NxVL™ System Controller Databook

PRELIMINARY
March 4, 1994

NexGen™ Microproducts, Inc.
1623 Buckeye Drive
Milpitas, CA 95035

Order # NxDOC-DB002-01-W

The goal of this databook is to enable our customers to make informed purchase decisions and to design systems around our described products. Every effort is made to provide accurate information in support of these goals. However, representations made by this databook are not intended to describe the internal logic and physical design. Wherever product internals are discussed, the information should be construed as conceptual in nature. No presumptions should be made about the internal design based on this document. Information about the internal design of NexGen products is provided via nondisclosure agreement ("NDA") on a need to know basis.

The material in this document is for information only and is subject to change without notice. NexGen reserves the right to make changes in the product specification and design without reservation and without notice to its users. THIS DOCUMENT DOES NOT CONSTITUTE A WARRANTY OF ANY KIND WITH RESPECT TO THE NEXGEN INC. PRODUCTS, AND NEXGEN INC. SHALL NOT BE LIABLE FOR ANY ERRORS THAT APPEAR IN THIS DOCUMENT.

All purchases of NexGen products shall be subject to NexGen's standard terms and conditions of sale. THE WARRANTIES AND REMEDIES EXPRESSLY SET FORTH IN SUCH TERMS AND CONDITIONS SHALL BE THE SOLE WARRANTIES AND THE BUYER'S SOLE AND EXCLUSIVE REMEDIES, AND NEXGEN INC. SPECIFICALLY DISCLAIMS ANY AND ALL OTHER WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING THE IMPLIED WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE, AGAINST INFRINGEMENT AND OF MERCHANTABILITY. No person is authorized to make any other warranty or representation concerning the performance of the NexGen products. In particular, NexGen's products are not specifically designed, manufactured or intended for sale as components for the planning, design, construction, maintenance, operation or use of any nuclear facility or other ultra-hazardous activity, and neither NexGen nor its suppliers shall have any liability with respect to such use

Trademark Acknowledgments

NexGen, Nx586, Nx587, RISC86, NexBus, NxPCI, and NxVL are trademarks of NexGen Microproducts, Inc..

IBM, AT, and PS/2 are registered trademarks of International Business Machines, Inc. Intel is a registered trademark of Intel Corporation. i386, i387, i486 and Pentium are trademarks of Intel Corporation. Tri-state is a registered trademark of National Semiconductor Corporation. VL-Bus is a trademark of Video Electronics Standards Association.

Restricted Rights and Limitations

Use, duplication, or disclosure by the Government is subject to restrictions set forth in subparagraph (c)(1)(ii) of the Rights in technical Data and Computer Software clause at 252.2777-7013

Contents

Figures

Preface

This databook covers the NxVL™ systems logic, the systems controller for the Nx586™ and Nx587™ processors. This book is written for system designers considering the use of NexGen™ products in their designs. We assume an experienced audience, familiar not only with system design conventions but also with the x86 architecture. The *Glossary* at the end of the book defines NexGen's terminology, and the *Index* gives quick access to the subject matter.

Notation

The following notation and conventions are used in this book:

Chip and Bus Names

- *NxVL*—The NxVL system controller chip described in this book.
- *Processor or CPU*—The Nx586 processor chip described in the *Nx586 Processor and Nx587 Floating Point Coprocessor Databook.*
- *Floating Point Coprocessor or NP*—The Nx587 floating-point unit chip described in the *Nx586 Processor and Nx587 Floating Point Coprocessor Databook.*
- *NexBus™System Bus*—The Nx586 processor bus, including its multiplexed address/status and data bus (NxAD<63:0>) and related control signals.

Signals *and* Timing Diagrams

- *Active-Low Signals*—Signal names that are followed by an asterisk, such as ALE*, indicate active-low signals. They are said to be "asserted" in their low-voltage state and "negated" in their high-voltage state.
- *Bus Signals*—In signal names, the notation <*n:m*> represents bits *n* through *m* of a bus.
- *Reserved Bits and Signals*—Signals or bus bits marked "reserved" must be driven inactive or left unconnected, as indicated in the signal descriptions. These bits and signals are reserved by NexGen for future implementations. When software reads registers with reserved bits, the reserved bits must be

masked. When software writes such registers, it must first read the register and change only the non-reserved bits before writing back to the register.

- *Source*—In timing diagrams, the left-hand column indicates the "Source" of each signal. This is the device or logic that generates the signal. When signals are driven by multiple sources, all sources are shown, in the order in which they drive the signal. In some timing diagrams, bus signals take on different names as outputs cross buses through transceivers or are ORed in group-signal logic. In these cases, the signal source is shown with a subscript, where the subscript indicates the device or logic that originally caused the change in the signal.
- *Tri-state®*—In timing diagrams, signal ranges that are high impedance are shown as a straight horizontal line half-way between the high and low level.
- *Invalid and Don't Care*—In timing diagrams, signal ranges that are invalid or don't care are filled with a screen pattern.

Data

- *Quantities*—A word is two bytes (16 bits), a dword or doubleword is four bytes (32 bits), and a qword or quadword is eight bytes (64 bits).
- Addressing—Memory is addressed as a series of bytes on eight-byte (64-bit) boundaries, in which each byte can be separately enabled.
- *Abbreviations*—The following notation is used for bits and bytes:

Bits	b	as in "64b/qword"
Bytes	B	as in "32B/block"
kilo	k	as in "4kB/page"
Mega	M	as in "1Mb/sec"
Giga	G	as in "4GB of memory space"

- *Little* Endian Convention—The byte with the address xx...xx00 is in the least-significant byte position (little end). In byte diagrams, bit positions are numbered from right to left: the little end is on the right and the big end is on the left. Data structure diagrams in memory show small addresses at the bottom and high addresses at the top. When data items are "aligned," bit notation on a 64-bit data bus maps directly to bit notation in 64-bit-wide memory. Because byte addresses increase from right to left, strings appear in reverse order when illustrated.
- *Bit Ranges*—In a range of bits, the highest and lowest bit numbers are separated by a colon, as in <63:0>.
- *Bit Values*—Bits can either be *set* to 1 or *cleared* to 0.
- *Hexadecimal and Binary Numbers*—Unless the context makes interpretation clear, hexadecimal numbers are followed by an *h*, binary numbers are followed by a *b*, and decimal numbers are followed by a *d*.

Related Publications

The following books discuss various aspects of computer architecture, that may be useful for your understanding of NexGen products:

NexGen Products

- *Nx586 Processor and Nx587 Coprocessor Databook*, NexGen, Milpitas, CA, Tel: (408) 435-0202.

VL-Bus Architecture

- *VL-Bus™ Proposal*, Video Electronics Standards Association (VESA), San Jose, CA, 1992. Tel: (408) 435-0333.

ISA-Bus Architecture

- Edward Solari, *AT Bus Design, IEEE P996 Compatible*, Annabooks, San Diego, CA, 1990.

Other Products and Architecture

- *82C206 Integrated Peripheral Controller Data Sheet*, OPTi, Tel: (408) 980-8178; Chips and Technologies, Tel: (408) 434-0600; UMC; and others.

NxVL Features and Signals

The NxVL system controller provides most of the systems logic required to implement a high performance Nx586 based, VL-bus, PC/AT compatible system. It functions as a main-memory controller, NexBus Arbiter for a single Nx586 processor, and the system-logic interface (alternate-bus interface) between the NexBus and two other system buses—the 32-bit VL bus and the 16-bit ISA bus. It arbitrates accesses by masters on any bus to slaves on any other bus, and mediates the operations of an Integrated Peripherals Controller (IPC). The NxVL provides the following features:

- **Bus Interface**—Implements bus-crossing accesses between the Nx586/587 processors and the VL-bus or ISA bus (acts as the *alternate-bus interface* for the Nx586 processor).
- **Supports VL-Bus™ and ISA-Bus Devices**—Supports the standard VESA VL-bus and ISA-bus signals.
- **Global Bus Arbiter**—Arbitrates between the Nx586 processor, VL-bus masters, and ISA-bus masters. Acts as the NexBus Arbiter.
- **Main-Memory Controller**—Supports high-performance main memory acesses from 2MB to 256MB, with NexBus prefetch queue and read-on-the-fly write queue.
- **Implements Cache Coherency**—Implements NexBus MESI modified write-once cache coherency bus-snooping protocol.
- **Support for Integrated Peripheral Control**—Interfaces to 82C206 peripheral controller.
- **VL-Bus Decoupled from NexBus**—Operations on the VL bus and its derivative, the ISA bus, occur independently of operations on the NexBus.
- **Multiple Bus Master and Data Bursting** —The NxVL is capable of supporting, and arbitrating VL-bus Masters. In addition, the NxVL supports the VL-Bus data bursting protocols.

All NexBus signals are synchronous to the NexBus clock (CLK). They transition at the rising edge of the clock, except for the asynchronous signals, NMI* and GATEA20. All bi-directional NexBus signals are floated unless they are needed, as specified in the *Bus Operation* chapter. The normal state for all reserved bits is high.

Figure 1 shows the signal organization for the NxVL systems controller. Three types of signals deserve special mention:

- *NexBus Addresses and Data*—Address/status and data phases are multiplexed on the NexBus. This is a buffered bus that can be interfaced to NexBus devices through transceivers, for which control signals are provided on the Nx586 processor. In systems that use the NxVL product, the NexBus transceivers are optional; the NxVL chip has internal transceivers.
- *Group Signals*—There are several group signals on the NexBus, denoted by signal names beginning with the letter "G." In systems with multiple devices on the NexBus, group signals are generated by backplane logic. With NxVL, these group signals are generated within the NxVL device itself, rather than by external glue logic.
- *NexBus Arbiter and Bus Interface*—Several NexBus signals are used for NexBus arbitration. The NxVL serves as the processor's NexBus Arbiter and the interface to the VL and ISA buses, as well as for memory control and other system tasks. Since there are no other devices in the NexBus arbitration, the processor by default gets back-to-back use of the bus when no device on another bus needs access. See the *Bus Operations* chapter.

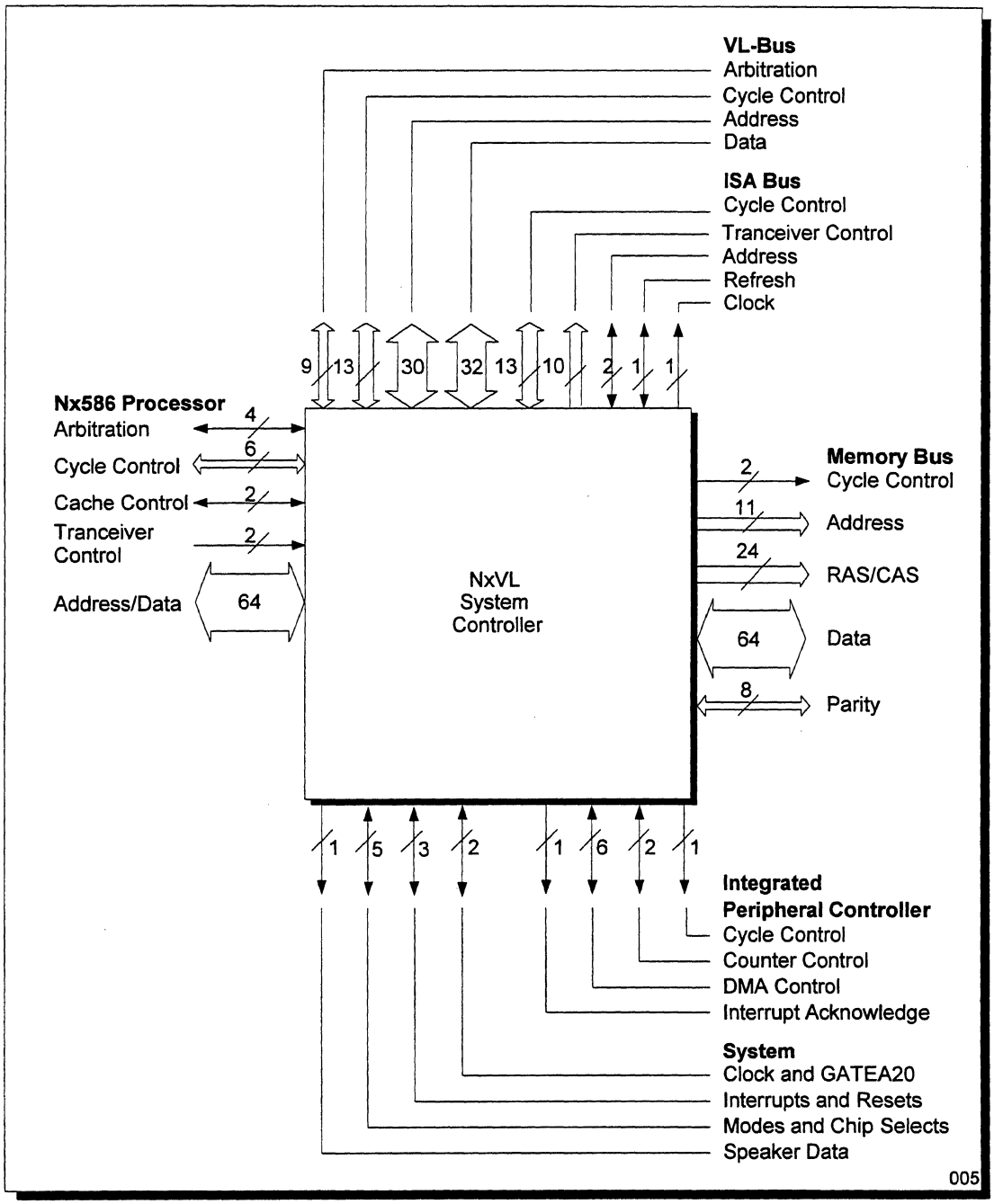


Figure 1 NxVL Signal Organization

NxVL Pinouts by Signal Names

Pin	Type	Signal
11	I	ADSTB16
104	I	ADSTB8
140	O	AEN
269	I	AEN16*
157	I	AEN8*
105	I	ALE*
217	I	AREQ*
158	O	ASRTC
195	O	BALE
103	I/O	BLAST*
57	I	BCLKSEL
268	O	BRDY*
75	I	BUFMODE
346	O	CASA<0>*
187	O	CASA<1>*
297	O	CASA<2>*
244	O	CASA<3>*
294	O	CASA<4>*
241	O	CASA<5>*
32	O	CASA<6>*
129	O	CASA<7>*
131	O	CASB<0>*
80	O	CASB<1>*
296	O	CASB<2>*
186	O	CASB<3>*
127	O	CASB<4>*
183	O	CASB<5>*
239	O	CASB<6>*
77	O	CASB<7>*
7	I	CLK
106	I	DCL*
74	O	DRAMBFDIR
28	O	FLASHCS*
277	O	GALE
101	O	GATE2
159	O	GATEA20
161	O	GBLKNBL
279	I/O	GNT*
322	O	GTAL
223	O	GXACK
160	O	GXHLD
321	I	HHOLD
270	O	HLDA
155	O	INTA*
194	I	IOCHCHK*
214	I/O	IOCHRDY
197	I	IOCS16*
304	I/O	IOR*
55	I/O	IOW*
211	O	ISABCLK
89	O	KBDCS*
8	I/O	LADS*
50	I/O	LBADR<2>
152	I/O	LBADR<3>
56	I/O	LBADR<4>
99	I/O	LBADR<5>
264	I/O	LBADR<6>
209	I/O	LBADR<7>
153	I/O	LBADR<8>
51	I/O	LBADR<9>
98	I/O	LBADR<10>
315	I/O	LBADR<11>
5	I/O	LBADR<12>
208	I/O	LBADR<13>
151	I/O	LBADR<14>
263	I/O	LBADR<15>
49	I/O	LBADR<16>
207	I/O	LBADR<17>
97	I/O	LBADR<18>
4	I/O	LBADR<19>
262	I/O	LBADR<20>
150	I/O	LBADR<21>
356	I/O	LBADR<22>
48	I/O	LBADR<23>
2	I/O	LBADR<24>
260	I/O	LBADR<25>
46	I/O	LBADR<26>
148	I/O	LBADR<27>
94	I/O	LBADR<28>
204	I/O	LBADR<29>
259	I/O	LBADR<30>
1	I/O	LBADR<31>
42	I/O	LBDATA<0>
141	I/O	LBDATA<1>
90	I/O	LBDATA<2>
198	I/O	LBDATA<3>
255	I/O	LBDATA<4>
142	I/O	LBDATA<5>
43	I/O	LBDATA<6>
91	I/O	LBDATA<7>
308	I/O	LBDATA<8>
44	I/O	LBDATA<9>
256	I/O	LBDATA<10>
143	I/O	LBDATA<11>
92	I/O	LBDATA<12>
357	I/O	LBDATA<13>
309	I/O	LBDATA<14>
200	I/O	LBDATA<15>
201	I/O	LBDATA<16>
202	I/O	LBDATA<17>
257	I/O	LBDATA<18>
258	I/O	LBDATA<19>
146	I/O	LBDATA<20>
93	I/O	LBDATA<21>
310	I/O	LBDATA<22>
203	I/O	LBDATA<23>
206	I/O	LBDATA<24>
96	I/O	LBDATA<25>
47	I/O	LBDATA<26>
149	I/O	LBDATA<27>
261	I/O	LBDATA<28>
95	I/O	LBDATA<29>
205	I/O	LBDATA<30>
312	I/O	LBDATA<31>
210	I/O	LBD/C*
265	I/O	LBE<0>*
6	I/O	LBE<1>*
100	I/O	LBE<2>*
154	I/O	LBE<3>*
52	I/O	LBM/IO*
320	I	LBS16*
266	I/O	LBW/R*
102	I	LDEV<0>*
212	I	LDEV<1>*
196	I	LDEV<2>*
10	O	LGNT<0>*
213	O	LGNT<1>*
237	O	LGNT<2>*
272	I	LOCK*
9	I/O	LRDY*
54	I	LREQ<0>*
156	I	LREQ<1>*
125	I	LREQ<2>*
243	O	MA<0>
182	O	MA<1>
345	O	MA<2>
242	O	MA<3>
130	O	MA<4>
33	O	MA<5>
295	O	MA<6>
185	O	MA<7>
78	O	MA<8>
31	O	MA<9>
79	O	MA<10>
39	I	MASTER*
303	I/O	MD<0>
250	I/O	MD<1>
38	I/O	MD<2>
193	I/O	MD<3>
249	I/O	MD<4>
37	I/O	MD<5>
84	I/O	MD<6>
248	I/O	MD<7>
348	I/O	MD<8>
134	I/O	MD<9>
299	I/O	MD<10>
82	I/O	MD<11>
133	I/O	MD<12>
189	I/O	MD<13>
245	I/O	MD<14>
188	I/O	MD<15>
351	I/O	MD<16>
137	I/O	MD<17>
302	I/O	MD<18>
85	I/O	MD<19>
136	I/O	MD<20>
192	I/O	MD<21>
349	I/O	MD<22>
135	I/O	MD<23>
247	I/O	MD<24>
35	I/O	MD<25>
190	I/O	MD<26>
246	I/O	MD<27>
34	I/O	MD<28>
81	I/O	MD<29>
298	I/O	MD<30>
132	I/O	MD<31>
290	I/O	MD<32>
124	I/O	MD<33>
339	I/O	MD<34>
179	I/O	MD<35>
26	I/O	MD<36>
235	I/O	MD<37>
178	I/O	MD<38>
25	I/O	MD<39>
287	I/O	MD<40>
121	I/O	MD<41>
336	I/O	MD<42>
176	I/O	MD<43>
23	I/O	MD<44>
69	I/O	MD<45>
119	I/O	MD<46>
174	I/O	MD<47>
27	I/O	MD<48>
236	I/O	MD<49>
73	I/O	MD<50>
289	I/O	MD<51>
123	I/O	MD<52>
72	I/O	MD<53>
288	I/O	MD<54>
122	I/O	MD<55>

Figure 2 NxVL Pin List, By Signal Name

Pin	Type	Signal
24	I/O	MD<56>
233	I/O	MD<57>
70	I/O	MD<58>
286	I/O	MD<59>
120	I/O	MD<60>
232	I/O	MD<61>
175	I/O	MD<62>
231	I/O	MD<63>
306	I/O	MEMCS16*
252	I/O	MEMR*
41	I/O	MEMW*
300	I/O	MPAR<0>
191	I/O	MPAR<1>
36	I/O	MPAR<2>
83	I/O	MPAR<3>
337	I/O	MPAR<4>
71	I/O	MPAR<5>
234	I/O	MPAR<6>
177	I/O	MPAR<7>
273	O	NMI*
271	I	NREQ*
282	I/O	NxAD<0>
334	I/O	NxAD<1>
67	I/O	NxAD<2>
171	I/O	NxAD<3>
168	I/O	NxAD<4>
169	I/O	NxAD<5>
115	I/O	NxAD<6>
21	I/O	NxAD<7>
280	I/O	NxAD<8>
116	I/O	NxAD<9>
172	I/O	NxAD<10>
118	I/O	NxAD<11>
166	I/O	NxAD<12>
167	I/O	NxAD<13>
112	I/O	NxAD<14>
17	I/O	NxAD<15>
113	I/O	NxAD<16>
20	I/O	NxAD<17>
225	I/O	NxAD<18>
226	I/O	NxAD<19>
218	I/O	NxAD<20>
109	I/O	NxAD<21>
274	I/O	NxAD<22>
325	I/O	NxAD<23>
165	I/O	NxAD<24>
276	I/O	NxAD<25>
16	I/O	NxAD<26>
164	I/O	NxAD<27>
60	I/O	NxAD<28>
14	I/O	NxAD<29>
59	I/O	NxAD<30>
108	I/O	NxAD<31>
285	I/O	NxAD<32>
230	I/O	NxAD<33>
284	I/O	NxAD<34>
173	I/O	NxAD<35>
229	I/O	NxAD<36>
114	I/O	NxAD<37>
224	I/O	NxAD<38>
66	I/O	NxAD<39>
22	I/O	NxAD<40>
117	I/O	NxAD<41>
68	I/O	NxAD<42>
227	I/O	NxAD<43>
327	I/O	NxAD<44>
222	I/O	NxAD<45>
63	I/O	NxAD<46>
62	I/O	NxAD<47>
65	I/O	NxAD<48>
281	I/O	NxAD<49>
19	I/O	NxAD<50>
170	I/O	NxAD<51>
12	I/O	NxAD<52>
162	I/O	NxAD<53>
219	I/O	NxAD<54>
15	I/O	NxAD<55>
111	I/O	NxAD<56
110	I/O	NxAD<57>
221	I/O	NxAD<58>
61	I/O	NxAD<59>
163	I/O	NxAD<60>
220	I/O	NxAD<61>
324	I/O	NxAD<62>
13	I/O	NxAD<63>
228	I	OSC14M
126	O	OSCBY12
53	I	OUT2
184	O	RAS<0>*
240	O	RAS<1>*
128	O	RAS<2>*
293	O	RAS<3>*
30	O	RAS<4>*
292	O	RAS<5>*
76	O	RAS<6>*
238	O	RAS<7>*
267	O	RDYRTN*
86	I/O	REFRESH*
215	I	RESET*
58	O	RESETCPU*
291	O	ROMCS*
283	I/O	RESETOUT*
145	N/C	RESERVE
40	I/O	SA0
305	I/O	SA1
88	I/O	SBHE*
147	O	SDIR1*
138	O	SDIR2*
251	O	SDOE01*
254	O	SDOE<0>*
199	O	SDOE<1>*
144	O	SDOE<2>*
3	O	SDOE<3>*
139	O	SMEMR*
87	O	SMEMW*
333	O	SPKR
181	I	TURBO
311	I	VCC4
314	I	VCC4
319	I	VCC4
323	I	VCC4
326	I	VCC4
330	I	VCC4
331	I	VCC4
335	I	VCC4
343	I	VCC4
347	I	VCC4
350	I	VCC4
354	I	VCC4
355	I	VCC4
360	I	VCC4
364	I	VCC4
371	I	VCC4
375	I	VCC4
379	I	VCC4
382	I	VCC4
386	I	VCC4
390	I	VCC4
393	I	VCC4
401	I	VCC4
180	I	VCC5
275	I	VCC5
278	I	VCC5
301	I	VCC5
307	I	VCC5
313	I	VCC5
318	I	VCC5
332	I	VCC5
338	I	VCC5
342	I	VCC5
368	I	VCC5
397	I	VCC5
45	O	VLASADIR*
316	I	VSS
317	I	VSS
328	I	VSS
329	I	VSS
340	I	VSS
341	I	VSS
352	I	VSS
358	I	VSS
359	I	VSS
361	I	VSS
362	I	VSS
363	I	VSS
365	I	VSS
366	I	VSS
367	I	VSS
369	I	VSS
370	I	VSS
353	I	VSS
372	I	VSS
373	I	VSS
374	I	VSS
376	I	VSS
377	I	VSS
378	I	VSS
380	I	VSS
381	I	VSS
383	I	VSS
384	I	VSS
385	I	VSS
387	I	VSS
388	I	VSS
389	I	VSS
391	I	VSS
392	I	VSS
394	I	VSS
395	I	VSS
396	I	VSS
398	I	VSS
399	I	VSS
400	I	VSS
344	O	WE*
216	I	XACK*
18	I	XBCKE*
64	I	XBOE*
29	O	XDEN*
253	O	XDIR*
107	I	XHLD*

Figure 3 NxVL Pin List, By Signal Name (continued)

NxVL Pinouts by Pin Numbers

Pin	Type	Signal
1	I/O	LBADR<31>
2	I/O	LBADR<24>
3	O	SDOE<3>*
4	I/O	LBADR<19>
5	I/O	LBADR<12>
6	I/O	LBE<1>*
7	I	CLK
8	I/O	LADS*
9	I/O	LRDY*
10	O	LGNT<0>*
11	I	ADSTB16
12	I/O	NxAD<52>
13	I/O	NxAD<63>
14	I/O	NxAD<29>
15	I/O	NxAD<55>
16	I/O	NxAD<26>
17	I/O	NxAD<15>
18	I	XBCKE*
19	I/O	NxAD<50>
20	I/O	NxAD<17>
21	I/O	NxAD<7>
22	I/O	NxAD<40>
23	I/O	MD<44>
24	I/O	MD<56>
25	I/O	MD<39>
26	I/O	MD<36>
27	I/O	MD<48>
28	O	FLASHCS*
29	O	XDEN*
30	O	RAS<4>*
31	O	MA<9>
32	O	CASA<6>*
33	O	MA<5>
34	I/O	MD<28>
35	I/O	MD<25>
36	I/O	MPAR<2>
37	I/O	MD<5>
38	I/O	MD<2>
39	I	MASTER*
40	I/O	SA0
41	I/O	MEMW*
42	I/O	LBDATA<0>
43	I/O	LBDATA<6>
44	I/O	LBDATA<9>
45	O	VLASADIR*
46	I/O	LBADR<26>
47	I/O	LBDATA<26>
48	I/O	LBADR<23>
49	I/O	LBADR<16>
50	I/O	LBADR<2>
51	I/O	LBADR<9>
52	I/O	LBM/IO*
53	I	OUT2
54	I	LREQ<0>*
55	I/O	IOW*
56	I/O	LBADR<4>
57	I	BCLKSEL
58	O	RESETCPU*
59	I/O	NxAD<30>
60	I/O	NxAD<28>
61	I/O	NxAD<59>
62	I/O	NxAD<47>
63	I/O	NxAD<46>
64	I	XBOE*
65	I/O	NxAD<48>
66	I/O	NxAD<39>
67	I/O	NxAD<2>

Pin	Type	Signal
68	I/O	NxAD<42>
69	I/O	MD<45>
70	I/O	MD<58>
71	I/O	MPAR<5>
72	I/O	MD<53>
73	I/O	MD<50>
74	O	DRAMBFDIR
75	I	BUFMODE
76	O	RAS<6>*
77	O	CASB<7>*
78	O	MA<8>
79	O	MA<10>
80	O	CASB<1>*
81	I/O	MD<29>
82	I/O	MD<11>
83	I/O	MPAR<3>
84	I/O	MD<6>
85	I/O	MD<19>
86	I/O	REFRESH*
87	O	SMEMW*
88	I/O	SBHE*
89	O	KBDCS*
90	I/O	LBDATA<2>
91	I/O	LBDATA<7>
92	I/O	LBDATA<12>
93	I/O	LBDATA<21>
94	I/O	LBADR<28>
95	I/O	LBDATA<29>
96	I/O	LBDATA<25>
97	I/O	LBADR<18>
98	I/O	LBADR<10>
99	I/O	LBADR<5>
100	I/O	LBE<2>*
101	O	GATE2
102	I	LDEV<0>*
103	I/O	BLAST*
104	I	ADSTB8
105	I	ALE*
106	I	DCL*
107	I	XHLD*
108	I/O	NxAD<31>
109	I/O	NxAD<21>
110	I/O	NxAD<57>
111	I/O	NxAD<56
112	I/O	NxAD<14>
113	I/O	NxAD<16>
114	I/O	NxAD<37>
115	I/O	NxAD<6>
116	I/O	NxAD<9>
117	I/O	NxAD<41>
118	I/O	NxAD<11>
119	I/O	MD<46>
120	I/O	MD<60>
121	I/O	MD<41>
122	I/O	MD<55>
123	I/O	MD<52>
124	I/O	MD<33>
125	I	LREQ<2>*
126	O	OSCBY12
127	O	CASB<4>*
128	O	RAS<2>*
129	O	CASA<7>*
130	O	MA<4>
131	O	CASB<0>*
132	I/O	MD<31>
133	I/O	MD<12>
134	I/O	MD<9>

Pin	Type	Signal
135	I/O	MD<23>
136	I/O	MD<20>
137	I/O	MD<17>
138	O	SDIR2*
139	O	SMEMR*
140	O	AEN
141	I/O	LBDATA<1>
142	I/O	LBDATA<5>
143	I/O	LBDATA<11>
144	O	SDOE<2>*
145	N/C	RESERVED
146	I/O	LBDATA<20>
147	O	SDIR1*
148	I/O	LBADR<27>
149	I/O	LBDATA<27>
150	I/O	LBADR<21>
151	I/O	LBADR<14>
152	I/O	LBADR<3>
153	I/O	LBADR<8>
154	I/O	LBE<3>*
155	O	INTA*
156	I	LREQ<1>*
157	I	AEN8*
158	O	ASRTC
159	O	GATEA20
160	O	GXHLD
161	O	GBLKNBL
162	I/O	NxAD<53>
163	I/O	NxAD<60>
164	I/O	NxAD<27>
165	I/O	NxAD<24>
166	I/O	NxAD<12>
167	I/O	NxAD<13>
168	I/O	NxAD<4>
169	I/O	NxAD<5>
170	I/O	NxAD<51>
171	I/O	NxAD<3>
172	I/O	NxAD<10>
173	I/O	NxAD<35>
174	I/O	MD<47>
175	I/O	MD<62>
176	I/O	MD<43>
177	I/O	MPAR<7>
178	I/O	MD<38>
179	I/O	MD<35>
180	I	VCC5
181	I	TURBO
182	O	MA<1>
183	O	CASB<5>*
184	O	RAS<0>*
185	O	MA<7>
186	O	CASB<3>*
187	O	CASA<1>*
188	I/O	MD<15>
189	I/O	MD<13>
190	I/O	MD<26>
191	I/O	MPAR<1>
192	I/O	MD<21>
193	I/O	MD<3>
194	I	IOCHCHK*
195	O	BALE
196	I	LDEV<2>*
197	I	IOCS16*
198	I/O	LBDATA<3>
199	O	SDOE<1>*
200	I/O	LBDATA<15>
201	I/O	LBDATA<16>

Figure 4 NxVL Pin List, By Pin Number

Pin	Type	Signal
202	I/O	LBDATA<17>
203	I/O	LBDATA<23>
204	I/O	LBADR<29>
205	I/O	LBDATA<30>
206	I/O	LBDATA<24>
207	I/O	LBADR<17>
208	I/O	LBADR<13>
209	I/O	LBADR<7>
210	I/O	LBD/C*
211	O	ISABCLK
212	I	LDEV<1>*
213	O	LGNT<1>*
214	I/O	IOCHRDY
215	I	RESET*
216	I	XACK*
217	I	AREQ*
218	I/O	NxAD<20>
219	I/O	NxAD<54>
220	I/O	NxAD<61>
221	I/O	NxAD<58>
222	I/O	NxAD<45>
223	O	GXACK
224	I/O	NxAD<38>
225	I/O	NxAD<18>
226	I/O	NxAD<19>
227	I/O	NxAD<43>
228	I	OSC14M
229	I/O	NxAD<36>
230	I/O	NxAD<33>
231	I/O	MD<63>
232	I/O	MD<61>
233	I/O	MD<57>
234	I/O	MPAR<6>
235	I/O	MD<37>
236	I/O	MD<49>
237	O	LGNT<2>*
238	O	RAS<7>*
239	O	CASB<6>*
240	O	RAS<1>*
241	O	CASA<5>*
242	O	MA<3>
243	O	MA<0>
244	O	CASA<3>*
245	I/O	MD<14>
246	I/O	MD<27>
247	I/O	MD<24>
248	I/O	MD<7>
249	I/O	MD<4>
250	I/O	MD<1>
251	O	SDOE01*
252	I/O	MEMR*
253	O	XDIR*
254	O	SDOE<0>*
255	I/O	LBDATA<4>
256	I/O	LBDATA<10>
257	I/O	LBDATA<18>
258	I/O	LBDATA<19>
259	I/O	LBADR<30>
260	I/O	LBADR<25>
261	I/O	LBDATA<28>
262	I/O	LBADR<20>
263	I/O	LBADR<15>
264	I/O	LBADR<6>
265	I/O	LBE<0>*
266	I/O	LBW/R*
267	O	RDYRTN*
268	O	BRDY*

Pin	Type	Signal
269	I	AEN16*
270	O	HLDA
271	I	NREQ*
272	I	LOCK*
273	O	NMI*
274	I/O	NxAD<22>
275	I	VCC5
276	I/O	NxAD<25>
277	O	GALE
278	I	VCC5
279	I/O	GNT*
280	I/O	NxAD<8>
281	I/O	NxAD<49>
282	I/O	NxAD<0>
283	I/O	RESETOUT*
284	I/O	NxAD<34>
285	I/O	NxAD<32>
286	I/O	MD<59>
287	I/O	MD<40>
288	I/O	MD<54>
289	I/O	MD<51>
290	I/O	MD<32>
291	O	ROMCS*
292	O	RAS<5>*
293	O	RAS<3>*
294	O	CASA<4>*
295	O	MA<6>
296	O	CASB<2>*
297	O	CASA<2>*
298	I/O	MD<30>
299	I/O	MD<10>
300	I/O	MPAR<0>
301	I	VCC5
302	I/O	MD<18>
303	I/O	MD<0>
304	I/O	IOR*
305	I/O	SA1
306	I/O	MEMCS16*
307	I	VCC5
308	I/O	LBDATA<8>
309	I/O	LBDATA<14>
310	I/O	LBDATA<22>
311	I	VCC4
312	I/O	LBDATA<31>
313	I	VCC5
314	I	VCC4
315	I/O	LBADR<11>
316	I	VSS
317	I	VSS
318	I	VCC5
319	I	VCC4
320	I	LBS16*
321	I	HHOLD
322	O	GTAL
323	I	VCC4
324	I/O	NxAD<62>
325	I/O	NxAD<23>
326	I	VCC4
327	I/O	NxAD<44>
328	I	VSS
329	I	VSS
330	I	VCC4
331	I	VCC4
332	I	VCC5
333	O	SPKR
334	I/O	NxAD<1>
335	I	VCC4

Pin	Type	Signal
336	I/O	MD<42>
337	I/O	MPAR<4>
338	I	VCC5
339	I/O	MD<34>
340	I	VSS
341	I	VSS
342	I	VCC5
343	I	VCC4
344	O	WE*
345	O	MA<2>
346	O	CASA<0>*
347		VCC4
348	I/O	MD<8>
349	I/O	MD<22>
350	I	VCC4
351	I/O	MD<16>
352	I	VSS
353	I	VSS
354	I	VCC4
355	I	VCC4
356	I/O	LBADR<22>
357	I/O	LBDATA<13>
358	I	VSS
359	I	VSS
360	I	VCC4
361	I	VSS
362	I	VSS
363	I	VSS
364	I	VCC4
365	I	VSS
366	I	VSS
367	I	VSS
368	I	VCC5
369	I	VSS
370	I	VSS
371	I	VCC4
372	I	VSS
373	I	VSS
374	I	VSS
375	I	VCC4
376	I	VSS
377	I	VSS
378	I	VSS
379	I	VCC4
380	I	VSS
381	I	VSS
382	I	VCC4
383	I	VSS
384	I	VSS
385	I	VSS
386	I	VCC4
387	I	VSS
388	I	VSS
389	I	VSS
390	I	VCC4
391	I	VSS
392	I	VSS
393	I	VCC4
394	I	VSS
395	I	VSS
396	I	VSS
397	I	VCC5
398	I	VSS
399	I	VSS
400	I	VSS
401	I	VCC4

Figure 5 NxVL Pin List, By Pin Number (continued)

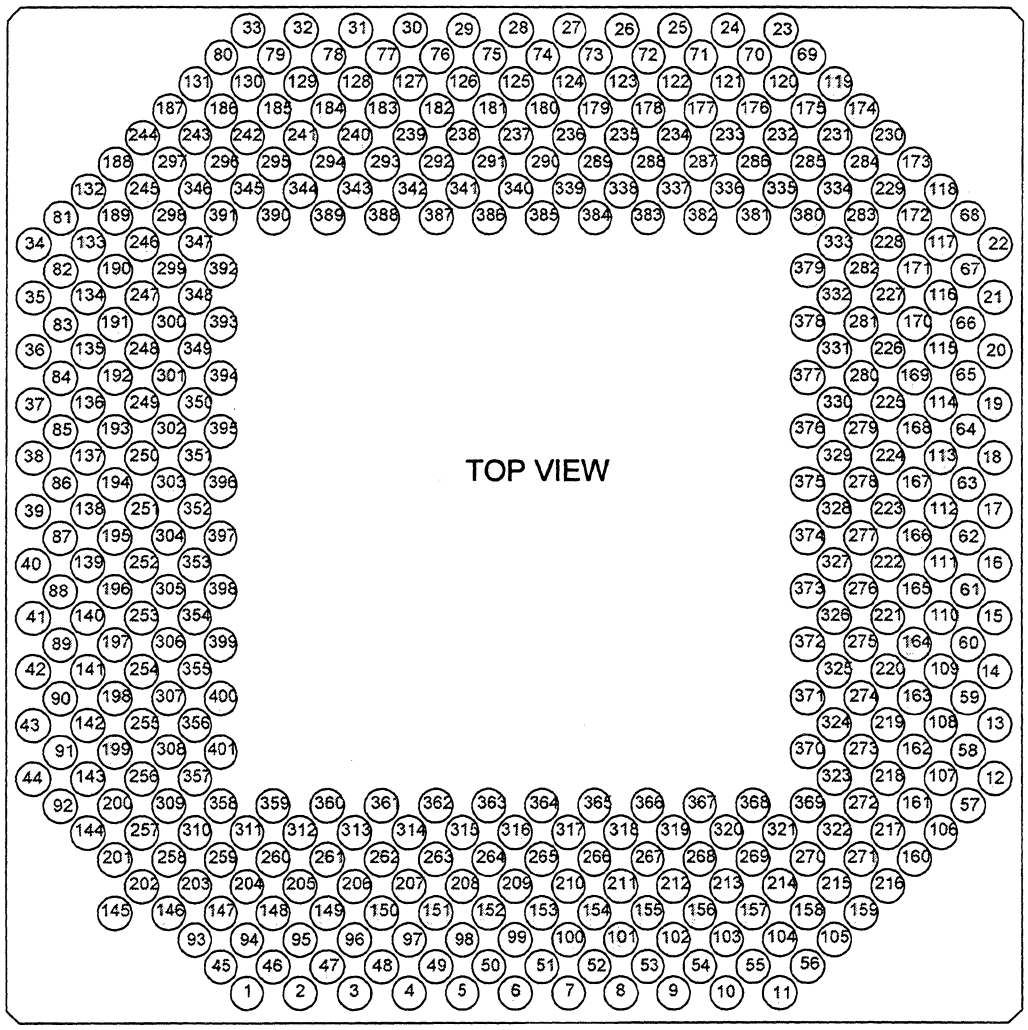


Figure 6 NxVL Pinout Diagram (Top View)

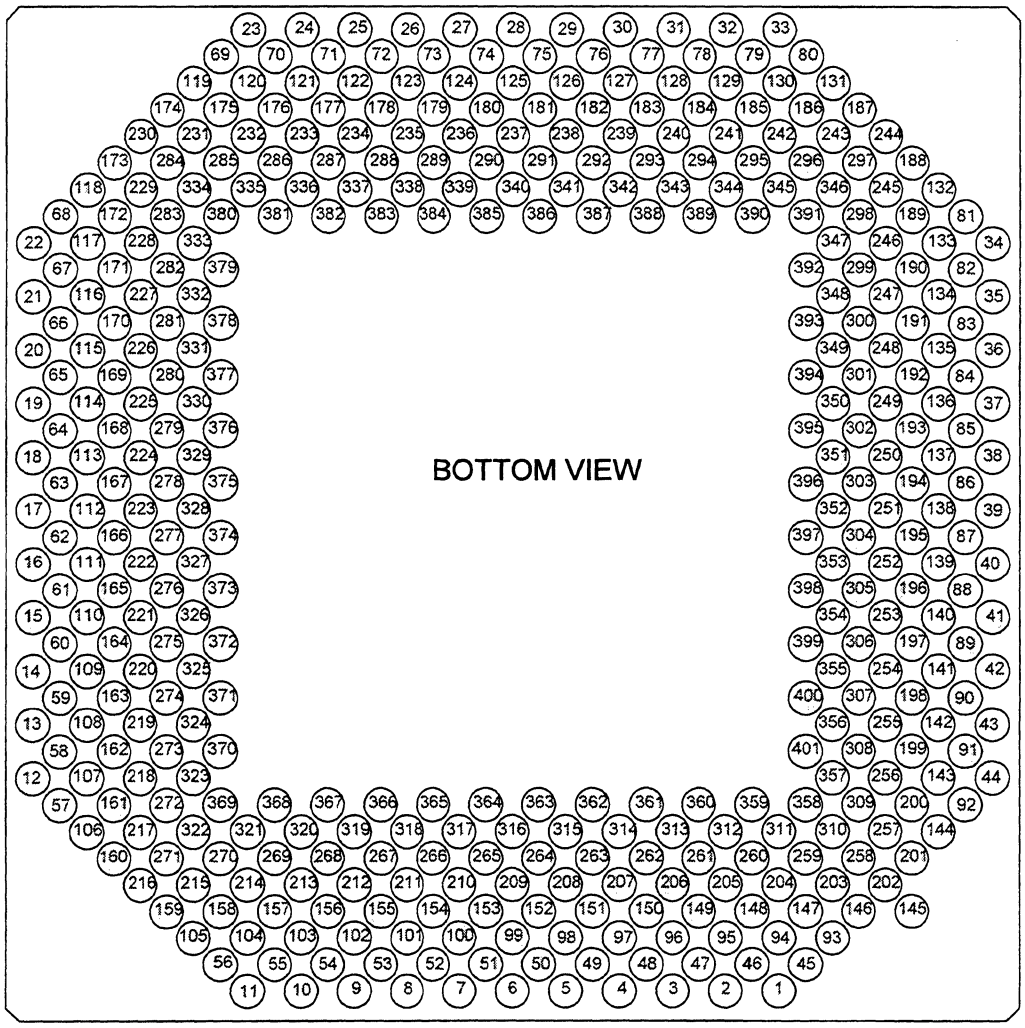


Figure 7 NxVL Pinout Diagram (Bottom View)

NexBus Signals

NexBus Arbitration

NREQ*	I	**NexBus Request**—Asserted by the Nx586 processor to secure control of the NexBus. This signal is driven active by the master (Nx586) until GNT* is received from the NxVL, although during speculative reads the processor may deactivate NREQ* before GNT* is received if the transfer is no longer needed. NREQ* is treated the same as AREQ*; when NxVL asserts GNT*, the processor is granted access to all buses in the system—NexBus, VL bus, and ISA bus.
LOCK*	I	**Bus Lock**—Asserted by the Nx586 processor to sustain a bus grant that was obtained by the assertion of NREQ* or AREQ*. This signal is used to determine the end of a bus sequence. Cache-block fills are not locked; they are implicitly treated as atomic reads.
AREQ*	I	**Alternate-Bus Request**—Asserted by the Nx586 processor to secure locked control of the NexBus and all alternate buses (VL bus or ISA bus). This signal is driven active until GNT* is received from the NxVL. Unlike NREQ*, the processor does not make speculative requests for the VL bus or ISA bus with AREQ*. The NxVL does not issue GNT* until the VL bus or ISA bus are available. AREQ* and NREQ* have the same effect in the sense that either one causes the NxVL system to grant all buses to the winning requester at the end of the current bus cycle. However, AREQ* locks the use of the buses until it is negated.
GNT*	O	**Grant NexBus**—Asserted to the Nx586 processor to grant the processor access to all buses. The buses are given to the processor (a) at reset, (b) after a transfer when NREQ* or AREQ* are asserted, (c) when no other master is requesting the buses, or (d) when the processor asserts DCL* as an intervenor during a NexBus snoop cycle initiated by a VL bus master or ISA-bus master access.

NexBus Cycle Control

ALE*	I	**Address Latch Enable**—Asserted by the Nx586 processor to indicate the presence of a valid address on the NxAD<63:0> bus. It is used by NxVL to latch the NexBus address and to signal to the internal state machines that a NexBus cycle has started.
GALE	O	**Group Address Latch Enable**—Asserted to the Nx586 processor to indicate that valid address and status information can be latched from the NxAD<63:0> bus. It indicates that (a) an ALE* input was received from the processor, or (b) NxVL is initiating a NexBus snoop cycle.
GTAL	O	**GTAL**—This signal must be connected to the Nx586's GTAL signal pin.
XACK*	I	**Transfer Acknowledge**—Asserted by the Nx586 processor during a NexBus snoop cycle, when the processor determines that it has data from the snooped address.
GXACK	O	**Group Transfer Acknowledge**—Asserted to the Nx586 processor to indicate that (a) an XACK* input was received from the processor during a NexBus snoop cycle, or (b) the addressed slave—whether main memory or a slave on the VL bus or ISA bus—is prepared to respond as a slave to the processor's current operation.
XHLD*	I	**Transfer Hold**—Asserted by the Nx586 processor when it is unable to respond as a slave on the next clock after GXACK. NxVL responds by inserting wait states in the cycle.
GXHLD	O	**Group Transfer Hold**—Asserted to the Nx586 processor to indicate that (a) an XHLD* input was received from the processor during a NexBus snoop cycle, or (b) wait states are needed in the current NexBus cycle. At the end of snoop cycles or wait sequences, GXACK is negated one clock after GXHLD is negated. During a bus-crossing read by the processor, the simultaneous assertion of GXACK and negation of GXHLD indicates that valid data is available on the bus. During a bus-crossing write, the same signal states indicate that data has been accepted by the slave.

NexBus Cache Control

DCL*	I	**Dirty Cache Line**—Asserted by the Nx586 processor during snoop cycles on the NexBus. It indicates that the location being accessed is cached by the processor's level-2 cache in a *modified* (dirty) state. In response, NxVL causes the requesting master's cycle to be aborted so that the processor, as an intervenor, can preemptively gain control of the NexBus and write back its modified data to main memory.
GBLKNBL	O	**Group Block (Burst) Enable**—Asserted to the Nx586 processor during termination of an access to enable burst transfers, and to indicate that the addressed space may be cached. GBLKNBL is negated if the accessed location is among those stored in the Non-Cacheable Registers (NCR1-NCR0) or if the cycle crosses over to the VL bus or ISA bus. That is, only main memory is cacheable, and only those parts that are not recorded in the Non-Cacheable Registers.

NexBus Transceiver Control

XBCKE*	I	**Transceiver Clock Enable**—Asserted by the Nx586 processor to clock registered transceivers and latch addresses and data on the NxAD<63:0> NexBus (see Figure 14). With respect to NxVL, these transceivers are integrated and are enabled when the BUFMODE signal is tied high and XBCKE* is tied to the same-named output on the Nx586 processor. XBCKE* indicates to NxVL that the address or data should be latched on the rising edge of the clock.
XBOE*	I	**Transceiver Output Enable**—Asserted by the Nx586 processor to enable the registered transceivers and drive addresses and data onto the NxAD<63:0> NexBus from the AD<63:0> bus (see Figure 14). With respect to NxVL, these transceivers are integrated and are enabled when the BUFMODE signal is tied high and XBOE* is tied to the same-named output on the Nx586 processor.

NexBus Address and Data Bus

NxAD<63:0>	I/O	**NexBus Address and Status, or Data**—As an input, the signal is driven by the Nx586 processor during the address phases of all processor-initiated cycles and during the data phases of processor write cycles. As an output, the bus is driven to the Nx586 processor during the address phases of NxVL snoop cycles and the data phases of processor read cycles. When the NxVL system is configured to emulate NexBus transceivers (BUFMODE tied high), this bus is tied directly to the NxAD<63:0> bus on the Nx586 processor, as shown in Figure 14. For a detailed description of signaling on pins NxAD<63:32> during the address and status phase, see the signal descriptions in the *Nx586 Processor and Nx587 Floating Point Coprocessor Databook.* NxAD<63:0> is a tristate bus.

VL-Bus Signals

VL-Bus Arbitration

LREQ<2:0>*	I	**VL-Bus Request**—Asserted by VL-bus masters to gain access to the system buses. There can be up to three VL-bus masters. The NxVL arbitration unit gives circular priority to the three masters in the order, 3, 1, 2. See the section entitled *Bus Arbitration* in the *Bus Operations* chapter.
LGNT<2:0>*	O	**VL-Bus Grant**—One of these three outputs may be asserted to a VL-bus master in response to that master's assertion of its LREQ<n>* signal. The NxVL arbitor thereby grants mastership of all three buses to a requesting master according to the NxVL arbitration protocol. See *Bus Arbitration* in the *Bus Operations* chapter.
LDEV<2:0>*	I	**VL-Bus Device**—Asserted by VL-bus devices to indicate when an address on the VL bus is addressing that device. The timing relationships for this signal differ with different clock speeds; see Figure 41.

VL-Bus Cycle Control

LADS*	I/O	**VL-Bus Address Strobe**—As an input, this signal is asserted by the VL-bus master to indicate the presence of a valid address on the VL bus. As an output, LADS* is asserted to the VL bus during bus-crossing cycles initiated by the Nx586 processor, DMA controller, or ISA-bus master. It indicates the presence of a valid address on the VL bus.
LBD/C*	I/O	**VL-Bus Data or Code**—As an input, this signal is driven by the VL-bus master to indicate a data (negated) or instruction (asserted) access. As an output, LBD/C* is driven to the VL-bus slaves. It indicates a data or instruction access during bus-crossing cycles initiated by the Nx586 processor, DMA controller, or ISA-bus master. In both directions, the signal is interpreted in conjunction with LADS*, LBADR<31:2>, LBM/IO*, LBW/R*, and LBE<3:0>*.

LBM/IO*	I/O	**VL-Bus Memory or I/O**—As an input, this signal is driven by a VL-bus master high to indicate a memory access or low for an I/O access. As an output, NxVL drives LBM/IO* for VL-bus slaves during memory or I/O bus-crossing cycles initiated by the Nx586 processor, a DMA controller, or an ISA-bus master.
LBW/R*	I/O	**VL-Bus Write or Read**—As an input, this signal is driven by a VL-bus master high to indicate a write access or low for a read access. As an output, NxVL drives LBW/R* for VL-bus slaves during write or read bus-crossing cycles initiated by the Nx586 processor, a DMA controller, or an ISA-bus master.
LBS16*	I	**VL-Bus Size 16**—Asserted by a VL-bus slave to indicate that the slave is a 16-bit device. If more than 16 bits are to be transferred, the NxVL will perform multiple 16-bit transfers to complete the request.
LBE<3:0>*	I/O	**VL-Bus Byte Enables**—As an inputs, these signal are driven by the VL-bus master to indicate the valid bytes in the currently addressed data. The numbering of the LBE<n>* signals corresponds to the byte location within the data: if LBE<0> is asserted, the least-significant byte of the data is valid. As an output, NxVL drives LBE<3:0>* for VL-bus slaves to indicate valid bytes in the currently addressed qword during bus-crossing cycles initiated by the Nx586 processor, a DMA controller, or an ISA-bus master.
LRDY*	I/O	**VL-Bus Ready**—As an input, this signal is driven active by a VL-bus slave drives to indicate the end of a bus-crossing transfer initiated by the Nx586 processor, a DMA controller, or an ISA-bus master. As an output, LRDY* is asserted to the VL-bus to indicate that the requested data is valid on the VL-bus. It indicates the end of the current transfer.
RDYRTN*	O	**VL-Bus Ready Return**—Asserted to all VL-bus devices in response to an LRDY* input when a VL-bus slave finishes a cycle. This signal can be asserted in response to a VL-bus master cycle, a VL-bus slave, or a bus-crossing cycle with a VL-bus slave that is initiated by the Nx586 processor, a DMA controller, or an ISA-bus master.

BLAST*	I/O	**VL-Bus Burst Last**—As an input, this signal is asserted by a VL-bus master to indicate the last data transfer of either a burst or a non-burst VL-bus cycle. The cycle ends with the next assertion of BRDY* or LRDY*. BLAST* is negated by the VL-bus master during non-burst transfers. As an output, BLAST* is asserted to the VL-bus during bus-crossing cycles. It indicates a non-burst cycle—i.e., that the next time LRDY* is asserted, the cycle will be complete. The Nx586 processor cannot perform bursts to the VL-bus, but a VL-bus master can request bursts to main memory.
BRDY*	O	**VL-Bus Burst Ready**—Asserted to a VL-bus master during burst memory cycles. It indicates that data for one of four (4) data transfers is currently valid on the VL-bus. Whenever the VL-bus master addresses main memory, the NxVL controller attempts to perform burst transfers and terminates each transfer with BRDY*. If the VL-bus master addresses memory on the ISA-bus, NxVL will not support burst transfers and will respond instead with LRDY*. BRDY* is a tristate signal.

VL-Bus Address

LBADR<31:2>	I/O	**VL-Bus Address**—As an input, these signals are driven by the VL-bus master with the address being accessed. As an output, NxVL drives LBADR<31:2> with an address to the VL-bus slaves during bus-crossing cycles initiated by the Nx586 processor, a DMA controller, or an ISA-bus master.

VL-Bus Data

LBDATA<31:0>	I/O	**VL-Bus Data**—As an input, these signals are driven by the VL-bus master during read cycles. As an output, NxVL drives LBDATA<31:0> to the VL-bus master during memory write cycles.

ISA Bus Signals

ISA-Bus Cycle Control

MASTER*	I	**ISA-Bus Master**—Asserted by an ISA-bus master to indicate that it is in control of the ISA bus. This also causes the integrated peripheral controller to *not* participate as a DMA controller in the transfers.
AEN	O	**ISA-Bus Address Enable**—Asserted only by NxVL to ISA-bus slaves, during DMA bus-master cycles on the ISA bus. AEN *disables* address decoding by I/O devices on the ISA bus. The signal is negated during all other types of cycles to enable address decoding by I/O devices on the ISA bus.
SBHE*	I/O	**ISA-Bus High Byte Enable**—As an input, this signal is asserted by an ISA-bus master during ISA-bus cycles to indicate that the master is either requesting or providing valid data on the ISA-bus high byte, SD<15:8>. As an output, NxVL assertes SBHE* to the ISA bus during bus-crossing cycles initiated by the Nx586 processor or a VL-bus master, or during DMA cycles. Again, it indicates that the master is either requesting or providing valid data on the ISA-bus high byte, SD<15:8>.
BALE	O	**ISA-Bus Address Latch Enable**—Asserted by NxVL to ISA-bus devices to indicate that the ISA-bus address, AEN, and SBHE* signals are valid. For ISA-bus master cycles, this signal is asserted throughout that master's cycle.
MEMR*	I/O	**ISA-Bus Memory Read**—As an input, this signal is asserted by the a DMA controller or an ISA-bus master during DMA reads. As an output, MEMR* is asserted by the Nx586 processor or by a VL-bus master to an ISA-bus memory slave during bus-crossing reads. NxVL also asserts MEMR* when an ISA-bus master asserts REFRESH*.
MEMW*	I/O	**ISA-Bus Memory Write**—As an input, the signal is asserted by a DMA controller or an ISA-bus master during DMA writes. As an output, MEMW* is asserted to an ISA-bus memory slave during bus-crossing writes by the Nx586 processor or by a VL-bus master.

MEMCS16*	I/O	**ISA-Bus Memory Chip Select 16**—As an input, this signal is driven by an ISA-bus memory slave during bus-crossing cycles by the Nx586 processor or by a VL-bus master. NxVL responds by performing 16-bit memory cycles. MEMCS16* is open drain signal. As an output, MEMCS16* is driven to an ISA-bus memory slave for cycles in the 512kB-to-1MB range of memory by a DMA controller or an ISA-bus master. The Output is enabled when bit 18 of the Configuration Register is set to 1.
SMEMR*	O	**ISA-Bus System Memory Read**—Asserted to an ISA memory slave during read cycles addressed under 1MB. SMEMR* is derived from the MEMR* signal. SMEMR* is a tristate signal.
SMEMW*	O	**ISA-Bus System Memory Write**—Asserted to an ISA memory slave during write cycles addressed under 1MB. The SMEMW* is derived from the MEMW* signal. SMEMW* is a tristate signal.
IOR*	I/O	**ISA-Bus I/O Read**—As an input, this signal is asserted by ISA-bus masters during I/O read cycles on the ISA-bus. As an output, IOR* is driven to an ISA-bus slave during I/O read cycles by the Nx586 processor, DMA controller or a VL-bus master.
IOW*	I/O	**ISA-Bus I/O Write**—As an input, the signal is asserted by ISA-bus masters during I/O write cycles on the ISA-bus. As an output, IOW* is asserted to an ISA-bus slave during I/O write cycles by the Nx586 processor, DMA Controller or a VL-bus master.
IOCS16*	I	**ISA-Bus I/O Chip Select 16**—Asserted by an ISA-bus I/O slave to signal that the slave can support either 16-bit or 8-bit access cycles.
IOCHCK*	I	**ISA-Bus I/O Channel Check Error**—Asserted by an ISA-bus device to indicate an error condition on the ISA-bus. It is used to generate NMI* to the Nx586 processor. The state of the signal can also be read from bit 6 of I/O port 61h in the NxVL system.

IOCHRDY	I/O	**ISA-Bus I/O Channel Ready**—As an input, this signal is asserted by an ISA device to indicate the end of a bus-crossing cycle initiated by the Nx586 processor or a VL-bus master. If the signal is negated, NxVL will add wait states. IOCHRDY is an open drain signal. As an output, IOCHRDY is driven during DMA or ISA-master cycles to indicate that the cycle is finished. If the signal is negated, the DMA controller or ISA-master must add wait states.

ISA-Bus Transceiver Control

VLASADIR*	O	**VL-to-ISA Address Direction**—Asserted for external transceivers to enable a VL-bus address onto the ISA-bus. VLASADIR* is negated to enable an ISA-bus address onto the VL bus. Refer to Figure 20.
SDIR1* **SDIR2***	O	**SD-Buffer Direction**—Asserted for external transceivers to select the direction of data transfers between the ISA-bus (SD bus) and the VL-bus. Refer to Figure 19.
SDOE0* **SDOE1*** **SDOE2*** **SDOE3***	O	**SD-Buffer Output Enable**—Asserted to external transceivers to enable transfers between the ISA bus (SD bus) and the VL-bus. Refer to Figure 19.
SDOE01*	O	**SD-Buffer Output Enable**—Asserted for external transceivers to enable transfers between the ISA-bus (SD bus) and the VL-bus. Refer to Figure 19.
XDEN*	O	**XD-Buffer Enable**—Asserted for external transceivers to enable data flow between the SD and XD buses. XDEN* is driven active during all I/O transactions and at the beginning of DMA transfers. Refer to Figure 21
XDIR*	O	**XD-Buffer Direction**—Asserted for external transceivers to direct data from the SD-bus to XD bus. When high it directs data from the XD-bus to the SD-bus. Refer to Figures 20 and 22.

ISA-Bus Address, Refresh, and Clock

SA<1:0>	I/O	**ISA-Bus Address Bits 1 and 0**—As an input, these signals are asserted by an ISA-bus master or the Integrated Peripheral Controller during ISA-bus cycles. These signals are the lower two bits of address on the SA bus; the rest of the address is translated from the VL bus. As outputs, theses signals are driven to an ISA-bus slave during bus-crossing cycles initiated by the Nx586 processor, DMA controller or a VL-bus master. They provide the lower two bits of address, which are derived from the VL-bus byte-enables bits, LBE<3:0>*.
REFRESH*	I/O	**ISA-Bus Refresh**—As an input, this signal is asserted by the ISA-bus master when the master performs a refresh cycle for ISA-bus memory. As an output, REFRESH* is driven to memory on the ISA-bus when the NxVL ISA refresh logic performs an ISA-bus memory refresh cycle. REFRESH* is an open drain signal. .
OSC14M	I	**14MHz Clock Input**—This signal should by driven by a 14.31818 MHz clock.
OSCBY12	O	**14MHz Divided by 12**—This signal is equal to OSC14M divided by 12.
BCLKSEL	I	**BCLK SELECT**—This signal determines the source clock for the ISABLK signal. When high or active, the ISA bus clock is derived from the internal programmable counter. If BCLKSEL is low or inactive, ISABCLK is set equal to OSC14M divided by 2.
ISABCLK	O	**ISA-Bus Clock**—This signal is simply an output specified for the ISA-bus. ISABCLK runs at a programmable division of the NexBus clock or OSC14M divided by 2 as determined by BCLKSEL. The NexBus clock can be divide by 3, 4, 5 or 6 for ISABCLK, as specified in bits 11:10 of Configuration Register CFG0.

Memory-Bus Signals

DRAMBFDIR	O	**DRAM Buffer Direction**—Asserted during write cycles to main memory. DRAMBFDIR is driven low during read cycles.
WE*	O	**Write Enable**—Asserted to select main memory during a write cycle. If WE* is negated and CASA<7:0>* or CASB<7:0>* is asserted, a read cycle is assumed.
MA<10:0>	O	**Memory Address**—Driven to main memory with muliplexed row and column addresses during memory accesses and refresh cycles.
RAS<7:0>*	O	**Row Address Strobes**—Asserted to the main memory DRAM modules during memory accesses and refresh cycles.
CASA<7:0>*	O	**Column Address Strobes (Bank A)**—Asserted to bank A (1) of the main memory modules during memory read and write cycles.
CASB<7:0>*	O	**Column Address Strobes (Bank B)**—Asserted to bank B (2) of the main memory modules during memory read and write cycles.
MD<63:0>	I/O	**Memory Data**—As inputs, these signals are driven by main memory during read cycles. As outputs, these signals are driven to main memory during memory write cycles.
MPAR<7:0>	I/O	**Memory Parity**—As inputs, these signals are driven by main memory during read cycles to indicate even parity on the MD<63:0> bus. MPAR<0>* corresponds to the byte on MD<7:0>, and MPAR<7>* corresponds to the byte on MD<63:56>. As outputs, these signals are driven in a similar manner to main memory during memory write cycles. However, parity errors are only reported via NMI* if main memory parity checking is enabled by setting bit-15 in Configuration Register CFG0 to 1.

Integrated Peripheral Controller (IPC) Signals

ASRTC	O	**Address Strobe for Real-Time Clock**—Asserted to the AS or ASRTC signal of a IPC . It is used on the falling edge to latch the address from the XD-bus.
HHOLD	I	**Hold Request**—Asserted by the IPC to indicate that access to the buses is needed for a DMA cycle between main memory and the ISA-bus. Either the IPC or an ISA-bus master can be a DMA master and support transfers between main memory and the ISA-bus. The VL-bus does not support DMA transfers. The NexBus supports DMA transfers, but not in the single-processor configuration implemented with the NxVL.
HLDA	O	**Hold Acknowledge**—Asserted to the IPC in response to latter's assertion of HHOLD. It indicates that the IPC has been granted the buses.
ADSTB16	I	**Address Strobe for 16-Bit DMA**—Asserted by the IPC during 16-bit DMA transfers between main memory and the ISA-bus. It is used to latch the upper byte of address from the XD<7:0> data bus onto the SA<16:9> address bus.
AEN16*	I	**Address Enable for 16-Bit DMA**—Asserted by the IPC during 16-bit DMA transfers between main memory and the ISA-bus. It is used to enable output of the upper byte of DMA address (A9-A16).
ADSTB8	I	**Address Strobe for 8-Bit DMA**—Asserted by the IPC during 8-bit DMA transfers between main memory and the ISA-bus. It is used to latch the upper byte of address from the XD<7:0> data bus onto the SA<15:8> address bus.
AEN8*	I	**Address Enable for 8-Bit DMA**—Asserted by the IPC during 8-bit DMA transfers between main memory and the ISA-bus. It is used to output of the upper byte of DMA address (A8-A15) onto the ISA-bus.
OUT2	I	**Counter 2 Out**—Asserted by the IPC to control speaker frequency. The signal is ANDed in the NxVL with the value of bit 0 in port 61h. The state of the signal can be read at bit 5 of port 61h.
GATE2	O	**Gate For Counter 2**—Asserted to the 82C206 peripheral controller. It enables counter 2 on the controller. The signal is activated by writing bit 0 of I/O port 61h on the NxVL chip. It is typically used to control the frequency of the speaker.

INTA*	O	**Interrupt Acknowledge**—Asserted to the IPC in response to decoding the Nx586 processor's Interrupt Acknowledge cycle. As with all processor cycles not intended for main memory, the NxVL translates the Interrupt Acknowledge cycle onto the VL-bus. The IPC responds by placing the interrupt vector on the XD-bus, from which the NxVL translates in onto the NexBus.

NxVL System Signals

CLK	I	**NexBus Clock**—The NexBus clock. Its frequency must be 25, 33, or 40 MHz. It must be the same clock that is used for the VL-bus clock. Bits 9:8 of the Configuration Register, which generate wait states, must be set to the speed of this clock.
NMI*	O	**Non-Maskable Interrupt**—Asserted to the Nx586 processor three clocks after a main-memory parity error or an ISA-bus I/O channel check error (IOCHCK*). To function in this manner, however, the NMI* signal must be enabled by writing a 1 to bit 7 of port 70h, and for I/O channel checks on the ISA bus, bit 2 of port 61h must cleared to zero (enabled).
RESET*	I	**Global Reset**—Asserted by external glue logic to perform a hard or system reset. The NxVL responds by resetting all internal state machines, loading default values into the Configuration Registers, and asserting RESETCPU* to the Nx586 processor and RESETOUT* to the sub-systems. See the *Reset and Initialization* section in the *Configuration and Testing* chapter for more infromation.
RESETOUT*	O	**System Reset**—This signal is asserted for 128 clocks after RESET* goes inactive.
RESETCPU*	O	**Processor Reset**—Asserted to the Nx586 processor in any of four cases: (1) when the NxVL receives RESET*, (2) the processor writes a 1 to bit 0 of port 92h, called the *fast CPU reset bit*, (3) the processor runs a shutdown cycle, or (4) the processor does a keyboard reset cycle. See the *Reset and Initialization* section in the *Configuration and Testing* chapter for additional information.

GATEA20	O	**Gate for Address 20**—Asserted to the Nx586 processor when the current value of address bit 20 is needed on the NxAD<63:0> bus, and negated when a 0 is needed on address bit 20. This signal is used to replicate the IBM PC's method of handling 80286 address wraparound for addresses above the 20-bit limit. See GATEA20 in the *Configuration and Testing* chapter.
TURBO	I	**Turbo Switch**—Asserted by glue logic or a front-panel button to place the system in *fast* mode. When the signal is negated *(slow* mode) the NxVL extends the length of the ISA refresh cycles by a predetermined number of clocks, as specified in bits 7:0 of the Configuration Register (CFG0).
BUFMODE	I	**Buffered Mode**—Tied high to select the NexBus-buffer emulation mode. In this mode, the NxVL's NexBus interface emulates the function of external bus registered transceivers. When negated, external registered transceivers must be provided between the NxVL and NexBus. Typically, if a single Nx586 processor is used with the NxVL, transceivers are not needed and BUFMODE should be tied high. See Figure 33 and Figure 14.
ROMCS*	O	**ROM Chip Select**—Asserted to select ROM, if FLASHCS* is not enabled. This signal is derived by decoding addresses 0xE0000h to 0xFFFFFh on the ISA-bus and MEMR*.
FLASHCS*	O	**Flash-ROM Chip Select**—Asserted to select Flash ROM, if enabled by bit 27 (write) and/or bit 28 (read) of the Configuration Register CFG0. When either of these register bits are set to 1, the Flash ROM will be selected and ROMCS* will not be generated.
KBDCS*	O	**Keyboard Chip Select**—Asserted to the keyboard controller when I/O address 60h or 64h is accessed.
SPKR	O	**Speaker Data**—Asserted by the NxVL to the speaker driver.

NxVL Alphabetical Signal Summary

ADSTB16	I	Address Strobe for 16-Bit DMA
ADSTB8	I	Address Strobe for 8-Bit DMA
AEN	O	ISA-Bus Address Enable
AEN16*	I	Address Enable for 16-Bit DMA
AEN8*	I	Address Enable for 8-Bit DMA
ALE*	I	Address Latch Enable
AREQ*	I	Alternate-Bus Request
ASRTC	O	Address Strobe for Real-Time Clock
BALE	O	ISA-Bus Bus Address Latch Enable
BLAST*	I/O	VL-Bus Burst Last
BRDY*	O	VL-Bus Burst Ready
BUFMODE	I	Buffered Mode
CASA<7:0>*	O	Column Address Strobes (Bank A)
CASB<7:0>*	O	Column Address Strobes (Bank B)
CLK	I	NexBus Clock
DCL*	I	Dirty Cache Line
DRAMBFDIR	O	DRAM Buffer Direction
FLASHCS*	O	Flash-ROM Chip Select
GALE	O	Group Address Latch Enable
GATE2	O	Gate For Counter 2
GATEA20	O	Gate For Address 20
GBLKNBL	O	Group Block Enable
GNT*	O	Bus Grant
GTAL	O	*Connect to Nx586 Directly*
GXACK	O	Group Transfer Acknowledge
GXHLD	O	Group Transfer Hold
HHOLD	I	Hold Request
HLDA	O	Hold Acknowledge
INTA*	O	Interrupt Acknowledge
IOCHCK*	I	ISA-Bus I/O Channel Check Error
IOCHRDY	I/O	ISA-Bus I/O Channel Ready

Figure 8 NxVL Alphabetical Signal Summary

ICCS16*	I	ISA-Bus I/O Chip Select 16
IOR*	I/O	ISA-Bus I/O Read
IOW*	I/O	ISA-Bus I/O Write
ISABCLK	O	ISA-Bus Clock
KBDCS*	O	Keyboard Chip Select
LADS*	I/O	VL-Bus Address Strobe
LBADR<31:2>	I/O	VL-Bus Address
LBDATA<31:0>	I/O	VL-Bus Data
LBD/C*	I/O	VL-Bus Data or Code
LBE<3:0>*	I/O	VL-Bus Byte Enables
LBM/IO*	I/O	VL-Bus Memory or I/O
LBS16*	I	VL-Bus Size 16
LBW/R*	I/O	VL-Bus Write or Read
LDEV<2:0>*	I	VL-Bus Device
LGNT<2:0>*	O	VL-Bus Grant
LOCK*	I	Lock NexBus
LRDY*	I/O	VL-Bus Ready
LREQ<2:0>*	I	VL-Bus Request
MA<11:0>	O	Memory Address
MASTER*	I	ISA-Bus Master
MD<63:0>	I/O	Memory Data
MEMCS16*	I/O	ISA-Bus Memory Chip Select 16
MEMR*	I/O	ISA-Bus Memory Read
MEMW*	I/O	ISA-Bus Memory Write
MPAR<7:0>	I/O	Memory Parity
NMI*	O	Non-Maskable Interrupt
NREQ*	I	NexBus Request
NxAD<63:0>	I/O	NexBus Address and Status, or Data
OUT2	I	Counter 2 Out
PARERR*	I	*Connect to Nx586 Directly*
RAS<7:0>*	O	Row Address Strobes
RDYRTN*	O	VL-Bus Ready Return
REFRESH*	I/O	ISA-Bus Refresh

Figure 8 NxVL Alphabetical Signal Summary (Continued)

RESERVED	I/O	*Reserved*
RESET*	I	Global Reset
RESETCPU*	O	Processor Reset
RESETOUT*	O	System Reset
ROMCS*	O	ROM Chip Select
SA<1:0>	I/O	ISA-Bus Address Bits 1 and 0
SBHE*	I/O	ISA-Bus High Byte Enable
SDIR1*, SDIR2*	O	SD-Buffer Direction
SDOE0*,-1*,-2*,-3*	O	SD-Buffer Output Enable
SDOE01*	O	SD-Buffer Output Enable
SMEMR*	O	ISA-Bus System Memory Read
SMEMW*	O	ISA-Bus System Memory Write
SPKR	O	Speaker Data
TURBO	I	Turbo Switch
VLASADIR*	O	VL-to-ISA Address Direction
WE*	O	Write Enable
XACK*	I	Transfer Acknowledge
XBCKE*	I	NexBus-Transceiver Clock Enable
XBOE*	I	NexBus-Transceiver Output Enable
XDEN*	O	XD-Buffer Enable
XDIR*	O	XD-Buffer Direction
XHLD*	I	Transfer Hold

Figure 8 NxVL Alphabetical Signal Summary (Continued)

Hardware Architecture

System Overview

The NxVL provides most of the systems logic required to implement a high performance Nx586/Nx587 based system. It is capable of interfacing to the 64-bit NexBus, the 32-bit VL-bus and the 16-bit ISA-bus. It contains the state machines which arbitrates bus-crossing operations between the Nx586 processor and masters or slaves on the VL-bus or ISA-bus. There can be caching devices on the VL-bus, but they must use a write-through caching policy. As the Nx586 processor's NexBus Arbiter, the NxVL arbitrates accesses by all masters on any bus to all system resources. This device also mediates the operations of an Integrated Peripherals Controller during interrupts, DMA operations, and other peripheral functions.

A single processor based system has the following basic parts as its minimum configuration:

- Nx586 Processor
- Nx587 Floating-Point Coprocessor (Optional)
- NxVL System Controller
- 82C206, Integrated Peripherals Controller
- Keyboard Controller, such as the 8742 or 8042
- BIOS EPROM, such as the 27C010
- 2MB to 256MB DRAMS for main memory
- Bus transceivers and latches

These sub-systems or components are connected in the manner shown in Figure 9. The details of the NexBus and its timing information are covered in the *Nx586 Processor and Nx587 Floating Point Coprocessor Databook.* A complete

design example is described at the end of this chapter, and schematics are available from NexGen.

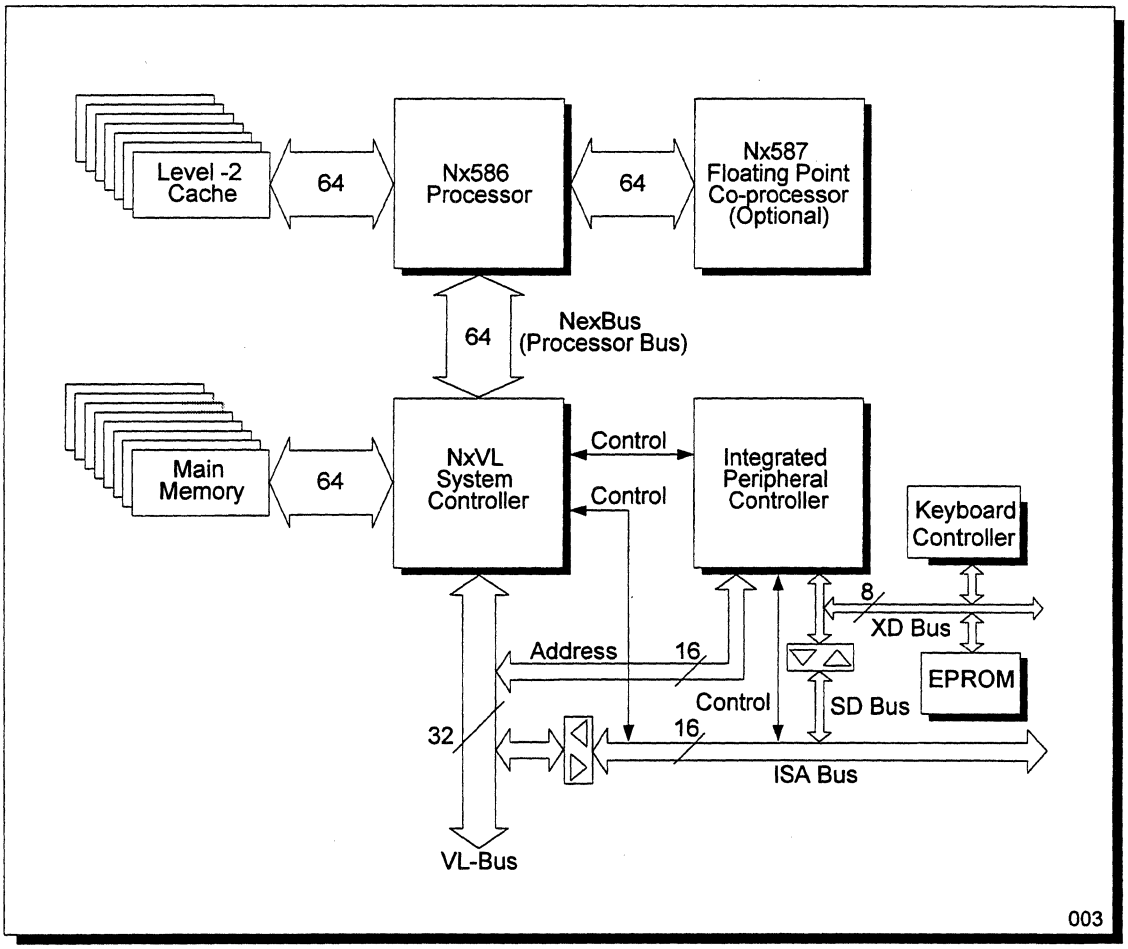


Figure 9 System Block Diagram

The Nx586 processor supports three 64-bit buses: The NexBus (the processor bus), the level-2 cache SRAM bus, and the Floating Point Coprocessor bus. The NxVL is attached to the NexBus. As shown in Figure 9, the NxVL is utilizing its internal NexBus registered transceivers for a single processor design (BUFMODE pin is tied high). The level-2 cache is the Nx586's external cache system. The level-2 cache is also known as L2-cache, second-level cache or secondary cache.

Three types of memory can exist in an NxVL system:

- Memory attached to the NxVL Memory-bus *(main or local memory)*
- Memory on the VL-bus
- Memory on the ISA-bus

The terms *memory, main memory or local memory*, unless otherwise specified, refers only to the storage memory devices attached to the NxVL Memory-bus. From the Nx586 processor's perspective, main memory exists in a hierarchy of other memory structures. In addition to the processor's level-2 cache structure, the processor has storage structures between it and main memory that contribute directly to the speed of accessing data: a prefetch queue, a branch prediction cache (BPC), and a write-reservation queue. The NxVL also maintains a prefetch queue between the level-2 cache and main memory that continuously pre-loads two eight-byte cache blocks in anticipation of the processor's next request for a cache fill.

Figure 10 shows the organization of cache and memory during a read cycle. Figure 11 shows the analogous organization during a write cycle. All levels of cache and memory are interfaced through 64-bit data buses. Physically, transfers between the L2-cache and main memory go through the processor via NexBus, and transfers between L1 and L2 cache go through the processor via the dedicated L2-cache bus. While the NexBus is multiplexed between address/status and data phases, the L2-cache data bus carries only data at 64 bits every NexBus clock cycle. The disk subsystem and software disk caches are included in the figures for completeness of the hierarchy; software disk caches are maintained in memory by some operating systems. Bus masters on VL-bus can maintain caches, but they must be write-through (not write-back) caches. Bus masters on the ISA-bus cannot maintain caches that map main memory.

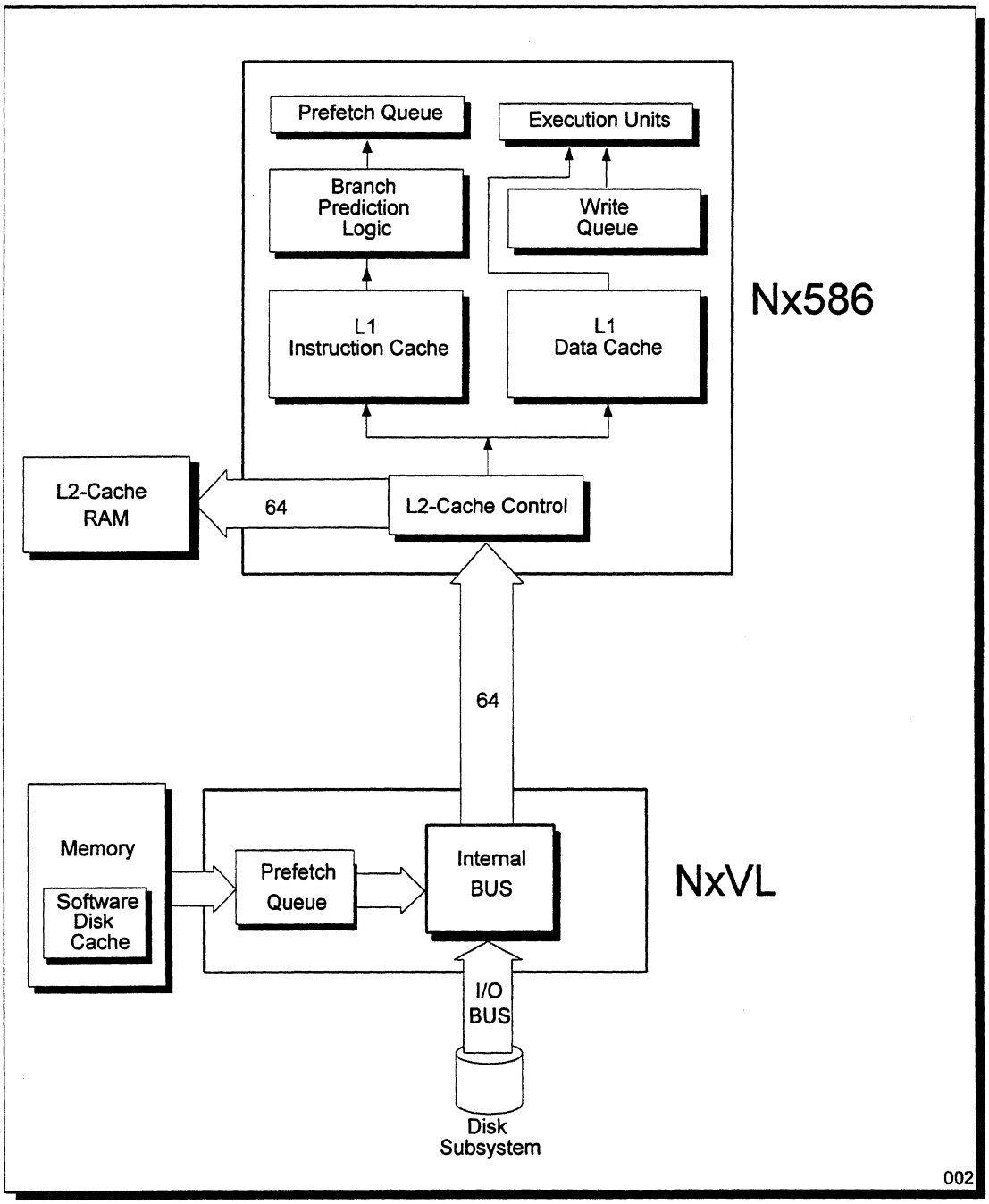


Figure 10 Storage Hierarchy (Nx586 Reads)

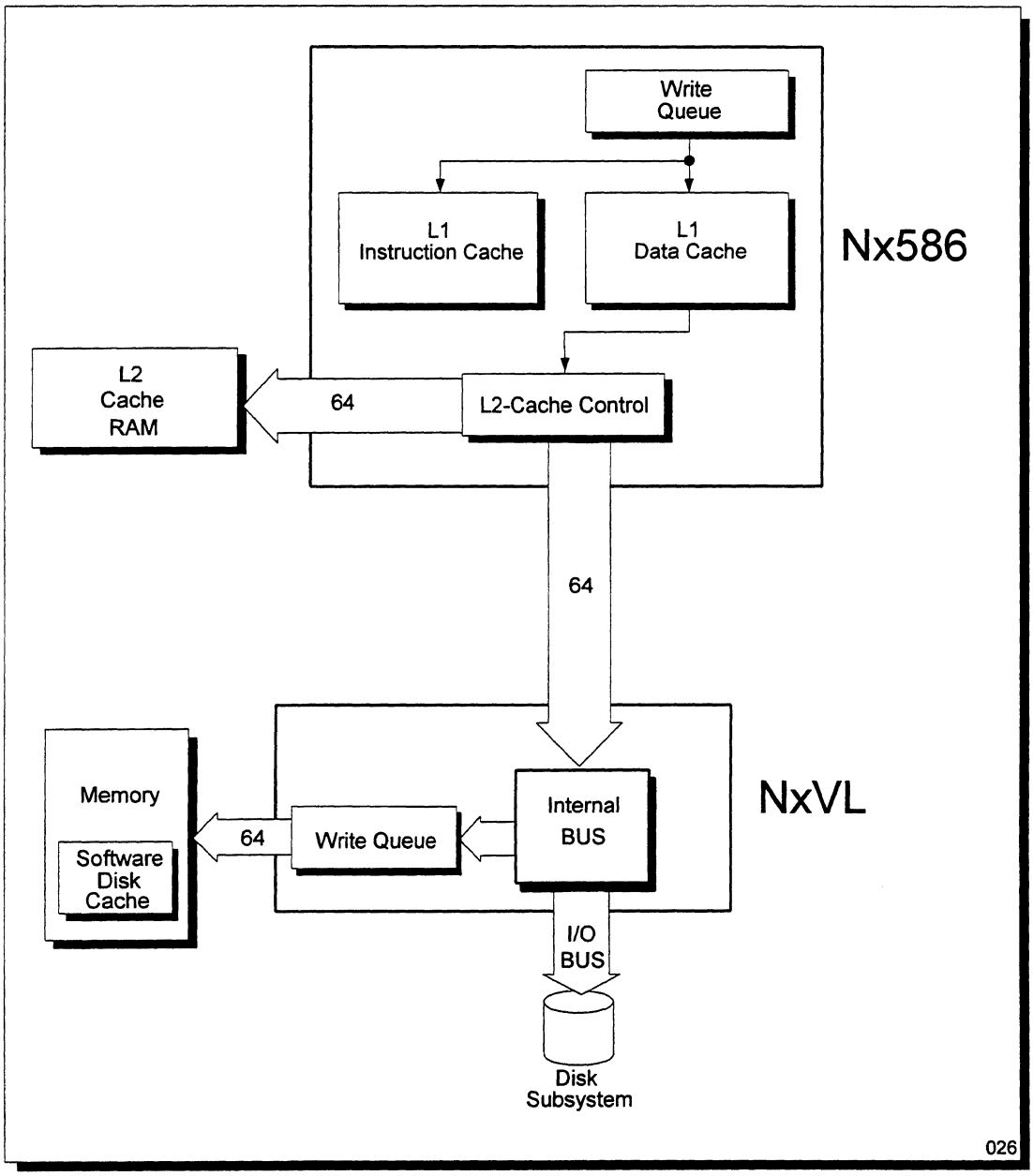


Figure 11 Storage Hierarchy (Nx586 Writes)

Internal Architecture

The various blocks of the NxVL's internal architecture are shown in Figure 12. These include blocks for the NexBus interface, VL-bus interface, ISA-bus control, main memory write, read or prefetch queues for all buses, bus arbitration, memory arbitration, memory control (including refresh), and snoop control.

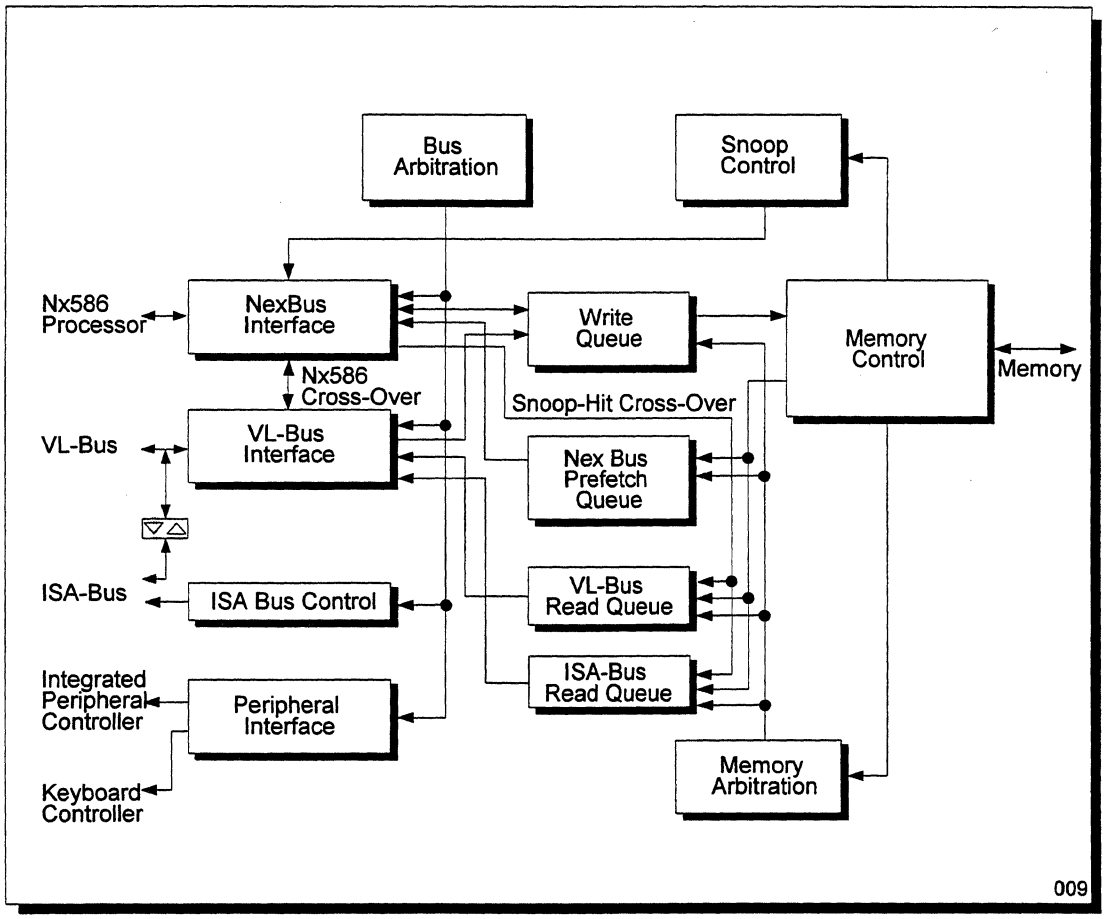


Figure 12 Internal Architecture

The 64-bit NexBus interface allows the Nx586 processor to access main memory via the NxVL. A cross-over path to the VL-bus also allows the processor to read and write devices on either the VL-bus or ISA-bus.

The 32-bit VL-bus interface allows VL-bus masters and ISA-bus masters to access main memory. It also allows the Nx586 processor and ISA-bus masters to access VL-bus memory.

The 16-bit ISA-bus interface operates in tandem with the interface to the industry standard Integrated Peripherals Controller, or IPC. These interfaces allow ISA masters and the IPC DMA controller to access main memory and VL-bus memory. Other ISA functions such as DMA control, interrupt control, and timers are handled by the IPC.

All memory or I/O writes—whether by tbe Nx586 processor or any other master—are buffered in the NxVL. A write queue and three read queues (one for each bus) sits between main memory and the bus interfaces. The write queue buffers writes to main memory by masters on all buses, allowing the master to continue with other work after filling the queue. The three read queues—the NexBus prefetch queue, the VL-bus read queue, and the ISA-bus read queue—buffer fetched main memory locations for each separate bus. Like cache within memory, they minimize latency for reads.

The bus arbitration logic supports one Nx586 processor, three VL-bus masters, and one Integrated Peripherals Controller (which in turn supports up to seven DMA masters, including six ISA-bus masters). The bus arbitration logic also handles arbitration between buses during bus-crossing transfers and snoop cycles.

The memory control logic generates the DRAM control signals (RAS, CAS, write-enable, and address) during accesses to main memory from all buses. The memory arbitration block handles simultaneous accesses and refresh to main memory. The ISA memory refresh logic on the NxVL ensures that memory on the ISA-bus is properly refreshed, no matter which master currently controls the buses.

Snoop cycles are performed on the NexBus whenever a VL-bus or ISA-bus master reads main memory. The snoop-control logic performs this function. It requests the NexBus and translates the cycle onto the NexBus. This causes the Nx586 processor to compare the address with data in its caches and take any actions required by the processor's MESI cache-coherency protocol, such as writing back modified data to memory in an intervenor operation and/or invalidating a cached copy of the data. If a write-back occurs, the VL-bus or ISA-bus master that initiated the read cycle can (if so configured) read the snoop-hit data on the fly as it is being written back to main memory by the Nx586 processor. A snoop-hit cross-over path between the NexBus interface and the VL-bus and ISA-bus read queues is provided for this purpose.

Main-Memory Write Queue

A 64-byte write queue or FIFO (First-In-Firs-Out) buffer, shown in Figure 12, interfaces the NexBus, VL-bus, and ISA-bus to main memory. This FIFO is organized as eight qwords. Each qword has an associated address tag, byte-enable bits, and decoded row-select bits. The queue buffers all writes to memory: single-qword and four-qword block writes from the Nx586 processor, single-dword or 16-byte block writes from VL-bus masters, and single byte or word writes from the ISA-bus master or the IPC DMA function.

A set of FIFO status signals are checked during write cycles. They indicate whether the write queue is empty, half-full, or full. If the queue is full, the write is blocked at the bus-interface until the queue is emptied.

During a write cycle, both the data and its related address information are written into the write queue. By retaining this information in the write queue, the write queue can be emptied to memory more quickly. When the write queue gains access to memory, no decoding is needed; the queue contains all of the information it needs to write to memory.

Data can be written into the queue at the rate of one clock per qword. Transfers from the write queue to memory occur at a slower rate: either two or three cycles per qword, depending on the clock speed and the configuration of DRAM operation. During write-queue transfers to memory, the queue can simultaneously be read on the fly into the NexBus prefetch queue or the VL-bus or ISA-bus read queue in two clocks per qword.

The FIFO status indicators are also checked during snoop cycles, which are initiated by read cycles. These indicators determine whether the write queue contains the location being read. If it does, the data in the write queue is first written to memory before being read from memory by the master that initiated the read.

Bus Structure

NxVL-Support for bus transactions includes:

- Main-memory accesses by masters on any of the buses
- Bus snooping by the Nx586 processor or a VL-bus master
- Bus-crossing operations, in which:
  - — the Nx586 processor can access resources on any other bus
  - — a VL-bus master can access resources on the ISA-bus, or
  - — an ISA-bus master can access resources on the VL-bus

Of the three buses, the NexBus and VL-bus are synchronous to the NexBus clock. The ISA-bus, which is derived through bus transceivers from the VL-bus, operates asynchronously to the NexBus clock. The structures and operating characteristics of the three buses are described in the following.

NexBus

The 64-bit synchronous NexBus supports all signals and bus protocols needed for cache-coherent multiprocessing, although only one Nx586 processor is supported by the NxVL device. The NxVL reflects all addresses accessed on the VL-bus and ISA-bus to the NexBus, so that the processor can monitor (snoop) the NexBus to guarantee coherency between the processor's caches and main memory.

The NexBus is multiplexed. An address-and-status phase is interleaved with a data phase, with a guaranteed dead clock cycle between the two phases. The dead cycle simplifies system design by providing time for address-driving devices to get off the bus before data-driving devices get on, without requiring systems logic to monitor these states. The NxVL supports an average sustainable read and write bandwidth on NexBus of 152 MBytes per second for the 66MHz Nx586 processor, and a peak transfer rate of 267 MBytes per second for the 66MHz Nx586 processor.

Figure 13 shows the address/status and data phases on the NexBus.

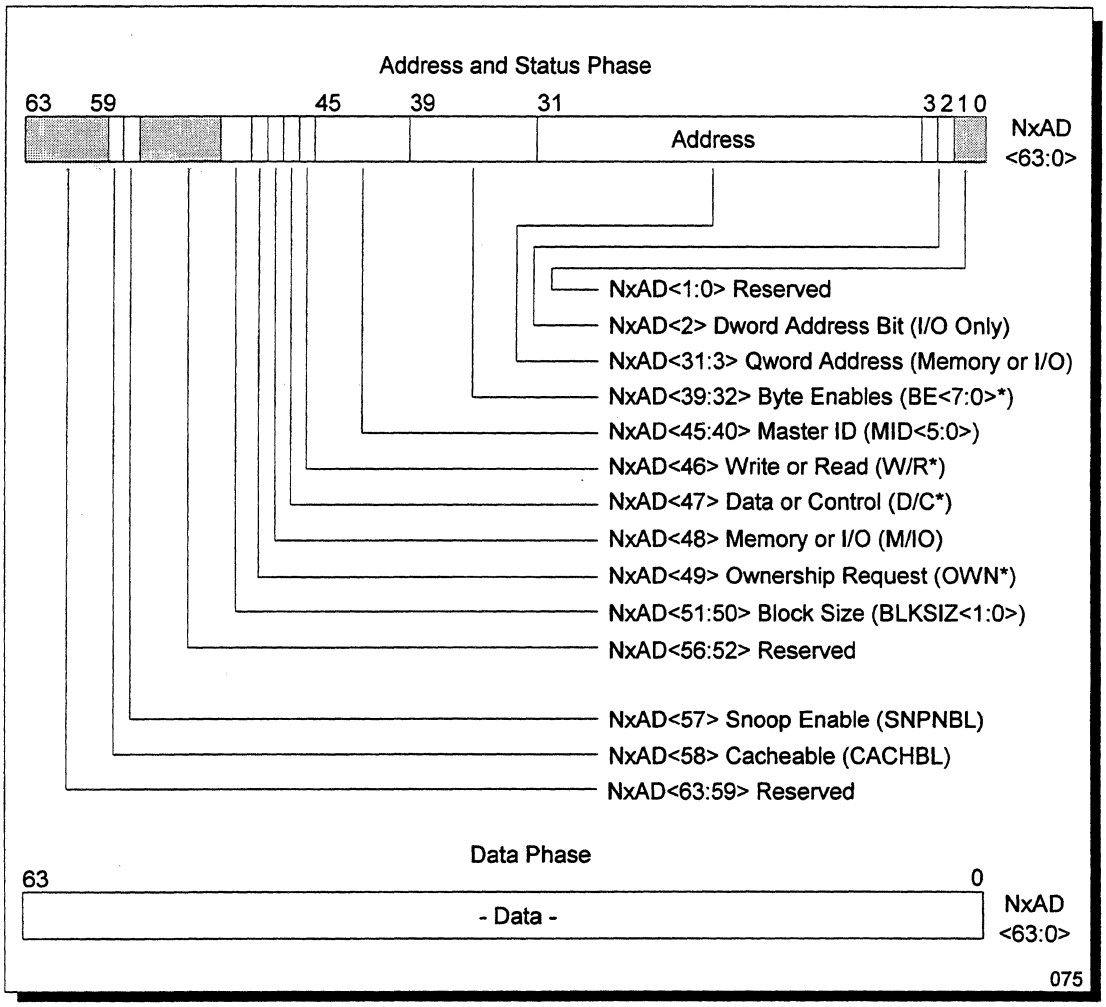


Figure 13 NexBus Address/Status and Data

The NexBus clock (CLK) drives not only the Nx586 processor but also the NxVL chip and the VL-bus devices. Internally, the NxVL chip uses both phases of the clock for certain external signals. Memory-read timing, for example, can operate in one of two programmable modes: in the slow memory-read mode, CAS assertion on reads takes two clocks and deassertion takes one clock; in the fast memory-read mode, however, CAS assertion on reads takes 1.5 clocks and deassertion takes 0.5 clock.

The NexBus is a buffered bus, designed with all the signals and bus protocols needed for multiprocessing or multimasters. Many types of devices can be interfaced to the NexBus, including a backplane, one or more processors (with optional coprocessors), one or more memory subsystems shared between processors, high-speed I/O devices, a NexBus Arbiter, and a NxVL to interface to other system buses (called an alternate-bus interface, ISA and VL.) In a multiprocessing environment, Nx586 processors are normally attached to the NexBus through registered transceivers (such as 29FCT52Bs). NxAD<63:0> is on the processor side of the transceivers, and AD<63:0> is on the shared-bus side of the transceivers. Control signals for the transceivers (XBCKE* and XBOE*) are provided by the Nx586 processor.

However, in single-processor systems designed with the NxVL, the external registered transceivers are not necessary. The registered transceivers are integrated into the NxVL and are enabled when the BUFMODE pin is pulled high. Connect XBCKE* and XBOE* from the Nx586 to the NxVL for a single processor design. Figure 14 shows the two alternative ways in which the NxVL can be connected to the Nx586 processor. The NxVL is fully capable of emulating the function of the NexBus transceivers and, while the external transceivers are optional, they are typically not used in a single processor design. For additional information on the transceivers, see the *Nx586 Processor and Nx587 Floating Point Coprocessor Databook.*

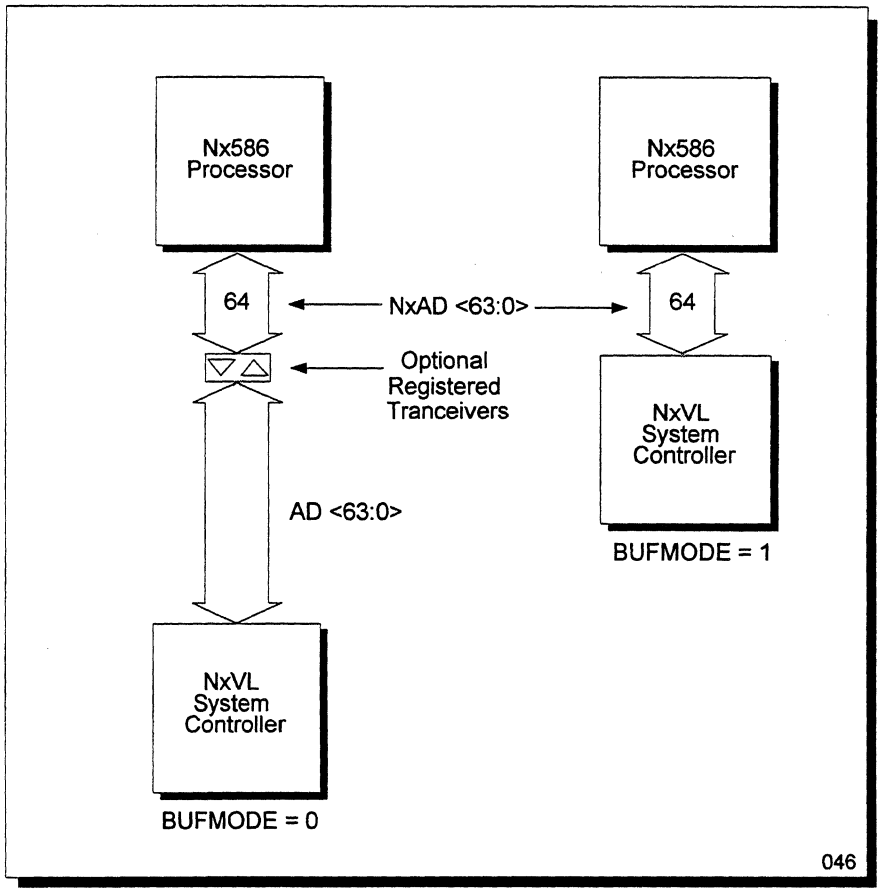


Figure 14 NxVL Connection Alternatives to the NexBus

VL-Bus

The 32-bit synchronous VL-bus is supported by the NxVL. The VESA standard for VL-bus is derived from the i386™ Local Bus and the i486™ Processor Bus; it is sometimes called the *local bus.*

The NxVL translates cycles to VL-bus slaves that are initiated by the Nx586 processor, an ISA-bus master, or the IPC DMA controller. Up to three VL-bus masters are supported. All of them can perform 16-byte burst transfers to or from main memory, and all can cache main-memory data if a write-through caching policy is used. The NxVL translates addresses from the VL-bus onto the NexBus so that the Nx586 processor can snoop main-memory accesses by VL-bus masters.

Figure 15 shows the addresses and data on the VL-bus. Figure 17 shows the complete set of VL-bus signals, indicating the subset that the NxVL supports and the directionality for the NxVL and VL-bus masters and slaves. For detailed specifications of the VL-bus, see the *VL-Bus™ Proposal* by the Video Electronics Standards Association (VESA).

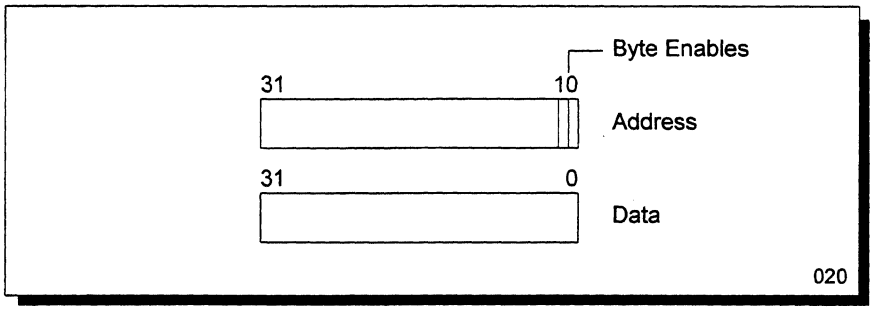


Figure 15 VL-Bus Address and Data

ISA-bus

Unlike the VL-bus, the 16-bit asynchronous Industry-Standard Architecture (ISA) bus—with its derivatives XA and XD and buffered SA and SD buses—does not have its own data path through the NxVL. Instead, the ISA-bus addresses and data are derived from the VL-bus. The NxVL chip generates only the ISA-bus clock (ISABCLK), control signals, and two address bits: SA<1:0>. The ISABCLK clock runs at a programmable division of the NexBus clock (CLK) frequency or OSC14M divided by two. The source of ISABCLK is determined by the BCLKSEL pin. ISABCLK is equal to OSC14M divided by two when BCLKSEL is tied low. Other ISA functions such as DMA control, interrupt control, and timers are provided by an Integrated Peripherals Controller, which can support up to three 8-bit ISA-bus masters and three 16-bit ISA-bus masters using its DREQ <3:1> and DREQ<7:5> signals. The NxVL supports cycles by the Nx586 processor or VL-masters to ISA-bus slaves, cycles by ISA masters to main memory or VL-bus devices, DMA accesses from ISA I/O to main memory or to VL-bus memory, and refresh cycles for ISA-bus memory.

Figure 16 shows the addresses and data on the ISA-bus. Figure 18 shows the complete set of ISA-bus signals, including the subset that the NxVL supports directly. For detailed specifications of the ISA-bus, see Edward Solari's book, *AT Bus Design,* published by Annabooks.

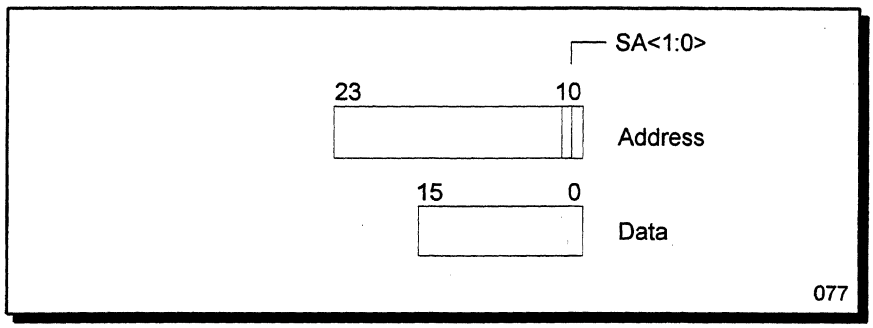


Figure 16 ISA-Bus Address and Data

Figure 9 shows the interfacing between the VL-bus and ISA-bus. These buses are connected via transceivers and buffers, plus a high-byte DMA-address latch in the NxVL. Data is transferred between the LBDATA<31:0> bus and the SD<15:0> bus via five data-bus transceivers—one for each byte of the VL-bus, plus a fifth to wrap 16-bit transfers onto an 8-bit device, as shown in Figure 19. DMA data reaches the ISA-bus through a data-bus transceiver between the XD and SD buses, as shown in Figure 21. Address are transferred through transceivers and buffers, as shown in Figure 20, plus the high-byte address latch that is built into the NxVL.

During DMA cycles, the DMA controller within the IPC drives the lower eight bits of the DMA address plus some of the upper bits, as follows: during 8-bit DMA cycles, the IPC first sends address bits <15:8> to the NxVL latch on the XD<7:0> bus and strobes the latch with its ADSTB8 signal. Then, the IPC DMA controller outputs address bits <23:16> and <7:0> while driving the latched bits with its AEN8* signal. For 16-bit DMA, the IPC first sends address bits <16:9> to the NxVL latch on the XD<7:0> bus and strobes the latch with its ADSTB16 signal. Then, the IPC DMA controller outputs address bits <23:17> and <8:1> (bit 0 is always 0) while driving the latched bits with its AEN16* signal.

A 32-bit transfer from the Nx586 processor to an alternate-bus slave involves one cycle on the VL-bus but either two or four cycles on the ISA-bus, depending on whether the ISA slave is a 16-bit or 8-bit device. The size of the slave, specified by the MEMCS16* signal, becomes known only after the first ISA-bus transfer. Either a memory slave drives MEMCS16* (IOCS16* for an I/O slave) within the time limit specified by the ISA standard, thereby indicating a 16-bit device, or it does not, thereby indicating an 8-bit device. In the first ISA-bus cycle, the lower two transceivers (labeled "Byte 0" and "Byte 1" in Figure 19) are both enabled. At that point, the size of the ISA-bus slave is not yet known. By the next cycle, the size of the ISA slave becomes known from the state of MEMCS16*. If it is a 16-bit slave, the upper two bytes of data are sent to their transceivers (labeled "Byte 2" and "Byte 3" in Figure 19), and the

transfer finishes. If the slave is a 8-bit device, only the first byte is received through transceiver 0 on the first cycle, and another cycle is required to place the second byte on the lower eight bits of the ISA-bus. In this second cycle, transceiver 0 is turned off and transceiver 1 remains on while the bridge transceiver (labeled "8-Bit Wrap" in Figure 19) is also turned on. The third byte is sent by enabling transceiver 2 with all others off. Finally, the fourth byte is sent by enabling only transceivers 3 and 4.

VL-Bus Signal	*Description*	*NxVL Signal Name*	*NxVL*	*VL Master*	*VL Slave*
ADR<31:2>	Local-Bus Address	LBADR<31:2>	I/O	O	I
ADS*	Local-Bus Address Strobe	LADS*	I/O	O	I
BE<3:0>	Local-Bus Byte Enables	LBE<3:0>*	I/O	O	I
BLAST*	Burst Last	same	I/O	O	I
BRDY*	Burst Ready	same	O	I	U
D/C*	Local-Bus Data or Code	LBD/C*	I/O	O	I
DAT<31:0>	Local-Bus Data	LBDATA<31:0>	I/O	I/O	I/O
ID<4:0>	Host Identifiers	—	—	I	I
IRQ9	Interrupt Request Line 9	—	—	O	O
LBS16*	Local-Bus Size 16	same	I	I	O
LCLK	Local CPU Clock	CLK	I	I	I
LDEV<2:0>*	Local-Bus Device	same	I	—	O
LEADS*	Local External Address Data Strobe	—	—	O	I
LGNT<2:0>*	Local-Bus Grant	same	O	I	—
LKEN*	Local Cache Enable	—	—	I	I
LRDY*	Local-Bus Ready	same	I/O	I	O
LREQ<2:0>*	Local-Bus Request	same	I	O	O
M/IO*	Local-Bus Memory or I/O	LBM/IO*	I/O	O	I
RDYRTN*	Ready Return	same	O	I	I
RESET*	Global Reset	same	I	I	I
W/R*	Local-Bus Write or Read	LBW/R*	I/O	O	I
WBACK*	Write Back	—	—	I	—

Figure 17 VL-Bus Signals

ISA-Bus Signal	*Description*	*NxVL Signal Name*	*NxVL*	*ISA Master*	*ISA Slave*
AEN	Address Enable	same	O	—	I
BALE	Bus Address Latch Enable	same	O	—	I
DACK*	DMA Acknowledge	—	—	I	—
DRQ	DMA Request	—	—	O	—
IOCHCK*	I/O Channel Check Error	same	I	O	O
IOCHRDY	I/O Channel Ready	same	I/O	I	O
IOCS16*	I/O Chip Select 16	same	I	I	O
IOR*	I/O Read	same	I/O	—	I
IOW*	I/O Write	same	I/O	—	I
IRQ	Interrupt Request	—	—	O	O
LA<23:17>	High Address Lines	LBADR<23:17>	I/O	O	I
MASTER*	ISA-Bus Master	same	I	O	—
MEMCS16*	ISA-Bus Memory Chip Select 16	same	I/O	I	O
MEMR*	ISA-Bus Memory Read	same	I/O	O	I
MEMW*	ISA-Bus Memory Write	same	I/O	O	I
OSC	ISA-Bus Clock	ISABCLK	O	I	I
REF*	ISA-Bus Refresh	REFRESH*	I/O	I/O	I
RESETDRV	ISA-Bus Reset	—	—	I	I
SA<19:2>	ISA-Bus Address	LBADR<19:2>	I/O	O	I
SA<1:0>	ISA-Bus Address	SA<1:0>	O	I	I
SBHE*	ISA-Bus High Byte Enable	same	I/O	—	I
SD<15:0>	ISA-Bus Data	LBDATA<15:0>	I/O	I/O	I/O
SMEMR*	ISA-Bus System Memory Read	same	O	—	I
SMEMW*	ISA-Bus System Memory Write	same	O	—	I
SYSCLK	ISA-Bus System Clock	ISABCLK	O	I	I
TC	Terminal Count	—	—	—	I
SRDY*	Zero Wait State	—	—	—	O

Figure 18 ISA-Bus Signals

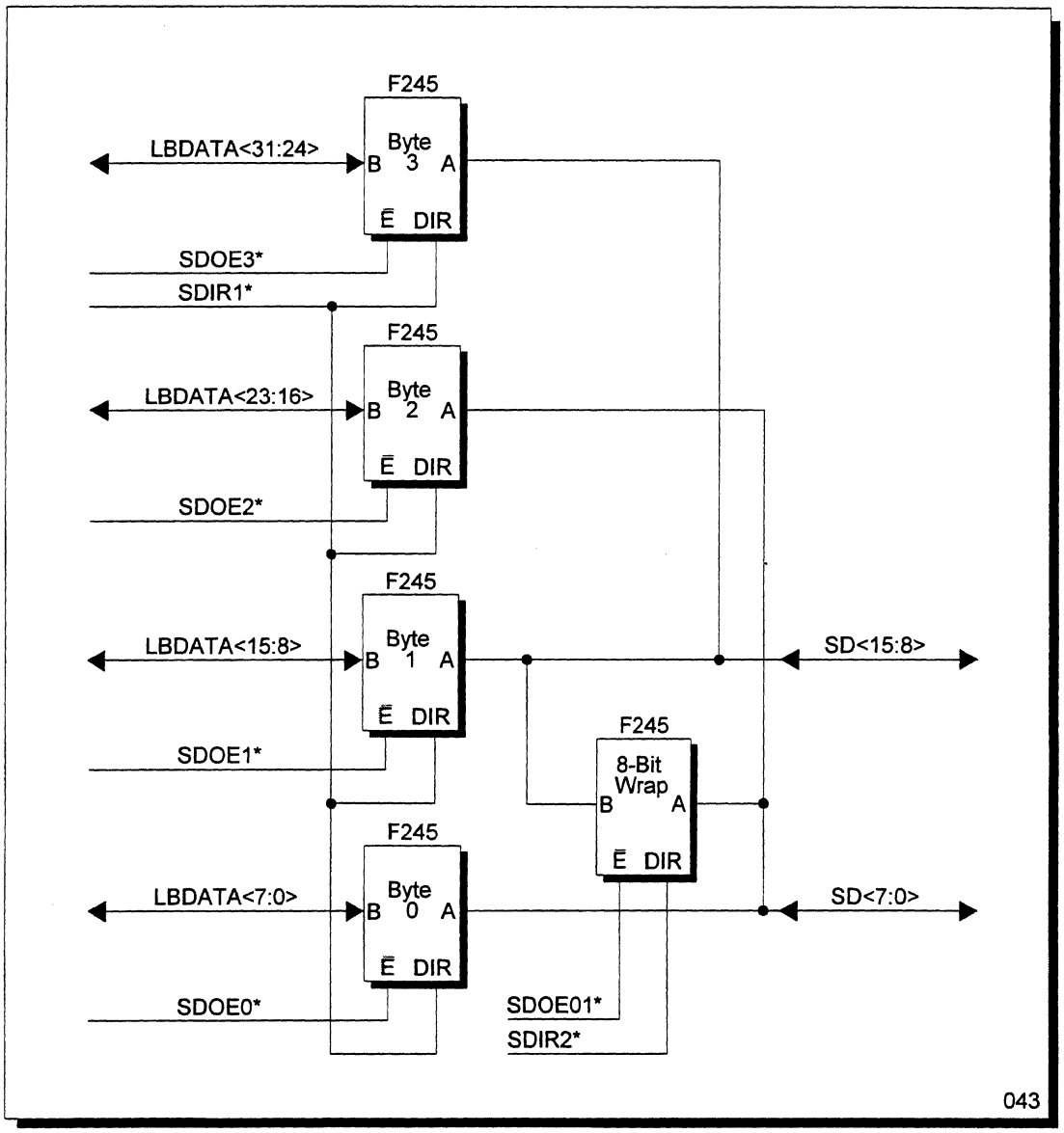


Figure 19 VL-Bus to ISA-Bus Data Transceivers

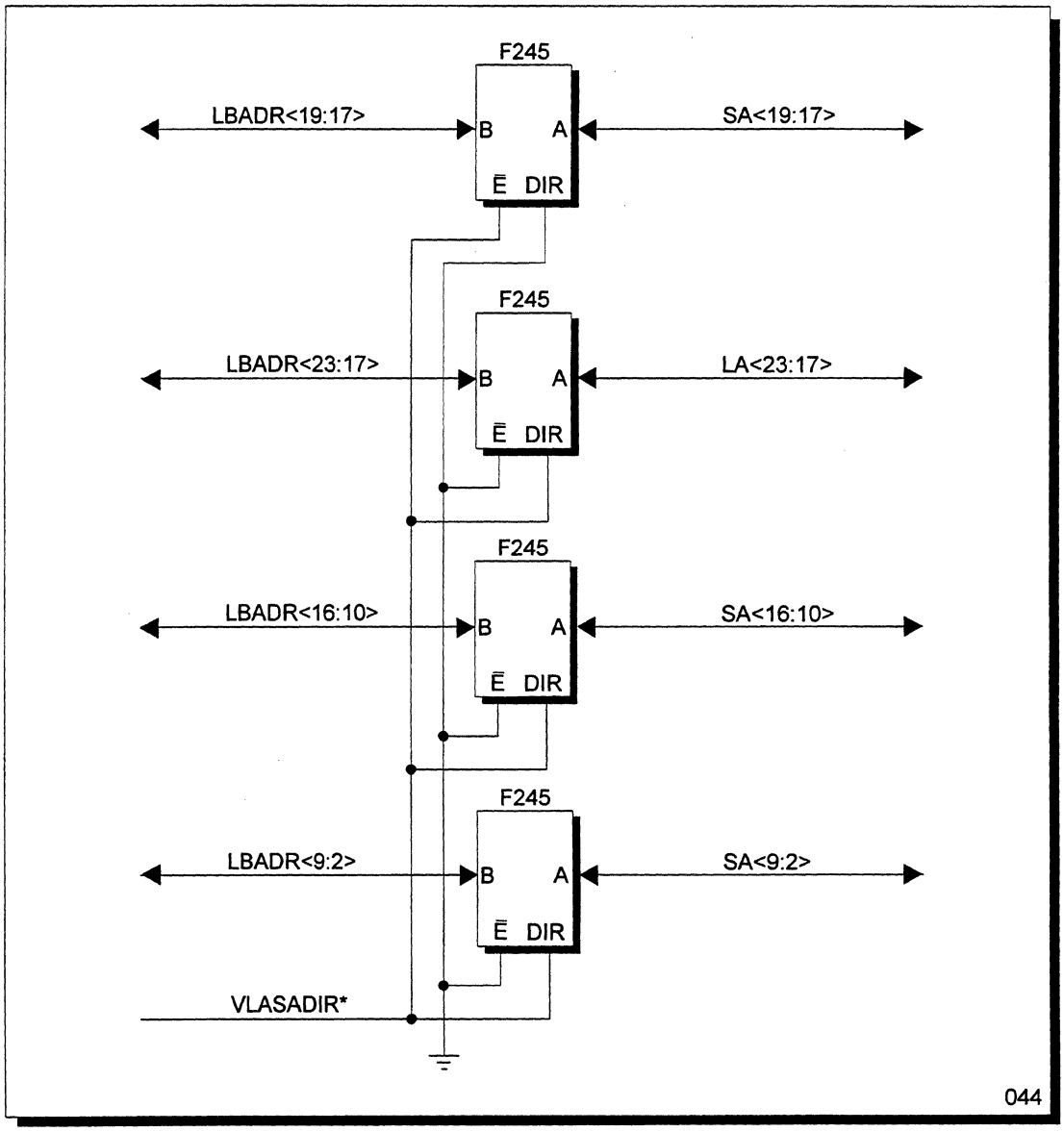


Figure 20 VL-Bus to ISA-Bus Address Transceivers

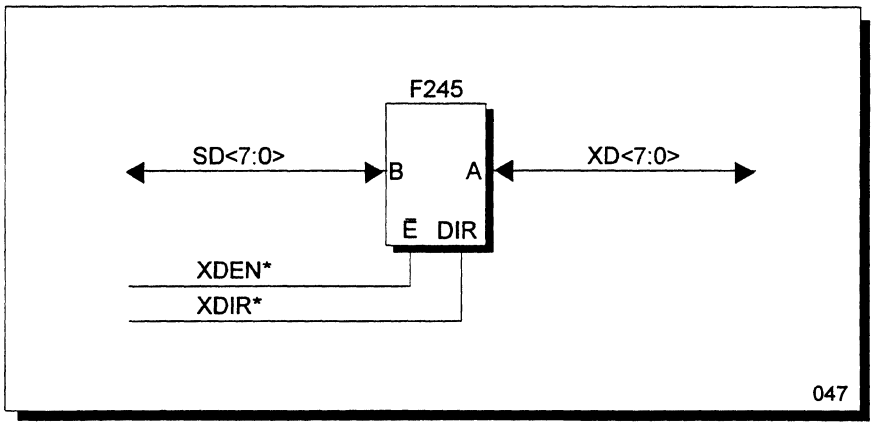


Figure 21 IPC SD-Bus to XD-Bus Data Transceiver

Bus Arbitration

A bus is said to be *owned* by a master when that master can initiate operations (cycles) on the bus. In fact, the master to which bus ownership is granted only controls its own interface with the NxVL. The NxVL, on behalf of that master, controls main memory and the VL-bus and ISA-bus.

The NxVL arbitrates between competing masters with a global bus arbiter, which grants access to all buses and main memory as a group. It can arbitrate bus ownership between the following requesting sources:

- *Nexbus*—Requests from one Nx586 processor (and its companion Nx587 coprocessor) through the NREQ*, AREQ*, and /or LOCK* signals.
- *VL-bus*—Requests from up to three VL-bus masters through the LREQ(2:0)* signals.
- *ISA-bus*—Requests mediated by the 82C206 peripherals controller through its HHOLD signal for (a) DMA by the 82C206 peripherals controller, (b) ISA-bus memory refresh by an ISA-bus master through the REFRESH* signal, or (c) ISA-bus control by an ISA-bus master through its MASTER* signal.

At any time, one master controls all three buses and main memory. In terms of overall prioritization, all buses have equal priority, although the Nx586 processor has twice as many chances to obtain the buses than do the VL-bus masters (as a group) or an ISA-bus master.

By default, the Nx586 processor is the master of all buses. It is given the buses (a) at reset, (b) when no other master is requesting them, or (c) when the Nx586 processor intervenes in a cycle initiated by another master so as to write back modified cache data. When the processor obtains the buses, it continues to own

them as long as it asserts one of its request signals (NREQ*, AREQ*, or LOCK*). The processor relinquishes the buses whenever it is not requesting them for its own use, and there is a request from a master on another bus. Between granting the buses to a VL-bus master or an ISA requesting source, the NexBus has an opportunity to regain the buses. Figure 28 in the chapter entitled *Bus Operations* shows the global bus-arbitration states.

Within this global arbitration scheme, each time the VL-bus rises to maximum global priority, only one of three VL-bus masters is granted the buses. A VL-bus arbiter manages a register that is loaded with pending VL-bus requests when the register is empty. All pending VL-bus requests are granted before the register is reloaded.

Bus-Crossing Operations

A *bus-crossing operation* is a bus cycle that originates on one bus and accesses a resource on another bus. If the addressed location of a cycle initiated by the NexBus processor, DMA controller, or ISA-bus master is not in main memory, the cycle is converted ("falls through") to a VL-bus cycle and thereby becomes a bus-crossing cycle. If a VL-bus device does not respond in a specified time period (one clock for 33MHz operation or lower, two clocks for 40MHz operation or above), the cycle falls through to an ISA-bus cycle.

Nx586 processor cycles to VL-bus or ISA-bus devices can be memory or I/O reads or writes. The NxVL handles all qword assembly and disassembly that may be required to match the different widths of buses. Block (burst) transfers are not supported on crossover cycles from the Nx586 processor to VL-bus or ISA-bus devices.

Write cycles by the Nx586 processor to VL-bus or ISA-bus memory or an I/O devices are buffered. Since the VL-bus or ISA-bus is already owned, the latency on finishing these writes is not large. The NxVL may, however, break the cycle into multiple cycles on the VL-bus or ISA-bus if there is a size mismatch.

Read cycles by the Nx586 processor from VL-bus or ISA-bus devices are held up until the requested read is performed on the VL-bus or ISA-bus. When the data is collected, the read cycle is terminated.

Memory

Main memory is attached to the NxVL via a dedicated 64-bit memory bus, and alternate memory can optionally be attached to the VL-bus or ISA-bus. In this book, the term *memory*, unless otherwise specified, refers only to the main memory attached to the NxVL.

Organization

The NxVL can support 2MB to 256MB of main memory. Memory options are based on memory-increment desired and number of buffers in the design. Main memory is organized in two 64-bit banks (A and B), plus eight bits of parity. Bank A is enabled by CASA<7:0>* and Bank B is enabled by CASB<7:0>*. Figures 22 through 25 show bank organizations for various memory module organizations. Figure 26 shows configurations that can be used with SIMM devices.

Bank	*Column Address*	*Module 1*	*Module 2*
Bank A	CASA<7:0>*	RAS0*	RAS0*
Bank B	CASB<7:0>*	RAS4*	RAS4*

Figure 22 Memory Banks—4 32-bit memory modules

Bank	*Column Address*	*Module 1*	*Module 2*
Bank A	CASA<7:0>*	RAS0*	RAS0*
		RAS1*	RAS1*
Bank B	CASB<7:0>*	RAS4*	RAS4*
		RAS5*	RAS5*

Figure 23 Memory Banks—4 32-bit interleaved memory modules

Bank	*Column Address*	*Module 1*	*Module 2*
Bank A	CASA<7:0>*	RAS0*	RAS0*
	CASA<7:0>*	RAS1*	RAS1*
Bank B	CASB<7:0>*	RAS4*	RAS4*
	CASB<7:0>*	RAS5*	RAS5*

Figure 24 Memory Banks—8 32-bit memory modules

Bank	*Column Address*	*Module 1*	*Module 2*
Bank A	CASA<7:0>*	RAS0*	RAS0*
		RAS1*	RAS1*
	CASA<7:0>*	RAS2*	RAS2*
		RAS3*	RAS3*
Bank B	CASB<7:0>*	RAS4*	RAS4*
		RAS5*	RAS5*
	CASB<7:0>*	RAS6*	RAS6*
		RAS7*	RAS7*

Figure 25 Memory Banks—8 32-bit interleaved memory modules

SIMM Type	*Memory Sizes of one SIMM*	*Number of SIMMs*
x36s	1, 2, 4, 8, 16, or 32 MB	Four
x36s	1, 2, 4, 8, 16, or 32 MB	Eight

Figure 26 SIMM Configurations

For details on the configuration of DRAMs and memory mapping, see the *Configuration and Testing* chapter.

Read/Write Reordering

Reads have higher priority than writes, except when there is a read to a pending write contained in the write queue. All three of the NxVL's read queues (the NexBus prefetch queue, VL-bus read queue, and ISA-bus read queue) support read-bypassing. That is, if a write is pending in the write queue when a read request is received, the read will be done first, if possible. However, if the read is to a location contained in the write queue, the contents of that queue are first written to memory before the read is performed. Prefetching of 32-byte blocks into the NexBus prefetch queue have lower priority than either reads or writes.

DMA Transfers

All transfers between main memory and any other device go through the NxVL. Transfers between main memory and masters on the VL-bus or ISA-bus are controlled by those masters and are referred to as bus-master transfers. When

there is no master on the ISA-bus, *DMA transfers* can be made between memory (main memory, VL-bus memory, or ISA-bus memory) and I/O on the ISA-bus. These transfers are controlled by the DMA controller within the IPC.

For information about DMA-cycle configurations, see a data sheet for an F82C206 integrated peripherals controller.

Bus Snooping and Cache Coherency

The Nx586 processor relies on its caches to support high performance. In addition, VL-bus masters can maintain caches, although these must use a write-through caching policy rather than the write-back policy used by the Nx586 processor. ISA-bus masters cannot maintain caches. The NxVL prefetch and read queues for all buses (the NexBus prefetch queue, the VL-bus read queue, and the ISA-bus read queue shown in Figure 12) snoop all writes to main memory from any other bus so as to invalidate stale read-queue entries.

The NxVL supports the Nx586 processor's cache-coherency protocol during memory accesses by any bus master. It does this by (1) initiating snooping cycles on the NexBus, and (2) maintaining a set of four snoop-tag registers with the four most-recently snooped addresses and their status with respect to the Nx586 cache. Whenever a VL-bus or ISA-bus master reads or writes memory, the bus master asserts signals that causes the NxVL to look the address up in its snoop-tag registers, and (if the location is not in the registers) initiate a snoop cycle on the NexBus. The snoop-tag registers prevents redundant snooping when any of the last four locations are repetitively accessed by the VL-bus or ISA-bus masters.

The Nx586 processor observes a modified, exclusive, shared, invalid (MESI) protocol. In this protocol, a cache block may be in one of four states:

- Exclusive—Data copied into a single bus-master's cache. The master then has the exclusive right (not yet exercised) to modify the cached data. Also called *owned clean* data.
- Modified—Data copied into a single bus-master's cache (originally in the exclusive state) but that has subsequently been written to. Also called *dirty, owned dirty,* or *stale* data.
- Shared—Data that may be copied into multiple bus-masters' caches and can therefore only be read, not written.
- Invalid—Cache locations in which the data is not correctly associated with the tag for that cache block. Also called *absent* or *not present* data.

For details on the MESI protocol, see the *Nx586 Processor and Nx587 Floating Point Coprocessor Databook.*

The address tags in the NxVL's write queue are snooped by masters on all buses (NexBus, VL-bus, ISA-bus) during read cycles. The VL-bus and ISA-bus read queues are snooped by the Nx586 processor during write cycles.

Design Example

Figure 9 at the beginning of the *Hardware Architecture* chapter shows a system block diagram for a typical design using the NxVL and a single Nx586 processor. Figure 27 lists the active components used in such a design. Detailed schematics for the design, plus supporting data files, are available from NexGen.

Quantity	*Part Type*
1	Nx586 Processor
8	SRAM Level-2 Cache
1	Nx587 Floating Point Coprocessor socket
1	NxVL System Controller
1	82C206 Integrated Peripherals Controller
1	BIOS EPROM
1	Keyboard Controller
8	72 pin SIMMs
8	Octal Buffers and Line Tri-state Drivers
10	Octal Bus Tri-state Transceivers
1	Quad 2-Input NAND gates
1	Hex Inverters
1	Hex Non-Inverting OC gates
1	CMOS Invertering gates
1	14.31818MHz Oscillator
1	NexBus Oscillator

Figure 27 Active Components in a Typical Single Processor Design

Bus Operations

This chapter describes the bus cycles on all of the buses supported by the NxVL. The operations on the VL-bus and ISA-bus cover primarily those performed by the NxVL bus control logic, not all possible operations on those buses. As in other chapters, the term "clock" refers to the NexBus clock. For details of operations on those buses, see the *VL-Bus™ Proposal*, published in 1992 by the Video Electronics Standards Association (VESA) and Edward Solari's book, *AT Bus Design, IEEE P996 Compatible*, published in 1990 by Annabooks, San Diego, CA.

Arbitration Protocols

To begin operation, a master requesting access must be granted all buses through the NxVL's arbitration process. The NxVL arbitration logic responds to the following types of bus-access requests:

- *NexBus*—Requests from the Nx586 processor through the NREQ*, AREQ*, and/or LOCK* signals.
- *VL-bus*—Requests from up to three VL-bus masters using the LREQ<2:0>* signals.
- *ISA-bus*—Requests mediated by the Integrated Peripherals Controller (IPC) through its HHOLD signal for
    (a) DMA controller within the IPC,
    (b) ISA-bus memory refresh by an ISA-bus master using REFRESH*, or
    (c) ISA-bus control by an ISA-bus master through its MASTER* signal.

Bus-Arbitration Protocol

Figure 28 shows a state diagram of the bus arbitration protocol. The states are:

- *Grant to Nx586 Processor*—This state is entered at reset and continues when the Nx586 processor continues to request bus accesses, or when no VL-bus or ISA-bus master is requesting access. This state is left if a VL-bus or ISA-bus master requests the buses while the Nx586 processor is not requesting or locking them. If the buses are granted to another master, the Nx586 processor has an opportunity to regain access after that master is finished and before another master can gain access.
- *Grant to Highest-Priority VL-Bus Master*—This state is entered when a VL-bus master requests the buses while the Nx586 processor is not requesting or locking them. The NxVL decides which of up to three VL-bus masters gets the buses. This state is left if the VL-bus master's request is withdrawn, or if a read by that master hits in the Nx586 cache and thereby initiates an intervenor (DCL*) cycle by the Nx586 processor.
- *Grant to ISA-Bus Master*—This state is entered when (a) an ISA-bus memory-refresh request is generated within the NxVL, or (b) the DMA-control logic in the IPC asserts HHOLD, while the Nx586 processor is not requesting or locking the buses. This state is left if the ISA-bus master's request is withdrawn or if a read by the ISA-bus master or a DMA controller hits in the Nx586 cache and thereby initiates an intervenor (DCL*) cycle by the Nx586 processor.
- *Grant to Nx586 for Intervenor (DCL*) Cycle*—This state is entered if any read hits in the Nx586 cache, thereby initiating a DCL* (intervenor) cycle by the Nx586 processor. This state is left when the Nx586 processor completes the DCL* cycle.

The conditions for state changes shown in Figure 28 are:

- NEXREQ—The Nx586 processor asserts AREQ*, NREQ*, or LOCK. Any of these signals has the same effect: it causes the arbiter to grant all three buses to the NexBus processor at the end of the current bus cycle.
- LOCK—The Nx586 processor asserts its LOCK* signal.
- VLREQ—The top-priority VL-bus master asserts its LREQ<*n*>* signal.
- ISAHOLDREQ—The NxVL's internal SA-bus refresh logic requests an ISA-bus memory refresh, or the DMA-control logic in the Integrated Peripheral Controller asserts HHOLD.
- LASTGNTTOISA—The last non-NexBus grant was given the ISA-bus master rather than to a VL-bus master.

- DCLREQ—The Nx586 processor asserts its DCL* (dirty cache line) signal when a snoop hit occurs.
- DCLCYCLE—The Nx586 processor performs an intervenor operation, in which it invalidates or writes back modified cached data.

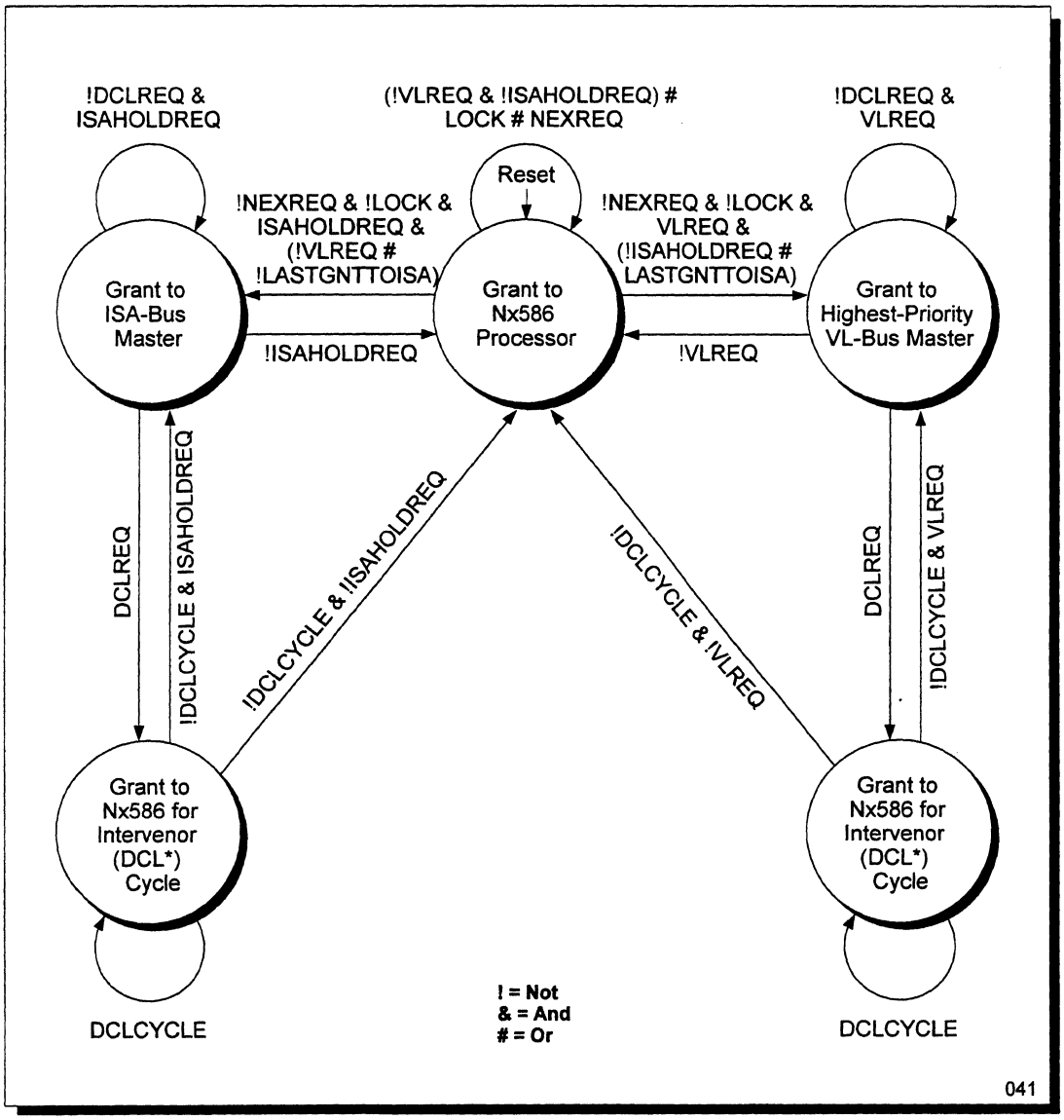


Figure 28 Global Bus Arbitration Protocol

The state diagram is symmetric between all three buses. The top-middle state, *Grant to Nx586 Processor*, is the default. This is the state in which the processor has access to all buses. From there, essentially the same transitions are shown for the ISA-bus (on the left side of the diagram) and the VL-bus (on the right side of the diagram). Figure 29 show the transition times from state to state.

From	*To*	*Clocks*
Grant to Nx586 Processor	Grant to ISA-Bus Master	2
Grant to Nx586 Processor	Grant to Highest-Priority VL-Bus Master	2
Grant to ISA-Bus Master	Grant to Nx586 for Intervenor (DCL*) Cycle	1
Grant to Nx586 for Intervenor (DCL*) Cycle	Grant to ISA-Bus Master	1
Grant to Nx586 for Intervenor (DCL*) Cycle	Grant to Nx586 Processor	1
Grant to Highest-Priority VL-Bus Master	Grant to Nx586 for Intervenor (DCL*) Cycle	1
Grant to Nx586 for Intervenor (DCL*) Cycle	Grant to Highest-Priority VL-Bus Master	1

Figure 29 Bus-Arbitration State Transition Times

The transition from *Grant to Nx586 Processor* to *Grant to Highest-Priority VL-Bus Master* indicates the invocation of the VL-bus arbiter. This arbiter decides which of up to three VL-bus masters (0, 1, or 2) gets the buses. The priority among VL-bus masters is circular. It begins with master 2, then goes to master 0, then to master 1, and back again to master 2. After one VL-bus master has secured control, another requesting VL-bus master must wait until both the Nx586 processor and any pending ISA request (in that order) have had an opportunity to secure control.

Arbitration by ISA masters is simpler than for VL masters. m The only sources of ISA-bus cycles are the ISA-bus memory-refresh logic within the NxVL, which generates refresh requests every 15μsec if so configured, or the Integrated Peripherals Controller which makes DMA requests by asserting HHOLD. If both requests are asserted, the ISA-bus memory-refresh requests receives higher priority.

Main-Memory Arbitration Protocol

The NxVL handles main-memory refresh, memory-access arbitration by any bus master, and prefetching to the NexBus prefetch queue. Figure 30 shows these types of accesses and their priorities. Reads have higher priority than writes, except when there is a read to a pending write location contained in the write queue (see Figure 12 at the beginning of the *Hardware Architecture* chapter).

Priority	*Memory Access Type*
1	Memory refresh when a read or write request is pending.
2	Reads, except when the read is to a pending write location, in which case the write is done before the read.
3	Writes.
4	Memory refresh when no read or write request is pending.

Figure 30 Main-Memory Access Priorities

Figure 31 shows a state diagram for memory arbitration. At reset, the NxVL chip comes up in the idle state. The conditions for state changes are:

- REFRESHREQ—A main-memory refresh request generated within the NxVL.
- READREQ—A read request from any bus master to its associated NxVL read queue (i.e., to the NexBus prefetch queue, the VL-bus read queue, or the ISA-bus read queue).
- WRITEREQ—A write request from any bus master to the NxVL's write queue, including write-backs by the Nx586 processor during snoop hits.
- PENDINGWRITE—A WRITEREQ to the same location as a READREQ.

Block prefetches to fill the NexBus prefetch queue, in response to the queue being half-empty, are considered as an integral part of NexBus cache fills and, as such, are treated as part of the read requests.

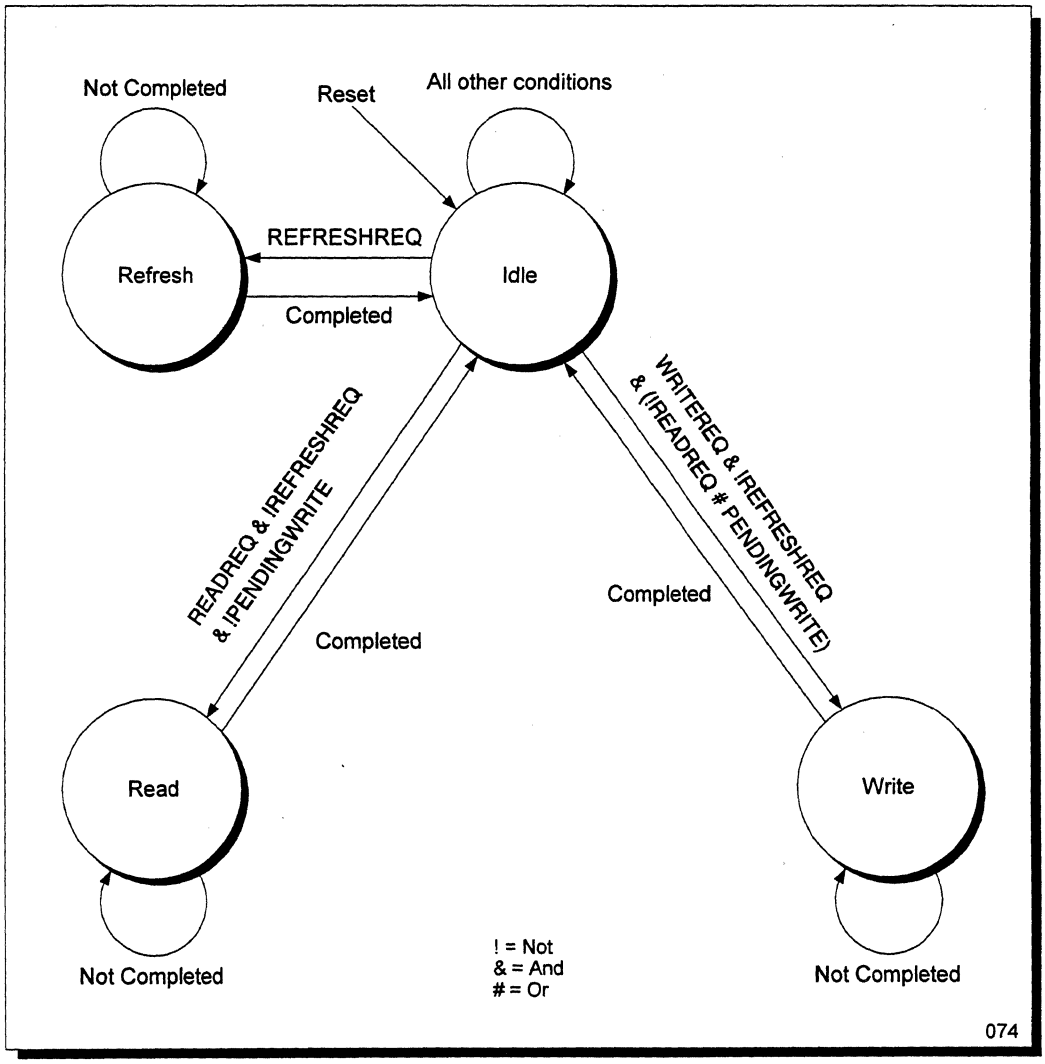


Figure 31 Main-Memory Arbitration State Diagram

Nx586 Processor Operations

With the NxVL, the Nx586 processor can perform single or block (burst) reads and writes to main memory. The processor can also perform single reads and writes to memory on the VL-bus and to memory or I/O on the ISA-bus.

A processor read or write that lies outside the main-memory address range specified in the configuration and mapping registers (explained in the *Configuration and Testing* chapter) is diverted first to the VL-bus and subsequently to the ISA-bus in search of a device that supports that address. Such a cycle is said to fall through to the VL-bus, and subsequently to the ISA-bus. The NxVL handles all word assembly and disassembly required to match different bus widths.

All writes—whether by the Nx586 processor or any other master—are buffered in the NxVL, whether they are to memory or to I/O. The processor latency for main-memory and bus-crossing writes is quite short. During bus-crossing reads, however, the processor is held up until the requested read is performed on the VL-bus or ISA-bus.

Bus Arbitration, Address Phase, and Data Phase

Figure 32 shows the processor's bus arbitration and address phase. The notation regarding Source in the left-hand column of the figure indicates the device or logic that generates the signal. In some timing diagrams, bus signals take on different names as outputs cross buses through transceivers or are logically ORed in group-signal logic. In these cases, the signal source is shown with a subscript, where the subscript indicates the device or logic that originally caused the change in the signal.

In Figure 32, the processor asserts NREQ*, LOCK*, and/or AREQ* to obtain control of all system buses. For bus-granting purposes, the NxVL treats NREQ*, AREQ*, and LOCK* identically. The NxVL arbiter responds in the next clock by asserting GNT*, if no VL or ISA master is currently using the buses (and excluding cases in which the processor needs bus access to write back snooped data, as will be described later). Automatic re-grant of the buses occurs when the NxVL holds GNT* asserted at the time the processor samples it, in which case the processor need not assert NREQ*, LOCK*, or AREQ* and can immediately begin its operation. The NxVL holds GNT* asserted in this manner when no VL or ISA master is requesting the buses.

NREQ*, when asserted, remains active until GNT* is received from the NxVL, although during speculative reads the processor will negate NREQ* before GNT* is received if the transfer is no longer needed. AREQ* is asserted by the processor when it knows that it must secure control not only of the NexBus but also of the VL-bus or ISA-bus. Unlike NREQ*, the processor does not make speculative requests for the VL-bus or ISA-bus with AREQ*. The NxVL does

not assert GNT* in response to AREQ* until the required bus is available. AREQ* and NREQ* have the same effect in the sense that either one causes the NxVL arbiter to grant all buses to the winning requester at the end of the current bus cycle. However, AREQ* locks the use of the buses until the processor negates the signal. LOCK* is asserted by the processor to sustain a bus grant that was obtained by the assertion of NREQ* or AREQ*. The signal is used by the NxVL logic to determine the end of a bus sequence. Cache-block fills (32 bytes) are not locked; they are implicitly treated as atomic reads.

The timing for bus cycles depends on whether the BUFMODE signal is tied high or low:

- *BUFMODE Tied Low (0)*—If BUFMODE is 0, external registered bus transceivers are used between the processor and the NxVL, the XBOE* and XBCKE* signals on the NxVL are tied high, and the NxVL is interfaced to the NexBus, NxAD<63:0>. Figure 32 shows the arbitration, address, and data phases when BUFMODE is 0. Figure 14 shows a block diagram of this bus-connection alternative.
- *BUFMODE Tied High (1)*—If BUFMODE is 1, the processor (including its XBOE* and XBCKE* signals) is connected directly to the NxVL, the NxVL NexBus ports emulates the bus transceivers, and the timing of the address and data signals on the NexBus is somewhat different than when BUFMODE is 0. Figure 33 shows the arbitration, address, and data phases when BUFMODE is 1. Figure 14 shows a block diagram of this bus-connection alternative.

As shown in Figure 33, when BUFMODE is 1, addresses or data going *from the processor to the NxVL* (addresses for all processor-initiated cycles and data for processor-initiated write cycles) appear at least one clock before the corresponding information when BUFMODE = 0. XBCKE* is asserted to latch the addresses and data into the NxVL, although the latched addresses and data are not used until the correct phase in the cycle, which is different than for BUFMODE = 0. For addresses or data going *from the NxVL* to the *processor* (addresses for NxVL-initiated snoop cycles and data for processor-initiated read cycles), the addresses and data are delayed by one clock when compared to BUFMODE = 0, and XBOE* is asserted to enable the NxAD buffers in the NxVL. To provide adequate setup time for the processor which latches data at the falling edge of the clock, the data is switched by the NxVL at the falling edge of the clock.

When BUFMODE is 1, the processor can assert the address/status signals in the same clock as it asserts its bus request. For memory operations, the address of a qword is driven on NxAD<31:3> and the status is driven on the higher bits. For I/O operations, the address of a dword is driven on NxAD<15:2>, with the status on the higher bits. In that clock, the processor asserts XBCKE* so that the NxVL can latch the address/status into its emulated transceivers. If another address is latched into the NxVL before the cycle starts, the old address is

overwritten. Such overwriting may occur during speculative reads in which the read is aborted at the last moment. When GNT* is asserted by the NxVL, the processor asserts ALE* for one NexBus clock, causing the NxVL to assert GALE a few nanoseconds after ALE* goes active. The bus cycle starts when GALE is asserted and the address and status are latched from the NxVL's emulated transceivers into the NxVL's address-decoding logic. That completes the address phase of the cycle.

When BUFMODE is 0, the processor can enable its NxAD buffers onto the bus only after it has control of the bus (samples GNT* asserted). In this mode, the processor asserts its request and waits for a grant. After it samples GNT* asserted, it asserts the address/status for one NexBus clock onto the NxAD lines as described above. Simultaneously, it asserts ALE* for one NexBus clock, causing the NxVL to assert GALE a few nanoseconds after ALE* goes active. The NxVL then latches the address from the bus into its internal latches, bypassing its internally emulated transceivers, thus completing the address phase.

The clock following the address phase is spent decoding the destination for the cycle. Any cycle not destined for main memory falls through onto the VL-bus, and if no response if received, onto the ISA-bus. In the next clock, (the start of the data phase), the NxVL asserts GXACK and drives GBLKNBL valid. If the NxVL is ready to provide data for a read cycle, or ready to accept data for a write cycle, it will not assert GXHLD. Otherwise, it will assert GXHLD to add wait states until it is ready.

During a write cycle when BUFMODE is 1, the data to be written is driven by the processor in the first clock of the data phase (in which GXACK is first asserted). XBCKE* is also asserted along with the data for one clock. This latches the data into the NxVL's internally emulated transceivers. During a read cycle when BUFMODE is 1, the NxVL outputs valid data onto NxAD<63:0> in the second clock after GXACK is sampled asserted and GXHLD is sampled negated. By comparison, during a read cycle when BUFMODE is 0, valid data is required on NxAD<63:0> in the clock after GXACK is sampled asserted and GXHLD is sampled negated.

The group signal, GALE, is the logical NOR of all ALE* signals in the system, but in NxVL-based systems there is only one ALE* signal (the one from the Nx586 processor). For multiprocessor systems, the group signals must be generated by an external NexBus arbiter. In Figure 32, the "C_P" symbol in the "Source" column indicates that the source of GALE is the NxVL but that its underlying activator is the Nx586 processor. In most subsequent timing diagrams that show group signals such as GALE, the corresponding activating signal is not shown but is indicated by the subscript on the Source symbol.

☞ *In all subsequent discussions in this section, we assume that BUFMODE is tied high (1).*

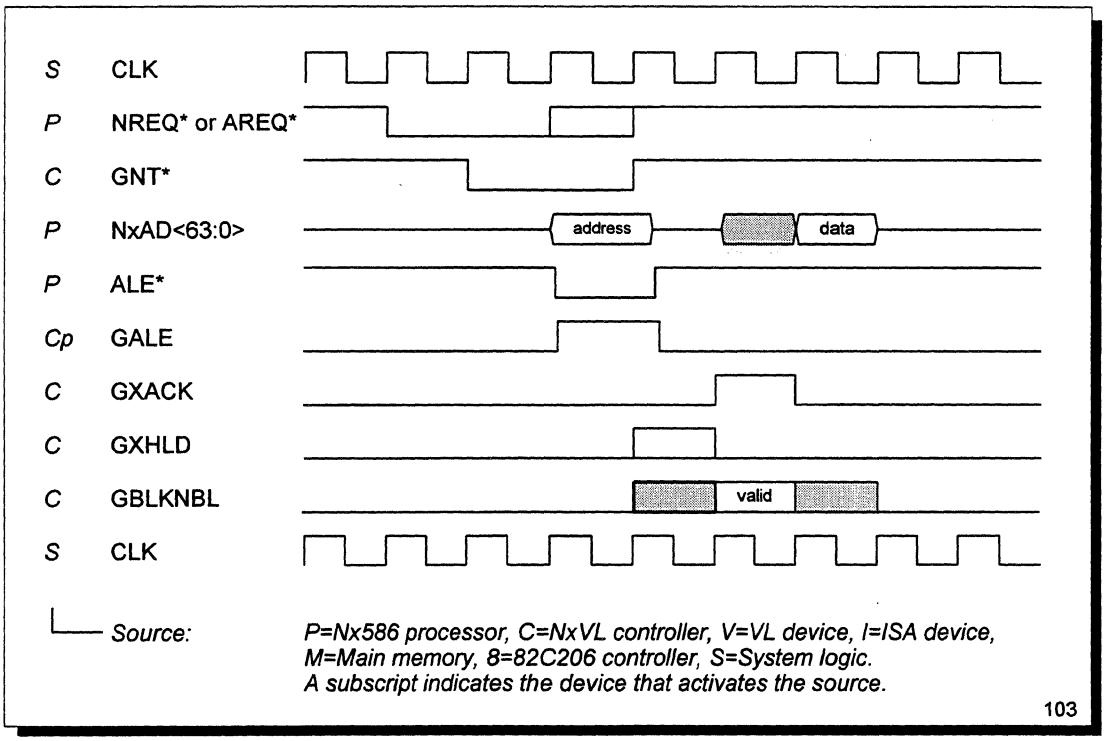


Figure 32 Bus Arbitration, Address, and Data Phase (BUFMODE = 0)

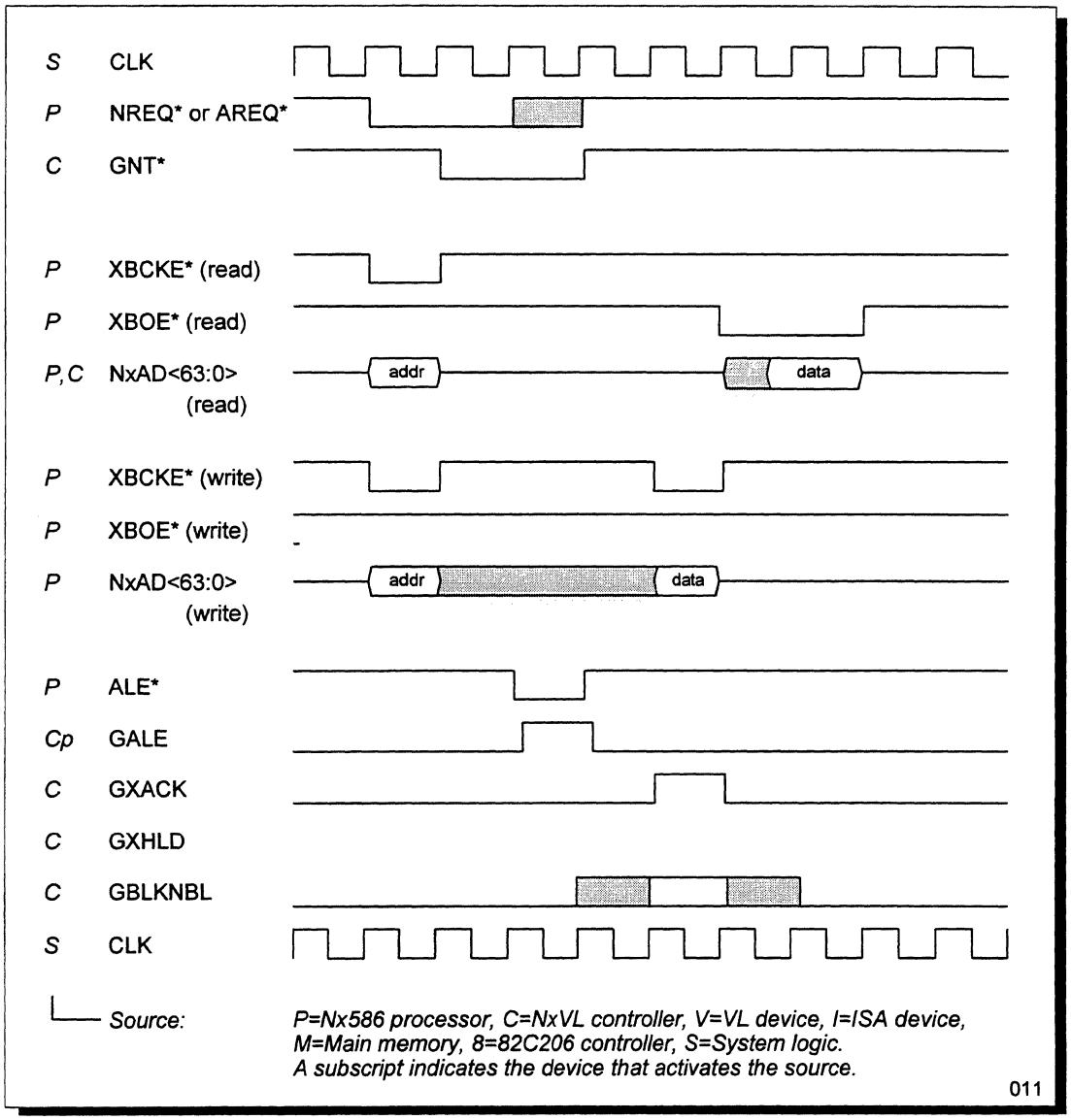


Figure 33 Bus Arbitration, Address, and Data Phase (BUFMODE = 1)

Processor Write to Main Memory

Processor writes to main memory are buffered in the NxVL's eight-qword write queue. The processor's cycle terminates three clocks after the address/status is latched. (Hereafter, the word "address" is used to mean both address and status.) The NxVL subsequently transfers the data to main memory. An empty write queue can accommodate up to two 32-byte blocks (bursts). Wait states are introduced into the cycle only if the write queue is full, in which case the queue is written to memory before the processor's write can complete.

Figure 34 shows a single-qword memory write. The processor begins by driving the address on the NxAD<63:0> bus and asserting XBCKE* for one clock. At the end of this clock, the NxVL latches the address into its internal transceivers. In the next clock, the processor asserts ALE* (not shown). The NxVL asserts GALE a few nanoseconds later in response to ALE*. The next clock is for decoding, during which time the latched address is compared with the NxVL's memory-map registers. In the next clock, the processor outputs the data to be written and asserts XBCKE* for one clock. At the end of this clock, the data is latched into the NxVL's transceivers. The NxVL simultaneously acknowledges the processor without a wait by asserting GXACK with GXHLD negated.

In the case of a burst write, the transactions are identical to the single write described above except that, following the first qword transfer, three more qwords are written at the rate of one qword per clock. In such a cycle, XBCKE* and GXACK are each active for three additional clocks.

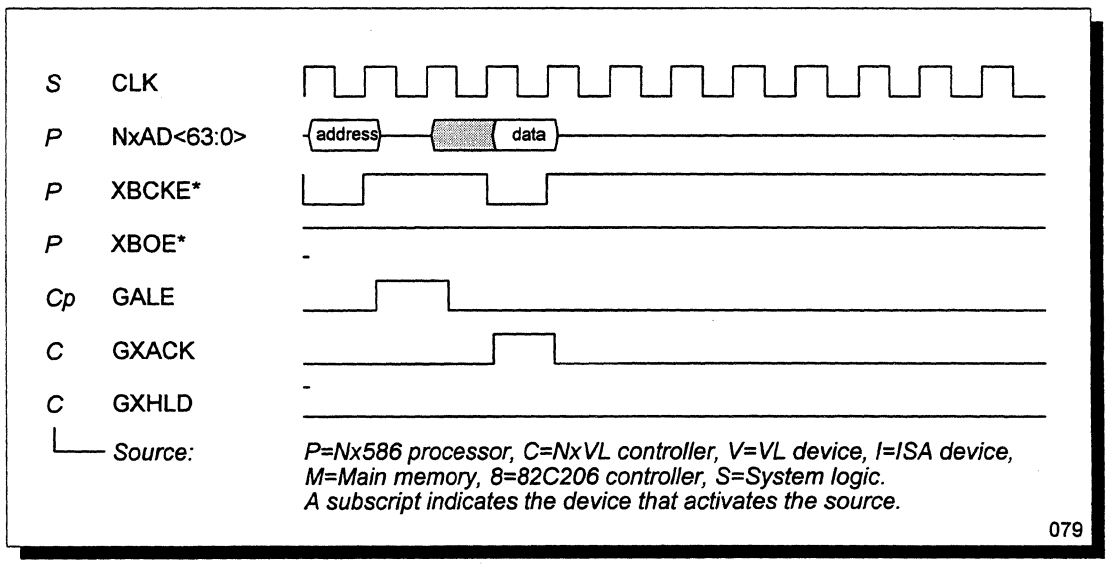


Figure 34 Processor 64-Bit Write to Main Memory (Queue Hit)

Processor Read from Main Memory

Figure 35 shows a single-qword memory read that hits in the NexBus prefetch queue. The timing of GALE, GXACK, and GXHLD are identical to the memory write shown in Figure 34. The address phase for a read cycle is the same as for a write cycle. The processor drives the address on the NxAD<63:0> bus and asserts XBCKE* for one clock. At the end of this clock, the NxVL latches the address into its internally emulated transceivers. In the next clock, the processor asserts ALE* and the NxVL responds by asserting GALE. Two clocks after GALE, the NxVL responds by asserting GXACK. Two clocks later, valid data is driven on the NxAD bus. The multiplexed NxAD<63:0> bus is shown with two sources because this is a read cycle: the processor provides the address and the NxVL provides the data.

In the case of a burst read, the transactions are identical to the single read described above except that the GXACK and XBOE* signals are extended for three more clocks and, following the first qword transfer, three more qwords of data are output onto the NxAD bus at the rate of one qword per clock. (A block-transfer request is indicated by a status bit during the address phase.)

The NxVL maintains a 64-byte NexBus prefetch queue, providing the processor with two quickly accessible 32-byte cache blocks of instructions or data. If the read hits in the second of the two 32-byte prefetched blocks in the queue, or if it misses the prefetch queue, memory-access latency is added to the read time.

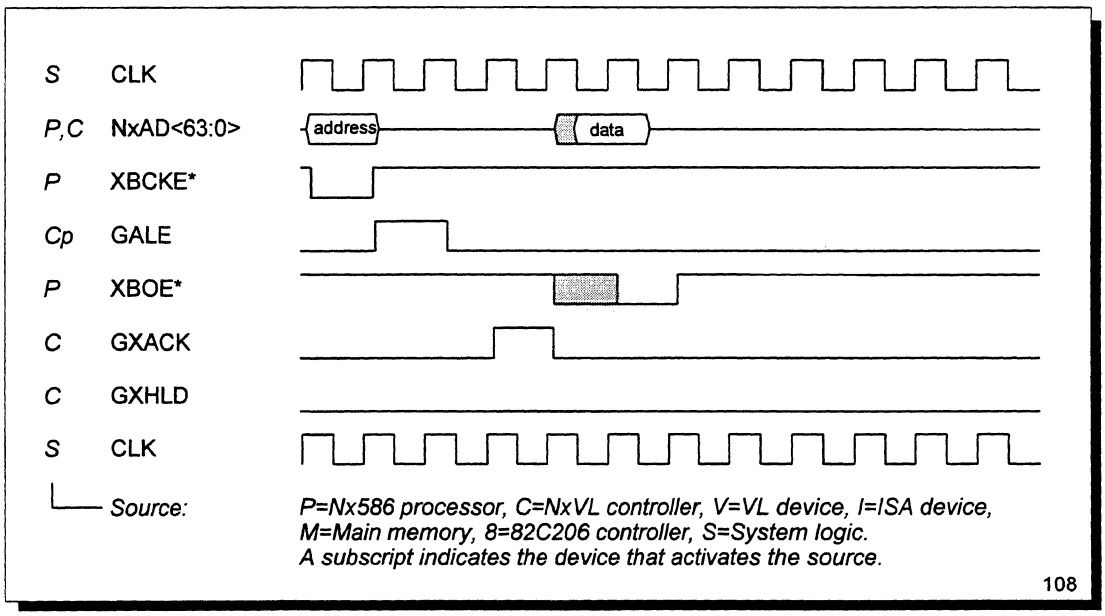


Figure 35 Processor 64-Bit Read from Main Memory (Queue Hit)

Figure 36 shows a single-qword read that misses the NexBus prefetch queue and misses the main-memory DRAM page. The NxVL asserts both GXACK and GXHLD two clocks after the address is latched. Two clocks later, RAS<n>* is negated for the precharge. When bit 12 is set to 1 in the CFG0 register, the precharge takes two clocks. When bit 12 is cleared to 0 in the CFG0 register, the precharge takes three clocks. During the precharge, the NxVL begins driving the row address on the memory address bus, MA<10:0>.

As the column addresses are driven on MA<10:0>, the MD<63:0> data bus returns two 32-byte cache blocks (lines) in a sequence of qwords. The first four qwords contain the operand addressed by the processor. That block plus the next (labeled "Prefetch") fills the NexBus prefetch queue and leaves the queue half-read. The addressed qword appears on NxAD<63:0> as early as twelve clocks after GALE is sampled active by the processor.

If the read cycle hit in the first of the two prefetch-queue blocks (lines), the cycle would end in three clocks as shown above in Figure 35 and the NxVL would not do a prefetch. If the read had hit in the second prefetch-queue block, the processor's cycle would end in three clocks, and thereafter the NxVL would prefetch two new 32-byte blocks to fill the queue with entirely new data.

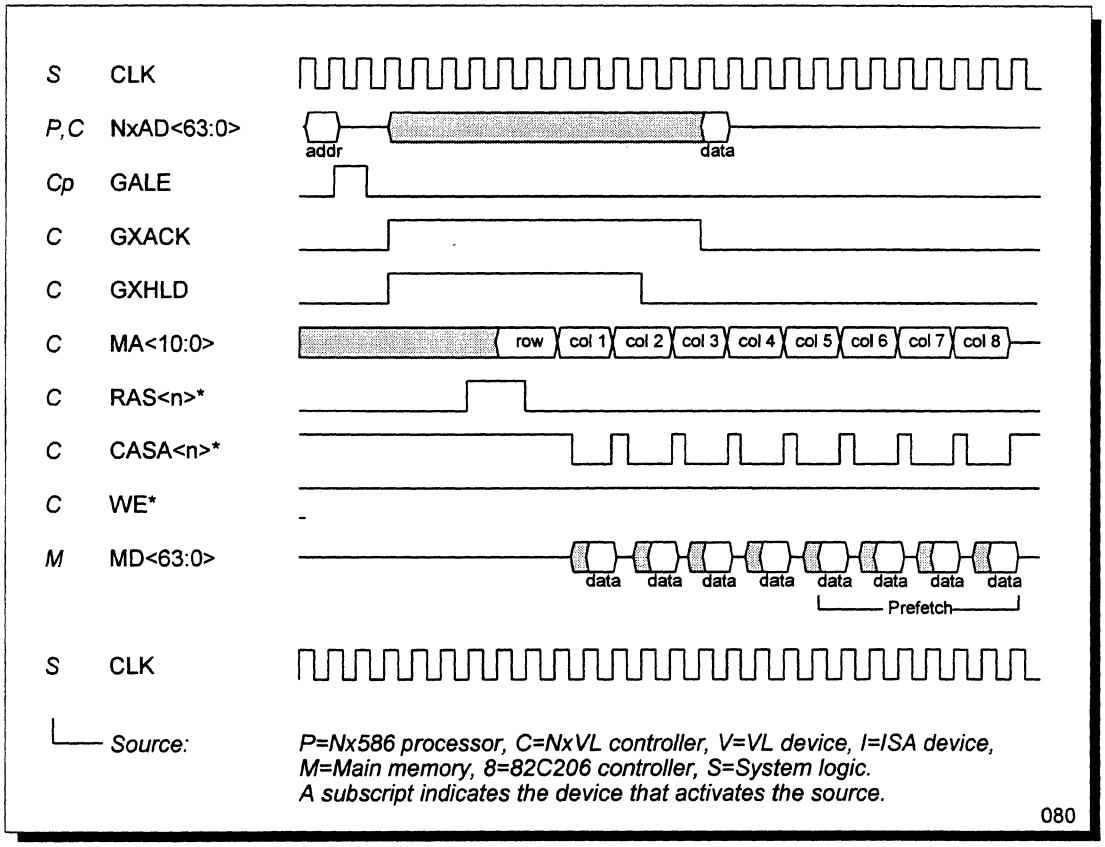


Figure 36 Processor 64-Bit Read from Main Memory (DRAM-Page Miss)

Memory reads that fill the processor's caches are by far the most common types of reads. Figure 37 shows a series of four-qword block (burst) reads. In this example, the first block is a prefetch-queue miss. It assumes that memory is initially inactive and does not require a precharge. Thereafter, subsequent blocks that access sequential locations are all prefetch-queue hits because the NxVL fills the prefetch queue as an integral part of the read cycle.

The cycle in Figure 37 begins in the same manner as Figure 36, except that the precharge time is eliminated. GXACK goes active one clock after GALE and remains active until one clock before the block's last qword appears on NxAD<63:0>. For the first block fetched from memory, the NxVL asserts GXHLD to the processor to indicate that one wait state is added between each qword being returned. Thereafter, prefetching continues to fill the prefetch queue, and the second four-qword block addressed by the processor is returned without wait states. The prefetch queue continues to be filled as long as the processor continues addressing sequential locations.

The example assumes that fast DRAM accesses are enabled by setting bits 12 and 22 of the configuration register CFG0 to 1.

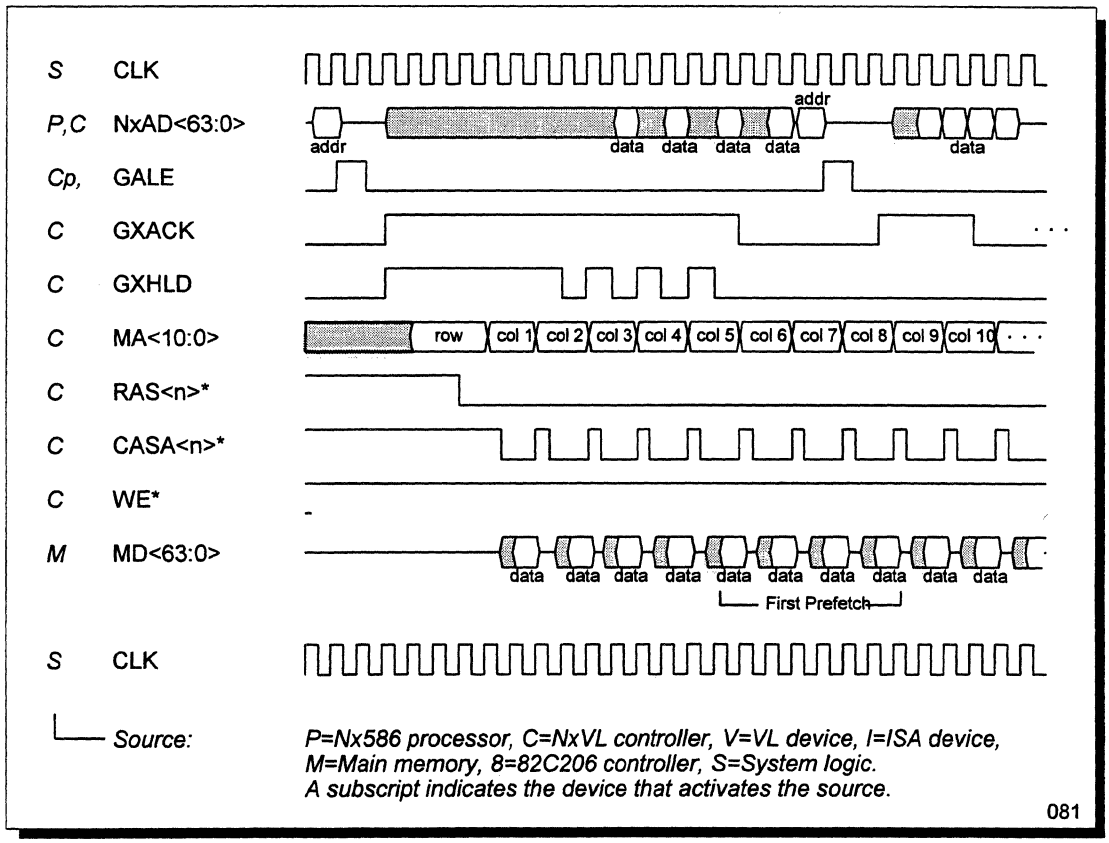


Figure 37 Processor 32-Byte Block Read from Main Memory

Processor Write to VL Slave

Figure 38 shows a processor qword write to a 16-bit VL-bus memory slave. The processor begins just as in a write to main memory. In the clock after GALE, the NxVL compares the address with the memory-map and finds that it is not in main memory. The NxVL asserts GXACK with GXHLD negated to the processor while it drives the address, cycle definition, and byte-enable bits onto the VL-bus (LBADR<31:2>, LBM/IO*, LBW/R*, and LBE<3:0>*). Shortly thereafter, the VL-bus slave memory responds with its device number (LDEV<n>*). In the next clock, the NxVL buffers the processor's data and the processor is finished with the cycle.

The NxVL then asserts a sequence of four LADS* strobe on the VL-bus (one for each 16 bits of the processor's qword) as it drives the byte-enables, addresses, and data. The VL-bus slave asserts LBS16* to indicate a 16-bit device at the same time that it asserts LRDY*. Initially, all byte-enable bits are asserted under the assumption that the slave is a 32-bit device. The assertion of LBS16* causes the NxVL to change the byte-enable bits (without changing the address or data) for the next 16-bit transfer. After the first 32-bits have been written, the NxVL changes the address on LBADR<31:2> to write the next 32 bits with the same sequence.

The NxVL does not support bursts on bus-crossing cycles. It negates GBLKNBL to the processor, thus preventing bursts. The NxVL asserts BLAST* throughout any bus-crossing cycle to indicate to the VL slave that the cycle is not a burst.

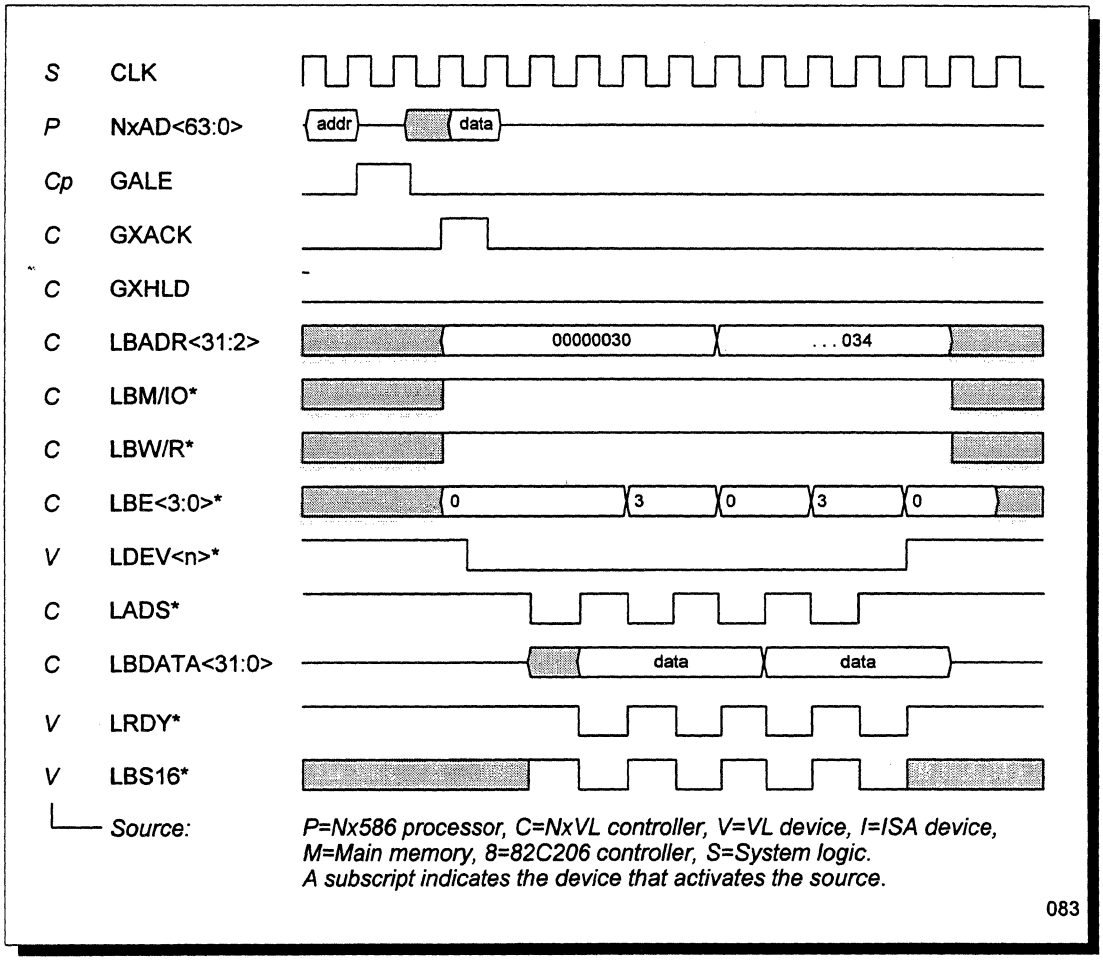


Figure 38 Processor 64-Bit Write to VL-Bus 16-Bit Memory Slave

Figure 39 shows the response timing of LDEV<n>* for two different system configurations. In accordance with the VL-bus specification, each VL-bus device looks at the address when LADS* is asserted. The device to which the address maps must return its LDEV<n>* within one or two clocks. If bit 25 of the CFG0 register is set to 1 (Fast VL-Bus Transfer Enable) and the NexBus clock is 33MHz or less, the device is expected to respond in one clock; otherwise, in two clocks as shown for a 40MHz clock.

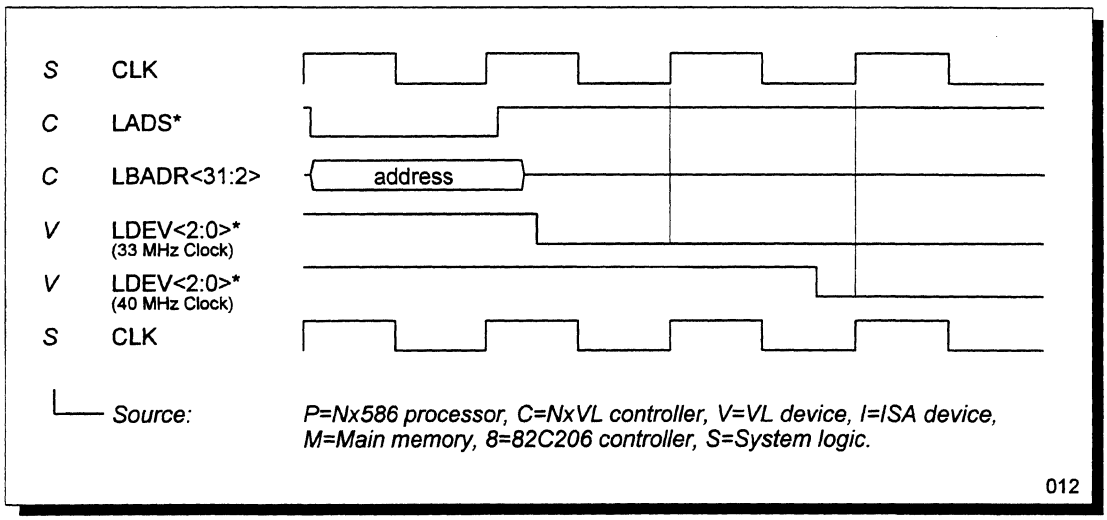


Figure 39 Response by a VL-Bus Device to Address Strobe

Processor Read from VL Slave

Figure 40 shows a processor qword read from a VL-bus 32-bit memory slave. The cycle begins like the write cycle described in Figure 38, but in the data phase, wait states are inserted by the simultaneous assertion of GXACK and GXHLD. The NxVL translates the address onto LBADR<31:2>, asserts all byte-enable bits on LBE<3:0>*, and asserts LADS*. At the same time, the VL-bus slave responds with its LDEV<n>*.

For reads, the VL-bus specification requires one wait state between LADS* and LRDY* from the VL slave. Thus, one clock after LADS* is negated, the slave drives the first 32 bits of data valid on LBDATA<31:0>, asserts LRDY*, and the NxVL responds with RDYRTN*. Two iterations of this sequence supply the processor with the required qword. During both 32-bit iterations, the NxVL asserts all byte-enable bits. One clock after the last dword is returned by the slave, the NxVL negates GXHLD, followed in one clock by the negation of GXACK and the data on NxAD<63:0> in the next clock.

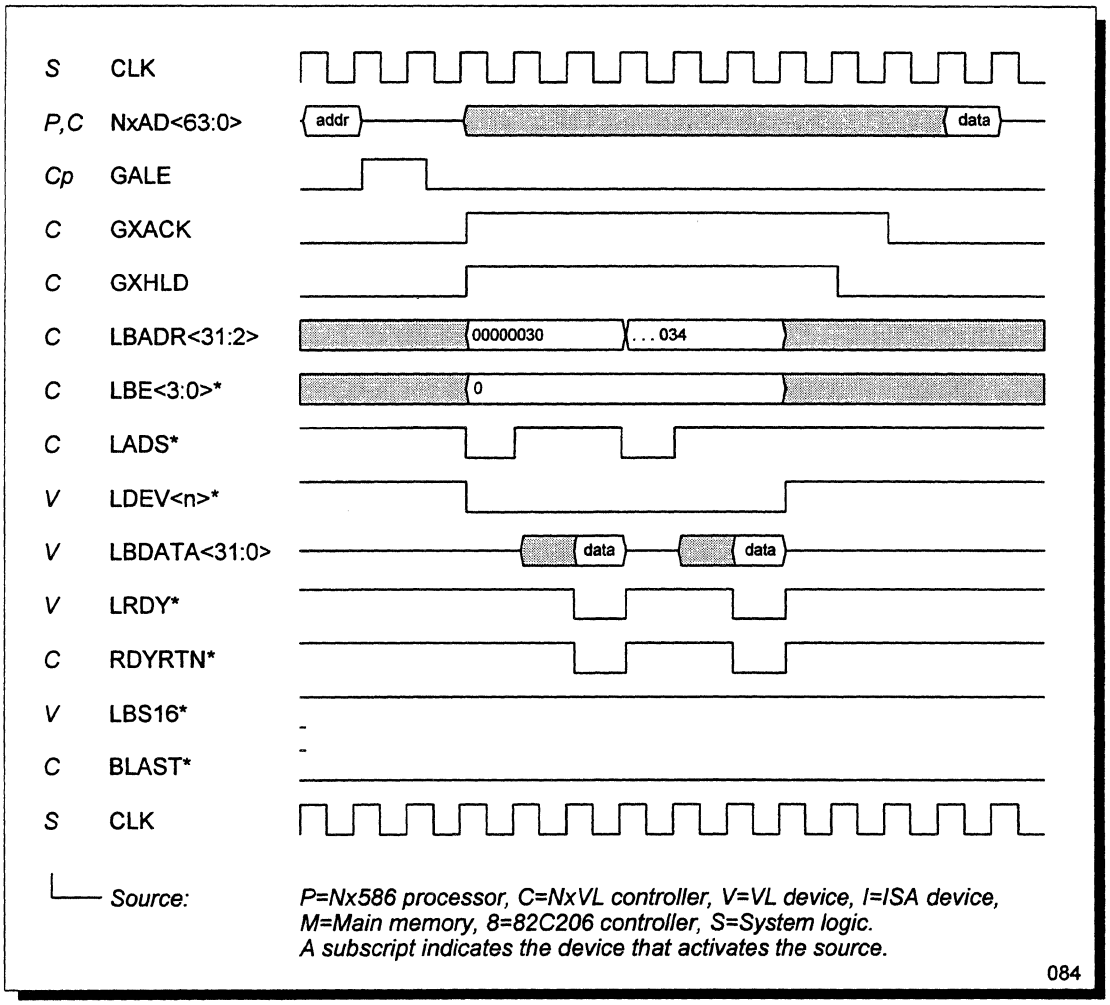


Figure 40 Processor 64-Bit Read from VL-Bus 32-Bit Memory Slave

Processor Write to ISA Slave

If a write location is neither to main memory nor to the VL-bus (because no device responds with its LDEV<n>* in the VL-bus time limit), the write goes to the ISA-bus. The NxVL AT-bus state machine generates the ISA control signals to perform the requested cycle on the ISA-bus, and it controls all of the byte, word, and dword assembly and disassembly for the cycle.

Figure 41 shows a processor 16-bit write to an ISA 16-bit memory slave (with most of the DMA-controller interactions omitted; these are explained later).

The processor's cycle ends after it places the data on the NexBus. As the data is driven, the NxVL translates the address onto LA<23:17> and SA<19:0>, asserts SBHE* (to enable the high byte for a 16-bit transfer), and negates AEN (to enable address decoding by the slave). The slave responds by asserting MEMCS16* to indicate that it is 16 bits wide. The NxVL then asserts BALE, drives the data on SD<15:0>, and asserts MEMW*. After MEMW* is asserted, the NxVL briefly negates IOCHRDY (for reasons not related to this cycle) prior to re-asserting and sampling the signal. The cycle terminates a specific time after the NxVL samples IOCHRDY active.

The NxVL drives LA<23:17> through bus transceivers with LBADR<23:17> and it drives SA<19:0> with a combination of LBADR<19:2> and SA<1:0>. The three bits of overlap between LBADR<23:17> and LBADR<19:2> are driven identically. For details on the mapping of data between the VL-bus and the ISA-bus, see Figures 20 and 21 in the *Hardware Architecture* chapter.

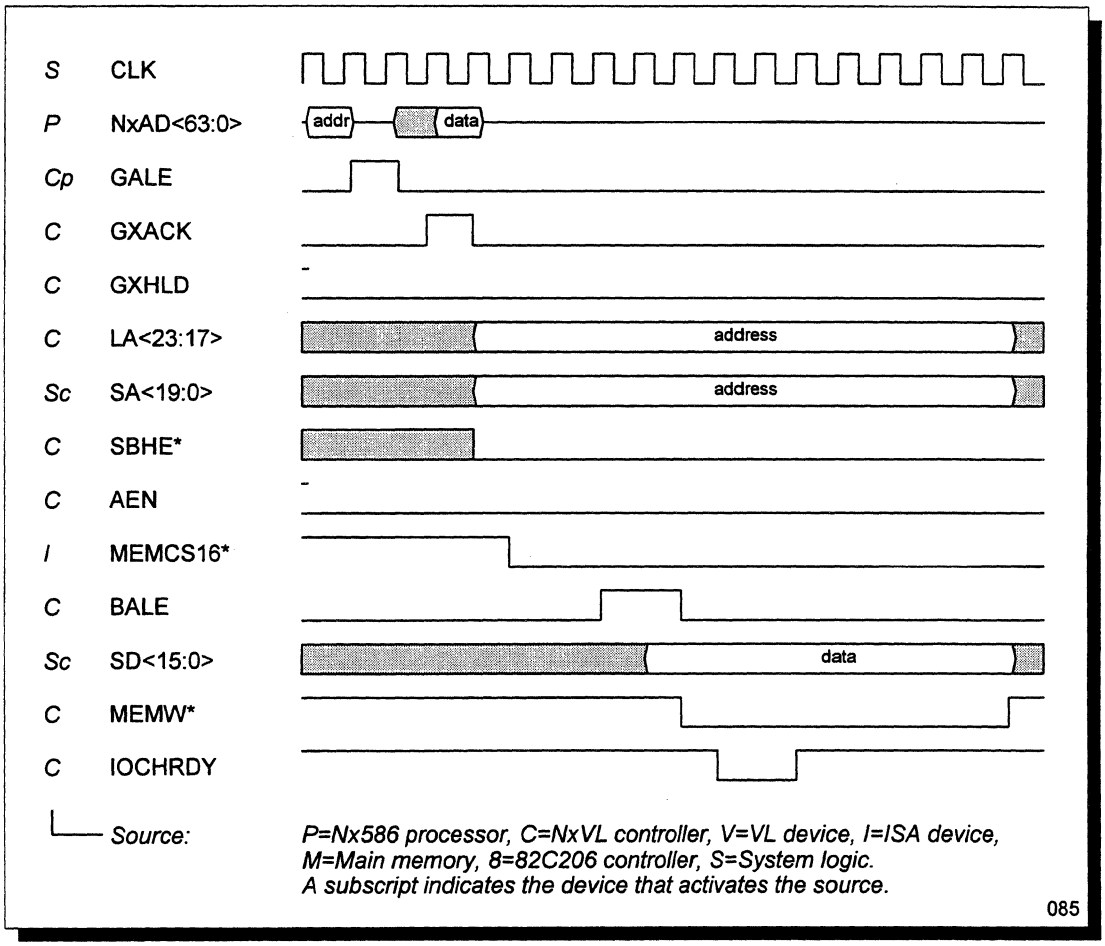


Figure 41 Processor 16-Bit Write to ISA-Bus 16-Bit Memory

A 32-bit transfer from the processor to an ISA-bus slave involves one cycle on the VL-bus but either two or four cycles on the ISA-bus, depending on whether the ISA slave is a 16-bit or 8-bit device. In a 64-bit transfer, the first 32 bits are done first. Then, when the first 32 bits are transferred to the slave, the next 32 bits are sent from the processor. While the intermediate bus-crossing transfers are occurring between the VL-bus and the ISA-bus slave, the processor can continue with other work from its cache, unless it accesses a location outside its cache.

Processor Read from ISA Slave

Figure 42 shows the processor reading 64 bits from a 16-bit ISA memory slave. After the processor drives the address and status onto the NexBus, the NxVL asserts both GXACK and GXHLD for the entire time required to read and accumulate all four 16-bit transfers in a latch. Each 16-bit transfer on SD<15:0> is initiated with the assertion of BALE, and MEMR*. When all four transfers are finished, the NxVL negates GXHLD and (one clock later) GXACK, and drives the 64 bits of data onto the NexBus. As in the previous example, the NxVL briefly negates IOCHRDY (for reasons not related to this cycle) prior to re-asserting and sampling the signal during each 16-bit transfer.

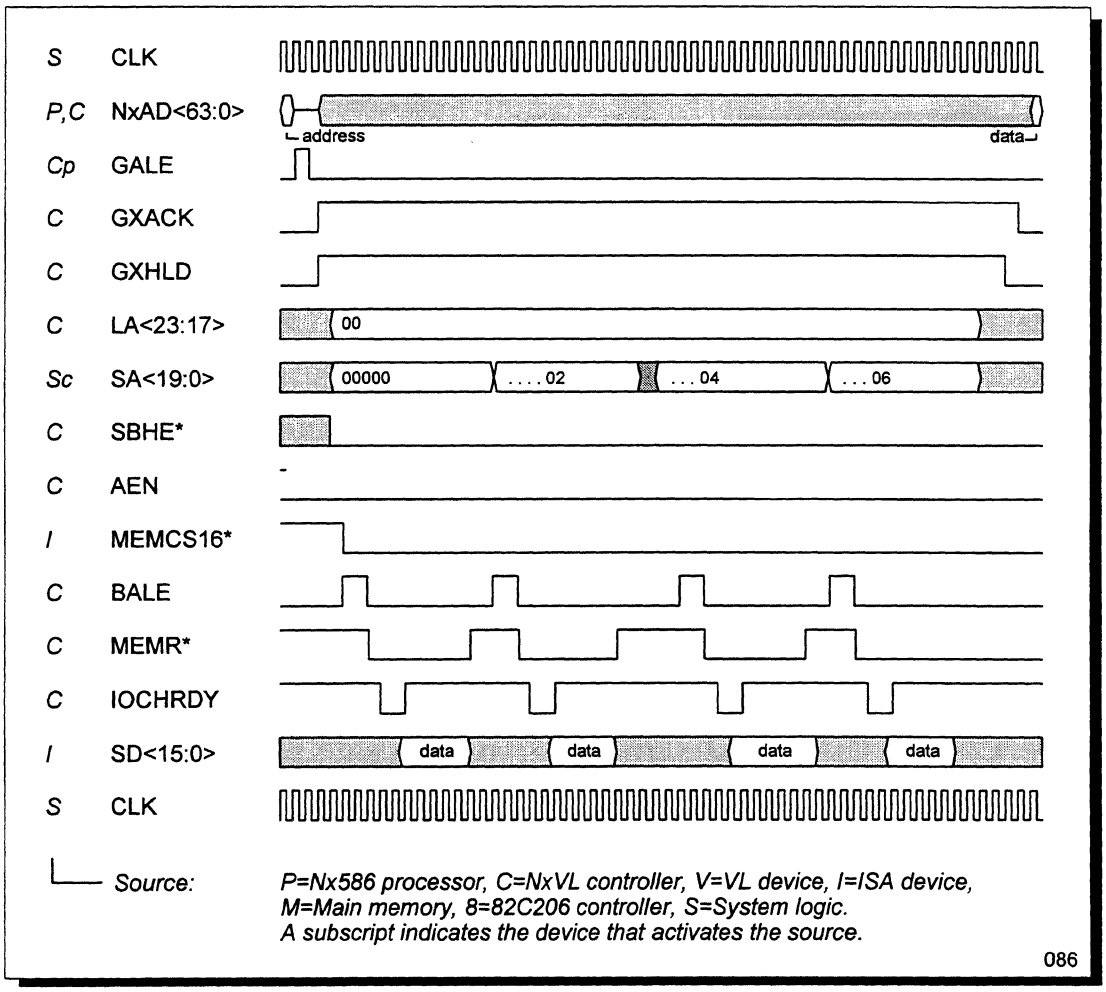


Figure 42 Processor 64-Bit Read from ISA-Bus 16-Bit Memory

Snooping and Processor Intervention

When a VL-bus master or an ISA-bus master initiates a read or write cycle to main memory, the NxVL guarantees memory coherency with the Nx586 caches by generating snoop cycles to the Nx586 processor. However, the NxVL may not need to generate a snoop cycle for every access from a VL or ISA master. The NxVL keeps track of the four most-recently snooped addresses. If the cycle address matches one of these four snooped addresses, a snoop cycle on the processor bus is avoided.

If a snooped address is cached by the Nx586 processor in either the *shared* or *exclusive* state, the cache entry will be changed to the *invalid* state. If the snooped address is cached in the *modified* state, the Nx586 processor will assert DCL*, wait for the NxVL to grant it access to the buses, write the modified data back to main memory, and change the cache entry to the *invalid* state. If the initiating cycle is a read that hits a *modified* location, the Nx586 processor will write the entire cache entry back to memory; if the initiating cycle is a write that hits a *modified* location, the Nx586 processor will write back only that portion of its cache entry that is not being written by the initiating master. Since VL-bus masters can cache data using a write-through protocol, *any* read by a VL-bus master is treated by the Nx586 processor as a MESI-protocol *exclusive* read (read-to-own), and during the snoop cycle the Nx586 processor will change its cached copy to the *invalid* state (or write-back its copy if it is in the *modified* state).

Figure 43 shows a snoop hit during a VL-master read from main memory. In the first clock, the NxVL is asserting GNT* to the Nx586 processor but there is no activity on the NexBus. This granting is the default condition when no other master is requesting the buses. The next several clocks show the VL-master's bus arbitration, address strobe, and the driving of the read address and cycle definition. This cycle, which continues during the snoop cycle, is explained in the next section, entitle *VL-Bus Master Operations*.

After the VL master drives the read address, the NxVL drives the same address on the NexBus and asserts GALE without stimulation by an ALE* from the processor. This is the snoop address. If the processor's tag for that location is available, the processor asserts XACK* one clock later, the NxVL responds with GXACK, and the processor's DCL* assertion becomes valid. If the processor's snoop tag is not immediately available, the processor asserts GXHLD for one clock before asserting GXACK. DCL* is only valid when GXACK is asserted while GXHLD is negated.

Three clocks after the first GXACK, the processor drives the snoop-hit address (the same address driven earlier by the NxVL) along with ALE*, which causes the NxVL to assert GALE and GNT*. In the next clock, the NxVL asserts GXACK for four clocks (without stimulation by an XACK* from the processor), and the write-back of four qwords (one cache line) to the NxVL write queue begins one clock after GXACK goes active.

While the processor is writing back to the write queue, the data is simultaneously written to the NxVL's VL-bus read queue and onto the VL-bus. A snoop-hit cross-over path is provided for this purpose, as shown in Figure 12 in the *Hardware Architecture* chapter. One clock after the last qword of data is written back, the VL-bus master asserts LRDY* to terminate the cycle. The VL master then negates its LREQ<n>* and the NxVL negates LGNT<n>*. Even if the VL master negates LREQ<n>* early, the master's cycle will not be terminated by the NxVL until the snoop has actually begun on the NexBus.

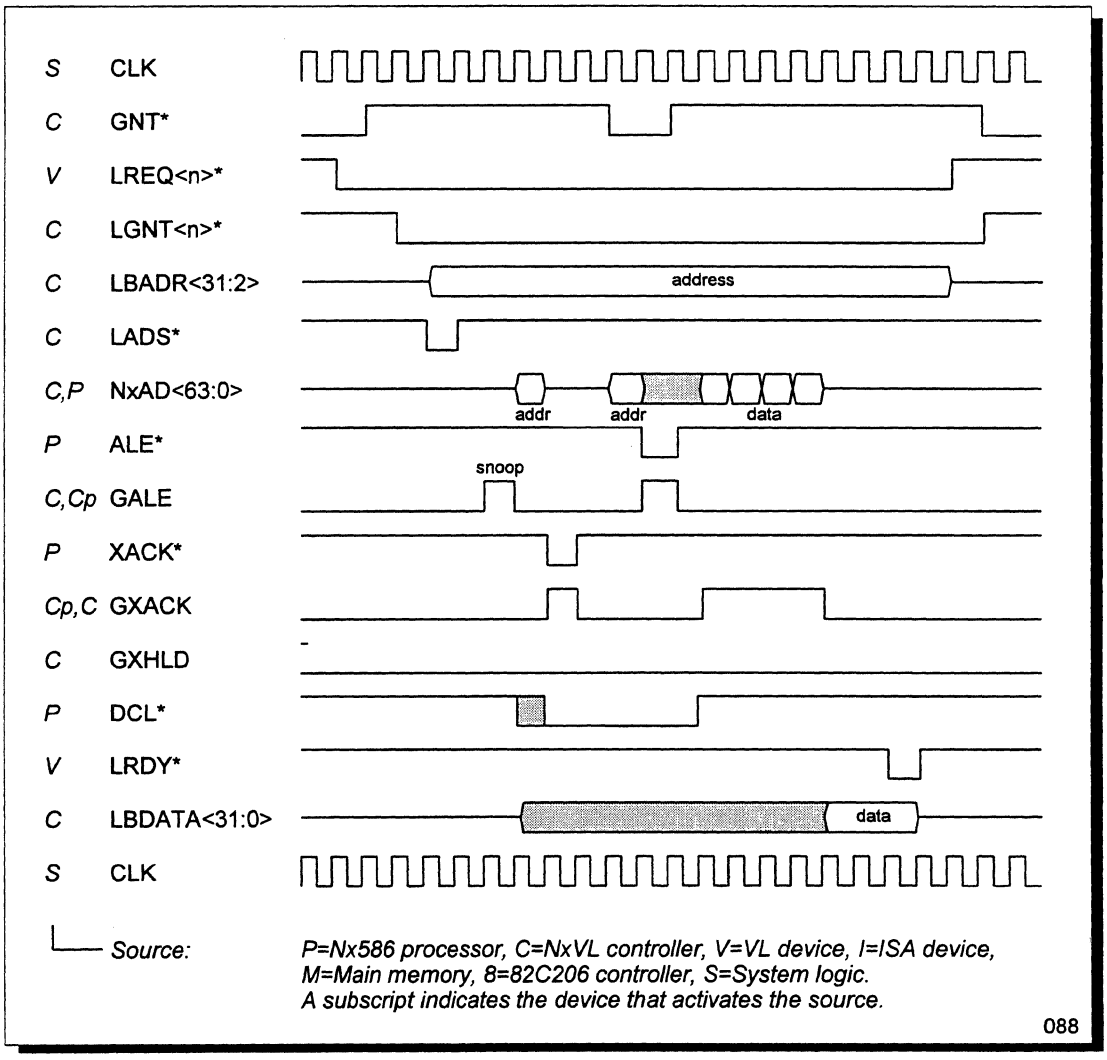


Figure 43 Snoop Hit During VL-Master Read from Main Memory

VL-Bus Master Operations

When a VL master arbitrates for the bus and initiates a cycle, the NxVL compares the address of the cycle with the main-memory address map initiated by the BIOS. If the cycle is not addressed to main memory, it is translated onto the ISA-bus. The NxVL, by default, supports 16-byte burst operations on the VL-bus and between VL masters and main memory. If a VL master cannot support bursts, it must assert BLAST* throughout its cycles to main memory.

During all VL-master cycles to main memory, whether read or write, the NxVL snoops the Nx586 processor. VL masters can cache data, but they must use a write-through policy since the NxVL cannot rely on VL masters maintaining a MESI cache-coherency protocol that is compatible with the Nx586 processor. As a precaution against VL masters that cache data, *any* read by a VL master is treated by the Nx586 processor as a MESI-protocol *exclusive* read, as described above in the section entitled *Snooping and Processor Intervention.*

All VL-bus operations performed by the NxVL complies with the *VL-Bus™ Proposal*, published in 1992 by the Video Electronics Standards Association (VESA).

Bus Arbitration

Figure 44 shows the arbitration process used by a VL master to obtain control of all system buses. In the first clock, the NxVL asserts GNT*, which gives the buses to the Nx586 processor, even though the processor is not requesting the buses (NREQ* is negated). This is the default condition when no master is requesting the buses.

To obtain bus control, a VL master asserts its LREQ<n>* (one of the LREQ<2:0>* signals). The NxVL responds by asserting the associated LGNT<n>* one clock after it negates GNT* to the Nx586 processor. In the next clock, the VL master drives its address (not shown) and LADS*, and the NxVL initiates a snoop cycle on the NexBus. At the end of the VL-master's cycle, the slave asserts LRDY*, or the NxVL asserts LRDY* or BRDY*, whereupon the master negates LREQ<n>*. One clock later, the NxVL negates LGNT<n>*, and another clock later it asserts GNT* if either (1) the Nx586 processor is explicitly requesting access by asserting NREQ*, AREQ* or LOCK*, or (2) no other master is requesting access.

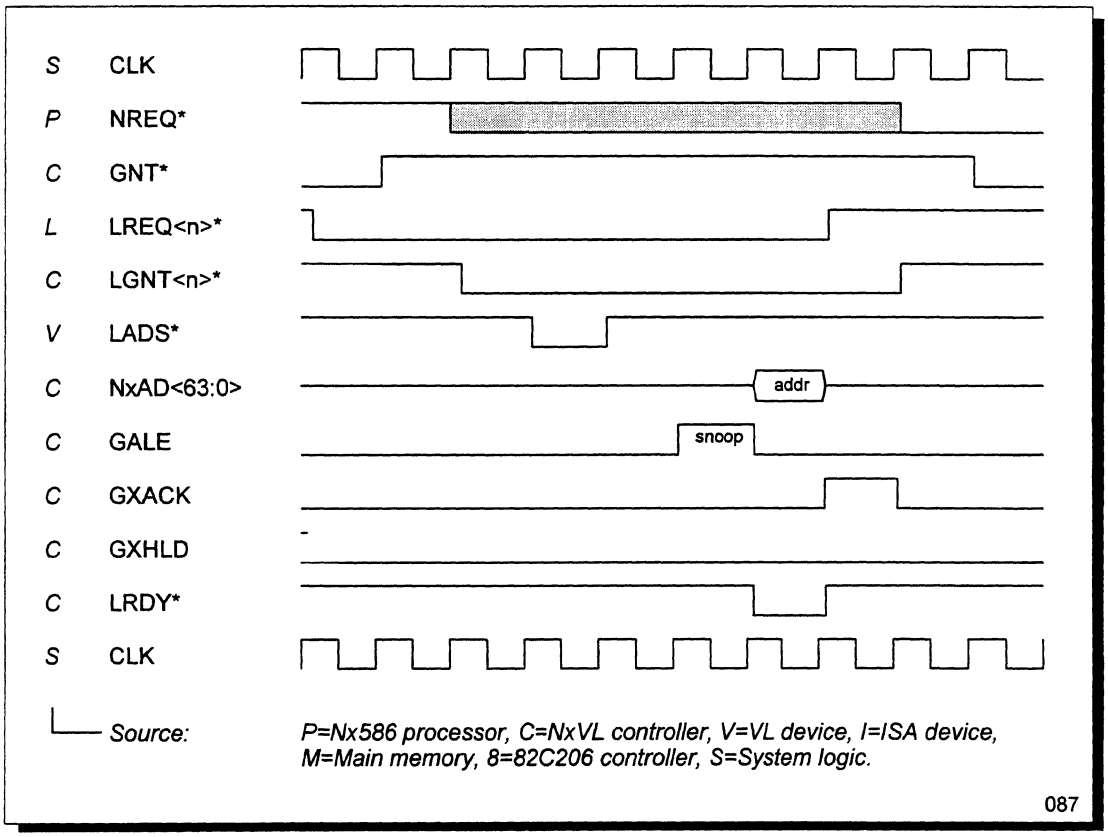


Figure 44 Bus Arbitration by VL-Bus Master

VL Write to Main Memory

Figure 45 shows a VL-master 32-bit write to main memory. The cycle begins with the VL-master's bus arbitration (not shown), as described in the section immediately above. The VL master then drives the address onto LBADR<31:2>, defines the type of cycle with LBM/IO* and LBW/R*, asserts LADS*, and one clock later drives the data on LBDATA<31:0>.

One clock after LADS* goes inactive, the NxVL snoops the Nx586 processor. In Figure 45, the snoop is a miss. The VL-master's write cycle continues during the snoop. BLAST* becomes valid one clock after LADS* goes inactive. The NxVL then asserts LRDY* two clocks after LADS* to terminate the cycle.

While the VL-bus specification allows LRDY* to be asserted one clock after LADS* goes inactive, the NxVL adds one wait state so that BLAST* can be properly sampled to determine whether LRDY* or BRDY* indicates the

termination of the cycle. In burst operations, BRDY* is asserted one clock earlier, as shown in Figure 46, below. The NxVL is designed with the assumption that most VL-bus devices will support bursts, so burst cycles are given a performance advantage over single cycles.

For details of how data is subsequently transferred by the NxVL to main memory, see the section below entitled *Main-Memory Operations*.

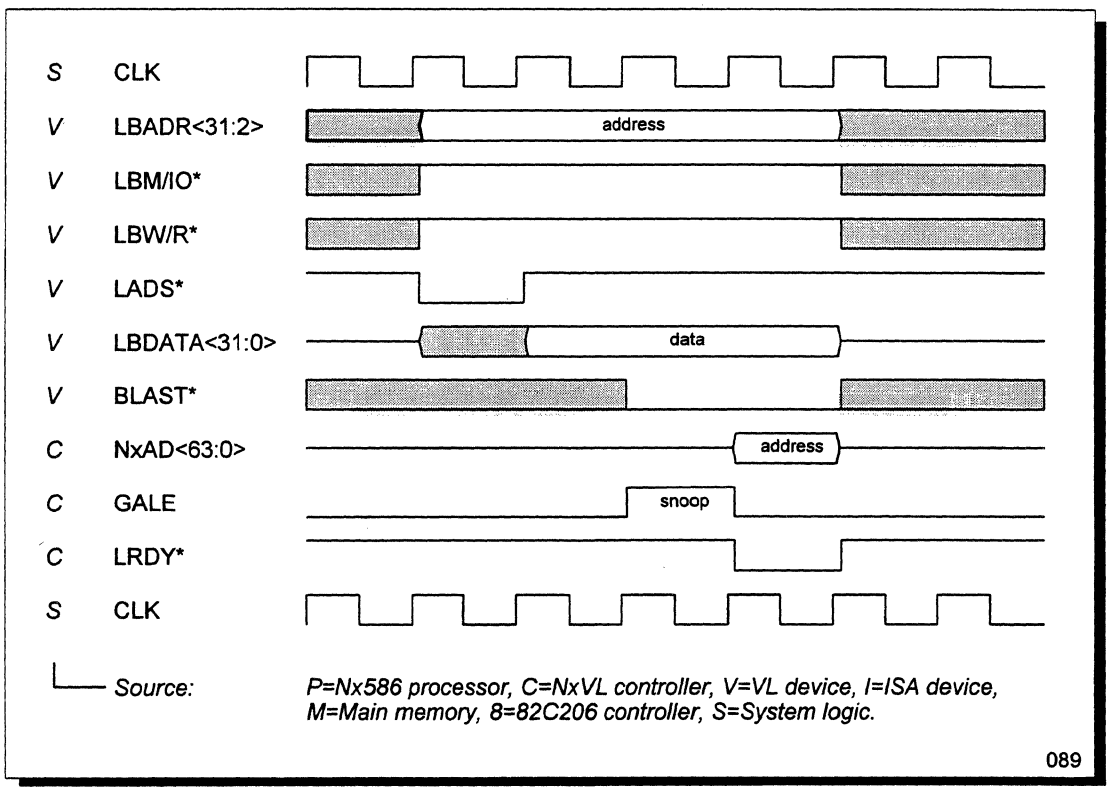


Figure 45 VL-Master 32-Bit Write to Main Memory

VL Read from Main Memory

Figure 46 shows a VL-master 16-byte burst read from main memory that hits in the NxVL's read queue. The VL master drives the address, cycle-definition signals (not shown), and all byte-enable bits (not shown). At the same time, it asserts LADS*. The first address is latched one clock after LADS* goes inactive. BLAST* becomes valid at this same time and is held negated until the last of the four 32-bit transfers. The transfers begin when the NxVL asserts BRDY*, which is held asserted for all four transfers, during which time the VL

master increments the addresses. The VL master latches the first 32-bits of data on the rising edge of the clock after sampling BRDY* asserted. The NxVL negates BLAST* to indicate the last transfer of the burst.

The example in Figure 46 does not include a snoop cycle on the NexBus because it assumes that the NxVL has found the status of the address among the last four snoop addresses that it stores. Any read cycle initiated by a VL master is treated as a MESI-protocol *exclusive* read. If the read hits the Nx586 processor's cache, the processor will (1) change its copy to the *invalid* state if the copy is in the *exclusive* state, or (2) intervene in the VL master's cycle to write-back its copy if it is in the *modified* state.

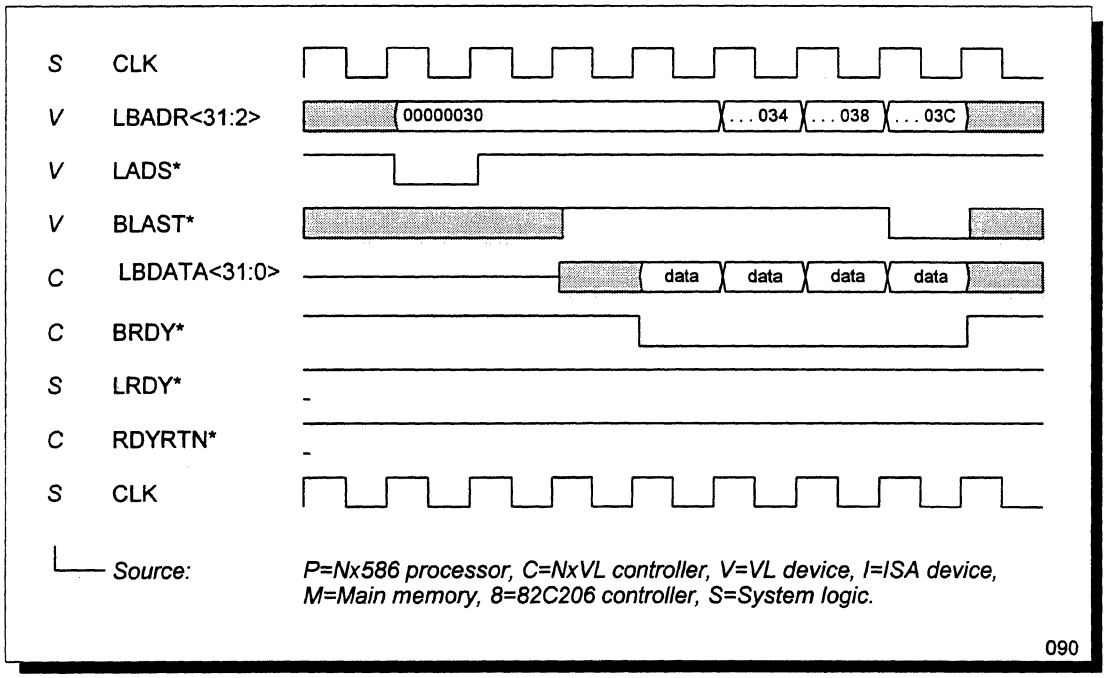


Figure 46 VL-Master 16-Byte Burst Read from Main Memory (Queue Hit)

Figure 47 shows a VL-master burst read from main memory that misses the read queue but hits the DRAM page. The cycle begins like a queue hit but BRDY* is held negated. The address on LBADR<31:2> is driven on the memory bus, MA<10:0> and the column addresses for four qwords are driven on CASA<n>*. The corresponding data is fetched in four cycles on MD<63:0>. The first two qwords of data subsequently appear as four 32-bit transfers on the LBDATA<31:0> bus. The second two qwords fetched from memory (those labeled "Prefetch") are the additional data needed to fill the 32-byte VL-bus prefetch queue.

The requested data is then driven on LBDATA<31:0> along with corresponding addresses on LBADR<31:2>. Throughout the transfer of these four dwords, BRDY* is asserted. During the last transfer, BLAST* is asserted for one clock and the cycle ends. At that point, the VL-bus prefetch queue holds all four qwords, two of which were requested and read by the VL master.

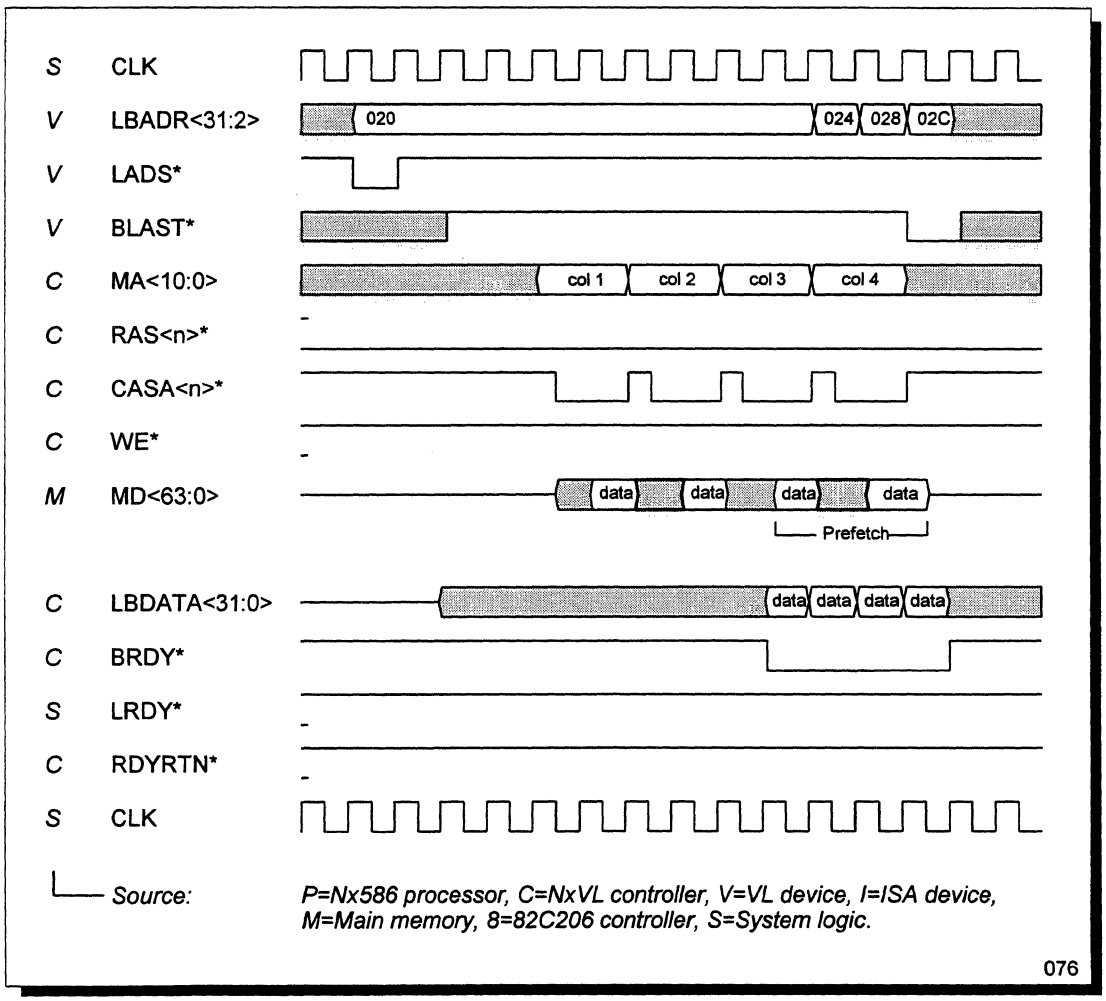


Figure 47 VL-Master 16-Byte Burst Read from Main Memory

VL Write to ISA Slave

When a VL-bus address does not map to main memory, the NxVL translates the VL-bus cycle onto the ISA-bus and handles all word assemblies. Figure 48 shows a VL-master 32-bit write to an ISA-bus 16-bit I/O slave. The VL master drives the address on LBADR<31:2> and asserts LADS*. Shortly after, the data is valid on LBDATA<31:0> and the address appears on SA<19:0>. The NxVL asserts SBHE* to request 16-bit transfers and asserts BALE for each of two 16-bit addresses on SA<19:0>. The transfers on SD<15:0> become valid when BALE and IOW* are asserted. The cycle ends when the NxVL asserts LRDY* to the VL master.

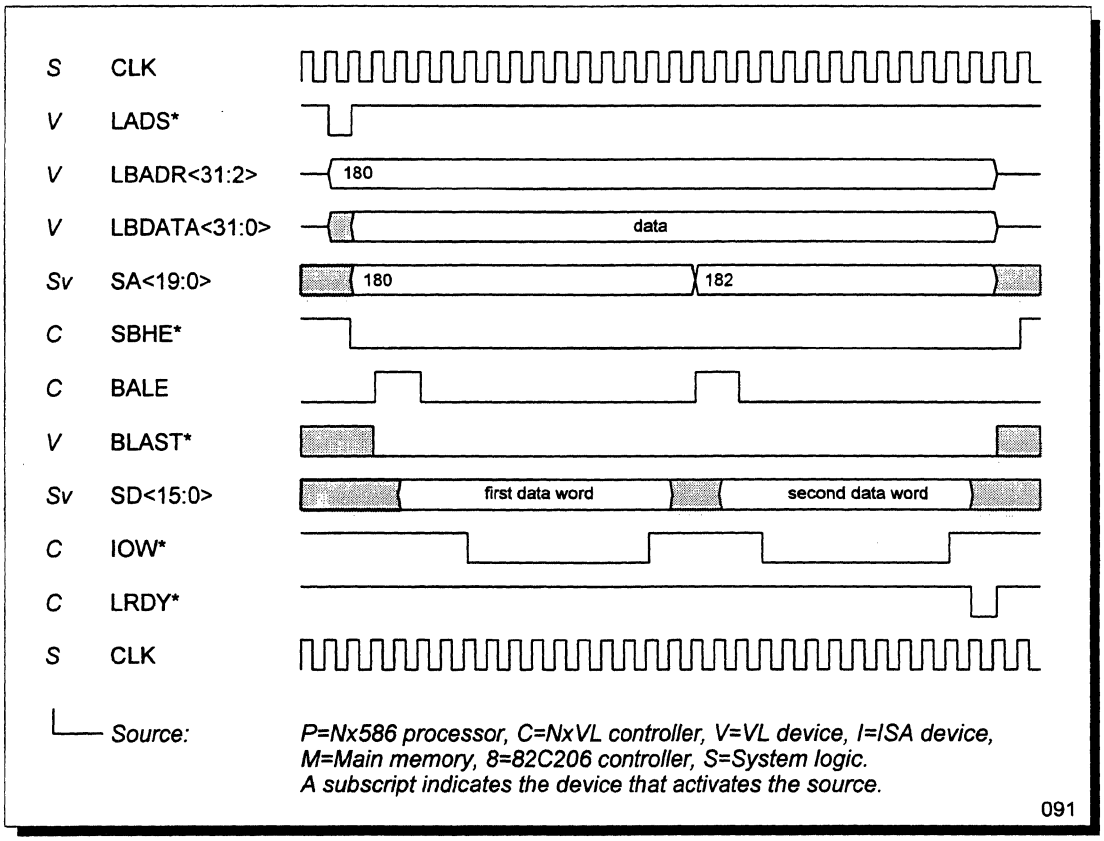


Figure 48 VL-Master 32-Bit Write to ISA-Bus 16-Bit I/O

VL Read from ISA Slave

Figure 49 shows a VL-master 32-bit read from an ISA 16-bit I/O slave. The timing is the same as the VL-write-to-ISA-slave shown in Figure 48, except that IOR* is asserted rather than IOW*, the data on SD<15:0> is driven by the ISA slave, and the arrival of data on LBDATA<31:0> is later.

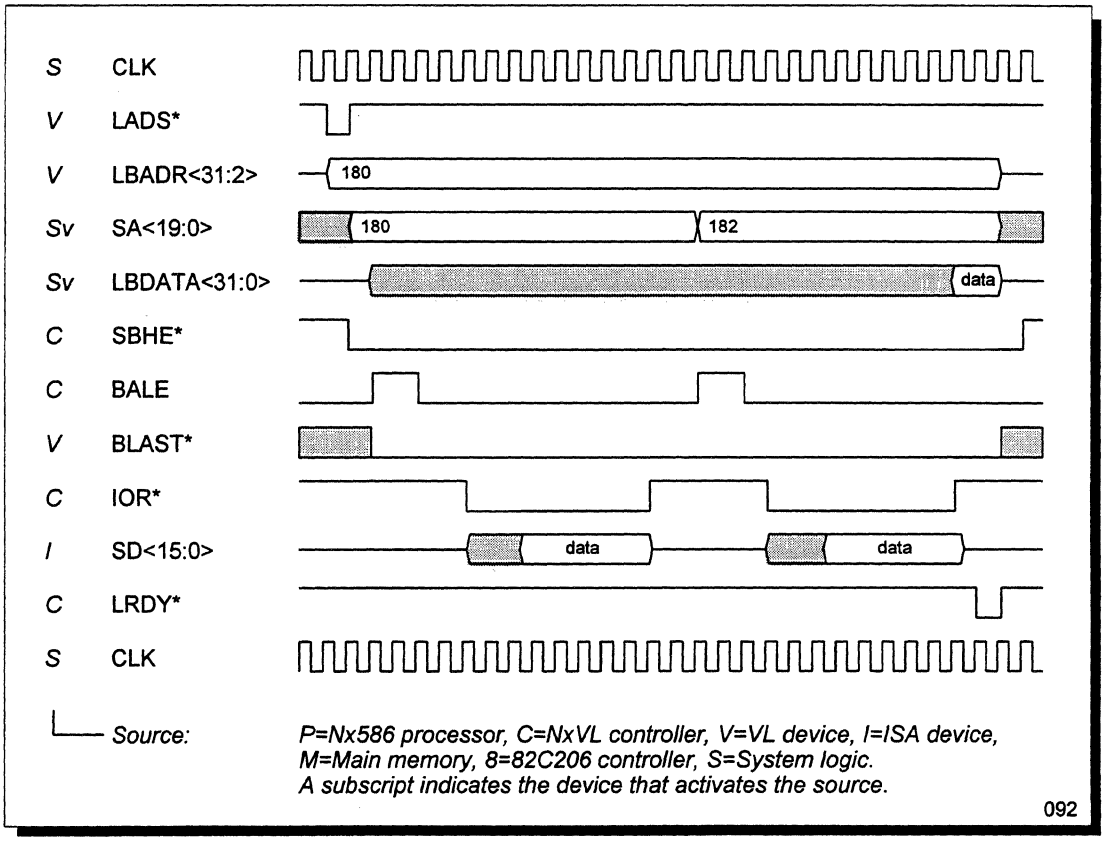


Figure 49 VL-Master 32-Bit Read from ISA-Bus 16-Bit I/O

ISA-Bus Master Operations

The NxVL can translate cycles initiated by the Nx586 processor or a VL master onto the ISA-bus. It can also translate ISA-master cycles to main memory or the VL-bus, and (when enabled) perform ISA-bus refresh cycles. DMA, interrupt control, and timers are provided by the 82C206 peripheral controller. During cycle translation, the NxVL handles word assembly and disassembly between the ISA-bus and other resources. Only the SA<1:0> portion of ISA-bus addresses is driven by the NxVL. All other address and data lines are interfaced to the VL-bus through the transceivers shown in Figures 20 and 21, in the *Hardware Architecture* chapter.

As with VL masters, any main-memory read by an ISA master is treated as a MESI-protocol *exclusive* read. If the read hits in the Nx586 caches, the processor will either change its copy to the *invalid* state or—if its copy is in the *modified* state—intervene in the ISA-master's cycle to write back the modified copy.

Bus Arbitration

To request control of all system buses, the ISA-bus master requests service from the Integrated Peripherals Controller (82C206) through that chip's DREQ<n> and DACK<n>* protocol. The 82C206 then asserts HHOLD to the NxVL on behalf of the ISA master. The NxVL responds in the next clock by negating GNT* to the Nx586 processor, if the processor or any other higher-priority bus master is not itself requesting the buses. One clock later the NxVL asserts HLDA to the 82C206, followed by the assertion of BALE to the ISA master one clock later.

After the ISA cycle completes, the IPC negates HHOLD. One clock later the NxVL negates HLDA and asserts GNT*. It negates BALE one clock after the negating HLDA.

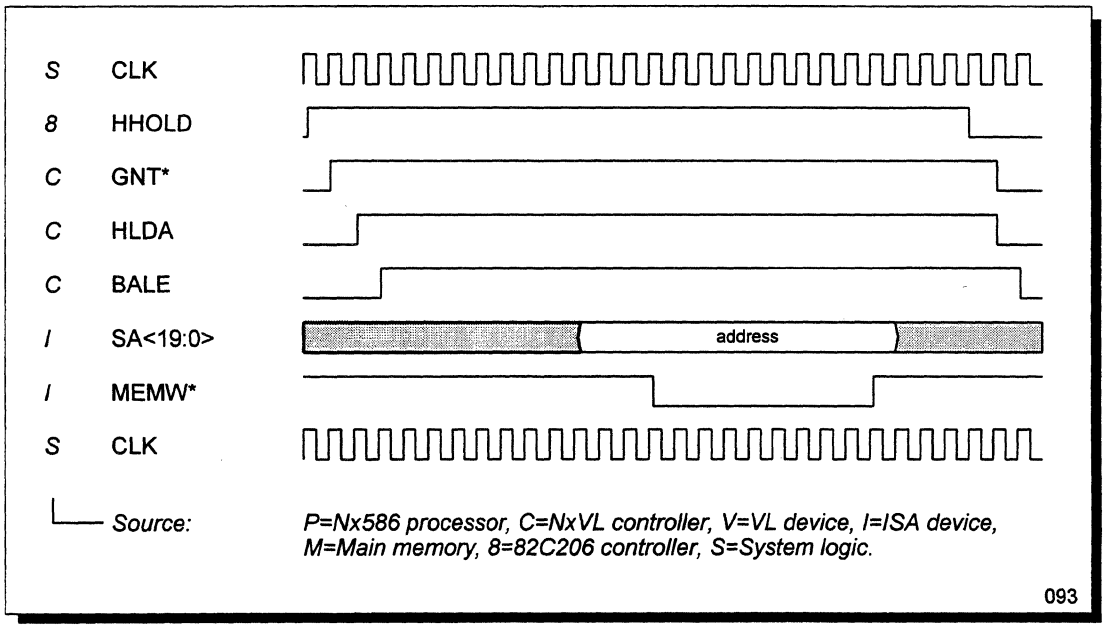


Figure 50 Bus Arbitration by ISA-Bus Master

ISA Write to Main Memory

Figure 51 shows an ISA-master 16-bit write to main memory. After the 82C206 has arbitrated for the buses and the NxVL asserts BALE, as described in the last section, the ISA master drives the address on LA<23:17> and SA<19:0> and asserts SBHE*. When the ISA master asserts MEMW* and drives the data on SD<15:0> (and onto LBDATA<15:0>), the NxVL negates IOCHRDY to add wait states. During this time, the NxVL snoops the Nx586 processor. In this example, the snoop is a miss, IOCHRDY is subsequently sampled asserted by the NxVL, and the cycle concludes with the negation of MEMW*.

For details of how data is subsequently transferred by the NxVL to main memory, see the section below entitled *Main-Memory Operations*.

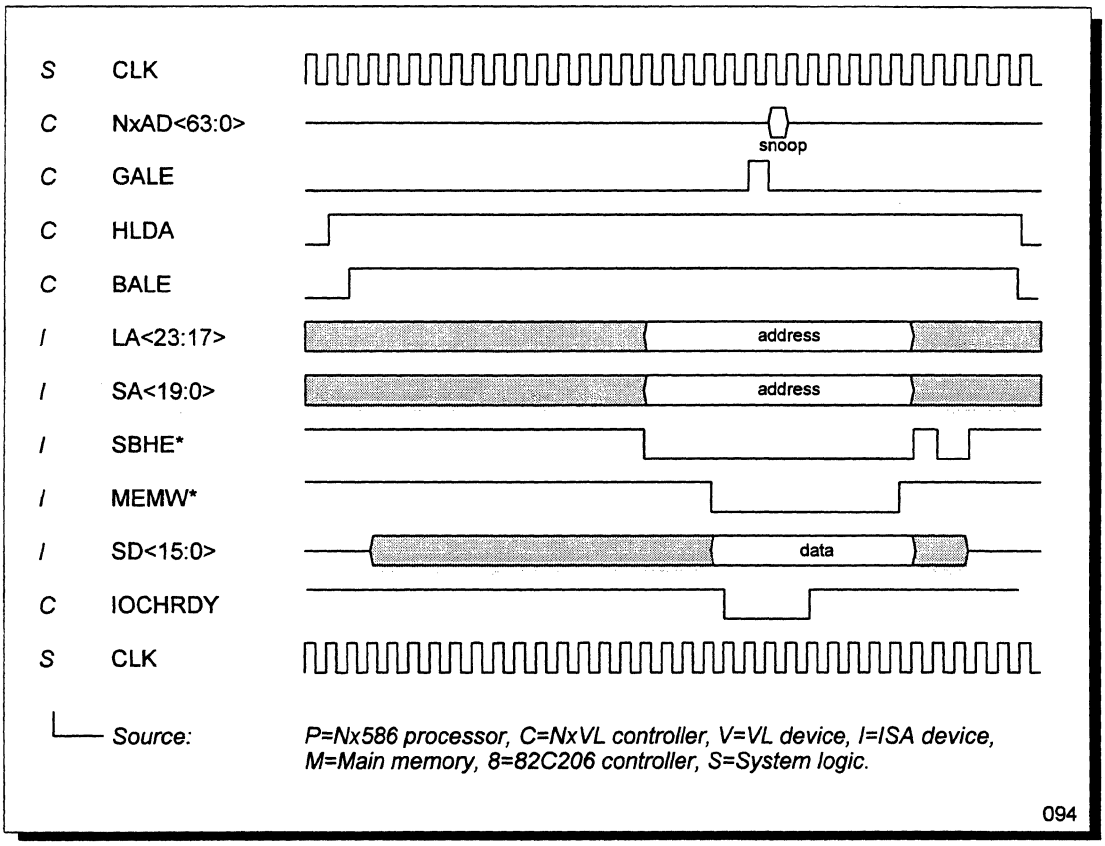


Figure 51 ISA-Master 16-Bit Write to Main Memory

ISA Read from Main Memory

Figure 52 shows an ISA-master 16-bit read from main memory. The timing is the same as for writes (Figure 51) except that MEMR* is asserted instead of MEMW* and the data is driven later and for a shorter time on SD<15:0>. For details of how data is fetched by the NxVL from main memory, see the section below entitled *Main-Memory Operations.*

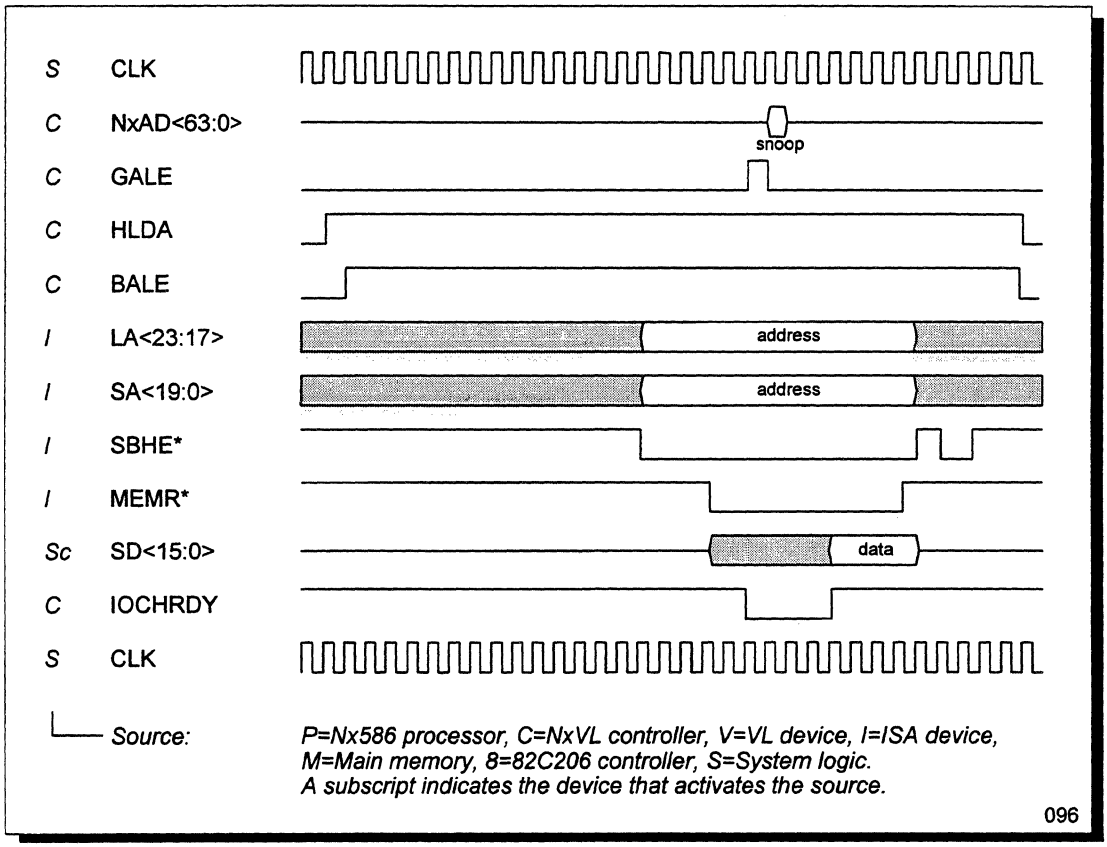


Figure 52 ISA-Master 16-Bit Read from Main Memory

ISA Write to VL Slave

When the address for an ISA cycle does not lie in main memory, the NxVL translates the cycle onto the VL-bus. If no VL-bus device responds with LDEV<n>*, the NxVL ignores the ISA master cycle.

Figure 53 shows an ISA-master 16-bit write to a VL memory slave. The ISA master drives the address on LA<23:17> and SA<19:0> and it asserts SBHE*.

The address appears on LBADR<31:2>. Shortly after, the VL slave responds with LDEV<n>* and the NxVL asserts MEMCS16*.

The ISA master then drives the data on SD<15:0> (which also appears on LBDATA<31:0>) and asserts MEMW*. The NxVL negates IOCHRDY, asserts LADS*, drives LBM/IO* high, and selects the lower two bytes with LBE<3:0>*. The VL slave assets LRDY* after or in the same clock in which it samples the data. Then, the NxVL re-asserts IOCHRDY and samples it active. The cycle ends when the ISA master negates MEMW*.

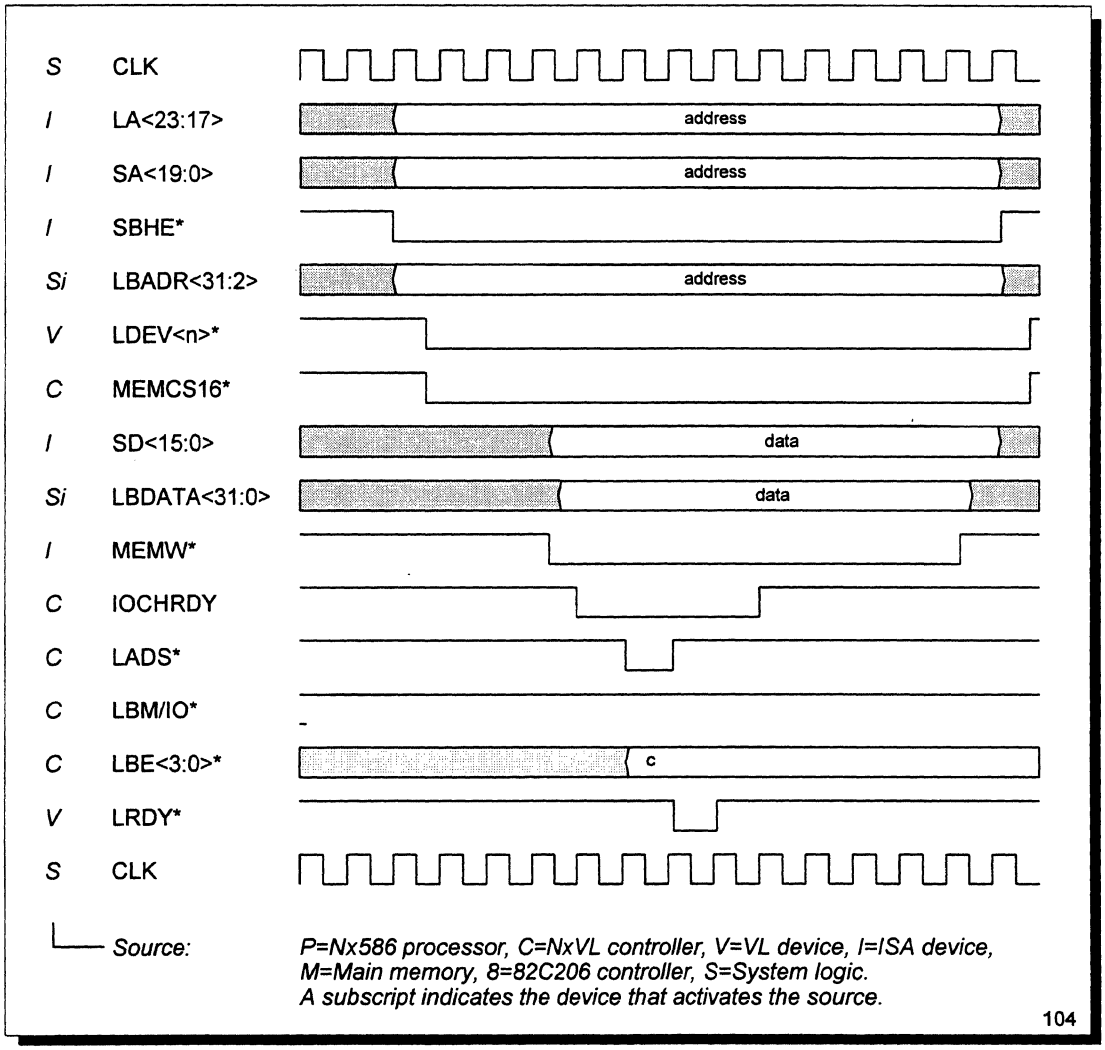


Figure 53 ISA-Master 16-Bit Write to VL-Bus Memory

ISA Read from VL Slave

Figure 54 shows an ISA-master 16-bit read from a VL I/O slave. The ISA master drives the write address on SA<19:0> (LA<23:17> is not used for I/O accesses) and asserts SBHE*. The address appears on LBADR<31:2>. Shortly after, the ISA master asserts IOR* and the NxVL negates IOCHRDY. System logic holds IOCS16* negated. The NxVL then asserts LADS*, drives LBM/IO* low, and selects the low byte on LBE<3:0>. The VL slave responds with LDEV<n>*, drives the data on LBDATA<15:0>, and asserts LRDY*. In the next clock, the data appears on SD<15:0>. One clock later, the NxVL asserts IOCHRDY and samples it. The cycle ends when the ISA master negates IOR*. Shortly after, the NxVL asserts RDYRTN* for one clock.

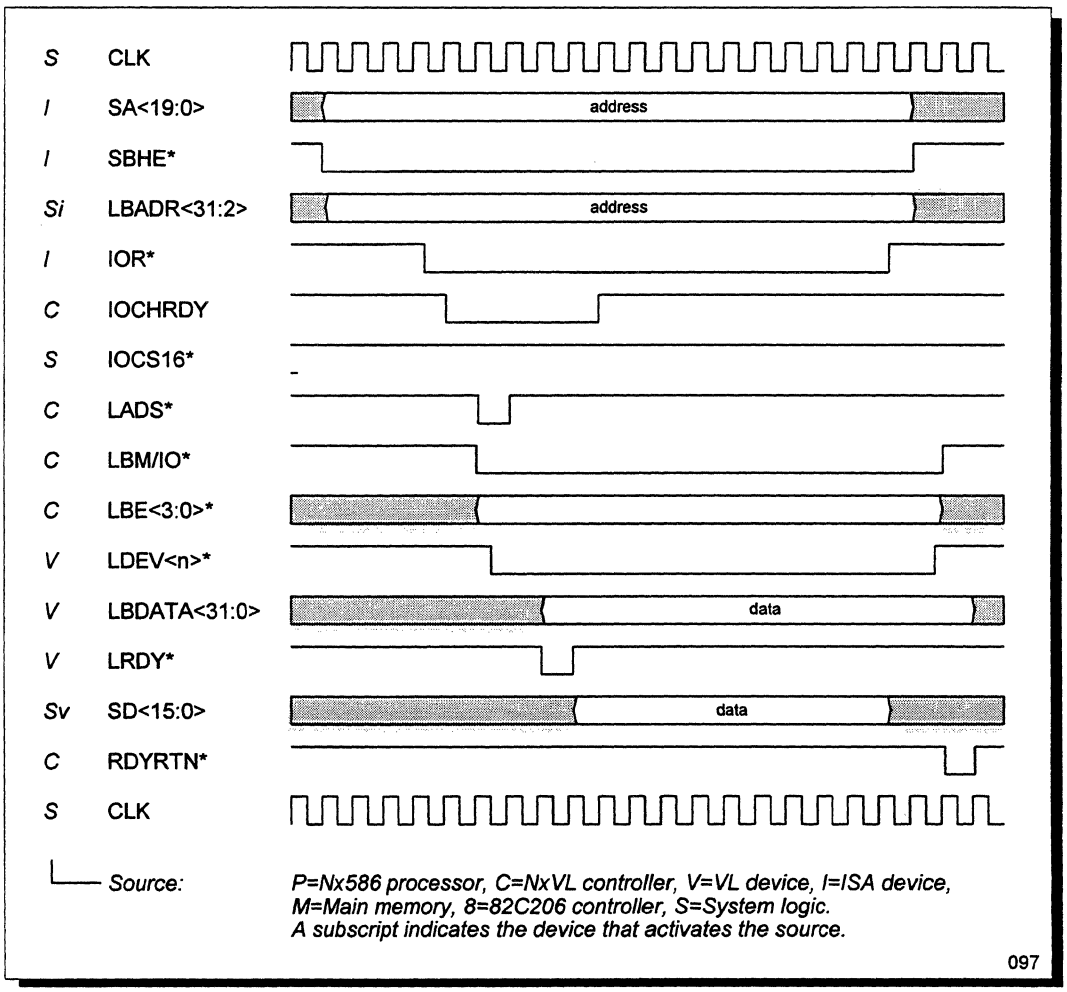


Figure 54 ISA-Master 16-Bit Read from VL-Bus I/O

ISA-Bus Memory Refresh

The NxVL can be configured in software to generate ISA-bus refresh cycles every 15μsec, as specified in the ISA protocol. If an ISA master controls the buses during the time when a refresh cycle is required, the master will assert REFRESH*. The NxVL, which maintains the refresh counter, sees REFRESH* and provides the address.

Figure 55 shows the refresh cycle for a NexBus clock. The cycle begins with the assertion of REFRESH* by an ISA master or the NxVL. Three refresh address is subsequently driven by the NxVL on SA<19:0> and MEMR* is asserted. IOCHRDY can be negated to extend the refresh cycle.

The time that REFRESH* remains asserted depends on the enabling of *turbo* mode. Turbo mode is enabled by bit 2 of the Port 92 register, and a non-turbo ISA HOLD speed is specified by bits 7:0 of the CFG0 configuration register. When turbo mode is disabled, the processor runs at the reduced speed specified in the CFG0 register. Non-turbo mode is used to slow down timing loops in software designed for pre-i386 processors. The slow-down is implemented by extending the ISA-bus refresh cycle using a counter.

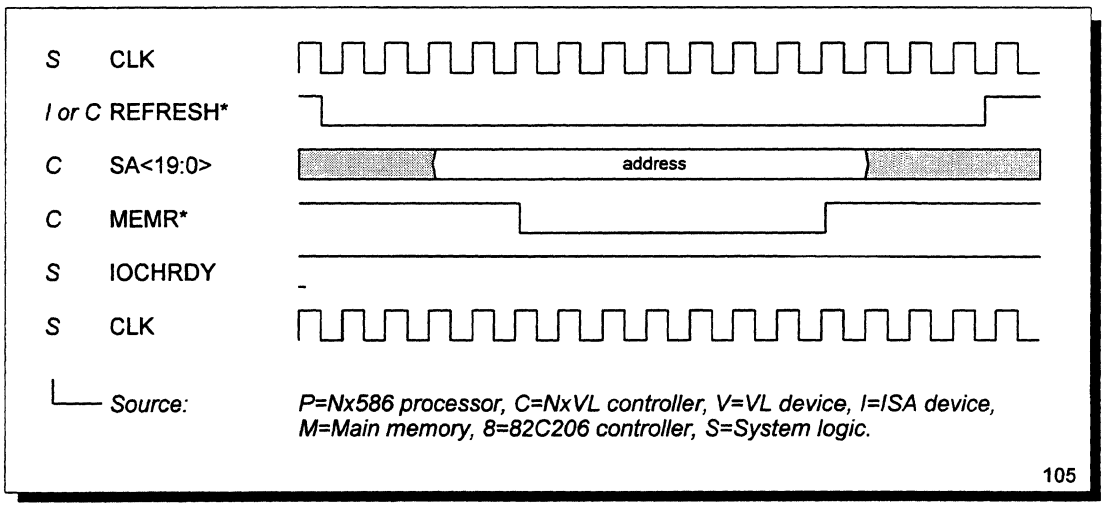


Figure 55 ISA-Bus Memory Refresh Cycle

DMA Operations

When there is no master on the ISA-bus, DMA transfers can be made between memory (main memory, VL-bus memory, or ISA-bus memory) and ISA-bus I/O. These transfers are controlled by the DMA controller within the IPC (82C206). The NxVL monitors the memory-control signals during these cycles. If the address lies within the main-memory or VL-bus address range, the NxVL translates the cycle to that source or destination.

ISA I/O to VL Memory

Figure 56 shows a DMA transfer from an 8-bit I/O device on the ISA-bus to a VL-bus memory. The 82C206 drives the address on LA<23:17> and SA<19:0>. The same address appears on LBADR<31:0> for the VL slave. The VL slave responds by asserting its LDEV<n>* and the NxVL asserts MEMCS16* based on LDEV<n>*.

The 82C206 asserts IOR* for the read from the ISA device, which then drives the data on SD<7:0>. The data appears on LBDATA<7:0>. Shortly after, the 82C206 asserts MEMW* for the write to the VL slave, causing the NxVL to negate IOCHRDY (to add wait states) and assert LADS*. MEMW* is held asserted throughout the write. The VL slave then asserts LRDY*. When LRDY* is negated, it causes the NxVL to re-assert IOCHRDY and sample the data. The 82C206 ends the cycle later by negating IOR* and MEMW*.

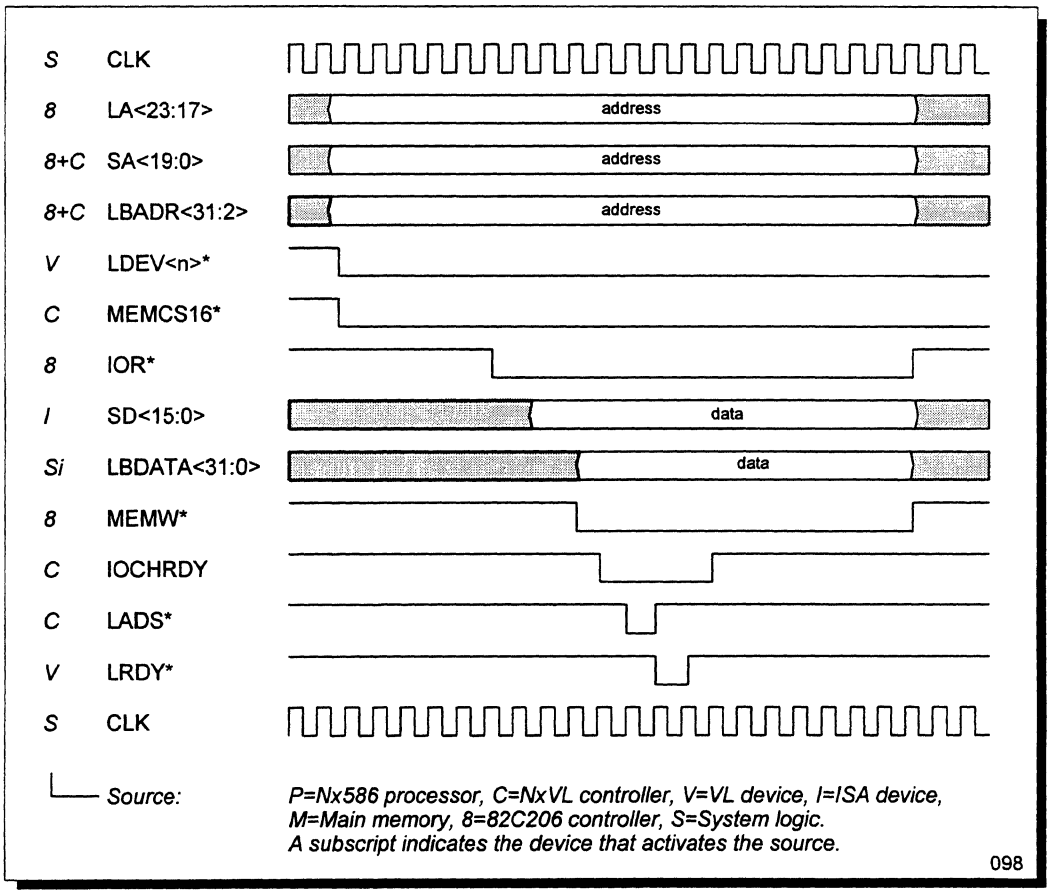


Figure 56 DMA Transfer from ISA-Bus 8-Bit I/O to VL-Bus Memory

Main Memory to ISA I/O

Figure 57 shows a DMA transfer from main memory to an 8-bit I/O slave on the ISA-bus. The 82C206 drives LA<23:17> and SA<19:0>, and the same address appears on LBADR<31:0> for main memory. No VL slave responds on the LDEV<3:0>* lines. Instead, the NxVL finds the address in its main-memory map and asserts MEMCS16*.

The 82C206 asserts MEMR* and IOW* simultaneously, causing the NxVL to negate IOCHRDY to add wait states. The arrival of data on LBDATA<7:0> and SD<7:0> depends on the state of the NxVL's ISA-bus read queue. When the data appears, the NxVL re-asserts IOCHRDY, samples the data, and the 82C206 ends the cycle by negating MEMR* and IOW*.

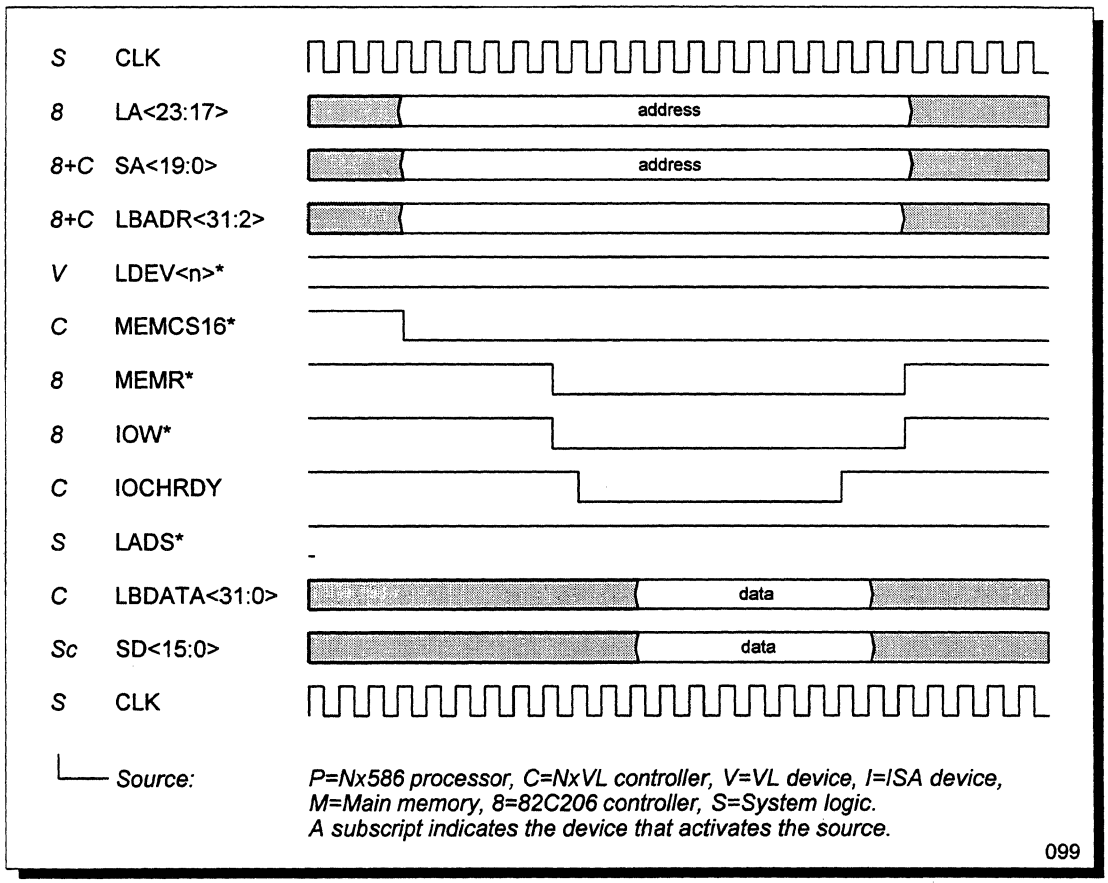


Figure 57 DMA Transfer from Main Memory to ISA-Bus 8-Bit I/O

Main-Memory Operations

In the prior bus-operation sections, write cycle have shown only the timing between the master initiating the write and the NxVL's write queue. The subsequent transfers from the write queue to main memory were not illustrated. In the case of read cycles, memory timing was only shown when the read missed its read queue and resulted in a fetch from main memory.

This section covers only the activity between the NxVL's read or write queues and the main memory, which typically happens *after* a master's read or write cycle terminates.

Slow Main-Memory Cycles

Figure 58 shows a slow write to main memory with no precharge. No precharge is required in this example because it assumes that RAS<n>* was negated before the access for at least 2, 3 or 4 clocks (depending on DRAM and processor speed). The timing assumes 80ns DRAMs, a 25MHz or 33MHz processor, and that CFG0 register bit 23 (fast memory-write enable) is cleared to 0 to specify slow writes.

One clock after the NxVL asserts WE*, drives the row address on MA<10:0>, and drives the data on MD<63:0>, it asserts RAS<n>*. Two clocks later, it asserts CASA<n>* (or CASB<n>*) for two clocks. The cycle ends when CASA<n>* and WE* are both negated.

Compared with the fast write shown in Figure 61, this slow write last one clock longer.

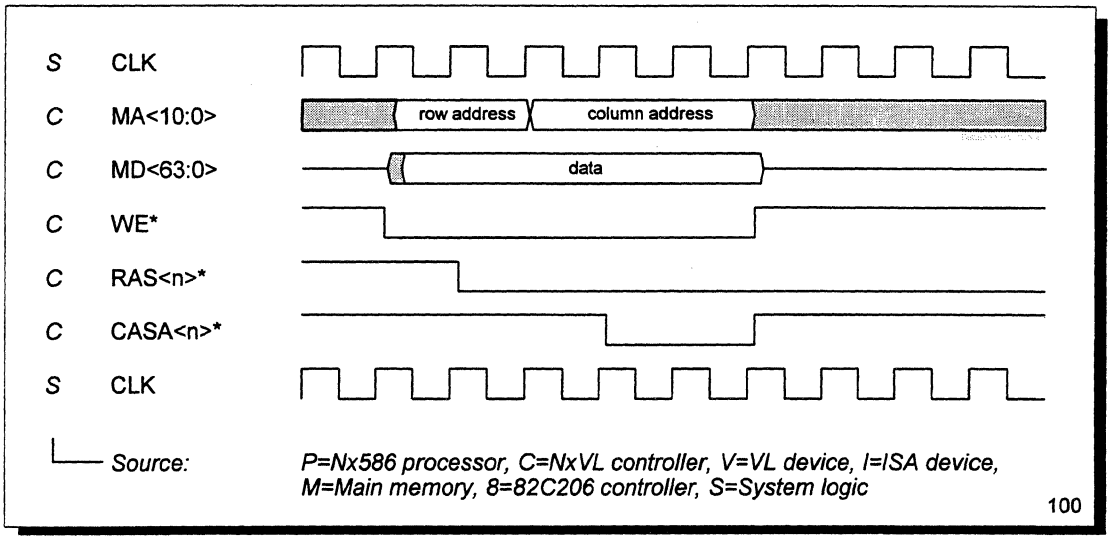


Figure 58 Slow Write to Main Memory (No Precharge)

Figure 59 shows a slow read from main memory with a precharge. The precharge and other timing in this example assumes 80ns DRAMs, a 25MHz or 33MHz processor, and that CFG0 register bit 22 (fast memory-read enable) is cleared to 0 to specify slow reads.

When the NxVL drives the row address on MA<10:0>, it negates RAS<n>* for the two-clocks precharge. One clock after RAS is asserted, the NxVL drives the column address on MA<10:0>. Another clock later, it asserts CASA<n>* (or CASB<n>*) for two clocks, during which the data is returned on MD<63:0>. The negation of WE* implies a read cycle. The cycle ends when CASA<n>* is negated.

Compared with the fast read shown in Figure 62, this slow read last two clocks longer.

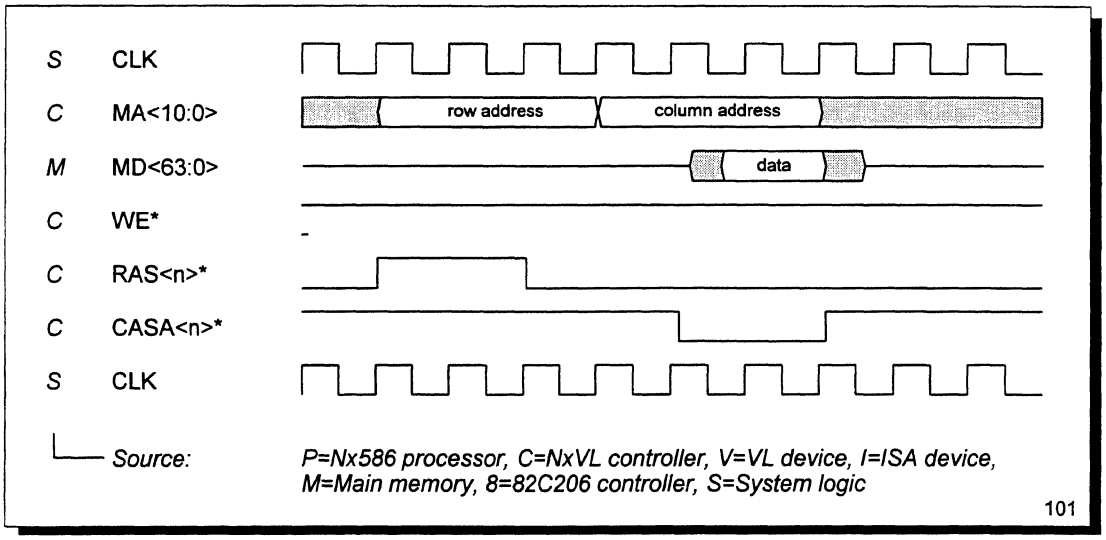


Figure 59 Slow Read from Main Memory (Precharge)

Figure 60 shows a slow read from main memory without a precharge, but one in which an overlapping precharge takes place on a row-address strobe (RAS<1>*) that is active at the beginning of the cycle for a DRAM chip other than the one accessed in the illustrated cycle. This example is based on the same assumptions regarding DRAM and processor speed as in Figure 60.

When the NxVL drives the row address on MA<10:0>, it negates the unused row-address strobe (RAS<1>*) for its precharge. One clock after the row address appears, the NxVL asserts RAS<0>* to address the required DRAM. Two clocks later, the NxVL asserts CASA<n>* (or CASB<n>*) for two clocks with WE* negated, during which the data is returned on MD<63:0>. The cycle ends when CASA<n>* is negated.

Compared with the fast read shown in Figure 62, this slow read with overlapping precharge last one clock longer.

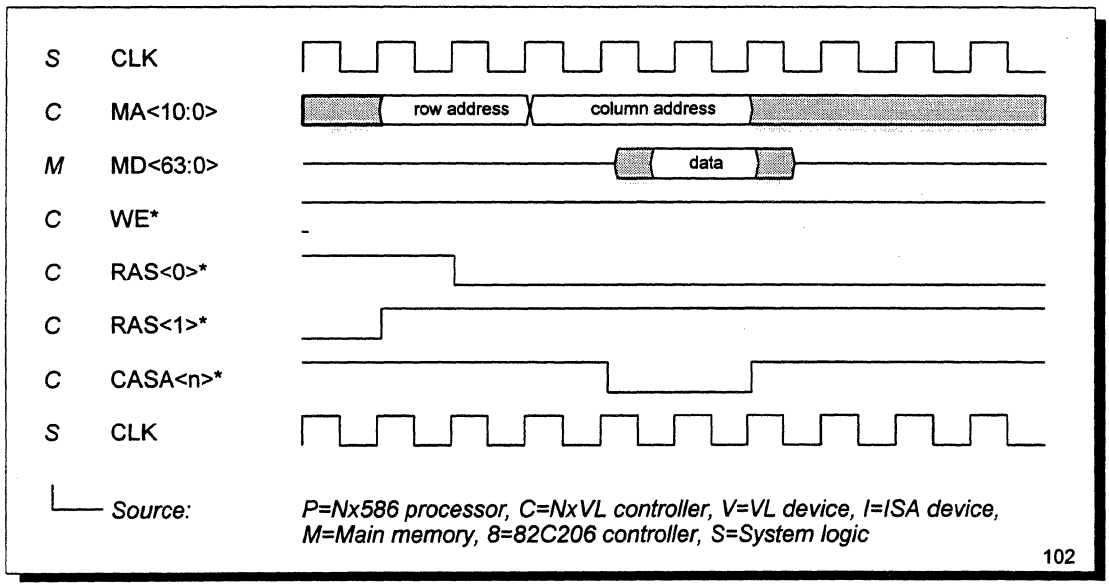


Figure 60 Slow Read from Main Memory (Overlapping Precharge)

Fast Main-Memory Cycles

Figure 61 shows a fast write to main memory with no precharge. This fast write is one clock faster than the slow write in Figure 59, and it is based on the same assumptions as the slow write except that fast writes are enabled with CFG0 register bit 23 (fast memory-write enable) set to 1.

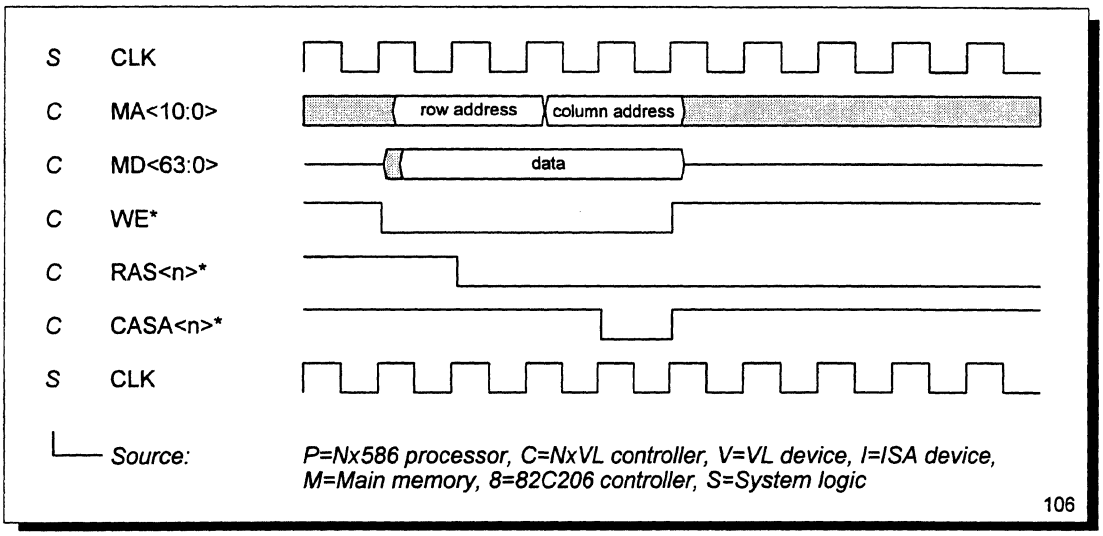


Figure 61 Fast Write to Main Memory (No Precharge)

Figure 62 shows a fast read from main memory with no precharge. This fast read is one clock faster than the slow read in Figure 60, and it is based on the same assumptions as the slow read except that fast reads are enabled with CFG0 register bit 22 (fast memory-read enable) set to 1.

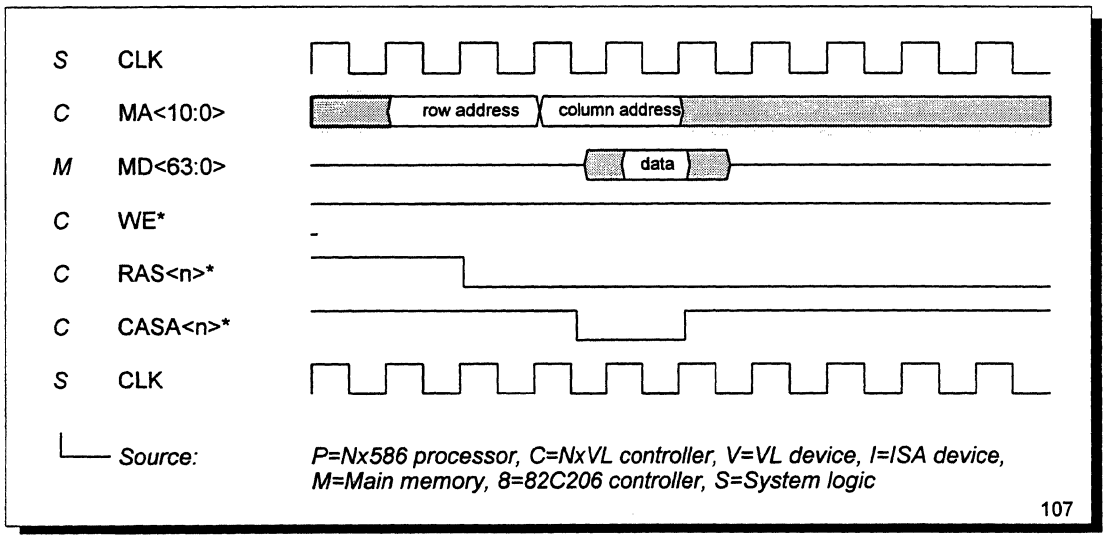


Figure 62 Fast Read from Main Memory (No Precharge)

Reset and Configuration Registers

Reset and Initialization

After reset, all registers that have default values are set to those values, and the system operates at the slowest, safest configuration. RESET is generated at power-up by external glue logic. The Nx586 processor and Nx587 floating pointing coprocessor are also reset by the NxVL at this time.

In addition to this power-up reset logic, the NxVL asserts RESETCPU* to the Nx586 processor under any of three conditions:

- Shutdown *or* Halt *Cycle*—When the Nx586 processor runs a shutdown or halt cycle, as seen by the NxVL on the NxAD<63:0> bus during the address and status phase. The RESETCPU* signal is asserted for 32 clocks (CLK).
- Keyboard Reset—The NxVL shadows writes to the keyboard controller (Ports 60h and 64h). A write to port 64h with the data F0 or, F2 or, ..., or FE (bit 0 being zero) causes the NxVL to assert RESETCPU* for 32 clocks (CLK), but after a delay of 288 clocks.
- Fast CPU Reset—A write to Port 92h with the data xxxxxxx1h causes the NxVL to assert RESETCPU* for 32 clocks (CLK), but after a delay of 288 clocks.

Configuration and Mapping Registers

There are five registers (the Configuration Register and Ports 60, 61, 64, 70, and 92) for configuring the functions and interface miscellaneous parameters within the NxVL. Figures 63 and 64 show the registers, and the following sections describe their contents.

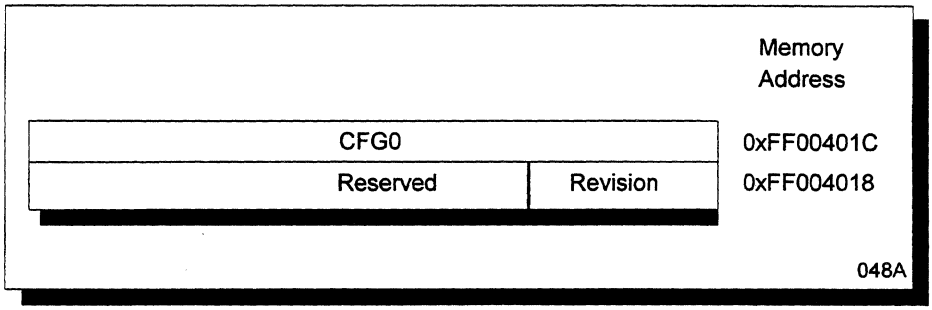


Figure 63 Configuration Registers

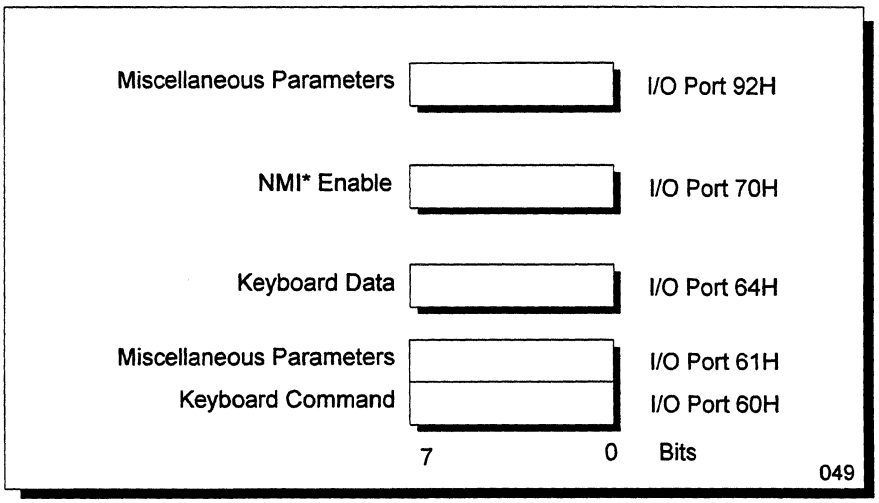


Figure 64 Configuration Registers (I/O-Mapped)

Configuration Register (CFG0)

The single 32-bit memory-mapped Configuration Register (CFG0) is accessed at address 0xFF00401C. It specifies a miscellaneous collection of functions and parameters for the NxVL, including clocking, memory access and refresh speed, port enabling, parity and transfer modes, ISA-bus memory cycle size, write-queue bypass, and use of FLASH EPROM.

Bit definitions for the CFG0 (address 0xFF00401C) are:

Bits	*Function*		
7:0	Non-Turbo ISA HOLD Speed: The number of clocks that the ISA refresh request is extended when the TURBO signal is negated (non-turbo mode). The default is 0x00h.		
9:8	Nexbus Clock (CLK) Speed:		
	00: 50MHz (default)		
	01: 40MHz		
	10: 33MHz		
	11: 25MHz		
11:10	ISA-Bus Clock Speed		
	00: Divide by six (default)	CLK/6	
	11: Divide by five	CLK/5	
	10: Divide by four	CLK/4	(default)
	01: Divide by three	CLK/3	
12	80ns Memory Enable: When set to 1, enables 80ns timing for RAS/CAS precharge and pulse width. When cleared to 0, enables timing for slower memory. The default is 0 (disabled). Figure 65 shows the minimum RAS-low, RAS-high, and precharge times for each setting of bit 12.		

bit 12	*Parameter*	*25MHz*	*33MHz*	*40MHz*
1	Minimum RAS-Low Time	2 clocks (80 ns)	3 clocks (90 ns)	4 clocks (100 ns)
	Minimum RAS-High Time	2 clocks (80 ns)	3 clocks (90 ns)	3 clocks (90 ns)
	RAS Precharge Time	2 clocks (80 ns)	3 clocks (90 ns)	3 clocks (75 ns)
0	Minimum RAS-Low Time	3 clocks (120 ns)	4 clocks (120 ns)	4 clocks (100 ns)
	Minimum RAS-High Time	3 clocks (120 ns)	4 clocks (120 ns)	4 clocks (100 ns)
	RAS Precharge Time	3 clocks (120 ns)	4 clocks (120 ns)	4 clocks (100 ns)

Figure 65 CFG0 Bit-12 Minimum Times

13 *Reserved:* Must be set to 1.

14 Port 92H Enable: When set to 1, enables reading and writing port 92h. When cleared to 0, cycles to this port are translated onto the VL-bus or ISA-bus. The default is 0 (disabled).

15 Memory Parity-Error Enable: When set to 1, enables reporting of errors on the main-memory bus. When enabled, parity errors are reported by the assertion of NMI*, if NMI* is enabled in Port 70. The default is 0 (disabled).

16 *Reserved:* Must be cleared to 0.

17 *Reserved:* Must be cleared to 0.

18 *Reserved:* Must be cleared to 0.

19 *Reserved:* Must be set to 0.

20 *Reserved:* Must be set to 1.

21 *Reserved:* Must be cleared to 0.

22 Fast Memory-Read Enable: When set to 1, CAS-low takes 1 clock and CAS-high takes 1 clock on writes. When cleared to 0, CAS-low takes 2 clocks and CAS-high takes 1 clock. The default is 0 (slow write). Figure 66 summarizes these CAS-low and CAS-high times.

bit 22	*CAS-Low Time*	*CAS-High Time*
1	1.0 clock	1.0 clocks
0	2.0 clocks	1.0 clock

Figure 66 CFG0 Bit-22 CAS Timing for Memory Reads

23 Fast Memory-Write Enable: When set to 1, CAS-low takes 1 clock and CAS-high takes 1 clock on writes. When cleared to 0, CAS-low takes 2 clocks and CAS-high takes 1 clock. The default is 0 (slow write). Figure 67 summarizes these CAS-low and CAS-high times.

bit 23	*CAS-Low Time*	*CAS-High Time*
1	1.0 clock	1.0 clocks
0	2.0 clocks	1.0 clock

Figure 67 CFG0 Bit-23 CAS Timing for Memory Writes

24 *Reserved:* Must be cleared to 0.

25 Fast VL-Bus Transfer Enable: When set to 1, enables fast transfers on the VL-bus. Clearing the bit to 0 slows down VL-bus transfers by providing an additional clock of setup time for addresses (before LADS* is asserted during NxVL cycles), and for data. When cleared to 0, RDYRTN* is delayed by a clock on reads to enable the NxVL to sample data with faster setup times; also, during VL-master cycles, the NxVL delays its assertion of LRDY* so as to enable faster setup times for data. The bit is provided as a fall-back in case of timing problems on the VL-bus at speeds of 40MHz and above. When bits 9:8 are chosen to implement a 40MHz NexBus clock, bit 25 is cleared to 0. The default is 0 (slow transfers).

26 *Reserved:* Must be cleared to 0.

27 FLASH EPROM Write Enable: When set to 1, enables writes to FLASH EPROM. The default is 0 (disabled).

28 FLASH EPROM Read Enable: When set to 1, enables reads to FLASH EPROM. The default is 0 (disabled).

29:31 *Reserved:* Must be cleared to 0.

Port 61

The 8-bit I/O-mapped register at Port 61h specifies miscellaneous I/O parameters relating to ISA-bus events, timer detection, and speaker data.

Bit definitions for I/O Port 61h are:

Bit	*Function*	*Type*	
7	Parity Error—When set to 1, a parity error occurred in main-memory. (see bit 2)	R/O	
6	I/O Channel Check—When set to 1, there was a parity error on ISA-bus memory. (see bit 3)	R/O	
5	Timer OUT2 Detect—When set to 1, the timer 2 output is set.	R/O	
4	ISA Refresh Detect—When set to 1, there was an ISA-bus memory refresh detected.	R/O	
3	I/O Channel-Check Enable—When set to 0, I/O channel checking is enabled. (see bit 6).	R/W	Default 0 (enabled)
2	Parity-Check Enable—When set to 0, parity-error checking of main memory is enabled. (see bit 7)	R/W	Default 0 (enabled)
1	Speaker Data—When set to 1, the speaker output port is enabled.	R/W	Default 0 (disabled)
0	Timer Gate 2 Enable—When set to 1, the timer gate 2 is enabled.	R/W	Default 0 (disabled)

Port 70 (NMI*)

Bit 7 of the I/O-mapped shadow register at Port 70h controls the non-maskable interrupt signal, NMI*. When bit 7 is set to 0, the NMI* signal is enabled (default set to 0, enabled). A write to this port on the ISA-bus is also shadowed into this bit. The state of NMI* enabling can be read from bit 3 of Port 92h.

Port 92

The 8-bit I/O-mapped register at Port 92h specifies miscellaneous parameters, including the state of non-maskable interrupt enabling, the state of Turbo mode, and enabling of fast GATEA20 and RESETCPU* signals. To function, however, Port 92h must be enabled by setting bit 14 to 1 in the Configuration Register (CFG0).

Bit definitions for I/O Port 92h are:

Bits	*Function*	*Type*	
Bits 7:5	*Reserved (must be set to 1)*	R/O	
Bits 4	*Reserved (must be set to 1)*	R/O	
Bits 3	NMI* Enable—When set to 1, non-maskable interrupts are enabled.	R/O	Default 0 (disabled)
Bits 2	Turbo Mode Enable—When set to 1, turbo mode is enabled.	R/O	
Bits 1	Fast GATEA20—When set to 1, GATEA20 is asserted.	R/W	Default 0 (disabled)
Bits 0	Fast RESETCPU* —When set to 1, the NxVL asserts RESETCPU* for 32 clocks (CLK), but after a delay of 288 clocks.	R/W	Default 0 (disabled)

Port 60 and Port 64 (Keyboard)

The NxVL shadows successive writes to the I/O-mapped keyboard shadow registers at command-port 60h and data-port 64h to generate the GATEA20 and RESETCPU* signals. The shadowing is done because of the 4-Volt operation of the NxVL and Nx586 chips; this arrangement allows a 5-Volt keyboard controller to be used without the need for the voltage translator between that controller and the Nx586 processor.

A write to command-port 64h with data 0xD1, followed by a write to data-port 60h, results in data bit 1 being reflected on the internal KBDG20. A write to command-port 64h with data 0xD1, followed by a write to data-port 60h with data 110111111 (DFh) causes the KBDG20 to go high if it were low.

A write to command-port 64h with data 0x1111xxx0 causes the internal KBDRST to pulse, which in turn cause RESETCPU* to be asserted for 32 Nexbus clocks after 6.72µs. The NxVL also supports continuous driving of the internal KBDRST by first writing to command-port 64h with data 0xD1h and then writing a 0 to bit 0 of data-port 60h.

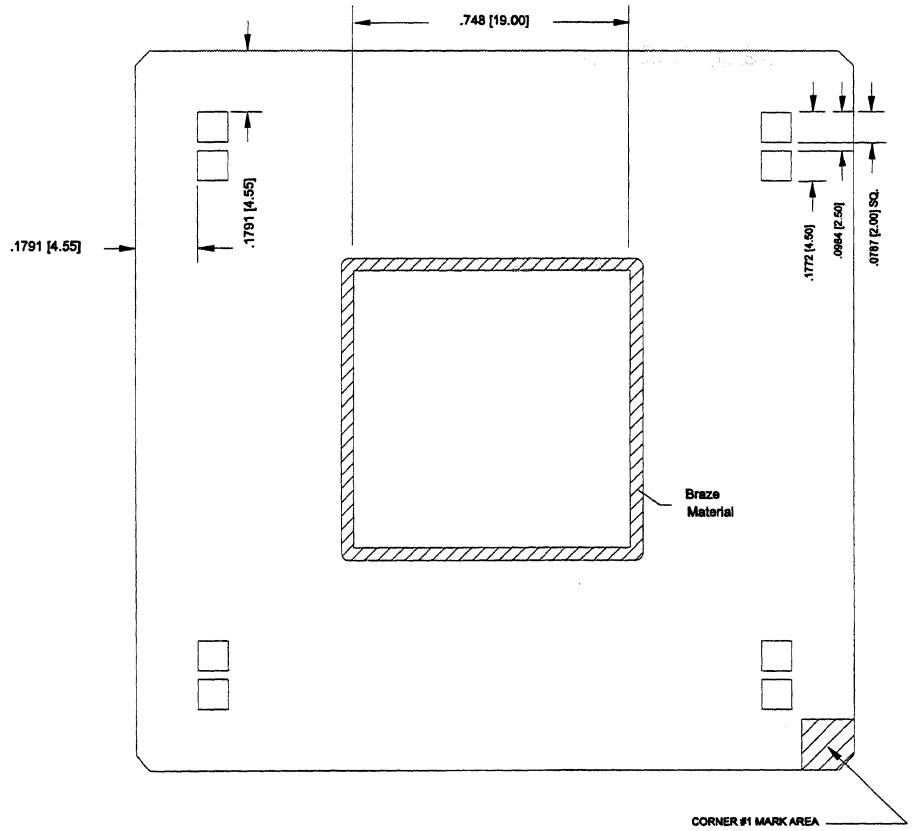
Electrical Data

For Electrical Data See Document "NxVL Electrical Specifications"

Order # NxDOC-ES002-01-W

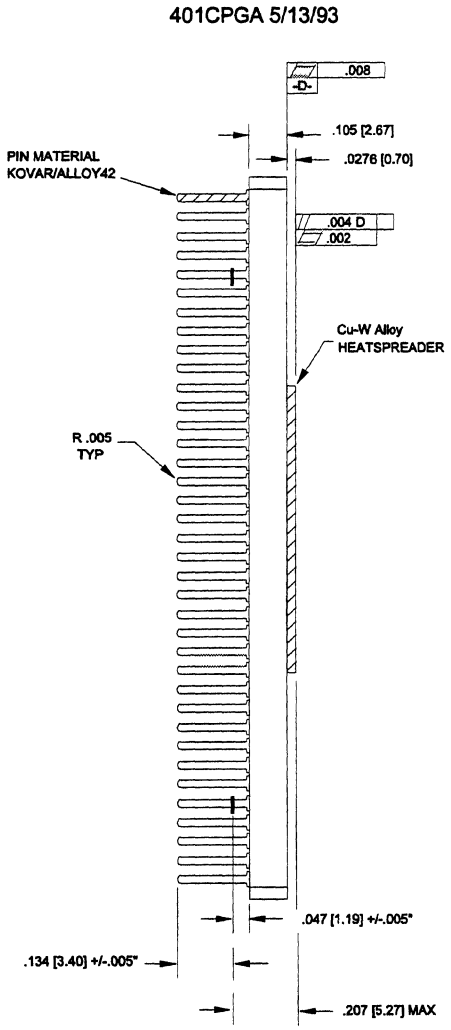
Mechanical Data

401CPGA 5/13/93



NOTE: ALL DIMENSIONS ARE +/- 1% UNLESS OTHERWISE INDICATED.

Figure 68 NxVL Package Diagram (top)



NOTE: ALL DIMENSIONS ARE +/- 1% UNLESS OTHERWISE INDICATED.

Figure 69 NxVL Package Diagram (side)

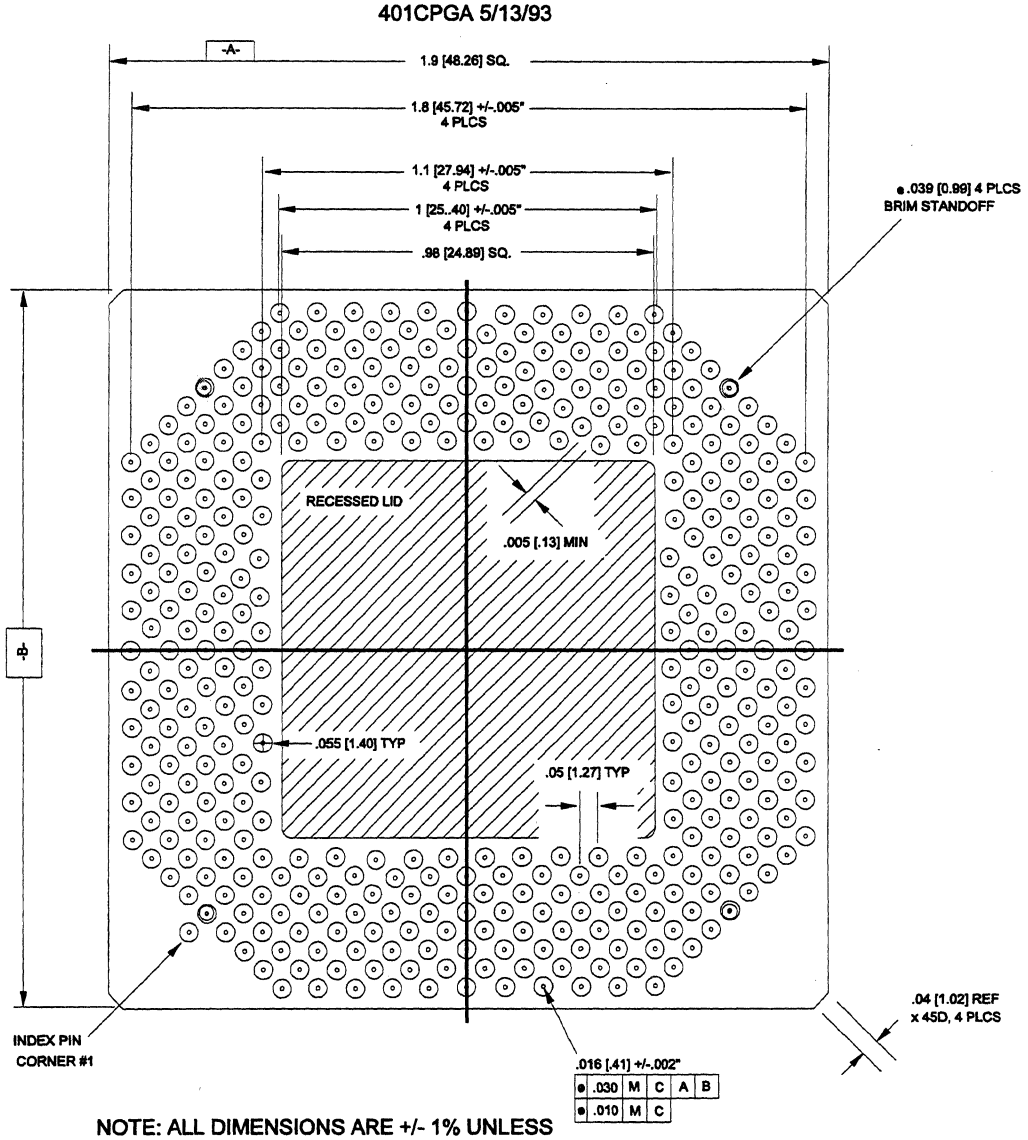


Figure 70 NxVL Package Diagram (bottom)

Glossary

Access—A bus master is said to "have access to a bus" when it can initiate a *bus cycle* on that bus. Compare *bus ownership.*

Adapter—A central processor, memory subsystem, I/O device, or other device that is attached to a slot on the NexBus, VL-Bus, or ISA bus. Also called a *slot.*

Aligned—Data or instructions that have been shifted until the relevant bytes begin in the least-significant byte position.

Allocating Write—A read-to-own (read for exclusive ownership of cacheable data) followed by a write to the cache.

Arbiter—A resource-conflict resolver, such as the NexBus arbiter. The NxVL chip includes a NexBus arbiter.

b—Bit.

B—Byte.

Block—See *cache block.*

Block Operation—Burst transfers of four qwords (32 bytes). Compare *cache block.*

Bus Cycle—A complete transaction between a bus master and a slave. For the Nx586 processor, a bus cycle is typically composed of an address and status phase, a data phase, and any necessary idle phases. Also called a *bus operation,* or simply *operation.*

Bus-Master Transfers—Transfers between main memory and masters on the VL-Bus or ISA bus that are controlled by those masters. See *DMA transfers.*

Bus Operation—Same as *bus cycle.*

Bus Ownership—A bus is said to be owned by a master when the master can initiate cycles on the bus. In single-processor systems supported by the NxVL chip, the NxVL chip arbitrates access to all buses. The master to which bus ownership is granted controls only its own interface with the NxVL chip. The NxVL chip, on behalf of that master, acts as a master on the other buses in the system. It does this so as to support the master in the event that a bus-crossing operation is requested. Compare *access.*

Bus Phase—Part of bus cycle that lasts one or more bus clocks. For example, it may be a transfer of address and status, a transfer of data, or idle clocks.

Bus Sequence—A sequence of bus cycles (or operations) that must occur sequentially due to their being explicitly locked by the continuous assertion of the master's AREQ* and/or LOCK* signals, or implicitly locked by the GDCL signal.

Cache Hit—An access to a cache block whose state is *modified, exclusive,* or *shared* (i.e., not *invalid).* Compare *cache miss.*

Cache Lookup—Comparison between a processor address and the cache tags and state bits in all four sets (ways) of a cache.

Cache Miss—An access to a cache block whose state is *invalid.* Compare *cache hit.*

Clean—Same as *exclusive.*

Clock Cycle—Unless otherwise stated, this a *processor-clock cycle* rather than a bus-clock cycle. The Nx586 processor's clock runs at twice the frequency of the NexBus clock (CLK). The level-1 cache runs at the same frequency as the processor clock. The level-2 cache runs at the same frequency as the NexBus clock (CLK).

Clock Phase—One-half of a processor clock cycle.

Crossing Operation—Same as *bus-crossing operation.*

Cycle—See *bus cycle, clock cycle, bus phase,* and *clock phase.*

Device—Same as *adapter.*

Dirty—Same as *modified.*

DMA *Transfers*—In systems using the NxVL chip, transfers between memory (main memory, VL-Bus memory, or ISA-bus memory) and ISA-bus I/O that are controlled by the 82C206 peripherals controller. DMA transfers are done when there is no master on the ISA bus to handle such transfers. The NexBus can support DMA, but not in the single-processor configurations implemented with the NxVL chip.

Dword—A doubleword. A four-byte (32-bit) unit of data that is addressed on an four-byte boundary. Also called a *dword* (doubleword). Same as *quad.*

Exclusive—One of the four states that a 32-byte cache block can have in the MESI cache-coherency protocol. *Exclusive* data is owned by a single caching device and is the only known-correct copy of data in the system. Also called *clean* data. When exclusive data is written over, it is called *modified* (or *dirty)* data.

Flush—(1) To write back a cache block to memory and invalidate the cache location, also called *write-back and invalidate,* or (2) to invalidate a storage location such as a register without writing the contents to any other location.

Group Signal—A NexBus control signal that represents the logical OR of several inputs. These signals typically have signal names that begin with the letter "G". They provide fast control response for systems with multiple devices on the NexBus. An active-low signal (such as ALE*) is driven by each NexBus device to a NAND gate on the backplane to produce an active-high group signal (such as GALE). This group signal—in effect an logical OR of all its inputs—is then distributed back to each NexBus device. This type of signaling works faster than open-collector buses.

Intervenor—A caching device on the NexBus that intervenes in the NexBus operation of another master to provide data from one of its modified cache blocks. To maintain cache coherency, the intervenor must update memory before other NexBus masters can access that location in memory.

Invalid—One of the four states that a 32-byte cache block can have in the MESI cache-coherency protocol. *Invalid* data is not correctly associated with the tag for its cache block.

Invalidate—To change the state of an cache block to *invalid.*

ISA Bus—Industry-Standard Architecture bus, an architecture derived from the IBM PC/AT architecture. It supports 24 bits of address and 8 or 16 bits of data. The SA and XA address buses and the SD and XD data buses are part of the ISA bus.

LA Bus—A 7-bit address bus on the ISA bus. In ISA terminology, it is called LA<23:17> and is used together with the SA bus (SA<19:0>) to form the entire ISA-bus address. In systems using the NxVL system controller, the LA bus is driven by LBADR<23:17> on the VL-Bus, and the SA bus is driven by a combination of LBADR<19:2> and SA<1:0>; the two bits of overlap between LBADR<23:17> and LBADR<19:2> are driven identically.

L1—The level-1 cache located on the Nx586 processor.

L2—The level-2 cache located in SRAM connected to the processor's SRAM bus and controlled by logic on the Nx586 processor.

Line—See *cache block.*

Main Memory—See *memory.*

Memory—A RAM or ROM subsystem located on any bus, including the *main memory* most directly accessible to a processor. In single-processor systems using the NxVL, main memory is the DRAM on the NxVL's memory bus.

MESI—The cache-coherency protocol used in the Nx586 processor. In the protocol, cached blocks in the L2 write-back cache can have four states (modified, exclusive, shared, invalid), hence the acronym MESI. See *modified, exclusive, shared,* and *invalid.*

Modified Write-Once Protocol—The cache-coherency protocol used in the Nx586 processor. See *MESI.*

Modified—One of the four states that a 32-byte cache block can have in the MESI cache-coherency protocol. *Modified* data is *exclusive* data that has been written to after being read from lower-level memory, and is therefore the only valid copy of that data. Also called *dirty or stale.*

MWO—See *modified write-once protocol.*

NexBus—A 64-bit synchronous, multiplexed, multiprocessor bus defined by NexGen. The Nx586 and NxVL are interfaced on this bus. When bus transceivers are used between the Nx586 processor and other system devices, the NexBus on the processor (unbuffered) side of the transceivers is called NxAD<63:0> and on the buffered side is called AD<63:0>. When no bus transceivers are used, as in systems using the NxVL system controller (which can emulate the bus transceivers), the NexBus is called NxAD<63:0>.

Non-Turbo Mode—An NxVL mode in which the system runs at a reduced speed. The mode is entered by clearing by bit 2 of Port 92 and is affected by the value in bits 7:0 of configuration register CFG0. It is used to correctly implement timing loops in software designed for pre-386 processors. The slow-down is implemented by extending the ISA-bus refresh cycle using a counter, thereby preventing access to the bus by the processor. Compare *turbo mode.*

No-Op—A single-qword operation with BE<7:0>* all negated. No-ops address no bytes and do nothing except consume processor cycles.

NP—Same as *Nx587* and *floating point coprocessor.*

Numerical Coprocessor—The Nx587 floating point coprocessor (NP). The logic in the coprocessor is integrated into the parallel pipeline of the Nx586. The Nx587 is binary-compatible with all x87 and i486 floating-point code.

Nx586—The Nx586 processor (CPU).

Nx587—The Nx587 floating point coprocessor (NP).

NxVL—A NexBus systems logic that supports a single the Nx586/587, main memory, 82C206 peripheral controller, VL-Bus, and ISA-bus.

Octet—Same as *qword.*

Operation—See *bus operation* and *micro-operation.*

Owned—A cache block whose state is *exclusive* (owned clean) or *modified* (owned dirty). See also *bus ownership.*

Ownership—See *bus ownership.*

Page—(1) a DRAM page of 256kb (single-sided SIMMs) or 512kb (double-sided SIMMs), or (2) a 4kB block of adjacent memory locations, used by the paging hardware.

Peripheral Controller—A chip that supports interrupts, DMA, timer/counters, and a real-time clock. Also known as the IPC or 82C206.

Phase—See *bus phase* and *clock phase.*

Present—Same as *valid.*

Processor—Unless otherwise specified, an Nx586 processor. When the NxVL chip is used in a system design, there can be only one processor on the NexBus so "the processor" refers to this single Nx586 processor.

Processor Clock—The Nx586 processor clock. See *clock cycle.*

Qword—A quadword. An eight-byte unit of data that is addressed on an eight-byte boundary. Also called an *octet.*

RTC—Real-time clock. In systems using the NxVL, this is supplied by an IPC.

SA Bus—A 20-bit address bus on the ISA bus. In ISA terminology, it is called SA<19:0> and is used together with the LA bus (LA<23:17>) to form the entire ISA-bus address. In systems using the NxVL system controller, the LA bus is driven by LBADR<23:17> on the VL-Bus, and the SA bus is driven by a combination of LBADR<19:2> and SA<1:0>; the two bits of overlap between LBADR<23:17> and LBADR<19:2> are driven identically.

SD Bus—A 16-bit data bus on the ISA bus.

Shared—One of the four states that a 32-byte cache block can have in the MESI cache-coherency protocol. *Shared* data is valid data that can only be read, not written.

Snoop—To compare an address on a bus with a tag in a cache, so as to detect operations that are inconsistent with cache coherency.

Snoop Hit—A snoop in which the compared data is found to be in a *modified* state. Compare *snoop miss.*

Snoop Miss—A snoop in which the compared data is not found, or is found to be in a *shared* state. Compare *snoop hit.*

Source—In timing diagrams, the left-hand column indicates the "Source" of each signal. This is the device or logic that output the signal. When signals are driven by multiple sources, all sources are shown, in the order in which they drive the signal. In some timing diagrams, bus signals take on different names as outputs cross buses through transceivers or are logical ORed in group-signal logic. In these cases, the signal source is shown with a subscript, where the subscript indicates the device or logic that originally caused the change in the signal.

Stale—Same as *modified.*

System Bus—A bus to which the NexBus interfaces. The NxVL chip supports two system buses, VL bus and ISA bus.

System Controller—The device or logic that provides NexBus arbitration and interfacing to main memory and any other buses in the system. The NxVL chip is a system controller.

Turbo Mode—An NxVL mode enabled by bit 2 of Port 92. Compare *non-turbo mode.*

VESA—Video Electronics Standards Association. The organization that specifies the VL bus.

VL-Bus—A 32-bit bus architecture derived from the Intel i386 and i486 Processor Bus and specified by the Video Electronics Standards Association (VESA).

Word—An two-byte (16-bit) unit of data.

XA Bus—A 16-bit address bus on the ISA bus.

XD Bus—An 8-bit data bus on the ISA bus.

Index