Fine Tuning the ALU Carry Path

National Semiconductor
Application Note 230
Harry Holt
May 1979

Most applications information for the IDM2902 Look-Ahead Carry Generator Family show three standard connections for 16-, 32-, and 64-bit Arithmetic Logic Units (ALUs). The three methods are shown in Figure 1.

With ALU cycle times in the 200 ns area, the standard connections shown in Figure 1 were quite adequate. A 5 to 10 ns overall savings did not warrant the time spent to examine alternative look-ahead carry methods. However, with the introduction in 1978 of the IDM2901A-1, cycle times began to approach 100 ns. This was further reduced to less than 80 ns (for a 16-bit ALU) with the introduction of the IDM2901A-2 in 1979. Now, obviously, a 5 to 10 ns savings is significant and well worth a new look at look-ahead carry techniques. The purpose of this application note is to do just that and, as will be shown, some of the results do not favor the standard approaches.

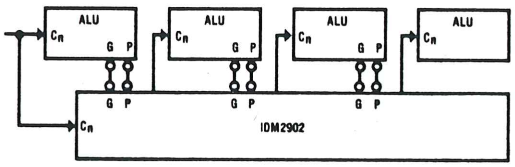


Figure 1(a). Conventional 16-Bit

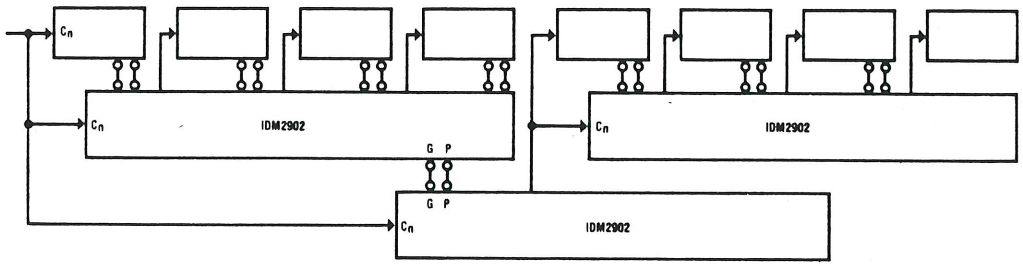


Figure 1(b). Conventional 32-Bit

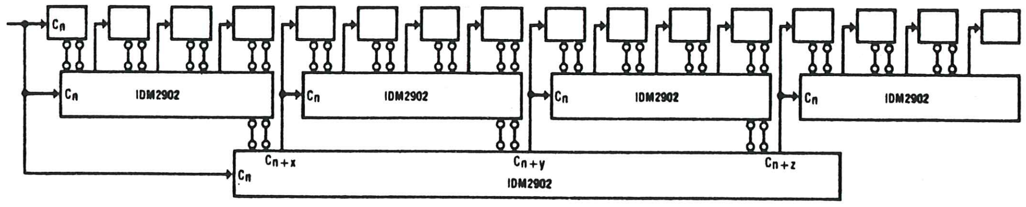


Figure 1(c). Conventional 64-Bit

BASIC METHODS

The basic methods examined in this
can be divided into four categories:

1. Ripple Carry
2. Conventional Single Level
3. Multi-Level
4. Shifted

Ripple carry is generally considered to
bits it turns out to be the fastest metho
shown that ripple carry can be used in c
other methods to eliminate parts while
to system cycle time. In most cas
methods will result in tradeoffs betw
and system speed. It will be shown that
however, generate the highest perfor
fewest parts!

Single-level will be used to describe
single layer of look-ahead carry even th
this is a multi-level solution since the
across four bits. An example of a singl
is the 16-bit solution of Figure 1(a).

The conventional 32-bit connection [F
example of multi-level look-ahead. In th
carry-in is connected to two IDM2902s.

The least well known of the four metho
approach shown in Figure 2. Although
slightly slower method in the 16-bit
there are word sizes where it can be the
Furthermore, freeing a set of G, P pins
can have advantages in certain applicat
extension is required. (See Reference

As alluded to above, the various m
combined in a number of ways. The foll
the various combinations that were e
study. Almost all were applied to ALU s
bits to identify the advantages and d
each. These will be summarized later.

Table I. Look-Ahead Carry Me

1. Ripple
2. C_n
3. Shifted
4. Chained
5. Shifted Chain
6. Two Level
7. Two Level with Helpers
8. Shifted Two Level
9. Shifted Two Level with Help
10. Double Shifted Two Level
11. Double Shifted Two Level wi
12. Three Level
13. Shifted Three Level
14. Double Shifted Three Level

BASIC METHODS

The basic methods examined in this application note can be divided into four categories:

1. Ripple Carry
2. Conventional Single Level
3. Multi-Level
4. Shifted

Ripple carry is generally considered to be slow, but at 8 bits it turns out to be the fastest method. Also, it will be shown that ripple carry can be used in combination with other methods to eliminate parts while adding very little to system cycle time. In most cases, the various methods will result in tradeoffs between parts count and system speed. It will be shown that some solutions, however, generate the highest performance with the *fewest* parts!

Single-level will be used to describe a system with a single layer of look-ahead carry even though technically this is a multi-level solution since the ALU itself looks across four bits. An example of a single-level approach is the 16-bit solution of Figure 1(a).

The conventional 32-bit connection [Figure 1(b)] is an example of multi-level look-ahead. In this approach, the carry-in is connected to two IDM2902s.

The least well known of the four methods is the shifted approach shown in Figure 2. Although this results in a slightly slower method in the 16-bit solution shown, there are word sizes where it can be the fastest method. Furthermore, freeing a set of G, P pins on the IDM2902 can have advantages in certain applications where sign-extension is required. (See Reference 1.)

As alluded to above, the various methods can be combined in a number of ways. The following is a list of the various combinations that were examined in this study. Almost all were applied to ALU sizes from 4 to 64 bits to identify the advantages and disadvantages of each. These will be summarized later.

Table I. Look-Ahead Carry Methods

1. Ripple
2. C_n
3. Shifted
4. Chained
5. Shifted Chain
6. Two Level
7. Two Level with Helpers
8. Shifted Two Level
9. Shifted Two Level with Helpers
10. Double Shifted Two Level
11. Double Shifted Two Level with Helpers
12. Three Level
13. Shifted Three Level
14. Double Shifted Three Level

FACTORS AFFECTING CHOICE

Before applying the look-ahead carry methods to the various word length ALUs, it may be worthwhile to look at some of the factors — other than raw speed — that could affect the choice of method. Some of these are:

1. parts count
2. board-to-board considerations
3. board space
4. sign extend
5. sequencer cycle time
6. board layout
7. word length expansion
8. different system architectures
9. current spiking

While parts count, board space, and board layout are more or less obvious considerations, the others deserve a brief comment:

A. Board-to-board considerations refer to those systems where half of the ALU is on one board and half is on another. Obviously all methods would not be readily adaptable to this situation if a sufficient number of connector pins is not available.

B. Sign-extend requirements may favor the method that frees a G,P input on one of the look-ahead carry circuits. This is explained more fully in Reference 1.

C. Sequencer cycle time, in a pipelined system, may be the limiting factor in overall system speed. Thus, saving a few nanoseconds in the ALU may not be worthwhile.

D. Future word length expansion is a consideration if several models of the same basic system are required. For example, 16 bits of address can address 64K words; twenty bits can address 1M words. If the ALU is used to compute addresses, the carry method optimized for 20 bits may be desirable.

E. The architecture that was assumed for this study will not be used in every system. Thus, the availability and timing of input signals, worst-case delay paths, and added components will affect the results shown in the following section. Thus, each design could require a separate study to achieve optimized results.

F. Current spiking is a consideration when one method causes several ALUs to change output states within a few nanoseconds of each other. If this causes system noise problems, perhaps an alternate method would be desirable.

APPLYING THE VARIOUS METHODS

In the following discussion, the IDM2901A-1 timing is used for the register-ALU elements. Because several different choices of pipeline register are available, the times shown do *not* include the clock-to-register output delay. Finally, the comparisons are based on the time required to add two registers and obtain a valid output, i.e., $A + B \rightarrow Y$.

4, 8 Bits: Ripple carry is clearly the best from all considerations and thus no further discussion is necessary. Register-to-register add time for 8 bits is 75ns.

12 Bits: At present, the conventional single-level method is best (77ns). However, if future bit-slices feature A, B → C_{n+4} as fast as A, B → $\overline{G}, \overline{P}$ and C_n → C_{n+4} as fast as the IDM2902's $\overline{G}$, $\overline{P}$ – C_{n+y}, ripple carry can be just as fast. (This illustrates the need for designers to continually rethink the problem as new parts become available.)

16 Bits: Without considering sign extend, the conventional approach [Figure 1(a)] is optimum (75ns). The shifted method (Figure 2) is 8.5ns slower under the assumptions made above, but if sign extend is required, it may well be as fast, in addition to eliminating multiplexers. (See Reference 1.) (This illustrates the fact that two parts of a system optimized independently may result in an overall slower system.)

20 Bits: As word width increases above 16 bits, some of the less conventional approaches begin to have some advantage. First, consider the more obvious approaches; the single level and chained approaches are shown in Figures 3(a) and 3(b). Another solution can be obtained by deleting parts from the conventional 32-bit solution of Figure 1(b). This is shown in Figure 3(c).

Note: In the figures that follow, connecting lines are simplified and terminal labels are eliminated for clarity.

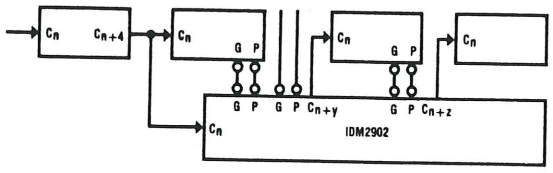


Figure 2. 16-Bit Shifted Look-Ahead

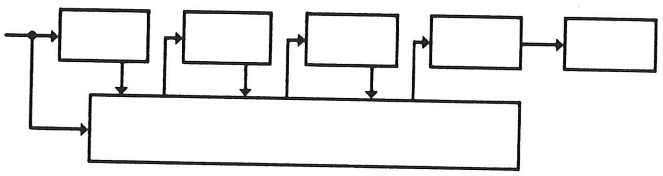


Figure 3(a). 20-Bit, Single-Level Method

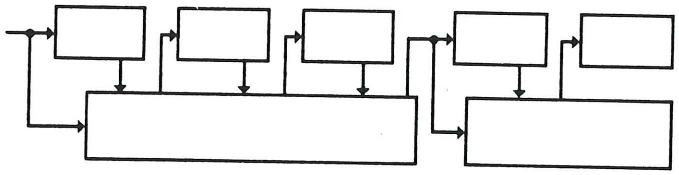


Figure 3(b). 20-Bit, Chained

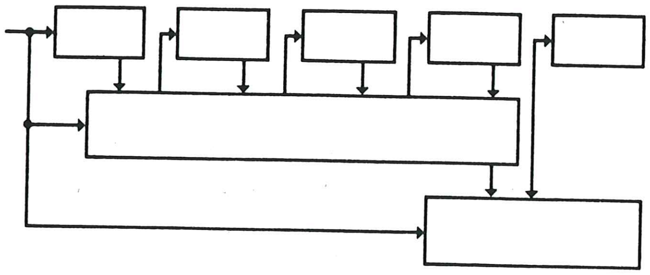


Figure 3(c). 20-Bit, Two-Level

From a timing standpoint, (b) and
87.5ns compared to 93ns fo
method [Figure 3(a)] is superio
standpoint, requiring a single l
than two. A closer look at Figur
that only a small portion of the s
circuit is used. Furthermore,
replaced by a circuit consisting c
of a 74S51 as shown in Figure 4
power, lower cost solution, th
second look-ahead carry circuit

Even more surprising is the fact t
shown in Figure 5 not only has
also runs faster (85.5ns) than the
Here is a situation where the
BOTH favor the same solution!

24, 28 Bits: Using the convent
(deleting one or two ALUs) yields
for both 24 and 28 bits. The sh
however, uses fewer parts and is
same parts count, the chained a
yield the fastest times (87.5ns) f
ALUs.

32 Bits: The conventional approa
Figure 1(b) is an example of the
method. For 32 bits, the register-
98ns for this method. This is a fas
chained and two-level methods
optimum for 24 and 28 bits. Ar

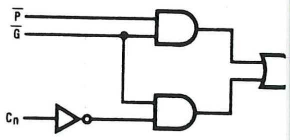
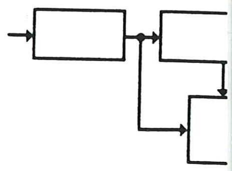
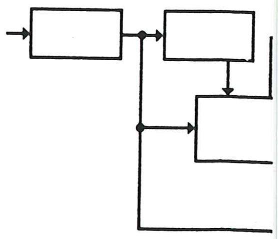


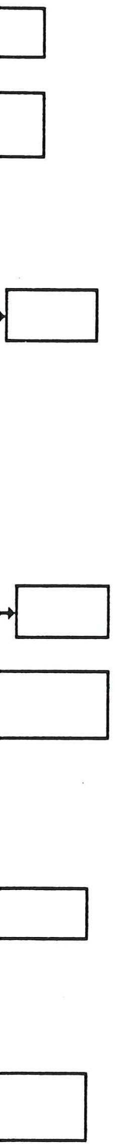
Figure 4. Partial Look-Ahead Ca





st, in addition to eliminating multi-
nce 1.) (This illustrates the fact that
tem optimized independently may
slower system.)

th increases above 16 bits, some of
al approaches begin to have some
consider the more obvious
gle level and chained approaches
s 3(a) and 3(b). Another solution can
eting parts from the conventional
gure 1(b). This is shown in Figure



From a timing standpoint, (b) and (c) of Figure 4 are both 87.5 ns compared to 93 ns for (a). The single-level method [Figure 3(a)] is superior from a parts count standpoint, requiring a single look-ahead carry rather than two. A closer look at Figure 3(c), however, reveals that only a small portion of the second look-ahead carry circuit is used. Furthermore, this portion can be replaced by a circuit consisting of 1/6 of a 74S04 and 1/2 of a 74S51 as shown in Figure 4. In addition to a lower power, lower cost solution, the replacement of the second look-ahead carry circuit actually saves 1.5 ns!

Even more surprising is the fact that the shifted method shown in Figure 5 not only has the fewest parts, but also runs faster (85.5 ns) than the other methods shown. Here is a situation where the speed-cost tradeoffs BOTH favor the same solution!

24, 28 Bits: Using the conventional 32-bit solution (deleting one or two ALUs) yields identical times (98 ns) for both 24 and 28 bits. The shifted chain (Figure 6), however, uses fewer parts and is faster (96 ns). With the same parts count, the chained and two-level methods yield the fastest times (87.5 ns) for both 24- and 28-bit ALUs.

32 Bits: The conventional approach for 32 bits shown in Figure 1(b) is an example of the two-level with helpers method. For 32 bits, the register-to-register add time is 98 ns for this method. This is a faster approach than the chained and two-level methods (103.5 ns) that were optimum for 24 and 28 bits. Another method — the shifted two-level — again uses fewer parts and is considerably faster than the conventional approach (87.5 ns). This is illustrated in Figure 7.

36, 40, 44 Bits: A 98 ns solution can be obtained for 36-bit ALUs by simply deleting parts from the 64-bit solution of Figure 1(c). A word of caution: this is *not* the path to the most significant slice (MSS). It turns out that this path is only 87.5 ns. The 98 ns path is to the output of the second MSS. The shifted two-level with helper method (Figure 8) will also produce a 98 ns result, but if the "helper" is replaced by the circuit of Figure 4, the result is 96.5 ns.

Another solution, requiring only two parts, is shown in Figure 9. This solution — the double shifted two-level method — turns out to be the best two-part solution for all word sizes from 36 to 64 bits. Speed for this method is 101.5 ns.

The shifted two-level with helper and double-shifted two-level methods turn out to be the optimum three-part and two-part solutions for 40- and 44-bit ALUs also.

48 Bits: As mentioned above, the double shifted two-level method shown in Figure 9 is also the optimum two-part solution for 48 bits. Three-part solutions are shown in Figure 10. The shifted three-level solution of Figure 10(a) results in a 114 ns system. The double shifted three-level solution is 101.5 ns. Note how the worst-case path varies between the two solutions. This points out the fact that several paths must be evaluated to ensure that the longest one has been found.

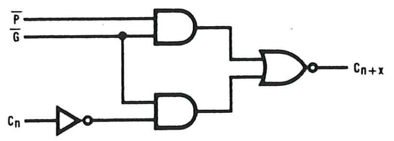


Figure 4. Partial Look-Ahead Carry Circuit

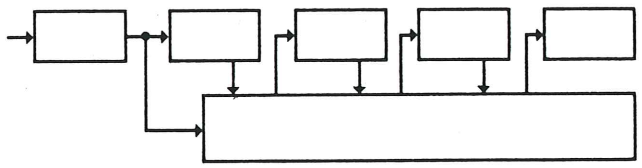


Figure 5. 20-Bit, Shifted

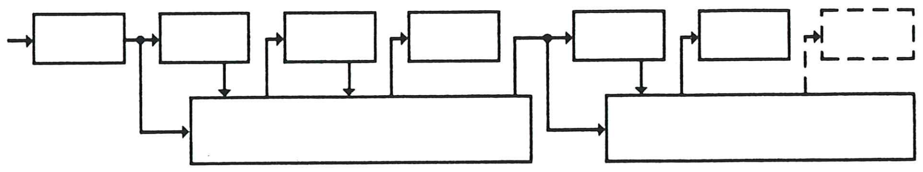


Figure 6. 24-, 28-Bit, Shifted Chain

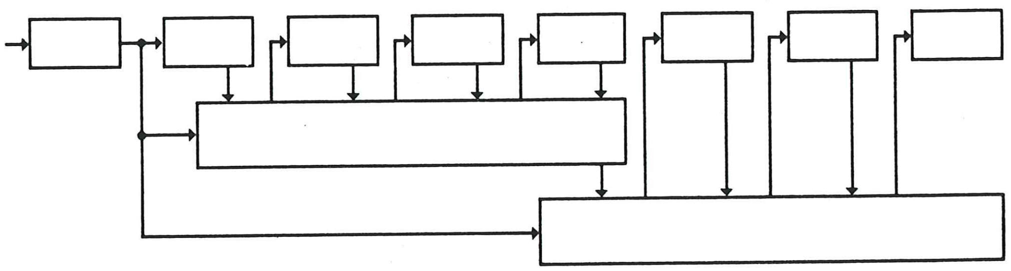


Figure 7. 32-Bit, Shifted Two-Level

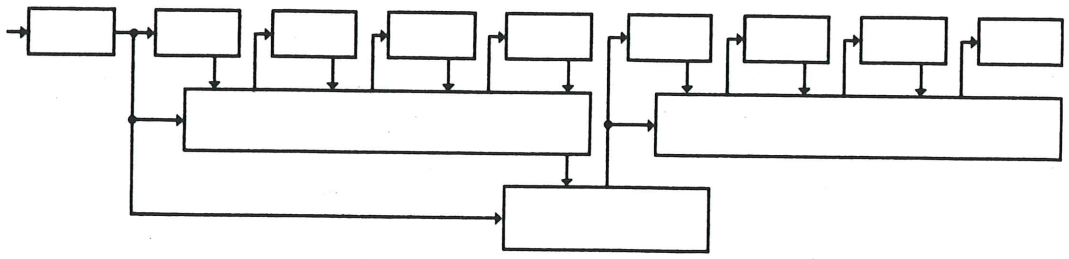


Figure 8. 36-Bit, Shifted Two-Level with Helper

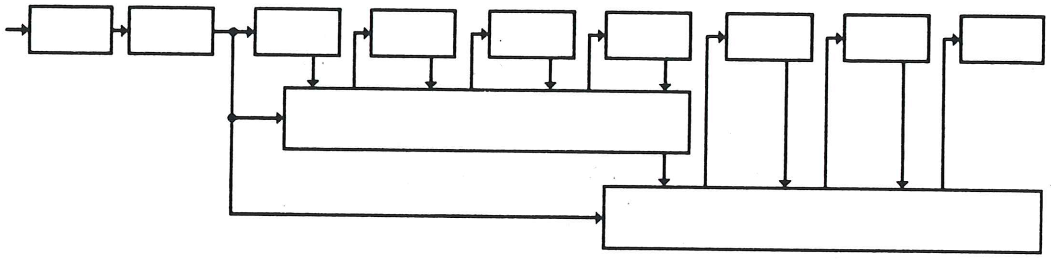


Figure 9. 36-Bit, Double-Shifted, Two-Level

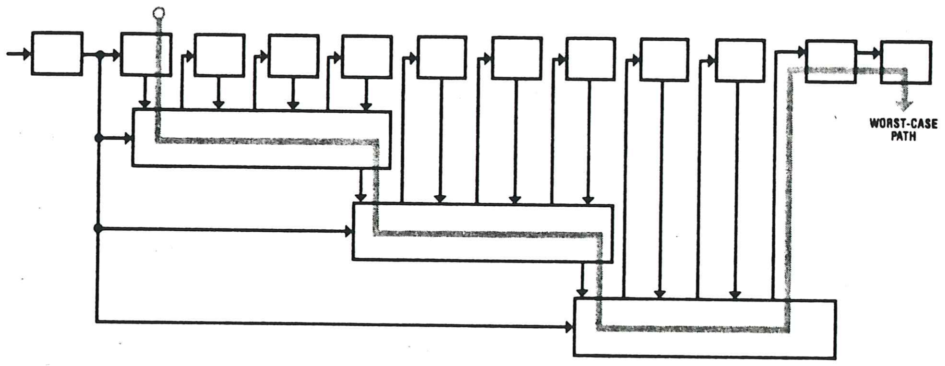


Figure 10(a). 48-Bit, Shifted, Three-Level

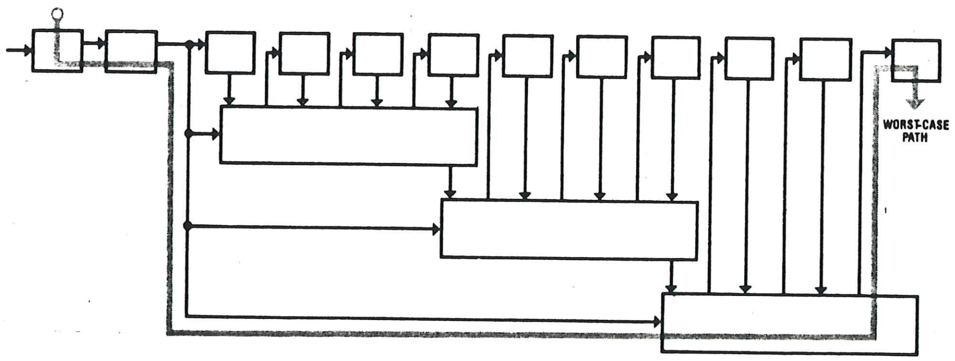
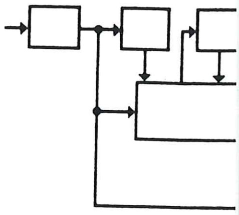


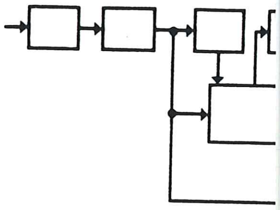
Figure 10(b). 48-Bit, Double Shifted, Three-Level

The four-part system of Figure
mance slightly (98ns) and may
additional expense. It is, howeve
choice from 48 to 60 bits. This me
shifted two-level with helpers.

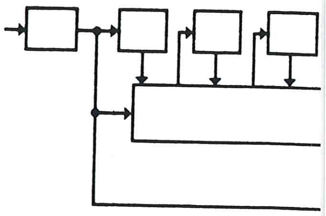
52 Bits: Not a particularly popula
system nevertheless provides an c
strate another look-ahead carry m
shifted, two-level with helpers. Thi
the fastest three-part method for w
64 bits. This method, illustrated in
117.5ns for 52-bit systems.

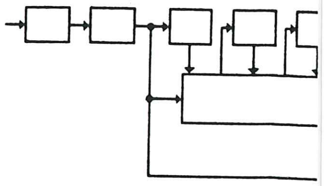
56 Bits: ALUs of 56 bits are be
floating point systems using 56 bi



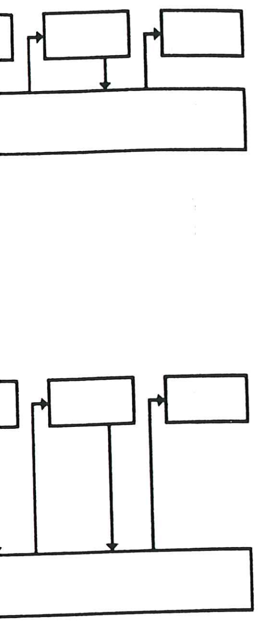


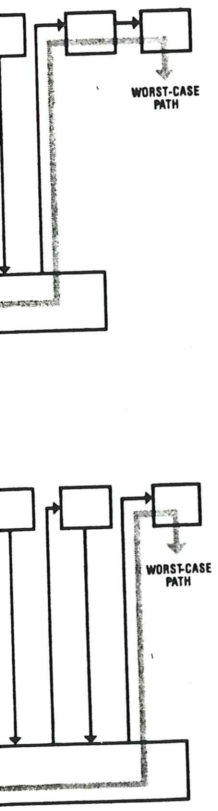
F





F





The four-part system of Figure 11 improves performance slightly (98 ns) and may not be worth the additional expense. It is, however, the best four-part choice from 48 to 60 bits. This method is referred to as shifted two-level with helpers.

52 Bits: Not a particularly popular ALU size, the 52-bit system nevertheless provides an opportunity to demonstrate another look-ahead carry method — the double-shifted, two-level with helpers. This also turns out to be the fastest three-part method for word widths from 52 to 64 bits. This method, illustrated in Figure 12, results in 117.5 ns for 52-bit systems.

56 Bits: ALUs of 56 bits are becoming common in floating point systems using 56 bits of mantissa and 8 bits of exponent. Deleting components from the conventional 64-bit approach [Figure 1(c)] results in a 98 ns solution. This same speed, however, can be achieved with four parts using the shifted two-level with helper method of Figure 13.

60 Bits: The fastest 60-bit solution is the conventional approach for 64 bits [Figure 1(c)] with one alu deleted. This results in a 98 ns solution. The shifted two-level with helper method (Figures 11 and 13) is a four-part solution that results in 103.5 ns.

64 Bits: Again, the fastest 64-bit solution (98 ns) is the conventional approach of Figure 1(c). The fastest four-part solution is a double-shifted two-level with helper method (117.5 ns) illustrated in Figure 14.

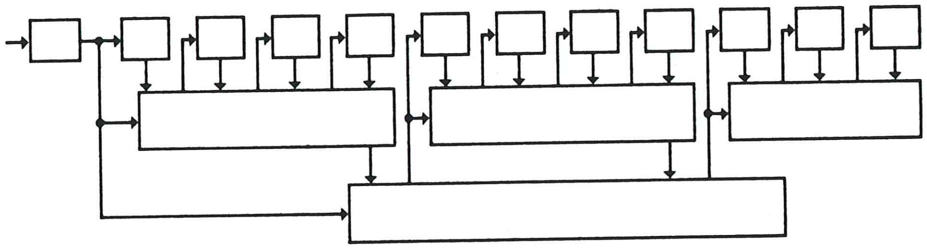


Figure 11. 48-Bit, Shifted Two-Level with Helpers

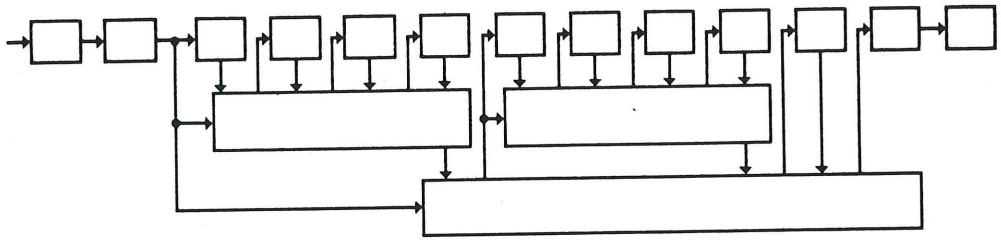


Figure 12. 52-Bit, Double-Shifted, Two-Level with Helpers

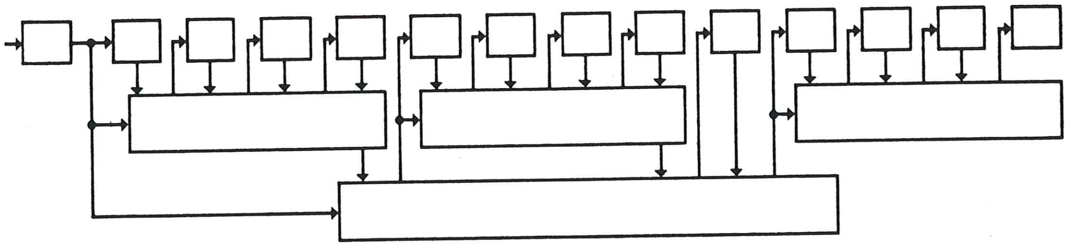


Figure 13. 56-Bit, Shifted, Two-Level with Helper

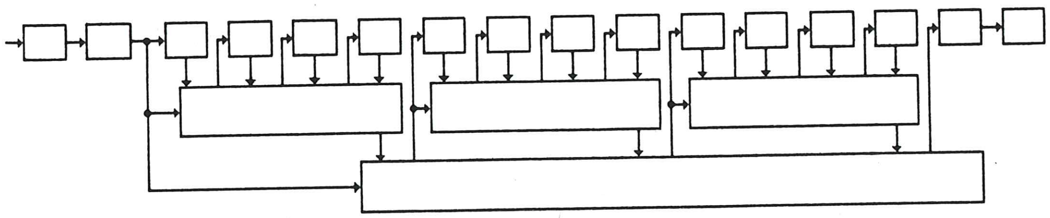


Figure 14. 64-Bit, Double Shifted, Two-Level with Helper

Table II is a summary of the data generated from this study. It lists, for each word size, the fastest solution for look-ahead carry parts count from 0 to 5. The number under the "Method" column corresponds to the numbered list of Table I. The fastest solution for each word size is shaded.

Table II. Optimum Speed for 4–64-Bit ALUs

	Number of IDM2902s											
	0		1		2		3		4		5	
Word Size	Method	Time	Method	Time	Method	Time	Method	Time	Method	Time	Method	Time
4	1	50										
8	1	75										
12	1	91	2	77								
16	1	107	2	77								
20	1	123	3	85.5	4	87.5						
24	1	139	3	101.5	4, 6, 8	87.5	7	98				
28	1	155	3	117.5	4, 6, 8	87.5	9	98				
32	1	171	3	133.5	8	87.5	9	98				
36	1	187	3	149.5	10	101.5	7, 9, 12	98				
40	1	203	3	165.5	10	117.5	4, 7, 9, 12	98				
44	1	219	3	181.5	10	133.5	9, 13	98				
48	1	235	3	197.5	10	149.5	14	101.5	7, 9	98		
52	1	251	3	213.5	10	165.5	11	117.5	7, 9	98		
56	1	267	3	229.5	10	181.5	11	133.5	9, 13	98	7	98
60	1	283	3	245.5	10	197.5	11	149.5	9	103.5	7, 9	98
64	1	299	3	261.5	10	213.5	11	165.5	11	117.5	7, 9	98

CONCLUSIONS

It may appear that a lot of time was spent investigating alternate look-ahead carry schemes that save "a few nanoseconds" in overall speed. While this is certainly true for systems with ALU cycle times in the 200 ns range, it has been shown that with the more recent 2900 components from National Semiconductor, ALU cycle times in the 100 ns region are certainly feasible, and here those same few nanoseconds could become significant. This will be even more apparent with the introduction of the IDM2901A-2, which will improve the register/ALU times listed in Table III by another 20–25%.

It has also been shown that no one solution is "best" for all applications. Even the "fastest" solution may not be optimum for a specific system when parts count, system wiring, board space, etc., are considered.

Table III. Delay Times Used to Calculate Cycle Time

IDM2901A-1:		IDM2902:	
AB → Y	50 ns	$\overline{G}, \overline{P} \rightarrow \overline{G}, \overline{P}$	10.5 ns
AB → $\overline{G}, \overline{P}$	45 ns	$C_n \rightarrow C_{n+x,y,z}$	10.5 ns
AB → C_{n+4}	50 ns	$\overline{G}, \overline{P} \rightarrow C_{n+x,y,z}$	7.0 ns
$C_n \rightarrow C_{n+4}$	16 ns		
$C_n \rightarrow Y$	25 ns		

Finally, the entire study will soon be obsolete as new components with different (albeit faster) specifications are introduced. Therefore, the only conclusion that seems legitimate is that each application should be considered individually with the requirements of the system, the devices available, and sound engineering judgement determining the optimum solution. It is hoped that the information contained in this application note will provide some guidelines for finding that solution.

REFERENCES

1. AN-203, Bit-Slice Microprocessor Design Takes a Giant Step Forward with "Schottky-Coupled-Logic" Circuits.

2. AN-217, High Speed Bit-Slice Microsequencing Design.