Circuit Design Techniques for a Floating-Point Processor

Timothy Hu

University of California, Berkeley
Department of Electrical Engineering
and Computer Science

ABSTRACT

This report presents some novel circuit design techniques used in the data-path of the SPUR floating-point unit. Three most interesting circuit blocks are discussed which include a fast adder, a leading one detector and a shifter. Mixed logic with static and dynamic circuits are used. The chip is implemented in a 1.6 micron, N-well, double-metal CMOS process (HP CMOS40).

The timing and area of the above three modules are as follows:

66 bit adder

- Delay - Crystal 36 ns, SPICE 33 ns
- Size - 4757 × 553 lambda, which is 3806 × 442 μm.

67 bit Leading one detector

- Delay - Crystal 20.5 ns, SPICE 18 ns
- Size - 4901 × 463 lambda, which is 3920 × 320 μm.

67 bit shifter with Sticky Logic

- Delay - Shifting 15 ns, Sticky bit (latched into output latch) 25 ns
- Size - The whole module with decoder is 5359 × 1414 lambda, which is 4287 × 1131 μm.

September 23, 1987

ACKNOWLEDGMENT

I am very grateful to Professor David A. Hodges, my research advisor for the opportunity to participate in the SPUR project. His technical and financial support are largely responsible for the success of this project. I would also like to thank Professor Paul R. Gray, the second reviewer of this report for his patience and support.

I am in debted to Greg Uehara and B. K. Bose for their time and effort on proofreading this report. A special thanks has to go to Bose whom I have benefitted greatly from his personal advise, encouragement, and support throughout this project.

Finally, I would like to thank my parents, Hung-Nye and Annie, for their endless encouragement and patience. I wish them health and happiness for the rest of their years together.

Table of Contents

1. Introduction

This report presents some novel circuit design techniques which may find application in a *Floating-Point Unit* (FPU). Examples are taken from the design of a VLSI FPU for a LISP-based multi-processor system - *Symbolic Processing Using RISCs* (SPUR) [Hill86]. The FPU implements the IEEE Floating-point Standard P-754 for the essential operations, supporting single, double and extended precision operands. A four-phase clocking scheme is used, with a cycle time of 140ns (25 ns phase time and 10 ns guard time between phases). The chip is being designed in a 1.6 micron, N-well, double-metal CMOS process. *Mixed Logic* with *static* and *dynamic* circuits are employed.

Following a brief description of the SPUR FPU chip is a discussion of three most interesting circuit blocks which differ from conventional processor design approaches. The interesting examples included in this report are :

- Fast 66 bits Parallel Prefix Adder
- 67 bit Leading One Detector
- 67 bit Shifter with Sticky Logic

Essential features of the floating-point format and IEEE Floating-point Standard are provided for background information. This is given in order to outline the reasons for certain implementation decisions. For more detail, [1] and [5] are recommended.

1.1. Floating-Point Format

Floating-point numbers stored in a computer are often divided into two parts. The first part represents a fixed point number called the *fractional part* (and sometimes *mantissa* or *coefficient*). The second part designates the position of the radix point and is called the *exponent* (or *characteristic*). The two parts represent a number obtained from multiplying the mantissa(m) times a radix(r) raised to the value of the exponent(e), thus: $m \times r^e$. For binary arithmetic, the radix is equal to 2. The location of the radix point and the value of radix are assumed and are not stored in the computer.

A floating-point number is said to be *normalized* if the most significant position of the coefficient contains a nonzero digit. In this way, the coefficient has no leading zeros and contains the maximum possible number of significant digits. When two normalized mantissas are added, the sum may contain an overflow digit. An overflow can be corrected easily by shifting the sum once to the right and incrementing the exponent. When two numbers are subtracted, the result may contain most significant zeros. A floating-point number that has a zero in the most significant position of the mantissa is said to have an *underflow*. To normalize a number that contains an underflow, it is necessary to shift the mantissa to the left and decrement the exponent until a nonzero digit appears in the first position. In most computers, a normalization procedure is performed after each operation to ensure that all results are in a normalized form.

1.2. IEEE Floating-Point Standard

The major considerations for the standard are their *format*, *accuracy* and *exception handling*. There are two basic formats for the floating-point operands: 32 bits and 64 bits. Radix 2 was chosen in order to maintain a high degree of precision. The standard specifies extended-precision operations including intermediate results which have a range and precision greater than the basic format. To achieve this goal, the fraction has 64 bits and exponent has 15 bits with three additional bits are used. The three additional bits are a *guard bit*, a *round bit* and a *sticky bit*. They are placed in the immediate right of the lower-order fraction bit. The guard bit and the round bit are extra bits appended to the lower-order end. At the lowest-order is the sticky bit. The sticky bit is set to 1 and remains set if a 1 is shifted into it during alignment. They are used for directed rounding.

2. SPUR Floating-Point Unit

In this chapter, a brief description of the SPUR FPU is given [Lee86][Bose86]. This is the actual implementation of the IEEE floating-point standard in the SPUR FPU chip. A description of the data formats is given, followed by a description of the datapaths.

2.1. Data Formats

There are 3 different formats as shown in Table 1.

Format	exponent length (range)	significant length (precision)
single	8	24
double	11	53
extended	17	64

Table 1 SPUR FPU Data Formats

Extended-precision arithmetic is implemented in hardware. Since the internal register file holds data in the extended-precision format only, *single* and *double* formats are subjected to an implicit conversion to extended whenever they are loaded. Bearing in mind that for the operations in the datapath, there is a *sign bit*, a *17-bits exponent* and a *64-bits fraction*.

2.2. Datapaths

There are two datapaths for the FPU. One is the exponent datapath which handles the exponent part of the floating-point number and the other is the fraction datapath which handles the fraction part.

2.2.1. Exponent Datapath

The exponent datapath lies on the critical path of all arithmetic operations in the FPU. The results of each add, subtract, multiply and divide instructions must have its exponent adjusted for normalization at the end of the operation. A block diagram of the exponent datapath is shown in figure 1.

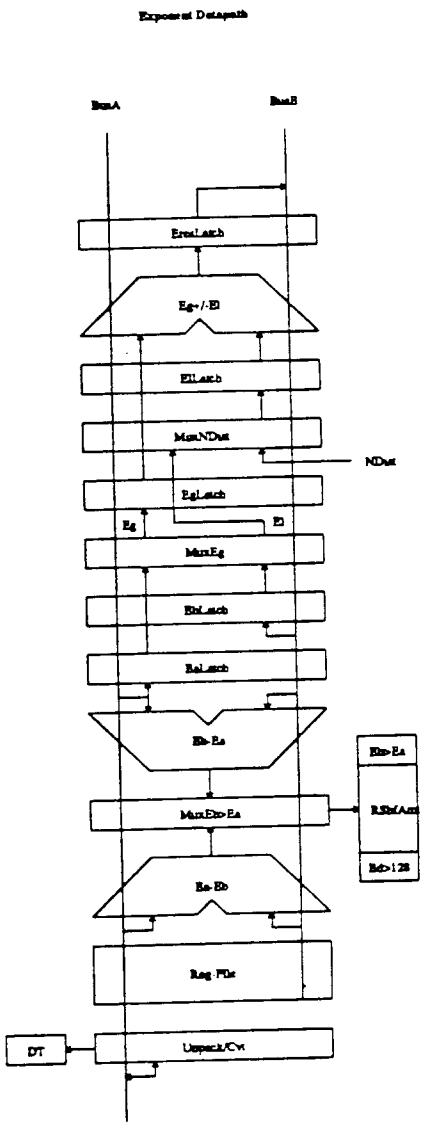


Figure 1 Block diagram for the exponent datapath

2.2.2. Fraction Datapath

The fraction datapath performs add and subtract operations on unsigned 65 bit numbers. Its use however is not limited to the implementation of the floating-point add and subtract instructions. In the multiply and divide instructions, the datapath is used to complement one of the operands and to gather the final result. It is used again during divides to compute the remainder of the division; this number is then used to find the quotient. A block diagram of the fraction datapath is shown in figure 2.

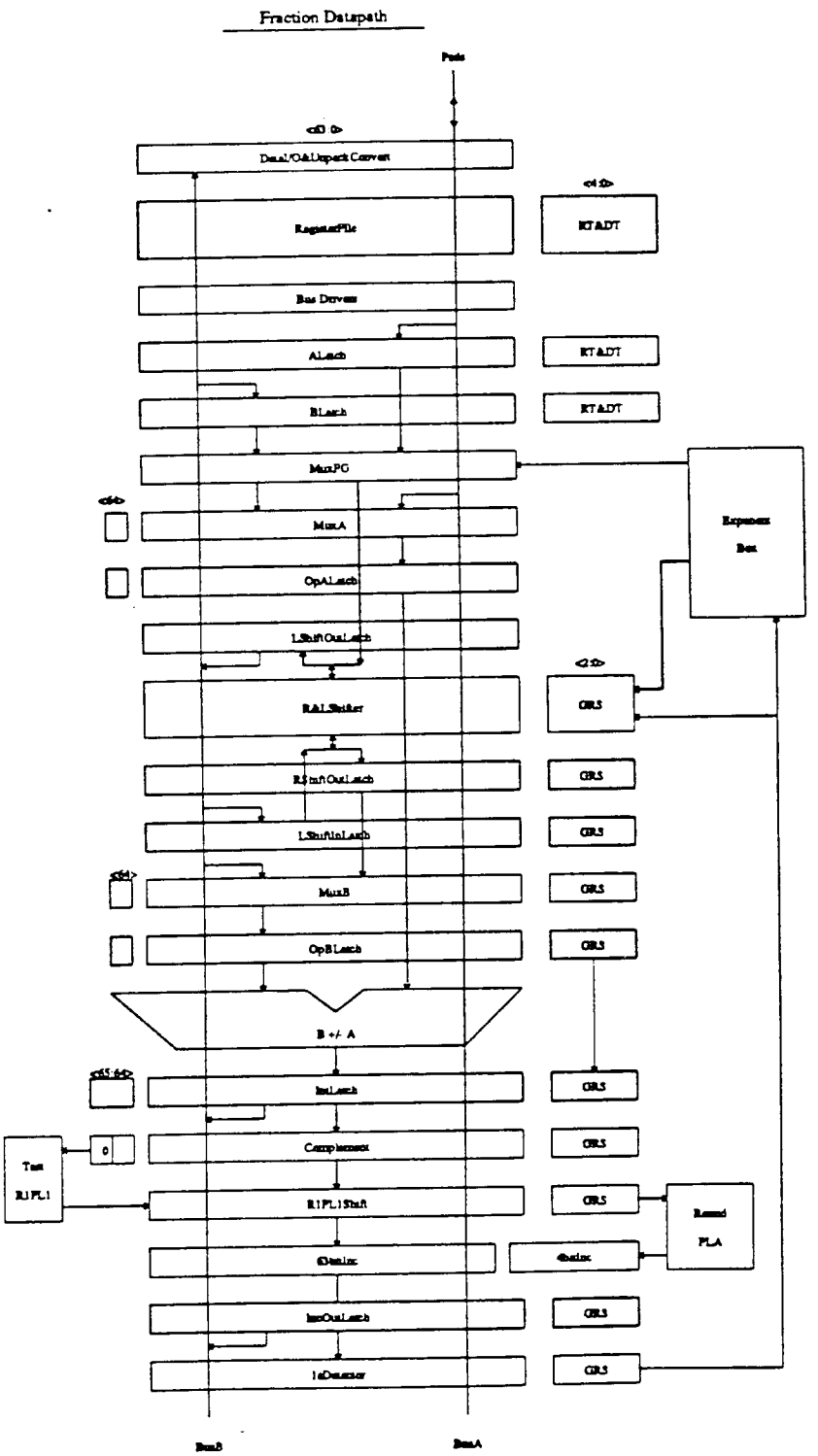


Figure 2 Block diagram for the fraction datapath

3. Adders

3.1. Introduction

Adder is the critical module of every micro-processor performing arithmetic. For a FPU, which is supposed to be a *number cruncher*, a fast adder is needed because it could be the element that determines the cycle time. It is needed in almost every computations done by the FPU. An adder can perform *addition*, *subtraction* (invert the input and set the C_{in}), *multiplication* (shift and add) and *division* (shift and subtract).

Besides *speed*, it must be able to handle a *large number of inputs*. The adder is needed in the exponential data path for computation of the exponent. The exponent seldom exceeds 20 bits. Therefore designing the adder for the exponential data path isn't too much a problem. But on the fractional datapath, the adder has to handle the mantissa which usually ranges from 32 to 64 bits.

Since the adder is used in VLSI FPU, regularity is also important for the purpose of layout using CAD tools. Traditional kinds of *TTL design* for which gate count and delay are performance measures are not sufficient to evaluate a design for possible implementation in VLSI circuits.

Different architectures for VLSI fast adder have been proposed [2][9]. Out of those, the Brent and Kung adder [BrK82] is a simple, fast and regular structured adder suitable to VLSI applications. The disadvantage is that the fan-out is limited to 2 and is too restrictive. Fan-out can be traded off for shorter interconnects and a smaller area, which may in fact result in a faster circuit.

For the SPUR FPU, these effects are considered in designing the VLSI adder [Wei85]. It is based on the idea of Ladner and Fischer's *parallel prefix computation (PPC)* [LaFi80]. A simple delay model of the building blocks are assumed. The associated signal delay is computed with drivers modeled explicitly. Using dynamic programming, the optimal adder configuration for 1 to 66 bit of inputs are calculated by Wei. This information is used to design the adder used in the SPUR FPU.

3.2. Design Specification

For the SPUR FPU supporting the IEEE Floating-point Standard several adders are needed. Three are in the exponent datapath and one is in the fractional datapath.

The one in the fractional datapath has the following requirements. There are two *65 bit inputs* and a C_{in} that produces one *65 bit output* and a C_{out}.

For the three in the exponential datapath, all have two *17 bit inputs* and a C_{in} and produce *17 bit outputs* and a C_{out}. Two of them are hard-wired to perform subtractions (with inverters in front to complement one input and C_{in} set to 1). One can be an *add-or-subtractor* (with XOR in front with the other inputs and C_{in} connected to one control line CE_{sub}. If CE_{sub} is 0 then the XOR are simple pass and C_{in} is 0 so that the whole thing is an adder. If CE_{sub} is 1 then XOR inverts all inputs and set Cin 1 so that the whole thing is a subtractor).

The timing requirement for the adder in the exponential datapath is one phase time (25 ns) and that of the adder in the fractional datapath is two phase time (50 ns) plus one guard time (10ns).

3.3. Logic Implementation - Parallel Prefix Computation for Adder

A brief introduction of parallel prefix computation (PPC) is reproduced in this section. For detail, please refer to [LaFi80] [3] [7].

A prefix computation is a computation where for each output position, the output depends only on its low-order inputs, but not the higher-order inputs.

Binary addition can be transform into a parallel computation by introducing an associative operator $\circ$. Define the carry generate terms, gIN_i, and the carry propagate term, pIN_i, for each bit position i ,then the carry c_i for each bit obeys the following recurrence relation:

$$gIN_i = a_i \cdot b_i \tag{3.1}$$

$$pIN_i = a_i \oplus b_i \tag{3.2}$$

$$c_i = G_i \quad \text{for } i = 1, 2, \cdots, n \tag{3.3}$$

where

$$(G_i , P_i) = \begin{cases} (gIN_1 , pIN_1) & \text{if } i=1 \\ (gIN_i , pIN_i) \circ (G_{i-1} , P_{i-1}) & \text{if } n \geq i > 1 \end{cases} \tag{3.4}$$

and $\circ$ is a concatenation operator defined as follows:

$$(g_l , p_l) \circ (g_r , p_r) = (g_l + p_l \cdot g_r , p_l \cdot p_r) \tag{3.5}$$

Note that $\circ$ is not commutative. Its left argument (g_l , p_l) is treated differently from its right argument (g_r , p_r). After the carry bit c_i is computed, the sum bit s_i is :

$$s_i = pIN_i \oplus c_{i-1} \quad \text{for } i = 2, \cdots , n \tag{3.6}$$

$$s_1 = p_1 \tag{3.7}$$

Given the fact that $\circ$ is associative, choose a m such that $i \geq m > 1$ and rewrite $(G_{i,1} , P_{i,1})$ as follows:

$$(G_{i,1} , P_{i,1}) = (G_{i,m} , P_{i,m}) \circ (G_{m-1,1} , P_{m-1,1}) \tag{3.8}$$

where

$$(G_{i,m} , P_{i,m}) = \begin{cases} (gIN_m , pIN_m) & \text{if } i=m \\ (gIN_i , pIN_i) \circ (G_{i-1,m} , P_{i-1,m}) & \text{if } i>m \end{cases} \tag{3.9}$$

Observe that $(G_{i,m} , P_{i,m})$ and $(G_{i-m+1,1} , P_{i-m+1,1})$ have similar functional forms. Both are functions of $i-m+1$ consecutive input bits and both require $i-m$ applications of the associative operator $\circ$. As a result, both can be computed by the same circuit.

3.4. Circuit and Layout Design

Fully static CMOS is used for the design. This is because one can make use of both the phase time (25 ns) and the dead time (10 ns) if the design is static. Though the requirement gives two phase time, the design is aimed to complete all computations in one phase time plus one dead time (35 ns)

Three blocks are used to implement the above PPC scheme for an adder (Figure 3).

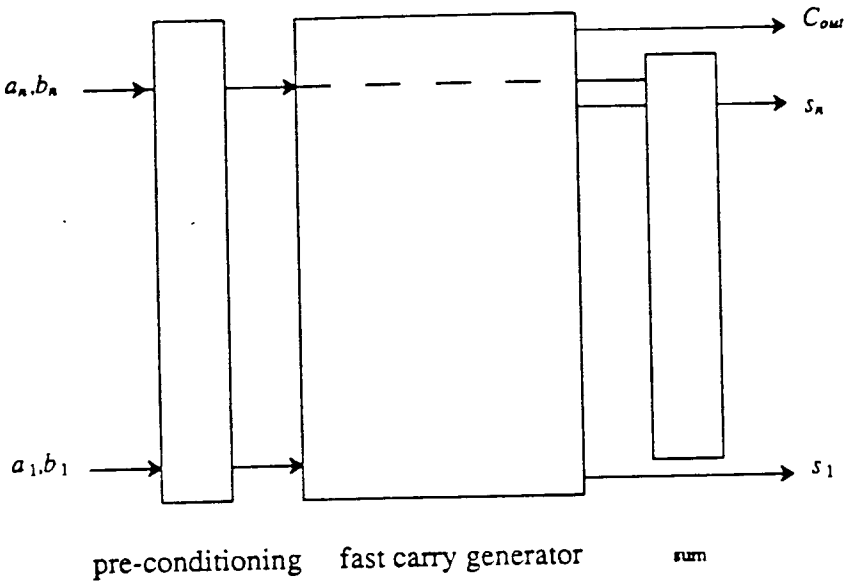


Figure 3 Floor Plan for the adder

The front end is a *pre-condition block*, the heart of the module is a *fast carry generator* and a *sum block* is appended at the end to give the results.

3.4.1. Pre-Conditioning

The pre-conditioning block gates in the inputs a_i and b_i to generate the initial pIN_i and gIN_i for each bit position i (3.1) (3.2). The computed pIN_i and gIN_i terms are then fed into the fast carry generator. A schematic diagram of the XOR gate used is shown in figure 4a.

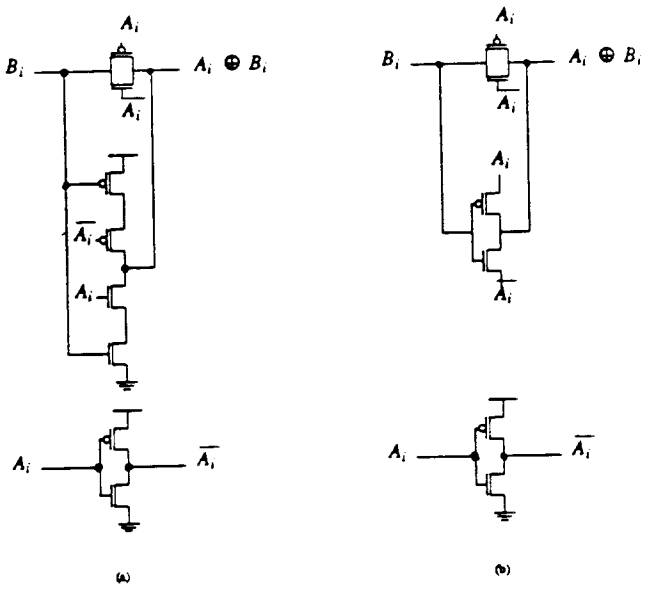


Figure 4 XOR schematics

The reason for using such configuration (which is not an optimal) over the one in figure 4b (optimal) is that the logic simulator used at the beginning of the project is Esim, and it failed to simulate the later circuit. All simulations and timing analysis for the adder had been done before MOSSIM was used. If time allowed, changes should be made to the XOR circuit used.

3.4.2. Fast Carry Generator

The fast carry generator performs the operations defined in (3.4) (3.5) which is the heart of the whole adder. Three basic type of cells are used to implement this block. They are the *black cells*, the *white cells* and the *driver cells* (figure 5).

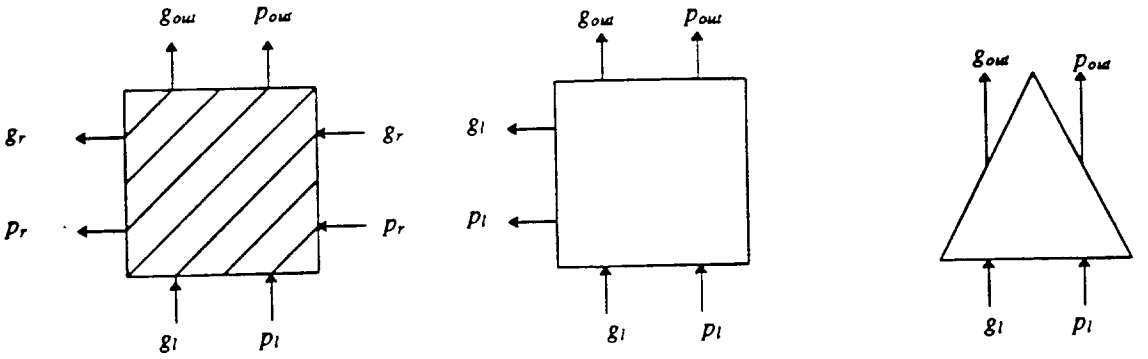


Figure 5 Three basic Types of Adder Cells

The are designed and layout in such a manner so that no routing have to be done. The is very useful because one can have template of the cells and use CAD tools to piece them up once the configuration of the matrix for the fast carry generator is obtained for any desired number of inputs.

The black cells performs the operations in (3.5). There are two types of black cells, a and b which are shown in figure 6.

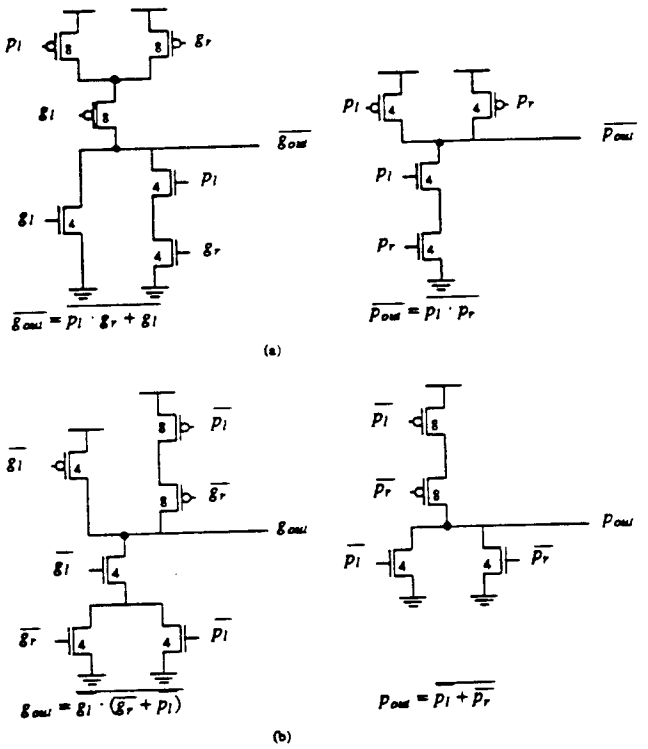
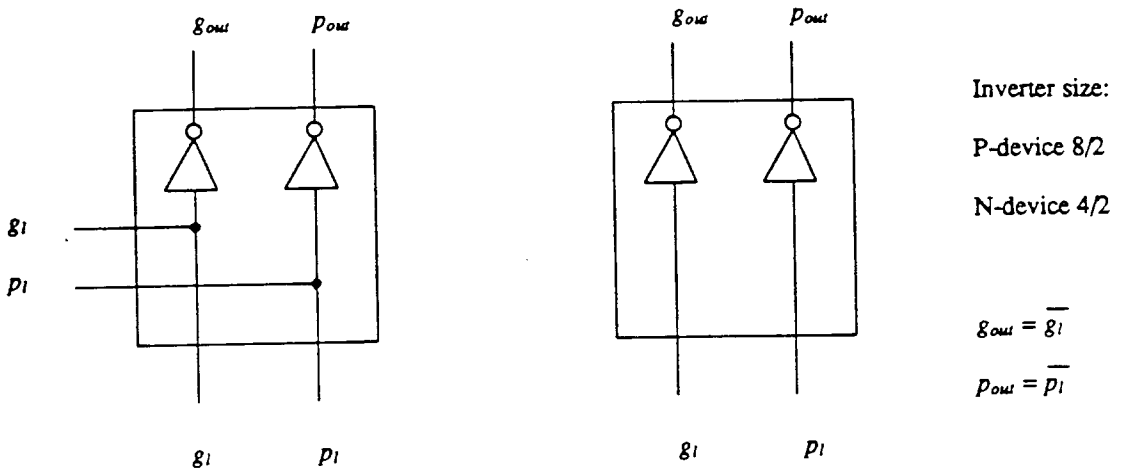


Figure 6 Black Cells Schematics

The *a* cells gates in active HIGH signals and produces active LOW outputs. The *b* cells gates in active LOW signals and produces active HIGH outputs. Each cell gives two outputs p_{out} and g_{out} which are the results of (3.5). The white cells are used to convert signals between active HIGH and active LOW inputs and outputs for the subsequent black cell operations. Again two types are available, *c* and *n* (figure 7).



(c) (n)

Figure 7 White Cells schematics

The *c* cells have connections that passes the input to subsequent cells and the *n* cells have no such connections.

The drivers cells are used when there is long wire connects or large fan-outs. The are ratioed to three sizes to optimize circuit performance.

The configuration of the matrix for the fast carry generator is shown in table 2. The configuration is for inputs from 1 to 66 bits with a constrain of only three buffer sizes are available.

n	n-m	m	driver stages	depth
1	0	0	0	0
2	0	0	0	1
3	1	2	0	2
4	2	2	0	2
5	3	2	1	3
6	4	2	1	3
7	5	2	2	4
8	5	3	1	4
9	6	3	1	4
10	7	3	2	5
11	7	4	2	5
12	8	4	2	5
13	9	4	2	5
14	10	4	3	6
15	10	5	2	6
16	11	5	2	6
17	12	5	2	6
18	13	5	2	6
19	14	5	3	7
20	15	5	3	7
21	15	6	3	7
22	16	6	3	7
23	17	6	3	7
24	18	6	3	7
25	19	6	2	8
26	19	7	3	8
27	20	7	3	8
28	21	7	3	8
29	22	7	3	8
30	23	7	3	8
31	24	7	3	8
32	24	8	3	8
33	24	9	3	8
34	27	7	2	9
35	29	6	3	9
36	30	6	3	9
37	27	10	3	9
38	28	10	3	9
39	29	10	3	9
40	30	10	3	9
41	31	10	3	9
42	32	10	3	9
43	33	10	3	9
44	35	9	3	10
45	36	9	3	10
46	37	9	3	10
47	38	9	3	10
48	39	9	3	10
49	40	9	3	10
50	41	9	3	10
51	42	9	3	10
52	43	9	3	10
53	39	14	3	10
54	44	10	3	11
55	44	11	3	11
56	44	12	3	11
57	44	13	3	11
58	45	13	3	11
59	46	13	3	11
60	47	13	3	11
61	48	13	3	11
62	49	13	3	11
63	50	13	3	11
64	51	13	3	11
65	52	13	3	11
66	57	9	3	12

Table 2 66 bit adder constrained to 3 buffer stages

The first column of the table indicates how many bits you want. The second and third columns indicates how the bits should be broken down. The fourth column indicates how many buffer stages are needed. The fifth column indicates how many levels of delay are required. The circuits are designed such

that each level of delay is about 2 ns.

A 5 bit example is given below to illustrate how to use the table. Search for the first column for n = 5, then $n-m$ is 3 and m is 2. This means that there should be 3 black cells and 2 white cells at the outermost level. The a cell should always be used at the outermost level. The c cell should be used to connect the black and white cells. Then recursively, the same table s used to construct the remaining levels. Figure 8 shows the configuration.

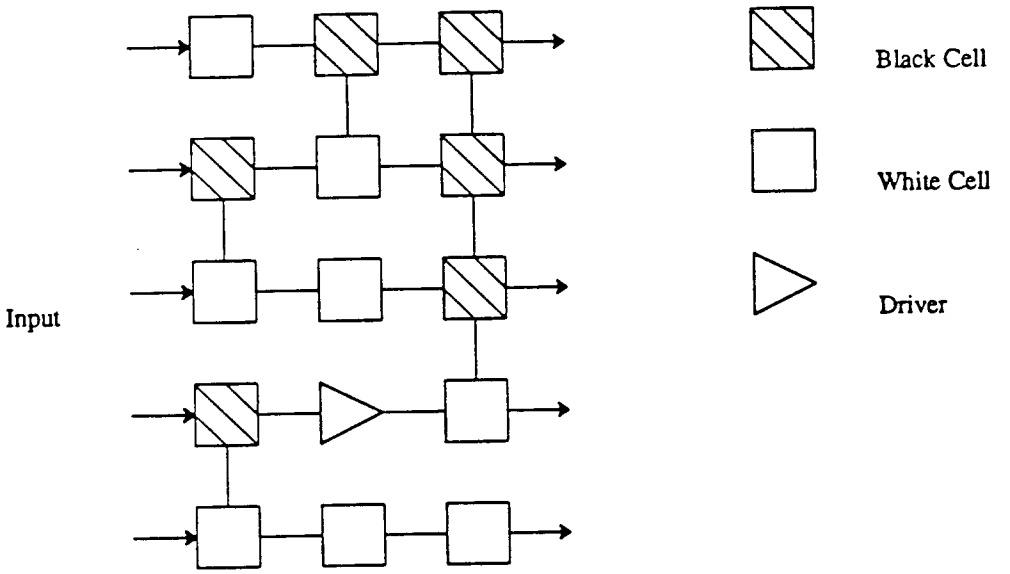


Figure 8 An example for 5 bit fast carry generator

A program was written to generate the fast carry generator based on the table given. It reads the table and gives the configuration as output. A sample run is as follows:

```
% ppc 5 > add5
Computed 1,2 - Depth L=1, R=1
Computed 3,2 - Depth L=2, R=1
PPC Carry Propagate Width = 5, Depth = 3
% cat add5
(O)N(O)B(O)A
(O)A(O)C(O)A
(O)C(O)N(O)A
(O)A(O)1(O)C
(O)C(O)N(O)N
%
```

The (O) is for spacing informations so that once a template file with all the cells in it is available, the output can be piped directly to another program called *Mquilt* written by Bob Mayo to generate the whole layout automatically.

3.4.3. Sum Bit

The last block is just an array of XOR implementing (3.6). Note that because of (3.7), all that is needed is $n-1$ such cells for a n bit adder.

3.4.4. Adders

The delay of the whole adder can be estimated quickly by adding the following:

$$delay\ of\ pre-condition + delay\ of\ each\ level \times number\ of\ level + delay\ of\ sum\ bit$$

For a 66 bit adder, the delay is approximately

$$5\ ns + 2\ ns \times 12 + 5\ ns = 34\ ns$$

For a 17 bit add-or-subtractor (XOR plus 18 bit adder), the delay is approximately

$$5\,ns + 5\,ns + 2\,ns \times 5 + 5\,ns = 25\,ns$$

Optimization of the XOR gates are necessary if time is allowed.

3.5. Simulation results

It is not easy to determine the worst case path by inspection because it is optimized to give equal delay to every input. Crystal is used to find the worst case path and SPICE is used to simulate the time delay.

The delays computed by Crystal should be larger than those computed by SPICE. The difference comes from the fact that Crystal is too 'honest'. When a CMOS passgate is used, Crystal will use the worst case path neglecting the fact that both gates are open at the same time. For the 66 bit adder, Crystal gives a delay of 36 ns while SPICE gives 33 ns (adding the passgate of the opposite type back to the SPICE deck) . For the 17 bit add-or-subtractor, Crystal gives 22 ns while SPICE gives 24ns (without modification).

3.6. Summary

A fast adder suitable for VLSI application has been designed. The larger the number of inputs, the more efficient this adder is compared with other type of adders. This is because an additional level only costs 2 ns and can handle on the average 10 more bits (if the number of inputs is larger than 30). There is a fixed cost of the two XOR to implement the pre-conditioning and sum blocks. If the number of inputs is small, this adder may not be as efficient as other type of adders. For a FPU where the inputs to the adder ranges from 30 to 80 bits, this type of adder is a better choice over the others.

Characteristics:

66 bit adder

- Inputs - 131 bits , two 65 bit data and one C_{in}
- Output - 66 bits, 65 bit results and one C_{out}
- Delay - Crystal 36 ns, SPICE 33 ns
- Size - 4757 × 553 lambda, which is 3806 × 442 μm.

17 bit add-or-subtractor

- Inputs - 35 bits , two 17 bit data and one CE_{sub}
- Output - 18 bits, 17 bit results and one C_{out}
- Delay - Crystal 22 ns, SPICE 24 ns
- Size - 1301 × 315 lambda, which is 1041 × 252 μm.

4. Leading One Detector

4.1. Introduction

Normalization of floating-point number has to be done after each arithmetic operation to ensure that all results are in a normalized form. It is necessary to shift the fractional part to the left until a leading nonzero digit is in the most significant position. The exponent should also be decremented by the shift amount.

The task of a *leading one detector* is to find the position of the leading nonzero digit and output the shift amount coded in binary format to the shifter for shifting left and the *exponent box* for subtraction. With some modification, it can also perform the task of a *zero detector* when no leading one is found. Since this module is lying in the critical path in most floating-point processors and is used in every operation, its speed is a main concern. It is quite a challenge in terms of logic and circuit design to design a leading one detector running at high speed. With a systematic layout and circuit design, CAD tools can be used in the future for similar requirements. To lay the detector out in a systematic manner for VLSI applications pushes the whole design up to a higher degree of difficulty.

4.2. Design Specification

For application in the SPUR project's FPU which supports the IEEE Floating-point Standard, the following is required. There are *67 inputs* namely B_{63} to B_0 and the G (Guard), R (Round) and S (Sticky) bits. To encode the shift amount in binary format requires *7 bits*, which is O_6 to O_0. Also, if no leading nonzero digit is found (the number is zero), '*All* 0' is set. The whole operation must be performed within one phase time which is 25 ns.

4.3. Logic Design

Let the inputs be in_i where i is between 66 and 0. If in_{66} is 1, then the shift amount is 0 because no shifting is required reguardless of the bit pattern of the rest of the bits. Again if in_{66} is 0 and in_{65} is 1, then the shift amount is 1 as there is one leading zero. In general, if inputs in_i to in_{j+1} are zero and in_j is 1 then

the shift amount is $i-j$ where i is the most significant bit and j is between 0 and $i-1$. Let S_n be the shift amount n. Then

$$S_n = \overline{in_{66}}\,\overline{in_{65}} \cdots \overline{in_{j+1}}\,in_j \tag{4.1}$$

Let the outputs be O_k where k is between 6 and 0. Then O_k is the binary code for the shift amount.

A straightforward PLA implementation would be too large and therefore slow. Hence, a slightly unusual approach that borrows ideas from PLA types of layouts with an *AND plane* in the front for generating the S_n from the in_i's and an *OR plane* at the back to generate the O_k has been considered and used. This is a direct intuitive implementation of the function itself without doing any sophisticated logic reduction. The reason for doing so is that there is a pattern that one can follow to make a systematic design and layout. Reducing the logic to its simplest form may end up in randomness which can make the circuit and layout design extremely difficult. Besides, the design approach presented here is general and can accommodate any number of inputs. To implement the required logic for the SPUR FPU with traditional circuit techniques requires an extreme of a 67 inputs AND gate and a 34 input OR gate. A clever way in circuit design and layout to achieve this is introduced in the next section.

4.4. Circuit and Layout Design

To generate the desired logic function with minimum area and still meet the timing requirements, *dynamic circuits* are used for their higher speed and density. Since there is no logic inversion required, Domino Logic [KrLL82] is used so that both the AND and OR plane can be clocked at the same phase.

A CMOS Domino OR gate with a large fan-in is easy to design. It is just a dynamic NOR gate with an output inverter. The larger number of inputs introduces only a larger *load capacitance* C_1 at the output node and a larger *node capacitance* C_2 between the input N-channel devices M_i and the *evaluation* N-channel device M_e as shown in figure 9. The speed of the OR plane is thus limited by the speed with which the N-channel devices can discharge C_1 and C_2. One can use the largest N-channel device possible, limited only by the height of the pitch of each bit and a matching evaluation N-channel device for maximum driving capability.

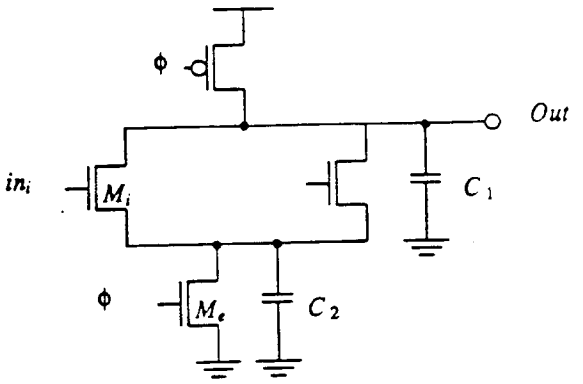


Figure 9 Dynamic NOR Gate with Multiple Fan-in

The problem arises in the implementation of AND gates with a large fan-in. Of course a one stage 67 bit AND gate is impossible! Several approaches were investigated. One can use the Multi-stage *Domino AND gate* [KrLL82] or use an *alternate AND-OR gate (NORA)* [GoMa82] to implement the function. Domino AND gates with maximum fan-in of 4 were chosen to be used.

A larger fan-in for each AND gate may result in a *charge sharing* problem because the simplest form of Domino gates were used without P-channel feedback or extra precharging P-channel devices for simplicity and higher density. The P and N devices of the output inverter of each gate was sized with a ratio of 1:3 to make sure the gates are trouble free from charge sharing while maintaining a reasonable safety margin and transfer characteristic for output. Worst case simulation using SPICE shows that the precharged node will dropped down to 3.54 V due to charge sharing while the output voltage remains at around 0.1V with a noise margin of at least 1V including device parameter variation.

A smaller fan-in may result in the requirement for additional stages for implementation of the multi-stage AND function. Three stages of 4 input AND gates can provide the equivalent of a 64 input AND function. If a 3 input AND is used, 4 stages are required. The design at first was intended for a 64 bit leading one detector forgetting about the three rounding bits. For a 67 bit leading one detector, using 3 input AND gates instead of 4 input AND gates will perform at least as well if not better (higher speed and less charge sharing problem) at no overhead because 4 stages will be required regardless of the choice.

In the layout of the Domino AND gates, the width of the long chain of N-channel inputs is scaled with the smallest at the top. This has been proven to give better performance and higher speed [Shoji82] [Shoji85]. This is due to the fact that successive N-channel device in the chain has to drive less capacitance. By doing so, the speed of the detector *improved by 15%*.

A problem still unaddressed is the optimization of the number of gates required keeping in mind that one must maintain the *symmetry* and *generality*. A simplified example is given to illustrate the approach. Consider designing an 8 bit leading one detector with the constraint that only 2-input AND gate can be used. The digits of the binary data and their complements are given as inputs.

To perform the function, 3 stages are required to make an 8-input AND. By observing (4.1), a lot of terms can be shared. A symbolic logic and layout is shown in figure 10 for illustration.

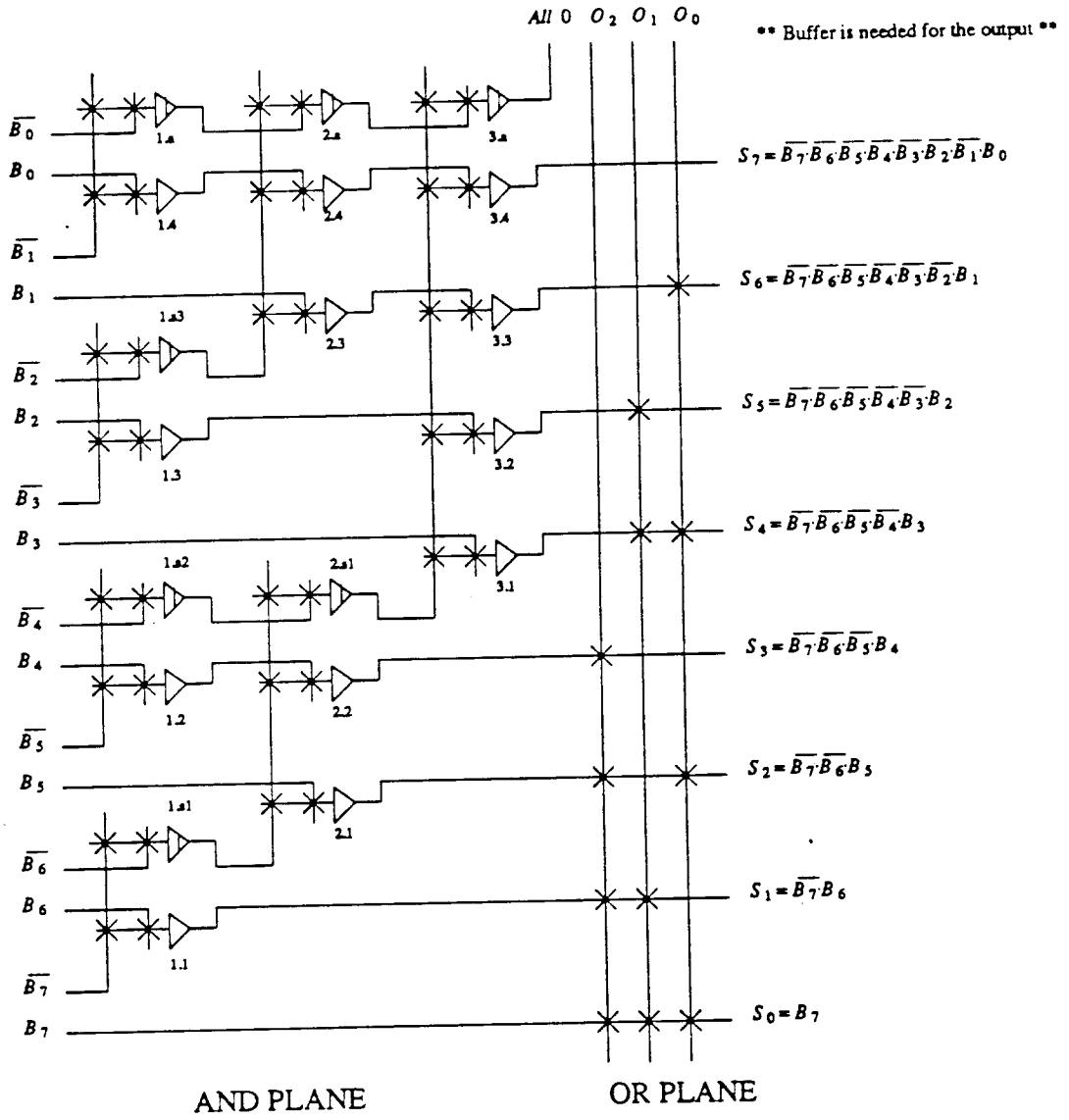


Figure 10 Symbolic Representation of an 8 bit Leading One

and All Zero Detector

Gates 1.1 to 1.4 form the first stage of results. They are used for generating the last two terms in (4.1). Note that for the first stage, the odd number inputs just pass through the first stage leaving room for *intermediate AND terms* 1.s1 to 1.s3 to generate intermediate results needed in the second stage. For the second stage, 2.1 to 2.4 are used to generate higher order terms with the intermediate results from the first stage. Since the lower order terms are generated in the first stage, there is room again for gate 2.s1 to generate intermediate results for the third stage. Gates 3.1 to 3.4 combined with the intermediate result of the

second stage form the higher order terms. With an OR plane at the back, the 8 bit leading one detector is completed.

Note that the output is inverted because only the NOR function is included in the module, *the output inverter is missing!* This is deliberately done because these outputs have to drive other circuits such as the shifter decoder, buffers are required in between, the correct number of inversions is assumed to be done by the buffers.

As one can see, S_7 isn't used in the OR plane. Therefore there is no difference between the case where all digits are zero, and all digits are zero except the least significant bit. This is in some sense true because for both cases, we need to shift the fractional part by the maximum amount. But with a few additional gates, the leading one detector can also serve as a zero detector. Gates 1.a to 3.a are additional gates for zero detection.

$$All\ 0 = \overline{in_{66}} \cdot \overline{in_{65}} \cdots \overline{in_{j+1}} \cdot \overline{in_j} \tag{4.2}$$

Note the high degree of similarity for each stage which allows the design of one cell and array it to form the different stages.

The critical path is through each intermediate gate. In this case, it is through gate 1.s1 to 2.s1 then through any of the third stage AND gates to any of the output OR lines because the gate of the last stage and in the OR plane are identical.

For the 67 bit leading one detector, a similar idea is used. The only problem is the large fan-out of the intermediate gates. For the 8 bit example, 1.s1 has to drive 3 second stage gates (2 normal + 1 intermediate). Gate 2.s1 has to drive 4 third stage gates. The same is true for the 67 bit leading one detector. The intermediate gate of the first stage has to drive 17 gates (16 normal + 1 intermediate) and the intermediate gate of the second stage has to drive 48 gates!! For a fan-out of 17, the output inverter of the Domino gates can still handle the load. However for a fan-out of 48, buffers have to be used.

4.5. Simulation Results

A complete schematic showing the critical path for SPICE simulation is shown in figure 11 with the device sizes and critical capacitances sizes shown in figure 12. The parameters used are extracted from actual layout. The critical path is found by Crystal and agrees with what was expected by inspection. M25 to M32 forms the first intermediate stage, M18 to M24 with M36 froms the second intermediate stage. M14 to M16 forms the buffer. M7 to M15 with M35 forms the third stage. M2 to M5 with M34 forms the last stage. M1, M33 and M37 forms the output OR stage.

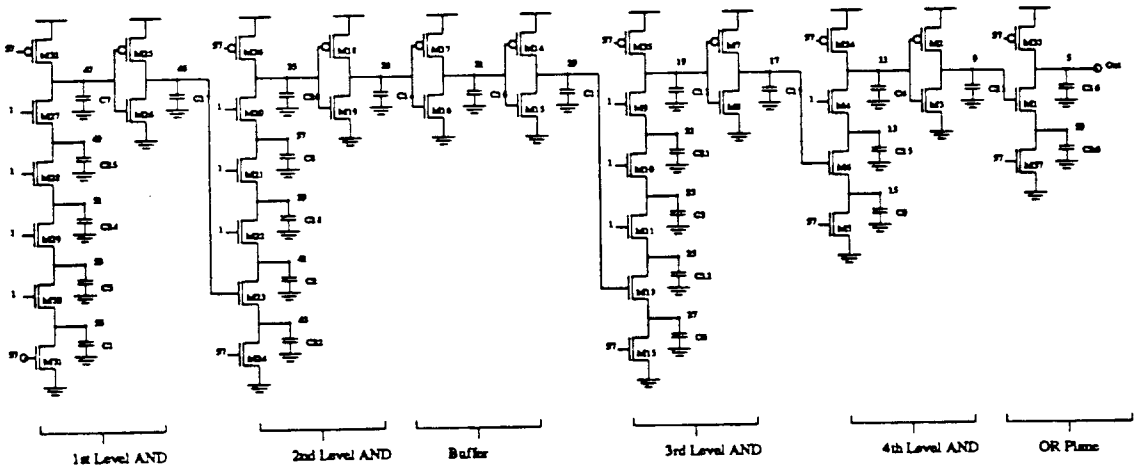


Figure 11 SPICE Model Used For Simulation

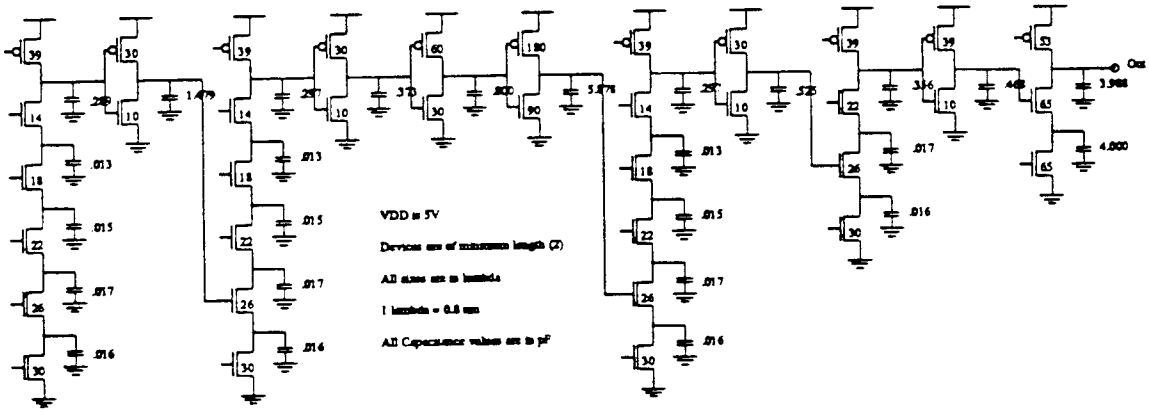


Figure 12 Device Sizes and Critical Capacitance for SPICE

According to Crystal, the delay is 20.5 ns. According to SPICE simulation using typical parameters, the delay is 15 ns. MOSSIM is used to verify the correct functionality of the layout. The final layout is of

size 4901 × 463 lambda which corresponds to 3920 × 320 μm.

4.6. Summary

A leading one detector with zero detection capability which meets the requirements for use in the SPUR FPU has been designed and verified logically and electrically. Its function is to detect the position of the leading nonzero digit and code the result in binary format for normalization.

Characteristics :

- Inputs - 67 bit, B_{63} to B_0 with G, R and S
- Output - 8 bit, O_6 to O_0, $All\ 0$
- Delay - Crystal 20.5 ns, SPICE 18 ns
- Size - 3920 × 320 μm

5. Shifter with Sticky Logic

5.1. Introduction

The shifter is an important part in the FPU fractional datapath because it is the work horse that performs all the shifting required. Shifting is needed in the *normalization*, *multiplication*, and *division* operations. The shifter must be able to shift *any number of digits* and in *both directions*. The different requirement is that to support the IEEE Floating-point Standard, special treatment is required for right shifting. During a right shift, 3 extra bits are generated for high precision rounding. They are the G, R and S bits. The G and R bits are extra bits used for storing the data shifted out of the normal lower-order end. The S or *Sticky bit* is set and remains set if a 1 is shifted into it during right shifting.

5.2. Design Specification

In the SPUR FPU, operations are different for right shift and left shift. For both shifts, the shifter decoder accepts a *7 bit* shift amount namely I_6 to I_0. They are the binary code for the shift amount so that a decoder is needed to decode the inputs to 67 control lines, namely Shf_0 through Shf_{66}. Shf_0 means no shifting and Shf_i means shift i bits. For a right shift, a *64 bit* input is shifted right to form a *67 bit* result as explained in the introduction. If the shift amount is larger than or equal to 66, then an extra input is used call CF_{shfmax} which is once enabled, the shifter is going to shift the maximum amount, 66 bits, to the right. Since the decoder is fully decoded, all other *shfi*'s will be disabled. For the left shift, a *67 bit* input is shifted to the left to form back a *64 bit* result.

The whole operation has to be done within one phase time which is 25 ns long.

5.3. Logic design

Mixed-logic is used in the shifter, i.e. a signal can be active high or active low. An active high signal means that a 1 is represented by a HIGH voltage, and an active low signal means that a 1 is represented by a LOW voltage. The design itself is simple, all that is needed is a decoder to decode the shift amount and a barrel shifter to do the shifting. The barrel shifter is nothing but a big array of passgates enabled by the

Shf_i control lines to perform the required shifting. For more detail, see [2] [8] [9]. To generate the sticky bit is a little bit tricky in here. Logically, it depends on the shift amount and all inputs. To have the highest speed, a brute force method can be used and will be shown later. The decoder basically is a NOR decoder with one level of predecoding.

5.4. Circuit and Layout Design

The whole shifter can be divided into the *decoder*, the *barrel shifter* and the *sticky bit generator*. The floor plan of the whole shifter is shown in figure 13.

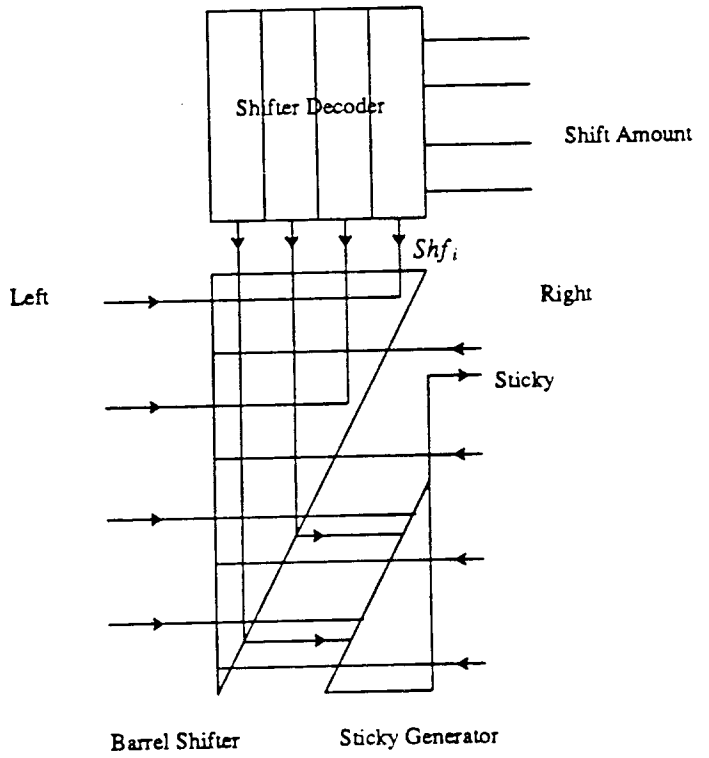


Figure 13 Floor Plan for the Shifter

The decoder is on top accepting the shift amount and provides the Shf_i signals to the barrel shifter and the sticky bit generator. The barrel shifter is a big triangular array of passgates to perform the shifting of any amount in both directions. The sticky bit generator is another big triangular module that can be folded into the barrel shifter wasting no area at all.

Since both the barrel shifter and part of the OR gates that are folded into the shifter are regular structures, CAD tools can be used again. A program has also been written to generate the desired pattern for any number of inputs. Together with a template file, Mquilt is used again to generate the whole block automatically.

5.4.1. Barrel Shifter

Because the barrel shifter is a big module, the layout objective is to design it to be *as narrow as possible* for each bit. Dynamic circuits are used so that the passgate can be just an N-channel device instead of full CMOS passgates. Only the passgate and necessary vertical connections are used to make the cells for each bit as narrow as 18 lambda.

The logic used is shown in figure 14.

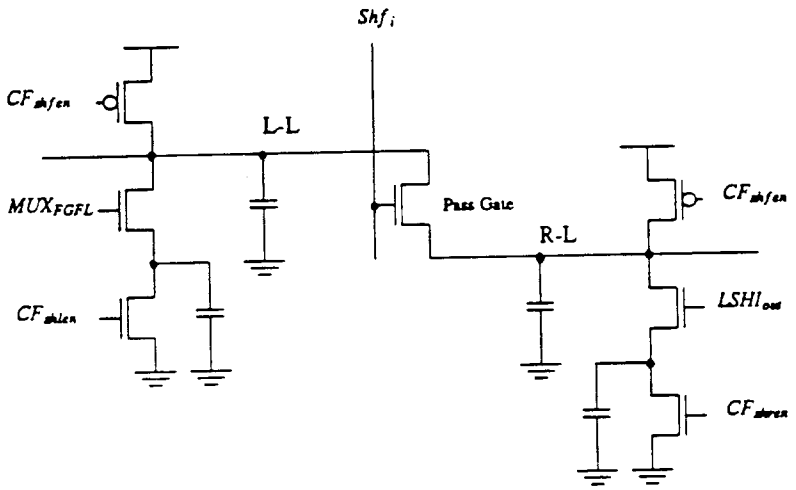


Figure 14 Dynamic Bidirectional Shifter Cell

When CF_{shfen} is LOW, it is precharging both L and R lines. When it is HIGH, the shifter is active. Depending on left shift or right shift, the CF_{shlen} or CF_{shren} is set to HIGH respectively and the other remain LOW. Taking a right shift as an example, CF_{shren} is set to HIGH, so whatever is in the MUX_{FGFL} output (a multiplexer that selects the input to the shifter on right shift) is going to selectively discharge line L. Thus making L an active LOW input (voltage LOW represent 1). When the Shf_i line is activated, the signal is passed to R making it also active LOW. Because the CF_{shlen} remains LOW, the only possible

discharge path of R is through the passgate to L controlled by the input. Similarly, the left shift is done in the same manner. For left shifting, the input is taken from $LSHI_{out}$ (Left-SHift-In-latch out). It must be careful that the output of the shifter is active LOW, so that the correct logic has to be converted back afterwards.

Electrically, the N-channel device for the passgate can be a minimum device because shifting is fast through a passgate. The time determining factor is the input driver discharging the L and R lines. The capacitance on those lines consists mainly of the diffusion capacitance of the passgates (a maximum of 67). Therefore, it would be wise to make the passgates small. 6/2 lambda devices were used.

5.4.2. Shifter Decoder

For the purpose of speed, the decoder has been designed with a dynamic scheme. To fit it into the narrow pitch of the shifter makes the decoder plane a little different from other decoders. The idea is shown in figure 15.

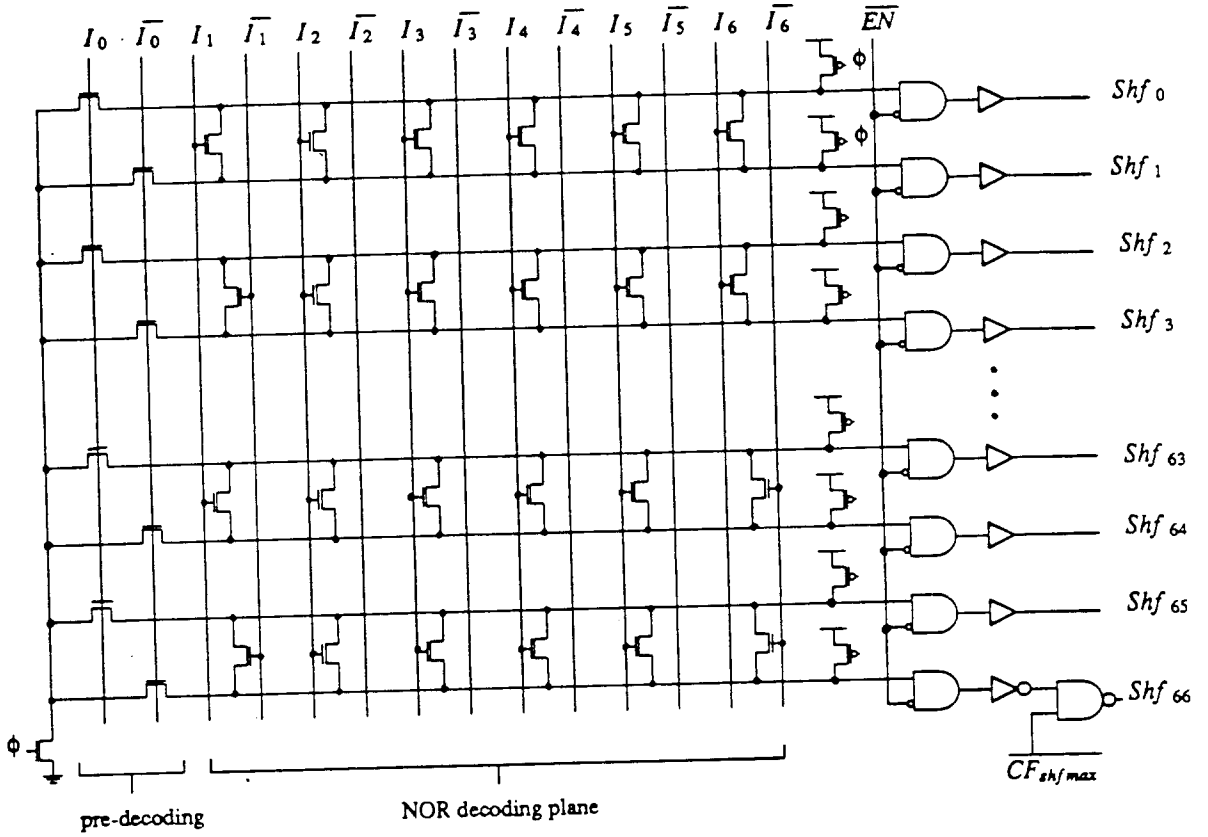


Figure 15 Floor Plan for Shifter Decoder

It is basically a NOR decoder with one level of predecoding. It is also self-enabled as explained later. Normally, ϕ is low for precharging, the $\overline{EN}$ is HIGH forcing the output of the AND gates to be LOW so that every Shf_i is low. When ϕ is high, it is in the evaluation stage. A first stage of predecoding is done by the I_0 and $\overline{I_0}$ which selectively discharges one out of every two lines. The rest of the decoding is done by the special passgates matrix shown. To explain how it works, an example is shown in figure 16.

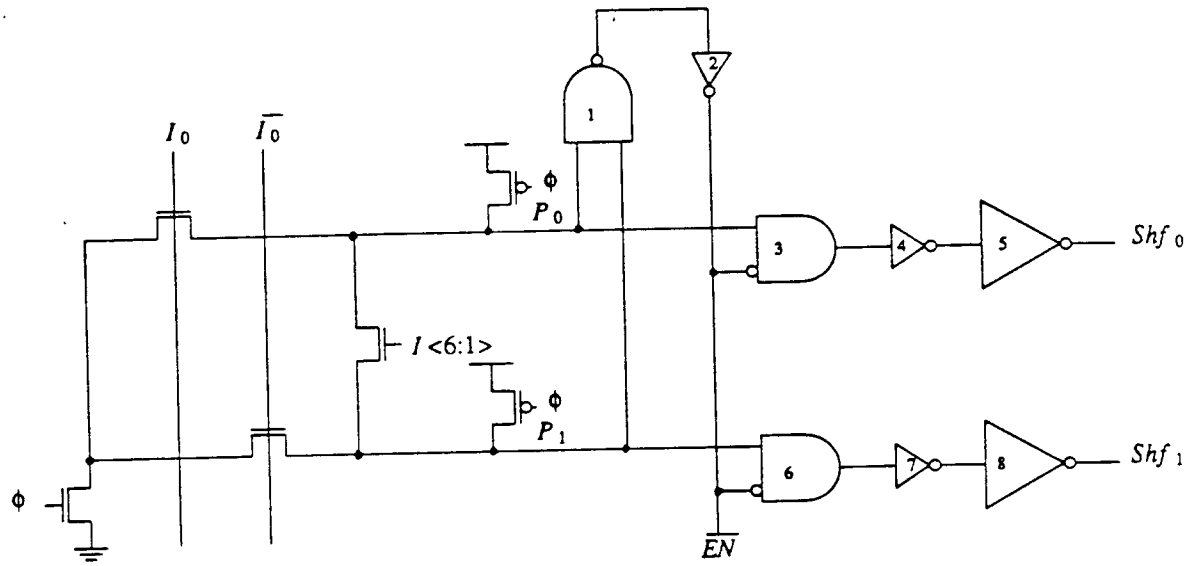


Figure 16 An example for the Shifter Decoder

Assuming the shift amount is 0, then I will be 0000000, and $\overline{I}$ will be 1111111. With the first level of predecoding, all the lines connected to ground through the evaluation N-channel device and N-channel gates controlled by $\overline{I_0}$ will be discharged. Therefore P_1 is discharged. Since the connections between P_0 and P_1 are all formed by N-channel devices controlled by I_6 to I_1, P_0 stays HIGH. While all other pairs of lines have some sort of N-channel devices to short them out, only P_0 will stay HIGH and the rest all go LOW. When this happens, the $\overline{EN}$ signal should be LOW to activate the AND gate so that the corresponding Shf_0 line can be driven HIGH.

Generation of the $\overline{EN}$ signal is the hard part. Decoding and shifting have to be done in the same phase and only one clock line is available to control the charging and precharging. If the $\overline{EN}$ line is enabled (by setting it LOW) at the same time as ϕ, the discharging of the P lines takes time and during those periods, all the Shf_i lines are HIGH and the barrel shifter logic is ruined. So there must be a self-time method to enable this control signal. A peripheral logic circuit is connected to the P_0 and P_1 line. Since during the decode stage, only one of the P lines are going to stay HIGH, an AND gate can be used to indicate that the decoding is done. Normally, when P_0 and P_1 are precharged, the output of gate 1 is LOW and that of gate 2 is set HIGH, thus forcing the output of gate 3 LOW so the Shf_i line will remain LOW. When the decoding is done, at least one of P_0 or P_1 is LOW so that $\overline{EN}$ is LOW enabling the correct Shf_i.

By doing this, the timing of the dynamic circuit is correct.

Going back to diagram 15, one can see that there is also some special circuits hanging out of the Shf_{66} bit. Since the shift amount may exceed 66, and the decoder is fully decoded, none of the Shf_i bits will go HIGH but still a maximum shift have to be performed. The circuit in there makes this possible by keeping $\overline{CF_{shfmax}}$ normally HIGH, so that what ever signal comes out of the decoder can pass through. If a maximum shift is desired, one can enable $\overline{CF_{shfmax}}$ by setting it LOW forcing Shf_{66} to be HIGH. The decoder will keep the rest of the Shf_i lines LOW by giving a shift amount larger than 65.

5.4.3. Sticky Bit Generator

For high precision rounding, 3 extra bits are used for shifting right. They are the G, R, and S bits. The G and R bits are a natural extension of the lower-order bits but the S or *Sticky* bit is different. It stores information about all the bits shifted out of the R bit. It is set and remain set if a 1 is shifted into it during shifting. A brute force method must be used to ensure the result can be obtained in 25 ns. The circuit used is shown in figure 17.

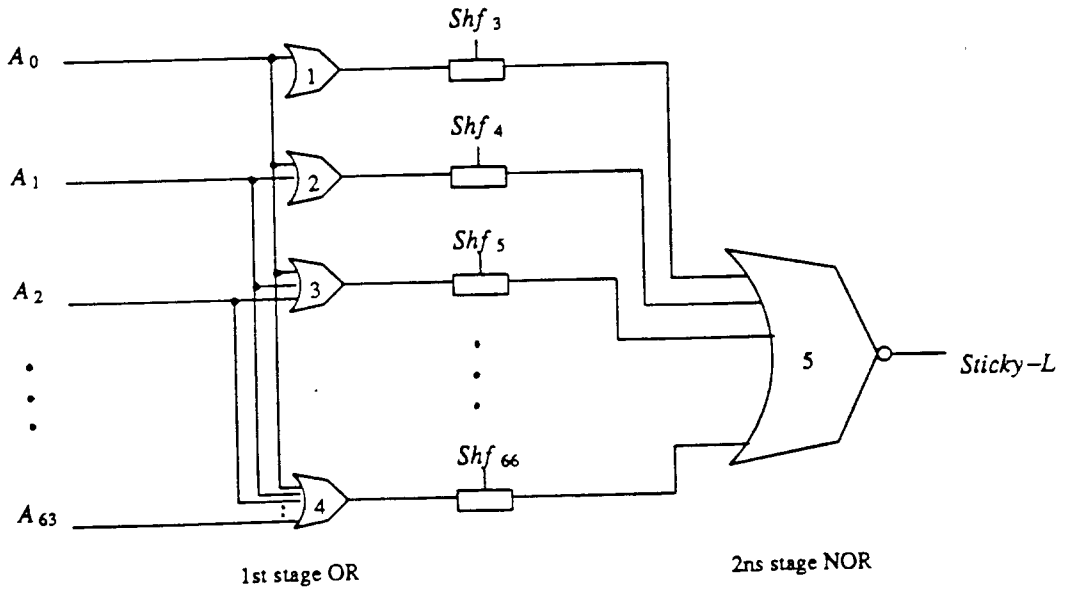


Figure 17 Sticky Bit Generation

The inputs to this circuit are A_{63} to A_0, depending on the shift amount. The correct result is passed through the passgate to the output NOR gate to generate the Sticky bit which is active LOW. The reason of making

it active LOW is for compatibility with the rest of the output from the shifter. Note that A_0 is gated through Shf_3. This is because only after shifting right for three bits will A_0 arrive at S (remember the G and R bits). Others are gated similarly.

In order to implement this with separate circuits, a large amount of area is needed. Fortunately, this can be folded into the barrel shifter, thus effectively makes use of the unused area of the barrel shifter. The way it is implemented is shown in figure 18.

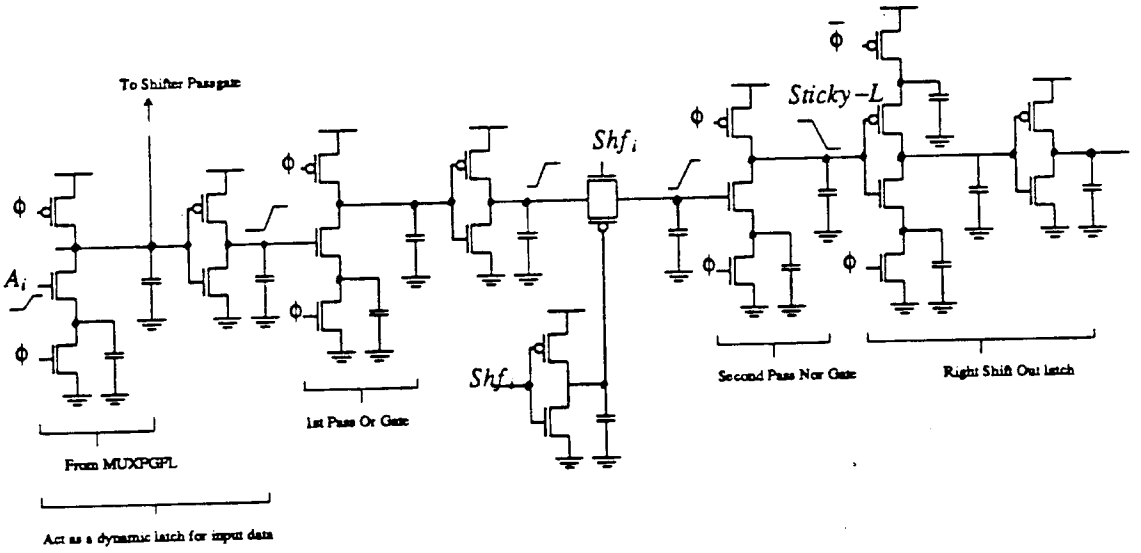


Figure 18 Sticky Bit Generator Circuit Diagram

The input stage is the same as the barrel shifter, an inverter is put in front to convert the polarity of the input back to active HIGH. Then there is a stage of domino OR gates, gated through a CMOS passgate controlled by Shf_i to another big NOR gate. A dynamic shift out latch is included for completeness of the whole scheme. To lay this out is a major headache, because it has to have the same pitch as the barrel shifter. An array of the first pass OR gates with only the N-channel devices are squeezed beneath the barrel shifter. The precharging P-channel devices and the rest of the circuit can stay out of the module.

5.5. Simulation Results

5.5.1. Shifter Decoder

The worst case path of the decoder is through the enabling scheme driving Shf_0 which is the line with most capacitance because there are 67 N-channel diffusions hanging on there. Simulation results from Crystal shows that this is indeed the case. A circuit model used for SPICE for timing analysis is shown in figure 19.

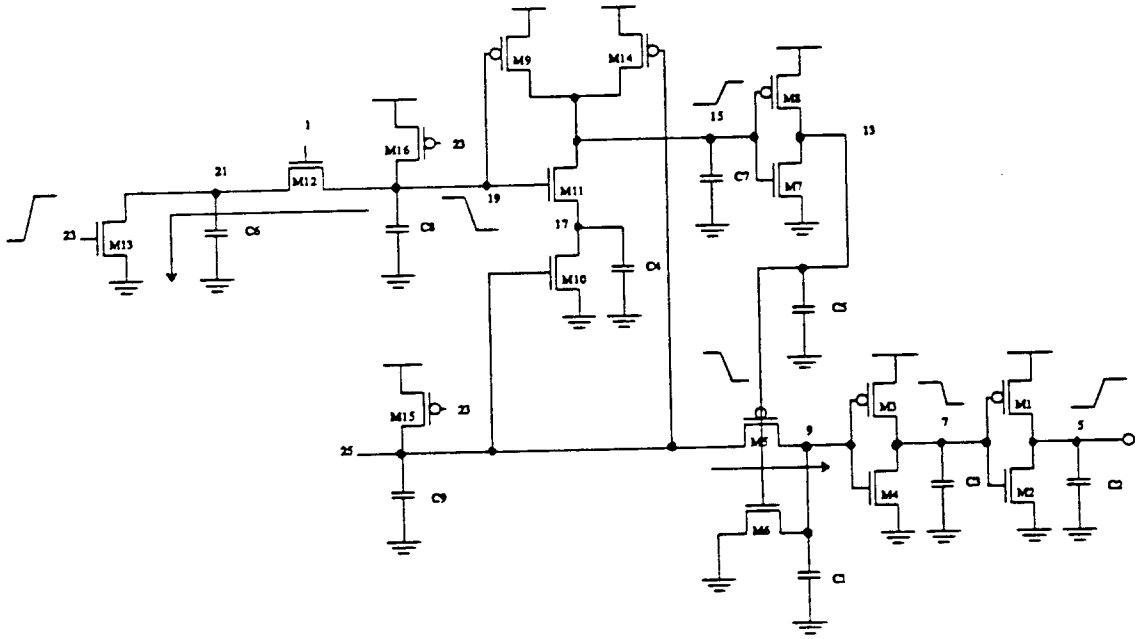


Figure 19 Shifter Decoder Model for SPICE Simulation

M13 is the evaluation device which has to drive 6 pF of capacitance and is therefore made as wide as the decoder itself for highest speed. M12 is the predecode passgate which is also made as big as possible within the pitch limit. Since none of the passgates between P_0 and P_1 is on, P_0 and P_1 are separated. Node 19 is P_1 and it is discharged through M12 and M13 to ground. The output of the static NAND gate is therefore HIGH and the output driver inverter LOW. M5 and M6 forms a special AND circuit for enabling. Since inverter M7 and M8 have to drive 67 of these special AND gates, the size of M5 and M6 have to be reasonably small. Then, M1 to M4 make up the output driver that drive the output Shf_i lines and have to be big to drive a low of about 5 pF. The device size and loading conditions are shown in figure 20.

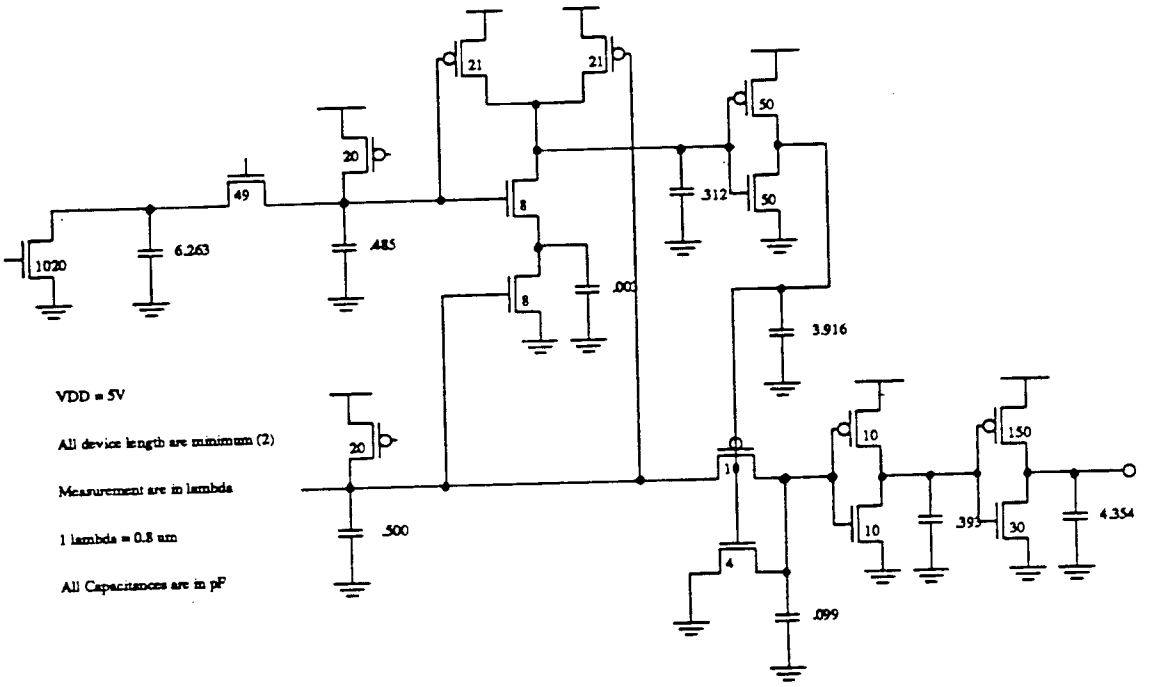


Figure 20 Shifter Decoder Device Sizes for SPICE Simulation

According to SPICE, the delay from evaluation to the setting of Shf_i takes about 9 ns. MOSSIM was used to verify, at the switch level, the logic of the decoder showing no problem at all. The layout of the decoder is of size 534 × 1272 lambda which corresponds to $427 \times 1018\ \mu m$.

5.5.2. Barrel Shifter

The circuit for the shifter itself is simple. Left shifting is simulated because of the higher capacitance on the L lines (node 59). The L lines are more capacitive because it has to drive a big inverter to the sticky bit generator. The model and device sizes are shown in figure 21.

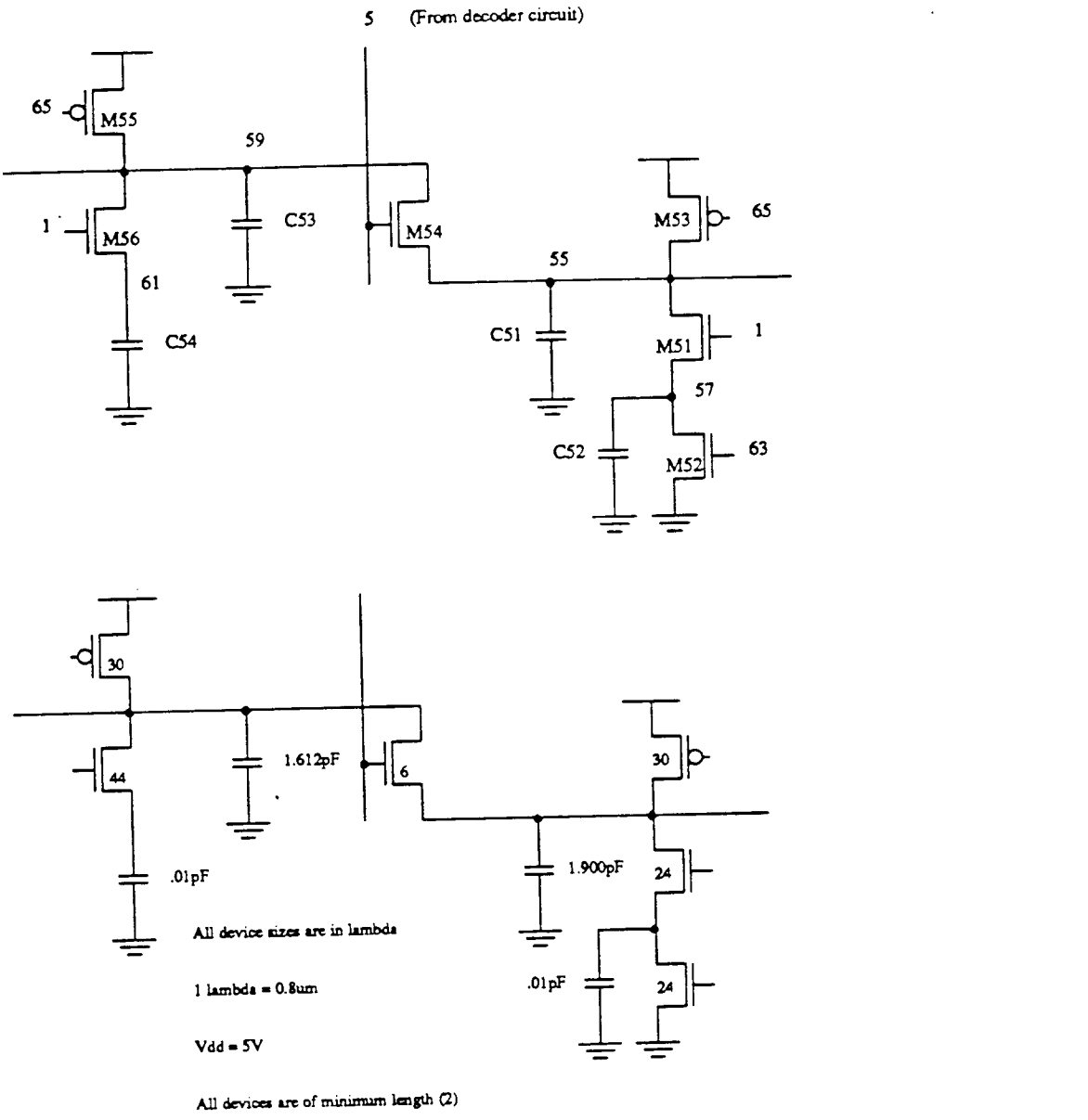


Figure 21 Barrel Shifter Model and Device Sizes for SPICE simulation

Note the on the left, M56 is included to account for the extra charging needed if the left input is turn on while the CF_{shren} is off.

According to SPICE, the worst case delay started from decoding is 15 ns. MOSSIM is used to verify the whole matrix starting from decoding and shows that everything works in the switch level.

5.5.3. Sticky Bit Generator

The models and sizes used for SPICE simulation are shown in figure 22.

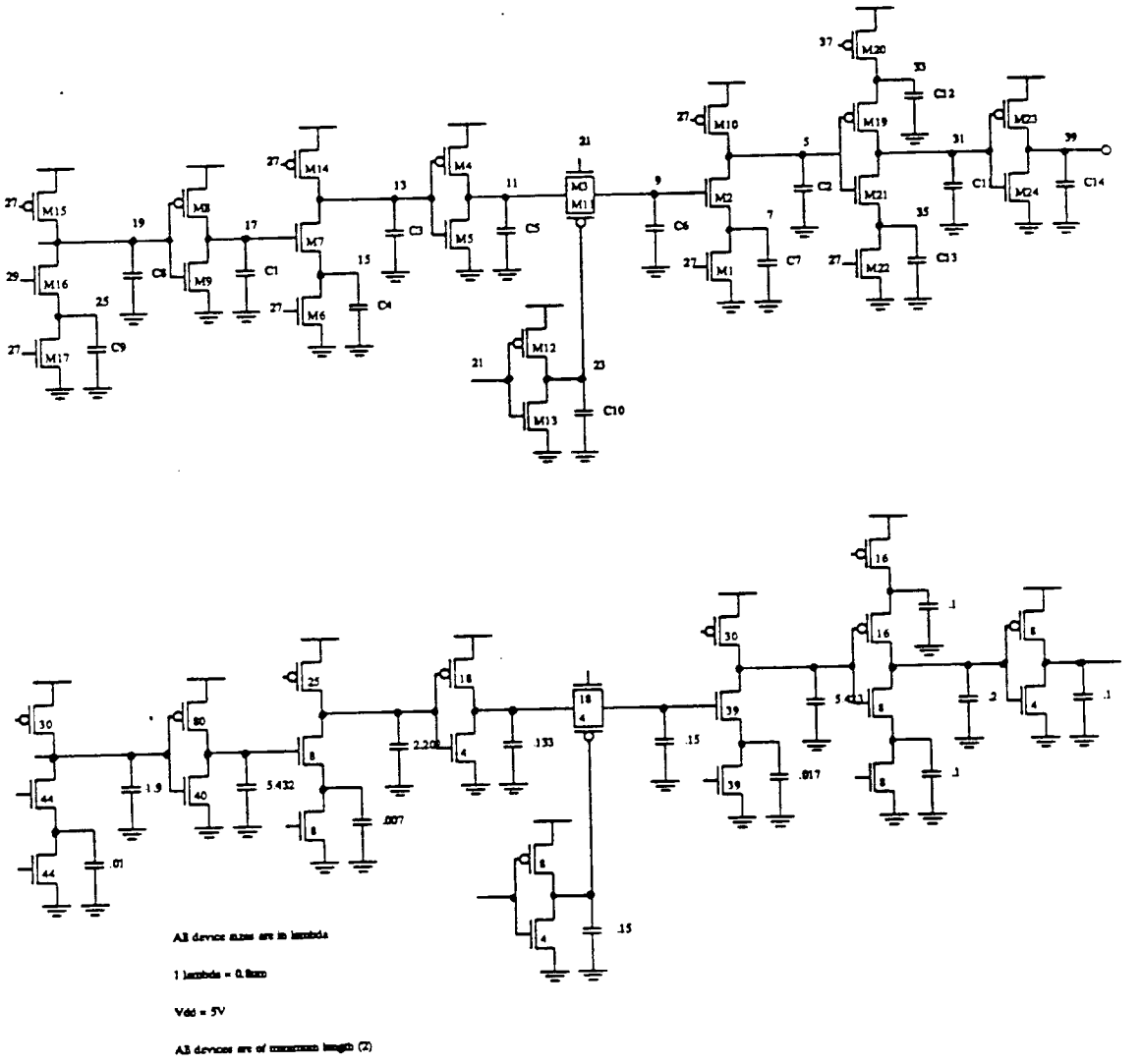


Figure 22 Sticky Bit Generator Model and Device Sizes for SPICE simulation

The worst case delay is when Shf_{66} is enabled and only one of the A_i is activated. Running Crystal gives the same worst case path. From SPICE simulation, the delay till the data is latched into the dynamic latch of the next module takes 25ns. The way the circuit just match the specification really doesn't matter in here because it still have a dead time of 10 ns for the data to settle. Besides, the sticky bit is not needed until the next cycle, so this impose no trouble with the design. The final layout of the barrel shifter and the sticky bit generator is 4874 × 1224 lambda which is 3899 × 979 μm.

5.6. Summary

A shifter that accepts a binary coded number and perform a right or left shift of any amount supporting IEEE standard is designed. It has been simulated electrically by SPICE and logically by MOSSIM.

Characteristics:

- Inputs - Binary coded shift amount *I* <6:0>, data *L* <63:0> or *R* <63:0>*GRS*
- Output - Perform the shifting supporting IEEE standard.
- Delay - Shifting 15 ns, Sticky bit (latched into output latch) 25ns
- Size - The whole module with decoder is 5359 × 1414 lambda, which is 4287 × 1131 μm.

6. Remarks

6.1. CAD Tools Used

For a project of this size, CAD tools play an important role in the whole design process. The major tools for VLSI layout are the Berkeley VLSI Tools [Walt86]. They provides a good environment for VLSI design by integrating a lot of design tools together.

The layout tool used is Magic which has a lot of supporting software for easier non-array-like layouts (e.g. Mpack, Mpanda, Mpla, Mquilt, ... just to name a few). Magic does layout and design rule checking simultaneously which is a nice feature to have. This allows immediate feedback and time efficient because the design is being checked while the designer is idle. Magic also does the extraction of the layout. The extraction file from Magic can be converted to different format for other CAD Tools to read in as input.

For switch-level simulation, Esim was used at the beginning. It is not a good software because it does not have the idea of device sizes and cannot handle feedback circuits. The design of the XOR gates used was influenced by this as explained in the report (3.4.1). Later MOSSIM is used which is a much better software. Though it is a switch-level simulator, it has idea about device sizes and capacitances. It can detect potential race conditions and charge sharing problems which helps a lot in debugging the circuits.

For timing information, Crystal and SPICE were used. Crystal helps in performance tuning by pointing out paths that limit clock speed and approximate delay of each gate in the path. SPICE helps in circuit-level simulation for final verification of timing for each module.

Running all these programs demands a lot of computing power. SUN3 workstations make interactive layout possible. Several supermini-computers were used for running other simulations. VAX-8650, VAX-8800 and IBM 3081 were used for large SPICE simulations.

Without all these nice tools, this project might take two to five times longer to finish.

6.2. Lessons Learned

Making the initial decisions was the hardest thing to do in this project. The initial decision of using metal 1 as vertical control lines and metal 2 as horizontal data lines for the datapaths. This proved to be a good decision because it greatly simplified the layouts and global routing.

The pitch that was used in the datapaths for each bit is the spacing for eight metal 2 lines to pass through. The assignment for each 'track' is two for Vdd and GND, two for the A and B buses, leaving four lines free for input or passing data through modules. This arrangement proved to be a bad choice. For an example, the adder takes two inputs and the fast carry generator needs one track to pass the pIN_i from the pre-conditioning block to the sum block. The layout of the fast carry generator needed one track as a jumper for the cells leaving no free track behind. As a result, no signal can pass through the adders. The floor plan to the datapath has to be made carefully, bearing in mind that no signal other than the A and B buses can pass through the adders.

A similar problem occurred in the design of the shifter. The Vdd track must be used to pass a signal through the shifter. This is possible because the barrel shifter contains passgates only and N-channel devices were used. Only the GND line is used for well-contact of each N-channel devices.

Another lesson learned is to be careful about the limitations of Magic design rule checking. The design rule checker of Magic uses the technology file provided by the designer. The technology file used allowing diffusion width of 2 lambda which is 1.6 μm. An IC designer should recognize immediately that with the present processing techniques, the above is impossible because just the 'bird-beak' may eat up at least 1 μm leaving so little room behind that it is unsave to use a 2 lambda diffusion width. The second thing is that the checker doesn't check for well contacts. It is extremely easy to forget to put in well-contact especially for passgates.

Lots of CMOS digital circuit design techniques were used in this project. Putting things learned in class and from papers into practice has to take into account a lot of practical issues that a designer may easily overlook. Especially for dynamic circuits which signals depend only on stored charges. The designer has to be careful of problems like charge sharing, leakage paths, timing ... etc.

6.3. Future Work

Other than the datapaths, there were also the multiply and divide unit [Bose87], and other control parts. Global routing of the whole FPU chip is in progress. After routing, global simulation of the FPU chip is the next target. Testing will be done once the chip is fabricated.

Appendix

A.1 SPICE Model Used

```
*********************************************************************
*          TYPICAL Device parameters for the HP CMOS40 Process      *
*                                                                   *
*                      Released 2/6/86 by Rich Duncombe             *
*        NOTE:  These parameters are intended for digital design only.*
*                                                                   *
*********************************************************************
*
* Use N and P models for W >= 4U and L <= 2U
*
.MODEL N NMOS LEVEL=2 VTO= 0.75 KP=76.0U GAMMA=.40 LAMBDA=.025 TOX=25N
+ NSUB=4E16 TPG=+1 XJ=.25U LD=.20U UEXP=.16 VMAX=5.5E4 JS=1000U
+ CGSO=220P CGDO=220P CJ=230U CJSW=260P CGBO=400P
*
.MODEL P PMOS LEVEL=2 VTO=-0.75 KP=27.0U GAMMA=.50 LAMBDA=.045 TOX=25N
+ NSUB=2.0E16 TPG=-1 XJ=.20U LD=.05U UEXP=.15 VMAX=9.0E4 JS=1000U
+ CGSO=220P CGDO=220P CJ=670U CJSW=215P CGBO=400P
*
* Use NBIG and PBIG models for W >= 4U and L >= 2U
*
.MODEL NBIG NMOS LEVEL=2 VTO= 0.75 KP=62.0U GAMMA=.40 LAMBDA=.001 TOX=25N
+ NSUB=4E16 TPG=+1 XJ=.25U LD=.20U UEXP=.02 VMAX=5.5E4 JS=1000U
+ CGSO=220P CGDO=220P CJ=230U CJSW=260P CGBO=400P
*
.MODEL PBIG PMOS LEVEL=2 VTO=-0.75 KP=21.0U GAMMA=.50 LAMBDA=.005 TOX=25N
+ NSUB=2.0E16 TPG=-1 XJ=.20U LD=.05U UEXP=.05 VMAX=9.0E4 JS=1000U
+ CGSO=220P CGDO=220P CJ=670U CJSW=215P CGBO=400P
*
* Use NMIN and PMIN models for W <= 4U and L <= 2U
*
.MODEL NMIN NMOS LEVEL=2 VTO= 0.75 KP=58.0U GAMMA=.40 LAMBDA=.025 TOX=25N
+ NSUB=4E16 TPG=+1 XJ=.25U LD=.20U UEXP=.18 VMAX=5.5E4 JS=1000U
+ CGSO=220P CGDO=220P CJ=230U CJSW=260P CGBO=400P
*
.MODEL PMIN PMOS LEVEL=2 VTO=-0.75 KP=17.5U GAMMA=.50 LAMBDA=.045 TOX=25N
+ NSUB=2.0E16 TPG=-1 XJ=.20U LD=.05U UEXP=.15 VMAX=9.0E4 JS=1000U
+ CGSO=220P CGDO=220P CJ=670U CJSW=215P CGBO=400P
*
*********************************************************************
```

A.2 Input Decks and Plots

```
WORST CASE PATH FOR EGsubEL

**************************************************
*                                                *
*       EGsubEL                                  *
*       -------                                  *
*       A 17 bit adder subtractor with control   *
*       line CWsub.                              *
*       Consist of a 18 bit adder (17 bit data   *
*       and Cin) and an input XOR                *
*                                                *
*       Delay is ____24 ns_____                  *
*                                                *
*                        7/8/86 8:20pm           *
*                                                *
**************************************************

C1 9 0 0.189pf
C2 39 0 0.002pf
C3 27 0 0.027pf
C4 7 0 0.007pf
C5 11 0 0.027pf
C6 47 0 0.104pf
C7 37 0 0.092pf
C8 31 0 0.002pf
C9 23 0 0.125pf
C10 35 0 0.197pf
C11 5 0 0.139pf
C12 41 0 0.252pf
C13 19 0 0.952pf
C14 17 0 0.045pf
C15 13 0 0.100pf
C16 43 0 0.007pf
C17 15 0 0.002pf
C18 33 0 0.045pf
C19 29 0 0.283pf
C20 21 0 0.153pf
C21 25 0 0.007pf
M1 5 1 7 0 n l=1.6u w=3.2u
M2 7 9 0 0 n l=1.6u w=3.2u
M3 1 9 5 1 p l=1.6u w=12.8u
M4 1 13 9 1 p l=1.6u w=3.2u
M5 0 1 11 0 n l=1.6u w=4.0u
M6 11 13 9 0 n l=1.6u w=4.0u
M7 15 1 0 0 n l=1.6u w=3.2u
M8 13 19 15 0 n l=1.6u w=3.2u
M9 1 19 17 1 p l=1.6u w=6.4u
M10 13 0 17 1 p l=1.6u w=6.4u
M11 19 21 1 1 p l=1.6u w=12.8u
M12 19 21 0 0 n l=1.6u w=6.4u
M13 0 23 21 0 n l=1.6u w=3.2u
M14 1 23 21 1 p l=1.6u w=6.4u
M15 25 0 1 1 p l=1.6u w=6.4u
M16 23 29 25 1 p l=1.6u w=6.4u
```

```
M17 0 29 27 0 n l=1.6u w=4.0u
M18 27 1 23 0 n l=1.6u w=4.0u
M19 31 1 0 0 n l=1.6u w=3.2u
M20 29 35 31 0 n l=1.6u w=3.2u
M21 1 35 33 1 p l=1.6u w=6.4u
M22 29 0 33 1 p l=1.6u w=6.4u
M23 1 37 35 1 p l=1.6u w=8.0u
M24 0 37 35 0 n l=1.6u w=4.0u
M25 37 1 39 0 n l=1.6u w=3.2u
M26 39 41 0 0 n l=1.6u w=3.2u
M27 37 41 1 1 p l=1.6u w=6.4u
M28 41 0 43 1 p l=1.6u w=10.4u
M29 43 47 1 1 p l=1.6u w=10.4u
M30 0 47 41 0 n l=1.6u w=3.2u
M31 47 49 0 0 n l=1.6u w=3.2u
M32 47 49 1 1 p l=1.6u w=6.4u

* Initial conditions:
.ic v(9)=0.000000
.ic v(39)=5.000000
.ic v(27)=0.000000
.ic v(7)=5.000000
.ic v(11)=0.000000
.ic v(47)=5.000000
.ic v(37)=5.000000
.ic v(31)=0.000000
.ic v(23)=0.000000
.ic v(35)=0.000000
.ic v(5)=5.000000
.ic v(41)=0.000000
.ic v(19)=0.000000
.ic v(17)=5.000000
.ic v(13)=5.000000
.ic v(43)=0.000000
.ic v(15)=0.000000
.ic v(33)=5.000000
.ic v(29)=5.000000
.ic v(21)=5.000000
.ic v(25)=5.000000

vdd 1 0 5.0
vin 49 0 pulse(0 5 0ns 2ns 0ns)
.tran 1ns 50ns
.plot tran V(5) V(49) (0,5)
***** OPTIONS *****
.OPTIONS VNTOL=0.001 ABSTOL=1.0E-8 ITL3=50 ITL4=50 LIMTIM=40 CPTIME=180
+ RELTOL=0.001 CHGTOL=1E-14 PIVTOL=1E-18
+ METHOD=GEAR MAXORD=3
.WIDTH OUT=80
*******************************************************************************
*          TYPICAL Device parameters for the HP CMOS40 Process                *
*                                                                             *
*                        Released 2/6/86 by Rich Duncombe                     *
*        NOTE:  These parameters are intended for digital design only.        *
*                                                                             *
*******************************************************************************
*
* Use N and P models for W >= 4U and L <= 2U
*
.MODEL N NMOS LEVEL=2 VTO= 0.75 KP=76.0U GAMMA=.40 LAMBDA=.025 TOX=25N
+ NSUB=4E16 TPG=+1 XJ=.25U LD=.20U UEXP=.16 VMAX=5.5E4 JS=1000U
+ CGSO=220P CGDO=220P CJ=230U CJSW=260P CGBO=400P
*
.MODEL P PMOS LEVEL=2 VTO=-0.75 KP=27.0U GAMMA=.50 LAMBDA=.045 TOX=25N
+ NSUB=2.0E16 TPG=-1 XJ=.20U LD=.05U UEXP=.15 VMAX=9.0E4 JS=1000U
```

```
+ CGSO=220P CGDO=220P CJ=670U CJSW=215P CGBO=400P
*
*******************************************************************************
.END
```

WORST CASE PATH FOR Adder66

```
*****************************************************
*                                                   *
*       Add66                                       *
*       -----                                       *
*       A 65 bit adder with Cin                     *
*       Consist of a 66 bit adder (65 bit data      *
*       and Cin)                                    *
*                                                   *
*       Delay is ____33ns_____                      *
*                                                   *
*                    7/8/86 9:30pm                  *
*                                                   *
*****************************************************

C1 47 0 0.217pf
C2 17 0 0.007pf
C3 31 0 0.111pf
C4 11 0 0.098pf
C5 55 0 0.005pf
C6 45 0 0.014pf
C7 41 0 0.602pf
C8 9 0 0.376pf
C9 29 0 0.111pf
C10 7 0 0.005pf
C11 5 0 0.098pf
C12 33 0 0.116pf
C13 49 0 0.015pf
C14 15 0 0.027pf
C15 39 0 0.014pf
C16 53 0 0.002pf
C17 35 0 0.015pf
C18 25 0 0.178pf
C19 21 0 0.015pf
C20 13 0 0.166pf
C21 51 0 0.453pf
C22 37 0 0.147pf
C23 27 0 0.135pf
C24 23 0 3.244pf
C25 43 0 0.116pf
C26 19 0 0.152pf

* add back p-device for passgate *
M41 9 13 5 1 p l=1.6u w=3.2u

M1 1 0 7 1 p l=1.6u w=6.4u
M2 7 0 9 1 p l=1.6u w=6.4u
M3 9 11 5 0 n l=1.6u w=3.2u
M4 11 13 1 1 p l=1.6u w=6.4u
M5 11 13 0 0 n l=1.6u w=3.2u
M6 15 1 13 0 n l=1.6u w=3.2u
M7 15 19 0 0 n l=1.6u w=3.2u
M8 17 19 1 1 p l=1.6u w=6.4u
M9 13 0 17 1 p l=1.6u w=6.4u
M10 1 23 19 1 p l=1.6u w=3.2u
M11 21 1 0 0 n l=1.6u w=3.2u
M12 19 23 21 0 n l=1.6u w=3.2u
M13 23 25 0 0 n l=1.6u w=12.8u
M14 23 25 1 1 p l=1.6u w=28.8u
M15 1 27 25 1 p l=1.6u w=12.8u
M16 0 27 25 0 n l=1.6u w=6.4u
M17 0 29 27 0 n l=1.6u w=3.2u
M18 1 29 27 1 p l=1.6u w=6.4u
M19 1 31 29 1 p l=1.6u w=6.4u
M20 0 31 29 0 n l=1.6u w=3.2u
```

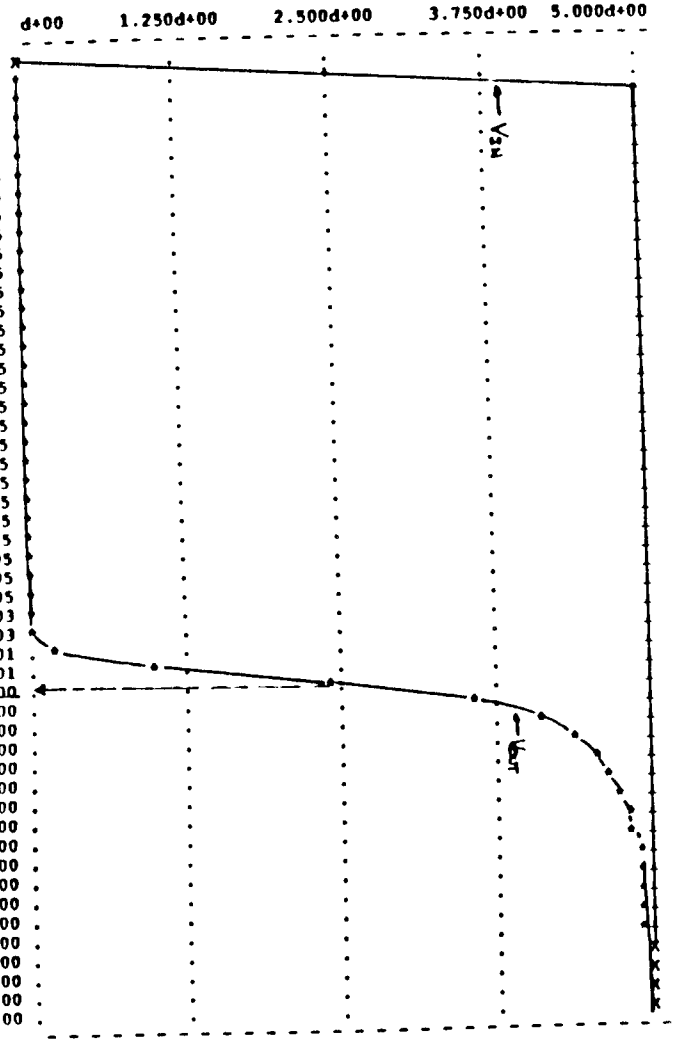
```
M21 0 33 31 0 n l=1.6u w=3.2u
M22 1 33 31 1 p l=1.6u w=6.4u
M23 33 37 1 1 p l=1.6u w=3.2u
M24 35 37 0 0 n l=1.6u w=3.2u
M25 33 1 35 0 n l=1.6u w=3.2u
M26 0 41 37 0 n l=1.6u w=3.2u
M27 39 0 1 1 p l=1.6u w=6.4u
M28 37 41 39 1 p l=1.6u w=6.4u
M29 1 43 41 1 p l=1.6u w=6.4u
M30 0 43 41 0 n l=1.6u w=3.2u
M31 0 47 43 0 n l=1.6u w=3.2u
M32 45 0 1 1 p l=1.6u w=6.4u
M33 43 47 45 1 p l=1.6u w=6.4u
M34 1 51 47 1 p l=1.6u w=3.2u
M35 49 1 0 0 n l=1.6u w=3.2u
M36 47 51 49 0 n l=1.6u w=3.2u
M37 53 1 51 0 n l=1.6u w=3.2u
M38 0 57 53 0 n l=1.6u w=3.2u
M39 1 57 55 1 p l=1.6u w=6.4u
M40 55 0 51 1 p l=1.6u w=6.4u

* Initial conditions:
.ic v(47)=0.000000
.ic v(17)=5.000000
.ic v(31)=5.000000
.ic v(11)=0.000000
.ic v(55)=5.000000
.ic v(45)=5.000000
.ic v(41)=0.000000
.ic v(9)=5.000000
.ic v(29)=0.000000
.ic v(7)=5.000000
.ic v(5)=0.000000
.ic v(33)=0.000000
.ic v(49)=0.000000
.ic v(15)=5.000000
.ic v(39)=5.000000
.ic v(53)=5.000000
.ic v(35)=0.000000
.ic v(25)=0.000000
.ic v(21)=0.000000
.ic v(13)=5.000000
.ic v(51)=5.000000
.ic v(37)=5.000000
.ic v(27)=5.000000
.ic v(23)=5.000000
.ic v(43)=5.000000
.ic v(19)=0.000000

vdd 1 0 5.0
vin 57 0 pulse(0 5 0ns 2ns 0ns)
.tran 1ns 50ns
.plot tran V(5) V(57) (0,5)
***** OPTIONS *****
.OPTIONS VNTOL=0.001 ABSTOL=1.0E-8 ITL3=50 ITL4=50 LIMTIM=40 CPTIME=180
+ RELTOL=0.001 CHGTOL=1E-14 PIVTOL=1E-18
+ METHOD=GEAR MAXORD=3
.WIDTH OUT=80
*******************************************************************************
*                                                                             *
*          TYPICAL Device parameters for the HP CMOS40 Process                *
*                                                                             *
*                                                                             *
*                        Released 2/6/86 by Rich Duncombe                     *
*        NOTE:  These parameters are intended for digital design only.        *
*                                                                             *
*******************************************************************************
*
```

```
* Use N and P models for W >= 4U and L <= 2U
*
.MODEL N NMOS LEVEL=2 VTO= 0.75 KP=76.0U GAMMA=.40 LAMBDA=.025 TOX=25N
+ NSUB=4E16 TPG=+1 XJ=.25U LD=.20U UEXP=.16 VMAX=5.5E4 JS=1000U
+ CGSO=220P CGDO=220P CJ=230U CJSW=260P CGBO=400P
*
.MODEL P PMOS LEVEL=2 VTO=-0.75 KP=27.0U GAMMA=.50 LAMBDA=.045 TOX=25N
+ NSUB=2.0E16 TPG=-1 XJ=.20U LD=.05U UEXP=.15 VMAX=9.0E4 JS=1000U
+ CGSO=220P CGDO=220P CJ=670U CJSW=215P CGBO=400P
*
*******************************************************************************
.END
```

*******08/30/87 ******** SPICE 2G.6 3/15/83 ********19:59:26*****
WORST CASE PATH FOR ADDER66 TEMPERATURE = 27.000 DEG C
**** TRANSIENT ANALYSIS

LEGEND:
*: V(5)
+: V(57)

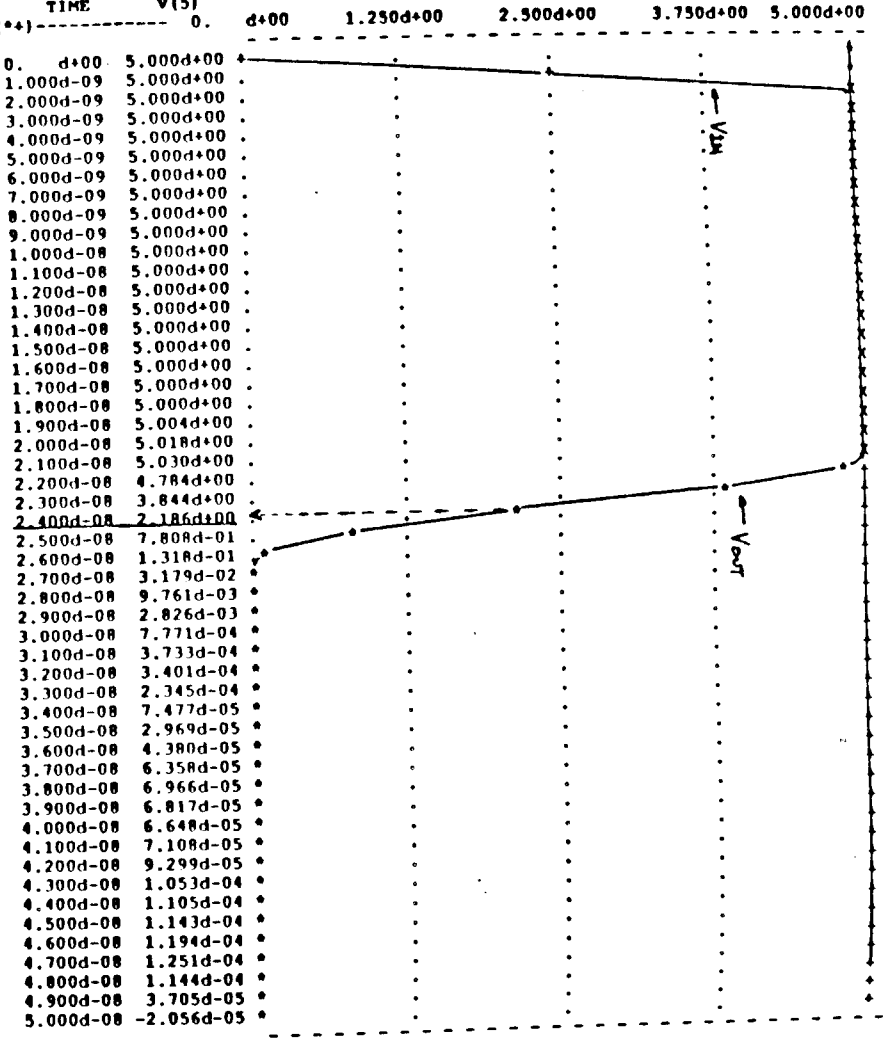
TIME	V(5)
0.	6.933d-12
1.000d-09	-1.010d-05
2.000d-09	-1.652d-05
3.000d-09	-2.535d-05
4.000d-09	-2.821d-05
5.000d-09	-2.928d-05
6.000d-09	-3.001d-05
7.000d-09	-3.055d-05
8.000d-09	-3.099d-05
9.000d-09	-3.147d-05
1.000d-08	-3.209d-05
1.100d-08	-3.312d-05
1.200d-08	-3.439d-05
1.300d-08	-3.538d-05
1.400d-08	-3.554d-05
1.500d-08	-3.573d-05
1.600d-08	-3.642d-05
1.700d-08	-3.753d-05
1.800d-08	-4.000d-05
1.900d-08	-4.204d-05
2.000d-08	-4.437d-05
2.100d-08	-4.755d-05
2.200d-08	-4.989d-05
2.300d-08	-5.699d-05
2.400d-08	-6.074d-05
2.500d-08	-4.881d-05
2.600d-08	-1.291d-05
2.700d-08	4.787d-05
2.800d-08	-2.319d-05
2.900d-08	-1.303d-03
3.000d-08	7.421d-03
3.100d-08	2.004d-01
3.200d-08	9.823d-01
3.300d-08	2.394d+00
3.400d-08	3.612d+00
3.500d-08	4.135d+00
3.600d-08	4.367d+00
3.700d-08	4.522d+00
3.800d-08	4.634d+00
3.900d-08	4.717d+00
4.000d-08	4.781d+00
4.100d-08	4.830d+00
4.200d-08	4.868d+00
4.300d-08	4.898d+00
4.400d-08	4.921d+00
4.500d-08	4.939d+00
4.600d-08	4.953d+00
4.700d-08	4.963d+00
4.800d-08	4.972d+00
4.900d-08	4.978d+00
5.000d-08	4.983d+00



*******08/30/87 ******** SPICE 2G.6 3/15/83 ********19:30:21*****
WORST CASE PATH FOR EGSUBEL TEMPERATURE = 27.000 DEG C
**** TRANSIENT ANALYSIS

LEGEND:
*: V(5)
+: V(49)

TIME	V(5)
0.	5.000d+00
1.000d-09	5.000d+00
2.000d-09	5.000d+00
3.000d-09	5.000d+00
4.000d-09	5.000d+00
5.000d-09	5.000d+00
6.000d-09	5.000d+00
7.000d-09	5.000d+00
8.000d-09	5.000d+00
9.000d-09	5.000d+00
1.000d-08	5.000d+00
1.100d-08	5.000d+00
1.200d-08	5.000d+00
1.300d-08	5.000d+00
1.400d-08	5.000d+00
1.500d-08	5.000d+00
1.600d-08	5.000d+00
1.700d-08	5.000d+00
1.800d-08	5.000d+00
1.900d-08	5.004d+00
2.000d-08	5.018d+00
2.100d-08	5.030d+00
2.200d-08	4.784d+00
2.300d-08	3.844d+00
2.400d-08	2.186d+00
2.500d-08	7.808d-01
2.600d-08	1.318d-01
2.700d-08	3.179d-02
2.800d-08	9.761d-03
2.900d-08	2.826d-03
3.000d-08	7.771d-04
3.100d-08	3.733d-04
3.200d-08	3.401d-04
3.300d-08	2.345d-04
3.400d-08	7.477d-05
3.500d-08	2.969d-05
3.600d-08	4.380d-05
3.700d-08	6.358d-05
3.800d-08	6.966d-05
3.900d-08	6.817d-05
4.000d-08	6.648d-05
4.100d-08	7.108d-05
4.200d-08	9.299d-05
4.300d-08	1.053d-04
4.400d-08	1.105d-04
4.500d-08	1.143d-04
4.600d-08	1.194d-04
4.700d-08	1.251d-04
4.800d-08	1.144d-04
4.900d-08	3.705d-05
5.000d-08	-2.056d-05



Leading One Detector for a 67 bits input Charge Sharing Simulation 6/28/87

```
*****************************************************************
*                                                               *
*  67 bits-Leading One Detector Charge Sharing Simulation       *
*  ======================================================       *
*  All N-channel input on except the bottom one.                *
*  For a 4-input AND Domino gate.                               *
*                                                               *
*  WORST CASE HIGH VOLTAGE AT PRECHARGE NODE : 3.54V            *
*  WORST CASE LOW VOLTAGE AT OUTPUT          : 0.12V            *
*                                                               *
*****************************************************************

C1 55 0 0.016pf
C5 53 0 0.017pf
C7 47 0 0.289pf
C17 45 0 1.479pf
C24 51 0 0.015pf
C25 49 0 0.013pf
M25 1 47 45 1 p l=1.6u w=24.0u
M26 0 47 45 0 n l=1.6u w=8.0u
M27 47 1 49 0 n l=1.6u w=11.2u
M28 49 1 51 0 n l=1.6u w=14.4u
M29 51 1 53 0 n l=1.6u w=17.6u
M30 53 0 55 0 n l=1.6u w=20.8u
M31 55 57 0 0 n l=1.6u w=24.0u

* Initial conditions:
.ic v(45)=0.000000
.ic v(47)=5.000000
.ic v(49)=0.000000
.ic v(51)=0.000000
.ic v(53)=0.000000
.ic v(55)=0.000000

vdd 1 0 5.0
vin 57 0 pulse(0 5 0ns 2ns 2ns 23ns 100ns)
.tran 0.50ns 25ns
.print tran V(45) V(47) V(57)
.plot tran V(45) V(47) V(57) (0,5)
***** OPTIONS *****
*
.OPTIONS VNTOL=0.01 ABSTOL=1.0E-7 ITL3=500 ITL4=500 LIMTIM=40 CPTIME=1800
+ ITL5=20000
+ RELTOL=0.01 CHGTOL=1E-14 PIVTOL=1E-18
+ METHOD=GEAR MAXORD=4
+ TRTOL=6
*
.WIDTH OUT=80
*****************************************************************
*          TYPICAL Device parameters for the HP CMOS40 Process  *
*                                                               *
*                      Released 2/6/86 by Rich Duncombe         *
*        NOTE:  These parameters are intended for digital design only.
*                                                               *
*****************************************************************
*
* Use N and P models for W >= 4U and L <= 2U
*
.MODEL N NMOS LEVEL=2 VTO= 0.75 KP=76.0U GAMMA=.40 LAMBDA=.025 TOX=25N
+ NSUB=4E16 TPG=+1 XJ=.25U LD=.20U UEXP=.16 VMAX=5.5E4 JS=1000U
+ CGSO=220P CGDO=220P CJ=230U CJSW=260P CGBO=400P
+ XQC=0.49
*
.MODEL P PMOS LEVEL=2 VTO=-0.75 KP=27.0U GAMMA=.50 LAMBDA=.045 TOX=25N
```

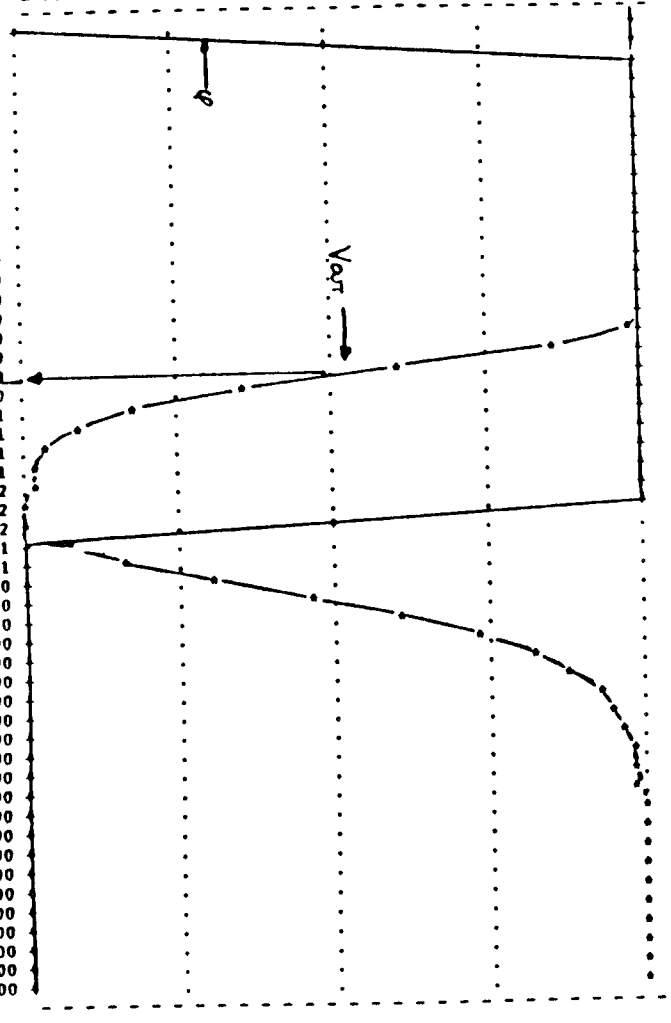
```
+ NSUB=2.0E16 TPG=-1 XJ=.20U LD=.05U UEXP=.15 VMAX=9.0E4 JS=1000U
+ CGSO=220P CGDO=220P CJ=670U CJSW=215P CGBO=400P
+ XQC=0.49
*
*********************************************************************************
.END
```

```
*****************************************************************
*                                                               *
*  67 bits-Leading One Detector Worst Case Delay Simulation     *
*  ========================================================     *
*  INPUT : 57  Clock Phil for Evaluation and Precharge          *
*  OUTPUT: 5   Output of the OR plane - O1                      *
*                                                               *
*  WORST CASE DELAY:  18ns                                      *
*                                                               *
*  PRECHARGE TIME  :  15ns                                      *
*                                                               *
*****************************************************************

C1 55 0 0.016pf
C2 41 0 0.017pf
C3 23 0 0.015pf
C4 11 0 0.356pf
C5 53 0 0.017pf
C6 27 0 0.016pf
C7 47 0 0.289pf
C8 37 0 0.013pf
C9 15 0 0.016pf
C10 31 0 0.800pf
C11 29 0 5.878pf
C12 25 0 0.017pf
C13 19 0 0.297pf
C14 33 0 0.373pf
C15 13 0 0.017pf
C16 5 0 3.988pf
C17 45 0 1.479pf
C18 39 0 0.015pf
C19 17 0 0.525pf
C20 35 0 0.297pf
C21 21 0 0.013pf
C22 43 0 0.016pf
C23 9 0 0.468pf
C24 51 0 0.015pf
C25 49 0 0.013pf
C26 59 0 4pf
M1 59 9 5 0 n l=1.6u w=52.0u
M2 1 11 9 1 p l=1.6u w=24.0u
M3 0 11 9 0 n l=1.6u w=8.0u
M4 11 1 13 0 n l=1.6u w=17.6u
M5 15 57 0 0 n l=1.6u w=24.0u
M6 13 17 15 0 n l=1.6u w=20.8u
M7 1 19 17 1 p l=1.6u w=24.0u
M8 0 19 17 0 n l=1.6u w=8.0u
M9 19 1 21 0 n l=1.6u w=11.2u
M10 21 1 23 0 n l=1.6u w=14.4u
M11 23 1 25 0 n l=1.6u w=17.6u
M12 27 57 0 0 n l=1.6u w=24.0u
M13 25 29 27 0 n l=1.6u w=20.8u
M14 29 31 1 1 p l=1.6u w=144.0u
M15 29 31 0 0 n l=1.6u w=72.0u
M16 0 33 31 0 n l=1.6u w=24.0u
M17 1 33 31 1 p l=1.6u w=48.0u
M18 1 35 33 1 p l=1.6u w=24.0u
M19 0 35 33 0 n l=1.6u w=8.0u
M20 35 1 37 0 n l=1.6u w=11.2u
M21 37 1 39 0 n l=1.6u w=14.4u
M22 39 1 41 0 n l=1.6u w=17.6u
M23 43 57 0 0 n l=1.6u w=24.0u
M24 41 45 43 0 n l=1.6u w=20.8u
M25 1 47 45 1 p l=1.6u w=24.0u
```

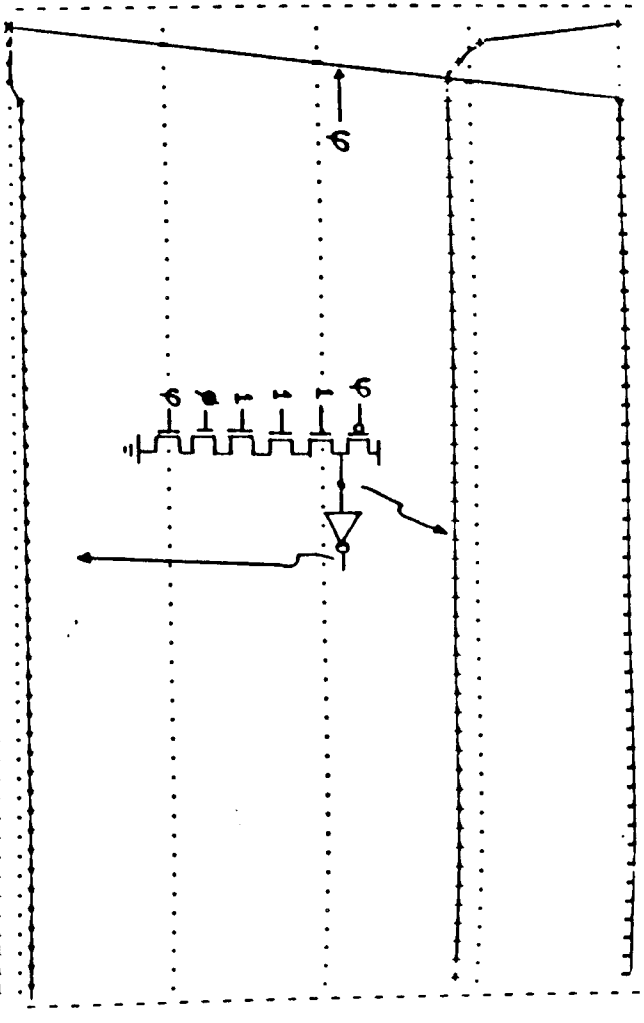
```
M26 0 47 45 0 n l=1.6u w=8.0u
M27 47 1 49 0 n l=1.6u w=11.2u
M28 49 1 51 0 n l=1.6u w=14.4u
M29 51 1 53 0 n l=1.6u w=17.6u
M30 53 1 55 0 n l=1.6u w=20.8u
M31 55 57 0 0 n l=1.6u w=24.0u
M32 47 57 1 1 p l=1.6u w=31.2u
M33 5 57 1 1 p l=1.6u w=42.4u
M34 11 57 1 1 p l=1.6u w=31.2u
M35 19 57 1 1 p l=1.6u w=31.2u
M36 35 57 1 1 p l=1.6u w=31.2u
M37 59 57 0 0 n l=1.6u w=52.0u

* Initial conditions:
.ic v(55)=5.000000
.ic v(41)=5.000000
.ic v(23)=5.000000
.ic v(11)=5.000000
.ic v(53)=5.000000
.ic v(27)=0.000000
.ic v(47)=5.000000
.ic v(37)=5.000000
.ic v(15)=0.000000
.ic v(31)=5.000000
.ic v(29)=0.000000
.ic v(25)=5.000000
.ic v(19)=5.000000
.ic v(33)=0.000000
.ic v(13)=5.000000
.ic v(5)=5.000000
.ic v(45)=0.000000
.ic v(39)=5.000000
.ic v(17)=0.000000
.ic v(35)=5.000000
.ic v(21)=5.000000
.ic v(43)=0.000000
.ic v(9)=0.000000
.ic v(51)=5.000000
.ic v(49)=5.000000
.ic v(59)=5.000000

vdd 1 0 5.0
vin 57 0 pulse(0 5 0ns 2ns 2ns 23ns 100ns)
.tran 1ns 50ns
.plot tran V(5) V(57) (0,5)
***** OPTIONS *****
*
.OPTIONS VNTOL=0.01 ABSTOL=1.0E-7 ITL3=500 ITL4=500 LIMTIM=40 CPTIME=1800
+ ITL5=20000
+ RELTOL=0.01 CHGTOL=1E-14 PIVTOL=1E-18
+ METHOD=GEAR MAXORD=4
+ TRTOL=6
*
.WIDTH OUT=80
*******************************************************************************
*                                                                             *
*          TYPICAL Device parameters for the HP CMOS40 Process                *
*                                                                             *
*                        Released 2/6/86 by Rich Duncombe                     *
*        NOTE:  These parameters are intended for digital design only.        *
*                                                                             *
*******************************************************************************
*
* Use N and P models for W >= 4U and L <= 2U
*
.MODEL N NMOS LEVEL=2 VTO= 0.75 KP=76.0U GAMMA=.40 LAMBDA=.025 TOX=25N
+ NSUB=4E16 TPG=+1 XJ=.25U LD=.20U UEXP=.16 VMAX=5.5E4 JS=1000U
```

```
+ CGSO=220P CGDO=220P CJ=230U CJSW=260P CGBO=400P
+ XQC=0.49
*
.MODEL P PMOS LEVEL=2 VTO=-0.75 KP=27.0U GAMMA=.50 LAMBDA=.045 TOX=25N
+ NSUB=2.0E16 TPG=-1 XJ=.20U LD=.05U UEXP=.15 VMAX=9.0E4 JS=1000U
+ CGSO=220P CGDO=220P CJ=670U CJSW=215P CGBO=400P
+ XQC=0.49
*
**********************************************************************
.END
```

```
*******08/30/87 ********  SPICE 2G.6    3/15/83 ********17:08:14*****
LEADING ONE DETECTOR FOR A 67 BITS INPUT WORST CASE SIMULATION 6/27/87 1:30AM
****      TRANSIENT ANALYSIS                    TEMPERATURE =   27.000 DEG C
******************************************************************************

LEGEND:
*: V(5)
+: V(57)
     TIME          V(5)
(*+)------------  0.    d+00      1.250d+00      2.500d+00      3.750d+00   5.000d+00
0.    d+00   5.000d+00
1.000d-09    5.029d+00
2.000d-09    5.051d+00
3.000d-09    5.051d+00
4.000d-09    5.051d+00
5.000d-09    5.051d+00
6.000d-09    5.051d+00
7.000d-09    5.051d+00
8.000d-09    5.051d+00
9.000d-09    5.051d+00
1.000d-08    5.051d+00
1.100d-08    5.051d+00
1.200d-08    5.051d+00
1.300d-08    5.051d+00
1.400d-08    5.051d+00
1.500d-08    5.051d+00
1.600d-08    4.932d+00
1.700d-08    4.286d+00
1.800d-08    3.019d+00
1.900d-08    1.754d+00
2.000d-08    8.861d-01
2.100d-08    4.241d-01
2.200d-08    2.017d-01
2.300d-08    1.000d-01
2.400d-08    5.500d-02
2.500d-08    3.166d-02
2.600d-08    4.246d-02
2.700d-08    3.492d-01
2.800d-08    8.401d-01
2.900d-08    1.504d+00
3.000d-08    2.303d+00
3.100d-08    3.052d+00
3.200d-08    3.653d+00
3.300d-08    4.099d+00
3.400d-08    4.407d+00
3.500d-08    4.612d+00
3.600d-08    4.746d+00
3.700d-08    4.834d+00
3.800d-08    4.891d+00
3.900d-08    4.929d+00
4.000d-08    4.953d+00
4.100d-08    4.969d+00
4.200d-08    4.980d+00
4.300d-08    4.987d+00
4.400d-08    4.991d+00
4.500d-08    4.994d+00
4.600d-08    4.996d+00
4.700d-08    4.998d+00
4.800d-08    4.999d+00
4.900d-08    4.999d+00
5.000d-08    5.000d+00
```




```
*******06/28/87 ********  SPICE 2G.6    3/15/83 ********04:37:39*****
LEADING ONE DETECTOR FOR A 67 BITS INPUT CHARGE SHARING SIMULATION 6/28/87
****      TRANSIENT ANALYSIS                    TEMPERATURE =   27.000 DEG C
******************************************************************************

LEGEND:
*: V(45)
+: V(47)
=: V(57)
     TIME          V(45)
(*+=)----------  0.    d+00      1.250d+00      2.500d+00      3.750d+00   5.000d+00
0.    d+00   6.962d-12
5.000d-10   -9.217d-03
1.000d-09    5.856d-03
1.500d-09    2.718d-02
2.000d-09    4.727d-02
2.500d-09    6.275d-02
3.000d-09    7.526d-02
3.500d-09    8.502d-02
4.000d-09    9.286d-02
4.500d-09    9.908d-02
5.000d-09    1.040d-01
5.500d-09    1.079d-01
6.000d-09    1.111d-01
6.500d-09    1.136d-01
7.000d-09    1.156d-01
7.500d-09    1.172d-01
8.000d-09    1.184d-01
8.500d-09    1.194d-01
9.000d-09    1.202d-01
9.500d-09    1.208d-01
1.000d-08    1.213d-01
1.050d-08    1.218d-01
1.100d-08    1.221d-01
1.150d-08    1.223d-01
1.200d-08    1.224d-01
1.250d-08    1.226d-01
1.300d-08    1.227d-01
1.350d-08    1.228d-01
1.400d-08    1.229d-01
1.450d-08    1.229d-01
1.500d-08    1.230d-01
1.550d-08    1.230d-01
1.600d-08    1.230d-01
1.650d-08    1.230d-01
1.700d-08    1.230d-01
1.750d-08    1.230d-01
1.800d-08    1.230d-01
1.850d-08    1.229d-01
1.900d-08    1.229d-01
1.950d-08    1.228d-01
2.000d-08    1.228d-01
2.050d-08    1.228d-01
2.100d-08    1.227d-01
2.150d-08    1.226d-01
2.200d-08    1.226d-01
2.250d-08    1.226d-01
2.300d-08    1.225d-01
2.350d-08    1.225d-01
2.400d-08    1.225d-01
2.450d-08    1.225d-01
2.500d-08    1.224d-01
```



Shifter Worst Case Delay ANalysis 6/29/87 6:30am

```
*************************************************************
*                                                           *
*  Shifter Worst Case Delay ANalysis                        *
*  =================================                        *
*  Input : 67 - LSHI (assume stable = HIGH)                 *
*          63 - CFshren                                     *
*          65 - CFshfen                                     *
*           5 - Shf (use results of Decoder simulation)     *
*                                                           *
* Output:  59 - Lshfout                                     *
*                                                           *
*  Delay     Time : ____15ns____                            *
*  Precharge Time : ____10ns____                            *
*                                                           *
*************************************************************

*************************************
* Obtained from Sdec.spice for Shf
C1 9 0 0.099pf
C2 5 0 4.354pf
C3 7 0 0.393pf
C4 17 0 0.003pf
C5 13 0 3.916pf
C6 21 0 6.263pf
C7 15 0 0.312pf
C8 19 0 0.485pf
C9 25 0 0.500pf
M1 5 7 1 1 p l=1.6u w=120.0u
M2 5 7 0 0 n l=1.6u w=24.0u
M3 7 9 0 0 n l=1.6u w=8.0u
M4 7 9 1 1 p l=1.6u w=8.0u
M5 9 13 1 1 p l=1.6u w=8.0u
M6 9 13 0 0 n l=1.6u w=3.2u
M7 13 15 0 0 n l=1.6u w=40.0u
M8 13 15 1 1 p l=1.6u w=40.0u
M9 15 19 1 1 p l=1.6u w=16.8u
M10 0 25 17 0 n l=1.6u w=6.4u
M11 17 19 15 0 n l=1.6u w=6.4u
M12 19 1 21 0 n l=1.6u w=39.2u
M13 21 23 0 0 n l=1.6u w=816.0u
M14 15 25 1 1 p l=1.6u w=16.8u
M15 25 23 1 1 p l=1.6u w=16u
M16 19 23 1 1 p l=1.6u w=16u
*************************************

C51 55 0 1.612pf
C52 57 0 0.01pf
C53 59 0 1.612pf
C54 61 0 0.001pf
M51 55 1 57 0 n l=1.6u w=19.2u
M52 57 63 0 0 n l=1.6u w=19.2u
M53 55 65 1 1 p l=1.6u w=24u
M54 55 5 59 0 n l=1.6u w=4.8u
M55 59 65 1 1 p l=1.6u w=24u
M56 59 1 61 0 n l=1.6u w=20u

* Initial conditions:
.ic v(9)=0.000000
.ic v(5)=0.000000
.ic v(7)=5.000000
.ic v(17)=0.000000
.ic v(13)=5.000000
.ic v(21)=5.000000
.ic v(15)=0.000000
```

```
.ic v(19)=5.000000
.ic v(25)=5.000000


.ic v(55)=5.000000
.ic v(57)=5.000000
.ic v(59)=5.000000
.ic v(61)=5.000000


vdd 1 0 5.0
vCFshfen 65 0 pulse(0 5 0ns 2ns 2ns 23ns 100ns)
vCFshren 63 0 pulse(0 5 0ns 2ns 2ns 23ns 100ns)
vin 23 0 pulse(0 5 0ns 2ns 2ns 23ns 100ns)
.tran 0.50ns 70ns
.plot tran V(59) V(5) V(23) (0,5)
***** OPTIONS *****
.OPTIONS VNTOL=0.001 ABSTOL=1.0E-8 ITL3=50 ITL4=50 LIMTIM=40 CPTIME=1800
+ RELTOL=0.01 CHGTOL=1E-14 PIVTOL=1E-18
+ METHOD=GEAR MAXORD=3
.WIDTH OUT=80
*******************************************************************************
*          TYPICAL Device parameters for the HP CMOS40 Process                *
*                                                                             *
*                        Released 2/6/86 by Rich Duncombe                     *
*       NOTE:  These parameters are intended for digital design only.         *
*                                                                             *
*******************************************************************************
*
* Use N and P models for W >= 4U and L <= 2U
*
.MODEL N NMOS LEVEL=2 VTO= 0.75 KP=76.0U GAMMA=.40 LAMBDA=.025 TOX=25N
+ NSUB=4E16 TPG=+1 XJ=.25U LD=.20U UEXP=.16 VMAX=5.5E4 JS=1000U
+ CGSO=220P CGDO=220P CJ=230U CJSW=260P CGBO=400P
*
.MODEL P PMOS LEVEL=2 VTO=-0.75 KP=27.0U GAMMA=.50 LAMBDA=.045 TOX=25N
+ NSUB=2.0E16 TPG=-1 XJ=.20U LD=.05U UEXP=.15 VMAX=9.0E4 JS=1000U
+ CGSO=220P CGDO=220P CJ=670U CJSW=215P CGBO=400P
*
*******************************************************************************
.END
```

```
Sticky Bit Worst Case Delay  7/9/87  12:11pm

**************************************************
*                                                *
*   Sticky Bit Worst Case Generation             *
*   ================================             *
*   Simulated till output latched into output    *
*   latch.                                       *
*                                                *
*   Delay Time: ____25ns_____                    *
*   Precharge Time: ___25ns_____                 *
*                                                *
**************************************************

C1 17 0 5.432pf
C2 5 0 5.432pf
C3 13 0 2.202pf
C4 15 0 0.007pf
C5 11 0 0.133pf
C6 9 0 0.150pf
C7 7 0 0.017pf
C8 19 0 1.9pf
C9 25 0 0.01pf
C10 23 0 0.15pf

C11 31 0 0.2pf
C12 33 0 0.1pf
C13 35 0 0.1pf
C14 39 0 0.1pf

M1 7 27 0 0 n l=1.6u w=31.2u
M2 5 9 7 0 n l=1.6u w=31.2u
M3 11 23 9 1 p l=1.6u w=14.4u
M4 11 13 1 1 p l=1.6u w=14.4u
M5 11 13 0 0 n l=1.6u w=3.2u
M6 15 27 0 0 n l=1.6u w=6.4u
M7 13 17 15 0 n l=1.6u w=6.4u
M8 1 19 17 1 p l=1.6u w=64u
M9 0 19 17 0 n l=1.6u w=32u
M10 1 27 5 1 p l=1.6u w=24u
M11 9 21 11 0 n l=1.6u w=3.2u
M12 1 21 23 1 p l=1.6u w=6.4u
M13 0 21 23 0 n l=1.6u w=3.2u
M14 1 27 13 1 p l=1.6u w=20u
M15 1 27 19 1 p l=1.6u w=24u
M16 19 29 25 0 n l=1.6u w=35.2u
M17 0 27 25 0 n l=1.6u w=35.2u

M19 33 5 31 1 p l=1.6u w=12.8u
M20 33 37 1 1 p l=1.6u w=12.8u
M21 35 5 31 0 n l=1.6u w=6.4u
M22 35 27 0 0 n l=1.6u w=6.4u
M23 1 31 39 1 p l=1.6u w=6.4u
M24 0 31 39 0 n l=1.6u w=3.2u

.subckt ShfDec 1 5 23

C1 9 0 0.099pf
C2 5 0 4.354pf
C3 7 0 0.393pf
C4 17 0 0.003pf
C5 13 0 3.916pf
C6 21 0 6.263pf
C7 15 0 0.312pf
C8 19 0 0.485pf
C9 25 0 0.500pf
```

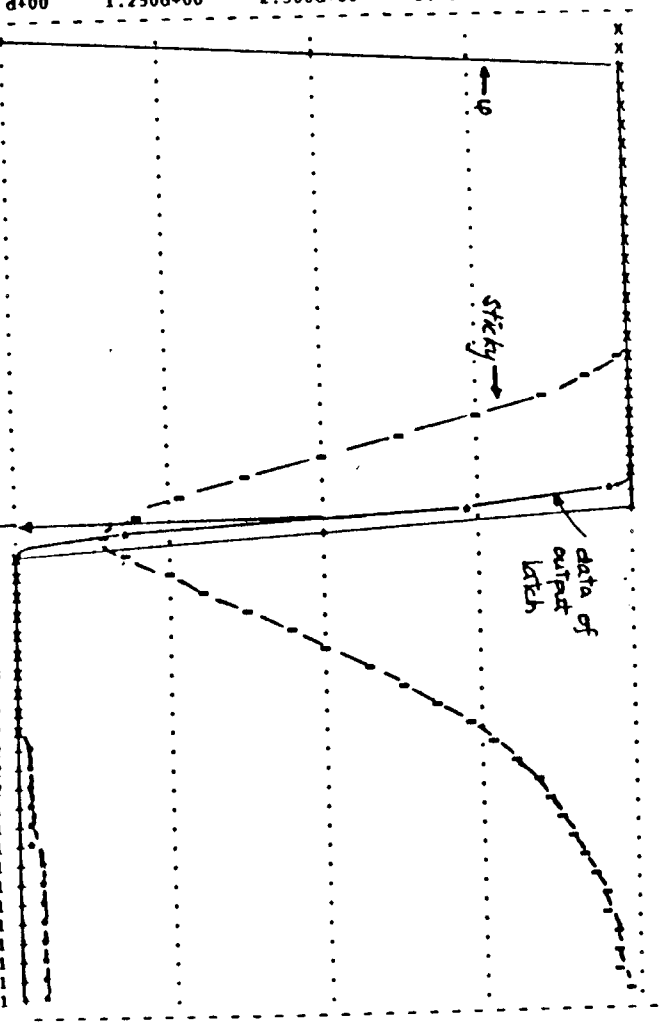
```
M1 5 7 1 1 p l=1.6u w=120.0u
M2 5 7 0 0 n l=1.6u w=24.0u
M3 7 9 0 0 n l=1.6u w=8.0u
M4 7 9 1 1 p l=1.6u w=8.0u
M5 9 13 1 1 p l=1.6u w=8.0u
M6 9 13 0 0 n l=1.6u w=3.2u
M7 13 15 0 0 n l=1.6u w=40.0u
M8 13 15 1 1 p l=1.6u w=40.0u
M9 15 19 1 1 p l=1.6u w=16.8u
M10 0 25 17 0 n l=1.6u w=6.4u
M11 17 19 15 0 n l=1.6u w=6.4u
M12 19 1 21 0 n l=1.6u w=39.2u
M13 21 23 0 0 n l=1.6u w=816.0u
M14 15 25 1 1 p l=1.6u w=16.8u
M15 25 23 1 1 p l=1.6u w=16u
M16 19 23 1 1 p l=1.6u w=16u

* Initial conditions:
.ic v(9)=0.000000
.ic v(5)=0.000000
.ic v(7)=5.000000
.ic v(17)=0.000000
.ic v(13)=5.000000
.ic v(21)=5.000000
.ic v(15)=0.000000
.ic v(19)=5.000000
.ic v(25)=5.000000

.ends shfdec

X1 1 21 27 ShfDec

* initial conditions:
.ic V(17) = 0.000000
.ic V(5) = 5.000000
.ic V(13) = 5.000000
.ic V(15) = 0.000000
.ic V(11) = 0.000000
.ic V(9) = 0.000000
.ic V(7) = 0.000000
.ic V(19) = 5.000000
.ic V(21) = 0.000000
.ic V(23) = 5.000000
.ic V(25) = 5.000000

.ic V(31) = 0.000000

Vdd 1 0 5.0
Vin 29 0 pulse(0 5 0ns 2ns 2ns 23ns 100ns)
Vphi 27 0 pulse(0 5 0ns 2ns 2ns 23ns 100ns)
Vphibar 37 0 pulse(5 0 0ns 2ns 2ns 23ns 100ns)
.tran 1ns 50ns
.plot tran V(39) V(29) V(5) (0,5)
***** OPTIONS *****
.OPTIONS VNTOL=0.001 ABSTOL=1.0E-8 ITL3=50 ITL4=50 LIMTIM=40 CPTIME=1800
+ RELTOL=0.01 CHGTOL=1E-14 PIVTOL=1E-18
+ METHOD=GEAR MAXORD=3
.WIDTH OUT=80
*******************************************************************************
*          TYPICAL Device parameters for the HP CMOS40 Process                *
*                                                                             *
*                        Released 2/6/86 by Rich Duncombe                     *
*        NOTE:  These parameters are intended for digital design only.        *
*                                                                             *
*******************************************************************************
*
```

```
* Use N and P models for W >= 4U and L <= 2U
*
.MODEL N NMOS LEVEL=2 VTO= 0.75 KP=76.0U GAMMA=.40 LAMBDA=.025 TOX=25N
+ NSUB=4E16 TPG=+1 XJ=.25U LD=.20U UEXP=.16 VMAX=5.5E4 JS=1000U
+ CGSO=220P CGDO=220P CJ=230U CJSW=260P CGBO=400P
*
.MODEL P PMOS LEVEL=2 VTO=-0.75 KP=27.0U GAMMA=.50 LAMBDA=.045 TOX=25N
+ NSUB=2.0E16 TPG=-1 XJ=.20U LD=.05U UEXP=.15 VMAX=9.0E4 JS=1000U
+ CGSO=220P CGDO=220P CJ=670U CJSW=215P CGBO=400P
*
*******************************************************************************
.END
```

*******08/30/87 ******** SPICE 2G.6 3/15/83 ********16:49:26*****
STICKY BIT WORST CASE DELAY 7/9/87 12:11PM
**** TRANSIENT ANALYSIS TEMPERATURE = 27.000 DEG C

LEGEND:
*: V(39)
+: V(29)
=: V(5)

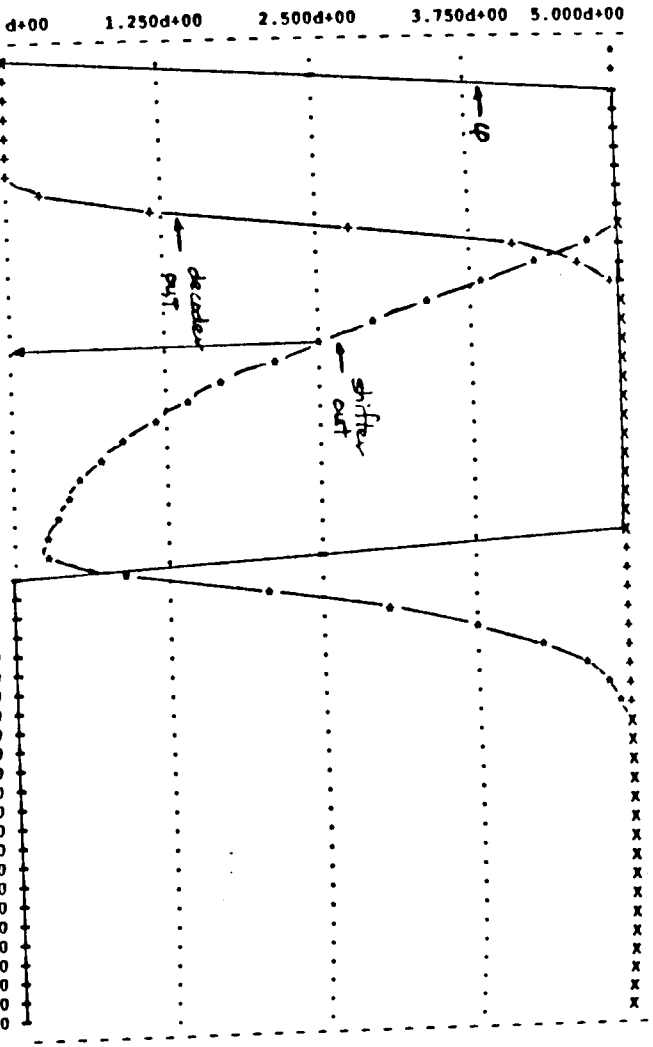
TIME	V(39)
0.	5.000d+00
1.000d-09	5.002d+00
2.000d-09	5.000d+00
3.000d-09	5.000d+00
4.000d-09	5.000d+00
5.000d-09	5.000d+00
6.000d-09	5.000d+00
7.000d-09	5.000d+00
8.000d-09	5.000d+00
9.000d-09	5.000d+00
1.000d-08	5.000d+00
1.100d-08	5.000d+00
1.200d-08	5.000d+00
1.300d-08	5.000d+00
1.400d-08	5.000d+00
1.500d-08	5.000d+00
1.600d-08	5.000d+00
1.700d-08	5.000d+00
1.800d-08	5.000d+00
1.900d-08	5.000d+00
2.000d-08	5.001d+00
2.100d-08	5.003d+00
2.200d-08	5.008d+00
2.300d-08	5.009d+00
2.400d-08	4.798d+00
2.500d-08	3.675d+00
2.600d-08	9.358d-01
2.700d-08	-7.605d-04
2.800d-08	-2.104d-04
2.900d-08	-9.550d-05
3.000d-08	9.541d-04
3.100d-08	2.785d-03
3.200d-08	5.623d-03
3.300d-08	9.789d-03
3.400d-08	1.659d-02
3.500d-08	2.696d-02
3.600d-08	4.027d-02
3.700d-08	5.536d-02
3.800d-08	7.132d-02
3.900d-08	8.753d-02
4.000d-08	1.036d-01
4.100d-08	1.188d-01
4.200d-08	1.325d-01
4.300d-08	1.422d-01
4.400d-08	1.472d-01
4.500d-08	1.485d-01
4.600d-08	1.482d-01
4.700d-08	1.477d-01
4.800d-08	1.473d-01
4.900d-08	1.469d-01
5.000d-08	1.468d-01



*******08/30/87 ******** SPICE 2G.6 3/15/83 ********16:55:27*****
SHIFTER WORST CASE DELAY ANALYSIS 6/29/87 6:30AM
**** TRANSIENT ANALYSIS TEMPERATURE = 27.000 DEG C

LEGEND:
*: V(59)
+: V(5)
=: V(23)

TIME	V(59)
0.	5.000d+00
1.000d-09	5.038d+00
2.000d-09	5.067d+00
3.000d-09	5.067d+00
4.000d-09	5.067d+00
5.000d-09	5.067d+00
6.000d-09	5.067d+00
7.000d-09	5.067d+00
8.000d-09	5.056d+00
9.000d-09	4.965d+00
1.000d-08	4.689d+00
1.100d-08	4.280d+00
1.200d-08	3.839d+00
1.300d-08	3.396d+00
1.400d-08	2.959d+00
1.500d-08	2.528d+00
1.600d-08	2.104d+00
1.700d-08	1.721d+00
1.800d-08	1.394d+00
1.900d-08	1.123d+00
2.000d-08	9.001d-01
2.100d-08	7.186d-01
2.200d-08	5.713d-01
2.300d-08	4.528d-01
2.400d-08	3.580d-01
2.500d-08	2.810d-01
2.600d-08	3.014d-01
2.700d-08	9.353d-01
2.800d-08	2.042d+00
2.900d-08	3.029d+00
3.000d-08	3.787d+00
3.100d-08	4.304d+00
3.200d-08	4.624d+00
3.300d-08	4.808d+00
3.400d-08	4.907d+00
3.500d-08	4.958d+00
3.600d-08	4.982d+00
3.700d-08	4.993d+00
3.800d-08	4.998d+00
3.900d-08	4.999d+00
4.000d-08	5.000d+00
4.100d-08	5.000d+00
4.200d-08	5.000d+00
4.300d-08	5.000d+00
4.400d-08	5.000d+00
4.500d-08	5.000d+00
4.600d-08	5.000d+00
4.700d-08	5.000d+00
4.800d-08	5.000d+00
4.900d-08	5.000d+00
5.000d-08	5.000d+00



REFERENCE

[Bose86] B. K. Bose, "Datapath Description of the SPUR Floating-Point Unit", Notes, Januray 1986

[Bose87] B. K. Bose, "Fast Multiply and Divide for a VLSI Floating-Point Unit", Proceeding of Eighth International Symposium on Computer Arithmetic, May 1987

[BrK82] R. P. Brent, H. T. Kung, "A Regular Layout for Parallel Adders", IEEE Trans. on Computers, vol. C-31, no. 3, March 1982.

[GoMa82] N. F. Goncalvs, H. J. DeMan, "NORA: A Racefree Dynamic CMOS Technique for Pipelined Logic Structures", IEEE Journal of Solid-Sate Circuits, June 1983 pp.261-266

[Hill86] M. D. Hill et al, "SPUR - A Multiprocessor Workstation", IEEE Computer, November 1986

[KrLL82] R. H. Krambeck, C. M. Lee and H-F. S. Law, "High-Speed Compact Circuits with CMOS", IEEE Journal of Solid-State Circuits, June 1982 pp.614-619

[LaFi80] R. E. ladner, M. J. Fischer, "Parallel Prefix Computation", JACM, vol.27, no.4, pp.831-838, October, 1980.

[Lee86] Corinna Lee, "Description of the SPUR Floating-Point Unit", Notes, October 1986

[Shoji82] M. Shoji, "Electrical Design of the BELLMAC-82A Microprocessor", Proc. ICCC, New York, January 1982 pp.112-115

[Shoji85] M. Shoji, "FET Scaling in Domino CMOS Gates"A, IEEE Journal of Solid-State Circuits, October 1985 pp.1067-1071

[Walt86] S. S. Walter et al, "1986 VLSI Tools: Still More Works by the Original Artists", Berkeley Technical Report No. UCB/CSD 86/272, December 1985

[Wei85] B. W-Y. Wei, C. Thompson, Y-F. Chen, "Time-Optimal Design of a CMOS Adder", Nineteenth Annual Asilomar Conference, 1985.

BIBLIOGRAPHY

[1] Cavanagh, J. J. F. *Digital Computer Arithmetic : Design and Implementation* McGram Hill, 1984

[2] Glasser, L. A. & Dobberpuhl D. W. *The Design and Analysis of VLSI Circuits* Addison-Wesley, 1985

[3] Hayes, J. P. *Computer Architecture and Organization* McGraw Hill, 1979

[4] Hodges, D. A. & Jackson, H. G., *Analysis and Design of Digital Integrated Circuits* McGraw Hill, 1983

[5] IEEE, *IEEE Standard 754 for Binary Floating-Point Arithmetic* IEEE, 1985

[6] Katz, R. H., *Proceedings of CS292i: Implementation of VLSI System (Spring 1985)* UCB/CSD, 1985

[7] Mano, M. M., *Computer System Architecture*, Prentice-Hall, 1982

[8] Mead, C. & Conway, L., *Introduction to VLSI systems* Addison Wesley, 1980

[9] Weste, N. and Eshraghian K. *Principle of CMOS VLSI Design :A System Perspective* Addison Wesley, 1985