may be obtained from the nearest SGS-THOMSON sales office.

DC PARAMETERS

Symbol	Parameter	Description
V_{CC}	Supply Voltage	The range of power supply voltage over which the device is guarantied to operate within specified limits.
V_{IH}	Input HIGH Voltage	The range of input voltages that represents a logic HIGH in the system.
V_{IL}	Input LOW Voltage	The range of input voltages that represents a logic LOW in the system.
V_{OH}	Output HIGH Voltage (min.)	The minimum voltage at an output terminal for the specified output current I_{OH} and the minimum value of V_{CC}.
V_{OL}	Output LOW Voltage (max.)	The maximum voltage at an output terminal sinking the maximum specified LOW current I_{OL}.
I_{CC}	Supply Current	The current flowing into the VCC supply terminal of a circuit with the specified inputs conditions and the outputs open. When not specified, input conditions are chosen to guarantee worst case operation.
I_{IH}	Input Leakage Current (High)	The current flowing into an input when a specified HIGH level voltage is applied to that input.
I_{IL}	Input Leakage Current (Low)	The current flowing out of an input when a specified LOW voltage is applied to the input.
I_{OH}	Output High Current	The current flowing out of an output which is in the HIGH state.
I_{OL}	Output Low Current	The current flowing into an output which is in the LOW state.
I_{OS}	Output Short Circuit Current	The current flowing out of an output which is in the HIGH state when that output is connected to a reference of 0.5 V.
I_{BH}	Bidirectional pin Leakage Current HIGH	The current flowing into a disabled 3-state output with a specified HIGH output voltage applied.
I_{BL}	Bidirectional pin Leakage Current LOW	The current flowing out of a disabled 3-state output with a specified LOW output voltage applied.

AC PARAMETERS

Symbol	Parameter	Description
f_{clk}, f_{clkf}	Clock Frequency without and with Feedback	The maximum input frequency at a clock input for predictable performance. Above this frequency the device may cease to function.
t_{pd}, t_{co}	Combinational Propagation Delay and Clock to Output Delay	The time between the specified reference points on the input and output waveforms with the output changing from the defined LOW level to the defined HIGH level or vice versa.
t_{dis}, t_{disr}	Product Term Output Disable to Output and Output Register Disable to Output	The delay time between the specified reference points on the input and output voltage waveforms with the 3-state output changing from the HIGH (or LOW) level to a high impedance "off" state.
t_{en}, t_{enr}	Product Term Output Enable to Output and Output Register Enable to Output	The delay time between the specified reference points on the input and output voltage waveforms with the 3-state output changing from a high impedance "off" state to the HIGH (or LOW) level.
t_h	Input or Feedback Hold Time (after Clock Rise)	The interval immediately following the active transition of the timing pulse (usually the clock pulse) or following the transition of the control input to its latching level, during which interval the data to be recognized must be maintained at the input to ensure its continued recognition. A negative set-up time indicates that the correct logic level may be released prior to the active transition of the timing pulse and still be recognized.
t_{su}	Input or Feedback Set-up Time (before Clock Rise)	The interval immediately preceding the active transition of the timing pulse (usually the clock pulse) or preceding the transition of the control input to its latching level, during which interval the data to be recognized must be maintained at the input to ensure its recognition. A negative set-up time indicates that the correct logic level may be initiated sometimes after the active transition of the timing pulse and still be recognized.
t_{wh}, t_{wl}	Minimum Clock Width, High and Low	The time between the specified reference points on the leading and trailing edges of a pulse.

GAL DATASHEETS

SGS-THOMSON
MICROELECTRONICS

GAL6001S

E^{2}PROM CMOS PROGRAMMABLE LOGIC DEVICE

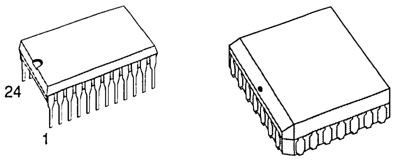
- ELECTRICALLY ERASABLE CELL TECHNOLOGY
  - Instantly reconfigurable logic
  - Instantly reprogrammable cells
  - Guaranteed 100% yields
- HIGH PERFORMANCE E^{2}CMOS TECHNOLOGY
  - Low power: 90mA typical
  - High speed: 12ns max. clock to output delay, 25ns max. setup time, 30ns max. propagation delay
- TTL COMPATIBLE INPUTS AND OUTPUTS
- UNPRECEDENTED FUNCTIONAL DENSITY
  - 10 Output Logic Macrocells
  - 8 Buried Logic Macrocells
  - 20 Input and I/O Logic Macrocells
- HIGH-LEVEL DESIGN FLEXIBILITY
  - 78 x 64 x 36 FPLA Architecture
  - Separate buried register and input clock pins
  - Functionally supersets existing 24 pin PAL® and IFL™ devices
  - Asynchronous or Synchronous clocking
- SPACE SAVING 24 PINS, 300 MILS DIP
- HIGH SPEED PROGRAMMING ALGORITHM
- 20 YEAR DATA RETENTION

DESCRIPTION

Using a high performance E^{2}CMOS technology, SGS-THOMSON has produced a next-generation programmable logic device, the GAL6001S. Using FPLA architecture known for its superior flexibility in state machine design, the GAL6001S offers the highest degree of functional integration and flexibility currently available in a 24 pin, 300 mils package.

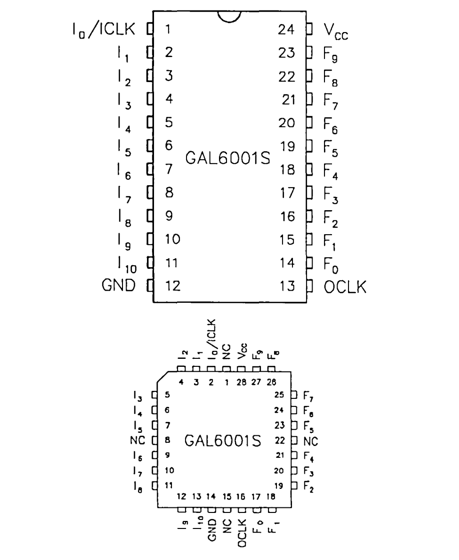
The GAL6001S has 10 programmable Output Logic Macrocells (OLMCs) and 8 programmable Buried Logic Macrocells (BLMCs). In addition, there are 10 Input Logic Macrocells (ILMCs) and 10 I/O Logic Macrocells (IOLMC). Two Clock inputs are provided for independent control of the Input and Output Macrocells.

Advanced features that simplify programming and reduce test time, coupled with E^{2}PROM CMOS reprogrammable cells, enable complete AC, DC, programmability, and functionality test of each GAL6001S during manufacture. This allows SGS-THOMSON to guarantee 100% field programmability and functionality to datasheet specifications.



B PDIP24 **C** PLCC28

Pin Connections

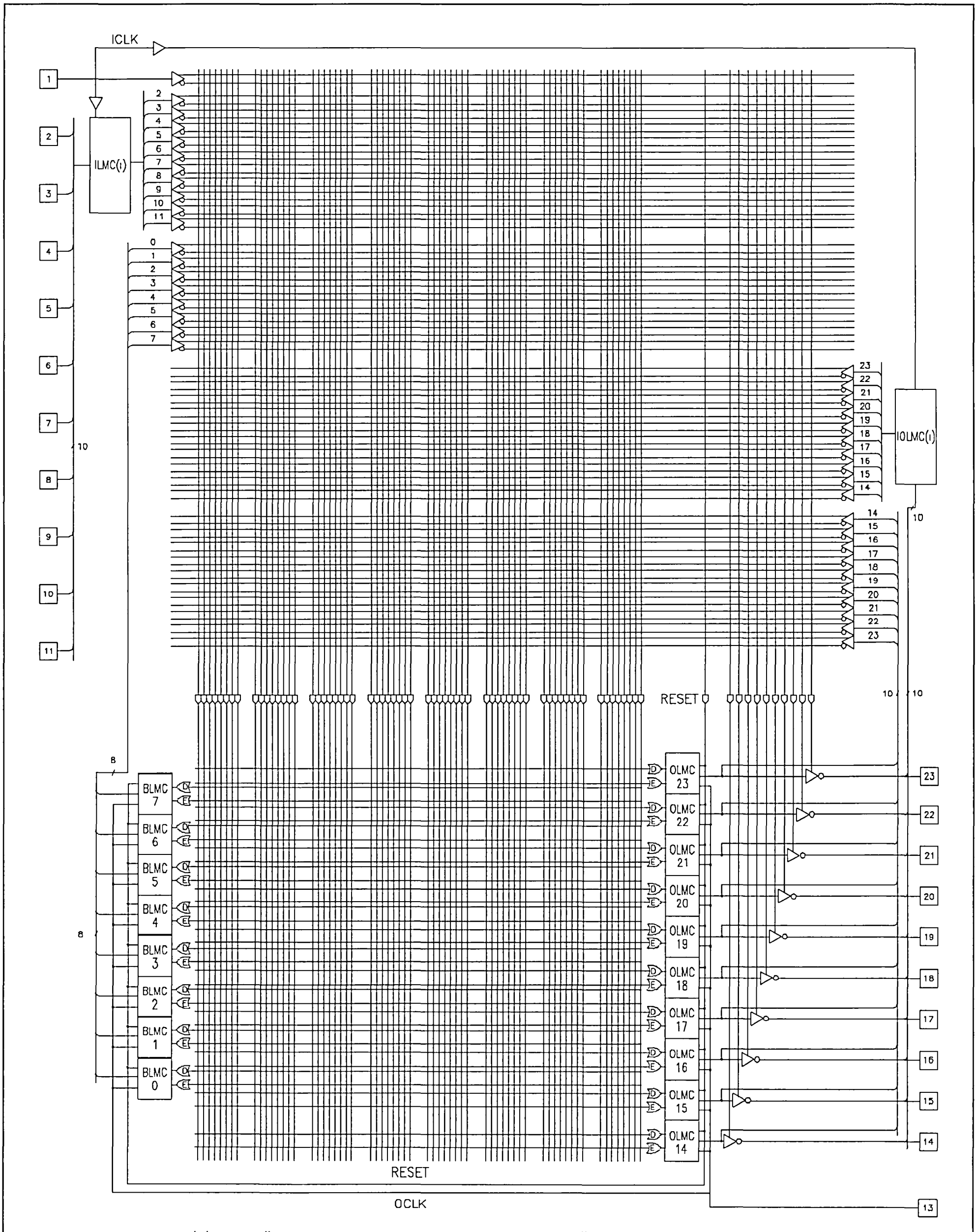


Pin Names

I0–I10	Input
F0–F9	I/O
ICLK	Input Clock
OCLK	Output Clock
VCC	Power
GND	Ground

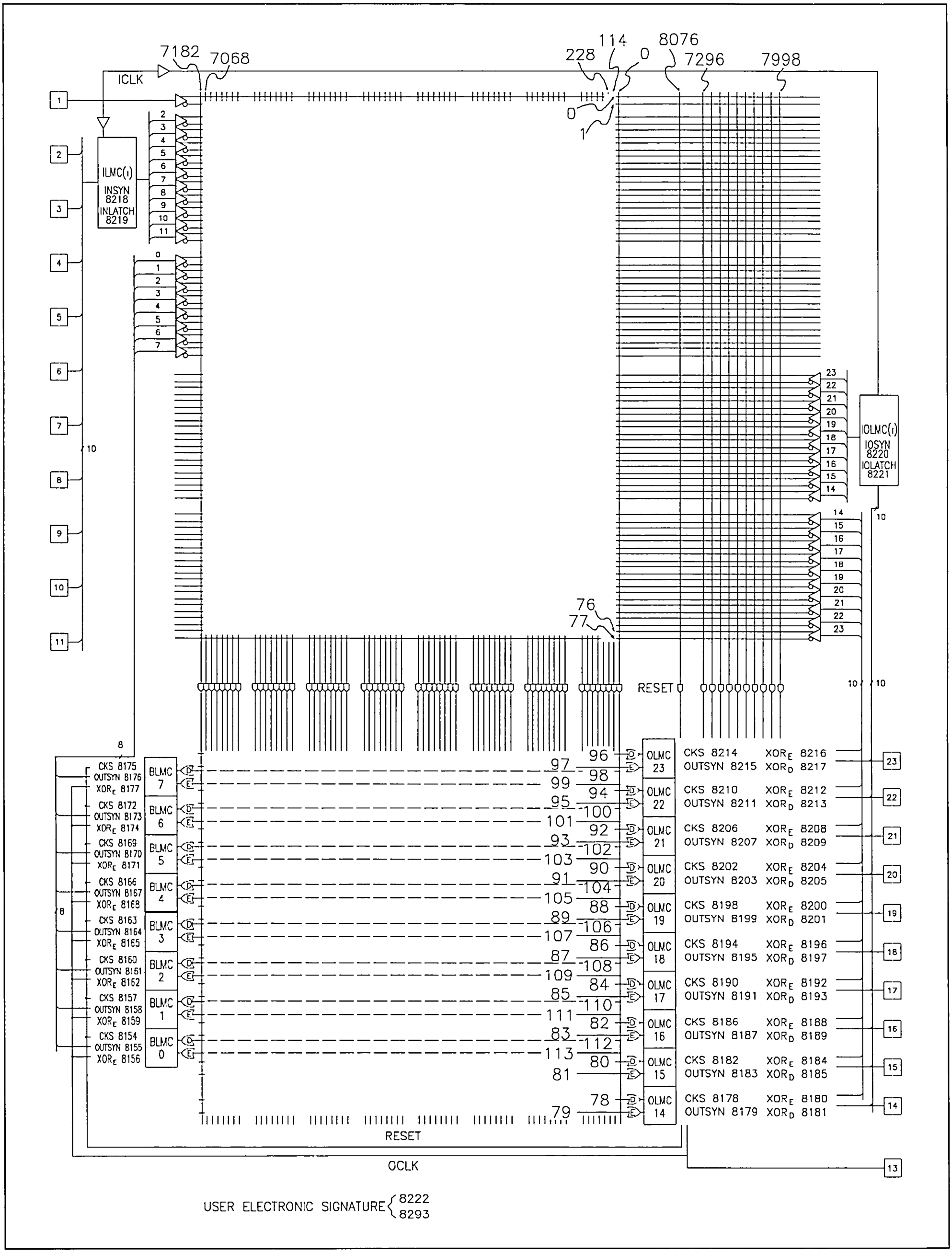
GAL® is a registered trademark of Lattice Semiconductor Corp.; PAL® is a registered trademark of Monolithic Memories Inc.

GAL6001S Logic Diagram



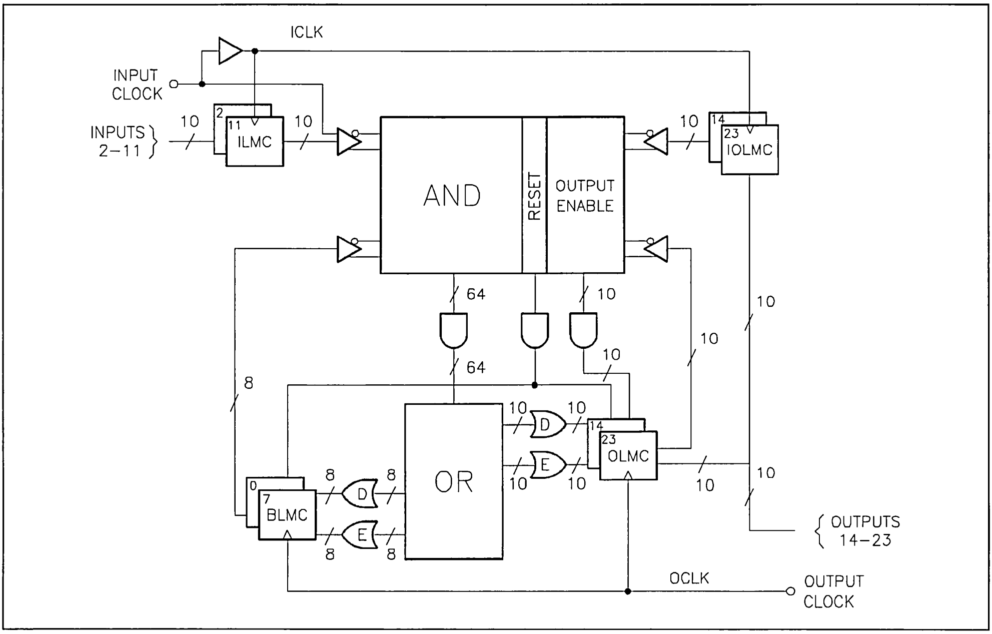
(*) See "Differential Product Term Switching" section

GAL6001S Jedec Map



Programming is accomplished using standard hardware and software tools. SGS-THOMSON guarantees a minimum of 100 erase write cycles, and data retention to exceed 20 years. An Electronic Signature word has been provided for user-defined data. In addition, a security cell is available to protect proprietary designs.

GAL6001S Functional Block Diagram



Macrocells Names

ILMC	Input Logic Macrocell
IOLMC	I/O Logic Macrocell
BLMC	Buried Logic Macrocell
OLMC	Output Logic Macrocell

Absolute Maximum Ratings

Symbol	Parameter	Value	Unit
V_{CC}	Supply Voltage	-0.5 to +7	V
V_I	Input Voltage Applied	-2.5 to V_{CC}+1	V
V_B	Off-State Output (Bidirectional) Voltage Applied	-2.5 to V_{CC}+1	V
T_{STG}	Storage Temperature	-65 to +125	˚C
T_J	Junction Temperature (Operating)	-40 to +125	˚C
T_L	Lead Temperature (Soldering)	260 (for 10s max.)	˚C

Stresses above those listed under the “Absolute Maximum Ratings” may cause permanent damage to the device. These are stress only ratings and functional operation of the device at these or at any other conditions above those indicated in the operational sections of this specification is not implied.

Switching Test Conditions

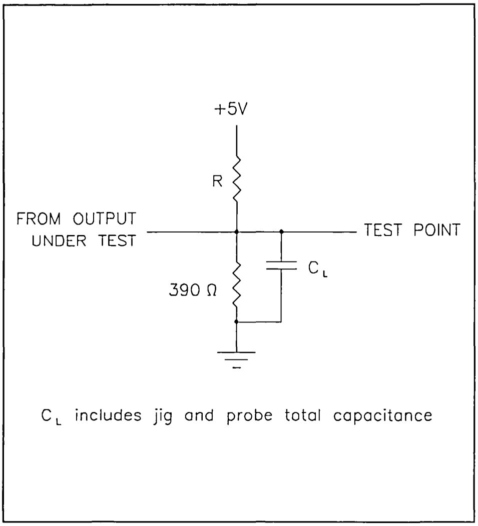
Input Pulse Levels	GND to 3.0V
Input Rise and Fall Times	3ns 10%-90%
Input Timing Reference Levels	1.5V
Output Timing Reference Levels	1.5V
Output Load	See figure

3-state levels are measured 0.5V from steady-state active level.

Test Conditions

#	R [Ω]	C_L [pF]
1	300	50
2	Active High: ∞ Active Low: 300	50
3	Active High: ∞ Active Low: 300	5

Switching Test Circuit



C_L includes jig and probe total capacitance

Capacitance (T_A=25°C, f=1.0MHz, V_{CC}=5V)

Symbol	Parameter	Test Conditions	Maximum⁘	Units
C_I	Input Capacitance	V_I=2V	8	pF
C_B	Bidirectional Pin Capacitance	V_B=2V	10	pF

⁘ Guarantied but not 100% tested

DC Operating Conditions

Symbol	Parameter	Commercial Temperature Range		Industrial Temperature Range		Units
		Min.	Max.	Min.	Max.	
V_{CC}	Supply Voltage	4.75	5.25	4.5	5.5	V
T_A	Ambient Temperature	0	70	-40	85	°C
V_{IL}	Input Low Voltage	V_{SS}⋄-0.5	0.8	V_{SS}⋄-0.5	0.8	V
V_{IH}	Input High Voltage	2.0	V_{CC}+1	2.0	V_{CC}+1	V
I_{OL}	Low Level Output Current	–	16	–	16	mA
I_{OH}	High Level Output Current	-3.2	–	-3.2	–	mA

⋄ V_{SS} is the voltage applied to the GND pin

Electrical Characteristics Over Operating Conditions

Symbol	Parameter	Test Conditions	Commercial Temperature Range		Industrial Temperature Range		Units
			Min.	Max.	Min.	Max.	
I_{IH}, I_{IL}	Input Leakage Current	GND≤V_I≤$V_{CC\ Max}$	–	±10	–	±10	µA
I_{BH}, I_{BL}	Bidirectional Pin Leakage Current	GND≤V_I≤$V_{CC\ Max}$	–	±10	–	±10	µA
I_{CC}	Operating Power Supply Current	f=15MHz V_{IL}=0.5V V_{IH}=3.0V	–	150	–	180	mA
I_{OS}⁘	Output Short Circuit Current	V_{CC}=5.0V, V_B=0.5V	-130	-30	-130	-30	mA
V_{OL}	Output Low Voltage	–	–	0.5	–	0.5	V
V_{OH}	Output High Voltage	–	2.4	–	2.4	–	V

⁘ One output at a time for a maximum duration of one second.

Switching Characteristics Over Operating Conditions

Symbol	Parameter	From	To	6001S-30	6001S-35	Units	Test Cond.✦
				Max.✧	Max.⁘		
t_{pda}	Combinational Propagation Delay (ILMC Async.)	Input	Output	30	35	ns	1
t_{pdf}	Combinational Propagation Delay (With Feedback)	Output, Registered Output	Output	30	35	ns	1
t_{pdl}	Combinational Propagation Delay (ILMC Latch)	Input	Output	35	40	ns	1
t_{coir}	Input Clock to Output Delay (ILMC Reg., OLMC Comb.)	ICLK	Output	35	40	ns	1
t_{coil}	Input Clock to Output Delay (ILMC Latch, OLMC Comb.)	ICLK	Output	35	40	ns	1
t_{coo}	Output Clock to Registered Output Delay (OLMC D/E Reg.)	OCLK	Registered Output	12	13.5	ns	1
t_{cos}	Sum Term Clock to Registered Output Delay (OLMC D Reg.)	STCLK	Registered Output	35	40	ns	1
t_{en}	Product Term Output Enable to Output Delay	Input, I/O	Output, Registered Output	25	30	ns	2
t_{dis}	Product Term Output Disable to Output Delay	Input, I/O	Output, Registered Output	25	30	ns	3
t_{res}	Register Reset Delay	Input, I/O	Registered Output	35	35	ns	1

✦ Refer to Switching Test Conditions”.

✧ Commercial Temperature range only

⁘ Industrial Temperature range only

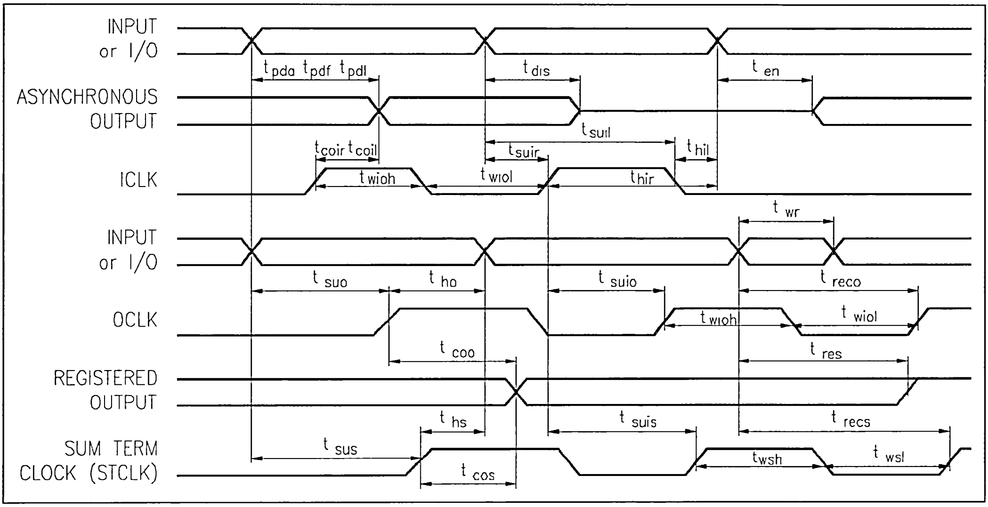
AC Operating Conditions

Symbol	Parameter	6001S-30*		6001S-35✢		Units
		Min.	Max.	Min.	Max.	
t_{suil}	Input Setup Time Before ICLK Fall (ILMC Latch)	–	2.5	–	3.5	ns
t_{suir}	Input Setup Time Before ICLK Rise (ILMC Reg.)	–	2.5	–	3.5	ns
t_{suo}	Input or Feedback Setup Time Before OCLK Rise (OLMC D/E Reg.)	–	25	–	30	ns
t_{sus}	Input or Feedback Setup Time Before STCLK Rise (OLMC D Reg.)	–	7.5	–	10	ns
t_{suio}	ICLK Rise Setup Time Before OCLK Rise (OLMC D/E Reg.)	–	30	–	35	ns
t_{suis}	ICLK Rise Setup Time Before STCLK Rise (OLMC D Reg.)	–	15	–	17	ns
t_{hil}	Hold Time After ICLK Fall (ILMC Latch)	–	5	–	5	ns
t_{hir}	Hold Time After ICLK Rise (ILMC Reg.)	–	5	–	5	ns
t_{ho}	Hold Time After OCLK Rise (OLMC D/E Reg.)	–	5	–	5	ns
t_{hs}	Hold Time After STCLK Rise (OLMC D Reg.)	–	10	–	12.5	ns
t_{wioh}	ICLK or OCLK Pulse Duration High	–	10	–	10	ns
t_{wiol}	ICLK or OCLK Pulse Duration Low	–	10	–	10	ns
t_{wsh}	STCLK Pulse Duration High	–	15	–	15	ns
t_{wsl}	STCLK Pulse Duration Low	–	15	–	15	ns
t_{wr}	Reset Pulse Duration	–	15	–	15	ns
t_{reco}	Reset to OCLK Recovery Time	–	20	–	20	ns
t_{recs}	Reset to STCLK Recovery Time	–	10	–	10	ns
f_{clk}	OCLK or STCLK Maximum Frequency	27	–	22.9	–	MHz

* Commercial Temperature range only

✢ Industrial Temperature range only

Switching Waveforms



INPUT LOGIC MACROCELL (ILMC) AND I/O LOGIC MACROCELL (IOLMC)

The GAL6001S features two configurable input sections.

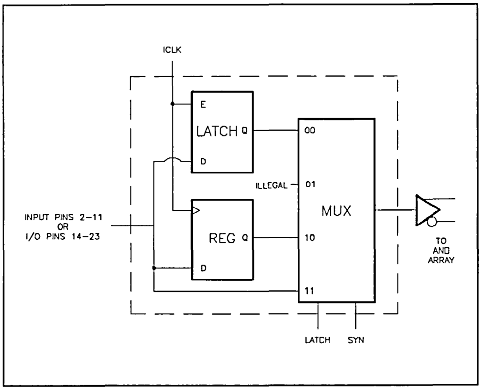
The ILMC section corresponds to the dedicated input pins (2-11) and the IOLMC section to the I/O pins (14-23). Each input section is configurable as a block for asynchronous, latched, or registered inputs. Pin 1 (ICLK) is used as an enable input for latched macrocells (transparent when high) and as a clock for registered macrocells (positive edge triggered).

Configurable input blocks can be used to advantage by system designers. Registered inputs are popular for synchronization and data merging. Transparent latches are useful when the input data is invalid outside a known time window. Direct inputs are used in systems where the input data is well ordered in time. With the GAL6001S, external registers and latches are not necessary.

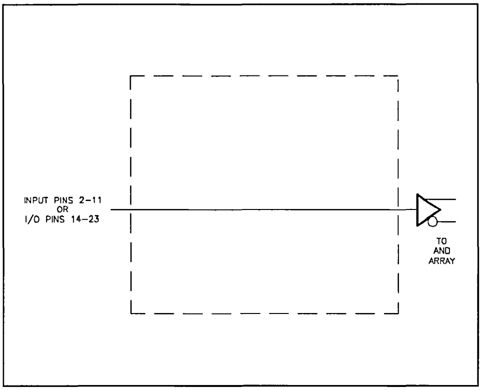
The various configurations of the Input and I/O Macrocells are controlled by programming four architecture control bits (LATCH and SYN both for Input and I/O Macrocells) within the 68 bits Architecture Control Word. The SYN bits determine whether the macrocells will have register/latch capability or will be strictly asynchronous. The LATCH bits select between latched and registered inputs.

The three valid macrocell configurations are shown in the macrocell equivalent diagrams shown below.

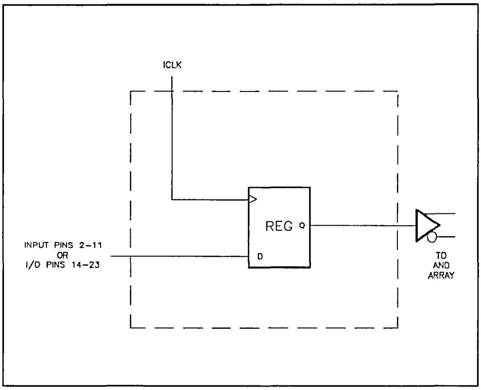
ILMC/IOLMC Generic Block Diagram



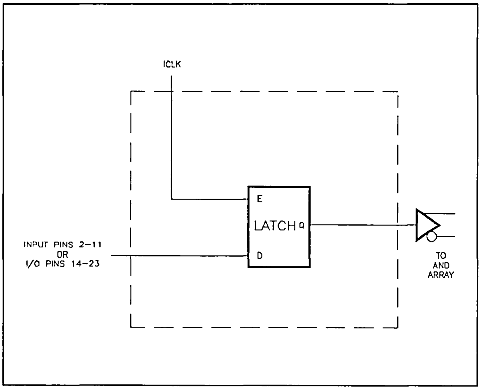
Asynchronous Input (LATCH=1, SYN=1)



Registered Input (LATCH=1, SYN=0)



Latched Input (LATCH=0, SYN=0)



OUTPUT LOGIC MACROCELL (OLMC) AND BURIED LOGIC MACROCELL (BLMC)

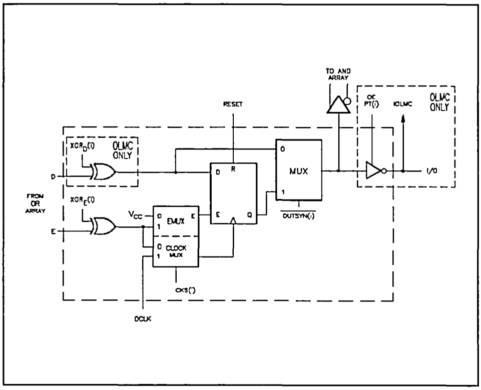
The outputs of the OR array feed two groups of macrocells. One group of eight macrocells is buried; its output feed back directly into the AND array rather than to device pins. These cells are called the Buried Logic Macrocells (BLMC): they are useful for building state machines. The second group of macrocells consists of 10 cells whose outputs, in addition to feeding back into the AND array, are available at the device pins. Cells in this group are known as Output Logic Macrocells (OLMC).

Like the ILMCs and IOLMCs discussed above, Output and Buried Logic Macrocells are configured by programming specific bits in the Architecture Control Word (CKS(i), OUTSYN(i), XOR_D(i), XOR_E(i)), but unlike the Input Macrocells which must be configured in blocks, these macrocells are configurable on a macrocell-by-macrocell basis. Throughout this data sheet, i=[14..23] for OLMCs and i=[0...7] for BLMCs.

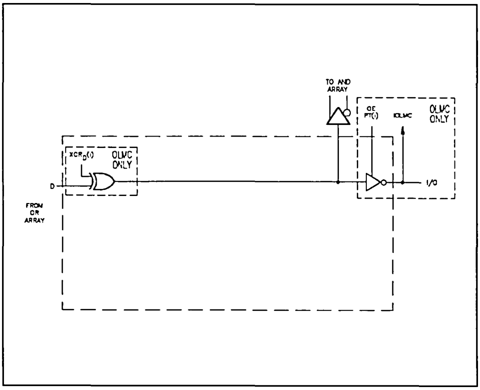
Buried and Output Logic Macrocells may be set to one of three valid configurations: combinational, D type registered with sum term (asynchronous) clock or D/E type registered.

Output macrocells always have I/O capability, with directional control provided by the 10 output enable (OE) product terms. Additionally, the polarity of each OLMC output is selectable through the XOR_D(i) architecture bits. Polarity selection is available for BLMCs, since both the true and complemented forms of their outputs are available in the AND array.

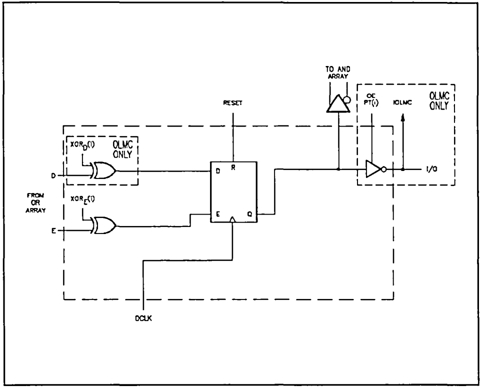
OLMC/BLMC Generic Block Diagram

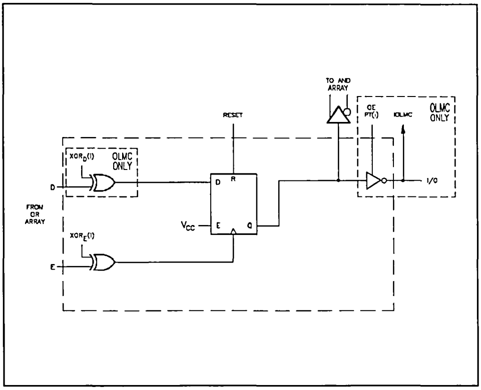


Combinational (CKS(i)=0, OUTSYN(i)=1)



D/E Type Registered (CKS(i)=1, OUTSYN(i)=0)





Polarity of all "E" sum terms is selectable through the $XOR_E(i)$ architecture control bits.

When CKS(i) = 1 and OUTSYN(i) = 0, macrocell "i" is set as "D/E type registered". In this configuration, the register is clocked from the common OCLK and the register clock enable input is controlled by the associated "E" sum term. This configuration is useful for building counters and state-machines with state hold functions.

When the macrocell is configured as a "D type registered with a sum term asynchronous clock" (CKS(i) = 0 and OUTSYN(i) = 0), the register is always enabled and its "E" sum term is routed directly to the clock input. This permits asynchronous programmable clocking, selected on a register-by-register basis.

When CKS(i) = 0 and OUTSYN(i) = 1, macrocell "i" is set as "combinational". Configuring a BLMC in this manner turns it into a complement array. Complement arrays are used to construct multi-level logic.

Registers in both the Output and Buried Logic Macrocells feature a common RESET product term. This active high product term allows the registers to be asynchronously reset. Registers are reset to a logic zero. If connected to an output pin, a logic one will occur because of the inverting output buffer.

There are two possible feedback paths from each OLMC: one directly from the OLMC (this feedback is before the output buffer and always present), and one from OLMC after the output buffer through the IOLMC. The second path is usable as a feedback only when the associated bidirectional pin is being used as an output. With this dual feedback arrangement, the OLMC can be permanently buried (the associate OLMC pin is an input), or dynamically buried with the use of the output enable product term.

The D/E registers used in this device offer the designer the ultimate in flexibility and utility. The D/E register architecture can emulate RS, JK, and T type registers with the same efficiency as a dedicated RS, JK, or T register.

The three valid macrocell configurations are shown in the macrocell equivalent diagrams shown in the previous page.

ARRAY DESCRIPTION

The GAL6001S E^2 reprogrammable array is subdivided into two smaller arrays; the first is an AND and the second is an OR array. These arrays are described in detail below.

AND ARRAY

The AND array is organized as 78 input terms by 75 product term outputs. The 10 ILMC, 10 I/O Logic Macrocells, 8 BLMC feedbacks, 10 OLMC feedbacks, and ICLK comprise the 39 inputs to this array (each available in true and complemented forms).

Product terms 0-63 serve as inputs to the OR array. Product term 64 is the RESET PT; it generates the RESET signal described in the earlier discussion of Output and Buried Logic Macrocells. Product terms 65-74 are the output enable product terms; they control the output buffers, thus enabling device pins 14-23 to be bidirectional or 3-state.

OR ARRAY

The OR array is organized as 64 inputs by 36 sum term outputs. 64 product terms from the AND array serve as the inputs to the OR array. Of the 36 sum term outputs, 18 are data ("D") terms and 18 are enable/clock ("E") terms. These terms feed into the 10 OLMCs and 8 BLMCs, one "D" term and one "E" term to each.

The programmable OR array offers unparalleled versatility in product term usage. This programmability allows from 1 to 64 product terms to be connected to a single sum term. A programmable OR array is more flexible than a fixed, shared, or variable product term architecture.

ARCHITECTURE CONTROL WORD

The various configurations of the GAL6001S are enabled by programming cells within the Architecture Control Word. This 68 bits word contains all of the chip configuration data. This data includes: $XOR_D(i)$, $XOR_E(i)$, CKS(i), OUTSYN(i), and LATCH and SYN bits both for ILMCs and IOLMCs. The function of each of these bits has been previously explained.

USER ELECTRONIC SIGNATURE WORD

An User Electronic Signature word (UES) is provided with GAL6001S device. The User Electronic Signature word is a 72 bits user definable storage area, which can be used to save inventory control data, pattern revision numbers, manufacture date, etc. Signature data is always available to the user, regardless of the state of the security cell.

Note: UES is included in checksum calculations. Changing the UES will alter the checksum.

SECURITY CELL

A security cell is provided with GAL6001S device as a deterrent to unauthorized copying of the array patterns. Once programmed, this cell prevents further read access to the AND and OR arrays. This cell can be erased only during a bulk erase cycle, so the original configuration can never be examined once this cell is programmed. User Electronic Signature data is always available to the user, regardless of the state of this control cell.

BULK ERASE

Before writing a new pattern into a previously programmed part, the old pattern must first be erased. This erasure is done automatically by the programming hardware as part of the programming cycle and takes only 50 milliseconds.

REGISTERED PRELOAD

When testing state machine designs, all possible states and state transitions must be verified, not just those required during normal operations. This verification is necessary because in system operation certain events may occur that cause the logic to assume an illegal state: power-up, brown out, line voltage glitches, etc. To test a design for proper management of these conditions, a method must be provided to break the feedback paths and force any desired state (e.g. an illegal state) into the registers. Then the machine can be sequenced and the outputs tested for correct next state generation. All registers in the GAL6001S can be preloaded, including the ILMC, IOLMC, OLMC, and BLMC registers. The programming hardware takes care or all preload timing and voltage requirements.

INPUT BUFFERS

GAL devices are designed with TTL level compatible input buffers. These buffers, with their characteristically high impedance, load driving logic much less than traditional bipolar devices.

This allows for a greater fan out from the driving logic.

GAL6001S does not include active pull-ups within its input structures. As a result, SGS-THOMSON recommends that all unused inputs and 3-state I/O pins be connected to another active input, V_{CC}, or GND. This precaution improves the noise immunity and reduces the I_{CC} consumption.

POWER-UP RESET

Circuitry within the GAL6001S provides a reset signal to all registers during power-up. All internal registers will have their Q outputs set low after a specified time (t_{reset}=10µs). As a result, the state on the registered output pins (if they are enabled) will always be high after power-up, regardless of the programmed polarity of the output pins. This feature can greatly simplify state machine design by providing a known state upon power-up.

The timing diagram for power-up is shown below. Because of the asynchronous nature of system power-up, the V_{CC} rise must be monotonic to guarantee a valid power-up reset of the GAL6001S. The registers will reset within a maximum of t_{reset} time. As in normal system operation, avoid clocking the device until all input and feedback path setup times have elapsed (i.e. avoid clocking before the t_{pr}=t_{reset}+t_{su} time interval).

DIFFERENTIAL PRODUCT TERM SWITCHING (DPTS)

The number of “Differential Product Term Switching” (DPTS) for a given design is calculated by taking the absolute value of:

- the total number of product terms that are switching from a logical level high to a logical level low
- minus the total number of those switching from a logical level low to a logical level high

within a 5ns time window.

$$DPTS = | PT_{LH} - PT_{HL} |$$

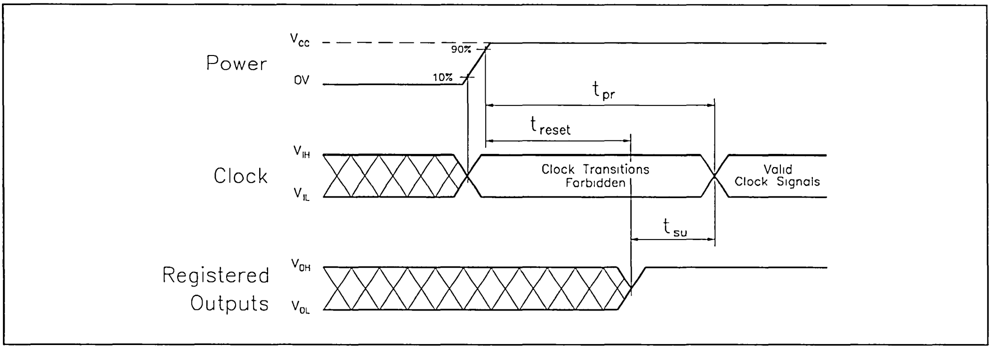
The correct behaviour of the device is guaranteed for applications where the number of DPTS is not greater than 15. This limit is believed to be largely conservative. For the device to exhibit an incorrect behaviour, other conditions of supply voltage, clock timing, temperature, etc., should be simultaneously present. As each of these conditions may, to some extent, lay partly within the operating range limits, it is simpler to refer to a DPTS boundary that ensures ample margin in all conditions for a correct operation.

There is no limit on the number of product terms that can be used at the same time.

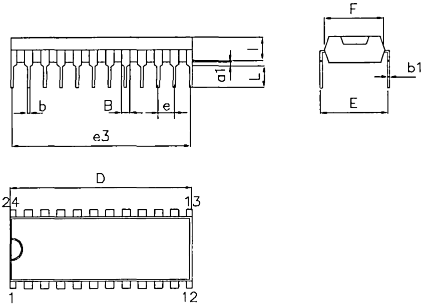
LATCH-UP PROTECTION

GAL6001S devices are designed with an on-board charge pump to negatively bias the substrate. The negative bias is of sufficient magnitude to prevent external disturbances from causing the circuitry to latch. Additionally, outputs are designed with n-channel pull-ups instead of the traditional p-channel pull-ups to eliminate any possibility of SCR induced latching.

Power-Up Reset Timing Diagram

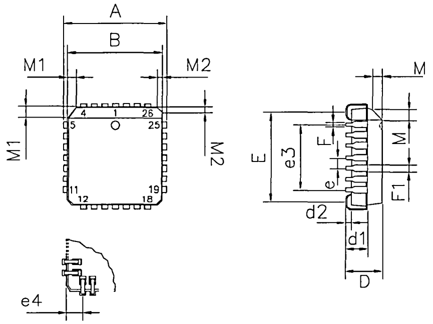


PDIP 24 Pins



Dim.	mm			inches		
	Min	Typ	Max	Min	Typ	Max
a1	0.38			0.015		
B	1.27		1.65	0.050		0.065
b		0.46			0.018	
b1		0.25			0.010	
D			31.88			1.255
E		7.62			0.300	
e		2.54			0.100	
e3		27.94			1.100	
F			6.86			0.27
I			4.32			0.170
L		3.30			0.130	

PLCC 28 Pins



Dim.	mm			inches		
	Min	Typ	Max	Min	Typ	Max
A	12.32		12.57	0.485		0.495
B	11.43		11.58	0.450		0.456
D	4.20		4.57	0.165		0.180
d1	2.29		3.04	0.090		0.120
d2	0.51			0.020		
E	9.91		10.92	0.390		0.430
e		1.27			0.050	
e3		7.62			0.300	
e4			1.99			0.078
F		0.46			0.018	
F1		0.71			0.028	
M		1.24			0.049	
M1		1.143			0.045	
M2			0.51			0.020

Seating Plane: 0.101 mm/0.004 inches

Ordering Informations*

SGS-THOMSON GAL®s are available in a variety of package and temperature ranges.
General ordering code is reported below.

GAL6001S - s w p t

Temperature	**1**	0˚C to +70˚C (only for 30ns Speed selection)
	3	- 40˚C to +85˚C (only for 35ns Speed selection)
Package	**B**	24 Pins PDIP
	C	28 Pins PLCC
Power	**H**	Half Power
Speed	**30**	30ns (Commercial Temperature range only)
	35	35ns (Industrial Temperature range only)

Example: ordering code for a GAL6001S, 30ns speed and Half Power in PDIP is **GAL6001S-30HB1**

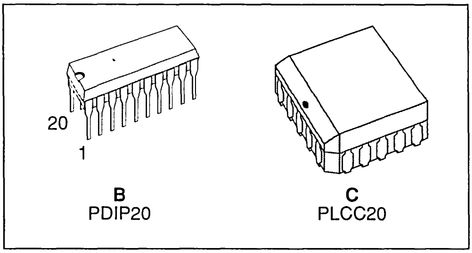
* Please contact local Product Marketing for latest update on package / temperature range availability.

SGS-THOMSON
MICROELECTRONICS

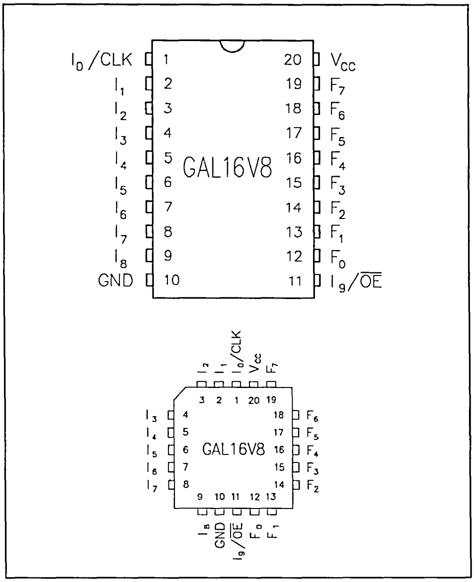
GAL16V8AS

E^2PROM CMOS PROGRAMMABLE LOGIC DEVICE

- HIGH PERFORMANCE SGS-THOMSON SINGLE-POLY E^2PROM CMOS TECHNOLOGY
  - 10ns maximum propagation delay (GAL16V8AS-10*xxx*)
  - F_{max} = 62.5MHz
  - 7ns max. from clock input to data output
  - TTL compatible 24mA outputs
  - SGS-THOMSON proprietary Single-Poly F3-G™ technology
- GLITCH FREE DEVICE
  - Enhanced design minimises ground bounce
- VERY LOW POWER
  - 90mA typ. (115mA max.) I_{CC} Half power selection, 45mA typ. (55 mA max.) I_{CC} Quarter power selection, 27mA typ. (30 mA max.) I_{CC} Eighth power selection
- ELECTRICAL ERASABLE CELL TECHNOLOGY
  - Reconfigurable logic/reprogrammable cells
  - 100% tested: guaranteed 100% final programming yield
  - High speed electrical program & erase
- EIGHT OUTPUT MACROCELLS
  - Maximum flexibility for complex logic design
  - Programmable output polarity
  - Also emulates 21 types of 20 pin PAL® devices with full function/fuse map/parametric compatibility
- PRELOAD AND POWER-ON RESET OF ALL REGISTERS
  - 100% functional testability
- ELECTRONIC SIGNATURE FOR USER'S IDENTIFICATION



Pin Connections



Pin Names

I_0–I_9	Input
CLK	Clock Input
F_0–F_7	I/O
$\overline{OE}$	Output Enable
V_{CC}	Power
GND	Ground

DESCRIPTION

The GAL16V8AS, at 10ns maximum propagation delay time, combines a high performance CMOS process with Electrical Erasable Single-Poly F3-G™ technology — SGS-THOMSON proprietary — to provide one of the highest speed-power performance products available in PLD market.

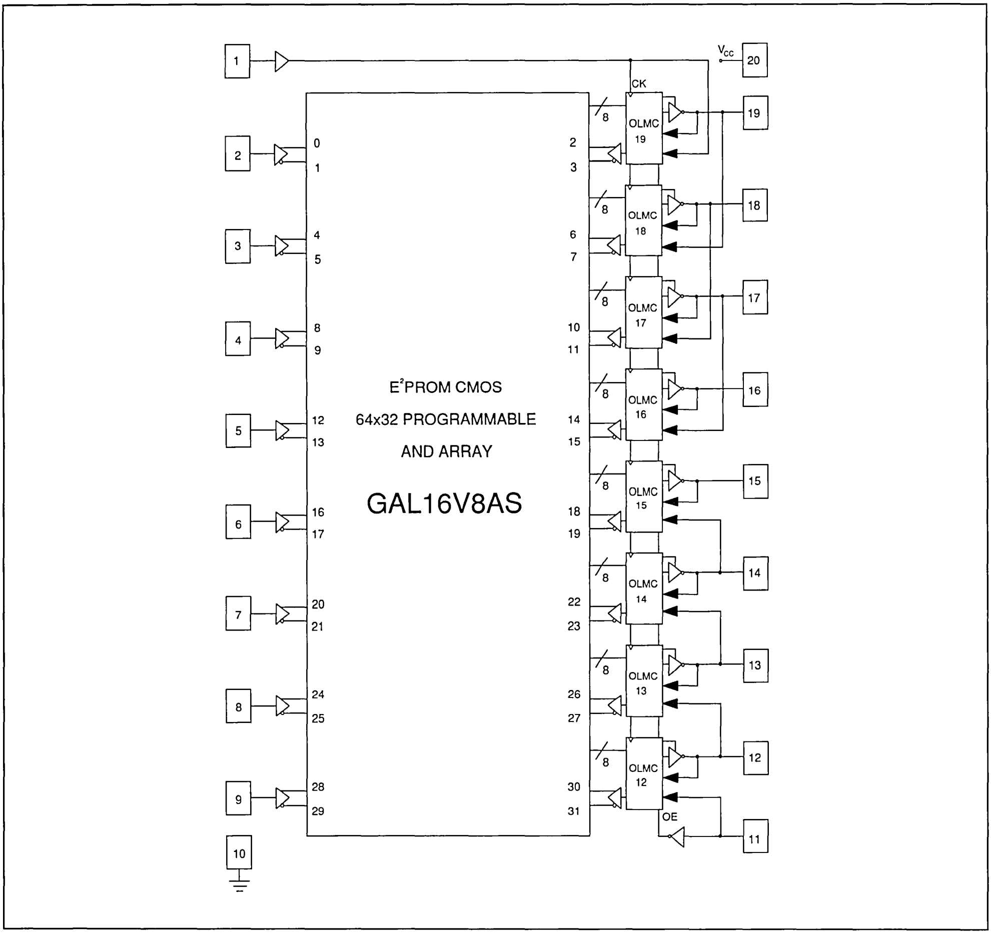
CMOS circuit allows GAL16V8AS to consume just 27mA typ. I_{CC} (Eighth power selection, 15ns) which represents a 75% saving in power when compared to its bipolar counterparts. Its E^2PROM CMOS technology offers high speed (50ms) erase time providing the ability to reprogram or reconfigure the device quickly and efficiently.

GAL© is a registered trademark of Lattice Semiconductor Corp ; PAL© is a registered trademark of Monolithic Memories Inc.

GAL16V8AS features 8 programmable Output Logic Macro Cells (OLMCs) allowing each output to be configured by the user. Additionally, the GAL16V8AS is capable of emulating, in a functional/fuse map/parametric compatible mode, 21 types of 20 pin PAL® devices. Unique test circuits and reprogrammable cells allow complete AC, DC and functional testing during manufacture.

Therefore, SGS-THOMSON guarantees 100% field programmability and functionality of GAL® devices. SGS-THOMSON also guarantees 100 erase/write cycles and data retention exceeding 20 years.

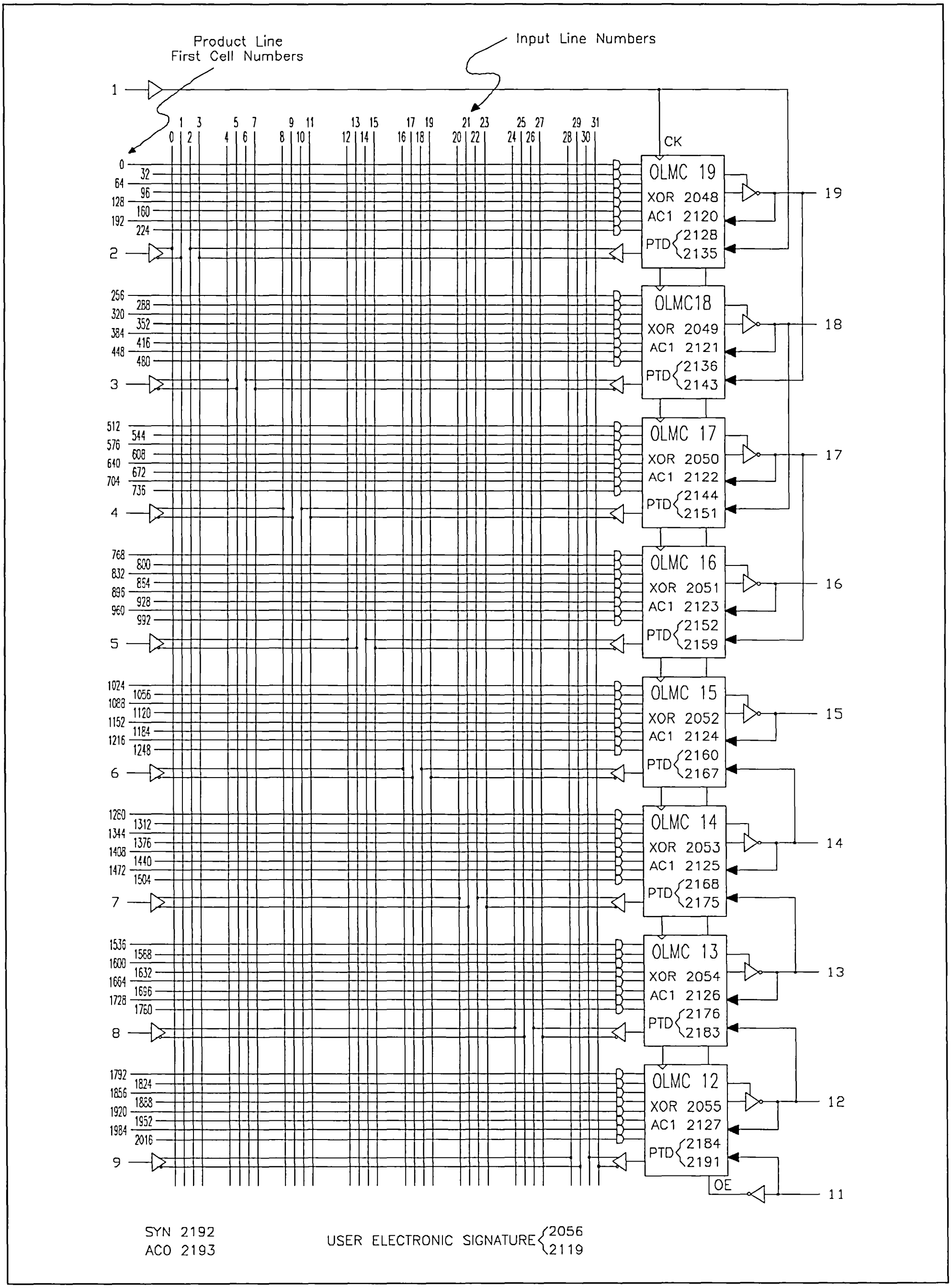
GAL16V8AS Block Diagram



GAL16V8AS PAL® Architecture Emulation

16L8	16R8	16RP6	16L2	14L4	12L6	10L8
16H8	16RP8	16R4	16H2	14H4	12H6	10H8
16P8	16R6	16RP4	16P2	14P4	12P6	10P8

GAL16V8AS Logic Diagram



Absolute Maximum Ratings

Symbol	Parameter	Value	Unit
V_{CC}	Supply Voltage	-0.5 to +7	V
V_I	Input Voltage Applied	-2.5 to V_{CC}+1	V
V_B	Off-State Output (Bidirectional) Voltage Applied	-2.5 to V_{CC}+1	V
T_{STG}	Storage Temperature	-65 to +125	˚C
T_J	Junction Temperature (Operating)	-40 to +125	˚C
T_L	Lead Temperature (Soldering)	260 (for 10s max.)	˚C

Stresses above those listed under the “Absolute Maximum Ratings” may cause permanent damage to the device. These are stress only ratings and functional operation of the device at these or at any other conditions above those indicated in the operational sections of this specification is not implied.

ESD Immunity

Test Method: Human Body Model (HBM)
ESD Tolerance ≥ 2000V
(See MIL-STD 883c).

Test Method: Charge Device Model (CDM)
ESD Tolerance ≥ 500V
Test Instrument: KeyTek ZapMaster

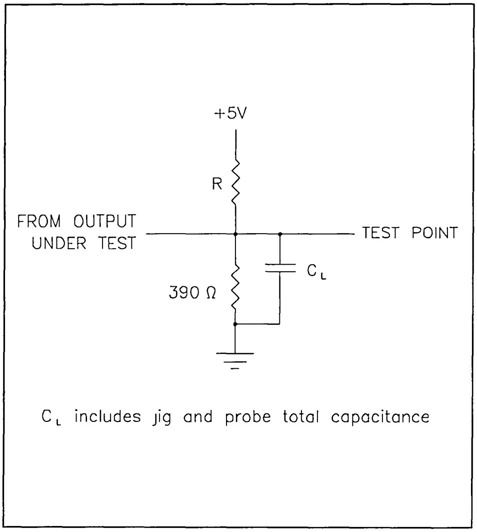
CDM is an additional test only for GAL®s not yet adopted as a company standard test.

Switching Test Conditions

Input Pulse Levels	GND to 3.0V
Input Rise and Fall Times	3ns 10%-90%
Input Timing Reference Levels	1.5V
Output Timing Reference Levels	1.5V
Output Load	See figure

3-state levels are measured 0.5V from steady-state active level.

Switching Test Circuit



C_L includes jig and probe total capacitance

Test Conditions

#	R [Ω]	C_L [pF]
1	200	50
2	Active High: ∞ Active Low: 200	50
3	Active High: ∞ Active Low: 200	5

Capacitance (T_A=25˚C, f=1.0MHz, V_{CC}=5V)

Symbol	Parameter	Test Conditions	Maximum∵	Units
C_I	Input Capacitance	V_I=2V	8	pF
C_B	Bidirectional Pin Capacitance	V_B=2V	10	pF

∵ Guarantied but not 100% tested

DC Operating Conditions

Symbol	Parameter	Commercial Temperature Range		Industrial Temperature Range		Units
		Min.	Max.	Min.	Max.	
V_{CC}	Supply Voltage	4.75	5.25	4.5	5.5	V
T_A	Ambient Temperature	0	70	-40	85	˚C
V_{IL}	Input Low Voltage	V_{SS}[†]-0.5	0.8	V_{SS}[†]-0.5	0.8	V
V_{IH}	Input High Voltage	2.0	V_{CC}+1	2.0	V_{CC}+1	V
I_{OL}	Low Level Output Current	–	24	–	24	mA
I_{OH}	High Level Output Current	-3.2	–	-3.2	–	mA

[†] V_{SS} is the voltage applied to the GND pin

Electrical Characteristics Over Operating Conditions (Commercial Temperature Range)

Symbol	Parameter	Test Conditions		Min.	Max.	Units
I_{IH}, I_{IL}	Input Leakage Current	GND≤V_I≤$V_{CC\,Max}$		–	±10	µA
I_{BH}, I_{BL}	Bidirectional Pin Leakage Current	GND≤V_I≤$V_{CC\,Max}$		–	±10	µA
I_{CC}	Operating Power Supply Current	f=15MHz (Q, E) f=25MHz (H) V_{CC} = $V_{CC\,Max}$ V_{IL}=0.5V V_{IH}=3.0V	Half Power	–	115	mA
			Quarter Power (only 15 and 20ns)	–	55	
			Eighth Power (only 15ns)	–	30	
I_{OS}[*]	Output Short Circuit Current	V_{CC}=5.0V, V_B=0.5V		-150	-30	mA
V_{OL}	Output Low Voltage	–		–	0.5	V
V_{OH}	Output High Voltage	–		2.4	–	V

Electrical Characteristics Over Operating Conditions (Industrial Temperature Range)

Symbol	Parameter	Test Conditions		Min.	Max.	Units
I_{IH}, I_{IL}	Input Leakage Current	GND≤V_I≤$V_{CC\,Max}$		–	±10	µA
I_{BH}, I_{BL}	Bidirectional Pin Leakage Current	GND≤V_I≤$V_{CC\,Max}$		–	±10	µA
I_{CC}	Operating Power Supply Current	f=15MHz (Q, E) f=25MHz (H) V_{CC} = $V_{CC\,Max}$ V_{IL}=0.5V V_{IH}=3.0V	Half Power	–	130	mA
			Quarter Power (only 15 and 20ns)	–	65	
			Eighth Power (only 15ns)	–	40	
I_{OS}[*]	Output Short Circuit Current	V_{CC}=5.0V, V_B=0.5V		-150	-30	mA
V_{OL}	Output Low Voltage	–		–	0.5	V
V_{OH}	Output High Voltage	–		2.4	–	V

[*] One output at a time for a maximum duration of one second

Switching Characteristics Over Operating Conditions

Symbol	Parameter	From	To	16V8AS 10	16V8AS 12	16V8AS 15	16V8AS 20	Units	Test Cond.‡
				Max.☆	Max.☆	Max.	Max.		
t_{pd}	Combinational Propagation Delay	Input	Output	10	12	15	20	ns	1
t_{co}	Clock to Output Delay	Clock	Registered Output	7	10	10	15	ns	1
t_{en}	Product Term Output Enable to Output	Input	Output	10	12	15	20	ns	2
t_{enr}	Output Register Enable to Output	$\overline{OE}$	Registered Output	10	12	15	18	ns	2
t_{dis}	Product Term Output Disable to Output	Input	Output	10	12	15	20	ns	3
t_{disr}	Output Register Disable to Output	$\overline{OE}$	Registered Output	10	12	15	18	ns	3

AC Operating Conditions

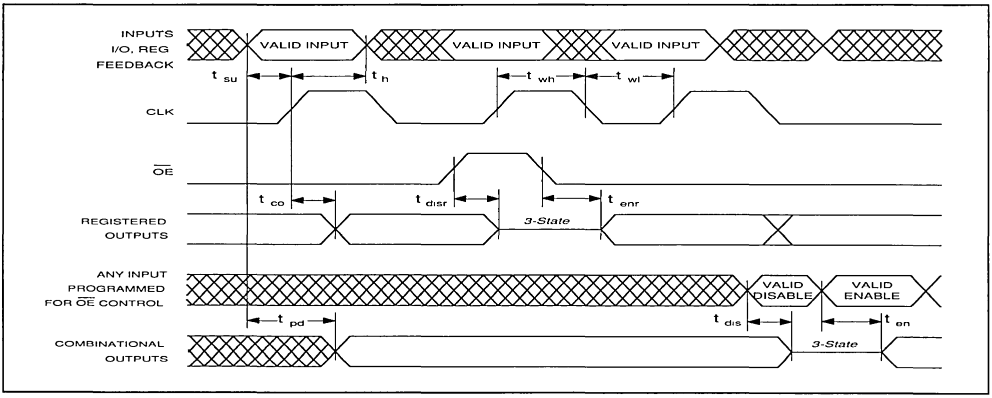
Symbol	Parameter	16V8AS 10☆		16V8AS 12☆		16V8AS 15		16V8AS 20		Units	Test Cond.‡
		Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.		
t_{su}	Input or Feedback Setup Time (Before Clock Rise)	–	10	–	12	–	12	–	15	ns	–
t_h	Input or Feedback Hold Time (After Clock Rise)	–	0	–	0	–	0	–	0	ns	–
t_{wh}	Minimum Clock Width High	–	8	–	8	–	10	–	12	ns	–
t_{wl}	Minimum Clock Width Low	–	8	–	8	–	10	–	12	ns	–
f_{clk}◇	Clock Frequency Without Feedback	62.5	–	62.5	–	50	–	41.7	–	MHz	1
f_{clkf}✦	Clock Frequency With Feedback	58.8	–	48.5	–	41.6	–	33.3	–	MHz	1

☆ Commercial Temperature range only.

‡ Refer to "Switching Test Conditions"

◇ $f_{clk} = \frac{1}{t_{wh} + t_{wl}}$ ✦ $f_{clkf} = \frac{1}{t_{su} + t_{co}}$

Switching Waveforms



FUNCTIONAL DESCRIPTION

GAL16V8AS has a programmable AND array whose output terms feed a fixed (non programmable) OR array, as bipolar PAL®. The 2 × 8 input lines enter the AND array as true or complemented form. 64 product terms are available allowing standard Sum of Products Logic implementation. Each product term is obtained by appropriate connections between the input lines and the product term line. The connections can be made by programming the E^{2}PROM memory cell at each intersection of the AND matrix (2048 memory cells). The 64 product terms are divided into eight groups of 8 terms each. One product term for each group can be used to provide Output Enable control for combinational output, the others are connected with an OR gate into the corresponding OLMC (Output Logic Macrocell). The output buffer is in 3-state when the corresponding output enable signal is low.

OUTPUT LOGIC MACROCELL (OLMC)

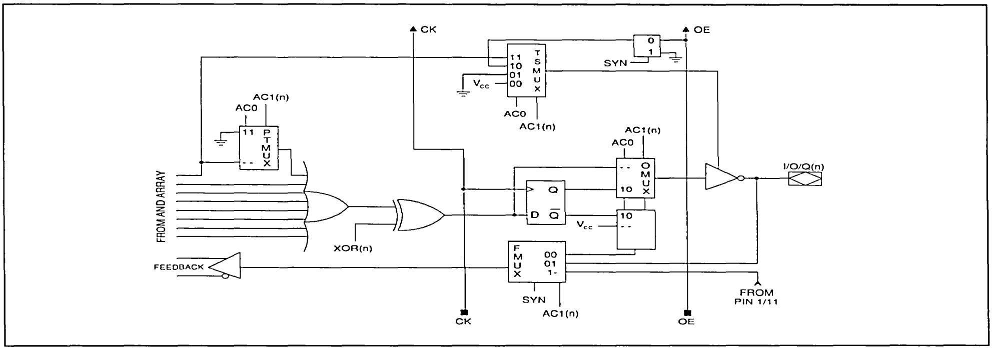
The following discussion pertains to configuring the output logic macrocells. It should be noted that actual implementation is accomplished by development software/hardware and is completely transparent to the user.

The outputs of the AND array are fed into an OLMC, where each output can be individually set to active high or active low, with either combinational (asynchronous) or registered (synchronous) configurations. A common output enable is connected to all registered outputs; product terms can be used to provide individual output enable control for combinational outputs. All outputs have always programmable polarity.

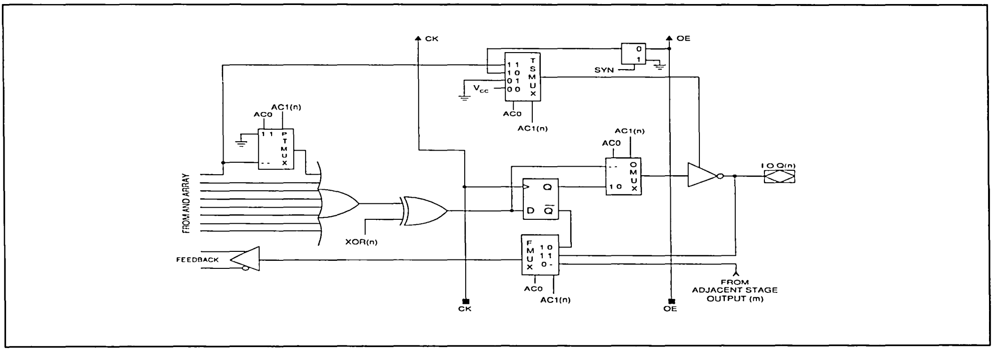
The output logic macrocell provides the designer with maximum output flexibility in matching signal requirements, thus providing more functions than existing 20 pin PAL® devices.

Three different configuration modes of the OLMCs are possible: registered, complex and simple. The output of an OLMC in registered mode can be either registered or combinational. Different modes cannot be mixed: i.e. all OLMCs of a device have to be configured in simple, complex or registered mode.

GAL16V8AS Output Logic Macrocell Pin 12 and 19



GAL16V8AS Output Logic Macrocell Pin 13 to 18

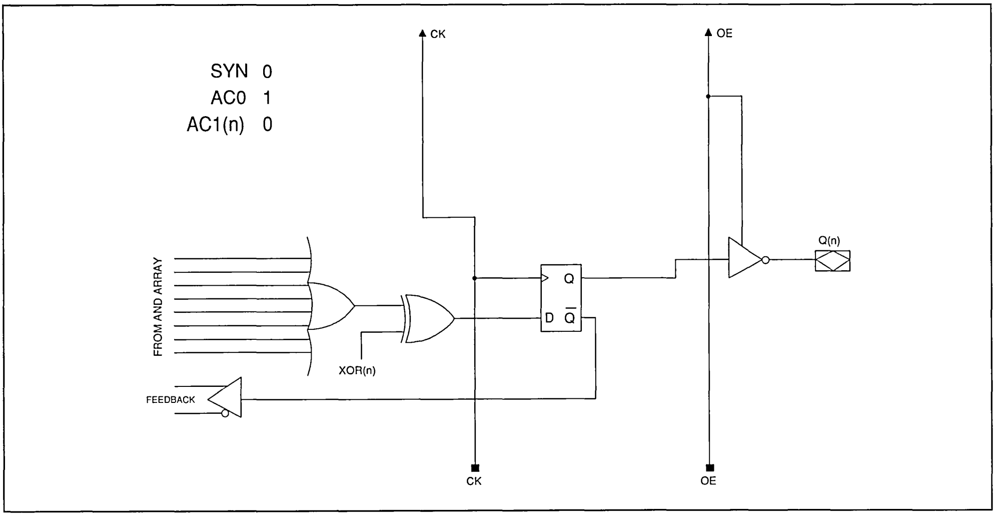


REGISTERED MODE

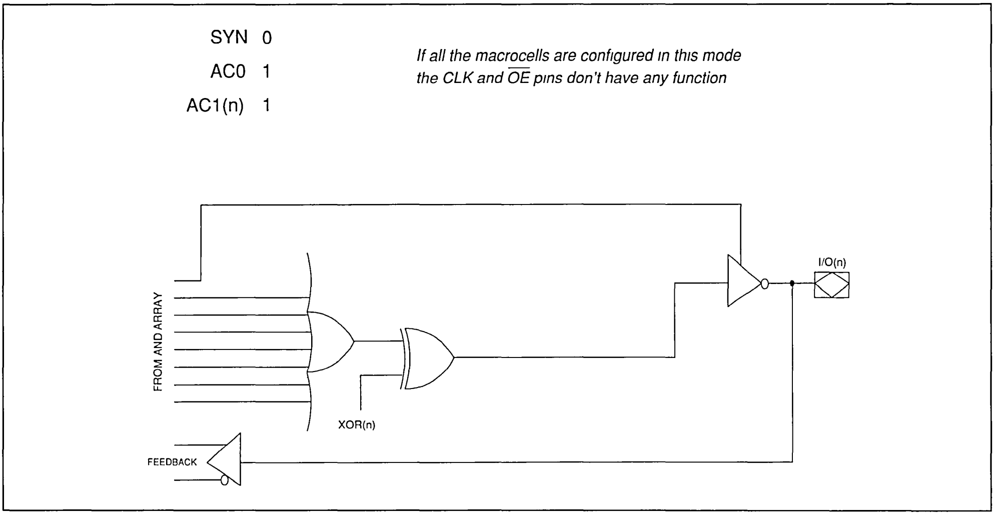
In registered mode macrocells are configured as registered outputs or combinational inputs/outputs. Any macrocell can be configured as registered output or combinational input/output. Up to 8 registered outputs or up to 8 inputs/outputs are possible in this mode.

All registered macrocells share common clock and output enable control. Registered outputs have 8 data product terms per output, while combinational inputs/outputs have only 7 data product terms per output: in the latter case the eighth product term serves as individual output enable control for each macrocell.

Registered Output with Programmable Polarity



Combinational Input/Output with Programmable OE and Polarity



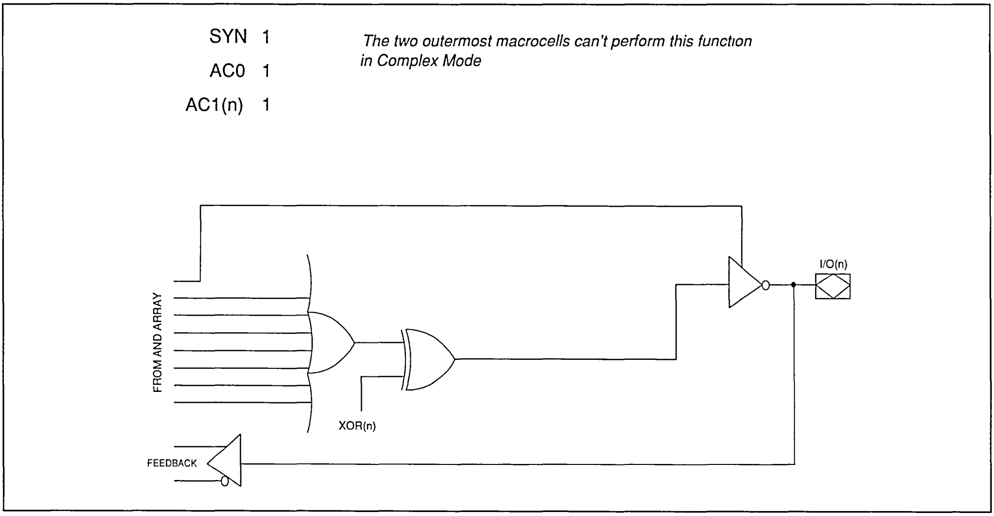
If all the macrocells are configured in this mode the CLK and $\overline{OE}$ pins don't have any function

COMPLEX MODE

In complex mode macrocells are configured as combinational inputs/outputs or outputs only. The two outermost macrocells (12 and 19) do not have input capability: so only 6 inputs/outputs are possible in this mode. Applications requiring 8 inputs/outputs must be implemented in registered mode.

All macrocells have 7 data product terms per output; the eighth product term is used as individual output enable control for each macrocell. The clock and output enable pins (pins 1 and 11 respectively) are always available as inputs.

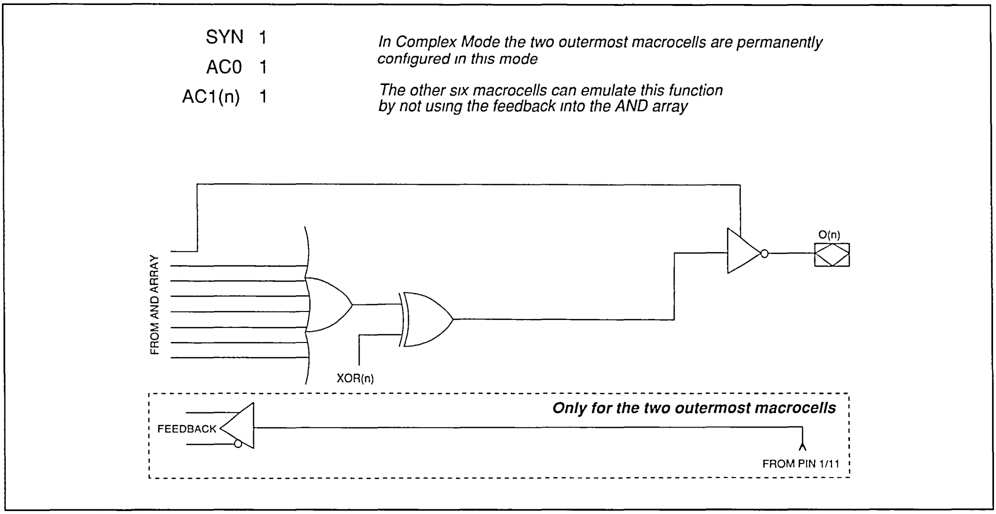
Combinational Input/Output with Programmable OE and Polarity



SYN 1
AC0 1
AC1(n) 1

The two outermost macrocells can't perform this function in Complex Mode

Combinational Output with Programmable OE and Polarity



SYN 1
AC0 1
AC1(n) 1

In Complex Mode the two outermost macrocells are permanently configured in this mode

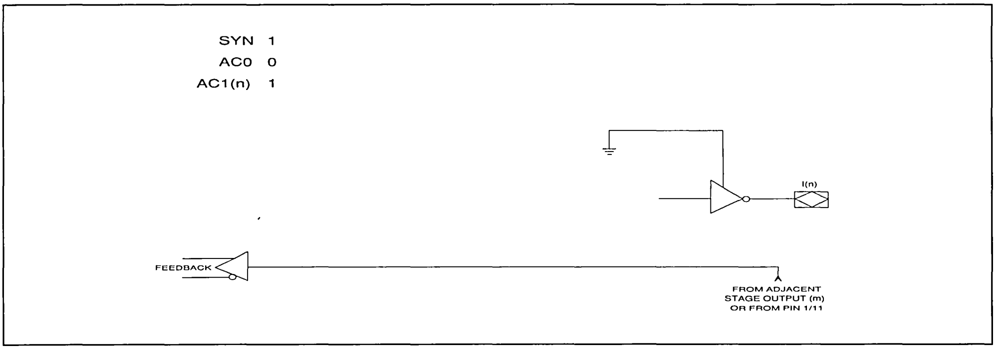
The other six macrocells can emulate this function by not using the feedback into the AND array

Only for the two outermost macrocells

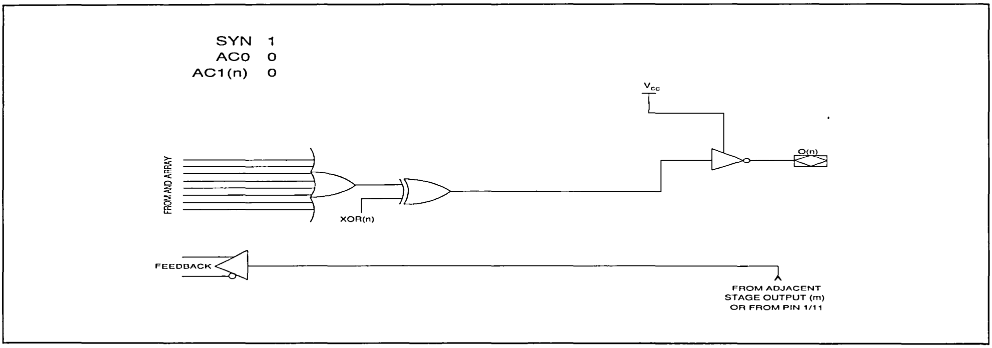
SIMPLE MODE

In simple mode macrocells are configured as dedicated inputs or as dedicated, always active, combinational outputs. All macrocells have 8 data product terms per output. The clock and output enable pins (pins 1 and 11 respectively) are always available as inputs.

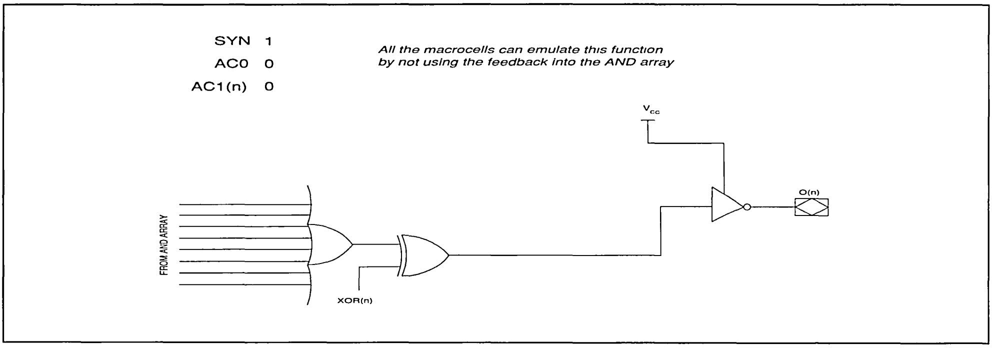
Dedicated Input Mode



Dedicated Combinational Output with Feedback and Programmable Polarity



Dedicated Combinational Output with Programmable Polarity



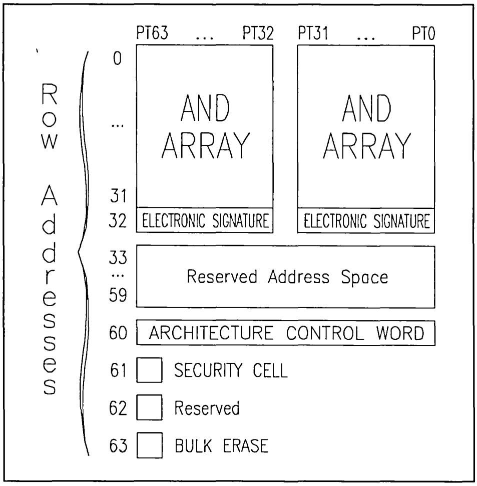
All the macrocells can emulate this function by not using the feedback into the AND array

ROW ADDRESS MAP DESCRIPTION

There are a total of 36 unique row addresses available to the user when programming the GAL16V8AS device. Row addresses 0-31 each contain 64 bits of input term data. This is the AND array where the custom logic pattern is programmed. Row 32 is the Electronic Signature Word. It has 64 bits available for any user defined purpose. Row 33-59 are reserved by the manufacturer and are not available to users.

Row 60 contains the architecture and output polarity information. The 82 bits within this word are programmed to configure the device for a specific application. Row 61 contains a one bit security cell that when programmed prevents further pattern verification of the array. Row 63 is the row that is addressed to perform a bulk erase of the device, resetting it back to a virgin state. Each of these functions is described in the following sections.

GAL16V8AS Row Addresses Map Block Diagram



ELECTRONIC SIGNATURE WORD DESCRIPTION

An electronic signature word is provided with every GAL16V8AS device. It resides at row address 32 and contains 64 bits of reprogrammable memory that can contain user-defined data. Some uses include user ID codes, revision numbers, or inventory control. This signature data is always available to the user independent of the state of the security cell.

ARCHITECTURE CONTROL WORD

All the various output configurations of the GAL16V8AS devices are controlled by programming cells within the 82 bit Architecture Control Word that resides at row 60. The location of specific bits within the Architecture Control Word is shown in the control word diagram in figure below. The function of the SYN, AC0 and AC1(n) bits have been explained in the OUTPUT LOGIC MACROCELL description. The eight polarity bits determine each output's polarity individually. The numbers below the XOR(n) and AC1(n) bits in the architecture control word diagram shows the output device pin number that the polarity bits control.

SECURITY CELL

Row address 61 contains the Security Cell (one bit). The Security Cell is provided on all GAL16V8AS devices as a deterrent to unauthorized copying of the array configuration patterns. Once programmed, the circuitry enabling array access is disabled, preventing further verification of the array (rows 0-31). The cell can be erased only in conjunction with the array during a bulk erase cycle, so the original configuration can never be examined once this cell is programmed. Signature data is always available to the user.

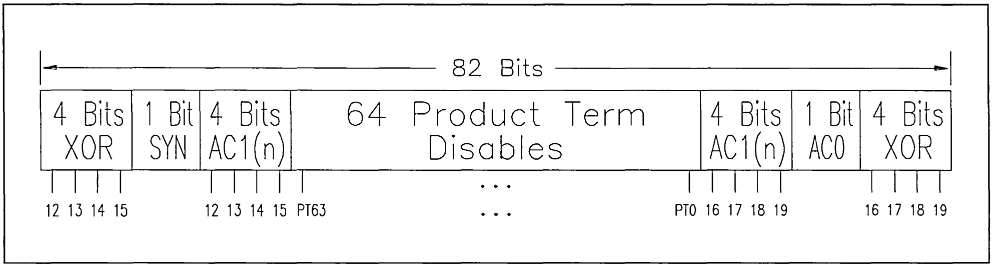
BULK ERASE MODE

By addressing row 63 during a programming cycle, a clear function performs a bulk erase of the array and the Architecture Control Word. In addition, the Electronic Signature Word and the Security Cell are erased. This mode resets a previously configured device back to its virgin state.

OUTPUT REGISTER PRELOAD

When testing state machine designs, all possible

GAL16V8AS Architecture Control Word Diagram



states and state transitions must be verified in the design, not just those required in the normal machine operations. This is because in system operation, certain events occur that may throw the logic into an illegal state (power-up, line voltage glitches, brown-outs, etc.). To test a design for proper management of these conditions, a method must be provided to break the feedback paths, and force any desired (e.g. illegal) state into a register. Then the machine can be sequenced and the outputs tested for the correct next state condition. The GAL16V8AS device includes circuitry that allows each registered output to be synchronously set either high or low. Thus, any present state condition can be forced for test sequencing. If necessary, approved GAL programmers capable of executing test vectors can perform output register preload automatically.

The figure on the right shows the pin functions necessary to preload the register. This test mode is entered by raising PRLD to V_{IES} (register preload input voltage, typically 15V), which enables the serial data in (S_{DIN}) buffer and the serial data out (S_{DOUT}) buffer. Data is then serially shifted into the registers on each rising edge of the clock, D_{CLK}. Only the macrocells with registered output configurations are loaded. If only 3 outputs have registers, then only 3 bits need be shifted in. The registers are loaded from the bottom up as shown in the figure on the right.

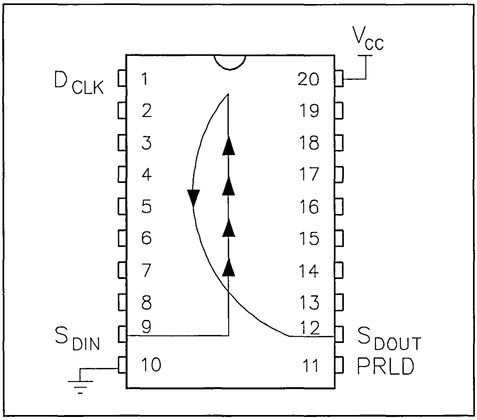
LATCH-UP PROTECTION

GAL® devices are designed with an on board charge pump to negatively bias the substrate. The negative bias is of sufficient magnitude to prevent input undershoots from causing the circuitry to latch. Additionally, outputs are designed with n-channel pullups instead of the traditional p-channel pullups to eliminate any possibility of SCR induced latching.

POWER-UP RESET

Circuitry within the GAL16V8AS provides a reset signal to all registers during power-up. All internal registers will have their Q outputs set low after a specified time (t_{reset}=10µs). As a result, the state on the registered output pins (if they are enabled through $\overline{OE}$) will always be high on power-up, regardless of the programmed polarity of the output pins. This features can greatly simplify state machine design by providing a known state on power-up.

Output Register Preload Pinout

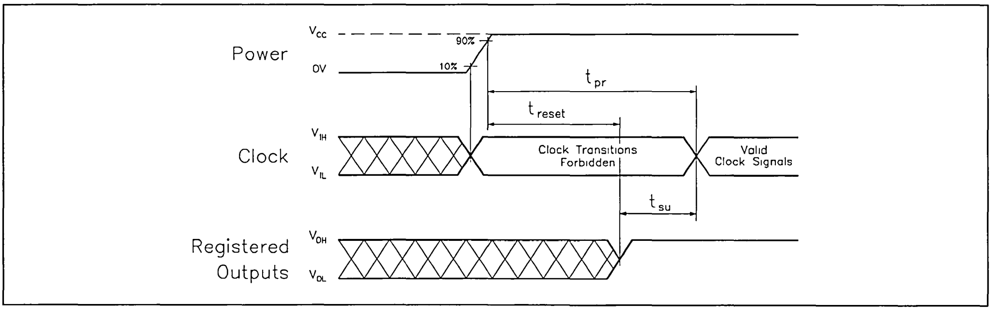


The timing diagram for power-up is shown below. Because of the asynchronous nature of system power-up, the V_{CC} rise must be monotonic to guarantee a valid power-up reset of the GAL16V8AS. The registers will reset within a maximum of t_{reset} time: before this time any clock transition from low to high is forbidden to avoid undesired commutations. As in normal system operation, avoid clocking the device until all input and feedback path setup times have been met (i.e. avoid clocking before the $t_{pr}=t_{reset}+t_{su}$ time interval).

DEVICES PROGRAMMING

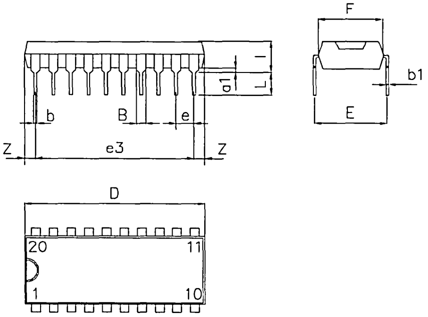
SGS-THOMSON strongly recommends the use of qualified programming hardware. Programming on unapproved equipment will invalidate all guarantees.

Power-Up Reset Timing Diagram



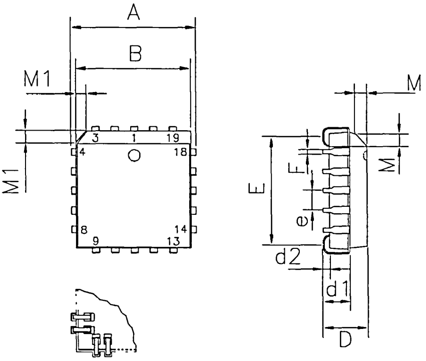
PACKAGE MECHANICAL DATA

PDIP 20 Pins



Dim.	mm			inches		
	Min	Typ	Max	Min	Typ	Max
a1	0.254			0.010		
B	1.39		1.65	0.054		0.064
b		0.45			0.017	
b1		0.25			0.009	
D			25.4			1.000
E		8.50			0.334	
e		2.54			0.100	
e3		22.86			0.900	
F			7.10			0.279
I			3.93			0.154
L		3.30			0.129	
Z		1.27	1.34		0.050	0.052

PLCC 20 Pins



Dim.	mm			inches		
	Min	Typ	Max	Min	Typ	Max
A	9.78		10.03	0.385		0.395
B	8.89		9.04	0.350		0.356
D	4.20		4.57	0.165		0.180
d1		2.54			0.100	
d2		0.56			0.022	
E	7.37		8.38	0.290		0.330
e		1.27			0.050	
F		0.38			0.015	
M		1.27			0.050	
M1		1.14			0.045	

Seating Plane: 0.101 mm/0.004 inches

Ordering Informations*

SGS-THOMSON GAL®s are available in a variety of package and temperature ranges.
General ordering code is reported below.

GAL16V8AS - s w p t

Temperature **1** 0°C to +70°C
3 -40°C to +85°C (only for 15 and 20ns Speed selection)

Package **B** 20 Pins PDIP
C 20 Pins PLCC

Power **H** Half Power
Q Quarter Power (only for 15 and 20ns Speed selection)
E Eighth Power (only for 15ns Speed selection)

Speed **10** 10ns
12 12ns
15 15ns
20 20ns

Example: ordering code for a GAL16V8AS, 12ns speed and Half Power in PDIP is
GAL16V8AS-12HB1

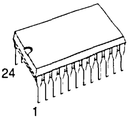
* Please contact local Product Marketing for latest update on package / temperature range availability.

SGS-THOMSON
MICROELECTRONICS

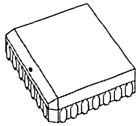
GAL20V8AS

E^{2}PROM CMOS PROGRAMMABLE LOGIC DEVICE

- HIGH PERFORMANCE SGS-THOMSON SINGLE-POLY E^{2}PROM CMOS TECHNOLOGY
  - 10ns maximum propagation delay (GAL20V8AS-10*xxx*)
  - F_{max} = 62.5MHz
  - 7ns max. from clock input to data output
  - TTL compatible 24mA outputs
  - SGS-THOMSON proprietary Single-Poly F3-G™ technology
- GLITCH FREE DEVICE
  - Enhanced design minimises ground bounce
- VERY LOW POWER
  - 90mA typ. (115mA max.) I_{CC} Half power selection, 45mA typ. (55mA max.) I_{CC} Quarter power selection, 27mA typ. (30mA max.) I_{CC} Eighth power selection
- ELECTRICAL ERASABLE CELL TECHNOLOGY
  - Reconfigurable logic/reprogrammable cells
  - 100% tested: guaranteed 100% final programming yield
  - High speed electrical program & erase
- EIGHT OUTPUT MACROCELLS
  - Maximum flexibility for complex logic design
  - Programmable output polarity
  - Also emulates 21 types of 24 pin PAL® devices with full function/fuse map/parametric compatibility
- PRELOAD AND POWER-ON RESET OF ALL REGISTERS
  - 100% functional testability
- ELECTRONIC SIGNATURE FOR USER'S IDENTIFICATION

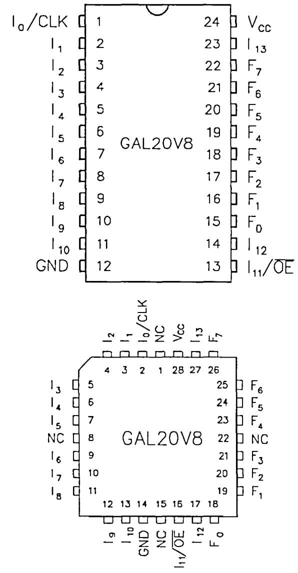


B
PDIP24



C
PLCC28

Pin Connections



DESCRIPTION

The GAL20V8AS, at 10ns maximum propagation delay time, combines a high performance CMOS process with Electrical Erasable Single-Poly F3-G™ technology — SGS-THOMSON proprietary — to provide one of the highest speed-power performance products available in PLD market.

CMOS circuit allows GAL20V8AS to consume just 27mA (typ.) I_{CC} (Eighth power selection, 15ns) which represents a 75% saving in power when compared to its bipolar counterparts. Its E^{2}PROM CMOS technology offers high speed (50ms) erase time providing the ability to reprogram or reconfigure the device quickly and efficiently.

Pin Names

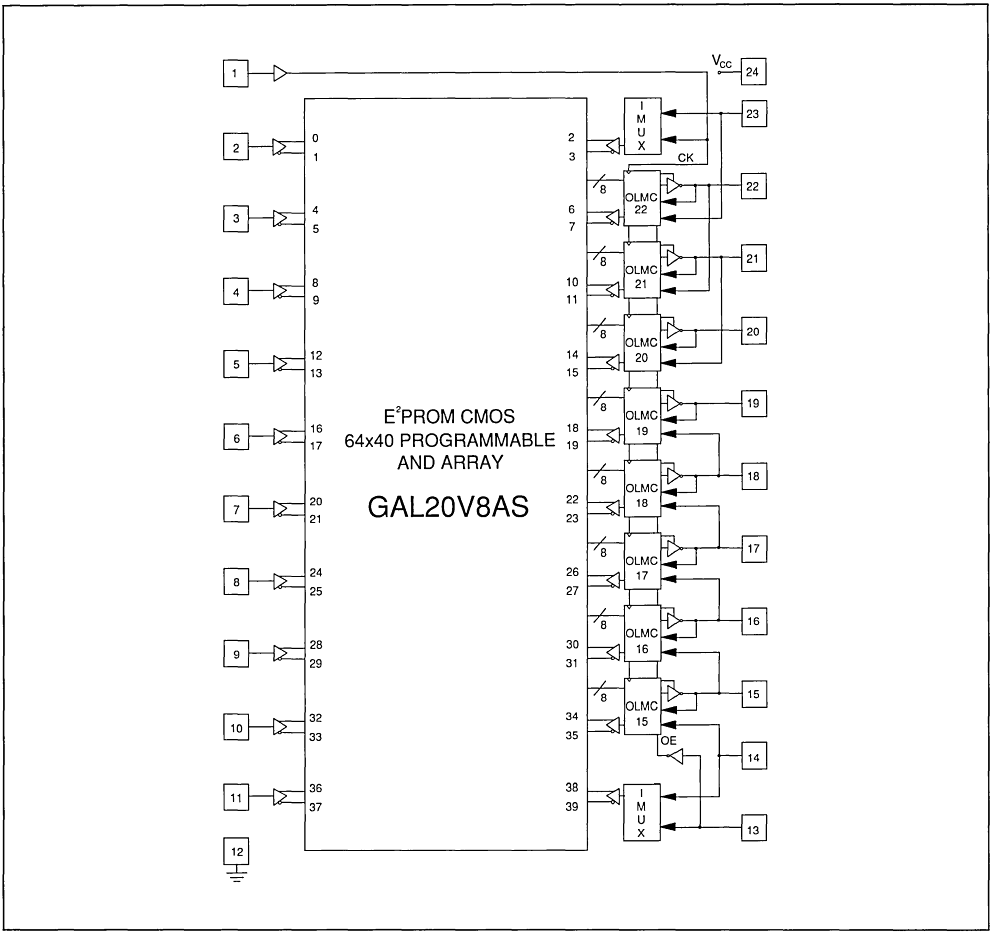
I_0–I_{13}	Input
CLK	Clock Input
F_0–F_7	I/O
$\overline{OE}$	Output Enable
V_{CC}	Power
GND	Ground

GAL® is a registered trademark of Lattice Semiconductor Corp.; PAL® is a registered trademark of Monolithic Memories Inc

GAL20V8AS features 8 programmable Output Logic Macro Cells (OLMCs) allowing each output to be configured by the user. Additionally, the GAL20V8AS is capable of emulating, in a functional/fuse map/parametric compatible mode, 21 types of 24 pin PAL® devices. Unique test circuits and reprogrammable cells allow complete AC, DC and functional testing during manufacture.

Therefore, SGS-THOMSON guarantees 100% field programmability and functionality of GAL® devices. SGS-THOMSON also guarantees 100 erase/write cycles and data retention exceeding 20 years.

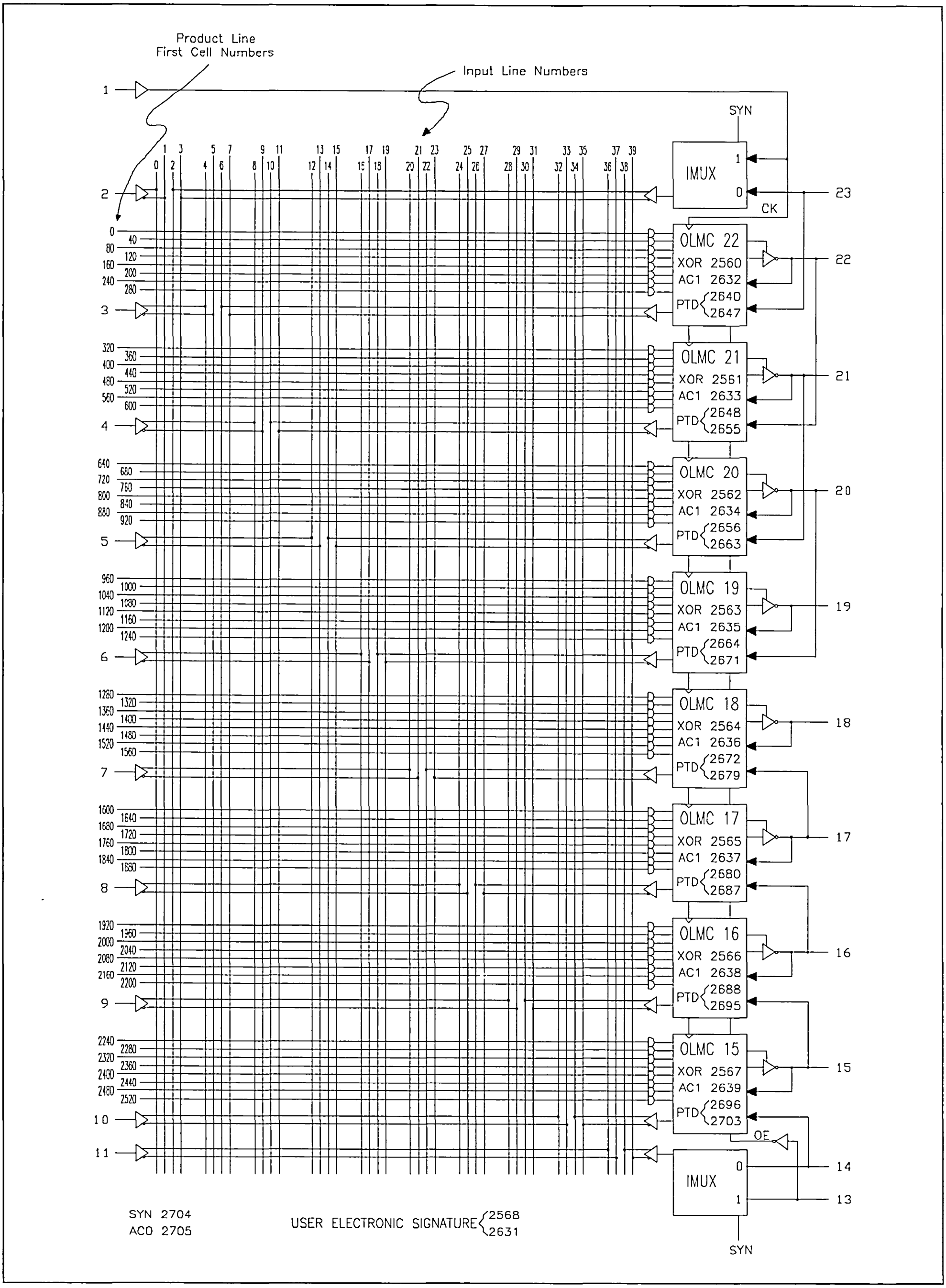
GAL20V8AS Block Diagram



GAL20V8AS PAL® Architecture Emulation

20L8	20R8	20RP6	20L2	18L4	16L6	14L8
20H8	20RP8	20R4	20H2	18H4	16H6	14H8
20P8	20R6	20RP4	20P2	18P4	16P6	14P8

GAL20V8AS Logic Diagram



Absolute Maximum Ratings

Symbol	Parameter	Value	Unit
V_{CC}	Supply Voltage	-0.5 to +7	V
V_I	Input Voltage Applied	-2.5 to V_{CC}+1	V
V_B	Off-State Output (Bidirectional) Voltage Applied	-2.5 to V_{CC}+1	V
T_{STG}	Storage Temperature	-65 to +125	˚C
T_J	Junction Temperature (Operating)	-40 to +125	˚C
T_L	Lead Temperature (Soldering)	260 (for 10s max.)	˚C

Stresses above those listed under the “Absolute Maximum Ratings” may cause permanent damage to the device. These are stress only ratings and functional operation of the device at these or at any other conditions above those indicated in the operational sections of this specification is not implied.

ESD Immunity

Test Method: Human Body Model (HBM)
ESD Tolerance ≥ 2000V
(See MIL-STD 883c).

Test Method: Charge Device Model (CDM)
ESD Tolerance ≥ 500V
Test Instrument: KeyTek ZapMaster

CDM is an additional test only for GAL®s not yet adopted as a company standard test.

Switching Test Conditions

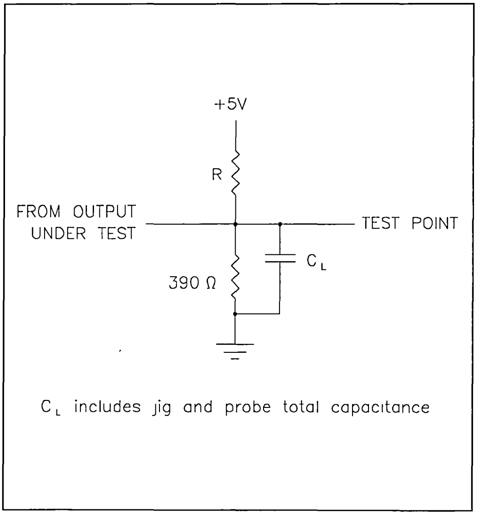
Input Pulse Levels	GND to 3.0V
Input Rise and Fall Times	3ns 10%-90%
Input Timing Reference Levels	1.5V
Output Tıming Reference Levels	1.5V
Output Load	See figure

3-state levels are measured 0.5V from steady-state active level.

Test Conditions

#	R [Ω]	C_L [pF]
1	200	50
2	Active High: ∞ Active Low: 200	50
3	Active High: ∞ Active Low: 200	5

Switching Test Circuit



C_L includes jig and probe total capacitance

Capacitance (T_A=25˚C, f=1.0MHz, V_{CC}=5V)

Symbol	Parameter	Test Conditions	Maximum⁘	Units
C_I	Input Capacitance	V_I=2V	8	pF
C_B	Bidirectıonal Pın Capacıtance	V_B=2V	10	pF

⁘ Guarantıed but not 100% tested

DC Operating Conditions

Symbol	Parameter	Commercial Temperature Range		Industrial Temperature Range		Units
		Min.	Max.	Min.	Max.	
V_{CC}	Supply Voltage	4.75	5.25	4.5	5.5	V
T_A	Ambient Temperature	0	70	-40	85	˚C
V_{IL}	Input Low Voltage	VSS✢-0.5	0.8	V_{SS}✢-0.5	0.8	V
V_{IH}	Input High Voltage	2.0	V_{CC}+1	2.0	V_{CC}+1	V
I_{OL}	Low Level Output Current	–	24	–	24	mA
I_{OH}	High Level Output Current	-3.2	–	-3.2	–	mA

✢ V_{SS} is the voltage applied to the GND pin.

Electrical Characteristics Over Operating Conditions (Commercial Temperature Range)

Symbol	Parameter	Test Conditions		Min.	Max.	Units
I_{IH}, I_{IL}	Input Leakage Current	GND≤V_I≤V $_{CC\ Max}$		–	±10	µA
I_{BH}, I_{BL}	Bidirectional Pin Leakage Current	GND≤V_I≤V $_{CC\ Max}$		–	±10	µA
I_{CC}	Operating Power Supply Current	f=15MHz (Q, E) f=25MHz (H) V_{CC} = V $_{CC\ Max}$ V_{IL}=0.5V V_{IH}=3.0V	Half Power	–	115	mA
			Quarter Power (only 15 and 20ns)	–	55	
			Eighth Power (only 15ns)	–	30	
I_{OS}✻	Output Short Circuit Current	V_{CC}=5.0V, V_B=0.5V		-150	-30	mA
V_{OL}	Output Low Voltage	–		–	0.5	V
V_{OH}	Output High Voltage	–		2.4	–	V

Electrical Characteristics Over Operating Conditions (Industrial Temperature Range)

Symbol	Parameter	Test Conditions		Min.	Max.	Units
I_{IH}, I_{IL}	Input Leakage Current	GND≤V_I≤V $_{CC\ Max}$		–	±10	µA
I_{BH}, I_{BL}	Bidirectional Pin Leakage Current	GND≤V_I≤V $_{CC\ Max}$		–	±10	µA
I_{CC}	Operating Power Supply Current	f=15MHz (Q, E) f=25MHz (H) V_{CC} = V $_{CC\ Max}$ V_{IL}=0.5V V_{IH}=3.0V	Half Power	–	130	mA
			Quarter Power (only 15 and 20ns)	–	65	
			Eighth Power (only 15ns)	–	40	
I_{OS}✻	Output Short Circuit Current	V_{CC}=5.0V, V_B=0.5V		-150	-30	mA
V_{OL}	Output Low Voltage	–		–	0.5	V
V_{OH}	Output High Voltage	–		2.4	–	V

✻ One output at a time for a maximum duration of one second.

Switching Characteristics Over Operating Conditions

Symbol	Parameter	From	To	20V8AS 10	20V8AS 12	20V8AS 15	20V8AS 20	Units	Test Cond.✻
				Max.✩	Max.✩	Max.	Max.		
t_{pd}	Combinational Propagation Delay	Input	Output	10	12	15	20	ns	1
t_{co}	Clock to Output Delay	Clock	Registered Output	7	10	10	15	ns	1
t_{en}	Product Term Output Enable to Output	Input	Output	10	12	15	20	ns	2
t_{enr}	Output Register Enable to Output	$\overline{OE}$	Registered Output	10	12	15	18	ns	2
t_{dis}	Product Term Output Disable to Output	Input	Output	10	12	15	20	ns	3
t_{disr}	Output Register Disable to Output	$\overline{OE}$	Registered Output	10	12	15	18	ns	3

AC Operating Conditions

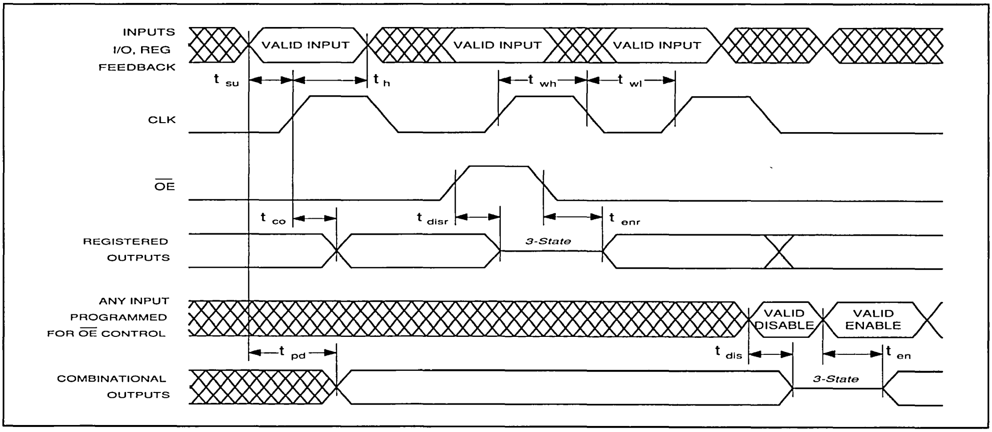
Symbol	Parameter	20V8AS 10✩		20V8AS 12✩		20V8AS 15		20V8AS 20		Units	Test Cond.✻
		Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.		
t_{su}	Input or Feedback Setup Time (Before Clock Rise)	–	10	–	12	–	12	–	15	ns	–
t_h	Input or Feedback Hold Time (After Clock Rise)	–	0	–	0	–	0	–	0	ns	–
t_{wh}	Minimum Clock Width High	–	8	–	8	–	10	–	12	ns	–
t_{wl}	Minimum Clock Width Low	–	8	–	8	–	10	–	12	ns	–
f_{clk}◇	Clock Frequency Without Feedback	62.5	–	62.5	–	50	–	41.7	–	MHz	1
f_{clkf}✦	Clock Frequency With Feedback	58.8	–	48.5	–	41.6	–	33.3	–	MHz	1

✩ Commercial Temperature range only

✻ Refer to “Switching Test Conditions”.

◇ $f_{clk} = \dfrac{1}{t_{wh} + t_{wl}}$ ✦ $f_{clkf} = \dfrac{1}{t_{su} + t_{co}}$

Switching Waveforms



FUNCTIONAL DESCRIPTION

GAL20V8AS has a programmable AND array whose output terms feed a fixed (non programmable) OR array, as bipolar PAL®. The 2 × 10 input lines enter the AND array as true or complemented form. 64 product terms are available allowing standard Sum of Products Logic implementation. Each product term is obtained by appropriate connections between the input lines and the product term line. The connections can be made by programming the E^{2}PROM memory cell at each intersection of the AND matrix (2560 memory cells). The 64 product terms are divided into eight groups of 8 terms each. One product term for each group can be used to provide Output Enable control for combinational output, the others are connected with an OR gate into the corresponding OLMC (Output Logic Macrocell). The output buffer is in 3-state when the corresponding output enable signal is low.

OUTPUT LOGIC MACROCELL (OLMC)

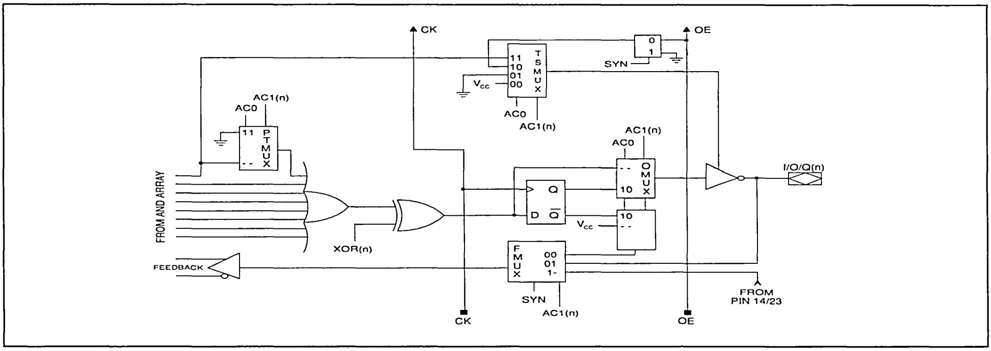
The following discussion pertains to configuring the output logic macrocells. It should be noted that actual implementation is accomplished by development software/hardware and is completely transparent to the user.

The outputs of the AND array are fed into an OLMC, where each output can be individually set to active high or active low, with either combinational (asynchronous) or registered (synchronous) configurations. A common output enable is connected to all registered outputs; product terms can be used to provide individual output enable control for combinational outputs. All outputs have always programmable polarity.

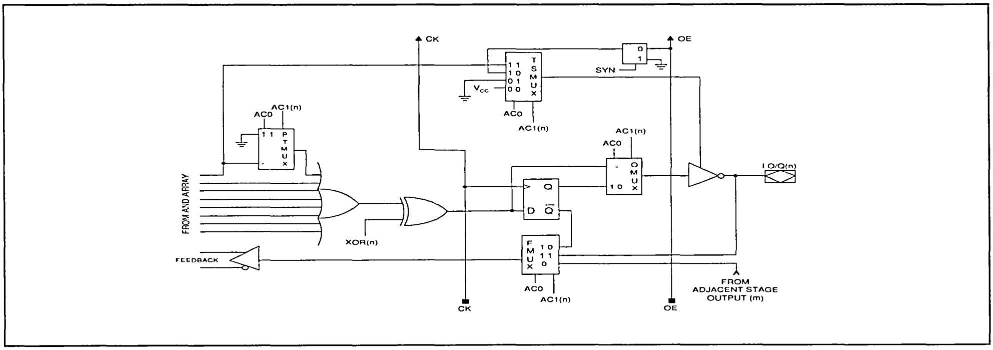
The output logic macrocell provides the designer with maximum output flexibility in matching signal requirements, thus providing more functions than existing 24 pin PAL® devices.

Three different configuration modes of the OLMCs are possible: registered, complex and simple. The output of an OLMC in registered mode can be either registered or combinational. Different modes cannot be mixed: i.e. all OLMCs of a device have to be configured in simple, complex or registered mode.

GAL20V8AS Output Logic Macrocell Pin 15 and 22



GAL20V8AS Output Logic Macrocell Pin 16 to 21

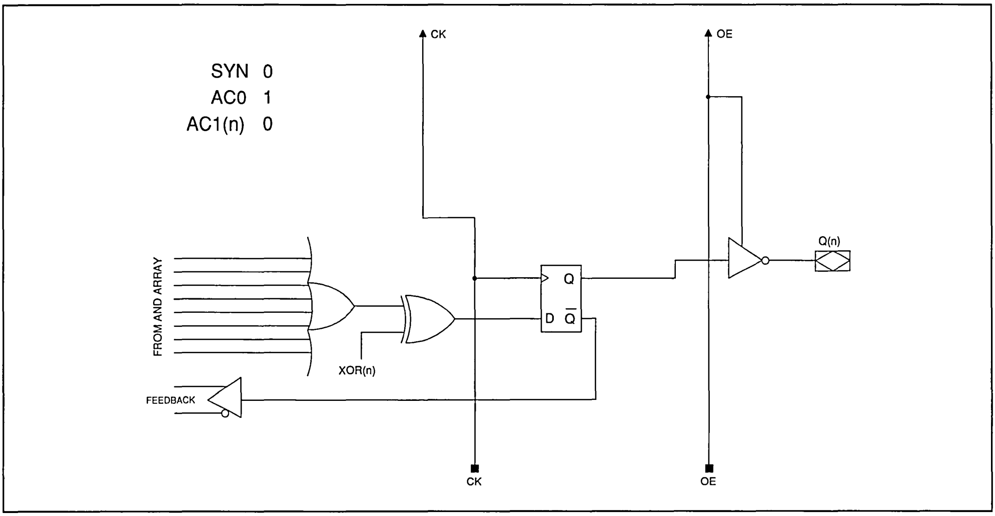


REGISTERED MODE

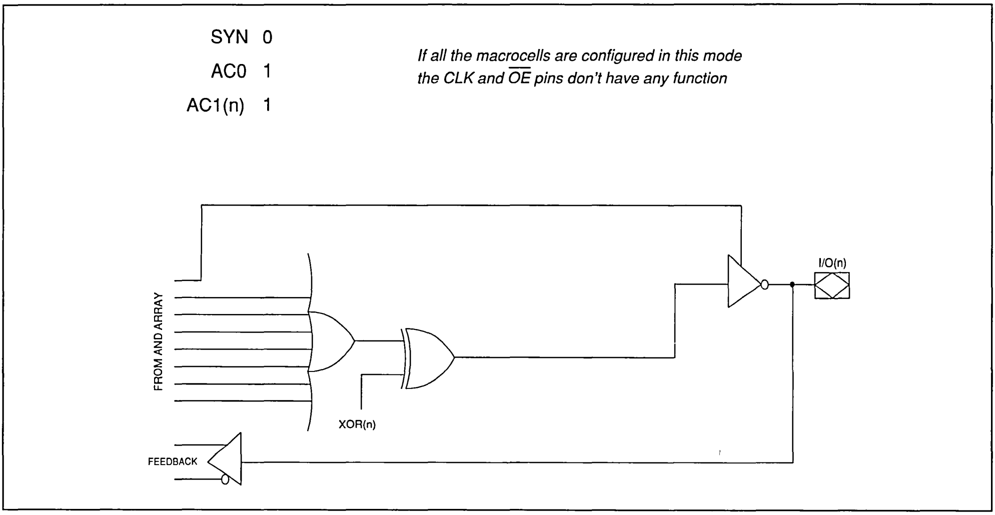
In registered mode macrocells are configured as registered outputs or combinational inputs/outputs. Any macrocell can be configured as registered output or combinational input/output. Up to 8 registered outputs or up to 8 inputs/outputs are possible in this mode.

All registered macrocells share common clock and output enable control. Registered outputs have 8 data product terms per output, while combinational inputs/outputs have only 7 data product terms per output: in the latter case the eighth product term serves as individual output enable control for each macrocell.

Registered Output with Programmable Polarity



Combinational Input/Output with Programmable OE and Polarity



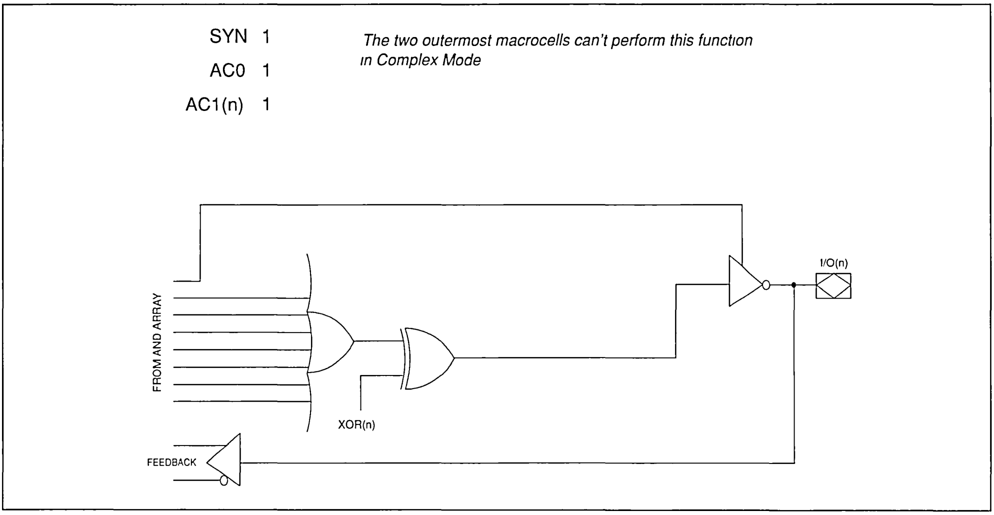
If all the macrocells are configured in this mode the CLK and $\overline{OE}$ pins don't have any function

COMPLEX MODE

In complex mode macrocells are configured as combinational inputs/outputs or outputs only. The two outermost macrocells (15 and 22) do not have input capability: so only 6 inputs/outputs are possible in this mode. Applications requiring 8 inputs/outputs must be implemented in registered mode.

All macrocells have 7 data product terms per output; the eighth product term is used as individual output enable control for each macrocell. The clock and output enable pins (pins 1 and 13 respectively) are always available as inputs.

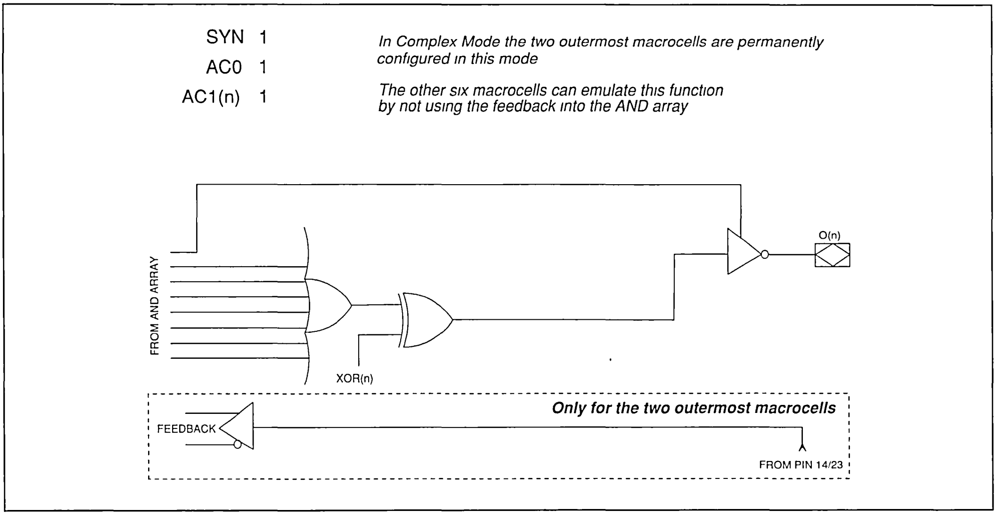
Combinational Input/Output with Programmable OE and Polarity



SYN 1
AC0 1
AC1(n) 1

The two outermost macrocells can't perform this function in Complex Mode

Combinational Output with Programmable OE and Polarity



SYN 1
AC0 1
AC1(n) 1

In Complex Mode the two outermost macrocells are permanently configured in this mode

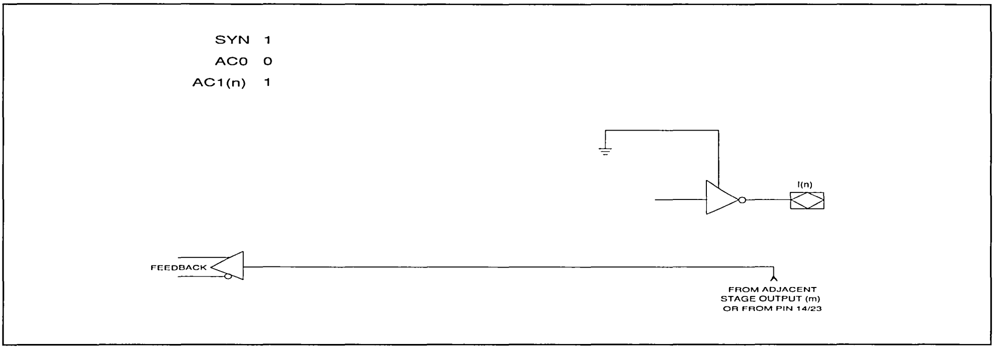
The other six macrocells can emulate this function by not using the feedback into the AND array

Only for the two outermost macrocells

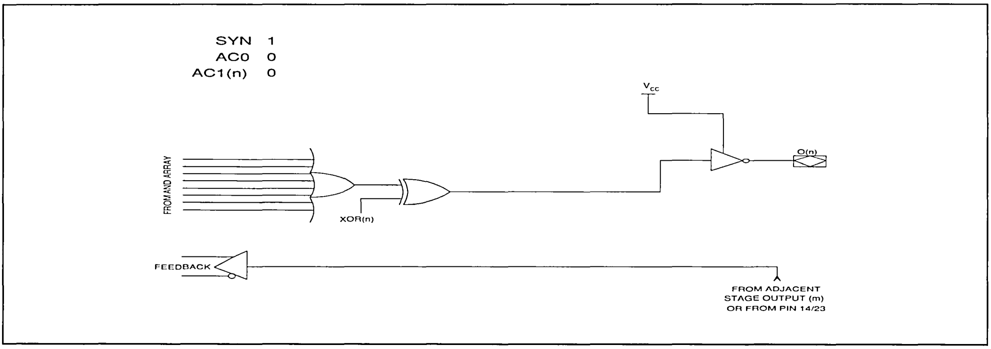
SIMPLE MODE

In simple mode macrocells are configured as dedicated inputs or as dedicated, always active, combinational outputs. Only the two outermost macrocells (15 and 22) can be configured as dedicated inputs. All macrocells have 8 data product terms per output. The clock and output enable pins (pins 1 and 13 respectively) are always available as inputs.

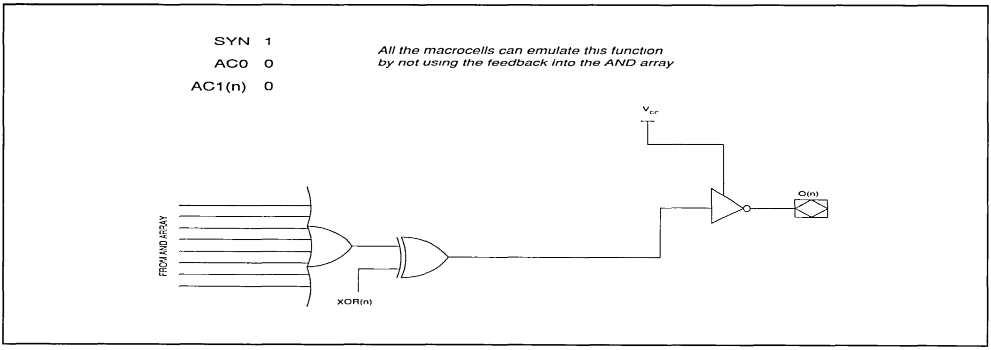
Dedicated Input Mode



Dedicated Combinational Output with Feedback and Programmable Polarity



Dedicated Combinational Output with Programmable Polarity

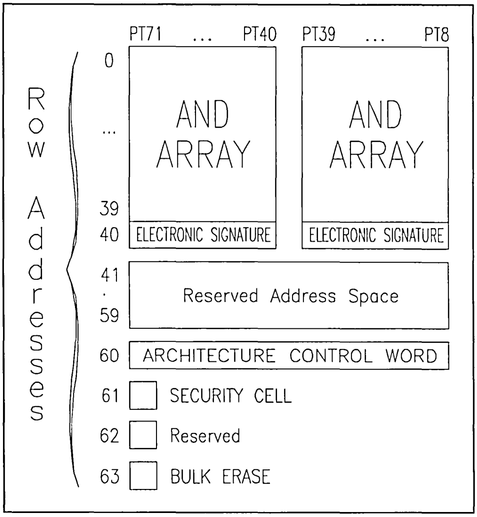

All the macrocells can emulate this function by not using the feedback into the AND array

ROW ADDRESS MAP DESCRIPTION

There are a total of 44 unique row addresses available to the user when programming the GAL20V8AS device. Row addresses 0-39 each contain 64 bits of input term data. This is the AND array where the custom logic pattern is programmed. Row 40 is the Electronic Signature Word. It has 64 bits available for any user defined purpose. Row 41-59 are reserved by the manufacturer and are not available to users.

Row 60 contains the architecture and output polarity information. The 82 bits within this word are programmed to configure the device for a specific application. Row 61 contains a one bit security cell that when programmed prevents further pattern verification of the array. Row 63 is the row that is addressed to perform a bulk erase of the device, resetting it back to a virgin state. Each of these functions is described in the following sections.

GAL20V8AS Row Addresses Map Block Diagram



ELECTRONIC SIGNATURE WORD DESCRIPTION

An electronic signature word is provided with every GAL20V8AS device. It resides at row address 40 and contains 64 bits of reprogrammable memory that can contain user-defined data. Some uses include user ID codes, revision numbers, or inventory control. This signature data is always available to the user independent of the state of the security cell.

ARCHITECTURE CONTROL WORD

All the various output configurations of the GAL20V8AS devices are controlled by programming cells within the 82 bit Architecture Control Word that resides at row 60. The location of specific bits within the Architecture Control Word is shown in the control word diagram in figure below. The function of the SYN, AC0 and AC1(n) bits have been explained in the OUTPUT LOGIC MACROCELL description. The eight polarity bits determine each output's polarity individually. The numbers below the XOR(n) and AC1(n) bits in the architecture control word diagram shows the output device pin number that the polarity bits control.

SECURITY CELL

Row address 61 contains the Security Cell (one bit). The Security Cell is provided on all GAL20V8AS devices as a deterrent to unauthorized copying of the array configuration patterns. Once programmed, the circuitry enabling array access is disabled, preventing further verification of the array (rows 0-39). The cell can be erased only in conjunction with the array during a bulk erase cycle, so the original configuration can never be examined once this cell is programmed. Signature data is always available to the user.

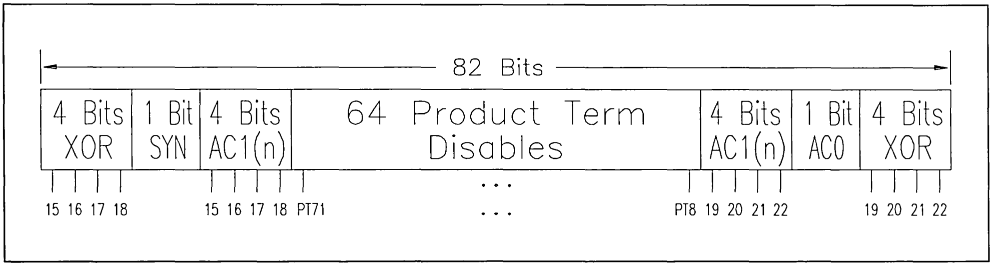
BULK ERASE MODE

By addressing row 63 during a programming cycle, a clear function performs a bulk erase of the array and the Architecture Control Word. In addition, the Electronic Signature Word and the Security Cell are erased. This mode resets a previously configured device back to its virgin state.

OUTPUT REGISTER PRELOAD

When testing state machine designs, all possible states and state transitions must be verified in the

GAL20V8AS Architecture Control Word Diagram



design, not just those required in the normal machine operations. This is because in system operation, certain events occur that may throw the logic into an illegal state (power-up, line voltage glitches, brown-outs, etc.). To test a design for proper management of these conditions, a method must be provided to break the feedback paths, and force any desired (e.g. illegal) state into a register. Then the machine can be sequenced and the outputs tested for the correct next state condition. The GAL20V8AS device includes circuitry that allows each registered output to be synchronously set either high or low. Thus, any present state condition can be forced for test sequencing. If necessary, approved GAL programmers capable of executing test vectors can perform output register preload automatically.

The figure on the right shows the pin functions necessary to preload the register. This test mode is entered by raising PRLD to V_{IES} (register preload input voltage, typically 15V), which enables the serial data in (S_{DIN}) buffer and the serial data out (S_{DOUT}) buffer. Data is then serially shifted into the registers on each rising edge of the clock, D_{CLK}. Only the macrocells with registered output configurations are loaded. If only 3 outputs have registers, then only 3 bits need be shifted in. The registers are loaded from the bottom up as shown in the figure on the right.

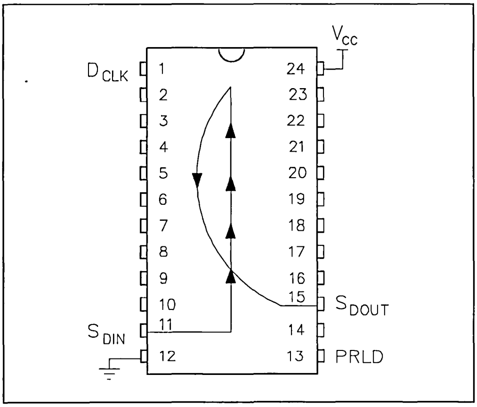
LATCH-UP PROTECTION

GAL® devices are designed with an on board charge pump to negatively bias the substrate. The negative bias is of sufficient magnitude to prevent input undershoots from causing the circuitry to latch. Additionally, outputs are designed with n-channel pullups instead of the traditional p-channel pullups to eliminate any possibility of SCR induced latching.

POWER-UP RESET

Circuitry within the GAL20V8AS provides a reset signal to all registers during power-up. All internal registers will have their Q outputs set low after a specified time (t_{reset}=10µs). As a result, the state on the registered output pins (if they are enabled through $\overline{OE}$) will always be high on power-up, regardless of the programmed polarity of the output pins. This features can greatly simplify state machine design by providing a known state on power-up.

Output Register Preload Pinout

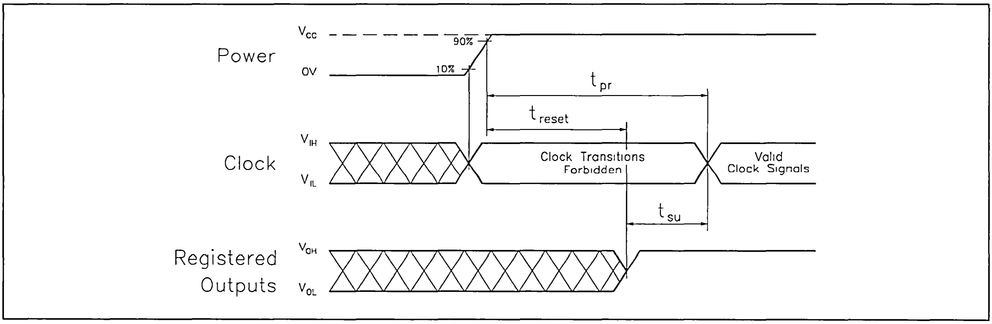


The timing diagram for power-up is shown below. Because of the asynchronous nature of system power-up, the V_{CC} rise must be monotonic to guarantee a valid power-up reset of the GAL20V8AS. The registers will reset within a maximum of t_{reset} time: before this time any clock transition from low to high is forbidden to avoid undesired commutations. As in normal system operation, avoid clocking the device until all input and feedback path setup times have been met (i.e. avoid clocking before the t_{pr}=t_{reset}+t_{su} time interval).

DEVICES PROGRAMMING

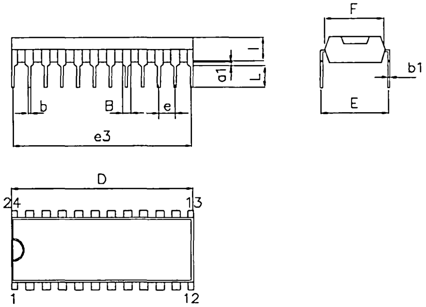
SGS-THOMSON strongly recommends the use of qualified programming hardware. Programming on unapproved equipment will invalidate all guarantees.

Power-Up Reset Timing Diagram



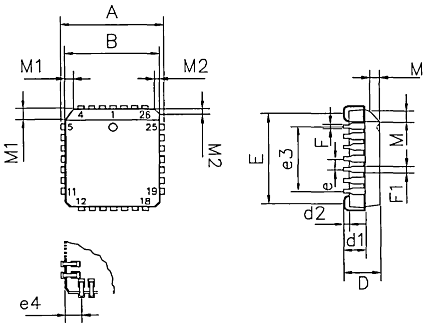
PACKAGE MECHANICAL DATA

PDIP 24 Pins



Dim.	mm			inches		
	Min	Typ	Max	Min	Typ	Max
a1	0.38			0.015		
B	1.27		1.65	0.050		0.065
b		0.46			0.018	
b1		0.25			0.010	
D			31.88			1.255
E		7.62			0.300	
e		2.54			0.100	
e3		27.94			1.100	
F			6.86			0.27
I			4.32			0.170
L		3.30			0.130	

PLCC 28 Pins



Dim.	mm			inches		
	Min	Typ	Max	Min	Typ	Max
A	12.32		12.57	0.485		0.495
B	11.43		11.58	0.450		0.456
D	4.20		4.57	0.165		0.180
d1	2.29		3.04	0.090		0.120
d2	0.51			0.020		
E	9.91		10.92	0.390		0.430
e		1.27			0.050	
e3		7.62			0.300	
e4			1.99			0.078
F		0.46			0.018	
F1		0.71			0.028	
M		1.24			0.049	
M1		1.143			0.045	
M2			0.51			0.020

Seating Plane: 0.101 mm/0.004 inches

Ordering Informations*

SGS-THOMSON GAL®s are available in a variety of package and temperature ranges.
General ordering code is reported below.

GAL20V8AS - s w p t j

PLCC Pinout **J** Jedec Pinout

Temperature **1** 0˚C to +70˚C
3 -40˚C to+85˚C (only for 15 and 20ns Speed selection)

Package **B** 24 Pins PDIP
C 28 Pins PLCC

Power **H** Half Power
Q Quarter Power (only for 15 and 20ns Speed selection)
E Eighth Power (only for 15ns Speed selection)

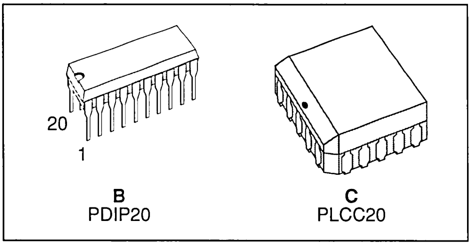
Speed **10** 10ns
12 12ns
15 15ns
20 20ns

Example: ordering code for a GAL20V8AS, 12ns speed and Half Power in PLCC (Jedec pinout) is
GAL20V8AS-12HC1J

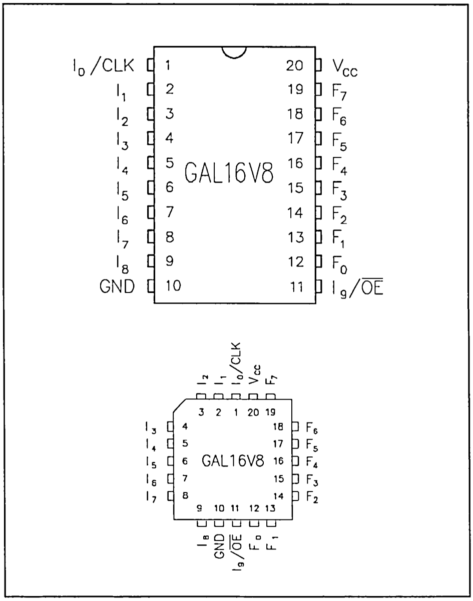
* Please contact local Product Marketing for latest update on package / temperature range availability.

E^2PROM CMOS PROGRAMMABLE LOGIC DEVICE

- HIGH PERFORMANCE SGS-THOMSON SINGLE-POLY E^2PROM CMOS TECHNOLOGY
  - 20ns maximum propagation delay (GAL16V8S-20E*xx*)
  - F_{max} = 41.6MHz
  - 15ns max. from clock input to data output
  - TTL compatible 24mA outputs
  - SGS-THOMSON proprietary Single-Poly F3-G™ technology
- VERY LOW POWER
  - 24mA typ. (27mA max.) I_{CC}
- ELECTRICAL ERASABLE CELL TECHNOLOGY
  - Reconfigurable logic/reprogrammable cells
  - 100% tested: guaranteed 100% final programming yield
  - High speed electrical program & erase
- EIGHT OUTPUT MACROCELLS
  - Maximum flexibility for complex logic design
  - Programmable output polarity
  - Also emulates 21 types of 20 pin PAL® devices with full function/fuse map/parametric compatibility
- PRELOAD AND POWER-ON RESET OF ALL REGISTERS
  - 100% functional testability
- ELECTRONIC SIGNATURE FOR USER'S IDENTIFICATION



Pin Connections



Pin Names

I_0–I_9	Input
CLK	Clock Input
F_0–F_7	I/O
$\overline{OE}$	Output Enable
V_{CC}	Power
GND	Ground

DESCRIPTION

The GAL16V8S, at 20ns maximum propagation delay time, combines a high performance CMOS process with Electrical Erasable Single-Poly F3-G™ technology — SGS-THOMSON proprietary — to provide one of the highest performance-cost2 produsct available in PLD market.

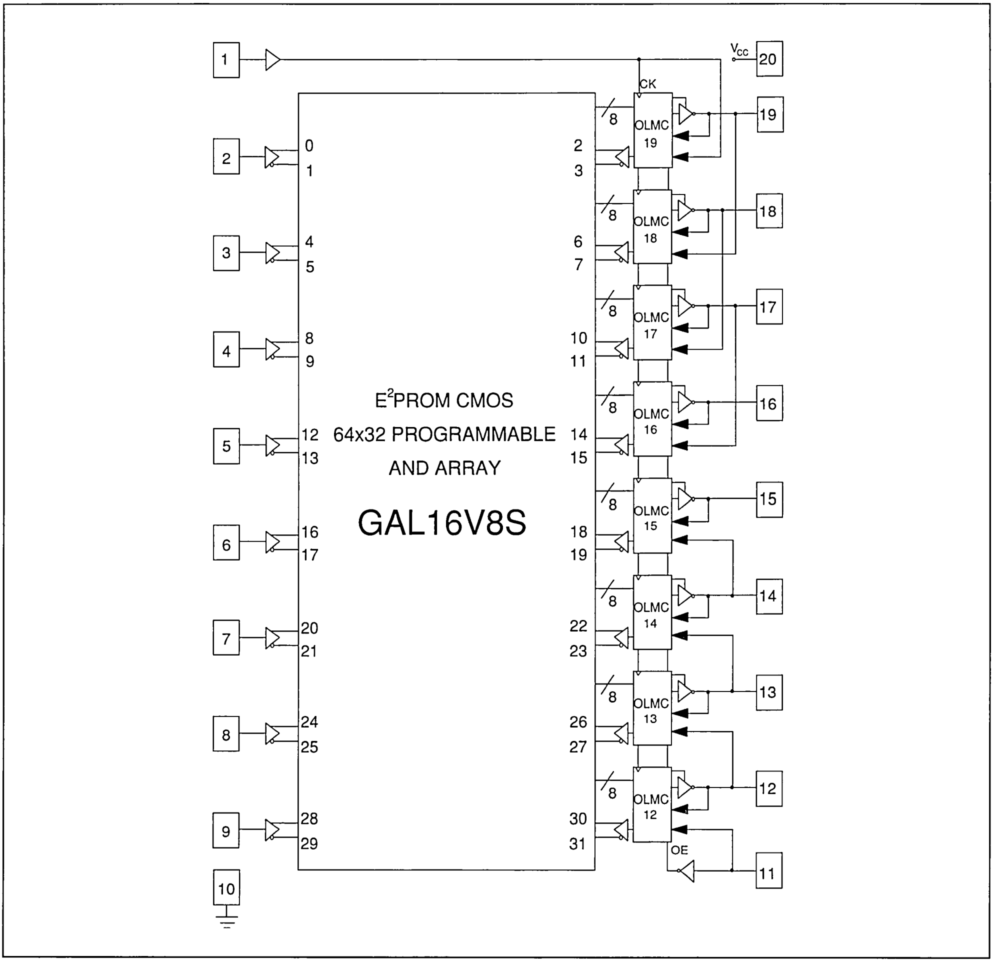
CMOS circuit allows GAL16V8S to consume just 24mA (typ.) I_{CC} which represents a 85% saving in power when compared to its bipolar counterparts. Its E^2PROM CMOS technology offers high speed (50ms) erase time providing the ability to reprogram or reconfigure the device quickly and efficiently.

GAL16V8S features 8 programmable Output Logic Macro Cells (OLMCs) allowing each output to be configured by the user. Additionally, the GAL16V8S is capable of emulating, in a functional/fuse

GAL® is a registered trademark of Lattice Semiconductor Corp.; PAL® is a registered trademark of Monolithic Memories Inc map/parametric compatible mode, 21 types of 20 pin PAL® devices. Unique test circuits and reprogrammable cells allow complete AC, DC and functional testing during manufacture.

Therefore, SGS-THOMSON guarantees 100% field programmability and functionality of GAL® devices. SGS-THOMSON also guarantees 100 erase/write cycles and data retention exceeding 20 years.

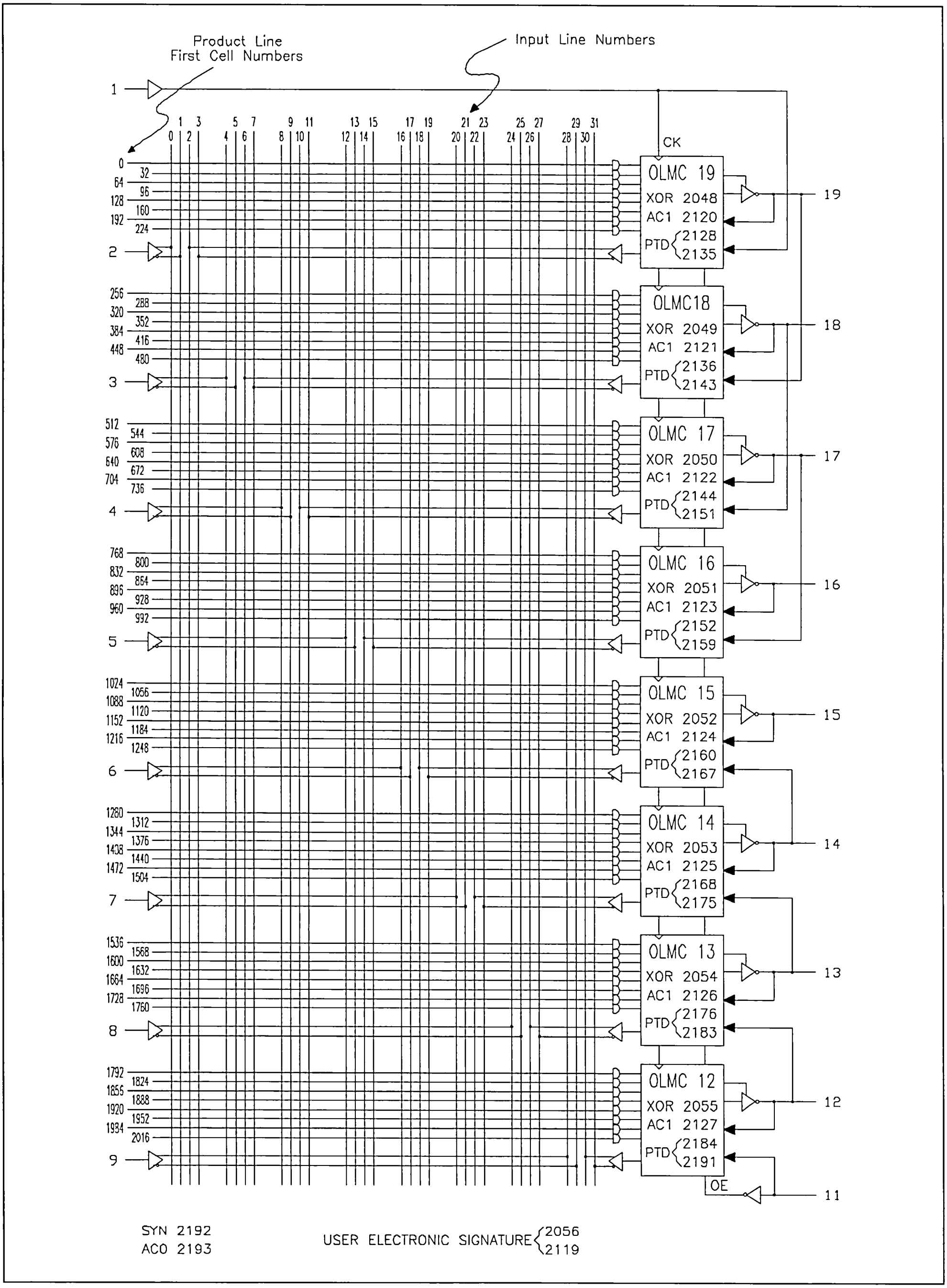
GAL16V8S Block Diagram



GAL16V8S PAL® Architecture Emulation

16L8	16R8	16RP6	16L2	14L4	12L6	10L8
16H8	16RP8	16R4	16H2	14H4	12H6	10H8
16P8	16R6	16RP4	16P2	14P4	12P6	10P8

GAL16V8S Logic Diagram



Absolute Maximum Ratings

Symbol	Parameter	Value	Unit
V_{CC}	Supply Voltage	-0.5 to +7	V
V_I	Input Voltage Applied	-2.5 to V_{CC}+1	V
V_B	Off-State Output (Bidirectional) Voltage Applied	-2.5 to V_{CC}+1	V
T_{STG}	Storage Temperature	-65 to +125	˚C
T_J	Junction Temperature (Operating)	-40 to +125	˚C
T_L	Lead Temperature (Soldering)	260 (for 10s max.)	˚C

Stresses above those listed under the “Absolute Maximum Ratings” may cause permanent damage to the device. These are stress only ratings and functional operation of the device at these or at any other conditions above those indicated in the operational sections of this specification is not implied.

ESD Immunity
Test Method: Human Body Model (HBM)
ESD Tolerance ≥ 2000V
(See MIL-STD 883c).

Test Method: Charge Device Model (CDM)
ESD Tolerance ≥ 500V
Test Instrument: KeyTek ZapMaster

CDM is an additional test only for GAL®s not yet adopted as a company standard test.

Switching Test Conditions

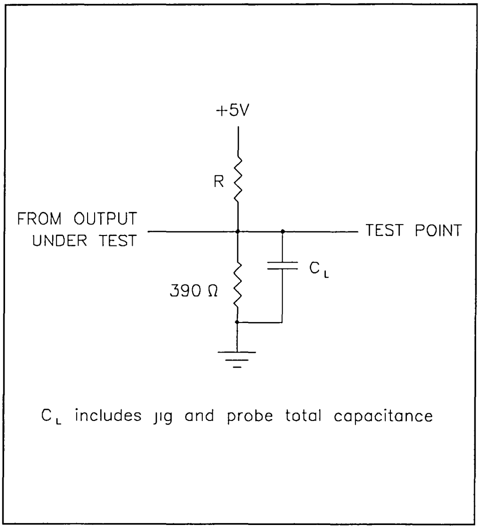
Input Pulse Levels	GND to 3.0V
Input Rise and Fall Times	3ns 10%-90%
Input Timing Reference Levels	1.5V
Output Timing Reference Levels	1.5V
Output Load	See figure

3-state levels are measured 0.5V from steady-state active level.

Test Conditions

#	R [Ω]	C_L [pF]
1	200	50
2	Active High: ∞ Active Low: 200	50
3	Active High: ∞ Active Low: 200	5

Switching Test Circuit



Capacitance (T_A=25˚C, f=1.0MHz, V_{CC}=5V)

Symbol	Parameter	Test Conditions	Maximum[*]	Units
C_I	Input Capacitance	V_I=2V	8	pF
C_B	Bidirectional Pin Capacitance	V_B=2V	10	pF

[*] Guarantied but not 100% tested.

DC Operating Conditions

Symbol	Parameter	Commercial Temperature Range		Industrial Temperature Range		Units
		Min.	Max.	Min.	Max.	
V_{CC}	Supply Voltage	4.75	5.25	4.5	5.5	V
T_A	Ambient Temperature	0	70	-40	85	˚C
V_{IL}	Input Low Voltage	VSS[†]-0.5	0.8	V_{SS}[†]-0.5	0.8	V
V_{IH}	Input High Voltage	2.0	V_{CC}+1	2.0	V_{CC}+1	V
I_{OL}	Low Level Output Current	–	24	–	24	mA
I_{OH}	High Level Output Current	-3.2	–	-3.2	–	mA

[†] V_{SS} is the voltage applied to the GND pin

Electrical Characteristics Over Operating Conditions (Commercial Temperature Range)

Symbol	Parameter	Test Conditions	Min.	Max.	Units
I_{IH}, I_{IL}	Input Leakage Current	GND≤V_I≤V CC_{Max}	–	±10	µA
I_{BH}, I_{BL}	Bidirectional Pin Leakage Current	GND≤V_I≤V CC_{Max}	–	±10	µA
I_{CC}	Operating Power Supply Current	f=15MHz V_{CC} = V CC_{Max} V_{IL}=0.5V V_{IH}=3.0V	–	27	mA
I_{OS}[*]	Output Short Circuit Current	V_{CC}=5.0V, V_B=0.5V	-150	-30	mA
V_{OL}	Output Low Voltage	–	–	0.5	V
V_{OH}	Output High Voltage	–	2.4	–	V

Electrical Characteristics Over Operating Conditions (Industrial Temperature Range)

Symbol	Parameter	Test Conditions	Min.	Max.	Units
I_{IH}, I_{IL}	Input Leakage Current	GND≤V_I≤V CC_{Max}	–	±10	µA
I_{BH}, I_{BL}	Bidirectional Pin Leakage Current	GND≤V_I≤V CC_{Max}	–	±10	µA
I_{CC}	Operating Power Supply Current	f=15MHz V_{CC} = V CC_{Max} V_{IL}=0.5V V_{IH}=3.0V	–	36	mA
I_{OS}[*]	Output Short Circuit Current	V_{CC}=5.0V, V_B=0.5V	-150	-30	mA
V_{OL}	Output Low Voltage	–	–	0.5	V
V_{OH}	Output High Voltage	–	2.4	–	V

[*] One output at a time for a maximum duration of one second

Switching Characteristics Over Operating Conditions

Symbol	Parameter	From	To	Max.	Units	Test Cond.*
t_{pd}	Combinational Propagation Delay	Input	Output	20	ns	1
t_{co}	Clock to Output Delay	Clock	Registered Output	15	ns	1
t_{en}	Product Term Output Enable to Output	Input	Output	20	ns	2
t_{enr}	Output Register Enable to Output	$\overline{OE}$	Registered Output	18	ns	2
t_{dis}	Product Term Output Disable to Output	Input	Output	20	ns	3
t_{disr}	Output Register Disable to Output	$\overline{OE}$	Registered Output	18	ns	3

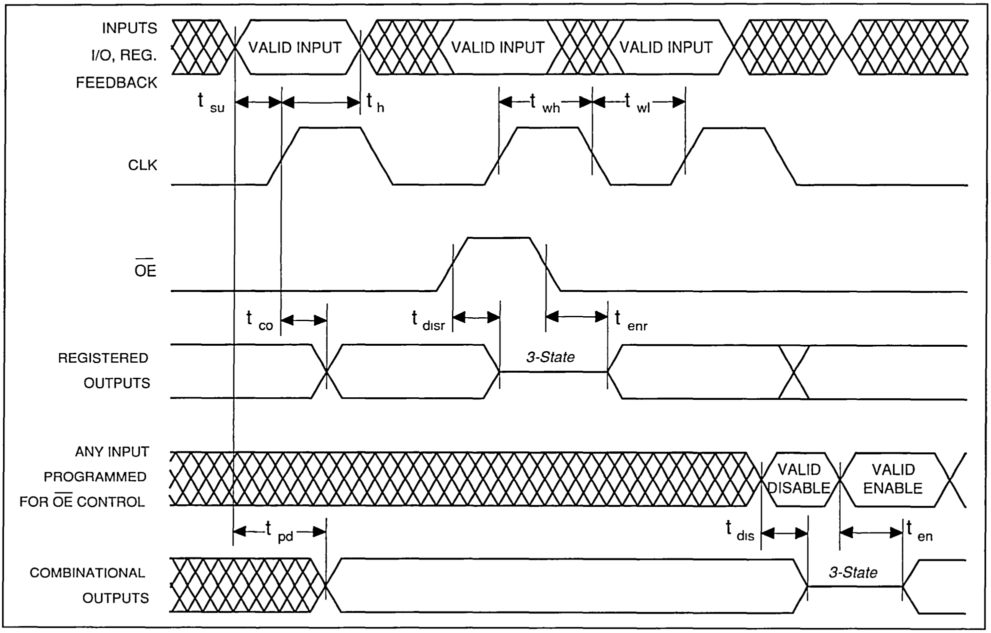
AC Operating Conditions

Symbol	Parameter	Min.	Max.	Units	Test Cond.*
t_{su}	Input or Feedback Setup Time (Before Clock Rise)	–	15	ns	–
t_h	Input or Feedback Hold Time (After Clock Rise)	–	0	ns	–
t_{wh}	Minimum Clock Width High	–	12	ns	–
t_{wl}	Minimum Clock Width Low	–	12	ns	–
f_{clk}✧	Clock Frequency Without Feedback	41.6	–	MHz	1
f_{clkf}✦	Clock Frequency With Feedback	33.3	–	MHz	1

* Refer to "Switching Test Conditions".

✧ $f_{clk} = \frac{1}{t_{wh} + t_{wl}}$ ✦ $f_{clkf} = \frac{1}{t_{su} + t_{co}}$

Switching Waveforms



FUNCTIONAL DESCRIPTION

GAL16V8S has a programmable AND array whose output terms feed a fixed (non programmable) OR array, as bipolar PAL®. The 2 × 8 input lines enter the AND array as true or complemented form. 64 product terms are available allowing standard Sum of Products Logic implementation. Each product term is obtained by appropriate connections between the input lines and the product term line. The connections can be made by programming the E^{2}PROM memory cell at each intersection of the AND matrix (2048 memory cells). The 64 product terms are divided into eight groups of 8 terms each. One product term for each group can be used to provide Output Enable control for combinational output, the others are connected with an OR gate into the corresponding OLMC (Output Logic Macrocell). The output buffer is in 3-state when the corresponding output enable signal is low.

OUTPUT LOGIC MACROCELL (OLMC)

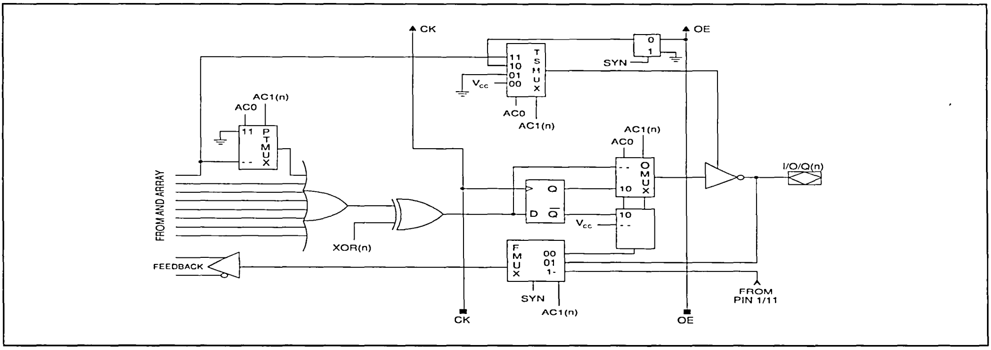
The following discussion pertains to configuring the output logic macrocells. It should be noted that actual implementation is accomplished by development software/hardware and is completely transparent to the user.

The outputs of the AND array are fed into an OLMC, where each output can be individually set to active high or active low, with either combinational (asynchronous) or registered (synchronous) configurations. A common output enable is connected to all registered outputs; product terms can be used to provide individual output enable control for combinational outputs. All outputs have always programmable polarity.

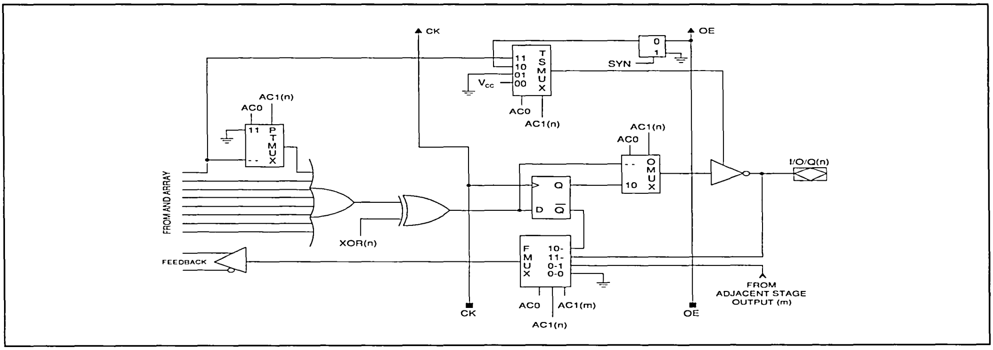
The output logic macrocell provides the designer with maximum output flexibility in matching signal requirements, thus providing more functions than existing 20 pin PAL® devices.

Three different configuration modes of the OLMCs are possible: registered, complex and simple. The output of an OLMC in registered mode can be either registered or combinational. Different modes cannot be mixed: i.e. all OLMCs of a device have to be configured in simple, complex or registered mode.

GAL16V8S Output Logic Macrocell Pin 12 and 19



GAL16V8S Output Logic Macrocell Pin 13 to 18

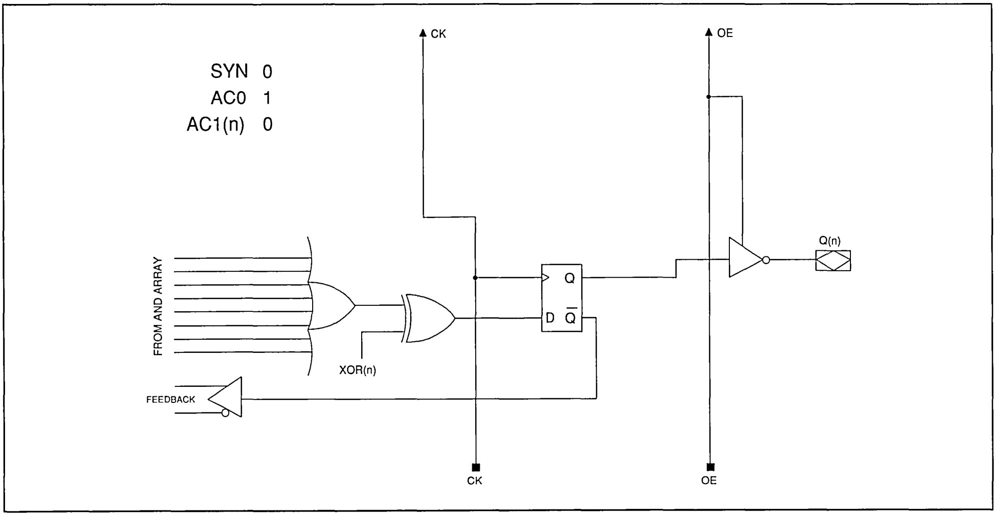


REGISTERED MODE

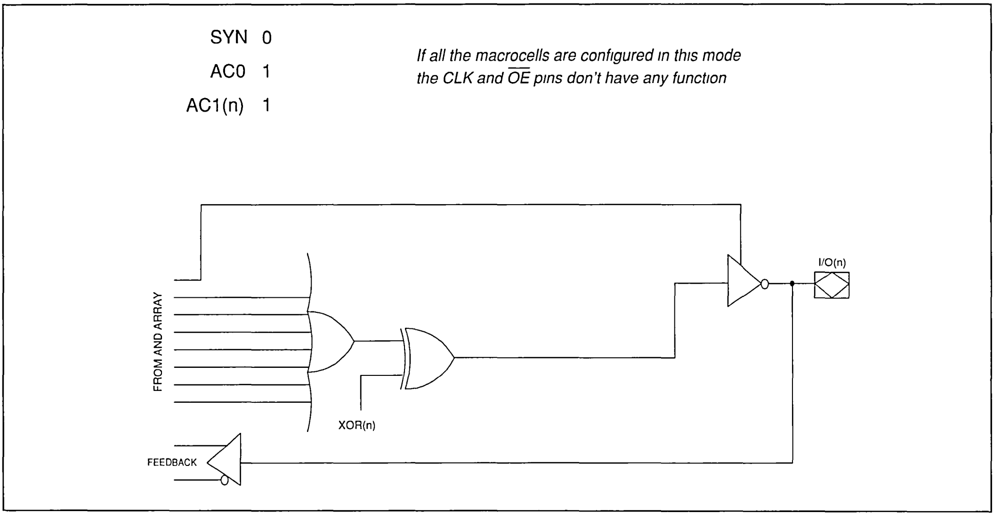
In registered mode macrocells are configured as registered outputs or combinational inputs/outputs. Any macrocell can be configured as registered output or combinational input/output. Up to 8 registered outputs or up to 8 inputs/outputs are possible in this mode.

All registered macrocells share common clock and output enable control. Registered outputs have 8 data product terms per output, while combinational inputs/outputs have only 7 data product terms per output: in the latter case the eighth product term serves as individual output enable control for each macrocell.

Registered Output with Programmable Polarity



Combinational Input/Output with Programmable OE and Polarity


SYN 0
AC0 1
AC1(n) 1
If all the macrocells are configured in this mode the CLK and $\overline{OE}$ pins don't have any function
FROM AND ARRAY
I/O(n)
XOR(n)
FEEDBACK

COMPLEX MODE

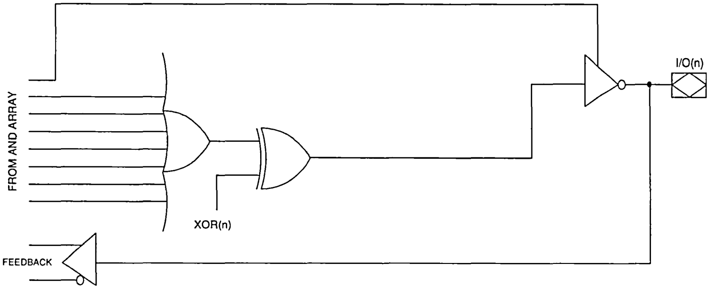
In complex mode macrocells are configured as combinational inputs/outputs or outputs only. The two outermost macrocells (12 and 19) do not have input capability: so only up to 6 inputs/outputs are possible in this mode. Applications requiring 8 inputs/outputs must be implemented in registered mode.

All macrocells have 7 data product terms per output; the eighth product term is used as individual output enable control for each macrocell. The clock and output enable pins (pins 1 and 11 respectively) are always available as inputs.

Combinational Input/Output with Programmable OE and Polarity

SYN 1
AC0 1
AC1(n) 1

The two outermost macrocells can't perform this function in Complex Mode

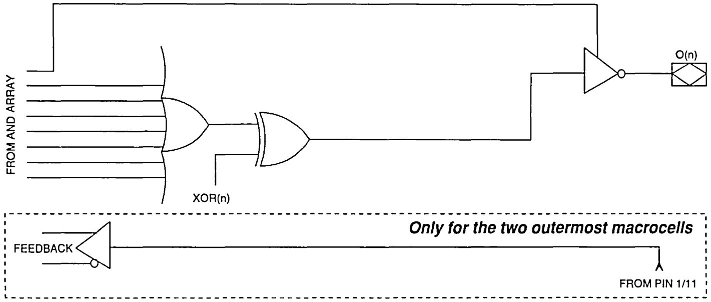


Combinational Output with Programmable OE and Polarity

SYN 1
AC0 1
AC1(n) 1

In Complex Mode the two outermost macrocells are permanently configured in this mode

The other six macrocells can emulate this function by not using the feedback into the AND array

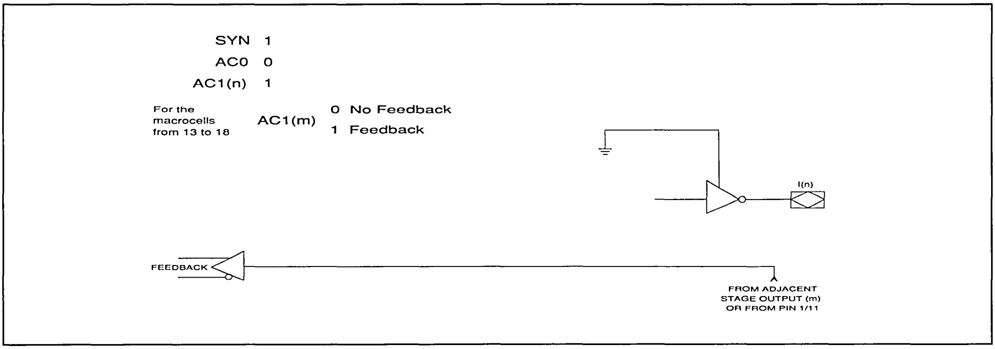


SIMPLE MODE

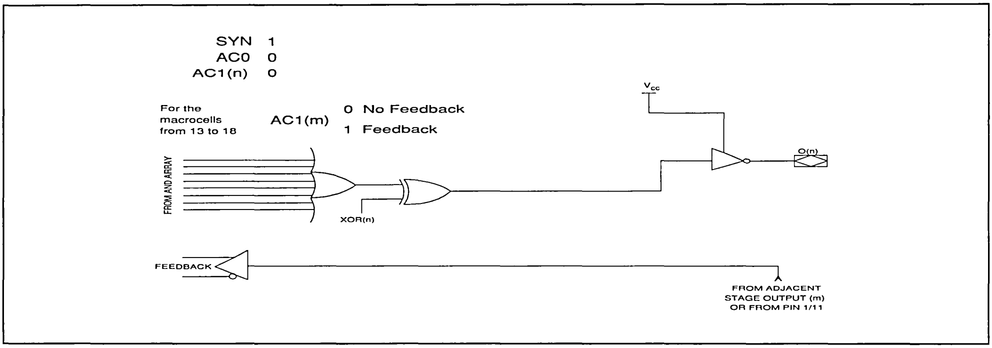
In simple mode macrocells are configured as dedicated inputs or as dedicated, always active, combinational outputs. The two central macrocells (15 and 16) cannot be used in the input configuration.

All macrocells have 8 data product terms per output. The clock and output enable pins (pins 1 and 11 respectively) are always available as inputs.

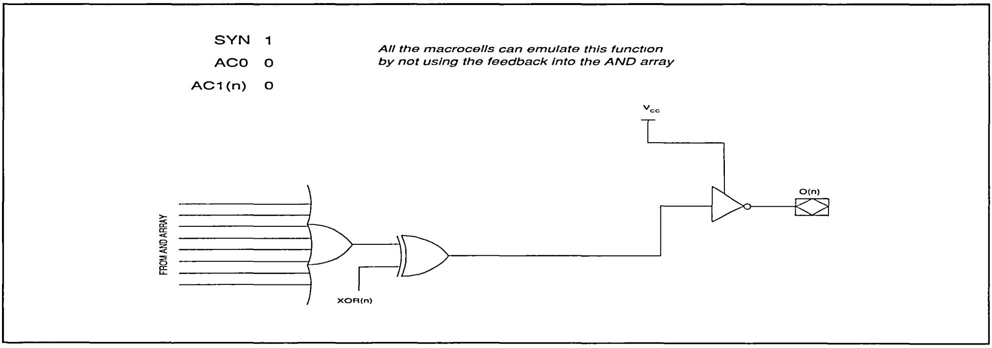
Dedicated Input Mode


SYN 1
AC0 0
AC1(n) 1
For the macrocells from 13 to 18
AC1(m)
0 No Feedback
1 Feedback
I(n)
FEEDBACK
FROM ADJACENT STAGE OUTPUT (m) OR FROM PIN 1/11

Dedicated Combinational Output with Feedback and Programmable Polarity



Dedicated Combinational Output with Programmable Polarity

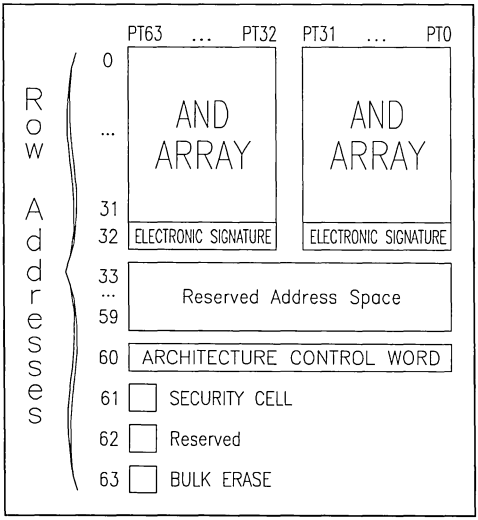

SYN 1
AC0 0
AC1(n) 0
All the macrocells can emulate this function by not using the feedback into the AND array
Vcc
O(n)
FROM AND ARRAY
XOR(n)

ROW ADDRESS MAP DESCRIPTION

There are a total of 36 unique row addresses available to the user when programming the GAL16V8S device. Row addresses 0-31 each contain 64 bits of input term data. This is the AND array where the custom logic pattern is programmed. Row 32 is the Electronic Signature Word. It has 64 bits available for any user defined purpose. Row 33-59 are reserved by the manufacturer and are not available to users.

Row 60 contains the architecture and output polarity information. The 82 bits within this word are programmed to configure the device for a specific application. Row 61 contains a one bit security cell that when programmed prevents further pattern verification of the array. Row 63 is the rcw that is addressed to perform a bulk erase of the device, resetting it back to a virgin state. Each of these functions is described in the following sections.

GAL16V8S Row Addresses Map Block Diagram



ELECTRONIC SIGNATURE WORD DESCRIPTION

An electronic signature word is provided with every GAL16V8S device. It resides at row address 32 and contains 64 bits of reprogrammable memory that can contain user-defined data. Some uses include user ID codes, revision numbers, or inventory control. This signature data is always available to the user independent of the state of the security cell.

ARCHITECTURE CONTROL WORD

All the various output configurations of the GAL16V8S devices are controlled by programming cells within the 82 bit Architecture Control Word that resides at row 60. The location of specific bits within the Architecture Control Word is shown in the control word diagram in figure below. The function of the SYN, AC0 and AC1(n) bits have been explained in the OUTPUT LOGIC MACROCELL description. The eight polarity bits determine each output's polarity individually. The numbers below the XOR(n) and AC1(n) bits in the architecture control word diagram shows the output device pin number that the polarity bits control.

SECURITY CELL

Row address 61 contains the Security Cell (one bit). The Security Cell is provided on all GAL16V8S devices as a deterrent to unauthorized copying of the array configuration patterns. Once programmed, the circuitry enabling array access is disabled, preventing further verification of the array (rows 0-31). The cell can be erased only in conjunction with the array during a bulk erase cycle, so the original configuration can never be examined once this cell is programmed. Signature data is always available to the user.

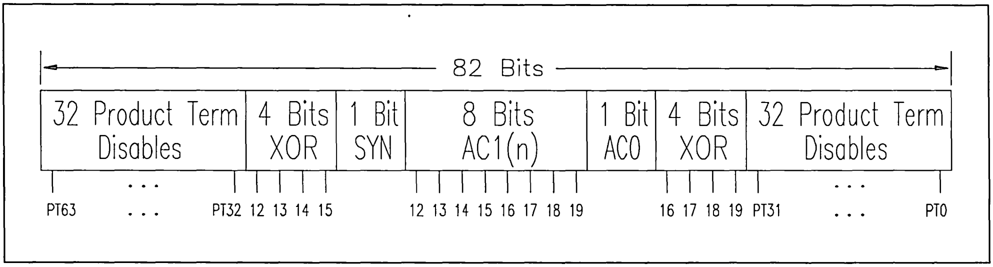
BULK ERASE MODE

By addressing row 63 during a programming cycle, a clear function performs a bulk erase of the array and the Architecture Control Word. In addition, the Electronic Signature Word and the Security Cell are erased. This mode resets a previously configured device back to its virgin state.

OUTPUT REGISTER PRELOAD

When testing state machine designs, all possible states and state transitions must be verified in the

GAL16V8S Architecture Control Word Diagram



design, not just those required in the normal machine operations. This is because in system operation, certain events occur that may throw the logic into an illegal state (power-up, line voltage glitches, brown-outs, etc.). To test a design for proper management of these conditions, a way must be provided to break the feedback paths, and force any desired (e.g. illegal) state into a register. Then the machine can be sequenced and the outputs tested for the correct next state condition. The GAL16V8S device includes circuitry that allows each registered output to be synchronously set either high or low. Thus, any present state condition can be forced for test sequencing. If necessary, approved GAL programmers capable of executing test vectors can perform output register preload automatically.

The figure on the right shows the pin functions necessary to preload the register. This test mode is entered by raising PRLD to V_{IES} (register preload input voltage, typically 15V), which enables the serial data in (S_{DIN}) buffer and the serial data out (S_{DOUT}) buffer. Data is then serially shifted into the registers on each rising edge of the clock, D_{CLK}. Only the macrocells with registered output configurations are loaded. If only 3 outputs have registers, then only 3 bits need be shifted in. The registers are loaded from the bottom up as shown in the figure on the right.

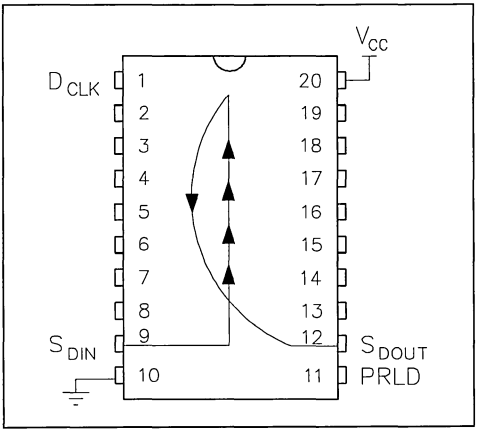
LATCH-UP PROTECTION

GAL® devices are designed with an on board charge pump to negatively bias the substrate. The negative bias is of sufficient magnitude to prevent input undershoots from causing the circuitry to latch. Additionally, outputs are designed with n-channel pullups instead of the traditional p-channel pullups to eliminate any possibility of SCR induced latching.

POWER-UP RESET

Circuitry within the GAL16V8S provides a reset signal to all registers during power-up. All internal registers will have their Q outputs set low after a specified time (t_{reset}=10μs). As a result, the state on the registered output pins (if they are enabled through $\overline{OE}$) will always be high on power-up, regardless of the programmed polarity of the output pins. This features can greatly simplify state machine design by providing a known state on power-up.

Output Register Preload Pinout

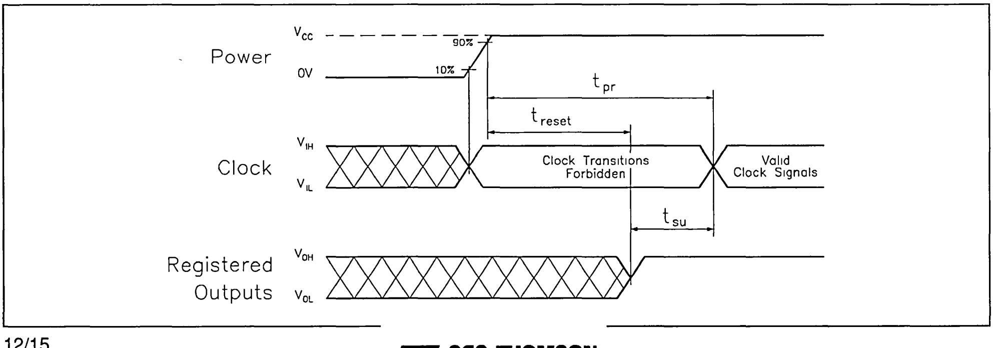


The timing diagram for power-up is shown below. Because of the asynchronous nature of system power-up, the V_{CC} rise must be monotonic to guarantee a valid power-up reset of the GAL16V8S. The registers will reset within a maximum of t_{reset} time: before this time any clock transition from low to high is forbidden to avoid undesired commutations. As in normal system operation, avoid clocking the device until all input and feedback path setup times have been met (i.e. avoid clocking before the $t_{pr}=t_{reset}+t_{su}$ time interval).

DEVICES PROGRAMMING

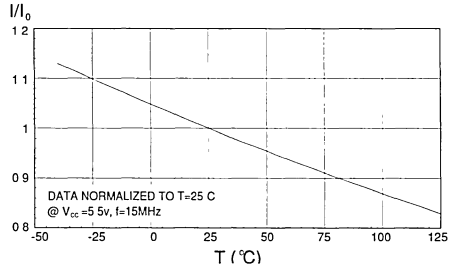
SGS-THOMSON strongly recommends the use of qualified programming hardware. Programming on unapproved equipment will invalidate all guarantees.

Power-Up Reset Timing Diagram

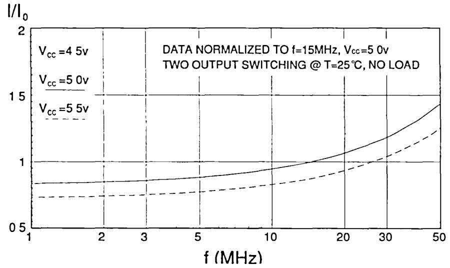


TYPICAL CHARACTERISTICS

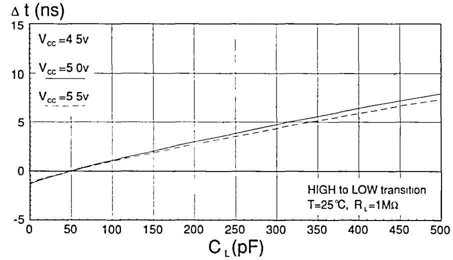
OPERATING CURRENT vs TEMPERATURE



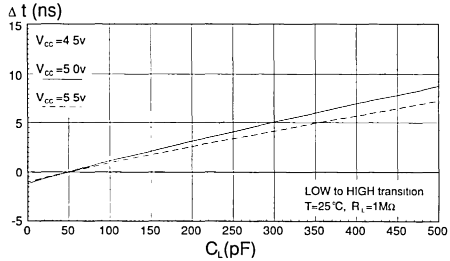
OPERATING CURRENT vs FREQUENCY



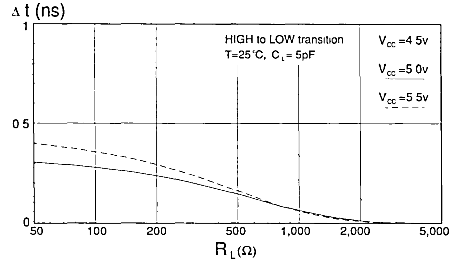
RELATIVE DELAY vs CAPACITIVE LOAD



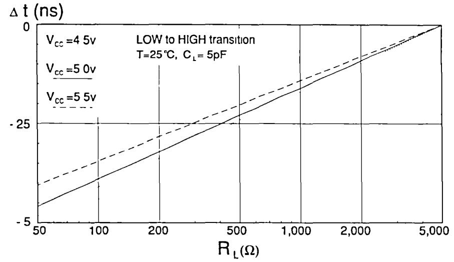
RELATIVE DELAY vs CAPACITIVE LOAD



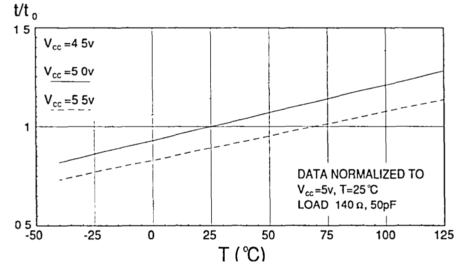
RELATIVE DELAY vs PULL-UP RESISTOR



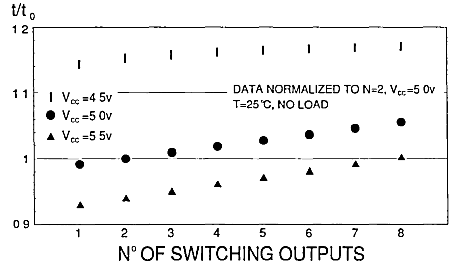
RELATIVE DELAY vs PULL-UP RESISTOR



NORMALIZED DELAY vs TEMPERATURE

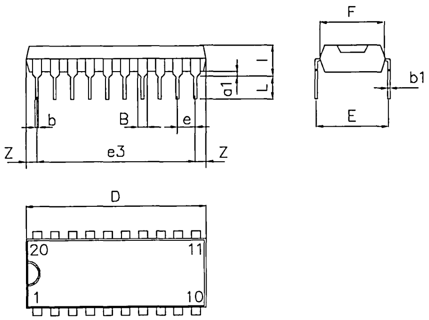


NORMALIZED t_{PD} vs SWITCHING OUTPUTS



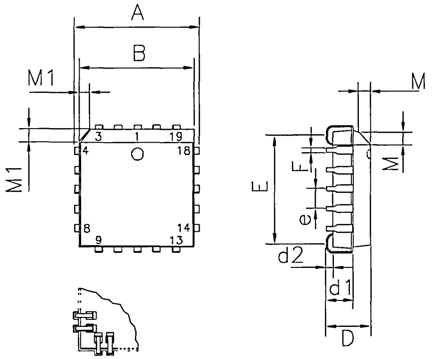
PACKAGE MECHANICAL DATA

PDIP 20 Pins



Dim.	mm			inches		
	Min	Typ	Max	Min	Typ	Max
a1	0.254			0.010		
B	1.39		1.65	0.054		0.064
b		0.45			0.017	
b1		0.25			0.009	
D			25.4			1.000
E		8.50			0.334	
e		2.54			0.100	
e3		22.86			0.900	
F			7.10			0.279
I			3.93			0.154
L		3.30			0.129	
Z		1.27	1.34		0.050	0.052

PLCC 20 Pins



Dim.	mm			inches		
	Min	Typ	Max	Min	Typ	Max
A	9.78		10.03	0.385		0.395
B	8.89		9.04	0.350		0.356
D	4.20		4.57	0.165		0.180
d1		2.54			0.100	
d2		0.56			0.022	
E	7.37		8.38	0.290		0.330
e		1.27			0.050	
F		0.38			0.015	
M		1.27			0.050	
M1		1.14			0.045	

Seating Plane: 0.101 mm/0.004 inches

Ordering Informations*

SGS-THOMSON GAL®s are available in a variety of package and temperature ranges.
General ordering code is reported below.

GAL16V8S - s w p t

Temperature	**1**	0°C to +70°C
	3	-40°C to +85°C
Package	**B**	20 Pins PDIP
	C	20 Pins PLCC
Power	**E**	Eighth Power
Speed	**20**	20ns

Example: ordering code for a GAL16V8S, 20ns speed and Eighth Power in PDIP is
GAL16V8S-20EB1

* Please contact local Product Marketing for latest update on package / temperature range availability.

SGS-THOMSON
MICROELECTRONICS

GAL20V8S

E^{2}PROM CMOS PROGRAMMABLE LOGIC DEVICE

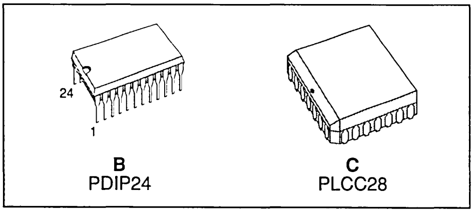
- HIGH PERFORMANCE SGS-THOMSON SINGLE-POLY E^{2}PROM CMOS TECHNOLOGY
  - 20ns maximum propagation delay (GAL20V8S-20E*xx*)
  - F_{max} = 41.6MHz
  - 15ns max. from clock input to data output
  - TTL compatible 24mA outputs
  - SGS-THOMSON proprietary Single-Poly F3-G™ technology
- VERY LOW POWER
  - 24mA typ. (27mA max.) I_{CC}
- ELECTRICAL ERASABLE CELL TECHNOLOGY
  - Reconfigurable logic/reprogrammable cells
  - 100% tested: guaranteed 100% final programming yield
  - High speed electrical program & erase
- EIGHT OUTPUT MACROCELLS
  - Maximum flexibility for complex logic design
  - Programmable output polarity
  - Also emulates 21 types of 24 pin PAL® devices with full function/fuse map/parametric compatibility
- PRELOAD AND POWER-ON RESET OF ALL REGISTERS
  - 100% functional testability
- ELECTRONIC SIGNATURE FOR USER'S IDENTIFICATION

DESCRIPTION

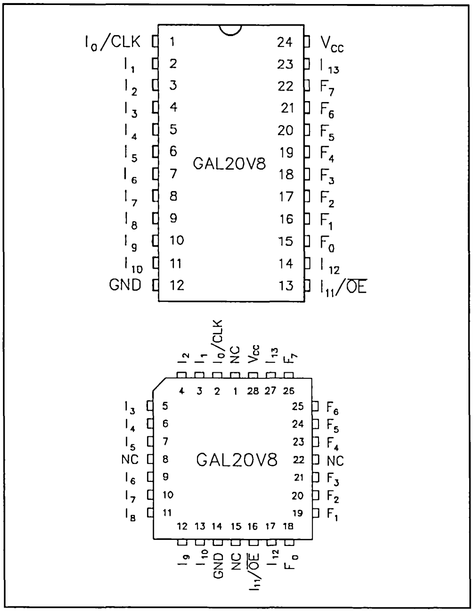
The GAL20V8S, at 20ns maximum propagation delay time, combines a high performance CMOS process with Electrical Erasable Single-Poly F3-G™ technology — SGS-THOMSON proprietary — to provide one of the highest performance-cost products available in PLD market.

CMOS circuit allows GAL20V8S to consume just 24mA (typ.) I_{CC} which represents a 85% saving in power when compared to its bipolar counterparts. Its E^{2}PROM CMOS technology offers high speed (50ms) erase time providing the ability to reprogram or reconfigure the device quickly and efficiently.

GAL20V8S features 8 programmable Output Logic Macro Cells (OLMCs) allowing each output to be configured by the user. Additionally, the GAL20V8S is capable of emulating, in a functional/fuse



Pin Connections



Pin Names

I_0–I_{13}	Input
CLK	Clock Input
F_0–F_7	I/O
$\overline{OE}$	Output Enable
V_{CC}	Power
GND	Ground

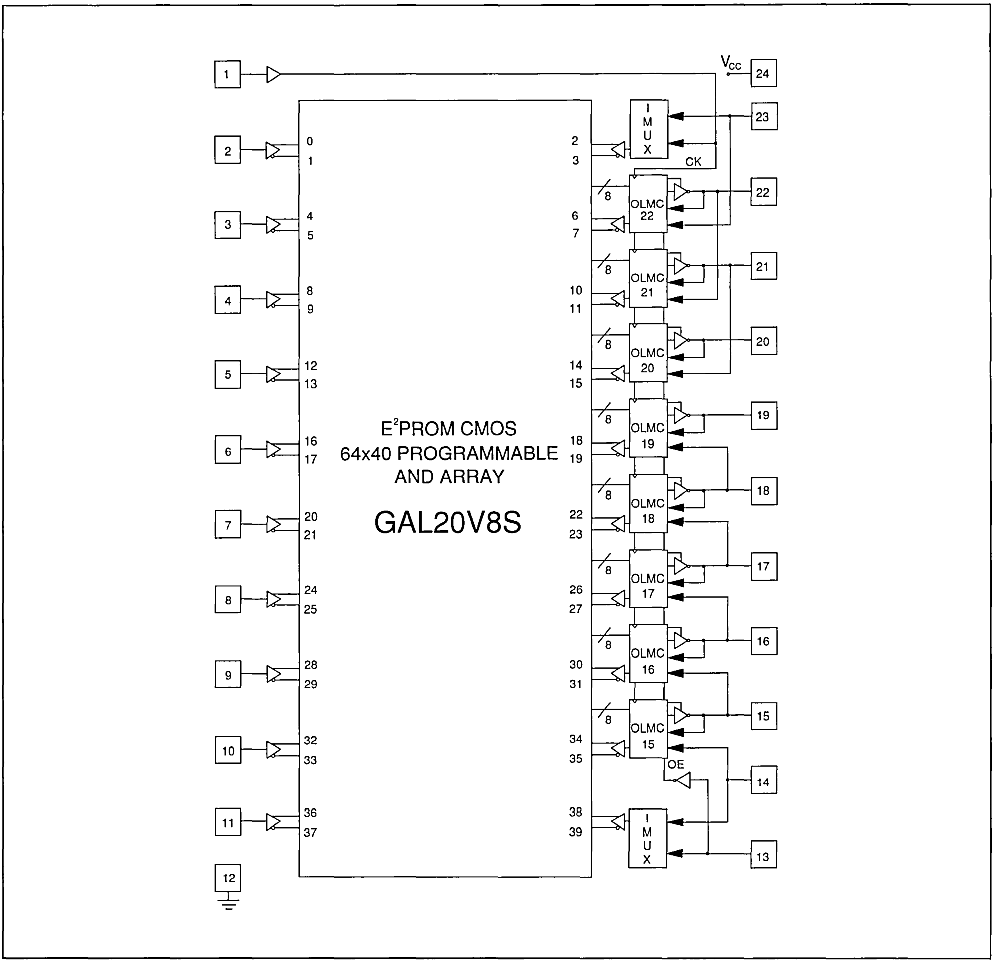
GAL® is a registered trademark of Lattice Semiconductor Corp.; PAL® is a registered trademark of Monolithic Memories Inc.

map/parametric compatible mode, 21 types of 24 pin PAL® devices.

Unique test circuits and reprogrammable cells allow complete AC, DC and functional testing during manufacture.

Therefore, SGS-THOMSON guarantees 100% field programmability and functionality of GAL® devices. SGS-THOMSON also guarantees 100 erase/write cycles and data retention exceeding 20 years.

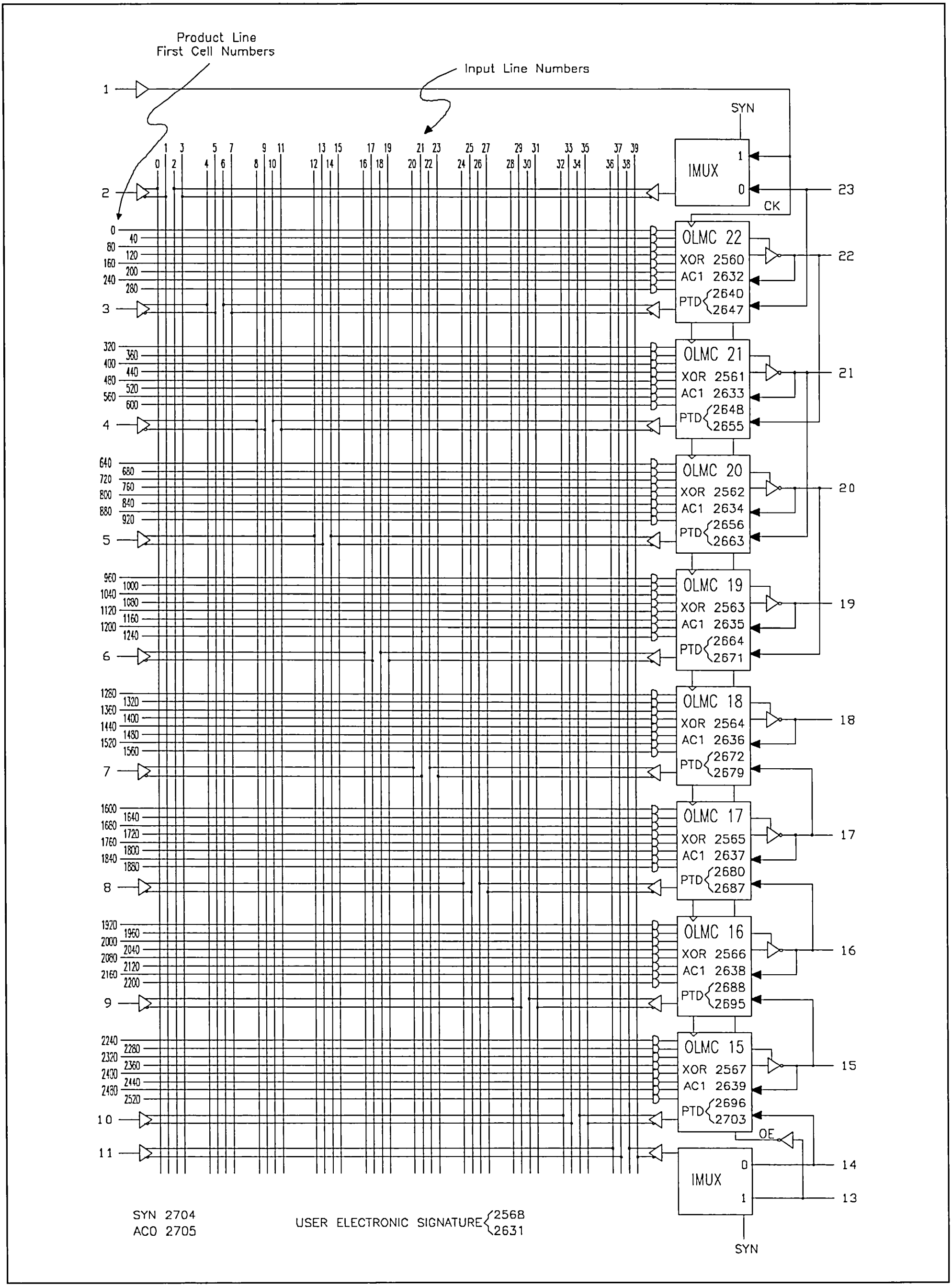
GAL20V8S Block Diagram



GAL20V8S PAL® Architecture Emulation

20L8	20R8	20RP6	20L2	18L4	16L6	14L8
20H8	20RP8	20R4	20H2	18H4	16H6	14H8
20P8	20R6	20RP4	20P2	18P4	16P6	14P8

GAL20V8S Logic Diagram



Absolute Maximum Ratings

Symbol	Parameter	Value	Unit
V_{CC}	Supply Voltage	-0.5 to +7	V
V_I	Input Voltage Applied	-2.5 to V_{CC}+1	V
V_B	Off-State Output (Bidirectional) Voltage Applied	-2.5 to V_{CC}+1	V
T_{STG}	Storage Temperature	-65 to +125	˚C
T_J	Junction Temperature (Operating)	-40 to +125	˚C
T_L	Lead Temperature (Soldering)	260 (for 10s max.)	˚C

Stresses above those listed under the “Absolute Maximum Ratings” may cause permanent damage to the device. These are stress only ratings and functional operation of the device at these or at any other conditions above those indicated in the operational sections of this specification is not implied.

ESD Immunity

Test Method: Human Body Model (HBM)
ESD Tolerance ≥ 2000V
(See MIL-STD 883c).

Test Method: Charge Device Model (CDM)
ESD Tolerance ≥ 500V
Test Instrument: KeyTek ZapMaster

CDM is an additional test only for GAL®s not yet adopted as a company standard test.

Switching Test Conditions

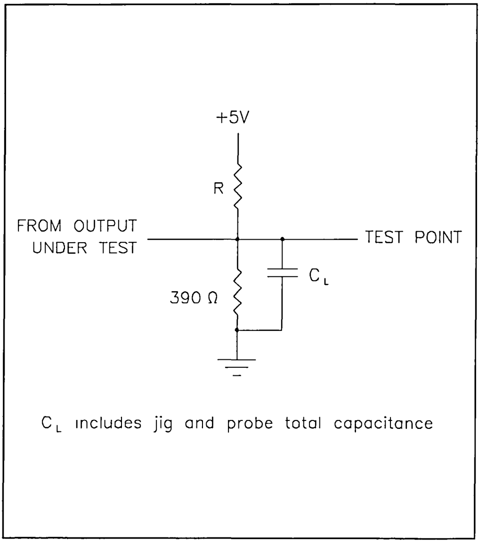
Input Pulse Levels	GND to 3.0V
Input Rise and Fall Times	3ns 10%-90%
Input Timing Reference Levels	1.5V
Output Timing Reference Levels	1.5V
Output Load	See figure

3-state levels are measured 0.5V from steady-state active level.

Test Conditions

#	R [Ω]	C_L [pF]
1	200	50
2	Active High: ∞ Active Low: 200	50
3	Active High: ∞ Active Low: 200	5

Switching Test Circuit



C_L includes jig and probe total capacitance

Capacitance (T_A=25°C, f=1.0MHz, V_{CC}=5V)

Symbol	Parameter	Test Conditions	Maximum*	Units
C_I	Input Capacitance	V_I=2V	8	pF
C_B	Bidirectional Pin Capacitance	V_B=2V	10	pF

* Guarantied but not 100% tested.

DC Operating Conditions

Symbol	Parameter	Commercial Temperature Range		Industrial Temperature Range		Units
		Min.	Max.	Min.	Max.	
V_{CC}	Supply Voltage	4.75	5.25	4.5	5.5	V
T_A	Ambient Temperature	0	70	-40	85	°C
V_{IL}	Input Low Voltage	V_{SS}*-0.5	0.8	V_{SS}*-0.5	0.8	V
V_{IH}	Input High Voltage	2.0	V_{CC}+1	2.0	V_{CC}+1	V
I_{OL}	Low Level Output Current	–	24	–	24	mA
I_{OH}	High Level Output Current	-3.2	–	-3.2	–	mA

* V_{SS} is the voltage applied to the GND pin.

Electrical Characteristics Over Operating Conditions (Commercial Temperature Range)

Symbol	Parameter	Test Conditions	Min.	Max.	Units
I_{IH}, I_{IL}	Input Leakage Current	GND≤V_I≤$V_{CC\ Max}$	–	±10	µA
I_{BH}, I_{BL}	Bidirectional Pin Leakage Current	GND≤V_I≤$V_{CC\ Max}$	–	±10	µA
I_{CC}	Operating Power Supply Current	f=15MHz V_{CC} = $V_{CC\ Max}$ V_{IL}=0.5V V_{IH}=3.0V	–	27	mA
I_{OS}*	Output Short Circuit Current	V_{CC}=5.0V, V_B=0.5V	-150	-30	mA
V_{OL}	Output Low Voltage	–	–	0.5	V
V_{OH}	Output High Voltage	–	2.4	–	V

Electrical Characteristics Over Operating Conditions (Industrial Temperature Range)

Symbol	Parameter	Test Conditions	Min.	Max.	Units
I_{IH}, I_{IL}	Input Leakage Current	GND≤V_I≤$V_{CC\ Max}$	–	±10	µA
I_{BH}, I_{BL}	Bidirectional Pin Leakage Current	GND≤V_I≤$V_{CC\ Max}$	–	±10	µA
I_{CC}	Operating Power Supply Current	f=15MHz V_{CC} = $V_{CC\ Max}$ V_{IL}=0.5V V_{IH}=3.0V	–	36	mA
I_{OS}*	Output Short Circuit Current	V_{CC}=5.0V, V_B=0.5V	-150	-30	mA
V_{OL}	Output Low Voltage	–	–	0.5	V
V_{OH}	Output High Voltage	–	2.4	–	V

* One output at a time for a maximum duration of one second.

Switching Characteristics Over Operating Conditions

Symbol	Parameter	From	To	Max.	Units	Test Cond.*
t_{pd}	Combinational Propagation Delay	Input	Output	20	ns	1
t_{co}	Clock to Output Delay	Clock	Registered Output	15	ns	1
t_{en}	Product Term Output Enable to Output	Input	Output	20	ns	2
t_{enr}	Output Register Enable to Output	$\overline{OE}$	Registered Output	18	ns	2
t_{dis}	Product Term Output Disable to Output	Input	Output	20	ns	3
t_{disr}	Output Register Disable to Output	$\overline{OE}$	Registered Output	18	ns	3

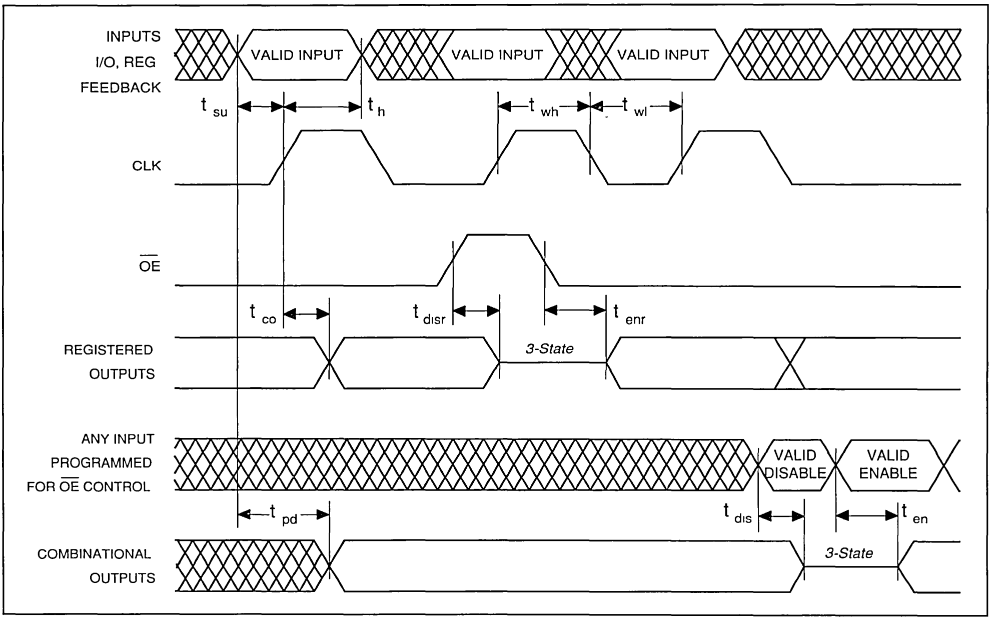
AC Operating Conditions

Symbol	Parameter	Min.	Max.	Units	Test Cond.*
t_{su}	Input or Feedback Setup Time (Before Clock Rise)	–	15	ns	–
t_h	Input or Feedback Hold Time (After Clock Rise)	–	0	ns	–
t_{wh}	Minimum Clock Width High	–	12	ns	–
t_{wl}	Minimum Clock Width Low	–	12	ns	–
f_{clk}◇	Clock Frequency Without Feedback	41.6	–	MHz	1
f_{clkf}✦	Clock Frequency With Feedback	33.3	–	MHz	1

* Refer to “Switching Test Conditions”.

◇ $f_{clk} = \frac{1}{t_{wh} + t_{wl}}$ ✦ $f_{clkf} = \frac{1}{t_{su} + t_{co}}$

Switching Waveforms



FUNCTIONAL DESCRIPTION

GAL20V8S has a programmable AND array whose output terms feed a fixed (non programmable) OR array, as bipolar PAL®. The 2 × 10 input lines enter the AND array as true or complemented form. 64 product terms are available allowing standard Sum of Products Logic implementation. Each product term is obtained by appropriate connections between the input lines and the product term line. The connections can be made by programming the E^{2}PROM memory cell at each intersection of the AND matrix (2560 memory cells). The 64 product terms are divided into eight groups of 8 terms each. One product term for each group can be used to provide Output Enable control for combinational output, the others are connected with an OR gate into the corresponding OLMC (Output Logic Macrocell). The output buffer is in 3-state when the corresponding output enable signal is low.

OUTPUT LOGIC MACROCELL (OLMC)

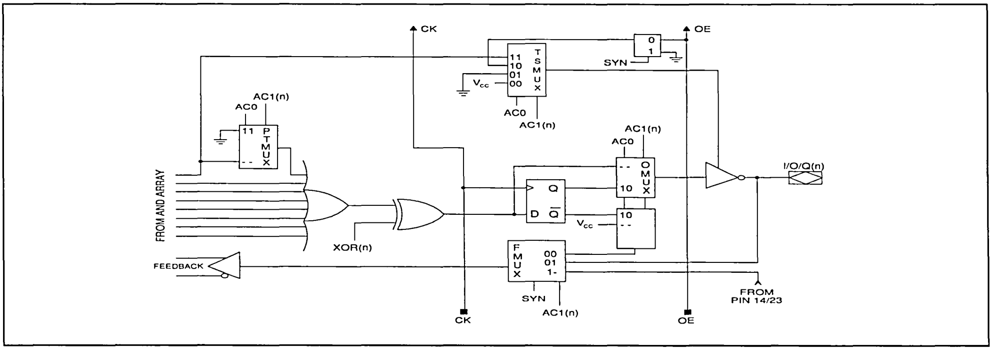
The following discussion pertains to configuring the output logic macrocells. It should be noted that actual implementation is accomplished by development software/hardware and is completely transparent to the user.

The outputs of the AND array are fed into an OLMC, where each output can be individually set to active high or active low, with either combinational (asynchronous) or registered (synchronous) configurations. A common output enable is connected to all registered outputs; product terms can be used to provide individual output enable control for combinational outputs. All outputs have always programmable polarity.

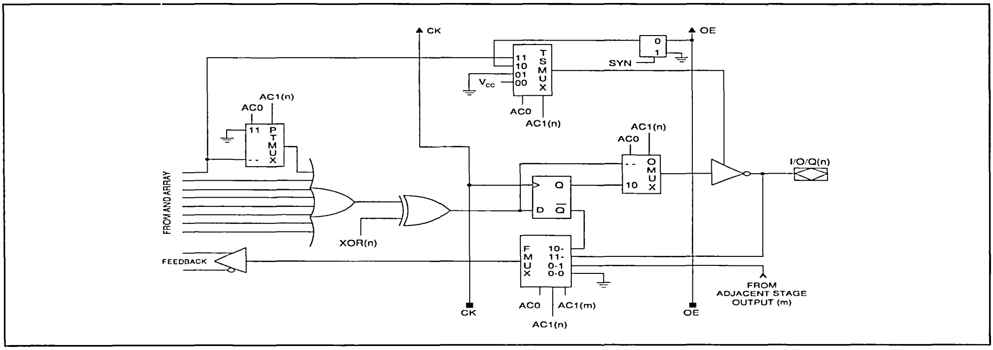
The output logic macrocell provides the designer with maximum output flexibility in matching signal requirements, thus providing more functions than existing 24 pin PAL® devices.

Three different configuration modes of the OLMCs are possible: registered, complex and simple. The output of an OLMC in registered mode can be either registered or combinational. Different modes cannot be mixed: i.e. all OLMCs of a device have to be configured in simple, complex or registered mode.

GAL20V8S Output Logic Macrocell Pin 15 and 22



GAL20V8S Output Logic Macrocell Pin 16 to 21

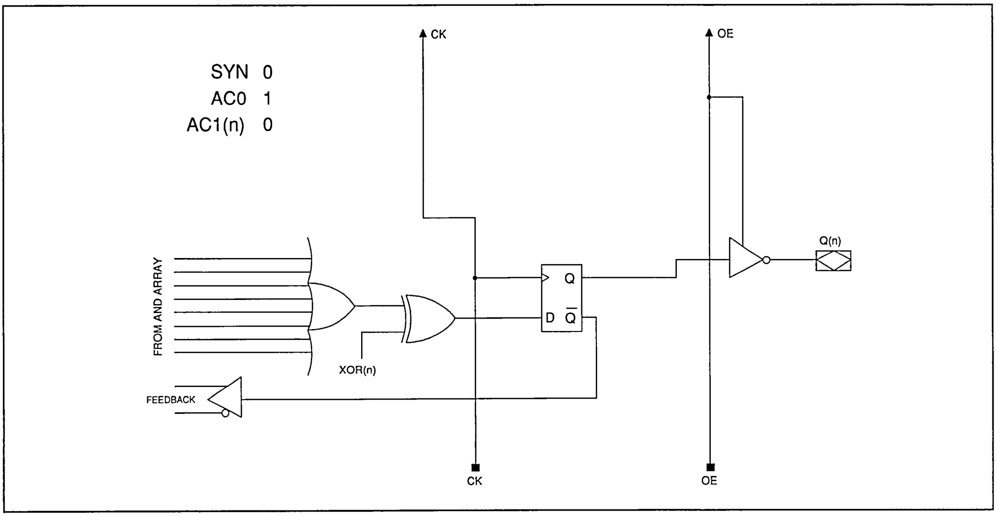


REGISTERED MODE

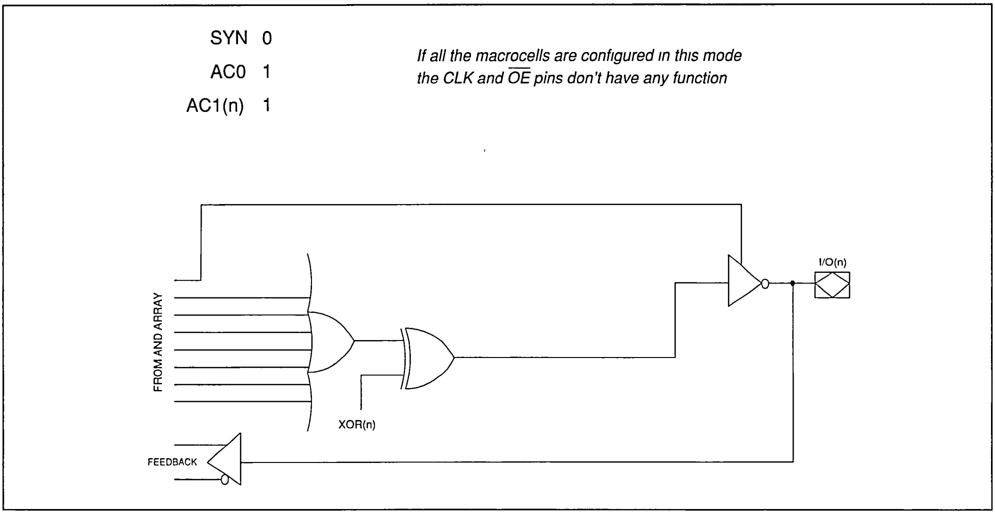
In registered mode macrocells are configured as registered outputs or combinational inputs/outputs. Any macrocell can be configured as registered output or combinational input/output. Up to 8 registered outputs or up to 8 inputs/outputs are possible in this mode.

All registered macrocells share common clock and output enable control. Registered outputs have 8 data product terms per output, while combinational inputs/outputs have only 7 data product terms per output: in the latter case the eighth product term serves as individual output enable control for each macrocell.

Registered Output with Programmable Polarity



Combinational Input/Output with Programmable OE and Polarity



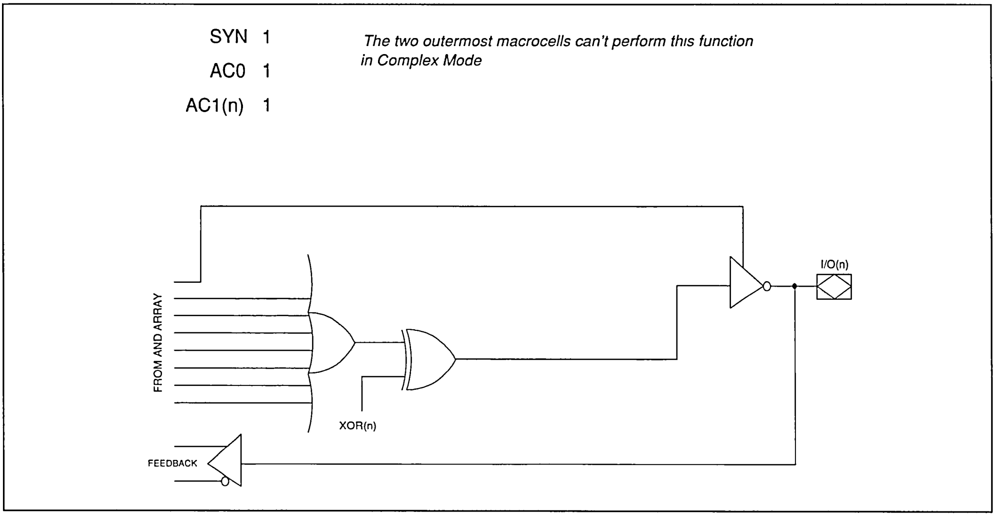
If all the macrocells are configured in this mode the CLK and $\overline{OE}$ pins don't have any function

COMPLEX MODE

In complex mode macrocells are configured as combinational inputs/outputs or outputs only. The two outermost macrocells (15 and 22) do not have input capability: so only up to 6 inputs/outputs are possible in this mode. Applications requiring 8 inputs/outputs must be implemented in registered mode.

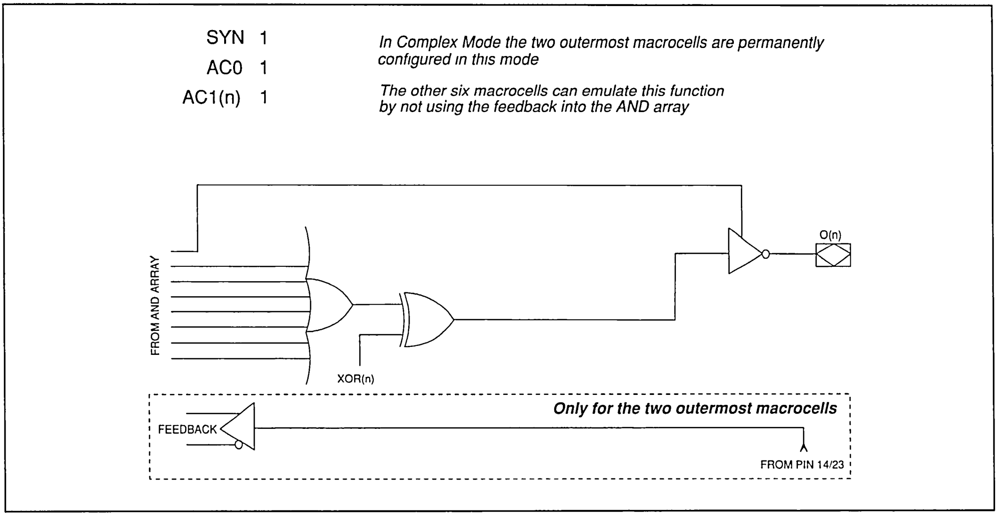
All macrocells have 7 data product terms per output; the eighth product term is used as individual output enable control for each macrocell. The clock and output enable pins (pins 1 and 13 respectively) are always available as inputs.

Combinational Input/Output with Programmable OE and Polarity



The two outermost macrocells can't perform this function in Complex Mode

Combinational Output with Programmable OE and Polarity



In Complex Mode the two outermost macrocells are permanently configured in this mode

The other six macrocells can emulate this function by not using the feedback into the AND array

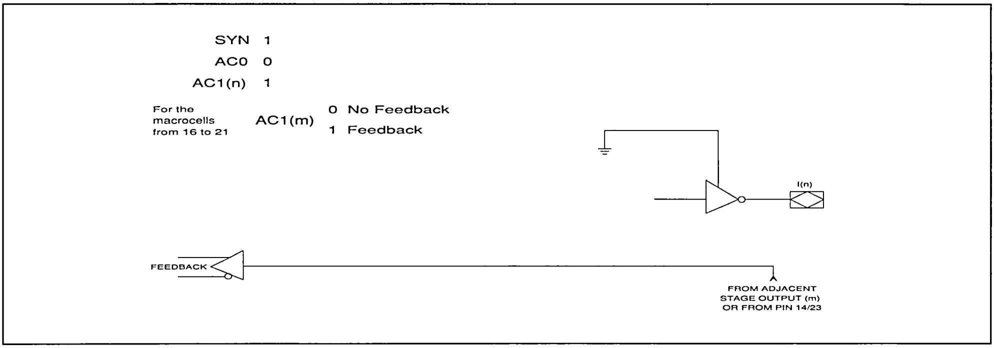
Only for the two outermost macrocells

SIMPLE MODE

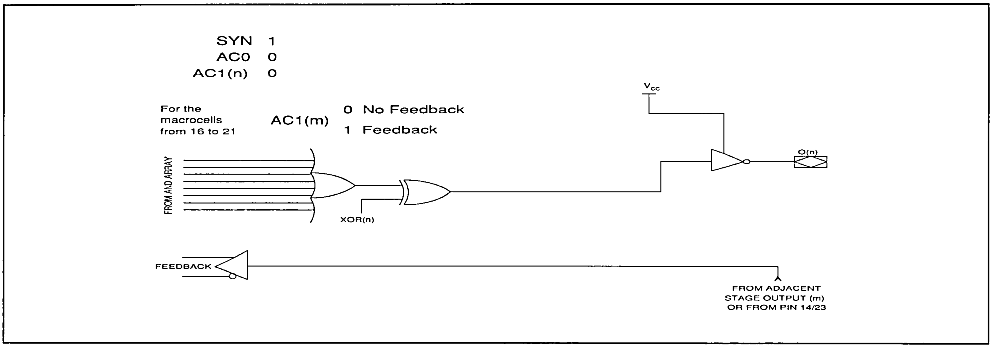
In simple mode macrocells are configured as dedicated inputs or as dedicated, always active, combinational outputs. The two central macrocells (18 and 19) cannot be used in the input configuration.

All macrocells have 8 data product terms per output. The clock and output enable pins (pins 1 and 13 respectively) are always available as inputs.

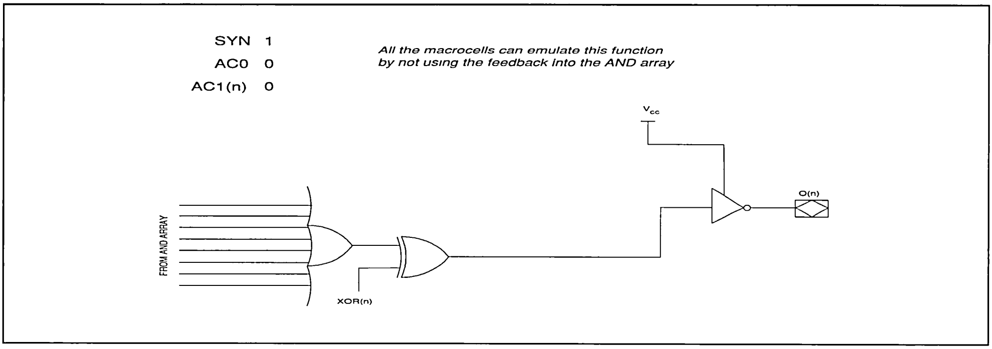
Dedicated Input Mode


SYN 1
AC0 0
AC1(n) 1
For the macrocells from 16 to 21
AC1(m)
0 No Feedback
1 Feedback
I(n)
FEEDBACK
FROM ADJACENT STAGE OUTPUT (m) OR FROM PIN 14/23

Dedicated Combinational Output with Feedback and Programmable Polarity



Dedicated Combinational Output with Programmable Polarity

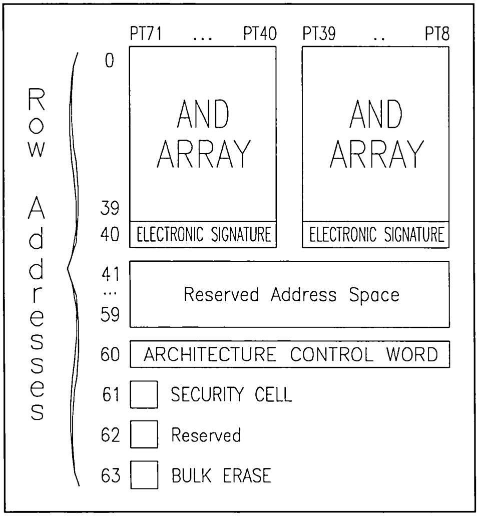

SYN 1
AC0 0
AC1(n) 0
All the macrocells can emulate this function by not using the feedback into the AND array
Vcc
O(n)
FROM AND ARRAY
XOR(n)

ROW ADDRESS MAP DESCRIPTION

There are a total of 44 unique row addresses available to the user when programming the GAL20V8S device. Row addresses 0-39 each contain 64 bits of input term data. This is the AND array where the custom logic pattern is programmed. Row 40 is the Electronic Signature Word. It has 64 bits available for any user defined purpose. Row 41-59 are reserved by the manufacturer and are not available to users.

Row 60 contains the architecture and output polarity information. The 82 bits within this word are programmed to configure the device for a specific application. Row 61 contains a one bit security cell that when programmed prevents further pattern verification of the array. Row 63 is the row that is addressed to perform a bulk erase of the device, resetting it back to a virgin state. Each of these functions is described in the following sections.

GAL20V8S Row Addresses Map Block Diagram



ELECTRONIC SIGNATURE WORD DESCRIPTION

An electronic signature word is provided with every GAL20V8S device. It resides at row address 40 and contains 64 bits of reprogrammable memory that can contain user-defined data. Some uses include user ID codes, revision numbers, or inventory control. This signature data is always available to the user independent of the state of the security cell.

ARCHITECTURE CONTROL WORD

All the various output configurations of the GAL20V8S devices are controlled by programming cells within the 82 bit Architecture Control Word that resides at row 60. The location of specific bits within the Architecture Control Word is shown in the control word diagram in figure below. The function of the SYN, AC0 and AC1(n) bits have been explained in the OUTPUT LOGIC MACROCELL description. The eight polarity bits determine each output's polarity individually. The numbers below the XOR(n) and AC1(n) bits in the architecture control word diagram shows the output device pin number that the polarity bits control.

SECURITY CELL

Row address 61 contains the Security Cell (one bit). The Security Cell is provided on all GAL20V8S devices as a deterrent to unauthorized copying of the array configuration patterns. Once programmed, the circuitry enabling array access is disabled, preventing further verification of the array (rows 0-39). The cell can be erased only in conjunction with the array during a bulk erase cycle, so the original configuration can never be examined once this cell is programmed. Signature data is always available to the user.

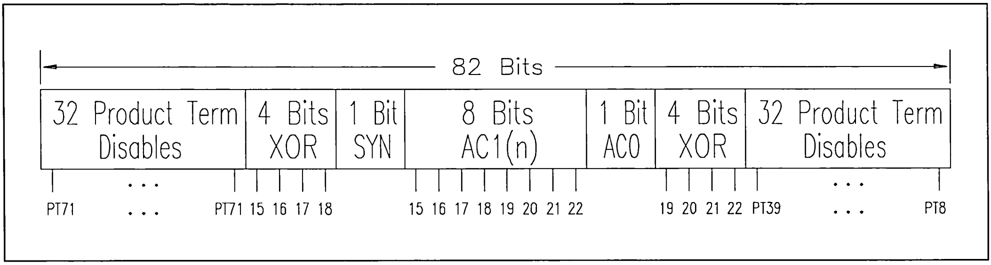
BULK ERASE MODE

By addressing row 63 during a programming cycle, a clear function performs a bulk erase of the array and the Architecture Control Word. In addition, the Electronic Signature Word and the Security Cell are erased. This mode resets a previously configured device back to its virgin state.

OUTPUT REGISTER PRELOAD

When testing state machine designs, all possible states and state transitions must be verified in the

GAL20V8S Architecture Control Word Diagram



design, not just those required in the normal machine operations. This is because in system operation, certain events occur that may throw the logic into an illegal state (power-up, line voltage glitches, brown-outs, etc.). To test a design for proper management of these conditions, a method must be provided to break the feedback paths, and force any desired (e.g. illegal) state into a register. Then the machine can be sequenced and the outputs tested for the correct next state condition. The GAL20V8S device includes circuitry that allows each registered output to be synchronously set either high or low. Thus, any present state condition can be forced for test sequencing. If necessary, approved GAL programmers capable of executing test vectors can perform output register preload automatically.

The figure on the right shows the pin functions necessary to preload the register. This test mode is entered by raising PRLD to V_{IES} (register preload input voltage, typically 15V), which enables the serial data in (S_{DIN}) buffer and the serial data out (S_{DOUT}) buffer. Data is then serially shifted into the registers on each rising edge of the clock, D_{CLK}. Only the macrocells with registered output configurations are loaded. If only 3 outputs have registers, then only 3 bits need be shifted in. The registers are loaded from the bottom up as shown in the figure on the right.

LATCH-UP PROTECTION

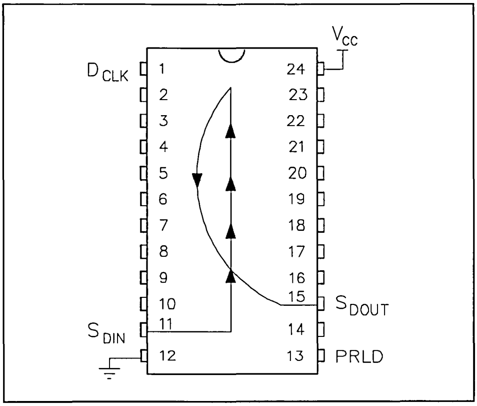
GAL® devices are designed with an on board charge pump to negatively bias the substrate. The negative bias is of sufficient magnitude to prevent input undershoots from causing the circuitry to latch. Additionally, outputs are designed with n-channel pullups instead of the traditional p-channel pullups to eliminate any possibility of SCR induced latching.

POWER-UP RESET

Circuitry within the GAL20V8S provides a reset signal to all registers during power-up. All internal registers will have their Q outputs set low after a specified time (t_{reset}=10µs). As a result, the state on the registered output pins (if they are enabled through $\overline{OE}$) will always be high on power-up, regardless of the programmed polarity of the output pins. This features can greatly simplify state machine design by providing a known state on power-up.

The timing diagram for power-up is shown below. Because of the asynchronous nature of system power-up, the V_{CC} rise must be monotonic to guarantee a valid power-up reset of the GAL20V8S. The registers will reset within a maximum of t_{reset} time: before this time any clock transition from low to high is forbidden to avoid undesired commutations. As in normal system operation, avoid clocking the device until all input and feedback path setup times have been met (i.e. avoid clocking before the $t_{pr}=t_{reset}+t_{su}$ time interval).

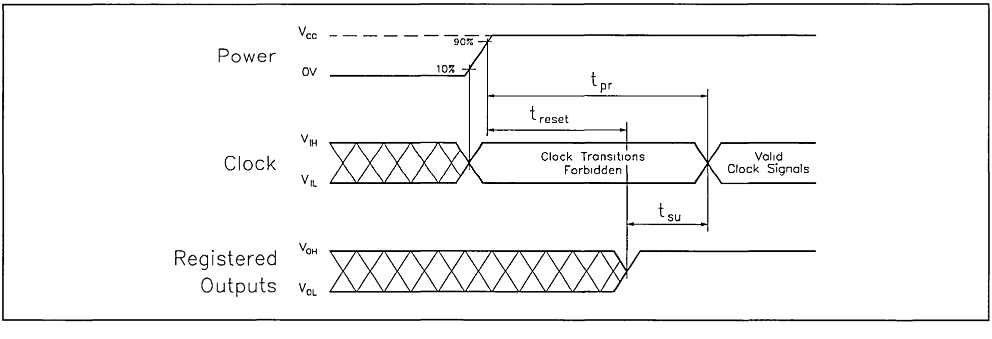
Output Register Preload Pinout



DEVICES PROGRAMMING

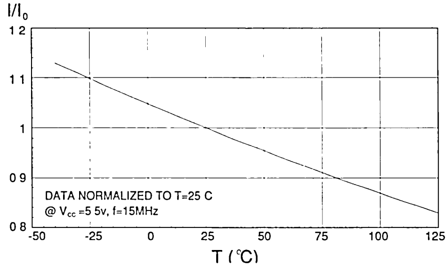
SGS-THOMSON strongly recommends the use of qualified programming hardware. Programming on unapproved equipment will invalidate all guarantees.

Power-Up Reset Timing Diagram

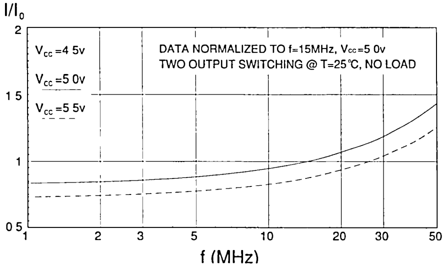


TYPICAL CHARACTERISTICS

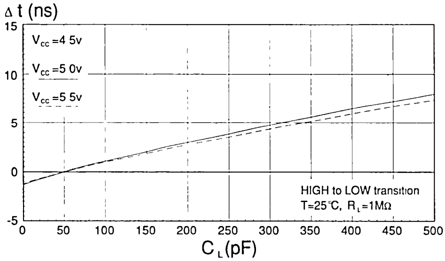
OPERATING CURRENT vs TEMPERATURE



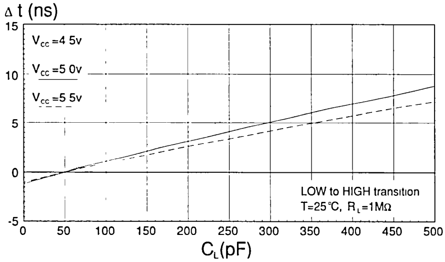
OPERATING CURRENT vs FREQUENCY



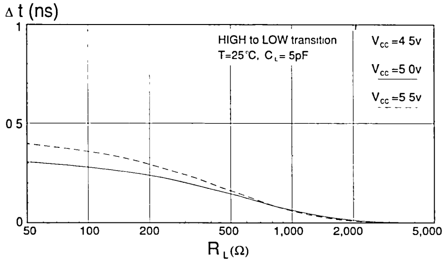
RELATIVE DELAY vs CAPACITIVE LOAD



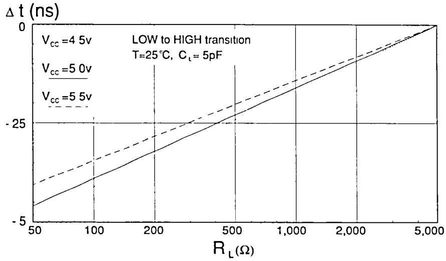
RELATIVE DELAY vs CAPACITIVE LOAD



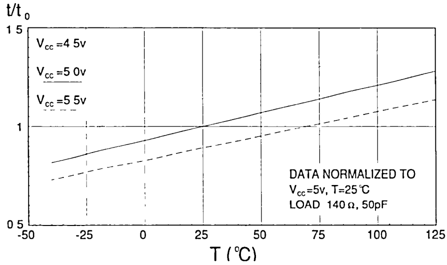
RELATIVE DELAY vs PULL-UP RESISTOR



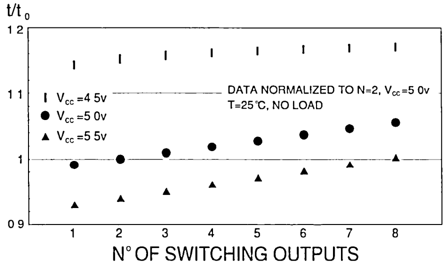
RELATIVE DELAY vs PULL-UP RESISTOR



NORMALIZED DELAY vs TEMPERATURE

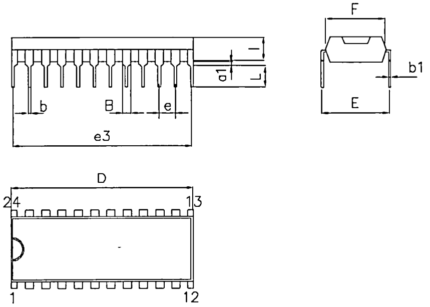


NORMALIZED t_{PD} vs SWITCHING OUTPUTS



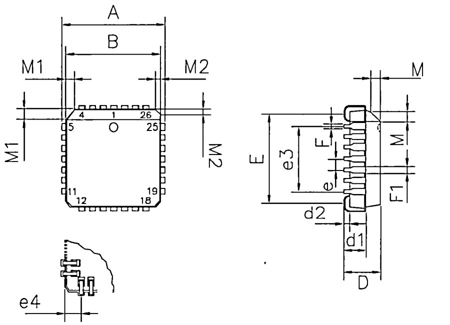
PACKAGE MECHANICAL DATA

PDIP 24 Pins



Dim.	mm			inches		
	Min	Typ	Max	Min	Typ	Max
a1	0.38			0.015		
B	1.27		1.65	0.050		0.065
b		0.46			0.018	
b1		0.25			0.010	
D			31.88			1.255
E		7.62			0.300	
e		2.54			0.100	
e3		27.94			1.100	
F			6.86			0.27
I			4.32			0.170
L		3.30			0.130	

PLCC 28 Pins



Dim.	mm			inches		
	Min	Typ	Max	Min	Typ	Max
A	12.32		12.57	0.485		0.495
B	11.43		11.58	0.450		0.456
D	4.20		4.57	0.165		0.180
d1	2.29		3.04	0.090		0.120
d2	0.51			0.020		
E	9.91		10.92	0.390		0.430
e		1.27			0.050	
e3		7.62			0.300	
e4			1.99			0.078
F		0.46			0.018	
F1		0.71			0.028	
M		1.24			0.049	
M1		1.143			0.045	
M2			0.51			0.020

Seating Plane: 0.101 mm/0.004 inches

Ordering Informations*

SGS-THOMSON GAL®s are available in a variety of package and temperature ranges.
General ordering code is reported below.

GAL20V8S - s w p t j

PLCC Pinout	**J**	Jedec Pinout
Temperature	**1** **3**	0°C to +70°C -40°C to +85°C
Package	**B** **C**	24 Pins PDIP 28 Pins PLCC
Power	**E**	Eighth Power
Speed	**20**	20ns

Example: ordering code for a GAL20V8S, 20ns speed and Eight Power in PLCC (Jedec pinout) is **GAL20V8S-20EC1J**

* Please contact local Product Marketing for latest update on package / temperature range availability.

ST-CUPL DOCUMENTATION

SGS-THOMSON MICROELECTRONICS

LOGICAL DEVICES, INC.

ST-CUPL Package

The *ST-CUPL™ Package* is a complete software development tool for design implementations with GAL®s, based on the popular *CUPL™ 4.0* compiler. It includes a high level description language — that permits truth tables, state machines and boolean equations entry — a logic simulator, a design partitioning tool and a schematic capture (together with a conversion utility).

On a standard PC platform, this software allows you to develop a complete design and to simulate the logic behaviour of the devices.

ST-CUPL™ Language

ST-CUPL™ features command line and menu driven operation, making it a functional and easy to use tool. It offers a choice of four logic minimization algorithms (i.e. Quick, Quine-McCluskey, Presto and Espresso minimization) that can be selected globally or on an equation by equation basis, in order to have the best exploitation of the device.

The *ST-CUPL™* preprocessor provides string substitution, file inclusion and conditional compilation. Together with a user definable syntax and high level macro and function definition, these capabilities allow modular definition of the design and utilization of general purpose units to be customized later for different applications. Indexed variables, set operations and bit field notation allow short and readable coding.

Designs can be entered in several ways:

- *Boolean Equations:* this is the simpler form to describe a design with a logic language; usually, the other forms of expression (truth table and state machine syntax) are first translated by the compiler to this form and then into the final file for the programmer.

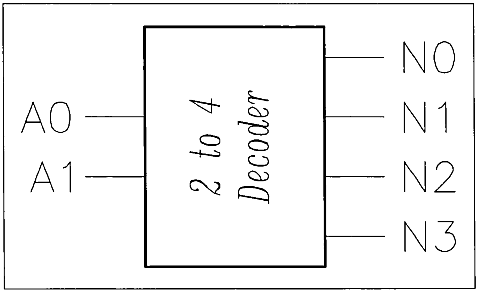
  The general form of a boolean equation is any valid combination of variables made with the logical operators NOT (*"!"*), AND (*"&"*), OR (*"#"*), XOR (*"$"*). The compiler minimizes the equations, generating, at the end of the optimization process, a sum of products form of the original equations.

  This boolean notation can be used for small designs, but becomes unmanageable when the size of the project grows. In order to preserve readability and maintainability, other more powerful notations must be used.

- *Truth Tables:* truth tables are a means to express the values some variables (*output variables*) must assume in function of the corresponding values of other variables (*input variables*).

  For example, a simple two to four decoder may be expressed in the following way:

```
field Input = (A1..0);
field Output = (N3..0);

table Input => Output
{
 'b'00 => 'b'0001;
 'b'01 => 'b'0010;
 'b'10 => 'b'0100;
 'b'11 => 'b'1000;
}
```



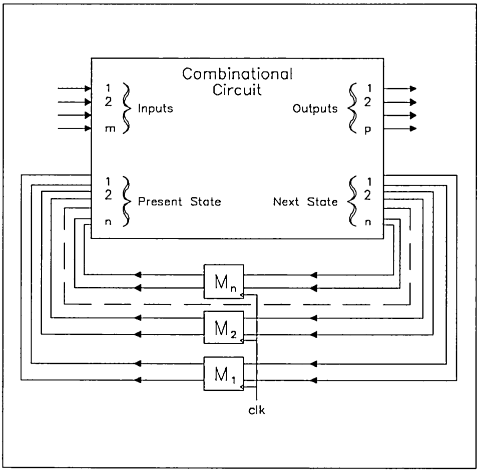
GAL® is a registered trademark of Lattice Semiconductor Corp.
ST-CUPL™ & ST-PLPartition™ are trademarks of SGS-THOMSON Microelectronics & Logical Devices, Inc.
CUPL™ & onCUPL™ are trademarks of Logical Devices, Inc
Schema-Quik™ is a trademark of Omation

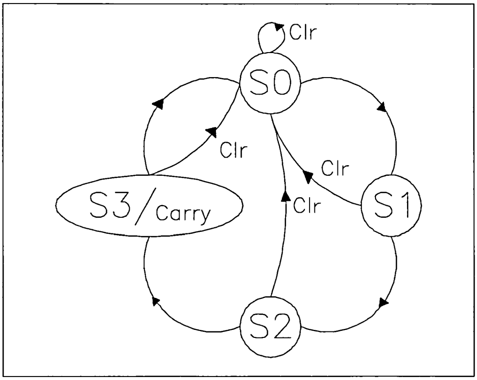
where the *'b'* in front of a sequence of digits indicates that the binary base is used and the *field* notation is used to group variables together.

- *State Machines:* the general structure of a *sequential machine* (or *state machine*) — i.e. a machine whose functioning depends both on the values of the input lines and on the history (the *state* of the machine conserved in its *memory* by means of bistables M_i) — is very similar to the physical structure of a GAL and so can be easily implemented in it. In fact the combinatorial part of the machine can fit into the AND and the OR arrays; its outputs then come into a series of flip-flops or go directly to the external outputs and eventually feed back into the combinatorial parts.

However, it is very useful to describe the state machine diagram in a high level notation that is translated automatically by the software into a series of logic equations.

In *ST-CUPL*™ there are many different types of statement that permit you to specify conditional or unconditional transitions and synchronous or asynchronous outputs, allowing in this way implementation of both Mealy and Moore machines.





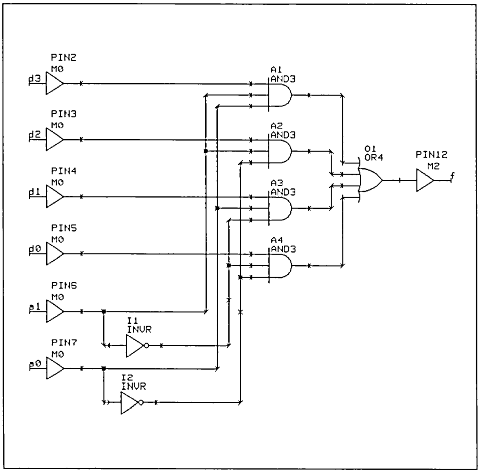
Multiple state machines can be easily handled in one design, with the ability to have them communicate and synchronize state transitions among each other.

An example of implementing a simple 2 bit counter with synchronous *clear* and *carry out* signals is listed below:

```
field Count = (Q1..0);

$define S0 'b'00
$define S1 'b'01
$define S2 'b'10
$define S3 'b'11

sequence Count
{
  present S0
    if Clr next S0;
    default next S1;
  present S1
    if Clr next S0;
    default next S2;
  present S2
    if Clr next S0;
    default next S3;
  present S3
    next S0;
    out Carry;
}
```

Clr is an input signal, *Carry* is connected to an output pin while *Count* defines the flip-flops that will contain the present state of the machine (its *memory*); the *$DEFINE* section simply assigns symbolic names (more readable and manageable) to corresponding values of the state memory elements.

As you can see, this kind of notation is easily readable: it also permits implementation of complicated state machines with a very small possibility of making errors.



- *Schematic Capture:* the schematic representation of an application can be directly translated into a form manageable by a logic compiler using software utilities that usually transform the schematic into a net list form and then into boolean equation syntax.
This file can be compiled and the device programmed.

Included in the *ST-CUPL*™ package there is the *Schema-Quik*™ schematic entry, together with a software utility — named *onCUPL*™ — that translate a netlist generated by the graphics software into the *ST-CUPL*™ syntax. It can translate netlists in EDIF format — so *ST-CUPL*™ can be interfaced with every graphics package that can handle such a format.

Simulation Tool

The simulator provides table oriented simulation to help the designer verify the logic design and generate the test vectors. Extra features are the handling of *don't care* and *high impedance* values, and the optional use of a *"*"* in the output field to indicate a generated rather than a supplied output value.

Simulation results can be displayed and stored on file as tables or waveforms for logic function check and design documentation purposes. The supplied test values can be added automatically to the JEDEC file as test vectors.

The simulation input file is kept separate from the logic description file. This lets you run successive simulations without having to re-compile each time.

ST-PLPartition™ (Interactive PLD Partitioning Software)

ST-PLPartition™ is a design partitioning software that takes a design created and compiled with *ST-CUPL*™ and fits it into one or more GALs.

ST-PLPartition™ reads the *ST-CUPL*™ generated document, allows you to select the desired GAL devices and then provides a list of design solutions.

PLD designs can be optimized with respect to device cost, power consumption and least number of devices.

Major features of the *ST-PLPartition*™ software tool are:

- *User Selectable Algorithms:* best fit algorithm or first fit algorithm are available to provide the results best suited to your particular design. Best fit determines the best devices for use from the list of available GAL devices on hand. First fit selects from the list of available devices the first fit device that will implement the design.
- *History-Based Partition Option:* allows you to modify your design without disturbing previous pin assignment. PLDs can be upgraded without re-laying out your PC board.
- *User-Specified Device Utilization:* designers can specify the percentage of usage of product terms per output to leave space for future ex-

pansion of product terms without requiring a different device.

- *Specify Maximum Number of Devices:* allows more control over final design.
- *Sorted Solution List:* resulting solutions can be sorted according to several criteria: cost, power consumption or number of devices.
- *Chip Connection Diagram:* the resulting chip connection can be displayed on any monitor with EGA or VGA capability.
- *Split Product Terms:* an option is provided to split product terms. Resulting split terms always reside in the same device. This feature should only be used when propagation delays are not critical.

To work with *ST-PLPartition*™ you have only to supply it with a list of available devices, select the optimization criteria and tell the software to find all possible solutions or a specific number of solutions. *ST-PLPartition*™ generates a list of possible solutions that define the types and quantities of the devices needed.

Best solutions are generated first so, if you choose to see only a few designs, the best solutions are always included. A design solution can be selected from those offered and then *ST-PLPartition*™ provides full information (including pin numbers) for device implementation on device programmers such as *Logical Devices' ALLPRO Universal Programmer* or *SGS-THOMSON GAL Starter Kit* programmer.

The input to *ST-PLPartition*™ is a *ST-CUPL*™ documentation file, and the resulting output is composed by *ST-CUPL*™ source files and a detailed report.

ST-CUPL™ package is distributed by all SGS-THOMSON and Logical Devices sales offices, authorized Distributors and Representatives.

Included in the package there is a rebate coupon valid for buying the complete library version of *CUPL*™ as well as *Logical Devices' ALLPRO Universal Programmers.*

Ordering Information

Ordering code for the *ST-CUPL*™ package is: ST-CUPL.

For more information, please contact:

PLD Marketing Group
SGS-THOMSON Microelectronics
via C. Olivetti, 2
20041 Agrate Brianza (MI)
Italy
FAX +39 39 603 5360
Tel. +39 39 603 5225/5438

Logical Devices, Inc.
1201 NW 65th Place.
Ft. Lauderdale, FL 33309
FAX +1 305 974 8531
Tel. +1 305 974 0967

LOGIC CONCEPT

LOGIC DESIGN FUNDAMENTALS

The digital logic design process is based on Boolean algebra. Here we deal only with the fundamentals of Boolean algebra necessary to implement basic logic functions in a programmable logic device.

Boolean Algebra – Basic Functions

The Boolean algebra is based on only two possible values: “yes” or “no” conditions, that can also be expressed as “true” or “false”, “high” or “low”, or in digital electronic terms as “1” or “0”.

All the operations in Boolean algebra are expressed in terms of these two states through the three basic functions: AND, OR and NOT as summarized in Figure 1.

As in normal algebra, the order of precedence in evaluating a formula follows the priority of the operators. Here NOT is first, then comes AND, then OR. To override such order of precedence, parentheses may be used: the operations within parentheses are performed first. For example:

$$C = \bar{A} + \bar{B} \qquad [1]$$

$$D = \overline{(A + B)} \qquad [2]$$

In case [1], values of A and B are inverted before evaluating the OR. In the case [2] the OR operator has to be evaluated before the inverting operator because of the parenthesis.

With the addition of a minimum of simple properties and postulates, the structure of the Boolean algebra is created.

Commutative property:

$$X \cdot Y = Y \cdot X$$
$$X + Y = Y + X$$

Associative property:

$$(X + Y) + Z = X + (Y + Z)$$
$$(X \cdot Y) \cdot Z = X \cdot (Y \cdot Z)$$

Additive and multiplicative identity elements:

$$X + 0 = X$$
$$X \cdot 1 = X$$

Distributive property:

$$X + (Y \cdot Z) = (X + Y) \cdot (X + Z)$$
$$X \cdot (Y + Z) = (X \cdot Y) + (X \cdot Z)$$

Complementation:

$$X + \bar{X} = 1$$
$$X \cdot \bar{X} = 0$$

Hereunder the most fundamental and useful theorems that can be deduced from the above.

Idempotence:

$$X \cdot X = X$$
$$X + X = X$$

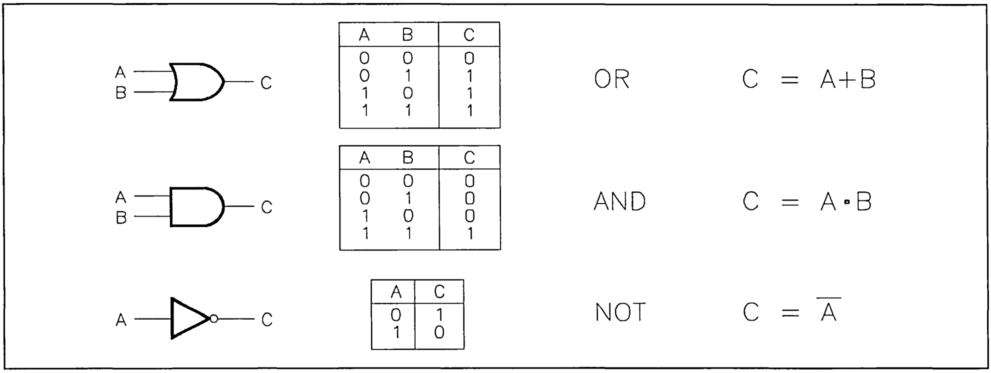
Special properties of 0 and 1:

$$X \cdot 0 = 0 \qquad \bar{0} = 1$$
$$X + 1 = 1 \qquad \bar{1} = 0$$
$$0 \cdot 0 = 0 \qquad 1 + 1 = 1$$
$$0 + 0 = 0 \qquad 1 \cdot 1 = 1$$

Absorption:

$$X \cdot (X + Y) = X$$
$$X + (X \cdot Y) = X$$

Figure 1. Boolean Operators



A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

OR $C = A + B$

A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

AND $C = A \bullet B$

A	C
0	1
1	0

NOT $C = \bar{A}$

Alternative notations for the AND operator — to the dot notation — are “&” or “∧” (e.g. A & B or A ∧ B).
Alternative notations for the OR operator — to the plus notation — are “#”, “|” or “∨” (e.g. A # B or A | B or A ∨ B).
Alternative notations for the NOT operator — to the bar notation — are “!” or “/” (e.g. !A or /A).

DeMorgan's Law:

$$\overline{(X + Y + Z + \ldots)} = \bar{X} \cdot \bar{Y} \cdot \bar{Z} \cdot \ldots$$

$$\overline{(X \cdot Y \cdot Z \cdot \ldots)} = \bar{X} + \bar{Y} + \bar{Z} + \ldots$$

Also very important in Boolean algebra is the Duality principle that states that if the following replacements are made, a logically equivalent expression can be generated:

replace every TRUE with a FALSE
replace every FALSE with a TRUE
replace every AND with an OR
replace every OR with an AND

Boolean algebra (or switching algebra) became popular and its use widespread because it perfectly fits the abstract description of digital electronics, long after the work of the late Mr. Boole had been forgotten.

Karnaugh Maps

Once the function to implement has been defined, it is very important to optimize it using the theorems already illustrated, or using the visual tool known as Karnaugh maps. This maps aid in the reduction of logic functions to one of two special formats that are easily transferred into PLD logic maps: these formats are the Sum Of Products (SOP) and the Product Of Sums (POS).

The goal of the development software designer is to transform the logic definition into an acceptable format. The POS format can be used to describe any combinatorial logic function. This two-level format consists of logical OR terms that are ANDed together. Thus

$$Y = A \cdot (C + D) + B \cdot C + B \cdot D \qquad [1]$$

can be simplified to:

$$Y = A \cdot (C + D) + B \cdot (C + D) \qquad [2]$$

and then

$$Y = (A + B) \cdot (C + D) \qquad [3]$$

which is an AND of sum terms.
The most common representation is the dual of the Product Of Sums format and is known as the Sum Of Products. The basic PLD array interconnects are of this form. The Sum Of Products (SOP) consists of several AND terms ORed (summed) together. Equation [1] can also be simplified to a SOP from:

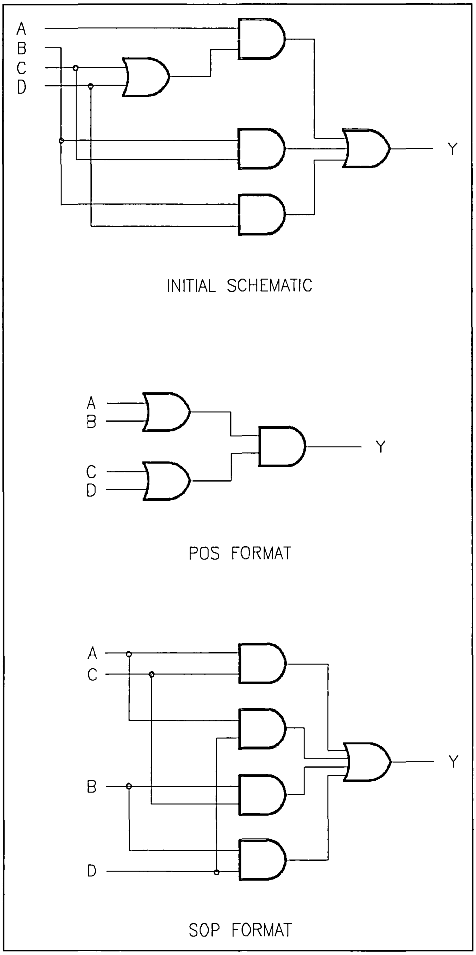
$$Y = A \cdot (C + D) + B \cdot C + B \cdot D \qquad [4]$$

and then

$$Y = A \cdot C + A \cdot D + B \cdot C + B \cdot D \qquad [5]$$

The above transformations are shown in Figure 2.

Figure 2. POS and SOP Formats



INITIAL SCHEMATIC

POS FORMAT

SOP FORMAT

DeMorgan's Law

A closer examination of the above two implementations of the same function in POS (equation [3]) and SOP (equation [5]) formats shows that the number of terms feeding into the final gate varies with the implementation. The SOP format required only 4 terms. This observation is critical, since the total number of terms in any PLD is limited. DeMorgan's law (defined earlier in this section) is a simple rule that can quickly convert SOP to POS or viceversa, without altering the final logic function. This can allow the number of terms to be reduced by as much as 50%, to overcome device limitations. Equation [3] can be converted to SOP as follows:

$$Y = (A + B)\cdot(C + D) \qquad [3]$$

using duality:

- the first step is to change every TRUE into a FALSE:

$$\overline{Y} = (\overline{A} + \overline{B})\cdot(\overline{C} + \overline{D}) \qquad [6]$$

- the second step is to change every AND into an OR and every OR into an AND:

$$\overline{Y} = (\overline{A}\cdot\overline{B}) + (\overline{C}\cdot\overline{D}) \qquad [7]$$

Equation [5] is the SOP form. Note, however, that the output function is inverted (or "active low"). A subsequent inversion function will be required to produce the original output function.

Reduction of Equations

Generally a complex logic function must be represented in a specific and reduced format to be implemented into a PLD. The various methods touched on above — Karnaugh maps, DeMorgan's law — are used to manipulate the equations in conjunction with the basic postulates and theorems.

Present generation software handles all equation minimisations and will actually allow the use of higher level notations, such as state description, truth tables and macro functions.

PLD REPRESENTATION

Let us now look at the logic conventions used to describe PLD devices. A typical PLD input buffer is shown in Figure 3. Its two outputs are the true and the complement of the input.

Figure 3. PLD Input Buffer

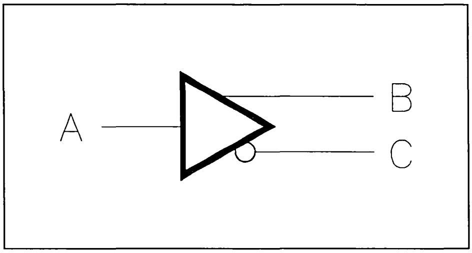
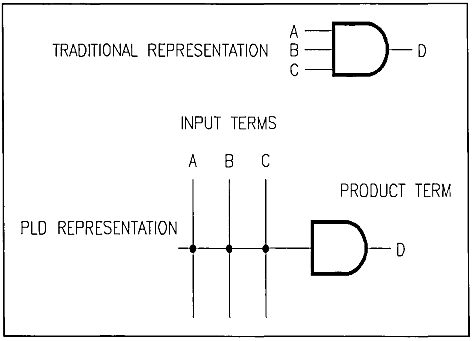


Figure 4 illustrates the convention used to reduce the complexity of a logic diagram without any loss of clarity. The traditional representation of an AND shows three inputs: A, B and C.

Figure 4. AND Gate Representations



The PLD representation has the same three inputs. This shorthand reflects the three distinct input terms of the prior drawing. The structure of a multiple-input AND gate is known as a Product Term.

Referring to Figure 5, we see that the solid-dot connection in the previous figure represents a permanent connection. A programmable interconnection would appear as an X over the intersection, as shown. The X implies that the connection is intact, whereas the absence of an X implies no connection.

Figure 5. PLD Connections

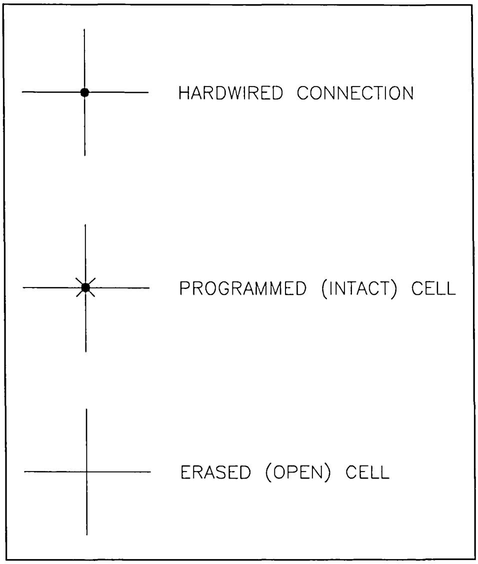
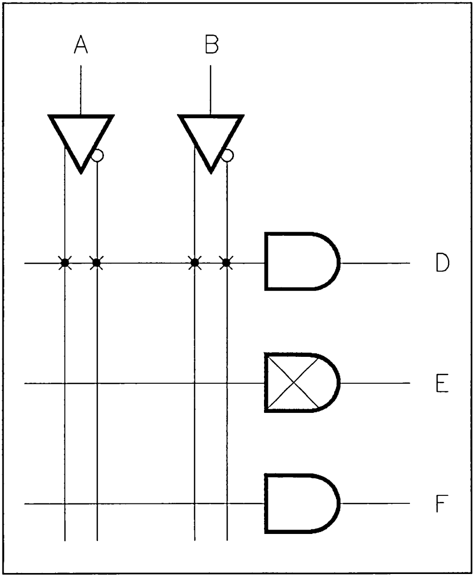


Figure 6 details the default conditions for AND gates.

Figure 6. AND Gate Default Conditions



From the diagram, you can see that the AND gate for output D is connected to all the input terms. The equation for D is

$$D = A \cdot \overline{A} \cdot B \cdot \overline{B}$$

which can be simplified using Boolean algebra

$$D = (A \cdot \overline{A}) \cdot (B \cdot \overline{B})$$
$$D = (0) \cdot (0)$$
$$D = 0$$

The connection of both the true and complement of a given input buffer to a single product term results in that product term always being a logic 0.

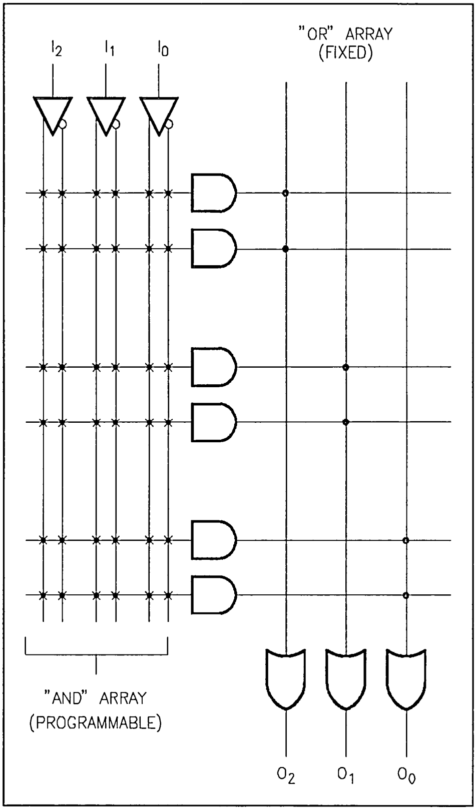
A shorthand notation for leaving all of the input buffers connected is illustrated on output E. Since logic diagram maps are usually supplied without any of the connections shows as intact, it is much simpler for the designer to connect a whole product term (the device default) by simply drawing "Xs" within the AND gates. Again, this product term will always output a logic 0.

Output F, in contrast, does not have any input terms connected to its product term. This product term will always float to a logic 1, resulting in a 1 on the output. In the following sections, where various PLD architectures will be examined in detail, we will see why this design practice is not recommended.

PAL® DEVICES

The PAL® structure in Figure 7 shows that a PAL® consists of a programmable AND array that feeds a fixed OR array. This approach offers the highest performance and the most efficient architecture for most logic functions.

Figure 7. Basic PAL® Architecture



The quantity of product terms per output is fixed by the hardwired OR array. The typical logic function requires some 3 to 4 product terms, well under the 7 to 8 available on current generation devices.

PAL® device architectures (the number of inputs, outputs and product terms) have been fixed by the manufacturers, based on the cumulated experience on what the designers most often need. However, the dozens of device types introduced over the last 10 years essentially offer various permutations of three basic output structures.

The first is shown in Figure 8, which illustrates an input and an output with six product terms. The output is always enabled and active low (note the inversion at the output of the OR gate). The true and complement of each input are available to the AND array.

The second output structure is actually an I/O pin, shown in Figure 9. The output logic is an active-low function of seven product terms. The signal from the I/O pin, also feeds back into the AND array.

Note that the output buffer is controlled by its own product term, allowing dynamic I/O control. This dynamic control can be used either to determine the ratio of device inputs to outputs, or to disable the outputs when connected in a bus environment.

Third is a sequential (or registered) output, shown in Figure 10. The logic OR of eight products terms is available to the designer. Here, the register state (both true and complement) feeds back into the array, as well as into an output buffer with a bank-controlled output enable feature. The clock is common as well, minimizing switching skew between buffers, and the register is a high speed D type. The feedback of the register data into the AND array allows the current state data (in the registers) to be part of the next state function. This is necessary for most sequential functions, such as counting and shifting operations.

Figure 8. Dedicated Output Structure

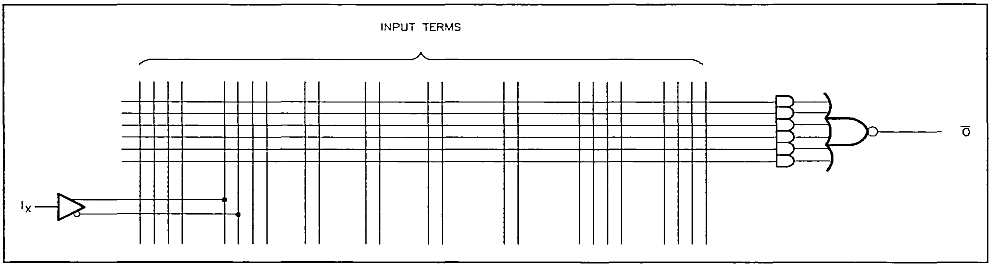


Figure 9. Asynchronous I/O Structure

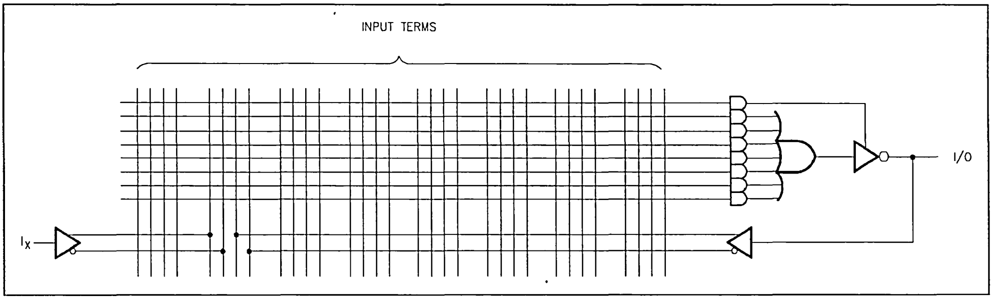
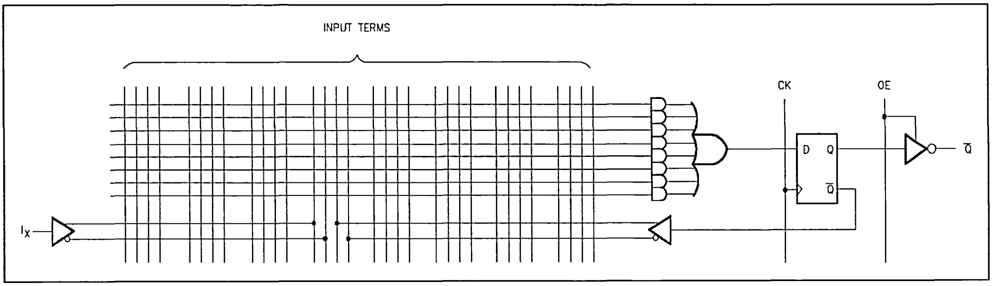


Figure 10. Sequential (Registered) Output Structure



GAL® DEVICES

GAL® devices have the same programmable AND array driving a fixed OR array. The difference is in the architecture and flexibility of the output functions.

The GAL® device integrates an Output Logic Macrocell (OLMC) on each of its output pins. The GAL16V8 and its eight OLMCs are shown in Figure 11.

The programmable polarity feature of each of the output macrocells deserves a special investigation. Located in the heart of the OLMC, the programmable polarity function is implemented by the exclusive-OR (XOR) gate that follows the OR gate from the array. Recalling the truth table of an XOR gate, it can be shown that the data can be inverted (control=1) or not inverted (control=0), depending on the state of the second input, as shown in the table below.

Programmable Polarity Function

Signal	Control	XOR Output
0	0	0
1	0	1
0	1	1
1	1	0

Programmable output polarity is used extensively in DeMorgan's Law to reduce the number of product terms required to implement a function. As a result, the GAL device can generally implement functions that appear to require more than 8 product terms per output. For example:

$$K = A + B + C + D + E + F + G + H + I$$

is a function of 9 product terms, each representing one input variable. This equation can be reduced to one product term if DeMorgan's Law is applied:

$$\overline{K} = \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} \cdot \overline{E} \cdot \overline{F} \cdot \overline{G} \cdot \overline{H} \cdot \overline{I}$$

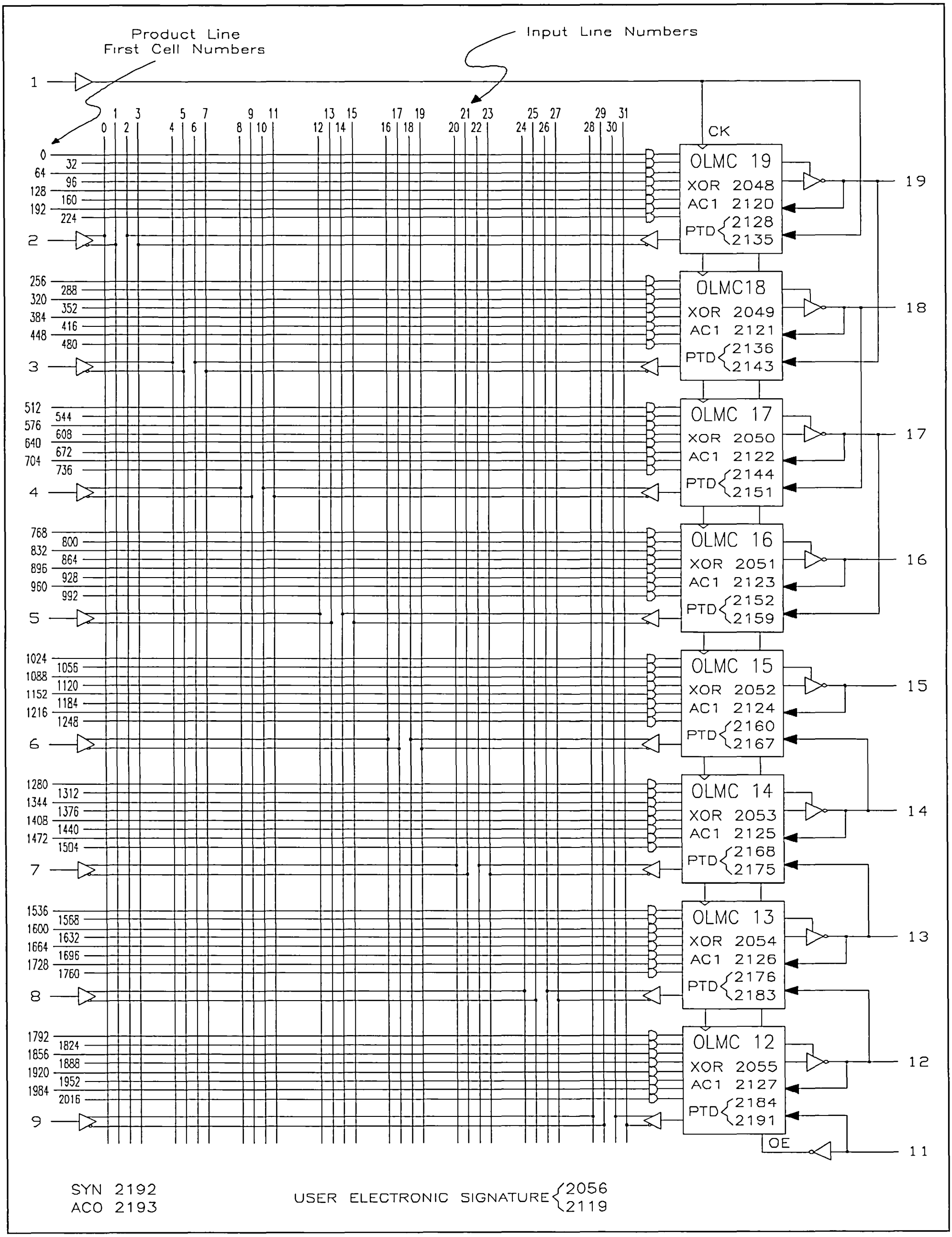
$\overline{K}$ is now a function of only 1 product term. To obtain K again, we only need to invert the function using the polarity feature of the GAL® OLMC.

The OLMCs are configurable by the designer to perform the various functions. For example, the designer merely specifies two active-low registers, one active-high register and the device is configured instantly.

Since each of the OLMCs contains the same logic, it is also possible to "tweak" an existing design for the convenience of the manufacturing department. One example might be moving a function to an adjacent pin to optimize board layout.

This could eliminate an interconnect level on a multi-level board, by swapping two functions and eliminating the need to cross traces. The GAL architecture is not fixed until the user specifies the requirements.

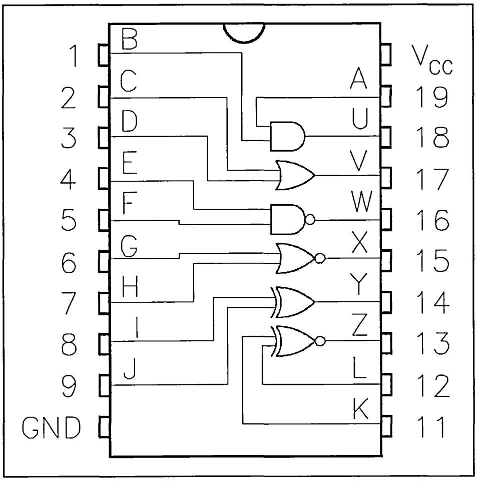
Figure 11. GAL16V8 Logic Diagram



GAL® CONFIGURATION EXAMPLE

This example shows how the various GAL16V8 architecture bits are derived to implement a device whit 12 inputs and 6 programmable combinatorial outputs (Figure 12). In practice, the users need not be concerned with these architecture bits, since they are automatically handled by qualified programming equipment.

Figure 12. Basic Gates Example



Deriving the Architecture Control Word

The architecture control word bits for this example are determined as follows:

AC1(12)=1	Input pin [1]
AC1(13)=0	Output pin [2]
AC1(14)=0	Output pin [2]
AC1(15)=0	Output pin [2]
AC1(16)=0	Output pin [2]
AC1(17)=0	Output pin [2]
AC1(18)=0	Output pin [2]
AC1(19)=1	Input pin [1]
AC0 = 0	No tri-state control on outputs
SYN = 1	All outputs are combinatorial (no registers; pin 1 and 11 available as inputs in this configuration).
XOR(12)	Input pin, XOR not used
XOR(13)=0	Invert
XOR(14)=1	Non-invert
XOR(15)=0	Invert
XOR(16)=0	Invert
XOR(17)=1	Non-invert
XOR(18)=1	Non-invert
XOR(19)	Input pin, XOR not used

[1] Output buffer disabled, tri-state.

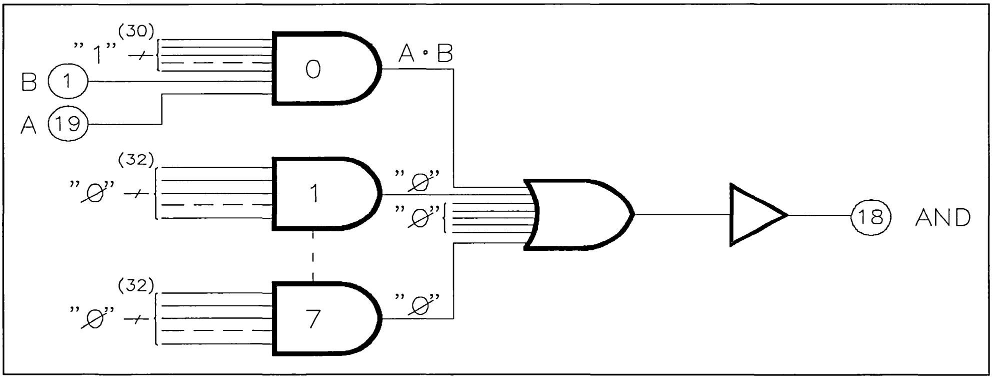
[2] Output buffer enabled.

Programming Basic Logic Gates

The examples of implementation of the basic gates AND, OR and XNOR, taken from the previous example, are shown here in greater detail. Programming of the other basic logic gates NAND, NOR and XOR can be obtained similarly.

In the following figures, the numbers within parentheses indicate how many input terms are fixed at the specified logic level.

Figure 13. The AND Basic Gate



The address of a memory cell may be obtained by adding the corresponding “input line number” to the “product line first cell number” found in the logic diagram.

To implement the AND, the cells 258 and 262 must be programmed; Figure 13 shows the equivalent circuit for the AND function.

Every programmed cell connects an input (or its negation) to a product term; for example the cell 258 connects the input pin 19 (A) to the first product term of the output pin 18 (AND).

Two product terms are involved in the implementation of the OR. The cells 512 and 548 must be programmed. The equivalent circuit for the OR function is shown in Figure 14.

Figure 14. The OR Basic Gate

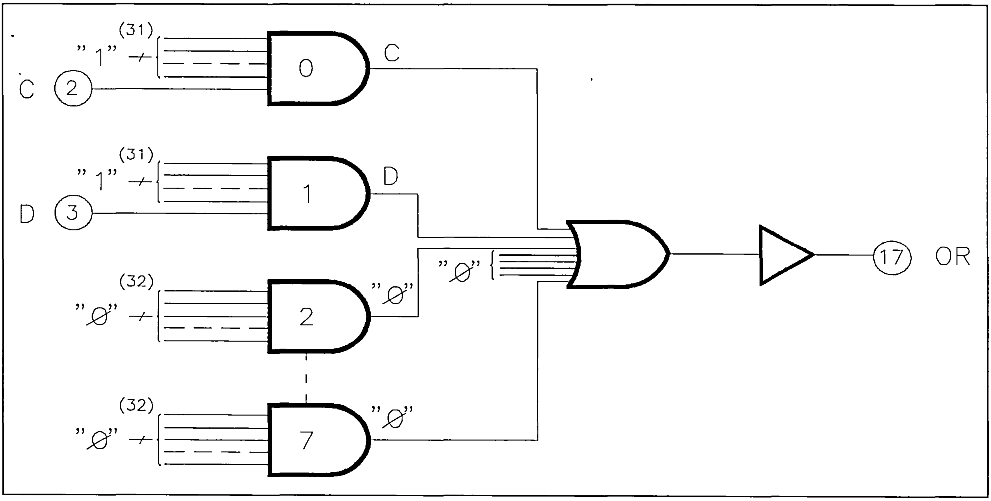
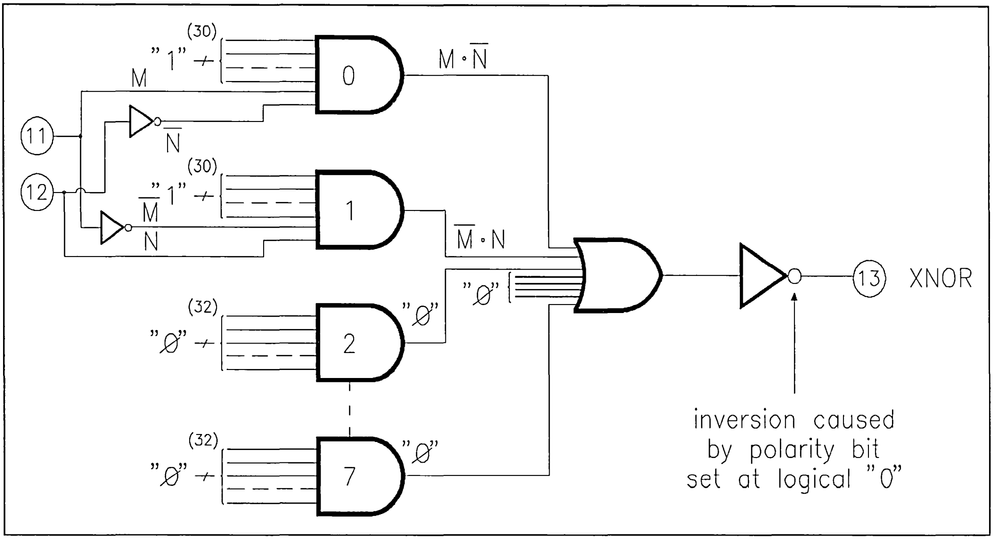


Figure 15. The XNOR Basic Gate



Also in the case of the XNOR function (Figure 15) two product terms are involved.

The Sum Of Product format of such function is then:

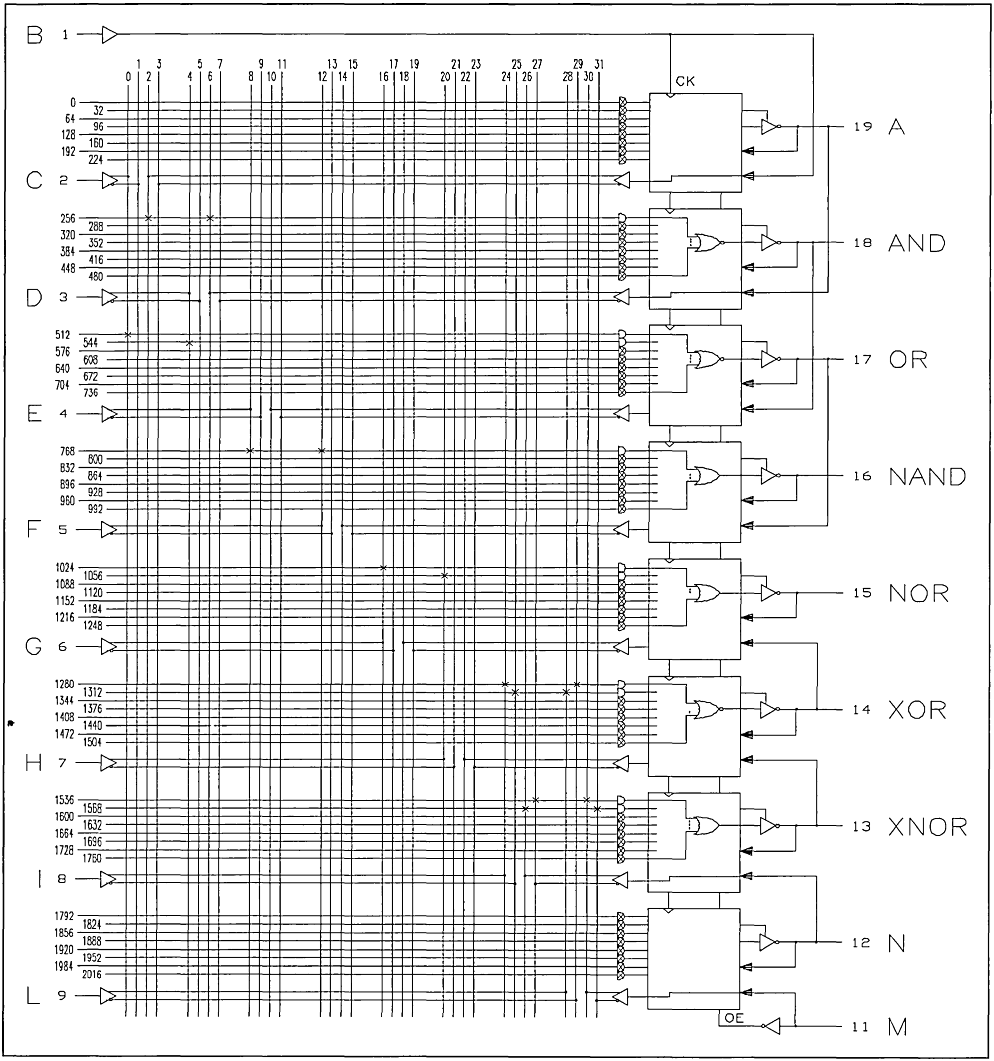
$$\overline{M \oplus N} = M \cdot \overline{N} + \overline{M} \cdot N$$

so four cells have to be programmed, specifically the ones at the locations 1563, 1566, 1594 and 1599.

The Resulting Programming Pattern

When a GAL16V8 has been programmed for the application shown in Figure 12, the logic diagram becomes that of Figure 16.

Figure 16. Basic Gates Logic Diagram



Basic Gates with ST-CUPL™

The previous example can be very quickly implemented using the ST-CUPL™ development software, as shown here below.

Basic Gates — Source File

```
Name          Basic_Gates;
Partno        B_GATES;
Date          None;
Revision      00;
Designer      None;
Company       SGS-THOMSON;
Assembly      None;
Location      None;
Device        G16V8;

/* Inputs */

Pin 19 = A;         /* AND input */
Pin  1 = B;         /* AND input */
Pin  2 = C;         /* OR input */
Pin  3 = D;         /* OR input */
Pin  4 = E;         /* NAND input */
Pin  5 = F;         /* NAND input */
Pin  6 = G;         /* NOR input */
Pin  7 = H;         /* NOR input */
Pin  8 = I;         /* XOR input */
Pin  9 = J;         /* XOR input */
Pin 11 = K;         /* XNOR input */
Pin 12 = L;         /* XNOR input */

/* Outputs */

Pin 18 = U;         /* AND output */
Pin 17 = V;         /* OR output */
Pin 16 = W;         /* NAND output */
Pin 15 = X;         /* NOR output */
Pin 14 = Y;         /* XOR output */
Pin 13 = Z;         /* XNOR output */


/* Logic Equations */

U = A & B;                  /* AND */
V = C # D;                  /* OR */
W = !(E & F);               /* NAND */
X = !(G # H);               /* NOR */
Y = I & !J # !I & J;        /* XOR */
Z = !(K & !L # !K & L);     /* XNOR */
```

Basic Gates — Documentation File

```
***********************************
           Basic_Gates
***********************************

ST-CUPL     4.0a Serial# ST-15000002
Device      g16v8s
Library     DLIB-h-200-9
Created     Mon Nov 04 15:42:08 1991
Name        Basic_Gates
Partno      B_GATES
Revision    00
Date        None
Designer    None
Company     SGS-THOMSON
Assembly    None
Location    None


====================================
      Expanded Product Terms
====================================
U = A & B
V = C # D
W = E & F
X = G # H
Y = I & !J # !I & J
Z = K & !L # !K & L
```

```
=====================================
            Symbol Table
=====================================
Pin  Var                       PT   Max  Min
Pol  Name     Ext     Pin  Type Used  PT  Level
-------------------
     A                19    V    -    -    -
     B                1     V    -    -    -
     C                2     V    -    -    -
     D                3     V    -    -    -
     E                4     V    -    -    -
     F                5     V    -    -    -
     G                6     V    -    -    -
     H                7     V    -    -    -
     I                8     V    -    -    -
     J                9     V    -    -    -
     K                11    V    -    -    -
     L                12    V    -    -    -
     U                18    V    1    8    1
     V                17    V    2    8    1
     W                16    V    1    8    1
     X                15    V    2    8    1
     Y                14    V    2    8    1
     Z                13    V    2    8    1

LEGEND       F : field
             D : default variable
             M : extended node
             N : node
             I : intermediate variable
             T : function
             V : variable
             X : extended variable
             U : undefined



=====================================
            Fuse Plot
=====================================

Syn   02192 - Ac0   02193 x

Pin #19  02048  Pol x  02120  Ac1 -
 00000 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00032 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00064 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00096 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00128 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00160 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00192 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00224 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #18  02049  Pol -  02121  Ac1 x
 00256 -x--x---------------------------
 00288 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00320 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00352 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00384 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00416 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00448 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00480 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #17  02050  Pol -  02122  Ac1 x
 00512 x-------------------------------
 00544 -x------------------------------
 00576 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00608 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00640 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00672 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00704 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00736 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #16  02051  Pol x  02123  Ac1 x
 00768 ----x--x------------------------
 00800 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00832 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00864 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00896 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00928 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00960 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00992 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #15  02052  Pol x  02124  Ac1 x
 01024 --------x-----------------------
 01056 ----------x---------------------
 01088 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01120 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01152 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01184 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01216 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01248 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #14  02053  Pol -  02125  Ac1 x
 01280 ------------x--x----------------
 01312 -------------x-x----------------
 01344 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01376 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01408 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01440 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01472 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01504 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
Pin #13  02054  Pol x  02126  Ac1 x
 01536 --------------x-x-
 01568 -------------x--x
 01600 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01632 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01664 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01696 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01728 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01760 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #12  02055  Pol x  02127  Ac1 -
 01792 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01824 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01856 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01888 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01920 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01952 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01984 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 02016 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

LEGEND       X : fuse not blown
             - : fuse blown
```

```
====================================
            Chip Diagram
====================================

             ______________
            | Basic_Gates  |
        B x---|1          20|---x Vcc
        C x---|2          19|---x A
        D x---|3          18|---x U
        E x---|4          17|---x V
        F x---|5          16|---x W
        G x---|6          15|---x X
        H x---|7          14|---x Y
        I x---|8          13|---x Z
        J x---|9          12|---x L
      GND x---|10         11|---x K
            |______________|
```

The alert reader would remark that the fuse map generated in this case by the ST-CUPL™ replicates the fuse map reported in Figure 16, but that it also shows the use of the polarity inversion implemented in the outputs 13, 15 and 16 (as well as all the architecture bits).

DEVELOPMENT TOOLS

SOFTWARE AND HARDWARE TOOLS

SGS-THOMSON Microelectronics is specialized in the business of designing and manufacturing EECMOS programmable logic device as well as a broad range of commodity and semi-custom semiconductor components. In addition SGS-THOMSON supplies software and hardware to program our line of GAL® devices (ST-CUPL™ and GAL Starter Kit). Moreover many excellent third party suppliers provide software and hardware which fully support the SGS-THOMSON GAL® devices as well as other PLD devices.

Most current vendors of PLD programming hardware and software support GAL® devices. However, an upgrade of some programming hardware may be necessary to support GAL®. Vendors and equipment model numbers which have been qualified on GAL® devices are on file and may be obtained from SGS-THOMSON sales offices, a listing of which is provided at the back of this handbook.

It may be instructive for the user to contact the local SGS-THOMSON sales office to determine if a specific vendor's hardware is qualified for programming GAL® devices. SGS-THOMSON recommends that only qualified hardware be used to program GAL® devices as this will ensure the 100% functional and programming yield of SGS-THOMSON GAL® devices.

Software Tools

The availability of user-friendly and functional software tools is the main contribution to the tremendous upsurge in the usage of PLD devices. In the early 70's PLDs had a difficult time being accepted by systems and board designers due to the lack of good programming software. At that time it was necessary to load each individual fuse location into the devices, after extensive analysis of the design requirements. As this was a slow, cumbersome process requiring the designer to learn the architecture of many different devices in addition to the fact that logic errors could not be automatically identified, the acceptance of PLDs into mainstream system design was quite limited.

The late 70's saw the development of assembler software by various PLD vendors in order to help the sales of these devices. The most popular of these assemblers is PALASM®, from Monolithic Memories. This assembler allows inputs only as Boolean equations, has a difficult command structure, allows equations only in a sum-of-products format, works on PAL devices only and has no intelligence i.e. unable to do logic minimisation or identify specific device types which will or will not work with a given set of Boolean equations.

Of course the alternative, manual fuse patterning of the device, was much worse so assemblers found wide market acceptance for PAL device programming. The most severe restriction of this type of approach was the inflexibility of the software to work on other vendors devices thus forcing designers to learn many different assemblers in order to have more than one device supplier to choose from. Fortunately for PLD manufactures and system designers a more generic software approach was not far away.

The development of compiler based software in the early 80's was a response to the need for more flexibility and utility in development tools. The original packages were developed by third-party manufacturers, not device vendors, with the goal of supporting all device types and all manufacturers. These original packages — CUPL™by Logical Devices Inc. (Fort Lauderdale, FL) and ABEL™ from Data I/O Corp. (Redmond, WA) — had the capability of logic equation minimisation, macros, truth table and state machine syntax, logic simulation and self-documentation (examples of use of the ST-CUPL™ package are contained in this guide).

The latest advance in the PLD development software technology has occurred in the mid-'80s. These programs allow schematic capture using pre-programmed macros in the software which allow a designer to simply create a logic schematic as the input to a translator. The translator converts the graphic representation to a network list that is then compiled to the fuse maps by the software tool. All the other functions of the software such as logic minimisation are then available to streamline the design before it is downloaded into a device. The most widely used of these tools are OrCAD® (OrCAD L.P., Hillsboro, Oregon) and DASH from FutureNet® (Data I/O Corp., Redmond, WA).

Hardware Tools

The hardware used to program GAL® devices (a list of qualified hardware vendors is available from SGS-THOMSON sales offices, listed at the back of this manual) can be divided roughly into two types:

- the so-called "universal programmers" (in this case "universal" means with respect to PLD device only and this terminology should not be confused with the broader sense of "universal" programmers, meaning those that program EPROM memories or EPROM arrays in microprocessors as well as PLDs, although many of the programmers listed do program the above mentioned device families too);
- the "GAL® only" type programmers.

In the category of universal type programmers are those from Data I/O, Logical Devices and Stag Microsystems, as well as many others. These programmers support many different PLD devices, including ECL, CMOS EPROM, standard bipolar PALs and GAL® EECMOS device types. These universal programmers also support many advanced functions such as test vectors, register preload and even chip handlers for automated handling in a production environment. The second type of programmers are the "GAL® only" type programmers such as those from Qwerty and the GAL Starter Kit programmer. The Qwerty also supports test vectors and register preload for full functional testing of the GAL® devices. The big advantage of the GAL® only programmers is, of course, their low cost which is from one-fourth to one-tenth of the cost of universal programmers. This makes them an excellent low cost development system for GAL® devices.

A further division can be made with respect to the operating mode of the programmers i.e. the "standalone" type units and those which require an IBM® PC or equivalent to operate.

Generally the stand-alone units can read a device, store the JEDEC file in a memory and download the information to a newly loaded device including test vectors such that a volume "program and test" operation is possible without having a PC attached to each programmer.

Of course a connection to a peripheral such as a PC is still necessary to load a software developed JEDEC file into the programmer's memory.

The type of tool chosen should reflect the environment it will be used in: e.g. a GAL® only programmer could be considered in an operation where GAL® only development or a small volume of production is occurring in a situation where a low-cost evaluation and programming of GAL® is necessary. However in a large development lab where many types of PLDs are being evaluated or a high volume production environment where automated handling of devices is necessary, a universal type programmer with chip handler may be more appropriate. A prime consideration should also be the necessary functions of the programmer as well. For example, although SGS-THOMSON guarantees a 100% programming yield on our devices, test vector and register preload support capability is recommended for the designer to verify that the device is doing exactly what was planned.

This is especially important in state machine design where bringing the outputs to a certain state simply by cycling the inputs can take an inordinate amount of time or may not even be possible.

With register preload the outputs can be driven to a specific state (one which may not occur in normal operation) and then next state operation can be checked to verify that the device returns to a normal, expected state. Where this can become an issue is after a power "glitch" in the system or after a power outage when the GAL® outputs may be driven to an unknown or illegal state. Then when the power to the device returns to normal the GAL® outputs may be in one of these indeterminate states.

Thus the concept of register preload allows the designer to test output conditions which may not normally occur, but which the designer wishes to know anyway what the next state will be.

The designer can then verify that his design ensures the return of the device, from any preloaded illegal state, at its next transition, into one of the legal states. This is just one example of the utility of the register preload function.

Applying test vectors is simply the process of forcing specific inputs to the device and monitoring the outputs to verify the correct logical functioning of the device.

Programming the GAL® is the process of providing it with the JEDEC file to perform a function and applying the specific series of voltage pulses to actually "burn" the JEDEC "fuse" map into the device. Also, if included in the JEDEC file, the programmer then applies the specific voltage pulses and looks at the outputs to run the test vector verification for functional testing of the device.

"Support" by a hardware manufacturer refers to his ability to provide the correct voltages and timing pulses and make the correct measurements on the outputs, if applicable, for the device. Given this the remainder of the process is merely downloading the JEDEC file to the GAL®.

Downloading is the process of loading the JEDEC "fuse" map into the programmer. This "fuse" map can come from a pre-programmed device (i.e. master device), from a computer (a PC or mainframe with appropriate software to construct the JEDEC file from a number of logic or graphic entry methods), or from an attached peripheral such as a tape drive. If the file is transferred in JEDEC format (which is recommended by SGS-THOMSON), a checksum is calculated and verified at the end of data transmission to ensure that data was not lost or corrupted during transfer. Most programmers require a simple keystroke or pushbutton to put the hardware into the download mode.

The programming of the GAL® is controlled by the programming hardware and therefore, to ensure complete reliability of the SGS-THOMSON GAL® device, SGS-THOMSON recommends that only approved programmers be used. A list of qualified vendors is available from SGS-THOMSON upon request.

Since the GAL® device is fabricated using an EECMOS technology it is erasable and reprogrammable. In fact the first step in the programming process is a bulk erase of the device. The actual number of program and erase cycles the device will tolerate is guarantee by SGS-THOMSON to be at least 100. We have seen devices in our lab, however, which have thousands of cycles on them and still continue to function.

The ultimate test is a successful verification of the device which occurs automatically after programming. If no error occur during the verify cycle the part is programmed and usable no matter how many program-erase cycles it has been through.

The patterning of the GAL® device array is done using a parallel programming scheme. This allows the device to be programmed very quickly and in fact it takes less than a second on most programming hardware. This is up to an order of magnitude faster than devices using the UV-CMOS approach. During this programming cycle, the logic array, the architecture matrix programming and the verify cycle are executed. The verify cycle checks programming and margins conservatively such that a minimum data retention of 20 years is ensured.

Finally the GAL® device has a security cell which is programmable and erasable (but it can only be erased during a bulk erase when the entire array will also be erased). This prevents a “read” and therefore does not allow re-verification of the device.

This feature is provided to protect proprietary designs from competitive eyes. The fact that the security cell is erasable (simultaneously with the rest of the device) allowing the device to be reused, is a unique cost saving feature of EECMOS technology.

PROGRAMMING THE GAL6001S

PROGRAMMING THE GAL6001S

GAL6001S is a device with a complex architecture, that can be exploited only with a deep knowledge of its internal features.

The architecture of this device offers a flexibility never before achieved with GAL®s, providing proportionally higher performance in a much larger range of potential applications.

This note describes the building blocks inside the device, and explains how to configure and program each block.

The ST-CUPL™ development software is used in the descriptions and examples given, because of its versatility and power.

Examples Format

All the examples are given using the ST-CUPL™ format. To describe syntax the following conventions are used:

- angular brackets indicate a parameter that must be provided by the user;
- a symbol in square brackets is optional, i.e. it may be stated or not; for example

  PIN *<pin_n>* = [!] *<var_name>*;

  means that you can write

  PIN 3 = rw;

  or

  PIN 4 = !wr_en;

Variables can be grouped using the list notation, in the form:

[*<var$_1$>*,*<var$_2$>*, ...,*<var$_k$>*,*<varX>*..*<varY>*]

where *<var$_i$>* is any variable name, while *<varX>*..*<varY>* are indexed variables, i.e. variables ending with a number; an example of a valid list is

[a0..2,a5,a8,a10..13,a16]

that is equivalent to

[a0,a1,a2,a5,a8,a10,a11,a12,a13,a16]

Of course, in the case of lists, square brackets do not indicate optional parameters, but are part of the notation.

The table below reports the logic operators — in order of precedence (from highest to lowest) — that can be used to write equations; the precedence may be changed by grouping sub-expressions within parentheses.

Operator	Symbol
NOT	!
AND	&
OR	#
XOR	$

GAL6001S Architecture

GAL6001S has an FPLA architecture, with both the AND and the OR array programmable: this means that the exact number of product terms needed for an equation can be employed, allowing the implementation of more complex logic functions and a better exploitation of the device.

The AND array is a 78×75 matrix organized as 39 inputs (each available both in true and complemented form and coming from Input, Input/Output, Buried and Output Logic Macrocells) and 75 product term outputs. Sixty-four of these product terms serve as inputs to the OR array, one is the reset signal for the Output and Buried Logic Macrocells while ten act as output enable for pins 14 through 23.

The OR array is a 64×36 matrix organized as 64 inputs coming from the AND array and 36 outputs towards Buried and Output Logic Macrocells; in each macrocell feeds one *Data* (*"D"*) term and one *Enable/clock* (*"E"*) term.

The device includes a total of 38 logic macrocells, divided into two major sections: the first is used for the input and input/output pins, while the other determines the configuration of the outputs and the *"buried nodes"*.

Input Logic Macrocells (ILMC) and Input/Output Logic Macrocells (IOLMC)

The GAL6001S features two configurable input sections. The ILMC section corresponds to the dedicated input pins (from 2 to 11) and the IOLMC section to the input/output pins (from 14 to 23): both sections are configurable as a block for asynchronous, latched or registered inputs.

Pin 1 (ICLK) is used as an enable signal for latched macrocells (transparent when high) and as a clock for registered macrocells (positive edge triggered). In any case, it is also available as direct input (without macrocell) to the AND array and can be used in any equation as any other signal.

Registered inputs can be efficiently used for synchronization and data merging; transparent latches are useful when the input data is invalid outside a known time window; direct inputs are used in systems where the input data is always valid. With the GAL6001S, external registers and latches are no longer needed: it is sufficient to employ the internal features of the device.

In order to configure ILMCs and IOLMCs, two different extensions are provided in the ST-CUPL™ language: if the signal coming from an input pin is used in an equation with the extension ".LQ", the whole block to which that input belongs (ILMC or IOLMC) is configured as latched; if the extension used is ".DQ" it is configured as registered; finally, if the signal is used without extensions, the block of macrocells is configured as asynchronous.

Extensions are added to a signal in the form

<name>.<ext>

where *<name>* is any valid variable name; for example

```
/* Inputs */
pin 2 = In_A;

/* Outputs */
pin 17 = L_Out;

/* Logic Equations */
L_Out = In_A.LQ;
```

configures the ILMCs as latched, and connects the output of the latch (pin 2) to the output pin 17.

You can not mix different extensions for signals coming from the same block of macrocells: for example, if you use the extension ".LQ" for a signal coming from input pin 2, you can not use the extension ".DQ" for a signal coming from input pin 5.

The following table reports the extensions used by ST-CUPL™ to configure ILMCs and IOLMCs, together with the side of equations (i.e. the side of the equal sign) in which the extension can be used.

Extension	Side of Equation	Description
.DQ	Right	Q output of D-type register
.LQ	Right	Q output of latch

Output Logic Macrocell (OLMC) and Buried Logic Macrocell (BLMC)

The outputs of the OR array feed two groups of macrocells. One group of eight macrocells, called Buried Logic Macrocells (BLMC), is buried; its output feeds back directly into the AND array rather than to device pins. These cells are useful for building state machines. The second group of macrocells consists of 10 cells whose outputs — in addition to feeding back into the AND array — are available at the device pins. Cells in this group are known as Output Logic Macrocells (OLMC).

Output Logic Macrocells always have I/O capability, with directional control provided by 10 output enable product terms, selected with the extension ".OE" after the pin name: when the output enable signal is low, the buffer is in tri-state and you can use the corresponding pin as input.

There are two possible feedback paths from each OLMC to the AND array: one directly from the OLMC (this feedback is before the output buffer and always present), and one from the OLMC after the output buffer through the IOLMC. The second path is usable as a feedback only when the associated bidirectional pin is being used as an output. With this dual feedback arrangement, the OLMC can be permanently buried (and in that case the associate pin is an input), or dynamically buried with the use of the output enable product term.

The two different feedbacks from the OLMCs can be selected by means of extensions; more precisely:

- the extension ".INT" selects the feedback before the output buffer, i.e. not affected by the output enable control;
- the extension ".IO" selects the feedback after the output buffer through IOLMCs configured as asynchronous;
- the extension ".IOL" selects the feedback after the output buffer through IOLMCs configured as latched;
- the extension ".IOD" selects the feedback after the output buffer through IOLMCs configured as registered.

In order to use Buried Logic Macrocells, a symbolic name must be assigned to them. This can be done in two different ways:

- using the keyword PINNODE with the following syntax

  PINNODE *<node_n>* = [!] *<var_name>*;

  where *<node_n>* is a valid node number; in this case the macrocell corresponding to *<node_n>* is associated to *<var_name>*;
- with the statement NODE in the form

  NODE [!] *<var_name>*;

  which allows the compiler to select any Buried Logic Macrocell not yet allocated.

By means of the keyword PINNODE, it is also possible to configure the Output Logic Macrocells as buried nodes, using the corresponding pin as an input, in addition to the feedback into the AND array (before the output buffer).

The pin numbers to be used with the PINNODE keyword are reported below.

Logic Macrocell	Node Number
BLMC 0	25
BLMC 1	26
BLMC 2	27
BLMC 3	28
BLMC 4	29
BLMC 5	30
BLMC 6	31
BLMC 7	32
OLMC 14	33
OLMC 15	34
OLMC 16	35
OLMC 17	36
OLMC 18	37
OLMC 19	38
OLMC 20	39
OLMC 21	40
OLMC 22	41
OLMC 23	42

For instance, a declaration of inputs, outputs and buried nodes is:

```
/* Inputs */

pin 2 = In_A;
pin 6 = Out_En0;
pin 7 = Out_En1;
pin 23 = In_B;

/* Outputs */

pin [22..19] = [Y0..3];

/* Declarations and Intermediate
   Variable Definitions */

pinnode 42 = Bur_Feed;
node L;
pinnode 30 = M;
```

Output Logic Macrocell 23 is declared as buried node (node number 42) and the corresponding pin is used as input (*In_B*). Output logic macrocell can be treated as buried registers (using the extension ".D" to specify the data input) or as buried combinatorial nodes (referring to the macrocell name without extensions).

The statement

```
pinnode 30 = M
```

assigns the symbolic name *M* to the Buried Logic Macrocell 5, while *L*, named in the statement

```
node L;
```

is any available BLMC.

The extensions to be used to select feedbacks from BLMCs and OLMCs are the followings.

Extension	Side of Equation	Description
.INT	Right	Internal feedback (before the output buffer)
.IO	Right	Feedback after the output buffer through asynchronous IOLMC
.IOD	Right	Feedback after the output buffer through registered IOLMC
.IOL	Right	Feedback after the output buffer through latched IOLMC

Assuming the declarations of the previous example, consider the following:

```
/* Logic Equations */

Y0 = In_B;
Y0.OE = Out_En0;

Bur_Feed = In_A;

Y1 = Bur_Feed;
Y1.OE = Out_En1;

Y2 = Y0.INT;
Y3 = Y0.IO;
```

In this case OLMC 23 is used as a buried combinatorial node, assigning to its input the signal *In_A* without using extensions.

The feedback signal from the Output Logic Macrocell 23 (*Bur_Feed*) is used in the equation

```
Y1 = Bur_Feed
```

The *Y2* signal is the feedback of *Y0* before the output buffer (i.e. not affected by the output enable control *Y0.OE = Out_En0*) and *Y3* is the feedback of the same signal (*Y0*) after the buffer, through an asynchronous Input/Output Logic Macrocell.

The last equation is equivalent to one in which no extension is used.

The default feedback path from the Output Logic Macrocells is different depending on the macrocell configuration:

- if the OLMC is asynchronous then the default feedback is after the output buffer, i.e.

```
pin [22..21] = [Y1..0];

Y0 = In0;
Y1 = Y0.IO;
```

  is equivalent to:

```
pin [22..21] = [Y1..0];

Y0 = In0;
Y1 = Y0;
```

  of course, the internal feedback can be always chosen using the extension ".INT".

- if the OLMC is registered then the default feedback is the internal one, i.e.

```
pin [22..21] = [Y1..0];

Y0.D = In0;
Y1 = Y0;
```

  selects the feedback before the output buffer (not affected by the output enable control); it is not possible to use the ".INT" extension with registered output logic macrocells; the feedback after the output buffer can be selected with the extension ".IO".

Buried and Output Logic Macrocells may be set to one of three valid configurations: combinatorial, D type registered with sum term (asynchronous) clock or D/E type registered; these macrocells can be configured on a macrocell by macrocell basis, regardless to which block they belong.

The *Enable/clock* sum terms are used as asynchronous clocks in the asynchronous D type registered configuration or as clock enable signals in the D/E type registered configuration.

The data input of any registered macrocell is specified by adding the extension ".D" to the macrocell name. The asynchronous clock input is selected by the extension ".CK", while the extension ".CE" specifies the clock enable signal for D/E type registers. In both cases the corresponding macrocell is automatically configured in the correct way. Of course, you can not add both the extensions ".CK" and ".CE" to the same macrocell name.

Registers in both the Output and Buried Logic Macrocells feature a common reset product term; this active high product term allows the registers to be asynchronously reset to a logic zero. If connected to an output pin, a logic one will occur because of the inverting output buffer.

The asynchronous reset signal can be specified using the extension ".AR"; for instance:

```
/* Inputs */
pin 3 = Rst;

/* Logic Equations */
L.AR = Rst;
M.AR = Rst;
```

The reset signal is generated by a unique product term, common to all the device: so it is not possible to write different reset expressions for the various macrocells.

Moreover the reset signal can not be generated by an equation not reducible to a logic product (because the product term is only one); for example:

```
M.AR = C & E # G;
```

is not a valid equation because the sum can not be eliminated (the implementation of such an equation would require two different product terms summed together).

The extensions reported below can be used to manage OLMCs and BLMCs.

Extension	Side of Equation	Description
.D	Left	Data input of register
.OE	Left	Output enable signal
.AR	Left	Asynchronous reset signal
.CE	Left	Clock enable signal of D/E type register
.CK	Left	Asynchronous clock of D type register

Fuse Plot Format in Documentation Files

The ST-CUPL™ compiler can generate a documentation file containing useful information about the design, such as expanded product terms, symbol table and chip diagram.

It comprehend also a fuse plot that represents in a more readable way the Jedec file that will be used to program the device; by means of this plot you can see how the part will be really programmed.

Examples of fuse plots can be found in Figures 3 and 8.

Assuming that a “0” (zero) in the Jedec file means that the corresponding memory cell in the device is programmed (i.e. the connection is made) and that a “1” (one) means that the cell is not programmed (i.e. there is no connection), the symbols used in the fuse plot are the following:

Fuse Plot Symbol	Jedec File Bits
A	0
.	1
0	00
H	01
L	10
-	11

The first two symbols are used in the macrocell configuration and in the OR array sections.

The last four symbols stand for two bits and are used in the AND array part; more precisely “-” means that no connection is made, “0” means that both connections are present (i.e. the corresponding product term is always at a logic level low), “L” means that the complemented input is connected, while “H” selects the true input.

The first lines of the plot show the configuration of the logic macrocells; more precisely:

- the first eight lines are referred to BLMCs; the node number, the starting memory address (i.e. the memory address of the first bit mentioned) and the mode of configuration (CKS(i), OUTSYN(i) and XOR$_E$(i) bits) are reported for every BLMC;
- the next ten lines contain pin number, node number, starting memory address and mode of configuration (CKS(i), OUTSYN(i), XOR$_E$(i) and XOR$_D$(i) bits) for every OLMC;
- then the configuration bits for the ILMC and IOLMC blocks are reported.

After these lines, there is the memory map of the AND and OR arrays:

- lines with starting address from 0000 to 7182 are referred to the AND array (first part of the line) and to the OR array (second part of the line);
- the ten output enable product terms are reported in lines from 7296 to 7998;
- the last line is the reset product term.

The address of any bit can be calculated adding the cardinal position of this bit in the line to the number reported at the beginning (taking care that bits in the AND section are two for every symbol).

Simulation File Format

The logic simulator included in the ST-CUPL™ package accepts as input an ASCII file composed by:

- the same header contained in the source file;
- an ORDER statement in which the signals involved in the simulation are declared (in the correct order);
- a VECTORS section that specifies the inputs and outputs signal values; this part can contain optional $MSG directives that indicate some strings to be included in the simulation output.

Input signals value must be specified with the following symbols:

Symbol	Signal Value
0	Input low
1	Input high
C	Clock input low, high, low
K	Clock input high, low, high

Output signals can be tested using the following symbols:

Symbol	Signal Value
L	Test output low
H	Test output high
Z	Test output for high impedance (tri-state)
X	Don’t care value, the output value is undefined (it may be high or low)

If the simulator finds that some values do not match, the user is alerted and both the expected and the actual values are specified.

It is also possible to specify asterisks (“*”) instead of output test values. In this case the simulator will calculate the output test values.

Moreover, the waveform simulation output can be automatically obtained.

PROGRAMMING EXAMPLES

Two programming examples that show how to configure the GAL6001S are reported below.

ILMC/IOLMC Programming Example

The example reported in Figure 1 (called *"ILMC/IOLMC example"*) shows how to configure ILMCs and IOLMCs as latched or registered inputs; more precisely Input Logic Macrocells are configured as latched, while Input/Output Logic Macrocells are registered. Two output pins are directly connected to the macrocell outputs in order to monitor their behaviour.

The functioning of the device can be checked with the logic simulator of the ST-CUPL™ (whose output is reported in Figure 2). As you can see, the input clock (ICLK) acts as a clock for the registers (triggering on the rising edge) and as enable for the latches (transparent when high). The corresponding timing diagram is shown in Figure 5.

Figure 3 reports the documentation file generated by ST-CUPL™, while Figure 4 contains the input file used by the simulator.

Figure 1. ILMC/IOLMC Example — Source Code

```
Name        ILMC6001 - ILMC/IOLMC GAL6001 Programming Example;
Partno      ILMC 6001;
Date        January 1992;
Revision    00;
Designer    Ernesto;
Company     SGS-THOMSON Microelectronics;
Assembly    None;
Location    None;
Device      G6001;

/* Inputs */
pin 1 = Iclk;        /* Clock of ILMCs and IOLMCs */
pin 2 = In_A;        /* Dedicated input pin */
pin 14 = In_B;       /* Bidirectional pin used as input */

/* Outputs */
pin 17 = L_Out;
pin 18 = R_Out;

/* Logic equations */
L_Out = In_A.LQ;        /* Input Logic Macrocells configured */
                        /* as Latched Inputs */
R_Out = In_B.DQ;        /* Input/Output Logic Macrocells configured */
                        /* as Registered Inputs */
```

Figure 2. ILMC/IOLMC Example — Simulation Results

```
==============================================================================
                               Simulation Results
==============================================================================
        Iclk     In_A   In_B     L_Out   R_Out
        ____________________

0001:    1        1      1         H       H
0002:    1        1      1         H       H
0003:    1        0      0         L       H
0004:    0        0      0         L       H
0005:    0        1      1         L       H
0006:    1        1      1         H       H
0007:    1        0      0         L       H
0008:    1        0      0         L       H
0009:    1        1      1         H       H
0010:    1        1      1         H       H
0011:    1        0      0         L       H
0012:    1        0      0         L       H
0013:    1        1      1         H       H
0014:    1        1      1         H       H
0015:    1        0      0         L       H
0016:    1        0      0         L       H
0017:    1        1      1         H       H
0018:    0        1      1         H       H
0019:    0        0      0         H       H
0020:    0        0      0         H       H
0021:    0        1      1         H       H
0022:    0        1      1         H       H
0023:    0        0      0         H       H
0024:    0        0      0         H       H
0025:    0        1      1         H       H
0026:    0        1      1         H       H
0027:    0        0      0         H       H
0028:    0        0      0         H       H
0029:    0        1      1         H       H
0030:    0        1      1         H       H
0031:    0        0      0         H       H
0032:    1        0      0         L       L
0033:    1        1      1         H       L
0034:    1        1      1         H       L
0035:    1        0      0         L       L
0036:    1        0      0         L       L
0037:    1        1      1         H       L
0038:    1        1      1         H       L
0039:    0        0      0         H       L
0040:    0        0      0         H       L
0041:    0        1      1         H       L
0042:    0        1      1         H       L
0043:    0        0      0         H       L
0044:    0        0      0         H       L
0045:    0        1      1         H       L
0046:    1        1      1         H       H
0047:    1        0      0         L       H
0048:    1        0      0         L       H
0049:    1        1      1         H       H
0050:    1        1      1         H       H
```

Figure 3. ILMC/IOLMC Example — Documentation File

```
*******************************************************************************
                                    ILMC6001
*******************************************************************************

CUPL             4.0a Serial# ST-15000002
Device           g6001  Library DLIB-h-200-13
Created          Tue Jan 7 12:00:00 1992
Name             ILMC6001 - ILMC/IOLMC GAL6001 Programming Example
Partno           ILMC 6001
Revision         00
Date             January 1992
Designer         Ernesto
Company          SGS-THOMSON Microelectronics
Assembly         None
Location         None


===============================================================================
                             Expanded Product Terms
===============================================================================

L_Out =
    In_A.lq

R_Out =
    In_B.dq

L_Out.oe  =
    1

R_Out.oe  =
    1


===============================================================================
                                  Symbol Table
===============================================================================

Pin Variable                                     Pterms   Max     Min
Pol   Name              Ext     Pin     Type      Used   Pterms  Level
--  -----         --     --     ---     ---   ---  ---

    Iclk                        1        V         -       -       -
    In_A                        2        V         -       -       -
    In_A                lq      2        X         -       -       -
    In_B                        14       V         -       -       -
    In_B                dq      14       X         -       -       -
    L_Out                       17       V         1       64      1
    R_Out                       18       V         1       64      1
    L_Out               oe      17       D         1       1       0
    R_Out               oe      18       D         1       1       0

LEGEND    F : field       D : default variable         M : extended node
          N : node        I : intermediate variable    T : function
          V : variable    X : extended variable        U : undefined

Total product terms merged: 0

Total product terms used by OR Arrays: 2
```

Figure 3. ILMC/IOLMC Example — Documentation File *(continued)*

```
===========================================================================
                               Fuse Plot
===========================================================================
Node #25  08154 Mode  AAA
Node #26  08157 Mode  AAA
Node #27  08160 Mode  AAA
Node #28  08163 Mode  AAA
Node #29  08166 Mode  AAA
Node #30  08169 Mode  AAA
Node #31  08172 Mode  AAA
Node #32  08175 Mode  AAA
Pin #14 Node #33  08178 Mode  AAAA
Pin #15 Node #34  08182 Mode  AAAA
Pin #16 Node #35  08186 Mode  AAAA
Pin #17 Node #36  08190 Mode  A.A.
Pin #18 Node #37  08194 Mode  A.A.
Pin #19 Node #38  08198 Mode  AAAA
Pin #20 Node #39  08202 Mode  AAAA
Pin #21 Node #40  08206 Mode  AAAA
Pin #22 Node #41  08210 Mode  AAAA
Pin #23 Node #42  08214 Mode  AAAA
INSYN  08218 A ILATCH  08219 A IONSYN  08220 A IOLATCH  08221 .
00000 -H------------------------------------- ......A.............................
00114 --------------H------------------------ ........A...........................
00228 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
00342 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
00456 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
00570 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
00684 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
00798 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
00912 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
01026 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
01140 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
01254 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
01368 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
01482 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
01596 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
01710 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
01824 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
01938 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
02052 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
02166 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
02280 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
02394 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
02508 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
02622 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
02736 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
02850 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
02964 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
03078 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
03192 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
03306 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
03420 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
03534 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
03648 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
03762 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Figure 3. ILMC/IOLMC Example — Documentation File *(continued)*

```
03876 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
03990 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
04104 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
04218 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
04332 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
04446 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
04560 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
04674 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
04788 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
04902 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
05016 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
05130 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
05244 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
05358 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
05472 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
05586 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
05700 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
05814 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
05928 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
06042 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
06156 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
06270 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
06384 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
06498 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
06612 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
06726 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
06840 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
06954 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
07068 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
07182 000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
07296 000000000000000000000000000000000000000
07374 000000000000000000000000000000000000000
07452 000000000000000000000000000000000000000
07530 000000000000000000000000000000000000000
07608 000000000000000000000000000000000000000
07686 --------------------
07764 --------------------
07842 000000000000000000000000000000000000000
07920 000000000000000000000000000000000000000
07998 000000000000000000000000000000000000000
08076 000000000000000000000000000000000000000

LEGEND     single fuse
           L : active level low        H : active level high
           A : active fuse not blown   . : active fuse blown
           double fuse
           0 : neither fuse blown      - : both fuses blown
           L : first fuse blown        H : second fuse blown
           A : generate                . : propagate
```

Figure 3. ILMC/IOLMC Example — Documentation File *(continued)*

```
===========================================================================
                                Chip Diagram
===========================================================================

                               ______________
                              |   ILMC6001   |
                      Iclk x---|1          24|---x Vcc
                      In_A x---|2          23|---x
                           x---|3          22|---x
                           x---|4          21|---x
                           x---|5          20|---x
                           x---|6          19|---x
                           x---|7          18|---x R_Out
                           x---|8          17|---x L_Out
                           x---|9          16|---x
                           x---|10         15|---x
                           x---|11         14|---x In_B
                       GND x---|12         13|---x
                              |______________|
```

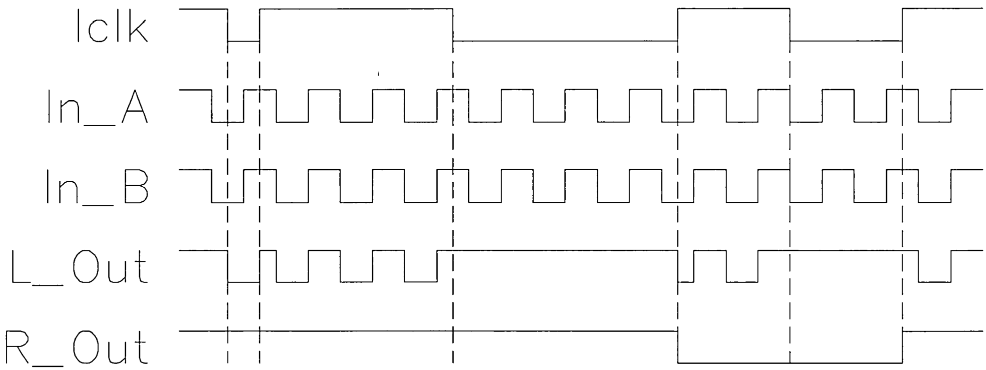
Figure 4. ILMC/IOLMC Example — Input File for Simulation

```
Name       ILMC6001 - ILMC/IOLMC GAL6001 Programming Example;
Partno     ILMC 6001;
Date       January 1992;
Revision   00;
Designer   Ernesto;
Company    SGS-THOMSON Microelectronics;
Assembly   None;
Location   None;
Device     G6001;

order: %2,Iclk,%7,In_A,%5,In_B,%7,L_Out,%6,R_Out;

vectors:

$msg “”;
$msg “          Iclk     In_A  In_B     L_Out  R_Out”;
$msg “          ____________________”;
$msg “”;

1 11 **
1 11 **
1 00 **
0 00 **
0 11 **
1 11 **
1 00 **
1 00 **
1 11 **
1 11 **
1 00 **
1 00 **
1 11 **
1 11 **
1 00 **
1 00 **
1 11 **
```

Figure 4. ILMC/IOLMC Example — Input File for Simulation *(continued)*

```
0 11 **
0 00 **
0 00 **
0 11 **
0 11 **
0 00 **
0 00 **
0 11 **
0 11 **
0 00 **
0 00 **
0 11 **
0 11 **
0 00 **
1 00 **
1 11 **
1 11 **
1 00 **
1 00 **
1 11 **
1 11 **
0 00 **
0 00 **
0 11 **
0 11 **
0 00 **
0 00 **
0 11 **
1 11 **
1 00 **
1 00 **
1 11 **
1 11 **
```

Figure 5. ILMC/IOLMC Example — Timing Diagram of the Simulation



OLMC/BLMC Programming Example

An example of the use of Output and Buried Logic Macrocells — together with other features of the device — is reported in Figure 6 (this example is called *"OLMC/BLMC example"*).

The bidirectional pin 23 is permanently used as an input utilizing the corresponding Output Logic Macrocell as a buried node by means of the statement:

```
pinnode 42 = Bur_Feed;
```

that associates the symbolic name *Bur_Feed* to OLMC 23.

Output *Y0* is directly connected to *In_B*; the output enable control is provided by the signal *Out_En0*.

OLMC 23 is configured as combinatorial buried node and its output is monitored from the external by directly connecting it to the output pin 21 (*Y1*).

The output signals *Y2* and *Y3* are the feedbacks of *Y0* (Output Logic Macrocell 22), respectively before and after the output buffer. Being the OLMC 22 configured as asynchronous, the default feedback is the internal one; so the equation for *Y3* should have been written as

```
Y3 = Y0;
```

form completely equivalent to the one used in the example.

The use of D/E type registers is illustrated with the statements for *YR0* and *L*; the clock enable signal

Figure 6. OLMC/BLMC Example — Source Code

```
Name        OLMC6001 - OLMC/BLMC GAL6001 Programming Example;
Partno      OLMC 6001;
Date        January 1992;
Revision    00;
Designer    Ernesto;
Company     SGS-THOMSON Microelectronics;
Assembly    None;
Location    None;
Device      G6001;

/* Inputs */
pin 13 = Oclk;      /* Clock of BLMCs and OLMCs */
pin 2 = In_A;       /* Dedicated input pin */
pin 23 = In_B;      /* Bidirectional pin used as input */
pin 3 = Rst;        /* Signal used as Asynchronous Reset of */
                    /* registered OLMCs and BLMCs */
pin 4 = As_Clk;     /* Signal used as Asynchronous Clock */
pin 5 = Hold;       /* Signal used as Clock Enable for D/E Registers */
pin 6 = Out_En0;    /* Signal used as Output Enable */
pin 7 = Out_En1;    /* Signal used as Output Enable */

/* Outputs */
pin [22..17] = [Y0..5];     /* Combinatorial outputs */
pin [16..15] = [YR0..1];    /* Registered outputs */

/* Declarations and Intermediate Variable Definitions */
pinnode 42 = Bur_Feed;     /* OLMC 23 declared as Buried Node */
node L;                    /* BLMC automatically selected */
pinnode 30 = M;            /* Selects BLMC 5 */

/* Logic equations */
Y0 = In_B;           /* Signal coming from pin 23 used as input */
Y0.OE = Out_En0;     /* Output Enable control */

Bur_Feed = In_A;     /* Input of OLMC 23 used as buried node */
```

is generated by connecting the negation of the input named *Hold* to the CE input of the registers. When the *Hold* signal is low (i.e. *!Hold* is high) the clock is enabled, so the flip-flop acts as a normal D type; if *Hold* is high, every commutation of the registers is forbidden, i.e. it is in a blocked state.

Output logic macrocell 18 (*YR1*) and buried logic macrocell 5 (*M*) are configured as D type registers with asynchronous clock; the clock signal is the one coming from input pin 4 (named *As_Clk*).

The *M* and *L* signals are brought to the external world connecting them to the output pins 17 and 18 (*Y5* and *Y4*).

In the example, for the sake of clarity, the expressions assigned to the clock enable and asynchronous clock signals are very simple (only a connection of a signal) and are the same for all the registers; obviously it is possible to use a different signal for every flip-flop, generated in any expression that can be implemented using the *"E"* sum term of the macrocells.

The reset signal — common to all registers — comes from input pin 3 (*Rst*); also in this case the expression used in the example is simple, but can be any one implementable using a single product term.

The results of the logic simulation is reported in Figure 7; the behaviour of every output signal is:

- the value of *Y0* is equal to *In_B*, except when *Out_En0* is low (vectors 4 to 6, 12 to 14 and 19 to 21); in these cases the output is in tri-state;
- *Y1* is equal to *Bur_Feed* (buried OLMC 23) that in turns is equal to *In_A*; again the output is in tri-state when *Out_En1* is low;
- the signal *Y2* is equal to the value assumed by Output Logic Macrocell 22 (*Y0*); so it is equal to *In_B* even when *Y0* is in tri-state (the feedback taken is the internal one);

Figure 6. OLMC/BLMC Example — Source Code *(continued)*

```
Y1 = Bur_Feed;       /* Feedback of OLMC 23 before the output buffer */
Y1.OE = Out_En1;     /* Output Enable control */

Y2 = Y0.INT;    /* Feedback before the output buffer */
Y3 = Y0.IO;     /* Feedback after the output buffer */
                /* This equation is equivalent to   Y3 = Y0; */

L.AR = Rst;     /* Asynchronous Reset of registers */
M.AR = Rst;     /* Asynchronous Reset of registers */
YR0.AR = Rst;   /* Asynchronous Reset of registers */
YR1.AR = Rst;   /* Asynchronous Reset of registers */

YR0.CE = !Hold;    /* Clock Enable for D/E type register */
YR0.D = In_B;      /* Data input of the register */

YR1.CK = As_Clk;   /* Asynchronous Clock for D type register */
YR1.D = In_B;      /* Data input of the register */

L.CE = !Hold;      /* Clock Enable for D/E type register */
L.D = In_A;        /* Data input of the register */

M.CK = As_Clk;     /* Asynchronous Clock for D type register */
M.D = In_A;        /* Data input of the register */

Y4 = L;
Y5 = M;
```

- the value of *Y3* is the same as *Y0*: when *Y0* is in tri-state, the value of *Y3* is undefined; in this case the feedback is affected by the output enable signal;
- *Y4* is the monitoring of *L* (a BLMC configured as D/E type register, having *In_A* as data input and *Hold* negated as clock enable input); from the simulation it can be noticed that when the *Hold* signal is high (vectors from 5 to 8) the state of the register is blocked (i.e. transitions are forbidden); outside this windows, the flip-flop acts as a D type register clocked by the *Oclk* signal;
- the monitoring of *M* (a BLMC configured as D type register, having *In_A* as data input and *As_Clk* as asynchronous clock input) is *Y5*; this flip-flop is clocked by the *As_Clk* signal;
- macrocell 16 (*YR0*) is configured as D/E type register, with *In_B* as data input and *Hold* negated as clock enable signal; its behaviour is analogous to that of *L*;
- macrocell 15 (*YR1*) is configured as D type asynchronous register, with *In_B* as data input and *As_Clk* as asynchronous clock signal; its behaviour is analogous to that of *M*.

All registers feature a common reset signal (in this case *Rst*): this signal causes a reset to a low state of all registers (vector 10). If the registers are connected to an output pin (*YR0* and *YR1*) a high value will be present on that pin, due to the inversion of the output buffer.

Figure 8 reports the documentation file generated by the ST-CUPL™ compiler, while Figure 9 contains the input file used by the simulator.

Figure 7. OLMC/BLMC Example — Simulation Results

```
==========================================================================================
                                  Simulation Results
==========================================================================================
    Oclk  In_A  In_B  Rst  Hold  As_Clk  Out_En0  Out_En1  Y0  Y1  Y2  Y3  Y4  Y5  YR0  YR1
    --------------------------------------------

0001: C     0     1     0    0     0          1        1   H   L   H   H   L   L    H    H
0002: C     1     1     0    0     1          1        1   H   H   H   H   H   H    H    H
0003: C     0     0     0    0     0          1        1   L   L   L   L   L   H    L    H
0004: C     1     1     0    0     0          0        1   Z   H   H   X   H   H    H    H
0005: C     0     1     0    1     1          0        1   Z   L   H   X   H   L    H    H
0006: C     1     1     0    1     0          0        0   Z   Z   H   X   H   L    H    H
0007: C     0     0     0    1     0          1        0   L   Z   L   L   H   L    H    H
0008: C     1     0     0    1     1          1        0   L   Z   L   L   H   H    H    L
0009: C     1     1     0    0     0          1        1   H   H   H   H   H   H    H    L
0010: C     0     1     1    0     0          1        1   H   L   H   H   L   L    H    H
0011: C     0     1     0    0     1          1        1   H   L   H   H   L   L    H    H
0012: C     1     0     0    0     0          0        1   Z   H   L   X   H   L    L    H
0013: C     0     1     0    0     0          0        1   Z   L   H   X   L   L    H    H
0014: C     1     1     0    0     1          0        0   Z   Z   H   X   H   H    H    H
0015: C     0     1     0    0     1          1        0   H   Z   H   H   L   H    H    H
0016: C     1     0     0    0     0          1        0   L   Z   L   L   H   H    L    H
0017: C     0     0     0    0     1          1        1   L   L   L   L   L   L    L    L
0018: C     1     0     0    0     0          1        1   L   H   L   L   H   L    L    L
0019: C     0     1     0    0     0          0        1   Z   L   H   X   L   L    H    L
0020: C     1     1     0    0     1          0        1   Z   H   H   X   H   H    H    H
0021: C     0     1     0    0     0          0        1   Z   L   H   X   L   H    H    H
```

Figure 8. OLMC/BLMC Example — Documentation File

```
*************************************************************************
                                   OLMC6001
**************************************************************************
CUPL            4.0a Serial# ST-15000002
Device          g6001  Library DLIB-h-200-13
Created         Tue Jan 7 12:00:00 1992
Name            OLMC6001 - OLMC/BLMC GAL6001 Programming Example
Partno          OLMC 6001
Revision        00
Date            January 1992
Designer        Ernesto
Company         SGS-THOMSON Microelectronics
Assembly        None
Location        None
==========================================================================
                            Expanded Product Terms
==========================================================================
Bur_Feed =
    In_A
L.ar  =
    Rst
L.ce  =
    !Hold
L.d  =
    In_A
M.ar  =
    Rst
M.ck  =
    As_Clk
M.d  =
    In_A
Y0 =
    In_B
Y0.oe  =
    Out_En0
Y1 =
    Bur_Feed
Y1.oe  =
    Out_En1
Y2 =
    Y0.int
Y3 =
    Y0.io
Y4 =
    L
Y5 =
    M
YR0.ar  =
    Rst
YR0.ce  =
    !Hold
```

Figure 8. OLMC/BLMC Example — Documentation File *(continued)*

```
YR0.d  =
    In_B
YR1.ar  =
    Rst
YR1.ck  =
    As_Clk
YR1.d  =
    In_B
In_B.oe  =
    0
Y2.oe  =
    1
Y3.oe  =
    1
Y4.oe  =
    1
Y5.oe  =
    1
YR0.oe  =
    1
YR1.oe  =
    1

===========================================================================
                                 Symbol Table
===========================================================================
Pin Variable                                         Pterms   Max      Min
Pol   Name                 Ext     Pin      Type      Used   Pterms   Level
-- ----               --   --      ---     ---  ---   ---
    As_Clk                         4        V         -       -        -
    Bur_Feed                       42       N         1       64       1
    Hold                           5        V         -       -        -
    In_A                           2        V         -       -        -
    In_B                           23       V         -       -        -
    L                              25       N         -       -        -
    L                      ar      25       M         1       1        1
    L                      ce      25       M         1       64       1
    L                      d       25       M         1       64       1
    M                              30       N         -       -        -
    M                      ar      30       M         1       1        1
    M                      ck      30       M         1       64       1
    M                      d       30       M         1       64       1
    Oclk                           13       V         -       -        -
    Out_En0                        6        V         -       -        -
    Out_En1                        7        V         -       -        -
    Rst                            3        V         -       -        -
    Y0                             22       V         1       64       1
    Y0                     int     22       X         -       -        -
    Y0                     io      22       X         -       -        -
    Y0                     oe      22       X         1       1        1
    Y1                             21       V         1       64       1
    Y1                     oe      21       X         1       1        1
    Y2                             20       V         1       64       1
    Y3                             19       V         1       64       1
```

Figure 8. OLMC/BLMC Example — Documentation File *(continued)*

```
      Y4                           18        V        1        64        1
      Y5                           17        V        1        64        1
      YR0                          16        V        -        -         -
      YR0               ar         16        X        1        1         1
      YR0               ce         16        X        1        64        1
      YR0               d          16        X        1        64        1
      YR1                          15        V        -        -         -
      YR1               ar         15        X        1        1         1
      YR1               ck         15        X        1        64        1
      YR1               d          15        X        1        64        1
      In_B              oe         23        D        1        1         0
      Y2                oe         20        D        1        1         0
      Y3                oe         19        D        1        1         0
      Y4                oe         18        D        1        1         0
      Y5                oe         17        D        1        1         0
      YR0               oe         16        D        1        1         0
      YR1               oe         15        D        1        1         0

  LEGEND     F : field      D : default variable        M : extended node
             N : node       I : intermediate variable   T : function
             V : variable   X : extended variable       U : undefined

  Total product terms merged: 6

  Total product terms used by OR Arrays: 9

  ==========================================================================
                                    Fuse Plot
  ==========================================================================

  Node #25  08154 Mode  .AA
  Node #26  08157 Mode  AAA
  Node #27  08160 Mode  AAA
  Node #28  08163 Mode  AAA
  Node #29  08166 Mode  AAA
  Node #30  08169 Mode  AAA
  Node #31  08172 Mode  AAA
  Node #32  08175 Mode  AAA
  Pin #14 Node #33  08178 Mode  AAAA
  Pin #15 Node #34  08182 Mode  AAA.
  Pin #16 Node #35  08186 Mode  .AA.
  Pin #17 Node #36  08190 Mode  A.A.
  Pin #18 Node #37  08194 Mode  A.A.
  Pin #19 Node #38  08198 Mode  A.A.
  Pin #20 Node #39  08202 Mode  A.A.
  Pin #21 Node #40  08206 Mode  A.A.
  Pin #22 Node #41  08210 Mode  A.A.
  Pin #23 Node #42  08214 Mode  A.AA
  INSYN  08218 . ILATCH  08219 . IONSYN  08220 . IOLATCH  08221 .
  00000 -H------------------ ..................A.....A.........A.
  00114 --L--------------- .....A..............................A
  00228 --H----------------- ...A......................A..........
  00342 ---------H--------- ..A.A...........A...................
  00456 ------------------H .............A.....................
  00570 ------------------H- ............A.......................
  00684 ---------H-------- ..........A.........................
  00798 ------H------------ ........A...........................
```

Figure 8. OLMC/BLMC Example — Documentation File *(continued)*

```
00912 --------H------------ ......A.............................
01026 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
01140 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
01254 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
01368 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
01482 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
01596 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
01710 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
01824 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
01938 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
02052 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
02166 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
02280 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
02394 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
02508 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
02622 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
02736 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
02850 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
02964 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
03078 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
03192 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
03306 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
03420 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
03534 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
03648 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
03762 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
03876 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
03990 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
04104 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
04218 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
04332 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
04446 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
04560 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
04674 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
04788 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
04902 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
05016 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
05130 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
05244 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
05358 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
05472 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
05586 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
05700 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
05814 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
05928 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
06042 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
06156 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
06270 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
06384 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
06498 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
06612 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
06726 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
06840 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
06954 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
07068 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
07182 0000000000000000000000000000000000000000 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
07296 0000000000000000000000000000000000000000
```

Figure 8. OLMC/BLMC Example — Documentation File *(continued)*

```
07374 ---H-----------------------------------
07452 ---H-----------------------------------
07530 ---------------------------------------
07608 ---------------------------------------
07686 ---------------------------------------
07764 ---------------------------------------
07842 ---------------------------------------
07920 ---------------------------------------
07998 000000000000000000000000000000000000000
08076 -H-------------------------------------

LEGEND    single fuse
          L : active level low        H : active level high
          A : active fuse not blown   . : active fuse blown
          double fuse
          0 : neither fuse blown      - : both fuses blown
          L : first fuse blown        H : second fuse blown
          A : generate                . : propagate

==========================================================================
                                Chip Diagram
==========================================================================

                             _______________
                            |   OLMC6001    |
                       x---|1            24|---x Vcc
                  In_A x---|2            23|---x In_B
                   Rst x---|3            22|---x Y0
                As_Clk x---|4            21|---x Y1
                  Hold x---|5            20|---x Y2
               Out_En0 x---|6            19|---x Y3
               Out_En1 x---|7            18|---x Y4
                       x---|8            17|---x Y5
                       x---|9            16|---x YR0
                       x---|10           15|---x YR1
                       x---|11           14|---x
                   GND x---|12           13|---x Oclk
                            |_______________|
```

Figure 9. OLMC/BLMC Example — Input File for Simulation

```
Name        OLMC6001 - OLMC/BLMC GAL6001 Programming Example;
Partno      OLMC 6001;
Date        January 1992;
Revision    00;
Designer    Ernesto;
Company     SGS-THOMSON Microelectronics;
Assembly    None;
Location    None;
Device      G6001;

order:
Oclk,%4,In_A,%4,In_B,%4,Rst,%3,Hold,%4,As_Clk,%7,Out_En0,%7,Out_En1,%4,Y0,%
2,Y1,%2,Y2,%2,Y3,%2,Y4,%2,Y5,%3,YR0,%3,YR1;
vectors:
$msg “”;
$msg “    Oclk In_A In_B Rst Hold As_Clk Out_En0 Out_En1 Y0 Y1 Y2 Y3 Y4
Y5 YR0 YR1";
$msg ″    ————————————————————————————————————";
$msg “”;
C 0100011 ********
C 1100111 ********
C 0000011 ********
C 1100001 ********
C 0101101 ********
C 1101000 ********
C 0001010 ********
C 1001110 ********
C 1100011 ********
C 0110011 ********
C 0100111 .********
C 1000001 ********
C 0100001 ********
C 1100100 ********
C 0100110 ********
C 1000010 ********
C 0000111 ********
C 1000011 ********
C 0100001 ********
C 1100101 ********
C 0100001 ********
```

PROGRAMMING EXAMPLES

PROGRAMMING EXAMPLES

The following examples demonstrate some typical application that can be implemented using GAL16V8 and GAL20V8. The ST-CUPL™ compiler is used for all the examples; a brief explanation of its syntax is reported below.

THE ST-CUPL™ SYNTAX

To describe syntax the following conventions are used:

- angular brackets indicate a parameter that must be provided by the user;
- a symbol in square brackets is optional, i.e. it may be stated or not; for example

  PIN *<pin_n>* = [!] *<var_name>*;

  means that you can write

  PIN 3 = rw;

  or

  PIN 4 = !wr_en;

File Header

The header contains information useful for archive and documentation purposes; some of them are effectively interpreted by the compiler:

- *Name* determines the name of the Jedec file generated by the compiler;
- *Partno* specifies the contents of the User Electronic Signature;
- *Device* indicates the type of device in which the design will be physically implemented (*G16V8* for the GAL16V8 and *G20V8* for the GAL20V8); the compiler automatically selects the correct configuration of the device, depending on the specific application; it is also possible to force the configuration using as device names the followings:
  - *G16V8S* or *G20V8S* to configure the device in simple mode (*small* mode);
  - *G16V8MA* or *G20V8MA* to configure the device in complex mode (*medium asynchronous* mode);
  - *G16V8MS* or *G20V8MS* to configure the device in registered mode (*medium synchronous* mode);

  sometimes it is useful to force the configuration of the device in order to exploit some special features (see the "Eight Set-Reset Flip-Flops" example).

Comments

Comments are an important part of the logic description file: they improve readability and are essential for documentation purposes (e.g. for subsequent modifications and maintenance of the design). Every text enclosed between the "/*" and the "*/" symbols is considered as a comment and therefore is completely ignored by the compiler.

Number Format

The default base in ST-CUPL™ for all numbers is hexadecimal, except for pin numbers and indexed variables that are always decimal. Different bases can be used by preceding the numbers with a prefix (chosen among those listed in table below).

Base Name	Base	ST-CUPL™ Prefix
Binary	2	'b'
Octal	8	'o'
Decimal	10	'd'
Hexadecimal	16	'h'

For example:

'b'1101

is a binary number equivalent to the decimal 13.

Preprocessor Commands

There are several preprocessor commands available in ST-CUPL™; these commands operate before the file is passed to the compiler, and include macro definition capabilities, conditional compilation, file inclusion and others.

The only preprocessor command used in the examples is $DEFINE, which has the following format:

$DEFINE *<arg$_1$>* *<arg$_2$>*

where *<arg$_1$>* is a variable name or an ASCII character while *<arg$_2$>* is a valid operator, a number or a variable name. For instance:

$DEFINE ON 'b'1.

will replace every occurrence of the string *ON* with the text *'b'1*.

List Notation

Variables can be grouped using the list notation, in the form:

[*<var$_1$>*,*<var$_2$>*, ...,*<var$_k$>*,*<varX>*..*<varY>*]

where *<var$_i$>* is any variable name, while *<varX>*..*<varY>* are indexed variables, i.e. variables ending with a number; an example of a valid list is

[a0..2,a5,a8,a10..13,a16]

that is equivalent to

[a0,a1,a2,a5,a8,a10,a11,a12,a13,a16]

Of course, in the case of lists, square brackets do not indicate optional parameters, but are part of the notation.

Bit Field Declaration

A bit field declaration assigns a single variable name to a group of bits, using the format:

FIELD *<var>* = [*<var1>*,*<var2>*,...,*<vark>*, *<varX>*..*<varY>*];

where *<var>* is any valid variable name, while in square brackets is enclosed a group of variable in the list notation.

For example:

FIELD addr = [a0..3,a7,a9];

is a legal bit field declaration.

The variable name assigned to a group of bits can then be used in any expression; the operation specified in the expression is applied to each bit in the group.

Pin Declaration

Pin declaration statements declare the pin numbers and assign to them symbolic names. The format of a pin declaration statement is:

PIN *<pin_n>* = [!] *<var_name>*;

where *<pin_n>* is a valid pin number, while *<var_name>* is any legal variable designator.

Pin numbers can be grouped using the list notation; for instance:

PIN [12..10] = [q2..0];

is a valid statement, that assigns to pins 10, 11 and 12 respectively the names *q0, q1* and *q2.*

Logic Operators

The table below reports the logic operators — in order of precedence (from highest to lowest) — that can be used to write equations; the precedence may be changed by grouping sub-expressions within parentheses.

Operator	Symbol
NOT	!
AND	&
OR	#
XOR	$

Equality Operator

The equality operator in ST-CUPL™ is represented by a colon; the format is

<var>:*<constant>*;

where *<var>* is a list of variables or a bit field variable, while *<constant>* is a number (hexadecimal by default).

The equality operator checks for bit equality and evaluates to a single Boolean value.

Alternatively this operator can work on a constant field constituted by a range of values instead of a single number. The format is as follows

<var>: [*<constant$_{low}$>*..*<constant$_{high}$>*];

The operators evaluates to true if <var> is a value comprised in the constant range indicated. For example:

sel = address: [C..F];

causes *sel* to be true if address is C, D, E or F and is equivalent to the expression

sel = address:C # address:D # address:E # address:F;

that uses the equality operator with single number constant field.

Extensions

Extensions are added to a variable name in the form

<var>.*<ext>*

and can be used to configure the Output Logic Macrocells. The extensions that can be used with the GAL16V8 and GAL20V8 are reported in the below table.

Extension	Side of Equation	Description
.D	Left	Data input of D-type register
.OE	Left	Output enable signal

Using the extension ".D", will configure automatically the corresponding OLMC as registered.

State Machine Syntax

The ST-CUPL™ syntax allows the user to describe the state machine diagram in a high level notation which is automatically translated by the compiler into a series of logic equations.

The general ST-CUPL™ notation to express the transitions of a state machine is the following:

```
SEQUENCE <state_vars_list>
{
 PRESENT <state1> <statement11>;
         <statement12>;
         ...
         <statement1k>;
 PRESENT <state2> <statement21>;
         <statement22>;
         ...
         <statement2j>;
 ...
 PRESENT <staten> <statementn1>;
         <statementn2>;
         ...
         <statementnh>;
}
```

where *<state_vars_list>* is a list of variables that keep the state of the machine, *<state$_i$>* is a value of that variables and *<statement$_{mn}$>* specifies the next state and the value of the outputs.

There are many different types of statement that permit the specification of conditional or unconditional transitions and the implementation of both Mealy and Moore machines (in a *Moore* machine the outputs depend only on the present state while in a *Mealy* machine they depend both on the present state *and* on the current value of the inputs).

The simpler statement is the unconditional transition:

```
NEXT <statej>;
```

indicating that the next state is *<state$_j$>* independently from the current value of inputs.

If you want to specify conditional transition the syntax is:

```
IF <cond1> NEXT <state1>;
IF <cond2> NEXT <state2>;
...
IF <condn> NEXT <staten>;
[DEFAULT NEXT <statem>;]
```

where *<cond$_i$>* is any valid logic expression, depending both on external inputs and/or current state; the optional DEFAULT statement specifies a next state *<state$_m$>* which is reached if all the previous conditions fail.

All the above listed statements can be followed by the expression:

```
OUT [!]<var1> OUT [!]<var2> ...
OUT [!]<varh>;
```

where *<var$_i$>* is a variable name associated with an output pin. In this case you are implementing a Moore machine, because the outputs are synchronous.

If you omit the NEXT section of the statement adding the OUT part, you will obtain asynchronous outputs, resulting in a Mealy machine.

Documentation File Format

The ST-CUPL™ compiler can generate a documentation file containing various information about the implementation of the design.

The first part of this file consist of a listing of the expanded product terms, i.e. the sum of product form of the original design. In fact, every application must be expressed in this form before it can be implemented in a GAL® device; this listing give an idea of the final physical form of the implementation.

The compiler accomplishes the job of translating the high level notation (complex equations, state machine, etc.) in the sum of product form and possibly minimize the equations using several algorithms.

Then a symbol table is reported, that contains all the variables used and some statistics on their physical implementation.

The documentation file also comprehends a fuse plot, that shows the memory map that will be written into the device. This plot is a more readable form of the Jedec file, used by the physical programmer to program the device.

Assuming that a "0" (zero) in the Jedec file means that the corresponding memory cell in the device is programmed (i.e. the connection is made) and that a "1" (one) means that the cell is not programmed (i.e. there is no connection), the corresponding symbols in the fuse plot are:

Fuse Plot Symbol	Jedec File Bits
	1
x	0

The first row in the fuse plot shows the SYN and AC0 bits (and the corresponding memory addresses) that are used to configure the whole device.

After follow eight sections (one for each output pin) containing:

- the pin number;
- the corresponding polarity bit (preceded by its memory address);
- the AC1 bit (with the memory address) that configures the specific OLMC;
- eight rows that represent the product terms feeding into the OLMC; each row begins with a number, representing the memory address of the first cell in the product term: the address of the other cell can be simply obtained adding to this number the cardinal position of the cell in the row.

Finally is reported a chip diagram, showing the name of each input and output pin.

Simulation File Format

The logic simulator included in the ST-CUPL™ package accepts as input an ASCII file composed by:

- the same header contained in the source file;
- an ORDER statement in which the signals involved in the simulation are declared (in the correct order);
- a VECTORS section that specifies the inputs and outputs signal values; this part can contain optional $MSG directives that indicate some strings to be included in the simulation output.

Input signals value must be specified with the following symbols:

Symbol	Signal Value
0	Input low
1	Input high
C	Clock input low, high, low
K	Clock input high, low, high
P	Preload internal registers

Output signals can be tested using the following symbols:

Symbol	Signal Value
L	Test output low
H	Test output high
Z	Test output for high impedance (tri-state)
X	Don't care value, the output value is undefined (it may be high or low)

If a preload operation is performed — indicating a "P" on the clock input — then output values must be specified (i.e. forced) using "0" and "1" instead of "L" and "H".

If the simulator finds that some values do not match, the user is alerted and both the expected and the actual values are specified.

It is also possible to specify asterisks ("*") instead of output test values. In this case the simulator will calculate the output test values.

Moreover, the waveform simulation output can be automatically obtained.

The $REPEAT directive, expressed in the form:

$REPEAT *<n>*

where *<n>* is a decimal value, can be used — in conjunction with asterisks as output values — to repeat *n* times the same input vector.

This directive is particularly useful when testing counters and state transitions.

Simple Gates Example

This example shows how ST-CUPL™ can implement a design containing simple gates; the design schematic is reported in Figure 1.

Figure 2 shows the source file, in Figure 3 are reported the simulation results, while Figures 4 and 5 contain respectively the documentation file generated by ST-CUPL™ and the input file for the simulator.

Figure 1. Simple Gates — Design Schematic

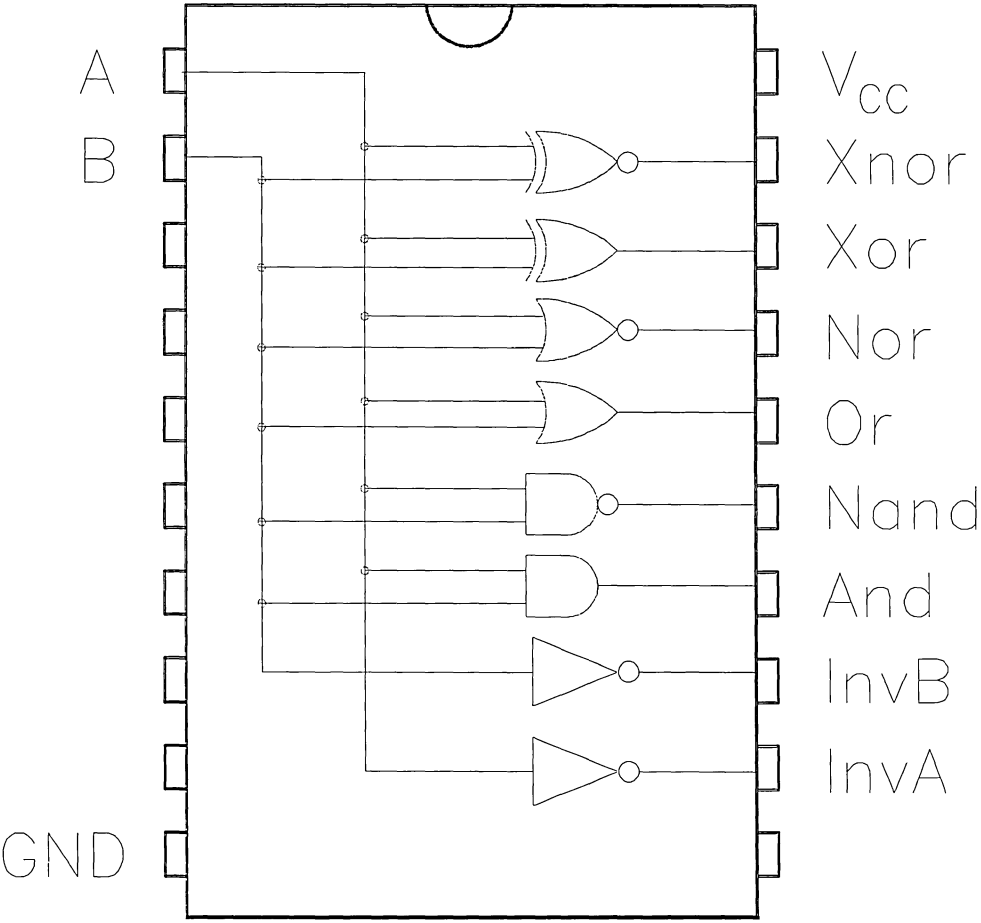


Figure 2. Simple Gates — Source Code

```
Name            Gates;
Partno          CA0001;
Revision        00;
Date            None;
Designer        G. Woolhiser;
Company         Logical Devices, Inc.;
Location        None;
Assembly        None;
Device          G16V8;

/****************************************************************/
/*                                                              */
/*      This is a example to demonstrate how ST-CUPL            */
/*      compiles simple gates.                                  */
/*                                                              */
/****************************************************************/
/*      Target Device: G16V8                                    */
/****************************************************************/

/* Inputs:  define inputs to build simple gates from */

Pin 1 =  A;
Pin 2 =  B;

/* Outputs:  define outputs as active HI levels */

Pin 12 = InvA;
Pin 13 = InvB;
Pin 14 = And;
Pin 15 = Nand;
Pin 16 = Or;
Pin 17 = Nor;
Pin 18 = Xor;
Pin 19 = Xnor;

/* Logic:  examples of simple gates expressed in ST-CUPL */

InvA = !A;              /* Inverters */
InvB = !B;              /* Inverters */
And  = A & B;           /* And gate */
Nand = !(A & B);        /* Nand gate */
Or   = A # B;           /* Or gate */
Nor  = !(A # B);        /* Nor gate */
Xor  = A $ B;           /* Exclusive Or gate */
Xnor = !(A $ B);        /* Exclusive Nor gate */
```

Figure 3. Simple Gates — Simulation Results

```
==========================================================================
                              Simulation Results
==========================================================================

                            Simple Gates Simulation

            Inverters  And      Nand     Or      Nor     Xor      Xnor
      A  B   !A  !B   A & B  !(A & B)  A # B  !(A # B)  A $ B  !(A $ B)
      -  -   -  -   ---  ----  ---  ----  ---  ----
0001: 0  0    H   H     L        H       L        H       L        H
0002: 0  1    H   L     L        H       H        L       H        L
0003: 1  0    L   H     L        H       H        L       H        L
0004: 1  1    L   L     H        L       H        L       L        H
0005: 1  X    L   X     X        X       H        L       X        X
0006: X  1    X   L     X        X       H        L       X        X
0007: 0  X    H   X     L        H       X        X       X        X
0008: X  0    X   H     L        H       X        X       X        X
0009: X  X    X   X     X        X       X        X       X        X
```

Figure 4. Simple Gates — Documentation File

```
*************************************************************************
                                    Gates
*************************************************************************

CUPL               4.0a Serial# ST-17202000
Device             g16v8s  Library DLIB-h-200-9
Created            Tue Jan 7 12:00:00 1992
Name               Gates
Partno             CA0001
Revision           00
Date               None
Designer           G. Woolhiser
Company            Logical Devices, Inc.
Assembly           None
Location           None

=========================================================================
                         Expanded Product Terms
=========================================================================

And =
    A & B

InvA =
    !A

InvB =
    !B

Nand =
    A & B

Nor =
    A
  # B

Or =
    A
  # B

Xnor =
    A & !B
  # !A & B
```

Figure 4. Simple Gates — Documentation File *(continued)*

```
Xor =
    A & !B
  # !A & B


==========================================================================
                                Symbol Table
==========================================================================

Pin Variable                                    Pterms   Max     Min
Pol   Name              Ext     Pin     Type     Used   Pterms  Level
-- ----            --     --     --    ---  ---  ---

    A                           1        V        -       -       -
    And                         14       V        1       8       1
    B                           2        V        -       -       -
    InvA                        12       V        1       8       1
    InvB                        13       V        1       8       1
    Nand                        15       V        1       8       1
    Nor                         17       V        2       8       1
    Or                          16       V        2       8       1
    Xnor                        19       V        2       8       1
    Xor                         18       V        2       8       1


LEGEND    F : field      D : default variable         M : extended node
          N : node       I : intermediate variable    T : function
          V : variable   X : extended variable        U : undefined


==========================================================================
                                  Fuse Plot
==========================================================================

Syn   02192 - Ac0   02193 x

Pin #19  02048  Pol x  02120  Ac1 x
 00000 -xx-----------------------------
 00032 x-x----------------------------
 00064 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00096 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00128 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00160 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00192 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00224 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Figure 4. Simple Gates — Documentation File *(continued)*

```
Pin #18  02049  Pol -  02121  Acl x
 00256 -xx---------------
 00288 x-x--------------
 00320 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00352 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00384 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00416 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00448 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00480 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #17  02050  Pol x  02122  Acl x
 00512 -x---------------
 00544 x----------------
 00576 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00608 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00640 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00672 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00704 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00736 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #16  02051  Pol -  02123  Acl x
 00768 -x---------------
 00800 x----------------
 00832 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00864 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00896 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00928 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00960 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00992 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #15  02052  Pol x  02124  Acl x
 01024 x-x---------------
 01056 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01088 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01120 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01152 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01184 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01216 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01248 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #14  02053  Pol -  02125  Acl x
 01280 x-x---------------
 01312 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01344 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01376 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01408 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01440 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01472 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01504 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Figure 4. Simple Gates — Documentation File *(continued)*

```
Pin #13  02054  Pol -  02126  Acl x
 01536 -x------------------------------
 01568 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01600 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01632 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01664 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01696 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01728 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01760 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #12  02055  Pol -  02127  Acl x
 01792 --x-----------------------------
 01824 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01856 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01888 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01920 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01952 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01984 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 02016 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx


LEGEND    X : fuse not blown
          - : fuse blown

===========================================================================
                                 Chip Diagram
===========================================================================


                               ______________
                              |    Gates     |
                        A x---|1           20|---x Vcc
                        B x---|2           19|---x Xnor
                          x---|3           18|---x Xor
                          x---|4           17|---x Nor
                          x---|5           16|---x Or
                          x---|6           15|---x Nand
                          x---|7           14|---x And
                          x---|8           13|---x InvB
                          x---|9           12|---x InvA
                      GND x---|10          11|---x
                              |______________|
```

Figure 5. Simple Gates — Input File for Simulation

```
Name            Gates;
Partno          CA0001;
Revision        00;
Date            None;
Designer        G. Woolhiser;
Company         Logical Devices, Inc.;
Location        None;
Assembly        None;
Device          G16V8;

/*****************************************************************/
/*                                                               */
/*      This is a example to demonstrate how ST-CUPL             */
/*      compiles simple gates.                                   */
/*                                                               */
/*****************************************************************/
/*      Target Device: G16V8                                     */
/*****************************************************************/

/* Order:  define order, polarity, and output
 * spacing of stimulus and response values */

Order: A, %2, B, %4, InvA, %3, InvB, %5, And, %8, Nand, %7, Or, %8, Nor, %7, Xor, %8, Xnor;

/* Vectors:  define stimulus and response values, with header
 *           and intermediate messages for the simulator listing. */

Vectors:

$msg “”;
$msg “                    Simple Gates Simulation”;
$msg “”;
$msg “         Inverters  And     Nand     Or     Nor    Xor     Xnor”;
$msg “     A  B   !A  !B   A & B  !(A & B)  A # B  !(A # B)  A $ B  !(A $ B)”;
$msg “     -  -   -   -    -----  -------   -----  -------   -----  -------”;

00 HHLHLHLH
01 HLLHHLHL
10 LHLHHLHL
11 LLHLHLLH
1X LXXXHLXX
X1 XLXXHLXX
0X HXLHXXXX
X0 XHLHXXXX
XX XXXXXXXX
```

Two-Bit Counter Example

In this example four different ways to implement a two-bit counter using D-type registers are shown. The flip-flops are clocked on the rising edge of the clock signal.

Figure 6 reports the source file, Figure 7 the simulation results, Figure 8 contains the documentation file and finally in Figure 9 there is the input file for the ST-CUPL™ simulator.

Figurue 6. Two-Bit Counter — Source Code

```
Name            Flops;
Partno          CA0002;
Revision        00;
Date            None;
Designer        G. Woolhiser;
Company         Logical Devices, Inc.;
Location        None;
Assembly        None;
Device          G16V8;

/****************************************************************/
/*                                                              */
/*      This example demonstrates the use of D-type flip-flops, */
/*      and flexibilty of expression with ST-CUPL. The          */
/*      following are four implementations of a two bit         */
/*      counter, which use the following timing diagram.        */
/*                ______    ______    ______    ______    ____  */
/*   clock      __|    |____|    |____|    |____|    |____|     */
/*              ___         ___________         ___________     */
/*      q0        |_________|         |_________|         |__   */
/*              ___                   _____________________     */
/*      q1        |___________________|                   |__   */
/*                                                              */
/****************************************************************/
/*      Target Device: G16V8                                    */
/****************************************************************/

Pin 1 =  clock;
Pin 2 =  reset;
Pin 11 = !enable;

/* Outputs:  define outputs and output active levels */

Pin 19 = qa0;
Pin 18 = qa1;
Pin 17 = qb0;
Pin 16 = qb1;
Pin 15 = qc0;
```

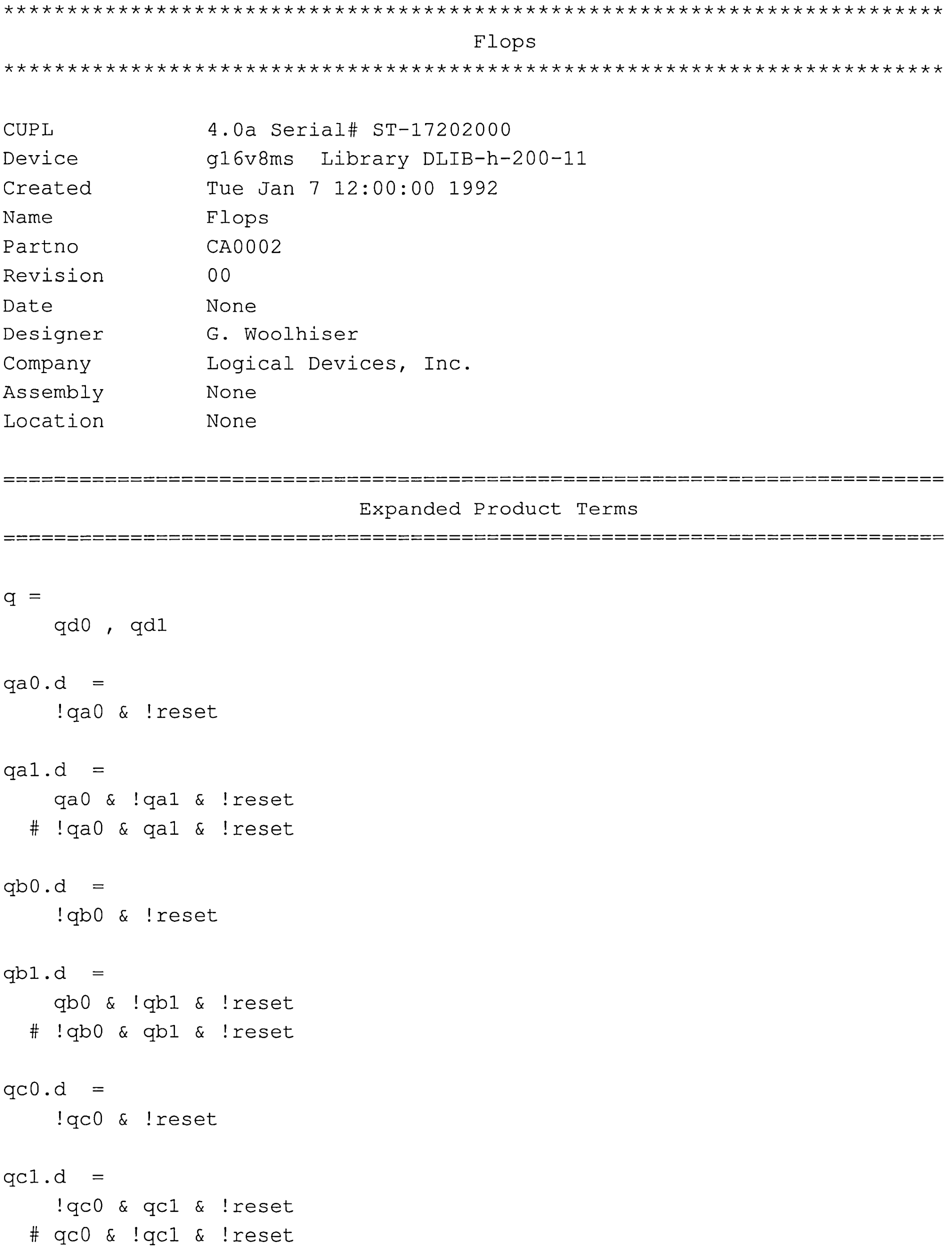
Figure 6. Two-Bit Counter — Source Code *(continued)*

```
Pin 14 = qc1;
Pin 13 = qd0;
Pin 12 = qd1;

/* Logic:  examples of two-bit counters using D-type flip-flops */

/* Two-bit counter example no. 1 */
/* Using software emulated exclusive or's */

qa0.d = !reset & !qa0;
qa1.d = !reset & (qa1 $ qa0);

/* Two-bit counter example no. 2 */
/* Using expanded exclusive or's */

qb0.d = !reset & (!qb0 & !qb1
                # !qb0 & qb1);
qb1.d = !reset & (!qb0 & qb1
                # qb0 & !qb1);

/* Two-bit counter example no. 3 */
/* Using bit fields on the right hand side of the equals sign */

field state = [qc1,qc0];

qc0.d = !reset & (state:0 # state:2);
qc1.d = !reset & (state:1 # state:2);

/* Two-bit counter example no. 4 */
/* Using bit fields on the left hand side of the equals sign */

field q = [qd0,qd1];

q.d = !reset & ([!qd0,qd1] & [!qd1,!qd0]
              # [!qd0,!qd1] & [qd1,qd0]);
```

Figure 7. Two-Bit Counter — Simulation Results

```
==========================================================================
                              Simulation Results
==========================================================================

                             counterA   counterB   counterC   counterD
    clock   !enable   reset   q0  q1     q0  q1     q0  q1     q0  q1
    ---   ----   ---  ----   ----   ----   ----

Reset to Zero

0001: C        0        1      L   L      L   L      L   L      L   L
0002: C        0        0      H   L      H   L      H   L      H   L
0003: C        0        0      L   H      L   H      L   H      L   H
0004: C        0        0      H   H      H   H      H   H      H   H
0005: C        0        0      L   L      L   L      L   L      L   L

Register Preload

0006: P        0        X      1   1      1   0      0   1      0   0
0007: 0        0        0      H   H      H   L      L   H      L   L
0008: C        0        0      L   L      L   H      H   H      H   L
0009: C        0        0      H   L      H   H      L   L      L   H
0010: C        0        0      L   H      L   L      H   L      H   H
0011: C        0        0      H   H      H   L      L   H      L   L
```

Figure 8. Two-Bit Counter — Documentation File



```
**************************************************************************
                                     Flops
**************************************************************************

CUPL            4.0a Serial# ST-17202000
Device          g16v8ms  Library DLIB-h-200-11
Created         Tue Jan 7 12:00:00 1992
Name            Flops
Partno          CA0002
Revision        00
Date            None
Designer        G. Woolhiser
Company         Logical Devices, Inc.
Assembly        None
Location        None

==========================================================================
                            Expanded Product Terms
==========================================================================

q =
    qd0 , qd1

qa0.d  =
    !qa0 & !reset

qa1.d  =
    qa0 & !qa1 & !reset
  # !qa0 & qa1 & !reset

qb0.d  =
    !qb0 & !reset

qb1.d  =
    qb0 & !qb1 & !reset
  # !qb0 & qb1 & !reset

qc0.d  =
    !qc0 & !reset

qc1.d  =
    !qc0 & qc1 & !reset
  # qc0 & !qc1 & !reset
```

Figure 8. Two-Bit Counter — Documentation File *(continued)*

```
qd0.d  =
    !qd0 & !reset

qd1.d  =
    qd0 & !qd1 & !reset
  # !qd0 & qd1 & !reset

state =
    qc1 , qc0


==========================================================================
                                 Symbol Table
==========================================================================

Pin Variable                                    Pterms   Max     Min
Pol   Name              Ext     Pin     Type     Used   Pterms  Level
-- ----            --     --     --    ---  ---  ---

    clock                           1        V        -       -       -
 !  enable                          11       V        -       -       -
    q                               0        F        -       -       -
    qa0                             19       V        -       -       -
    qa0                 d       19       X        1       8       1
    qa1                             18       V        -       -       -
    qa1                 d       18       X        2       8       1
    qb0                             17       V        -       -       -
    qb0                 d       17       X        1       8       1
    qb1                             16       V        -       -       -
    qb1                 d       16       X        2       8       1
    qc0                             15       V        -       -       -
    qc0                 d       15       X        1       8       1
    qc1                             14       V        -       -       -
    qc1                 d       14       X        2       8       1
    qd0                             13       V        -       -       -
    qd0                 d       13       X        1       8       1
    qd1                             12       V        -       -       -
    qd1                 d       12       X        2       8       1
    reset                           2        V        -       -       -
    state                           0        F        -       -       -

LEGEND    F : field      D : default variable        M : extended node
          N : node       I : intermediate variable   T : function
          V : variable   X : extended variable       U : undefined
```

Figure 8. Two-Bit Counter — Documentation File *(continued)*

```
===========================================================================
                                   Fuse Plot
===========================================================================

Syn   02192 x Ac0   02193 -

Pin #19  02048  Pol -  02120  Ac1 x
 00000 -x-x--------------
 00032 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00064 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00096 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00128 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00160 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00192 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00224 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #18  02049  Pol -  02121  Ac1 x
 00256 -xx--x------------
 00288 -x-x-x-------------
 00320 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00352 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00384 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00416 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00448 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00480 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #17  02050  Pol -  02122  Ac1 x
 00512 -x-----x----------
 00544 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00576 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00608 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00640 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00672 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00704 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00736 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #16  02051  Pol -  02123  Ac1 x
 00768 -x----x--x--------
 00800 -x-----x-x---------
 00832 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00864 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00896 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00928 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00960 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00992 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #15  02052  Pol -  02124  Ac1 x
 01024 -x---------x------
 01056 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Figure 8. Two-Bit Counter — Documentation File *(continued)*

```
  01088 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  01120 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  01152 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  01184 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  01216 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  01248 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #14  02053  Pol -  02125  Acl x
  01280 -x---------x-x-----
  01312 -x--------x--x----
  01344 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  01376 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  01408 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  01440 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  01472 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  01504 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #13  02054  Pol -  02126  Acl x
  01536 -x-------------x--
  01568 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  01600 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  01632 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  01664 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  01696 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  01728 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  01760 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #12  02055  Pol -  02127  Acl x
  01792 -x------------x--x
  01824 -x-------------x-x-
  01856 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  01888 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  01920 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  01952 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  01984 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  02016 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx


 LEGEND     X : fuse not blown
            - : fuse blown
```

Figure 8. Two-Bit Counter — Documentation File *(continued)*

```
===========================================================================
                                 Chip Diagram
===========================================================================

                             ______________
                            |     Flops    |
                   clock x---|1          20|---x Vcc
                   reset x---|2          19|---x qa0
                         x---|3          18|---x qa1
                         x---|4          17|---x qb0
                         x---|5          16|---x qb1
                         x---|6          15|---x qc0
                         x---|7          14|---x qc1
                         x---|8          13|---x qd0
                         x---|9          12|---x qd1
                     GND x---|10         11|---x !enable
                            |______________|
```

Figure 9. Two-Bit Counter — Input File for Simulation

```
Name            Flops;
Partno          CA0002;
Revision        00;
Date            None;
Designer        G. Woolhiser;
Company         Logical Devices, Inc.;
Location        None;
Assembly        None;
Device          G16V8;

/****************************************************************/
/*                                                              */
/*      This example demonstrates the use of D-type flip-flops, */
/*      and flexibilty of expression with ST-CUPL. The          */
/*      following are four implementations of a two bit         */
/*      counter, which use the following timing diagram.        */
/*                 ____      ____      ____      ____      ___  */
/*   clock      __|    |____|    |____|    |____|    |____|     */
/*              ___          __________          __________     */
/*      q0        |_________|          |_________|         |__   */
/*              ___                    ____________________     */
/*      q1        |___________________|                    |__   */
/*                                                              */
/****************************************************************/
/*      Target Devices: G16V8                                   */
/****************************************************************/
```

Figure 9. Two-Bit Counter — Input File for Simulation *(continued)*

```
Order:  clock, %8, !enable, %8, reset, %6, qa0, %3, qa1, %6, qb0,
        %3, qb1, %6, qc0, %3, qc1, %6, qd0, %3, qd1;

Vectors:

$msg “”;
$msg “                          counterA   counterB   counterC   counterD”;
$msg “   clock   !enable   reset   q0  q1    q0  q1    q0  q1    q0  q1 ”;
$msg “   ___   ____   ___  ____   ____   ____   ____”;
$msg “”;
$msg “Reset to Zero”;
$msg “”;
c01 LLLLLLLL
$repeat 4;
c00 ********
$msg “”;
$msg “Register Preload”;
$msg “”;
p0x 11100100
000 ********
$repeat 4;
c00 ********
```

Decade Up/Down Counter Example

This example describes a four-bit decade up/down counter with synchronous clear capacity and asynchronous carry output for cascading multiple devices; the state diagram is reported in Figure 10.

The input signal *dir* determines the direction of the counter: when *dir* is high the counter goes down, while when *dir* is low it goes up.

The bit field *mode* is used to combine the *clr* and *dir* inputs; the following equations

up = mode:0;
down = mode:1;
clear = mode:[2..3];

define three different variables:

- *up* is true when both *dir* and *clr* are low;
- *down* is true when *dir* is high and *clr* is low;
- *clear* is true when *clr* is high regardless of the value of *dir*.

Then these three signals are used in the definition of the state machine transitions: the code obtained in this way is certainly easily readable and maintainable.

The source file is contained in Figure 11, Figure 12 shows the simulation results, while Figure 13 and Figure 14 hold respectively the documentation file and the input file for the simulator.

Figure 10. Decade Up/Down Counter — State Diagram

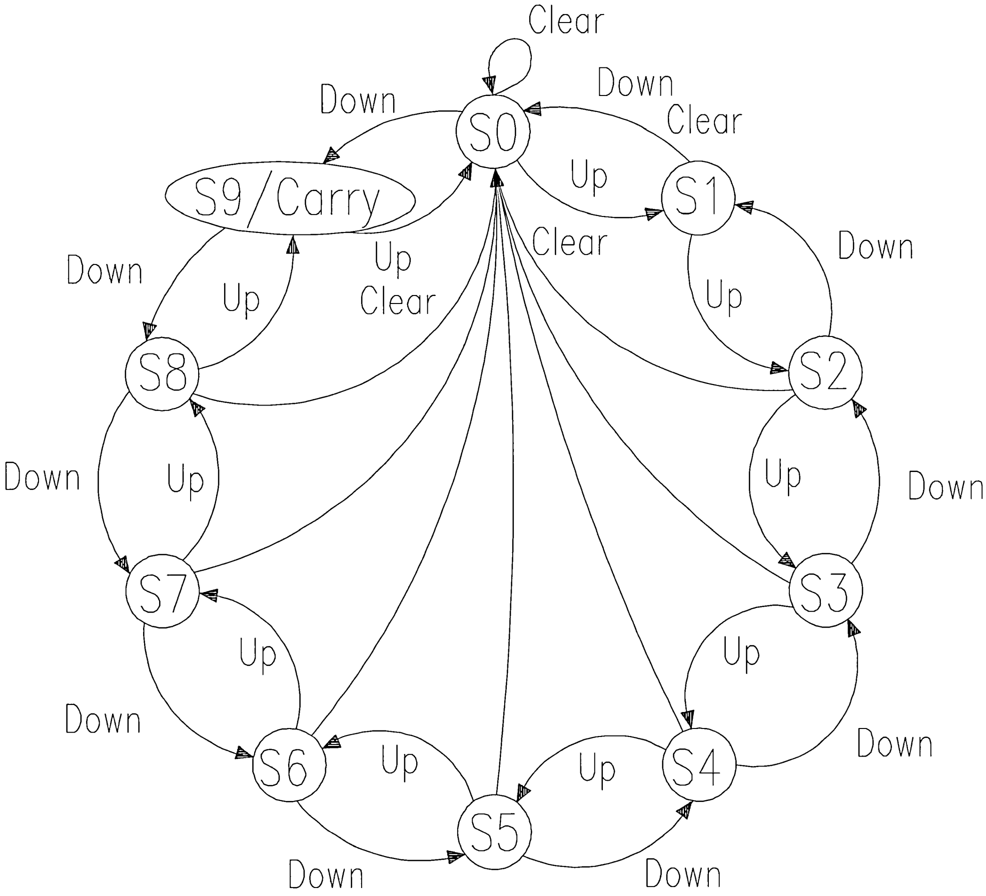


Figure 11. Decade Up/Down Counter — Source Code

```
Name      Count10;
Partno    CA0018;
Date      None;
Revision  00;
Designer  Kahl;
Company   Logical Devices, Inc.;
Assembly  None;
Location  None;
Device    G16V8;

/****************************************************************/
/*                                                              */
/* Decade Counter                                               */
/*                                                              */
/* This is a 4-bit up/down decade counter with synchronous      */
/* clear capability.  An asynchronous ripple carry output is    */
/* provided for cascading multiple devices.                     */
/* ST-CUPL state machine syntax is used.                        */
/****************************************************************/
/* Allowable Target Device Types :  GAL16V8                     */
/****************************************************************/

/**  Inputs  **/

pin 1        = clk;             /* Counter clock                */
pin 2        = clr;             /* Counter clear input          */
pin 3        = dir;             /* Counter direction input      */
pin 11       = !oe;             /* Register output enable       */

/**  Outputs  **/

pin [14..17] = [Q3..0];         /* Counter outputs              */
pin 18 = carry;                 /* Ripple carry out             */

/** Declarations and Intermediate Variable Definitions **/

field count = [Q3..0];          /* declare counter bit field */
$define S0 'b'0000              /* define counter states */
$define S1 'b'0001
$define S2 'b'0010
$define S3 'b'0011
$define S4 'b'0100
$define S5 'b'0101
$define S6 'b'0110
$define S7 'b'0111
```

Figure 11. Decade Up/Down Counter — Source Code *(continued)*

```
$define S8 'b'1000
$define S9 'b'1001

field mode = [clr,dir];         /* declare mode control field */
up = mode:0;                    /* define count up mode */
down = mode:1;                  /* define count down mode */
clear = mode:[2..3];            /* define count clear mode */

/** Logic Equations **/

sequence count {                        /* free running counter */

present S0      if up           next S1;
                if down         next S9;
                if clear        next S0;
                if down         out carry;
present S1      if up           next S2;
                if down         next S0;
                if clear        next S0;
present S2      if up           next S3;
                if down         next S1;
                if clear        next S0;
present S3      if up           next S4;
                if down         next S2;
                if clear        next S0;
present S4      if up           next S5;
                if down         next S3;
                if clear        next S0;
present S5      if up           next S6;
                if down         next S4;
                if clear        next S0;
present S6      if up           next S7;
                if down         next S5;
                if clear        next S0;
present S7      if up           next S8;
                if down         next S6;
                if clear        next S0;
present S8      if up           next S9;
                if down         next S7;
                if clear        next S0;
present S9      if up           next S0;
                if down         next S8;
                if clear        next S0;
                if up           out carry;     /* assert carry output */
}
```

Figure 12. Decade Up/Down Counter — Simulation Results

```
==========================================================================
                              Simulation Results
==========================================================================

      clk  clr  dir  !oe  Q3  Q2  Q1  Q0  carry
      ______________________

0001:  C    1    0   0    L   L   L   L     L
0002:  C    0    0   0    L   L   L   H     L
0003:  C    0    0   0    L   L   H   L     L
0004:  C    0    0   0    L   L   H   H     L
0005:  C    0    0   0    L   H   L   L     L
0006:  C    0    0   0    L   H   L   H     L
0007:  C    0    0   0    L   H   H   L     L
0008:  C    0    0   0    L   H   H   H     L
0009:  C    0    0   0    H   L   L   L     L
0010:  C    0    0   0    H   L   L   H     H
0011:  C    0    0   0    L   L   L   L     L
0012:  C    0    1   0    H   L   L   H     L
0013:  C    0    1   0    H   L   L   L     L
0014:  C    0    1   0    L   H   H   H     L
0015:  C    0    1   0    L   H   H   L     L
0016:  C    0    1   0    L   H   L   H     L
0017:  C    0    1   0    L   H   L   L     L
0018:  C    0    1   0    L   L   H   H     L
0019:  C    0    1   0    L   L   H   L     L
0020:  C    0    1   0    L   L   L   H     L
0021:  C    0    1   0    L   L   L   L     H
0022:  C    0    0   1    Z   Z   Z   Z     L
0023:  C    0    0   0    L   L   H   L     L
0024:  C    1    0   0    L   L   L   L     L
```

Figure 13. Decade Up/Down Counter — Documentation File

```
*************************************************************************
                                   Count10
*************************************************************************

CUPL                4.0a Serial# ST-17202000
Device              g16v8ms  Library DLIB-h-200-11
Created             Tue Jan 7 12:00:00 1992
Name                Count10
Partno              CA0018
Revision            00
Date                None
Designer            Kahl
Company             Logical Devices, Inc.
Assembly            None
Location            None

=========================================================================
                            Expanded Product Terms
=========================================================================

Q0.d  =
    !Q0 & !Q1 & !Q2 & Q3 & !clr
  # !Q0 & !Q3 & !clr

Q1.d  =
    !Q0 & !Q1 & !Q2 & Q3 & !clr & dir
  # Q0 & !Q1 & !Q3 & !clr & !dir
  # !Q0 & Q1 & !Q3 & !clr & !dir
  # Q0 & Q1 & !Q3 & !clr & dir
  # !Q0 & !Q1 & Q2 & !Q3 & !clr & dir

Q2.d  =
    !Q0 & !Q1 & !Q2 & Q3 & !clr & dir
  # Q0 & Q1 & !Q2 & !Q3 & !clr & !dir
  # !Q1 & Q2 & !Q3 & !clr & !dir
  # Q0 & Q2 & !Q3 & !clr & dir
  # !Q0 & Q1 & Q2 & !Q3 & !clr

Q3.d  =
    Q0 & !Q1 & !Q2 & Q3 & !clr & dir
  # !Q0 & !Q1 & !Q2 & !Q3 & !clr & dir
  # Q0 & Q1 & Q2 & !Q3 & !clr & !dir
  # !Q0 & !Q1 & !Q2 & Q3 & !clr & !dir
```

Figure 13. Decade Up/Down Counter — Documentation File *(continued)*

```
carry =
    !Q0 & !Q1 & !Q2 & !Q3 & !clr & dir
  # Q0 & !Q1 & !Q2 & Q3 & !clr & !dir

clear =
    clr

count =
    Q3 , Q2 , Q1 , Q0

down =
    !clr & dir

mode =
    clr , dir

up =
    !clr & !dir

carry.oe  =
    1

===========================================================================
                                 Symbol Table
===========================================================================

Pin Variable                                    Pterms   Max     Min
Pol   Name              Ext     Pin     Type     Used   Pterms  Level
-- -----           --     --     --     ---  ---  ---

    Q0                          17       V        -       -       -
    Q0                  d       17       X        2       8       1
    Q1                          16       V        -       -       -
    Q1                  d       16       X        5       8       1
    Q2                          15       V        -       -       -
    Q2                  d       15       X        5       8       1
    Q3                          14       V        -       -       -
    Q3                  d       14       X        4       8       1
    carry                       18       V        2       7       1
    clear                        0       I        1       -       -
    clk                          1       V        -       -       -
    clr                          2       V        -       -       -
    count                        0       F        -       -       -
    dir                          3       V        -       -       -
    down                         0       I        1       -       -
```

Figure 13. Decade Up/Down Counter — Documentation File *(continued)*

```
      mode                         0       F       -       -       -
   !  oe                           11      V       -       -       -
      up                           0       I       1       -       -
      carry               oe       18      D       1       1       0


 LEGEND     F : field      D : default variable       M : extended node
            N : node       I : intermediate variable  T : function
            V : variable   X : extended variable      U : undefined

 ==========================================================================
                                     Fuse Plot
 ==========================================================================


 Syn   02192 x Ac0   02193 -

 Pin #19  02048  Pol x  02120  Ac1 -
  00000 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  00032 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  00064 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  00096 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  00128 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  00160 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  00192 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  00224 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 Pin #18  02049  Pol -  02121  Ac1 -
  00256 ----------------
  00288 -x-x---x--x--x--x----
  00320 -x--x--x--x--x-x-----
  00352 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  00384 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  00416 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  00448 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  00480 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 Pin #17  02050  Pol -  02122  Ac1 x
  00512 -x-----x--x--x-x-----
  00544 -x-----x------x----
  00576 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  00608 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  00640 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  00672 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  00704 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  00736 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Figure 13. Decade Up/Down Counter — Documentation File *(continued)*

```
Pin #16  02051  Pol -  02123  Acl x
 00768 -x-x---x--x--x-x-----
 00800 -x--x--x--x----x----
 00832 -x--x---x-x----x----
 00864 -x-x---x--x----x----
 00896 -x-x---x--x-x--x----
 00928 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00960 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00992 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #15  02052  Pol -  02124  Acl x
 01024 -x-x---x--x--x-x-----
 01056 -x--x--x--x--x--x----
 01088 -x--x-----x-x--x----
 01120 -x-x---x----x--x----
 01152 -x-----x-x--x--x----
 01184 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01216 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01248 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #14  02053  Pol -  02125  Acl x
 01280 -x-x---x--x--x-x-----
 01312 -x-x---x--x--x--x----
 01344 -x--x--x--x--x--x----
 01376 -x--x---x--x--x-x-----
 01408 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01440 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01472 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01504 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #13  02054  Pol x  02126  Acl -
 01536 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01568 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01600 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01632 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01664 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01696 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01728 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01760 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #12  02055  Pol x  02127  Acl -
 01792 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01824 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01856 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01888 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01920 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01952 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Figure 13. Decade Up/Down Counter — Documentation File *(continued)*

```
  01984 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  02016 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx


LEGEND    X : fuse not blown
          - : fuse blown

==========================================================================
                                 Chip Diagram
==========================================================================


                               ______________
                              |   Count10    |
                      clk x---|1           20|---x Vcc
                      clr x---|2           19|---x
                      dir x---|3           18|---x carry
                          x---|4           17|---x Q0
                          x---|5           16|---x Q1
                          x---|6           15|---x Q2
                          x---|7           14|---x Q3
                          x---|8           13|---x
                          x---|9           12|---x
                      GND x---|10          11|---x !oe
                              |______________|
```

Figure 14. Decade Up/Down Counter — Input File for Simulation

```
Name      Count10;
Partno    CA0018;
Date      None;
Revision  00;
Designer  Kahl;
Company   Logical Devices, Inc.;
Assembly  None;
Location  None;
Device    G16V8;

/*************************************************************/
/*                                                           */
/* Decade Counter                                            */
/*                                                           */
/*************************************************************/

ORDER:  %1,clk,%4,clr,%4,dir,%4,!oe,%3,Q3,%3,Q2,%3,Q1,%3,Q0,%5,carry;
```

Figure 14. Decade Up/Down Counter — Input File for Simulation *(continued)*

```
VECTORS:

$msg "";
$msg "      clk  clr  dir  !oe  Q3  Q2  Q1  Q0  carry";
$msg "      ———————————————————-";
$msg "";

        C 100  LLLL L        /* synchronous clear to state 0 */
        C 000  LLLH L        /* count up to state 1          */
        C 000  LLHL L        /* count up to state 2          */
        C 000  LLHH L        /* count up to state 3          */
        C 000  LHLL L        /* count up to state 4          */
        C 000  LHLH L        /* count up to state 5          */
        C 000  LHHL L        /* count up to state 6          */
        C 000  LHHH L        /* count up to state 7          */
        C 000  HLLL L        /* count up to state 8          */
        C 000  HLLH H        /* count up to state 9 - carry  */
        C 000  LLLL L        /* count up to state 0          */
        C 010  HLLH L        /* count down to state 9        */
        C 010  HLLL L        /* count down to state 8        */
        C 010  LHHH L        /* count down to state 7        */
        C 010  LHHL L        /* count down to state 6        */
        C 010  LHLH L        /* count down to state 5        */
        C 010  LHLL L        /* count down to state 4        */
        C 010  LLHH L        /* count down to state 3        */
        C 010  LLHL L        /* count down to state 2        */
        C 010  LLLH L        /* count down to state 1        */
        C 010  LLLL H        /* count down to state 0 - carry*/
        C 001  ZZZZ L        /* test tri-state               */
        C 000  LLHL L        /* count up to state 2          */
        C 100  LLLL L        /* synchronous clear to state 0 */
```

Seven-Segment Display Decoder Example

This is an example of implementing a decoder that accepts as input an hexadecimal digit and generates seven signals able to control a seven-segment display driver. The design incorporates both a ripple-blanking input to inhibit the display of leading zeros and a ripple-blanking output for easy cascading of digits.

The signals that drive the display are grouped in the bit field *segment*; the equation for *segment* is written as a logical sum of several lines in the form:

> [$<st_a>$, $<st_b>$, $<st_c>$, $<st_d>$, $<st_e>$, $<st_f>$,
> $<st_g>$] & data:$<n>$

where $<st_k>$ is the status of the corresponding segment (i.e. *ON* or *OFF*, respectively binary 1 or 0), while *<n>* is a hexadecimal digit; the display of all possible digits is shown in Figure 15.

When *data* is equal to *<n>*, then *segment* assumes the value of the bits contained in the list. The expression for zero is ANDed with the negation of the input signal *rbi*: zeros are displayed only if *rbi* is low.

The ripple-blanking output signal is generated by the equation:

> rbo = rbi & data:0;

so *rbo* is high only if *rbi* is high (i.e. the display of leading zeros is inhibited) and the digit is a zero.

Figures 16 through 19 contain source file, simulation results, documentation file and the input file for the simulator.

Figure 15. Seven-Segment Display Decoder — Display of all possible Digits

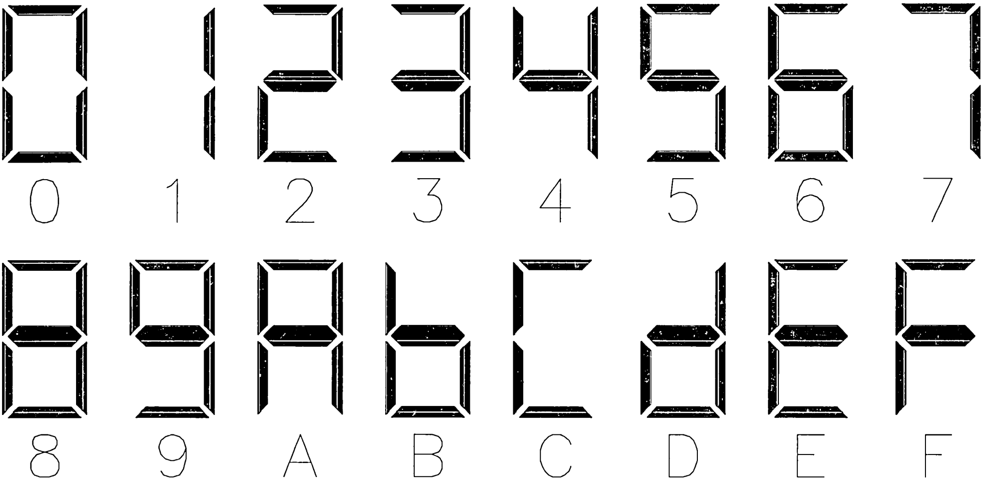


Figure 16. Seven-Segment Display Decoder — Source Code

```
Name            Hexdisp;
Partno          CA0007;
Date            None;
Revision        00;
Designer        T. Kahl;
Company         Logical Devices, Inc.;
Assembly        None;
Location        None;
Device          G16V8;

/****************************************************************/
/* This is a hexadecimal-to-seven-segment          a            */
/* decoder capable of driving common-annode      ---          */
/* LEDs.  It incorporates both a ripple-        |     |         */
/* blanking input (to inhibit displaying       f|     |b        */
/* leading zeroes) and a ripple-blanking        |  g  |         */
/* output to allow for easy cascading of         ---          */
/* digits.                                      |     |         */
/*                                             e|     |c        */
/*                                              |     |         */
/*                                               ---          */
/*                                                 d            */
/****************************************************************/
/* Allowable Target Device Types: G16V8                         */
/****************************************************************/

/** Inputs **/

pin [2..5] = [D0..3];  /* Data input lines to display           */
pin 6  = !rbi;         /* Ripple blanking input                 */

/** Outputs **/

pin [12..18] = ![a,b,c,d,e,f,g];  /* Segment output lines          */
pin 19       = !rbo;              /* Ripple blanking output        */

/** Declarations and Intermediate Variable Definitions **/

field data = [D3..0];             /* Hexadecimal input field  */
field segment = [a,b,c,d,e,f,g];  /* Display segment field    */

$define ON  'b'1        /* Segment LIT  when logically "ON"     */
$define OFF 'b'0        /* Segment DARK when logically "OFF"    */
```

Figure 16. Seven-Segment Display Decoder — Source Code *(continued)*

```
/** Logic Equations **/

               /*  a    b    c    d    e    f    g  */
segment    =
/* 0 */         [ ON,  ON,  ON,  ON,  ON,  ON, OFF] & data:0 & !rbi
/* 1 */     #  [OFF,  ON,  ON, OFF, OFF, OFF, OFF] & data:1
/* 2 */     #  [ ON,  ON, OFF,  ON,  ON, OFF,  ON] & data:2
/* 3 */     #  [ ON,  ON,  ON,  ON, OFF, OFF,  ON] & data:3
/* 4 */     #  [OFF,  ON,  ON, OFF, OFF,  ON,  ON] & data:4
/* 5 */     #  [ ON, OFF,  ON,  ON, OFF,  ON,  ON] & data:5
/* 6 */     #  [ ON, OFF,  ON,  ON,  ON,  ON,  ON] & data:6
/* 7 */     #  [ ON,  ON,  ON, OFF, OFF, OFF, OFF] & data:7
/* 8 */     #  [ ON,  ON,  ON,  ON,  ON,  ON,  ON] & data:8
/* 9 */     #  [ ON,  ON,  ON,  ON, OFF,  ON,  ON] & data:9
/* A */     #  [ ON,  ON,  ON, OFF,  ON,  ON,  ON] & data:A
/* B */     #  [OFF, OFF,  ON,  ON,  ON,  ON,  ON] & data:B
/* C */     #  [ ON, OFF, OFF,  ON,  ON,  ON, OFF] & data:C
/* D */     #  [OFF,  ON,  ON,  ON,  ON, OFF,  ON] & data:D
/* E */     #  [ ON, OFF, OFF,  ON,  ON,  ON,  ON] & data:E
/* F */     #  [ ON, OFF, OFF, OFF,  ON,  ON,  ON] & data:F;

rbo = rbi & data:0;
```

Figure 17. Seven-Segment Display Decoder — Simulation Results

```
===========================================================================
                              Simulation Results
===========================================================================


    !rbi D3 D2 D1 D0  !a !b !c !d !e !f !g  !rbo


0001: 0  0  0  0  0   H  H  H  H  H  H  H    L
0002: 1  0  0  0  0   L  L  L  L  L  L  H    H
0003: X  0  0  0  1   H  L  L  H  H  H  H    H
0004: X  0  0  1  0   L  L  H  L  L  H  L    H
0005: X  0  0  1  1   L  L  L  L  H  H  L    H
0006: X  0  1  0  0   H  L  L  H  H  L  L    H
0007: X  0  1  0  1   L  H  L  L  H  L  L    H
0008: X  0  1  1  0   L  H  L  L  L  L  L    H
0009: X  0  1  1  1   L  L  L  H  H  H  H    H
0010: X  1  0  0  0   L  L  L  L  L  L  L    H
0011: X  1  0  0  1   L  L  L  L  H  L  L    H
0012: X  1  0  1  0   L  L  L  H  L  L  L    H
0013: X  1  0  1  1   H  H  L  L  L  L  L    H
0014: X  1  1  0  0   L  H  H  L  L  L  H    H
0015: X  1  1  0  1   H  L  L  L  L  H  L    H
0016: X  1  1  1  0   L  H  H  L  L  L  L    H
0017: X  1  1  1  1   L  H  H  H  L  L  L    H
```

Figure 18. Seven-Segment Display Decoder — Documentation File

```
*************************************************************************
                                    Hexdisp
*************************************************************************

CUPL            4.0a Serial# ST-17202000
Device          g16v8s  Library DLIB-h-200-9
Created         Thu Jan  7 12:00:00 1992
Name            Hexdisp
Partno          CA0007
Revision        00
Date            None
Designer        T. Kahl
Company         Logical Devices, Inc.
Assembly        None
Location        None

=========================================================================
                            Expanded Product Terms
=========================================================================

a =
    !D0 & !D1 & !D2 & !D3 & !rbi
  # D0 & D1 & D2 & D3
  # D1 & !D2 & !D3
  # D0 & D2 & !D3
  # !D0 & D1 & D2
  # !D1 & !D2 & D3
  # !D0 & D1 & !D2 & D3
  # !D0 & !D1 & D2 & D3

b =
    !D0 & !D1 & !D2 & !D3 & !rbi
  # D0 & !D2 & !D3
  # !D0 & D1 & !D2
  # !D0 & !D1 & D2 & !D3
  # D0 & !D1 & D2 & D3
  # D0 & D1 & D2 & !D3
  # !D1 & !D2 & D3

c =
    !D0 & !D1 & !D2 & !D3 & !rbi
  # D0 & !D2 & !D3
  # D0 & !D1 & D2 & D3
  # !D2 & D3
  # D2 & !D3
```

Figure 18. Seven-Segment Display Decoder — Documentation File *(continued)*

```
d =
    !D0 & !D1 & !D2 & !D3 & !rbi
  # !D0 & !D1 & D2 & D3
  # D1 & !D2 & !D3
  # D0 & !D1 & D2
  # !D0 & D1 & D2
  # !D1 & !D2 & D3
  # D0 & D1 & !D2 & D3

data =
    D3 , D2 , D1 , D0

e =
    !D0 & !D1 & !D2 & !D3 & !rbi
  # !D0 & D1 & D2 & D3
  # !D0 & D1 & !D3
  # !D0 & !D2 & D3
  # D0 & D1 & D3
  # !D1 & D2 & D3

f =
    !D0 & !D1 & !D2 & !D3 & !rbi
  # D0 & D1 & D2 & D3
  # !D1 & D2 & !D3
  # !D0 & D1 & D2
  # !D0 & !D1 & D2 & D3
  # !D2 & D3

g =
    D0 & D2 & D3
  # D1 & !D2
  # !D1 & D2 & !D3
  # !D0 & D1 & D2
  # !D1 & !D2 & D3

rbo =
    !D0 & !D1 & !D2 & !D3 & rbi

segment =
    a , b , c , d , e , f , g
```

Figure 18. Seven-Segment Display Decoder — Documentation File *(continued)*

```
==========================================================================
                                 Symbol Table
==========================================================================

Pin Variable                                    Pterms   Max     Min
Pol   Name              Ext     Pin     Type     Used    Pterms  Level
-- -----           --     --     --    ---  ---  ---


    D0                          2        V        -       -       -
    D1                          3        V        -       -       -
    D2                          4        V        -       -       -
    D3                          5        V        -       -       -
 !  a                           12       V        8       8       1
 !  b                           13       V        7       8       1
 !  c                           14       V        5       8       1
 !  d                           15       V        7       8       1
    data                        0        F        -       -       -
 !  e                           16       V        6       8       1
 !  f                           17       V        6       8       1
 !  g                           18       V        5       8       1
 !  rbi                         6        V        -       -       -
 !  rbo                         19       V        1       8       1
    segment                     0        F        -       -       -


LEGEND    F : field      D : default variable        M : extended node
          N : node       I : intermediate variable   T : function
          V : variable   X : extended variable       U : undefined


==========================================================================
                                   Fuse Plot
==========================================================================


Syn   02192 - Ac0   02193 x

Pin #19  02048  Pol x  02120  Ac1 x
 00000 -x--x--x--x--x------------------
 00032 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00064 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00096 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00128 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00160 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00192 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00224 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Figure 18. Seven-Segment Display Decoder — Documentation File *(continued)*

```
Pin #18  02049  Pol x  02121  Acl x
 00256 x——-x--x————-
 00288 —x—x————
 00320 —-x-x—x————
 00352 -x-x--x—————-
 00384 —-x--x-x————-
 00416 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00448 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00480 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #17  02050  Pol x  02122  Acl x
 00512 -x--x--x--x-x———-
 00544 x--x--x--x————-
 00576 —-x-x—x————
 00608 -x-x--x—————-
 00640 -x--x-x--x————-
 00672 ——-x-x—————-
 00704 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00736 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #16  02051  Pol x  02123  Acl x
 00768 -x--x--x--x-x———-
 00800 -x-x--x--x————-
 00832 -x-x——x————
 00864 -x——-x-x————-
 00896 x--x——x————-
 00928 —-x-x--x————-
 00960 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00992 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #15  02052  Pol x  02124  Acl x
 01024 -x--x--x--x-x———-
 01056 -x--x-x--x————-
 01088 —x—x--x————
 01120 x—x-x—————-
 01152 -x-x--x—————-
 01184 —-x--x-x————-
 01216 x--x—x-x————-
 01248 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #14  02053  Pol x  02125  Acl x
 01280 -x--x--x--x-x———-
 01312 x——x--x————
 01344 x—x-x--x————-
 01376 ——-x-x—————-
 01408 ——x—x————
 01440 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01472 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01504 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Figure 18. Seven-Segment Display Decoder — Documentation File *(continued)*

```
Pin #13  02054  Pol x  02126  Acl x
 01536 -x--x--x--x-x--------
 01568 x----x--x---------
 01600 -x-x--x-----------
 01632 -x--x-x--x---------
 01664 x--x-x--x----------
 01696 x--x--x--x---------
 01728 ---x--x-x----------
 01760 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #12  02055  Pol x  02127  Acl x
 01792 -x--x--x--x-x--------
 01824 x--x--x--x----------
 01856 --x--x--x---------
 01888 x----x--x---------
 01920 -x-x--x------------
 01952 ---x--x-x----------
 01984 -x-x--x-x----------
 02016 -x--x-x--x----------


LEGEND    X : fuse not blown
          - : fuse blown

==========================================================================
                                 Chip Diagram
==========================================================================


                            ______________
                           |   Hexdisp    |
                       x---|1          20|---x Vcc
                    D0 x---|2          19|---x !rbo
                    D1 x---|3          18|---x !g
                    D2 x---|4          17|---x !f
                    D3 x---|5          16|---x !e
                  !rbi x---|6          15|---x !d
                       x---|7          14|---x !c
                       x---|8          13|---x !b
                       x---|9          12|---x !a
                   GND x---|10         11|---x
                           |______________|
```

Figure 19. Seven-Segment Display Decoder — Input File for Simulation

```
Name            Hexdisp;
Partno          CA0007;
Date            None;
Revision        00;
Designer        T. Kahl;
Company         Logical Devices, Inc.;
Assembly        None;
Location        None;
Device          G16V8;

/*****************************************************************/
/* Allowable Target Device Types: G16V8                          */
/*****************************************************************/

ORDER:
!rbi,%2,D3,%2,D2,%2,D1,%2,D0,%3,!a,%2,!b,%2,!c,%2,!d,%2,!e,%2,!f,%2,!g,%4,!rbo;

VECTORS:

$msg"";
$msg"    !rbi D3 D2 D1 D0  !a !b !c !d !e !f !g  !rbo";
$msg"    ——————————————————————";
$msg"";
     0  0000  HHHHHHH   L     /* ripple blanking input, no segments lit */
     1  0000  LLLLLLH   H     /* display “0" */
     X  0001  HLLHHHH   H     /* display “1" */
     X  0010  LLHLLHL   H     /* display “2" */
     X  0011  LLLLHHL   H     /* display “3" */
     X  0100  HLLHHLL   H     /* display “4" */
     X  0101  LHLLHLL   H     /* display “5" */
     X  0110  LHLLLLL   H     /* display “6" */
     X  0111  LLLHHHH   H     /* display “7" */
     X  1000  LLLLLLL   H     /* display “8" */
     X  1001  LLLLHLL   H     /* display “9" */
     X  1010  LLLHLLL   H     /* display “A” */
     X  1011  HHLLLLL   H     /* display “B” */
     X  1100  LHHLLLH   H     /* display “C” */
     X  1101  HLLLLHL   H     /* display “D” */
     X  1110  LHHLLLL   H     /* display “E” */
     X  1111  LHHHLLL   H     /* display “F” */
```

Memory Decoder Example

This is an implementation of a memory decoder; the peculiar feature of the language utilised in this application is the equality operator used to check the inclusion of a variable value in a range of constants.

The corresponding files are reported in Figures from 20 to 23.

Figure 20. Memory Decoder — Source Code

```
Name            Mdecode;
Partno          1;
Revision        00;
Date            None;
Designer        You;
Company         Logical Devices, Inc.;
Assembly        Forgotten Memory;
Location        1;
Device          G20V8;

/************************************************************************/
/* This device generates the memory RAS signals and initiates the       */
/* generation of CAS. It also enables the data bus transceiver for      */
/* both the memory and I/O read cycles.                                 */
/************************************************************************/
/* Allowable Target Device Types:    GAL20V8                            */
/************************************************************************/

/** Inputs **/

pin 2           = !ioacc ;              /* I/O cycle access             */
pin [3..6]      = [A19..16] ;           /* System addresses A16 - A19   */
pin 7           = altloc ;              /* Map RAM to 4000 thru 7FFFF   */
pin 8           = !refcyc ;             /* Memory refresh cycle         */
pin [9,10]      = ![memw,memr] ;        /* Memory read & write strobes  */
pin 11          = !ior ;                /* I/O read strobe              */
pin 14          = raminh ;              /* System RAM inhibit           */
pin 23          = !memacc ;             /* On-board memory access       */

/** Outputs **/

pin 15          = !casacc ;             /* Enable CAS generation        */
pin [16..19]    = ![ras0..ras3] ;       /* RAM RAS signals              */
pin 20          = rdbuff ;              /* Xceiver enable for reads     */

/** Declarations and Intermediate Variable Definitions **/

field memaddr  = [A19..16] ;
memreq  = memw # memr ;
```

Figure 20. Memory Decoder — Source Code *(continued)*

```
memacc_eqn  = !raminh & !refcyc & (memaddr:[00000..3FFFF] & !altloc
              # memaddr:[40000..7FFFF] & altloc) ;

/** Logic Equations **/

ras0  = !raminh & memreq & !refcyc & (memaddr:[00000..0FFFF] &
        !altloc # memaddr:[40000..4FFFF] & altloc) # refcyc ;

ras1  = !raminh & memreq & !refcyc & (memaddr:[10000..1FFFF] &
        !altloc # memaddr:[50000..5FFFF] & altloc) # refcyc ;

ras2  = !raminh & memreq & !refcyc & (memaddr:[20000..2FFFF] &
        !altloc # memaddr:[60000..6FFFF] & altloc) # refcyc ;

ras3  = !raminh & memreq & !refcyc & (memaddr:[30000..3FFFF] &
        !altloc # memaddr:[70000..7FFFF] & altloc) # refcyc ;

casacc  = memreq & memacc_eqn ;
rdbuff  = memacc & memr # ioacc & ior ;
```

Figure 21. Memory Decoder — Simulation Results

```
==========================================================================
                              Simulation Results
==========================================================================
                       !  !              !
                     ! m  r r a          c r
               ! !   i e  e a l  ! ! ! ! a d
               m m ! o m  f m t  r r r r s b
      A A A A  e e i a a  c i l  a a a a a u
      1 1 1 1  m m o c c  y n o  s s s s c f
      9 8 7 6  w r r c c  c h c  3 2 1 0 c f
     ____________________

0001: 0 0 0 0  0 1 1 1 0  1 0 0  H H H L L L
0002: 0 0 0 0  1 0 1 1 0  1 0 0  H H H L L H
0003: 0 0 0 1  0 1 1 1 0  1 0 0  H H L H L L
0004: 0 0 0 1  1 0 1 1 0  1 0 0  H H L H L H
0005: 0 0 1 0  0 1 1 1 0  1 0 0  H L H H L L
0006: 0 0 1 0  1 0 1 1 0  1 0 0  H L H H L H
0007: 0 0 1 1  0 1 1 1 0  1 0 0  L H H H L L
0008: 0 0 1 1  1 0 1 1 0  1 0 0  L H H H L H
0009: 0 1 0 0  0 1 1 1 0  1 0 1  H H H L L L
0010: 0 1 0 0  1 0 1 1 0  1 0 1  H H H L L H
0011: 0 1 0 1  0 1 1 1 0  1 0 1  H H L H L L
0012: 0 1 0 1  1 0 1 1 0  1 0 1  H H L H L H
0013: 0 1 1 0  0 1 1 1 0  1 0 1  H L H H L L
0014: 0 1 1 0  1 0 1 1 0  1 0 1  H L H H L H
0015: 0 1 1 1  0 1 1 1 0  1 0 1  L H H H L L
0016: 0 1 1 1  1 0 1 1 0  1 0 1  L H H H L H
0017: X X X X  1 1 1 1 1  0 0 0  L L L L H L
0018: X X X X  1 1 1 1 1  0 0 1  L L L L H L
0019: 0 0 0 0  1 1 0 0 1  1 0 0  H H H H H H
```

Figure 22. Memory Decoder — Documentation File

```
***************************************************************************
                                    Mdecode
***************************************************************************

CUPL            4.0a Serial# ST-17202000
Device          g20v8as  Library DLIB-h-200-1
Created         Tue Jan 7 12:00:00 1992
Name            Mdecode
Partno          1
Revision        00
Date            None
Designer        You
Company         Logical Devices, Inc.
Assembly        Forgotten Memory
Location        1

===========================================================================
                            Expanded Product Terms
===========================================================================

casacc =
    A18 & !A19 & altloc & memw & !raminh & !refcyc
  # A18 & !A19 & altloc & memr & !raminh & !refcyc
  # !A18 & !A19 & !altloc & memw & !raminh & !refcyc
  # !A18 & !A19 & !altloc & memr & !raminh & !refcyc

memacc_eqn =
    !A18 & !A19 & !altloc & !raminh & !refcyc
  # A18 & !A19 & altloc & !raminh & !refcyc

memaddr =
    A19 , A18 , A17 , A16

memreq =
    memw
  # memr

ras0 =
    !A16 & !A17 & !A18 & !A19 & !altloc & memw & !raminh & !refcyc
  # !A16 & !A17 & !A18 & !A19 & !altloc & memr & !raminh & !refcyc
  # !A16 & !A17 & A18 & !A19 & altloc & memr & !raminh & !refcyc
  # !A16 & !A17 & A18 & !A19 & altloc & memw & !raminh & !refcyc
  # refcyc
```

Figure 22. Memory Decoder — Documentation File *(continued)*

```
ras1 =
    A16 & !A17 & !A18 & !A19 & !altloc & memw & !raminh & !refcyc
  # A16 & !A17 & !A18 & !A19 & !altloc & memr & !raminh & !refcyc
  # A16 & !A17 & A18 & !A19 & altloc & memr & !raminh & !refcyc
  # A16 & !A17 & A18 & !A19 & altloc & memw & !raminh & !refcyc
  # refcyc

ras2 =
    !A16 & A17 & !A18 & !A19 & !altloc & memw & !raminh & !refcyc
  # !A16 & A17 & !A18 & !A19 & !altloc & memr & !raminh & !refcyc
  # !A16 & A17 & A18 & !A19 & altloc & memr & !raminh & !refcyc
  # !A16 & A17 & A18 & !A19 & altloc & memw & !raminh & !refcyc
  # refcyc

ras3 =
    A16 & A17 & !A18 & !A19 & !altloc & memw & !raminh & !refcyc
  # A16 & A17 & !A18 & !A19 & !altloc & memr & !raminh & !refcyc
  # A16 & A17 & A18 & !A19 & altloc & memr & !raminh & !refcyc
  # A16 & A17 & A18 & !A19 & altloc & memw & !raminh & !refcyc
  # refcyc

rdbuff =
    memacc & memr
  # ioacc & ior

==========================================================================
                                 Symbol Table
==========================================================================

Pin Variable                                    Pterms   Max     Min
Pol   Name              Ext     Pin     Type     Used   Pterms  Level
-- ----            --     --     --    ---  ---  ---

    A16                         6        V         -       -       -
    A17                         5        V         -       -       -
    A18                         4        V         -       -       -
    A19                         3        V         -       -       -
    altloc                      7        V         -       -       -
 !  casacc                      15       V         4       8       1
 !  ioacc                       2        V         -       -       -
 !  ior                         11       V         -       -       -
 !  memacc                      23       V         -       -       -
    memacc_eqn                  0        I         2       -       -
    memaddr                     0        F         -       -       -
 !  memr                        10       V         -       -       -
```

Figure 22. Memory Decoder — Documentation File *(continued)*

```
    memreq                      0        I        2       -       -
 !  memw                        9        V        -       -       -
    raminh                      14       V        -       -       -
 !  ras0                        16       V        5       8       1
 !  ras1                        17       V        5       8       1
 !  ras2                        18       V        5       8       1
 !  ras3                        19       V        5       8       1
    rdbuff                      20       V        2       8       1
 !  refcyc                      8        V        -       -       -


LEGEND    F : field      D : default variable        M : extended node
          N : node       I : intermediate variable   T : function
          V : variable   X : extended variable       U : undefined

==========================================================================
                                   Fuse Plot
==========================================================================

Syn   02704 - Ac0   02705 x

Pin #22  02560  Pol x  02632  Ac1 -
 00000 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00040 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00080 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00120 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00160 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00200 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00240 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00280 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #21  02561  Pol x  02633  Ac1 -
 00320 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00360 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00400 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00440 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00480 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00520 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00560 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00600 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #20  02562  Pol -  02634  Ac1 x
 00640 ----x-------------x---
 00680 -x------------------x-
 00720 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00760 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00800 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Figure 22. Memory Decoder — Documentation File *(continued)*

```
  00840 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  00880 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  00920 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 Pin #19  02563  Pol x  02635  Acl x
  00960 ——-x--x-x--x—x-x—x——-x——
  01000 ——-x--x-x--x—x-x————x-x——
  01040 ——-x-x--x--x--x--x————x-x——
  01080 ——-x-x--x--x--x--x—x——-x——
  01120 ————————————-x———————
  01160 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  01200 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  01240 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 Pin #18  02564  Pol x  02636  Acl x
  01280 ——-x--x-x—x--x-x—x——-x——
  01320 ——-x--x-x—x--x-x————x-x——
  01360 ——-x-x--x—x-x--x————x-x——
  01400 ——-x-x--x—x-x--x—x——-x——
  01440 ————————————-x———————
  01480 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  01520 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  01560 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 Pin #17  02565  Pol x  02637  Acl x
  01600 ——-x--x--x-x—x-x—x——-x——
  01640 ——-x--x--x-x—x-x————x-x——
  01680 ——-x-x—x-x--x--x————x-x——
  01720 ——-x-x—x-x--x--x—x——-x——
  01760 ————————————-x———————
  01800 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  01840 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  01880 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 Pin #16  02566  Pol x  02638  Acl x
  01920 ——-x--x--x--x--x-x—x——-x——
  01960 ——-x--x--x--x--x-x————x-x——
  02000 ——-x-x—x--x-x--x————x-x——
  02040 ——-x-x—x--x-x--x—x——-x——
  02080 ————————————-x———————
  02120 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  02160 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  02200 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 Pin #15  02567  Pol x  02639  Acl x
  02240 ——-x-x—————-x--x—x——-x——
  02280 ——-x-x—————-x--x————x-x——
  02320 ——-x--x—————-x-x—x——-x——
  02360 ——-x--x—————-x-x————x-x——
  02400 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Figure 22. Memory Decoder — Documentation File *(continued)*

```
 02440 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 02480 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 02520 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

LEGEND    X : fuse not blown
          - : fuse blown

==========================================================================
                                 Chip Diagram
==========================================================================


                              _______________
                             |   Mdecode     |
                          x---|1           24|---x Vcc
                   !ioacc x---|2           23|---x !memacc
                      A19 x---|3           22|---x
                      A18 x---|4           21|---x
                      A17 x---|5           20|---x rdbuff
                      A16 x---|6           19|---x !ras3
                   altloc x---|7           18|---x !ras2
                  !refcyc x---|8           17|---x !ras1
                    !memw x---|9           16|---x !ras0
                    !memr x---|10          15|---x !casacc
                     !ior x---|11          14|---x raminh
                      GND x---|12          13|---x
                             |_______________|
```

Figure 23. Memory Decoder — Input File for Simulation

```
Name            Mdecode;
Partno          1;
Revision        00;
Date            None;
Designer        You;
Company         Logical Devices, Inc.;
Assembly        Forgotten Memory;
Location        1;
Device          G20V8;

/************************************************************************/
/* This device generates the memory RAS signals and initiates the      */
/* generation of CAS. It also enables the data bus transceiver for     */
/* both the memory and I/O read cycles.                                */
/************************************************************************/
/* Allowable Target Device Types:       G20V8                          */
/************************************************************************/
```

Figure 23. Memory Decoder — Input File for Simulation *(continued)*

```
ORDER:  A19, %1, A18, %1, A17, %1, A16, %2,
        !memw, %1, !memr, %1, !ior, %1, !ioacc, %1, !memacc, %2,
        !refcyc, %1, raminh, %1, altloc, %2,
        !ras3, %1, !ras2, %1, !ras1, %1, !ras0, %1, !casacc, %1, rdbuff;

VECTORS:

$msg "                       !  !              !";
$msg "                     ! m  r r a          c r";
$msg "               ! !   i e  e a l  ! ! ! ! a d";
$msg "               m m ! o m  f m t  r r r r s b";
$msg "      A A A A  e e i a a  c i l  a a a a a u";
$msg "      1 1 1 1  m m o c c  y n o  s s s s c f";
$msg "      9 8 7 6  w r r c c  c h c  3 2 1 0 c f";
$msg "     --------------------";
0 0 0 0  0 1 1 1 0  1 0 0  H H H L L L  /* memory access wr 00000-0FFFF */
0 0 0 0  1 0 1 1 0  1 0 0  H H H L L H  /* memory access rd 00000-0FFFF */
0 0 0 1  0 1 1 1 0  1 0 0  H H L H L L  /* memory access wr 10000-1FFFF */
0 0 0 1  1 0 1 1 0  1 0 0  H H L H L H  /* memory access rd 10000-1FFFF */
0 0 1 0  0 1 1 1 0  1 0 0  H L H H L L  /* memory access wr 20000-2FFFF */
0 0 1 0  1 0 1 1 0  1 0 0  H L H H L H  /* memory access rd 20000-2FFFF */
0 0 1 1  0 1 1 1 0  1 0 0  L H H H L L  /* memory access wr 30000-3FFFF */
0 0 1 1  1 0 1 1 0  1 0 0  L H H H L H  /* memory access rd 30000-3FFFF */
0 1 0 0  0 1 1 1 0  1 0 1  H H H L L L  /* memory access wr 40000-4FFFF */
0 1 0 0  1 0 1 1 0  1 0 1  H H H L L H  /* memory access rd 40000-4FFFF */
0 1 0 1  0 1 1 1 0  1 0 1  H H L H L L  /* memory access wr 50000-5FFFF */
0 1 0 1  1 0 1 1 0  1 0 1  H H L H L H  /* memory access rd 50000-5FFFF */
0 1 1 0  0 1 1 1 0  1 0 1  H L H H L L  /* memory access wr 60000-6FFFF */
0 1 1 0  1 0 1 1 0  1 0 1  H L H H L H  /* memory access rd 60000-6FFFF */
0 1 1 1  0 1 1 1 0  1 0 1  L H H H L L  /* memory access wr 70000-7FFFF */
0 1 1 1  1 0 1 1 0  1 0 1  L H H H L H  /* memory access rd 70000-7FFFF */
X X X X  1 1 1 1 1  0 0 0  L L L L H L  /* memory refresh cycle         */
X X X X  1 1 1 1 1  0 0 1  L L L L H L  /* memory refresh cycle         */
0 0 0 0  1 1 0 0 1  1 0 0  H H H H H H  /* i/o read cycle               */
```

Eight Set-Reset Flip-Flops Example

This example shows how to configure a GAL20V8 in order to have eight feedbacks into the AND array in a device that does not use any macrocell as registered.

For a GAL16V8, the original specifications indicate that it is possible to use the feedbacks from pins 12 and 19 only if at least one macrocell is configured as registered.

The reason for this limitation is that using the feedback from pin 19 forbids the use of pin 1 as an input: this fact is not relevant if at least one macrocell is registered, because in this case pin 1 is the clock input.

If no macrocell is configured as registered, pin 1 could be used as an input, but in mutual exclusion with the feedback from pin 19: as shown in Figure 24, the *FMUX* multiplexer of Output Logic Macrocell 19 route to the AND array either the signal coming from pin 1 or the feedback from pin 19. More precisely:

- if SYN is 0 (registered mode) the feedback into the AND array is from the flip-flop (AC1(19)=0, registered output) or from the output pin (AC1(19)=1, combinatorial output);
- if SYN is 1 (complex or simple mode) the signal coming from pin 1 is directed into the AND array.

Actually, there is no reason to forbid the use of the feedback from pin 19 if pin 1 is not utilized as an input.

So, it is possible to use the feedback from pin 19 if:

- all the macrocell are combinatorial and pin 1 is not used, or
- at least one macrocell is registered, so pin 1 is the clock input.

A similar reasoning applies to pin 12 (in mutual exclusion with pin 11, used as global output enable of registered macrocells).

The compiler does not accept a feedback from pin 19 (or pin 12) if all the macrocells are combinatorial, without controlling if pin 1 (or pin 11 respectively) is declared or not.

The only way to force the compiler to accepts a feedback from pin 19 (or pin 12) also if no one macrocell is registered, is to force the device to be configured in registered mode, specifying *G16V8MS* as device type. Of course, in this case pin 1 (or pin 11) must be left unused.

If the device is a GAL20V8 then the feedbacks forbidden are those from pins 15 and 22.

Figures 25 to 28 contain the source file, the simulation results, the documentation file and the input file for simulation.

Figure 24. GAL16V8AS Output Logic Macrocell Pin 19

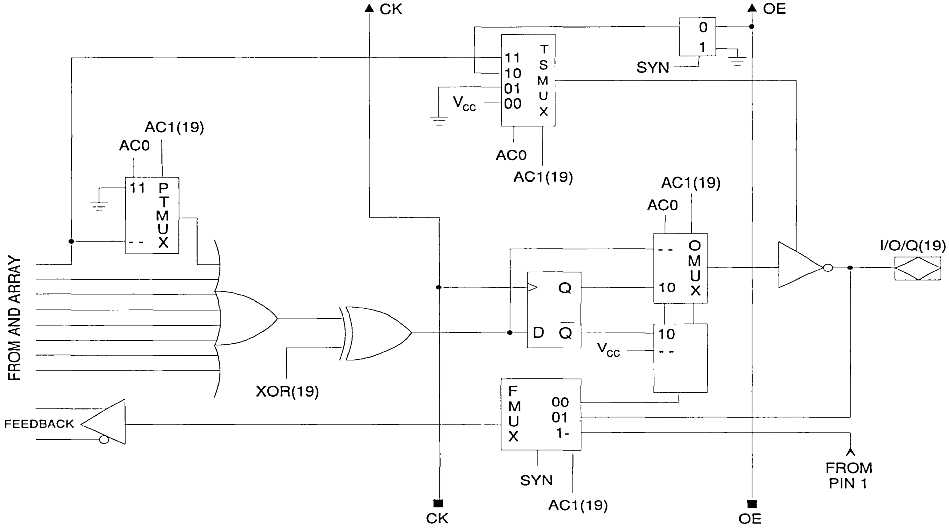


Figure 25. Eigth Set-Reset Flip-Flops — Source Code

```
Name            Flop_SR;
Partno          None;
Revision        00;
Date            None;
Designer        Ernesto;
Company         SGS-THOMSON Microelectronics;
Location        None;
Assembly        None;
Device          G20V8MS;

/* Inputs */

Pin [2..9] = [S1..8];     /* The Sx signal is the SET of the flip-flop x */
Pin 10 = RST;             /* Common RESET for all flip-flops */

/* Outputs - The Qx signal is the output of the flip-flop x */

Pin [22..15] = [Q1..8];

/* Logic equations implementing 8 SET-RESET flip-flops */

[Q1..8] = [S1..8] # !RST & [Q1..8];
```

Figure 26. Eigth Set-Reset Flip-Flops — Simulation Results

```
==========================================================================
                              Simulation Results
==========================================================================

      S1  S2  S8  RST Q1  Q2  Q8

0001: 0   0   0    1  L   L   L
0002: 0   0   1    0  L   L   H
0003: 0   1   0    0  L   H   H
0004: 1   0   0    0  H   H   H
0005: 0   0   0    1  L   L   L
0006: 0   1   1    0  L   H   H
0007: 0   0   0    1  L   L   L
0008: 1   1   0    0  H   H   L
0009: 0   0   0    1  L   L   L
0010: 1   0   1    0  H   L   H
0011: 0   0   0    1  L   L   L
0012: 1   1   1    0  H   H   H
0013: 0   0   0    1  L   L   L
```

Figure 27. Eigth Set-Reset Flip-Flops — Documentation File

```
***************************************************************************
                                    Flop_SR
***************************************************************************

CUPL            4.0a Serial# ST-17202000
Device          g20v8ms  Library DLIB-h-200-3
Created         Tue Jan  7 12:00:00 1992
Name            Flop_SR
Partno          None
Revision        00
Date            None
Designer        Ernesto
Company         SGS-THOMSON Microelectronics
Assembly        None
Location        None

===========================================================================
                            Expanded Product Terms
===========================================================================

Q1 =
    S1
  # Q1 & !RST

Q2 =
    S2
  # Q2 & !RST

Q3 =
    S3
  # Q3 & !RST

Q4 =
    S4
  # Q4 & !RST

Q5 =
    S5
  # Q5 & !RST

Q6 =
    S6
  # Q6 & !RST
```

Figure 27. Eigth Set-Reset Flip-Flops — Documentation File *(continued)*

```
Q7 =
    S7
  # Q7 & !RST

Q8 =
    S8
  # Q8 & !RST

Q1.oe  =
    1

Q2.oe  =
    1

Q3.oe  =
    1

Q4.oe  =
    1

Q5.oe  =
    1

Q6.oe  =
    1

Q7.oe  =
    1

Q8.oe  =
    1


==========================================================================
                                 Symbol Table
==========================================================================

Pin Variable                                    Pterms   Max     Min
Pol   Name                Ext     Pin     Type    Used   Pterms  Level
-- ----             --     --     --     ---  ---  ---

    Q1                          22       V        2       7       1
    Q2                          21       V        2       7       1
    Q3                          20       V        2       7       1
    Q4                          19       V        2       7       1
    Q5                          18       V        2       7       1
```

Figure 27. Eigth Set-Reset Flip-Flops — Documentation File *(continued)*

```
    Q6                          17      V       2       7       1
    Q7                          16      V       2       7       1
    Q8                          15      V       2       7       1
    RST                         10      V       -       -       -
    S1                          2       V       -       -       -
    S2                          3       V       -       -       -
    S3                          4       V       -       -       -
    S4                          5       V       -       -       -
    S5                          6       V       -       -       -
    S6                          7       V       -       -       -
    S7                          8       V       -       -       -
    S8                          9       V       -       -       -
    Q1                  oe      22      D       1       1       0
    Q2                  oe      21      D       1       1       0
    Q3                  oe      20      D       1       1       0
    Q4                  oe      19      D       1       1       0
    Q5                  oe      18      D       1       1       0
    Q6                  oe      17      D       1       1       0
    Q7                  oe      16      D       1       1       0
    Q8                  oe      15      D       1       1       0


  LEGEND    F : field      D : default variable       M : extended node
            N : node       I : intermediate variable  T : function
            V : variable   X : extended variable      U : undefined


  =========================================================================
                                  Fuse Plot
  =========================================================================

  Syn   02704 x Ac0   02705 -

  Pin #22  02560  Pol -  02632  Ac1 -
   00000 ----------------------------------------
   00040 x---------------------------------------
   00080 ------x--------------------------x------
   00120 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
   00160 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
   00200 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
   00240 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
   00280 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  Pin #21  02561  Pol -  02633  Ac1 -
   00320 ----------------------------------------
   00360 ----x-----------------------------------
   00400 ----------x----------------------x------
```

Figure 27. Eigth Set-Reset Flip-Flops — Documentation File *(continued)*

```
 00440 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00480 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00520 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00560 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00600 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #20  02562  Pol -  02634  Acl -
 00640 ----------------------------------------
 00680 --------x-------------------------------
 00720 --------------x------------------x------
 00760 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00800 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00840 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00880 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 00920 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #19  02563  Pol -  02635  Acl -
 00960 ----------------------------------------
 01000 ------------x---------------------------
 01040 ------------------x--------------x------
 01080 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01120 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01160 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01200 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01240 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #18  02564  Pol -  02636  Acl -
 01280 ----------------------------------------
 01320 ----------------x-----------------------
 01360 ----------------------x----------x------
 01400 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01440 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01480 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01520 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01560 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #17  02565  Pol -  02637  Acl -
 01600 ----------------------------------------
 01640 --------------------x-------------------
 01680 --------------------------x------x------
 01720 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01760 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01800 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01840 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 01880 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Pin #16  02566  Pol -  02638  Acl -
 01920 ----------------------------------------
 01960 ------------------------x---------------
 02000 ------------------------------x--x------
 02040 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Figure 27. Eigth Set-Reset Flip-Flops — Documentation File *(continued)*

```
  02080 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  02120 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  02160 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  02200 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 Pin #15  02567  Pol -  02639  Acl -
  02240 ----------------------------------------
  02280 ----------------------------x-----------
  02320 ---------------------------------xx-----
  02360 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  02400 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  02440 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  02480 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  02520 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

 LEGEND    X : fuse not blown
           - : fuse blown

 ==========================================================================
                                Chip Diagram
 ==========================================================================

                                  ______________
                                 |   Flop_SR    |
                              x---|1          24|---x Vcc
                           S1 x---|2          23|---x
                           S2 x---|3          22|---x Q1
                           S3 x---|4          21|---x Q2
                           S4 x---|5          20|---x Q3
                           S5 x---|6          19|---x Q4
                           S6 x---|7          18|---x Q5
                           S7 x---|8          17|---x Q6
                           S8 x---|9          16|---x Q7
                          RST x---|10         15|---x Q8
                              x---|11         14|---x
                          GND x---|12         13|---x
                                 |______________|
```

Figure 27. Eigth Set-Reset Flip-Flops — Input File for Simulation

```
Name            Flop_SR;
Partno          None;
Revision        00;
Date            None;
Designer        Ernesto;
Company         SGS-THOMSON Microelectronics;
Location        None;
Assembly        None;
Device          G20V8MS;

/* Only some flip-flops are simulated,                */
/* because the behaviour of the others is identical. */

Order: S1,%3,S2,%3,S8,%4,RST,%2,Q1,%3,Q2,%3,Q8;

Vectors:

$msg “”;
$msg “      S1   S2   S8   RST Q1   Q2   Q8";
$msg “      —————————————”;

0001 LLL
0010 LLH
0100 LHH
1000 HHH
0001 LLL
0110 LHH
0001 LLL
1100 HHL
0001 LLL
1010 HLH
0001 LLL
1110 HHH
0001 LLL
```

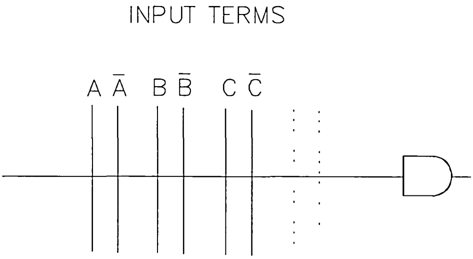
MEMORY CELL

GAL®: THE MEMORY STRUCTURE IN THE AND ARRAY

The so called “AND array” makes use of the main part of the EEPROM memory of the GAL®. (Smaller memory sections are present to implement the OLMC architecture options, the electronic signature, the security bit and some others bits for the device manufacturer use, that do not affect the operation of the device).

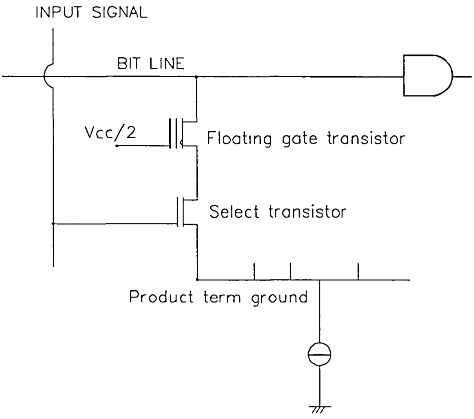
The AND array is basically made up of a repeated structure *(64 times in a GAL16V8 or in a GAL20V8)* called the “product term”.

The functional representation of a product term is (see also the chapter “Logic Concepts”):



The interconnections of an input signal *(or of its negation)* to the horizontal connection *(which is an actual metal strip in the chip, called BIT LINE because of its similarity to normal memory circuits)* can be programmed by the EEPROM cell present in each crossing.

The circuit implementing each crossing is:



An active input signal turns on the select transistor.

If the corresponding floating gate transistor has been programmed for a short circuit, the select transistor sinks current from the bit line. If , instead, the floating gate transistor has been programmed for an open circuit, the input signal has no effect whatsoever.

A current limiter gathers all the select transistors of a bit line, and limits the maximum amount of current that can be sunk from the bit line. *(There may be up to 16 select transistors active at the same time in a GAL16V8. There are 32 [33 including the PTD, see below] of them, but each pair is driven by an input and its complement).*

The Product Term Function

Unlike in standard memory circuits, in the GAL® array more than one select transistor may be active at the same time. This allows the implementation of a “wired NOR” function of the input signals complements, i.e. the AND of the input signals.

Consequently a current limiter is needed, to limit the power dissipation and to limit the voltage swing of the bit line, so reducing the slow-down effect of the parasitic capacitances affecting the long bit line.

At the end of each bit line, a current-to-voltage converter (the “sense amplifier”) detects whether at least one crossing is sinking current from its input, and translates this info into a CMOS level.

No one sinking =
= no programmed cell is selected =
= NOR
+ taking into account, when generating the programming pattern, the use of inverted inputs =
= AND =
= PRODUCT TERM

The sense amplifier must operate at very high speed with an input signal in the range of tens of millivolts. It is an analog piece of circuitry, and all of them together *(64 in a GAL16V8)* account for most of the d.c. power consumption of the GAL®.

The Product Term Disable

It is common practice that in the unused bit lines all the cells are programmed for a short circuit, thereby ensuring a constant 0 at the output of the product term.

In a GAL® it is possible to reduce the associated power consumption, that corresponds in the above situation to the current saturating the current limiter.

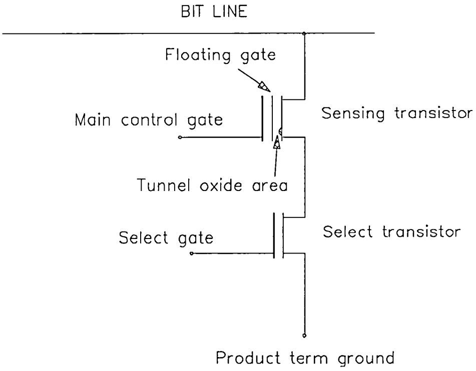
There is an additional cell that can ground the bit line as well: the Product Term Disable, or PTD.

By programming the PTD and leaving unprogrammed all the other cells of an unused product term, a steady saving of about 500 µW comes free from every unused product term so set.

The Single Poly Cell in GAL®

The structure made of a select transistor and the associated floating gate (= memory) transistor gives the GAL® its programmability.

The floating gate transistor may, during the programming phase, be permanently modified in its gate threshold voltage by an injection (or extraction) of electrons in the floating gate itself. If the floating gate is given electrons, the threshold seen from the external gate (the "main control gate") becomes more positive (+7 volt) because the negative field of those electrons has to be overcome; vice versa if electrons are taken out (+7 volt).



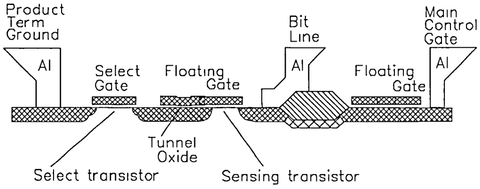
As the main control gates of all memory transistors are connected together, and set at 2.5 volt all during normal operation, these transistors act as permanent open or short circuits.

The floating gate is given or taken electrons during the programming phase. By applying a voltage well in excess of 10 volt across a thin oxide area below the floating gate, a "tunnel" current can flow.

After the programming phase the tunnel oxide spot never sees voltages high enough for a tunnel current of any significance to flow. All over the rest of the floating gate, a much thicker oxide layer isolates it.

During normal operation the voltage distribution across the tunnel oxide (and the oxide surronding the floating gate) remains low (so that the tunnel current is also extremely low) and the floating gate charge retention is unaffected for much longer than the specified 20 years of data retention.

GAL single poly EEPROM cell cross section

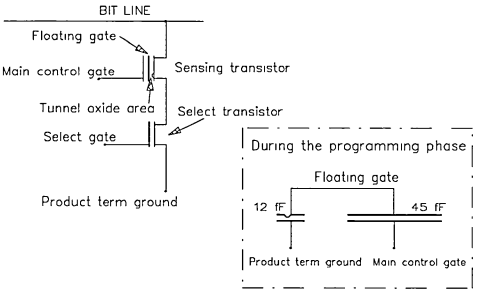


(What appears as two separate parts of the floating gate in the above figure, correspond in actuality to a unique element, that has the joining part behind the plane of the section shown, where it passes below the aluminium strip of the bit line).

The Voltage on the Floating Gate

To apply the programming voltage to the floating gate (that is electrically insulated) the cell is designed in order to exploit the capacitive coupling of the floating gate itself to its own source diffusion and to the main control gate.

The floating gate creates a relatively large capacitor with the diffusion of the main control gate beneath, and a significantly smaller capacitor with the small pieces of the memory transistor beneath *(source, drain diffusions, channel area and tunnel oxide area; the tunnel oxide is thinner (90 Å) than the oxide elsewhere in the described capacitors, (300 to 350 Å) but its surface is so small that the associated capacitance gives a small contribution).*



When programming, the bit line is disconnected, and the main control gate is grounded, while the high voltage is applied to the product term ground. In this case, if a given select transistor is on, the associated memory transistor can expel electrons from its floating gate.

In fact the high capacitive coupling with the main control gate attracts the floating gate close to ground, while the other side of the small capacitance, where the tunnel oxide is, receives the high voltage from the select transistor. The floating gate is set more positive, and in normal operation afterwards, the memory transistor acts as a short circuit.

Viceversa, always with the bit lines disconnected, the high voltage is given to the main control gate and ground to the PTG.

Selecting then the memory positions where to put open circuits, the relevant select transistors will insert electrons in the floating gates they address.

(In practice the programming is done first with a bulk erase of the entire memory, followed by the programming, one product term at a time, of only the cells that are to become short circuits)

Single Poly Versus Double Poly

It has been emphasized that the data retention is good if the floating gate is free from any leakage of the programmed charge.

The oxide surrounding the floating gate must be flawless all over, not only the small section of it made very thin on purpose to allow the programming via the tunnel effect. Moreover, the tunnel oxide must not be affected by any subsequent process step after its growth.

In this respect the single poly process used presently by SGS-THOMSON offers an additional security in comparison with the double poly process previously used for its GAL®, and still used by some competitors.

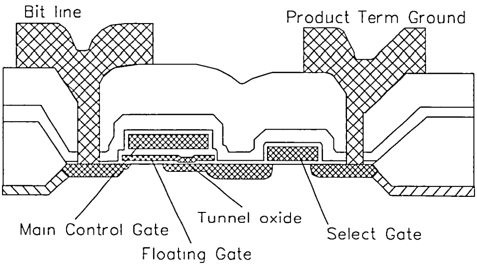
A double poly technology implements the capacitive structure described in the figure above putting the main control gate armature above the floating gate. This MCG armature is implemented by a second poly layer on top of the poly layer of the floating gate.

Apart from a general complexity increase of the process (the additional layer requires at least two more masking steps), the second poly layer implies additional processing of the wafers after the delicate tunnel oxide has been manufactured.

These additional steps are necessarily different in the case of the double poly, because the interpoly oxide must be relatively thin and well controlled in its thickness to implement the desired capacitor between the MCG and the floating gate.

The interpoly oxide must then be grown with a dry oxidation at high temperature, to precisely control its thickness. It is this high temperature that stresses the tunnel oxide beneath the first poly.

GAL double poly EEPROM cell cross section



In the case of the single poly process, the oxide that seals the floating gate needs not to be neither thin nor tightly controlled in its thickness. It is grown with a steam oxidation process at lower temperature, is thicker and seals at least as well, without jeopardizing the tunnel oxide integrity.

GAL® MEMORY CELL CONVENTIONS

The following table shows the conventions used to indicate the status of a memory cell.

	Written	Erased
Floating Gate	Missing Electrons (Positive)	With Electrons (Negative)
Voltage Threshold	Low	High
Logic Level	0	1
Bit Map	X X X	– – –

LATCH-UP IMMUNITY

LATCH-UP IMMUNITY

Abstract

The users of CMOS integrated circuits are aware of the inherent risk of latch-up in those devices.

GAL® are by design practically immune from this risk.

The basics of latch-up are recalled, and the solutions implemented in the GAL® devices are outlined, to offer a description on how their latch-up immunity has been designed-in since their conception.

The Latch-up

A CMOS structure cannot be implemented without implementing at the same time a parasitic PNPN structure, connected between V_{CC} and ground.

Such parasitic structure has, in all practical cases of CMOS devices, a capability of being triggered into a runaway state where it absorbs power from the power supply and quickly causes the chip destruction with the high power dissipation associated.

However, in the range of supply voltages, temperatures and external disturbances of practical use of the CMOS devices, they can be, and are, designed and manufactured so that the latch-up almost never presents itself.

A CMOS product can (according to its diffusion process and chip design) be more or less (but always only in extreme conditions) inclined to latching-up.

Figure 1 shows the essential elements of a CMOS inverter stage, along with the parasitic bipolar structures.

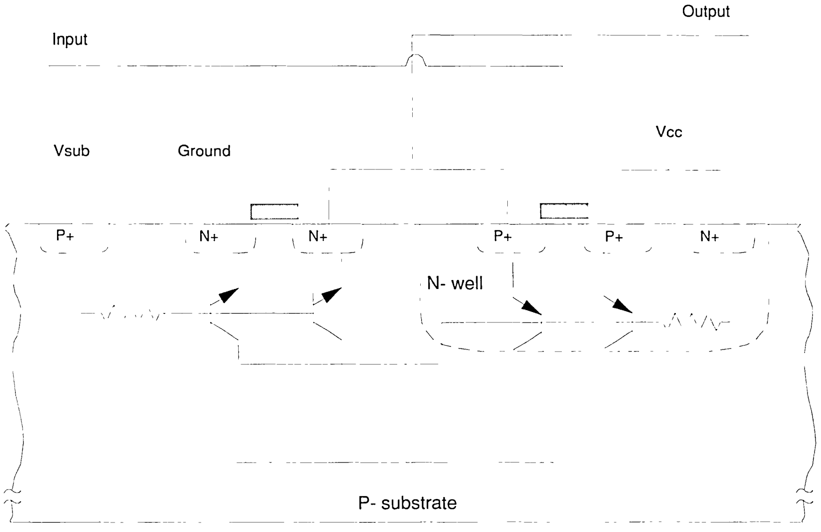
The figure shows the case of an N-well implementation (like GAL®); a perfectly dual figure would hold good for a P-well CMOS.

As Figure 1 may suggest, the substrate is often (but not in GAL®) connected to ground.

The P-channel transistor, from its source and drain diffusions, forms PN junctions with the N-well into which it is located. Taking into account also the P substrate, two PNP transistors can be identified. Their bases are biased to V_{CC}, but such connection is affected by a relatively high resistance, due to the N-well resistivity. Consequently they do not conduct, unless a transient disturbance can make current flow through the N-well and generate a voltage drop higher than a PN threshold (0.6 volt) across the base junctions, so that a base current can be made flow from the transistor bases.

Similarly, the source and drain diffusions of the N-channel transistor form the emitters of NPN structures, where the bases are made up by the substrate itself, and the collectors by the adjacent N-well. Again the bases of such NPN transistors

Figure 1. Parasitic Latch-up Structures



are negatively biased by the connection to the substrate bias contact on the chip surface. The connection exhibits unfortunately a non negligible resistance, so that an unwanted transient may make the substrate voltage oscillate and even reach the dangerous point where the bases become forward biased.

Figure 2 shows the equivalent circuit of the parasitic bipolar structures of Figure 1.

On the left side of Figure 2, the standard case of a CMOS logic is shown.

It is now easier to realize how a transient in the P-substrate (or in the N-well) can momentarily forward bias the transistor base that is one and the same with it.

This base current, amplified by the bipolar transistor gain, becomes a collector current. The magnitude of such collector current may, in turn, be enough to forward bias the other transistor shown in the left hand part of Figure 2. If this is the case, then the situation may become regenerative, with the second transistor reciprocating the gift from the first, providing further base current to it from its collector. The initial trigger current is no longer needed, and the permanent latch-up situation has been reached.

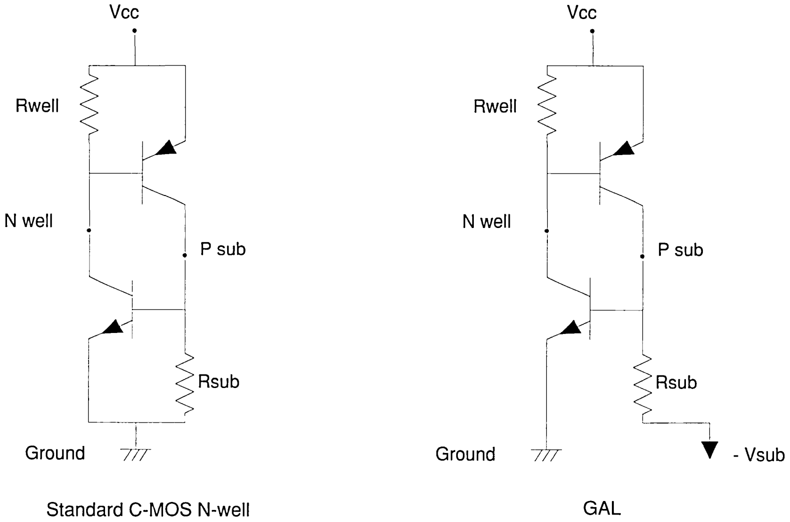
Once the latch-up has been triggered, the latch-up current rises to a very high value and can only be terminated by removing the supply voltage from the chip or (unfortunately more often) by the chip self destruction subsequent to the very high power dissipation generated.

Conditions For The Latch-up

Although the possibility of the latch-up always exists, it can be reduced to a level of negligible risk in a practical application. It is important to consider the factors influencing the phenomenon.

- The parasitic transistors gain. It is necessary that the product of the gains of the two parasitic transistors in each PNPN structure be greater than one. If not, the PNPN structure cannot be triggered into a regenerative condition. Unfortunately, in practice there are always in a CMOS chip several PNPN structures with high enough a gain. It must also be kept into consideration that a higher chip temperature corresponds to higher gains of the bipolar transistors, and to a reduced latch-up immunity.
- The base-to-emitter resistances. Lower resistances require more current before a voltage of 0.6 volt can be created across them. This means that a stronger disturbance is needed to reach the triggering point where a base current starts flowing into the base of a parasitic bipolar transistor.
- The level of the disturbance itself. It must be pointed out that the latch-up can only be trig-

Figure 2. Equivalent Circuit of the Parasitic Structure



gered by a signal generated outside the chip. As an immediate consequence it can be understood that the input and output stages of the chip are by far the best candidates for the latch-up because the external noises are reaching them first, with the minimum attenuation. The structures susceptible to the latch-up can be laid out in the chip with additional circuit solutions so that the external noise reaches them only after having undergone a substantial attenuation anyway. The waveform of the disturbance is also important. A square wave with a low duty-cycle is considerably less dangerous than a DC waveform of the same average DC value.

The input stages can be designed so that the voltage applied to them by external circuits has a negligible coupling with the parasitic PNPN structure associated (the input resistances are extremely high, the capacitances very low, and the input protection circuits can be implemented without adding regenerative PNPN structures).

The internal stages generate waveforms that are totally predictable, and the latch-up can be excluded at the design stage.

The output stages and the V_{CC} connection are left as the only possible entry points for noise pulses able to trigger the latch-up.

Figure 3 shows the circuit corresponding to the structure of Figure 1, where the output connection with the parasitic structures is identified.

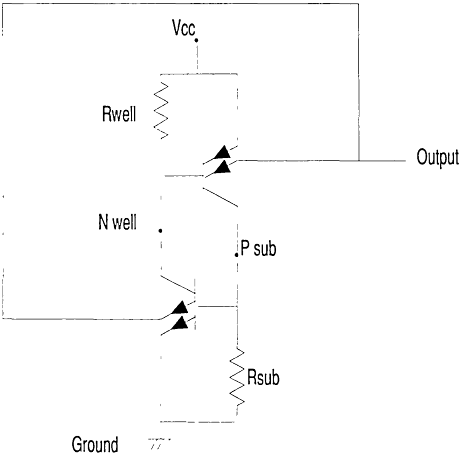
Precautions against the Latch-up

Several precautions altogether are normally implemented during a product design. The most common make use of:

- Diffusion process parameters. The doping concentrations of the substrate and of the wells play a role in the sense that lower concentrations are associated with higher gains of the parasitic transistors.
- Minimum distances between the relevant N and P diffusions, to get wide enough bases in the parasitic bipolar transistors, and accordingly lower gains.
- Additional guard rings. They are diffusion layout patterns guarding the path between two diffusions of the same type that may be part of a PNPN structure latching-up. These rings are biased to V_{CC} (or to the substrate bias voltage, according to their P or N characteristic) and intercept the current that may otherwise generate spurious bias voltages and the latch-up. The guard rings act either as:
  - additional collectors of some parasitic bipolar transistors, and they gather current that they deliver directly to V_{CC} or ground (or $-V_{BB}$ if the substrate bias is implemented), taking such current away from the regenerative path of a PNPN structure; or
  - redundant contacts for the biasing of the substrate (at ground or at $-V_{BB}$, according to the device family) and of the wells (at V_{CC}), so that the parasitic resistances are kept at as low a value as possible.

  Any current, that may otherwise affect the substrate (or a well) bias, shall find a short and conductive path very easily towards a nearby contact or an extra-collector, in all cases.

Figure 3. CMOS Output: the Structure that may Latch-up



Solutions Implemented in the GAL®

Apart from the mentioned points of:

- careful evaluation of the diffusion parameters,
- proper distances between diffusions for wide parasitic transistor bases,
- additional guard rings,

the GAL® have taken advantage, for their latch-up immunity, of two drastic and effective circuit solutions:

- negative bias (-2.5 volt) of the substrate,
- disconnection of the major part of the chip during the power-up transient.

Substrate Bias

All the SGS-THOMSON GAL® have an on-chip generator (driven by an oscillator free-running at 16 MHz) that, since a few microseconds after the supply voltage V_{CC} has been applied to the chip, forces the voltage of the chip substrate down to -2.5 volt.

This fact has two main beneficial effects:

- the N-channel transistors become faster, adding considerably to the overall switching speed of the device, and
- the latch-up becomes practically impossible, due to the very strong negative bias forced into the NPN transistors of the parasitic structures (see also the right hand side of Figure 2).

Power-on Protection

However the body bias is not effective during the short period since V_{CC} is applied until the body generator output has reached its final value.

To ensure the latch-up immunity also during such interval, the GAL® are equipped with a set of ancillary circuits, very effective in completing the shield against the latch-up.

Figure 4 shows the block diagram of these last circuits.

The circuits that are connected directly to the external supply voltage are made only with N-channel transistors, so that no parasitic regenerative structures are present.

Only after the body bias has been acknowledged as reached and stable by the V_{BB} detector, are the circuits with P-channel transistors given the supply voltage.

A large pass transistors takes care of blocking the V_{CC} until a completely safe situation has been reached.

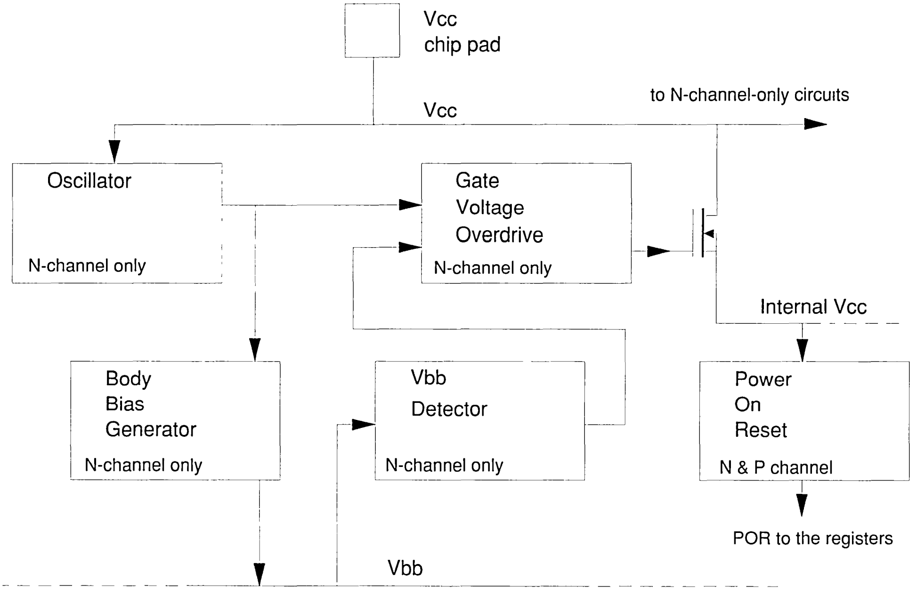
V_{CC} is directly connected to circuits made with N-channel transistors only. In addition to the ones shown in Figure 4, there are also the output stages and some sections of the sense amplifiers.

Output Buffer

A last precaution taken in the GAL® against the latch-up and worth mentioning is in fact the exclusive use of N-channel transistors in the last stage of the output circuits.

It can be seen from Figure 5 that the GAL® outputs are not CMOS stages, and that the upper N-channel transistor, with is gate at V_{CC}, can only yield an output level of (V_{CC} - V_{th}), where V_{th} is its threshold voltage, and has a value of about 1.0 volt.

Figure 4. Block Diagram of the Power-on Protection



The output swing is then from 0 to 4 volt, still well adequate for a TTL compatible signal, and the stages are totally immune from latch-up.

Experimental Evidence

The latch-up immunity is checked every time a new GAL® is introduced, running a set of tests on samples taken from different production lots.

Amongst others, the following tests are preformed on the parts:

- injection of overcurrent (positive and negative) into each input and I/O terminal (the I/O being programmed as an input),
- positive and negative overvoltages applied to the same terminals,
- positive overvoltage on the V_{CC} terminal.

At each test, the supply current readings, in normal operation before and after the applied stress, are contrasted.

Any variation is significant to the detection of a latch-up somewhere in the chip.

The upper temperature limits are also explored with the same methods, as well as a set of different (and sometimes even very unlikely!) power-up transients.

All the results gathered during such characterization exercises have always confirmed that the latch-up can not be induced, unless stresses well outside the absolute maximum ratings are applied.

Further evidence has been collected in the running activity of analysing the field rejects. Here again no failure could ever be suspected as a latch-up failure caused by stresses within the absolute maximum ratings (although a device destroyed by an upside-down insertion into its socket, with ground and V_{CC} inverted, simulates with its melted metal strips a latch-up failure!).

The most effective way to deliberately trigger a latch-up in one of these parts is to force V_{CC} values well in excess of 7.5 volt.

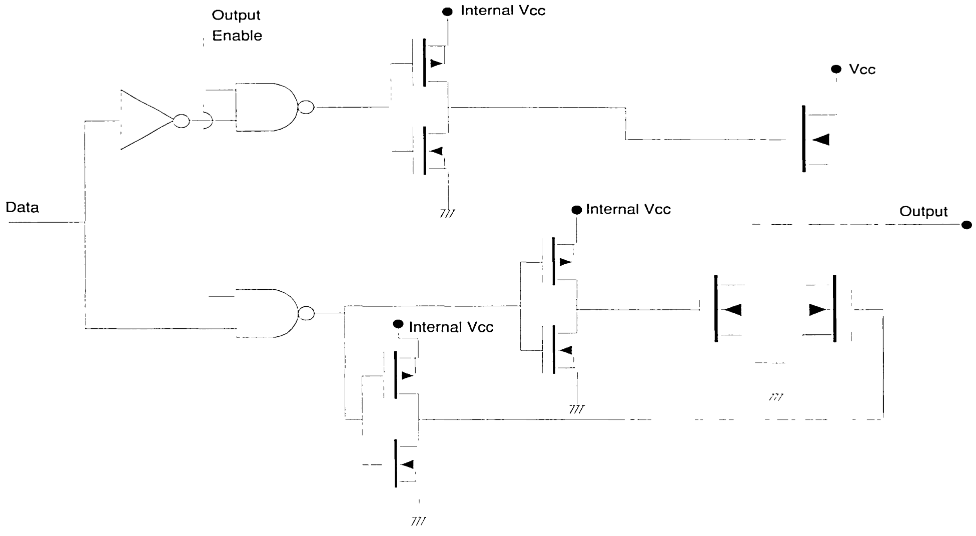
During such an experiment, a 10 Ω series resistor (between the power supply and the V_{CC} terminal) and a quick power-off action, are useful to reduce the risk of a device destruction following the activation of the latch-up.

Conclusion

The several precautions taken during the design of the GAL® products have reached their objective, and an effective immunity from the latch-up could be achieved.

The theory and the experimental evidence both demonstrate that the GAL® can be used without the least concern about the latch-up.

Figure 5. Output Circuits



POWER CONSUMPTION

EVALUATING THE POWER CONSUMPTION OF A GAL®

Due to its characteristic property of being programmable, a GAL® can be configured in one of several different modes.

According to the designer's choice, the logic function that the GAL® performs can vary substantially.

Moreover, the GAL® will absorb a supply current at a level that varies greatly (up to a few times!) as a direct consequence of that same choice.

The ability to predict the the GAL® supply current (or its power dissipation, which is basically the same thing) by calculation, when evaluating design alternatives, may be an important issue.

This note reports a simple but effective (and rather accurate) method for predicting the power dissipation in different situations of:

- architecture programmed into the GAL®;
- supply voltage;
- operating frequency;
- output load.

POWER DISSIPATION IN CMOS CIRCUITS

The power dissipation of a CMOS circuit can be separated into two components:

- DC power dissipation;
- AC power dissipation.

In several practical cases (and always with GAL®) both the DC and the AC components are significant.

Figures 1, 2 and 3 show the I_{CC} consumption of a GAL20V8S-25EB1, a GAL20V8AS-15QB1 and a GAL20V8AS-12HB1, respectively.

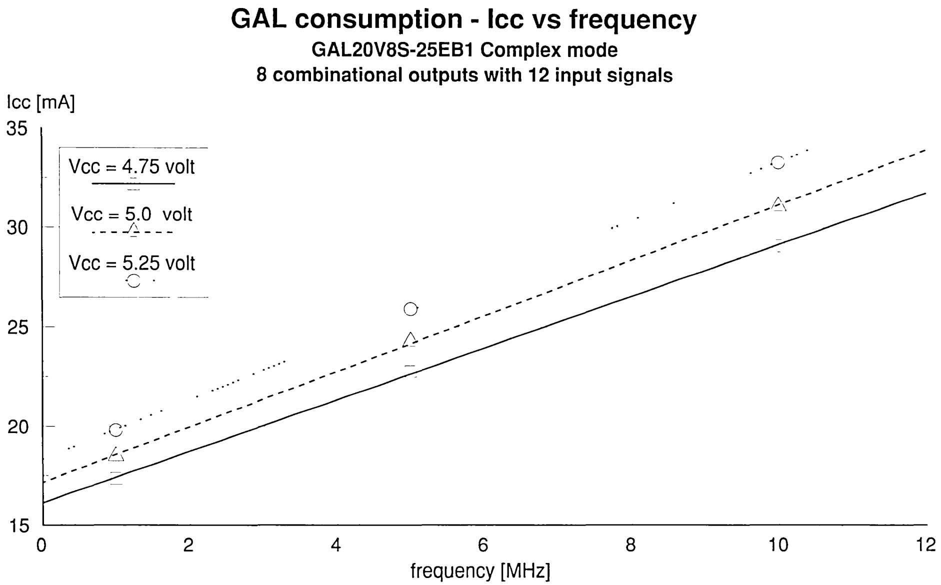
THE DC COMPONENT

In Figures 1, 2 and 3 the power consumption at very low frequencies of the input signals is still significant.

This "DC" component is associated with:

- the analog circuits. In the GAL®:
  - the sense amplifiers;
  - the bias generators (resistive dividers) that bias the analog circuitry;
- the circuitry generating (and switching with) a frequency of their own, i.e. the body bias generator and charge pump.

Figure 1. GAL20V8S-25EB1 Dissipation versus Frequency and Voltage



Ambient temperature = 25 degrees centigrades

Figure 2. GAL20V8AS-15QB1 Dissipation versus Frequency and Voltage

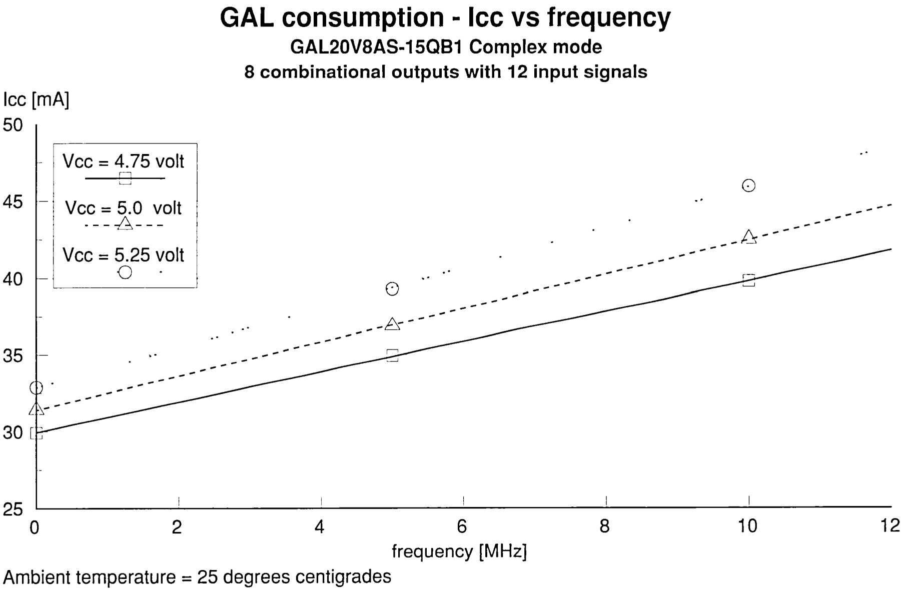
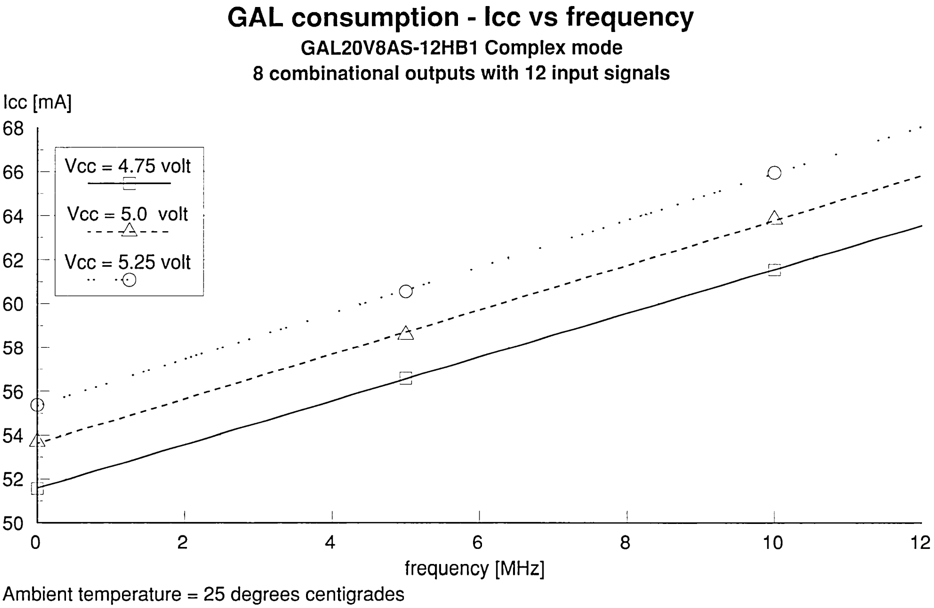


Figure 3. GAL20V8AS-12HB1 Dissipation versus Frequency and Voltage



The Fixed Pedestal

For the purpose of predicting the power consumption of a GAL®, the DC part due to the body biasing and to the other signal biasings can be considered fixed for all chips, with a value of:

2.8 mA

The Input Buffers

It must be emphasised that the actual levels of the input signals may contribute to the GAL® consumption.

Each of the input stages is in fact a CMOS circuit, with the classic Pch/Nch transistors. The dimensions of the two transistors are not perfectly complementary, in order to set the threshold not at $V_{CC}/2$, but rather at about 1.6 volt, because the input specifications require a behaviour of the "TTL compatible" type.

Figures 4, 5 and 6 show the current that an input buffer of each of three popular GAL® types absorbs from V_{CC} when its input is set at different voltage levels.

The data reported in Figure 4, 5 and 6 describe the dissipation in the input buffers only. They can be used to infer the total dissipation associated with an input level below 0.5 volt and above 3.0 volt.

Outside this range of 0.5–3.0 volt input levels, the output level of such buffer is a true CMOS level. However, if the input was set at a voltage close to 1.7 volt, the output level of the input buffer would be itself such as to set other internal stages in a condition of DC dissipation.

This condition must be avoided, and need not be discussed further.

The transient condition that occurs at every switching transition does in fact generate a very quick transient of dissipation in each CMOS stage, but this amount of energy is taken into account when characterising the AC component of the dissipation (see below).

Several consequences of practical significance may be drawn:

- the input stages do not contribute to the overall DC consumption as long as their input levels are within 1 volt from V_{CC} or ground (more precisely, as long as $0 \le V_{IL} < 1$ volt and $(V_{CC} - 1 \text{ volt}) < V_{IH} < V_{CC}$);
- when using input signals switching between 0.5 volt V_{IL} and 3 volt V_{IH} (normal assumption

Figure 4. GAL20V8S-25EB1 Input Buffer Consumption

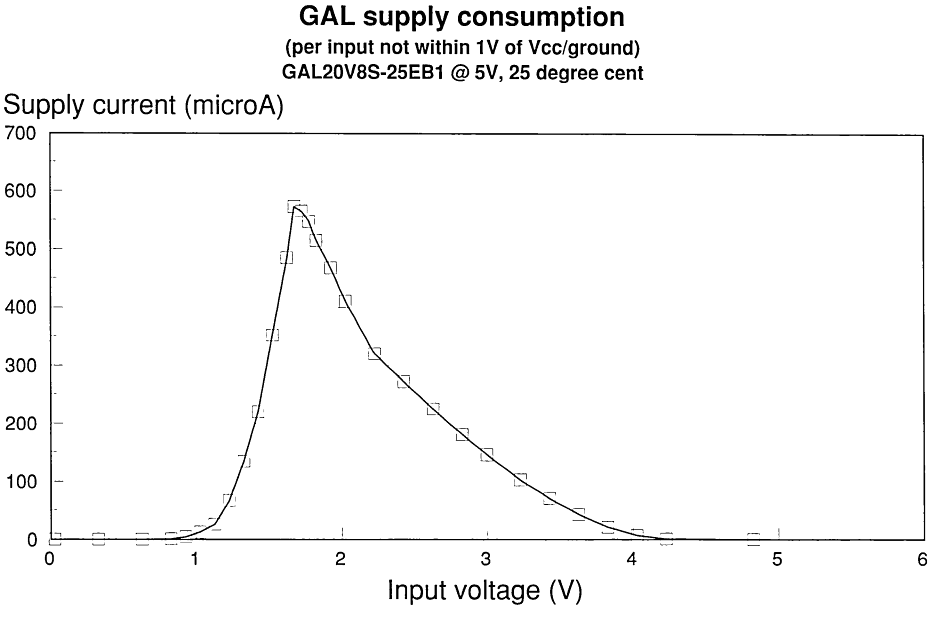


Figure 5. GAL20V8AS-15QB1 Input Buffer Consumption

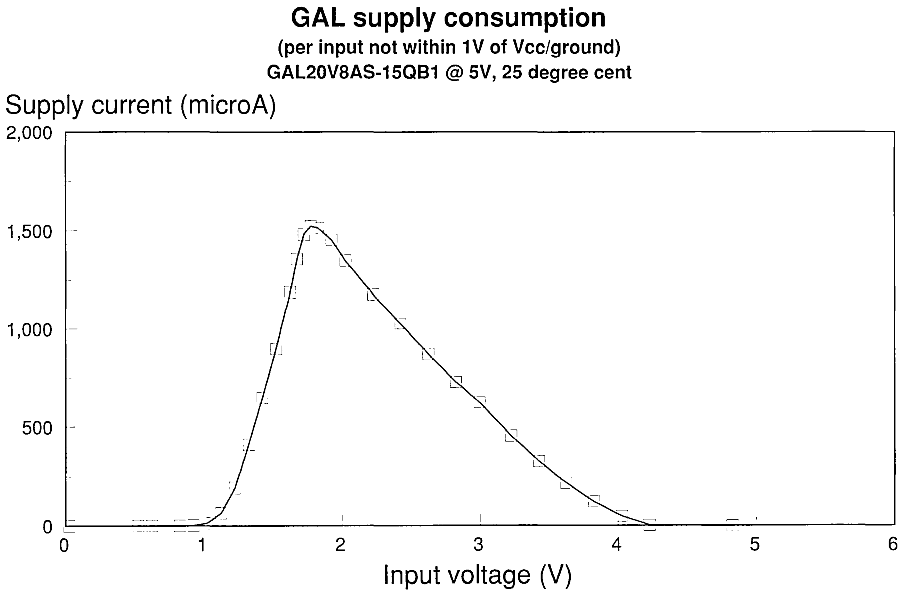
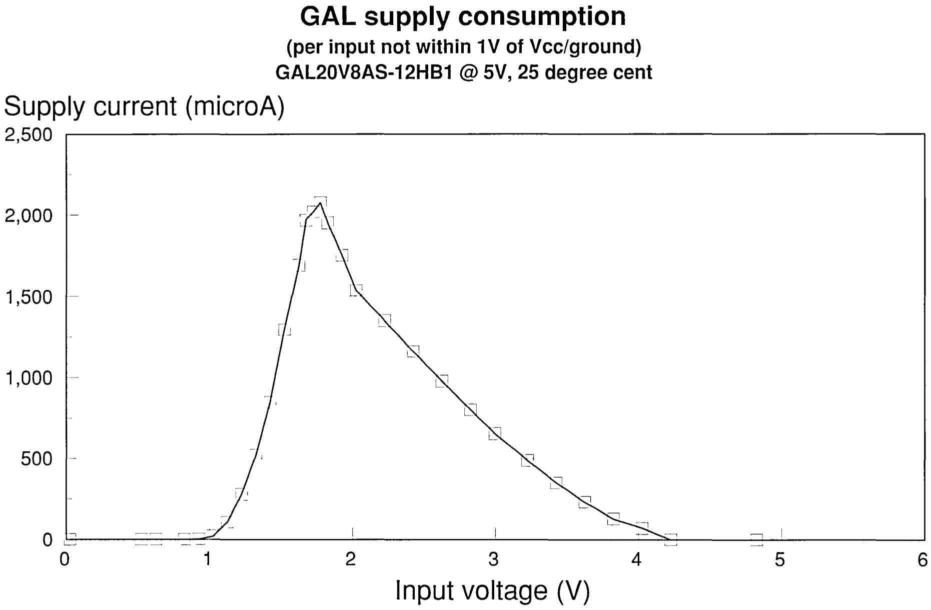


Figure 6. GAL20V8AS-12HB1 Input Buffer Consumption



in the data sheet specifications) the input buffer:

- does not absorb with input @ 0.5 volt;
- absorbs during the transition from 0.5 to 3 volt and from 3 to 0.5 volt according to the AC characterization described in this note (in this note, as well as in the data sheets, the transition times, both low-to-high and high-to-low, are 3 ns);
- absorbs in DC mode when the input signal is at 3 volt continuous (147 µA I_{CC} per input @ 3 volt is the value that can be found in Figure 4 for a GAL20V8S-25EB1).

- where the power dissipation must be minimized, a V_{IH} of 4 volt or more should be preferred (it can be seen that the trade-off with the dissipation of the external circuits on the input parasitic capacitances is more advantageous with lower input frequencies).

As a practical example, let's consider a GAL20V8S-25EB1 with an input switching at 5 MHz.

If the input swings between 0.5 and 3 volt, the power dissipation in the input circuits associated with that pin is:

$$\begin{array}{ll} \text{DC} & 0\,\mu\text{A}\cdot 50\% + 147\,\mu\text{A}\cdot 50\% + \\ \text{AC} & \dfrac{28\,\mu\text{A}}{\text{MHz}}\cdot 5\text{ MHz} = 213.5\,\mu\text{A} \end{array}$$

where the DC contribution is 73.5 µA, i.e. 38%.

If the input signal swings between 0.5 and 4 volt, then only the AC contribution remains, i.e.:

$$\text{AC}\quad \frac{28\,\mu\text{A}}{\text{MHz}}\cdot 5\text{ MHz} = 140\mu\text{A}$$

The advantage of a higher V_{IH} is more apparent if the input is switching at a low frequency, or steadily high.

The Sense Amplifiers

The DC component associated with the DC operation of each sense amplifier is also fixed (but the dependence to the supply voltage should be taken into account), but it is different according to whichever one of the two states the amplifier can assume while decoding the input signals.

The sense amplifier decodes whether one (or more) cell(s) are sinking current from the bit line connected to its input.

(Remember that the sense amplifier physically implements a NOR function of the signals routed to the bit line through the cells that have been written. The programming software writes the cells along the path of the signals inverted with respect to the ones specified by the designer when describing the circuit to implement. The overall result is the implementation of the AND function that each product term in the GAL® has to perform.)

When switching with a 50% duty cycle, the DC part of the consumption of the sense amplifier can be calculated as the average of the consumption in the two states.

In practice several Product Terms are often unused, and may be set in the low state with one cell (the Product Term Disable — PTD) steadily keeping it there. However most programming software implement the disabling of the unused product terms not via the PTD, but by programming as connected all the signal cells of the bit line. If a signal is not at the right level to sink from the bit line, then its negation will, or vice-versa.

This latter approach came into use with the bipolar PAL® that normally do not have provision for the PTD function, but the use of the PTD alone should be preferred with GAL®, because the power consumption can be, though slightly, reduced.

Table 1 gives the typical values for the most popular device types.

Table 1. DC Current Consumption Associated with each Product Term

State	Icc @ 5 volt Vcc [µA]		
	GAL16V8S-*xx*E*yy* and GAL20V8S-*xx*E*yy*	GAL16V8AS-*xx*Q*yy* and GAL20V8AS-*xx*Q*yy*	GAL16V8AS-*xx*H*yy* and GAL20V8AS-*xx*H*yy*
High - no cell sinking	165	451	965
Low - just the PTD (= one cell) sinking	261	350	505
Low - 3 or more cells sinking	280	437	568

For instance, for each Product Term of a GAL20V8S-25EB1:

State	Icc @ 5 volt Vcc
High - no cell sinking	165 µA
Low - 3 or more cells sinking	280 µA

If the PT is made switch at low frequency with duty cycle 50%:

$$\frac{(280\ \mu\text{A} + 165\ \mu\text{A})}{2} = 223\ \mu\text{A}$$

THE AC COMPONENT

In a CMOS circuit this component is generated by the continuous charging and discharging, every switching cycle, of the parasitic capacitances in every node of the circuit, and of the external load capacitances.

When charging a capacitor C from a fixed voltage source V, the energy involved during the transient, no matter how quick the transient is, results in:

- energy taken from the source CV^2 [J]
- energy stored in the capacitor $\frac{CV^2}{2}$ [J]
- energy dissipated in the Pch driving transistor $\frac{CV^2}{2}$ [J]

When, during the next coming transition of the switching cycle, the capacitor is discharged to ground:

- energy taken from the source 0 [J]
- energy given back by the capacitor $\frac{CV^2}{2}$ [J]
- energy dissipated in the Nch driving transistor $\frac{CV^2}{2}$ [J]

The total energy (taken from the supply and dissipated in the circuit), at every switching cycle, is:

$C \cdot V^2$ [J]

As long as the charging time constant of such capacitances in the CMOS circuit is much shorter than half of the switching period (which is always the case in practice), at a frequency of f switching cycles per second, the energy dissipated every second, i.e. the power dissipation, is:

$f \cdot C \cdot V^2$ [W]

The AC dissipation does receive a contribution also from the Pch/Nch pair of each CMOS stage that is in a transient short-circuit at every commutation.

However, the overall result is pretty much equivalent to the effect of an additional parasitic capacitance put across a CMOS stage without short-circuit transient.

Therefore it is convenient to use the above theoretical model in the discussion of the internal AC dissipation.

The Load Capacitances

The formula $f \cdot C \cdot V^2$ can be almost straightforwardly applied to obtain the power dissipation induced by the external capacitances.

Considering that the GAL® output ensures an output swing of about:

from 0 to $(V_{CC} - 1\ \text{volt})$ [V]

it can be seen that the current drawn from the supply with a capacitive load C_{ext} is:

$C_{ext} \cdot (V_{CC} - 1) \cdot f$ [A]

and that the power dissipation is:

$V_{CC} \cdot C_{ext} \cdot (V_{CC} - 1) \cdot f$ [W]

The evaluation of this item depends on the external circuit, is independent from the specific GAL® device, and can easily be calculated by the circuit designer. It is not discussed any further in this note.

Internal AC dissipation

The chip can be partitioned into blocks, that are or are not switching, according to the programmed architecture and to the input signals, in every given implementation.

Every such block can be characterized by an AC dissipation, measured as the current drawn from the supply per every MHz of the switching frequency.

The AC dissipation can then be obtained by adding, for all the blocks that are made switch in the application, the following amount:

$V_{CC} \cdot I_{CC}$ drawn by the block per MHz ·
· frequency in MHz [W]

Different AC Blocks

The internal blocks, that must be separately considered to achieve an accurate prediction of the AC power dissipation, are:

- each switching input (that includes the associated internal row drivers);
- each I/O switching (that includes the associated internal row drivers);

- each switching Product Term (the bit line and the sense amplifier);
- each switching output (that includes the associated feed-back row drivers, when connected to it), distinguishing whether:
  - programmed as a registered* output;
  - programmed as a combinatorial output;

  and also distinguishing whether its output buffer is enabled or not.

The typical values for the different types of SGS-THOMSON devices are shown in Table 2.

From Table 2, one interesting remark can be obtained, if the technological difference between the -S and the -AS types is taken into consideration.

Both the -S and the -AS devices are built with the SGS-THOMSON single-poly technology, for state of the art performances in memory retention.

However the -S devices are built with a single-metal process, to implement the best complexity/performance trade-off for these (relatively!) slow devices.

The -AS parts instead, where a faster switching is required, are built with a double-metal process. This last choice allows the implementation of a faster logic.

It can be seen that the -AS parts, that pay a cost in terms of DC dissipation to achieve high speed performances in the input buffers and the sense amplifiers — with respect to the -S counterparts —, are on the other hand characterized by lower specific AC consumptions because of their double-metal construction.

CONSUMPTION VERSUS SUPPLY VOLTAGE

Once the consumption at 5.0 volt V_{CC} has been evaluated, a simple, but accurate rule can be used if V_{CC} is not 5.0 volt:

- the DC component of the supply current varies +2.2% per every 100 mV above 5.0 volt V_{CC}, or
- the DC component of the supply current varies -2.2% per every 100 mV below 5.0 volt V_{CC}.

The 2.2%/100 mV coefficient can be used for all devices.

Note: the variation of the AC component of the supply current is just directly proportional to the V_{CC} variation.

CONSUMPTION VERSUS TEMPERATURE

The DC component, for all GAL16V8 and GAL20V8, can be estimated to vary:

- -0.23% per every degree centigrade of temperature increase.

For example:

- 5.75% less at 50˚C than it is at 25˚C
- 8.05% more at -10˚C than it is at 25˚C

Note: the AC component can be considered independent from the ambient temperature.

A PRACTICAL EXAMPLE

All throughout this note, a GAL20V8S-25EB1 has been used whenever an example was shown.

When a GAL20V8S-25EB1 is tested to verify the compliance with the datasheet limit of 27 mA for I_{CC} (Maximum Operating Power Supply Current), the pattern of Figure 7 is programmed in it.

* *The signal frequency is the reference for the number of MHz to be used when calculating the supply current. The clock frequency is twice as high as the signal frequency.*

Table 2. AC Consumption (µA/MHz) of the different Blocks @ 5 volt V_{CC}

	Switching Input	Switching I/O	Switching Product Term	Switching Registered* Output		Switching Combinatorial Output	
				Output Not Enabled	Output Enabled	Output Not Enabled	Output Enabled
GAL16V8S-*xx*E*yy* and GAL20V8S-*xx*E*yy*	28	30	4	45	93	15	105
GAL16V8AS-*xx*Q*yy* and GAL20V8AS-*xx*Q*yy*	20	22	3	52	82	10	82
GAL16V8AS-*xx*H*yy* and GAL20V8AS-*xx*H*yy*	20	22	3.5	52	82	10	82

V_{CC} is set at 5.25 volt, the pins 2 and 3 are driven with a square wave at 15 MHz while all the other input pins are kept to ground.

The architecture is set with AC0=SYN=AC1(n)=1, which corresponds to Complex Mode, with combinational Output Logic Macrocells and programmable OE and polarity.

An analysis of the pattern shown in Figure 3 would demonstrate that all outputs are constantly enabled, but only the outputs 22 and 18 switch at 15 MHz.

In order to predict the power consumption, let's first evaluate the DC component of the supply current:

Fixed pedestal	2.8 mA
Inputs 2 and 3 at 0/3.0 volt	
147µA · 50% · 2	0.147 mA
Sense amplifiers	
8 PT0 high 8 · 165 µA	1.32 mA
7 PTs of pins 15, 16, 17, 19, 20, 21	
all high 7 · 6 · 165 µA	6.93 mA
7 PTs of pins 18, 22 switching at 50%	
$7 \cdot 2 \cdot \frac{165\ \mu A + 261\ \mu A}{2}$	2.982 mA
Subtotal DC	14.179 mA

We can now evaluate also the AC contribution:

2 switching inputs	
2 · 28 µA / MHz · 15 MHz	0.84 mA
14 switching PTs	
14 · 4 µA / MHz · 15 MHz	0.84 mA
2 switching combinational outputs	
2 · 105 µA / MHz · 15 MHz	3.15 mA
Subtotal AC	4.83 mA

and the resulting total

Total	18.969 mA

To estimate this consumption at 5.25 volt instead of at 5.0 volt, the DC component of the supply current requires a correction of:

14.179 mA · 2.2% / 100 mV · 2.5 · 100 mV
= 0.78 mA

and the AC component of the supply current a correction of:

4.83 mA · 0.25 V / 5 V = 0.2415 mA

The total current consumption at 5.25 volt V_{CC} adds up to 19.99 mA, which is slightly overestimated when compared with the experimental result of 19.08 mA.

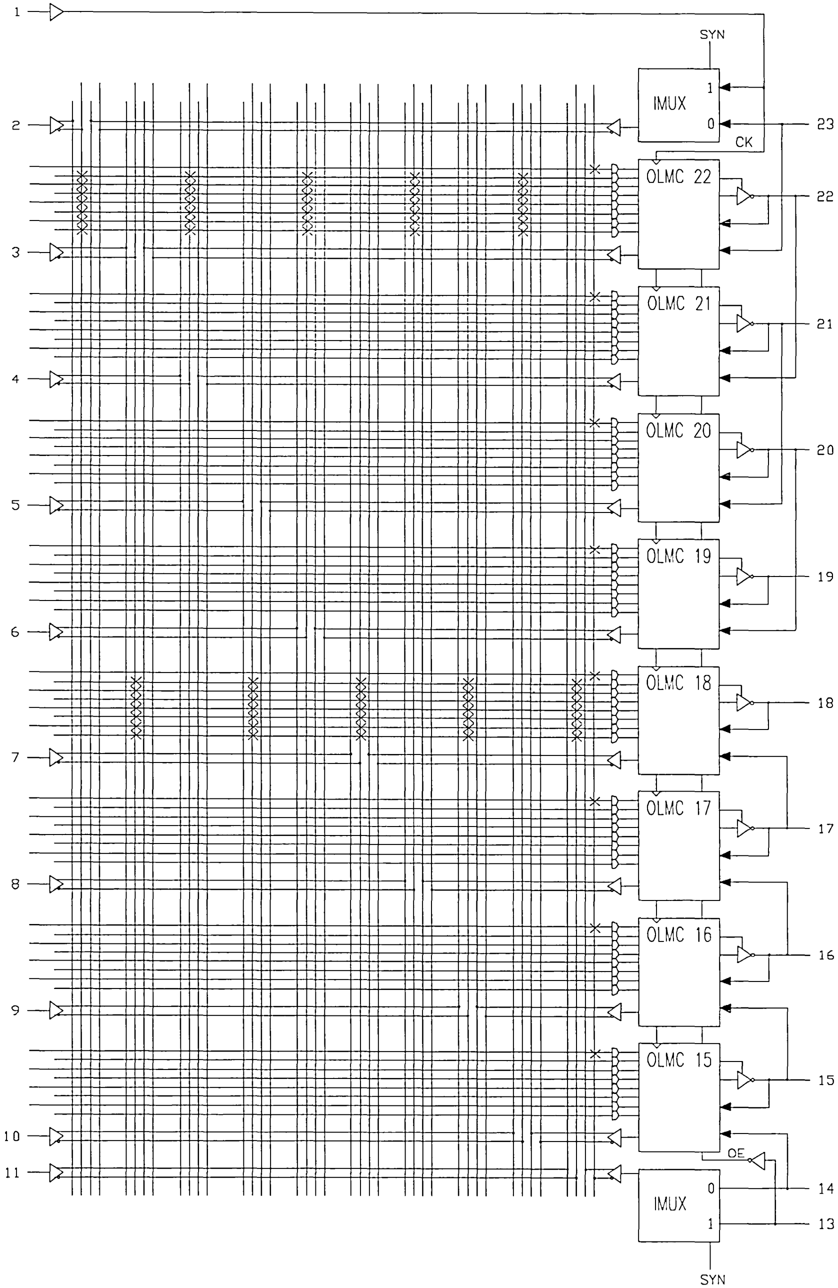
The datasheet limit is 27 mA, valid at 0˚C. The value of 19.99 mA at 25˚C must be corrected by:

14.179 mA – 0.23% / ˚C · – 25˚C =
= 0.815 mA

and becomes 20.81 mA at 0˚C.

It can also be concluded that the typical device considered here is a good "eighth power" part.

Figure 7. GAL20V8S-25EB1 — Pattern for the Dissipation Test



QUALITY AND RELIABILITY

THE RELIABILITY APPROACH

"Architects and designers have for millenia tried to design structures and products for long life. What is new is the movement to quantify reliability."

Dr. Joseph M Juran

In a customer's finished product, semiconductor devices must function normally in a stable manner under the given operational conditions throughout the specified life of the product.

SGS-THOMSON therefore, exercises meticulous care in the design and manufacturing stages and studies the various factors that affect the reliability of semiconductors such as operational and environmental conditions.

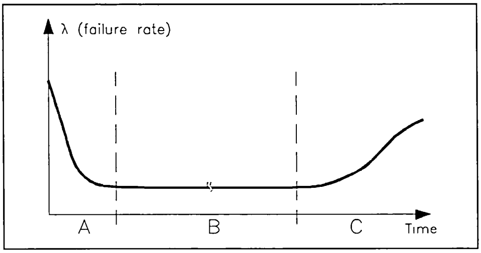
Component reliability is described in quantitative terms. The failure rate distribution of a typical device population follows the familiar bathtub curve shown in Figure 1. This curve is divided into three time zones, the length of which depends on device type and operation stresses. Zone A covers the infant mortality period which, as the name implies, represents the early failure of devices. These failures are usually associated with one or more manufacturing defects and are usually open or short circuits, complete functional failures or seriously degraded performance.

The predominant failure mechanisms are related to assembly defects (weak bonds, contamination, bad seal, etc.). Actions and checks throughout the process allow infant mortality failures to be reduced.

Zone B represents the random failure portion of the distribution curve related to the useful life of the device. This time duration, generally very long (more than hundreds of thousands of hours), depends on the stress (temperature, applied voltage, applied power, circuit complexity, etc.). Failure mechanisms include overstressed devices and residual wafer fab and process defects.

Failure in zone C are wear out failures consisting of catastrophic failures and degraded parameters. They are characterized by a rapid rising failure rate over time as devices wear out physically and electrically.

Figure1. Failure Rate distribution curve



Reliability Testing

Reliability testing is an on-going process adopted to identify and improve reliability performance. Accelerated tests are an important tool for evaluating long term reliability and stability of process and product parameters.

SGS-THOMSON performs rigourous tests throughout production to ensure that devices have the properly designed reliability.

Reliability tests are conducted at wafer and finished product level.

At wafer level, first a complete evaluation of all presently known reliability failure mechanisms is performed on all new processes/technologies through dedicated test patterns. Accelerated tests and targets are then selected for statistical reliability control on production lots.

At finished product level, engineering samples, during design and development stages, are tested to see if their Q&R corresponds to that called for in the design.

Reliability testing is usually performed on a small sample but for long periods or under very accelerated conditions to investigate wearout failures and to determine tolerances and limits of the design. For these tests it is also possible to use the step-stress procedure (e.g. ESD resistance evaluation).

A second type of test is performed periodically during production to check, maintain and improve the assured Q&R levels.

The reliability tests involve both environmental and endurance examination and are performed under conditions more severe than those met in the field. These conditions are chosen to accelerate the occurrence of failures that would appear in actual operation, and care is taken to ensure that the failure modes and mechanisms are unchanged. The data from reliability tests provide an objective tool for product performance evaluation under a wide range of conditions.

When a failure occurs, the SGS-THOMSON engineers conduct an in-depth analysis of the failure mechanism/mode to immediately apply suitable corrective actions.

Reliability testing activity during recent years has been extended to all SGS-THOMSON factories with new and advanced equipment enabling all the plants to perform all the main tests.

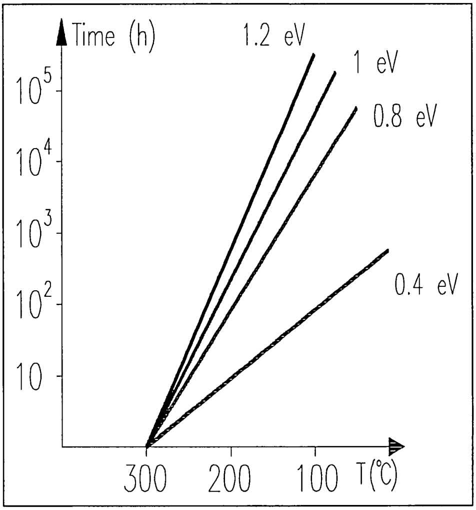
RELIABILITY PREDICTION MODELS

Reliability prediction for semiconductor devices is a very important tool in each stage of product life, from the design to the application by the users.

Since to prove reliability performance in normal application conditions is so expensive and time consuming, it is quite impractical to do it. The component manufacturer's usual approach is to obtain the estimated failure rate, derating the data collected from accelerated tests (generally operating life tests performed at T_j max.) to normal operating conditions.

Figure 2 shows the normalized time/temperature regression for various activation energy values.

Figure 2. Arrhenius Plot



The various lines correspond to the activation energies associated with the different failure mechanisms involved as shown, for typical values, in the following table.

Failure Mode	Activation Energy (eV)	Accelerating Factors
Surface charge	1.0 – 1.05	High temp. bias
Ionic contamination	1.0 – 1.4	High temp. bias
Dielectric defects	0.3 – 0.6	High temp. bias
Electromigration	0.5 – 1.2	High temp. bias
Intermetallic growth	1.0 – 1.05	High temp. bias, Storage
Metal corrosion	0.6 – 0.8	High humidity bias

For equipment manufacturers, on the other hand, knowing the reliability figures (failure rates) means having the possibility of adopting different alternatives during the design phases of the apparatus.

These considerations have increased the role of reliability predictions, and now we are facing an increasing number of customer purchasing specifications including reliability requirement in terms of failure rates.

Naturally, to make reliability predictions, in addition to the knowledge of the functional characteristics of the equipment, it is mandatory to know the reliability of the components used.

Even if sometimes the reliability figures can be obtained by manufacturers and national reliability data banks (thanks to the existence of international standards and to the presence in the semiconductor market of similar technologies), difficulties are present mainly due to:

- different applications
- fast technological evolution
- continuous reliability improvement
- trend to increase the use of VLSI components where failure definition is sometimes difficult.

To combat these difficulties, some reliability prediction models were created.

Current reliability prediction models, such as MIL-HDB-217, CNET and some others, give useful predictions for a wide variety of technologies in the low and medium complexity range. These models are derived from accelerated life tests, screening, burn-in, reliability tests, field experience, device characterization and failure data based on historical results.

MIL-HDBK-217 MODEL (USA)

$$\lambda_p = \pi_L \pi_Q (C_1 \pi_T \pi_V + C_2 \pi_E)\, 10^{-6}\, h^{-1}$$

CNET MODEL (FRANCE)

$$\lambda_p=(C_1\pi_T\pi_V+C_2\pi_B\pi_E\pi_S)\, \pi_Q \pi_L\, 10^{-9}\, h^{-1}$$

OTHER MODELS (i.e. NOKIA[1] - FINLAND)

$$\lambda_p = \lambda_o\ \pi_T \pi_E \pi_L \pi_V \pi_S \pi_F\, h^{-1}$$

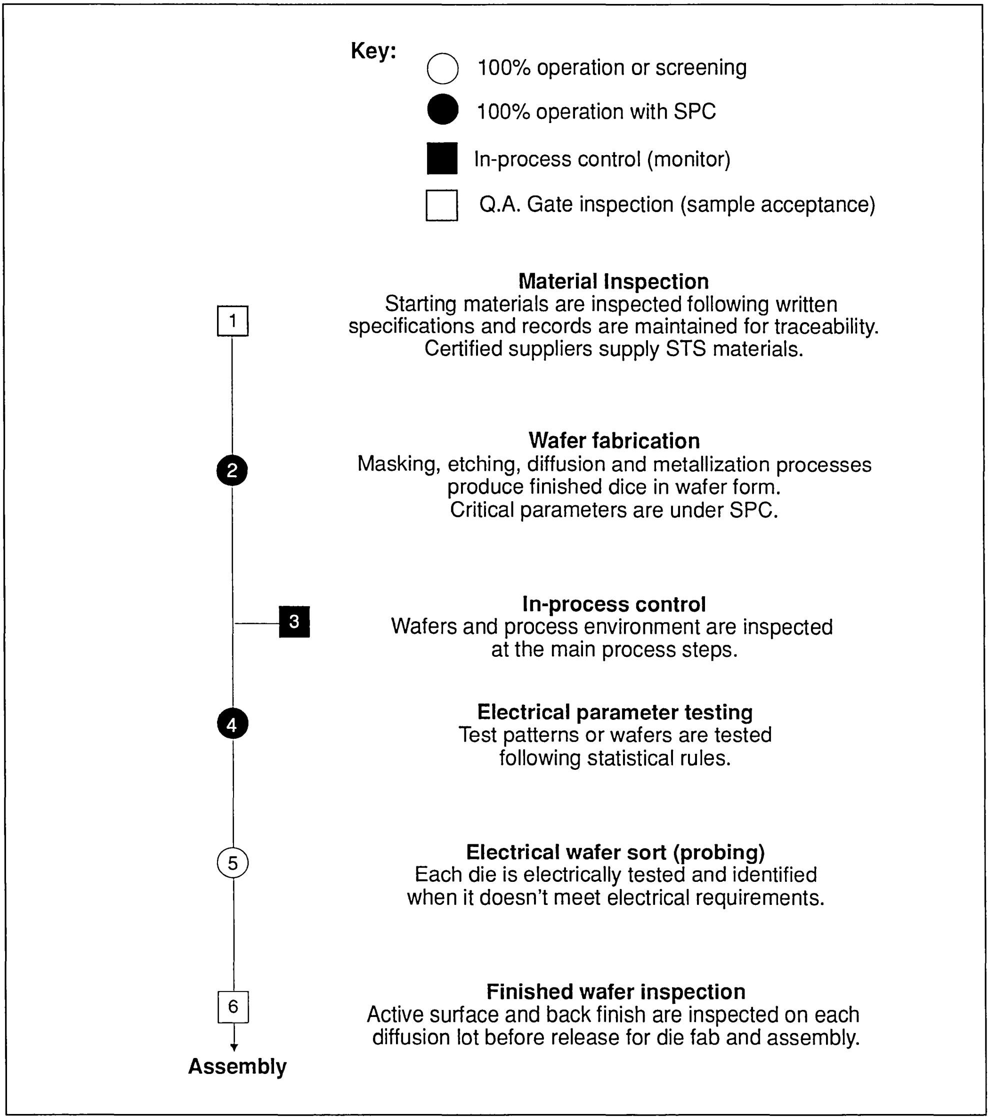
where

λ_p	=	reference failure rate (manufacturing period related)
λ_o	=	device failure rate
π_Q	=	quality factor
π_T	=	temperature acceleration factor
π_V	=	voltage derating stress factor
π_E	=	environmental stress factor
C_1	=	circuit complexity factor
$\pi_{S1}\ \pi_{S2}\ C_2$	=	package complexity factors
π_L	=	learning factor
π_F	=	field experience factor
h	=	hours.

[1] From Nokia Electronic Corporation

The MIL-HDBK-217 is the most popular up to date model and when semiconductor manufacturers data are missing or a reliability comparison is requested, reference to this HDBK represents for users one of the most widely available possibilities, even if the results obtained are generally very conservative and big caution has to be taken in their use.

Wafer Fab Typical Production Process Flow Chart



Key:

○ 100% operation or screening

● 100% operation with SPC

■ In-process control (monitor)

□ Q.A. Gate inspection (sample acceptance)

Material Inspection
Starting materials are inspected following written specifications and records are maintained for traceability. Certified suppliers supply STS materials.

Wafer fabrication
Masking, etching, diffusion and metallization processes produce finished dice in wafer form. Critical parameters are under SPC.

In-process control
Wafers and process environment are inspected at the main process steps.

Electrical parameter testing
Test patterns or wafers are tested following statistical rules.

Electrical wafer sort (probing)
Each die is electrically tested and identified when it doesn't meet electrical requirements.

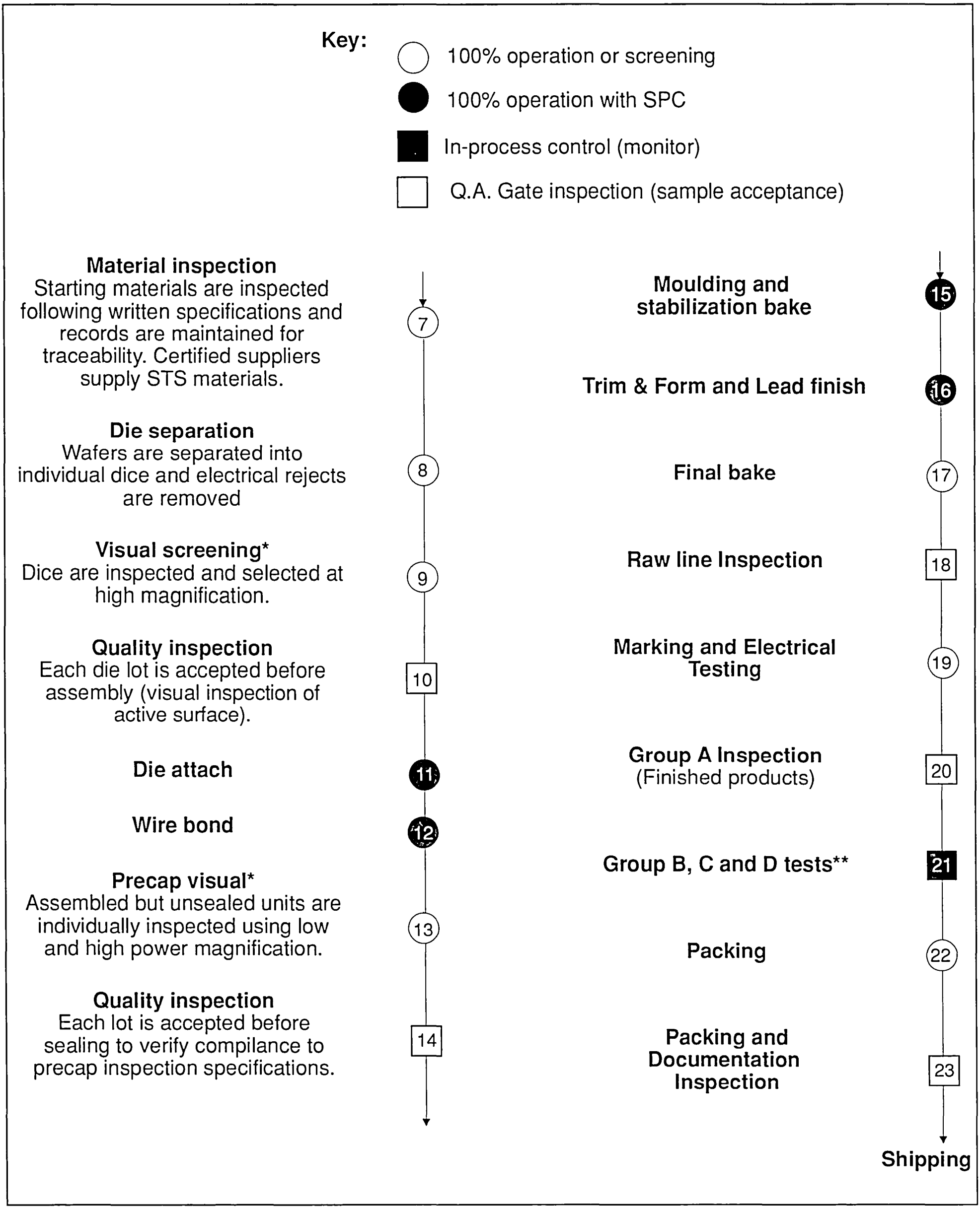
Finished wafer inspection
Active surface and back finish are inspected on each diffusion lot before release for die fab and assembly.

In-Process Control During Wafer Fabrication

The table emphasizes the most important fabrication steps with the relevant SPC measures and/or monitors performed.

Process Steps	In-Process Inspection/Monitoring
Oxidation	– Visual – Thickness – Refractive Index – CV plot (stability of ionic concentration and contamination control)
Deposition: Nitride, Poly Si	– Visual – Thickness – Refractive index – Doping content
Photo Lithography	– Mask and wafer cleanliness – Alignment and focusing accuracy – Critical dimensions
Etching	– Quality of etching and wafer cleanliness – Critical dimensions
Doping by Implant (P, As, B)	– Sheet resistance (dose and implant uniformity)
Doping by Diffusion ($POCl_3$, As)	– Sheet resistance – Thickness – CV plot (stability of ionic concentration and contamination control)
Metallization	–Wafer cleanliness – Visual – SEM (step coverage and film quality) – Thickness – CV plot (stability of ionic concentration and contamination control)
Intermediate and Final Passivation	– Thickness – Doping content – Passivation integrity (density of pinholes and cracks) – Visual
Back Finishing	– Wafer thickness – Back metal thickness – Metal adherence
Electrical Characterization	– Main parameters for active and parasitic structures (e.g. threshold voltage, saturation current, h_{FE}, resistance, capacitances ...)
Wafer Inspection	– Visual (microscope and/or laser surface inspection system)
All Depositions and Photolithography	– Surface scan (to detect and to measure foreign particles)

Assembly Typical Production Process Flow Chart



Key:

○ 100% operation or screening

● 100% operation with SPC

■ In-process control (monitor)

□ Q.A. Gate inspection (sample acceptance)

○ 7 **Material inspection**
Starting materials are inspected following written specifications and records are maintained for traceability. Certified suppliers supply STS materials.

○ 8 **Die separation**
Wafers are separated into individual dice and electrical rejects are removed

○ 9 **Visual screening***
Dice are inspected and selected at high magnification.

□ 10 **Quality inspection**
Each die lot is accepted before assembly (visual inspection of active surface).

● 11 **Die attach**

● 12 **Wire bond**

○ 13 **Precap visual***
Assembled but unsealed units are individually inspected using low and high power magnification.

□ 14 **Quality inspection**
Each lot is accepted before sealing to verify compilance to precap inspection specifications.

● 15 **Moulding and stabilization bake**

● 16 **Trim & Form and Lead finish**

○ 17 **Final bake**

□ 18 **Raw line Inspection**

○ 19 **Marking and Electrical Testing**

□ 20 **Group A Inspection**
(Finished products)

■ 21 **Group B, C and D tests****

○ 22 **Packing**

□ 23 **Packing and Documentation Inspection**

Shipping

* Omitted when the intrinsic quality meets the specified quality level.
** These reliability tests can be performed after step 18 on 100% electrically tested samples (when requested).

In-Process Control During Assembly Process

The table emphasizes the most important fabrication steps with the relevant SPC measures and/or monitors performed.

Process Steps	Tests	Description
11	Dıe Attach	MIL-STD-883 Method 2010 cond B (internal visual) and Method 2019 (die shear strength); CECC 90000
12	Wire Bond	MIL-STD-883 Method 2010 cond. B (internal visual) and Method 2011 cond. D (bond strength); CECC 90000
14	Quality Inspection	MIL-STD-883 Methods 2010 cond. B (internal visual); CECC 90000
15	Moulding and Stabilization Bake	– Visual and temperature process control
16	Trim & Form and Lead Finish	– Dimensions, thickness and contamination control – Solderability control: Aging as per groups B, C and D tests, subgroup 3 215 ± 5°C for 3 ± 0.5 sec. (SMD only) 235 ± 5°C for 2 ± 0.5 sec. 245 ± 5°C for 5 ± 0.5 sec.
17	Final Bake	For SMD only (according to internal specifications)
18	Raw Line Inspection	External Visual MIL-STD-883 Method 2009; CECC 90000 Note: at this step some reliability tests (pressure pot, temperature cycling, life test etc.) are performed as a monitor, generally on a weekly basis, to have fast feedback on process behaviour (Real Time Control Tests)
20	Group A Inspectıon	See relative table
21	Groups B, C and D Tests	Performed on the product family representative types (by rotation); the results are extended to all the other devıces of the same family according to the structure similarity concept
23	Packing and Documentation Inspection	Inspection for: – right quantıty – right type – right boxing – right labelling – right documentation – various

Group A Inspection - Finished Product Acceptance

Subgroup	Parameters	Minimum Sample Size	Acceptance Number
A1	Visual and mechanical inspection	315	0
A2+A3+A4	Cumulative electrical and inoperative mechanical failures	315	0

Notes

- This product acceptance is valid for standard production: for agreed customer programs other samplingplans can be applied.
- Specified temperature ranges according to SGS-THOMSON databooks

Groups B, C and D tests

Every week or every three months on raw line and/or finished products

Subgroup	Test Procedure	MIL-STD-883 Method	CECC 90000 Method	SGS-THOMSON Conditions	Minimum Sample Size	Acceptance Number	Notes (1)
1	Physical dimension	2016	4.3	Data Sheet Drawing	2	0	
2	Resistance to solvents	2015	4.4	1 minute immersion in solvent solution followed by 10 strokes with a hard brush as per MIL-STD method (the procedure shall be repeated 3 times)	4	0	
3	Solderability	2003	4.6.10 Cond 1	215±5°C 3±0.5 sec. 235±5°C 2±0.5 sec. 245±5°C 5±0.5 sec.	22	0	(2)
4	Operating Life Test or end point electrical parameters	1005	4.8	1000 h according to detail spec as per device spec.	45	0	(3)
5	Pressure pot end point electrical parameters	–	–	T_a = 121°C, 2 atm, 240 h min. as per device spec.	22	0	
6	HAST (Highly Accelerated Stress Test) end point electrical parameters	–	4.6.3 Cond2	130°C/85%RH with bias t=150h according to detail specification as per device spec.	22	0	

Notes.

(1) Sample can be increased according to LTPD table, till c=2

(2) Aging of 8h in steam vapor or 16h at 155°C. Soldering temperature of 215°C for SMD only.

(3) Ta such to have T_j=T_j max

Groups B, C and D tests

Every six months on raw line and/or finished products

Subgroup	Test Procedure	MIL-STD-883 Method	CECC 90000 Method	SGS-THOMSON Test Conditions	Minimum Sample Size	Acceptance Number	Notes (1)
1	Lead integrity		4.6.12 (2)	Lead Fatigue Cond. B2 – dual-in-line packages: the leads shall be bent 3 times simultaneously for at least 15° permanent bend, returning then to the original position	22	0	(2)
2	Temperature cycling end point electrical parameters	1010	4.6.8	Cond C; 100 cycles T_a=-65 to + 150°C as per device spec.	22	0	
3	Umidity test end point electrical parameters	–	4.6.3	85°C/85% RH with bias t = 1000h according to detail specification as per device spec.	45	0	

Notes.

(1) Sample can be increased according to LTPD table, till c=2

(2) Not for SMD

Groups B, C and D Reliability Tests(*)

Additional tests performed during qualification of GAL® products:

Test	MIL-STD-883 C	
	Method	Condition
Electrostatic Discharge (ESD) Tolerance	Human Body Model (HBM) 3015.7 Charged Device model (CDM)(*)	C = 100 pF R = 1.5KW ≥ 2000 V Test instrument: Key Tek ZapMaster ≥ 500 V
Latch-up susceptibility	Jedec Standard No. 17 (EIA Jedec JC-40.2 CMOS Logic Standardisation Committee)	See referred Document in "Method" Results available upon request

(*) This is an additional test for GAL® only, and not yet adopted as a Company Standard

SALES OFFICES

EUROPE

DENMARK

2730 HERLEV
Herlev Torv, 4
Tel (45-42) 94 85 33
Telex 35411
Telefax (45-42) 948694

FINLAND

LOHJA SF-08150
Karjalankatu, 2
Tel (358-12) 155 11
Telefax (358-12) 155 66

FRANCE

94253 GENTILLY Cedex
7 - avenue Gallieni - BP 93
Tel (33-1) 47 40 75 75
Telex 632570 STMHQ
Telefax (33-1) 47 40 79 10

67000 STRASBOURG
20, Place des Halles
Tel (33) 88 75 50 66
Telefax (33) 88 22 29 32

GERMANY

6000 FRANKFURT
Gutleutstrasse 322
Tel (49-69) 237492-3
Telex 176997 689
Telefax (49-69) 231957
Teletex 6997689=STVBP

8011 GRASBRUNN
Bretonischer Ring 4
Neukeferloh Technopark
Tel (49-89) 46006-0
Telex 528211
Telefax (49-89) 4605454
Teletex 897107=STDISTR

3000 HANNOVER 51
Rotenburger Strasse 28A
Tel (49-511) 615960
Telex 175118418
Teletex 5118418 CSFBEH
Telefax (49-511) 6151243

5202 HENNEF
Reuther Strasse 1A-C
Tel (49-2242) 6088
(49-2242) 4019/4010
Telefax. (49-2242) 84181

8500 NÜRNBERG 20
Erlenstegenstrasse, 72
Tel (49-911) 59893-0
Telex 626243
Telefax (49-911) 5980701

7000 STUTTGART 31
Mittlerer Pfad 2-4
Tel (49-711) 13968-0
Telex 721718
Telefax (49-711) 8661427

ITALY

20090 ASSAGO (MI)
V.le Milanofiori - Strada 4 - Palazzo A/4/A
Tel. (39-2) 89213 1 (10 linee)
Telex 330131 - 330141 SGSAGR
Telefax (39-2) 8250449

40033 CASALECCHIO DI RENO (BO)
Via R Fucini, 12
Tel. (39-51) 591914
Telex 512442
Telefax (39-51) 591305

00161 ROMA
Via A Torlonia, 15
Tel (39-6) 8443341
Telex 620653 SGSATE I
Telefax (39-6) 8444474

NETHERLANDS

5652 AR EINDHOVEN
Meerenakkerweg 1
Tel (31-40) 550015
Telex 51186
Telefax (31-40) 528835

SPAIN

08021 BARCELONA
Calle Platon, 6 4th Floor, 5th Door
Tel (34-3) 4143300-4143361
Telefax (34-3) 2021461

28027 MADRID
Calle Albacete, 5
Tel (34-1) 4051615
Telex 46033 TCCEE
Telefax (34-1) 4031134

SWEDEN

S-16421 KISTA
Borgarfjordsgatan, 13 - Box 1094
Tel (46-8) 7939220
Telex 12078 THSWS
Telefax (46-8) 7504950

SWITZERLAND

1218 GRAND-SACONNEX (GENEVA)
Chemin Francois-Lehmann, 18/A
Tel (41-22) 7986462
Telex 415493 STM CH
Telefax (41-22) 7984869

UNITED KINGDOM and EIRE

MARLOW, BUCKS
Planar House, Parkway
Globe Park
Tel (44-628) 890800
Telex 847458
Telefax (44-628) 890391

AMERICAS

BRAZIL

05413 SÃO PAULO
R Henrique Schaumann 286-CJ33
Tel (55-11) 883-5455
Telex (391)11-37988 "UMBR BR"
Telefax (55-11) 282-2367

U.S.A.

NORTH & SOUTH AMERICAN
MARKETING HEADQUARTERS
1000 East Bell Road
Phoenix, AZ 85022
(1-602) 867-6100

SALES COVERAGE BY STATE

ALABAMA
Huntsville - (205) 533-5995

ARIZONA
Phoenix - (602) 867-6217

CALIFORNIA
Santa Ana - (714) 957-6018
San Jose - (408) 452-8585

COLORADO
Boulder (303) 449-9000

ILLINOIS
Schaumburg - (708) 517-1890

INDIANA
Kokomo - (317) 455-3500

MASSACHUSETTS
Lincoln - (617) 259-0300

MICHIGAN
Livonia - (313) 462-4030

NEW JERSEY
Voorhees - (609) 772-6222

NEW YORK
Poughkeepsie - (914) 454-8813

NORTH CAROLINA
Raleigh - (919) 787-6555

TEXAS
Carrollton - (214) 466-8844

FOR RF AND MICROWAVE
POWER TRANSISTORS CONTACT
THE FOLLOWING REGIONAL
OFFICE IN THE U S.A

PENNSYLVANIA
Montgomeryville - (215) 361-6400

ASIA / PACIFIC

AUSTRALIA

NSW 2027 EDGECLIFF
Suite 211, Edgecliff centre
203-233, New South Head Road
Tel (61-2) 327 39 22
Telex 071 126911 TCAUS
Telefax (61-2) 327 61 76

HONG KONG

WANCHAI
22nd Floor - Hopewell centre
183 Queen's Road East
Tel (852-5) 8615788
Telex 60955 ESGIES HX
Telefax (852-5) 8656589

INDIA

NEW DELHI 110001
LiasonOffice
62, Upper Ground Floor
World Trade Centre
Barakhamba Lane
Tel (91-11) 3715191
Telex 031-66816 STMI IN
Telefax (91-11) 3715192

MALAYSIA

PULAU PINANG 10400
4th Floor - Suite 4-03
Bangunan FOP-123D Jalan Anson
Tel (04) 379735
Telefax (04) 379816

KOREA

SEOUL 121
8th floor Shinwon Building
823-14, Yuksam-Dong
Kang-Nam-Gu
Tel (82-2) 553-0399
Telex SGSKOR K29998
Telefax (82-2) 552-1051

SINGAPORE

SINGAPORE 2056
28 Ang Mo Kio - Industrial Park 2
Tel (65) 4821411
Telex RS 55201 ESGIES
Telefax (65) 4820240

TAIWAN

TAIPEI
12th Floor
325, Section 1 Tun Hua South Road
Tel (886-2) 755-4111
Telex 10310 ESGIE TW
Telefax (886-2) 755-4008

JAPAN

TOKYO 108
Nisseki - Takanawa Bld 4F
2-18-10 Takanawa
Minato-Ku
Tel (81-3) 3280-4121
Telefax· (81-3) 3280-4131

Information furnished is believed to be accurate and reliable However, SGS-THOMSON Microelectronics assumes no responsability for the consequences of use of such information nor for any infringement of patents or other rights of third parties which may result from its use No license is granted by implication or otherwise under any patent or patent rights of SGS-THOMSON Microelectronics Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all informations previously supplied. SGS-THOMSON Microelectronics products are not authorized for use as critical components in life support devices or system without express written approval of SGS-THOMSON Microelectronics.

SGS-THOMSON Microelectronics GROUP OF COMPANIES
Australia - Brazil - France - Germany - Hong Kong - Italy - Japan - Korea - Malaysia - Malta - Morocco - The Netherlands -
Singapore - Spain - Sweden - Switzerland - Taiwan - United Kingdom - U.S A.