USC-SIPI REPORT #194

VLSI Programmable Processor Design

Edited by

Bing J. Sheu and Min Chen

January 1992

Signal and Image Processing Institute
UNIVERSITY OF SOUTHERN CALIFORNIA
Department of Electrical Engineering-Systems
Electrical Engineering Building
University Park/MC-2564
Los Angeles, CA 90089 U.S.A.

VLSI Programmable Processor Design

Edited by Bing J. Sheu, Ph.D. and Min Chen

Table of Contents

(EE577 Computer Engineering)
(Term Projects, Fall Semester, 1991)

EE 577

TERM PROJECT

PART 1

SUBJECT:
A 32-BIT ADDER FOR THE HIGH PERFORMANCE PROCESS ELEMENT

NAME:
WENLIANG HONNG

ID #: **246-59-6718**

ABSTRACT:
By combining the technologies of Complementary Pass-Transistor Logic(CPL) and CMOS Logic, the response time of the 32-bit carry lookahead adder can be reduced to 2.35 ns. At the same time, the constant height can be maintained under 200 λ.

I. INTRODUCTION

The project is to design a 32-bit high performance adder as a component of a process element. The requirement of the adder includes: constant height ≤ 200 λ (λ= 0.6 μm), total delay time ≤ 4 ns, and the application of 0.5 μm technology. A new family of advanced differential CMOS logic, called complementary pass-transistor logic (CPL), is proposed and fully utilized on some critical paths to achieve very high speeds.

The concept of CPL is introduced in Section II, the CPL implementation of the adder is described in Section III, and the performance analysis of the adder is shown in Section IV. Conclusion is presented in Section V.

II. THE CONCEPT OF CPL

The main concept behind CPL[1] is the use of an nMOS pass-transistor network for logic organization, and elimination of the pMOS latch. CPL consists of complementary inputs/outputs, an nMOS pass transistor logic network, and CMOS output inverters. The pass transistors function as pull-down and pull-up devices. Thus the pMOS latch can be eliminated, allowing the advantage of the differential circuits to be fully utilized. Because the high level of the pass-transistor outputs is lower than the supply voltage level by the threshold voltage of the pass transistors, the signals have to be amplified by the output inverters. At the same time, the CMOS output inverters shift the logic threshold voltage and the drive the capacitive load. The logic threshold shift is necessary because the logic threshold voltage of the output inverter is lower than half the supply voltage, due to the lowering of the high signal level.

One attractive feature of CPL is that the complementary outputs are produced by the simple four-transistor circuits. Because inverters are unnecessary in CPL circuits, the number of critical-path gate stages can be reduced.

III.THE IMPLEMENTATION OF THE ADDER

The adder is implemented by applying both the CPL concept and the traditional CMOS technology. In the case of a 4-bit adder, as shown in Figure 1, the first three carry signals(C1, C2, C3) are produced by a 3-bit CLA adder using the traditional CMOS technology, and the last carry signal(C4) is produced by the CPL circuit. C4 and C4b are then used as the input-carry signals to the higher 4-bit adder. All of the generate signals, propagate signals, and the sum terms are implemented by using traditional CMOS technology.

The mathematical equations relevant to the 4-bit adder are shown as follows:
The carry of the ith stage, C_i, may be expressed as

$$C_i = G_i + P_i \cdot C_{i-1},$$

where

$G_i = A_i \cdot B_i$ generate signal

$P_i = A_i + B_i$ propagate signal

The sum Si is generated by

$$S_i = C_{i-1} \oplus A_i \oplus B_i$$
$$= C_{i-1} \oplus P_i$$

The four carry terms are expressed as

$$C_1 = G_1 + P_1C_0$$
$$C_2 = G_2 + P_2G_1 + P_2P_1C_0$$
$$C_3 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1C_0$$
$$C_4 = G_4 + P_4G_3 + P_4P_3G_2 + P_4P_3P_2G_1 + P_4P_3P_2P_1C_0$$

The implementation of the 3-bit CLA adder is shown in Figure 2. The physical layout of Figure 1 is shown in Figure 3. Figure 4 is the enlarged version of Figure 3. The complete schematic of the 32-bit CLA adder applying CPL technology is shown in Figure 5.

IV. PERFORMANCE ANALYSES

A 4-bit adder, as shown in Figure 1, is used in the performance analysis. The area of the 4-bit adder is 532 x 198 λ^2. The response time from C0 to C4 is 0.15 ns The response time of the last sum term (S4) is 1.15 ns. Thus, the estimated response time for the 32-bit adder is equal to 0.15 ns x 8 + 1.15 ns, which is 2.35 ns. The total area can be evaluated directly from Figure 5 and is equal to 4249 x 198 λ^2. The Spice-output diagrams for C4 and S4 are shown in Figure 6 and Figure 7, respectively. Figure 8 is the IRSIM diagram for the 4-bit adder. Figure 9 is the Spice file of the 4-bit adder.

Table 1 is a comparison between the 32-bit adder implemented in this project and a 32-bit ripple-carry adder using the schematic given in homework 3. The results show the response time is reduced from 31.26 ns to 2.35 ns, and the total area is reduced from 8524 x 168 λ^2 to 4249 x 198 λ^2.

Table 1: Comparison of Adders

Type of Adder	Response T C0-C32	Response T S4	Total Response T	Total Area
32-bit CLA/CPL Adder	1.2 0 ns	1.15 ns	2.35 ns	4249x198 λ^2
32-bit Ripple-carry Adder	26.80 ns	4.46 ns	31.26 ns	8524x168 λ^2

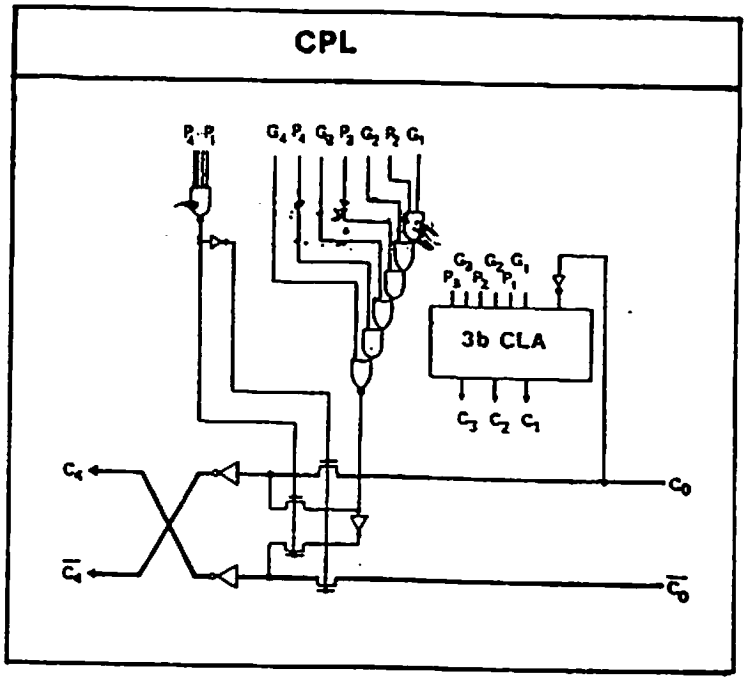
V. CONCLUSIONS

Based on the results shown in previous section, we can see the adder using CLA design has much better performance and less area than the one using ripple-carry design. By applying CPL circuits in the traditional CLA design, the adder can perform even better. Since the CPL circuit is not fully utilized in the 32-bit adder presented in this paper, a further improvement of the adder's response time is expected.

VI. REFERENCES

1.K. Yano, T. Yamanaka, T. Nishida, M. Saito, K. Shimohigashi, and A. Shimizu, "A 3.8-ns CMOS 16x16-b Multiplier Using Complementary Pass-Transistor Logic," IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 25, No. 2, April 1990. pp.388-395.

Figure 1: The Logic Design of 4-Bit CLA Adder Using CPL Circuit



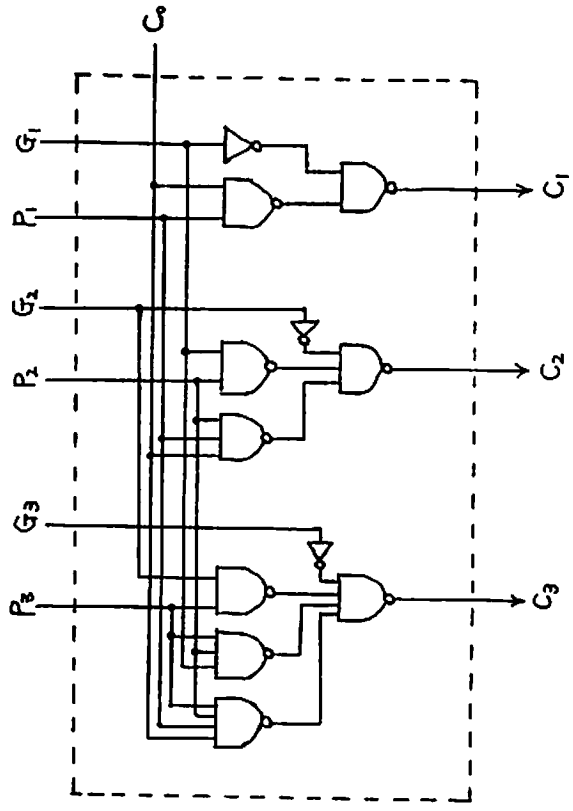
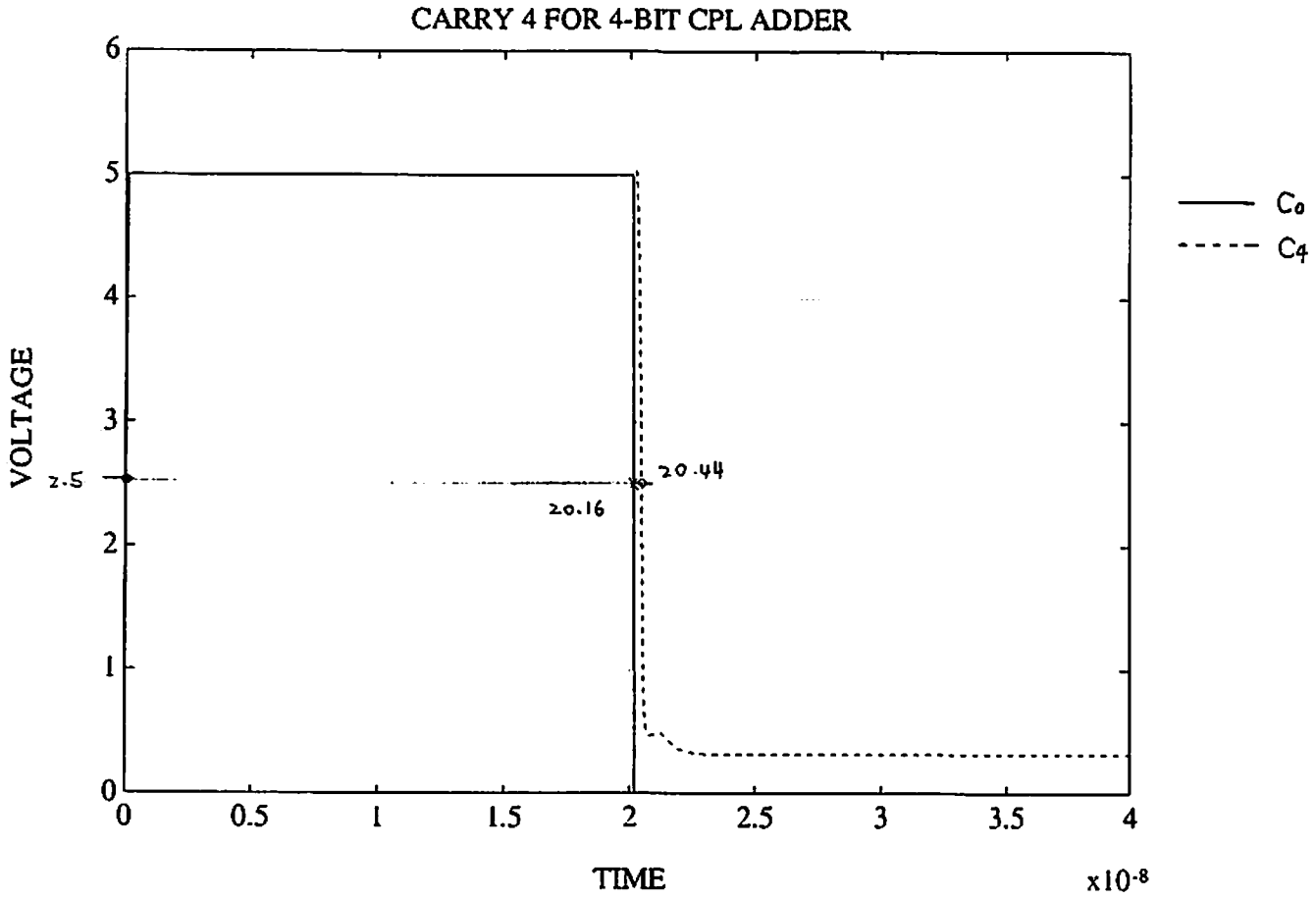
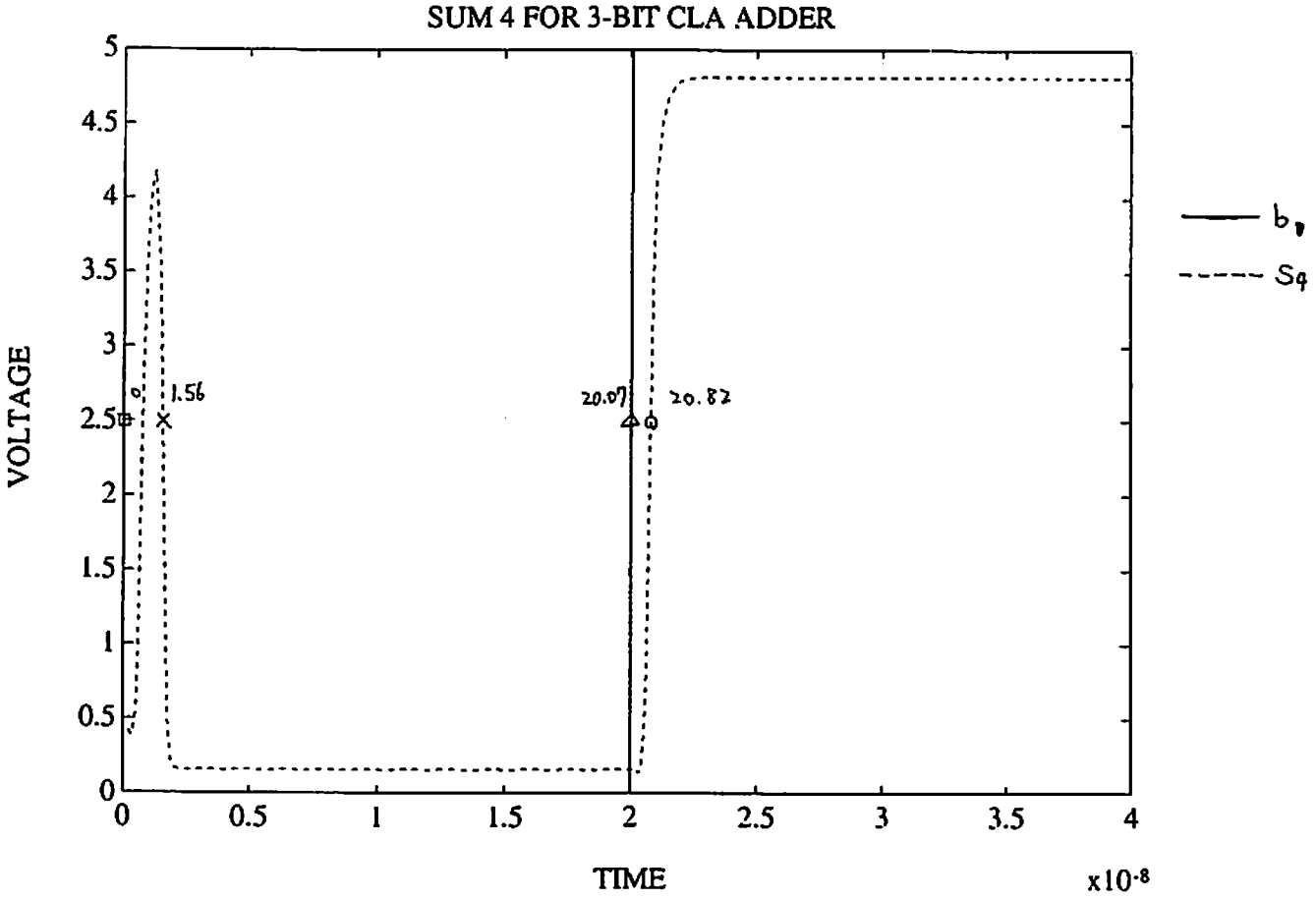


Figure 2: The Logic Design of 3-Bit CLA Adder



C_4 delay $= \frac{0 + (20.45 - 20.15)}{2} = 0.15$ (ns)

Figure 6: The Spice Diagram of C0 and C4



$$S_4 \text{ delay} = \frac{(156.0) + (20.82 - 20.07)}{2} = 1.15 \text{ (ns)}$$

Figure 7: The Spice Diagram of B1 and S4

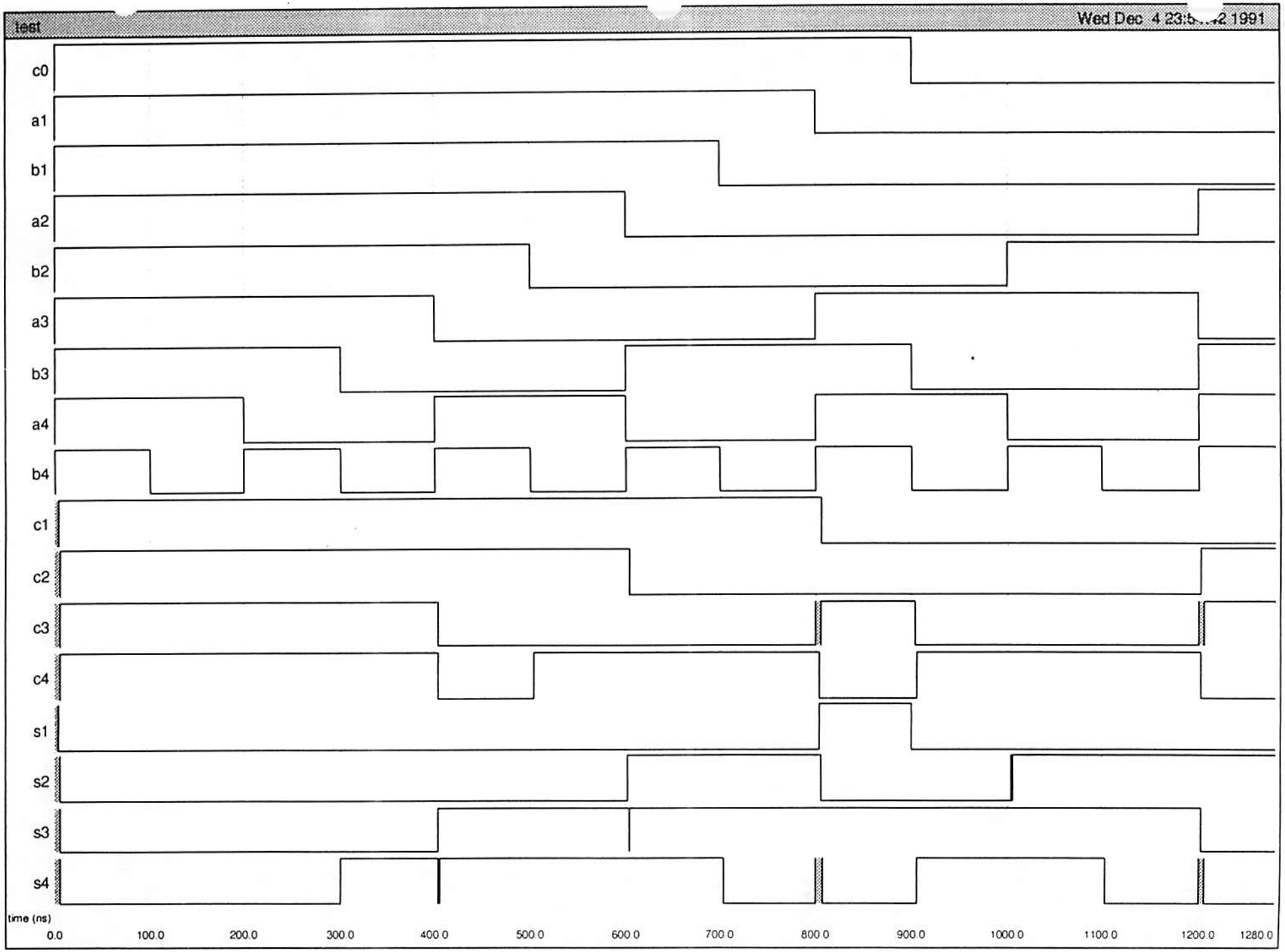


Figure 8: The IRSIM Logic Diagram for The 4-Bit CLA Adder

EE577 VLSI System Design
Term Project Report
by Chi-tat CHENG
(618-38-3116)

Instructor : Prof. Bing Sheu

A 16bit x 16bit high performance multiplier using Complementary Pass-Transistor Logic

Abstract - A 4.6ns 0.5μm 16bit by 16bit high performance multiplier is designed using Complementary Pass Transistor Logic.. Transistor count is reduced by 17.5% compared with CMOS implementation. Area of Wallace Tree is reduced by 46.6% by implementing it on a 14x32 full adder grid. A modified Binary CLA adder is used which reduces the adder depth from 8 to 5.

Introduction

The design of the multiplier centers around three issues. They are the optimization of the CPL logic, the design of a 16-to-2 Wallace Tree and a 32-bit Binary Carry-look-ahead Adder. While the Wallace Tree occupies 86.3% of the total area, it accounts for only 46.5% of the total delay. Therefore, the design aims at reducing the cell area and interconnection area of the Wallace Tree while reducing the delay through the Binary CLA.

Simulations on CPL Delay Optimizations

With minimum transistor sizing ($3\lambda/2\lambda$ for the pull-up transistor), the delay of the CPL logic depends mainly on three factors. They are the size of the n-pass-transistor, the ratio between the n and p transistors of the inverter buffer and their absolute sizes. Extensive simulations were performed which dealt with these factors in sequence. In making simulations, loading stages were explicitly connected to the outputs in the spice file. This presumably would offer a more accurate result than using capacitance estimation. In converting magic layouts to spice files, since the 0.5um extraction file was not available, a 1.2um extraction file was used instead and the 2-D scale down was performed on the spice file obtained.

Generally speaking, increasing the pass transistor size would decrease the channel resistance and hence reduce the rise and fall time. However, excessive increase in this size would introduce a large capacitive loading which might over-compensate the advantage offered by the p-type cross coupled pull up pairs. As a result, there exists optimal sizes for the transistors which lead to minimum delay. Simulation results show that minimal delay is attained with pass transistor size of 9λ / 2λ and p and n transistor of inverter buffer both of size 12λ / 12λ. It is worth noting that the n-p channel width ratio of the inverter is not at the conventional value of 1:2. It is because the n-type pass transistor would degrade the logic 1 input value and hence the logic switching threshold of the inverter must be smaller than 2.5V (1/2 Vdd). This is done by increasing the n-p ratio up to1:1.

When simulations were performed, there was a conjecture that the input signals that would

result in longest delay were the one with all inputs switching together. However, extensive simulations over most of the possible input signals showed that this was not the case. It is because when all inputs are switching together, there will be two parallel paths connecting the switching input and the output both transiently on. This effectively reduces the channel resistance for the switching signal and hence reduces the delay of the signal. Moreover, such a case would not happen with other input combinations.

Wallace Tree Design

Wallace tree[3] is used here because of its fast response time. However, it has the disadvantage of having an unregular layout with great interconnection area. To overcome this, the Wallace Tree is designed on a 32 x 14 full adder array[1]. The design makes use of the fact that the sum output always fanouts to the same bit column; whereas, the carry output always fanouts to the adjacent column on the same side. Therefore, by putting all full adders which calculates sums for the same bit in the same column, routing of the array is restricted between adjacent columns. This greatly reduces the routing congestion and hence the interconnection area. Detail calculation shows that an area reduction of 46.6% is obtained.

As regards the active cell area, naive CPL implementation of the CSA full adder would increase the transistor count compared with CMOS design. However, by taking advantage of the fact that both positive and complementary signals are present in CPL logic, the transistor count and the depth of a circuit can actually be decreased. It can be shown by Shannon expansion[4] that a function of n variables can be implemented in a CPL circuit with depth n-1. As a result, both the sum and carry logic can be implemented in 2 pass transistor stages. It is shown that the transistor count using CPL is reduced by 17.5% compared with CMOS design. Moreover, by designing circuits according to Shannon expansion, it can be made sure that in the steady state, one and only one path from the input to the output is on. Thus removing the possibility of short-circuiting between Vdd and GND.

---

1. Not all array positions are occupied by full adders.

Binary Carry Look Ahead (BCLA) Adder Design

The adder design is based on the BCLA adder proposed by R.P.Brent and H.T.Kung[2]. The original design constructs two tree architectures which incrementally calculate the G and P values. This results in 2 (log n - 1) CLA stages where n = number of bits to be added. This translates to 8 CLA stages with a maximum fanout of 6 in 32 bit adder design. This design is modified here and one with only log n (i.e. 5) CLA stages but with a maximum fanout of n/2 + 1 (17) is used. Under this new design, a delay reduction of 22.4% (0.61ns) can be obtained along the critical path. It is believed that in designing larger multipliers (32 bits or 64 bits) hierarchical buffering is essential to coping with high fanout problem.

While the present BCLA can be further compacted in area, we increase the width of the BCLA to that of the Wallace Tree array so that routing between the Wallace Tree and the BCLA is minimized. In the present design, their dimensions fit nicely with each other. The whole multiplier has a dimension of 3891λ x 5312λ occupying an area of 1.292 mm^2 with a delay of 4.6 ns using 0.5um technology.

References

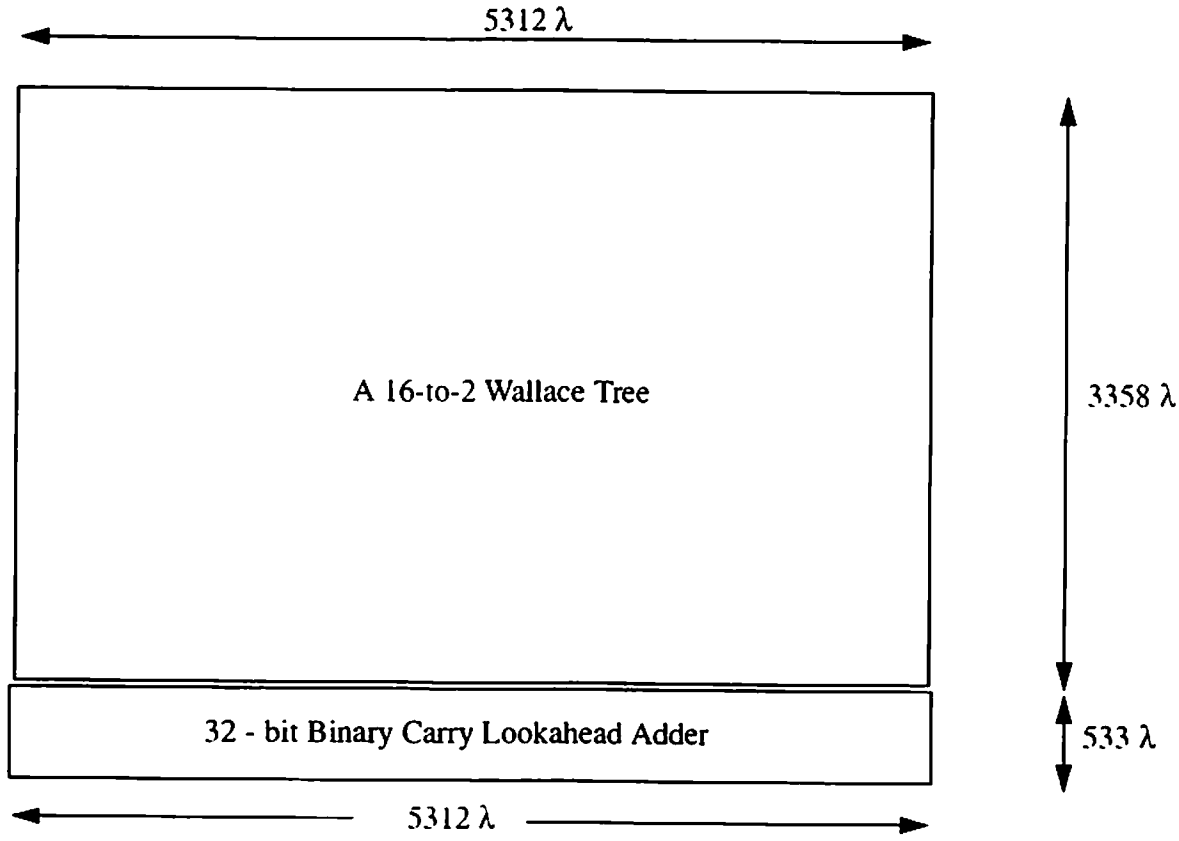
[1] K.Yano, T.Yamanaka, T.Nishida, M.Saito, K.Shimohigashi and A. Shimizu, "A 3.8-ns CMOS 16 X 16-b Multiplier Using COmkplementary Pass-Transistor Logic." 1990 IEEE Journal of Solid-State CIrcuits, Vol.25,No.2,pp388-394

[2] R.P.Brent and H.T.Kung, "A Regular Layout for Parallel Adders," IEEE Transactions on Computers, VOl. C-31, No.3, March 1982, pp.260-264

[3] C.S.Wallace. "A suggestion for a fast multiplier." IEEE Transactions on Electronic Computers, 13:14-17, 1964.

[4] C.E.Shannon. A symbolic analysis of relay and switching circuits." AIEE Transactions, 57:713-723, 1938.

Floorplan of the 16 bit x 16 bit Multiplier



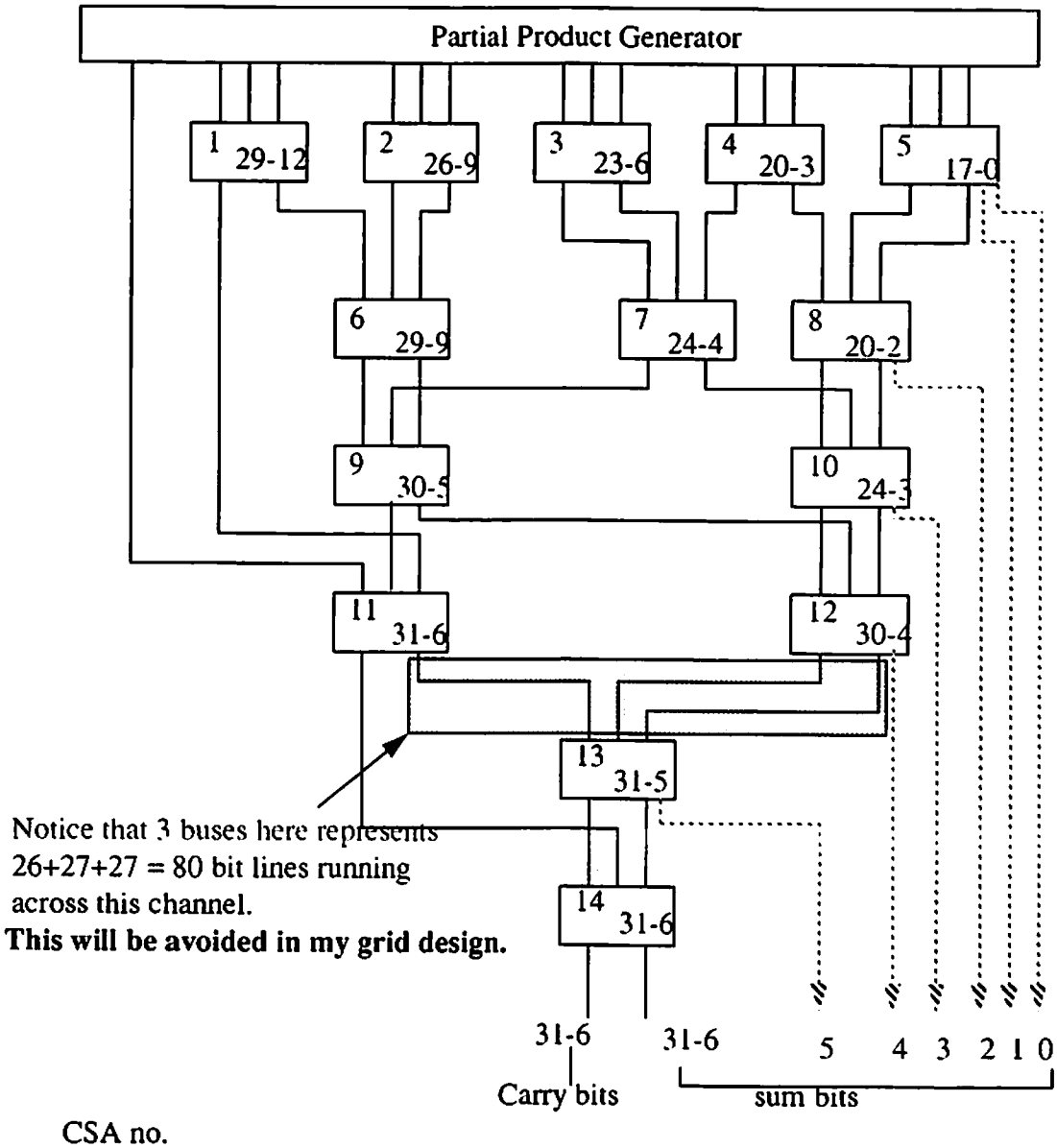
Area =3891 λ x 5312 λ = 1.292 mm^2

Wallace Tree for 16 bit by 16 bit multiplication

—— represents a BUS line
..... represents a one-bit line

Estimated Tree width = 5*18*adder width + one bus width = 8448λ
Estimated Tree height = 6*adder height + And-gate height + input routing + 13*routing channel width = 3955λ

INPUT: 16 x 16 x 2



CSA no.

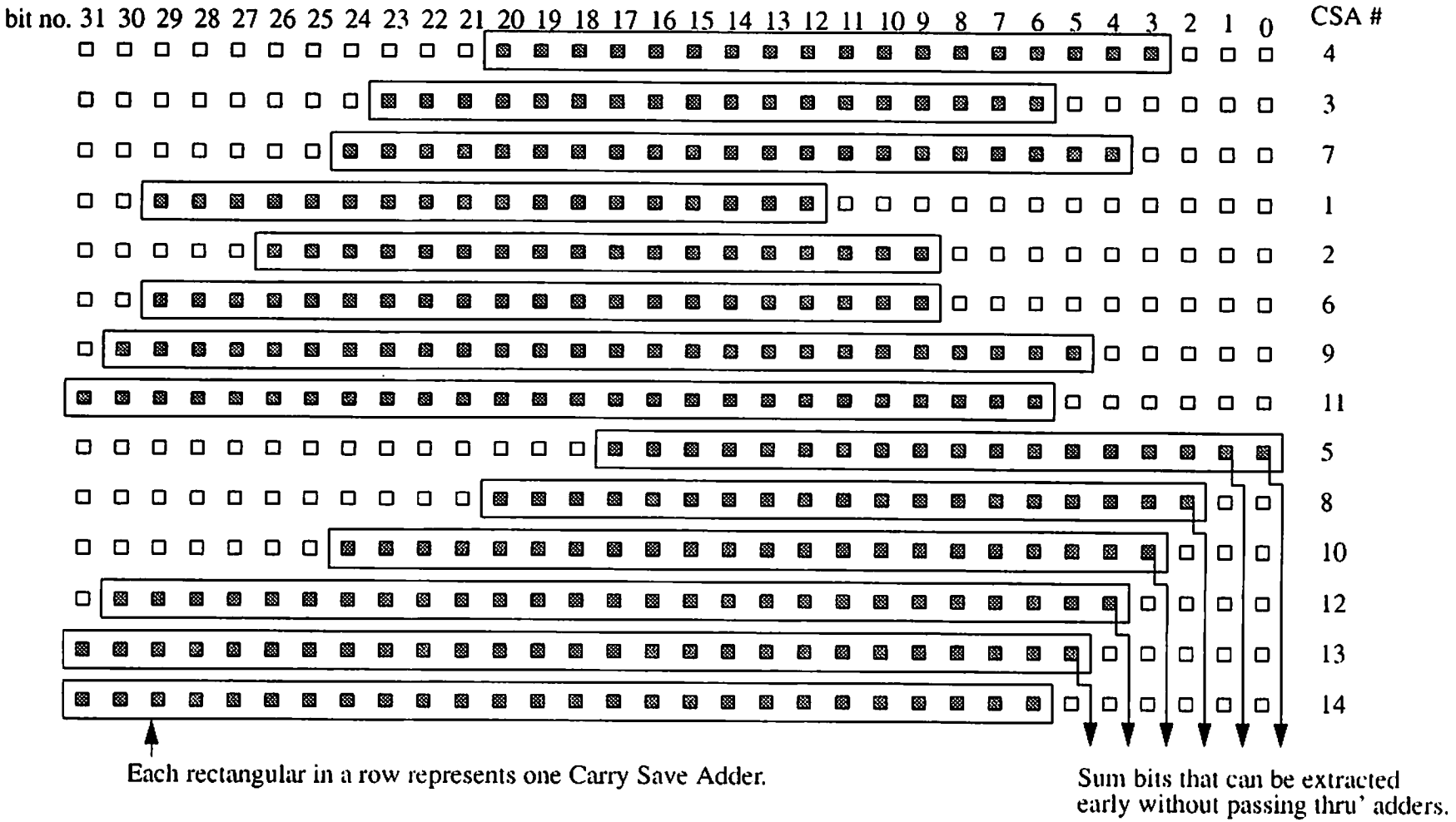
6 29-9 ← bit no: bits to be added by this adder

Carry Save Adder (composed of 21 full adders)

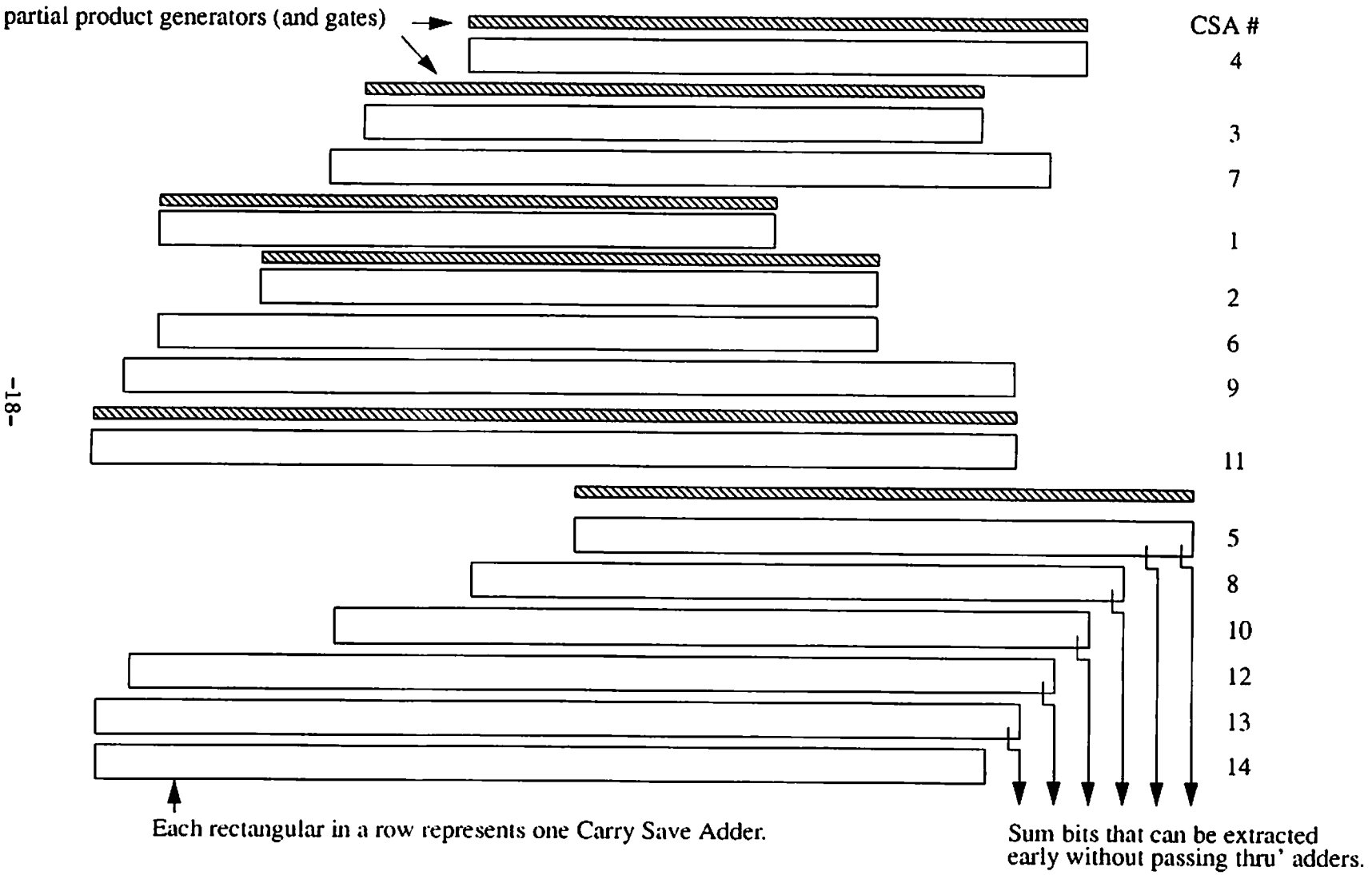
Implementation of Wallace tree in a 14 x 32 grid

Estimated Size = 3358λ x 5312 λ

■ denotes that the grid is occupied by a Full Adder. Each row corresponds to a Carry Save Adder whose number matches with that in the tree configuration.



Grid of Wallace Tree with addition of Partial Product Generators (And_gates)



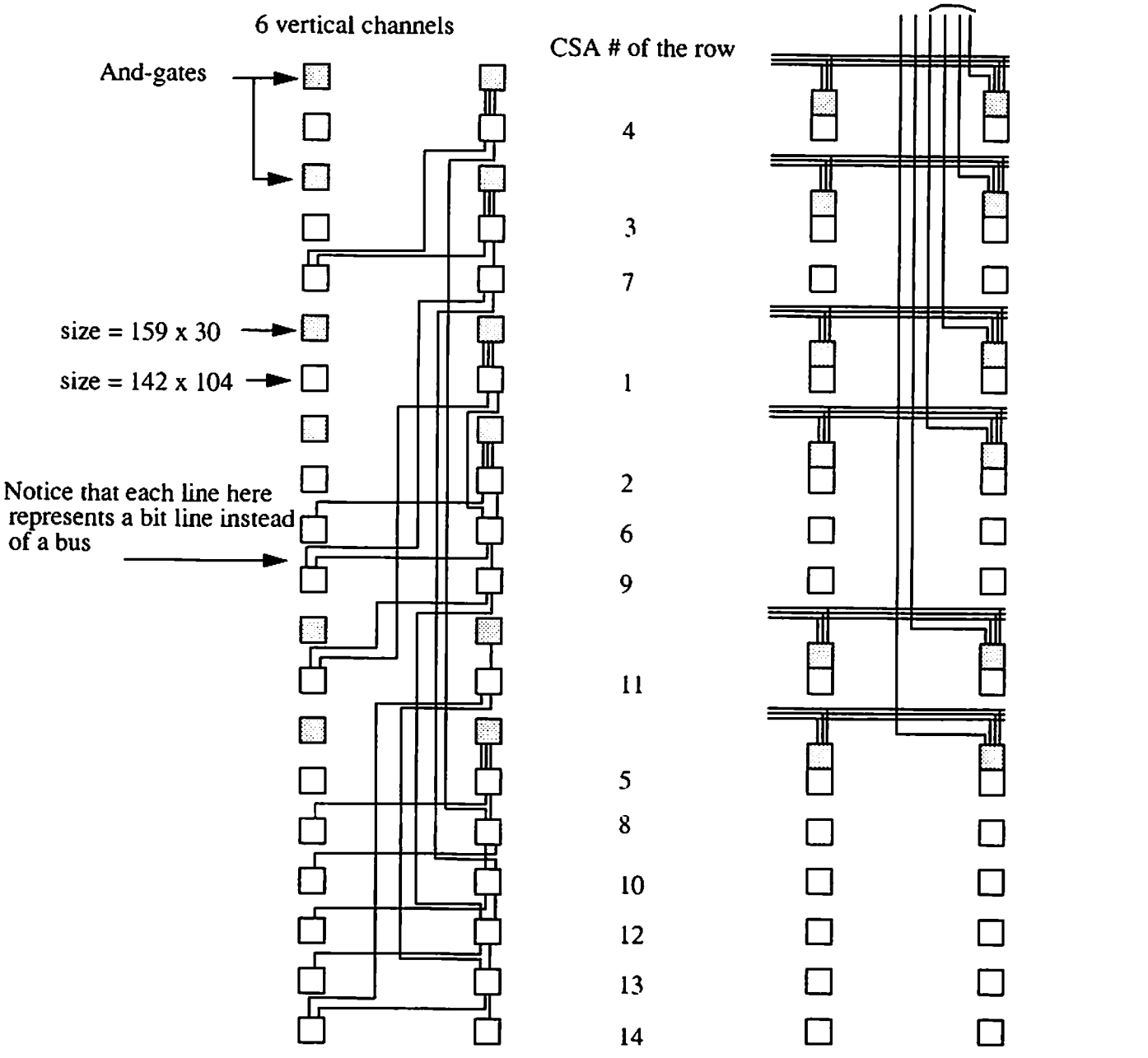
Routing of Wallace Tree in Grid Implementation with Area Estimation

Area reduction = 46.6 %

Routing between 2 adjacent columns of the Wallace Tree Grid

Routing of input lines

Note that these 4 lines can be routed in existing channels, So only 2 extra channels are required for input lines



Estimated Grid Height

= 6 * And gate height + 14 * adder height + 28*channel routing width+24 *input routing

= **3358 λ**

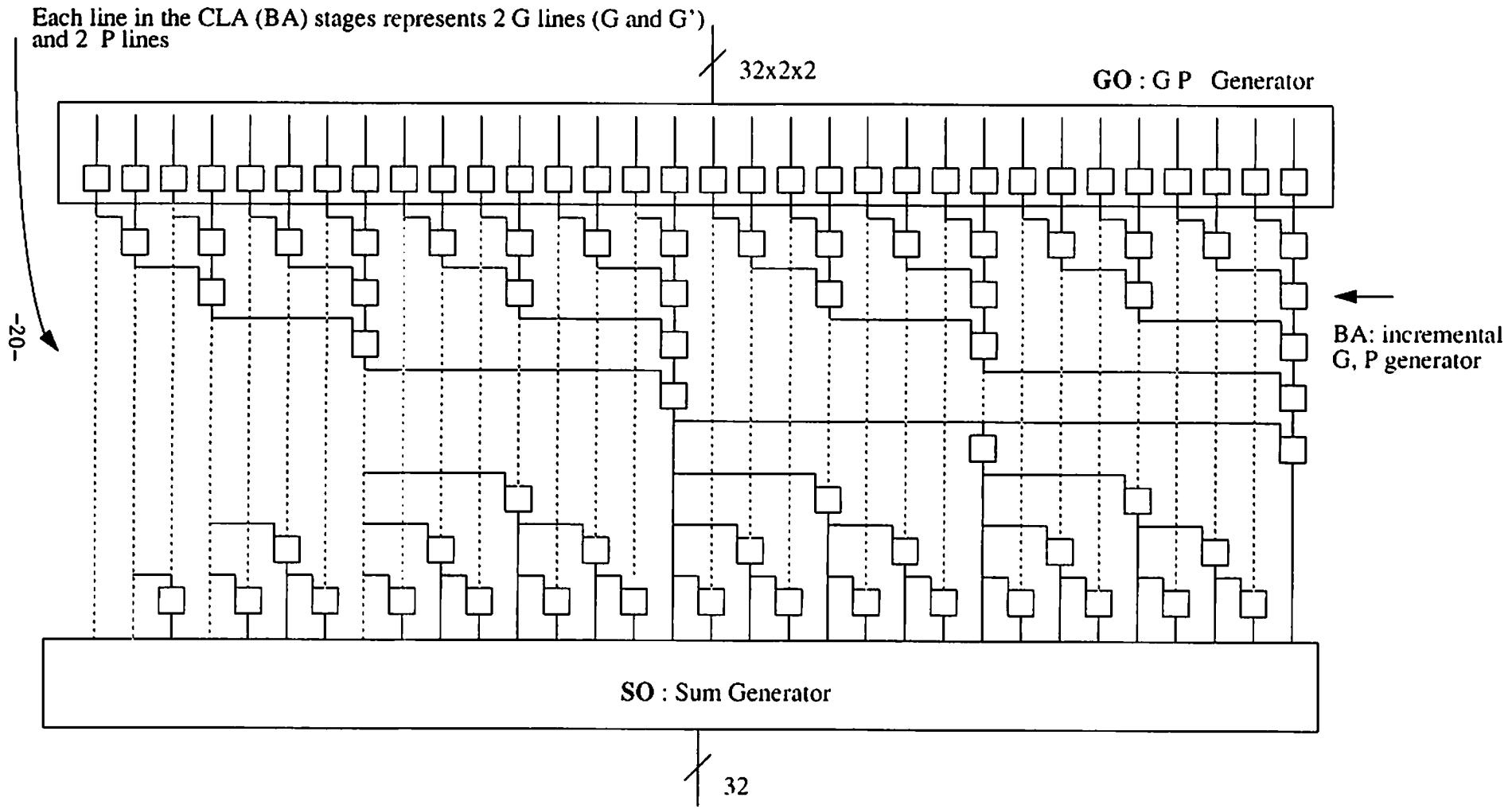
Esitimated Grid Width

= 32 * adder width + 31 * 6 * channel routing width + 31 * 2 * input routing width

= **5312 λ**

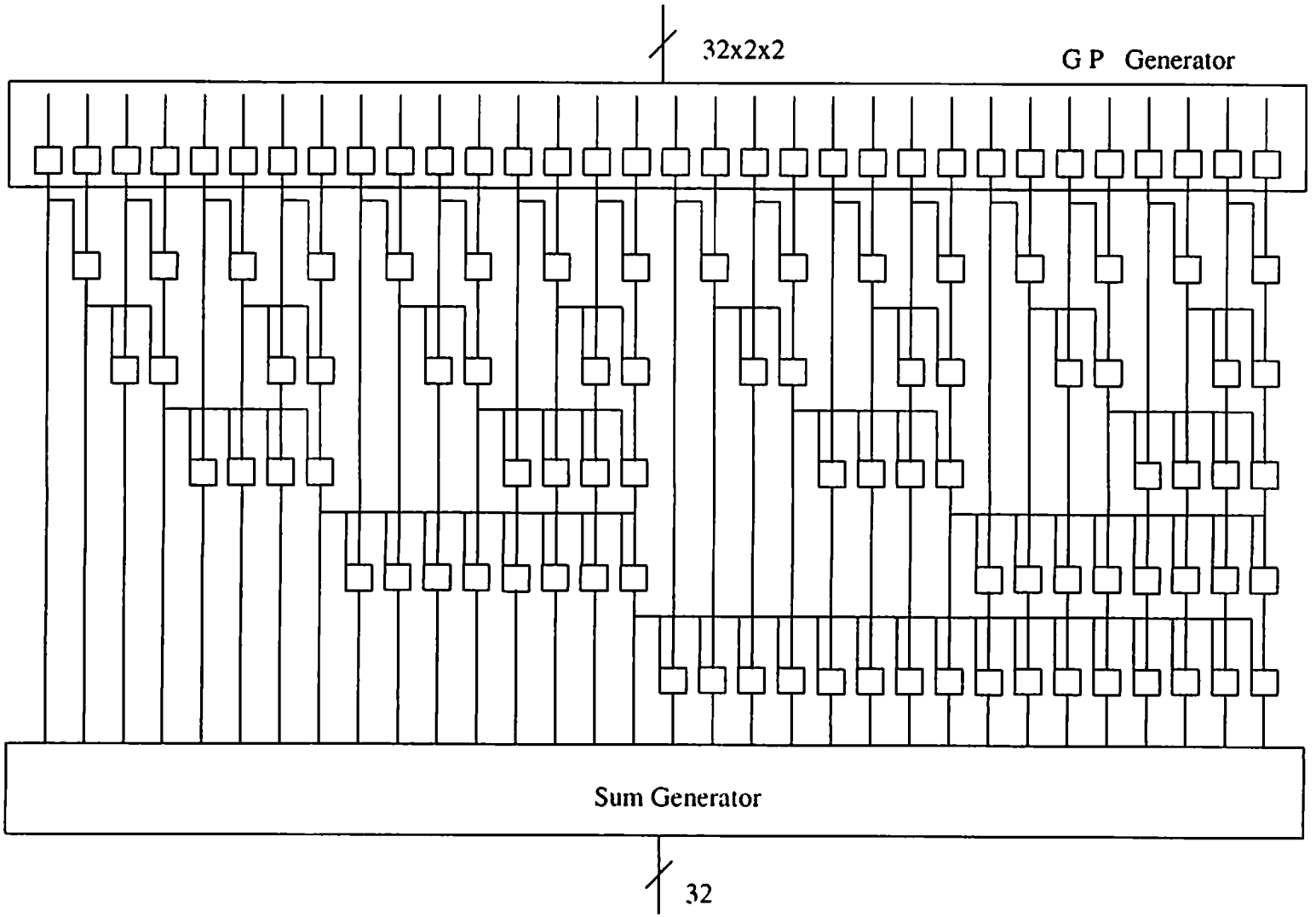
32 bit Binary Carry Lookahead Adder (before modification)

Estimated delay =3.07 ns



Optimized 32 bit Binary Carry Lookahead Adder

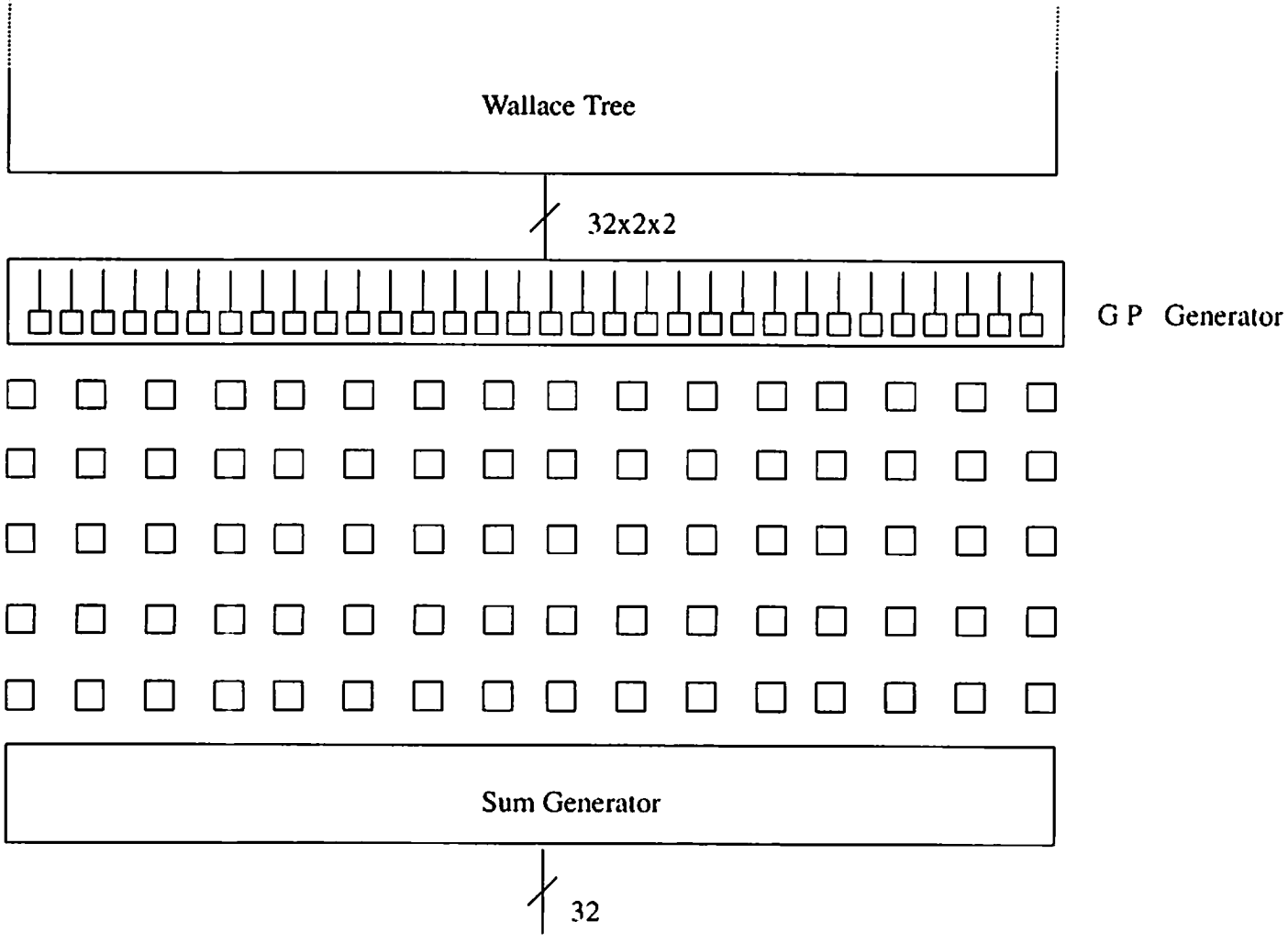
Estimated Delay = 0.2459 ns



Estimated height = 103 + 69*5 + 45 + 8*5 = 533 λ
Estimated width = 184 * 32 = 5888 λ

Area Compacted 32 bit Binary Carry Lookahead Adder

Estimated Delay = 0.2459 ns

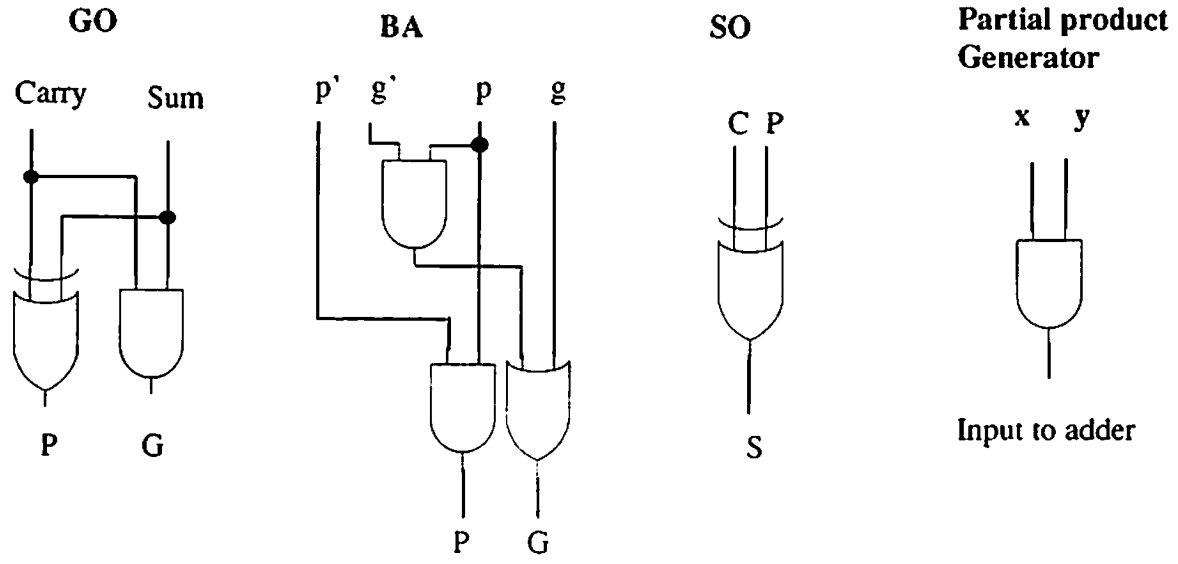


The CLA stages are first compacted to half of the original width and then re-expaned to align with the output lines of the Wallace Tree to reduce routing

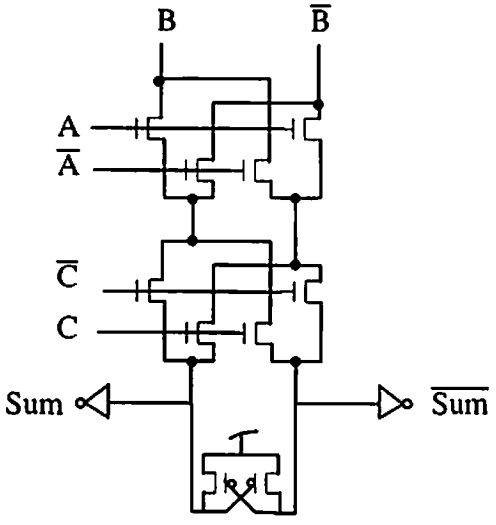
Estimated height = 103 + 69*5 + 45 + 8*5 = 533 λ

Estimated width = 5312 λ

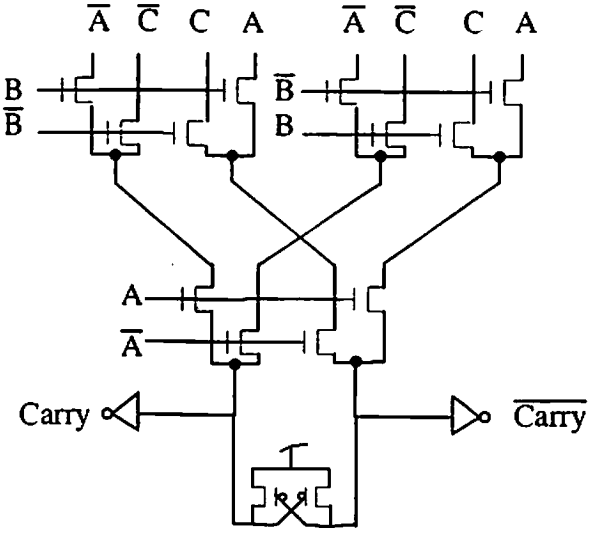
Block diagrams for BCLA components



Circuit Diagram for Sum Circuit



Circuit Diagram for Carry Circuit for Full Adder



Simulation Results

Optimization of Pass Transistor channel width

Pullup = 3λ/2λ (width/length), p-transistor of inverter = n-transistor of inverter = 12λ/2λ,

Pass Transistor width (λ)	Sum Delay			$\overline{\text{Sum}}$ Delay		
	rise (ns)	fall(ns)	average(ns)	rise (ns)	fall(ns)	average(ns)
7	0.30943	0.299804	0.3054	0.33104	0.26872	0.2999
8	0.30505	0.297756	0.3014	0.33384	0.26222	0.2980
9	**0.29791**	**0.295905**	**0.2969**	**0.33711**	**0.25859**	**0.2952**
10	0.30021	0.296794	0.2985	0.34521	0.25875	0.3020
11	0.30838	0.299428	0.3039	0.35524	0.25741	0.3063
12	0.30228	0.29872	0.3005	0.36860	0.25438	0.3115
optimum=9			0.2969			0.2952

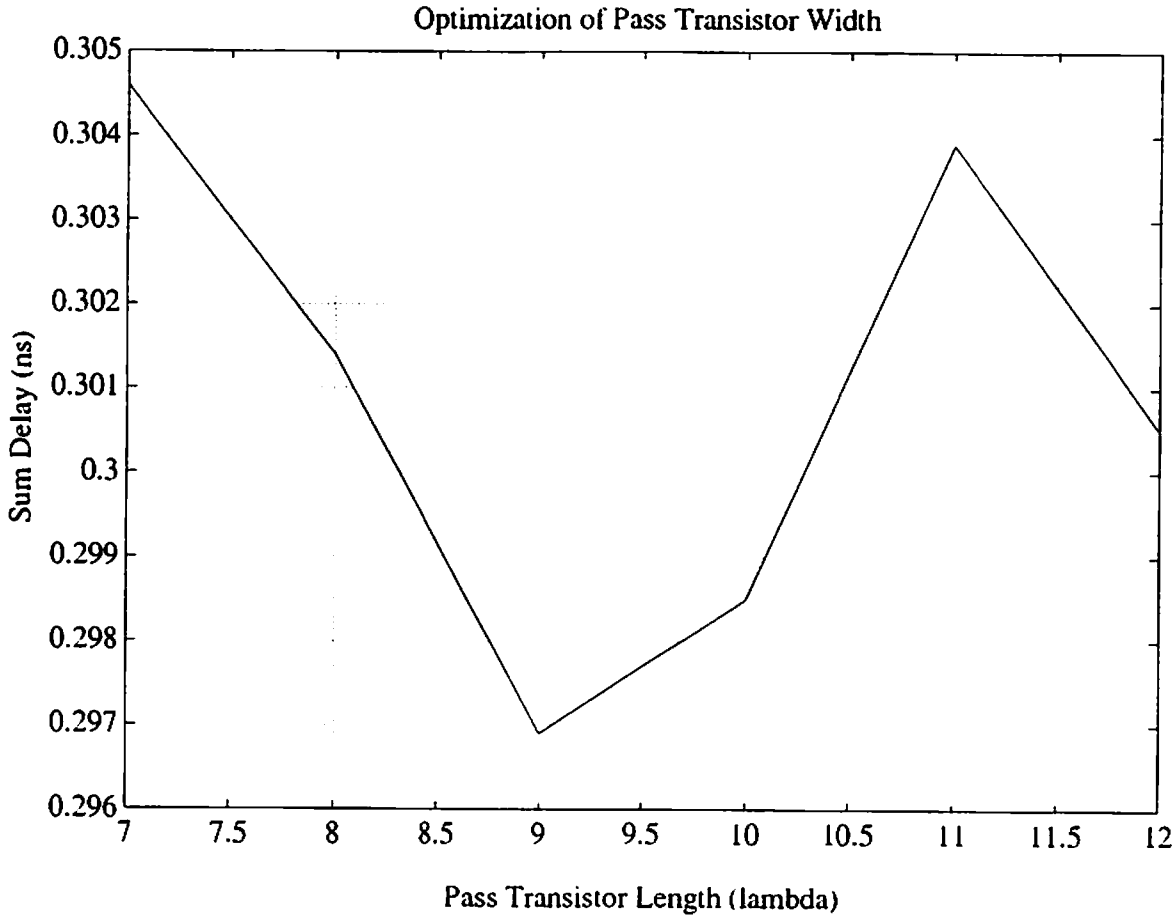
Simulation Results show that optimum Pass Transistor size = 9λ / 2λ.

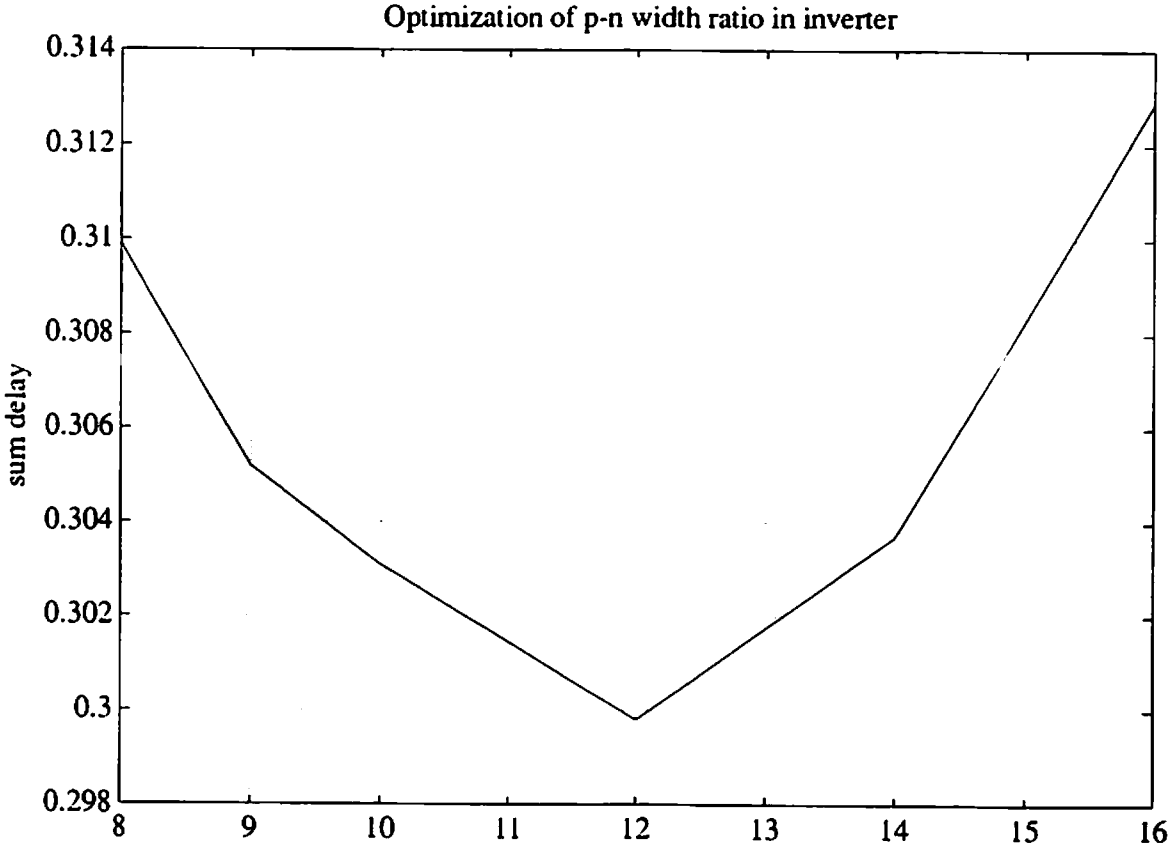
Optimization of p-n channel width ratio in inverter

Condition: after area compaction with pass transistor size = 9λ / 2λ

Transistor Width (λ)		Propagation Delay (ns)			
p	n	sum	$\overline{\text{sum}}$	carry	$\overline{\text{carry}}$
12	8	0.3099	0.3052	0.2655	0.2854
12	9	0.3052	0.3039	0.2694	0.2798
12	10	0.3031	0.3014	0.2519	0.2744
12	**12**	**0.2998**	**0.2985**	**0.2606**	**0.2777**
12	14	0.3037	0.3023	0.2604	0.2790
12	16	0.3129	0.3065	0.2638	0.2797

By minimizing the column of longest delay i.e. the 'sum' column, we get the optimum p-n ratio = 12 : 12.


Optimization of Pass Transistor Width
Sum Delay (ns)
0.305
0.304
0.303
0.302
0.301
0.3
0.299
0.298
0.297
0.296
7
7.5
8
8.5
9
9.5
10
10.5
11
11.5
12
Pass Transistor Length (lambda)


Optimization of p-n width ratio in inverter
sum delay
0.314
0.312
0.31
0.308
0.306
0.304
0.302
0.3
0.298
8
9
10
11
12
13
14
15
16
width of n-transistor of inverter

Optimization of transistor size in inverter under the p-n ratio of 12:12
Condition: with Pass Transistor size = 9λ/2λ and p-n ratio of inverter = 12:12

Transistor Width (λ)		Propagation Delay (ns)			
p	n	Sum	$\overline{\text{Sum}}$	Carry	$\overline{\text{Carry}}$
10	10	0.3079	0.3079	0.2659	0.2870
11	11	0.3045	0.3035	0.2653	0.2814
12	**12**	**0.2998**	**0.2985**	**0.2606**	**0.2777**
13	13	0.3025	0.2986	0.2614	0.2788
14	14	0.3042	0.2979	0.2628	0.2795
16	16	0.3061	0.3069	0.2653	0.2823

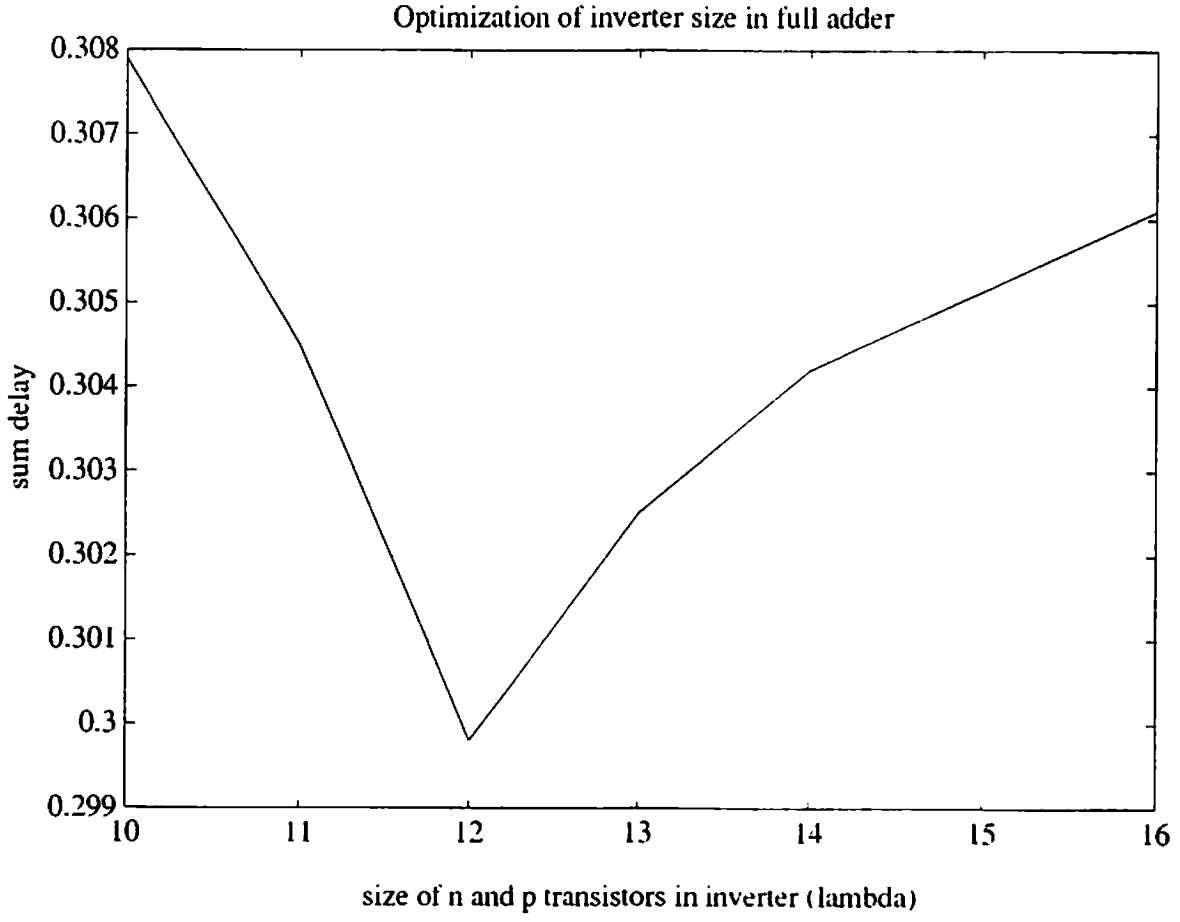
Once again, we minimize the column with longest delay, i.e. the 'sum' column, and get the optimum size of the p and n transistors for the inverter to be 12λ / 12λ.

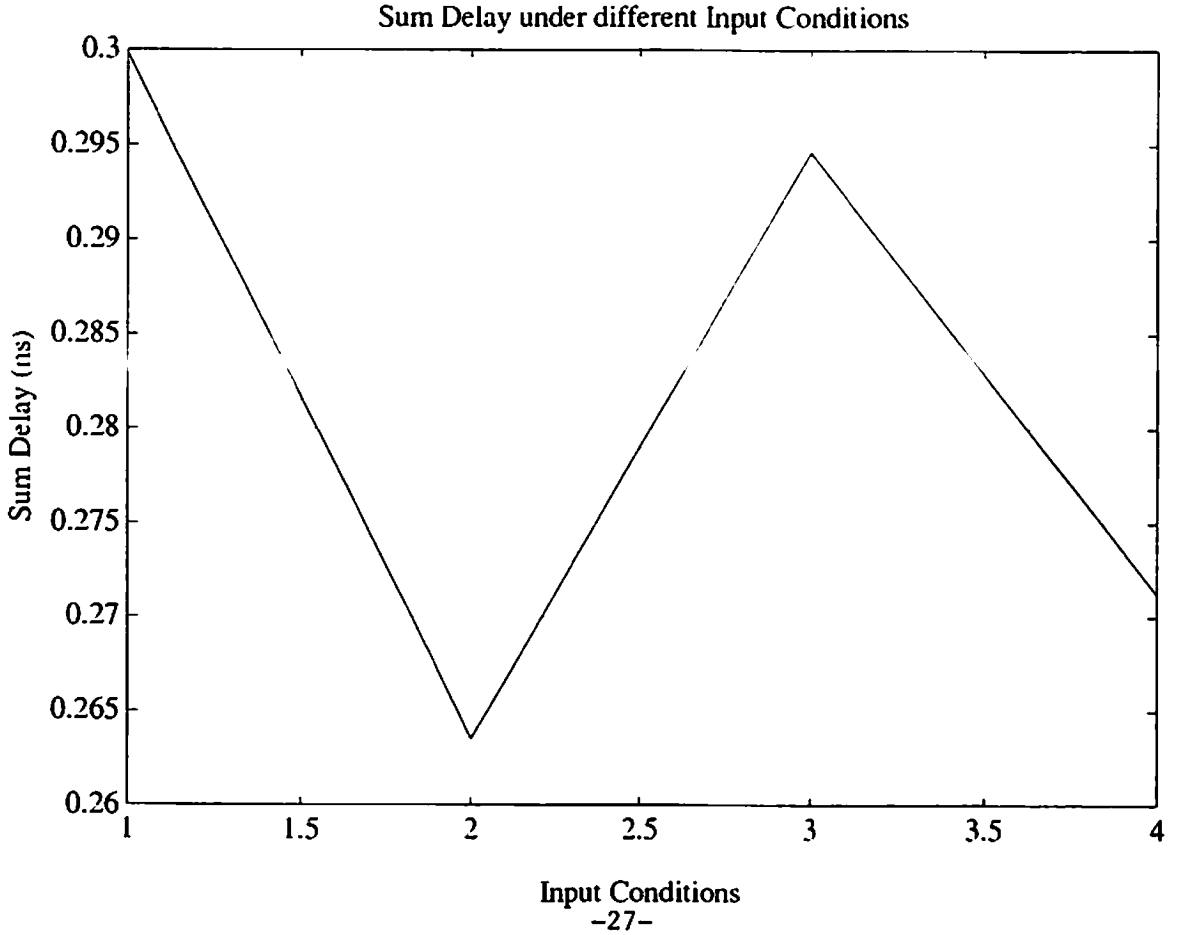
Throughtout the simulations, we use the set of input parameters in which only input A switches. Meanwhile we have compared all four possible input conditions that would result in switching in both sum and carry input. They are:

1) only input A switches and B is kept in logic 1, C in logic 0 or vice versa
2) all A and B and C switches from 0 to 1 or 1 to 0 together
3) only input B switches and A is kept in logic 1, C in logic 0 or vice versa
4) only input C switches and B is kept in logic 1, A in logic 0 or vice versa

Results are shown in the following table and it shows that input condition 1 leads to the longest delay.

Input Conditions	1	2	3	4
Sum delay (na)	**2.998**	2.635	2.946	2.712


Optimization of inverter size in full adder
sum delay
0.308
0.307
0.306
0.305
0.304
0.303
0.302
0.301
0.3
0.299
10
11
12
13
14
15
16
size of n and p transistors in inverter (lambda)


Sum Delay under different Input Conditions
Sum Delay (ns)
0.3
0.295
0.29
0.285
0.28
0.275
0.27
0.265
0.26
1
1.5
2
2.5
3
3.5
4
Input Conditions

Simulations were done on other stages as well and the results are as follows:
(longest delays are highlighted)

Delay for Partial Product Generator (ns)

Input Conditions	And	$\overline{\text{And}}$
Both inputs x and y switch	0.3310	0.2498
Input x switches	0.3219	0.2452
Input y switches	**0.3391**	0.2214

Delay for GO (ns)

G	$\overline{G}$	P	$\overline{P}$
0.1626	0.1573	0.1625	0.1570

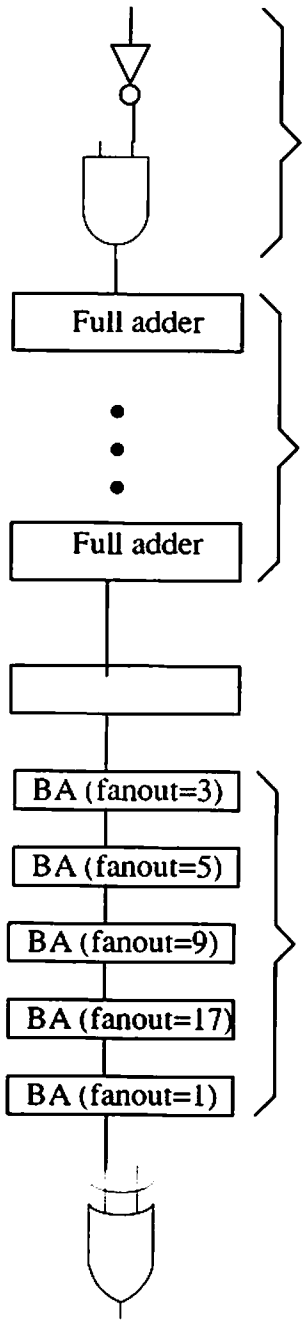
Delay for BA with different fanouts (ns)

no. of fanouts	G	$\overline{G}$	P	$\overline{P}$
1	**0.3159**	0.2586	0.1573	0.1278
3	**0.3459**	0.3086	0.1984	0.1788
5	**0.3802**	0.3518	0.2308	0.2269
9	0.4331	**0.4527**	0.3014	0.3406
16	0.5135	**0.6370**	0.3780	0.5269

Delay for SO (ns)

xor	xnor
0.1636	**0.1637**

Critical Path of the Multiplier



		Delay (ns)
Partial Product Generator		0.339
6 Full adders in series		0.300 x 6
GO		0.163
BA of different fanouts		0.346
		0.380
		0.453
		0.637
		0.316
SO		0.164

total delay = 4.598 ns

Transistor counts for CPL implementation

Component	Transistors per unit	No. of units	Total	Percentage
Partial Prod Generator	10	16x16	2560	18.7
Full Adder	28	305	8540	62.3
GO	16	32	512	3.7
BA	20	16x5	1600	11.7
SO	16	31	496	3.6
Total Count			13708	100.0

Transistor count for CMOS implementation

Component	Transistors per unit	No. of units	Total	Percentage
Partial Prod Generator	6	16x16	2560	15.4
Full Adder	40	305	12200	74.4
GO	12	32	384	2.3
BA	16	16x5	1280	7.7
SO	6	31	186	1.1
Total Count			16610	100.0

It shows that by using CPL implementation, we attain a reduction of **17.5%** in transistor count.

Hierarchical buffering of high fanout pins of Binary CLA.

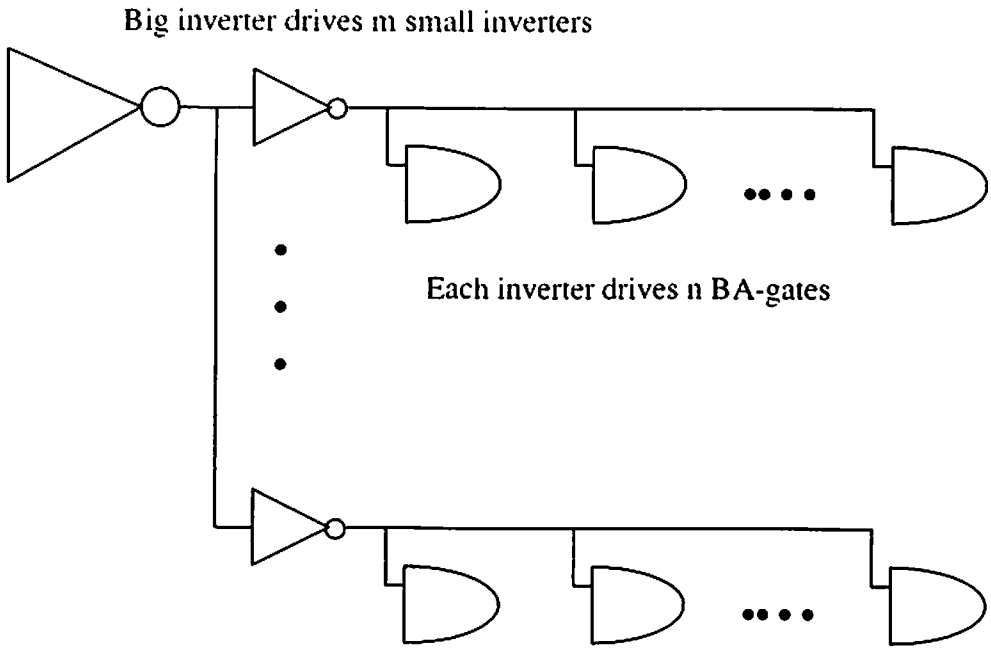


Table 1: Delay of big inverter with m fanouts

m	3	4	6	8	12	16
delay1 (ns)	0.095	0.12	0.18	0.28	0.40	0.57

Table 2: Delay of small inverter with n fanouts

n	1	2	3	4
delay2 (ns)	0.067	0.11	0.17	0.22

For Fanout = 16. we need to minimize total delay = delay1 + delay 2 under the constraint m x n >=16
we get optimum delay = 0.34 ns with m=4. n=4

For Fanout = 8 . we need to minimize total delay = delay1 + delay 2 under the constraint m x n >=8
we get optimum delay = 0.23ns with m=4. n=2

From the above results. it is believed that **in designing larger multipliers (32 bits or 64 bits) hierarchical buffering is essential to coping with high fanout problem.**

Decomposition of the Sum and Carry expression using Shannon Expansion.

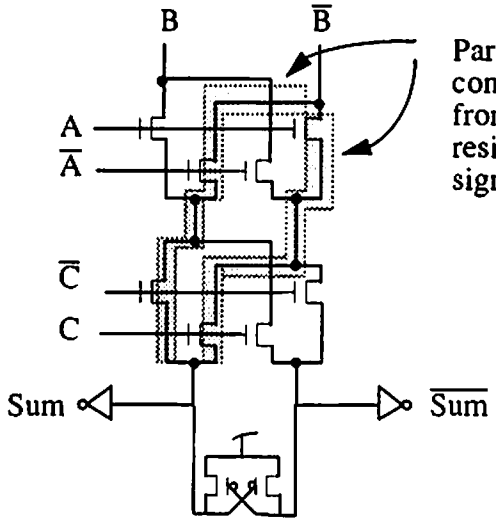
Carry
= AB +BC +CA
= A[B + C] + A'[BC]
= A[B(1) + B'(C)] + A'[B(C) +B'(0)]
= A[B(A) + B'(C)] + A'[B(C) +B'(A')]

Sum
= ABC + AB'C' + A'BC' + A'B'C
= C[AB + A'B'] + C'[AB' + A'B]
= C[A(B) + A'(B')] + C'[A(B') + A'(B)]

The above decomposition maps directly to the circuit diagram of the sum and carry stages of the full adder cell.

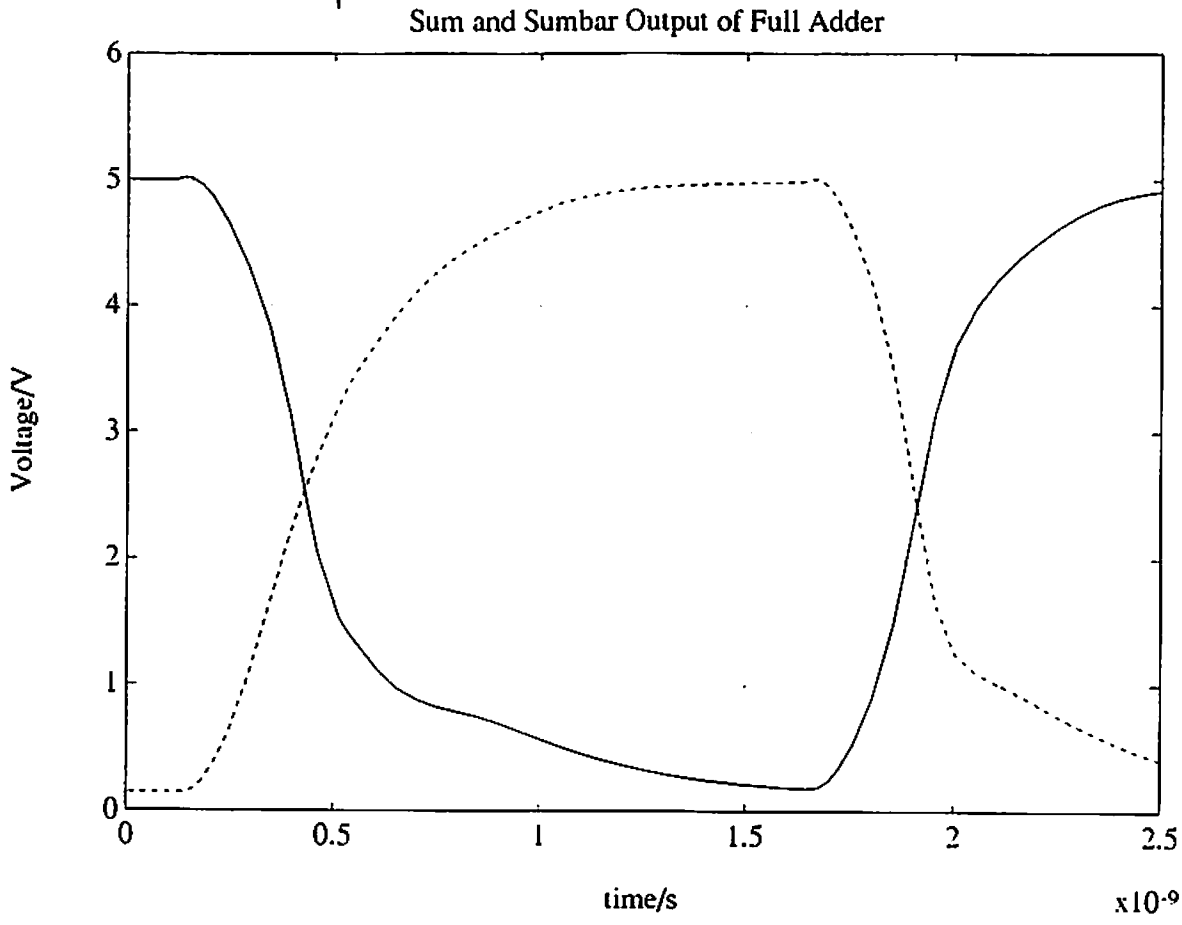
Parallel conducting paths results from all input switching together

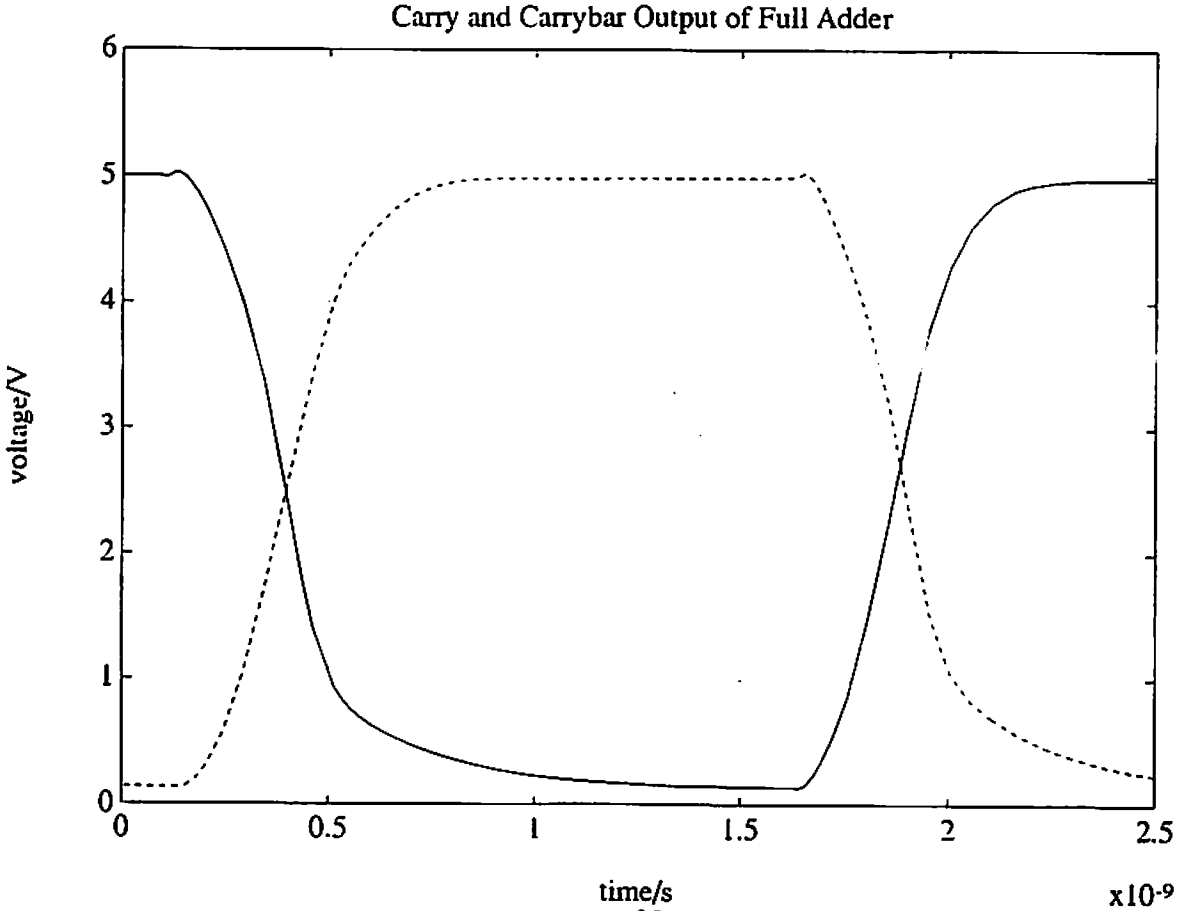
Circuit Diagram for Sum Circuit

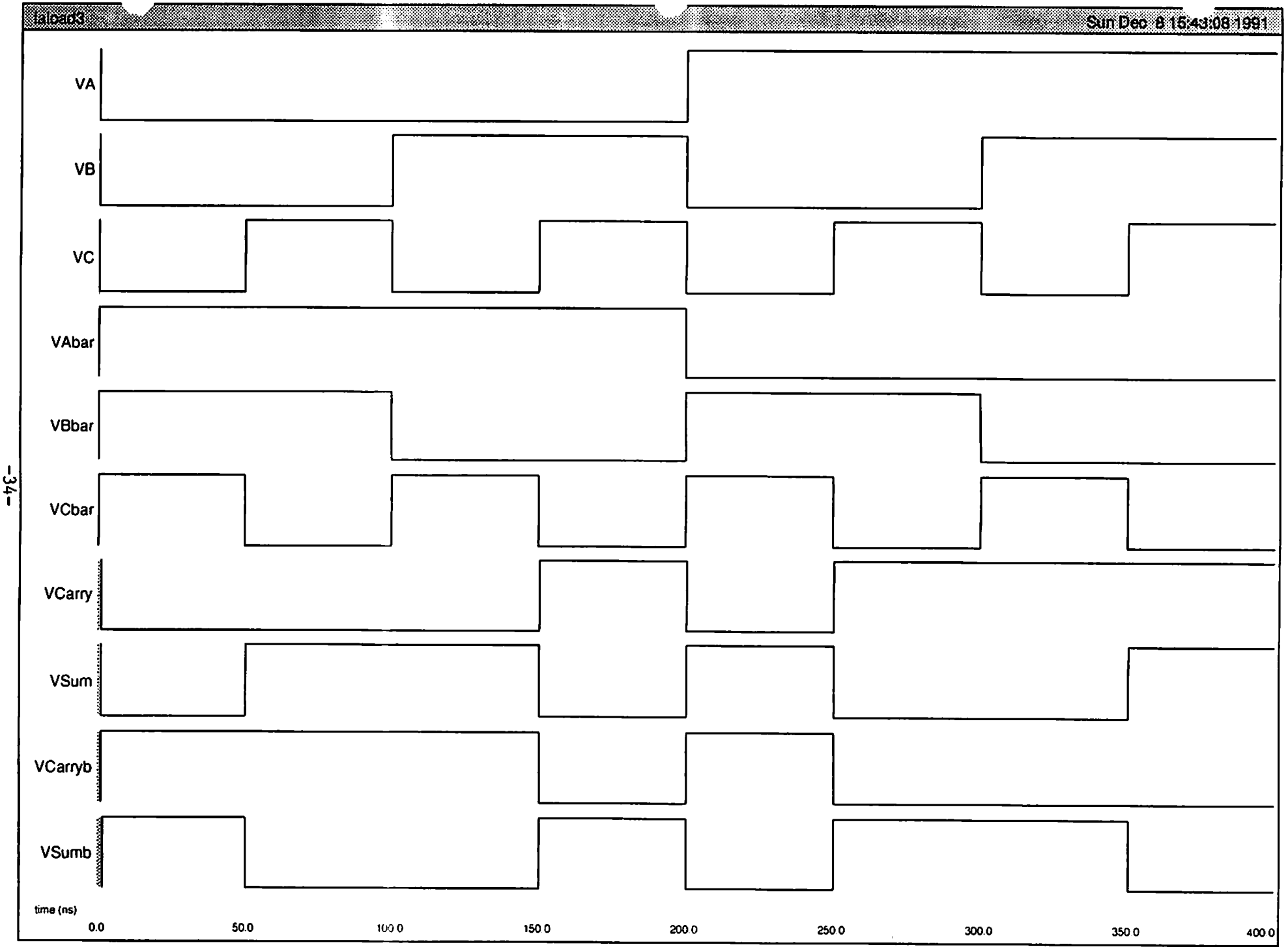


Parallel Paths that transiently conducts when all input switches from 1 to 0. This reduces the path resistance and hence reduces the signal rise time.

Spice Simulation Results







Irsim Result for Full Adder

EE 577 Term Project Report

The Design of a 16 × 16-b Multiplier for Unsigned Binary Integers

by Fai Li (092-72-7186)

Advisor: Associate Professor B. Sheu
Department of Electrical Engineering - Electrophysics
University of Southern California

Abstract

This term project is focused on the design of a 16 × 16-b multiplier for unsigned binary integers using 0.5 μm technology ($\lambda = 0.25$ μm). Architecture issues are considered first. It is followed by the comparisons of three implementations of the basic cells. Then the critical path simulation is presented to achieve a response time of 6.4 ns. Floorplan satisfying 450λ height requirement is also presented.

Introduction

Various aspects of the design of a 16 × 16-b multiplier for unsigned binary integers will be discussed. Issues on architecture, implementation of circuit elements, SPICE simulations on performance, and system floorplan will be included. The primary goal of this design is to achieve a response time close to 5 ns and to satisfy a constant height constraint of 450λ.

Arithmetic

The basic operation of an unsigned binary integer multiplication is shown in figure 1. For a N × N-b multiplication, N^2 partial product terms are needed. Figure 2 shows the carry-save addition. Examples using this method are shown in figure 3 [Hwan79]. This method has the advantage of higher speed over the carry-ripple addition method when the number of numbers to be summed is large.

Architecture

I. Array Architecture

Figure 4 shows the block diagram of a N × N-b array multiplier. Figure 5 shows the adder array block with a delay of (3N - 3)Δ, where Δ is the delay of single adder cell. An improved array architecture is shown in figure 6. This improvement is based on the utilization of zero carry-inputs to the next stage of adder cells on the right hand half of the adder array. Thick arrows in the figure show these altered arrangements. With this structure, the delay time is reduced to (2N - 1)Δ. To achieve a response time of 5ns for 16 × 16-b multiplication, $\Delta \leq (5ns) / (2N - 1) = 0.161ns$, which is not easily achievable. Therefore, improvements on architecture are needed.

II. Wallace Tree Architecture

The block diagram of this architecture is shown in figure 7.

A. Partial product generation block

This block consists of 256 (16^2) AND gates (shown in figure 8). These partial product terms will be fed into the Wallace tree adders.

B. Wallace tree adder block

The basic cells in this block are carry-save adders. They are connected as shown in figure 9 [HwBr84]. The number of carry-save adders needed are shown in each sub-block. A total of 253 adders are needed if we do the routing carefully. Otherwise, we can just put 32 adders in each sub-block and feed in zeros. This will result in a total of 448 adders which is not efficient in area consideration but may save some routing efforts. Six layers of addition must be finished before the final sum and carry terms of 31-bit each can be produced. These two numbers will then be processed by the following CLA (carry lookahead) adder block. Figure 10 shows a way to save on hardware by doing the addition iteratively [HwBr84].

C. CLA adder block

Figure 11 shows the block diagram of the CLA adder block. The generate and propagate signals generation block consists of 32 AND gates (figure 12) for the generate terms and 32 EXOR gates (figure 13) for the propagate terms.

For the CLA block, two approaches are introduced here. The first one is the BLC (binary lookahead carry) method [WeEs85]. The individual cell is shown in figure 14. Figure 15 shows the case for a 32-bit CLA by this method. The maximum delay for N-bit carry generation is $(\log_2 N)\Delta$, where Δ is the delay of individual cell. As N increases, the number of fan-outs at some nodes increases proportionally. So buffers may be needed to restore signal strengths at nodes with large fan-outs. For the 32-bit case, the total delay should be 5Δ.

The second approach is the BCLA (block carry lookahead) approach [Hwan79]. A 4-bit BCLA block is shown in figure 16 [Hwan79]. For this cell, there are two delays Δ_1 and Δ_2. Δ_1 is the maximum delay for generating C_2, C_1, or C_0 from C_{-1} (in this case, C_2 is the limiting factor). It can be approximated by a 3-bit BLC block with an additional gate (a four-input AND gate) to account for C_{-1}. Finally an OR gate will provide the final C_2. So we can approximate Δ_1 with 2.5Δ(0.5Δ accounts for the final OR gate) without much error. Δ_2 is the delay for generating G* and P* which are independent of C_{-1}. Δ_2 can be substituted by the delay for a 4-bit BLC block without loss of accuracy. So Δ_2 is 2Δ. Figure 17 shows the 8-by-4 configuration of BCLA [Hwan79]. Δ_3 is the delay for the 8-bit BCLA block which can also be approximated by an 8-bit BLC block without loss of accuracy. So Δ_3 is 3Δ. The maximum delay in this configuration is $\Delta_2 + \Delta_3 + \Delta_1 = 7.5\Delta$. The independency of G* and P* on C_{-1} makes this configuration possible.

The final sum generation block is just 32 EXOR gates (shown in figure 18).

Cell Implementation

Three types of cell implementations, namely the conventional CMOS, the CVSL (cascode voltage switch logic) [SoMi91], and the CPL (complementary pass-transistor logic) [Yano90], of basic AND, CARRY, and SUM cells are compared. The comparison is done in both the response time of the cell (with loading from the same kind of cell) and the number of transistors in the cell. The results are shown in table 1. Figure 19 shows the basic cells of the three implementations. The CPL implementation has a great advantage in speed over the other two but does not have any advantage in transistor count. Note that the SPICE simulations are done in 0.5 μm technology with the parameters of transistors shown in figure 20.

Figure 21 and 22 show the effect of the ratio of $\{(W_P/L_P)/(W_N/L_N)\}$ on the delay times of the CPL AND gate. The ratio $\{(W_P/L_P)/(W_N/L_N)\} = 1$ gives the best response time. Figure 23 and 24 show the output waveform with $\{(W_P/L_P)/(W_N/L_N)\} = 1$ and $\{(W_P/L_P)/(W_N/L_N)\} = 3$ respectively. With $\{(W_P/L_P)/(W_N/L_N)\} = 1$, we can have a better response time and a better waveform. Figure 25 and 26 show the effect of increasing transistor sizes (with $\{(W_P/L_P)/(W_N/L_N)\} = 1$) on the response time. The ratio $\{(W_P/L_P)/(W_N/L_N)\} = 5$ gives the best result. However, for area consideration, $\{(W_P/L_P)/(W_N/L_N)\} = 1$ is used even it is slower.

Critical-path Simulation

For the BLC approach, figure 27 shows the effect of different buffering schemes on delay times. The 2-scheme (one buffer for every two fan-outs) has a faster speed over others generally. Figure 28 shows the effect of $\{(W_P/L_P)/(W_N/L_N)\}$ of the buffer (CMOS inverter) on delay time. As this ratio increases, the delay time decreases to a minimum around $\{(W_P/L_P)/(W_N/L_N)\} = 5$ and then levels off. For simplicity, the conventional ratio of 2 is chosen. Figure 29 shows the effect of transistor size of the buffer (with $\{(W_P/L_P)/(W_N/L_N)\} = 2$) on delay time. It shows that the delay is reduced most at a ratio of 5. From these simulations, the 4-scheme with buffer of $\{(W_P/L_P)/(W_N/L_N)\} = 2$ and transistor sizes five times the minimum is chosen.

The critical-path of the multiplier is shown in figure 30. With individual worst delay time of each cell, we can approximate the system delay to be 4.71 ns (calculation shown in figure 31). However, when the worst case simulations are conducted on this path in SPICE, with the 4-scheme chosen, the average delay time is 6.34 ns. This shows the effect of loading on delay time. Figure 32 shows a typical waveform of this critical path.

Floorplan

The floorplan that complies with the constant height constraint of 450λ is shown in figure 33. This has an approximate dimension of $450\lambda \times 22000\lambda$. This floorplan has a very large width-to-height ratio. Due to this large ratio, delay time may even be worse than simulated due to longer line delays. The area is 0.62 mm^2. Figure 34 shows another floorplan of the same multiplier. This has an approximate dimension of $1000\lambda \times 13000\lambda$. Area of this floorplan is 0.81 mm^2. The latter floorplan is better in the sense that it is easier to be routed without the height constraint and the line delays may be smaller than that of the former floorplan. This is a trade-off between speed and area.

Conclusion

A 16 × 16-b multiplier for unsigned binary integers is designed with an response time of 6.4 ns and a height constant of 450λ in 0.5 μm technology. Rooms for improvements still exist. One possible improvement in speed is the improvement on parameters of the transistors. Another one is optimizing transistor sizes and obtaining more compact cell layouts.

References

[Hwan79] K. Hwang, Computer Arithmetic: Principles, Architecture, and Design. New York: John Wiley and Sons, 1979.

[HwBr84] K. Hwang and F. A. Briggs, Computer Architecture and Parallel Processing. New York: McGraw-Hill, 1984.

[SoMi91] P. J. Song and G. De Micheli, "Circuit and architecture trade-offs for high-speed multiplication," IEEE Journal of Solid-state Circuits, vol. 26, no. 9, pp. 1184-1198, September, 1991.

[WeEs85] N. H. E. Weste and K. Eshraghian, Principles of CMOS VLSI Design: A System Perspective. Reading, Massachusetts: Addison-Wesley, 1985.

[Yano90] K. Yano et al., "A 3.8-ns CMOS 16×16-b multiplier using complementary pass-transistor logic," IEEE Journal of Solid-state Circuits, vol. 25, no. 2, pp. 388-395, April, 1990.

				a_3	a_2	a_1	a_0
×)				b_3	b_2	b_1	b_0
				a_3b_0	a_2b_0	a_1b_0	a_0b_0
			a_3b_1	a_2b_1	a_1b_1	a_0b_1	
		a_3b_2	a_2b_2	a_1b_2	a_0b_2		
	a_3b_3	a_2b_3	a_1b_3	a_0b_3			
P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0

A total of 16 (4^2) partial product terms are needed.

Figure 1. 4×4 unsigned binary integer multiplication.

	a_3	a_2	a_1	a_0
	b_3	b_2	b_1	b_0
+)	s_3	s_2	s_1	s_0
c_3	c_2	c_1	c_0	
S_4	S_3	S_2	S_1	S_0

Figure 2. 4-b carry-save addition.

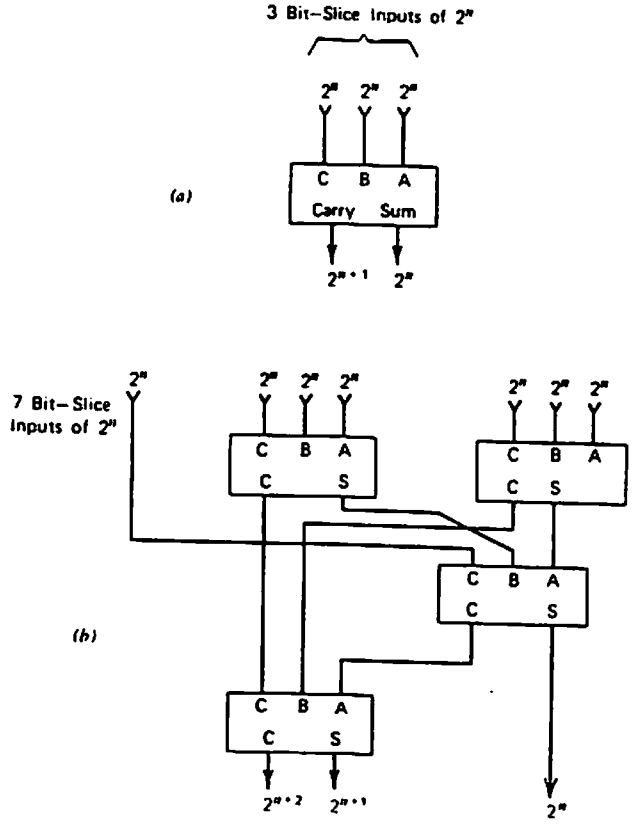


Figure 3. (a) A 3-bit-slice Wallace tree built with carry-save adder. ([Hwan79])
(b) A 7-bit-slice Wallace tree built with carry-save adders. ([Hwan79])

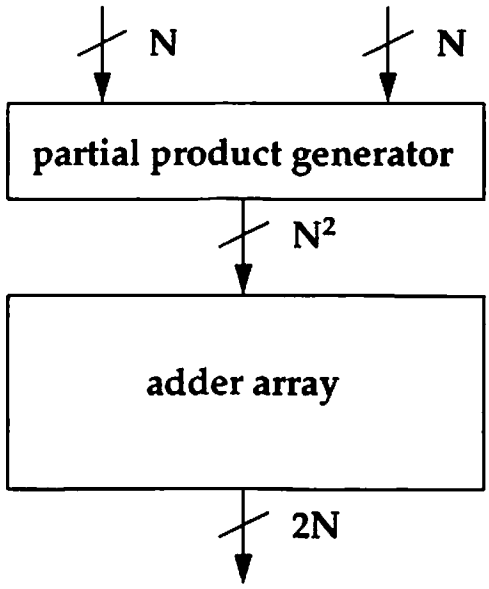


Figure 4. N × N-b array multiplier block diagram.

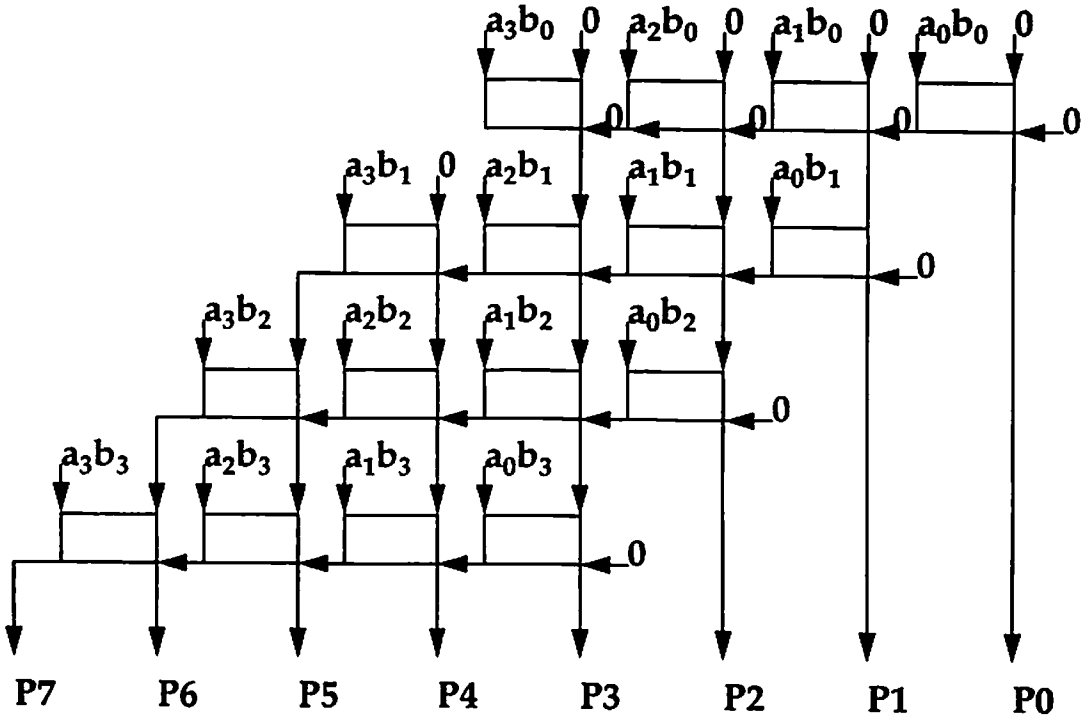


Figure 5. Adder block for a 4 × 4-b array multiplier.

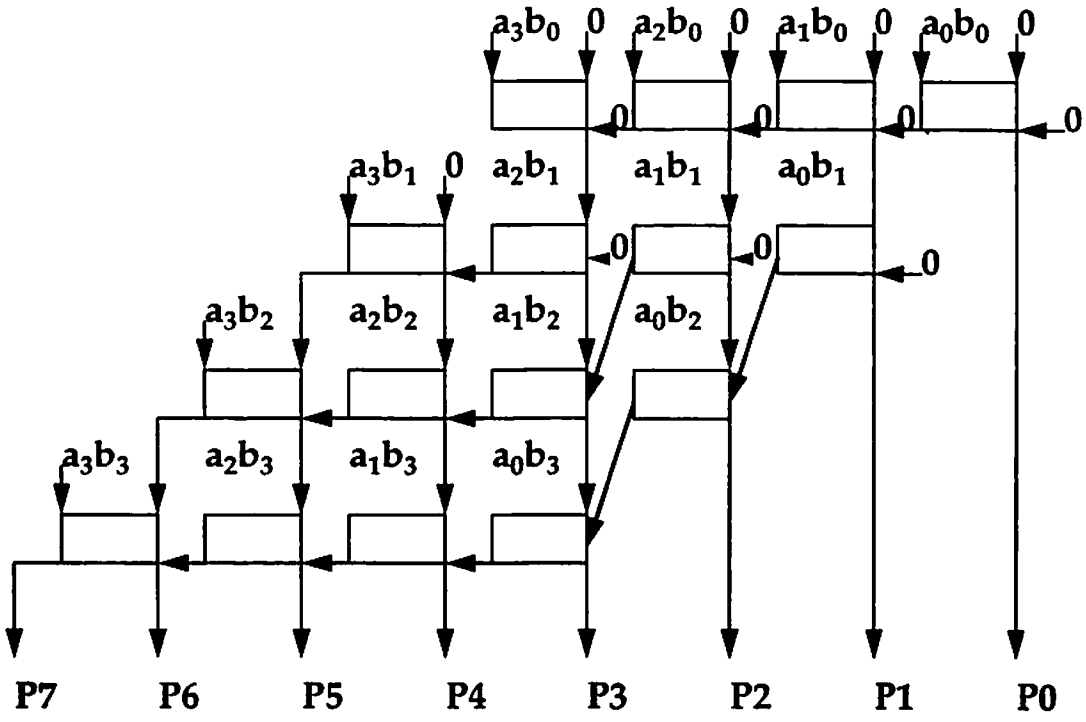


Figure 6. Improved adder block for a 4 × 4-b array multiplier.

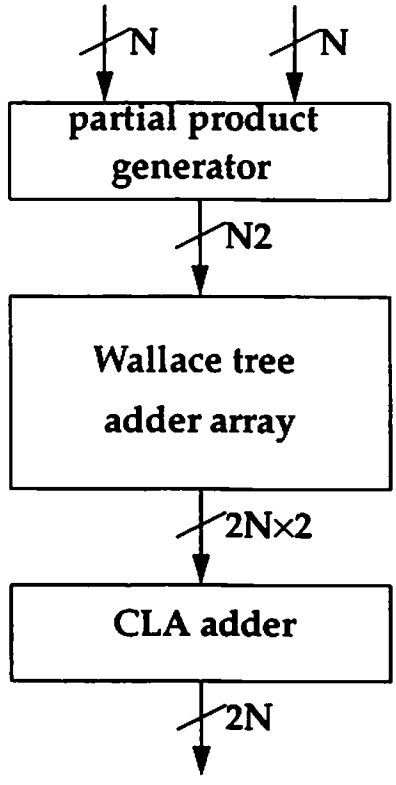


Figure 7. N × N-b Wallace tree multiplier block diagram.

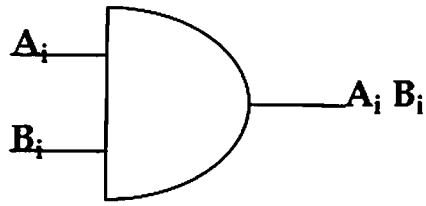
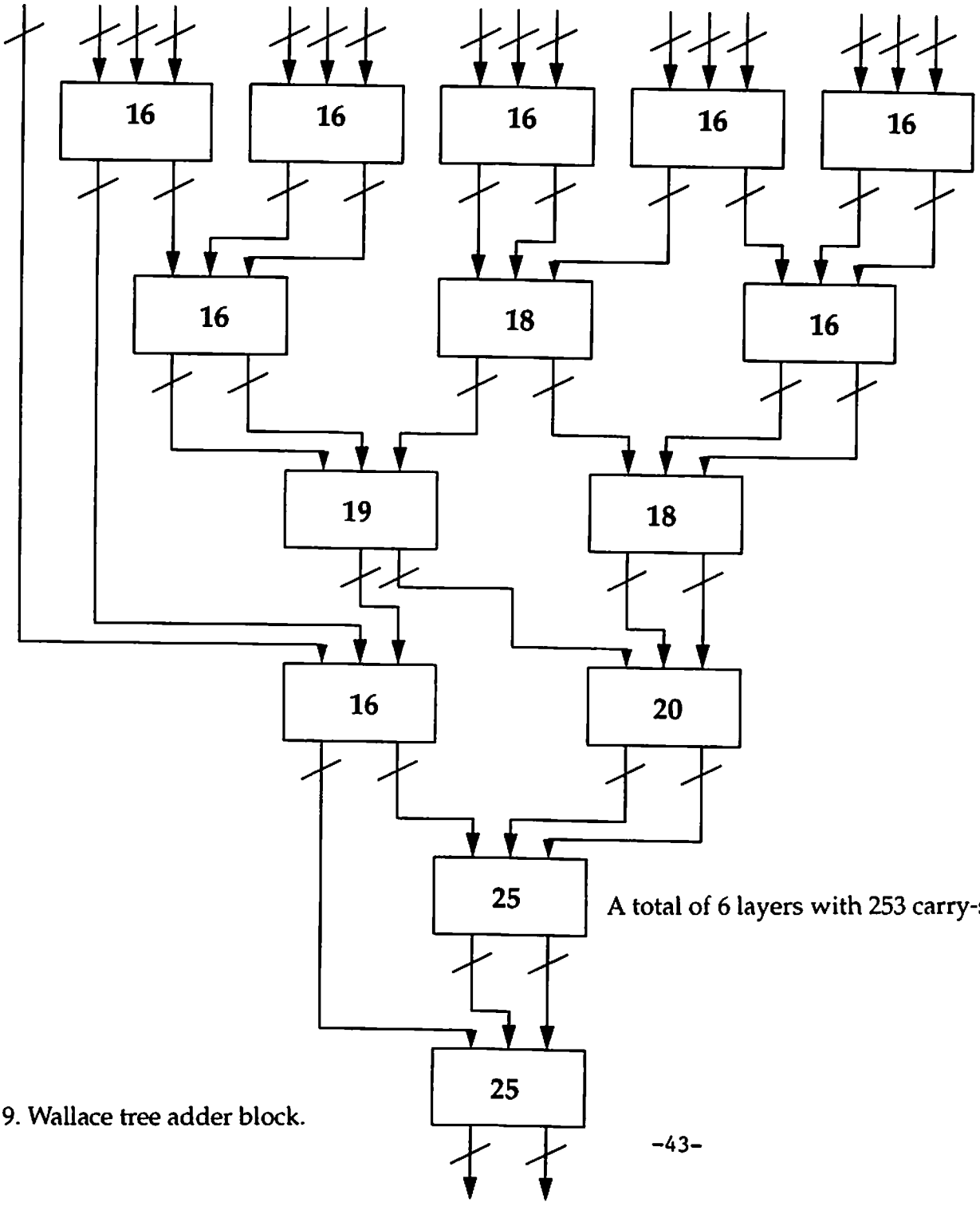


Figure 8. AND gate for partial product generation.



A total of 6 layers with 253 carry-save adders

Figure 9. Wallace tree adder block.

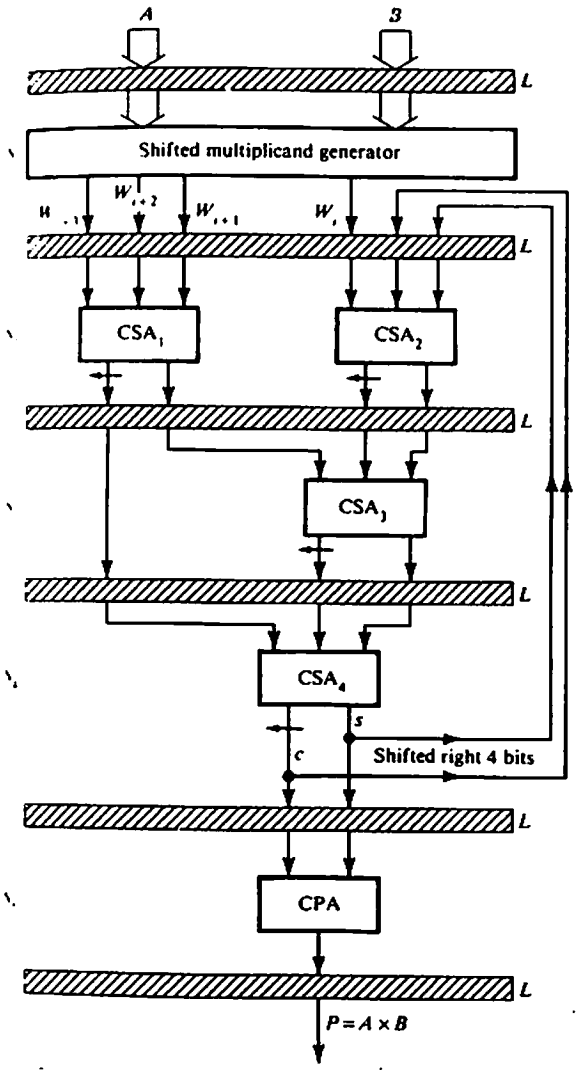


Figure 10. Iterative addition by Wallace tree structure. ([HwBr84])

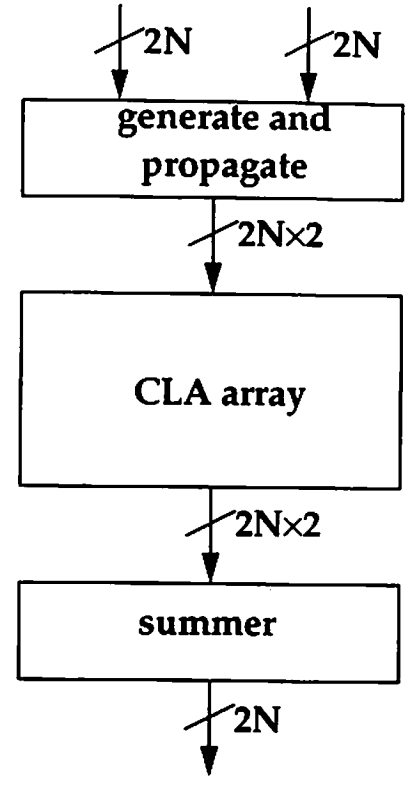


Figure 11. CLA adder block diagram.

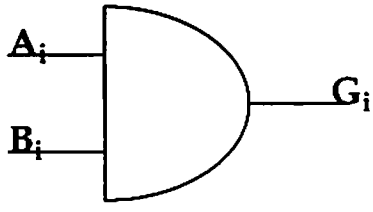


Figure 12. AND gate for 'generate' signal generation.

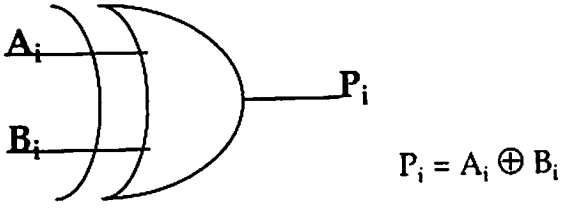


Figure 13. EXOR gate for 'propagate' signal generation.

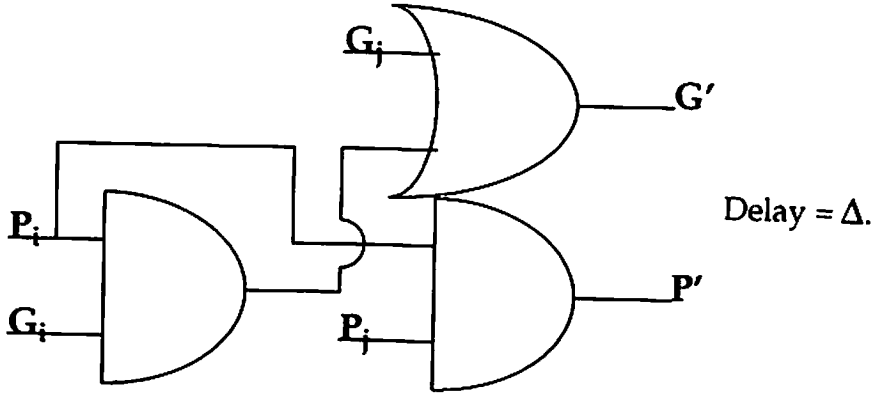
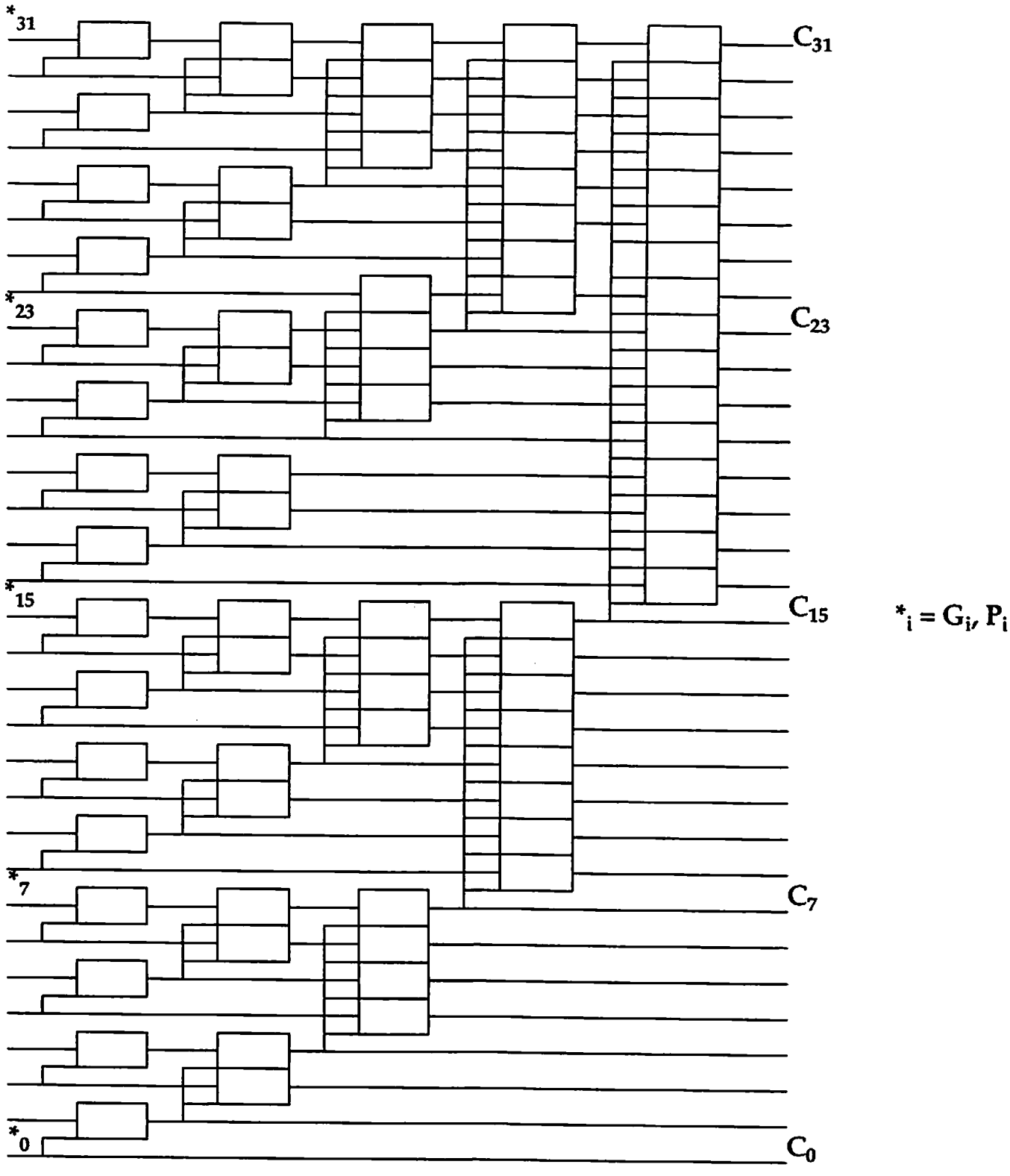
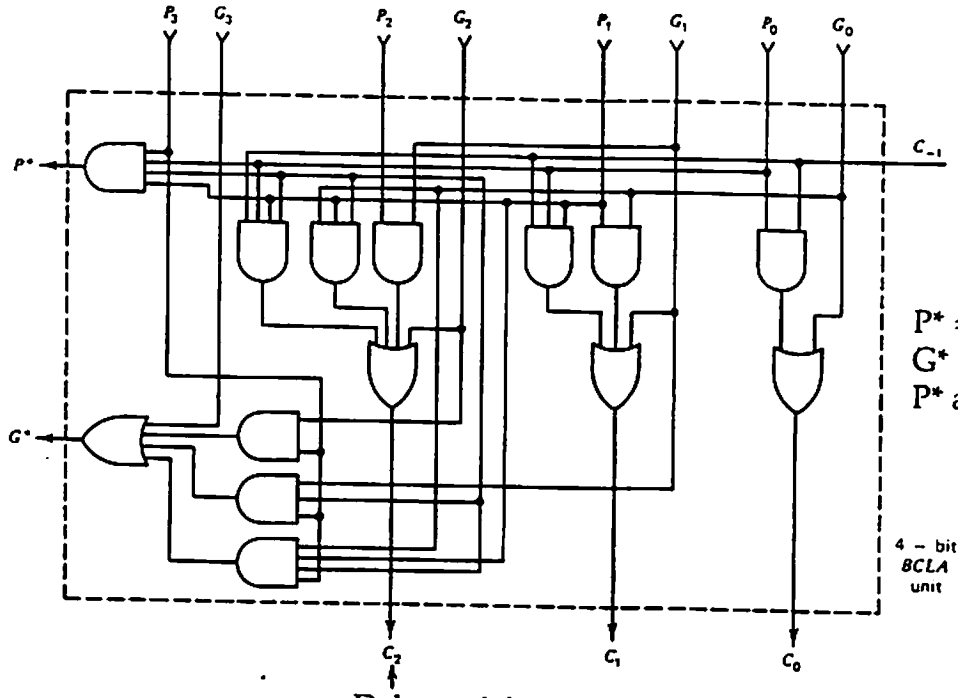


Figure 14. Basic cell for BLC method.



Delay = (log2N)Δ = 5Δ. Large fan-outs at certain nodes.

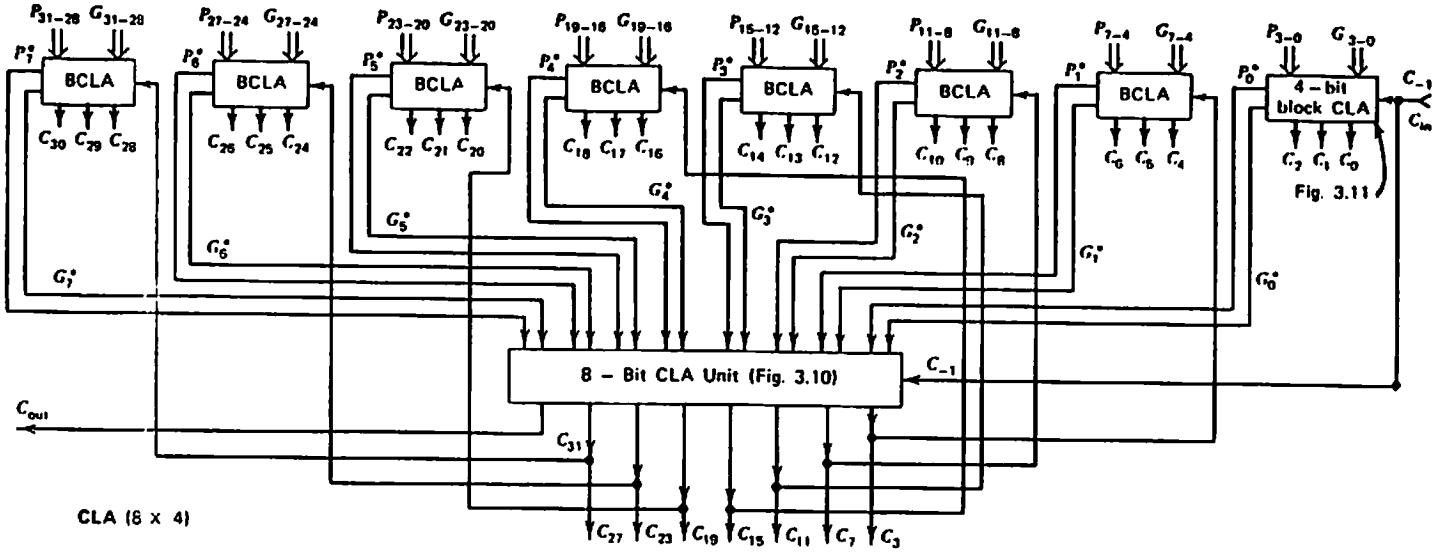
Figure 15. 32-bit BLC.



$P^* = P_0 P_1 P_2 P_3$.
$G^* = G_3 + G_2 P_3 + G_1 P_2 P_3 + G_0 P_1 P_2 P_3$
P^* and G^* are independent of C_{-1}.

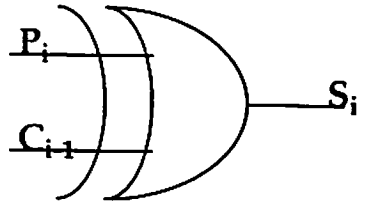
Delay = delay for 3-bit BLC circuit + OR gate delay ($\approx 0.5\Delta$) = $2\Delta + 0.5\Delta = 2.5\Delta$.

Figure 16. 4-bit BCLA circuit. ([Hwan79])



Delay = $\Delta_2 + \Delta_3 + \Delta_1 = 2\Delta + 3\Delta + 2.5\Delta = 7.5\Delta$

Figure 17. 8-by-4 configuration of BCLA method. ([Hwan79])



$S_i = P_i \oplus C_{i-1}$.

Figure 18. EXOR gate for sum generation.

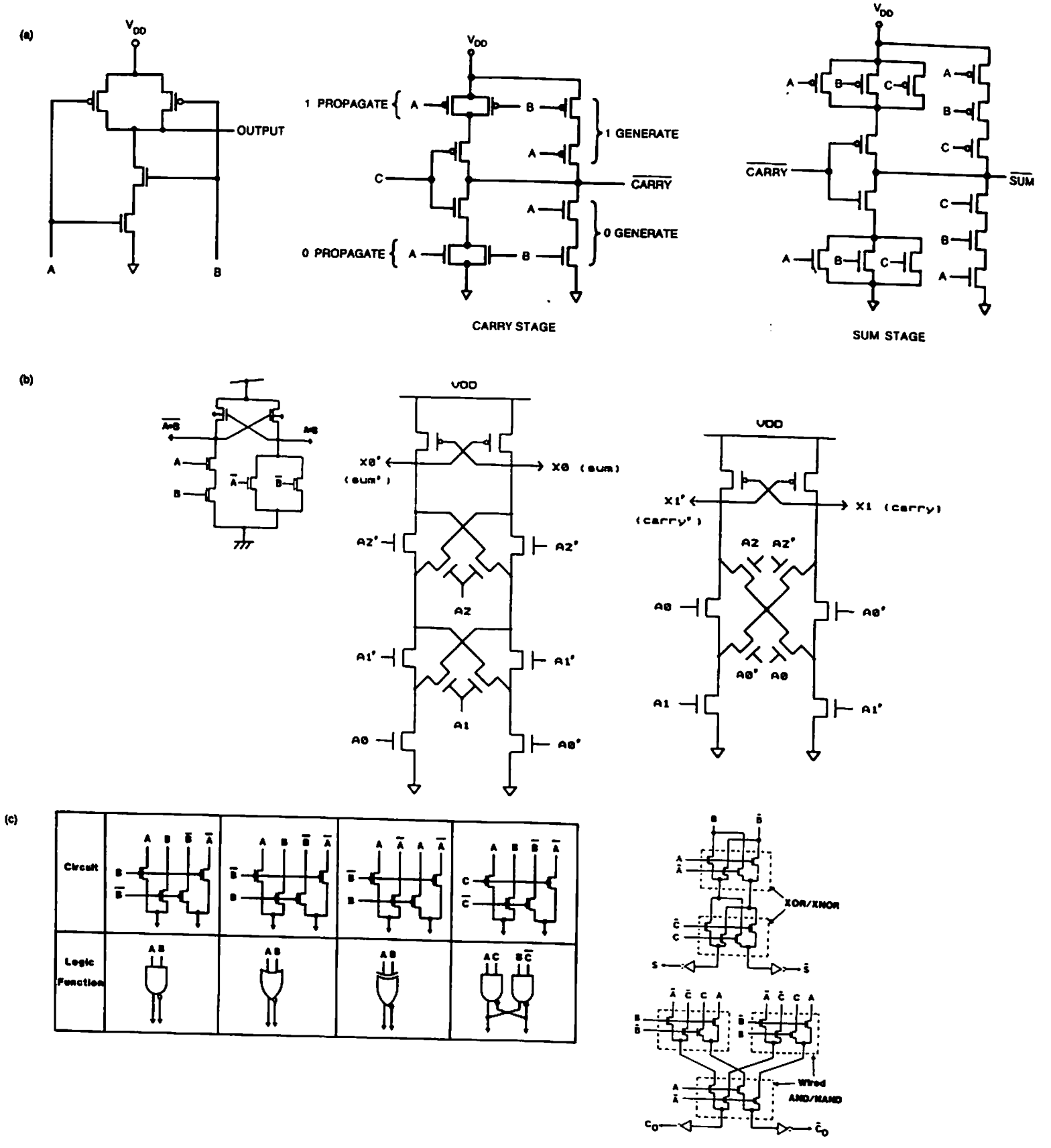


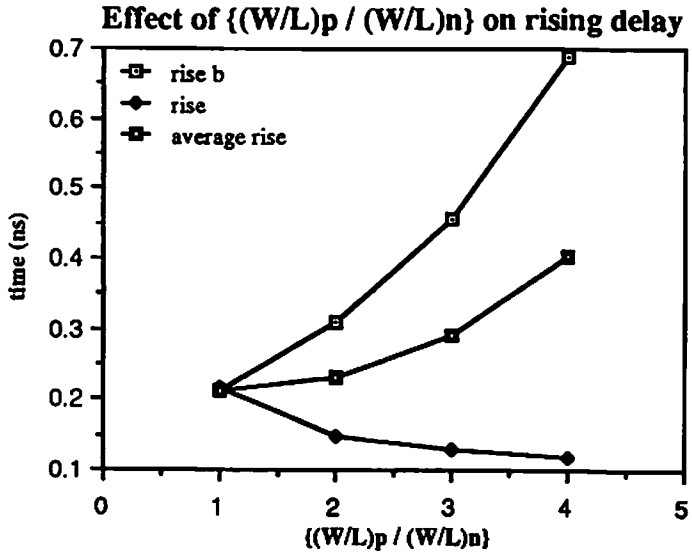
Figure 19. (a) AND, CARRY, and SUM circuits in conventional CMOS. ([WeEs85])
(b) AND, CARRY, and SUM circuits in CVSL implementation. ([SoMi91])
(c) AND, CARRY, and SUM circuits in CPL implementation. ([Yano90])

			CMOS	CVSL	CPL
AND:	delay:	rising:	0.388 ns	0.250 ns	0.201 ns
		falling:	0.298 ns	0.192 ns	0.189 ns
		average:	0.343 ns	0.221 ns	0.195 ns
transistor	count:		6	6	8
CARRY:	delay:	rising:	0.531 ns	0.422 ns	0.289 ns
		falling:	0.512 ns	0.469 ns	0.302 ns
		average:	0.522 ns	0.446 ns	0.297 ns
SUM:	delay:	rising:	0.687 ns	0.657 ns	0.297 ns
		falling:	0.667 ns	0.645 ns	0.320 ns
		average:	0.677 ns	0.651 ns	0.309 ns
transistor	count:		28	22	28

Table 1. Comparison between conventional CMOS, CVSL, and CPL.

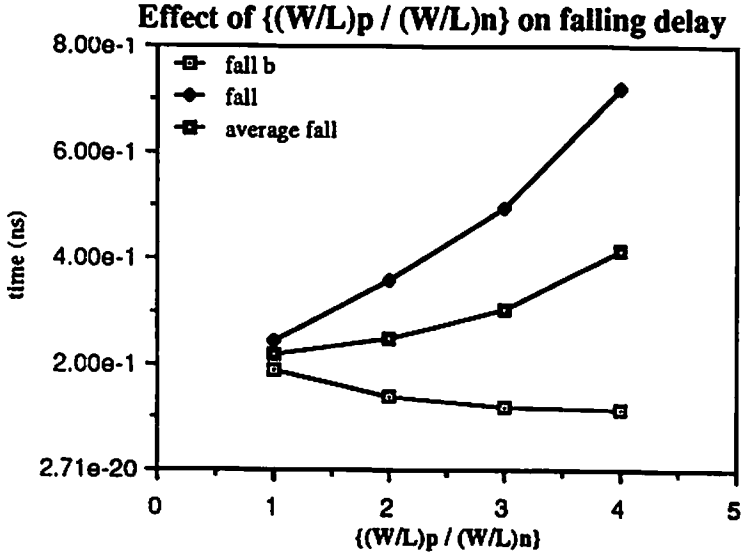
```
* The models
********
* NMOS
.model nfet nmos level=3 phi=0.6 tox=1.1e-8 xj=0.2u tpg=1 vto=0.8186
+ delta=1.757 ld=0 kp=9.1547e-5 uo=596.5 theta=1.085e-1 gamma=0.5266
+ nsub=1.968e16 nfs=5.5e12 vmax=1.942e5 eta=6.654e-2 kappa=1.121e-1
+ cgdo=2.6106e-10 cgso=2.6106e-10 cgbo=6.3402e-10 cj=3.1146e-4 mj=1.0667
+ cjsw=4.3777e-10 mjsw=0.15423 pb=0.8
* Weff = Wdrawn - Delta_W
* The suggested Delta-W is 1.9970e-7
* Note that the rsh is deleted
********
* The PMOS
.model pfet pmos level=3 phi=0.6 tox=1.1e-8 xj=0.2u tpg=-1 vto=-0.9456
+ delta=1.552 ld=0 kp=3.1646e-5 uo=206.2 theta=1.69e-1 gamma=0.4619
+ nsub=1.514e16 nfs=4.999e12 vmax=4.441e5 eta=1.635e-1 kappa=1e1
+ cgdo=2.6981e-11 cgso=2.6981e-11 cgbo=8.6508e-10 cj=4.7864e-4 mj=0.4973
+ cjsw=1.4771e-10 mjsw=0.190593 pb=0.85
* Weff = Wdrawn - Delta_W
* The suggested Delta-W is 3.128e-7
* Note that the rsh is deleted
```

Figure 20. SPICE transistor model parameters.



With minimum size N-transistors.

Figure 21. Effect of the ratio $\{(W_P/L_P)/(W_N/L_N)\}$ on rising edge delay time in CPL AND gate.



With minimum size N-transistors.

Figure 22. Effect of the ratio $\{(W_P/L_P)/(W_N/L_N)\}$ on falling edge delay time in CPL AND gate.

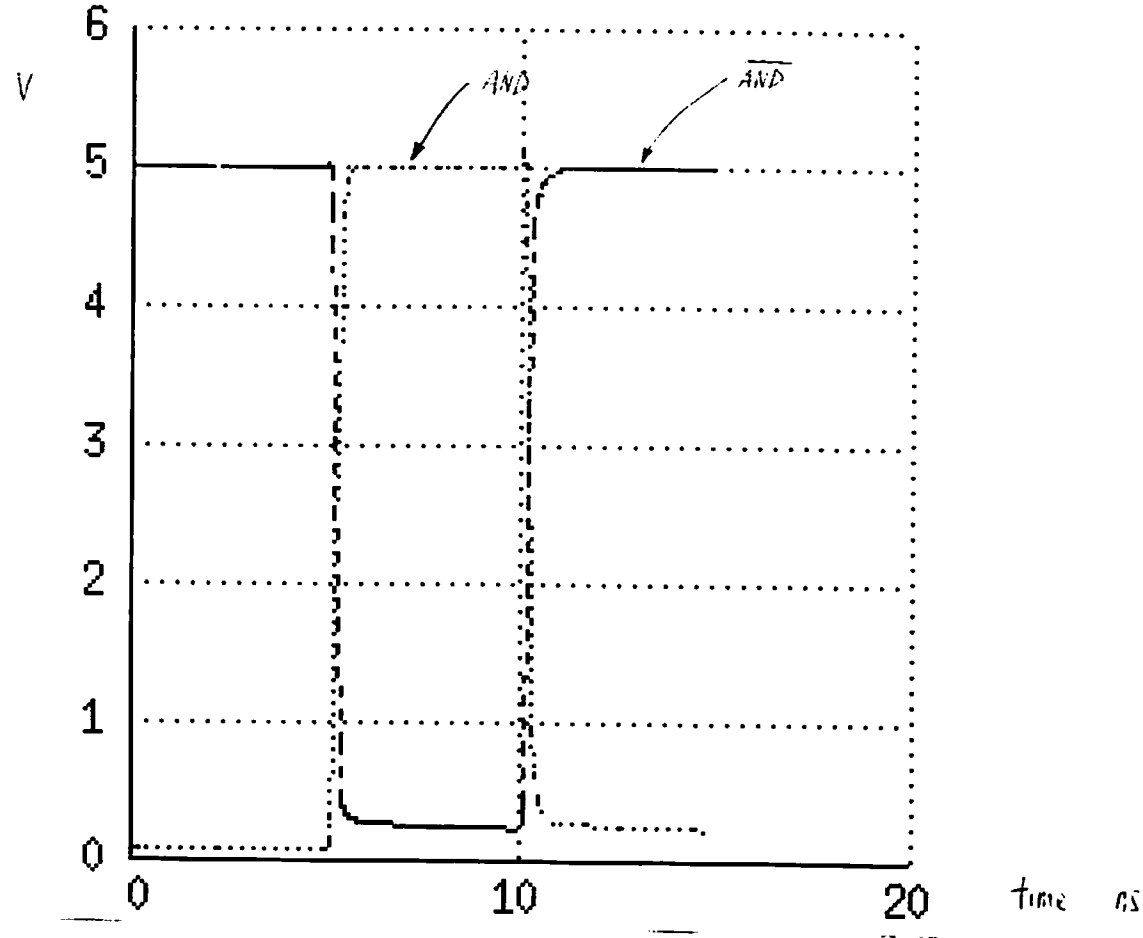


Figure 23. Waveform of CPL AND gate output with $\{(W_P/L_P)/(W_N/L_N)\} = 1$.

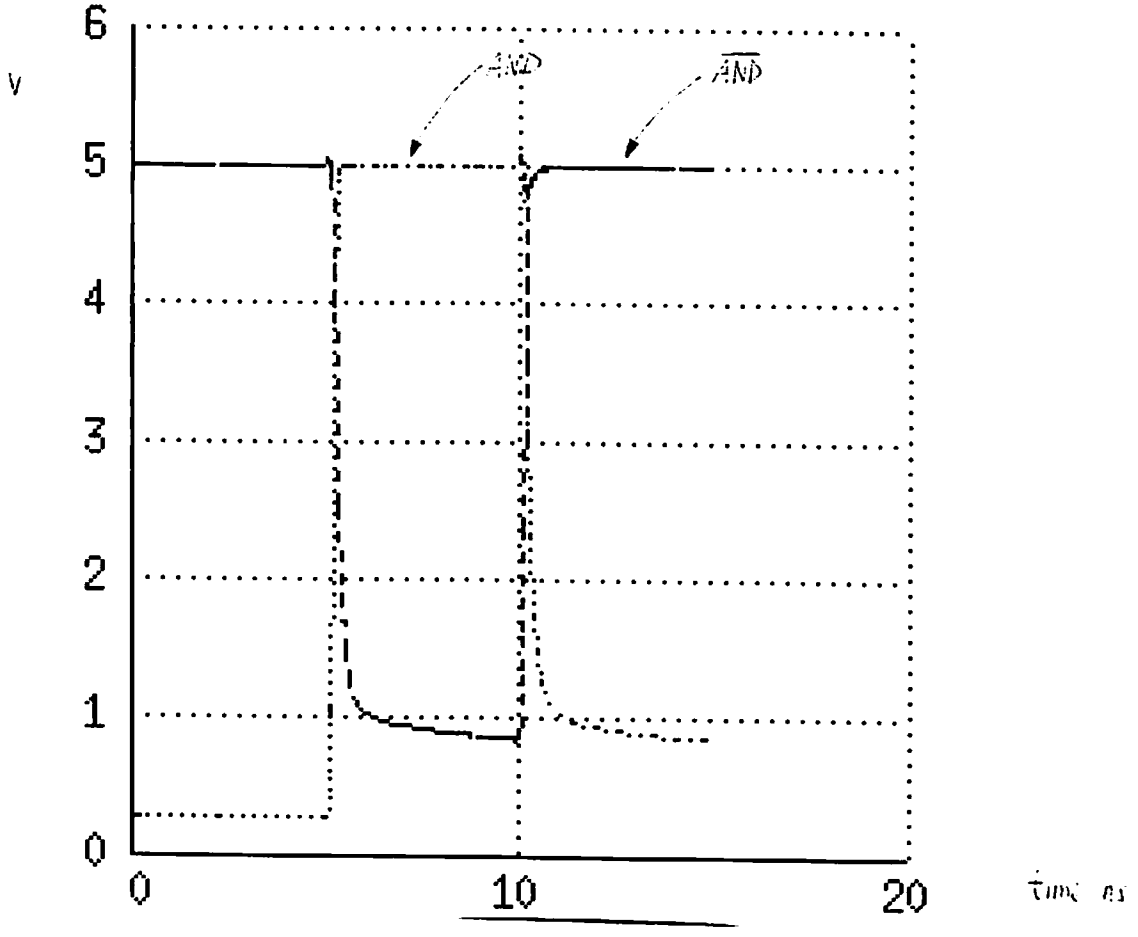
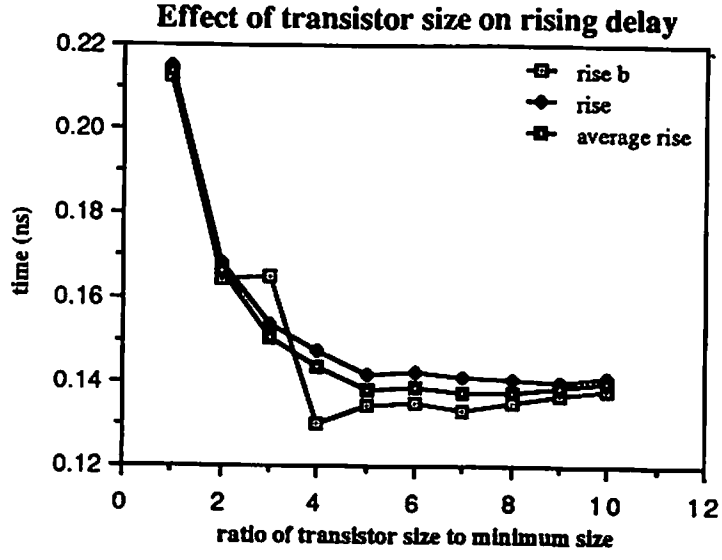
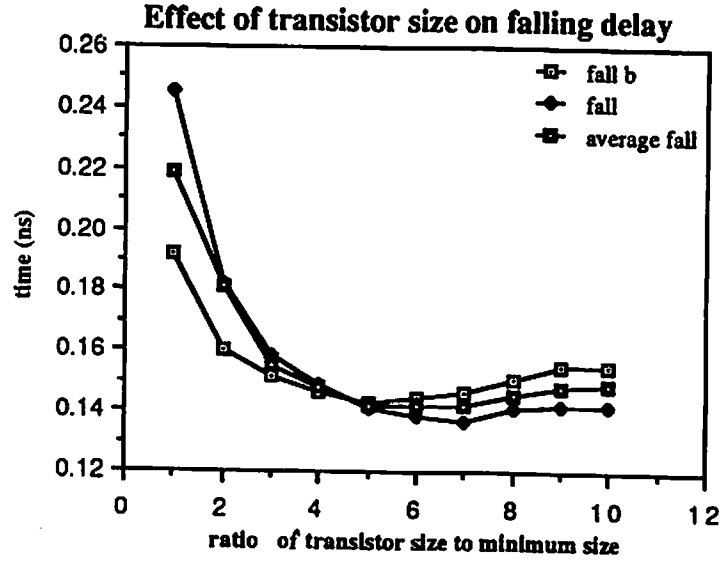


Figure 24. Waveform of CPL AND gate output with $\{(W_P/L_P)/(W_N/L_N)\} = 3$.



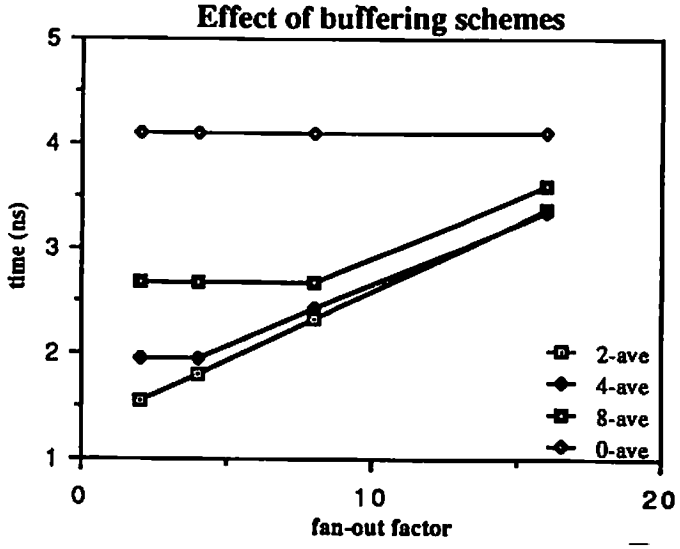
With $|(W_P/L_P)/(W_N/L_N)| = 1$.

Figure 25. Effect of transistor size on rising edge delay time of CPL AND gate.



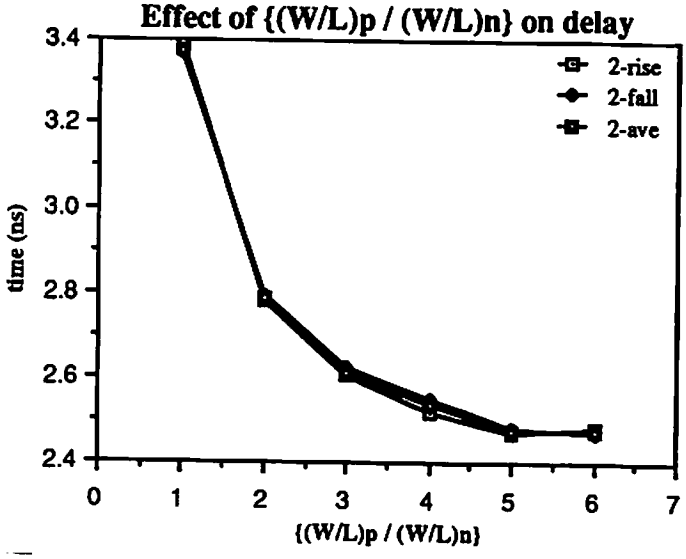
With $|(W_P/L_P)/(W_N/L_N)| = 1$.

Figure 26. Effect of transistor size on falling edge delay time of CPL AND gate.



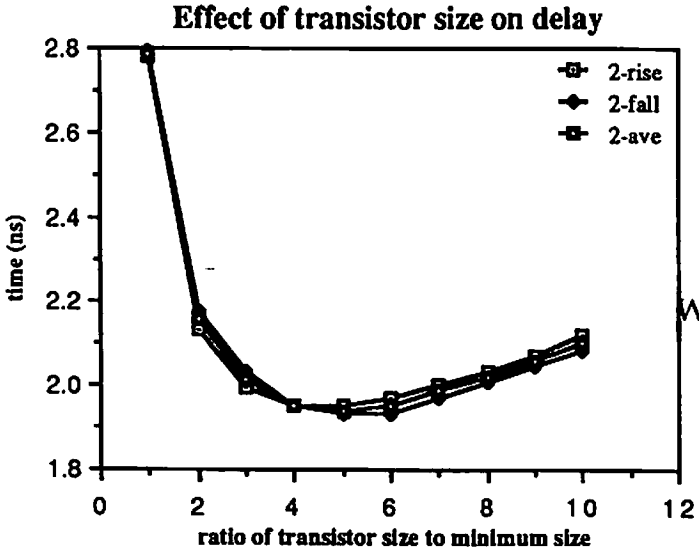
Minimum size P- and N-transistors.

Figure 27. Effect of various buffering schemes on delay time for BLC method.



With fan-out of 16.

Figure 28. Effect of $\{(W_P/L_P)/(W_N/L_N)\}$ of buffer on delay time.



With a fan-out of 16 and $\{(W_P/L_P)/(W_N/L_N)\}$ =

Figure 29. Effect of transistor size on delay time.

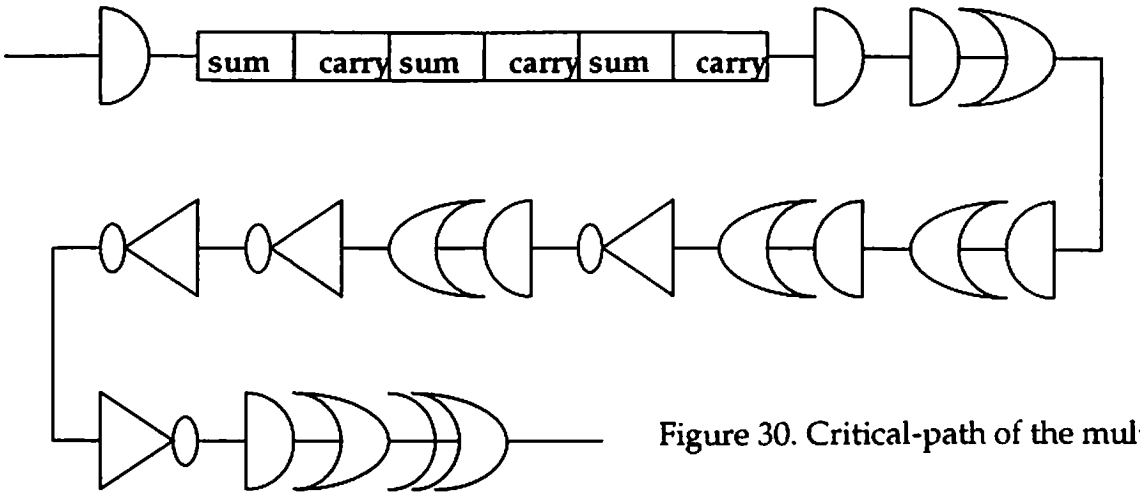


Figure 30. Critical-path of the multiplier circuit.

$$\text{Dealy} = \text{delay}|_{\text{partial product}} + \text{delay}|_{\text{Wallace tree}} + \text{delay}|_{\text{generate \& propagate}} + \text{delay}|_{CLA} + \text{delay}|_{\text{summer}}$$
$$= \text{AND}\,\Delta + 3\text{SUM}\,\Delta + 3\,\text{CARRY}\Delta + \text{AND/EXOR}\,\Delta + 5(\text{AND}\,\Delta + \text{OR}\,\Delta) + 4\text{INV}\,\Delta + \text{EXOR}\,\Delta$$
$$= 0.195 + 3 \times 0.309 + 3 \times 0.296 + 0.195 + 5 \times (0.195 + 0.195) + 4 \times 0.089 + 0.195 \text{ ns}$$
$$= 4.706 \text{ ns}$$

Figure 31. Approximation of critical-path delay from individual cell delays.

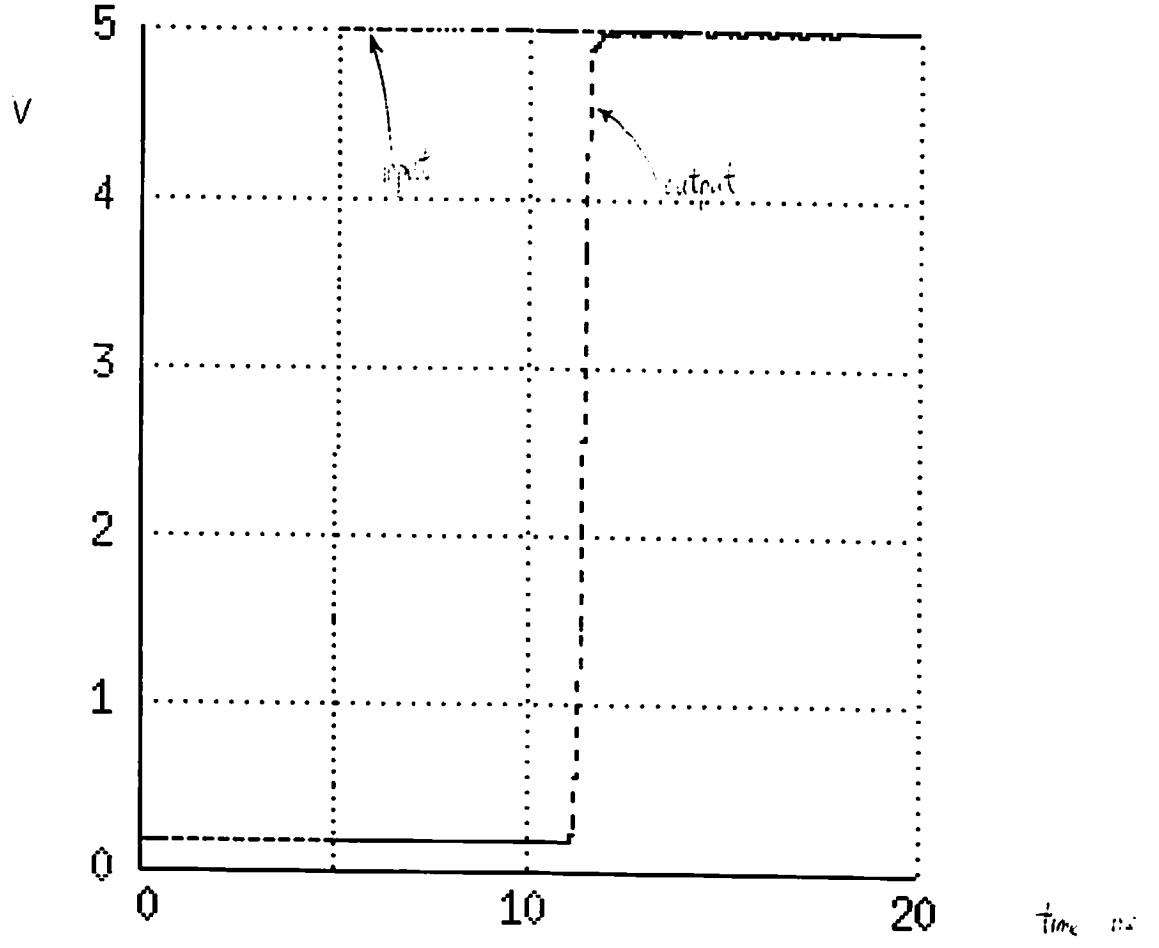


Figure 32. Typical waveform of input and output of critical-path delay simulation.

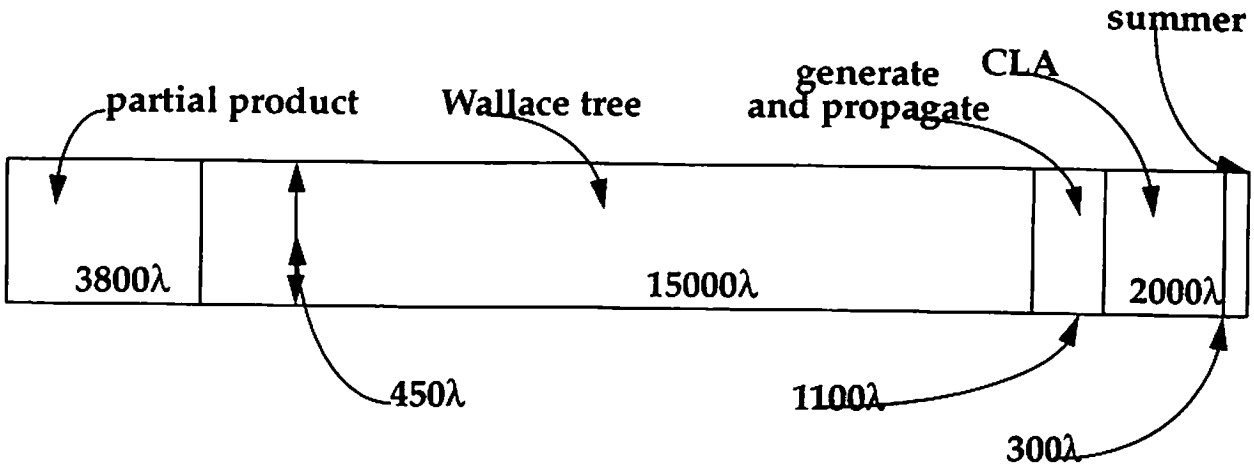


Figure 33. Floorplan satisfying 450λ height constraint.

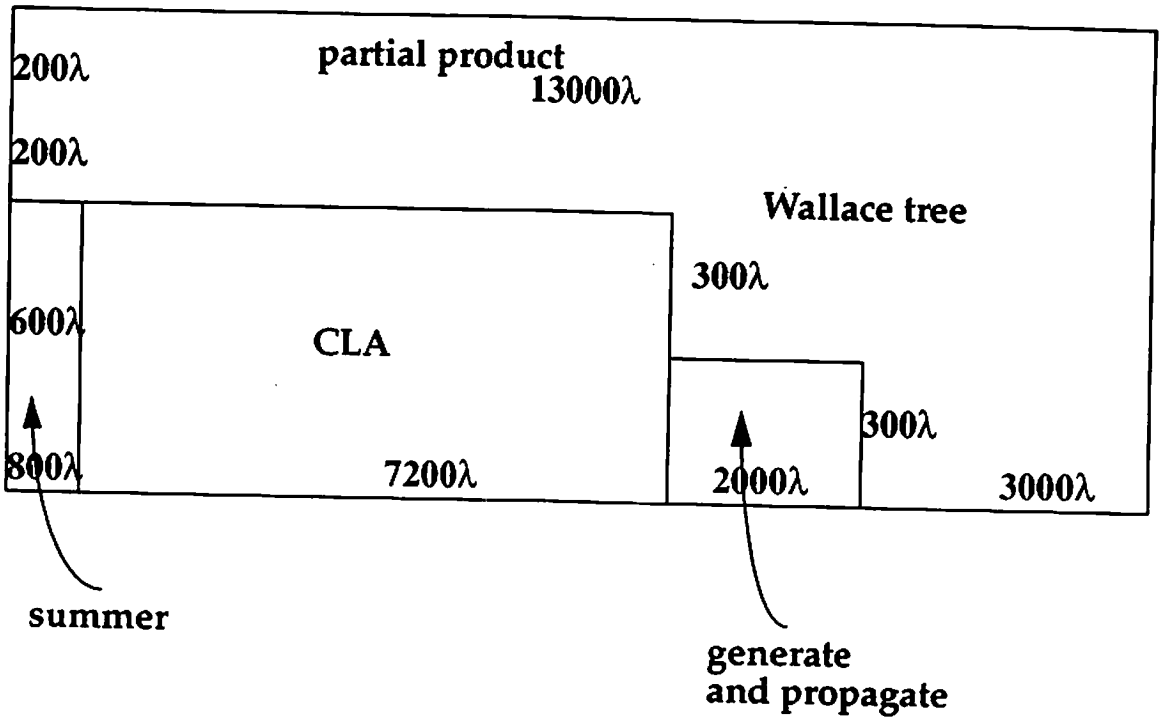


Figure 34. Floorplan for easier routing.

```
** SPICE file created for circuit critical_path
** Technology: scmos

** Sub-circuits transistor model
.include andcell
.include carrycell
.include exorcell
.include invcell
.include orcell
.include sumcell
.include model_131

** Components

** product terms
X0  2  3  4  5  6  7  1 and_cell

** Wallce tree depth
X1  6  7  6  7  8  9  10 11 1 sum_cell
X2  10 11 10 11 12 13 14 15 1 carry_cell
X3  14 15 14 15 16 17 18 19 1 sum_cell
X4  18 19 18 19 20 21 22 23 1 carry_cell
X5  22 23 22 23 24 25 26 27 1 sum_cell
X6  26 27 26 27 28 29 30 31 1 carry_cell

** propagate and generate term
X7  30 31 32 33 34 35 1 and_cell
*X7  30 31 32 33 34 35 1 exor_cell

** lookahead carry tree
X8  34 35 34 35 36 37 1 and_cell
X9  36 37 36 37 38 39 1 or_cell
X10 38 39 38 39 40 41 1 and_cell
X11 40 41 40 41 42 43 1 or_cell
X12 42 43 42 43 44 45 1 and_cell
X13 44 45 44 45 46 47 1 or_cell
X14 47 48 1 inv_cell
X15 46 49 1 inv_cell
X16 48 49 48 49 50 51 1 and_cell
X17 50 51 50 51 52 53 1 or_cell
X18 53 54 1 inv_cell
X19 52 55 1 inv_cell
X20 55 56 1 inv_cell
X21 54 57 1 inv_cell
X22 57 58 1 inv_cell
X23 56 59 1 inv_cell
X24 58 59 58 59 60 61 1 and_cell
X25 60 61 60 61 76 77 1 or_cell

** final adder
X40 76 77 78 79 80 81 1 exor_cell

** Sources

** power supply
Vdd    1  0 dc 5

** product terms
Va     2  0 pwl(0 0 5ns 0 5.1ns 5 20ns 5)
Vab    3  0 pwl(0 5 5ns 5 5.1ns 0 20ns 0)
Vb     4  0 pwl(0 0 5ns 0 5.1ns 5 20ns 5)
Vbb    5  0 pwl(0 5 5ns 5 5.1ns 0 20ns 0)

** Wallce tree depth
V8     8  0 pwl(0 0 5ns 0 5.1ns 5 20ns 5)
V9     9  0 pwl(0 0 0.1ns 5 5ns 5 5.1ns 0 20ns 0)
V12    12 0 dc 0
V13    13 0 dc 5
V16    16 0 pwl(0 0 5ns 0 5.1ns 5 20ns 5)
V17    17 0 pwl(0 5 5ns 5 5.1ns 0 20ns 0)
V20    20 0 dc 0
V21    21 0 dc 5
V24    24 0 pwl(0 0 5ns 0 5.1ns 5 20ns 5)
V25    25 0 pwl(0 5 5ns 5 5.1ns 0 20ns 0)
V28    28 0 dc 0
V29    29 0 dc 5

** propagate and generate term
V32    32 0 dc 5
V33    33 0 dc 0
*V32    32 0 dc 0
*V33    33 0 dc 5

** final adder
V78    78 0 dc 0
V79    79 0 dc 5

** Operations
** product terms
.ic v(2)=0 v(3)=5 v(4)=0 v(5)=5
+ v(8)=0 v(9)=0
+ v(12)=0 v(13)=5
+ v(16)=0 v(17)=5
+ v(20)=0 v(21)=5
+ v(24)=0 v(25)=5
+ v(28)=0 v(29)=5
+ v(32)=5 v(33)=0
+ v(78)=0 v(79)=5

.tran 10ps 20ns

.end
```



```
** SPICE file created for circuit and
** Technology: scmos

.subckt and_cell 122 115 113 108 102 104
* A Ab B Bb AND ANDb Vdd
** NODE: 0 = GND
** NODE: 1 = Vdd
** NODE: 2 = Error
RLUMP0 100 101 364.5
RLUMP1 102 103 163.5
M0 1 101 103 1 pfet L=0.5u W=0.75u
RLUMP2 104 105 162.5
RLUMP3 106 107 364.5
M1 105 107 1 1 pfet L=0.5u W=0.75u
RLUMP4 108 109 380.5
RLUMP5 108 110 380.5
RLUMP6 100 111 364.5
M2 109 110 111 0 nfet L=0.5u W=0.75u
RLUMP7 100 112 364.5
RLUMP8 113 114 310.5
RLUMP9 115 116 62.5
M3 112 114 116 0 nfet L=0.5u W=0.75u
RLUMP10 113 117 310.5
RLUMP11 108 118 380.5
RLUMP12 106 119 364.5
M4 117 118 119 0 nfet L=0.5u W=0.75u
RLUMP13 106 120 364.5
RLUMP14 113 121 310.5
RLUMP15 122 123 62.5
M5 120 121 123 0 nfet L=0.5u W=0.75u
RLUMP16 100 124 364.5
RLUMP17 102 125 163.5
M6 0 124 125 0 nfet L=0.5u W=0.75u
RLUMP18 104 126 162.5
RLUMP19 106 127 364.5
M7 126 127 0 0 nfet L=0.5u W=0.75u
** NODE: 0 = GND!
** NODE: 122 = A
** NODE: 115 = Ab
C0 113 0 1.910f
** NODE: 113 = B
C1 104 0 2.604f
** NODE: 104 = ANDb
C2 106 0 1.910f
** NODE: 106 = 6_20_30#
C3 1 0 2.257f
** NODE: 1 = Vdd!
C4 100 0 1.910f
** NODE: 100 = 6_20_78#
C5 102 0 2.951f
** NODE: 102 = AND
** NODE: 108 = Bb
.ends and_cell
```

EE 577 Term Project Report

*High Performance Multiplier Design a 5.5(ns) 32bit*32bit integer multiplier*

Professor: Bing Sheu, Ph.D.
Student: Koa-Shin Wu
S.S.#: 888-05-0865

Abstract

*In this term project report, a high performance multiplier which can output a 32bit*32bit product every 5.5 ns is desired. It will be implemented in CMOS VLSI circuit for a 0.25 um technology. Various high speed multiplier circuits were examined based on our design criterions. Our high performance multiplier achieved this speed requirement by employing pipelining. Finally, SPICE simulation results shows that our design for this high performance multiplier can generate a 64-bit product per 5.426 ns.*

1. Sequential Multiplier

Traditionally, multiplication is done by some form of repeated addition. One of the simplest multiplication methods is to add the multiplicand Y to itself X times, where X is the multiplier. This multiplication algorithm requires a simple logic circuit to implement and can easily accommodate complicated number codes. It can be realized by three up-down counters, X, Y, and Z, which store the multiplier, multiplicand and product, respectively, and the product P is formed by incrementing the counter Z a total of P times.[1]

More commonly, the multiplication of two fixed-point numbers is implemented by repeated add-shift operations, using an arithmetic logic unit (ALU) which has built in add and shift functions. This arithmetic logic unit is included in some smaller processors such as microprocessors, simple microcontrollers or RISCs.[2]

Both multiplication algorithms presented in the previous paragraphs are simple, but their execution speed is also fairly slow due to the nature of sequential execution. In the following, we will discuss various techniques for speeding up the multiplication process.

2. Speeding Up Multiplication

Usually, methods that increase the speed of multiplication can be divided into two classes: those that use a single adder (e.g., redundant multiplication, higher-radix multiplication) and those that use multiple adders. (e.g., a Wallace-tree multiplier, a full-adder array multiplier).[3] Since we are concerning with high performance multiplier design; only techniques using multiple adders will be discussed in this context.

3. Design Objectives and Criterions

Our design objectives of a high performance multiplier are to have the

1. hardware cost as low as possible.

2. execution time as low as possible; i.e., speed should be as fast as possible.

Moreover, this multiplier should have a very modular structure that can easily be implemented in CMOS VLSI circuit.

Therefore, it's a good criterion to calculate cost-speed product and choose the one which has the least value. The precise representation of both the cost and speed of a multiplier are complex and highly technology dependent. It depends on such factors as multiplier architecture, implementing circuitry, and signal drive capability. Hence, a simplified and technology independent representation method will be derived in this report and use it to represent the cost and speed of a multiplier.

4. Definition of the cost and speed of a multiplier

Cost : The number of full adders used to implement a specific multiplier will be considered as the hardware cost of this multiplier.

Speed : In general, the time a circuit takes to produce an output is proportional to the number of logic levels through which a signal travels. Therefore, when comparing execution time for nonpipelined multipliers, we simply compare the number of adder delays needed to output a product. For pipelined multipliers, same concept applies to the slowest pipeline stage.

It is important to realize that the total delay is not only well correlated with the number of stages but also depends strongly on the delay associated with the

interconnection wiring capacitance, technology and other factors.[5]

5. Nonpipelined Parallel Multipliers

Several most widespread used multipliers are summarized in Table-1. Detailed circuits can be found in related references. A comparison based on *Normalized Cost-Speed Product* is also presented in this table. The normalized cost-speed product is defined as the ratio between the cost-speed product value of a particular multiplier to the smallest cost-speed product value in this table. The smaller the numeric value, the more the cost-effective design.

From Table-1, a 8-pass CSA-tree (Wallace-tree) multiplier has the least cost-speed product value. In other words, it is the most cost-effective design based on our design criterion. However, its execution time is unacceptable long. It needs 40 carry propagation delays to generate one product.

The fastest one is the 1-pass CSA-tree with CPA (Carry Propagation Adder) using CLA (Carry Lookahead Adder). Actually, when multiplier is built out of MSI parts, a Wallace-tree is the design of choice for its high speed. There is, however, a problem with implementing it in VLSI. If we fill in all the adders and paths for the Wallace-tree, we will discover that it does not have nice, regular structure. That's why VLSI designers often use other designs such as binary tree multiplier.[3]

Today, pipelining is the key implementation technique used to make high speed multiplier. For example, Weitek's WTL3164 high performance floating processing unit(FPU), which can deliver a peak floating-point performance of 15 MFLOPS at 30 MHz.

*A Comparison of Nonpipelined 32b*32b Multipliers*

Type	*Cost* (*number of full adders*)	*Speed* (*number of adder delays*)	*Cost-Speed Product*	*Normalized Cost-Speed Product*
Combinational Full-adder Array [1,4]	32*32	63	32*32*63	*5.1*
1-Pass CSA-Tree Final stage : CPA= Ripple Carry Adder [2,3]	64*31	64+8	64*31*72	*11.2*
1-Pass CSA-Tree Final stage : CPA= 4-bit CLA [2,3]	64*31	16+8	64*31*24	*3.72*
8-Pass CSA-Tree Final stage : CPA= Ripple Carry Adder [2,3]	5*64	64+24	5*64*88	*2.2*
8-Pass CSA-Tree Final stage : CPA= 4-bit CLA [2,3]	5*64	16+24	5*64*40	*1*

Table-1

** CPA: Carry Propagation Adder. It may propagate its carries using ripples, carry lookahead or some other method.[*

** CSA: Carry Save Adder. For simplicity, each CSA is 64-bit long.[1]*

** CLA: Carry Lookahead Adder.*

By examining the process of multiplication, it is clear that the multiplication process is equivalent to the addition of multiple partial products. Multiple partial products addition can be realized with multiple adders. This serves the purpose of pipeline multiplier.[2] All methods presented in Table-1 can be pipelined to increase the system throughput.

6. Pipelined Parallel Multipliers

With the multipliers in Table-1, the corresponding pipelined multipliers' cost, speed, cost-speed product and normalized cost-speed are listed in Table-2. The hardware cost does not include the cost of latches, because the number of latches in a pipelined multiplier are proportional to the number of full adders in this multiplier.[1]

It is clearly that the 32-stage ripple carry full-adder array and the 1-pass CSA-tree (which has 8 CSA stages) with CPA using 4-bit CLA has the least cost-product value. The drawback of the ripple carry full-adder array is its long execution time. On the other hand, the 1-pass CSA-tree has the fastest speed, but it pays the highest hardware cost as well. Can we build a multiplier that has the fastest speed but not so costly? Obviously, the answer is positive.

We also notice that the ripple carry 1-bit full-adder array has more regular structure and balanced pipeline stage than that in a 1-pass CSA-tree multiplier. Thus, our high performance multiplier will be build based on this architecture. In addition, its low speed is caused by the carry signal which is rippling through stages in order to generate the correct sum and carry signals. This problem is easily solved by introducing a carry lookahead circuit to produce all required carry signals.

*A Comparison of Pipelined 32b*32b Multipliers*

Type	*Cost*	*Max. Speed*	*Cost-Speed Product*	*Normalized Cost-Speed Product*
Ripple Carry Full-adder Array [1]	32*32	32	32*32*32	4
1-Pass CSA-Tree CPA=64-bit *RCA* [2,3]	64*31	64	64*31*64	16
1-Pass CSA-Tree CPA=4-bit CLA [2]	64*31	16	64*31*16	4
8-Pass CSA-Tree CPA= 64-bit RCA [1,2,3]	64*5	64*8 *(MAL=8)*	64*5*64*8	20
8-Pass CSA-Tree CPA=4-bit CLA [1,2,3]	64*5	16*8 (MAL=8)	64*5*16*8	5
(My Design) 4-bit CLA Array	*32*32*	*8*	32*32*8	1

Table-2

RCA: Ripple Carry Adder.

MAL:Minimum Average Latency. A pipelined system's maximum throughput is achieved by an optimal scheduling strategy that achieves the MAL without collisions. [2]

7. Direct Implementation of Carry Lookahead Circuit(CLC)

The design for this multiplier is relatively straight-forward, with the main attention paid to minimize the time associated with carry propagation.

For an n-bit adder, the carry signal of the i+1th stage, Ci, can be expressed as

Ci=Gi+PiCi-1

where Gi=AiBi generate signal

Pi=Ai+Bi propagate signal

Ai, Bi adder inputs.

expending this recursive formula yields

Ci=Gi+PiGi-1+PiPi-1Gi-2+...+PiPi-1...P1G0+PiPi-1...P1P0Cin

Unfortunately, Direct implementation of above equation has three problems:[3]

1. A CLC for n bits requires a fan-in of n+1 at the OR gate as well as at the rightmost AND gate.

2. The Pn-1 signal must drive n AND gates.

3. The rather irregular structure makes it impractical to build a large CLC.

Thus, we will limit the number of carry lookahead stages to 4. The carries in a 4-stage CLC; for example, C0 and C3 are defined by

C0=G0+P0Cin

C3=G3+P3G2+P3P2G1+P3P2P1G0+P3P2P1P0Cin

If we realize C0(C3) in CMOS circuit directly, we need 2 logic levels to fonn P0(P3) and another 2 logic levels for C0(C3). In a multiplier, Ai or Bi is formed by logical ANDing of Xj and Yi; where X, Y is the multiplier, multiplicand, respectively, as shown in Fig-1. This needs 2 more logic levels. Hence, a total of 6 logic levels are required.

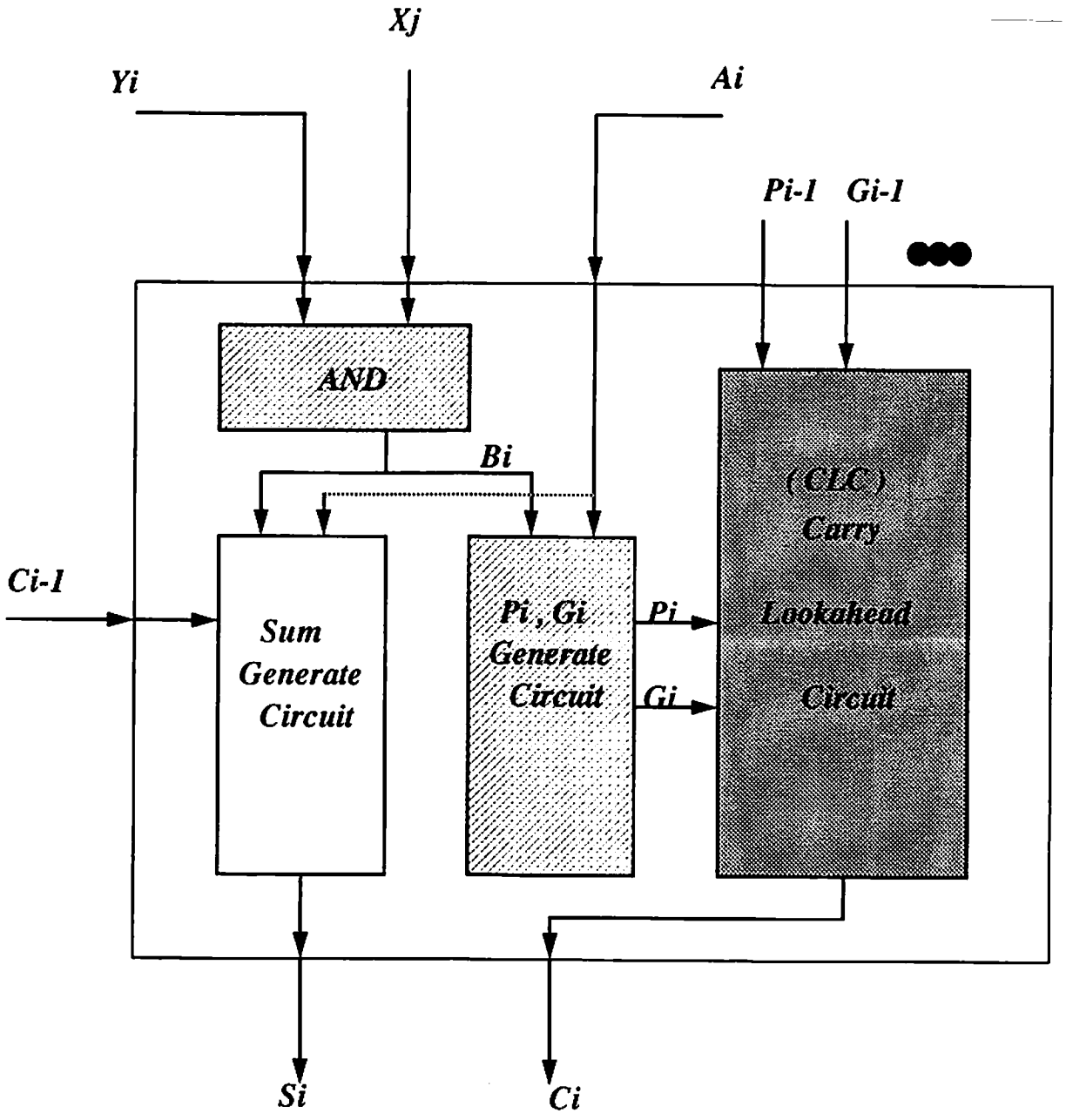


Figure-1 Direct Implementation of the CLC for a adder in the multiplier

Apparently, a multiplier based on this structure will have much longer execution time.

8. High Speed M-block Design

We may rewrite C0, C1, C2, C3 in a 4-bit carry lookahead adder as follows:

$C0=\overline{\overline{G0+P0Cin}}=\overline{\overline{G0}(\overline{P0}+\overline{Cin})}$

$C1=\overline{\overline{G1+P1(G0+P0Cin)}}=\overline{\overline{G1}(\overline{P1}+(\overline{G0}(\overline{P0}+\overline{Cin})))}$

$\overline{C2}=\overline{G2+P2C1}$ $\quad .C2=\overline{\overline{C2}}$

$\overline{C3}=\overline{G3+P3C2}=\overline{G3+P3(G2+P2C1)}$ $\quad .C3=\overline{\overline{C3}}$

where $\overline{Pi}=\overline{Ai+Bi}=\overline{Ai+XjYi}$ $\quad .i=0, 1$

$\overline{Gi}=\overline{AiBi}=\overline{AiXjYi}$ $\quad .i=0, 1$

$Pi=Ai+Bi=Ai+XjYi$ $\quad .i=2, 3$

$Gi=AiBi=AiXjYi$ $\quad .i=2, 3$

where Xj is the jth bit of the multiplier

Yi is the ith bit of the multiplicand.

Similarly, Sum output signals can be expressed as

$S0=\overline{\overline{C0}(A0+B0+Cin)+A0B0Cin}=\overline{C0(\overline{A0}+\overline{B0}+\overline{Cin})+\overline{A0}\,\overline{B0}\,\overline{Cin}}$

$S1=\overline{C1(\overline{A1}+\overline{B1}+\overline{C0})+\overline{A1}\,\overline{B1}\,\overline{C0}}$

$\overline{S2}=\overline{\overline{C2}(A2+B2+C1)+A2B2C1}$

$\overline{S3}=\overline{\overline{C3}(A3+B3+C2)+A3B3C2}$

Form these equations, we may define a cell " *M-block* " as shown in Figure-2. Also from Fig-2, it is interesting to notice that the same sum circuit layout can be used to generate S0 (S1) and $\overline{S2}$ ($\overline{S3}$) although these two circuits operate on totally complement data.

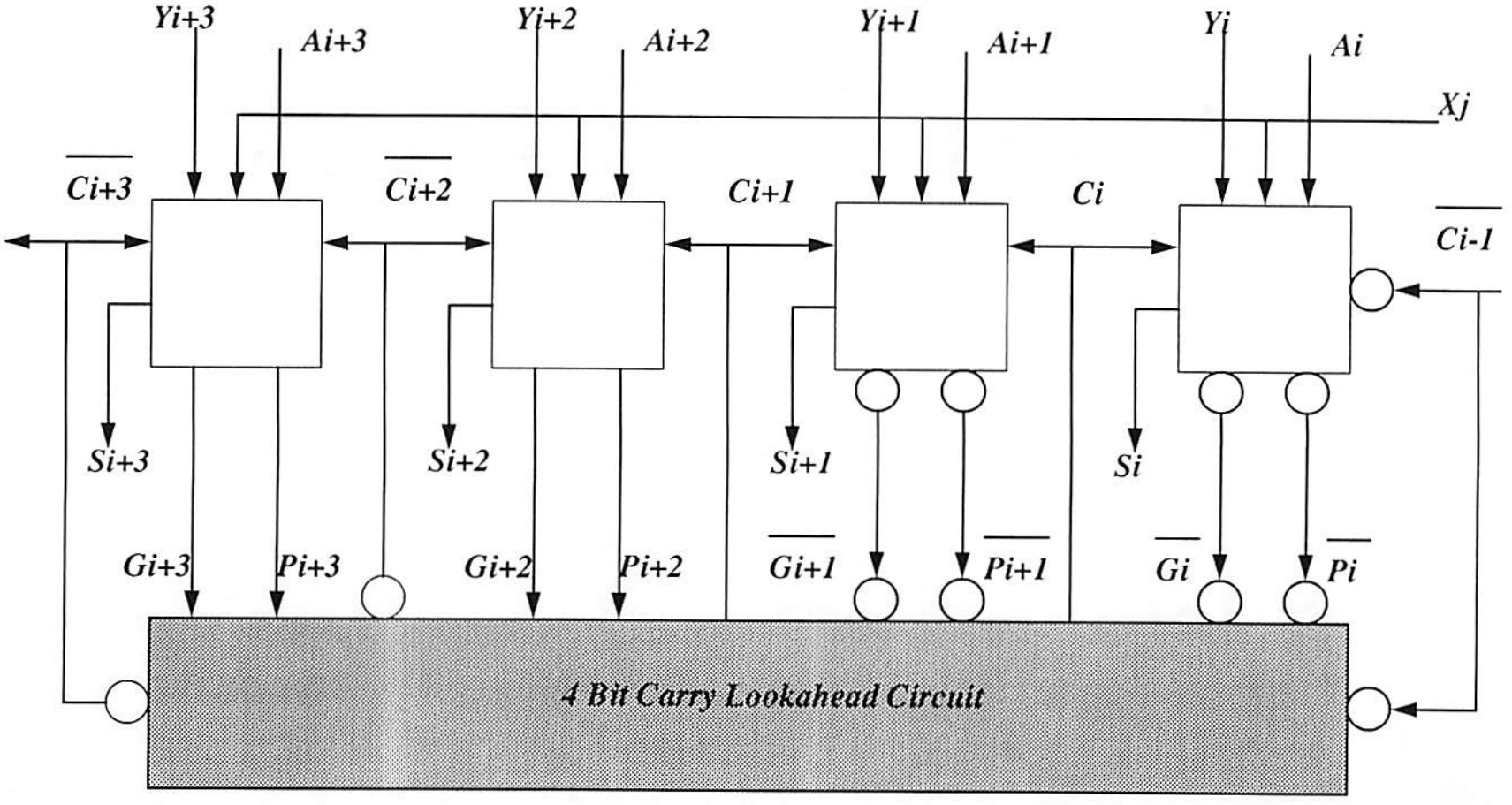


Figure-2 M-block for our high performance multiplier

The M-block will be used as a building block to construct our high performance multiplier.

A implementation of this M-block in a static CMOS circuit will need 1 logic level to produce $\overline{Pi}$ and $\overline{Gi}$ (i=0, 1), 1 logic level to produce C0 and C1, and another 1 logic level to generate $\overline{C2}$, $\overline{C3}$. Thus, a total of only 3 logic levels are required to produce C0, C1, $\overline{C2}$, and $\overline{C3}$. In later sections, we will use $\overline{C2}$ and $\overline{C3}$ instead of C2 and C3 to implement our pipelined multiplier.

Compared with direct implementation of a 4-bit carry lookahead circuit (which needs 6 logic level delays to generate Ci), this is a vast improvement. Furthermore, a 32-bit multiplier which is constructed based on our M-blocks will have about the same amount of logic gates as one is built based on 4-bit ripple carry adders, which contains 4 full-adders and some other logic gates for generating 1-bit logical ANDing of multiplier's bits and multiplicand's bits.

That's why we represent the hardware cost of a multiplier in Table-1 and Table-2 by simply counting the number of full adders used to implement it.

9. Implementation of a High Performance Multiplier

Since we employ pipelining to implement this specific 32bit*32bit high performance multiplier, high speed latches will be placed between any two pipeline stages to allow all necessary data items to be transferred between them without interfering with one another. The latches are clocked at the maximum possible rate which allows data to be transferred reliably between stages.

Our high performance 32bit*32bit multiplier will contain a total of

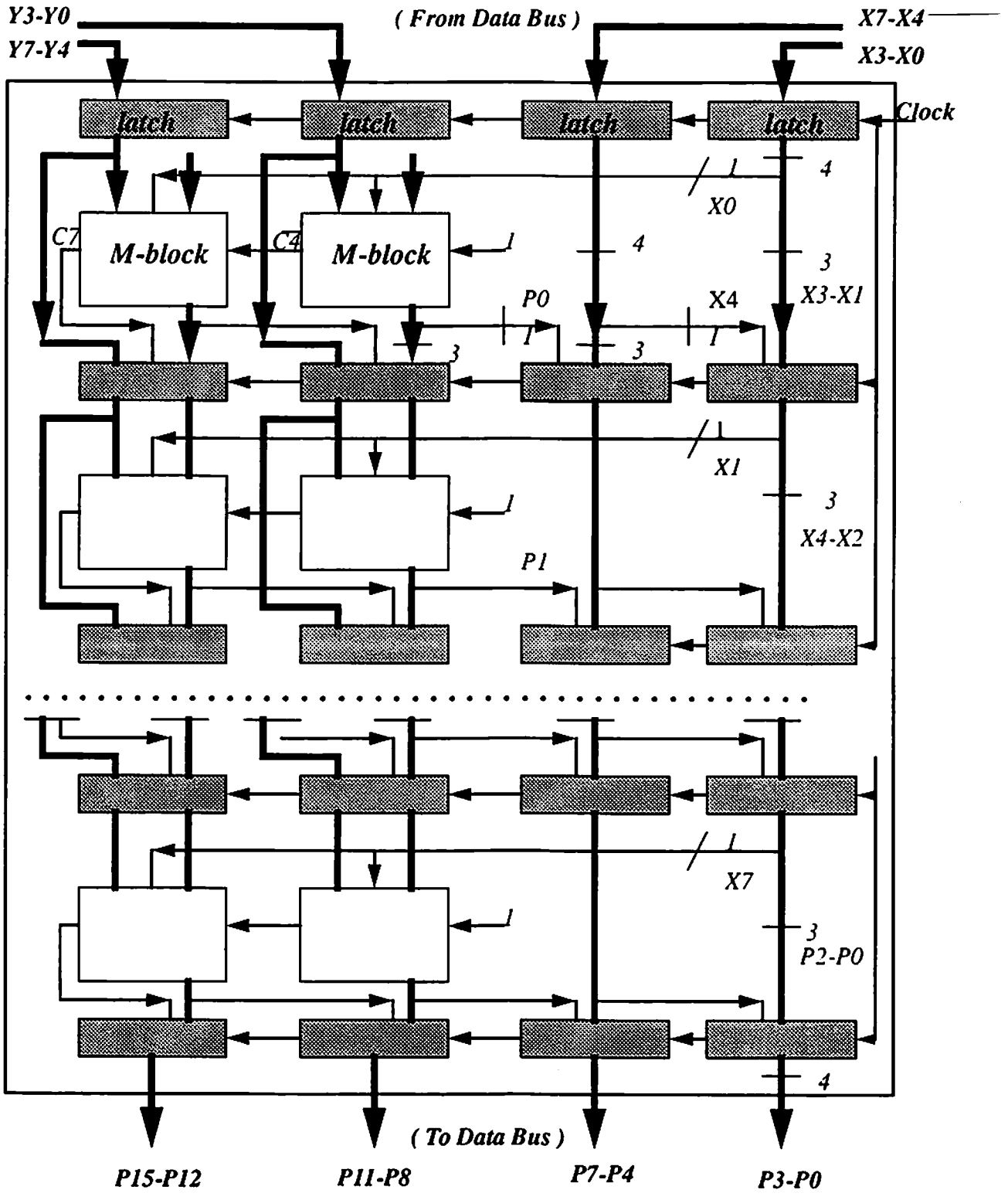


*Figure 3 : A pipelined 8b*8b multiplier using M-block array*

32 pipeline stages. Each pipeline stage contains 8 M-blocks, 32 latches for storing multiplicand, 32 latches for storing partial products, and another 32 latches for storing products and unused multiplier bits. In every stage, the carry bit out of a M-block is transferred to the carry input of its left neighbor.

Clearly, a 32-stage pipelined multiplier of this type can overlap the computation of 32 separate products, as required, for example, when multiplying fixed-point vectors, and can generate a new result every clock.

For the demonstration of interconnections between stages and M-blocks, a pipelined multiplier using m-blocks is shown in Fig-3.

Since this multiplier operates at very high speed, special attention had been paid to select the clocking strategy in order to reduce the clock skew. My method

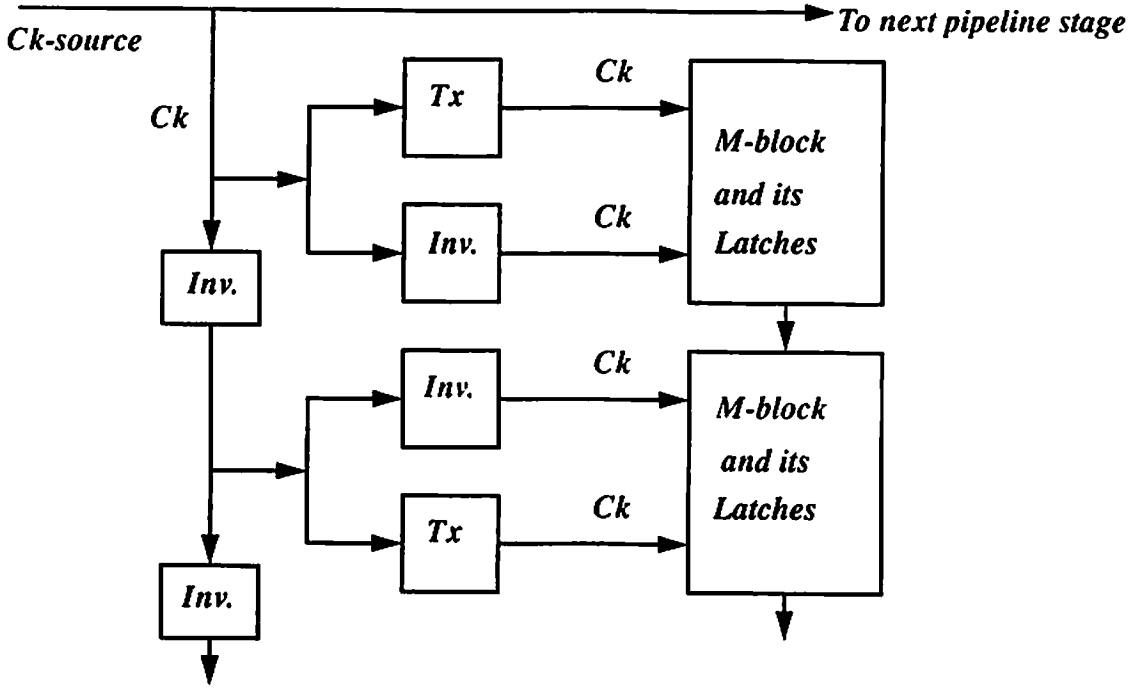


Fig-4 Clocking strategy in every pipeline stage

for achieving this is to balance the 2-phase clock by using one transmission gate and one inverter. Also, the transmission gate use the similar sized transistors as those used in inverter. This method is used for all latches in every M-block. In addition, this circuit is also used to buffer clock signals. This configuration is shown in Fig-4. 4 single phase clock inputs are used to reduce the clock distribution problem.

10. Circuit Simulation and Testing

For the convenience of testing and simulation, a test cell " *T-Cell* " is implemented in CMOS circuit with Lambda=0.25 um. This cell consists of a M-block, 8 latches,4 inverts and 1 transmission gate, and all required signal interconnections. Each component used to construct the M-block is first checked by IRSIM, then we perform CRYSTAL and SPICE simulation on it.

Circuit layouts and SPICE simulation results of these subcircuits are given in Appendix pages. We use MAGIC to extract their equivalent circuits for SPICE simulation.

Table-3 summaries SPICE-3 simulation results of these subcircuits. All SPICE parameters are default value except transistor parameter file which was released by MOSIS Service for 1.2 um CMOS on June 10, 1991. Since we implemented our multiplier in CMOS circuit for a 0.25 um technology, scaling related parameters down was the main technique used to estimate their electrical behaviors. Although it is unlikely that CMOS processing will scale in such a simple fashion, the results still give us some ideas about our designs.

Subcircuits of the M-block and their characteristics

Subcircuit Name	*Rise time/2 (ns)*	*Fall time/2 (ns)*	*Average Delay (ns)*	*Height * Width $(lambda)^2$*	*(W/L) n*
Latch	*0.05483*	*0.07050*	*0.06266*	*68*38*	*8/2*
Pi	*0.16623*	*0.04181*	*0.10402*	*53*28*	*8/2*
Gi	*0.047011*	*0.22454*	*0.13577*	*53*28*	*8/2*
C1	*0.23529*	*0.34314*	*0.28921*	*89*45*	*16/2*
C3	*0.23531*	*0.36275*	*0.29903*	*89*45*	*16/2*
Inverter	*0.07813*	*0.04947*	*0.06380*	*77*13*	*16/2*
Sum	*0.34332*	*0.57314*	*0.45823*	*113*109*	*8/2*
Tx-gate	*0.04764*	*0.06672*	*0.05718*	*66*24*	*16/2*

Table-3

Note: For all transistors, (W/L)p=2*(W/L)n.

11. Performance Evaluation

Before we decide this system's maximum clock frequency which also determines this pipeline multiplier's maximum throughput, we will see how it works first.

A common system clock causes all latches to change state synchronously at the start of a clock period of the pipeline. Each latch then receive a new data from the preceding stage, except latches in the first stage whose data come from data bus. This new data is the results computed by the preceding stage during the preceding clock period. Once a new data has been loaded, the latch is disconnected from its data source logically. Now, the pipeline processing elements can use this new input to compute a new output. Thus in each clock period, every pipeline stage transfers its old results to the next stage, and also computes new outputs.

Clearly, the clock period of this multiplier is determined by the pipeline stage which needs the longest time to compute its output. In our multiplier, each pipeline stage has the same execution time. However, we use 1 inverter to get the complement clock signal for latches from our single phase clock sources. This part will contribute 1 inverter delay to our system clock period.

Now, we will calculate the worst-case processing time in a pipeline stage. First, we want to determine the critical path. As we can see from our architecture, the carry signals which rippling through M-blocks constitute the critical path. Also from Fig-2, we know that we have to compare the sum output delay time with a inverter delay time which is required to generate the positive carry output bit in order to find the entire critical path.

This is easily done by reference Table-3. In addition, the latches which load data from preceding stage during every clock period also bring a latch delay time to our system clock period. All these factors added together determine our minimum clock period.

Therefore, the minimum system clock period should be

*Tmin=Inverter-delay+Latch-delay+Generate-delay+8*4-bit-block-delay+Sum-delay*
=0.06380+0.06266+0.13577+8(0.28921+0.29903)+0.45823*
=5.42638 (ns)

i.e., the maximum allowable input clock frequency is *184.285 MHz*.

Thus, our multiplier can generates one 32bit*32bit product every 5.426 ns if the input data array is large enough, or equivalently, the maximum bandwidth of this multiplier is about *184M (64-bit products)/sec*. A truly high speed multiplier compared with some 32bit*32bit high speed multiplies which are popular used in industry today. For example, Weitek's WTL 3164 has 15 MIPS for 32bit*32bit integer multiplication.

12 Floor Plan, Chip Layout and Pin Out Diagram

The floor plan of the M-block is shown in Fig-5

Figure 6 shows the floor plan of this high performance multiplier.

Also, a proposal pin out diagram for this chip is shown in Fig-7.

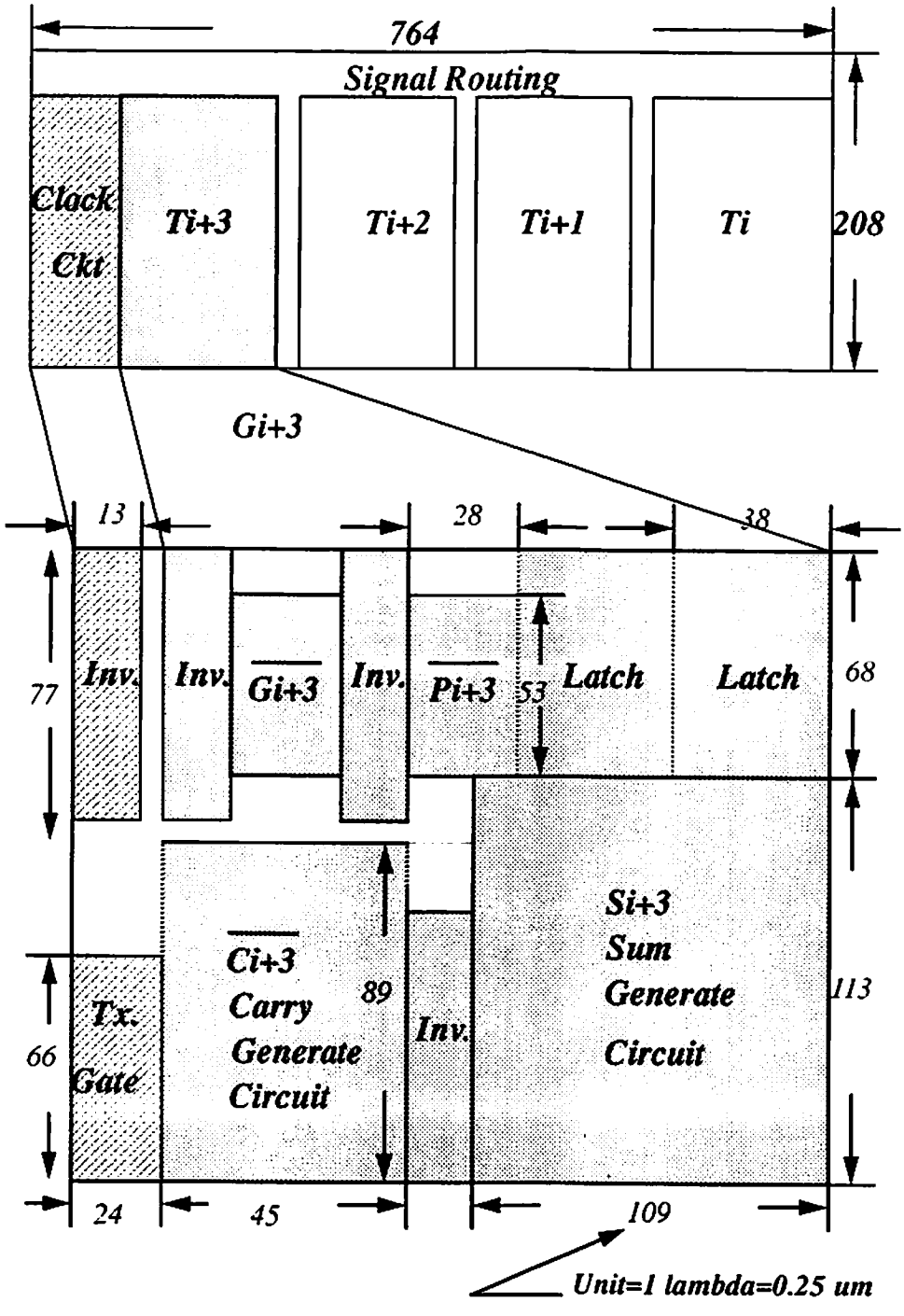
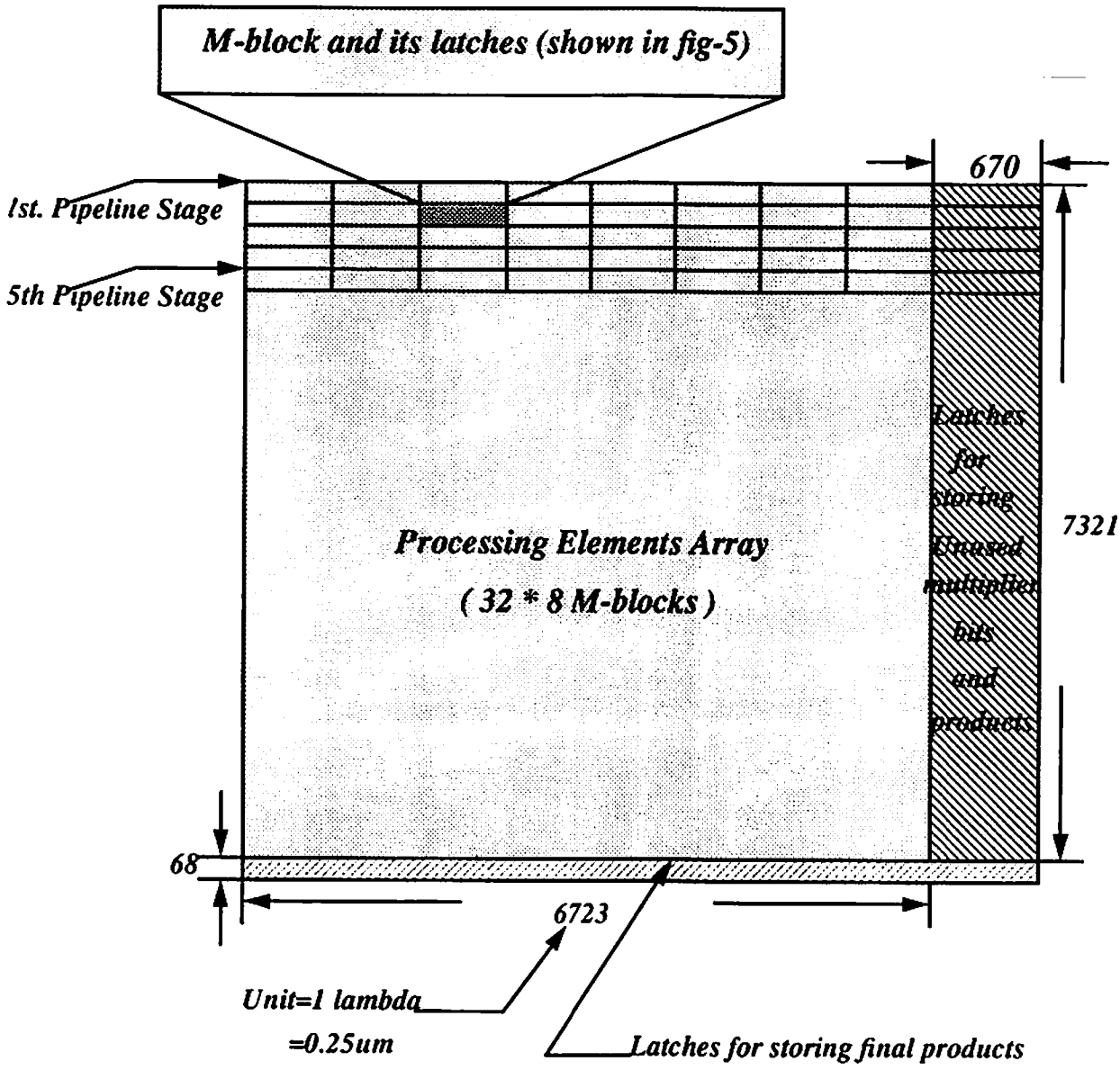


Figure 5 Floor plan of the M-block and its latches



670=16(Latches)*38(Height of latch)*1.1(Signal routing)
7321=32(Pipeline Stages)*208(Height of M-block)*1.1(Signal routing)
68=1(Latch)*68(Height of Latch)
6723=8(M-blocks)*764(Width of M-block)*1.1(Signal routing)

Chip Area=7393(Width)*7389(width)
=54626877(lambda)^2
=0.034142(Cm)^2

Fig-6 Floor plan of the 32bit*32bit pipelined multiplier

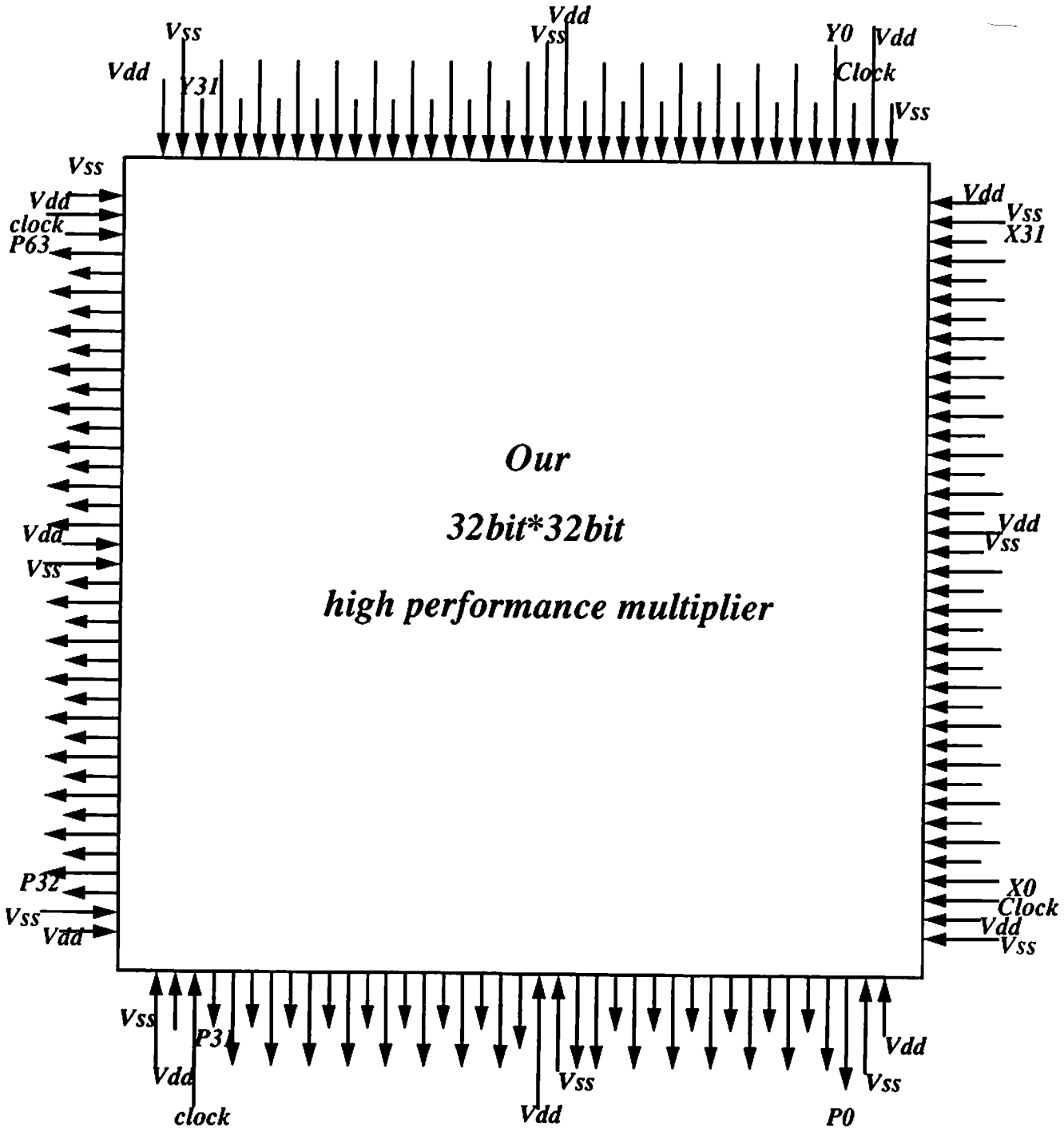


Figure 7 Pinout diagram

13. Conclusion

A high performance multiplier was constructed by using pipeline technique. Also, this multiplier was implemented in a CMOS VLSI circuit for a 0.25 um technology. The estimated maximum throughput is up to $184*10^6$ (64-bit products/sec). This means that our multiplier can produce a 32bit*32bit product every 5.426 ns if the size of input data array is large enough to fill in all pipes of this pipelined multiplier.

If a further speed improvement is desired, the carry select concept can be applied to replace the ripple carry method which is used in our circuit to transfer the output carry bit between M-blocks. However, this method will increase lots of hardware cost. Hence, a cost-effective analysis may be checked first based on our simple cost-speed product design criterion in order to choose the most efficient architecture.

References

1. *J.P. Hayes, Computer architecture and Organization, 2d. ed., McGraw-Hill, New York, N.Y.,1988.*
2. *K. Hwang and F.A. Briggs, Computer architecture and Parallel Processing, McGraw-Hill, New York, N.Y., 1984.*
3. *D.A. Patterson and J.L. Hennessy, Computer architecture: a quantitative approach, Morgan Kaufmann pub., Palo Alto, C.A., 1990.*
4. *N. Weste and K. Eshraghian, Principles of CMOS VLSI Design, a system perspective, Addison-Wesley, Reading, M.A., 1984.*
5. *P.J. Song and G.D. Micheli, "Circuit and architecture trade-offs high speed multiplication",IEEE Journal of Solid State circuits, Vol.26, No.9, 1991.*
6. *R.P. Brent and H.T. Kung, "A regular layout for parallel adders", IEEE Trans. on Computers, C-31, P260-264, 1982.*
7. *Texas Instruments, LSI Logic Data Book-MOS/CMOS, Texas Instruments, Dallas, T.X., 1986.*
8. *D. Lancaster, TTL Cookbook, Howard W.Sams & Co., Indiana, I.N.,1984.*

TERM PROJECT

EE 577

TITLE: Floating Point Multiplier
System Design

For: Dr. Bing Sheu

By: Philip Crary
Hughes Aircraft Company
ESN

Due: 12/11/91

Table of Contents

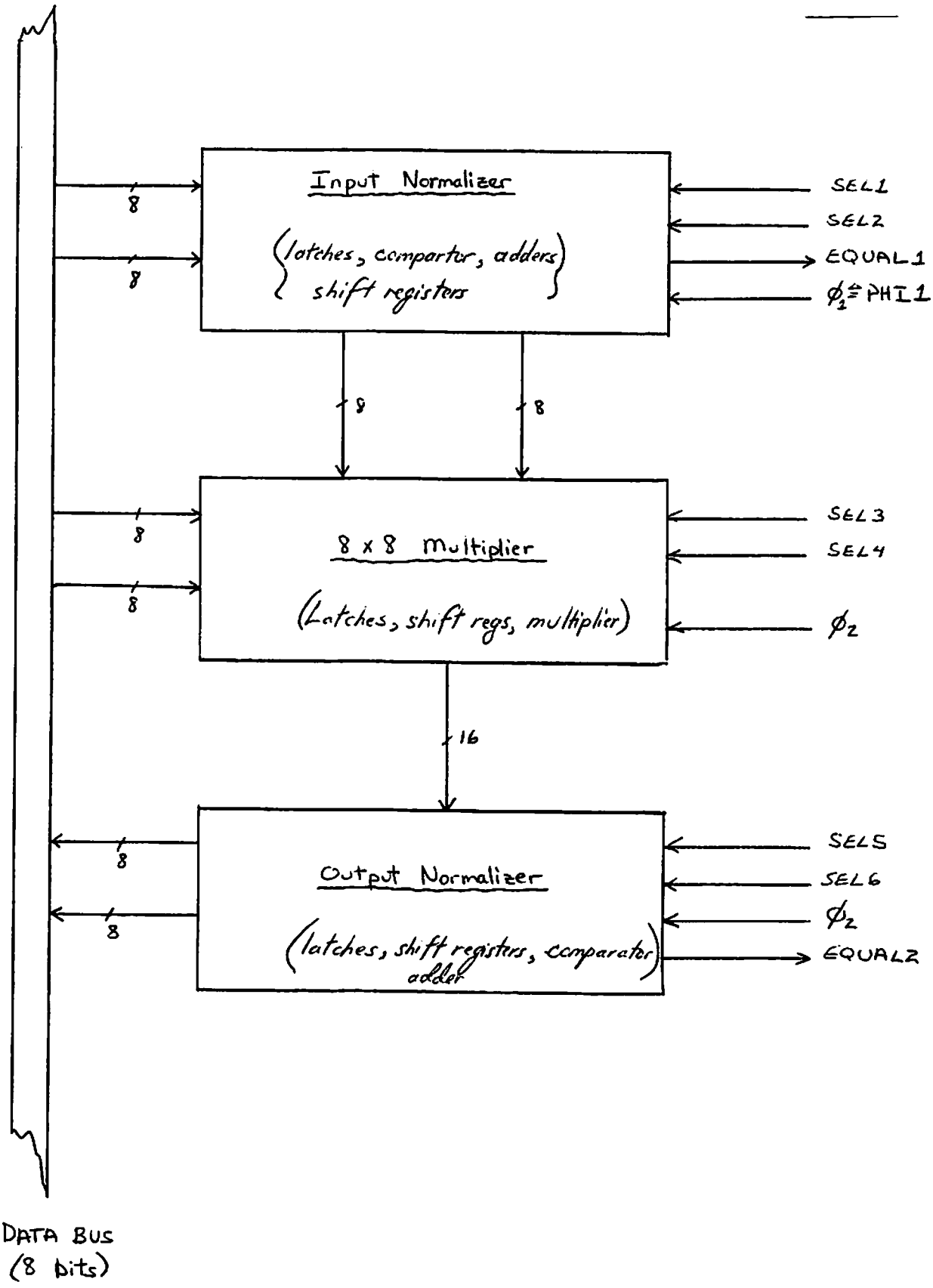
I. ABSTRACT OF THE PROJECT

For this project I chose to design an 8 bit by 8 bit floating point multiplier (i.e. 8 bit exponents and mantissas). This design was chosen because it requires a "system level" perspective in addition to the component level perspective stressed in the class assignments. The multiplier functions with any two "8 bit", floating point, 2's complement numbers. These numbers are assumed to not be normalized to each other. The output will be a normalized "8 bit" number; there is no check for overflow or underflow.

For this design I created some components, and used some from the homeworks. Then I found the maximum clock rate, and the maximum multiplication operation rate, of the circuit. I identified the frequency limitting components, and comment on the best ways to speed up the circuit.

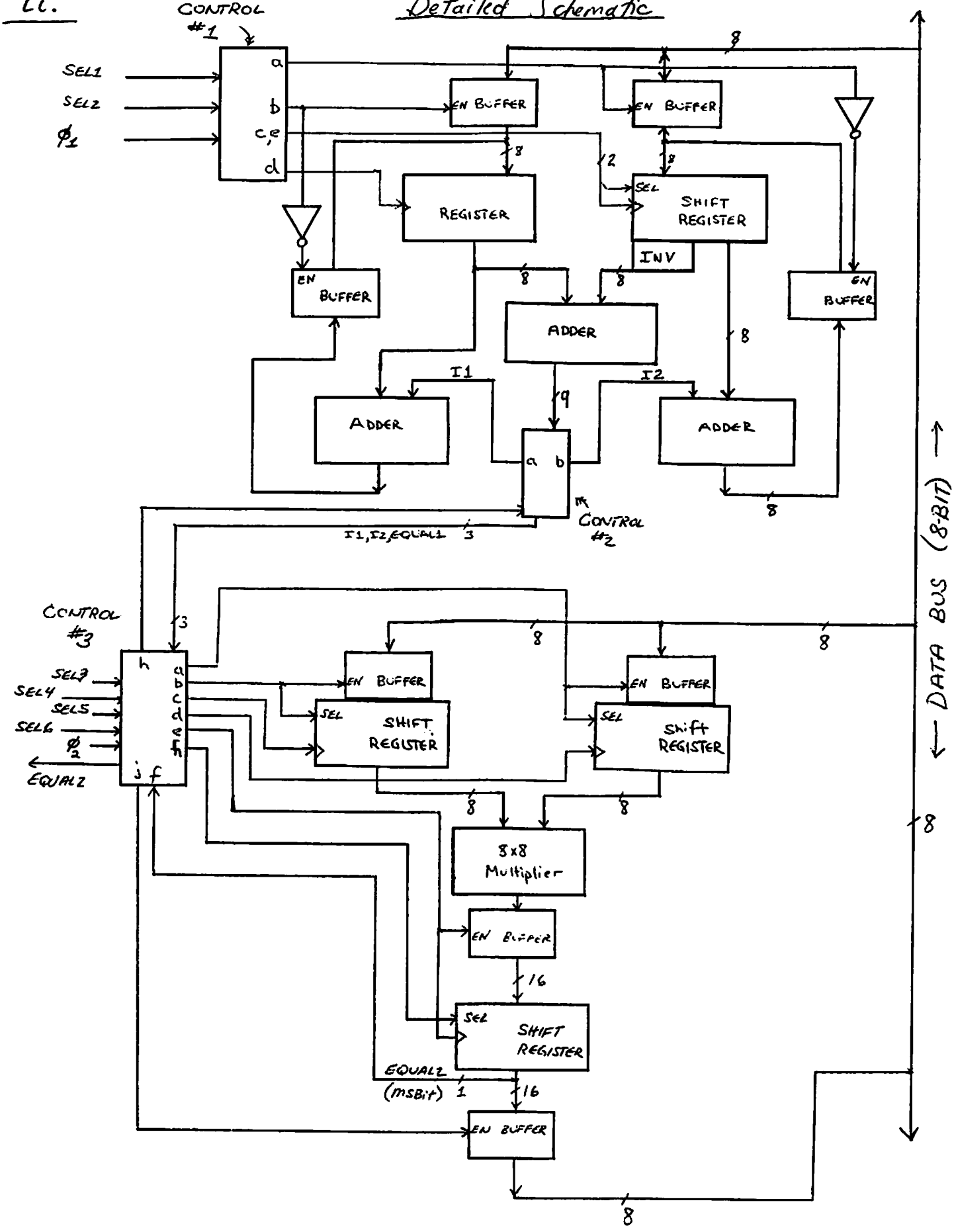
I used 0.6 um, 2 metal, CMOS technology, calculated the final silicon area, and performed a system level timing analysis based on the component timing analysis.

II. i. BLOCK DIAGRAM



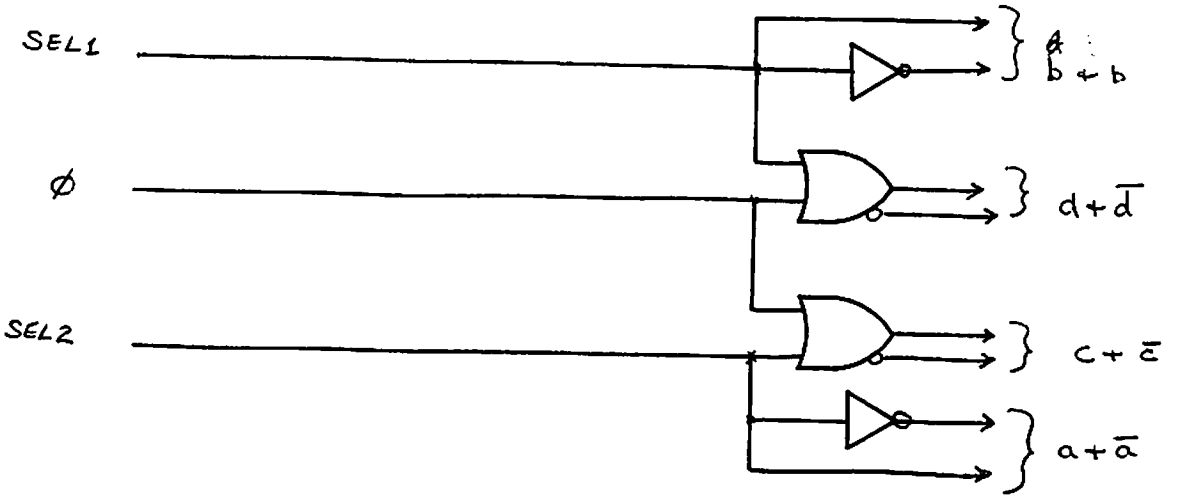
ii.

Detailed Schematic

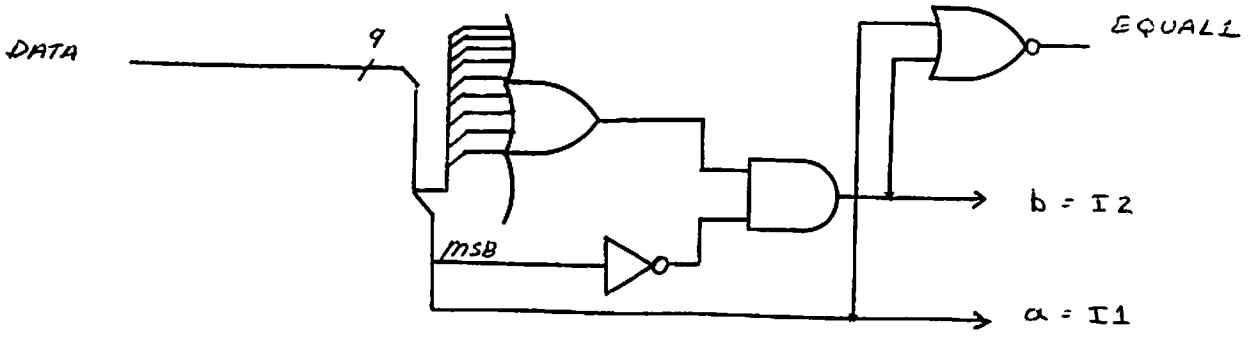


DETAILED BLOCK DIAGRAM

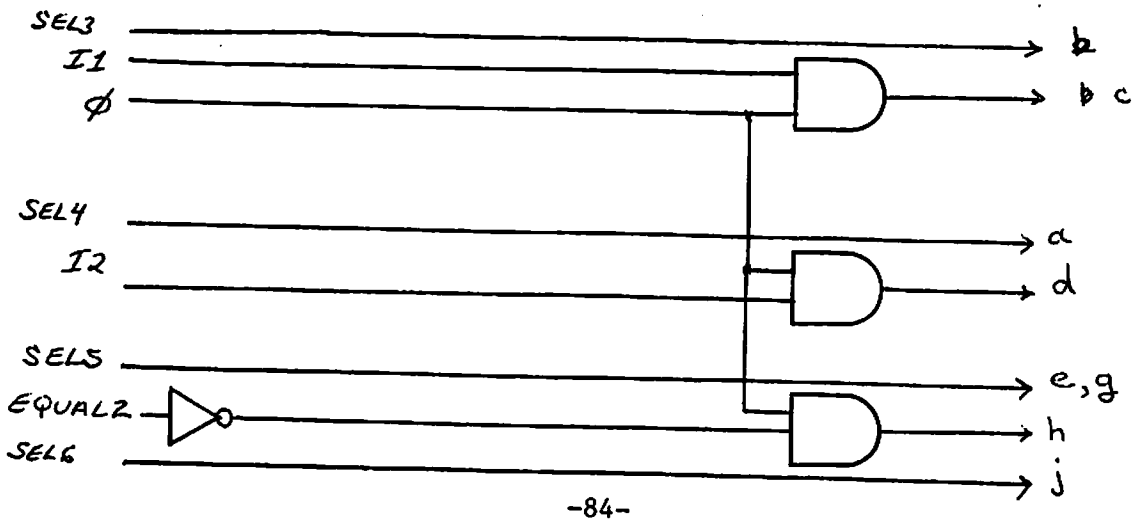
CONTROL BLOCK #1 — Detailed Schematic (cont'd)



CONTROL BLOCK #2



CONTROL BLOCK #3



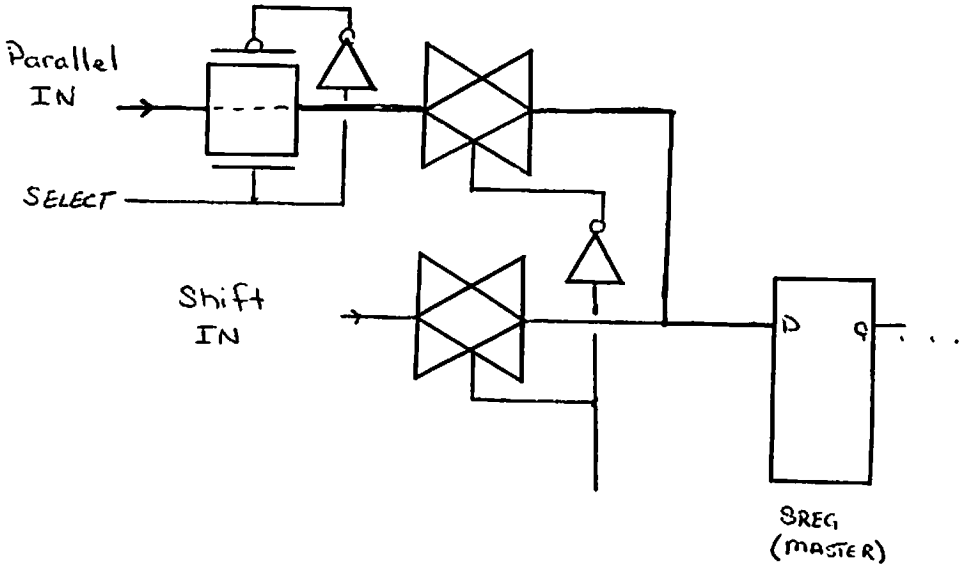
All need complement outputs

iii. TIMING DIAGRAM AND ANALYSIS

For this design I used a two phase, non-overlapping clocking scheme. I saw no advantage in going to other types of clocking techniques since I was mostly interested in a system analysis perspective.

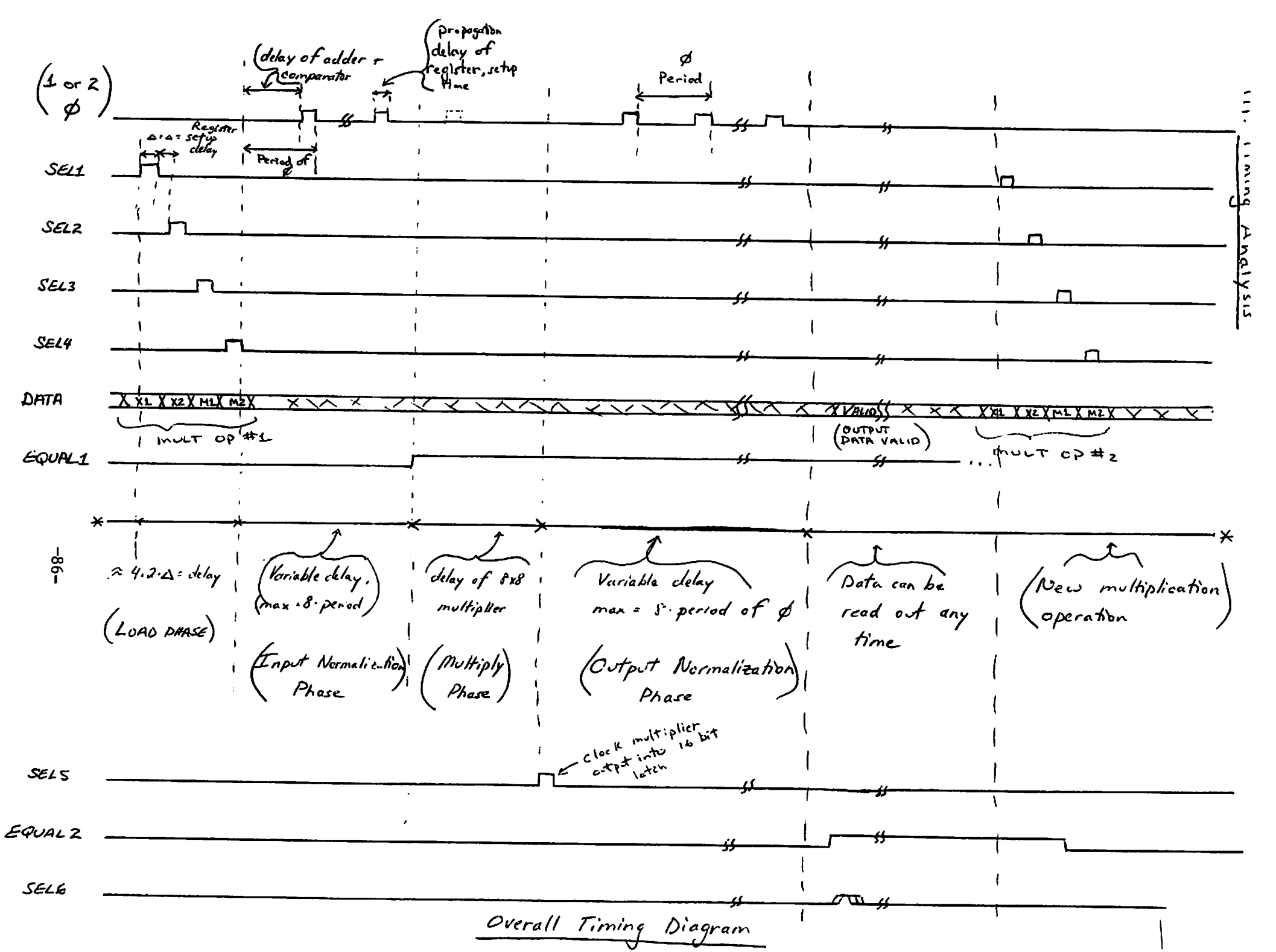
The high phase of PHI, the clock, is used to pass the data, which is already stable, from the generalized component inputs, to the slave cell inputs. Since I use the high phase of the same clock to load or shift throughout the chip, I needed to make this phase long enough to allow the data to pass through the buffers, the multiplexer, and then load the master cell of the parallel-load shift register (this is the longest delay).

The PHI clock must then be high longer than the delay of the following circuit:



——=critical path

or $T_{\phi HI} = T_{BUFF} + T_{SREG-MASTER} + T_{MUX} = 0.55 + 2.93 + 0.55$
$= 4.03$ nS.



Overall Timing Diagram

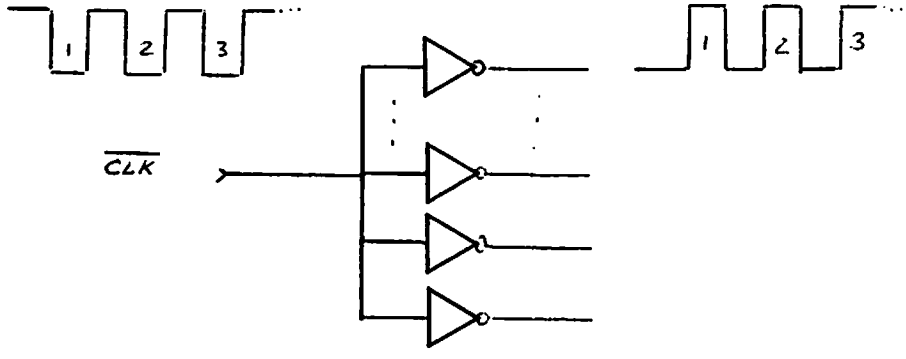
Since the low portion of the clock is used to load the slave cell, among other things, it needed to be at least the delay of the shift register slave cell long. After the loading of the shift register master cell is done, all control signals may change. If they change, they must again be stable on the rising edge of the clock. These control signals are driving at most two gates, of modest size. Because of this, the delay due to changing control signals is minimal and will not be mentioned here. So:

$$T_{Low} = T_{P\text{-}Sreg\text{-}Slave} + T_{Buff} = 2.93 + 0.55 = 3.43 \text{ nS.}$$

So, with $T_{\phi High} = 4.03$ nS and $T_{\phi Low} = 3.48$ nS, to the clock symmetric I made the duty cycle 50% with high pulse of 4.03 nS. A clock frequency of 1.23 MHz.

To enable the clock signal to drive so many gates, I used a parallel buffering scheme:

<u>Clock Source</u> <u>Buffering</u> <u>On Chip Signal</u>



Next, the maximum delay of the multiplication operation must be calculated. The following formula was extracted from the schematic, and the data used was produced by the CRYSTAL and SPICE simulations performed on the components.

$T_{MULT\text{-}OP}$ = load + input normalization + multiplication + output normalization + bus access time

$= 4^* T_{\phi\text{-}CLK} + 2^* 8^* T_{ADDER} + T_{\phi'\text{-}CLK} + T_{MULT} + T_{\phi\text{-}CLK}{}^*8 + 8^* T_{ADDER} + 8^* T_{\phi\text{-}CLK} + T_{\phi'\text{-}CLK}$

$= 21^* T_{\phi CLK} + 32^* T_{\phi\text{-}CLK} + 5^* T_{\phi\text{-}CLK} + T_{\phi MULT} + 16^* T_{\phi\text{-}CLK}$

= 602 nS/ multiplication operation.

(where $T_{\phi\text{-}CLK}$ is the period of one clock of PHI)

which is equivalent to 75 cylces of PHI clock per multiplication operation.

vi. OPERATION DESCRIPTION

The circuit, "chip", operates with 8 control inputs, 2 control outputs and an 8 bit data bus. Any two normalized floating point numbers, within the "8 bit" range can be multiplied. The numbers need not be normalized against each other prior to multiplication. There are no checks for overflow or underflow performed.

Initially, the two mantissas are loaded into two parallel-load shift registers. The #1 exponent is loaded into a parallel-load register, while the #2 exponent is loaded into a parallel-load, shift register. The two exponents are compared producing a signal called "EQUAL1". This signal, a chip output, indicating whether the input normalization phase is done, is added in with the exponent values in a way which increments the lower exponent toward the higher. EQUAL1 enables the correct mantissa to be shifted left, in conjunction with the incrementation of it's corresponding exponent. When EQUAL1 finally goes low, (exponents are equal) several things happen: the #2 exponent is shifted left to account for the doubling of the exponent during multiplication, the PHI clock to the input normalization circuitry stops, and the multiplication phase begins.

After a fixed delay for the 8 bit mantissa multiplication to be completed, the multiplication product is latched in a 16 bit parallel-load shift register. The msb of this register is checked for unity; if not the #2 exponent is decremented, and the 16 bit mantissa is shifted left. This continues until the product is normalized. The eight most significant bits of the product are passed through a buffer to the bus upon access by the controller.

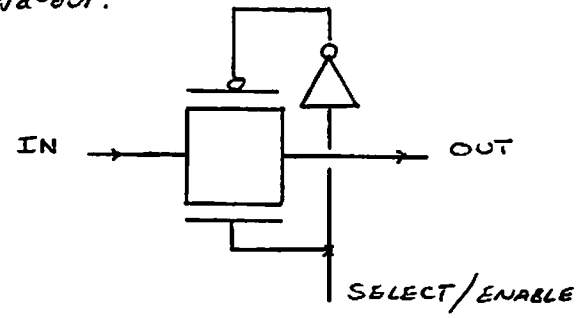
The external controller of the chip is minimally complex; a counter and some logic. All exact timing is displayed in either the timing diagram or data statistic sheets.

II. v. COMPONENT ANALYSIS

Component Data Sheet
($\lambda = 0.6\mu m$, CMOS)

BUFFER

- 8 bit, bidirectional, uses transmission gate
- size of silicon = $68\lambda \times 33\lambda = 2448\lambda^2$
- Transistor size = $2(L) \times 4(w)$ for both n-type and p-type.
- $\tau_p = 0.55ns$
  = max delay of cell = τ_{PLH}
  $\triangleq$ data-in to data-out.



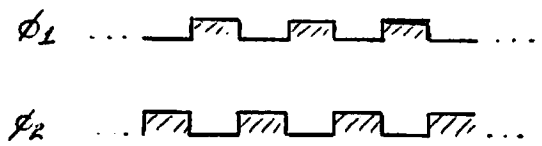
- See the layout for this part.

Description: It's a buffer; when "enable" is high, the OUT follows the IN (and Vice-Versa), when enable is low the OUT is isolated from the IN.

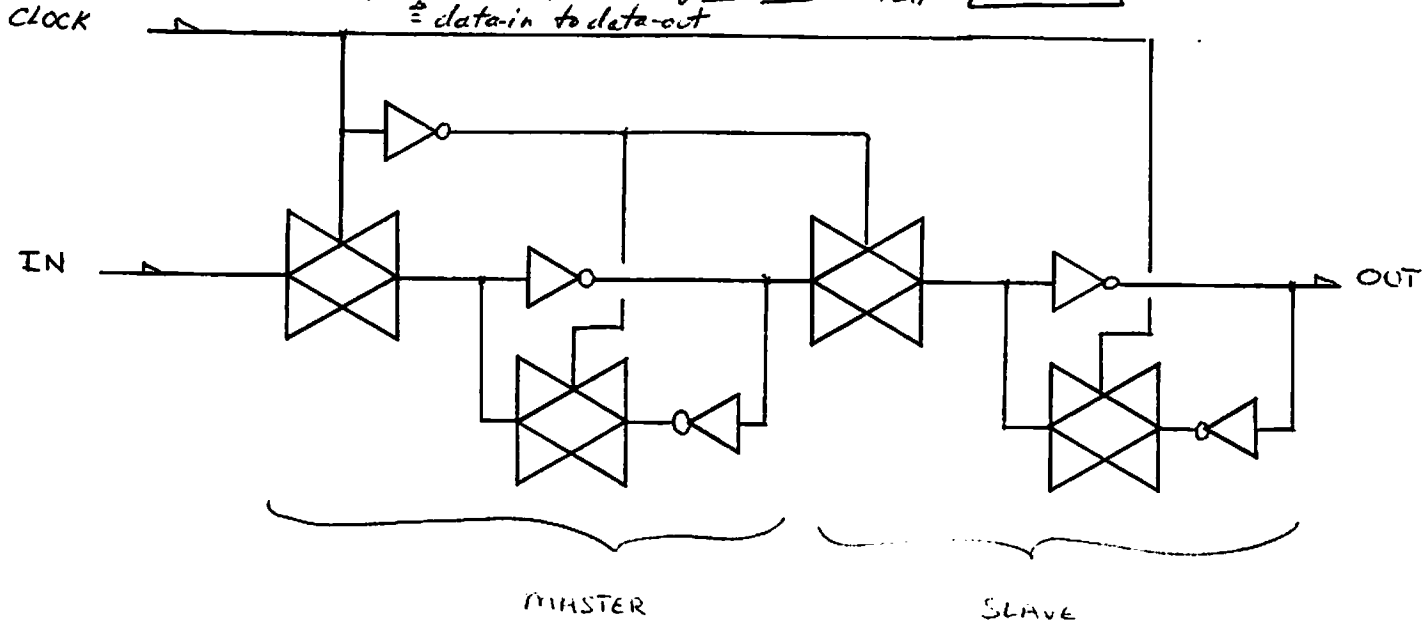
Component Data Sheet
($\lambda = 0.6\mu m$, CMOS)

REGISTER

- 8 BIT, from SREG.MAG cell used in H/W
- Size of Silicon = 998(w) x 91(h) = 90, 818 λ^2
- Transistor size - variable (see MAGIC Layout)
- Inverting and Non-inverting outputs
- Two phase clocking:



- τ_p = max delay of loading one cell = data-in to data-out $\tau_{PLH} = \boxed{2.93\,ns}$

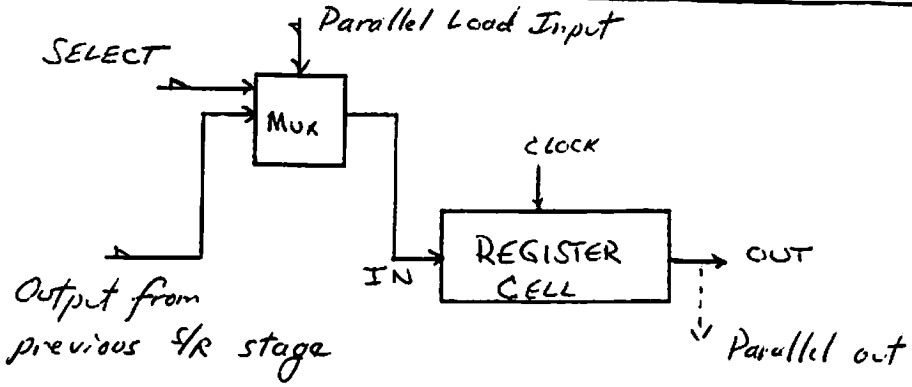


Description: The 2 phase clock loads the Master cell on the high portion of it's period; the low ~~phase~~ portion of the period disabled the feedback in the Master cell. The high portion also disables the loading of the slave cell. Once low, the clock loads the slave cell, with the now latched contents of the master cell. So one clock period will load this cell.

Component Data Sheet
($\lambda = 0.6\mu m$, CMOS)

Shift Register:

- 8 BIT or 16 BIT
- SIZE of SILICON =
  - 8 BIT = 1108(w) x 160(h) = 100,828 λ^2
  - 16 BIT = 2216(w) x 160(h) = 201,656 λ^2
- Transistor size = Variable see IMAGIC Layout
- Two phase clocking, Master/Slave arrangement
- Based on REGISTER CELL.
- $\hat{=}$ datain to data-out.
- T_p = max delay of loading cell = τ_{PLH} = 2.93+0.55 = 3.48ns



Functional Drawing

Description: Using the REGISTER cell from the previous page, I added a selectable input using a mux. Otherwise, this cell operates like the register cell.

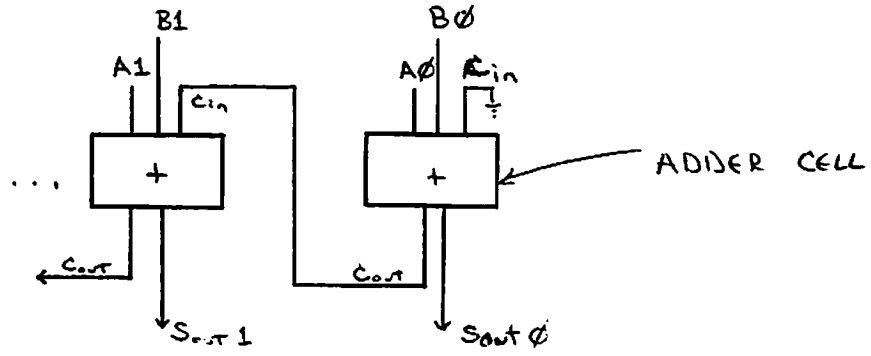
Component Data Sheet
($\lambda = 0.8\,\mu m$, CMOS)

ADDER: – 8 BIT + 8 BIT, based on "ADDER.MAG" from 4/w.
– size in Silicon (200(h) × 200(w) × 80) = 322,560 λ^2
– Timing Analysis:
(See Appendix for Print Outs of data)

	SPICE3	CRYSTAL	
τ_r = cin to c-out rising edge	1.9ns	2.42 ns	(21% off SPICE3)
τ_f = Cin to c-out falling edge	2.1ns	2.32 ns	(10% off SPICE3)

$\tau_p = \frac{\tau_r + \tau_f}{2} = 2.0$ ns (SPICE3)

τ_p = overall propagation delay = 2.0 × (8 cells) = 16. ns



Functional Drawing

Description: Each cell is provided an input bit from each of the 8bit words. Carry has NO Look-AHEAD circuitry.

Component Data Sheet
(λ = 0.6μm, CMOS)

Multiplier: – 8 Bit x 8 Bit, based on "MULL.MAG" and ADDER.MAG from H/W.

– Size in Silicon = (210(h) x 210(w)) x (9 rows) x (8 columns)
= 3,160,080 λ^2

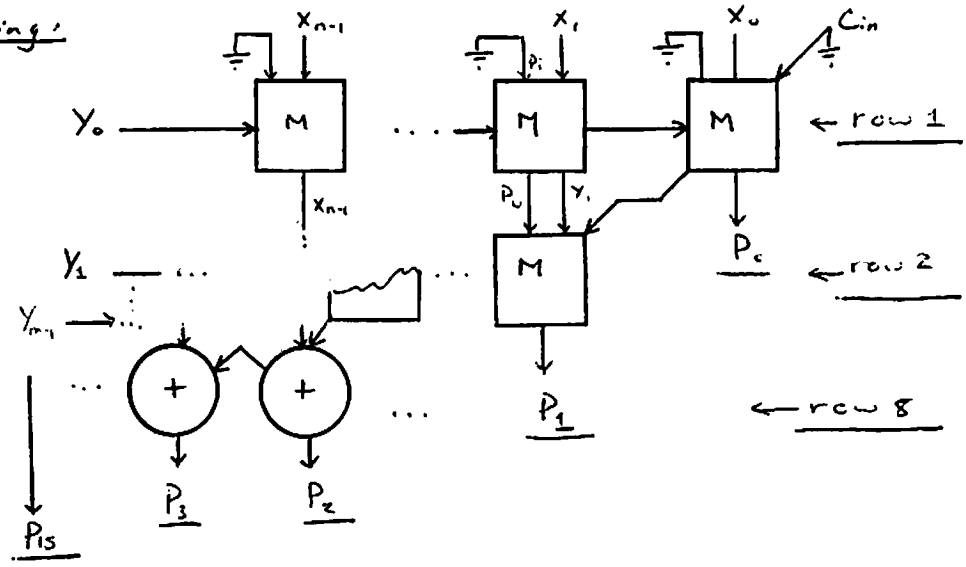
– Timing analysis: (See Appendix for printouts of data)

	SPICE	CRYSTAL
τ_{pxc} = propagation delay from x-input to C-out	7.5ns	7.93ns
τ_{ccm} = p-delay C-in to Cout of MUL.MAG	1.9ns	4.15ns (85% off SPICE)
τ_{cca} = p-delay C-in to C-out of adder cell	1.9ns	1.9ns

τ_p = max delay = $\tau_{pxc} + (\tau_{ccm} \cdot 7) + (\tau_{cca} \cdot 7)$

τ_p = max delay = for SPICE = 34.1ns
CRYSTAL = 50.28 ns

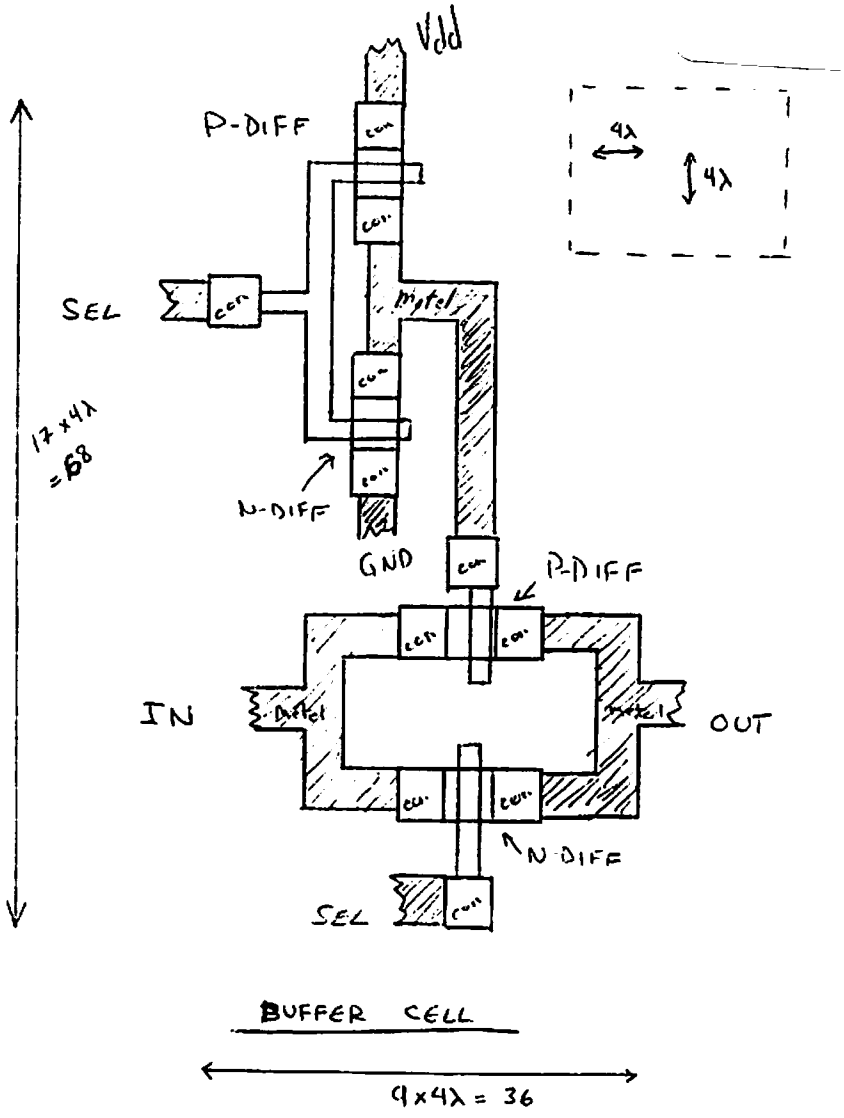
Functional Drawing:



The adders are used to complete the carry summation.

NOTE: Though this multiplier is by no means the fastest, it is the goal of this project to understand this circuit, the process to describe and test it, and show its functionality.

Buffer Cell



* Because the diffusion regions are symmetric, n-diff to p-diff, the rise and fall time will not be symmetric.

Because I am a remote student, I was unable to simulate this. I have had a lot of trouble going to campus + getting a color CAD station. I did the following analysis by hand and it should be fairly accurate.

Analysis

I used the NAND cell from H/W #1 as a basic gate delay. The τ_N = NAND delay = 0.55 ns for a gate area of:

$$\tau_N \propto (2\times3) + (2\times4) = 14\lambda^2$$

So, Since I have two paths, I took the ratios of gate areas and figured the respective times for the "select", and the "$\overline{\text{select}}$" signals

$$\tau_S \propto (2\times4)\lambda^2 \quad , \quad \tau_{\bar{S}} \propto (6\times4)+(2\times4)+(2\times4) = 24\lambda^2$$

So the average delay of these is

$$\tau_{BUFFER} \propto \frac{24+8}{2} = 16\lambda^2$$

Since the gate load on "select" is $(2\times4)+(2\times4)+(2\times4)$ also, I used

$$\tau_{BUFFER} = 0.55\text{ns}.$$

vi. ANALYSIS OF SILICON AREA

Control Block 1 - negligable size
2 - negligable size
3 - negligable size

#1 == relates the component to the first of the numbers to be multiplied.
#2 == relates the component to the second of the numbers to be multiplied.

BUFFERS:

EXPONENT #1 from BUS - 2448/cell* 8	=	19, 832
" " #2 from BUS - " "	=	19, 832
" " #1 from adder - " "	=	19, 832
" " #2 from adder - " "	=	19, 832
MANTISSA #1 from BUS - " "	=	19, 832
" " #2 from BUS - " "	=	19, 832
MULTIPLIER Output - 2x(" ")	=	39, 664
16 bit S/R Output - " "	=	19, 832

SHIFT REGISTERS:

EXPONENT #2 - 1108(w) * 160(h)	=	177, 280
MANTISSA #1 - " " " "	=	177, 280
MANTISSA #2 - " " " "	=	177, 280
PRODUCT (16 bit) - 2 * 1108 * 91	=	354, 560

REGISTER:

EXPONENT #1 - 1108 * 91	=	100, 828

ADDERS:

3 exactly the same - 3 * 322,560 (40,320/cell * 8 * 3)	=	967, 680

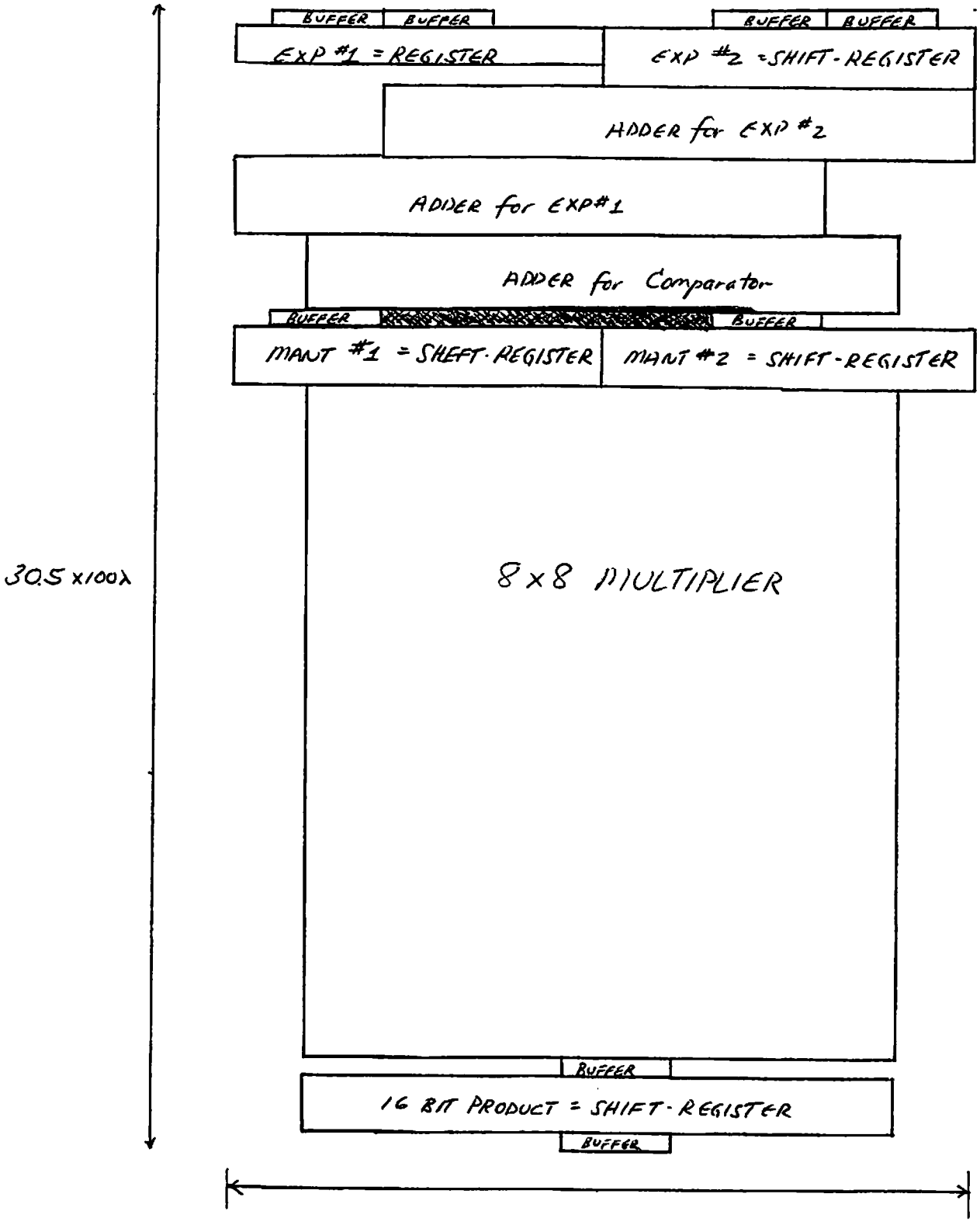
MULTIPLIER:

Only one - = 3, 160, 080
(43,890/cell * 8 columns * 8 rows)

TOTAL SILICON AREA = 5,293,476 λ^2

= 1.9 mm^2

= 6.1 mm^2 (from layout)



LAYOUT OF COMPONENTS
IN SILICON

BLOCKS ↕ ↔
=100λ/side

vii. NUMERICAL EXAMPLE

Number #1 = 0.101 e= -1
Number #2 = 0.001 e= +3

PHASE	EXP#1	EXP#2	MANT#1	MANT#2	PRODUCT
LOAD	-1	3	1010/0000	0010/0000	0/0/0/0
INPUT NORMALIZATION	0	3	0101/0000	0010/0000	-
	1	3	0010/1000	0010/0000	-
	2	3	0001/0100	0010/0000	-
	3	3	0000/1010	0010/0000	-
MULTIPLICATION	3	6	-	-	0/0001/0100/0
OUTPUT NORMALIZATION	3	5	-	-	0/0010/1000/0
	3	4	-	-	0/0101/0/0
	3	3	-	-	0/1010/0/0
	3	2	-	-	0001/0100/0/0
	3	1	-	-	0010/1000/0/0
	3	0	-	-	0101/0/0/0
	3	-1	-	-	1010/0/0/0

OUTPUT = 0.1010 e= -1 which is correct.

III. RESULTS AND BIBLIOGRAPHY

The chip I designed was by no means optimal in speed. I used no carry look-ahead on the adders and no reduction of scale to 0.25 um (with Tox reduced to 1.1nm), to name only the two most productive techniques to speed up the chip. I was mostly trying to see what was needed to design a working floating point multiplier. Yet the numbers show a 1.64 MFLM (M FLoating point Mult) rate is possible. Given the reduction rate (derived from homeworks) of 0.2 when going from 1.2 um technology to 0.5 um technology (with reduced Tox), a 1.65 * 5 MFLM, or 8.25 MFLM rate is easily possible.

I was unable to run IRSIM on this circuit as a whole. I had initially hoped to, but the memory and CPU time required by this circuit, and it's associcated files, was too great. I wasn't allowed to save files in MAGIC due to "QUOTA" problems, and I hadn't even drawn the whole thing. Finally the CPU was so very slow doing even small components, that I gave up the idea of a full IRSIM or SPICE3 simulation. Instead I worked the system timing out by hand and estimated the buffer performance by comparison to homeworks.

The only book used during the compilation of this project was the text used for class.

125MHz DSP RISC PE DESIGN

PROFESSOR: Dr. Bing Sheu

CHENG-JU HSIEH 606-38-4590 *ADDER*
CHIH-WEI TSAI 615-38-6758 *MULTIPLIER*
SHEN-TE HONG 888-04-0997 *SRAM*
WEI-LI WANG 888-05-7693 *I/O, REG., MUX.*

Electrical Engineering Department
University of Southern California

Dec. 1991

EE 577 TERM PROJECT

A 2.63ns CMOS 32 + 32-b Carry-Select Adder mixed with Manchester-Carry Adders

PROFESSOR: Dr. Bing Sheu
NAME: Cheng-Ju Hsieh
SS#: 606-38-4590
Electrical Engineering Department
University of Southern California

Dec. 1991

Abstract - A 2.63ns, based on 0.5um technology, CMOS 32 + 32-bit carry-select adder mixed with Manchester-carry adder is described. The adder cells using for carry-select adder are Manchester adders with variable sizes. 4- and 5-bit Manchester adders are used in this project. By comparing with the conventional carry -select adder, my adder design is more than twice faster than the conventional carry-select adder that uses ripple-carry adder cells. The Manchester lookahead circuitry has been applied to every block in order to reduce the delay time.

CONTENTS

I. INTRODUCTION

Adders form important components in many systems. Probably the simplest approach to designing an adder is to implement gates to yield the required majority logic function. The carry-lookahead adders improve the performance of the basic carry-ripple adder by making the slow signals arrive earlier. The carry-skip adders improve the performance of the basic carry-ripple adder by making the early signals more available, trading the available time against resources. In the carry-select adder, the early signals are duplicated, at the expense of additional resources, to reduce the number of levels in the adder. In my project, I try to combine carry-select adders with carry-skip adders in order to obtain the faster response time and focus on the analysis and design of CMOS Manchester carry-skip adders (Fig. 1) with variable size blocks [1], [2].

II. DEFINITIONS

I use some different cells to implement this adder in my project. Here are the definitions of those cells.

Definition 1: A 4- or 5-bit Manchester adder that the initial carry-in is 0 will be represented by ***mc40*** and ***mc50*** respectively (Fig 2a, 2b).

Definition 2: A 4- or 5-bit Manchester adder that the initial carry-in is 1 will be represented by ***mc41*** and ***mc51*** respectively (Fig 2c, 2d).

Definition 3: A two-input OR gate which is connected to a two-input NAND gate will be represented by ***ONA*** (Fig 3a).

Definition 4: A two-input AND gate which is connected to a two-input NOR gate will be represented by ***ANO*** (Fig. 3b)

Definition 5: A 8- or 10-input multiplexer will be represented by ***mx4*** and ***mx5*** respectively. Fig. 4 shows a two-input multiplexer.

III. CONCEPT AND STRUCTURE

Originally, the carry-select adder divides a ripple-carry adder into blocks, and for each

block two additions are performed in parallel. The first is performed assuming the carry-in to the block is 0; the second, assuming the carry-in to the block is 1. A set of multiplexers driven by the actual carry-in gives the final block sum bits and the block carry-out bit [3]. In this project, I design this adder by myself which is basically a carry-select adder mixed with Manchester adder in order to obtain the better response time and smaller area. I design a 32-bit adder composed of 7 blocks with variable sizes. My design method involves:

1) analyzing the timing of an n-bit adder block,

2) locating the critical path of the x-bit adder, and

3) determining the block sizes to reduce the delay of the critical path.

The whole structure of my design is shown in Fig. 5 and Fig. 6. The only difference of these two designs is the block size. The block sizes of Fig. 5 are 4 4 5 5 5 5 4 and of Fig. 6 are 4 4 4 5 5 5 5. All the Manchester adder cells are operating concurrently, so we can speed up the response time by making the block size bigger for the blocks of the more significant bit. The critical path is from the carry chain of the first block (*mc4x*) through the *ONA* or *ANO* cells in the following block except the last one and up to the last sum bit in last block. Manchester lookahead circuitry (Fig. 1d), which a special circuit associated with each block detects quickly if all the bits to be added are different (p=1 in all the blocks), has been applied. In this case the carry entering into this block may directly bypass it and so be transmitted to the next block.

IV. RESULT AND COMPARISION

All results in this project are based on the simulations of the longest latency for each block as shown in Fig. 7. Furthermore, I calculate the capacitance loading for *mx4*, *mx5*, *ANO* and *ONA* (Fig. 8) in order to do the SPICE simulation. The capacitance loadings of each carry out are shown in Table 1. After SPICE simulation (see Appendix II), the delays of each cell are shown in Table 2b and the delay of 4- and 5-bit modified ripple adders (from homework set #3d) are shown in Table 2a for comparison. The delay is measured by 50% - 50% criterion. From these two tables, we can see that the Manchester adders are more than twice faster than the modified ripple adders.

The Manchester adders seem to be a good compromise between ripple-carry adders (very simple, but slow) and sophisticated adders (like carry-lookahead adders, for instance), which involve both large silicon areas and design problems. This is the reason why I choose Manchester adders. Then, I do the timing analysis for conventional carry-select adder using modified ripple adders (Fig. 9a) and my design (Fig. 9b, Fig 9c). As I mentioned before, the delay will be affected by the different arrangement of variable size blocks. Obviously, Fig 9c is a better design compared with Fig. 9b and the delay time is just 2.63ns for 32-bit adder. Table 3 shows the area of different cells and blocks that I used in my design. The data in this table are from the actual layout (see Appendix III). My design is much smaller than the carry-select adder using ripple adders.

V. DISSCUSION AND CONCLUSION

In my own design, I present a way to obtain efficient and fast adders, built with blocks of different sizes and based on carry-select adder and Manchester adder. By changing the ripple adder to Manchester adder and adjusting the block size, we can obtain the better response time as shown in this report. Generally, my design is more than twice faster than the classical carry-select adder. The ripple-carry propagation delay is linearly proportional to the size of a block. The Manchester adder using dynamic (percharge) logic, where the carry propagation delay of a block is proportional to the square of its size [1].

Actually, The response time of my design can still be improved because it depends on the response time of certain cells such as "*mc4.x*", "*mc5.x*", "*ANO*", and "*ONA*". One way, which has been implemented by my teammate Chih-Wei Tasi, to improve it is to use transmission gates to replacing some logic cells that I just mention above or rearrange the block size to 4 4 4 4 5 5 6. The advantage of my design is that the delay time will just slightly increase when we add a big number of input bits. Thus, this adder seems to be of greatest use for adders that are larger than 16 bits and needs fast response time. The disadvantage is that some areas are wasted because of the duplication in carry-select adder. The final result compared to the conventional carry-select adder is shown in Table 4.

VI. REFERENCES

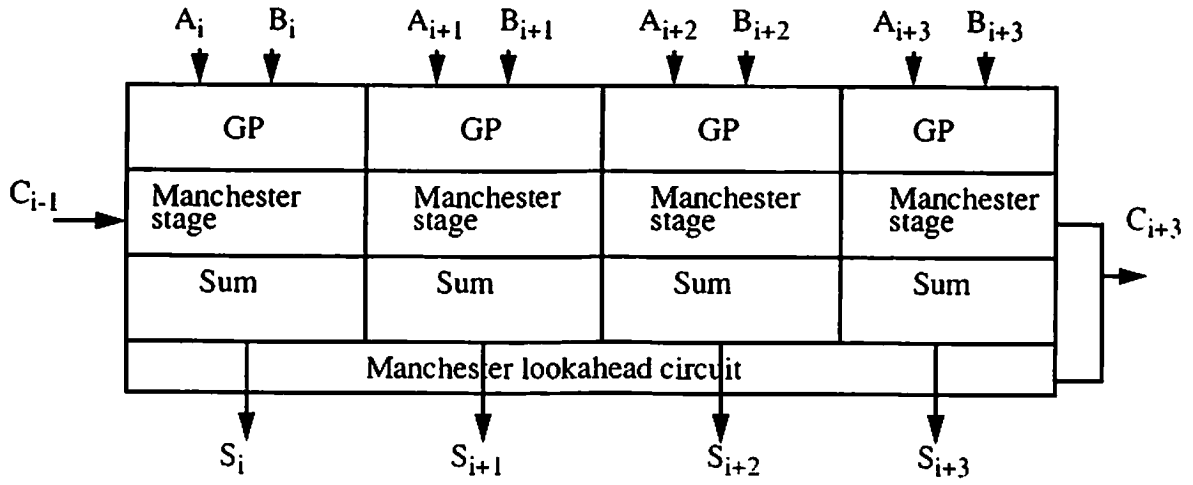
[1] P. K. Chan and M. D. F. Schlag, "Analysis and design of CMOS Manchester Adders wth variable carry-skip"*IEEE Trans. Comput.*, vol. 39, no. 8, pp. 983-991, Aug. 1990.

[2] A. Guyot, B. Hochet, and J. M. Muller, "A way to build efficient carry-skip adders", *IEEE Trans. Comput.*, vol. C-36, no. 10, pp. 1144-1151, Oct. 1987.

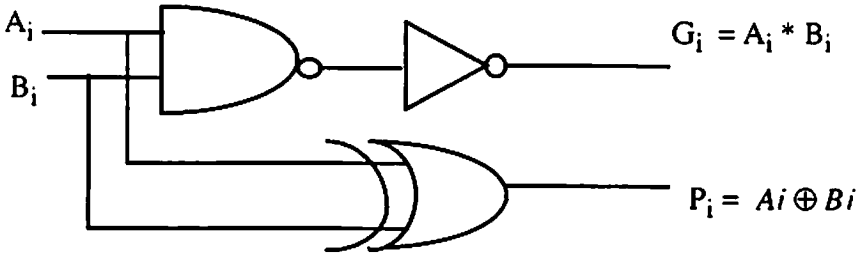
[3] N. Weste and K. Eshraghian, *Priciples of CMOS Design: A Systems Perspective*. Reading, MA: Addision-Wesley, 1985, pp. 322-333.

APPENDIX I

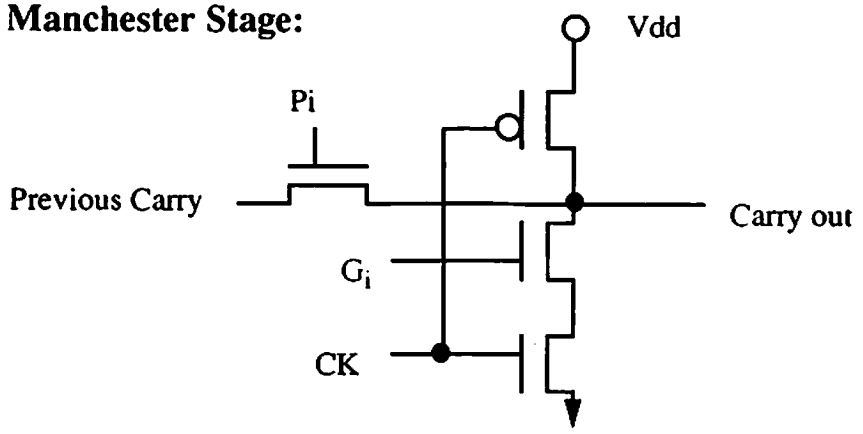
4-bit Manchester cell (mc4x):



a) G & P Generator:



b) Manchester Stage:



c) Sum Generator:

Fig.1 Manchester-Carry Adder

d) Manchester Lookahead Circuit:

If $P_i = P_{i+1} = P_{i+2} = P_{i+3} = 1$, then $\overline{Ci-1} = \overline{Ci+3}$

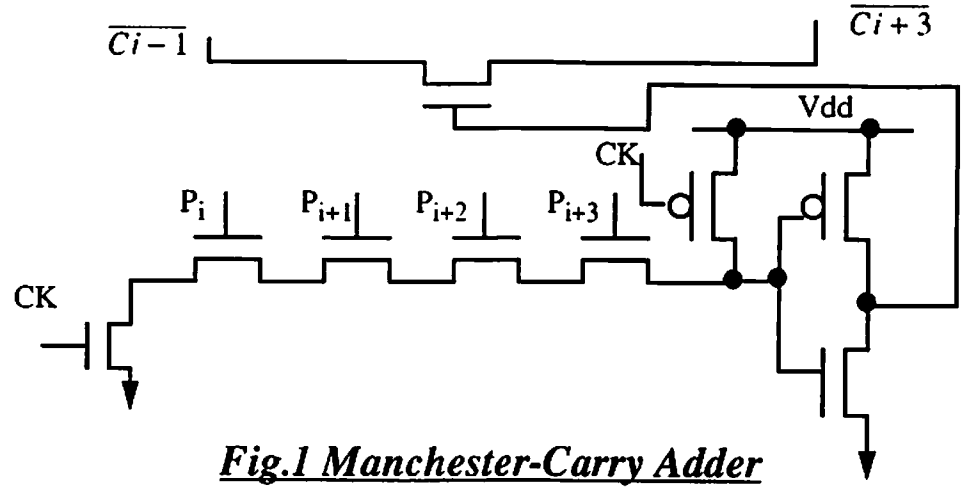
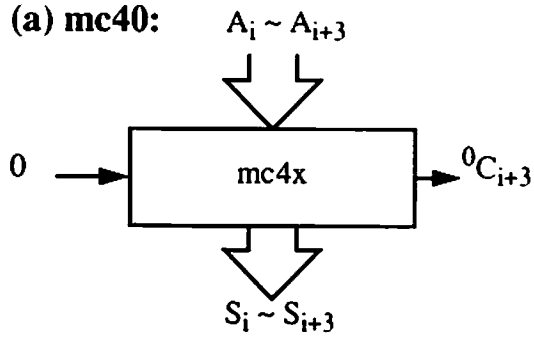
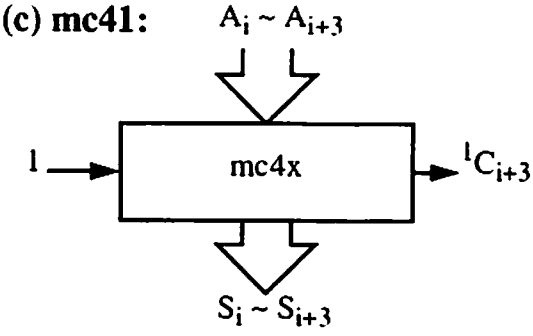


Fig.1 Manchester-Carry Adder

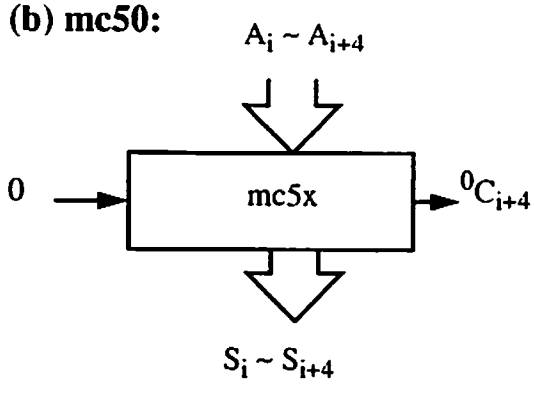
(a) mc40:



(c) mc41:



(b) mc50:



(d) mc51:

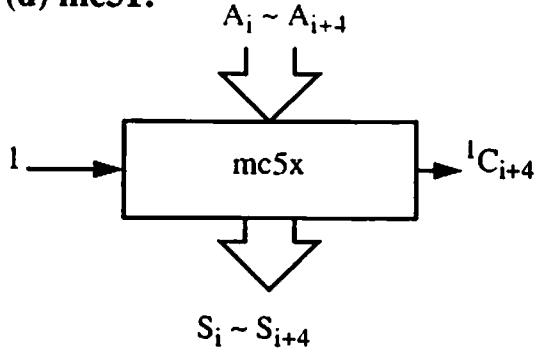
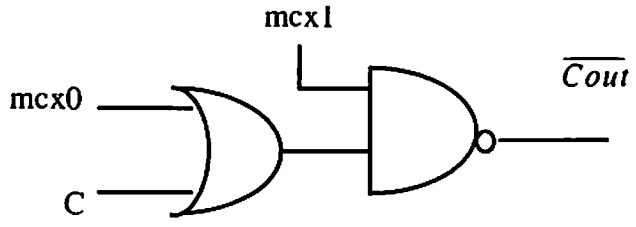
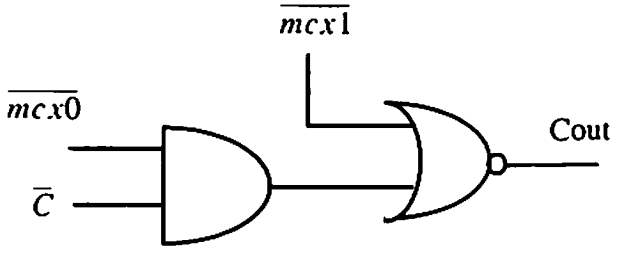


Fig. 2 n-bit Manchester-Carry Adder Cells



mcx0	mcx1	C	$\overline{Cout}$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Fig. 3a ONA (OR - NAND gate)

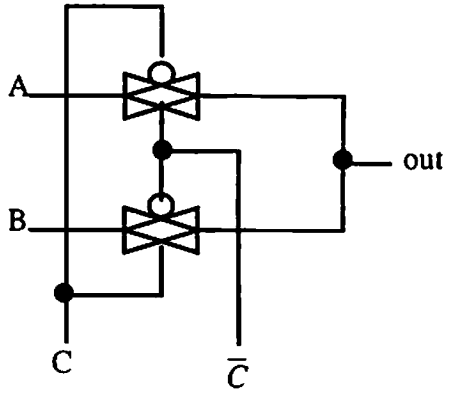


$\overline{mcx0}$	$\overline{mcx1}$	$\overline{C}$	Cout
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Fig. 3b ANO (AND - NOR gate)

mx4: 8-input multiplexer

mx5: 10-input multiplexer



2-input multiplexer

Fig. 4 Multiplexer

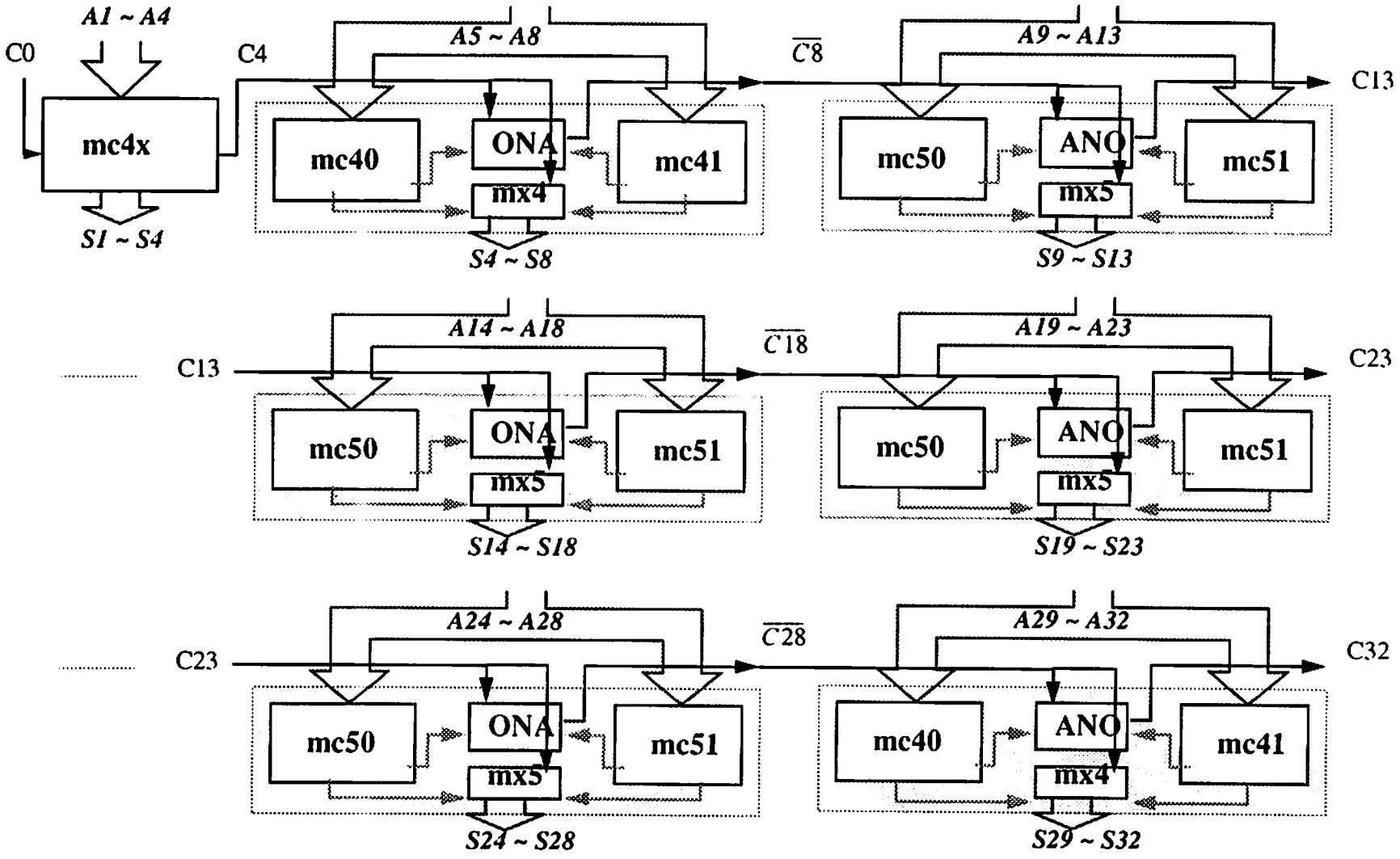


Fig. 5 THE STRUCTURE OF 32 + 32-b ADDER

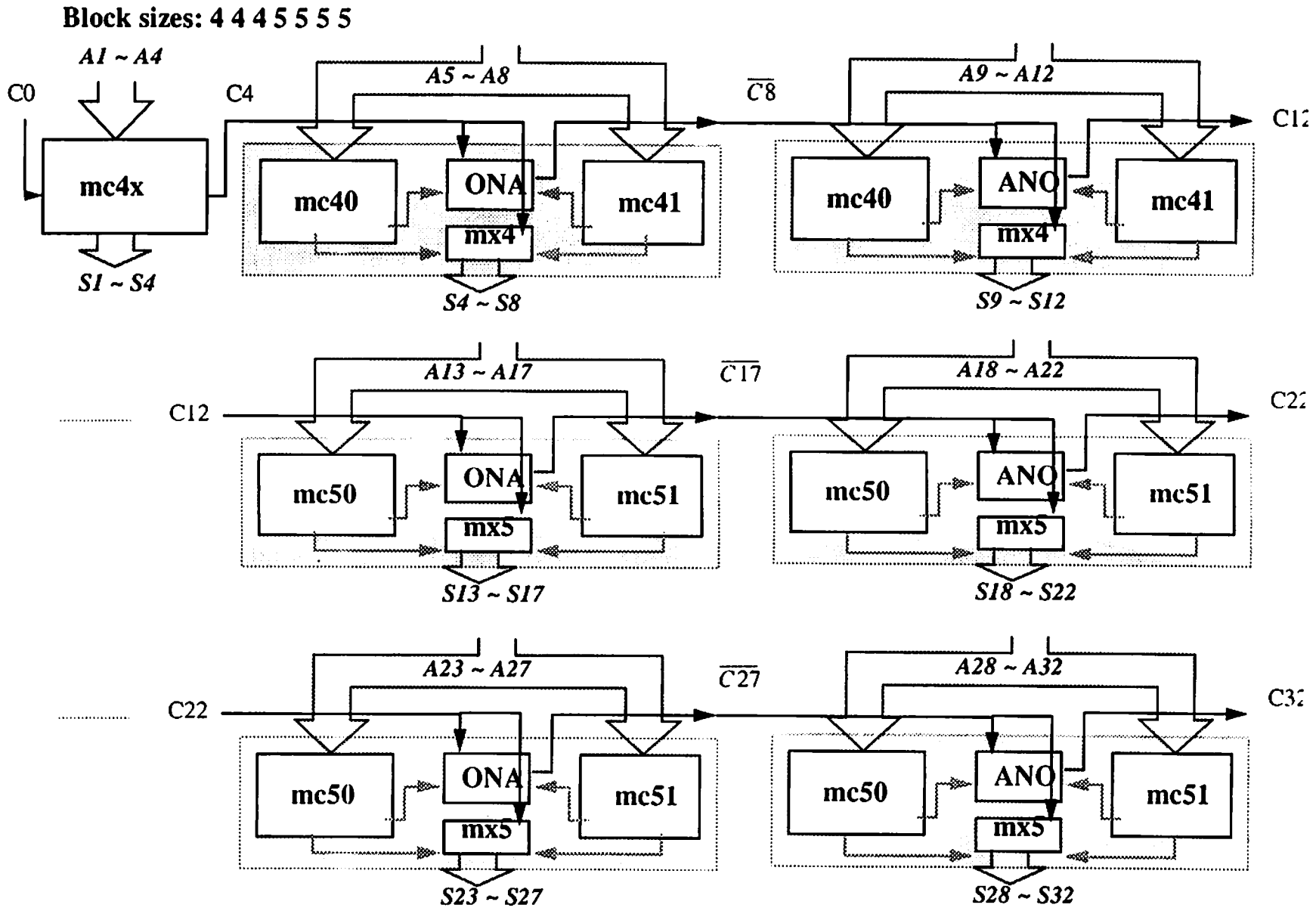
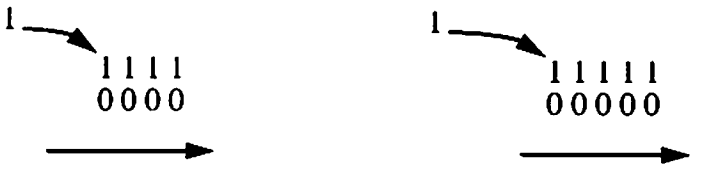


Fig. 6 THE STRUCTURE OF 32 + 32-b ADDER

1. The longest latency of one block without Manchester lookahead circuitry:

mc4: **mc5:**



2. The longest latency of one block with Manchester lookahead circuitry:

mc4: **mc5:**

The critical path of Manchester adder is the serially connected pass transistors in the carry chain. The bypass transistor (Manchester lookahead circuitry) improves the longest latency carry propagation time if all carry propagates P_1, P_2, ..., P_i are true [1].

All results and simulations in this project are using the longest latency analysis (2) for each block as above because every block has Manchester lookahead circuitry.

Fig. 7 THE LONGEST LATENCY ANALYSIS

$C_g = C_{ox} * A$

$$C_{ox} = \frac{\varepsilon_o \times \varepsilon_{ox}}{t_{ox}} = \frac{8.86 \times 10^{-12} \times 3.9}{1.1 \times 10^{-8}} = 3.14 \text{ fF/um}^2$$

$Cg = 3.14 \text{ fF/um}^2 * L * W \text{ (fF)}$

Capacitance loading for:

i) mx4: $3.14 \times (1+2) \times 0.5 \times 5 = 23.55fF$

ii) mx5: $3.14 \times (1+2) \times 0.5 \times 6 = 28.26fF$

iii) ANO: $3.14 \times (8+4) \times 0.5 = 18.84fF$

iv) ONA: $3.14 \times (4+4) \times 0.5 = 12.56fF$

Fig. 8 ESITIMATE CAPACITANCE LOADING

Table 1. CAPACITANCE LOADING OF:

loading	C_4	$\overline{C_8}$	C_{12}	$\overline{C_{17}}$	C_{22}	$\overline{C_{27}}$
mx4	23.55	23.55				
mx5			28.26	28.26	28.26	28.26
ONA	12.56		12.56		12.56	
ANO		18.84		18.84		18.84

Table 2. DELAY OF EACH CELL

a) Carry Select Adder using modified ripple Adder Cells:

	ripp4	ripp5	ONA	ANO	mx4	mx5
Rising Time	3.13ns (S) 3.88ns (C)	4.24ns (S) 4.78ns (C)	0.125ns	0.21ns	0.16ns	0.16ns
Falling Time	4.04ns (S) 2.73ns (C)	5.41ns (S) 4.34ns (C)	0.20ns	0.09ns	0.21ns	0.22ns
Delay	***3.58ns (S)*** ***3.30ns (C)***	***4.82ns (S)*** ***4.56ns (C)***	***0.16ns***	***0.15ns***	***0.18ns***	***0.19ns***

ps. the delay time of 4-bit ripple adder is from homework set #3.

b) Carry Select Adder using Manchester Carry Adder Cells:

	mc4	mc5	ONA	ANO	mx4	mx5
Rising Time	0.42ns (S) 2.33ns (C)	0.45ns (S) 0.22ns (C)	0.125ns	0.21ns	0.16ns	0.16ns
Falling Time	2.27ns (S) 0.29ns (C)	3.65ns (S) 3.73ns (C)	0.20ns	0.09ns	0.21ns	0.22ns
Delay	***1.34ns (S)*** ***1.31ns (C)***	***2.05ns (S)*** ***1.97ns (C)***	***0.16ns***	***0.15ns***	***0.18ns***	***0.19ns***

Block sizes: *4 4 4 5 5 5 5*

Delay = 4.56ns + 0.16ns + 0.15ns + 0.16ns + 0.19ns = **5.22ns**

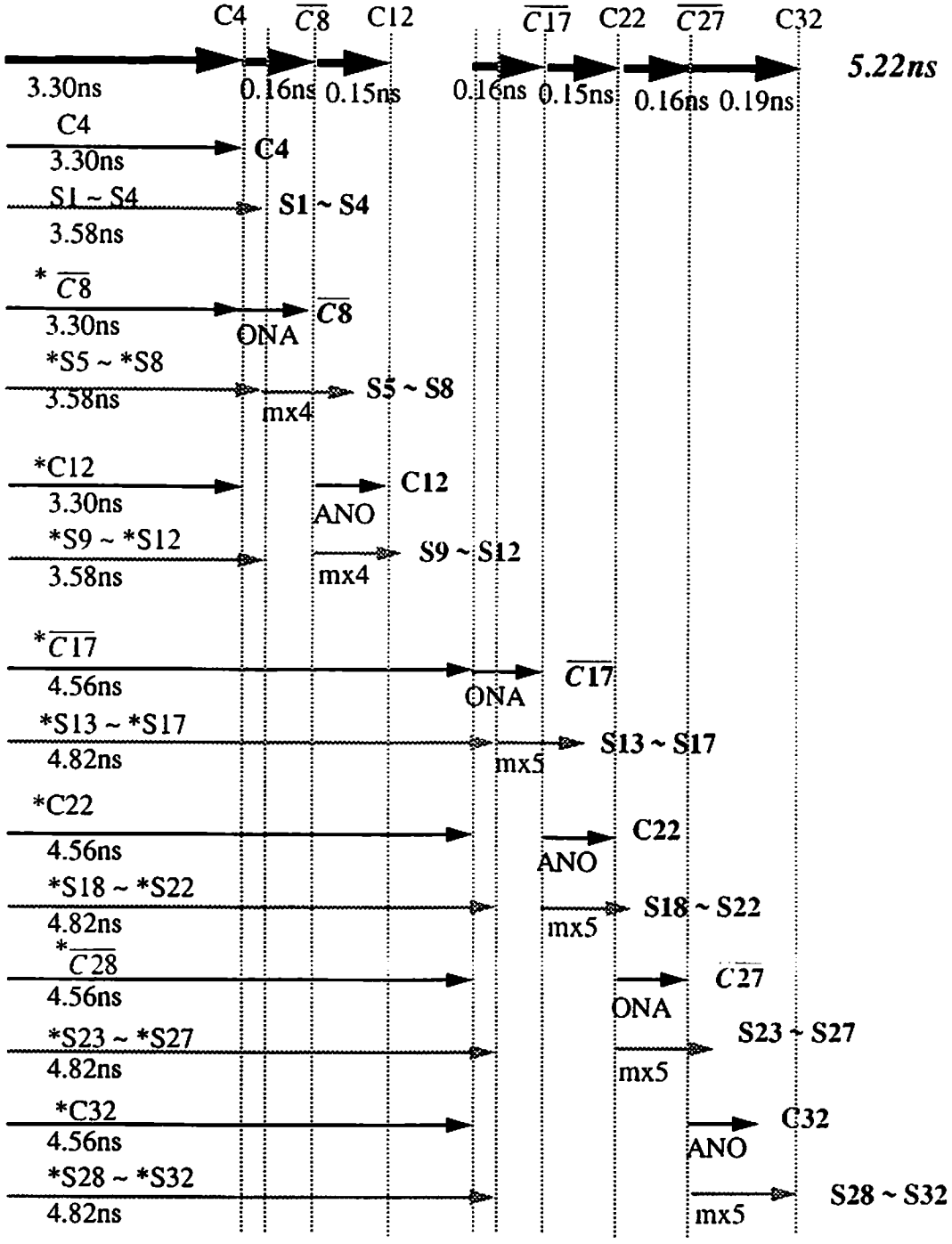


Fig. 9a Timing Analysis: (using Ripple Adder Cells)

Block sizes: *4 4 5 5 5 5 4*

Delay = 1.97ns + 0.15ns + 0.16ns + 0.15ns + 0.16ns +0.18ns = **2.77ns**

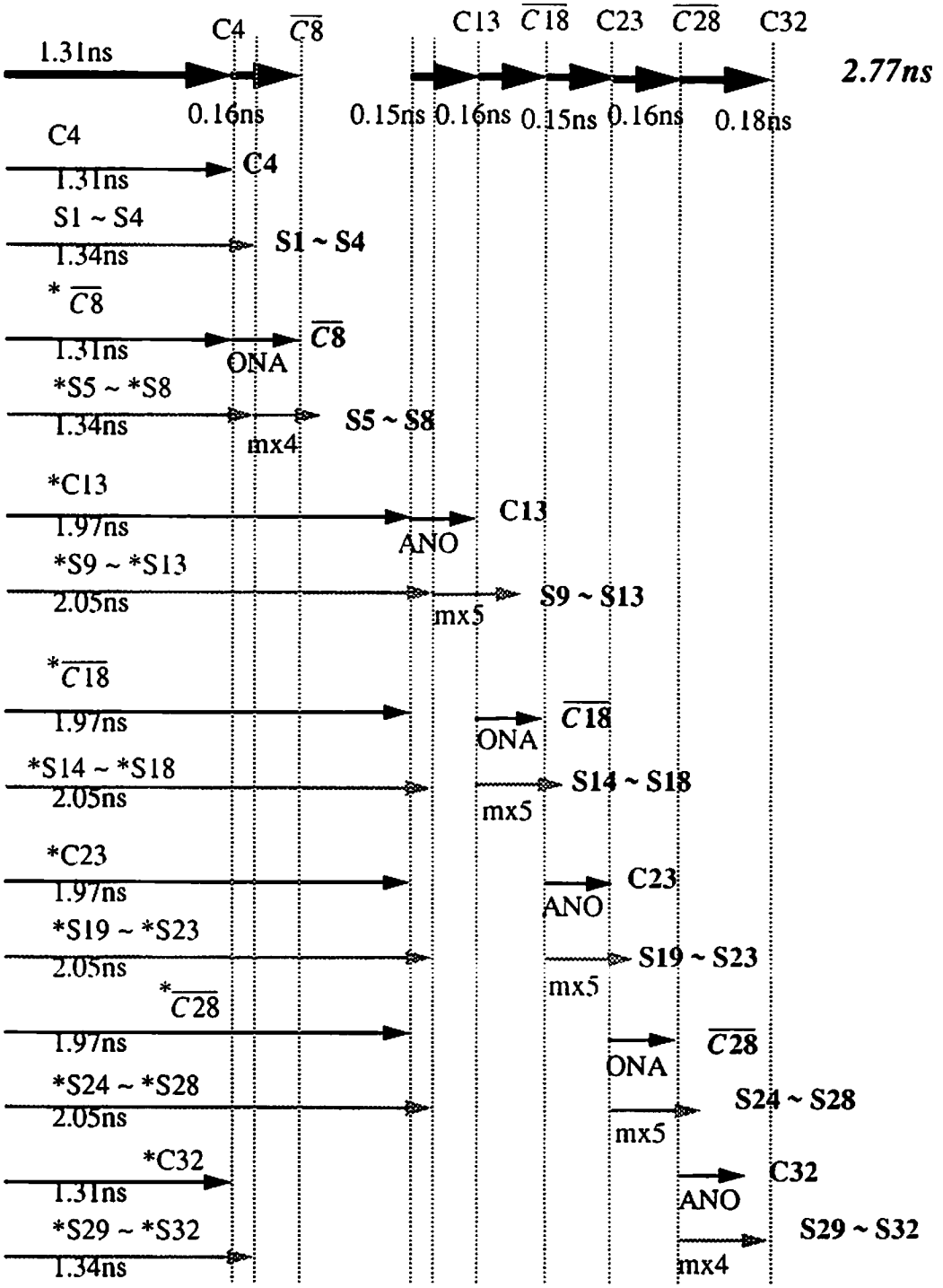


Fig. 9b Timing Analysis: (using Manchester Adder Cells)

Block sizes: *4 4 4 5 5 5 5*

Delay = 1.97ns + 0.16ns + 0.15ns + 0.16ns + 0.19ns = **2.63ns**

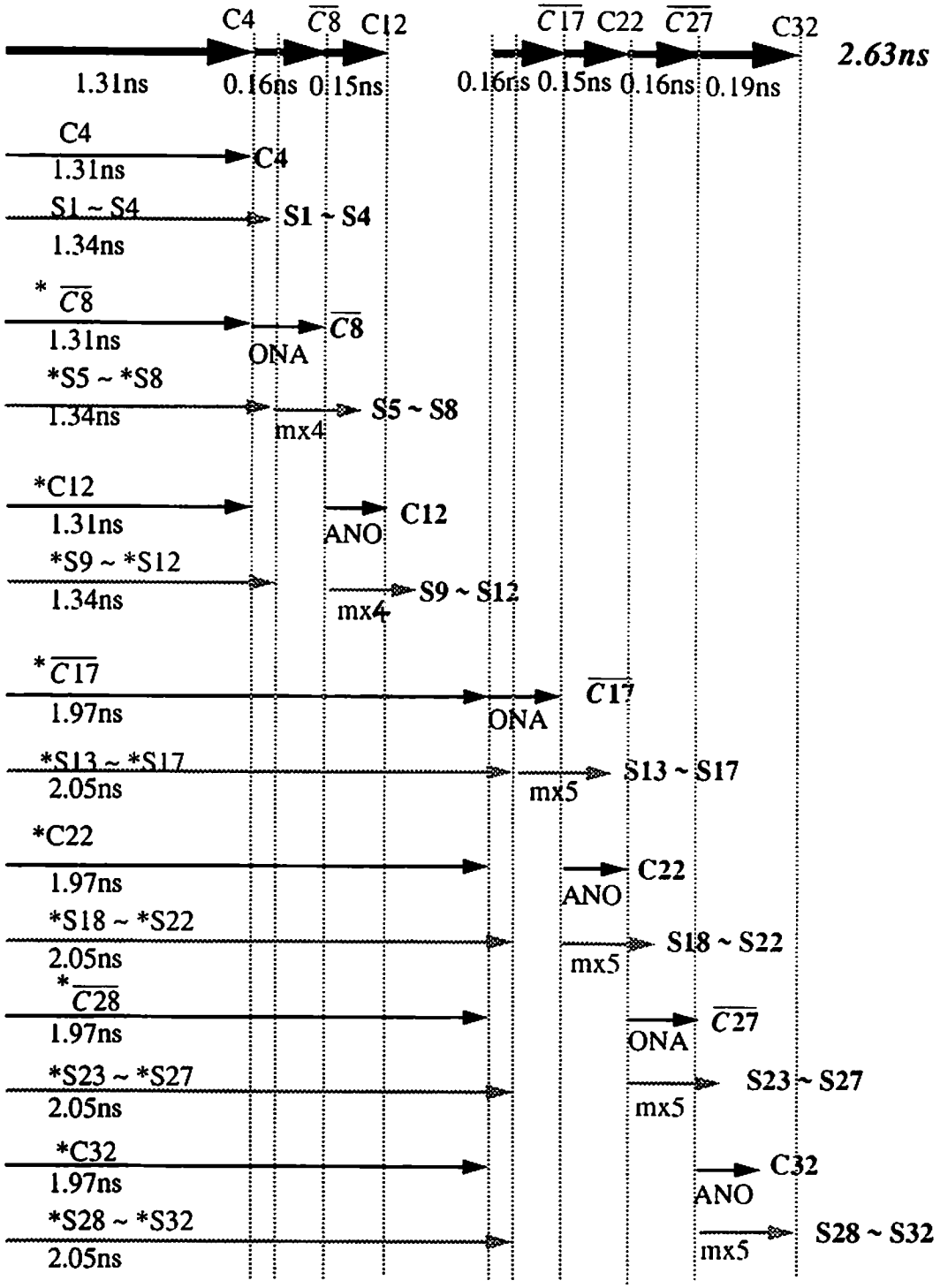


Fig. 9c Timing Analysis: (using Manchester Adder Cells)

Table 3. AREA

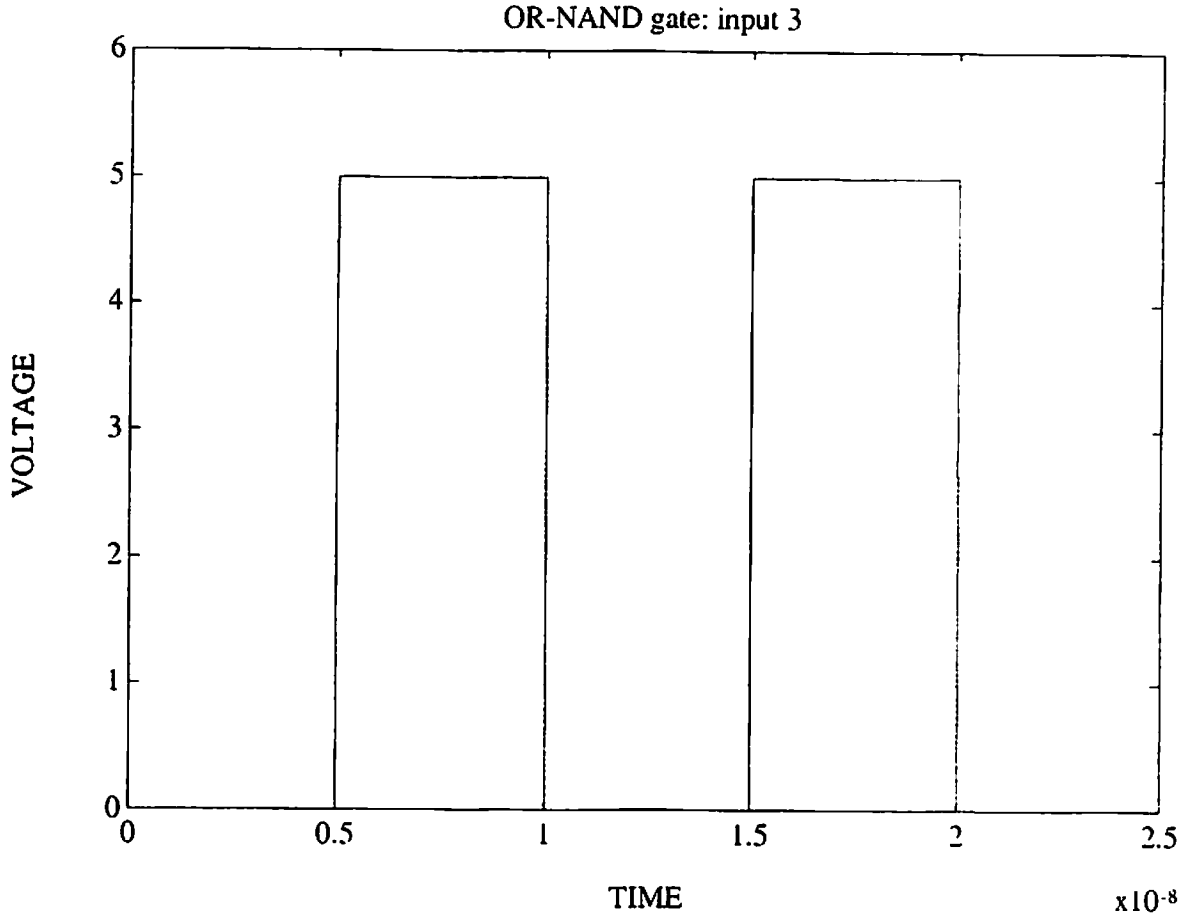
ripple	mx4	mx5	rippl4x	blk(4-b)	blk(5-b)	**total**
height λ	57	57	167	167	167	***167***
width λ	113	137	1100	2313	2837	***17074***
Manch.	mx4	mx5	mc4x	blk(4-b)	blk(5-b)	**total**
height λ	57	57	200	200	200	***200***
width λ	113	137	434	944	1132	***6920***

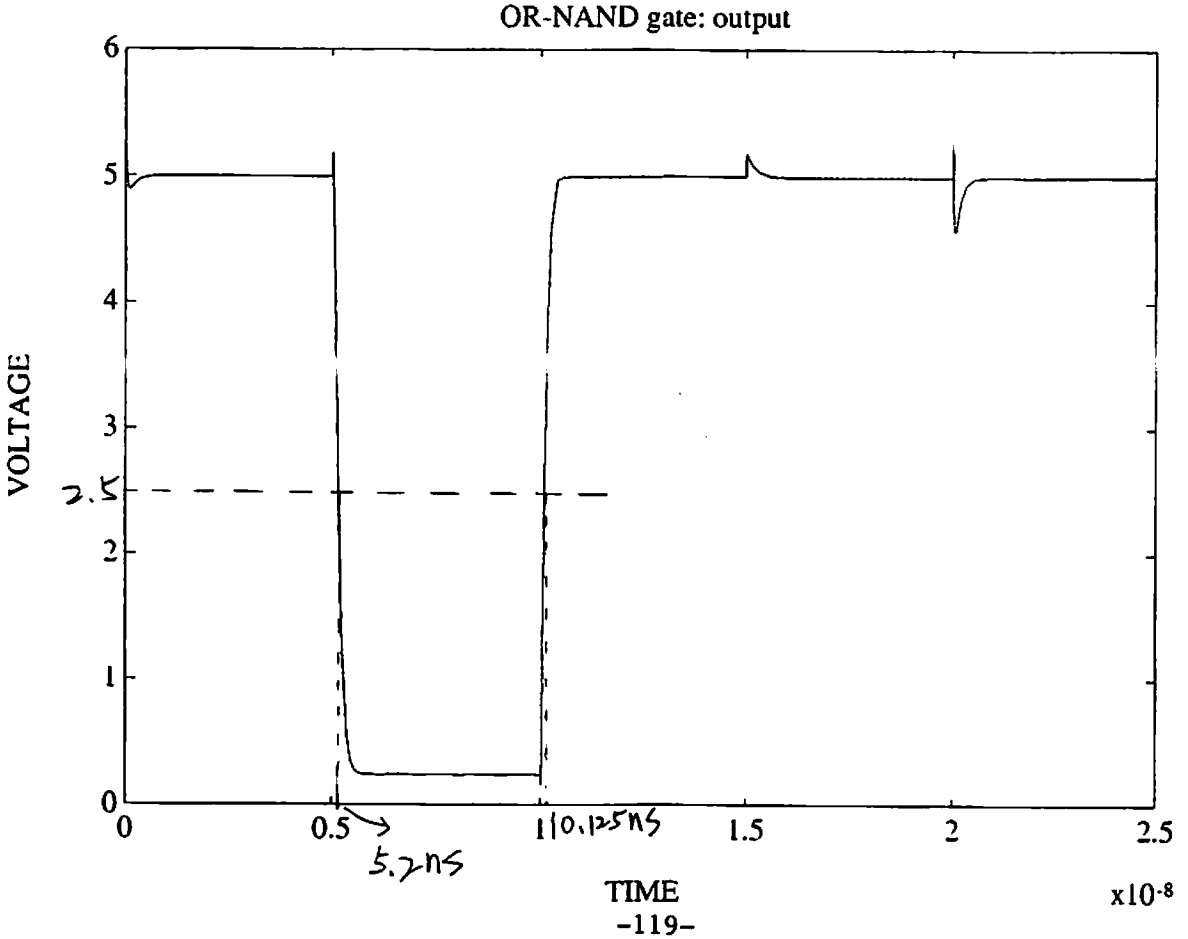
ps. the data of the ripple adder are from homework set #3.

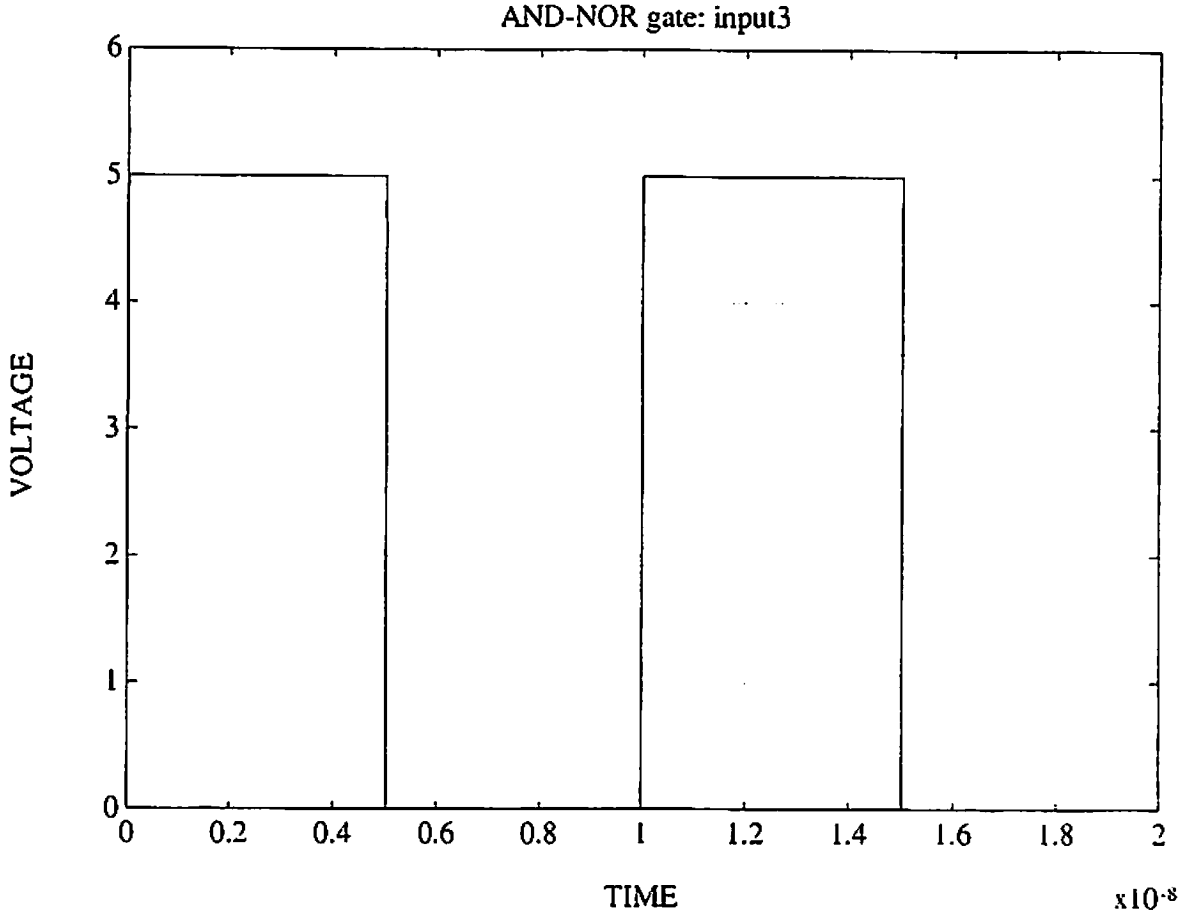
Table 4. FINAL RESULT

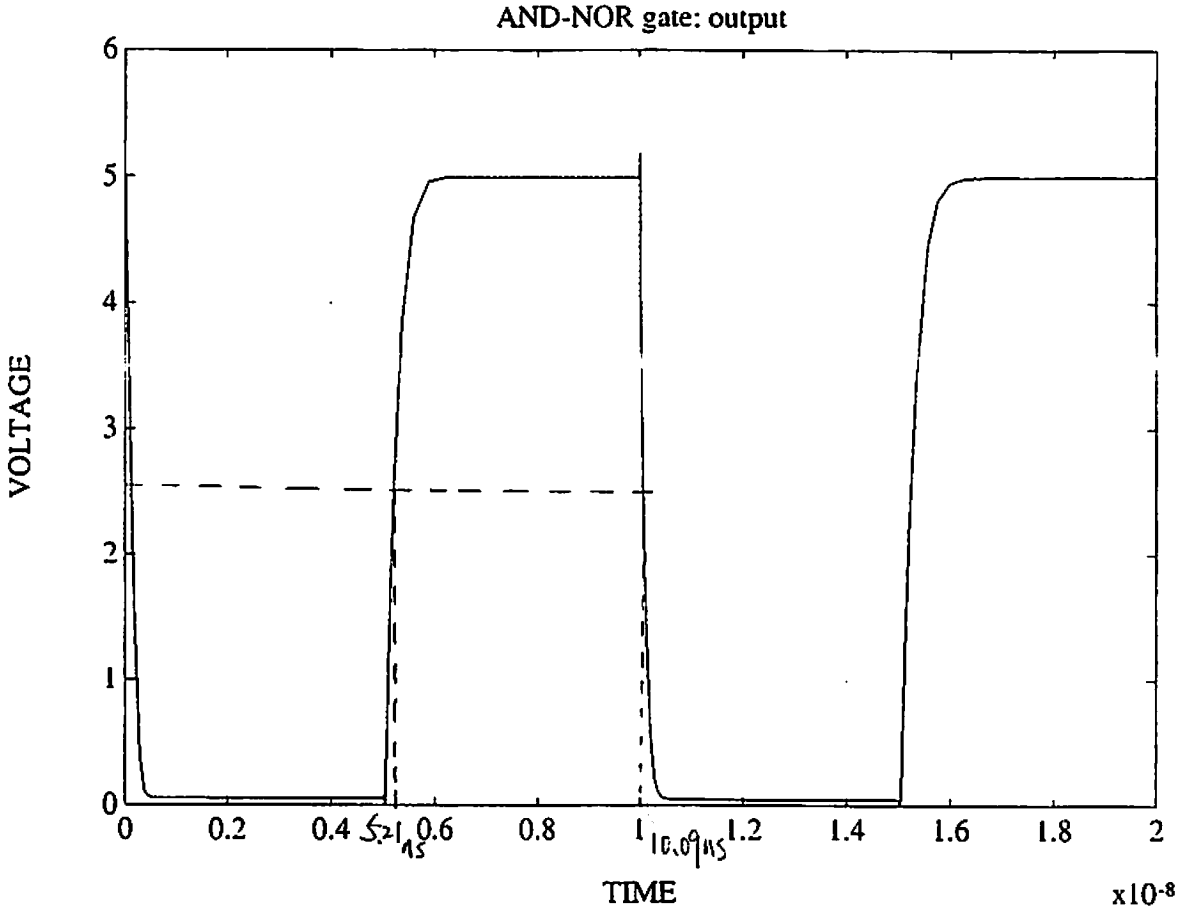
Type	Block sizes	Delay	Area
ripple	4 4 4 5 5 5 5	5.22ns	167λ × 17074λ
Manch.	4 4 5 5 5 5 4	2.77ns	200λ × 6920λ
Manch.	4 4 4 5 5 5 5	2.63ns	200λ × 6920λ

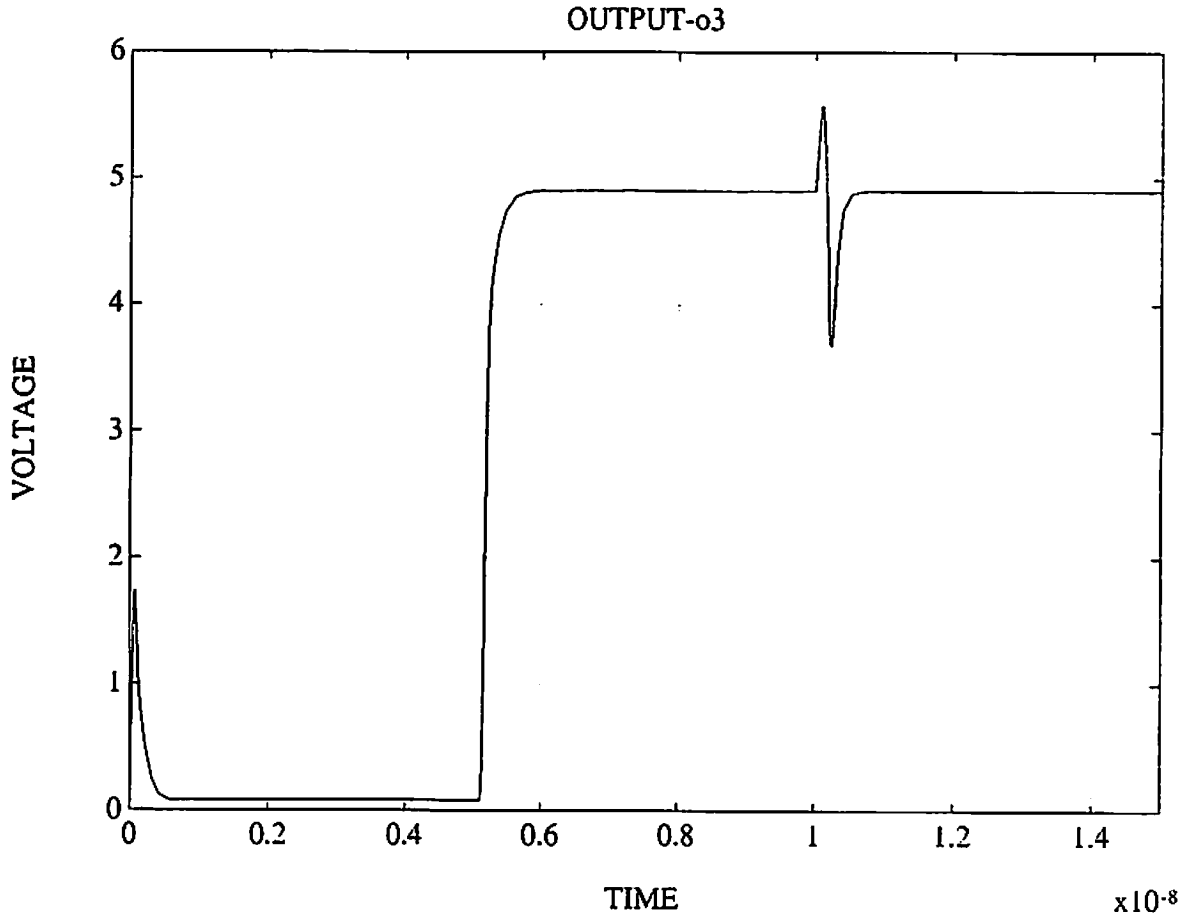
APPENDIX II: SIMULATION RESULT

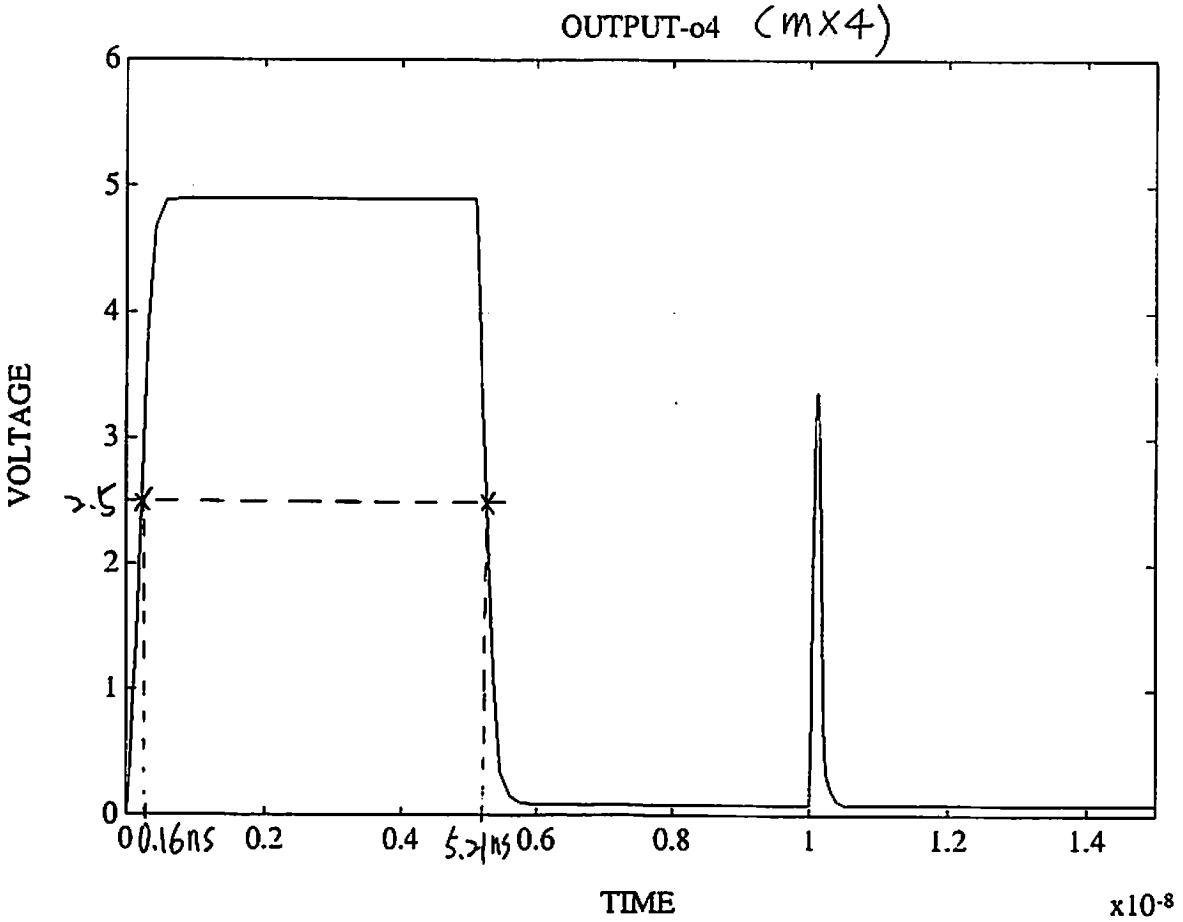


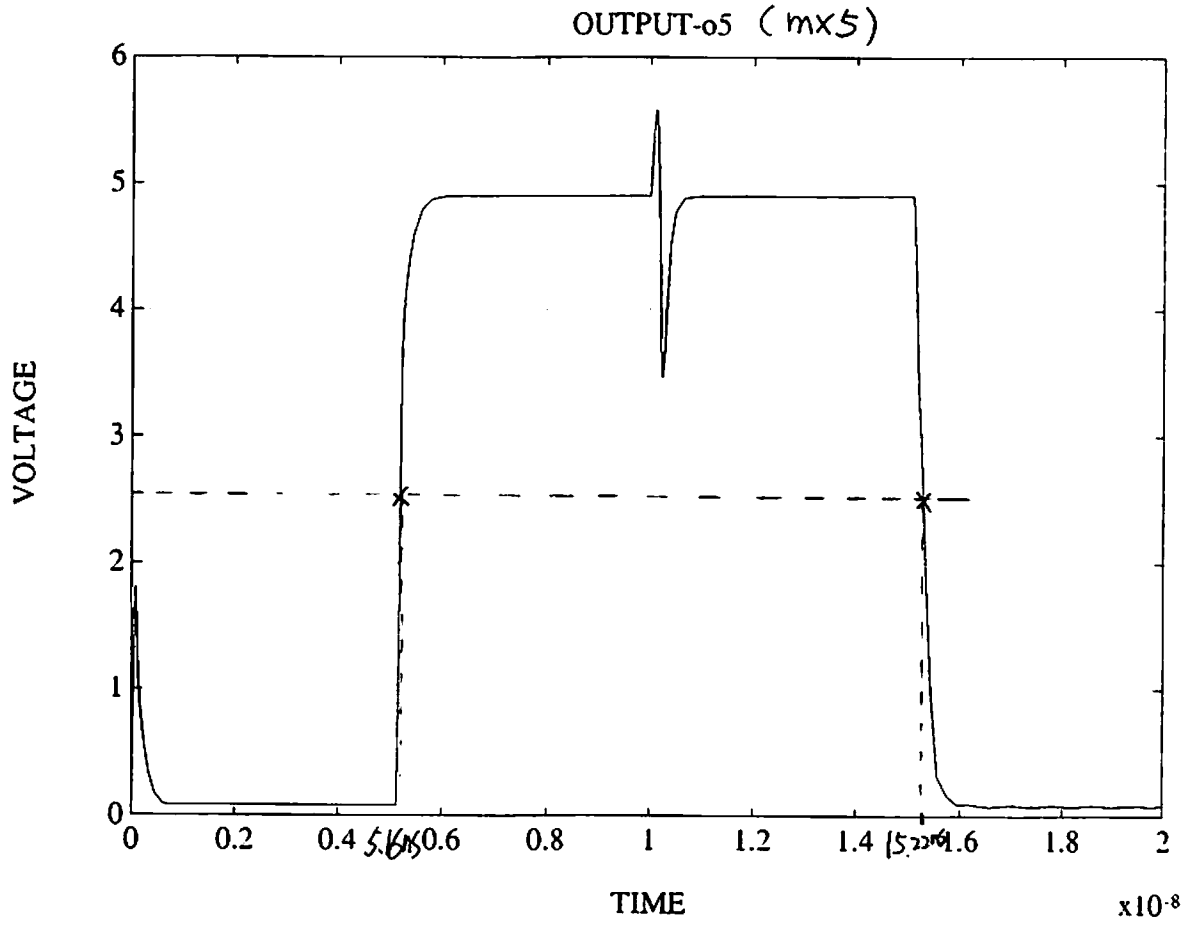


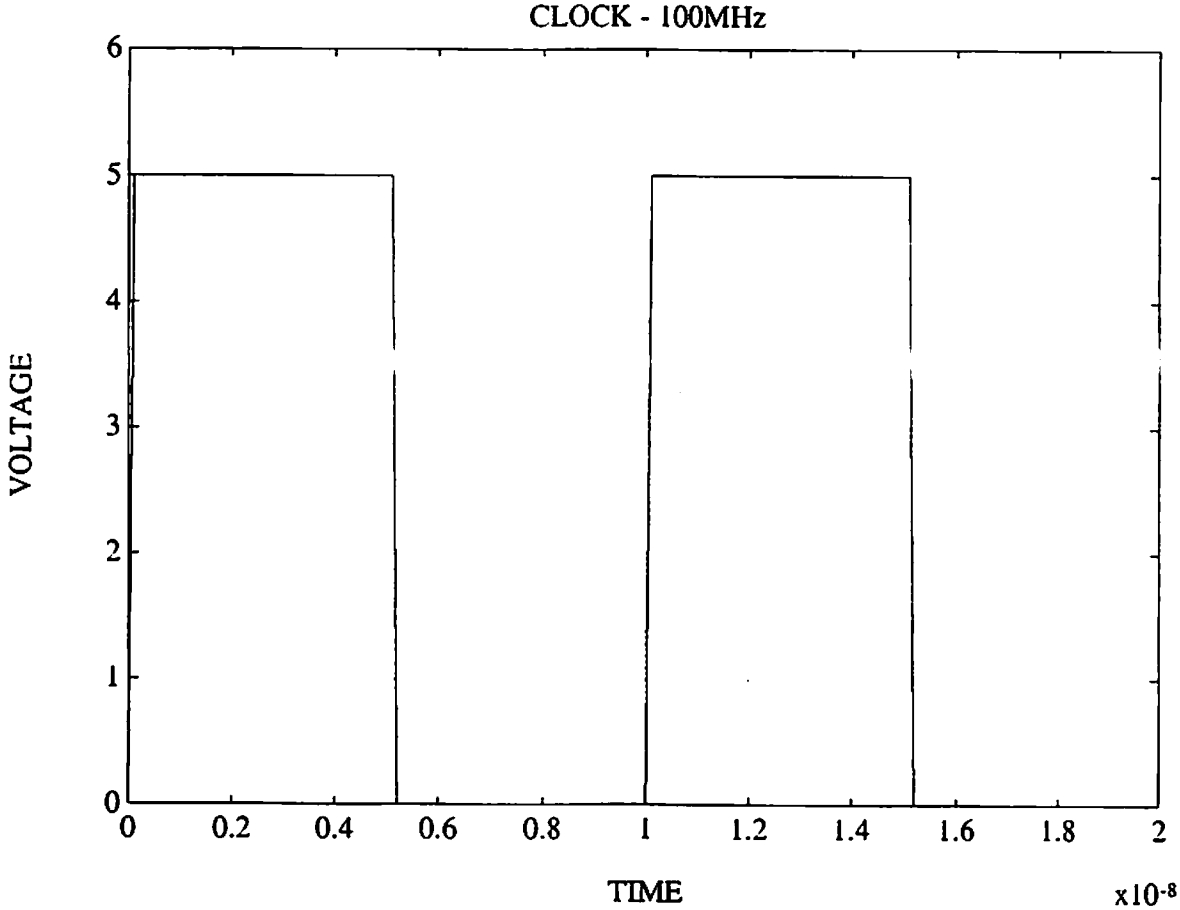

AND-NOR gate: input3
VOLTAGE
TIME
x10-8
0
1
2
3
4
5
6
0.2
0.4
0.6
0.8
1.2
1.4
1.6
1.8


AND-NOR gate: output
VOLTAGE
TIME
x10-8
0
1
2
3
4
5
6
0.2
0.4
0.6
0.8
1.2
1.4
1.6
1.8


OUTPUT-o3
VOLTAGE
TIME
x10-8
0
1
2
3
4
5
6
0
0.2
0.4
0.6
0.8
1
1.2
1.4


OUTPUT-o4 (mx4)
VOLTAGE
TIME
x10-8
2.5
0.16ns
5.2ns
0
1
2
3
4
5
6
0
0.2
0.4
0.6
0.8
1
1.2
1.4


OUTPUT-o5 (mx5)
VOLTAGE
TIME
x10-8


CLOCK - 100MHz
VOLTAGE
TIME
x10-8

```
** SPICE file created for circuit pgt
** Technology: scmos


** NODE: 0 = GND
** NODE: 1 = Vdd
** NODE: 2 = Error
RLUMP0 100 101  1698.5
RLUMP1 102 103  1847.0
RLUMP2 104 105  1654.5
M0 101 103 105 1 pfet L=0.5U W=8.0U
RLUMP3 102 106  1847.0
RLUMP4 104 107  1654.5
RLUMP5 100 108  1698.5
M1 106 107 108 1 pfet L=0.5U W=8.0U
RLUMP6 102 109  1847.0
RLUMP7 110 111   906.0
M2 1 109 111 1 pfet L=0.5U W=8.0U
RLUMP8 112 113   982.5
RLUMP9 114 115   671.0
M3 1 113 115 1 pfet L=0.5U W=8.0U
RLUMP10 112 116   982.5
RLUMP11 104 117  1654.5
M4 116 117 1 1 pfet L=0.5U W=8.0U
RLUMP12 102 118  1847.0
RLUMP13 112 119   982.5
M5 1 118 119 1 pfet L=0.5U W=8.0U
RLUMP14 100 120  1698.5
RLUMP15 110 121   906.0
RLUMP16 104 122  1654.5
M6 120 121 122 0 nfet L=0.5U W=4.0U
RLUMP17 110 123   906.0
RLUMP18 104 124  1654.5
RLUMP19 100 125  1698.5
M7 123 124 125 0 nfet L=0.5U W=4.0U
RLUMP20 102 126  1847.0
RLUMP21 110 127   906.0
M8 0 126 127 0 nfet L=0.5U W=4.0U
RLUMP22 112 128   982.5
RLUMP23 114 129   671.0
M9 0 128 129 0 nfet L=0.5U W=4.0U
RLUMP24 130 131   100.0
RLUMP25 104 132  1654.5
M10 131 132 0 0 nfet L=0.5U W=4.0U
RLUMP26 112 133   982.5
RLUMP27 102 134  1847.0
RLUMP28 130 135   100.0
M11 133 134 135 0 nfet L=0.5U W=4.0U
RLUMP29 136 137  2992.0
RLUMP30 138 139   300.0
M12 0 137 139 0 nfet L=0.5U W=4.0U
RLUMP31 136 140  2992.0
RLUMP32 141 142  1584.0
M13 1 140 142 1 pfet L=0.5U W=8.0U
RLUMP33 138 143   300.0
RLUMP34 114 144   671.0
RLUMP35 141 145  1584.0
M14 143 144 145 0 nfet L=0.5U W=4.0U
RLUMP36 141 146  1584.0
RLUMP37 100 147  1698.5
RLUMP38 148 149  1584.0
M15 146 147 149 0 nfet L=0.5U W=4.0U
RLUMP39 148 150  1584.0
RLUMP40 151 152   940.5
M16 1 150 152 1 pfet L=0.5U W=8.0U
RLUMP41 148 153  1584.0
RLUMP42 154 155   100.0
M17 0 153 155 0 nfet L=0.5U W=4.0U
RLUMP43 151 156   940.5
RLUMP44 100 157  1698.5
M18 156 157 1 1 pfet L=0.5U W=8.0U
RLUMP45 154 158   100.0
RLUMP46 100 159  1698.5
RLUMP47 151 160   940.5
M19 158 159 160 0 nfet L=0.5U W=4.0U
RLUMP48 161 162   935.0
RLUMP49 151 163   940.5
M20 162 163 1 1 pfet L=0.5U W=8.0U
RLUMP50 151 164   940.5
RLUMP51 165 166   274.5
M21 0 164 166 0 nfet L=0.5U W=4.0U
RLUMP52 148 167  1584.0
RLUMP53 168 169   320.0
M22 1 167 169 1 pfet L=0.5U W=8.0U
RLUMP54 165 170   274.5
RLUMP55 148 171  1584.0
RLUMP56 161 172   935.0
M23 170 171 172 0 nfet L=0.5U W=4.0U
RLUMP57 168 173   320.0
RLUMP58 100 174  1698.5
RLUMP59 161 175   935.0
M24 173 174 175 1 pfet L=0.5U W=8.0U
RLUMP60 161 176   935.0
RLUMP61 100 177  1698.5
RLUMP62 165 178   274.5
M25 176 177 178 0 nfet L=0.5U W=4.0U
RLUMP63 179 180  1699.0
RLUMP64 181 182  1847.0
RLUMP65 183 184  1654.5
M26 180 182 184 1 pfet L=0.5U W=8.0U
RLUMP66 181 185  1847.0
RLUMP67 183 186  1654.5
RLUMP68 179 187  1699.0
M27 185 186 187 1 pfet L=0.5U W=8.0U
RLUMP69 181 188  1847.0
RLUMP70 189 190   906.0
M28 1 188 190 1 pfet L=0.5U W=8.0U
RLUMP71 191 192   982.5
RLUMP72 193 194   671.0
M29 1 192 194 1 pfet L=0.5U W=8.0U
RLUMP73 191 195   982.5
RLUMP74 183 196  1654.5
M30 195 196 1 1 pfet L=0.5U W=8.0U
RLUMP75 181 197  1847.0
RLUMP76 191 198   982.5
M31 1 197 198 1 pfet L=0.5U W=8.0U
RLUMP77 179 199  1699.0
RLUMP78 189 200   906.0
RLUMP79 183 201  1654.5
M32 199 200 201 0 nfet L=0.5U W=4.0U
RLUMP80 189 202   906.0
RLUMP81 183 203  1654.5
RLUMP82 179 204  1699.0
M33 202 203 204 0 nfet L=0.5U W=4.0U
RLUMP83 181 205  1847.0
RLUMP84 189 206   906.0
M34 0 205 206 0 nfet L=0.5U W=4.0U
RLUMP85 191 207   982.5
```

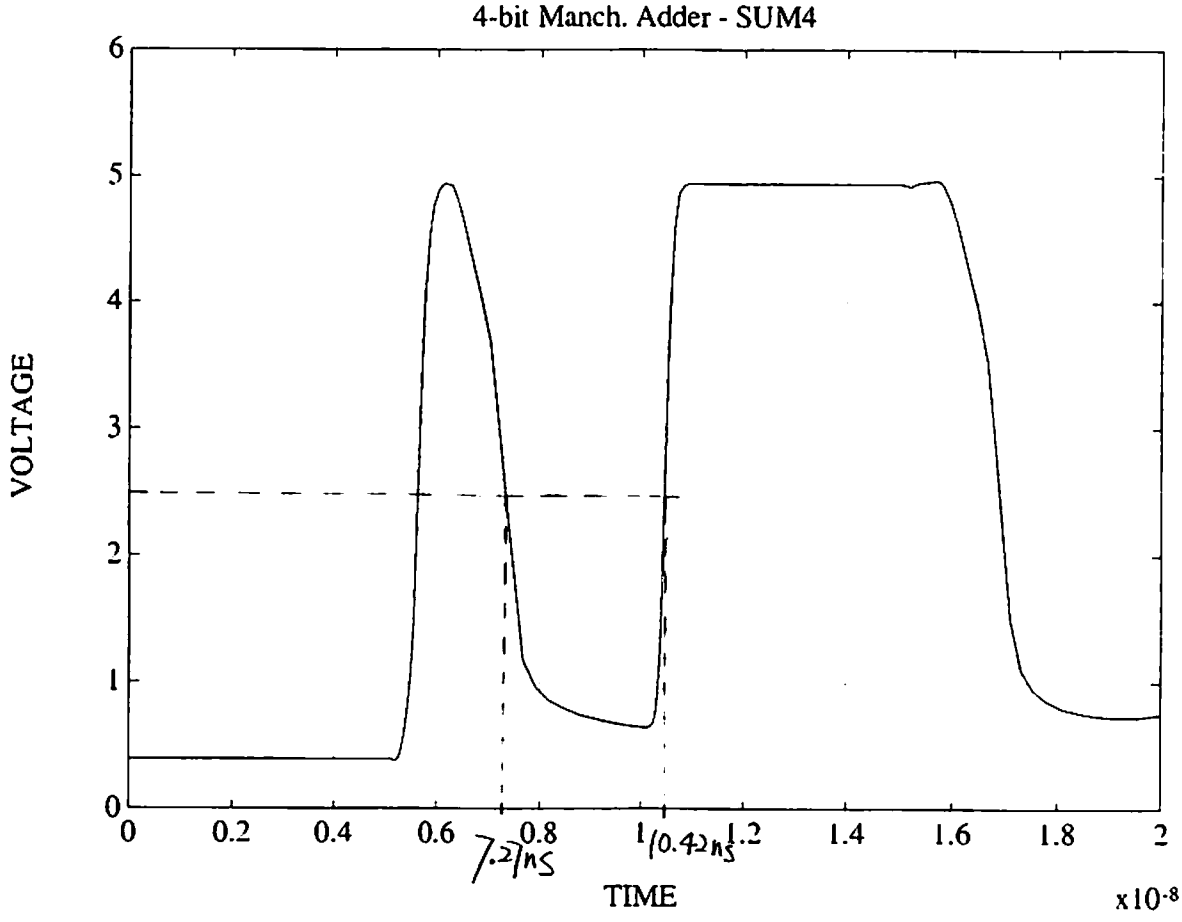
```
RLUMP86 193 208   671.0
M35 0 207 208 0 nfet L=0.5U W=4.0U
RLUMP87 209 210   100.0
RLUMP88 183 211  1654.5
M36 210 211 0 0 nfet L=0.5U W=4.0U
RLUMP89 191 212   982.5
RLUMP90 181 213  1847.0
RLUMP91 209 214   100.0
M37 212 213 214 0 nfet L=0.5U W=4.0U
RLUMP92 136 215  2992.0
RLUMP93 216 217   300.0
M38 0 215 217 0 nfet L=0.5U W=4.0U
RLUMP94 136 218  2992.0
RLUMP95 219 220   699.5
M39 1 218 220 1 pfet L=0.5U W=8.0U
RLUMP96 216 221   300.0
RLUMP97 193 222   671.0
RLUMP98 219 223   699.5
M40 221 222 223 0 nfet L=0.5U W=4.0U
RLUMP99 219 224   699.5
RLUMP100 179 225  1699.0
RLUMP101 141 226  1584.0
M41 224 225 226 0 nfet L=0.5U W=4.0U
RLUMP102 141 227  1584.0
RLUMP103 228 229   940.5
M42 1 227 229 1 pfet L=0.5U W=8.0U
RLUMP104 141 230  1584.0
RLUMP105 231 232   100.0
M43 0 230 232 0 nfet L=0.5U W=4.0U
RLUMP106 228 233   940.5
RLUMP107 179 234  1699.0
M44 233 234 1 1 pfet L=0.5U W=8.0U
RLUMP108 231 235   100.0
RLUMP109 179 236  1699.0
RLUMP110 228 237   940.5
M45 235 236 237 0 nfet L=0.5U W=4.0U
RLUMP111 238 239   935.0
RLUMP112 228 240   940.5
M46 239 240 1 1 pfet L=0.5U W=8.0U
RLUMP113 228 241   940.5
RLUMP114 242 243   274.5
M47 0 241 243 0 nfet L=0.5U W=4.0U
RLUMP115 141 244  1584.0
RLUMP116 245 246   320.0
M48 1 244 246 1 pfet L=0.5U W=8.0U
RLUMP117 242 247   274.5
RLUMP118 141 248  1584.0
RLUMP119 238 249   935.0
M49 247 248 249 0 nfet L=0.5U W=4.0U
RLUMP120 245 250   320.0
RLUMP121 179 251  1699.0
RLUMP122 238 252   935.0
M50 250 251 252 1 pfet L=0.5U W=8.0U
RLUMP123 238 253   935.0
RLUMP124 179 254  1699.0
RLUMP125 242 255   274.5
M51 253 254 255 0 nfet L=0.5U W=4.0U
RLUMP126 256 257  1729.0
RLUMP127 258 259  1847.0
RLUMP128 260 261  1654.5
M52 257 259 261 1 pfet L=0.5U W=8.0U
RLUMP129 258 262  1847.0
RLUMP130 260 263  1654.5
RLUMP131 256 264  1729.0
M53 262 263 264 1 pfet L=0.5U W=8.0U
RLUMP132 258 265  1847.0
RLUMP133 266 267   906.0
M54 1 265 267 1 pfet L=0.5U W=8.0U
RLUMP134 268 269   982.5
RLUMP135 270 271   671.0
M55 1 269 271 1 pfet L=0.5U W=8.0U
RLUMP136 268 272   982.5
RLUMP137 260 273  1654.5
M56 272 273 1 1 pfet L=0.5U W=8.0U
RLUMP138 258 274  1847.0
RLUMP139 268 275   982.5
M57 1 274 275 1 pfet L=0.5U W=8.0U
RLUMP140 256 276  1729.0
RLUMP141 266 277   906.0
RLUMP142 260 278  1654.5
M58 276 277 278 0 nfet L=0.5U W=4.0U
RLUMP143 266 279   906.0
RLUMP144 260 280  1654.5
RLUMP145 256 281  1729.0
M59 279 280 281 0 nfet L=0.5U W=4.0U
RLUMP146 258 282  1847.0
RLUMP147 266 283   906.0
M60 0 282 283 0 nfet L=0.5U W=4.0U
RLUMP148 268 284   982.5
RLUMP149 270 285   671.0
M61 0 284 285 0 nfet L=0.5U W=4.0U
RLUMP150 286 287   100.0
RLUMP151 260 288  1654.5
M62 287 288 0 0 nfet L=0.5U W=4.0U
RLUMP152 268 289   982.5
RLUMP153 258 290  1847.0
RLUMP154 286 291   100.0
M63 289 290 291 0 nfet L=0.5U W=4.0U
RLUMP155 136 292  2992.0
RLUMP156 293 294   300.0
M64 0 292 294 0 nfet L=0.5U W=4.0U
RLUMP157 136 295  2992.0
RLUMP158 148 296  1584.0
M65 1 295 296 1 pfet L=0.5U W=8.0U
RLUMP159 293 297   300.0
RLUMP160 270 298   671.0
RLUMP161 148 299  1584.0
M66 297 298 299 0 nfet L=0.5U W=4.0U
RLUMP162 148 300  1584.0
RLUMP163 256 301  1729.0
RLUMP164 302 303  1583.5
M67 300 301 303 0 nfet L=0.5U W=4.0U
RLUMP165 302 304  1583.5
RLUMP166 305 306   940.5
M68 1 304 306 1 pfet L=0.5U W=8.0U
RLUMP167 302 307  1583.5
RLUMP168 308 309   100.0
M69 0 307 309 0 nfet L=0.5U W=4.0U
RLUMP169 305 310   940.5
RLUMP170 256 311  1729.0
M70 310 311 1 1 pfet L=0.5U W=8.0U
RLUMP171 308 312   100.0
RLUMP172 256 313  1729.0
RLUMP173 305 314   940.5
M71 312 313 314 0 nfet L=0.5U W=4.0U
RLUMP174 315 316   935.0
RLUMP175 305 317   940.5
M72 316 317 1 1 pfet L=0.5U W=8.0U
```

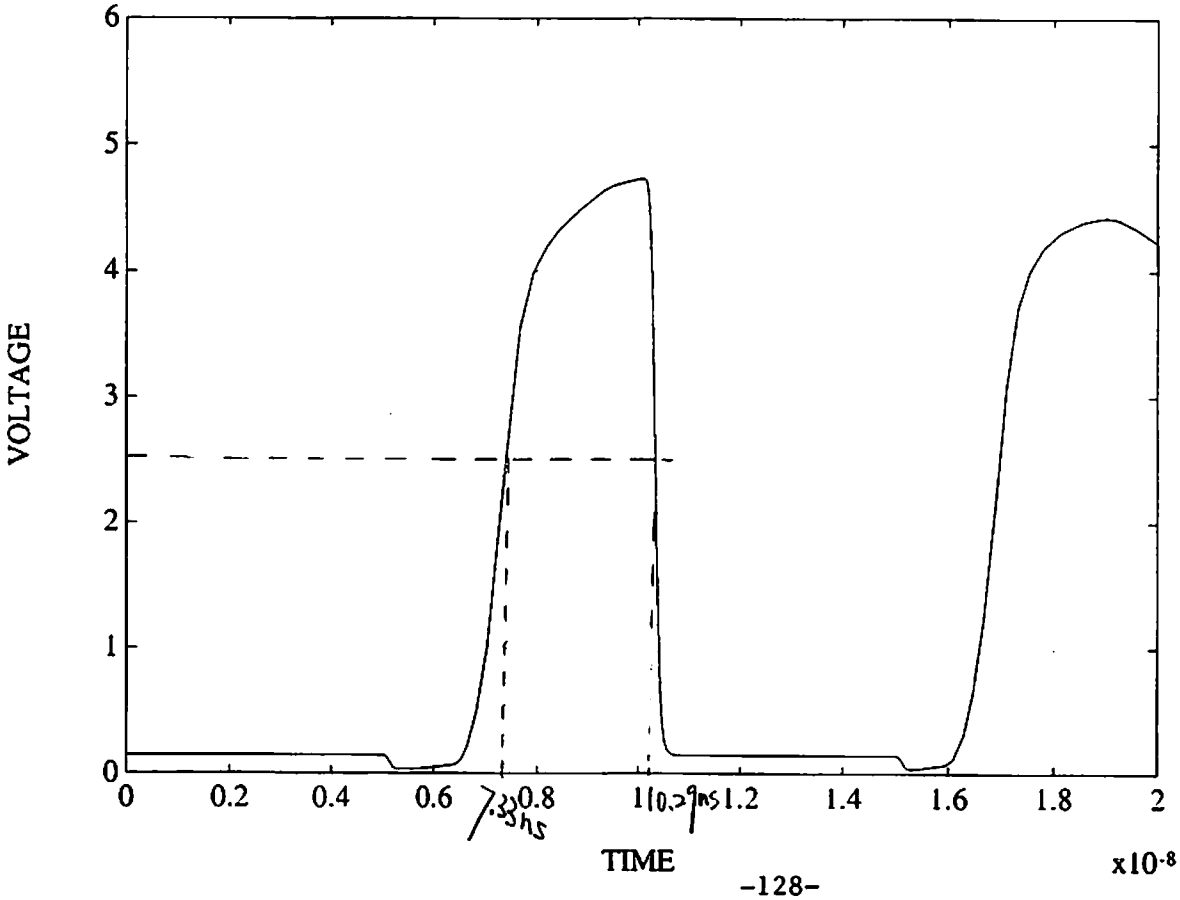
```
RLUMP176 305 318   940.5
RLUMP177 319 320   274.5
M73 0 318 320 0 nfet L=0.5U W=4.0U
RLUMP178 302 321  1583.5
RLUMP179 322 323   320.0
M74 1 321 323 1 pfet L=0.5U W=8.0U
RLUMP180 319 324   274.5
RLUMP181 302 325  1583.5
RLUMP182 315 326   935.0
M75 324 325 326 0 nfet L=0.5U W=4.0U
RLUMP183 322 327   320.0
RLUMP184 256 328  1729.0
RLUMP185 315 329   935.0
M76 327 328 329 1 pfet L=0.5U W=8.0U
RLUMP186 315 330   935.0
RLUMP187 256 331  1729.0
RLUMP188 319 332   274.5
M77 330 331 332 0 nfet L=0.5U W=4.0U
RLUMP189 333 334  1741.5
RLUMP190 335 336  1847.0
RLUMP191 337 338  1654.5
M78 334 336 338 1 pfet L=0.5U W=8.0U
RLUMP192 335 339  1847.0
RLUMP193 337 340  1654.5
RLUMP194 333 341  1741.5
M79 339 340 341 1 pfet L=0.5U W=8.0U
RLUMP195 335 342  1847.0
RLUMP196 343 344   906.0
M80 1 342 344 1 pfet L=0.5U W=8.0U
RLUMP197 345 346   982.5
RLUMP198 347 348   671.0
M81 1 346 348 1 pfet L=0.5U W=8.0U
RLUMP199 345 349   982.5
RLUMP200 337 350  1654.5
M82 349 350 1 1 pfet L=0.5U W=8.0U
RLUMP201 335 351  1847.0
RLUMP202 345 352   982.5
M83 1 351 352 1 pfet L=0.5U W=8.0U
RLUMP203 353 354   891.5
RLUMP204 355 356   660.0
M84 1 354 356 1 pfet L=0.5U W=8.0U
RLUMP205 353 357   891.5
RLUMP206 136 358  2992.0
M85 357 358 1 1 pfet L=0.5U W=8.0U
RLUMP207 219 359   699.5
RLUMP208 355 360   660.0
RLUMP209 361 362  1797.0
M86 359 360 362 0 nfet L=0.5U W=4.0U
RLUMP210 363 364   504.5
RLUMP211 219 365   699.5
M87 364 365 1 1 pfet L=0.5U W=8.0U
RLUMP212 353 366   891.5
RLUMP213 355 367   660.0
M88 0 366 367 0 nfet L=0.5U W=4.0U
RLUMP214 368 369   300.0
RLUMP215 136 370  2992.0
M89 369 370 0 0 nfet L=0.5U W=4.0U
RLUMP216 371 372   300.0
RLUMP217 333 373  1741.5
RLUMP218 368 374   300.0
M90 372 373 374 0 nfet L=0.5U W=4.0U
RLUMP219 375 376   300.0
RLUMP220 179 377  1699.0
RLUMP221 371 378   300.0
M91 376 377 378 0 nfet L=0.5U W=4.0U
RLUMP222 379 380   300.0
RLUMP223 256 381  1729.0
RLUMP224 375 382   300.0
M92 380 381 382 0 nfet L=0.5U W=4.0U
RLUMP225 353 383   891.5
RLUMP226 100 384  1698.5
RLUMP227 379 385   300.0
M93 383 384 385 0 nfet L=0.5U W=4.0U
RLUMP228 333 386  1741.5
RLUMP229 343 387   906.0
RLUMP230 337 388  1654.5
M94 386 387 388 0 nfet L=0.5U W=4.0U
RLUMP231 343 389   906.0
RLUMP232 337 390  1654.5
RLUMP233 333 391  1741.5
M95 389 390 391 0 nfet L=0.5U W=4.0U
RLUMP234 335 392  1847.0
RLUMP235 343 393   906.0
M96 0 392 393 0 nfet L=0.5U W=4.0U
RLUMP236 345 394   982.5
RLUMP237 347 395   671.0
M97 0 394 395 0 nfet L=0.5U W=4.0U
RLUMP238 396 397   100.0
RLUMP239 337 398  1654.5
M98 397 398 0 0 nfet L=0.5U W=4.0U
RLUMP240 345 399   982.5
RLUMP241 335 400  1847.0
RLUMP242 396 401   100.0
M99 399 400 401 0 nfet L=0.5U W=4.0U
RLUMP243 136 402  2992.0
RLUMP244 403 404   300.0
M100 0 402 404 0 nfet L=0.5U W=4.0U
RLUMP245 136 405  2992.0
RLUMP246 361 406  1797.0
M101 1 405 406 1 pfet L=0.5U W=8.0U
RLUMP247 363 407   504.5
RLUMP248 219 408   699.5
M102 407 408 0 0 nfet L=0.5U W=4.0U
RLUMP249 136 409  2992.0
RLUMP250 410 411   300.0
M103 0 409 411 0 nfet L=0.5U W=4.0U
RLUMP251 136 412  2992.0
RLUMP252 302 413  1583.5
M104 1 412 413 1 pfet L=0.5U W=8.0U
RLUMP253 403 414   300.0
RLUMP254 415 416   125.0
RLUMP255 361 417  1797.0
M105 414 416 417 0 nfet L=0.5U W=4.0U
RLUMP256 410 418   300.0
RLUMP257 347 419   671.0
RLUMP258 302 420  1583.5
M106 418 419 420 0 nfet L=0.5U W=4.0U
RLUMP259 302 421  1583.5
RLUMP260 333 422  1741.5
RLUMP261 361 423  1797.0
M107 421 422 423 0 nfet L=0.5U W=4.0U
RLUMP262 333 424  1741.5
RLUMP263 425 426   934.0
M108 1 424 426 1 pfet L=0.5U W=8.0U
RLUMP264 333 427  1741.5
RLUMP265 428 429   100.0
M109 0 427 429 0 nfet L=0.5U W=4.0U
RLUMP266 425 430   934.0
```

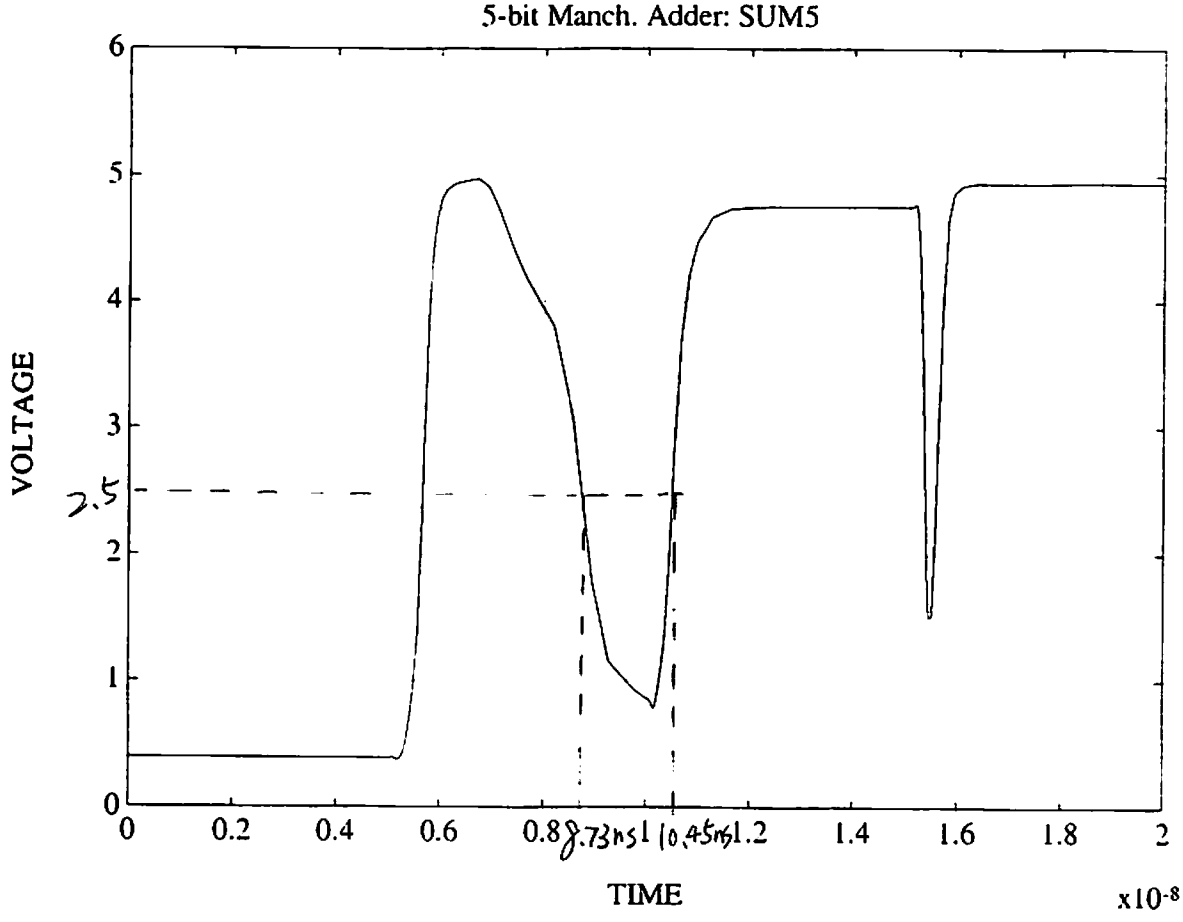
```
RLUMP267 361 431  1797.0
M110 430 431 1 1 pfet L=0.5U W=8.0U
RLUMP268 428 432   100.0
RLUMP269 361 433  1797.0
RLUMP270 425 434   934.0
M111 432 433 434 0 nfet L=0.5U W=4.0U
RLUMP271 435 436   935.0
RLUMP272 425 437   934.0
M112 436 437 1 1 pfet L=0.5U W=8.0U
RLUMP273 425 438   934.0
RLUMP274 439 440   274.5
M113 0 438 440 0 nfet L=0.5U W=4.0U
RLUMP275 333 441  1741.5
RLUMP276 442 443   320.0
M114 1 441 443 1 pfet L=0.5U W=8.0U
RLUMP277 439 444   274.5
RLUMP278 333 445  1741.5
RLUMP279 435 446   935.0
M115 444 445 446 0 nfet L=0.5U W=4.0U
RLUMP280 442 447   320.0
RLUMP281 361 448  1797.0
RLUMP282 435 449   935.0
M116 447 448 449 1 pfet L=0.5U W=8.0U
RLUMP283 435 450   935.0
RLUMP284 361 451  1797.0
RLUMP285 439 452   274.5
M117 450 451 452 0 nfet L=0.5U W=4.0U
C0 442 0 6F
** NODE: 442 = 6_61_235#
C1 439 0 6F
** NODE: 439 = 6_38_251#
C2 435 0 16F
** NODE: 435 = s1
C3 428 0 2F
** NODE: 428 = 6_38_167#
C4 425 0 10F
** NODE: 425 = 6_75_205#
** NODE: 410 = 6_60_89#
** NODE: 415 = c0
** NODE: 403 = 6_85_89#
C5 396 0 2F
** NODE: 396 = 6_26_39#
** NODE: 379 = 6_146_8#
** NODE: 375 = 6_138_8#
** NODE: 371 = 6_130_8#
** NODE: 368 = 6_122_8#
C6 363 0 9F
** NODE: 363 = c4
C7 361 0 19F
** NODE: 361 = c0b
C8 355 0 9F
** NODE: 355 = out
C9 353 0 9F
** NODE: 353 = 6_90_4#
C10 347 0 12F
** NODE: 347 = g1
C11 343 0 12F
** NODE: 343 = 6_77_43#
C12 333 0 15F
** NODE: 333 = p1
C13 345 0 11F
** NODE: 345 = 6_6_43#
C14 337 0 11F
** NODE: 337 = b1
C15 335 0 9F
** NODE: 335 = a1
C16 322 0 6F
** NODE: 322 = cell_0/6_61_235#
C17 319 0 6F
** NODE: 319 = cell_0/6_40_251#
C18 315 0 16F
** NODE: 315 = s2
C19 308 0 2F
** NODE: 308 = cell_0/6_40_167#
C20 305 0 10F
** NODE: 305 = cell_0/6_75_205#
C21 302 0 14F
** NODE: 302 = c1
** NODE: 293 = cell_0/6_57_89#
C22 286 0 2F
** NODE: 286 = cell_0/6_26_39#
C23 270 0 11F
** NODE: 270 = cell_0/g
C24 266 0 12F
** NODE: 266 = cell_0/6_77_43#
C25 256 0 14F
** NODE: 256 = cell_0/p
C26 268 0 11F
** NODE: 268 = cell_0/6_6_43#
C27 260 0 11F
** NODE: 260 = b2
C28 258 0 9F
** NODE: 258 = a2
C29 245 0 6F
** NODE: 245 = cell_1[1]/6_61_235#
C30 242 0 6F
** NODE: 242 = cell_1[1]/6_40_251#
C31 238 0 16F
** NODE: 238 = s4
C32 231 0 2F
** NODE: 231 = cell_1[1]/6_40_167#
C33 228 0 10F
** NODE: 228 = cell_1[1]/6_75_205#
** NODE: 0 = GND!
C34 141 0 15F
** NODE: 141 = c3
C35 219 0 12F
** NODE: 219 = c4b
** NODE: 216 = cell_1[1]/6_57_89#
C36 1 0 252F
** NODE: 1 = Vdd!
C37 136 0 13F
** NODE: 136 = clk
C38 209 0 2F
** NODE: 209 = cell_1[1]/6_26_39#
C39 193 0 11F
** NODE: 193 = cell_1[1]/g
C40 189 0 12F
** NODE: 189 = cell_1[1]/6_77_43#
C41 179 0 19F
** NODE: 179 = cell_1[1]/p
C42 191 0 11F
** NODE: 191 = cell_1[1]/6_6_43#
C43 183 0 11F
** NODE: 183 = b4
C44 181 0 9F
** NODE: 181 = a4
C45 166 0 6F
```

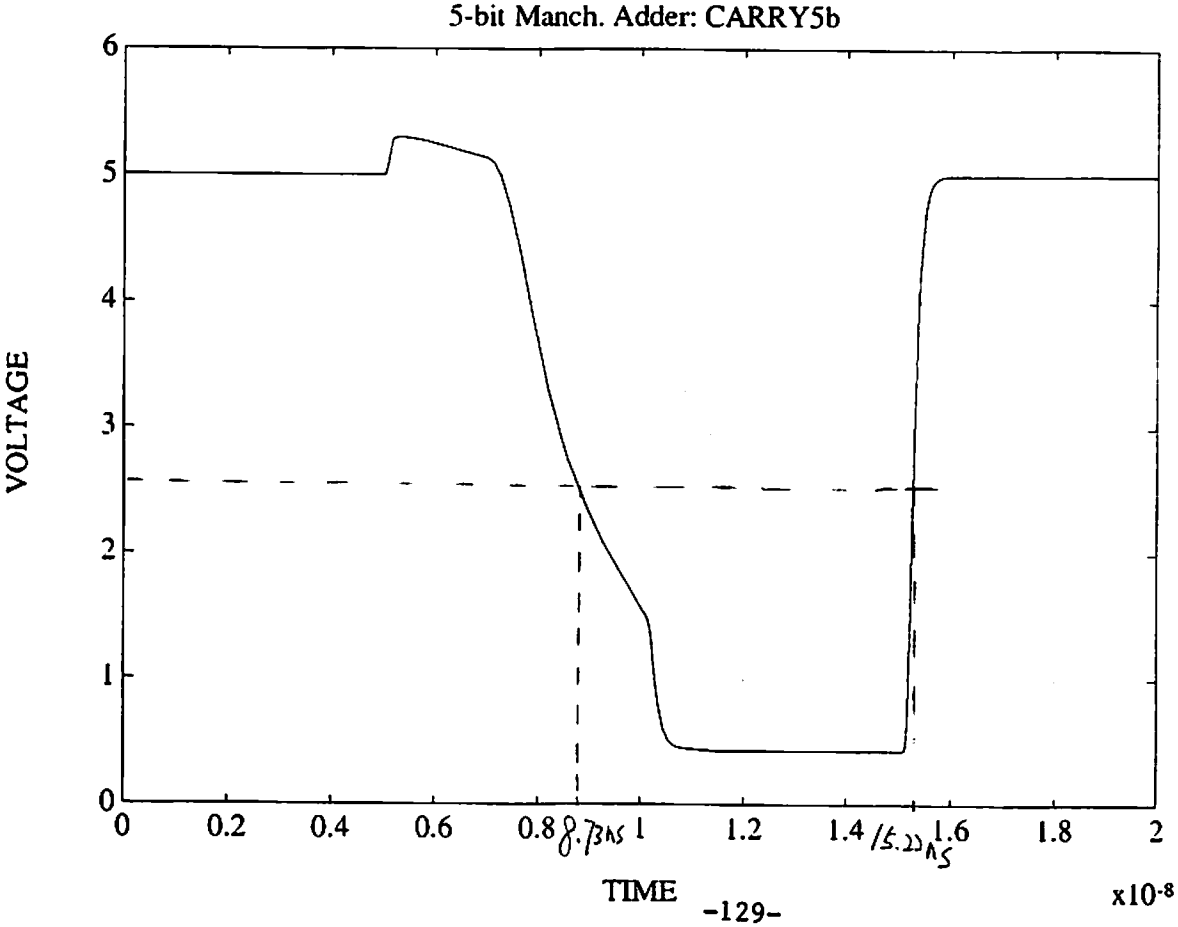
```
** NODE: 168 = cell_1[0]/6_61_235#
C46 165 0 6F
** NODE: 165 = cell_1[0]/6_40_251#
C47 161 0 16F
** NODE: 161 = s3
C48 154 0 2F
** NODE: 154 = cell_1[0]/6_40_167#
C49 151 0 10F
** NODE: 151 = cell_1[0]/6_75_205#
C50 148 0 15F
** NODE: 148 = c2
** NODE: 138 = cell_1[0]/6_57_89#
C51 130 0 2F
** NODE: 130 = cell_1[0]/6_26_39#
C52 114 0 11F
** NODE: 114 = cell_1[0]/g
C53 110 0 12F
** NODE: 110 = cell_1[0]/6_77_43#
C54 100 0 17F
** NODE: 100 = cell_1[0]/p
C55 112 0 11F
** NODE: 112 = cell_1[0]/6_6_43#
C56 104 0 11F
** NODE: 104 = b3
C57 102 0 9F
** NODE: 102 = a3

.MODEL nfet NMOS LEVEL=3 PHI=0.600000 TOX=1.1E-08 XJ=0.200000U TPG=1
+ VTO=0.8186 DELTA=1.7570E+00 LD=0 KP=9.1547E-05
+ UO=596.5 THETA=1.0850E-01 GAMMA=0.5266 NSUB=1.9680E+16
+ NFS=5.5000E+12 VMAX=1.9420E+05 ETA=6.6540E-02 KAPPA=1.1210E-01
+ CGDO=2.6106E-10 CGSO=2.6106E-10 CGBO=6.3402E-10
+ CJ=3.1146E-04 MJ=1.0667 CJSW=4.3777E-10 MJSW=0.154200 PB=0.800000
* Weff= Wdrawn - Delta_W
* The suggested Delta_W is 1.9970E-07

.MODEL pfet PMOS LEVEL=3 PHI=0.600000 TOX=1.1E-08 XJ=0.200000U TPG=-1
+ VTO=-0.9456 DELTA=1.5520E+00 LD=0 KP=3.1646E-05
+ UO=206.2 THETA=1.6900E-01 GAMMA=0.4619 NSUB=1.5140E+16
+ NFS=4.9990E+12 VMAX=4.4410E+05 ETA=1.6350E-01 KAPPA=1.0000E+01
+ CGDO=2.6981E-11 CGSO=2.6981E-11 CGBO=8.6508E-10
+ CJ=4.7864E-04 MJ=0.4973 CJSW=1.4771E-10 MJSW=0.190593 PB=0.850000
* Weff = Wdrawn - Delta_W
* The suggested Delta_W is 3.1280E-07

CC4 363 0 18.84F
CC44 363 0 28.26F

Vdd 1 0 dc 5v
Va1 335 0 dc 5v pulse(0 5 5ns 100ps 100ps 20ns 50ns)
Va2 258 0 dc 5v pulse(0 5 5ns 100ps 100ps 20ns 50ns)
Va3 102 0 dc 5v pulse(0 5 5ns 100ps 100ps 20ns 50ns)
Va4 181 0 dc 5v pulse(0 5 5ns 100ps 100ps 20ns 50ns)
Vb1 337 0 dc 5v pulse(0 5 5ns 100ps 100ps 20ns 50ns)
Vb2 260 0 dc 0v
Vb3 104 0 dc 0v
Vb4 183 0 dc 0v
Vc0 415 0 dc 5v pulse(0 5 5ns 100ps 100ps 10ns 20ns)
Vclk 136 0 dc 5v pulse(0 5 5ns 100ps 100ps 5ns 10ns)
.tran 0.1ns 20ns
.END
```

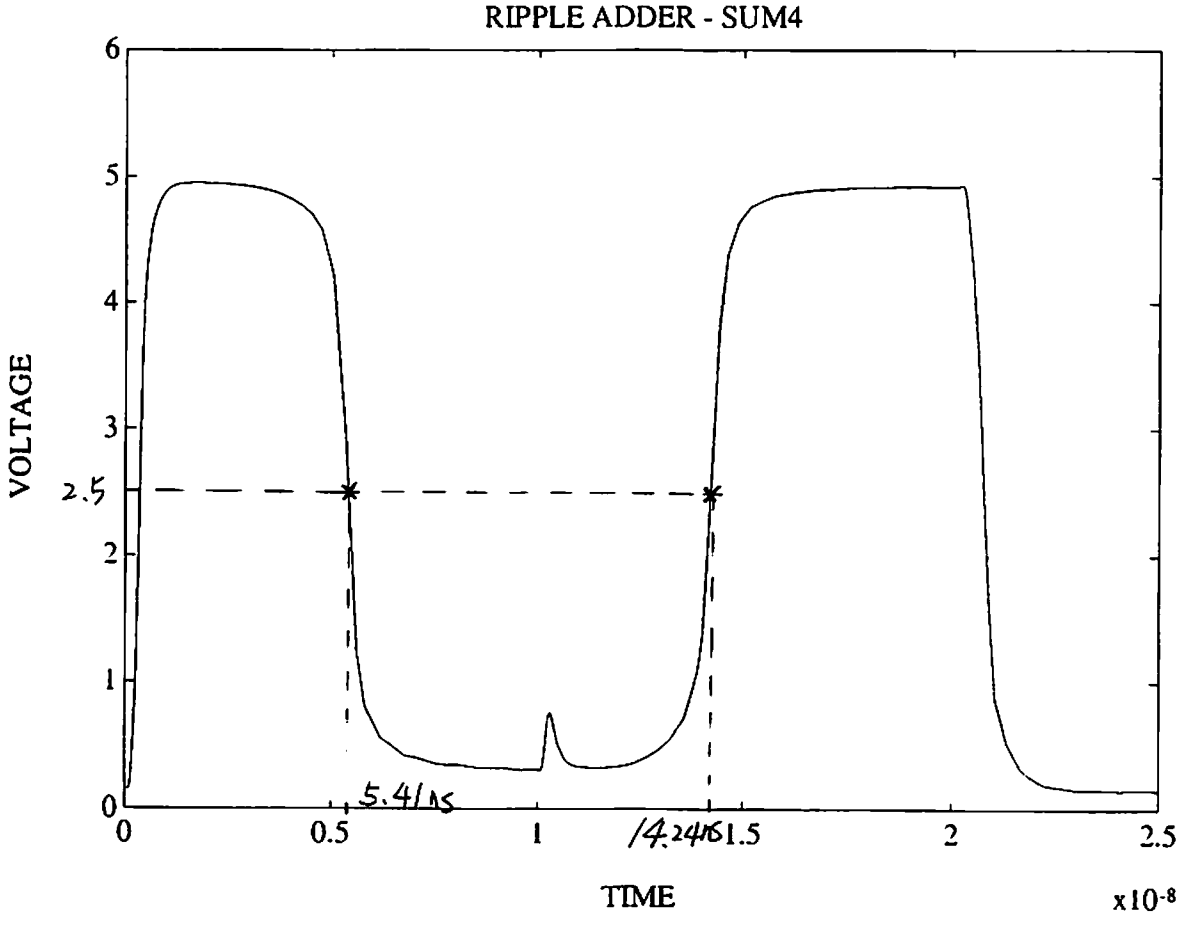
→ capacitance loading

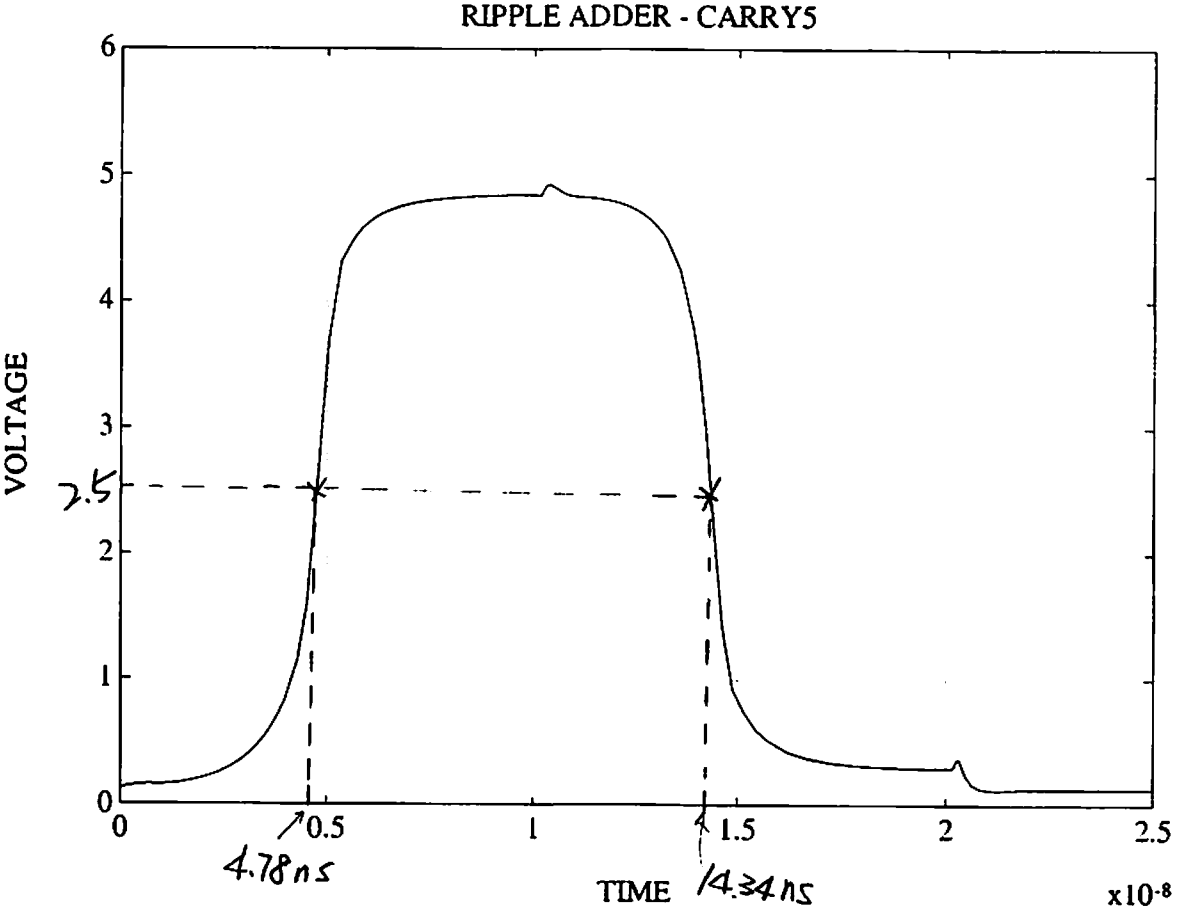

4-bit Manch. Adder - SUM4
VOLTAGE
TIME
x10-8
0
1
2
3
4
5
6
0.2
0.4
0.6
0.8
1.2
1.4
1.6
1.8


4-bit Manch. Adder - CARRY4
VOLTAGE
TIME
x10-8
0
1
2
3
4
5
6
0.2
0.4
0.6
0.8
1.2
1.4
1.6
1.8


5-bit Manch. Adder: SUM5
VOLTAGE
TIME
x10-8
0
1
2
3
4
5
6
0.2
0.4
0.6
0.8
1.2
1.4
1.6
1.8
2
2.5
8.73ns
10.45ns


5-bit Manch. Adder: CARRY5b
VOLTAGE
TIME
x10-8
0
1
2
3
4
5
6
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
8.73ns
15.22ns

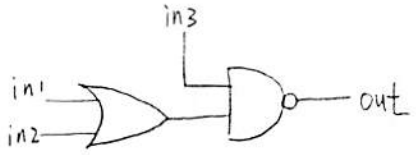

RIPPLE ADDER - SUM4
VOLTAGE
TIME
x10-8
2.5
5.41ns
14.24ns

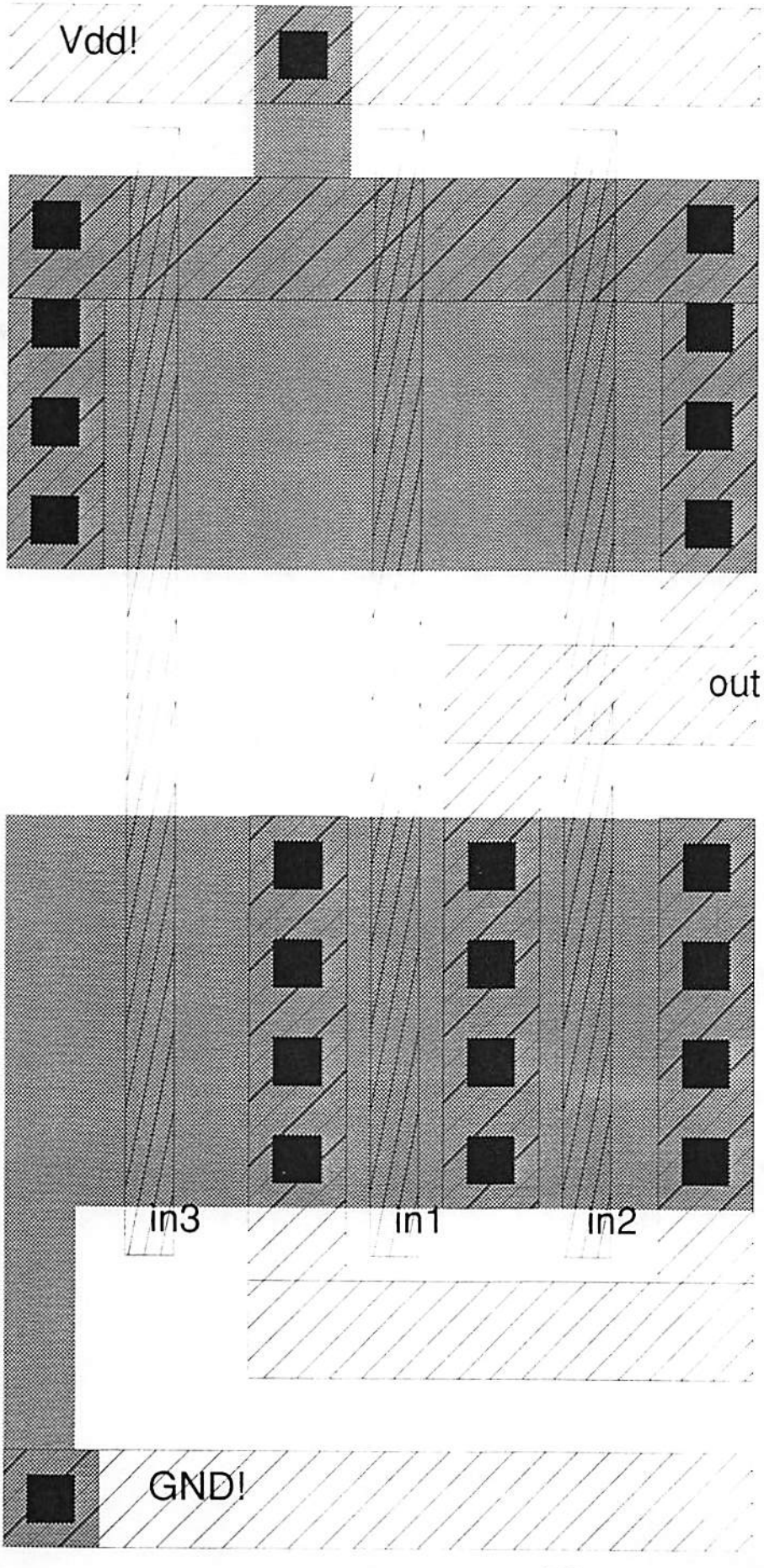

RIPPLE ADDER - CARRY5
VOLTAGE
TIME
x10-8
2.5
4.78ns
14.34ns

APPENDIX III:

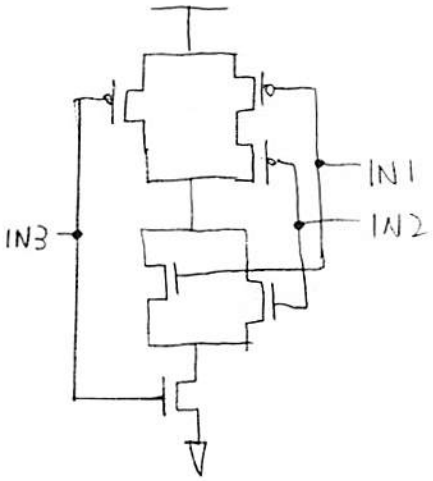
LAYOUT AND FLOORPLAN OF EACH BLOCK

ONA (OR - NAND)



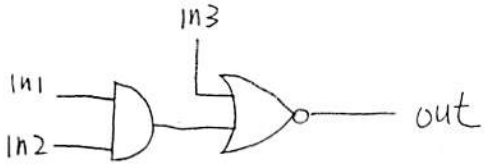


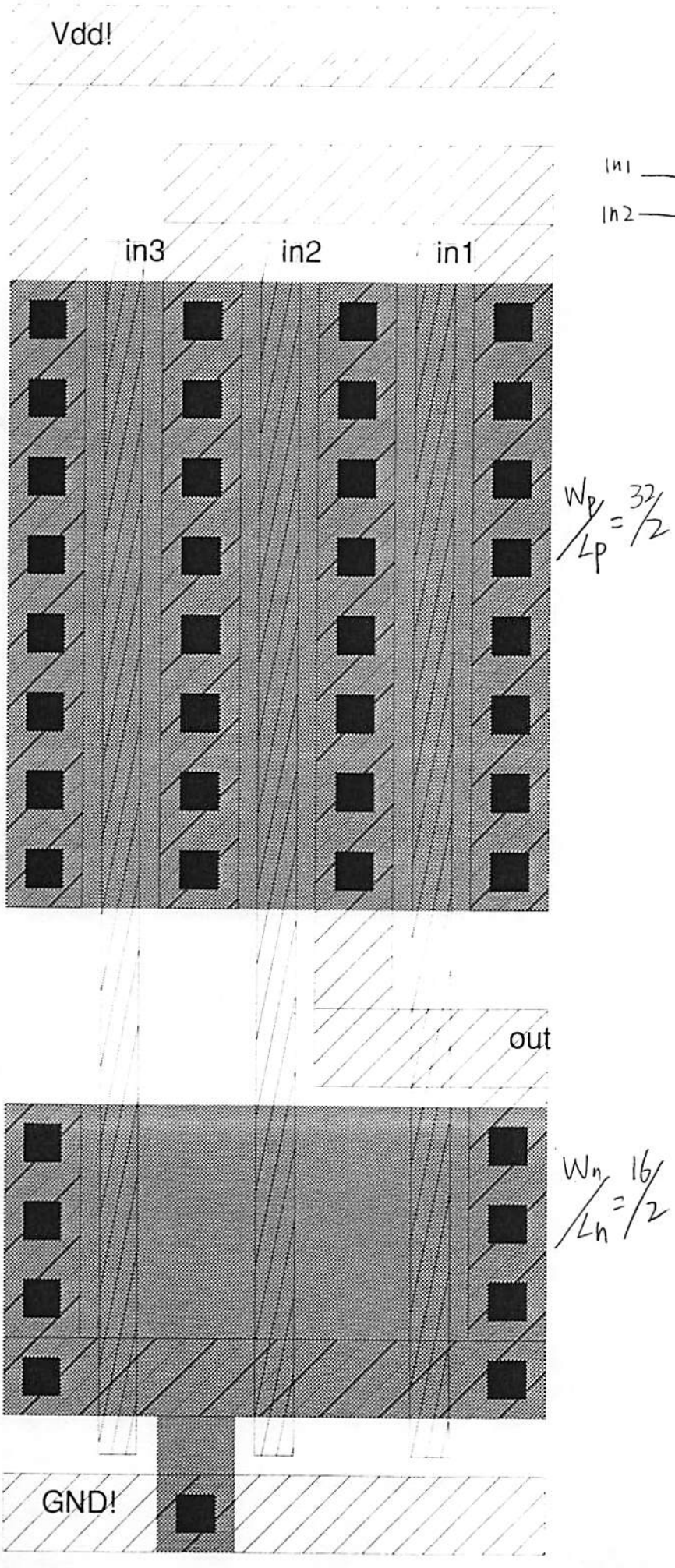
$16/2\,(W_P/L_P)$



$16/2\,(W_n/L_n)$

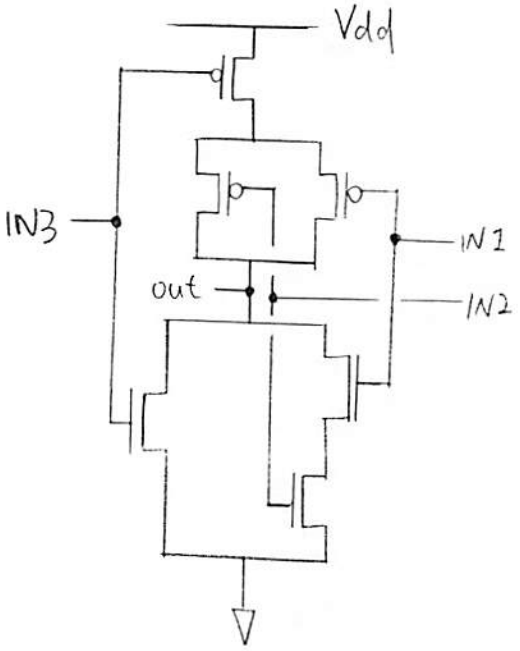
ANO (AND-NOR)



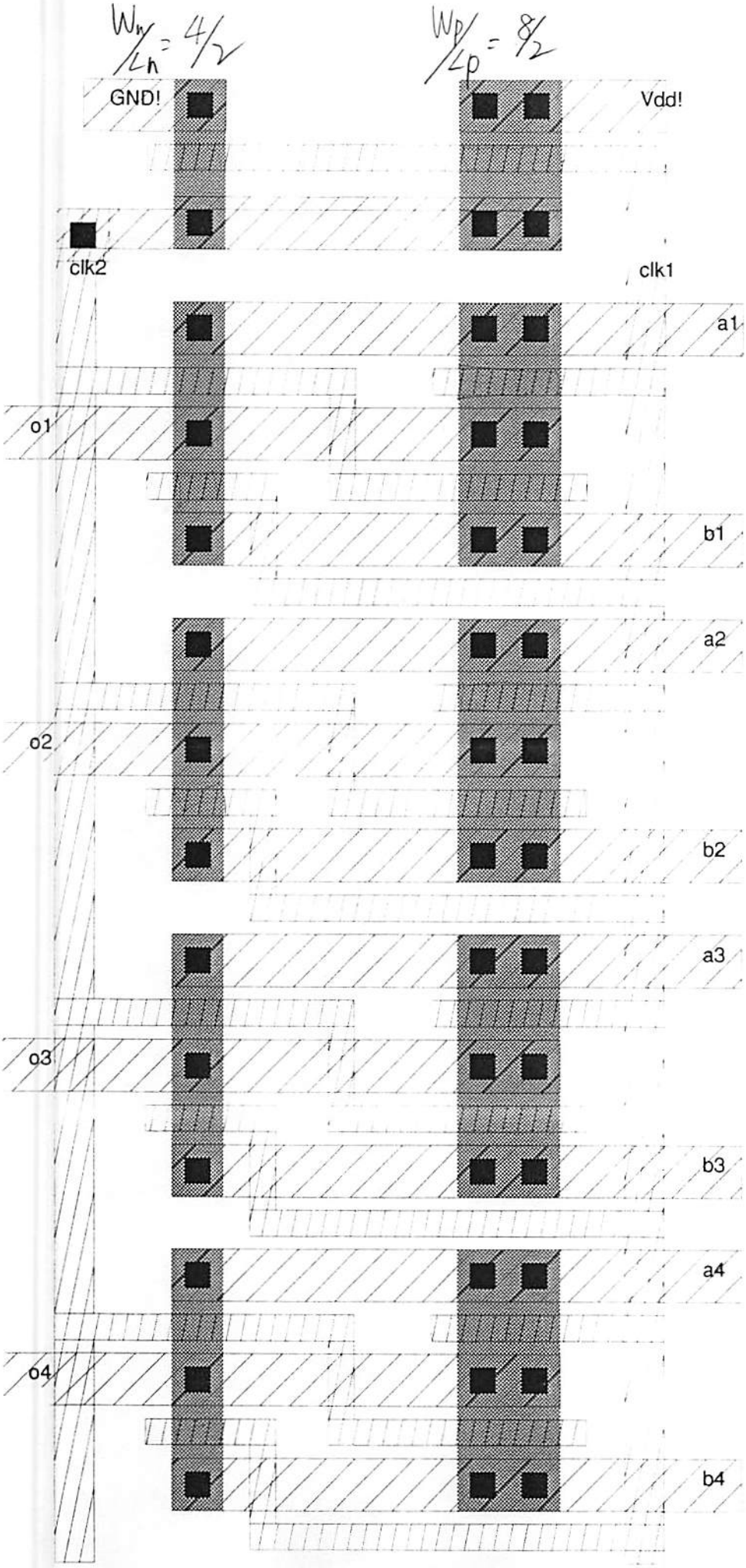


$W_p / L_p = 32/2$

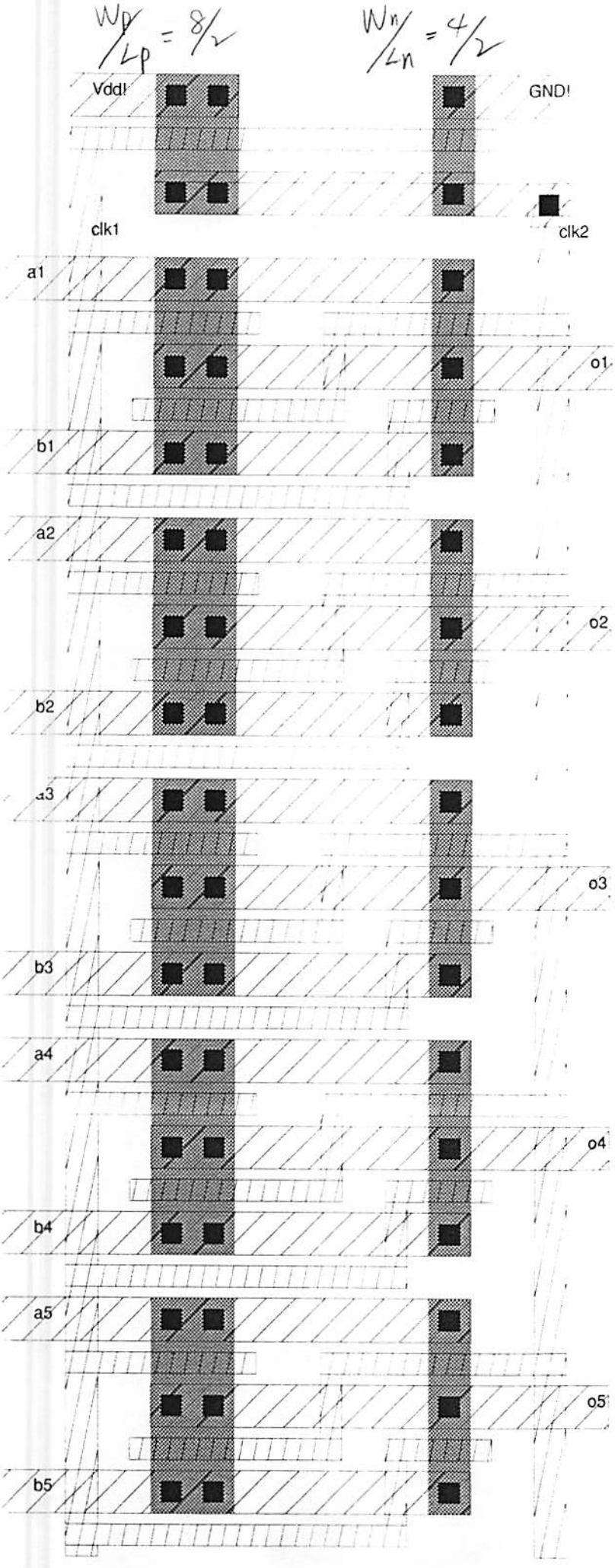
$W_n / L_n = 16/2$



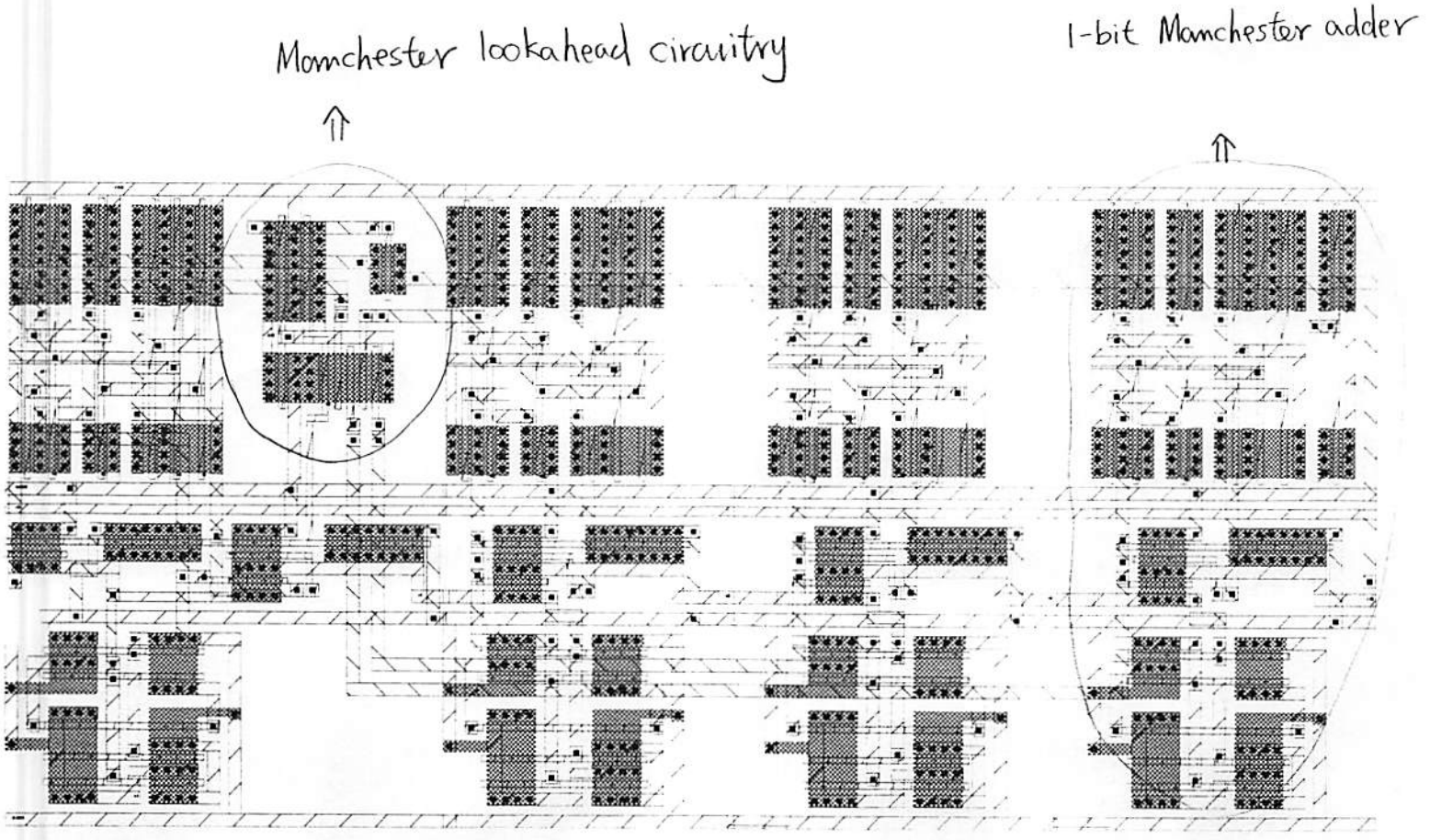
8-input multiplexer:



10-input multiplexer

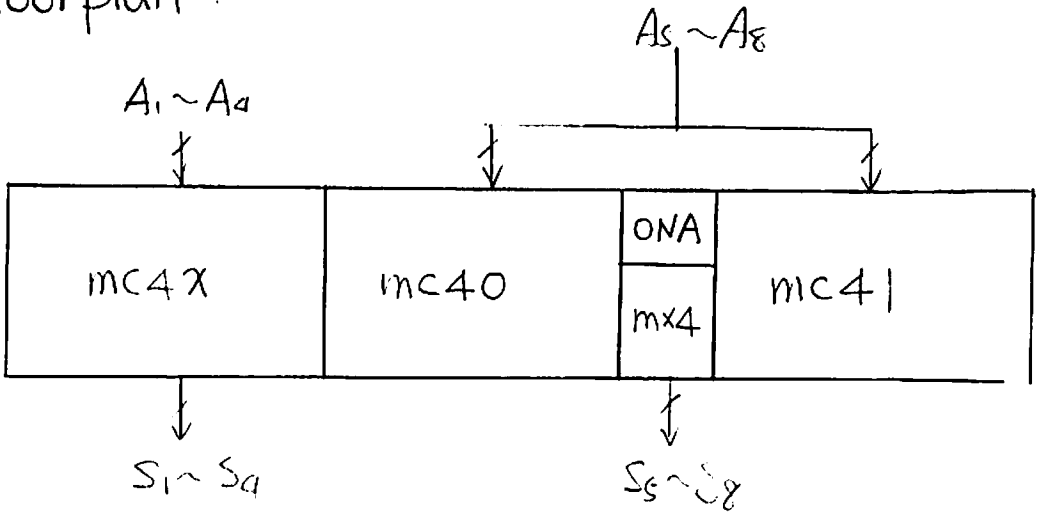


4-bit Manchester adder with Manchester lookahead circuitry: (mc4x)

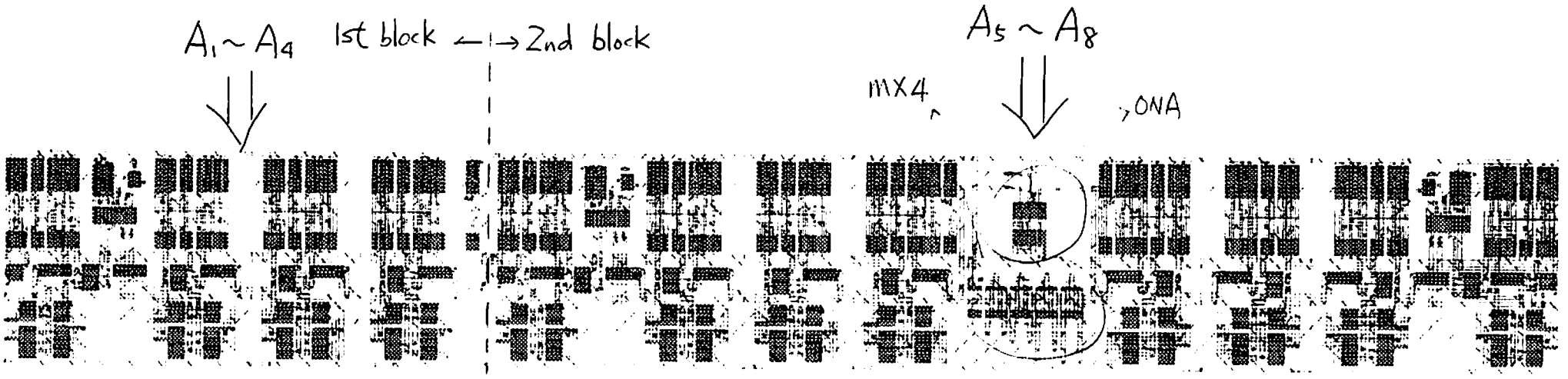


1st and 2nd blocks : ① Floorplan :

(4-bit) (4-bit)

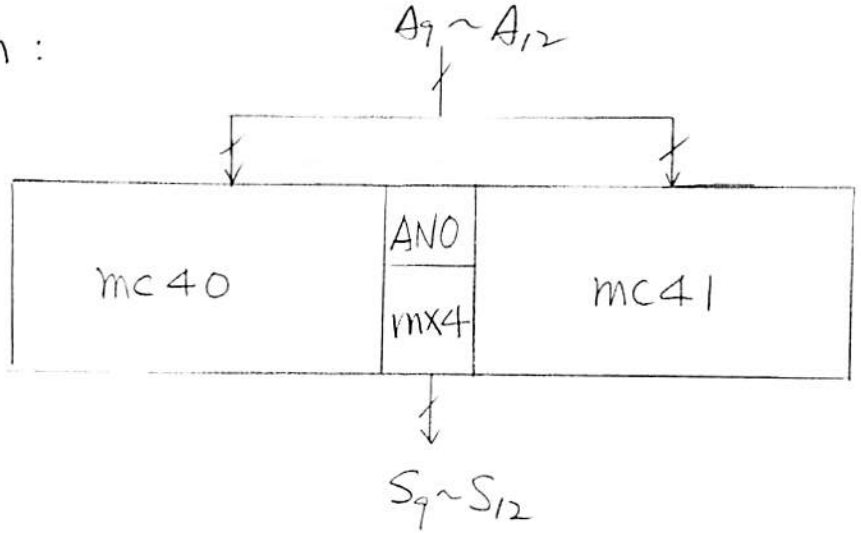


② layout :

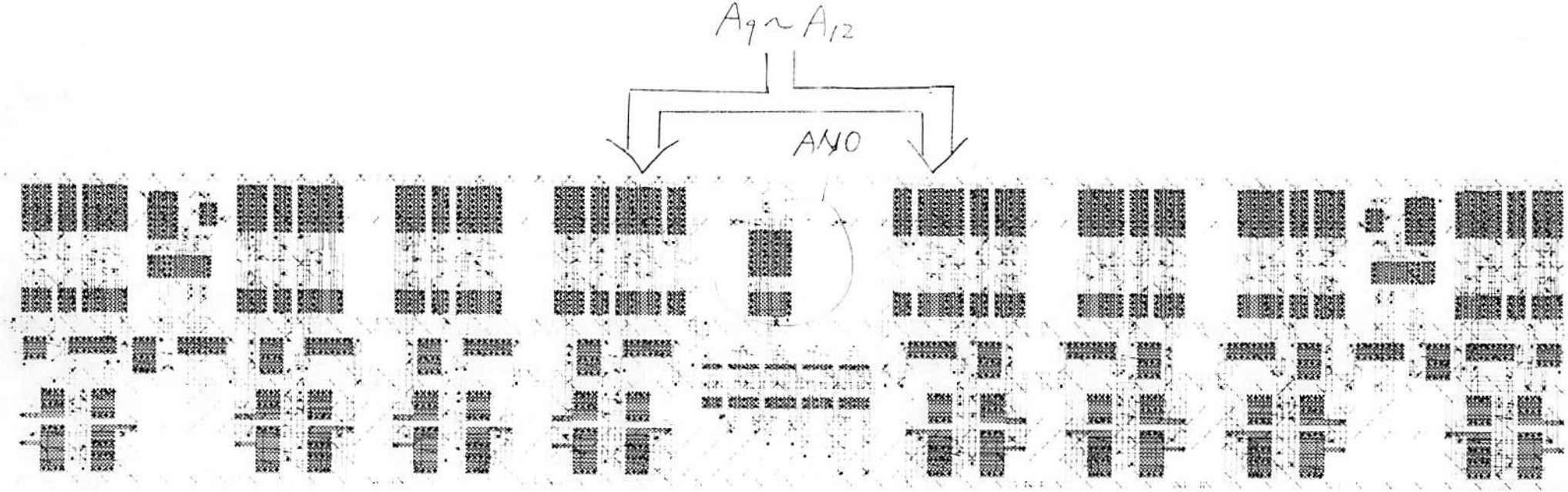


3rd block : (4-bit)

① Floorplan :

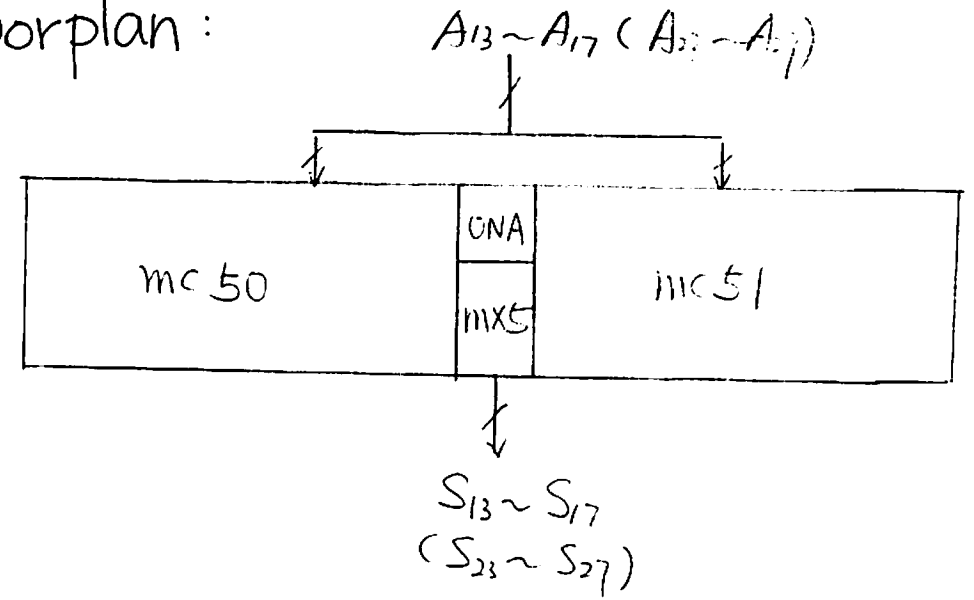


② Layout :

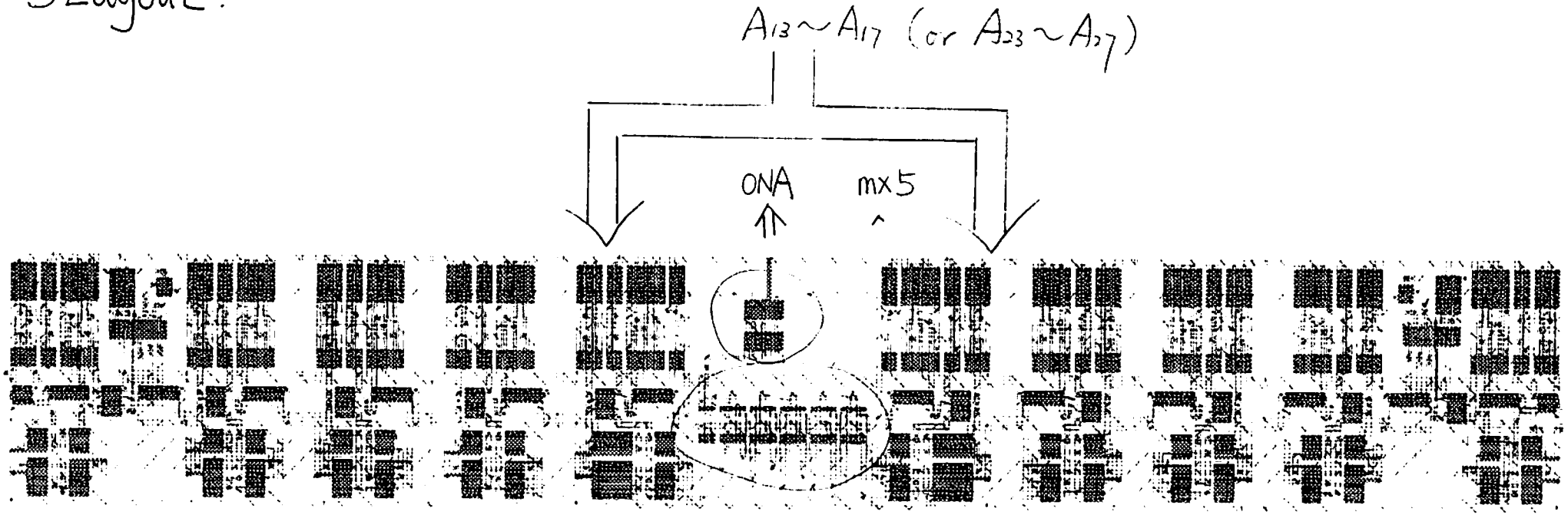


4tR or 6tR blocks :
(5-bit) (5-bit)

① Floorplan :

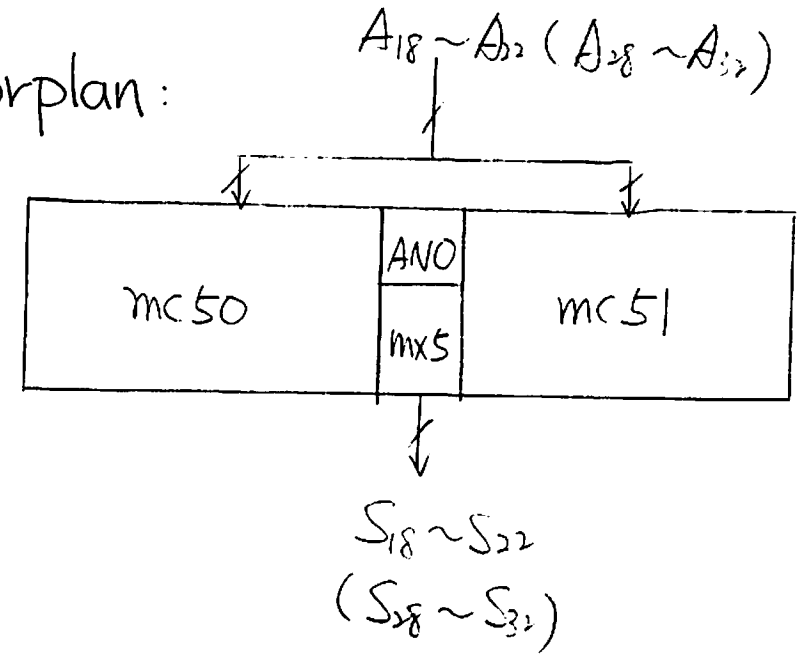


② Layout :

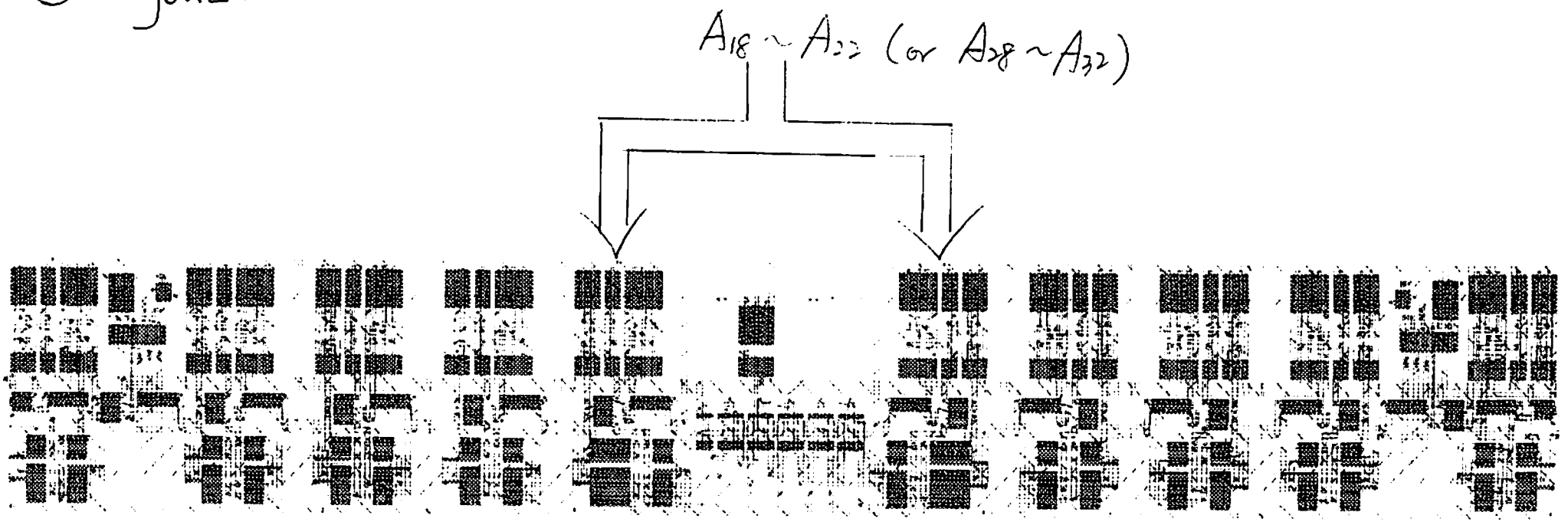


5th or 7th blocks :
(5-bit) (5-bit)

① Floorplan :



② Layout :



EE 577 TERM PROJECT:

HIGH-SPEED MULTIPLIER DESIGN

PROFESSOR: DR. BING SHEU

NAME : CHIH-WEI TSAI

ID # : 615-38-6758

ELECTRICAL ENGINEERING DEPARTMENT

UNIVERSITY OF SOUTHERN CALIFORNIA

DEC. 1991

Abstract:: A 4.5 ns CMOS 16*16-b multiplier based on 0.5um technology with a supply voltage of 5V is introduced in this report. The whole design applied complementary pass-transistor logic (CPL) to achieve high speed multiplication. The magic of the CPL logic is due to its lower input capacitance and high functionality, which paved the way to achieve high speed operations.

CONTENTS

I. INTRODUCTION

The speed of CMOS devices has been improved recently. Partially because the submicron technology is more mature, the new circuit technique also contributes to its success. This report implemented the idea in [1], which applied CPL logic to achieve high speed multiplication, and accomplished some analysis and performance evaluation of the CPL multiplier. First, the concept of the CPL and the architecture are described in Section II, and then the implementation of the CPL multiplier is described in Section III. Performance evaluation is briefly depicted in Section IV, and finally, the conclusion and discussion are shown in Section V.

II. CPL Logic Concepts and Multiplier Architecture

Using nMOS devices to implement the logic function and inverters to shift the logic threshold voltage and to strengthen the signal to drive the capacitive load are the major idea of CPL logic. The basic logic circuit of CPL is shown in Fig 1. Since pMOS is eliminated from the logic construction, the input capacitance is about the half of conventional CMOS circuits, which lead to higher speed and lower power dissipations. Moreover, arbitrary Boolean functions can be built from the CPL by using small amount of nMOS devices, which effectively reduced the transistor number, hence reduce the total area.

The architecture of my design of CPL multiplier consists of three major components: partial product generator (PPG), Wallace tree adder array and one 32-bit carry propagate adder, as shown in Fig 2. Partial product generator is basically constructed by 256 two-input CPL AND gates. A lot of carry save adders (CSA) formed Wallace adder array, which consists six stages and finally will generated two 32-bit sum signal and carry signal. The Wallace adder array has been discussed in [3] and is depicted in Fig 3. The architecture of CPA adopted the Manchester adder [2] mixed with carry select adder [2], which idea is come from my teammate, Cheng-Ju Hsieh. Fig 4 describes the block diagram of the 32-bit adder.

III. Design and Implementation

The following described simulations are performed by SPICE3 and the measurement of the delay was based on 50%-50% criterion.

Partial Product Generator(PPG):

A simple CPL AND gate, as shown is Fig 1, is the essential component of PPG. Actually, PPG contains 256 CPL AND gates and all of them will function parallelly. Therefore, the delay of PPG is the delay of one AND gate. The following are the key points in PPG design:

(1) The width of one AND gate should be close to one third of the width of one CSA. This is due to the input signal number of one CSA is three times of the output signal number of one AND gate, that is one CSA needs three AND gates to support it.

(2) The delay is dependent on the size of the devices. The systematic method to decide the size of the devices will be introduced in the next paragraph.

I connect one AND gate and one CSA to do SPICE simulation. The CSA is used to simulate the loading effect. Fig 5 shows the delay of one CPL AND gate and its layout.

Wallace Tree Adder Array:

To improve the speed of this component, we should focus on building one CSA with minimum delay. Fig 6 contains the circuits and the layout of one CSA cell. The delay of one CSA cell depends on the following four factors: the size of the nMOS device, the inputs, the ratio of pMOS to nMOS in inverter, and the size of the inverter. The goal of the CSA design is to achieve as small delay as possible under the longest latency input. The systematic method to achieve this goal by attacking those four factors is shown in Fig 7 and is briefly described in the following:

(1) The draft layout of one CSA will be created. The next step is to connect two CSA to do spice simulation. The second CSA is used to simulate the loading effect.

(2) To find the optimum size of the nMOS device, I changed the size of the nMOS size in the spice file to see which ones will lead to minimum delay. The results are shown in Table I, and its corresponding graph is in Fig 8.

(3) To find the inputs which lead to longest delay, I summarized eight kinds of inputs and try to find out which one will generate the longest delay. Those inputs will change the two pairs of output signals, sum, sum_bar, carry, and carry_bar simultaneously, and therefore, I assume one of them will be the longest-delay input. I do the simulation for these inputs and finally find the 'F' input is the longest-delay input. The waveforms of these inputs are shown in Fig 10.The results are in Table II.

(4) Once we know the size of the nMOS devices and the longest-delay input, we continue to find the optimum ratio of the inverter. I varied the ratio of the inverter and performed SPICE simulation. The results are in Table III. Its corresponding graph is in Fig 9.

(5) Varying the size of the pMOS and nMOS devices in the inverter while keeping the ratio fixed, which is the optimum result in step(4), to find the appropriate size for inverters. Table IV records the results and Fig 11 is its corresponding graph.

Actually, the delay may not linearly dependent on these four factors, so we can't solve them individually. We should iterate the steps as many times as possible. For my design, I iterated once and the delay is approximately 0.2652ns. The summation of the result is in Table V.

Carry Propagate Adder:

The design of my adder is based on the Manchester adder mixed with carry select adder, which has been fully developed by my teammate. These two different adders have been discussed in [2]. I tried to modify it to accommodate the advantages of CPL logic. The changes are shown in Fig 12. I used multiplexers to generate the P(the propagate signal) and the G(the generate signal). The reason is that the previous Wallace tree adder array will generate complimentary signals, hence we can save time by eliminating the inverters. As a result, the speed is enhanced, from 1.31ns to 0.95ns for 4-bit Manchester adder. The spice result is in Fig 13. Finally, the delay calculation of the adder is shown in Fig 14.

IV. Performance Evaluation

The delays and the areas of the critical components of the CPL multiplier are summarized in Table VI, which is based on the following assumptions. The floor planning is shown in Fig 15.

(1) Assume the delay for the Wallace tree adder is six times the delay on one CSA under loading effect. I've connected six CSAs to do simulation, but the longest delay input is difficult to explore; therefore, I select one set of input and get the simulation result, as shown in Fig 16.

(2) Assume the total delay can be computed by adding the delay of PPG, Wallace adder array and CPA.

The total area is estimated by adding the layout area of the individual components and appropriate routing area. The computation of routing area is briefly shown in Table V.

V. Conclusion and Discussion

The 4.5ns CPL multiplier is briefly described in this report. The layouts for individual parts are attached. Because we can't connect the whole circuit to do SPICE simulations, we need a lot of assumptions to help us come out the result. The advantages for applying CPL to design the multiplier are, high speed, less device count and less power dissipation. However, the routing area will increase drastically due to the complementary inputs and outputs of the CPL logic.

REPERENCES:

[1] K.YANO et.al., *"A 3.8-ns CMOS 16×16Multiplier Using Complementary Pass-Transistor Logic"*, IEEE Journal Solid-State Circuit, vol. 25, No. 2, pp. 388-394, April, 1987.

[2] N. Weste and K. Eshraghian *"Principles of CMOS VLSI Design. A System Perspective"*, MA:Addison-Wesley, 1985.

[3] K. Wang and F.A.Briggs *"Computer Architecture and Parallel Processing"*, McGraw-Hill, 1984.

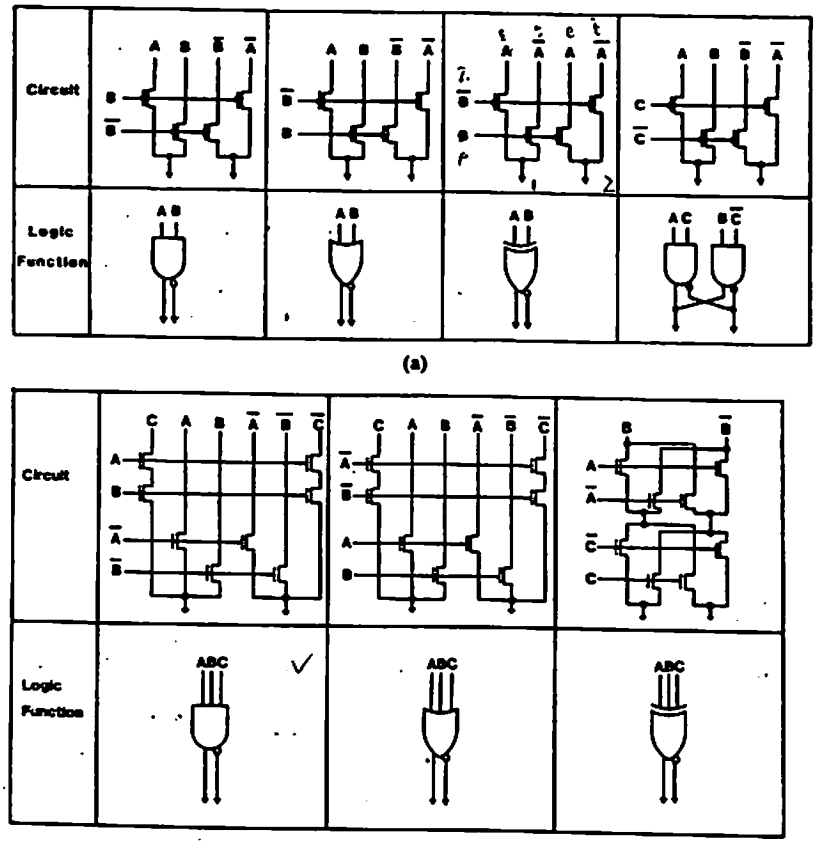


Fig 1. Basic CPL logic circuit.

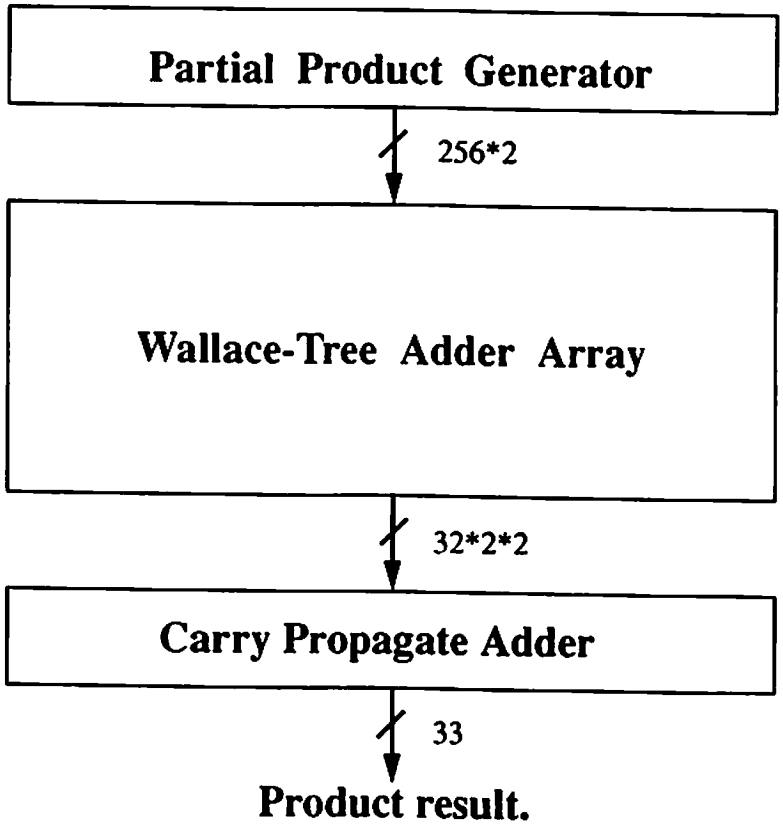


Fig 2. Block diagram of the 16*16-b multiplier.

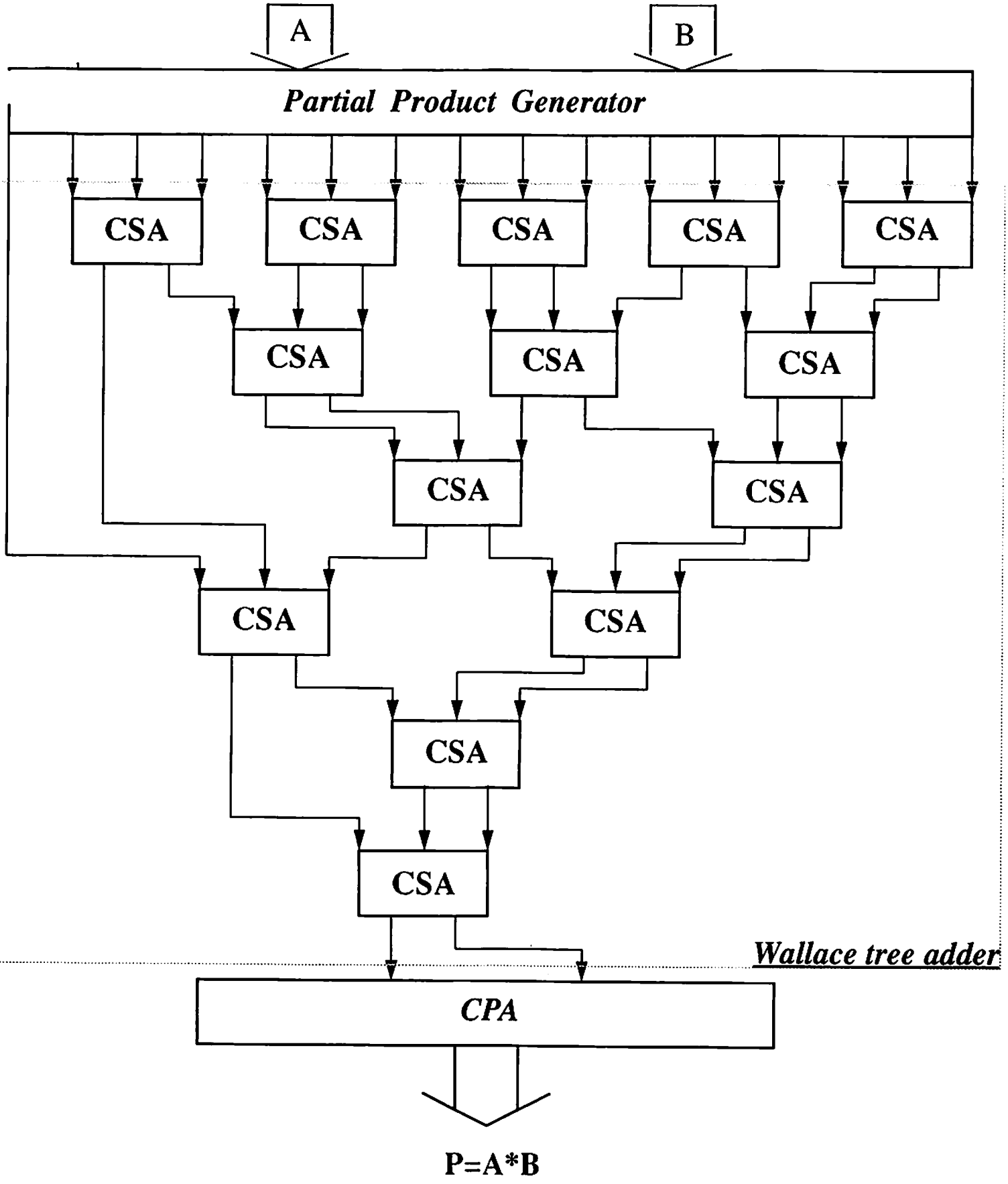


Fig 3. The connection of Wallace Tree Adder Array.

2. Block sizes: 4 4 4 5 5 5 5

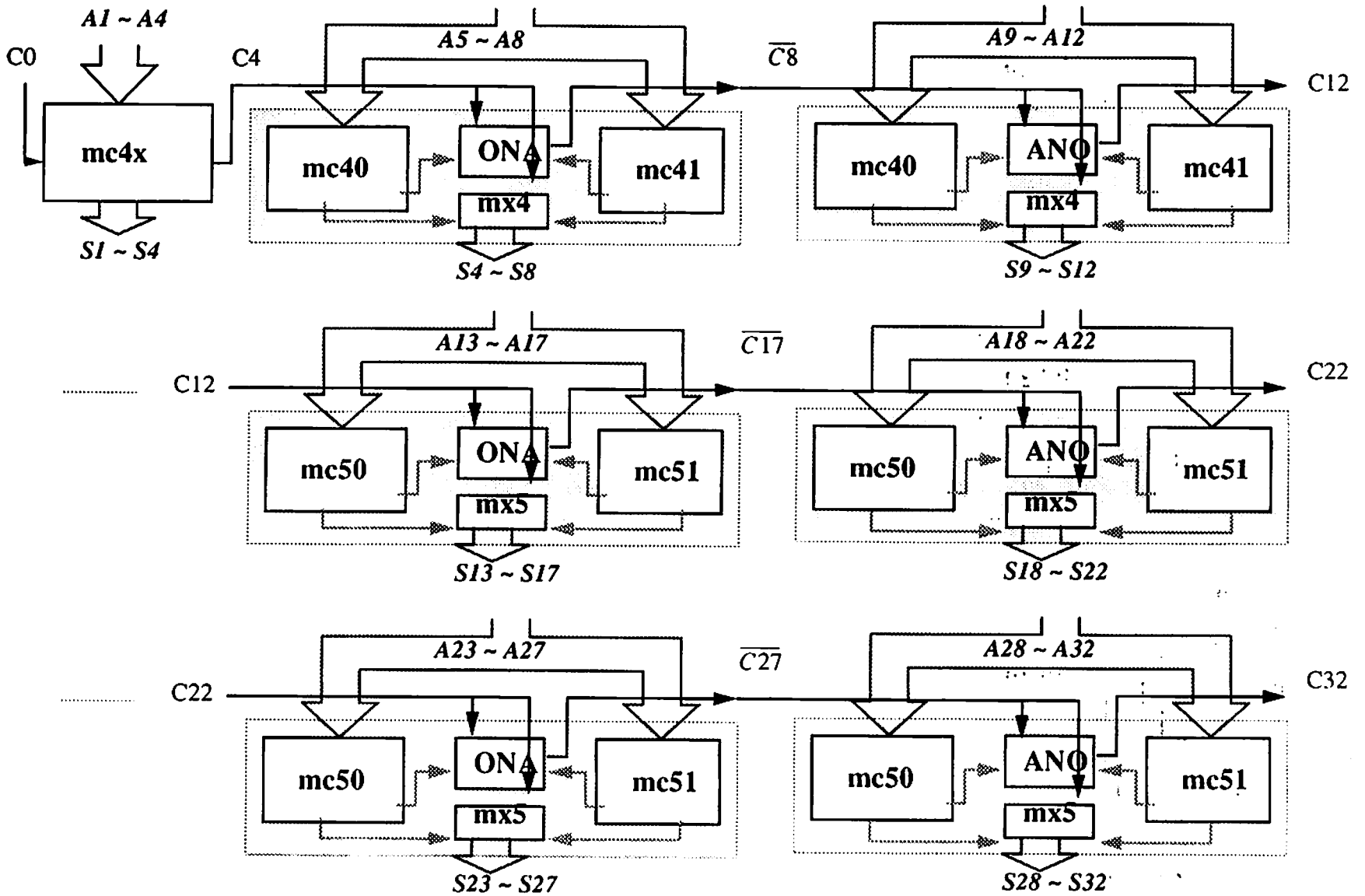
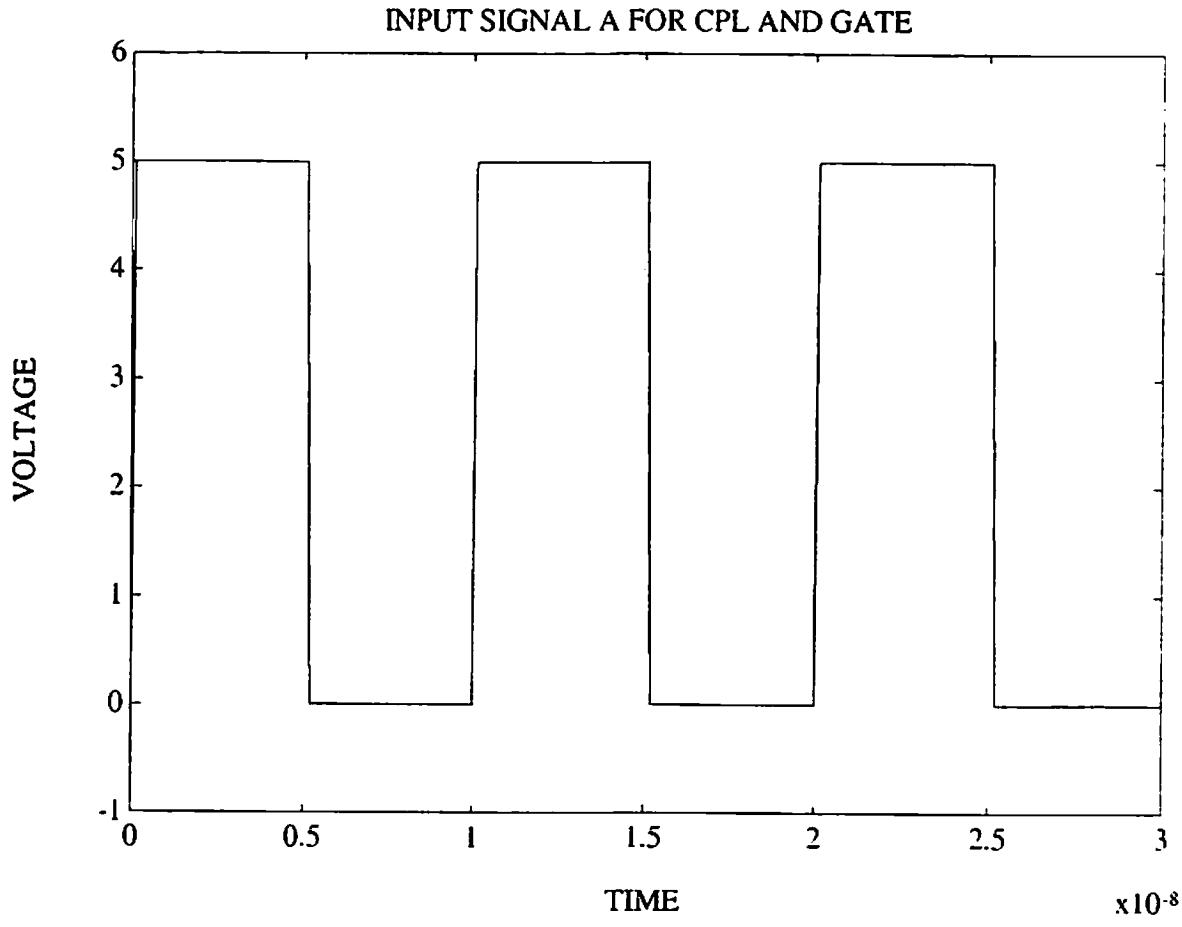


Fig 4. The block Diagram of a 32-bit adder



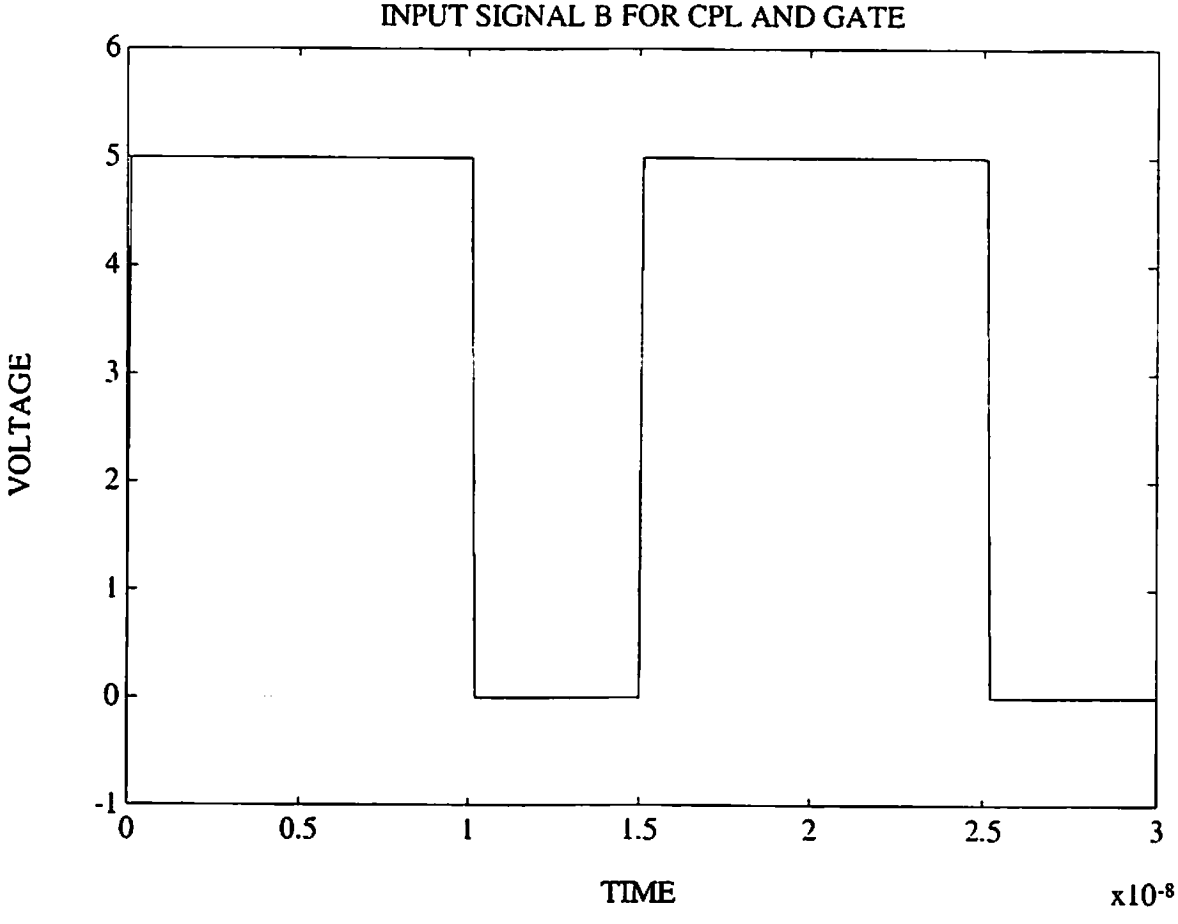
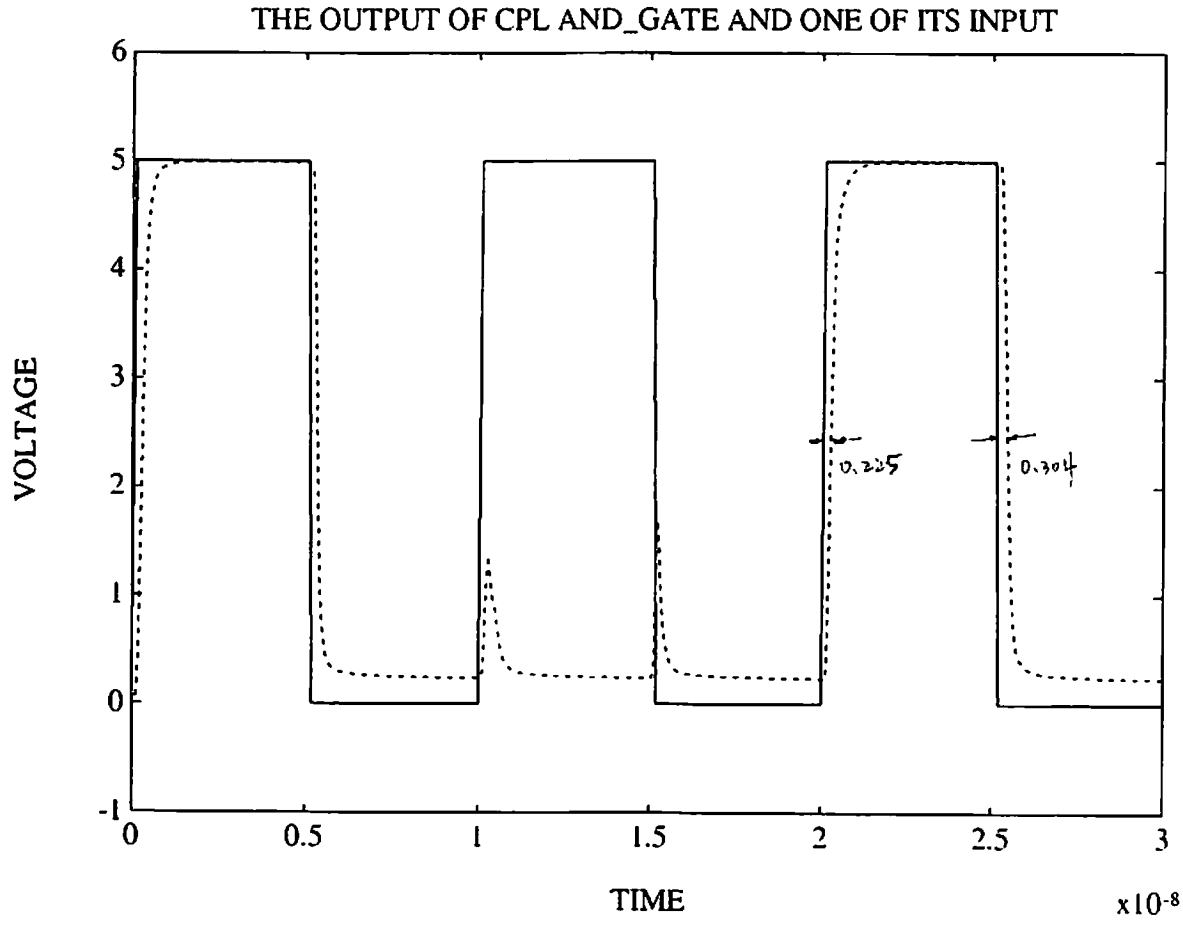
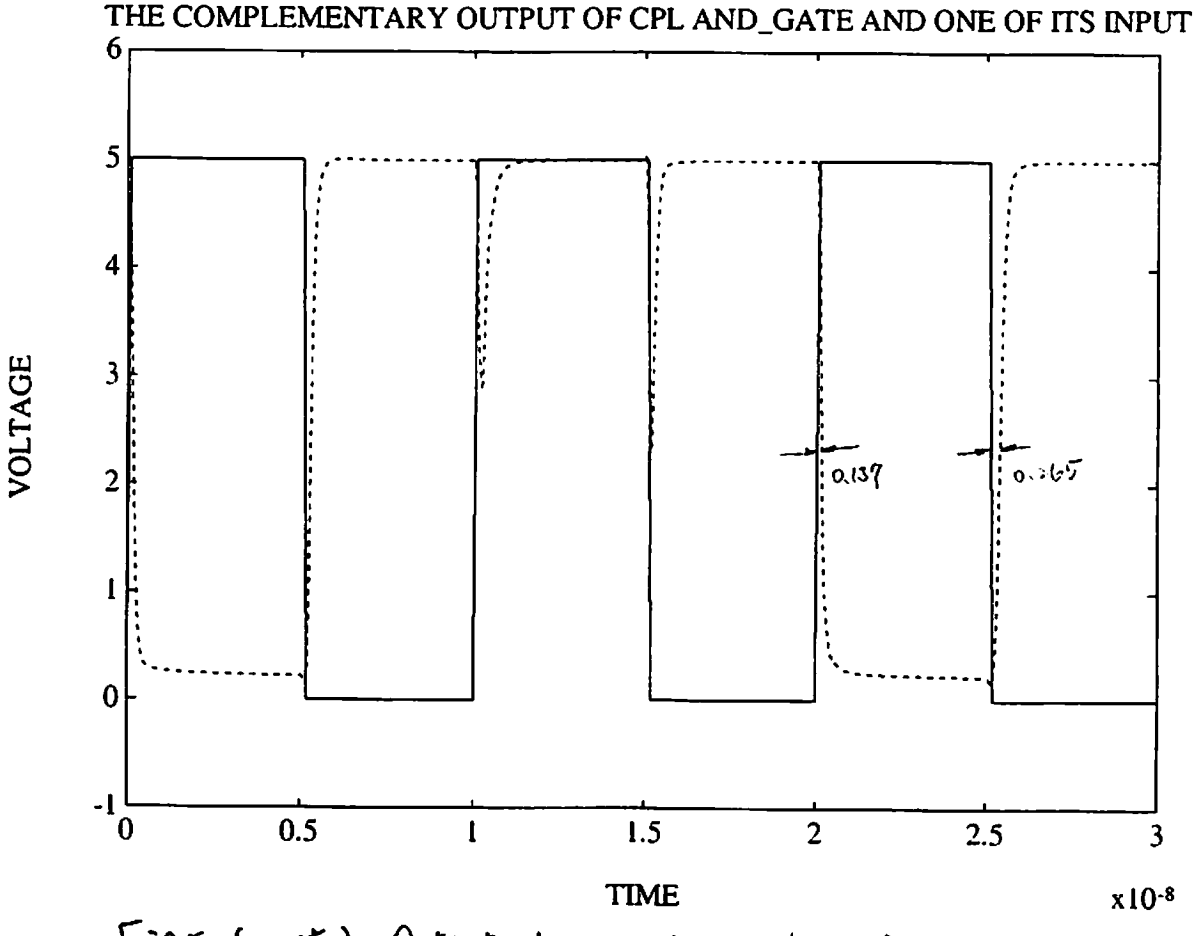


Fig 5. Input Signals for one CPL AND gate





Figs. (cont.) Output for a CPL AND gate

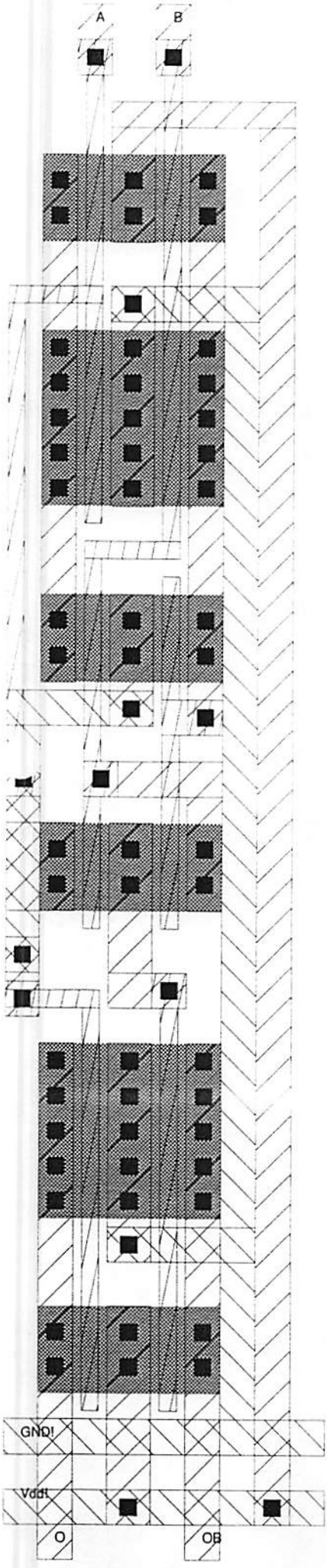


Fig 8.5 (cont.) Layout of one CPL AND gate

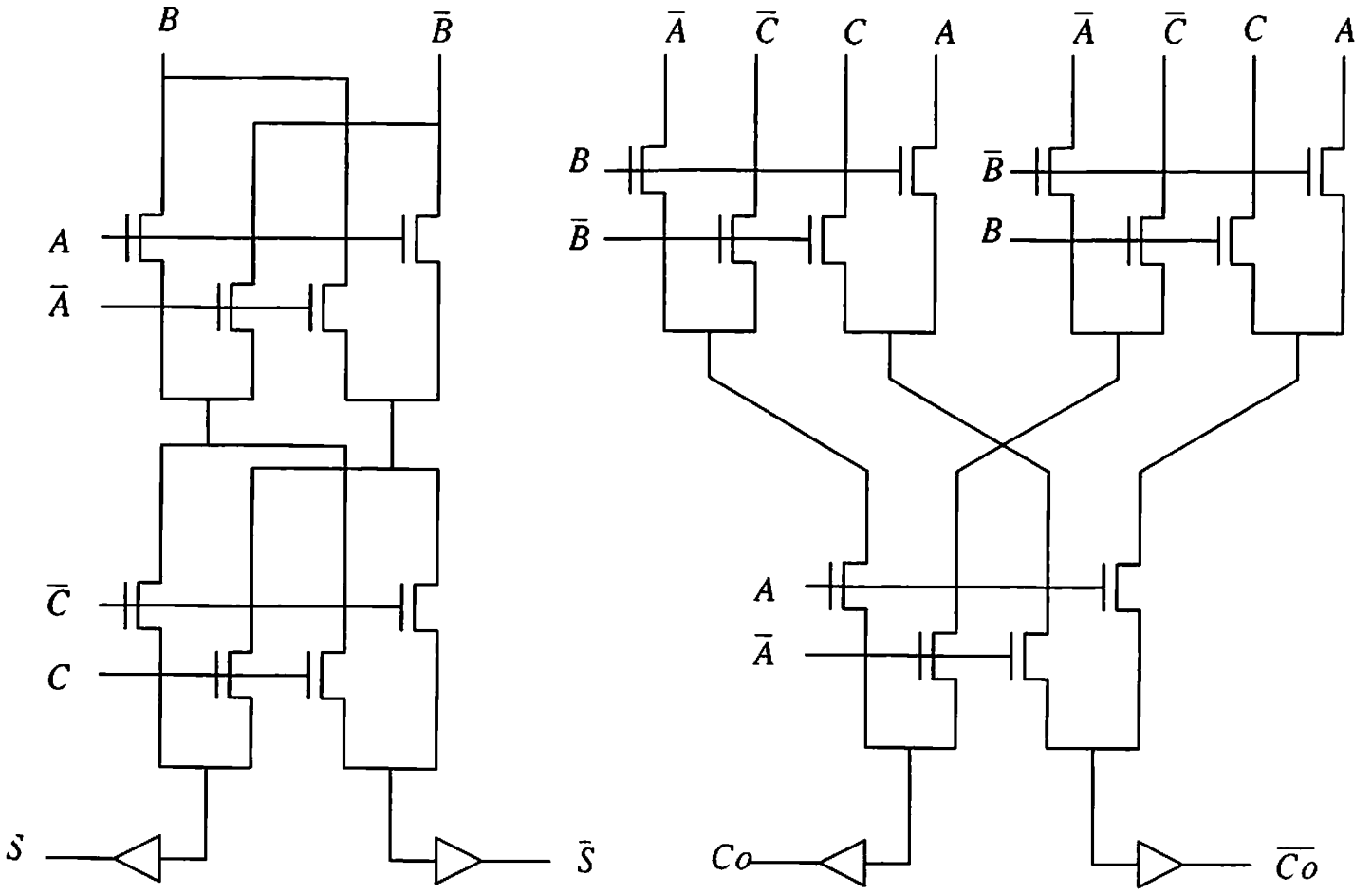


Fig 6. CPL full-adder circuit.

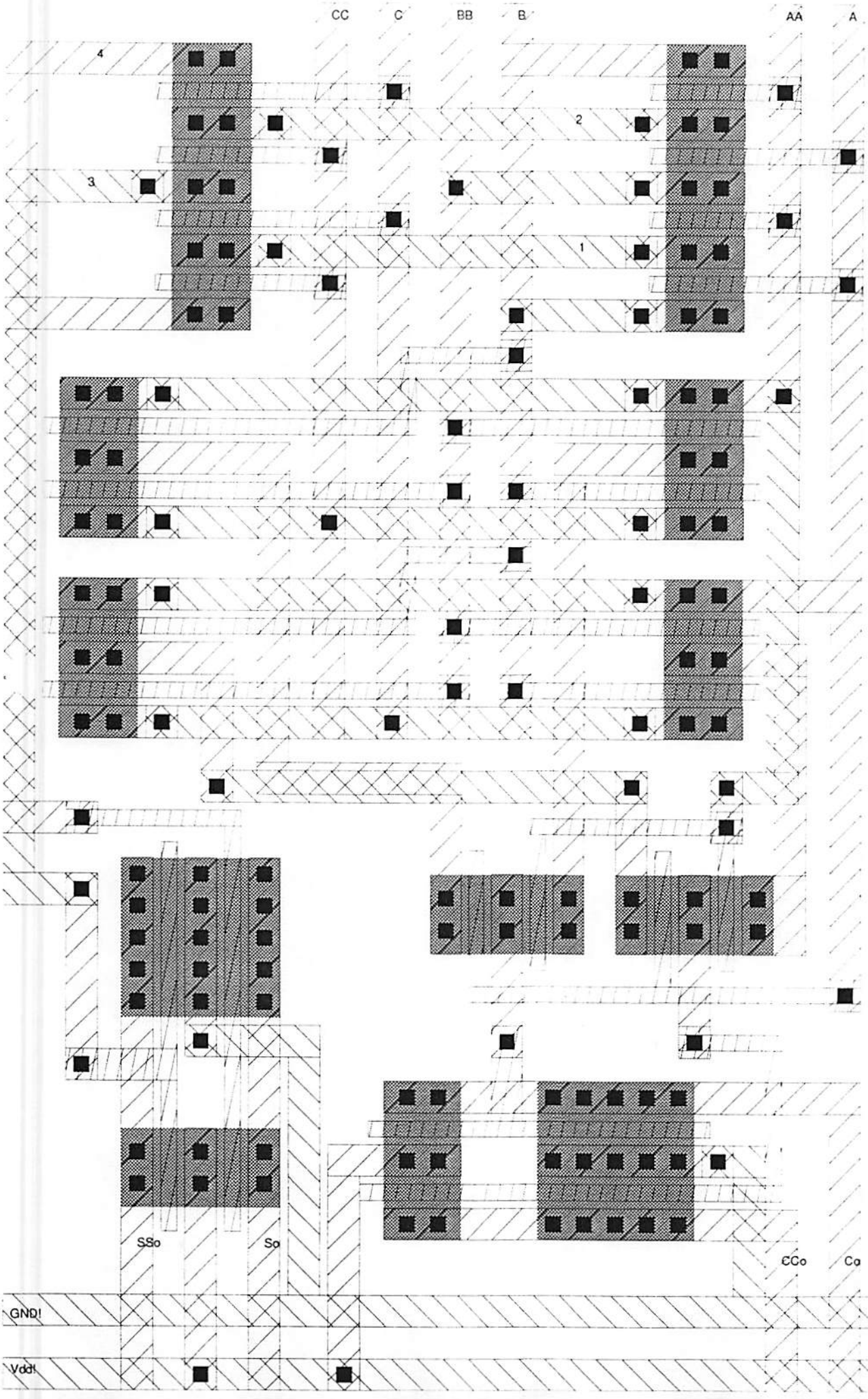


Fig 6. (cont.)
Layout of
one CSA cell

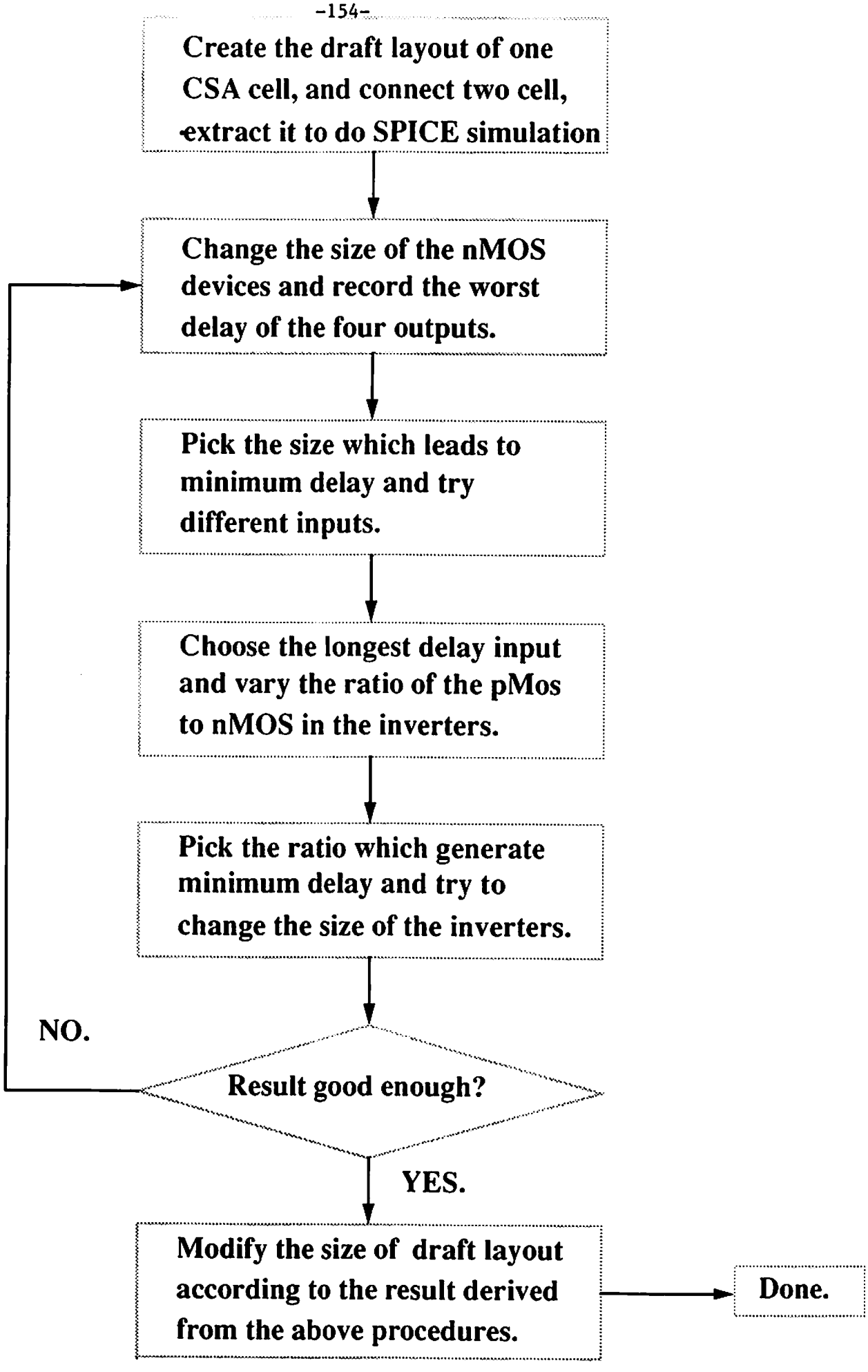


Fig 7. Procedure to explore the appropriate size of the devices.

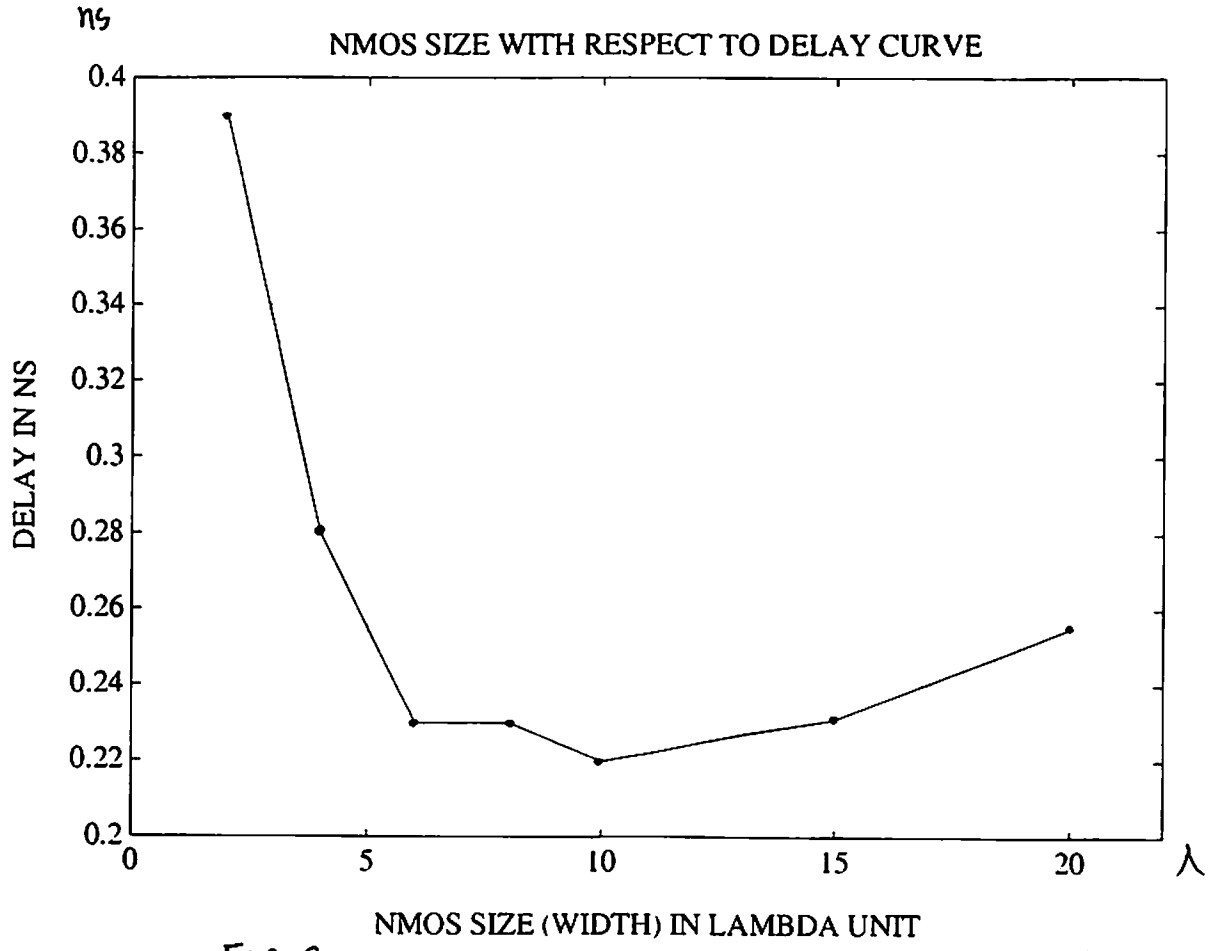


Fig 8. Delay corresponding to different device size.

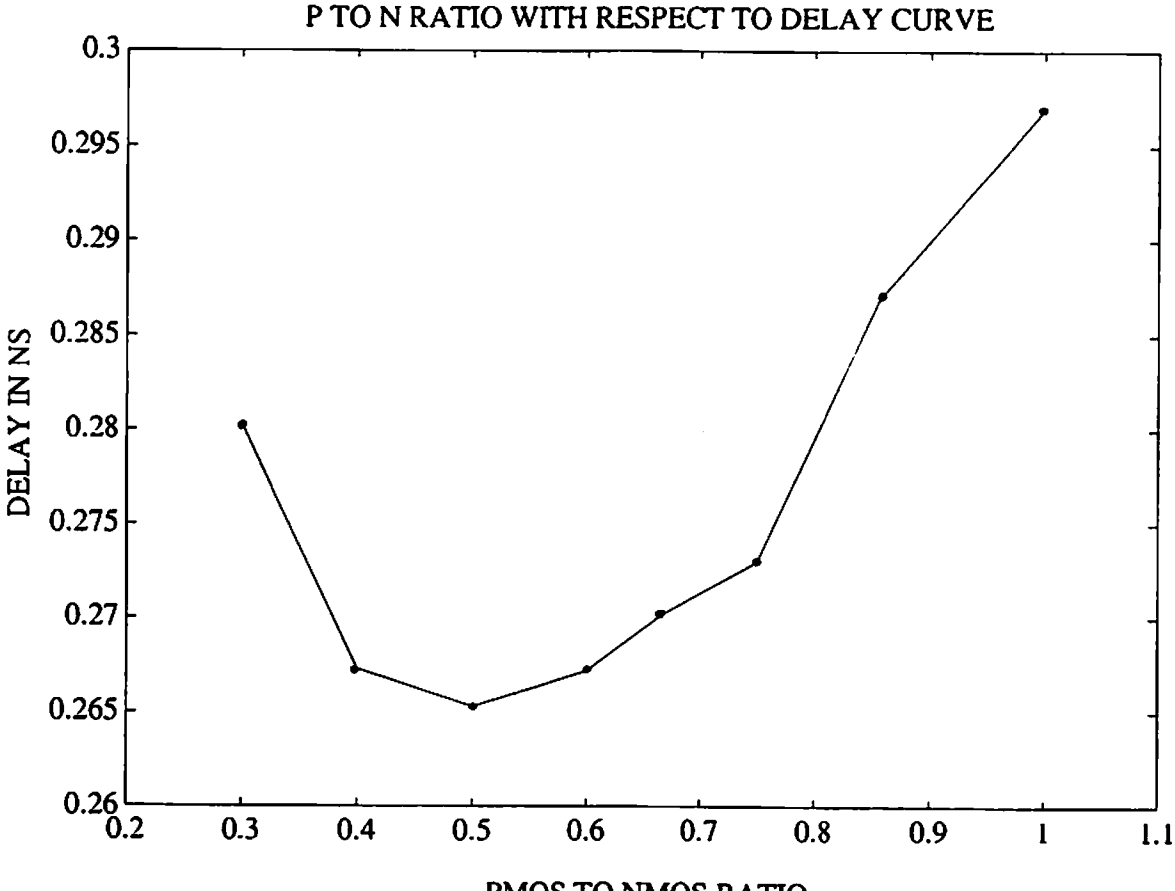


Fig 9. Delay corresponding to different ratio.

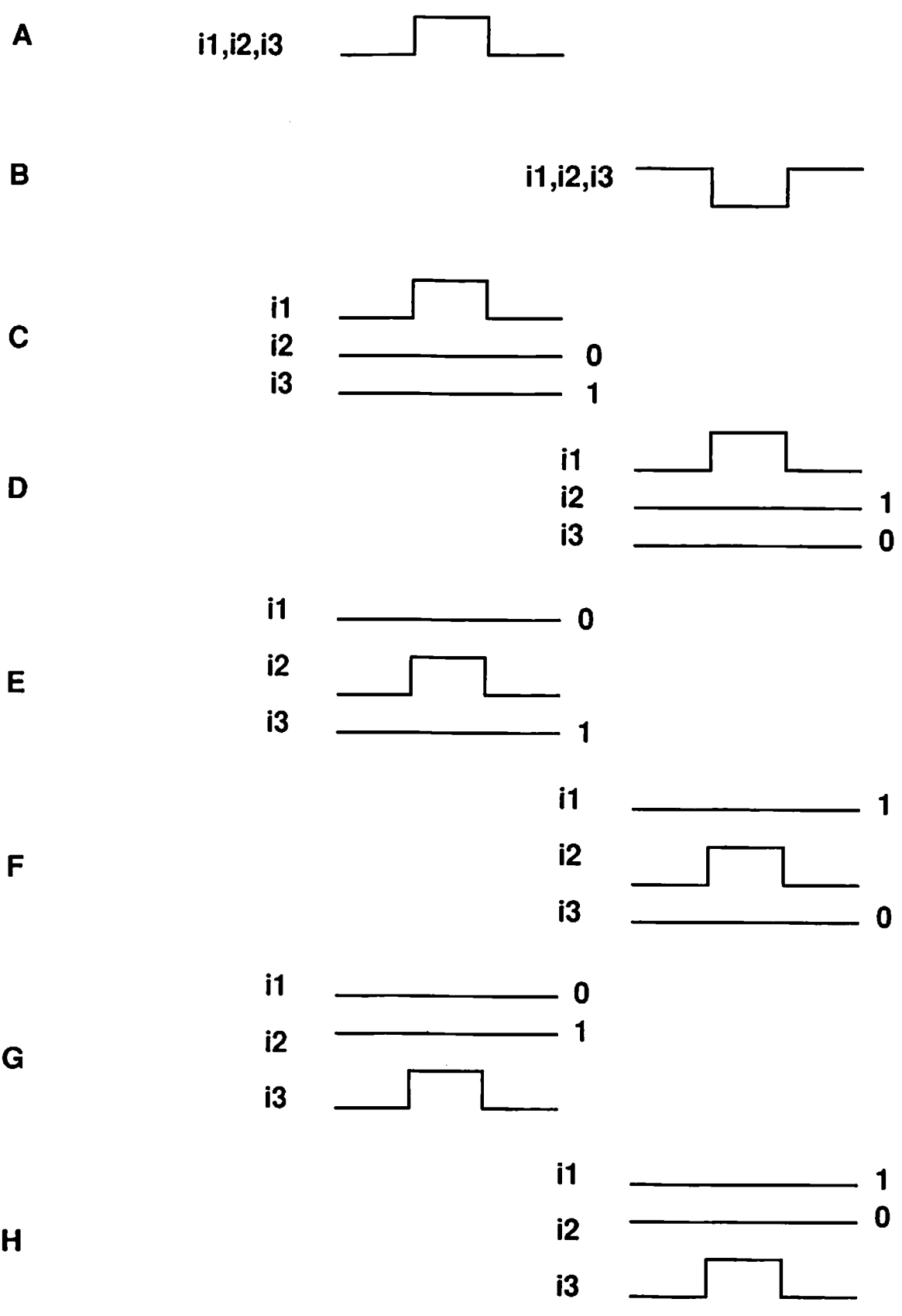


Fig 10. The corresponding waveforms of eight sets of inputs.

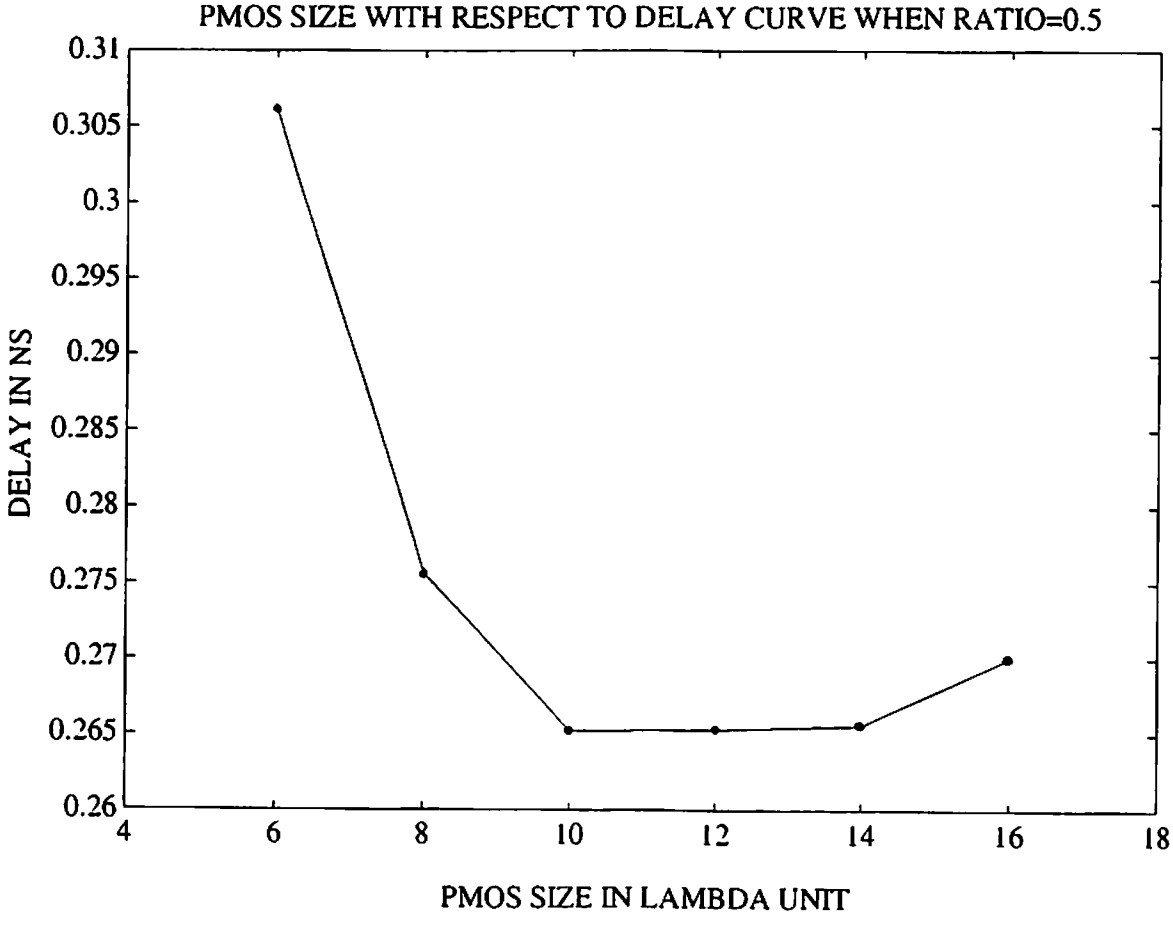


Fig 11. Delay Curve for different inverter size.

Table I . The delay for one adder cell when NMOS size change. Wp/Wn = 10/10 for inverter.

DELAY	2	4	6	8	10	11	13	15	20
SUM	0.39	0.26	0.23	0.23	0.22	0.222	0.227	0.231	0.255
SUMB	0.35	0.24	0.21	0.20	0.202	0.206	0.21	0.217	0.235
CARRY	0.39	0.25	0.20	0.20	0.20	0.20	0.202	0.206	0.222
CARRYB	0.38	0.28	0.22	0.20	0.19	0.20	0.20	0.199	0.22
WORST	0.39	0.28	0.23	0.23	0.22	0.222	0.227	0.305	0.255

Table II. The delay for one adder cell under different set on inputs. Wn = 10 and Wp/Wn = 12/12.

DELAY	A	B	C	D	E	F	G	H
SUM	0.22	0.236	0.254	0.272	0.291	0.297	0.204	0.236
SUMB	0.202	0.219	0.214	0.237	0.289	0.294	0.195	0.219
CARRY	0.20	0.199	0.187	0.203	0.211	0.216	0.207	0.199
CARRYB	0.19	0.196	0.185	0.19	0.209	0.202	0.191	0.196
WORST	0.22	0.236	0.254	0.272	0.291	0.297	0.207	0.236

Table III . The delay for one adder cell when NMOS to PMOS ratio change. Wn = 10 forthe other NMOS.

DELAY	12/12	12/14	12/16	12/18	12/20	12/24	12/30	12/40
SUM	0.297	0.287	0.273	0.2702	0.2672	0.2653	0.2675	0.277
SUMB	0.294	0.275	0.266	0.2619	0.2629	0.263	0.2658	0.2803
CARRY	0.216	0.213	0.209	0.215	0.2113	0.2205	0.2214	0.214
CARRYB	0.202	0.197	0.154	0.168	0.201	0.2015	0.2103	0.2319
WORST	0.297	0.287	0.273	0.2702	0.2672	0.2653	0.2675	0.2803

Tabel IV. The delay for one adder cell when changing the size of the inverter and keeping the ratio fixed.

DELAY	6/12	8/16	10/20	12/24	14/28	16/32
SUM	0.306	0.2756	0.2652	0.2653	0.2656	0.2700
SUMB	0.304	0.2700	0.2580	0.263	0.261	0.2683
CARRY	0.2371	0.2072	0.2142	0.2205	0.2200	0.2260
CARRYB	0.2248	0.210	0.2045	0.2015	0.208	0.2145
WORST	0.306	0.2756	0.2652	0.2653	0.2656	0.2700

Tabel V. Devices sizes, inverter ratio, and longest delay ouput summary for one-bit CSA cell.

nMOS device size	10 λ
inverter ratio	0.5
pMOS size in inverter	10 λ
longest delay input	F (waveform in Fig. 10)

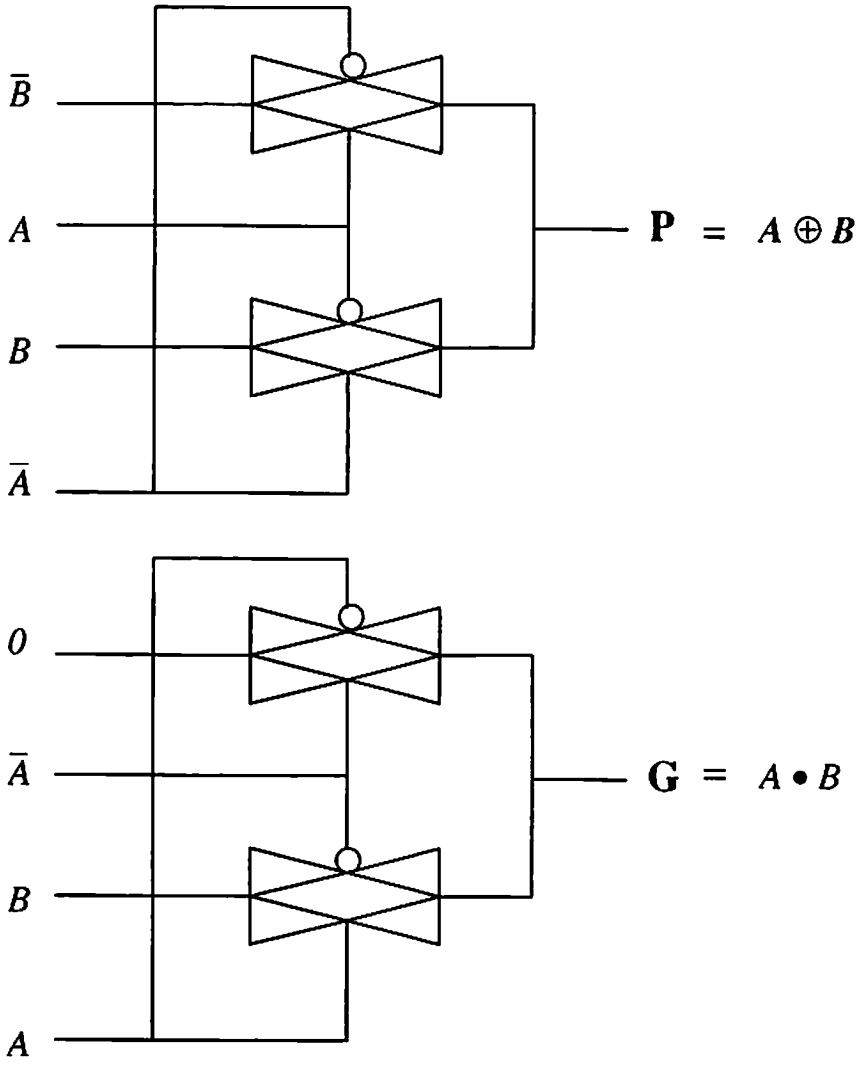


Fig 12. P and G signal implementation.

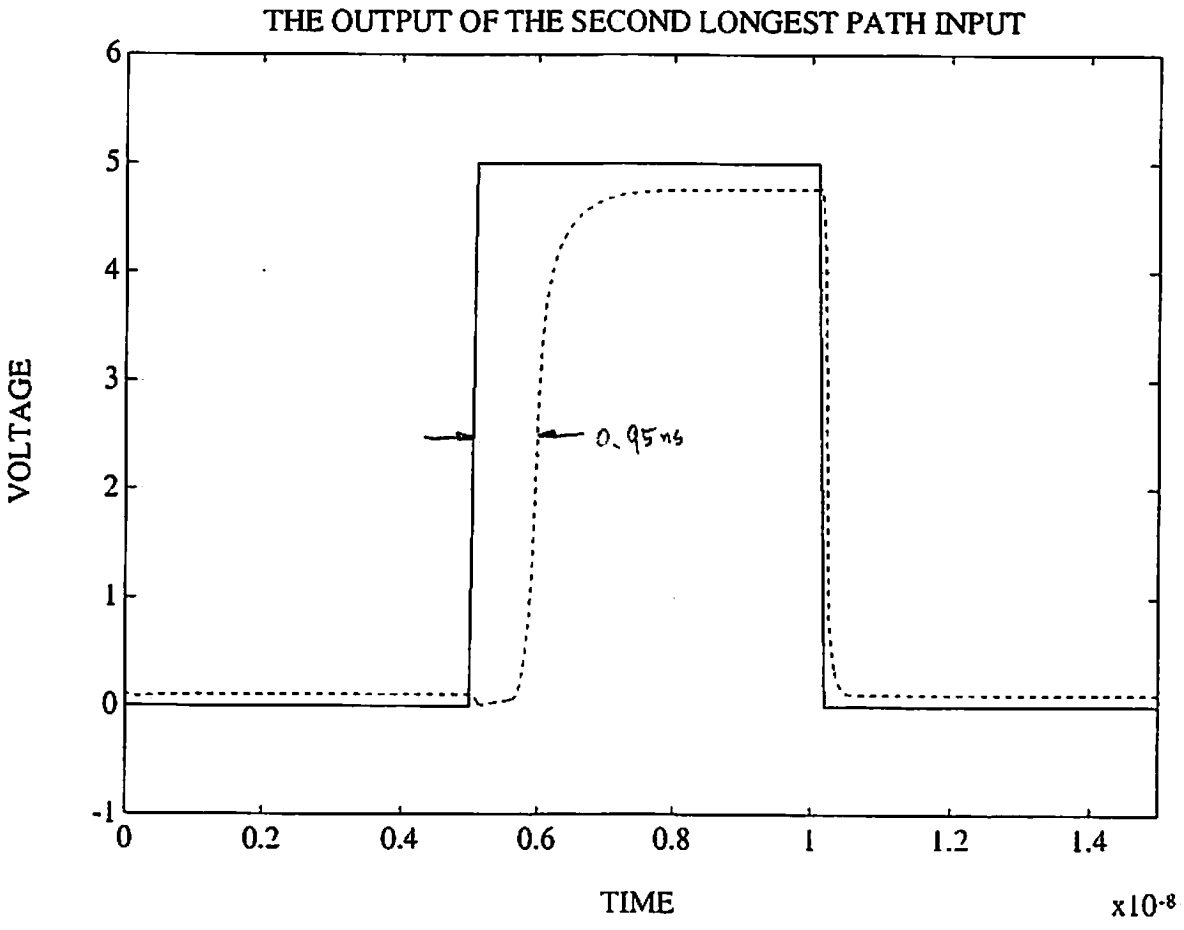


Fig 13. Simulation result for one 4-bit Manchester adder

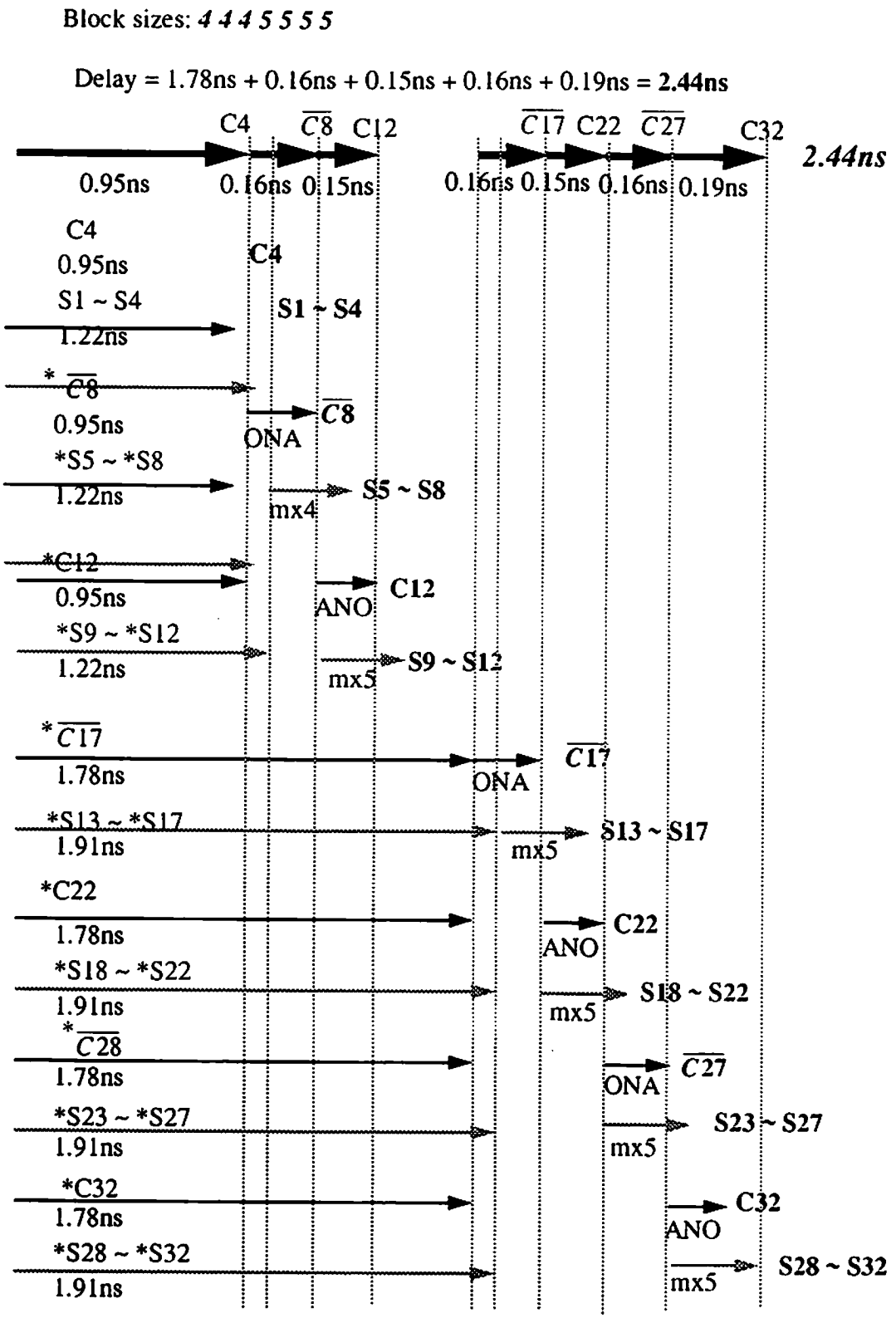


Fig 14. Timing Analysis of the adder.

Tabel VI. Delay and area of the 16*16-b multiplier.

Components	Delay ns	Area λ^2
one AND gate	0.2645	183*37
one cell CSA	0.2652	174*113
CPA	2.44	200*6920
Total	4.3	3400*9500

* NOTE: Delay computation

$$0.2654 + 0.2652*6 + 2.44 = 4.3 \text{ ns}$$

* **NOTE:** Area computation

1. Partial product generator: constant height 183 λ .
   width : 37 * 256 = 9472 λ .

2. Wallace tree : refer to Fig 19. , the floorplanning of the Wallace adder array.
   It's difficult to rearrange the CSA cells due to its huge number of signal lines. Therefore, I just move one 18-b CSA from stage 1 to stage 2. The routing area between the stages can be estimated:
   (1) routing between PPG and CSA : 50 λ.
   (2) routing between first stage to secnod stage: 500 λ
   (3) routing between secnod stage to third stage: 400 λ
   (4) routing between third to fourth: 200 λ
   (5) routing between fourth to sixth: 400λ
   (6) routing between sixth to CPA: 400 λ

   Total height: 174*6+50+1900= 3000 λ
   Total width : 113*18*4+16*2*8*4= 9160 λ

3. Adder: 200*6920 λ^2

4. Total : height : 183+ 3000+ 200 -> 3400 λ
   width : 9500 λ

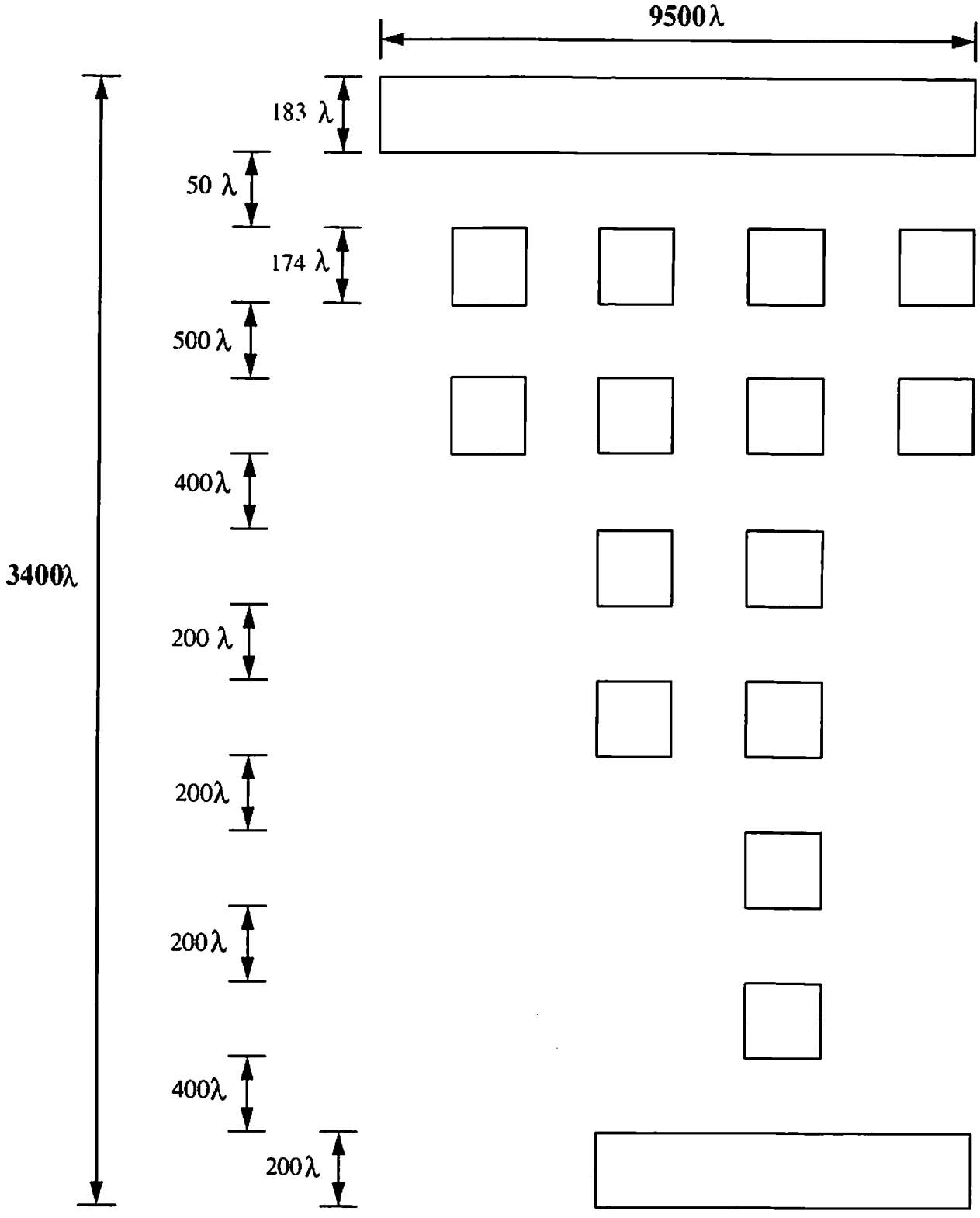


Fig 15. Floorplanning of the 16*16-b multiplier.

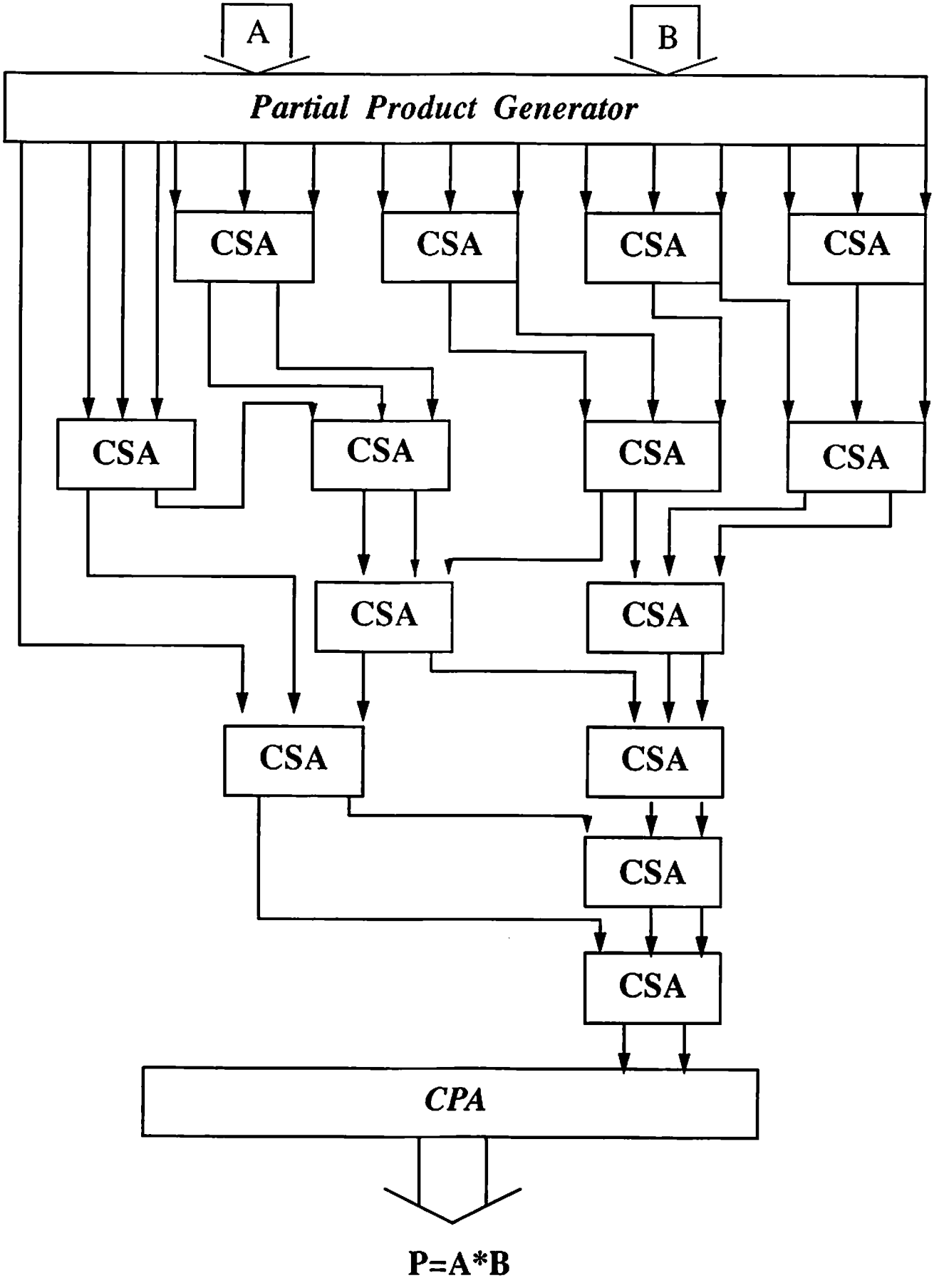
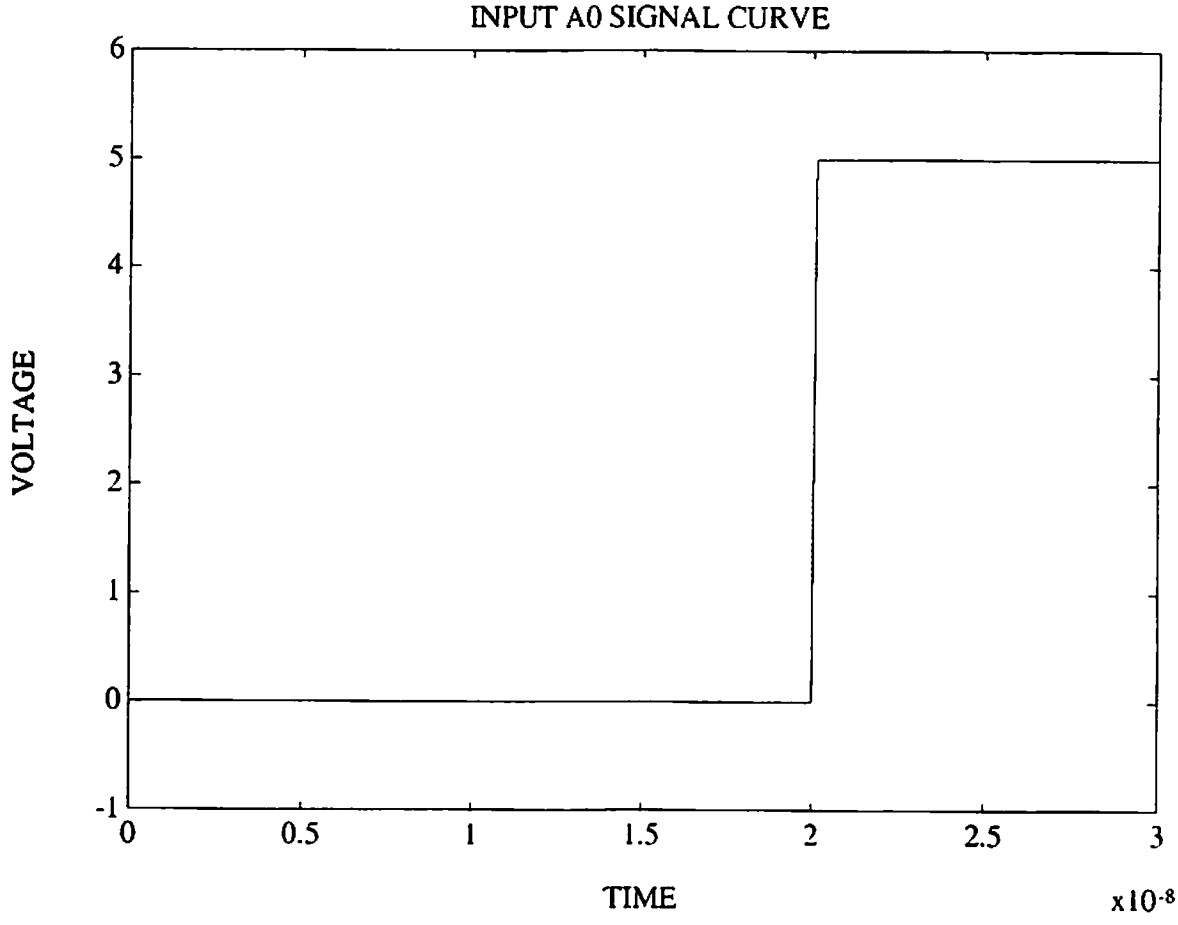


Fig 15(cont). Floorplanning and interconnection of the Wallace Tree.



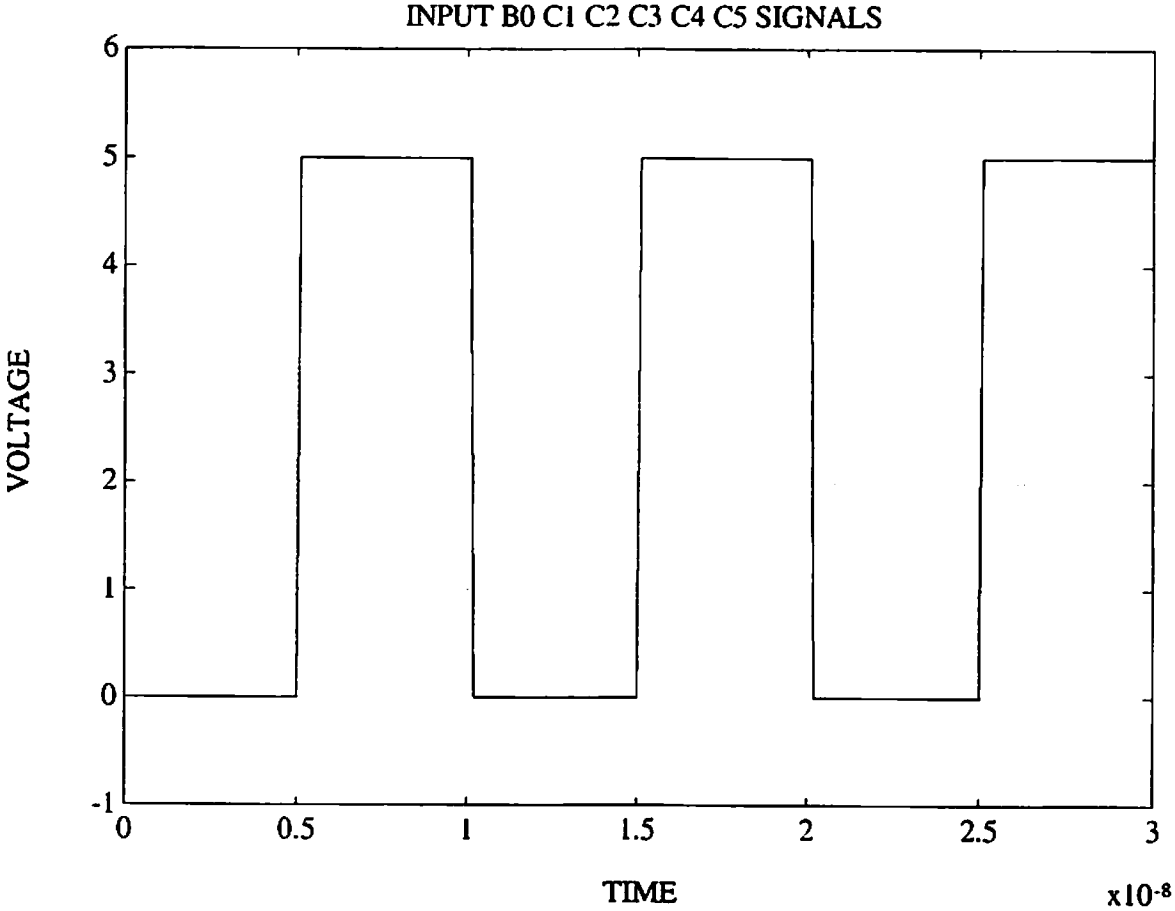


Fig16. Inputs for six-CSA simulation.

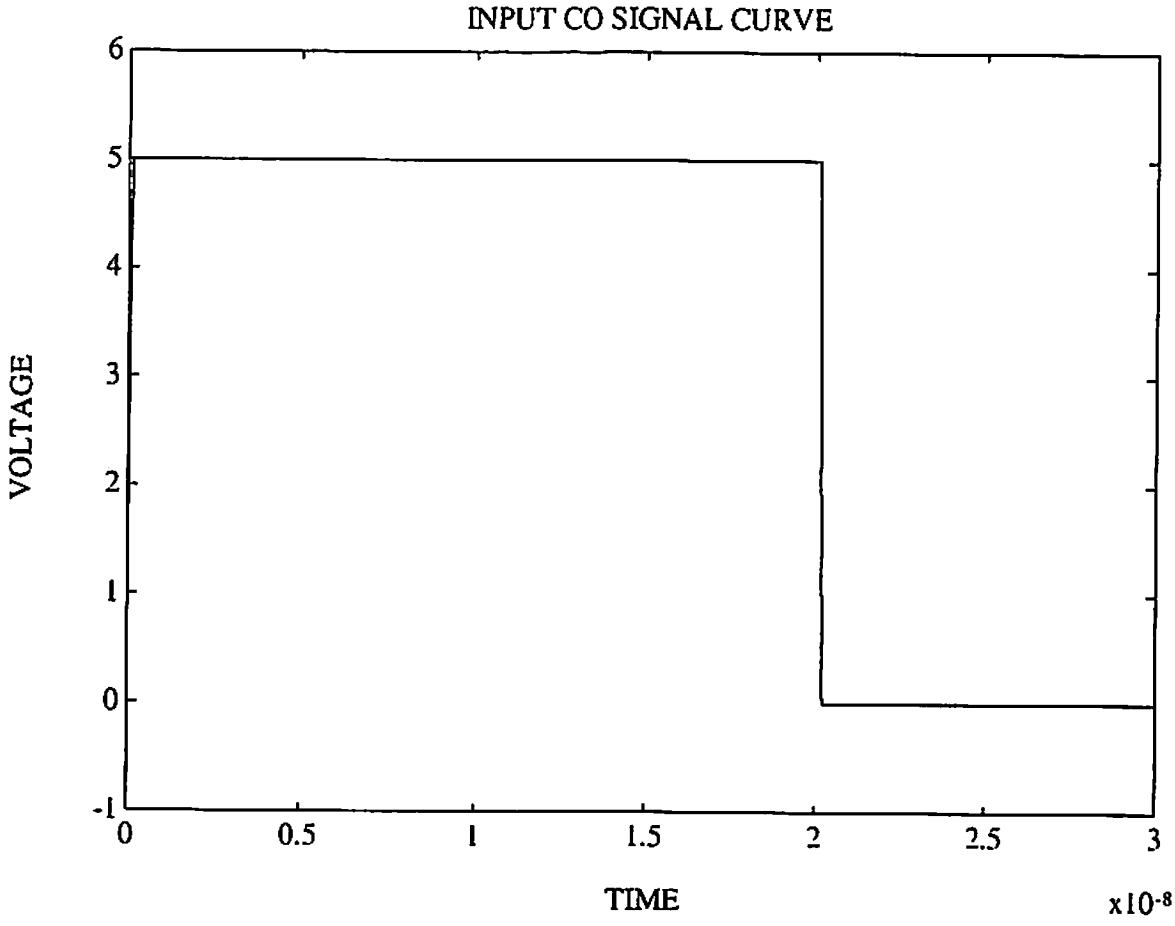
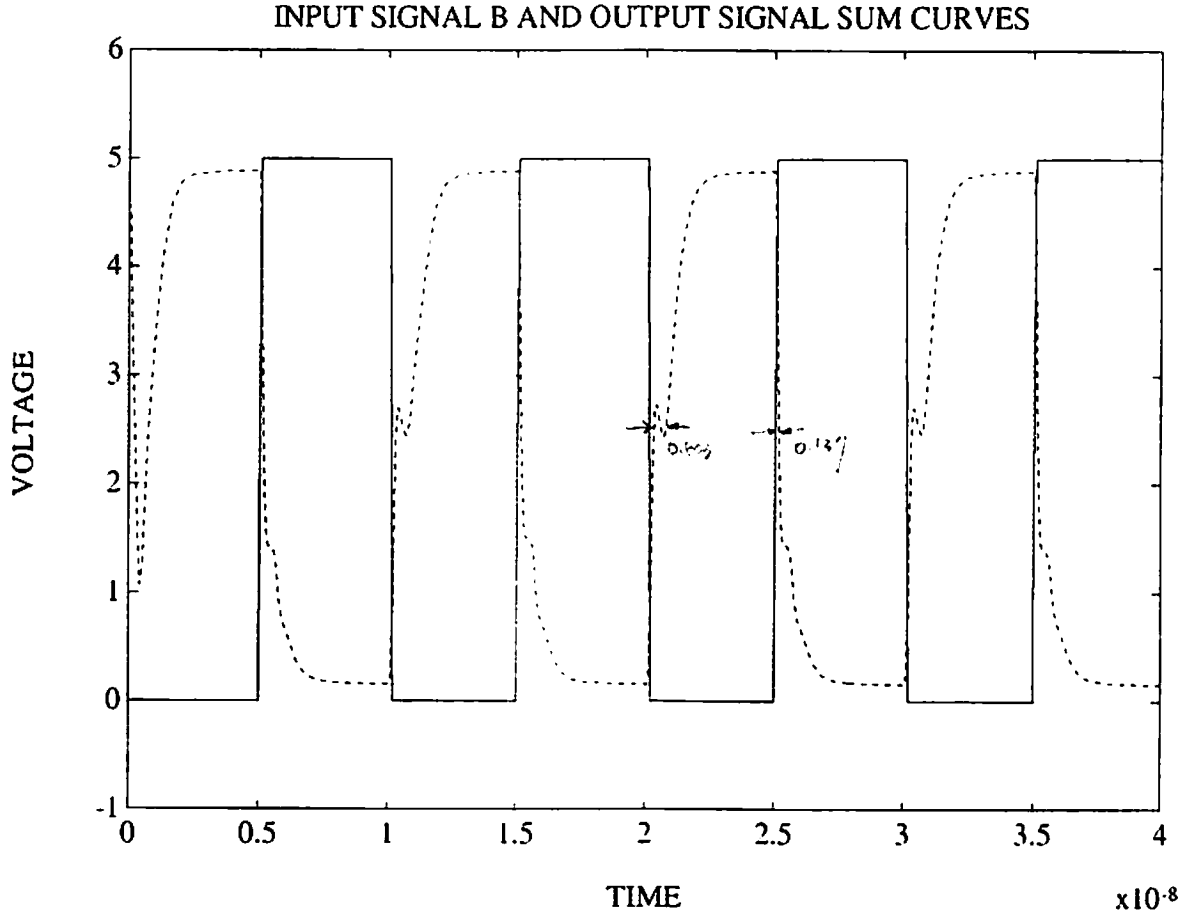

INPUT CO SIGNAL CURVE
VOLTAGE
6
5
4
3
2
1
0
-1
0
0.5
1
1.5
2
2.5
3
TIME
x10-8

Fig 16. (cont.)



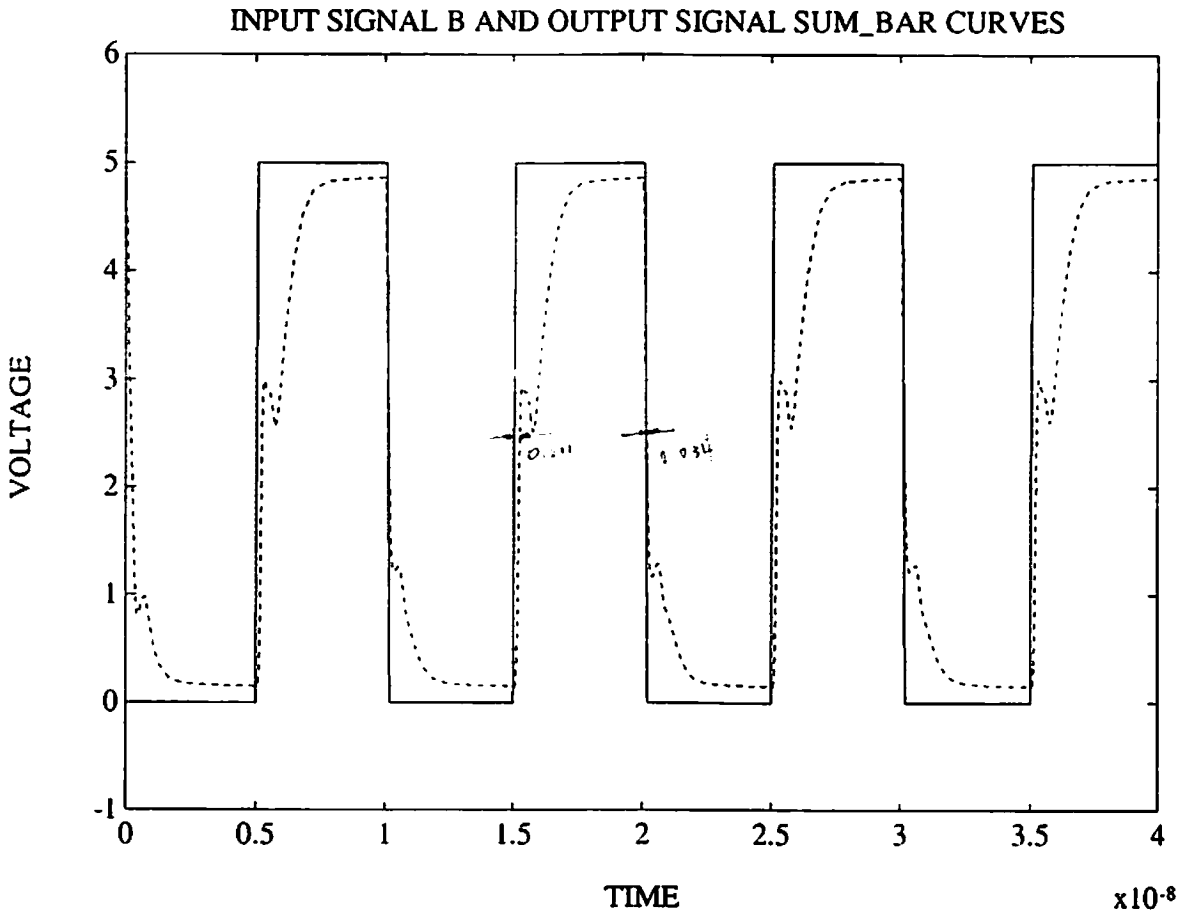
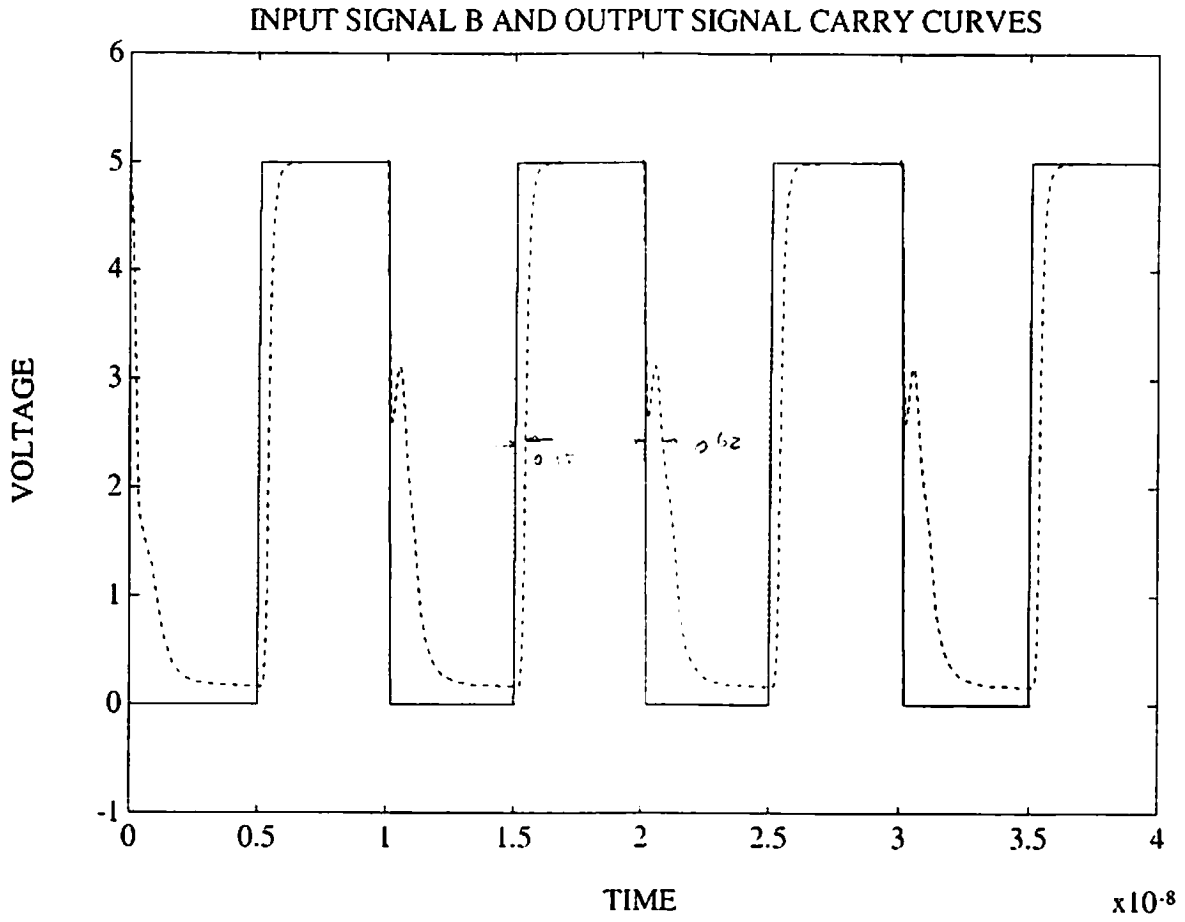


Fig16. (cont.) Output of 6-CSA simulation.



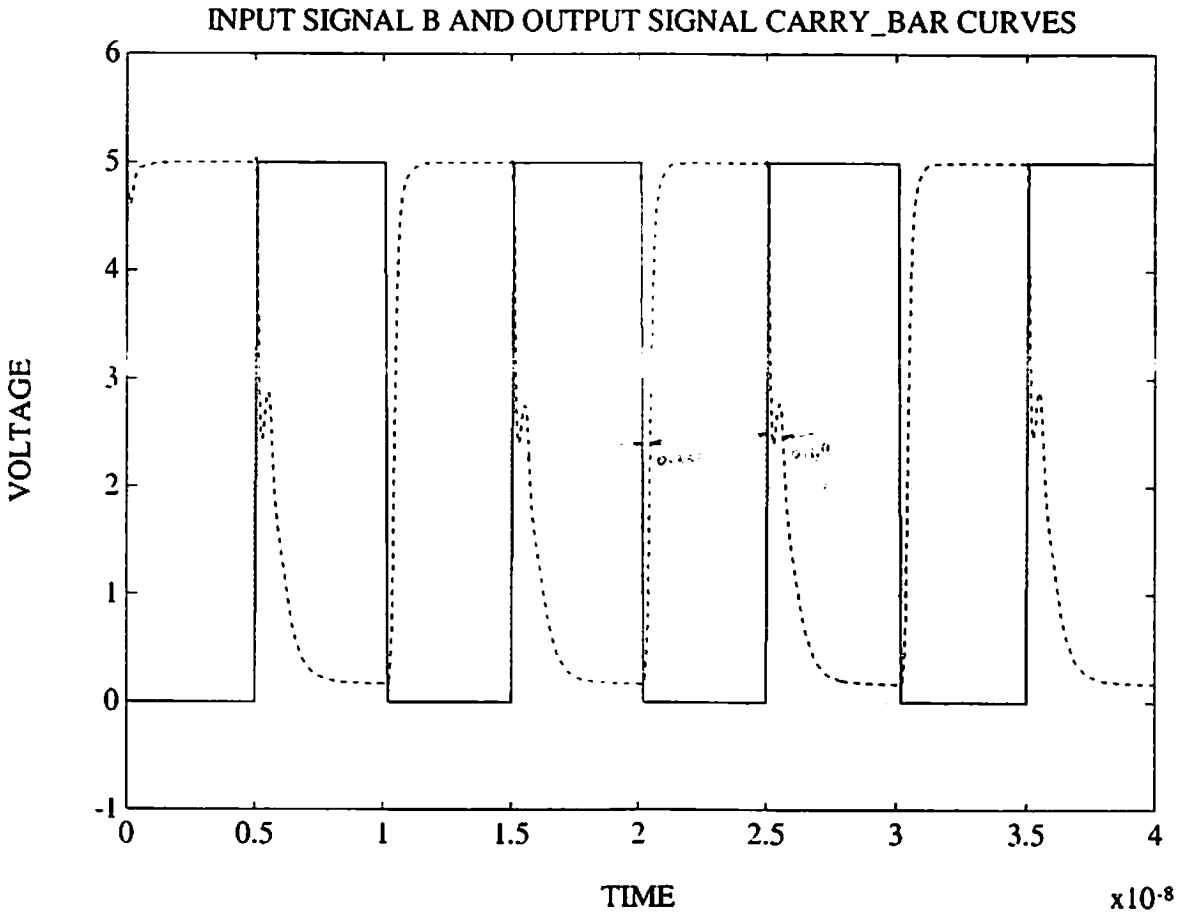


Fig 16. (cont.) Output of 6-CSA simulation

```
** SPICE file is scaled from 2cpl.spice 1.2um to 2cpl.sp 0.5um.
** Resistor scale = 1.00000
** Capacitor scale = 0.17360

** SPICE file created for circuit 2cpl
** Technology: scmos
**
** NODE: 0 = GND
** NODE: 1 = Vdd
** NODE: 2 = Error
RLUMP0 100 101 617.0
RLUMP1 102 103 480.0
RLUMP2 104 105 232.0
M0 101 103 105 0 nfet L=0.5U W=2.5U
RLUMP3 106 107 1177.5
RLUMP4 108 109 1682.5
RLUMP5 104 110 232.0
M1 107 109 110 0 nfet L=0.5U W=2.5U
RLUMP6 104 111 232.0
RLUMP7 112 113 574.0
RLUMP8 114 115 451.0
M2 111 113 115 0 nfet L=0.5U W=2.5U
RLUMP9 104 116 232.0
RLUMP10 117 118 1224.5
RLUMP11 119 120 1497.5
M3 116 118 120 0 nfet L=0.5U W=2.5U
RLUMP12 114 121 451.0
RLUMP13 102 122 480.0
RLUMP14 123 124 232.0
M4 121 122 124 0 nfet L=0.5U W=2.5U
RLUMP15 119 125 1497.5
RLUMP16 108 126 1682.5
RLUMP17 123 127 232.0
M5 125 126 127 0 nfet L=0.5U W=2.5U
RLUMP18 123 128 232.0
RLUMP19 112 129 574.0
RLUMP20 100 130 617.0
M6 128 129 130 0 nfet L=0.5U W=2.5U
RLUMP21 123 131 232.0
RLUMP22 117 132 1224.5
RLUMP23 106 133 1177.5
M7 131 132 133 0 nfet L=0.5U W=2.5U
RLUMP24 134 135 227.5
RLUMP25 117 136 1224.5
RLUMP26 137 138 381.5
M8 135 136 138 0 nfet L=0.5U W=2.5U
RLUMP27 137 139 381.5
RLUMP28 108 140 1682.5
RLUMP29 141 142 232.0
M9 139 140 142 0 nfet L=0.5U W=2.5U
RLUMP30 108 143 1682.5
RLUMP31 119 144 1497.5
RLUMP32 141 145 232.0
M10 143 144 145 0 nfet L=0.5U W=2.5U
RLUMP33 108 146 1682.5
RLUMP34 106 147 1177.5
RLUMP35 134 148 227.5
M11 146 147 148 0 nfet L=0.5U W=2.5U
RLUMP36 141 149 232.0
RLUMP37 106 150 1177.5
RLUMP38 112 151 574.0
M12 149 150 151 0 nfet L=0.5U W=2.5U
RLUMP39 134 152 227.5
RLUMP40 119 153 1497.5
RLUMP41 112 154 574.0
M13 152 153 154 0 nfet L=0.5U W=2.5U
RLUMP42 117 155 1224.5
RLUMP43 106 156 1177.5
RLUMP44 157 158 227.0
M14 155 156 158 0 nfet L=0.5U W=2.5U
RLUMP45 117 159 1224.5
RLUMP46 119 160 1497.5
RLUMP47 161 162 227.5
M15 159 160 162 0 nfet L=0.5U W=2.5U
RLUMP48 157 163 227.0
RLUMP49 117 164 1224.5
RLUMP50 165 166 381.0
M16 163 164 166 0 nfet L=0.5U W=2.5U
RLUMP51 157 167 227.0
RLUMP52 119 168 1497.5
RLUMP53 102 169 480.0
M17 167 168 169 0 nfet L=0.5U W=2.5U
RLUMP54 161 170 227.5
RLUMP55 106 171 1177.5
RLUMP56 102 172 480.0
M18 170 171 172 0 nfet L=0.5U W=2.5U
RLUMP57 165 173 381.0
RLUMP58 108 174 1682.5
RLUMP59 161 175 227.5
M19 173 174 175 0 nfet L=0.5U W=2.5U
RLUMP60 176 177 234.5
RLUMP61 137 178 381.5
M20 177 178 1 1 pfet L=0.5U W=2.5U
RLUMP62 176 179 234.5
RLUMP63 137 180 381.5
M21 179 180 0 0 nfet L=0.5U W=5.0U
RLUMP64 165 181 381.0
RLUMP65 182 183 235.0
M22 1 181 183 1 pfet L=0.5U W=2.5U
RLUMP66 165 184 381.0
RLUMP67 182 185 235.0
M23 0 184 185 0 nfet L=0.5U W=5.0U
RLUMP68 186 187 234.5
RLUMP69 100 188 617.0
M24 187 188 1 1 pfet L=0.5U W=2.5U
RLUMP70 186 189 234.5
RLUMP71 100 190 617.0
M25 189 190 0 0 nfet L=0.5U W=5.0U
RLUMP72 114 191 451.0
RLUMP73 192 193 234.5
M26 1 191 193 1 pfet L=0.5U W=2.5U
RLUMP74 114 194 451.0
RLUMP75 192 195 234.5
M27 0 194 195 0 nfet L=0.5U W=5.0U
RLUMP76 196 197 617.0
RLUMP77 198 199 480.0
RLUMP78 200 201 232.0
M28 197 199 201 0 nfet L=0.5U W=2.5U
RLUMP79 202 203 874.5
RLUMP80 204 205 1380.0
RLUMP81 200 206 232.0
M29 203 205 206 0 nfet L=0.5U W=2.5U
RLUMP82 200 207 232.0
RLUMP83 208 209 574.0
RLUMP84 210 211 451.0
M30 207 209 211 0 nfet L=0.5U W=2.5U
RLUMP85 200 212 232.0
RLUMP86 213 214 921.5
```

TABLE VII. SPICE file for simulation two CSAs.

```
RLUMP87 215 216 1164.0
M31 212 214 216 0 nfet L=0.5U W=2.5U
RLUMP88 210 217 451.0
RLUMP89 198 218 480.0
RLUMP90 219 220 232.0
M32 217 218 220 0 nfet L=0.5U W=2.5U
RLUMP91 215 221 1164.0
RLUMP92 204 222 1380.0
RLUMP93 219 223 232.0
M33 221 222 223 0 nfet L=0.5U W=2.5U
RLUMP94 219 224 232.0
RLUMP95 208 225 574.0
RLUMP96 196 226 617.0
M34 224 225 226 0 nfet L=0.5U W=2.5U
RLUMP97 219 227 232.0
RLUMP98 213 228 921.5
RLUMP99 202 229 874.5
M35 227 228 229 0 nfet L=0.5U W=2.5U
RLUMP100 230 231 227.5
RLUMP101 213 232 921.5
RLUMP102 233 234 381.5
M36 231 232 234 0 nfet L=0.5U W=2.5U
RLUMP103 233 235 381.5
RLUMP104 204 236 1380.0
RLUMP105 237 238 232.0
M37 235 236 238 0 nfet L=0.5U W=2.5U
RLUMP106 204 239 1380.0
RLUMP107 215 240 1164.0
RLUMP108 237 241 232.0
M38 239 240 241 0 nfet L=0.5U W=2.5U
RLUMP109 204 242 1380.0
RLUMP110 202 243 874.5
RLUMP111 230 244 227.5
M39 242 243 244 0 nfet L=0.5U W=2.5U
RLUMP112 237 245 232.0
RLUMP113 202 246 874.5
RLUMP114 208 247 574.0
M40 245 246 247 0 nfet L=0.5U W=2.5U
RLUMP115 230 248 227.5
RLUMP116 215 249 1164.0
RLUMP117 208 250 574.0
M41 248 249 250 0 nfet L=0.5U W=2.5U
RLUMP118 213 251 921.5
RLUMP119 202 252 874.5
RLUMP120 253 254 227.0
M42 251 252 254 0 nfet L=0.5U W=2.5U
RLUMP121 213 255 921.5
RLUMP122 215 256 1164.0
RLUMP123 257 258 227.5
M43 255 256 258 0 nfet L=0.5U W=2.5U
RLUMP124 253 259 227.0
RLUMP125 213 260 921.5
RLUMP126 261 262 381.0
M44 259 260 262 0 nfet L=0.5U W=2.5U
RLUMP127 253 263 227.0
RLUMP128 215 264 1164.0
RLUMP129 198 265 480.0
M45 263 264 265 0 nfet L=0.5U W=2.5U
RLUMP130 257 266 227.5
RLUMP131 202 267 874.5
RLUMP132 198 268 480.0
M46 266 267 268 0 nfet L=0.5U W=2.5U
RLUMP133 261 269 381.0
RLUMP134 204 270 1380.0
RLUMP135 257 271 227.5
M47 269 270 271 0 nfet L=0.5U W=2.5U
RLUMP136 106 272 1177.5
RLUMP137 233 273 381.5
M48 272 273 1 1 pfet L=0.5U W=2.5U
RLUMP138 106 274 1177.5
RLUMP139 233 275 381.5
M49 274 275 0 0 nfet L=0.5U W=5.0U
RLUMP140 261 276 381.0
RLUMP141 119 277 1497.5
M50 1 276 277 1 pfet L=0.5U W=2.5U
RLUMP142 261 278 381.0
RLUMP143 119 279 1497.5
M51 0 278 279 0 nfet L=0.5U W=5.0U
RLUMP144 117 280 1224.5
RLUMP145 196 281 617.0
M52 280 281 1 1 pfet L=0.5U W=2.5U
RLUMP146 117 282 1224.5
RLUMP147 196 283 617.0
M53 282 283 0 0 nfet L=0.5U W=5.0U
RLUMP148 210 284 451.0
RLUMP149 108 285 1682.5
M54 1 284 285 1 pfet L=0.5U W=2.5U
RLUMP150 210 286 451.0
RLUMP151 108 287 1682.5
M55 0 286 287 0 nfet L=0.5U W=5.0U
** NODE: 0 = GND!
C0 1 0 18F
** NODE: 1 = Vdd!
C1 261 0 2F
** NODE: 261 = addcpl2_1/6_27_169#
C2 257 0 2F
** NODE: 257 = addcpl2_1/6_27_183#
C3 253 0 1F
** NODE: 253 = addcpl2_1/6_27_153#
C4 237 0 2F
** NODE: 237 = addcpl2_1/6_27_103#
C5 233 0 2F
** NODE: 233 = addcpl2_1/6_27_87#
C6 230 0 2F
** NODE: 230 = addcpl2_1/6_27_71#
C7 219 0 1F
** NODE: 219 = addcpl2_1/6_27_39#
C8 215 0 6F
** NODE: 215 = BB0
C9 213 0 5F
** NODE: 213 = A0
C10 210 0 3F
** NODE: 210 = addcpl2_1/6_37_267#
C11 208 0 4F
** NODE: 208 = CCi0
C12 200 0 1F
** NODE: 200 = addcpl2_1/6_27_7#
C13 204 0 6F
** NODE: 204 = AA0
C14 202 0 6F
** NODE: 202 = B0
C15 198 0 4F
** NODE: 198 = Ci0
C16 196 0 5F
** NODE: 196 = addcpl2_1/6_37_251#
C17 192 0 5F
** NODE: 192 = SS2
C18 186 0 6F
```

```
** NODE: 186 = S2
C19 182 0 5F
** NODE: 182 = CC2
C20 176 0 6F
** NODE: 176 = C2
C21 165 0 2F
** NODE: 165 = addcpl2_0/6_27_169#
C22 161 0 2F
** NODE: 161 = addcpl2_0/6_27_183#
C23 157 0 1F
** NODE: 157 = addcpl2_0/6_27_153#
C24 141 0 2F
** NODE: 141 = addcpl2_0/6_27_103#
C25 137 0 2F
** NODE: 137 = addcpl2_0/6_27_87#
C26 134 0 2F
** NODE: 134 = addcpl2_0/6_27_71#
C27 123 0 1F
** NODE: 123 = addcpl2_0/6_27_39#
C28 119 0 12F
** NODE: 119 = CC1
C29 117 0 12F
** NODE: 117 = S1
C30 114 0 3F
** NODE: 114 = addcpl2_0/6_37_267#
C31 112 0 4F
** NODE: 112 = CC11
C32 104 0 1F
** NODE: 104 = addcpl2_0/6_27_7#
C33 108 0 11F
** NODE: 108 = SS1
C34 106 0 12F
** NODE: 106 = C1
C35 102 0 4F
** NODE: 102 = C11
C36 100 0 5F
** NODE: 100 = addcpl2_0/6_37_251#
VDD 1 0 5V
VA   213 0 DC 5V PULSE(0 5 0NS 100ps 100ps 20NS 30NS)
VAA  204 0 DC 5V PULSE(0 5 20NS 100PS 100PS 10NS 30NS)
VB   202 0 DC 5V PULSE(0 5 5NS 100ps 100ps 5NS 10NS)
VBB  215 0 DC 5V PULSE(0 5 0NS 100PS 100PS 5NS 10NS)
VC0  198 0 DC 5V PULSE(0 5 20NS 100PS 100PS 10NS 30NS)
VCC0 208 0 DC 5V PULSE(0 5 0NS 100PS 100PS 20NS 30NS)
VC1  102 0 DC 5V PULSE(0 5 5NS 100PS 100PS 5NS 10NS)
VCC1 112 0 DC 5V PULSE(0 5 0NS 100PS 100PS 5NS 10NS)
.MODEL nfet NMOS LEVEL=3 PHI=0.600000 TOX=1.100E-08 XJ=0.200000U TPG=1
+ VTO=0.8186 DELTA=1.7570E+00 LD=0 KP=9.1547E-05
+ UO=596.5 THETA=1.0850E-01 GAMMA=0.5266 NSUB=1.9680E+16
+ NFS=5.5000E+12 VMAX=1.9420E+05 ETA=6.6540E-02 KAPPA=1.1210E-01
+ CGDO=2.6106E-10 CGSO=2.6106E-10 CGBO=6.3402E-10
+ CJ=3.1146E-04 MJ=1.0667 CJSW=4.3777E-10 MJSW=0.154230 PB=0.800000
* Weff= Wdrawn - Delta_W
* The suggested Delta_W is 1.9970E-07
.MODEL pfet PMOS LEVEL=3 PHI=0.600000 TOX=1.100E-08 XJ=0.200000U TPG=-1
+ VTO=-0.9456 DELTA=1.5520E+00 LD=0 KP=3.1646E-05
+ UO=206.2 THETA=1.6900E-01 GAMMA=0.4619 NSUB=1.5140E+16
+ NFS=4.9990E+12 VMAX=4.4410E+05 ETA=1.6350E-01 KAPPA=1.0000E+01
+ CGDO=2.6981E-11 CGSO=2.6981E-11 CGBO=8.6508E-10
+ CJ=4.7864E-04 MJ=0.4973 CJSW=1.4771E-10 MJSW=0.190593 PB=0.850000
* Weff = Wdrawn - Delta_W
* The suggested Delta_W is 3.1280E-07
*.options Gmin=1.0E-10
.tran 0.01ns 15ns
.END
```

EE 577 Term Project

DSP RISC- Cache Memory

Professor : Bing Sheu

Name : Shen-Te Hong

ID # : 888-04-0997

ELECTRICAL ENGINEERING DEPARTMENT
UNIVERSITY OF SOUTHERN CALIFORNIA

Dec, 1991

Abstract : In this project, the cache memory of the DSP RISC is implemented by using 0.5 um CMOS technology. A simplified 1KB SRAM block with 6-transistor memory cell is bulit for the SPICE simulation. The clocking strategy used is a non-overlapping two-phase clock. It shows that the cache memory can work at 125MHz.

CONTENTS

1. Specification of DSP RISC Cache Memory

The cache memory part of the DSP RISC chip can be looked as an fast static random access memory with cache controller. Therefore the design of the cache memory can be simplified as sram design and leave the cache controller to the control unit of the DSP RISC chip. There are two kinds of cache memory in the process element : instruction cache memory and data cache memory. Their specifications are listed in the below :

	Instruction Cache Memory	Data Cache Memeory
Technology :	0.5um	0.5um
Memory Size :	1KB	2KB
Data Bus :	32 Bits	16 Bits
Address Bus :	16 Bits	16 Bits
Address Bits Needed :	8 Bits	10 Bits
Access Time :	5~10 ns	5~10 ns

The data cache memory can be divided into two 1KB SRAM blocks as same as the instruction cache. The difference between them is that instruction cache has more data bits but less address bits. Therefore this project will be concentrated on the design of the instruction cache memory (1 KB SRAM block) because it need less layout for SPICE simulation. From the result, we can estimate the design for the data cache memory too.

From the specification and references (seeAppendix A), a 64 word x 128 bit SRAM block is chose as the1 KB SRAM block of cache memory as shown in Fig 1.

2. Organization of SRAM

The organization of a static RAM block is shown in Fig 2. It can divided into 5 main parts as shown. The memory cells are organized in the form of a matrix. The address is decoded by two decoders - row decoder and column decoder int order to access the proper memory cells.

2.1 SRAM Cell :

The 6-transistor static CMOS memory cell is chose as the memory element. Although the cell size is larger than other type's, there are several advantages in the CMOS cell configuration, such as lower standby power dissipation and the wider operating margin of the cell.

In order to reduce the size and follow the MOSIS layout rules, the memory-cell matrix is formed by rows of memory array that are back to back to each other in order to reduce the space in between. The memory cell area is 5um x 10um. Once the memory cell size is set, the word-line height and bit-line height are also set. The word-line height is 5.5um and the bit-line height is 10.5um. The diagram of SRAM cell, sense amplifier and bit-line precharging circuit is shown in Fig 3.

2.2 Sense Amplifier :

To build a large memory array, the size of the meory cell must be as small as possible. By adding a sense amplifier in each column, the meory cell szie can be reduced since it will only sink current from the bit lines. The driving of the bit lines is done by a specially built flip-flop. Although the sense amplifier looks very similar to the flip-flop circuit of the meory cell, it is differnent from the memory cell with respect to the size of the transistors and the method of operation. The sense amplifiers only work during the read cycle. First, the bit-lines are precharged, then the meory cells to be read are open and the sense amplifiers are enabled. The differential sense amplifiers sense a samll difference between levels on the bit lines and amplify this to provide very fast sensing. In this project, a single-stage differntial sense amplifier is used.

2.3 Row Decoder :

In order to achieve high speed SRAM block (over 100 MHz), the dynamic row decoders are used. The dynamic row decoders are small, fast and relatively safe to design. The circuit diagram is shown in Fig 4. In this circuit a domino NOR gate does the decode. By using non-overlapping 2-phase clock, clocking precharge is during CLK1 = 1, and evaluation of the NOR gate is during CLK1 = 0. The second clocked stage is a dynamic NAND gate, which is precharged during CLK1 = 1 and evaluated during CLK2 = 1. This feed an inverter that drives the WORD line. The timing diagram is shown in Fig 5. This 2-phase clock is also used for precharging the bit line and internal data bus. Therefore it will also decide the access time of the SRAM block. In this project, the row decoder is put in the middle of the SRAM array. Two inverters are put on each side of the NOR gate and each drives 64 SRAM cells.

2.4 Column Decoder :

The column decoder is responsible for decoding the low-order address bits. Becasue the column decodr can has a long setup time, a multiplexer-type column decoder is used in this design. The ciruit diagram is shown in Fig 6. In order to speed up the response time, the internal data bus is first precharged. Then the column decoder select the proper bit lines to be connected with the internal data bus. In this project, the size of the column decoder is decided by the size of data bits.

2.5 Read/Write Control Circuit and Buffer

In order to connect with the system data bus, the chip-select and read/write control signals are provided for the contol unit of the PE. The circuit of R/W control and buffer is shown in Fig 7. The write operation is performed by setting CS = 0 and R/W = 0. Then the data will pass throght the transmission gate to the internal data bus. Then the row decoder and column decoder will select the proper SRAM cells to write in data. The read operation is performed by setting CS = 0 and R/W = 1. Fisrt the row decoder will select a particular row. Then the data will be output from the SRAM cell to the bit lines. The column decoder will select the bit lines that are needed and output them to the system data bus.

2.6 SRAM Block Floor Plan

By integrating all the above main parts and some other circuits, the complete SRAM block floor plan can be derived as shown in Fig 8. The feature of the SRAM block is listed in the following.

SRAM Matrix : 128 x 64 (1KB)
I/O : Chip_Select Read/Write Control System Timing Clock Address/Data Bus
Internal Signals : 2-Phase Clock Word-Line Select Signals Bit-Line Select Signals
Precharge Signals Sense Signal Internal Data/Address Bus

3. Simulation

Although we can simulate each part of the SRAM block to find out the response time for each. Then adds these response times together and estimates the access time of the SRAM block. But it seems to be not challenging to resolve the problem. Therefore I try to integrate all the parts of SRAM block together. Also we can get a floor plan from the experience of integration. A simplified SRAM block is layout in MAGIC for the simulation. It contains all the main parts of the SRAM block. In this way, the access time can be find out precisely.

3.1 Layout :

Before integrating all the parts together, a layout policy is defined for the routing. The Metal-1 is used in the horizontal direction while the Poly and Metal-2 are used in the vertical direction, except a few conditions. The layout is shown in Appendix B.

3.2 SPICE Simulation

The extracted SPICE file is converted to 0.5um technology by a converting program developed in C language. The SPICE simulation file is listed in Appendix C and the converting program is listed in Appendix D. The simulation is done by using the following parameters.

Tox = 1.1E-8 RSH = 0.0 LD = 0.0 CJ = 3.1146E-4 F/M
CJSW = 4.3777E-10 F/M Cox = 4 * 8.85 E-3 / 1.1 E-2 = 3.14 fF / um
Bit-Line Capacity Loading = (CJ * a * b + CJSW * (2*a + 2 *b)) * 31 = 127.84fF
Word-Line Capacity Loading = Cox * 8 * 2 * 63 = 197.82 fF
Column Decoder Capacity Loading = Cox * 20 * 2 * 31 = 243.35 fF
Read Control Line Capacity Loading = Cox * (24 * 2 + 5 * 2) = 352.86 fF
Write Control Line Capacity Loading = Cox * (20 * 2 + 12 * 2) = 389.36 fF

3.3 Simulation Input :

The input timing diagram of all the control signals of the simulation is show in Fig 9.

4. Results / Conclusions

4.1 SRAM Block Access Time :

From the above simulation, we can find out the access time for the SRAM block. The highest clock frequency that can be applied to is 125 MHz (8 ns). The SPICE output is shown in Fig 10 - 11. The capacity loading in the word-line and bit line confine the clock frequency. And the non-overlapping two phase clock limits the response time of the row decoder. The results in the below are all measured by logic "1" (3.5V) and logic "0" (1.5V) value.

4.2 Row Decoder Response Time :

The response time is measured from the beginning of the clock cycle to the time that tye word-line is driven to logic "1" . The response time is 4.72 ns.

4.3 SRAM Cell Read/Write Time :

The SRAM cell access time is measured from the beginning of the clock cycle to the time that the data is valid within the SRAM cell. The data output time is measured from the beginning of the clock cycle to the timw that the data output in the system data bus is valid.

	Time		Time
Write 0 :	5.68 ns	Read 0 :	5.22ns
Write 1 :	5.86 ns	Read 1 :	6.94ns

4.4 Area of the Cache Memory :

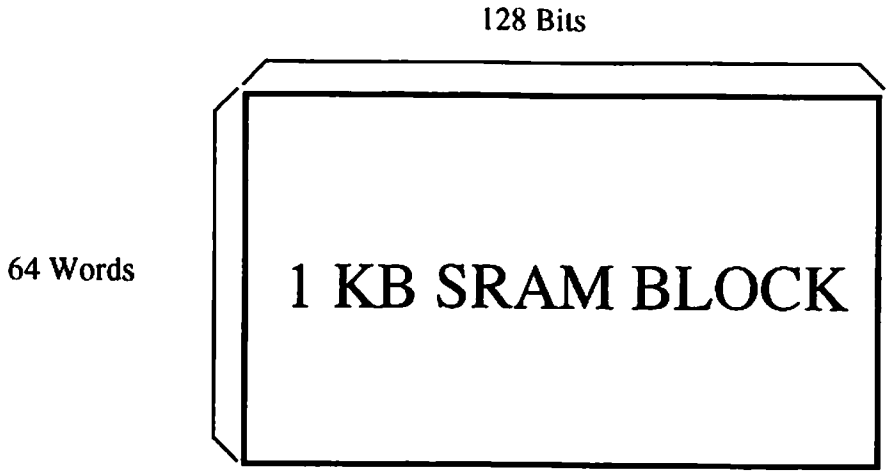
Data Cache Memory = 2 * (1565 * 900) um^2
Instruction Cache Memory = 1550 * 825 um^2

4.5 Access Timing Diagram of the Cache Memory :

The read/write cycle timing diagram is shown in Fig 12-13. We can find out that the timing diagram is similar to these of the commerical SRAM chips.

4.6 Dynamic Row Decoder :

The dynamic is small, fast and relatively safe, becasue it can tolerate the clock skew and addres skew during the CLK1 high period. But in this project, we find that it must be carefully used, because the precharge of the NAND gate and NOR gate will leak if the clock cycle is too long. The solution to this problem is that adopting higher frequency clocking or the size of the NAND gate and NOR gate must be carefully chose.



Fug1. 1 KB SRAM Block

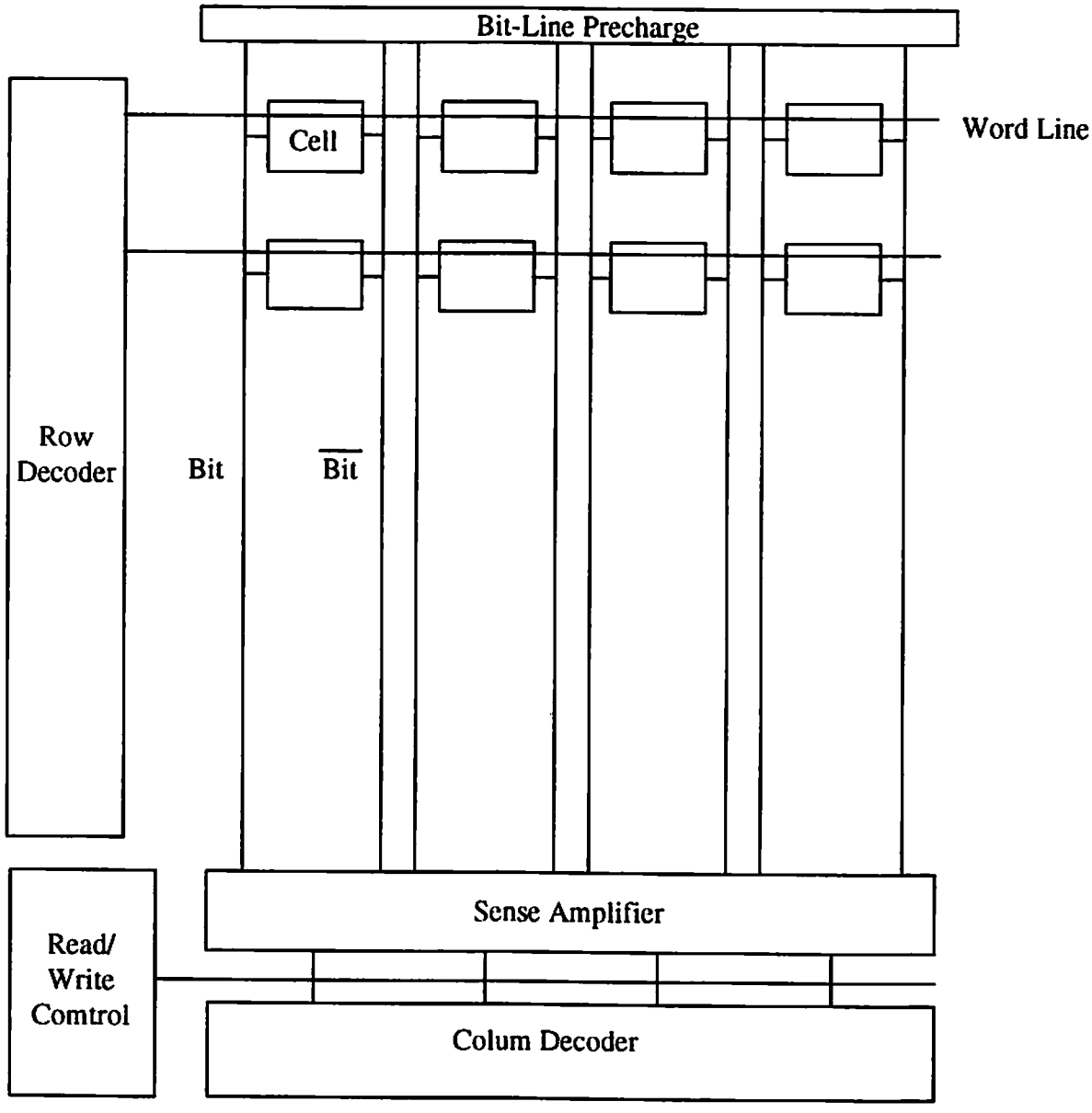


Fig 2. Organization of a static RAM.

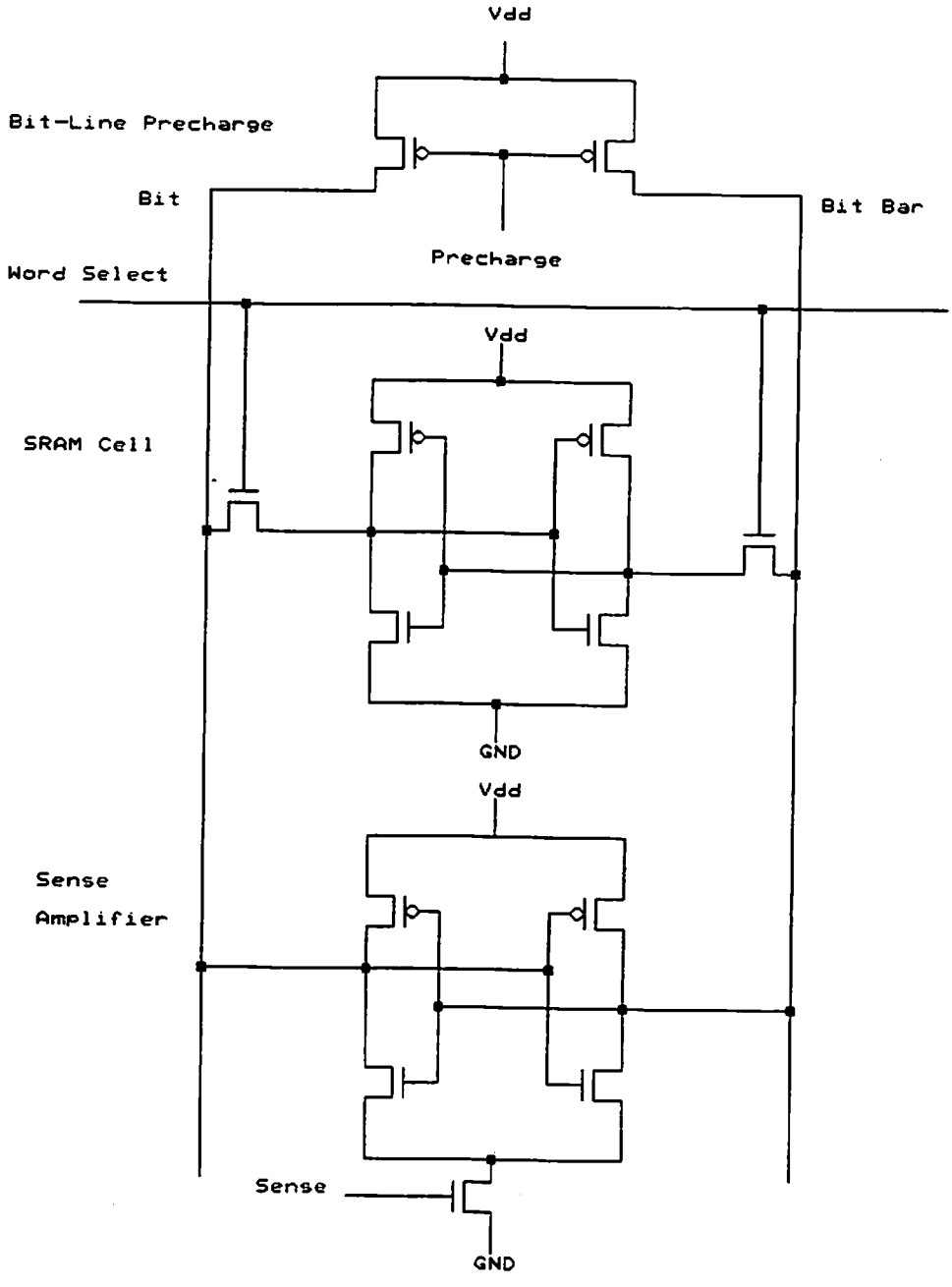


Fig 3. SRAM Cell, Sense Amplifier and Bit-Line Precharge Circuit Diagram

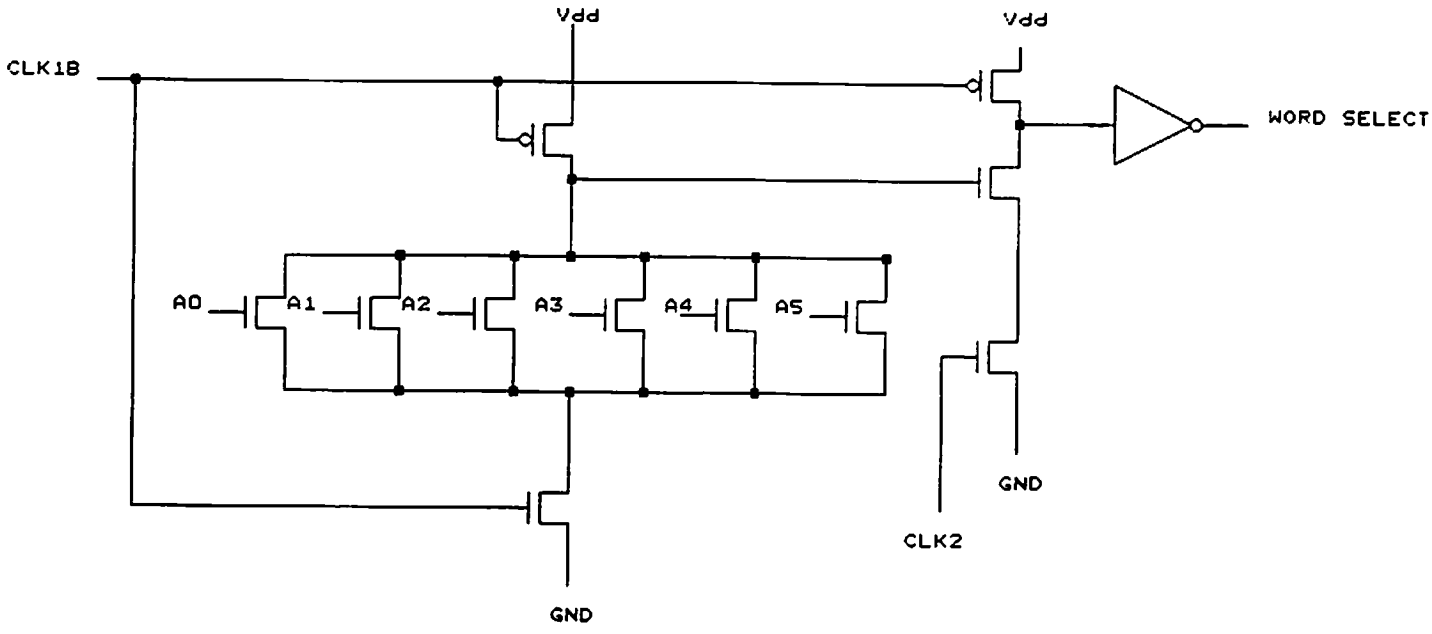


Fig 4. Dynamic Row Decoder

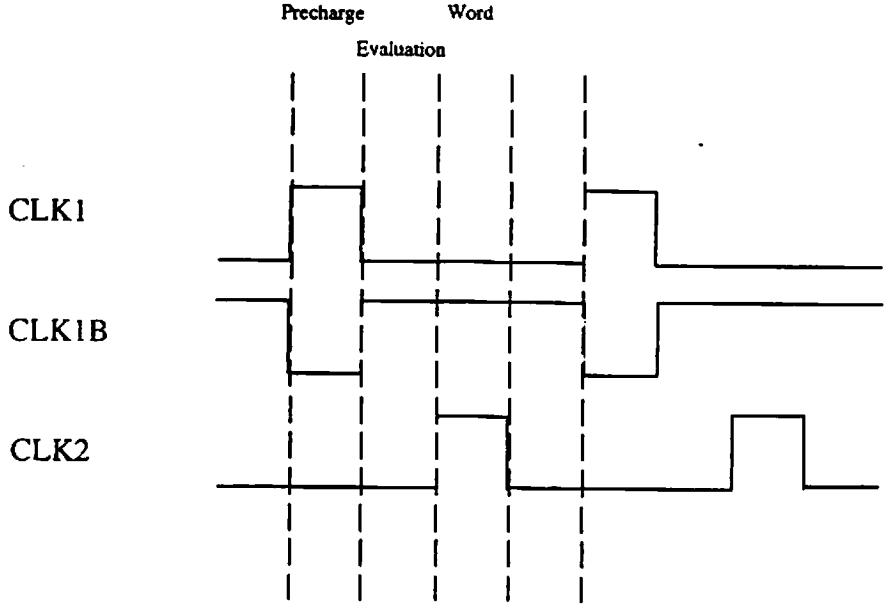


Fig 5. Timing Diagram for Dynamic Row Decoder

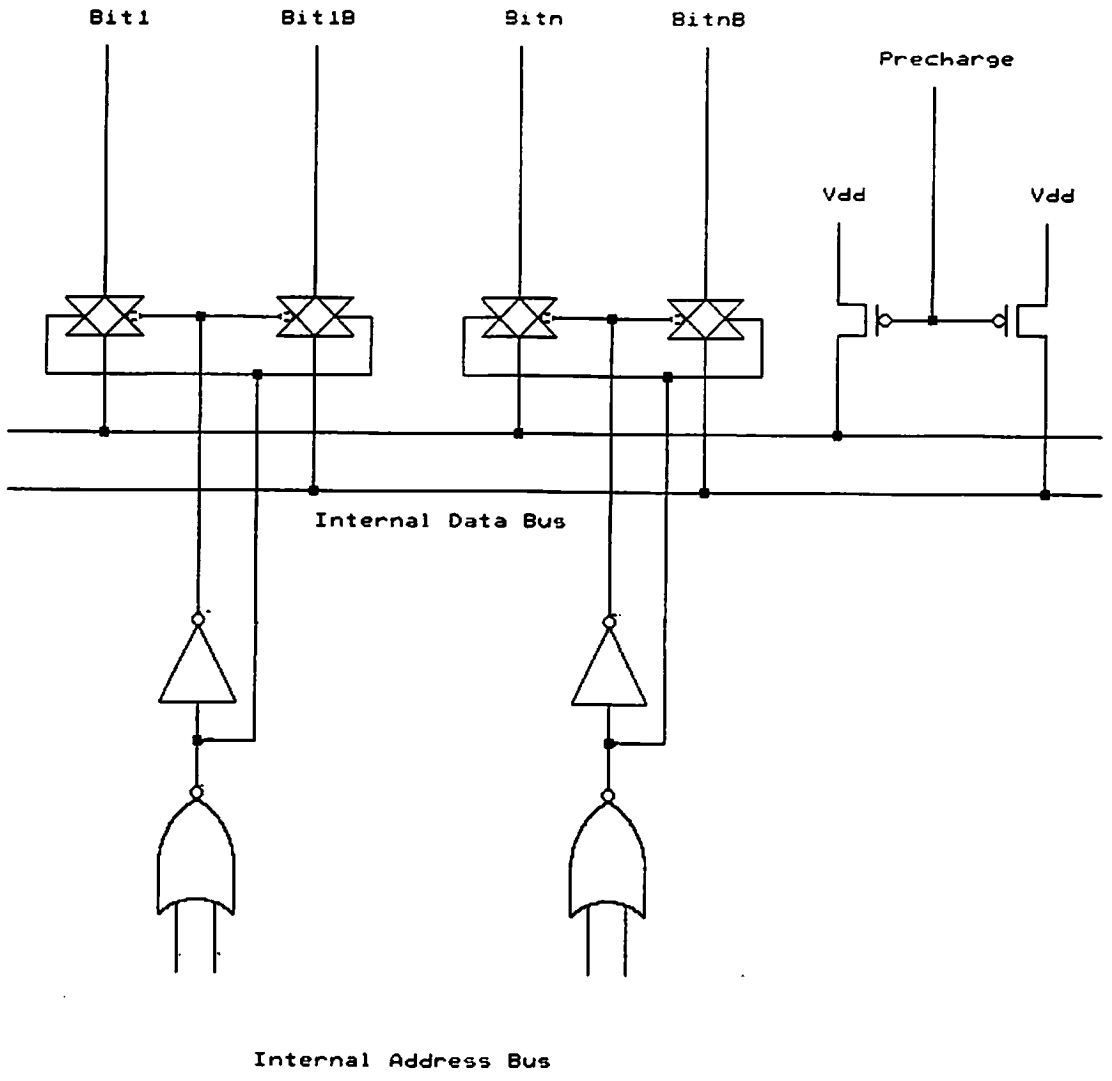


Fig 6. Column Decoder

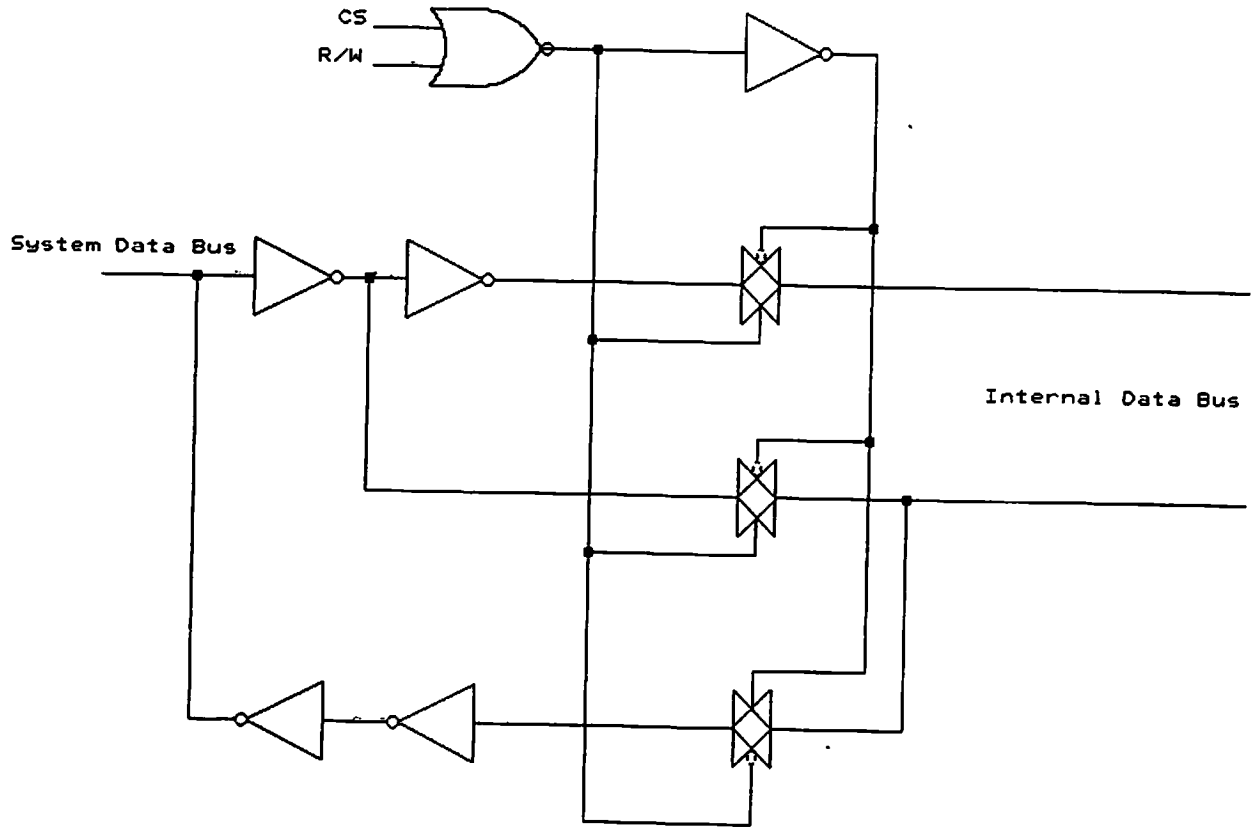


Fig 7. Read/Write Control Circuit and Buffer

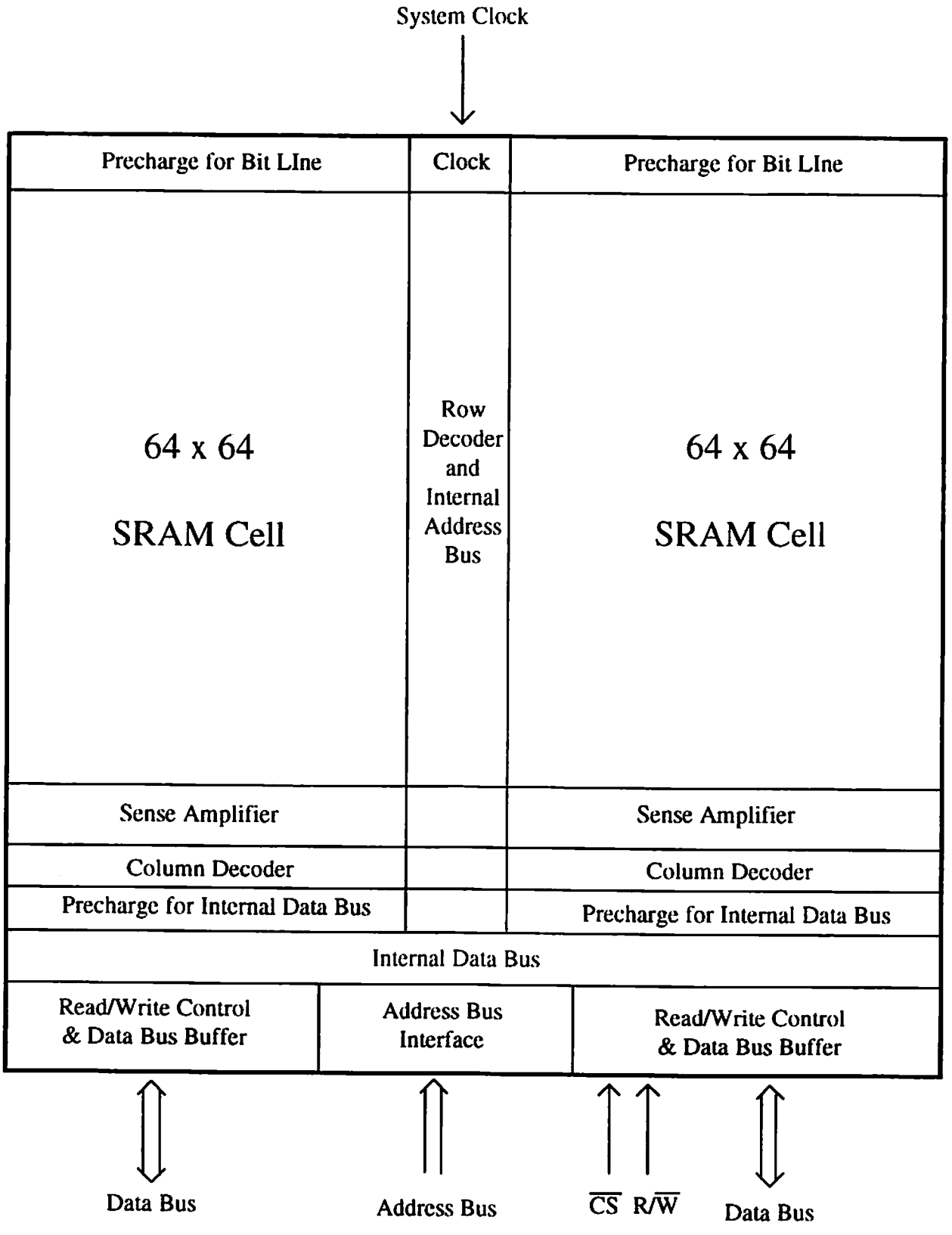


Fig 8. 1 KB SRAM-Block Floor Plan

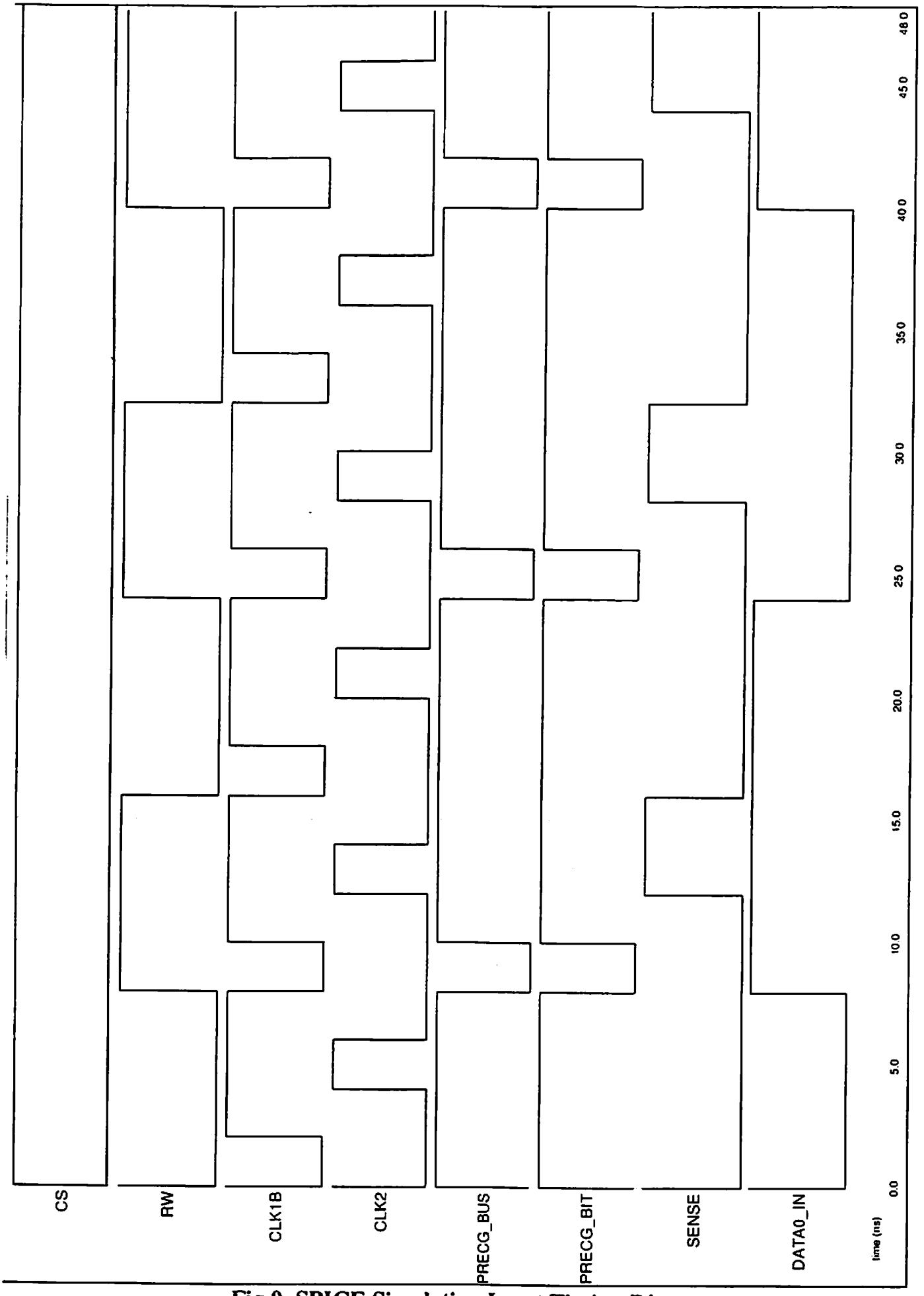


Fig 9. SPICE Simulation Input Timing Diagram

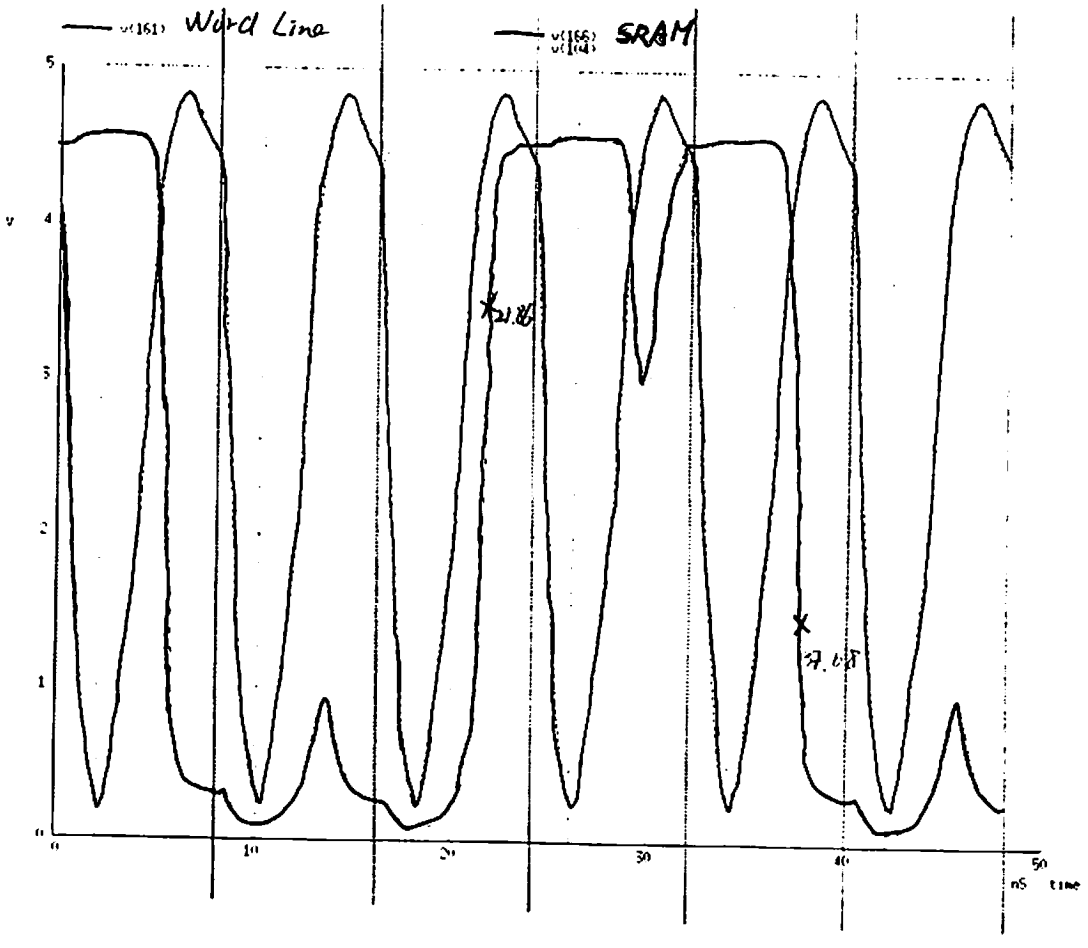


Fig 10 . SPICE Output Diagram of SRAM Write Operation

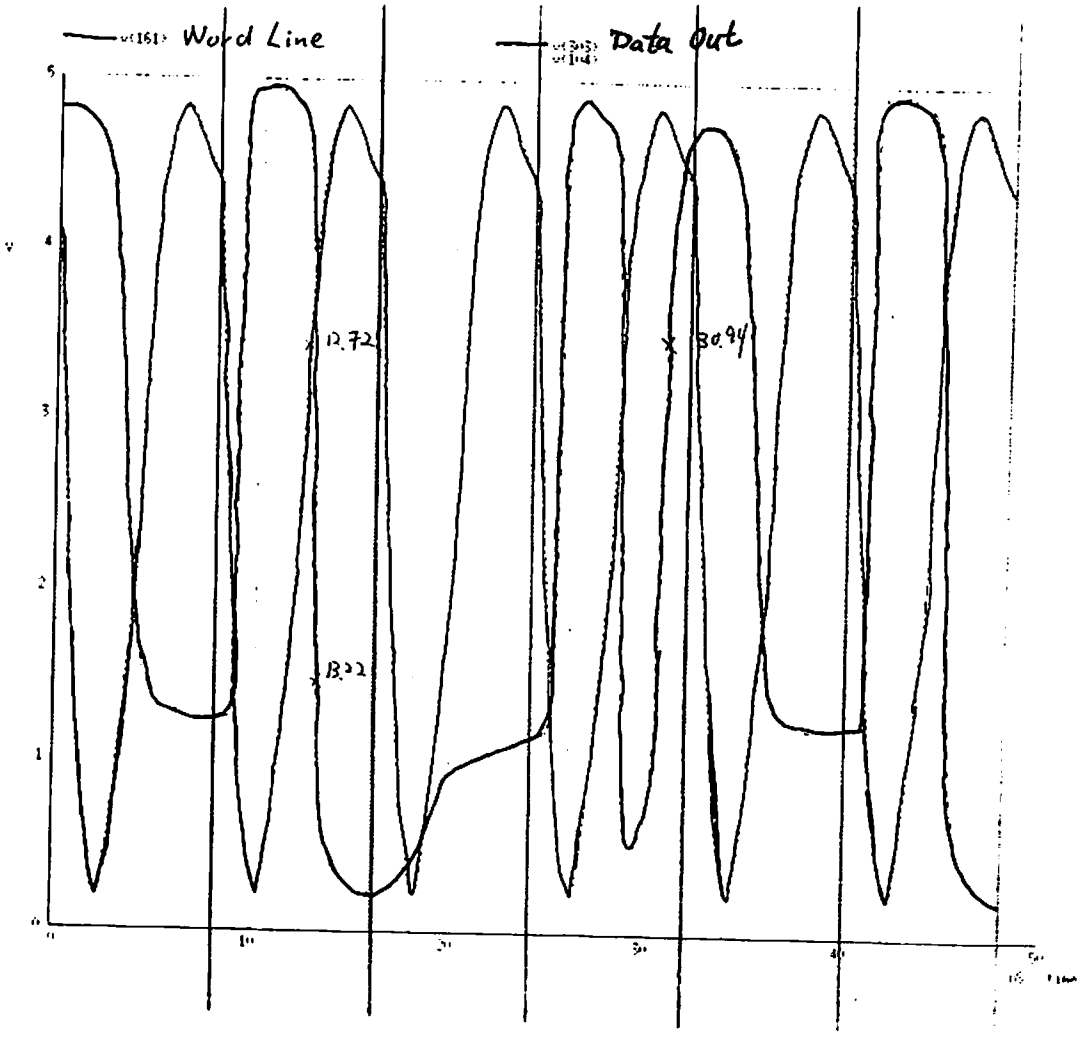


Fig 11. SPICE Output Diagram of SRAM Read Operation

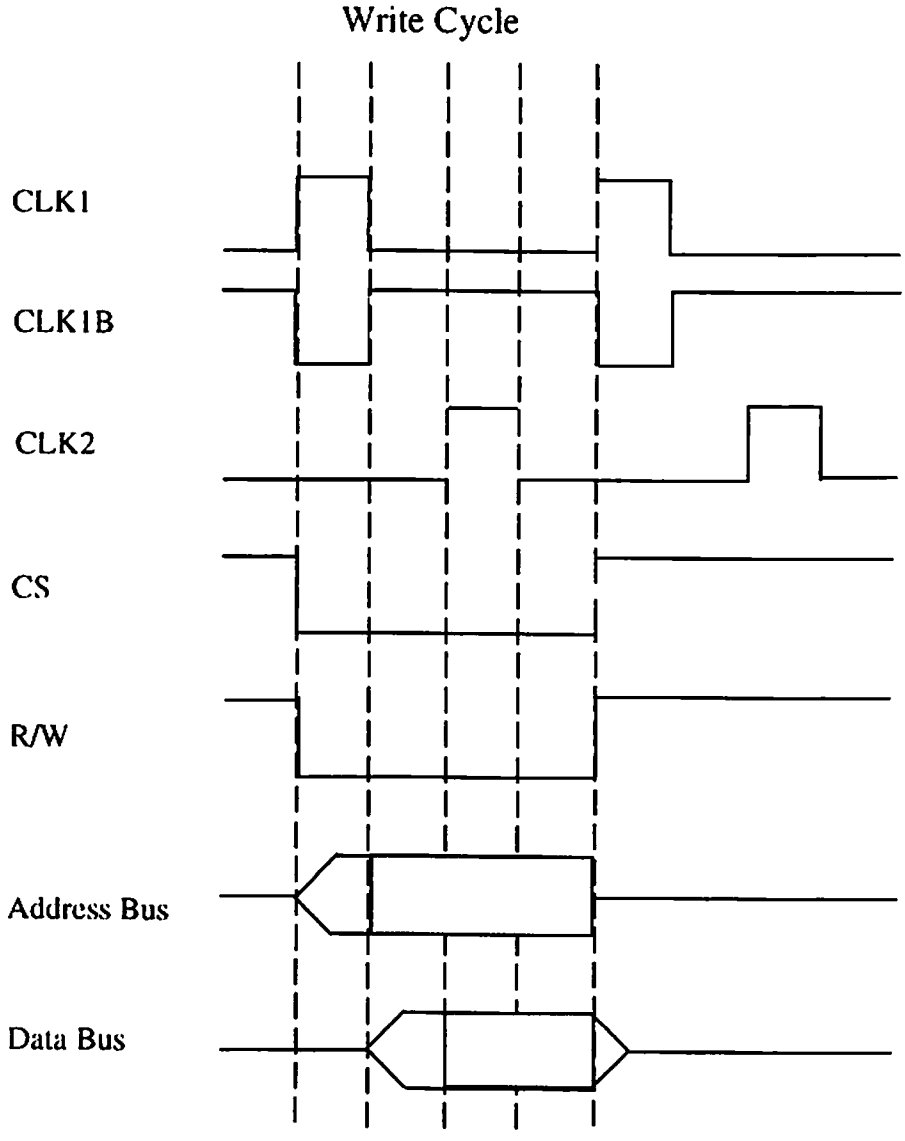


Fig 12. SRAM Block Write Cycle Timing Diagram

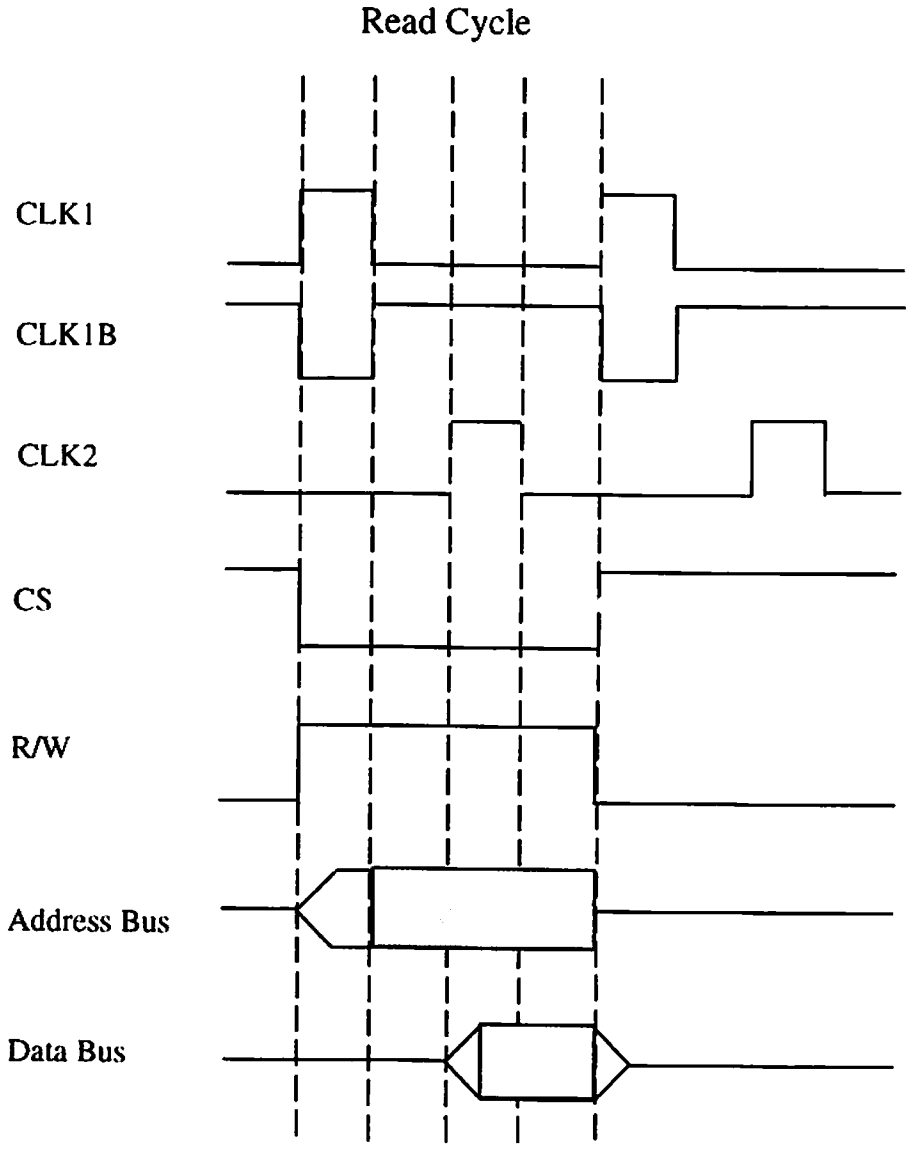


Fig 13. SRAM Block Read Cycle Timing Diagram

Appendix A.
References

1. N. Weste and K. Eshraghian, Principles of CMOS VLSI Design. Reading, Mass. : Addison-Wesley, 1985.

2. Amar Mukherjee, INTRODUCTION TO nMOS & CMOS VLSI Systems Design. Englewood Cl;iffs, N.J., 1986.

3. R.G. Stewart, "High Density CMOS ROM Arrays", IEEE J. Solid-State Circuits, vol. SC-12, pp. 502-506, Oct. 1977.

4. K. Ochii et al., "An Ultralow Power 8K x 8-Bit Full CMOS RAM with a siz-transistor Cell", IEEE J. Solid-State Circuits, vol. SC-17, pp. 798-803, Oct. 1982.

5. N. Okazaki et al., "A 16ns 2Kx8 Bit Full CMOS SRAM", IEEE J. Solid-State Circuits, vol. SC-19, pp. 552-556, Oct. 1984.

6. H. Okuyama et al., "A 7.5-ns 32K x 8 CMOS SRAM", IEEE J. Solid-State Circuits, vol. SC-23, pp. 1054-1059, Oct. 1988.

7. K. Sasaki et al., "A 9-ns 1-Mbit CMOS SRAM", IEEE J. Solid-State Circuits, vol. SC-24, pp. 1219-1225, Oct. 1989.

8. S.T. Flannagan et al., "8-ns CMOS 64K x 4 and 256K x 1 SRAMS", IEEE J. Solid-State Circuits, vol. SC-25, pp. 1049-1055, Oct. 1990.

Appendix B. The Simplified 1 KB SRAM Block Layout

```
** SPICE file is scaled from samp.spice 1.2um to s8.spice 0.5um.
** Resistor scale = 1.00000
** Capacitor scale = 0.17360

** SPICE file created for circuit samp
** Technology: scmos
**


.MODEL nfet NMOS LEVEL=3 PHI=0.600000 TOX=1.100E-08 XJ=0.200000U TPG=1
+ VTO=0.8186 DELTA=1.7570E+00 LD=0.0 KP=9.1547E-05
+ UO=596.5 THETA=1.0850E-01 GAMMA=0.5266 NSUB=1.9680E+16
+ NFS=5.5000E+12 VMAX=1.9420E+05 ETA=6.6540E-02 KAPPA=1.1210E-01
+ CGDO=2.6106E-10 CGSO=2.6106E-10 CGBO=6.3402E-10
+ CJ=3.1146E-04 MJ=1.0667 CJSW=4.3777E-10 MJSW=0.154230 PB=0.800000
* Weff = Wdrawn - Delta_W
* The suggested Delta_W is 1.9970E-07
.MODEL pfet PMOS LEVEL=3 PHI=0.600000 TOX=1.100E-08 XJ=0.200000U TPG=-1
+ VTO=-0.9456 DELTA=1.5520E+00 LD=0.0 KP=3.1646E-05
+ UO=206.2 THETA=1.6900E-01 GAMMA=0.4619 NSUB=1.5140E+16
+ NFS=4.9990E+12 VMAX=4.4410E+05 ETA=1.6350E-01 KAPPA=1.0000E+01
+ CGDO=2.6981E-11 CGSO=2.6981E-11 CGBO=8.6508E-10
+ CJ=4.7864E-04 MJ=0.4973 CJSW=1.4771E-10 MJSW=0.190593 PB=0.850000
*Weff = Wdrawn - Delta_W



** NODE: 0 = GND
** NODE: 1 = Vdd
** NODE: 2 = Error
RLUMP0 100 101 363.5
RLUMP1 102 103 121.0
M0 1 101 103 1 pfet L=0.5U W=2.25U
RLUMP2 104 105 3503.0
RLUMP3 100 106 363.5
M1 1 105 106 1 pfet L=0.5U W=2.25U
RLUMP4 107 108 1539.5
RLUMP5 104 109 3503.0
M2 108 109 1 1 pfet L=0.5U W=2.25U
RLUMP6 100 110 363.5
RLUMP7 102 111 121.0
M3 0 110 111 0 nfet L=0.5U W=1.5U
RLUMP8 112 113 125.0
RLUMP9 107 114 1539.5
RLUMP10 100 115 363.5
M4 113 114 115 0 nfet L=0.5U W=2.5U
RLUMP11 116 117 1264.0
RLUMP12 112 118 125.0
M5 0 117 118 0 nfet L=0.5U W=2.5U
RLUMP13 104 119 3503.0
RLUMP14 120 121 75.0
M6 0 119 121 0 nfet L=0.5U W=2.5U
RLUMP15 122 123 119.0
RLUMP16 107 124 1539.5
M7 0 123 124 0 nfet L=0.5U W=0.75U
RLUMP17 125 126 119.0
RLUMP18 107 127 1539.5
M8 0 126 127 0 nfet L=0.5U W=0.75U
RLUMP19 128 129 119.0
RLUMP20 107 130 1539.5
M9 0 129 130 0 nfet L=0.5U W=0.75U
RLUMP21 131 132 119.0
RLUMP22 107 133 1539.5
M10 0 132 133 0 nfet L=0.5U W=0.75U
RLUMP23 134 135 119.0
RLUMP24 107 136 1539.5
M11 0 135 136 0 nfet L=0.5U W=0.75U
RLUMP25 137 138 328.0
RLUMP26 139 140 1540.5
M12 1 138 140 1 pfet L=0.5U W=3.75U
RLUMP27 141 142 1540.5
RLUMP28 137 143 328.0
M13 142 143 1 1 pfet L=0.5U W=3.75U
RLUMP29 104 144 3503.0
RLUMP30 107 145 1539.5
M14 1 144 145 1 pfet L=0.5U W=2.25U
RLUMP31 146 147 388.5
RLUMP32 104 148 3503.0
M15 147 148 1 1 pfet L=0.5U W=2.25U
RLUMP33 149 150 119.0
RLUMP34 107 151 1539.5
M16 0 150 151 0 nfet L=0.5U W=0.75U
RLUMP35 152 153 75.0
RLUMP36 104 154 3503.0
M17 153 154 0 0 nfet L=0.5U W=2.5U
RLUMP37 155 156 125.0
RLUMP38 116 157 1264.0
M18 156 157 0 0 nfet L=0.5U W=2.5U
RLUMP39 146 158 388.5
RLUMP40 107 159 1539.5
RLUMP41 155 160 125.0
M19 158 159 160 0 nfet L=0.5U W=2.5U
RLUMP42 161 162 334.0
RLUMP43 146 163 388.5
M20 162 163 1 1 pfet L=0.5U W=3.0U
RLUMP44 164 165 369.5
RLUMP45 166 167 369.5
M21 1 165 167 1 pfet L=0.5U W=0.75U
RLUMP46 164 168 369.5
RLUMP47 166 169 369.5
M22 168 169 1 1 pfet L=0.5U W=0.75U
RLUMP48 166 170 369.5
RLUMP49 161 171 334.0
RLUMP50 139 172 1540.5
M23 170 171 172 0 nfet L=0.5U W=1.0U
RLUMP51 164 173 369.5
RLUMP52 166 174 369.5
M24 0 173 174 0 nfet L=0.5U W=1.5U
RLUMP53 164 175 369.5
RLUMP54 166 176 369.5
M25 175 176 0 0 nfet L=0.5U W=1.5U
RLUMP55 141 177 1540.5
RLUMP56 161 178 334.0
RLUMP57 164 179 369.5
M26 177 178 179 0 nfet L=0.5U W=1.0U
RLUMP58 161 180 334.0
RLUMP59 146 181 388.5
M27 180 181 0 0 nfet L=0.5U W=1.5U
RLUMP60 182 183 369.5
RLUMP61 184 185 187.5
RLUMP62 139 186 1540.5
M28 183 185 186 0 nfet L=0.5U W=1.0U
RLUMP63 187 188 369.5
RLUMP64 182 189 369.5
M29 0 188 189 0 nfet L=0.5U W=1.5U
RLUMP65 187 190 369.5
RLUMP66 182 191 369.5
M30 190 191 0 0 nfet L=0.5U W=1.5U
```

```
RLUMP67 141 192 1540.5
RLUMP68 184 193 187.5
RLUMP69 187 194 369.5
M31 192 193 194 0 nfet L=0.5U W=1.0U
RLUMP70 187 195 369.5
RLUMP71 182 196 369.5
M32 1 195 196 1 pfet L=0.5U W=0.75U
RLUMP72 187 197 369.5
RLUMP73 182 198 369.5
M33 197 198 1 1 pfet L=0.5U W=0.75U
RLUMP74 141 199 1540.5
RLUMP75 139 200 1540.5
M34 1 199 200 1 pfet L=0.5U W=1.0U
RLUMP76 141 201 1540.5
RLUMP77 139 202 1540.5
M35 201 202 1 1 pfet L=0.5U W=1.0U
RLUMP78 203 204 247.0
RLUMP79 141 205 1540.5
RLUMP80 139 206 1540.5
M36 204 205 206 0 nfet L=0.5U W=3.5U
RLUMP81 141 207 1540.5
RLUMP82 139 208 1540.5
RLUMP83 203 209 247.0
M37 207 208 209 0 nfet L=0.5U W=3.5U
RLUMP84 203 210 247.0
RLUMP85 211 212 390.5
M38 210 212 0 0 nfet L=0.5U W=2.75U
RLUMP86 211 213 390.5
RLUMP87 203 214 247.0
M39 0 213 214 0 nfet L=0.5U W=2.75U
RLUMP88 215 216 200.0
RLUMP89 217 218 288.5
M40 216 218 1 1 pfet L=0.5U W=5.0U
RLUMP90 219 220 1042.5
RLUMP91 221 222 307.0
RLUMP92 215 223 200.0
M41 220 222 223 1 pfet L=0.5U W=5.0U
RLUMP93 224 225 625.0
RLUMP94 219 226 1042.5
M42 225 226 1 1 pfet L=0.5U W=5.0U
RLUMP95 227 228 1529.0
RLUMP96 224 229 625.0
RLUMP97 139 230 1540.5
M43 228 229 230 1 pfet L=0.5U W=5.0U
RLUMP98 141 231 1540.5
RLUMP99 224 232 625.0
RLUMP100 233 234 1087.0
M44 231 232 234 1 pfet L=0.5U W=5.0U
RLUMP101 217 235 288.5
RLUMP102 219 236 1042.5
M45 0 235 236 0 nfet L=0.5U W=3.0U
RLUMP103 219 237 1042.5
RLUMP104 221 238 307.0
M46 237 238 0 0 nfet L=0.5U W=3.0U
RLUMP105 224 239 625.0
RLUMP106 219 240 1042.5
M47 239 240 0 0 nfet L=0.5U W=3.0U
RLUMP107 227 241 1529.0
RLUMP108 219 242 1042.5
RLUMP109 139 243 1540.5
M48 241 242 243 0 nfet L=0.5U W=2.5U
RLUMP110 141 244 1540.5
RLUMP111 219 245 1042.5
RLUMP112 233 246 1087.0
M49 244 245 246 0 nfet L=0.5U W=2.5U
RLUMP113 247 248 546.5
RLUMP114 227 249 1529.0
M50 1 248 249 1 pfet L=0.5U W=3.5U
RLUMP115 233 250 1087.0
RLUMP116 247 251 546.5
M51 250 251 1 1 pfet L=0.5U W=3.5U
RLUMP117 252 253 120.0
RLUMP118 254 255 243.5
M52 253 255 1 1 pfet L=0.5U W=3.0U
RLUMP119 256 257 986.5
RLUMP120 258 259 262.5
RLUMP121 252 260 120.0
M53 257 259 260 1 pfet L=0.5U W=3.0U
RLUMP122 261 262 587.0
RLUMP123 256 263 986.5
M54 262 263 1 1 pfet L=0.5U W=3.0U
RLUMP124 264 265 157.5
RLUMP125 261 266 587.0
RLUMP126 233 267 1087.0
M55 265 266 267 1 pfet L=0.5U W=3.0U
RLUMP127 268 269 861.5
RLUMP128 264 270 157.5
M56 1 269 270 1 pfet L=0.5U W=3.0U
RLUMP129 227 271 1529.0
RLUMP130 261 272 587.0
RLUMP131 268 273 861.5
M57 271 272 273 1 pfet L=0.5U W=3.0U
RLUMP132 274 275 508.5
RLUMP133 261 276 587.0
RLUMP134 227 277 1529.0
M58 275 276 277 0 nfet L=0.5U W=1.5U
RLUMP135 254 278 243.5
RLUMP136 256 279 986.5
M59 0 278 279 0 nfet L=0.5U W=1.5U
RLUMP137 256 280 986.5
RLUMP138 258 281 262.5
M60 280 281 0 0 nfet L=0.5U W=1.5U
RLUMP139 261 282 587.0
RLUMP140 256 283 986.5
M61 282 283 0 0 nfet L=0.5U W=1.5U
RLUMP141 264 284 157.5
RLUMP142 256 285 986.5
RLUMP143 233 286 1087.0
M62 284 285 286 0 nfet L=0.5U W=1.5U
RLUMP144 268 287 861.5
RLUMP145 264 288 157.5
M63 0 287 288 0 nfet L=0.5U W=1.5U
RLUMP146 227 289 1529.0
RLUMP147 256 290 986.5
RLUMP148 268 291 861.5
M64 289 290 291 0 nfet L=0.5U W=1.5U
RLUMP149 274 292 508.5
RLUMP150 256 293 986.5
RLUMP151 227 294 1529.0
M65 292 293 294 1 pfet L=0.5U W=3.0U
RLUMP152 295 296 412.5
RLUMP153 297 298 506.0
M66 0 296 298 0 nfet L=0.5U W=2.5U
RLUMP154 268 299 861.5
RLUMP155 297 300 506.0
M67 299 300 0 0 nfet L=0.5U W=2.5U
RLUMP156 301 302 543.5
RLUMP157 303 304 225.5
```

```
M68 0 302 304 0 nfet L=0.5U W=2.5U
RLUMP158 301 305 543.5
RLUMP159 274 306 508.5
M69 305 306 0 0 nfet L=0.5U W=2.5U
RLUMP160 295 307 412.5
RLUMP161 297 308 506.0
M70 1 307 308 1 pfet L=0.5U W=5.0U
RLUMP162 268 309 861.5
RLUMP163 297 310 506.0
M71 309 310 1 1 pfet L=0.5U W=5.0U
RLUMP164 301 311 543.5
RLUMP165 303 312 225.5
M72 1 311 312 1 pfet L=0.5U W=5.0U
RLUMP166 301 313 543.5
RLUMP167 274 314 508.5
M73 313 314 1 1 pfet L=0.5U W=5.0U
C0 315 0 4F
** NODE: 315 = A1B ===FLOATING===
C1 316 0 4F
** NODE: 316 = A0B ===FLOATING===
** NODE: 317 = 8_318_23# ===FLOATING===
** NODE: 318 = 8_274_23# ===FLOATING===
C2 319 0 9F
** NODE: 319 = A7B ===FLOATING===
C3 320 0 9F
** NODE: 320 = A6B ===FLOATING===
C4 321 0 9F
** NODE: 321 = A5B ===FLOATING===
C5 322 0 9F
** NODE: 322 = A4B ===FLOATING===
C6 323 0 9F
** NODE: 323 = A3B ===FLOATING===
C7 324 0 9F
** NODE: 324 = A2B ===FLOATING===
C8 303 0 8F
** NODE: 303 = DATA0_OUT
C9 301 0 7F
** NODE: 301 = 6_378_879#
C10 297 0 8F
** NODE: 297 = 6_296_875#
** NODE: 295 = DATA0_IN
C11 274 0 5F
** NODE: 274 = 6_394_785#
C12 264 0 3F
** NODE: 264 = 6_318_725#
C13 252 0 2F
** NODE: 252 = 6_222_691#
C14 268 0 12F
** NODE: 268 = 6_330_729#
C15 256 0 399F
** NODE: 256 = READ
C16 261 0 360F
** NODE: 261 = WRITE
C17 258 0 2F
** NODE: 258 = RW
C18 254 0 1F
** NODE: 254 = CS
C19 247 0 10F
** NODE: 247 = PRECG_BUS
C20 233 0 25F
** NODE: 233 = DATA0B
C21 227 0 29F
** NODE: 227 = DATA0
C22 215 0 4F
** NODE: 215 = 6_186_361#
C23 219 0 257F
** NODE: 219 = COL1
C24 224 0 497F
** NODE: 224 = COL1B
C25 221 0 5F
** NODE: 221 = A0
C26 217 0 5F
** NODE: 217 = A1
C27 211 0 10F
** NODE: 211 = SENSE
C28 203 0 4F
** NODE: 203 = 6_294_243#
C29 182 0 3F
** NODE: 182 = SRAM2
C30 187 0 3F
** NODE: 187 = SRAM2B
C31 184 0 2F
** NODE: 184 = WORD2R
C32 161 0 205F
** NODE: 161 = WORD1R
** NODE: 155 = 6_148_37#
** NODE: 152 = 6_114_37#
C33 166 0 3F
** NODE: 166 = SRAM1
C34 164 0 3F
** NODE: 164 = SRAM1B
C35 146 0 4F
** NODE: 146 = DEC1R
C36 141 0 147F
** NODE: 141 = BIT1B
C37 139 0 147F
** NODE: 139 = BIT1
** NODE: 120 = 6_185_37#
** NODE: 112 = 6_215_37#
C38 149 0 9F
** NODE: 149 = A7
C39 134 0 9F
** NODE: 134 = A6
C40 131 0 9F
** NODE: 131 = A5
C41 128 0 9F
** NODE: 128 = A4
C42 125 0 9F
** NODE: 125 = A3
C43 122 0 9F
** NODE: 122 = A2
** NODE: 0 = GND!
C44 107 0 13F
** NODE: 107 = DEC2
C45 1 0 153F
** NODE: 1 = Vdd!
C46 102 0 4F
** NODE: 102 = WORD1L
C47 100 0 4F
** NODE: 100 = DEC1L
C48 104 0 17F
** NODE: 104 = CLK1B
C49 116 0 10F
** NODE: 116 = CLK2
C50 137 0 10F
** NODE: 137 = PRECG_BIT
```

```
Vdd   1 0 DC 5V
Vp1 247 0 DC 5V PULSE (5 0 8.5ns 100ps 100ps 2ns 16ns)
Vp2 137 0 DC 5V PULSE (5 0 8.5ns 100ps 100ps 2ns 16ns)
VCS 254 0 DC 0V
VRW 258 0 DC 5V PULSE (5 0 0 100ps 100ps 8ns 16ns)
VSE 211 0 DC 5V PULSE (0 5 12ns 100ps 100ps  4ns 16ns)
VDB 295 0 DC 5V PULSE (0 5  8ns 100ps 100ps 16ns 32ns)

VW2 184 0 DC 0V

VA7 149 0 DC 0V
VA6 134 0 DC 0V
VA5 131 0 DC 0V
VA4 128 0 DC 0V
VA3 125 0 DC 0V
VA2 122 0 DC 0V
VA1 217 0 DC 0V
VA0 221 0 DC 0V

VA7B 319 0 DC 5V
VA6B 320 0 DC 5V
VA5B 321 0 DC 5V
VA4B 322 0 DC 5V
VA3B 323 0 DC 5V
VA2B 324 0 DC 5V
VA1B 315 0 DC 5V
VA0B 316 0 DC 5V

Vclk1b 104 0 DC 5V PULSE (5 0   0 100ps 100ps 2ns 8ns)
Vclk2  116 0 DC 5V PULSE (0 5 4ns 100ps 100ps 2ns 8ns)

.TRAN 1ns 48ns
.OPTIONS GMIN=5E-6
.END
```

scale.c


```
/* program to convert 1.2um SPICE file to 0.5um */
/*          Author : Shen-Te Hong                */
/*              Oct 21, 1991                     */

#include <stdio.h>
#include <math.h>
#include <string.h>

#define MAXSTRLEN 255

void fet (char line[], float ot, float nt);
void resistor (char line[], float scale);
void capacitor (char line[], float scale);
int readaline (FILE *fp, char line[]);


void main()
{
  FILE *infp, *outfp;
  char line[255], infname[50], outfname[50];
  float cscale, rscale, ot, nt;

  printf ("SPICE File Scaler V2.0\n");
  printf ("Input SPICE file name : ");
  scanf ("%s", infname);

  infp = fopen (infname,"r");

  if (infp == NULL) {
    printf ("File not found!\n");
    exit (1);
  }

  printf ("Output SPICE file name : ");
  scanf ("%s", outfname);

  outfp = fopen (outfname,"w");


  printf ("Input present technology (um) : ");
  scanf ("%f", &ot);

  printf ("Input new technology (um) : ");
  scanf ("%f", &nt);


  printf ("Input resistor scale : ");
  scanf ("%f", &rscale);

  printf ("Input capacitor scale : ");
  scanf ("%f", &cscale);

  printf ("SPICE file is scaled from %s %3.1fum to %s %3.1fum.\n",
     infname, ot, outfname, nt);
  printf ("Resistor scale = %7.5f\n", rscale);
  printf ("Capacitor scale = %7.5f\n", cscale);

  fprintf (outfp,"** SPICE file is scaled from %s %3.1fum to %s %3.1fum.\n",
     infname, ot, outfname, nt);
  fprintf (outfp,"** Resistor scale = %7.5f\n", rscale);
  fprintf (outfp,"** Capacitor scale = %7.5f\n", cscale);
  fprintf (outfp,"\n");

  while (!feof(infp)) {
    if (readaline (infp, line)) {
      printf ("Error! String too long ");
      printf ("or SPICE file doesn't end with .END or .end.\n");
      break;
    }


    if ((strcmp (line,".END") == 0) ||
        (strcmp (line,".end") == 0)) {
      fprintf (outfp,"%s\n", line);
      printf ("Done.....\n");
      break;
    }


    switch (line[0]) {
      case 'R' :  resistor (line, rscale);
                  break;

      case 'C' :  capacitor (line, cscale);
                  break;

      case 'M' :  fet (line, ot, nt);
                  break;

      default  :  break;
    }
    fprintf (outfp,"%s\n", line);

  }

  fclose (infp);
  fclose (outfp);
}



int readaline (FILE *fp, char line[])
{
  int c;
  int i=0;

  while ((c=fgetc(fp)) != '\n') {
    line[i++] = c;
    if (i > MAXSTRLEN)
      return (1);
  }

  line[i] = '\0';
  return (0);
}



void fet (char line[], float ot, float nt)
{
  int len;
  int i,j,count6=0, count7, space_no=0, digit;
  char temp[255], format[255];
  float r;

  len = strlen (line);

  while (space_no < 6)
```

```
    if (line[count6++] == ' ')
      space_no++;

  count7=count6+1;
  while (space_no <7)
    if (line[count7++] == ' ')
      space_no++;

  for (i=count7+2, j=0; i<=len-2; i++, j++)
    temp[j] = line[i];
  temp[j] = '\0';

  r = (float) atof (temp) * nt / ot;
  digit = log10((double) r);
  if (digit < 1) digit = 3;
  else digit +=3;

  sprintf (format,"L=%c3.1fU W=%c%d.%dfU", '%','%',digit,1);
  sprintf (temp,format, nt, r);
  line[count6]='\0';
  strcat (line, temp);
}


void resistor (char line[], float scale)
{
  int len;
  int i,j,count=0, space_no=0, digit;
  char temp[255], format[255];
  float r;

  len = strlen (line);

  while (space_no < 3)
    if (line[count++] == ' ')
      space_no++;

  for (i=count, j=0; i<=len; i++, j++)
    temp[j] = line[i];
  temp[j] = '\0';

  r = (float) atof(temp) * scale;
  digit = log10((double) r);
  if (digit < 1) digit = 3;
  else digit +=3;

  sprintf (format,"%c%d.%df", '%',digit,1);
  sprintf (temp,format,r);
  line[count] = '\0';
  strcat (line, temp);
}


void capacitor (char line[], float scale)
{
  int len;
  int i,j,count=0, space_no=0;
  char temp[255];
  int t;

  len = strlen (line);

  while (space_no < 3)
    if (line[count++] == ' ')
      space_no++;

  for (i=count, j=0; i<=len-2; i++, j++)
    temp[j] = line[i];
  temp[j] = '\0';

  t = atoi (temp) * scale;
  sprintf (temp,"%dF",t);
  line[count]='\0';
  strcat (line, temp);
}
```

DSP RISC PE Floor Plan

Shen-Te Hong

* Dealy / Area

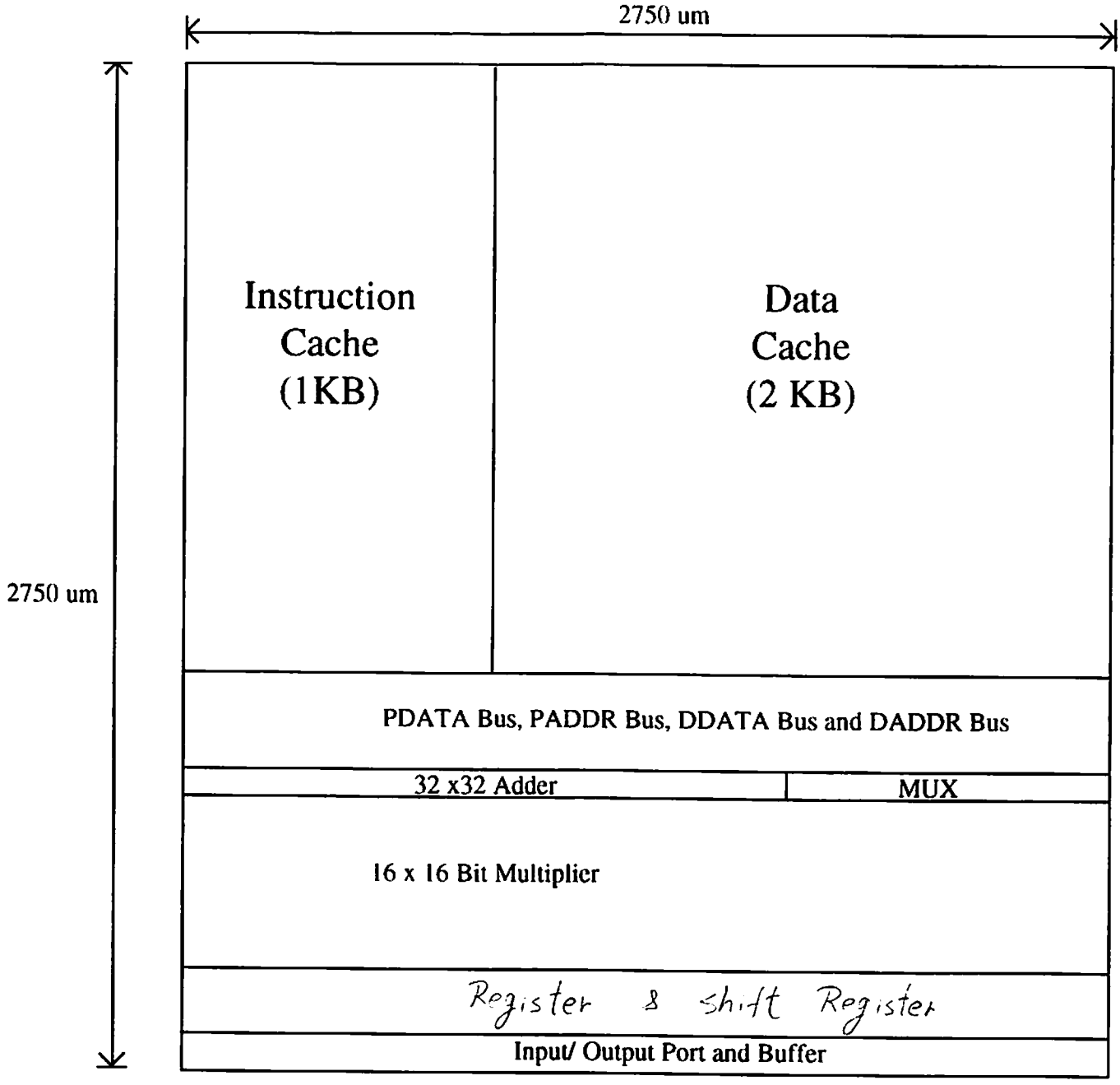
	Adder	Multiplier	Data Cache	Instruction Cache	I/O Port and Buffer
Delay	2.63 ns	4.3 ns	8 ns	8 ns	4.8 ns
Area (lambda= 0.25 um)	200 x 6920	3500 x 10000	7200 x 6260	6200 x 3300	3250 x 88

* Clock rate : The clock rate for the DSP RISC PE is 125 MHz according to the delay of the Cache Memory

* Pin : Clock : 1
CD : 16
CA : 16
$\overline{\text{CREQ}}$: 1
$\overline{\text{CACK}}$: 1
CSTROP : 1
Vdd : 9
GND : 9

Total Pins : 54

* Floor Plan : in the following page.



Clock Rate = 125 MHz
Area = 7.56 mm^2
Pins = 54

DSP RISC PE Floor Plan

EE 577 Term Project Report

RISC PROCESSING ELEMENT DESIGN

Name : Weili Wang

ID : 888-05-7693

Abstract ---The implementation of the Register ,Shift Register, Multiplexer, and I/O Buffer of Processor element is considered . The registers are implemented by 2-phase static D flip-flop to prevent clock race problem . The output buffers is implemented by cascaded stage buffer to achieve reasonable speed as normal output load is considered. The results are simulated by SPICE3 and layout area are shown.

1.0 Register implementation

The Register file is constructed by juxtaposing 32 bits two phase static D flip flops. The shift registers in the communication I/O port is implemented by cascade 21 bits D flip-flops.

1.1 Two phase static D flip-flop

Although it takes more transistors (14 transistors) to implement the D flip-flop containing a gated R-S flip-flop than other design , it uses only one clock phase and is clock race immune. The circuit is shown in figure 1.1 and its speed is Tpd =(.125 + .285) / 2 = .155 ns. The SPICE3 simulation result is shown in figure 1.2.

1.2 Register file

The layout floor plan is shown in figure 1.3. The total area is 101 * 2826 λ^2

1.3 Shift register

Construct 21-b shift register by 21 D flip-flops in serial , the floor planning is shown in figure 1.4 and the total area is 101 * 1848 λ^2

2.0 Multiplexer

The multiplexer is implemented by transmission gate, the circuit is shown in figure 2.1. The floor plan is shown in figure 2.2. The SPICE3 simulation result is shown in figure 2.3 .The total area of 16 bit multiplexer is 36 * 960 λ^2.

The Speed : Tpd = .05 ns

3.0 Cascaded Stage Output Buffer

To achieve acceptible output response time when driving outside chip capacitance load , cascaded output buffer must be used as shown in figure 3.1.

Let α represents Stage Ratio $\alpha = (C_{Load}/C_L)^{1/n}$

Assume Capcitance load for the output signal C_{Load} = 10 pf and let

$C_L = 2\ C_G = 16*\lambda^2*C_{ox}$

$C_{ox} = \varepsilon_{SiO2} * \varepsilon_0 / T_{OX} = 4 * 8.854 * 10^{-12} / (110 * 10^{-10}) = 32 * 10^{-4}$ pf/μm^2

C_L = 3.2 fF C_{Load}/C_L = 3000

If we Select α = 5 , then total 5 stages will be needed to achieve the best propagation delay.

The cascaded output buffer with 4 stages and stage ratio of 5 is simulated. The area for one cascaded stage buffer is 88 * 315 λ^2 and the speed Tpd is 2.6 ns when C_{LOAD} is 1 PF and 4.8 ns when C_{LOAD} is 10 PF. The simulation result is shown in figure 3.2

4.0 Tri-State Input / Output Buffer

Combining the cascaded output buffer into the tri-state input/output buffer as the circuit shown in figure 4.1 , we construct the tri-state input/output buffer for the 16 bits addresses and data signals also three communication signals. The total area is 88 * 5250 λ^2 for the all 35 signal lines.

Conclusion

To achieve high speed Processing element of RISC machine , not only the operation unit must gain high speed by using submicron technology

and good algorithm but also the communication I/O port driving capability and response time must be enhanced.

Reference

1. N. Weste " Principle of CMOS VLSI Design A Systems Perspective" Massachusetts, Addison-Wesley, 1984, p126-p127,p215, p227-229

2. Randell L. Geiger " VLSI Design Techniques for Analog and Digital Circuits" Newyork, McGraw-Hill , 1990, p590-p593

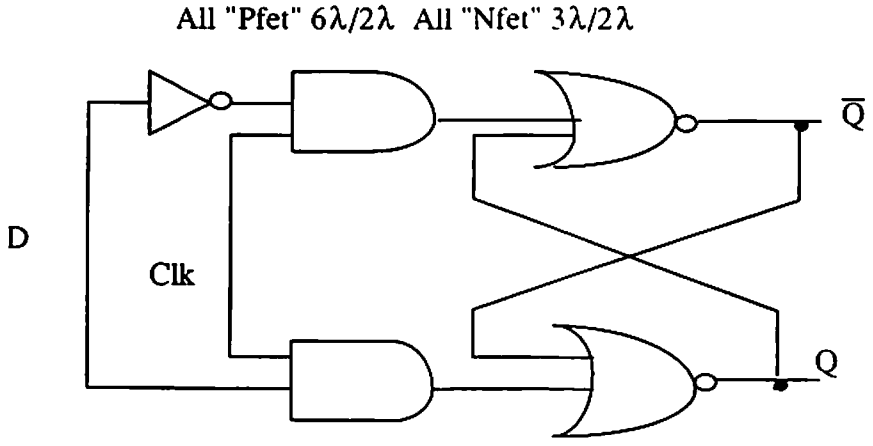


Figure 1.1 Two phase D flip flop circuit

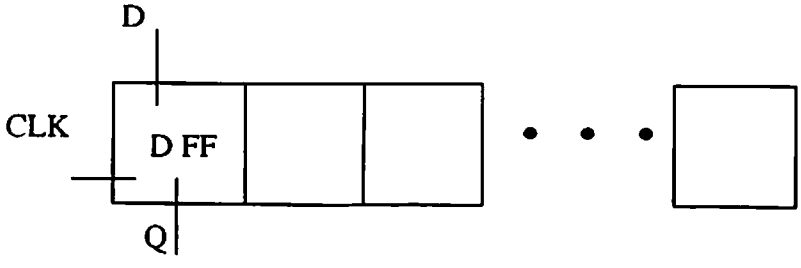


Figure 1.3 Register file floor plan

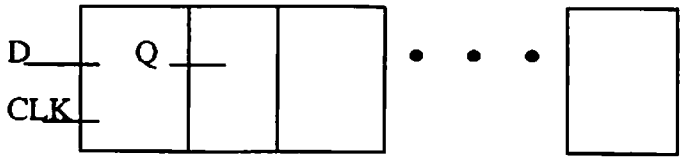


Figure 1.4 Shift register floor plan

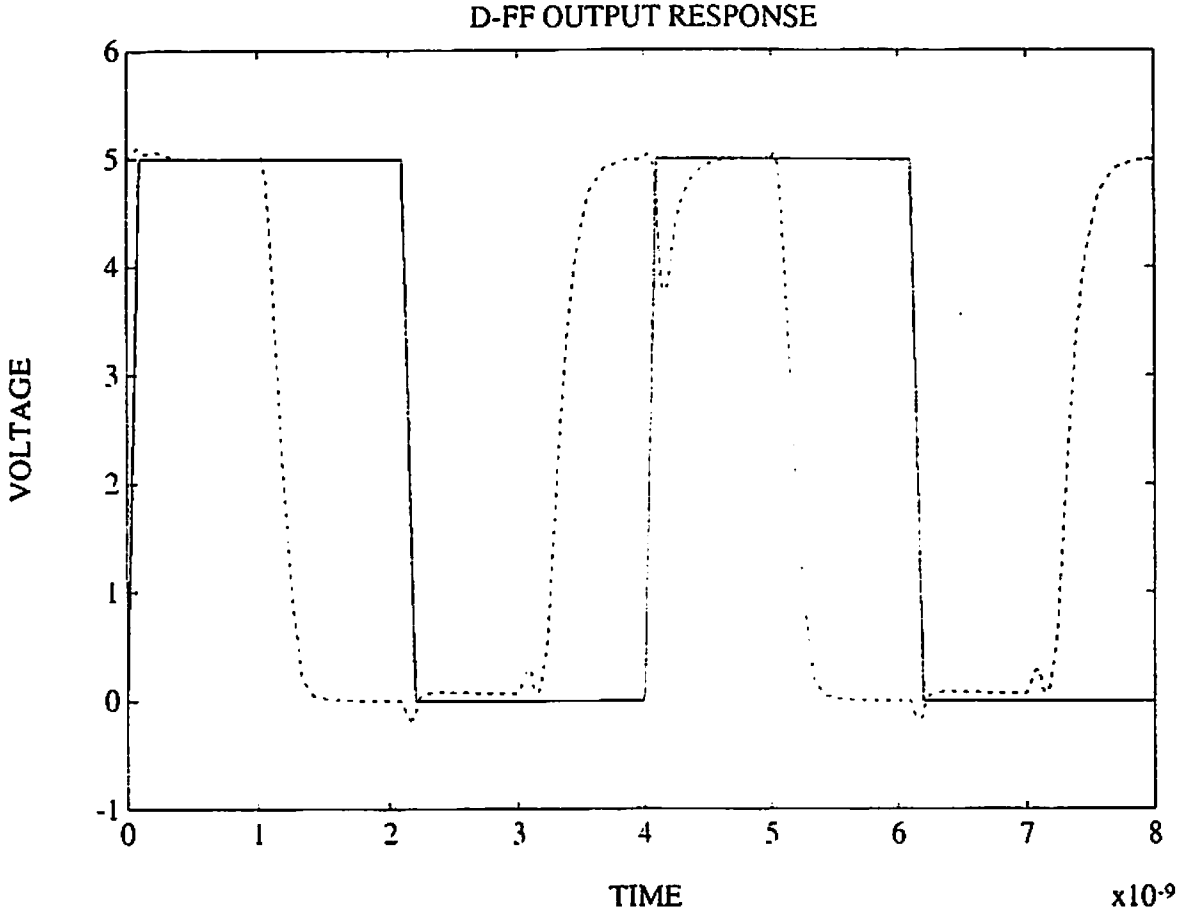
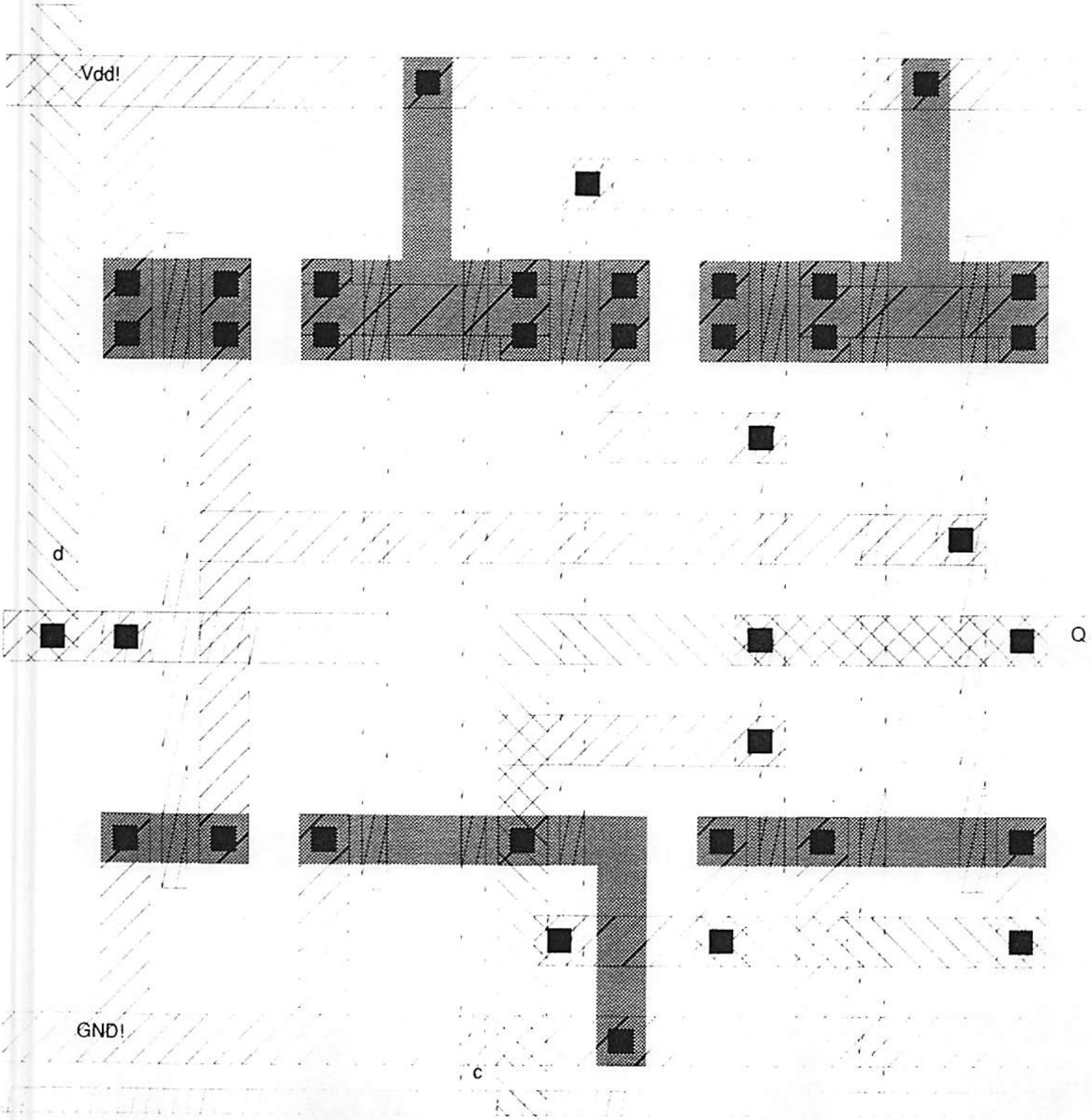


Figure 1.2 Spice simulation result of D flip flop output response

Figure 1.5 Two phase static D flip flop layout



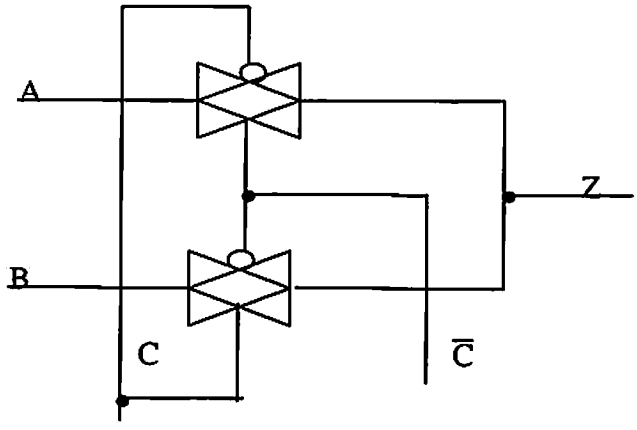


Figure 2.1 Multiplexer circuit

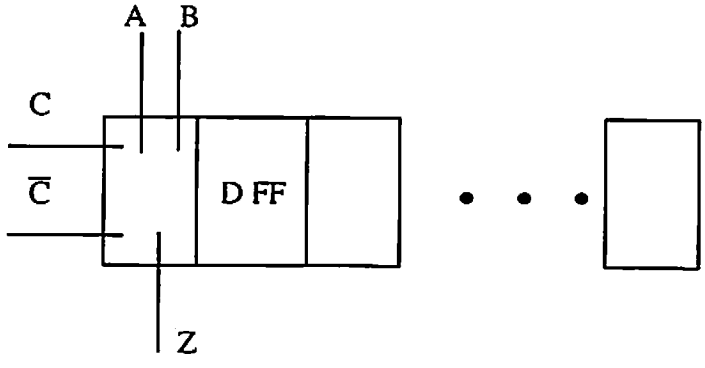


Figure 2.2 16-b multiplexer floor plan

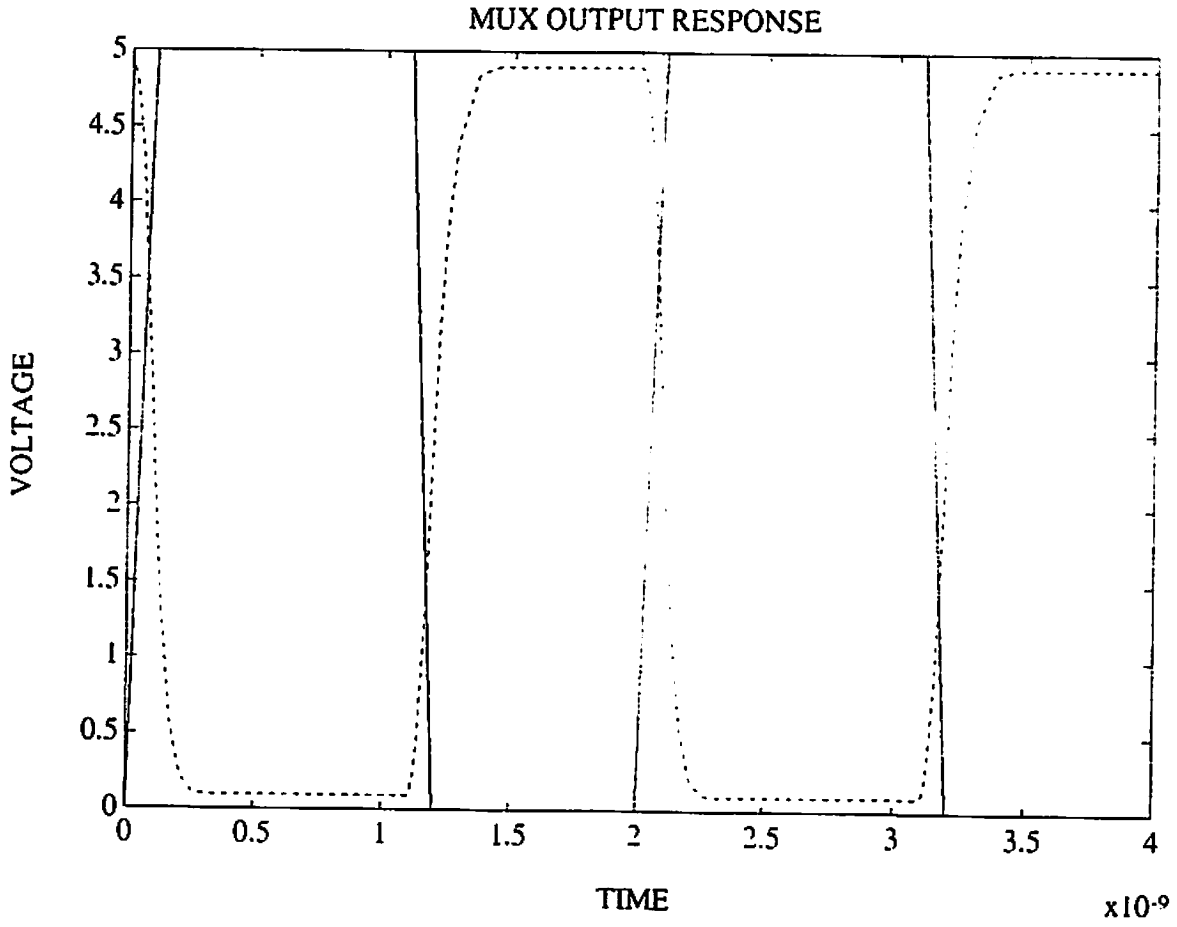
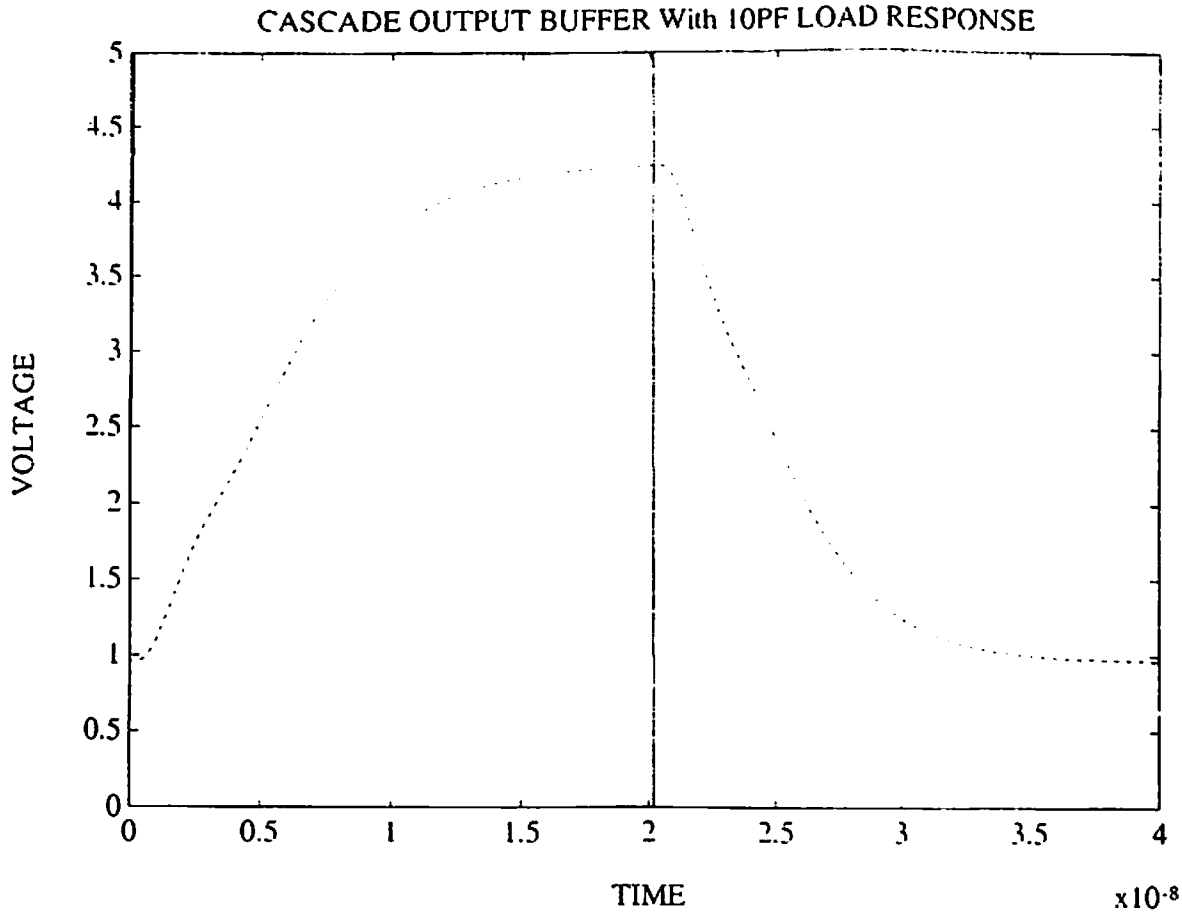


Figure 2.3 Multiplexer output response



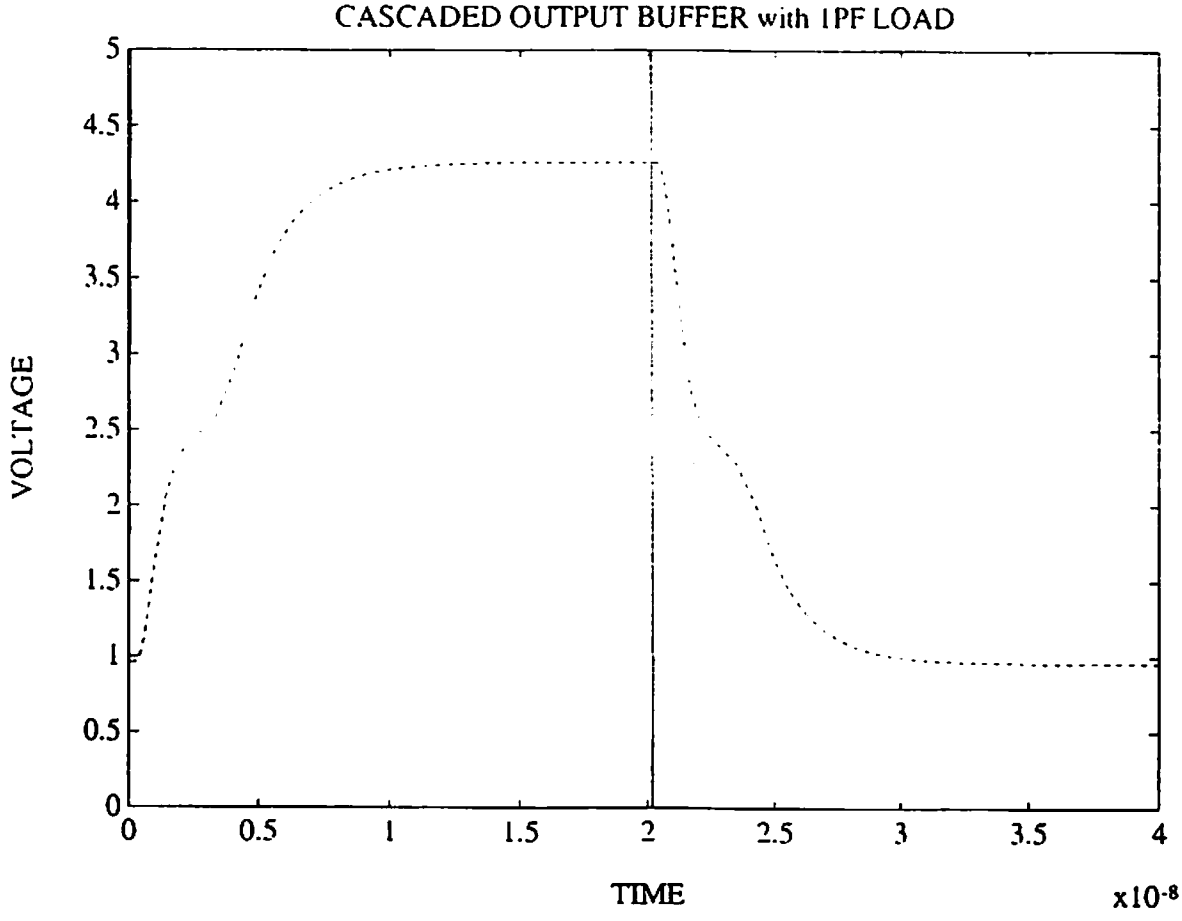


Figure 3.2 Cascaded output buffer output response

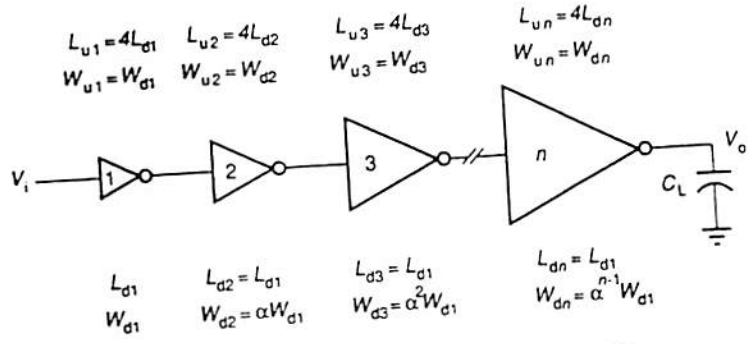
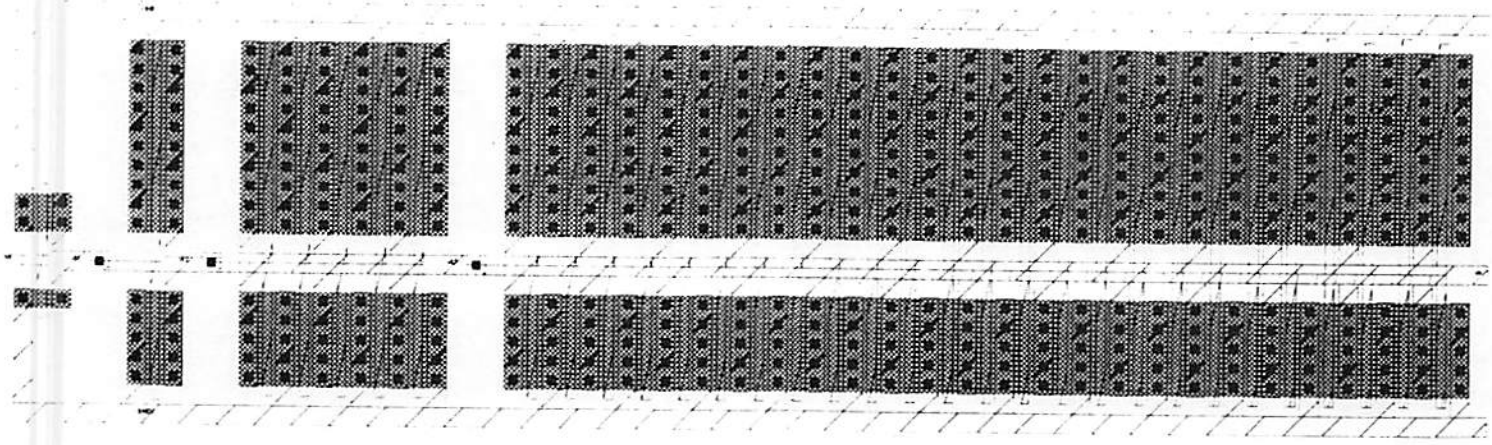


Figure 3.1 Cascaded driver for a concentrated Load



TRUTH TABLE

C	D	N	P	OUTPUT
0	X	0	1	Z (HIGH IMPEDANCE)
1	0	1	1	0
1	1	0	0	1

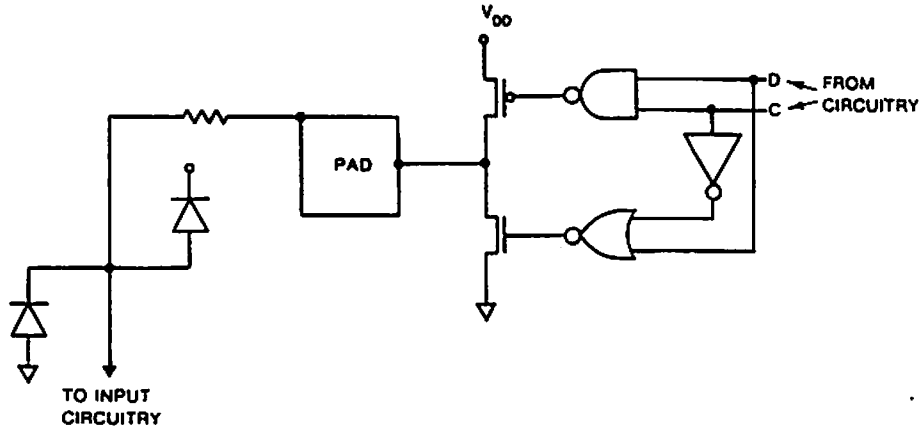


Figure 4.1 Circuit of tri-state input output buffer

Ring Interface of Multi-Multi

Term Project of EE-577, Professor Bing Sheu, Fall 1991

Chen-Chiu Joseph Teng

888-01-3061

cteng@arisbe.usc.edu

Computer Engineering Division
Department of E.E.
University of Southern California

Ring Interface of Multi-Multi

Chen-Chiu Joseph Teng

Term Project Report of EE-577 **CE DIVISION (EE)**

ABSTRACT

Some preliminary design issues of an ASIC chip for the ring interface tentatively used in the Trojan Multi-Multi shared-memory multiprocessor system are investigated in this project. When shared-memory multiprocessor systems are popularly explored to achieve better performance/cost gains over single processor techniques, the interconnection network plays one of the most important roles on the side of performance and system expansibility. A novel high speed shifting slotted ring is adopted as the base of interconnection network in Multi-Multi. After examining and estimating the initiatory design of Multi-Multi board, I found large amount of board area and power consumption are eaten by the ring interface, which is built of ECL chips to meet the speed requirement. A lot of other technical problems arise as well while dealing with so many parts, after including the rest blocks of each node. Hereby, a plan to implement the ring interface circuitry in a VLSI ASIC is desired in order to reduce chip count and power consumption, at the same time solve some technical problems and increase the reliability. To make it suitable for a less than one-month term project with only two students, and also meet with the future development, the project is divided into two streams in order to testify to its feasibility of running at 200MHz. One stream is concerning about modifying the original logical design of the ring interface, identifying critical paths and basic layout cells, and then developing it in a top-down way. The other stream is concerning about building cell library in a bottom-up manner, satisfying the requirements from top levels. This report is for the overall introduction, the top-down design portion and PAD design. The pin specification for the chip and a tentative floor plan are done. Critical data path and components on it are identified. Recognizing PAD delay is the critical issue to achieve a high ring speed, some study of it is included at the end.

Contents

1. Introduction

As the demand of computing power keeps going over the limit of state-of-art single processor technologies, parallel processing with multiple processing elements seems to be one possible solution. Multiprocessor systems built on interconnected microprocessors thus play important roles in computer architecture community due to their higher performance/cost ratios than supercomputers. Among those, shared-memory ones receive more interest because of their friendly programming environments than distributed systems. Different interconnection networks have been proposed or even built for shared-memory multiprocessors. Almost all successful commercial multiprocessor systems on the market today are based on single-bus schemes and hence the numbers of processors in those systems are quite constrained by the limited bandwidths of buses. In order to support a large number of processors, more complicate interconnection structures, such as multi-stage interconnection networks (MIN), hierarchies of buses, meshes, three-dimension cubes, etc., are being exploited. The success of a large-size shared-memory multiprocessor is very much dependent on efficient cache coherency enforcement. In Trojan Multi-Multi shared-memory multiprocessor system, pipeline slotted rings are adopted as the interconnection network for cache coherency and data transfer. The prototype will have 9 nodes connected by 6 uni-directional rings as a mesh scheme. Each node has a BAM processor, 32k+32k words instruction and data cache, 16M words main memory, a cache controller, a memory coherency controller, an MMU, an I/O processor, a SCSI controller, a VME interface, and a ring interface for two dimensions (Appendix A.1). The whole virtual memory address space is distributively partitioned into each node. A snoopy cache coherency protocol is adopted with the help of memory directory scheme. Each node snoops transactions on rings and responses appropriately. The I/O processor controls activities on the I/O bus. A local 0.5G-byte disk is attached to each node through SCSI interface for paging. VME bus is used as the back-plane to the host.

Each ring can be viewed as composed of circular uni-directional shift registers. The size of each register is 32 bits here. Each node monitors a fixed number, called window size Wi, of shift registers of the ith dimension ring to which it is connected. Window sizes can be different for each dimension, but they are the same in current design for the reason of symmetry. If the total number of nodes on the ring is N, then the ring is composed of N*W shift registers. Each node puts on, takes off, and reads data which are in its own window according to the ring protocol. Data is shifted in at one end and out at the other end synchronously. By synchronously, it means there is a single ring clock all over nodes on the same ring, and once the ring starts, it behaves like a clocked circular

pipeline with no stopping. The clocks for different dimensions are chosen to be the same, which is targeted at 200MHz, for the reasons of system symmetry and synchronous timing problem. Currently, Multi-Multi uses two dimensions of rings connected in a mesh organization for supporting high bandwidth requirement of data transfers among nodes in the system. A ring protocol and its corresponding interface are designed. To reduce the latency of transactions on rings, W is chosen as small as possible, which is 3 to allow the ring logic circuitry enough time to response. Owing to the high speed requirement, the ring interface is built of ECL chips if commercial parts are used. For a simple ring protocol supporting the basic mechanism to ensure correctness and compromise performance, the interface uses up to about 298 chips and consumes about 207 to 235W power for two dimensions of rings on each node, and cache keys are not even included yet. A ring interface block diagram and chip summary of ECL implementation are shown in Appendix A.2 and A.3 respectively.

After considering about the technical problems of ECL implementation, a VLSI ASIC for ring interface is very much desired. It can reduce the chip number, thus board area, and power dissipation a lot. The reliability of system is also increased at the same time. Inspecting the initiatory logic design of ring interface, the critical paths are easily identified. The output queue is one of them. (With careful snooping logic design, the cache access is not on the critical path anymore, which is a great advantage.) On the ring itself. to ensure the propagation delay across chips is less than 5ns is very important. Owing to that the transfer of data on rings has to use ECL signals for frequency at 200MHz, the delay of two converters has to be counted in, which makes very fast pads necessary. A special pad design for ring interface is done.

2. Ring Interface

2.1 Ring Protocol

A ring protocol is proposed, with its corresponding logic designed. Basically, most transactions on rings considered here are concerning about maintaining data consistency of the global shared memory through certain cache coherency protocol. However, to support other kinds of data transfers, such as page migration, can also be easily extended. Each transaction is formatted in a fixed data structure "frame", which is composed of six 32-bit words. The first word contains the head of a transaction, which includes a empty bit, sender ID (4 bits), transaction types (5 bits), destination address (6 bits, for some

transaction types), home node ID (the 6 most significant bits of long virtual address), acknowledge field (2 bits, for previous frame), etc. The second word contains the rest virtual address which is exactly 32 bits for BAM. The other four words are for data block. Briefly speaking, due to the existence of multiple entry points on rings, each transaction needs an acknowledge before it's completed. Different reply schemes send their responses at different time. Currently, my scheme encodes the response as two bits: "acknowledge" and "busy" in the first word of the next frame. Some basic operation protocols are described below. For transactions which have been on the ring for one circle, the sender takes them down and puts in the pending buffer for determining if it needs re-sending after receives the acknowledge in the following frame. For those needed to be routed to the other ring or buffered for further memory or cache operations, the ring controller will buffer (and even remove) the transaction while the corresponding latches are available. Otherwise, the transaction is passed by with a negative acknowledge sent. For snooping requests, the ring controller will snoop the appropriate cache directory and send reply according to the cache coherency protocol and the current cache state. If the cache is locked by the cache controller, a "busy" response is encoded in the acknowledge field, which is in the first slot of the next frame. At the output end of the window on each node, transactions are sent out while the frame is empty (decoded by the ring controller). If there is a transaction in the pending buffer, it's shifted out, otherwise waiting transactions in output queue is sent out. Current design assumes 16 entries in the output queue before further simulation result is got.

2.2 VLSI Implementation

The pin specification for ring interface ASIC is done and shown in Appendix A.4, with its description in A.5. A initiatory floor plan is shown in A.6. Basic cell library has been built. The delay of a output queue is equal to 2.5ns, assuming 0.5um TRW technology. The other portion contributing to the critical path is the pads, which is discussed in the next section. We are also considering to include the cache directories into the chip. Further design and layout are still undergoing and being evaluated.

3. Pad Design

3.1 BAM's Pad

As we all know, I/O structures require the most amount of circuit design expertise in association with detailed process knowledge. In most time, pads are provided to a system designer in a well characterized library. In the previous Berkeley BAM project, a pad library has been built in 1.2um technology. For the ring interface ASIC, only pins for ring connection need to run at 200MHz. Data-in pins use input pads and data-out pins use output pads. These are all uni-directional, thus no bi-direction and output enable pads will be discussed below. The basic circuit models and layouts for BAM's pads are shown Appendix B.1 and B.2. Because the intended connection has one TTL-ECL converter and one ECL-TTL converter between a pair of output and input pads, the simulation uses the standard load of converters, which is chosen as 5pF. The PCB wiring has 2ohm/in and 2pF/in impedance. In simulation, 2 inches of PCB wiring is selected as the worst case. The bonding wire has the worst case impedance of 1ohm, 20nH, and 25fF from the technology used by BAM. For each case, both the intrinsic delay and that of extracted circuit from Magic are simulated for comparison. Basically, the intrinsic delay represents the lower bound and the later one can be treated as the upper bound of the real delay. The real delay should be measured from the real fabricated pad. The previous measured results show that they are in fact between the bounds and even much closer to the intrinsic ones. The simulation results are shown below. All the SPICE simulation files and graphics are shown in Appendix C and Appendix D.

Table 1: Delay

1.2um	Intrinsic			Magic		
(unit: ns)	t_{pLH}	t_{pHL}	t_{max}	t_{pLH}	t_{pHL}	t_{max}
input pad	0.272	0.583	0.583	2.437	3.573	3.573
output pad	3.964	2.389	3.964	18.675	11.232	18.675
input-output	4.373	3.59	4.373	18.486	18.734	18.734

0.5um	Intrinsic			Magic		
(unit: ns)	t_{pLH}	t_{pHL}	t_{max}	t_{pLH}	t_{pHL}	t_{max}
input pad	0.159	0.425	0.425	1.343	1.693	1.693
output pad	2.396	1.288	2.396	10.05	7.794	10.05
input-output	2.219	1.486	2.219	7.834	9.289	9.289

Table 2: Power Bouncing

1.2um	Intrinsic				Magic			
(unit: mv)	GND		Vdd		GND		Vdd	
input pad	+11.39	-2.232	+2.221	-10.753	+125.0	-130.7	+7.567	-6.6
output pad	+21.91	-1.675	+1.480	-27.081	+1.722	-0.426	+0.664	-2.323

0.5um	Intrinsic				Magic			
(unit: mv)	GND		Vdd		GND		Vdd	
input pad	+13.13	-0.797	+0.818	-12.695	+1.7	-0.876	+0.528	-1.267
output pad	+26.90	-1.823	+1.949	-30.792	+2.486	-0.417	+0.713	-2.919

From the simulation results, the intrinsic delays of output pads have fallen down below 3ns after using 0.5um technology. But the results got from Magic extracted circuits are still too high and make the other numbers very unreliable. Actually, the scaling method applied to spice files got from 1.2um technology for 0.5um technology is not accurate according to previous experience. The simulated results show that the output pad of BAM has the worst delay of 18.675ns though the intrinsic delay is only 3.964ns in the testing environment. In fact, measured figures for the real BAM chip show that the delay of uni-direction output pad has a propagation delay ranging from 4.5ns ~ 5.5ns, which is much closer to the intrinsic value. If the case can be applied to scaled down pad as well, then a 3ns output pad is very feasible. For the reason of reliability, a pad design is thus tried below to have better simulation results.

3.2 Ring Pad

Because the simulation results show that the input pad has been well qualified for our requirement, no modification is tried on it. For the output pad, I have tried different transistor sizings in order to get better performance, at least it can satisfy our requirement. By just varying the sizes of transistors in the previous circuit model unfortunately cannot make the intrinsic delay less than 1ns. The reason I want to make the standard, i.e. 1ns

here, so strict is to ensure the real delay still fall in the range. Some more accurate simulation will be done after the TRW 0.5um technology file for Magic is set up. By adding more stages and varying their sizes, currently I get intrinsic delays of t_{pLH}=0.829 and t_{pHL}=0.837ns with four stages of inverting drivers. The delays are very close and with t_{max} only equal to 0.837ns. The power bouncing is only -74.353mv for Vdd and +43.663mv for GND. The exact parameters used in SPICE simulation and its simulation results are shown in Appendix E.

Table 3: Modified Output Pad (0.5um)

Intrinsic Delay (ns)			Power Bouncing (mv)			
t_{pLH}	t_{pHL}	t_{max}	GND		Vdd	
0.829	0.837	0.837	+43.663	-1.361	+2.306	-74.353

4. Future Work

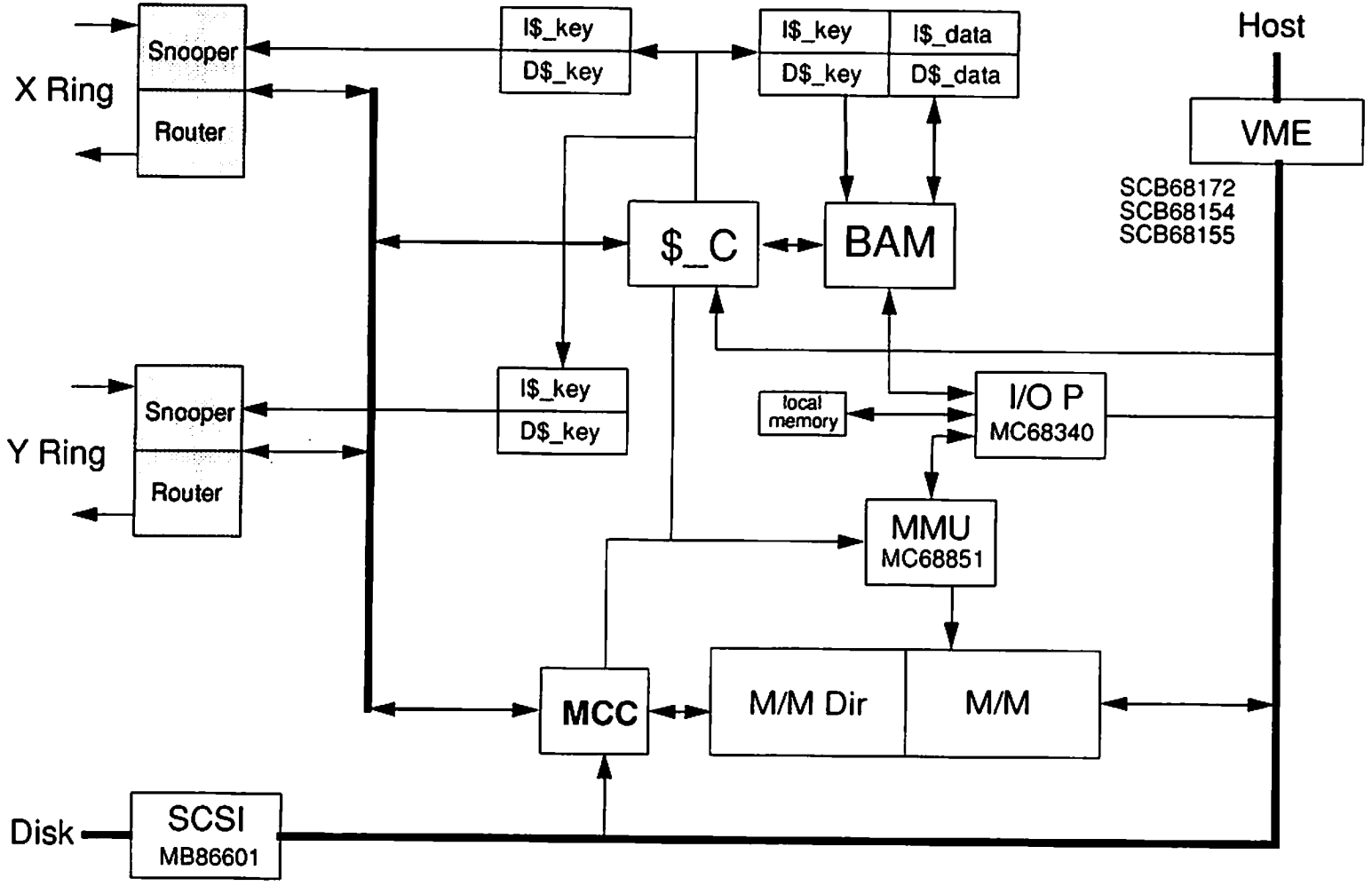
From the simulation results got so far, the critical path has a delay of 2.458ns + 0.837ns = 3.295ns which is about 34.1% below the requirement. If the real fabrication doesn't have a discrepancy more than that, then I am very confident about the feasibility of building our ring interface ASIC at 200MHz, if 0.5um technology is available. Because the pin number is more than 300 even with only 32 bits wide node system bus, the original design with 4 words (128bits) wide bus seems impractical, thus some compromise with performance is predicted. The architectural and logical design of the interface is now in the middle way, due to waiting for the critical path simulation. Another important party involved in the success of this chip is the on-board clock generating circuitry. In the initiatory design, 4 different phases of clocks generated from the ring master clock are used. Making sure the clock skew doesn't hurt the correctness of logic really needs lots of investigation. This is going to be scheduled in the near future. Basically, the architecture and thus the corresponding circuit layout is not complicate. We anticipate to do the layout within three months with primary simulations done.

Bibliography

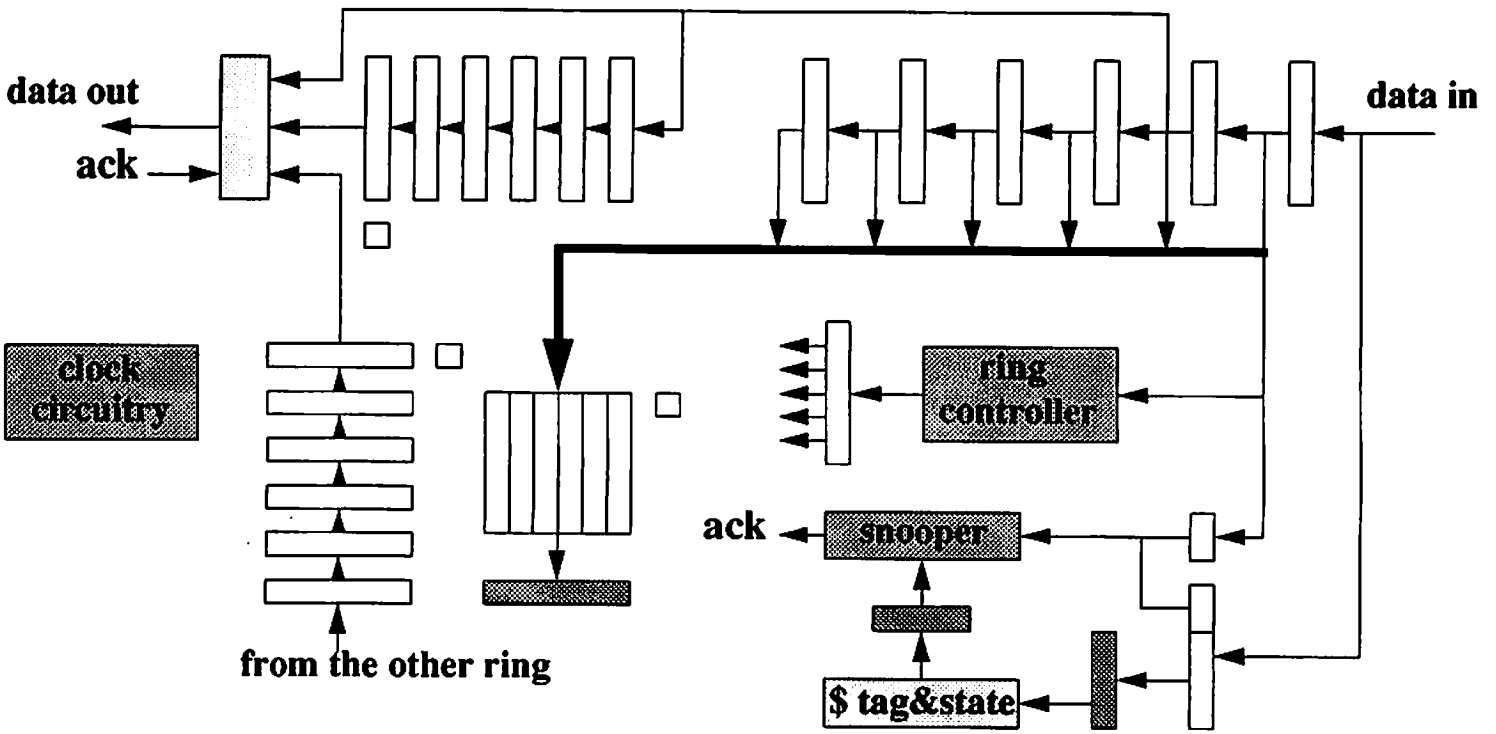
1. Neil Weste, Kamran Eshraghian, "Principles of CMOS VLSI Design. A System Perspective," Addison-Wesley, 1985.

2. Angela Cheng, A. Despain, "Input/Output Buffers for ASP," Technical Report, UC Berkeley, EECS, 1989.

Functional Modules of Multi-Multi Board



Ring Interface (one dimension only)



* Output queue shown here has only one entry.

Estimation of Chip Count and Power dissipation of Ring interface

	Parts	Chip/ring	Power/ring	Total* Chips	Total* Power
ECL latch	MC10E143	~78	~48.7 - 58.8	~156	~97.4 - 117.6
ECL 2:1 MUX_latch	MC10E167	~36	~17.6 - 21.1	~72	~35.2 - 42.2
ECL 4:1 MUX_latch	MC10E156	~11	~4.3 - 5.15	~22	~8.6 - 10.3
ECL PAL	CY10E302	~2	~1.77	~4	~3.54
Converter	CY10E383	~22	~30.9	~44	~61.8
Total		~149	~103.3 - 117.7	~298	~206.6 - 235.4

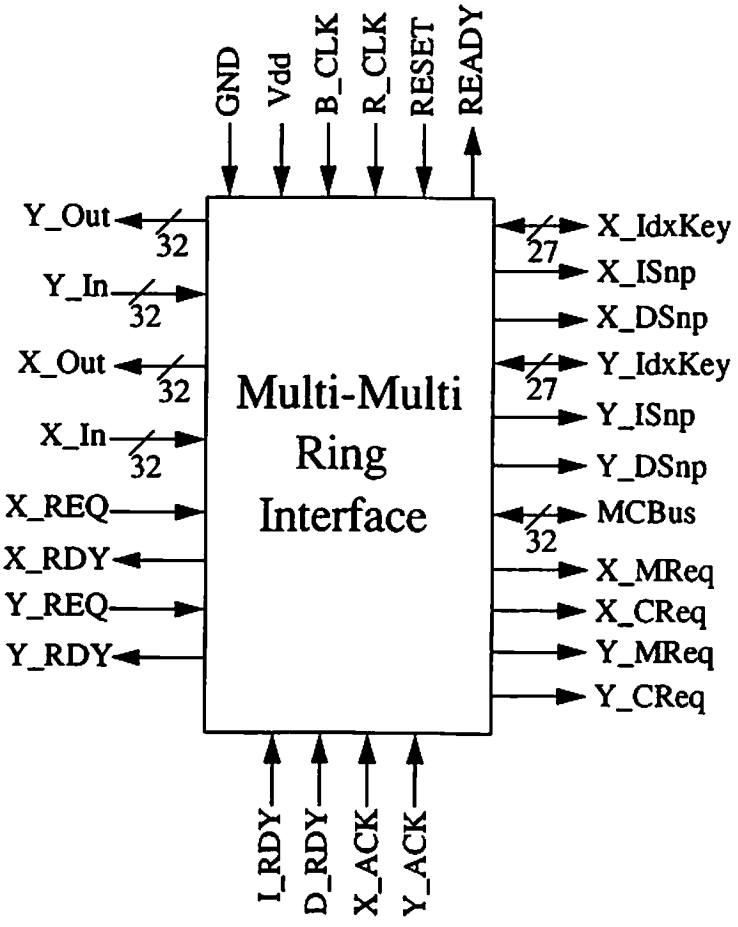
*: For two dimensions.

**: The clock circuitry is not included.

***: The TTL SRAMs for cache tag&state are not included.

Multi-Multi Ring Interface: Pin Specification

(Version 1.1)



Physical Pin Count

Signal: 9+11=20
I/O: 64+150=214
Vdd & GND: 6+75=81

Total: 315

Pin Description

Symbol # pin	Type	Name and Function
B_CLK 1	I	**Board Clock** The clock running on processor board (the board system clock) to synchronize ring interface, memory controller and cache controller.
D_RDY 1	I	**Data Cache Ready** The control signal used to tell ring interface that data cache key is available for snooping.
I_RDY 1	I	**Instruction Cache Ready** The control signal used to tell ring interface that instruction cache key is available for snooping.
MCBus 32	I/O/Z	**Memory/Cache Bus** The bus multiplexed for ring interface to transfer data frames to memory controller or cache controller, and the other direction.
R_CLK 1	I	**Ring Clock** The high speed clock running on rings to synchronize slot shifting. Different dimensions use the same clock frequency and run synchronously.
READY 1	O	**Ring Ready** The control signal used to indicate the reset sequence is properly completed.
RESET 1	I	**Ring Reset** The hardware reset signal for rings. When this signal is asserted, all latches in ring interface are reset, and frame marking is initialized.
X_ACK 1	I	**Bus Acknowledge for X Ring** The acknowledge signal from memory/cache bus arbitrator for X ring. When it's asserted, the bus is allocated for X ring to transfer data to memory controller or cache controller.
X_CReq 1	O	**Cache Request from X Ring** The control signal indicating a transaction is buffered at X ring and waiting for the service of cache controller.
X_DSnp 1	O	**Data Cache Snooping Request from X Ring** The request signal for data cache snooping from X ring.
X_IdxKey 27	I/O/Z	**Snooping address from X Ring** The multiplexed address lines used for cache index address and cache key. At first, cache index address is sent over X_IdxKey<15:0> to access the cache key. Later on, the tag and state of the corresponding cache line are returned on X_IdxKey<22:0> and X_IdxKey<26:23> respectively.
X_In 32	I	**X Ring Datain Bus** The bus carrying data into node from X ring.

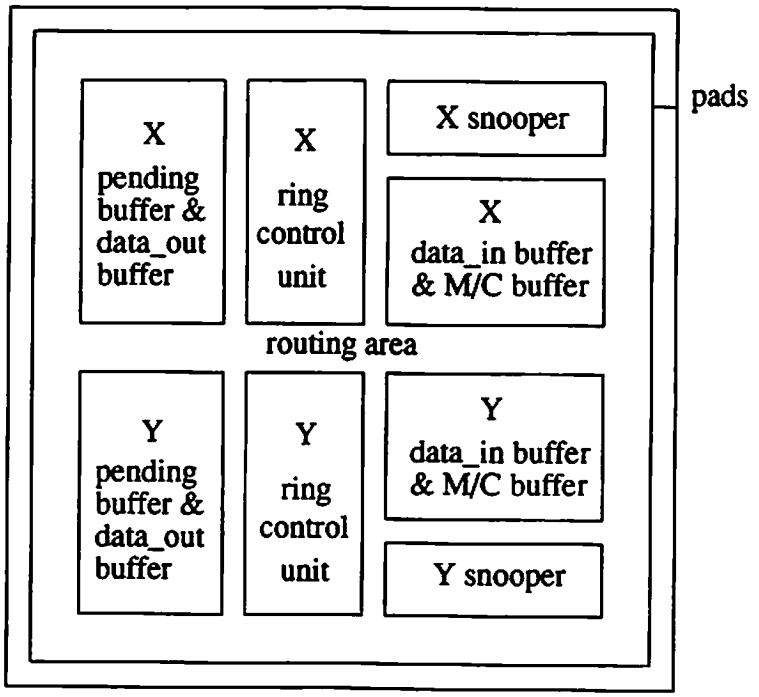
Pin Description

Symbol # pin	Type	Name and Function
X_ISnp 1	O	**Instruction Cache Snooping Request from X Ring** The request signal indicating there is a instruction cache snooping request from X ring.
X_MReq 1	O	**Memory Request from X Ring** The control signal indicating a transaction is buffered at X ring and waiting for the service of memory controller.
X_Out 32	O	**X Ring Dataout Bus** The bus carrying data out of node from X ring.
X_REQ 1	I	**Request for X Ring** The signal indicating memory controller or cache controller is requesting the X ring buffer for data transfers.
X_RDY 1	O	**X Ring Buffer Ready** The signal indicating the X ring buffer is ready to accept data from memory controller or cache controller.
Y_ACK 1	I	**Bus Acknowledge for Y Ring** The acknowledge signal from memory/cache bus arbitrator for Y ring. When it's asserted, the bus is allocated for Y ring to transfer data to memory controller or cache controller.
Y_CReq 1	O	**Cache Request from Y Ring** The control signal indicating a transaction is buffered at Y ring and waiting for the service of cache controller.
Y_DSnp 1	O	**Data Cache Snooping Request from Y Ring** The request signal for data cache snooping from Y ring.
Y_IdxKey 27	I/O/Z	**Snooping address from Y Ring** The multiplexed address lines used for cache index address and cache key. At first, cache index address is sent over Y_IdxKey<15:0> to access the cache key. Later on, the tag and state of the corresponding cache line are returned on Y_IdxKey<22:0> and Y_IdxKey<26:23> respectively.
Y_In 32	I	**Y Ring Datain Bus** The bus carrying data into node from Y ring.
Y_ISnp 1	O	**Instruction Cache Snooping Request from Y Ring** The request signal indicating there is a instruction cache snooping request from Y ring.
Y_MReq 1	O	**Memory Request from Y Ring** The control signal indicating a transaction is buffered at Y ring and waiting for the service of memory controller.
Y_Out 32	O	**Y Ring Dataout Bus** The bus carrying data out of node from Y ring.

Pin Description

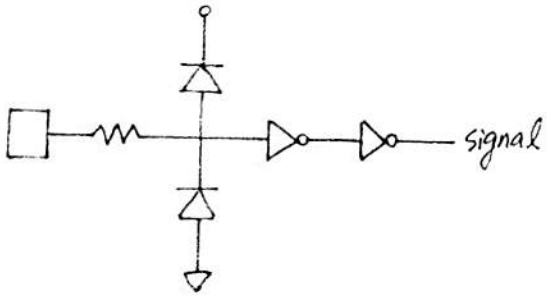
Symbol # pin	Type	Name and Function
Y_REQ 1	I	**Request for Y Ring** The signal indicating memory controller or cache controller is requesting the Y ring buffer for data transfers.
Y_RDY 1	O	**Y Ring Buffer Ready** The signal indicating the Y ring buffer is ready to accept data from memory controller or cache controller.

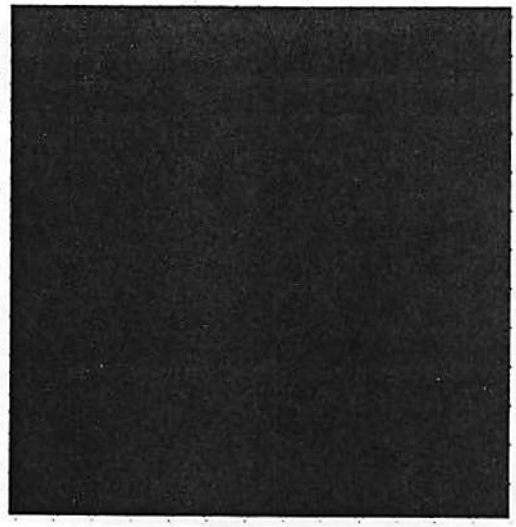
Intentative Floor Plan (version 1.1)

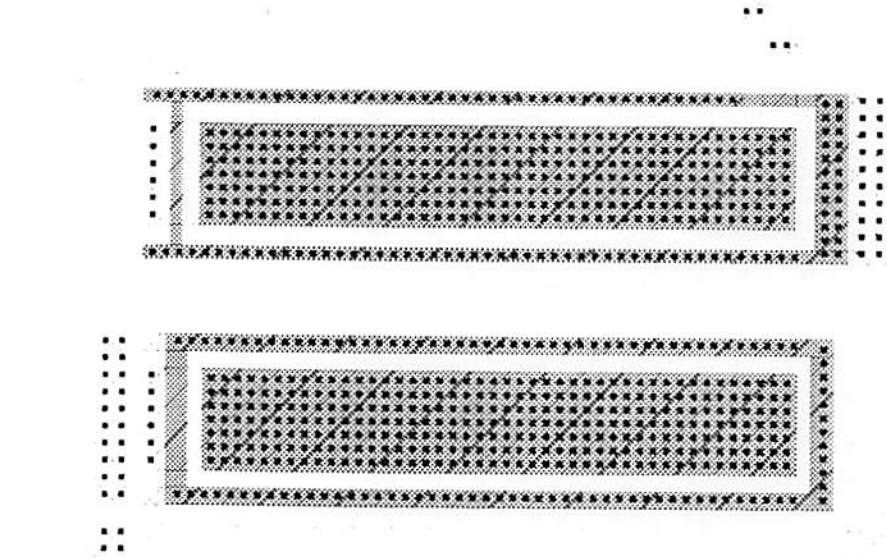


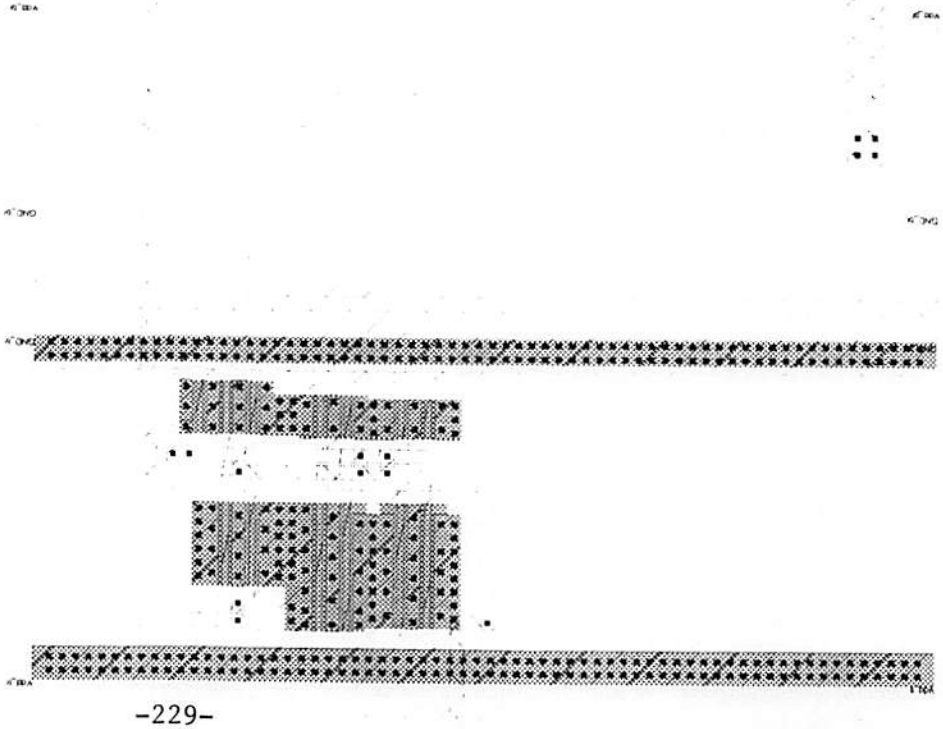
Appendix B.1

Layout of Input PAD and Circuit Model



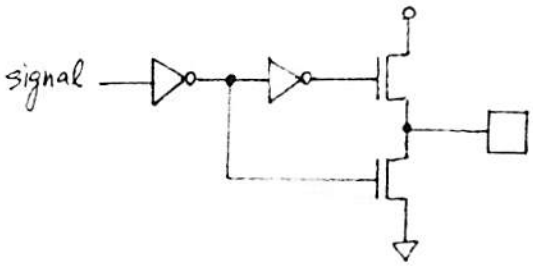


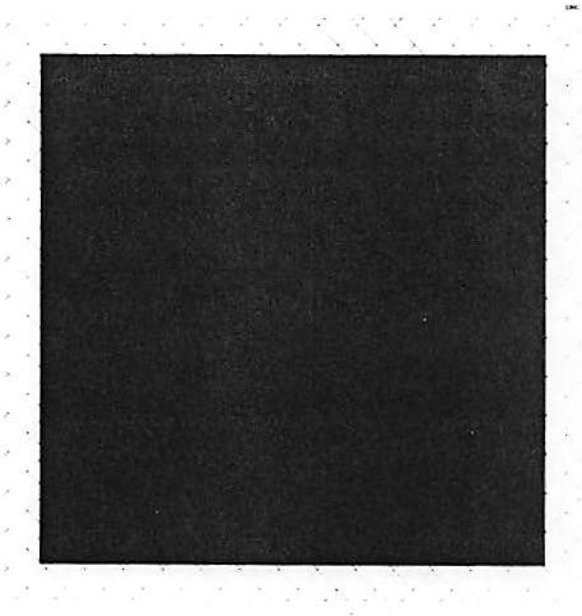


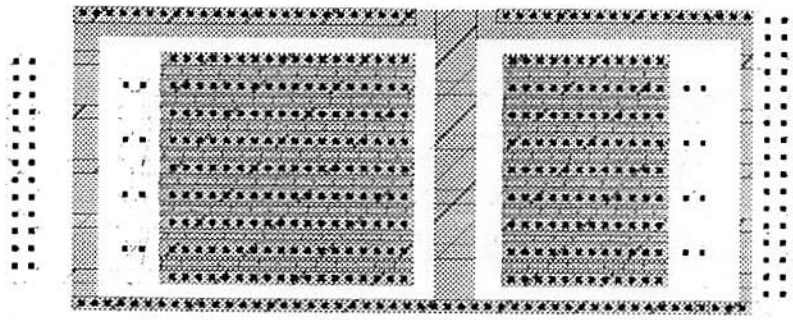


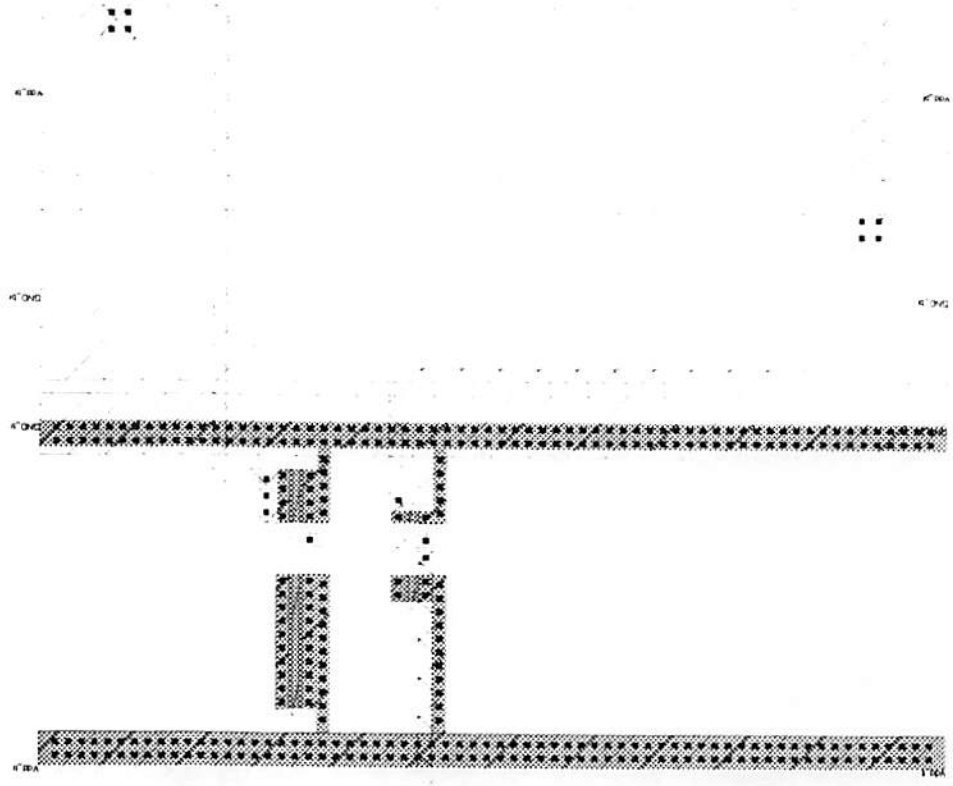
Appendix B.2

Layout of Output PAD and Circuit Model

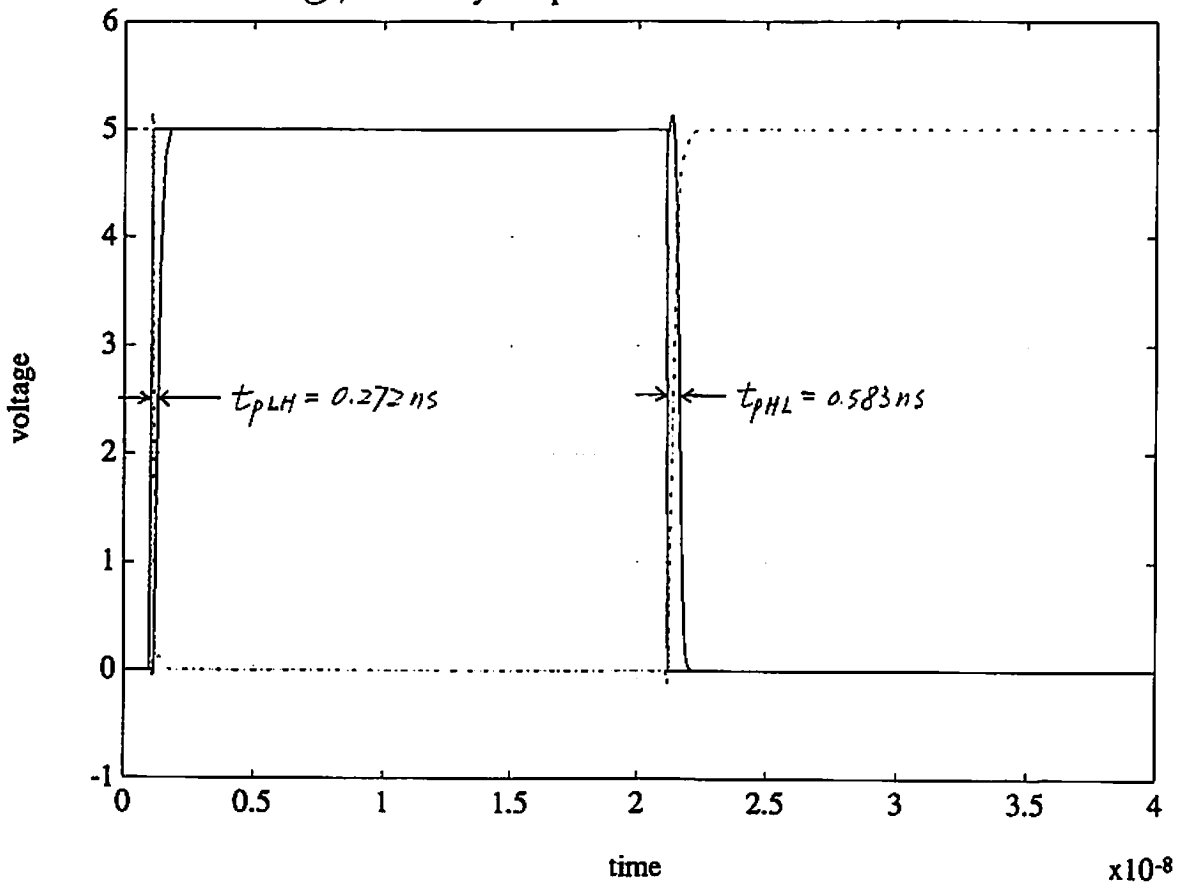




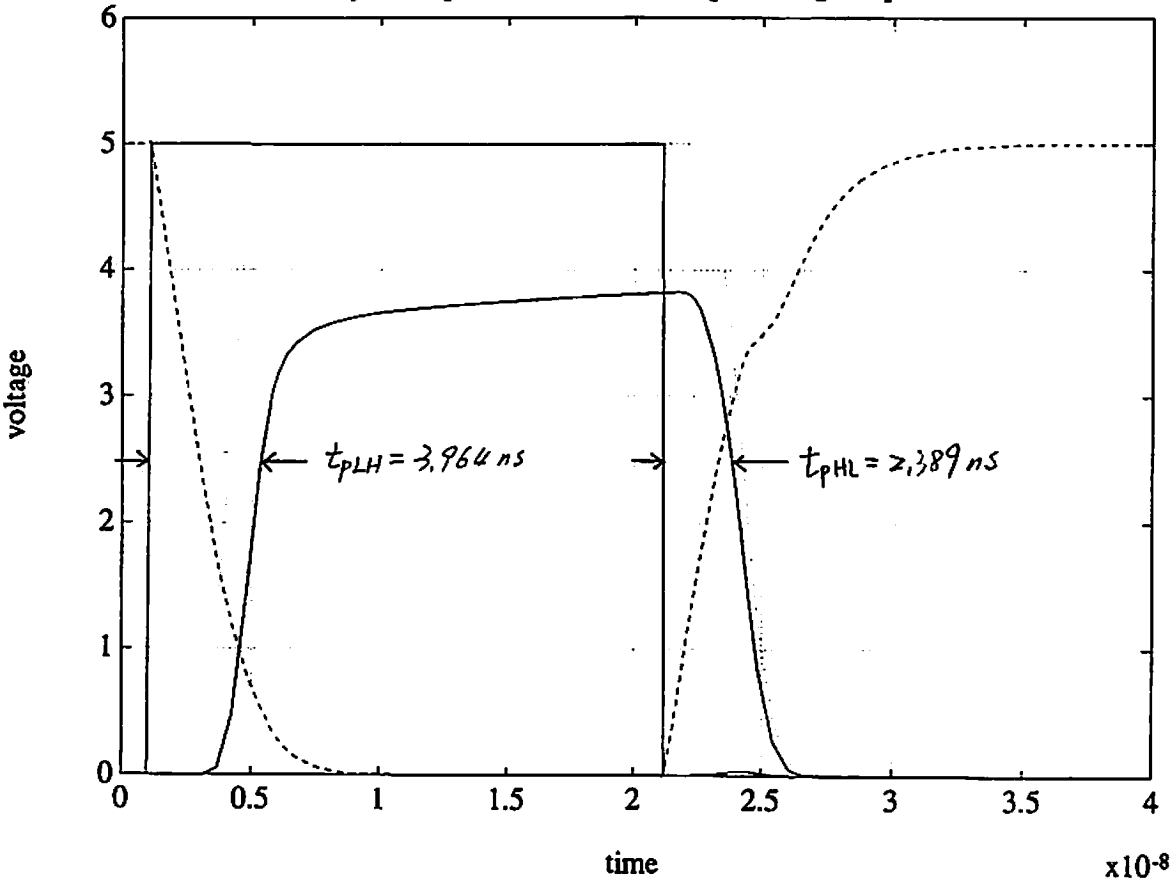




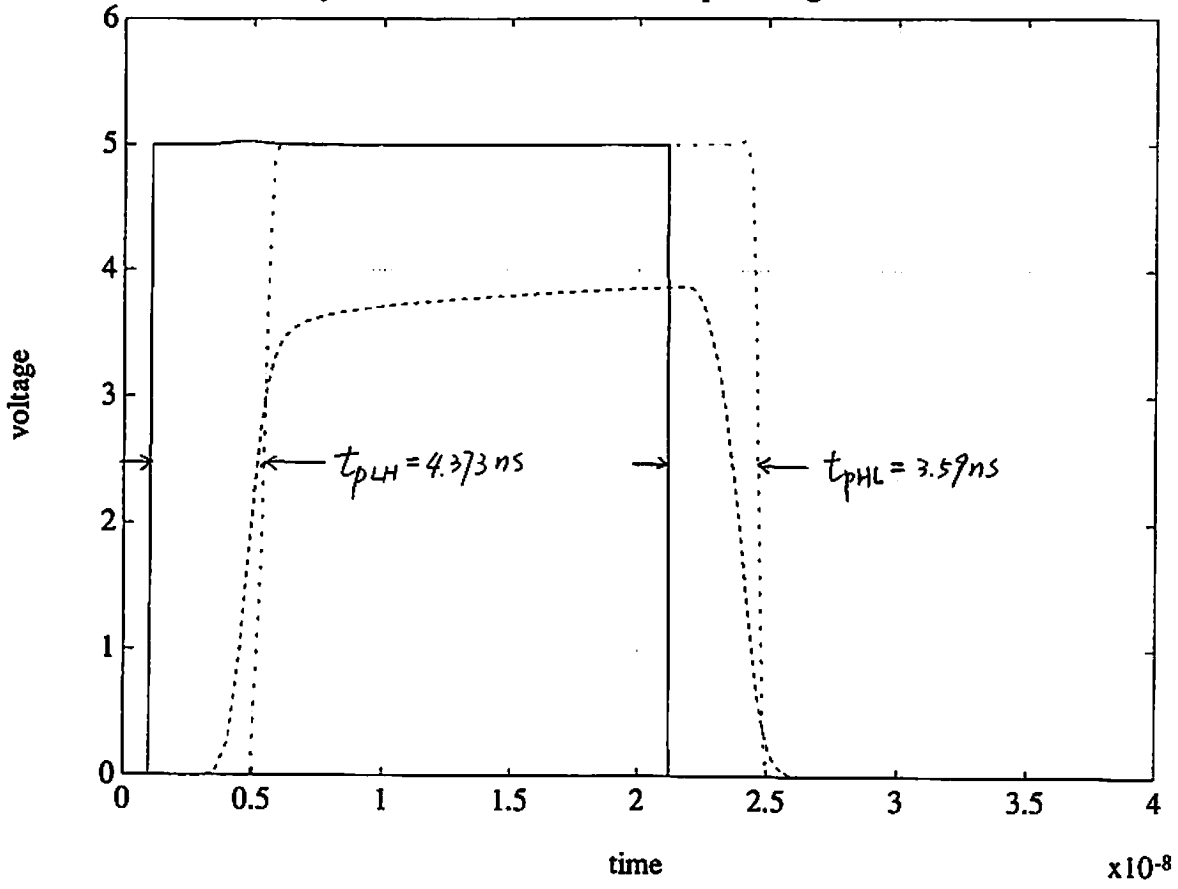
C1 Delay of input PAD with 500fF load.

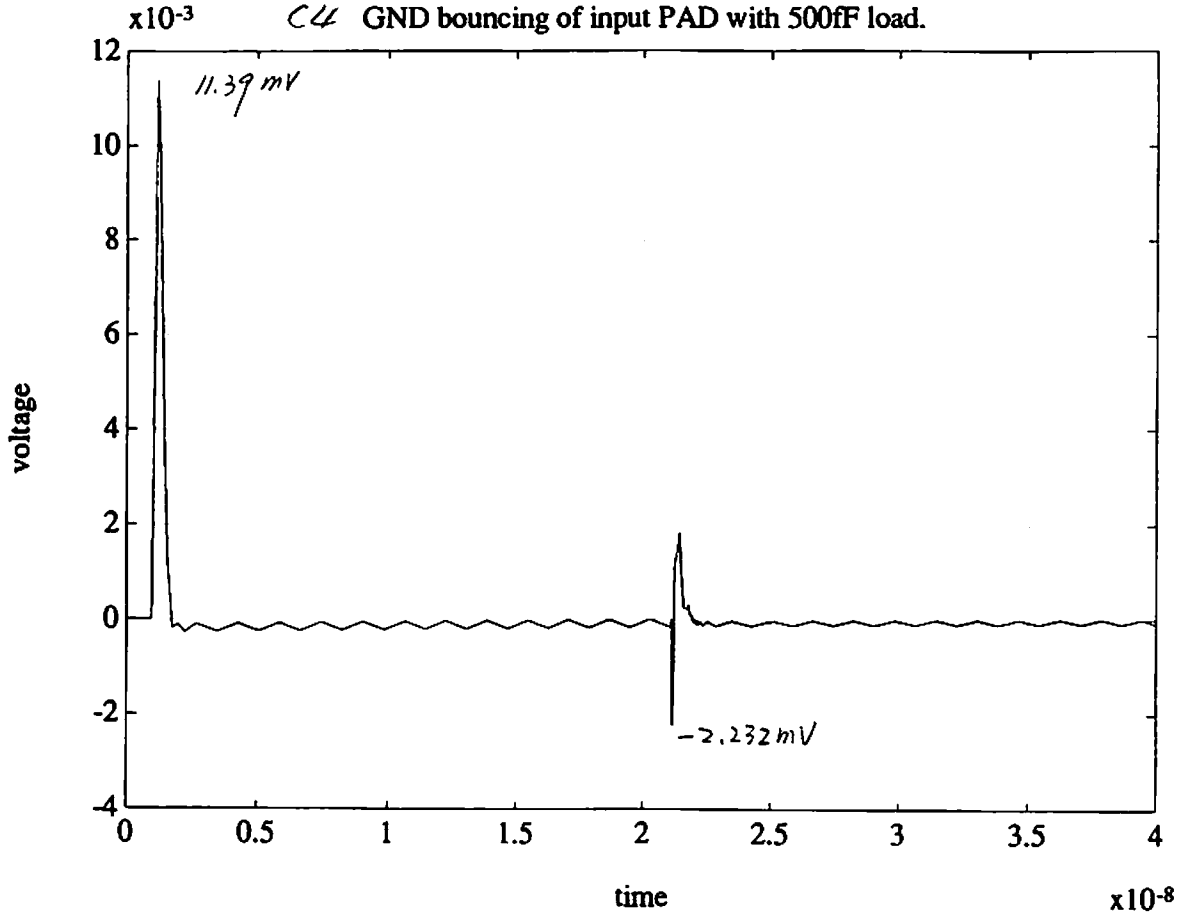


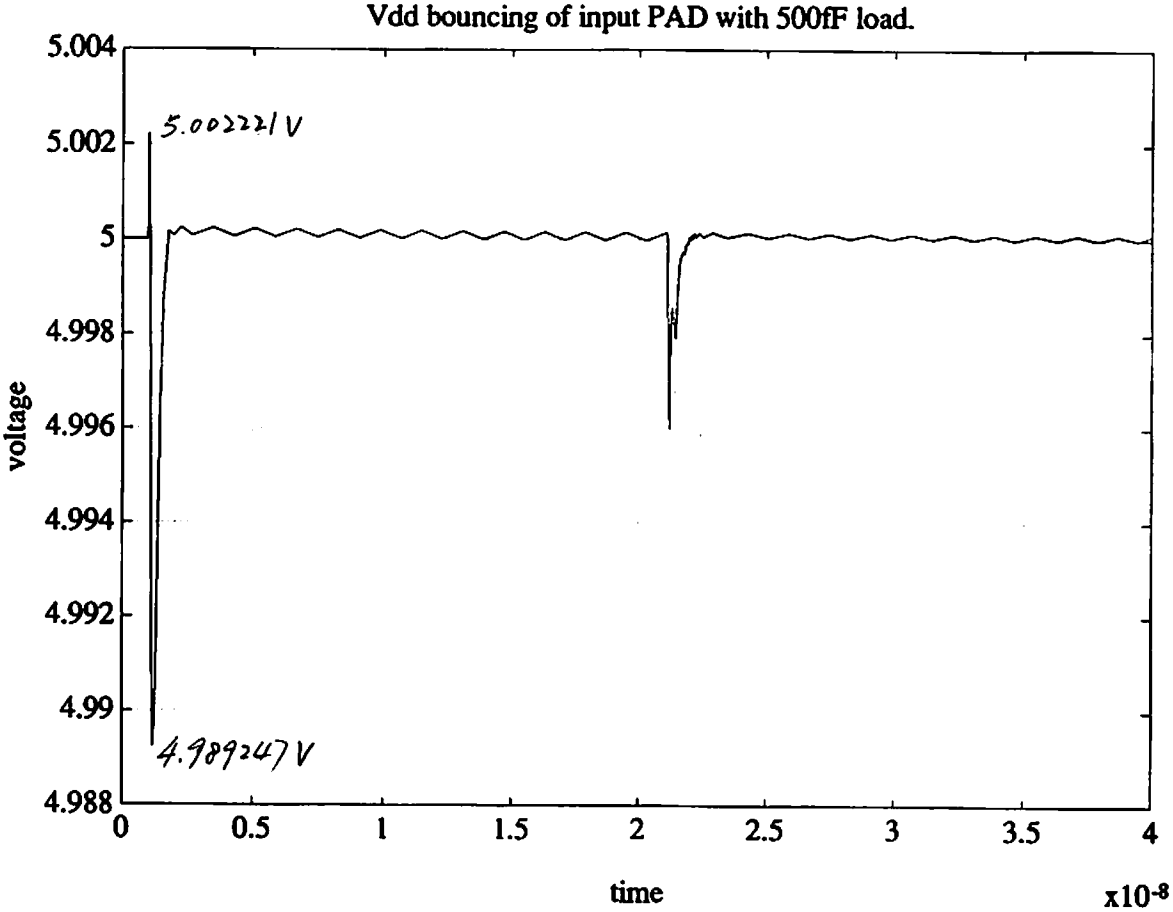
C 2. Delay of output PAD with 4ohm/4pF wiring & 5pF load.



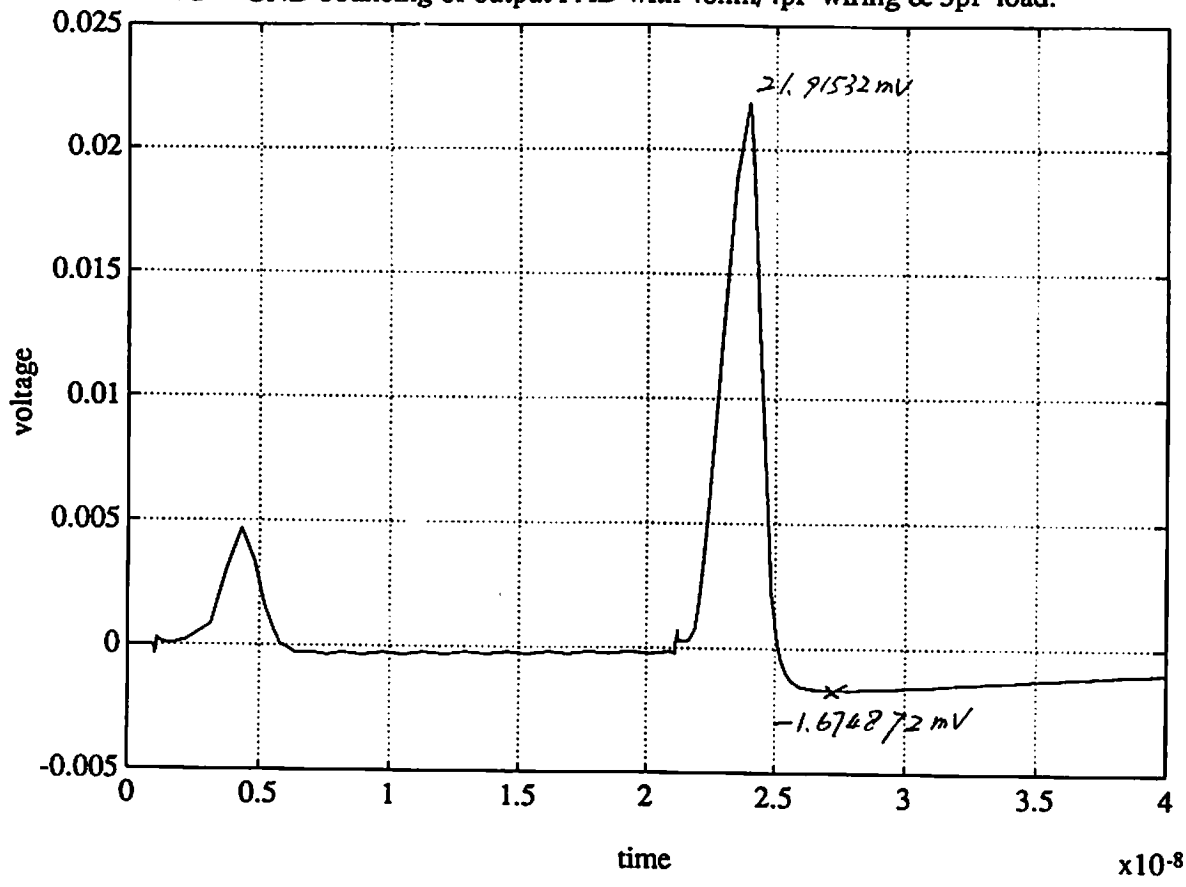
C3 Delay of PAD-to-PAD with 4ohm/4pF wiring and 500fF load.



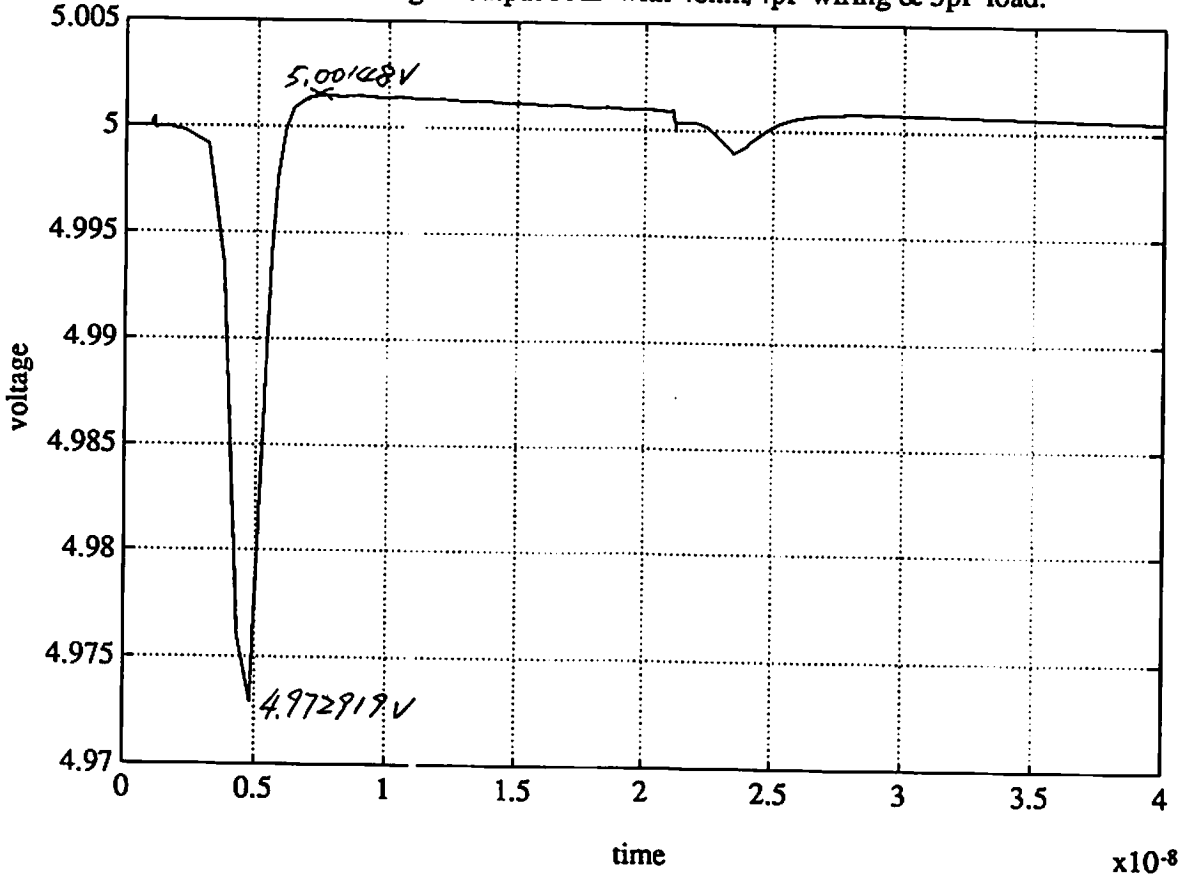

x10-3
C4 GND bouncing of input PAD with 500fF load.
11.39 mV
-2.232 mV
voltage
time
x10-8


Vdd bouncing of input PAD with 500fF load.
5.002221 V
4.989247 V
voltage
time
x10-8

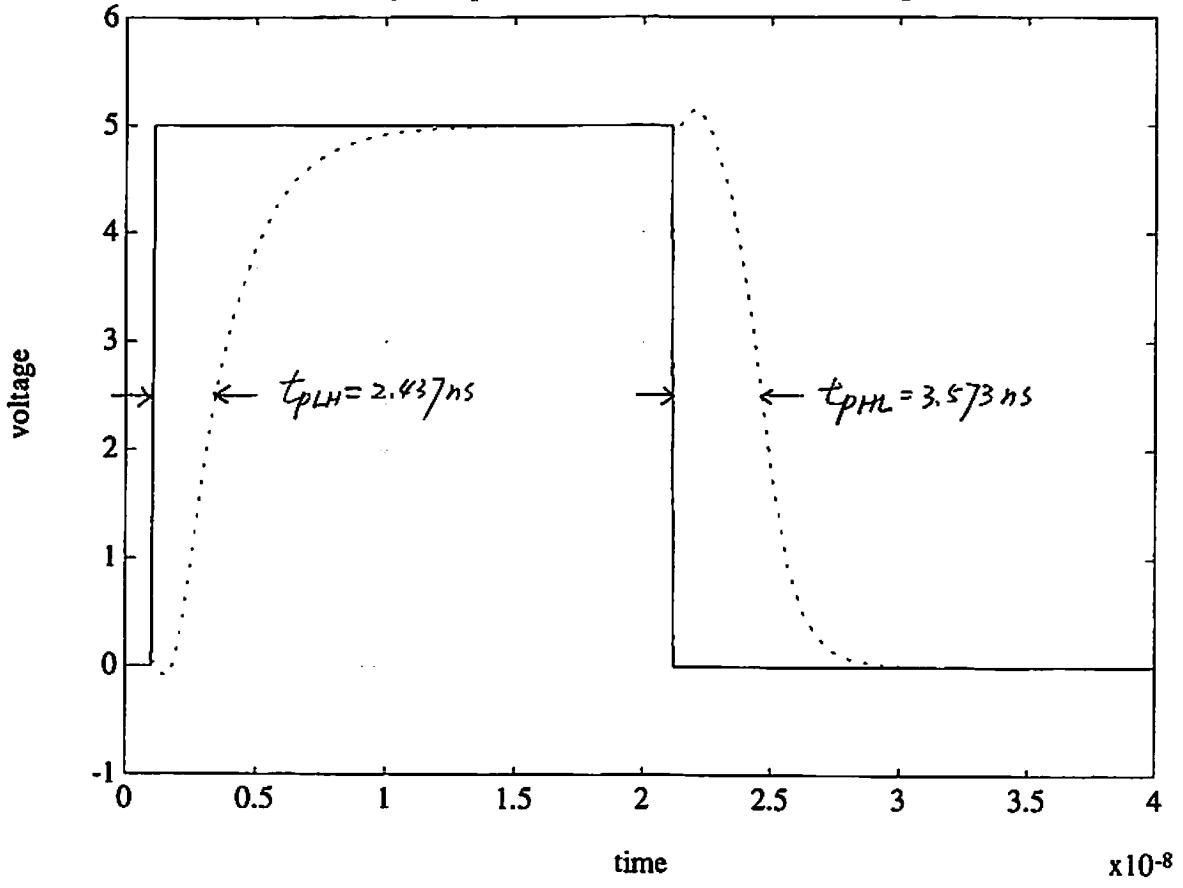
C5 GND bouncing of output PAD with 4ohm/4pF wiring & 5pF load.



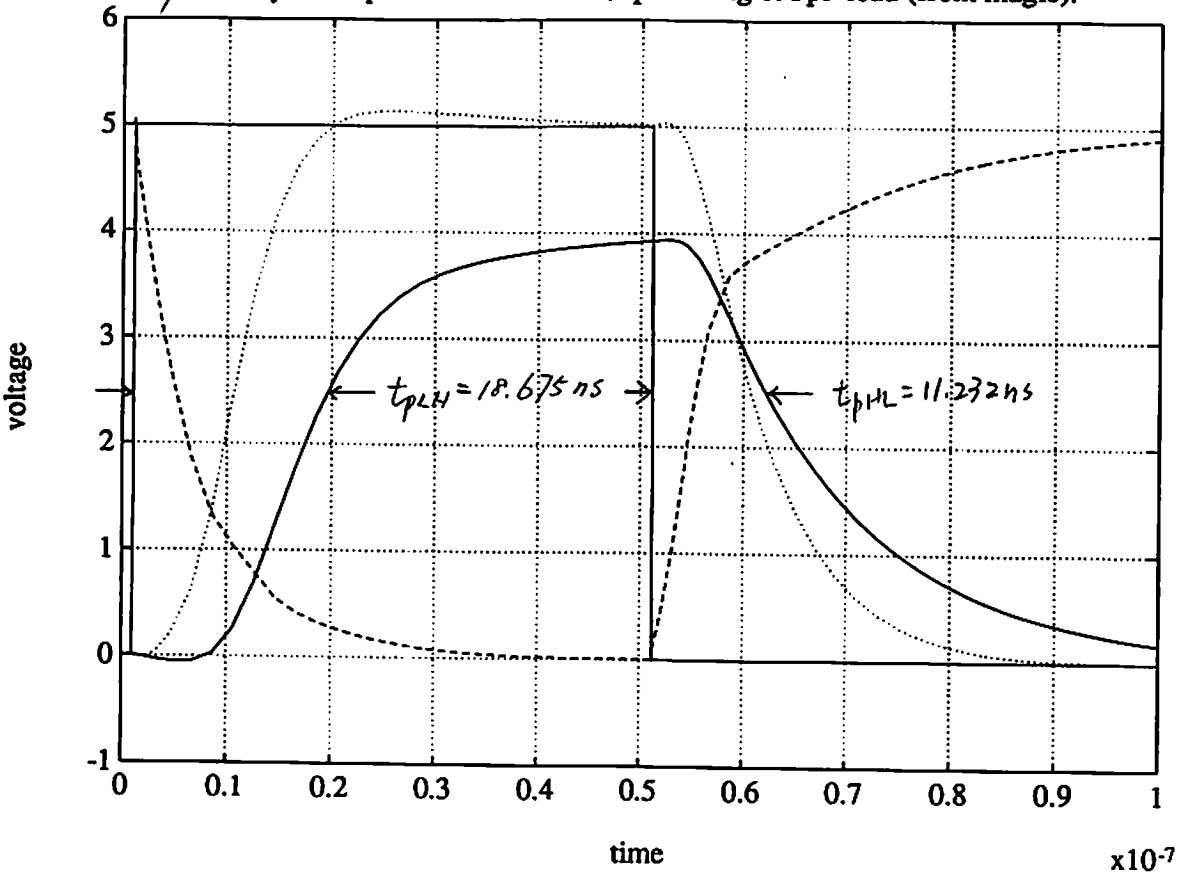
Vdd bouncing of output PAD with 4ohm/4pF wiring & 5pF load.



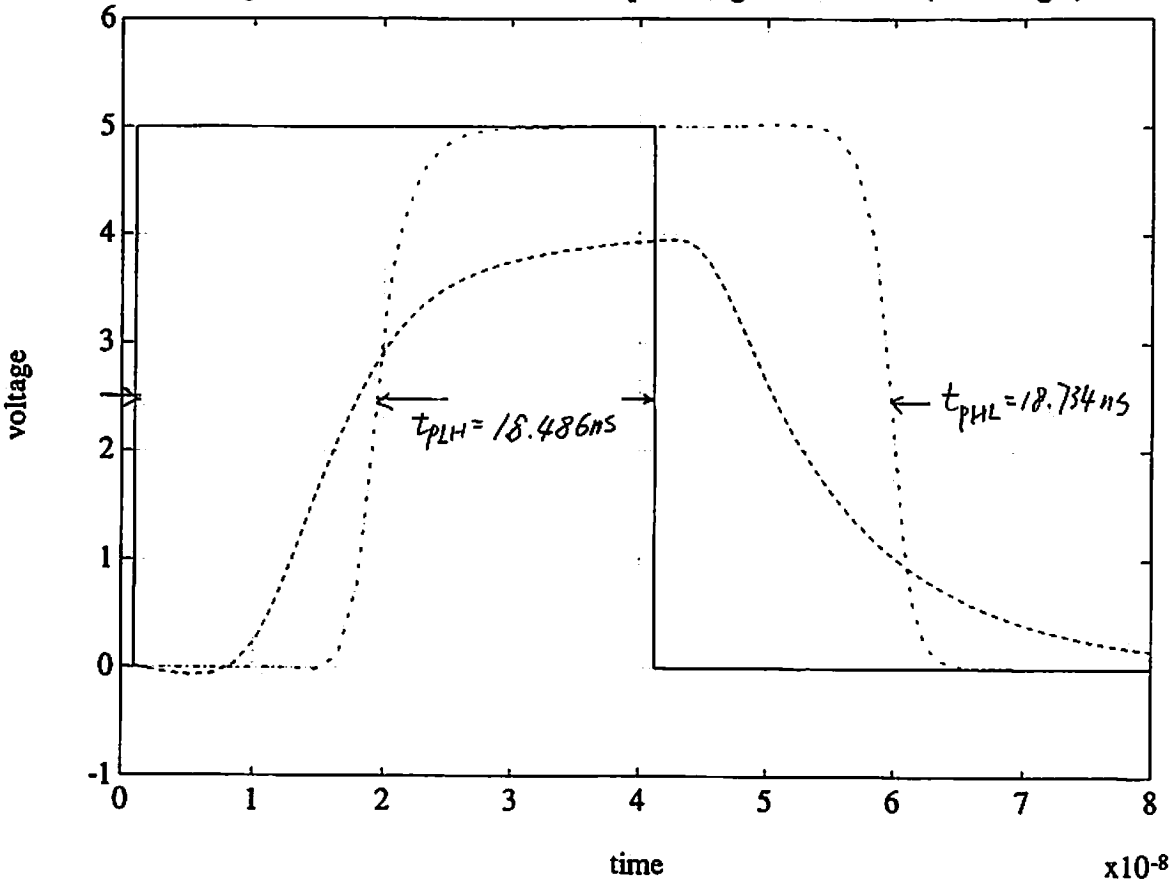
C6 Delay of input PAD with 500fF load (from magic).



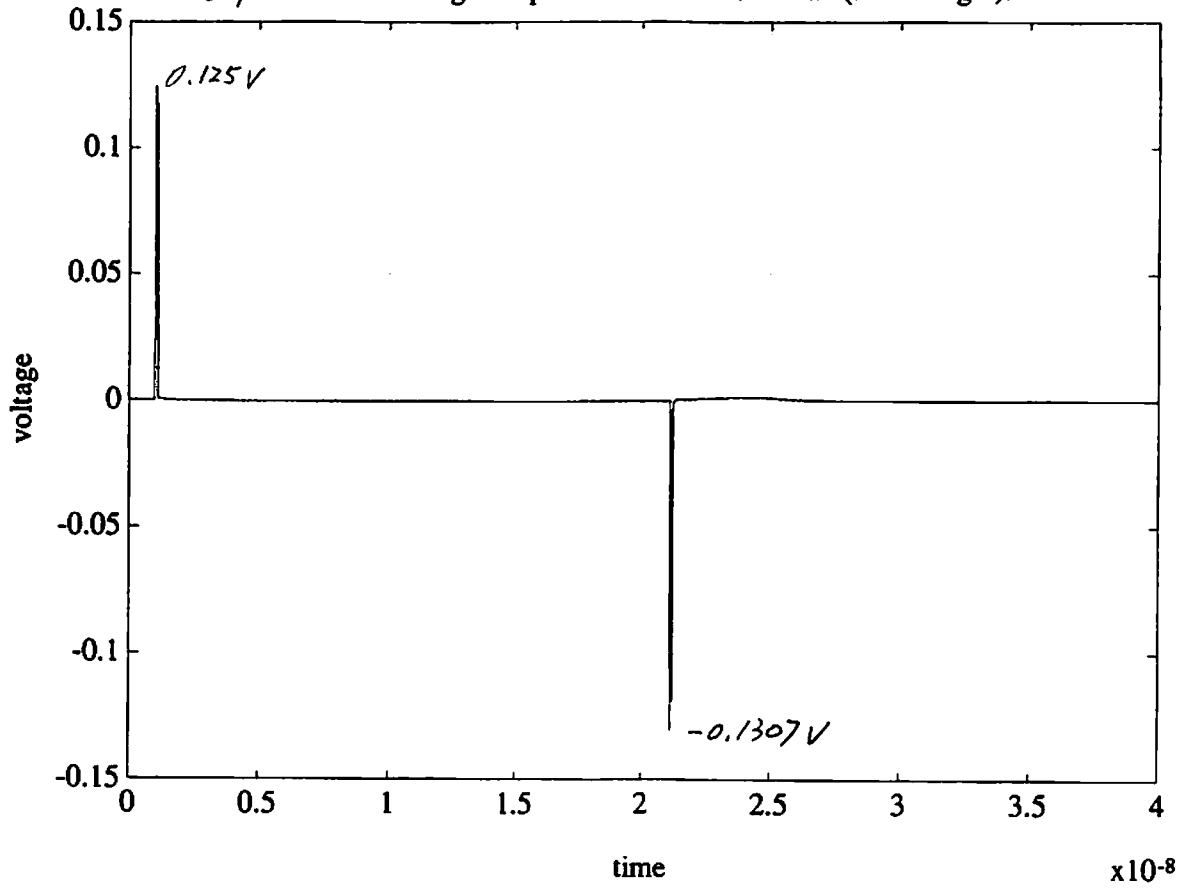
C7 Delay of output PAD with 4ohm/4pF wiring & 5pF load (from magic).



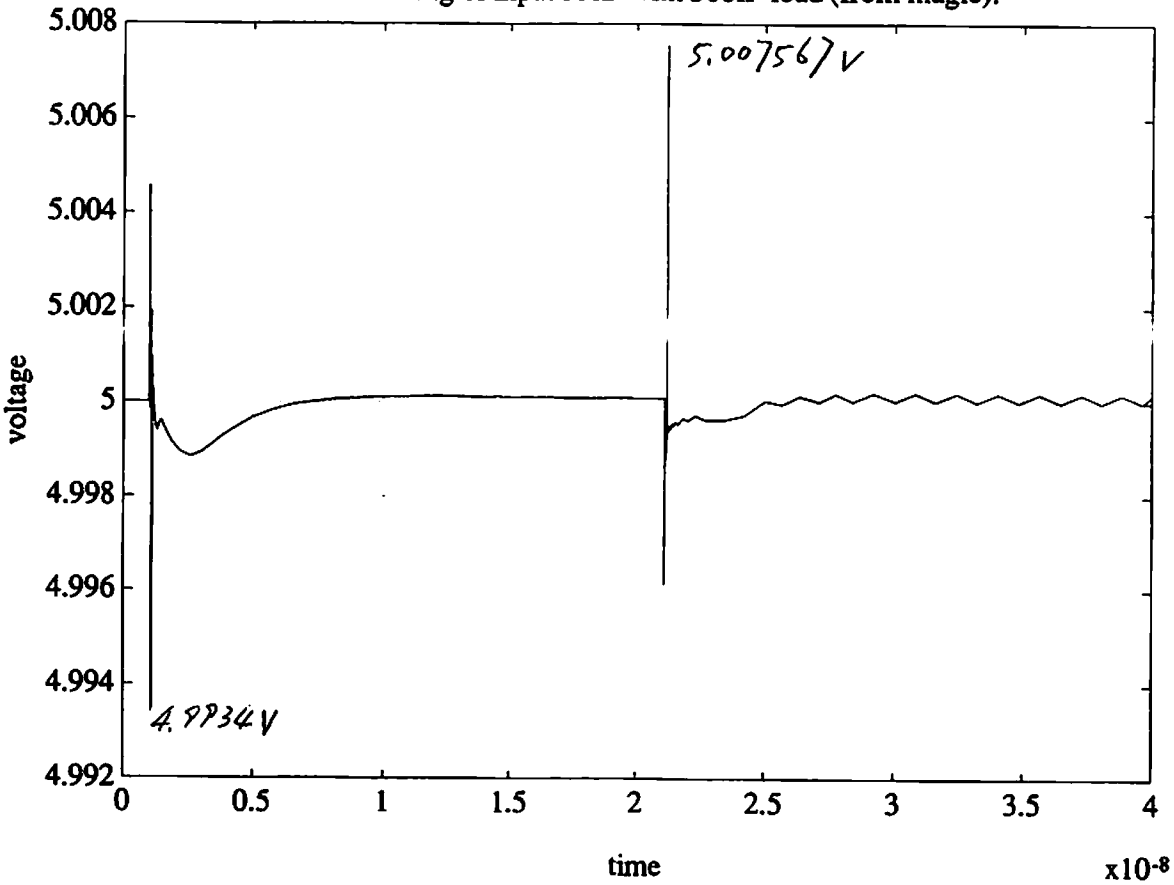
C8 Delay of PAD-to-PAD with 4ohm/4pF wiring & 500fF load (from magic).



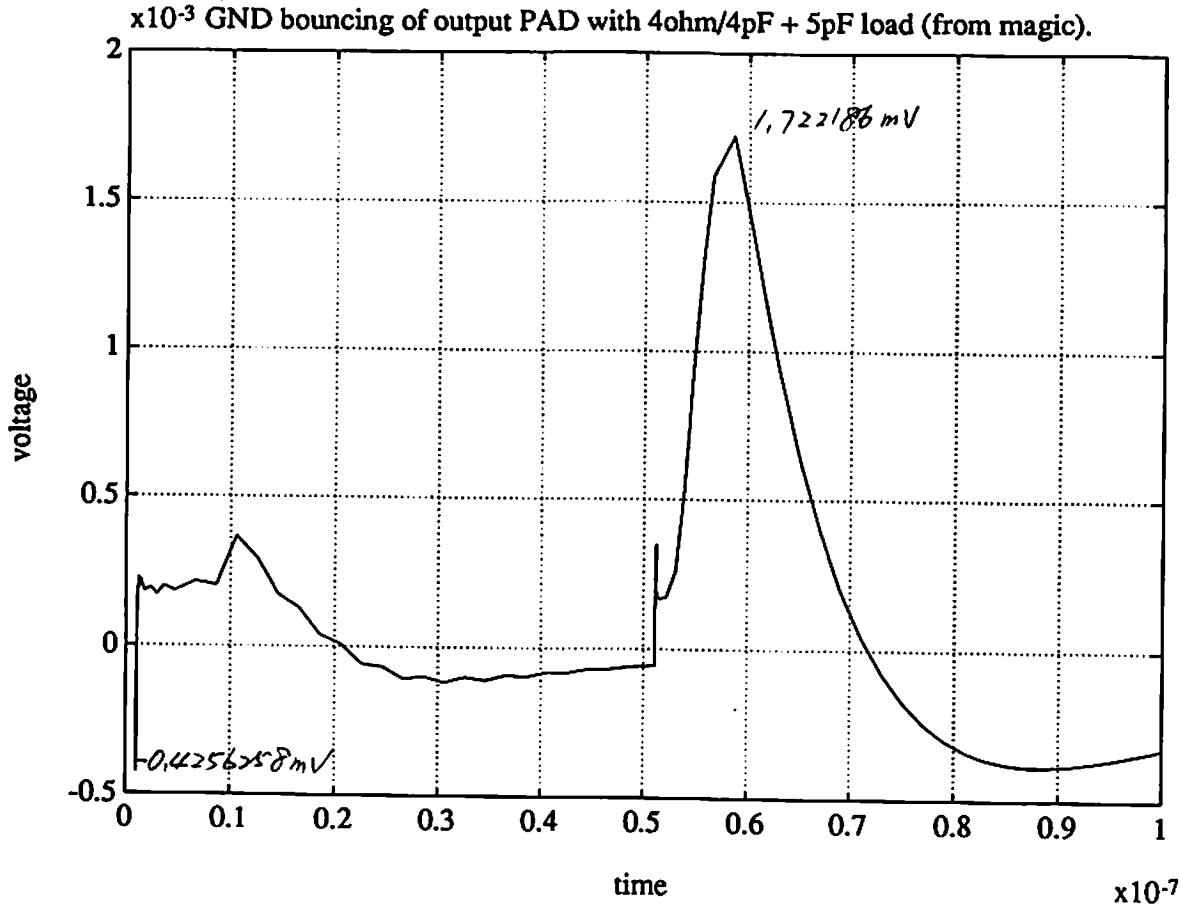
c 9 GND bouncing of input PAD with 500fF load (from magic).

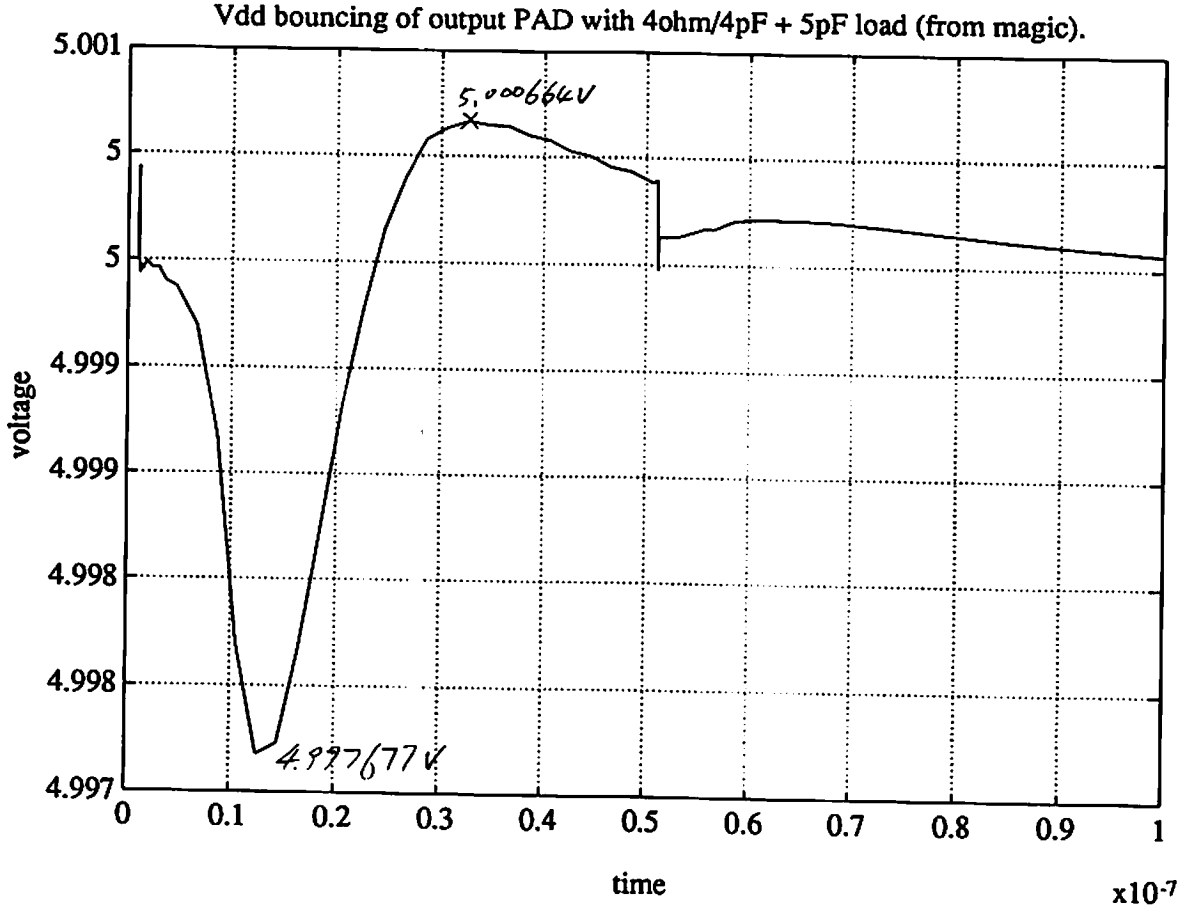


Vdd bouncing of input PAD with 500fF load (from magic).

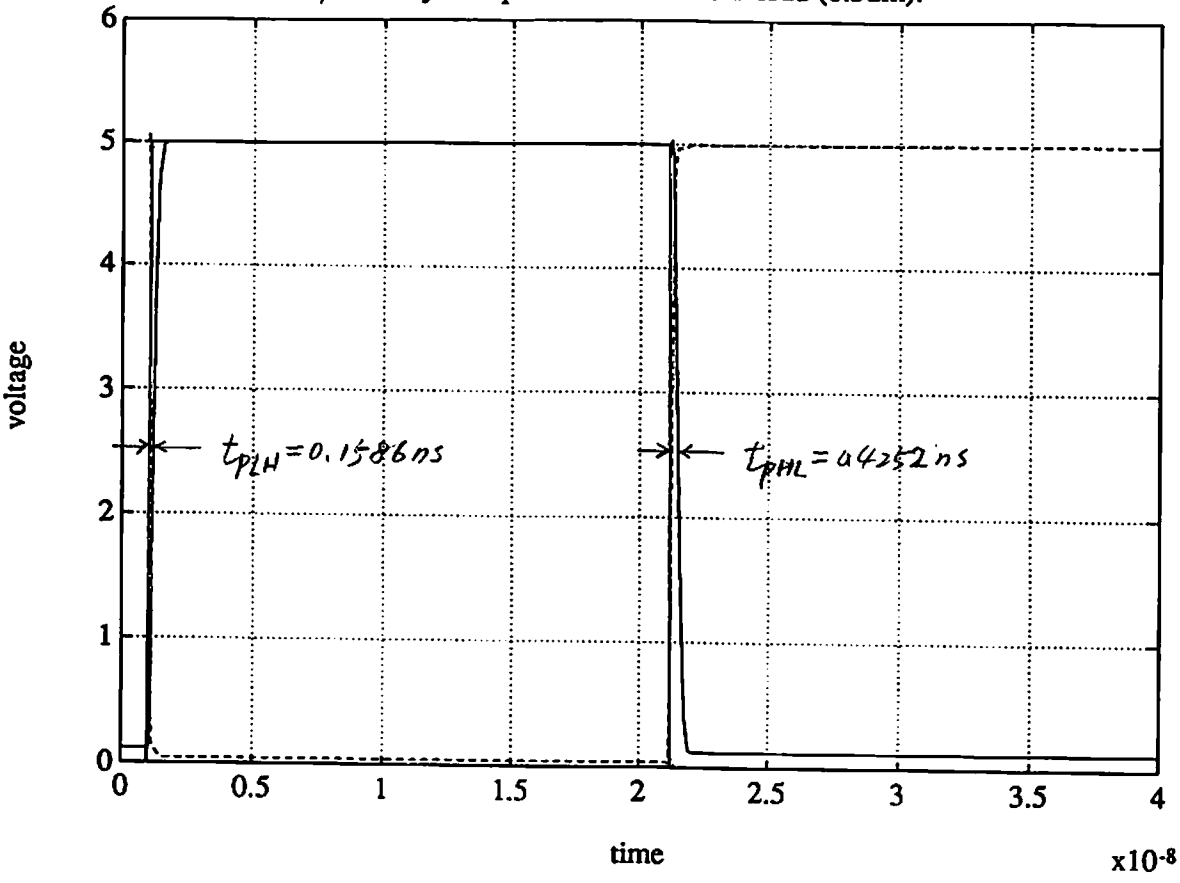


C10

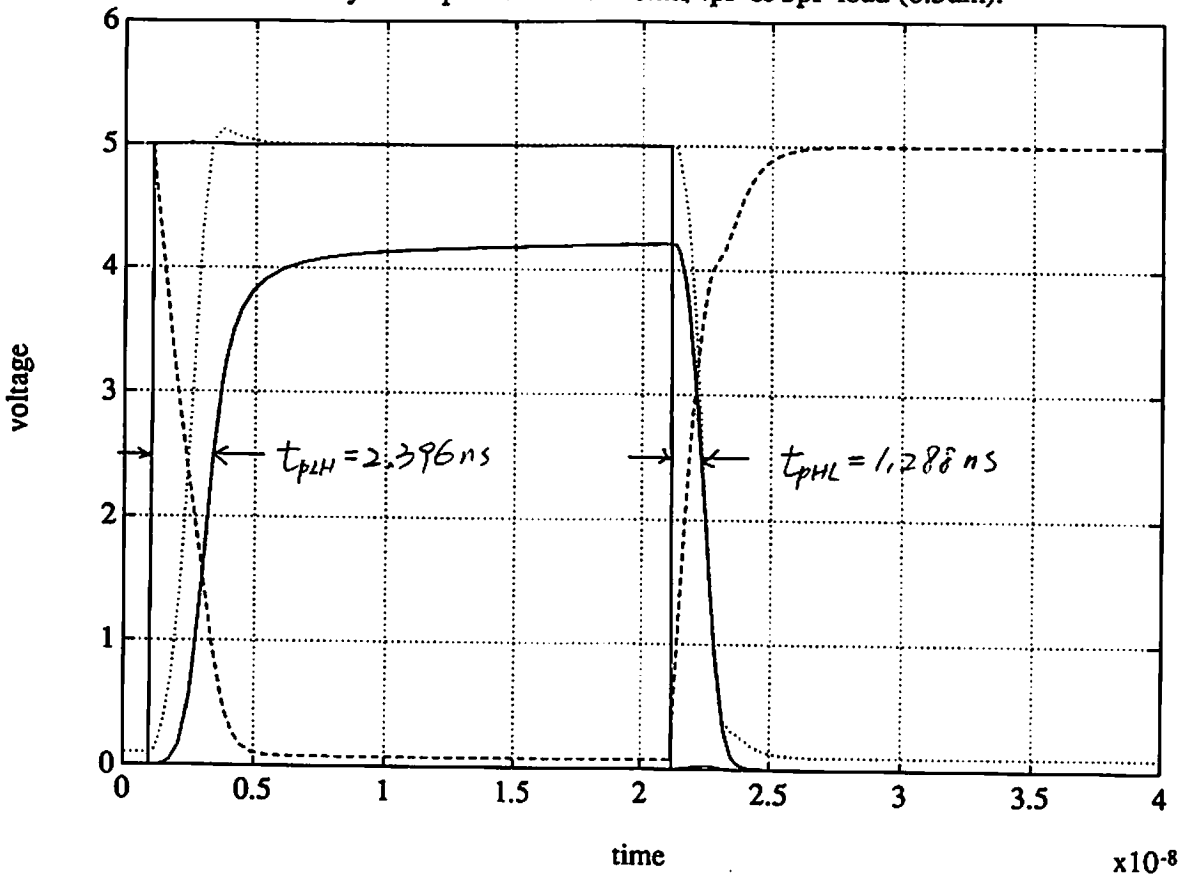

x10-3 GND bouncing of output PAD with 4ohm/4pF + 5pF load (from magic).
1.722186 mV
-0.4256258 mV
voltage
time
x10-7
2
1.5
1
0.5
0
-0.5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Vdd bouncing of output PAD with 4ohm/4pF + 5pF load (from magic).
5.000664V
4.997677V
voltage
time
x10-7
5.001
5
5
4.999
4.999
4.998
4.998
4.997
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

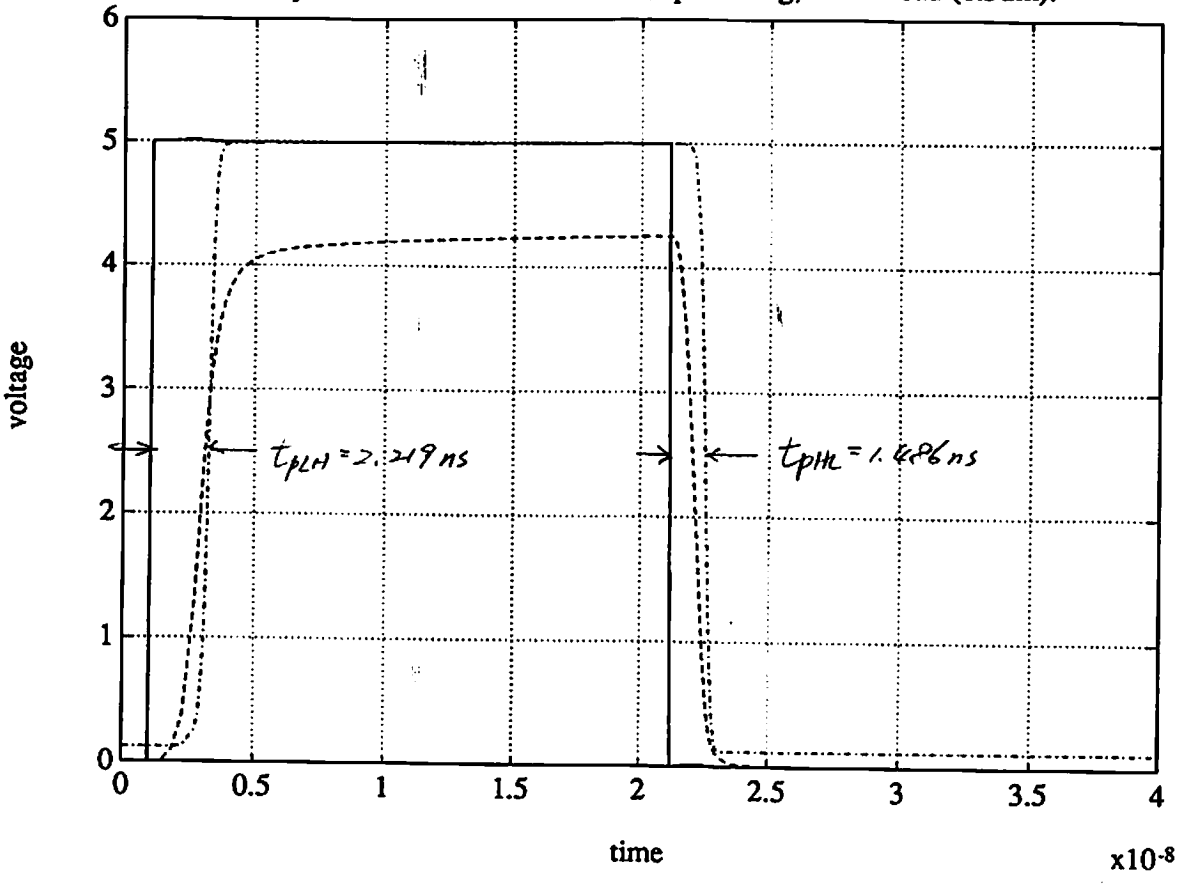
C// Delay of input PAD with 500fF load (0.5um).

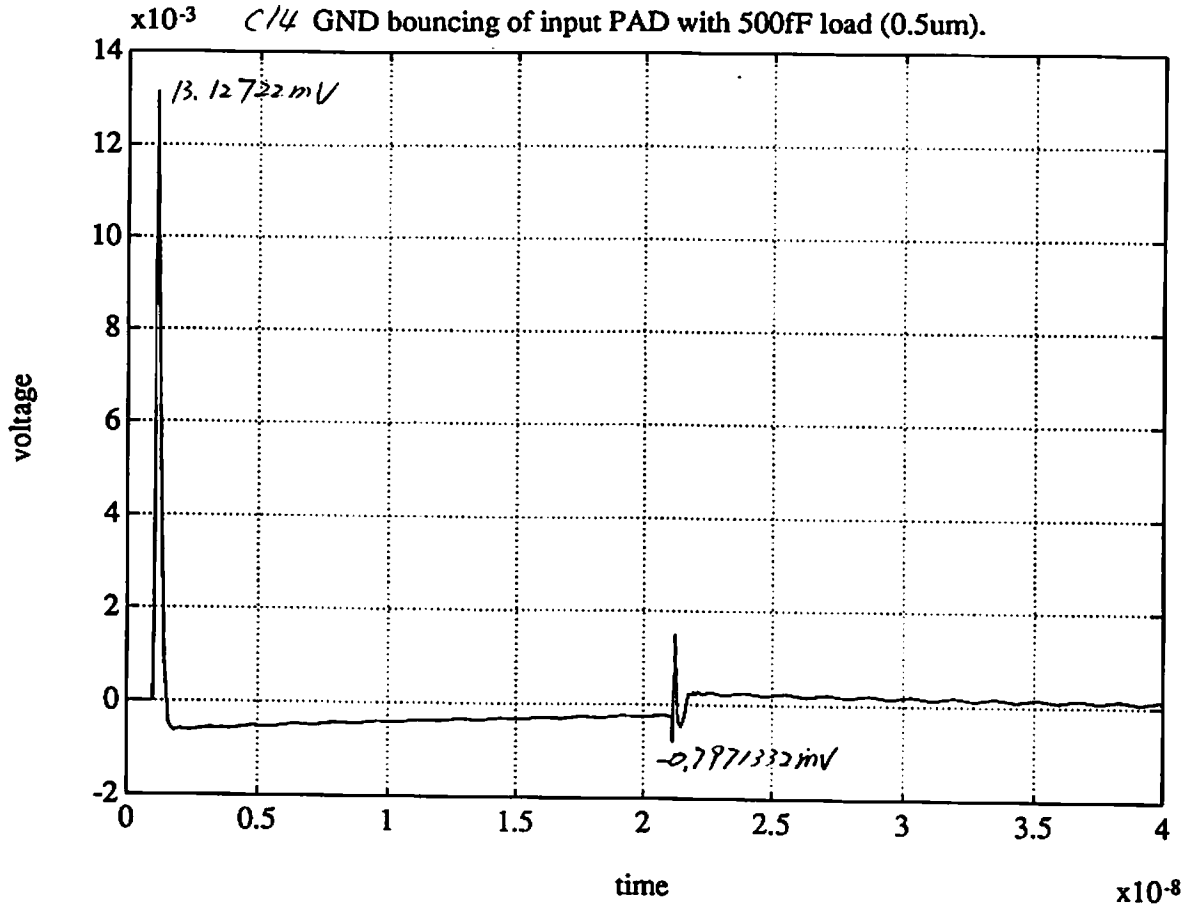


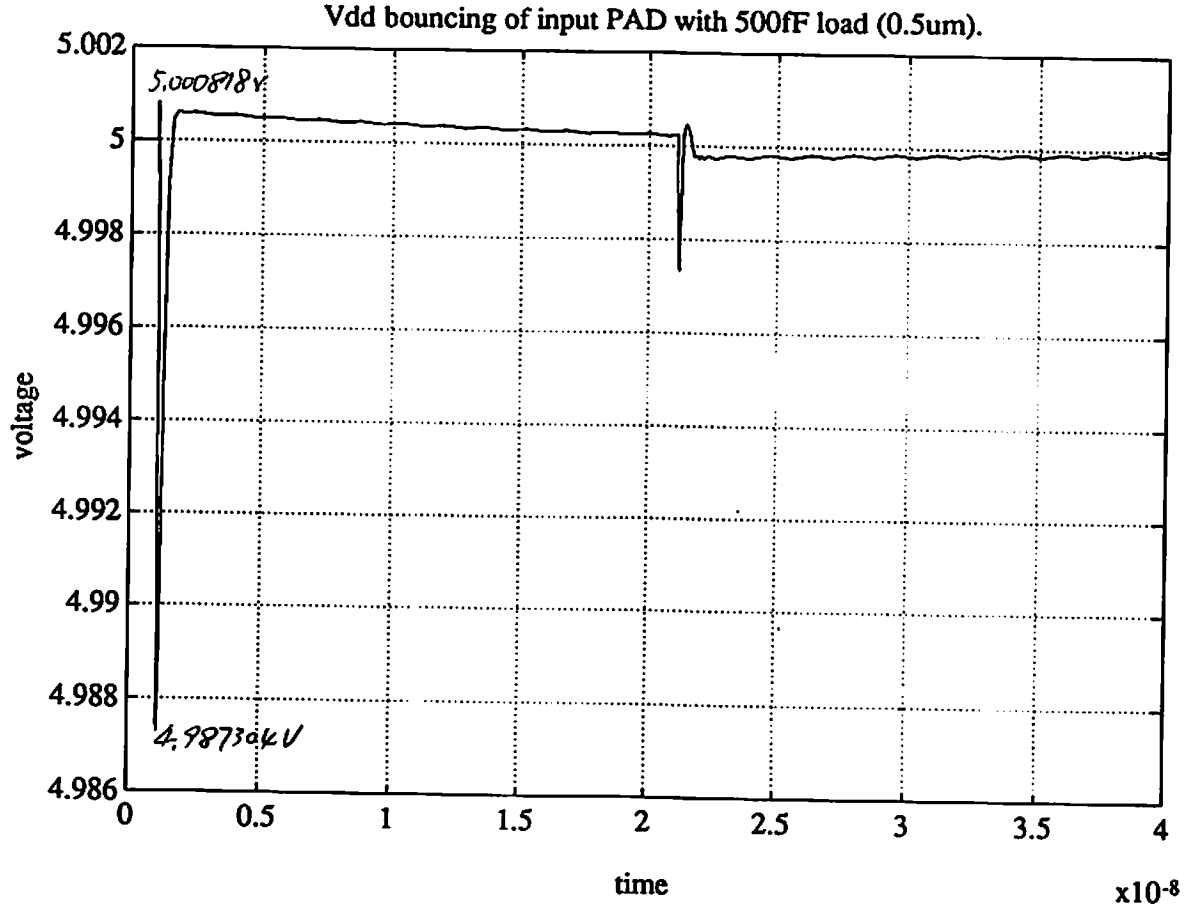
C12 Delay of output PAD with 4ohm/4pF & 5pF load (0.5um).



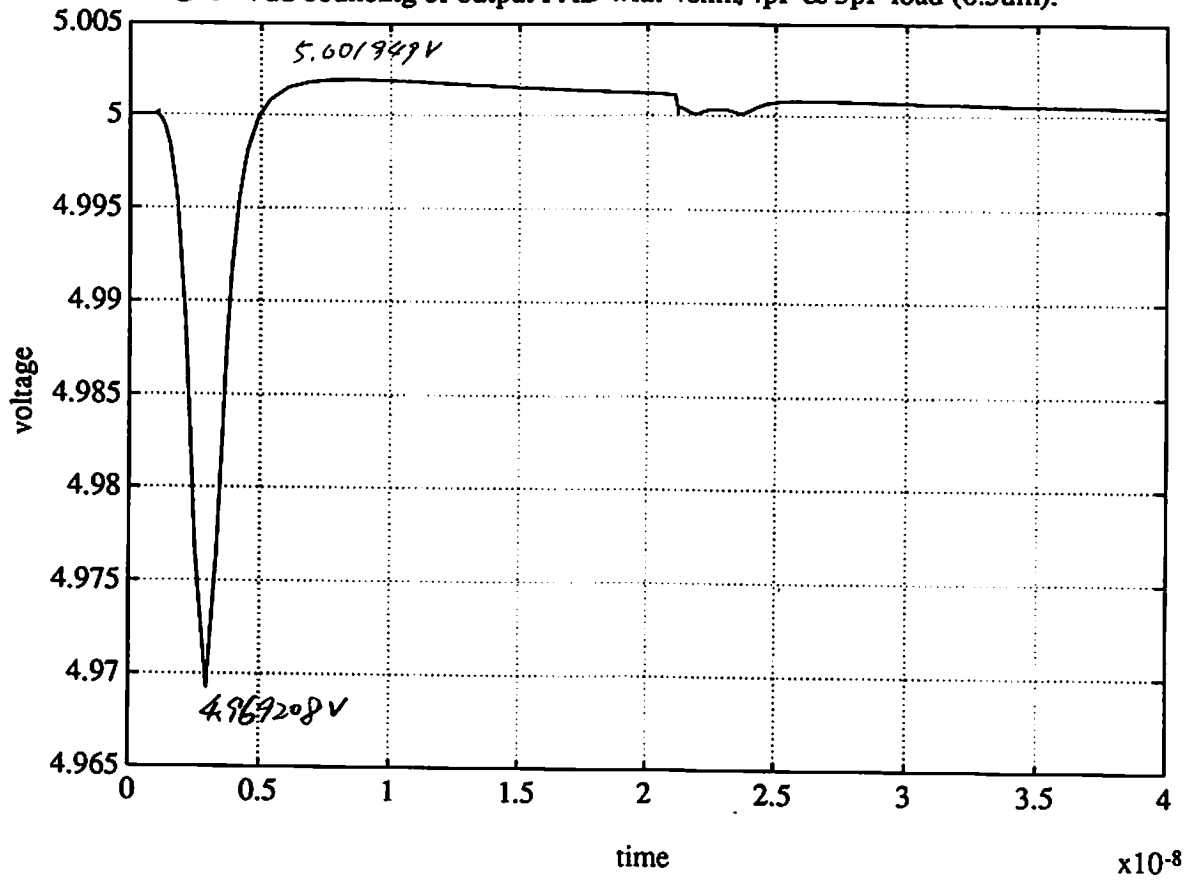
C13 Delay of PAD-to-PAD with 4ohm/4pF wiring, 500fF load (0.5um).



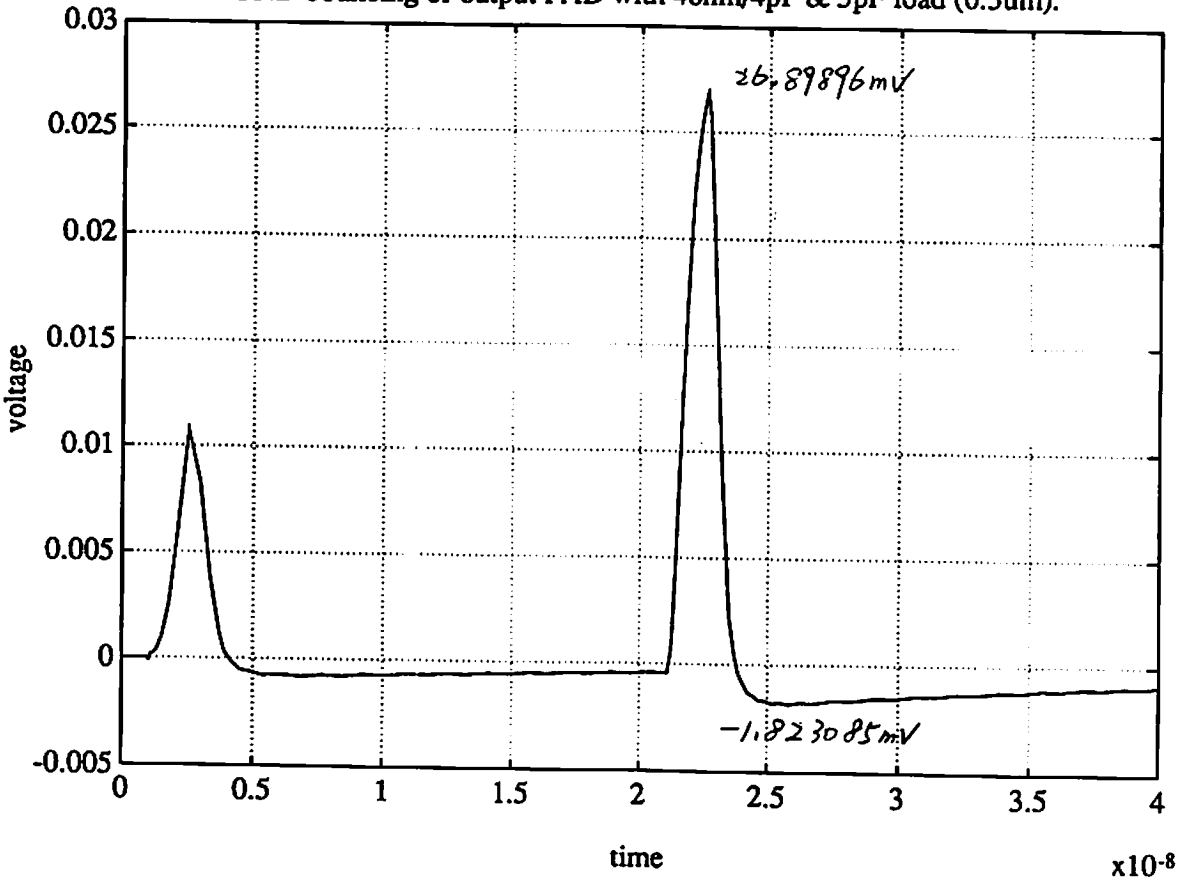

x10-3
C14 GND bouncing of input PAD with 500fF load (0.5um).
13.12722mV
-0.7971332mV
voltage
time
x10-8


Vdd bouncing of input PAD with 500fF load (0.5um).
5.000818V
4.987304V
voltage
time
x10-8

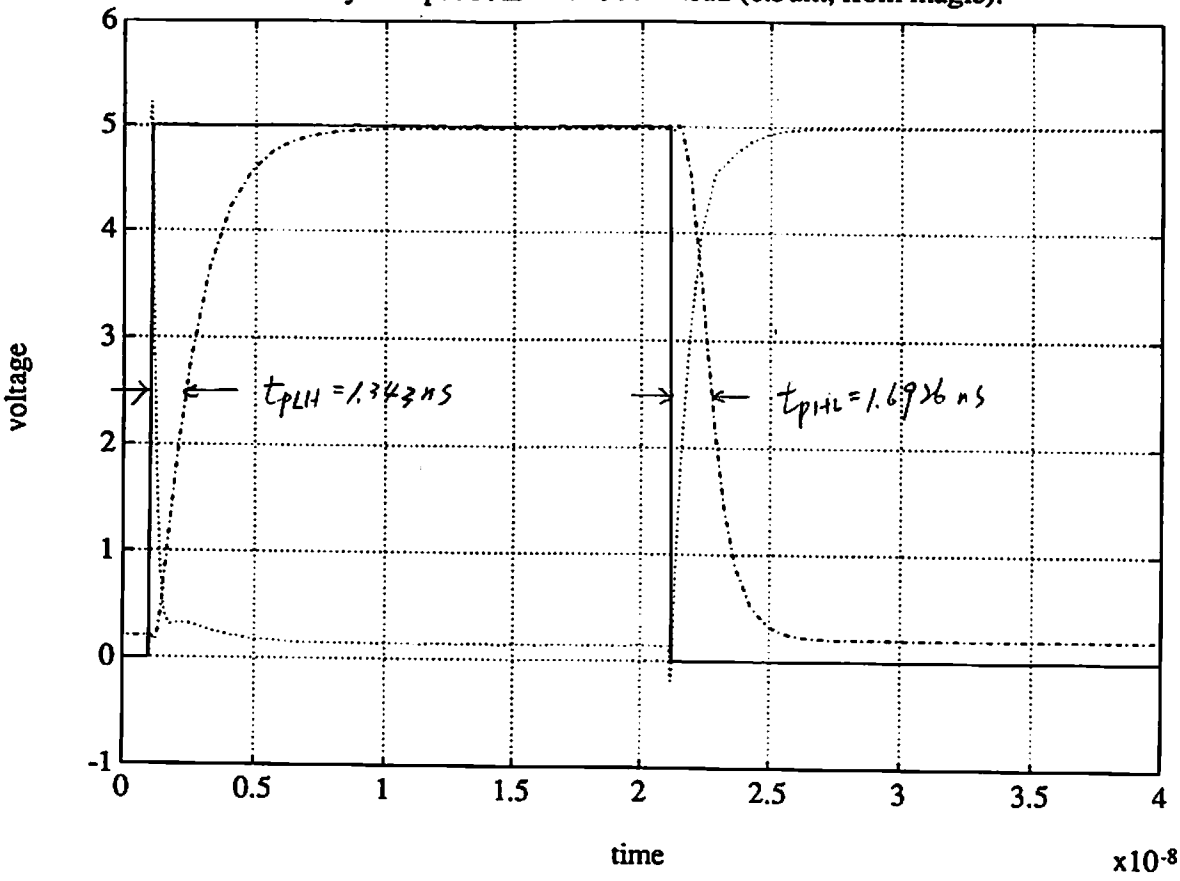
C15 Vdd bouncing of output PAD with 4ohm/4pF & 5pF load (0.5um).



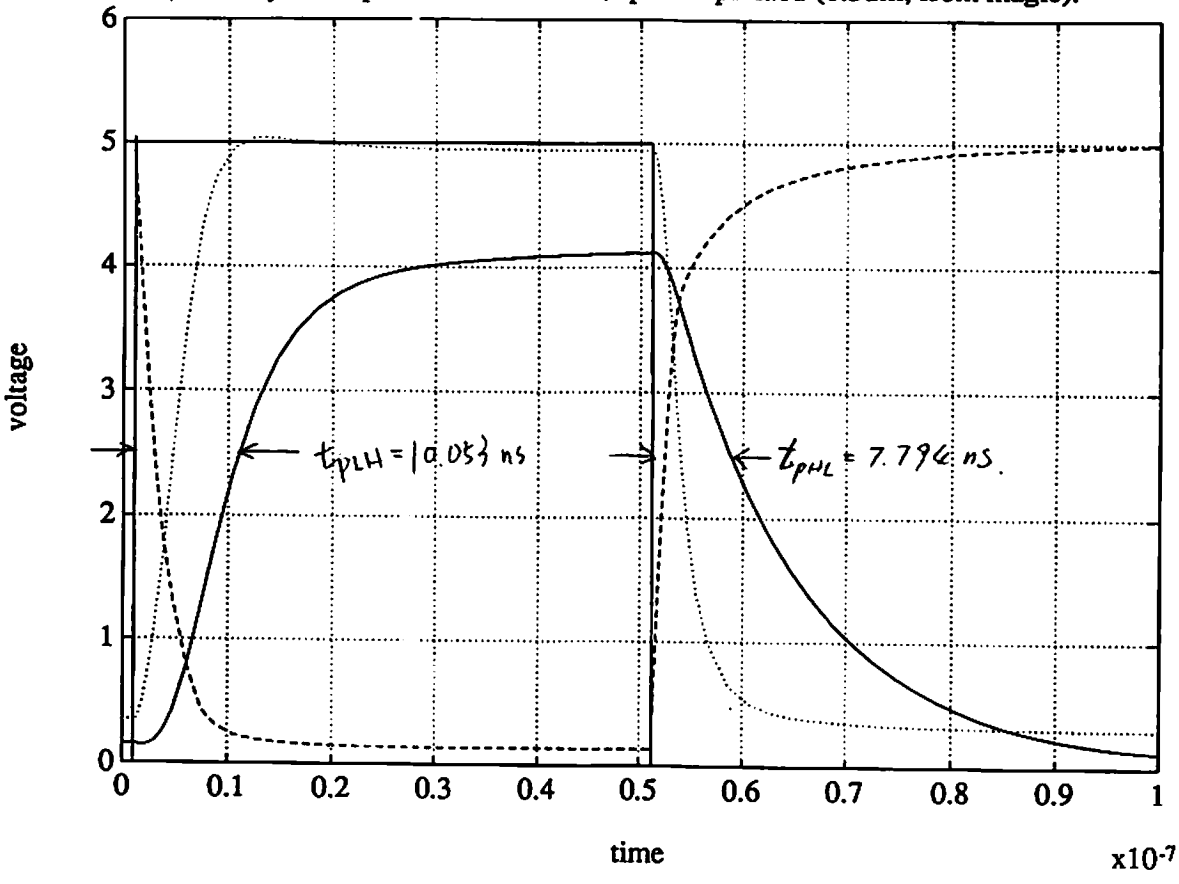
GND bouncing of output PAD with 4ohm/4pF & 5pF load (0.5um).



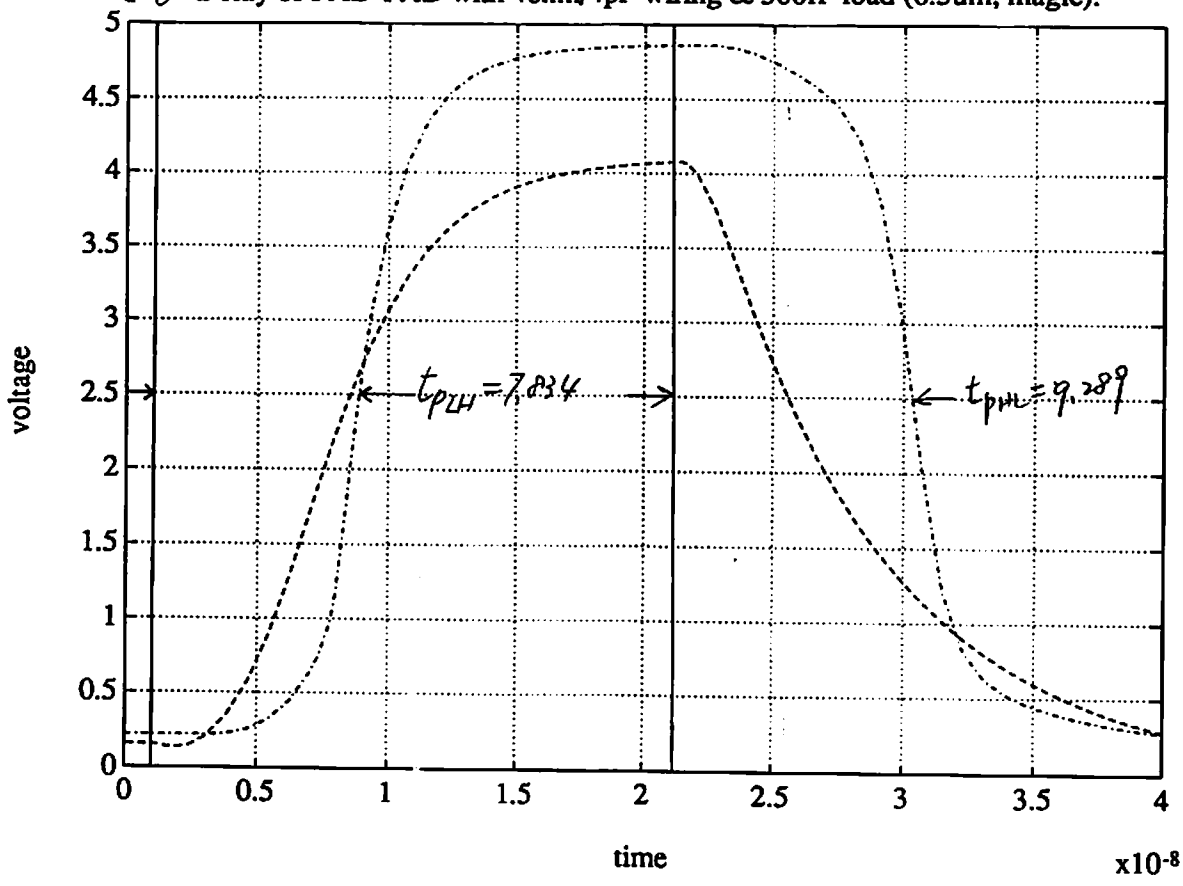
C16 Delay of input PAD with 500fF load (0.5um, from magic).



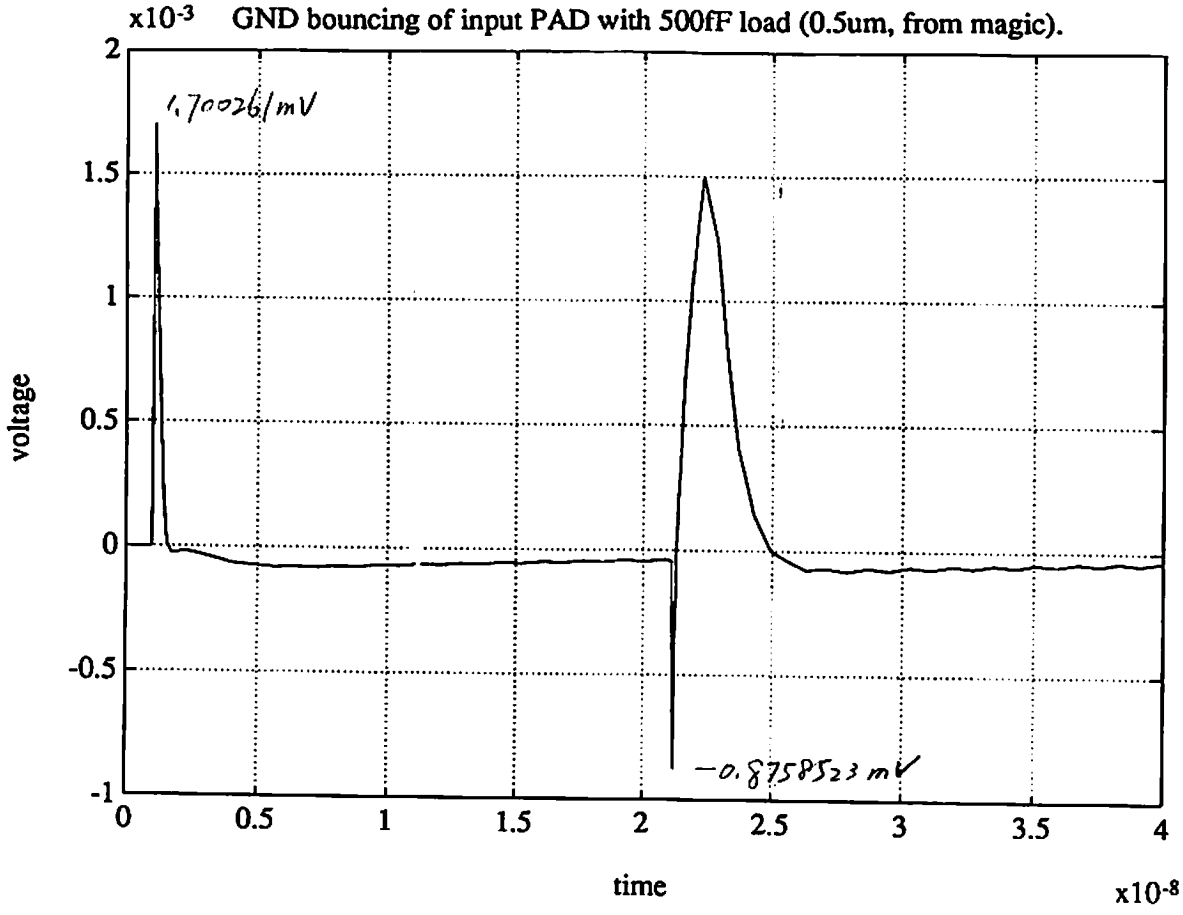
C17 Delay of output PAD with 4ohm/4pF & 5pF laod (0.5um, from magic).

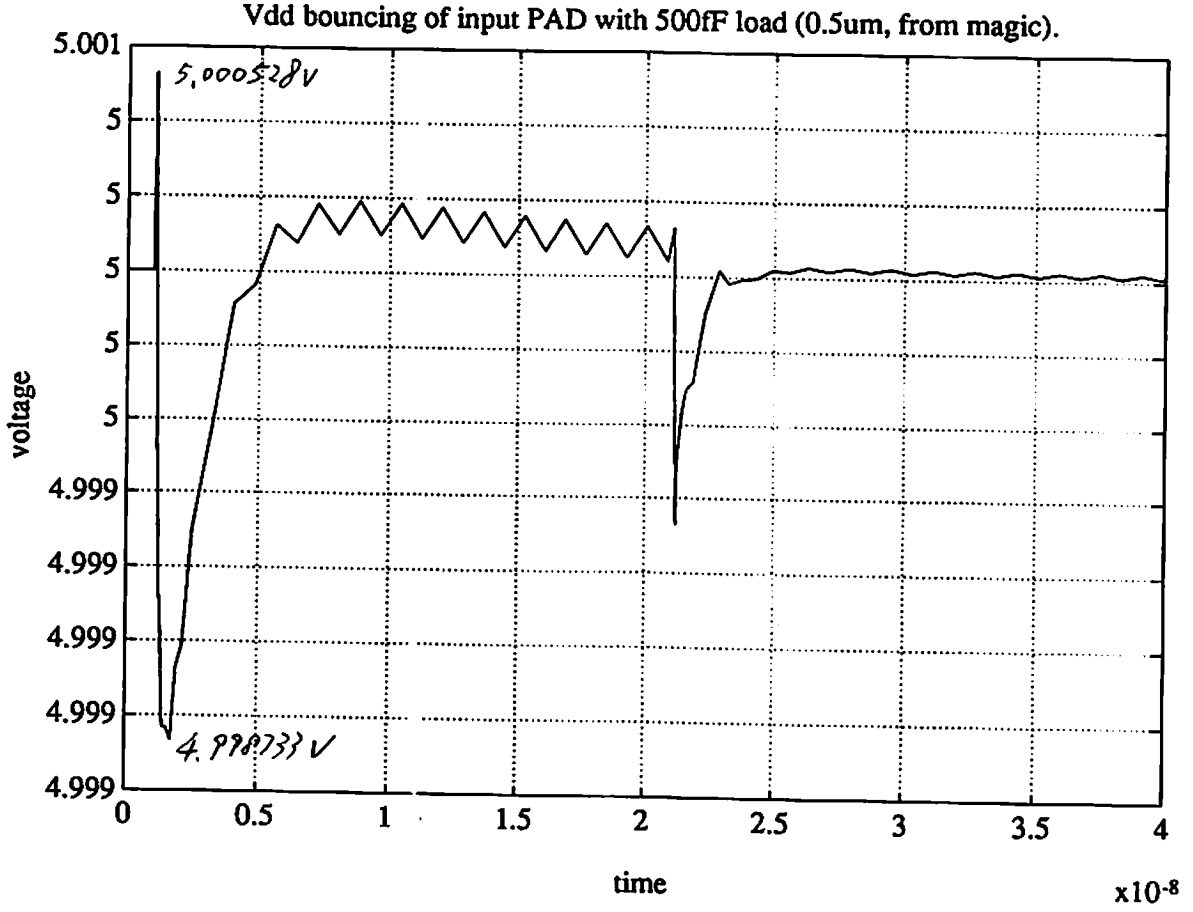


C18 Delay of PAD-PAD with 4ohm/4pF wiring & 500fF load (0.5um, magic).

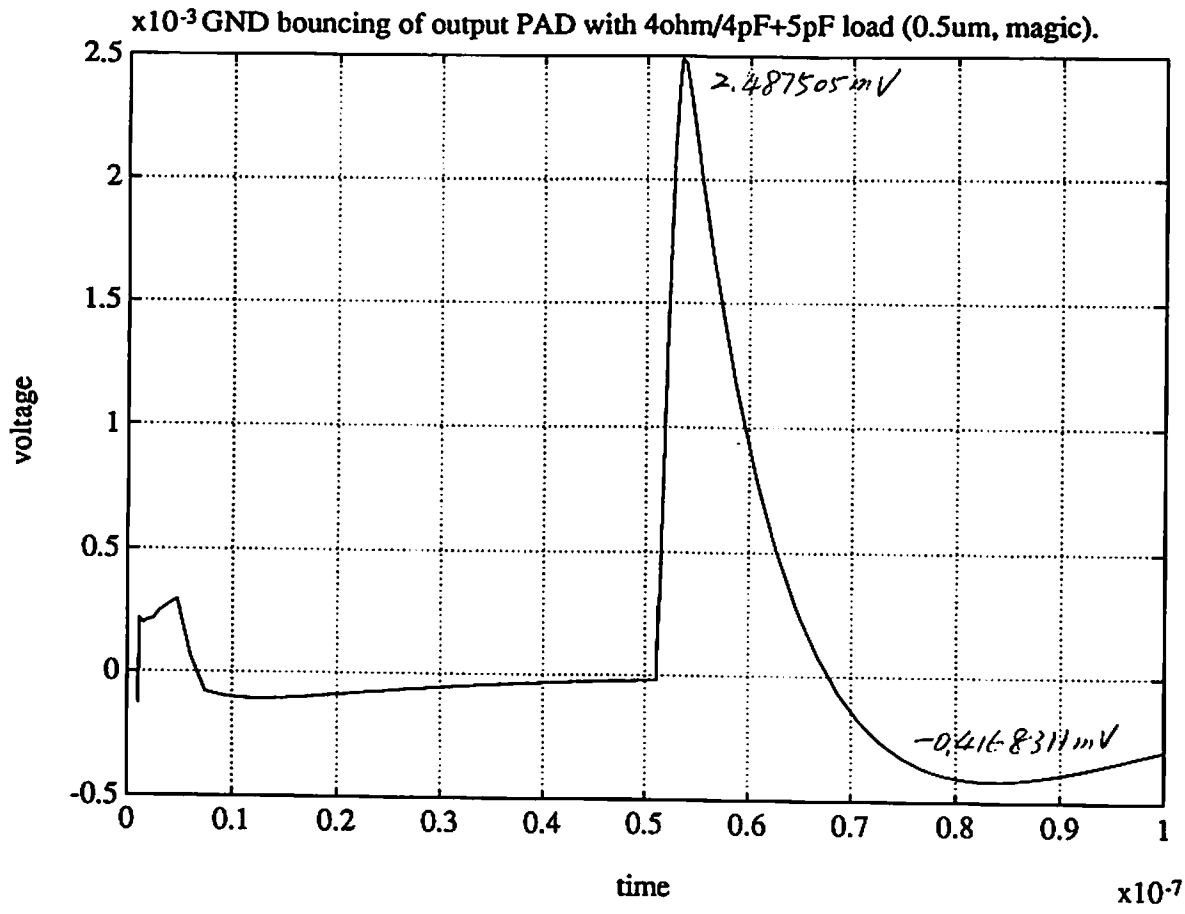


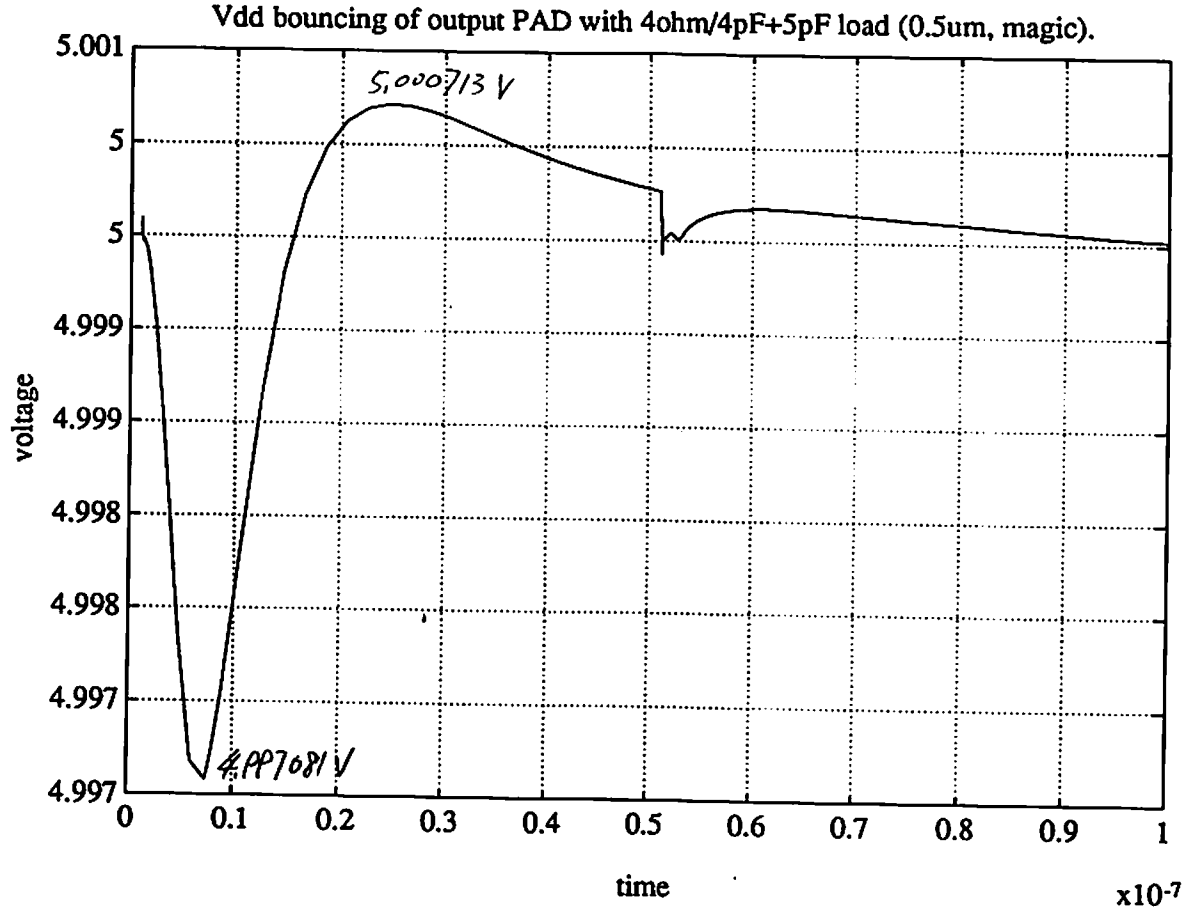
C19


x10-3
GND bouncing of input PAD with 500fF load (0.5um, from magic).
1.700261mV
-0.875823 mV
voltage
time
x10-8

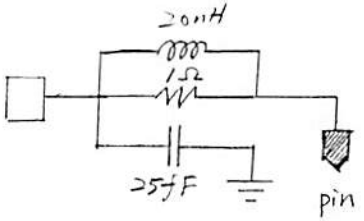

Vdd bouncing of input PAD with 500fF load (0.5um, from magic).
5.000528V
4.998733 V
voltage
time
x10-8

C20


x10-3 GND bouncing of output PAD with 4ohm/4pF+5pF load (0.5um, magic).
2.487505 mV
-0.4168311 mV
voltage
time
x10-7
2.5
2
1.5
1
0.5
0
-0.5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

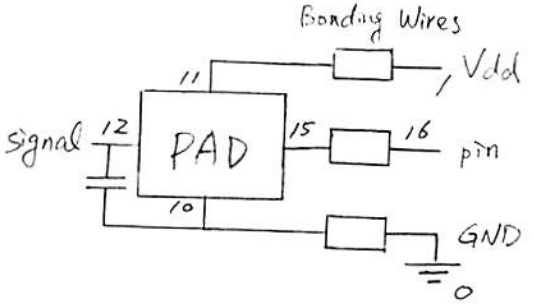

Vdd bouncing of output PAD with 4ohm/4pF+5pF load (0.5um, magic).
5.000713 V
4.997081 V
voltage
time
x10-7
5.001
5
5
4.999
4.999
4.998
4.998
4.997
4.997
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

```
** MOS model: 1.2um technology
**
.MODEL nfet NMOS LEVEL=3 PHI=0.6 TOX=2.25E-8 XJ=0.2U TPG=1
+ VTO=0.8186 DELTA=1.757 LD=1.134E-7 KP=9.1547E-5
+ UO=596.5 THETA=0.1085 GAMMA=0.5266 NSUB=1.968E+16
+ NFS=5.5E+12 VMAX=1.942E+5 ETA=6.654E-2 KAPPA=0.1121
+ CGDO=2.6106E-10 CGSO=2.6106E-10 CGBO=6.3402E-10
+ CJ=3.1146E-4 MJ=1.0667 CJSW=4.3777E-10 MJSW=0.15423 PB=0.8
* Weff = Wdrawn - Delta_W
* The suggested Delta_W is 1.997E-7
.MODEL pfet PMOS LEVEL=3 PHI=0.6 TOX=2.25E-8 XJ=0.2U TPG=-1
+ VTO=-0.9456 DELTA=1.552 LD=1.172E-8 KP=3.1646E-5
+ UO=206.2 THETA=1.69E-1 GAMMA=0.4619 NSUB=1.514E+16
+ NFS=4.999E+12 VMAX=4.441E+5 ETA=1.635E-1 KAPPA=10
+ CGDO=2.6981E-11 CGSO=2.6981E-11 CGBO=8.6508E-10
+ CJ=4.7864E-4 MJ=0.4973 CJSW=1.4771E-10 MJSW=0.190593 PB=0.85
* Weff = Wdrawn - Delta_W
* The suggested Delta_W is 3.128E-7

** DIODE model
**
.MODEL diode D IS=1E-14 RS=10 N=1 TT=1E-10 CJO=2E-..
+ VJ=0.6 M=0.5 EG=1.11 XTI=3 BV=40 TNOM=50

M0 12 13 11 11 pfet l=1.2u w=90u
M1 13 14 11 11 pfet l=1.2u w=30u
M2 12 13 10 10 nfet l=1.2u w=30u
M3 13 14 10 10 nfet l=1.2u w=28.8u
R1 14 15 200
D1 14 11 DIODE 1774.08p
D2 10 14 DIODE 1774.08p

** Bonding wires
RB0 10 0 1
CB0 10 0 25f
LB0 10 0 20n
RB1 11 1 1
CB1 11 0 25f
LB1 11 1 20n
RB2 15 16 1
CB2 15 0 25f
LB2 15 16 20n

** Load of Vin
CL 12 10 500f

Vdd 1 0 DC 5v
Vt 16 0 DC 5v pulse(0v 5v 1ns 100ps 100ps 20ns 40ns)
.TRAN 100ps 40ns
.END
```

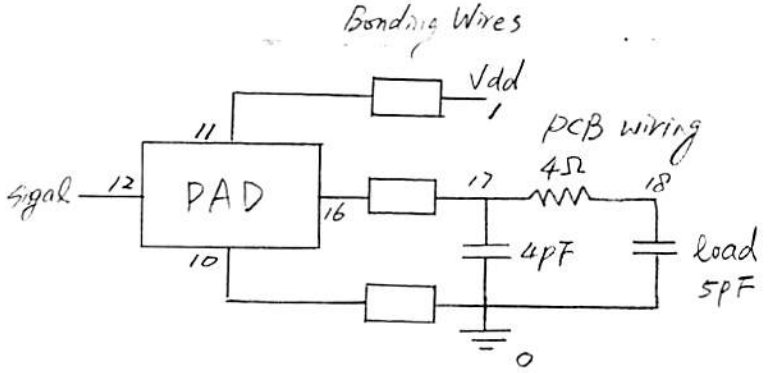


Bonding Wire
(parameters are for the worst case).

The areas of clamping diodes are from magic layout.



```
** MOS model: 1.2um technology
**
.MODEL nfet NMOS LEVEL=3 PHI=0.6 TOX=2.25E-8 XJ=0.2U TPG=1
+ VTO=0.8186 DELTA=1.757 LD=1.134E-7 KP=9.1547E-5
+ UO=596.5 THETA=0.1085 GAMMA=0.5266 NSUB=1.968E+16
+ NFS=5.5E+12 VMAX=1.942E+5 ETA=6.654E-2 KAPPA=0.1121
+ CGDO=2.6106E-10 CGSO=2.6106E-10 CGBO=6.3402E-10
+ CJ=3.1146E-4 MJ=1.0667 CJSW=4.3777E-10 MJSW=0.15423 PB=0.8
* Weff = Wdrawn - Delta_W
* The suggested Delta_W is 1.997E-7
.MODEL pfet PMOS LEVEL=3 PHI=0.6 TOX=2.25E-8 XJ=0.2U TPG=-1
+ VTO=-0.9456 DELTA=1.552 LD=1.172E-8 KP=3.1646E-5
+ UO=206.2 THETA=1.69E-1 GAMMA=0.4619 NSUB=1.514E+16
+ NFS=4.999E+12 VMAX=4.441E+5 ETA=1.635E-1 KAPPA=10
+ CGDO=2.6981E-11 CGSO=2.6981E-11 CGBO=8.6508E-10
+ CJ=4.7864E-4 MJ=0.4973 CJSW=1.4771E-10 MJSW=0.190593 PB=0.85
* Weff = Wdrawn - Delta_W
* The suggested Delta_W is 3.128E-7

M0 13 12 11 11 pfet l=1.2u w=4.8u
M1 13 12 10 10 nfet l=1.2u w=2.4u
M2 14 13 11 11 pfet l=1.2u w=24u
M3 14 13 10 10 nfet l=1.2u w=9.6u
M4 15 14 11 10 nfet l=1.2u w=360u
M5 15 13 10 10 nfet l=1.2u w=240u
R15 15 16 1
C15 15 10 1p

** Bonding wires
RB0 10 0 1
CB0 10 0 25f
LB0 10 0 20n
RB1 11 1 1
CB1 11 0 25f
LB1 11 1 20n
RB2 16 17 1
CB2 16 0 25f
LB2 16 17 20n

** PCB wiring and 5pF load
RW 17 18 4
CW 17 0 4p
CL 18 0 5p

Vdd 1 0 DC 5v
Vt 12 10 DC 5v pulse(0v 5v 1ns 100ps 100ps 20ns 40ns)
.TRAN 100ps 40ns
.END
```



```
** MOS model: 1.2um technology
**
.MODEL nfet NMOS LEVEL=3 PHI=0.6 TOX=2.25E-8 XJ=0.2U TPG=1
+ VTO=0.8186 DELTA=1.757 LD=1.134E-7 KP=9.1547E-5
+ UO=596.5 THETA=0.1085 GAMMA=0.5266 NSUB=1.968E+16
+ NFS=5.5E+12 VMAX=1.942E+5 ETA=6.654E-2 KAPPA=0.1121
+ CGDO=2.6106E-10 CGSO=2.6106E-10 CGBO=6.3402E-10
+ CJ=3.1146E-4 MJ=1.0667 CJSW=4.3777E-10 MJSW=0.15423 PB=0.8
* Weff = Wdrawn - Delta_W
* The suggested Delta_W is 1.997E-7
.MODEL pfet PMOS LEVEL=3 PHI=0.6 TOX=2.25E-8 XJ=0.2U TPG=-1
+ VTO=-0.9456 DELTA=1.552 LD=1.172E-8 KP=3.1646E-5
+ UO=206.2 THETA=1.69E-1 GAMMA=0.4619 NSUB=1.514E+16
+ NFS=4.999E+12 VMAX=4.441E+5 ETA=1.635E-1 KAPPA=10
+ CGDO=2.6981E-11 CGSO=2.6981E-11 CGBO=8.6508E-10
+ CJ=4.7864E-4 MJ=0.4973 CJSW=1.4771E-10 MJSW=0.190593 PB=0.85
* Weff = Wdrawn - Delta_W
* The suggested Delta_W is 3.128E-7

** DIODE model
**
.MODEL diode D IS=1E-14 RS=10 N=1 TT=1E-10 CJO=2E-12
+ VJ=0.6 M=0.5 EG=1.11 XTI=3 BV=40 TNOM=50

** input pad subcircuit
**
.SUBCKT inpad 1 10 16 12
M0 12 13 11 11 pfet l=1.2u w=90u
M1 13 14 11 11 pfet l=1.2u w=30u
M2 12 13 10 10 nfet l=1.2u w=30u
M3 13 14 10 10 nfet l=1.2u w=28.8u
R1 14 15 200
D1 14 11 DIODE 1774.08p
D2 10 14 DIODE 1774.08p

** Bonding wires
RB0 10 0 1
CB0 10 0 25f
LB0 10 0 20n
RB1 11 1 1
CB1 11 0 25f
LB1 11 1 20n
RB2 15 16 1
CB2 15 0 25f
LB2 15 16 20n
.ENDS inpad

** output pad subcircuit
**
.SUBCKT outpad 1 10 12 17
M0 13 12 11 11 pfet l=1.2u w=4.8u
M1 13 12 10 10 nfet l=1.2u w=2.4u
M2 14 13 11 11 pfet l=1.2u w=24u
M3 14 13 10 10 nfet l=1.2u w=9.6u
M4 15 14 11 10 nfet l=1.2u w=360u
M5 15 13 10 10 nfet l=1.2u w=240u
R15 15 16 1
C15 15 10 10p

** Bonding wires
RB0 10 0 1
CB0 10 0 25f
LB0 10 0 20n
RB1 11 1 1
CB1 11 0 25f
LB1 11 1 20n
RB2 16 17 1
CB2 16 0 25f
LB2 16 17 20n
.ENDS outpad

X1 1 10 2 3 outpad
X2 1 20 4 5 inpad

** PCB wiring
RW 3 4 4
CW 3 0 4p

** Load of Vin
CL 5 20 500f

Vdd 1 0 DC 5v
Vt 2 10 DC 5v pulse(0v 5v 1ns 100ps 100ps 20ns 40ns)
.TRAN 100ps 40ns
.END
```

```
** MOS model: 1.2um technology
**
.MODEL nfet NMOS LEVEL=3 PHI=0.6 TOX=2.25E-8 XJ=0.2U TPG=1
+ VTO=0.8186 DELTA=1.757 LD=1.134E-7 KP=9.1547E-5
+ UO=596.5 THETA=0.1085 GAMMA=0.5266 NSUB=1.968E+16
+ NFS=5.5E+12 VMAX=1.942E+5 ETA=6.654E-2 KAPPA=0.1121
+ CGDO=2.6106E-10 CGSO=2.6106E-10 CGBO=6.3402E-10
+ CJ=3.1146E-4 MJ=1.0667 CJSW=4.3777E-10 MJSW=0.15423 PB=0.8
* Weff = Wdrawn - Delta_W
* The suggested Delta_W is 1.997E-7
.MODEL pfet PMOS LEVEL=3 PHI=0.6 TOX=2.25E-8 XJ=0.2U TPG=-1
+ VTO=-0.9456 DELTA=1.552 LD=1.172E-8 KP=3.1646E-5
+ UO=206.2 THETA=1.69E-1 GAMMA=0.4619 NSUB=1.514E+16
+ NFS=4.999E+12 VMAX=4.441E+5 ETA=1.635E-1 KAPPA=10
+ CGDO=2.6981E-11 CGSO=2.6981E-11 CGBO=8.6508E-10
+ CJ=4.7864E-4 MJ=0.4973 CJSW=1.4771E-10 MJSW=0.190593 PB=0.85
* Weff = Wdrawn - Delta_W
* The suggested Delta_W is 3.128E-7

** SPICE file created for circuit pad6Bin
** Technology: scmos
**

** NODE: 0 = GND
** NODE: 1 = Vdd
** NODE: 2 = Error
RLUMP0 100 101 1149.5
RLUMP1 102 103 2333.0
RLUMP2 104 105 7813.5
M0 101 103 105 146 pfet L=1.2U W=22.2U
RLUMP3 104 106 7813.5
RLUMP4 102 107 2333.0
RLUMP5 100 108 1149.5
M1 106 107 108 146 pfet L=1.2U W=22.2U
RLUMP6 100 109 1149.5
RLUMP7 102 110 2333.0
RLUMP8 104 111 7813.5
M2 109 110 111 146 pfet L=1.2U W=22.8U
RLUMP9 104 112 7813.5
RLUMP10 102 113 2333.0
RLUMP11 100 114 1149.5
M3 112 113 114 146 pfet L=1.2U W=22.8U
RLUMP12 102 115 2333.0
RLUMP13 116 117 2433.0
RLUMP14 104 118 7813.5
M4 115 117 118 146 pfet L=1.2U W=15.0U
RLUMP15 104 119 7813.5
RLUMP16 116 120 2433.0
RLUMP17 102 121 2333.0
M5 119 120 121 146 pfet L=1.2U W=15.0U
RLUMP18 100 122 1149.5
RLUMP19 102 123 2333.0
RLUMP20 124 125 3708.0
M6 122 123 125 145 nfet L=1.2U W=7.2U
RLUMP21 124 126 3708.0
RLUMP22 102 127 2333.0
RLUMP23 100 128 1149.5
M7 126 127 128 145 nfet L=1.2U W=7.2U
RLUMP24 100 129 1149.5
RLUMP25 102 130 2333.0
RLUMP26 124 131 3708.0
M8 129 130 131 145 nfet L=1.2U W=7.8U
RLUMP27 124 132 3708.0
RLUMP28 102 133 2333.0
RLUMP29 100 134 1149.5
M9 132 133 134 145 nfet L=1.2U W=7.8U
RLUMP30 102 135 2333.0
RLUMP31 116 136 2433.0
RLUMP32 124 137 3708.0
M10 135 136 137 145 nfet L=1.2U W=9.6U
RLUMP33 124 138 3708.0
RLUMP34 116 139 2433.0
RLUMP35 102 140 2333.0
M11 138 139 140 145 nfet L=1.2U W=9.6U
RLUMP36 102 141 2333.0
RLUMP37 116 142 2433.0
RLUMP38 124 143 3708.0
M12 141 142 143 145 nfet L=1.2U W=9.6U
** NODE: 144 = 10_114_16# ===FLOATING===
C0 145 145 229F
** NODE: 145 = GND_br ===FLOATING===
C1 146 145 238F
** NODE: 146 = Vdd_br ===FLOATING===
C2 124 145 122F
** NODE: 124 = GND_tr
C3 116 145 2336F
** NODE: 116 = pad
C4 100 145 130F
** NODE: 100 = in
C5 102 145 83F
** NODE: 102 = 6_308_974#
C6 104 145 274F
** NODE: 104 = Vdd_tr
** NODE: 0 = GND!
** NODE: 1 = Vdd!

** Bonding wires
RB0 145 0 1
CB0 145 0 25f
LB0 145 0 20n
RB1 146 1 1
CB1 146 0 25f
LB1 146 1 20n
RB2 116 2 1
CB2 116 0 25f
LB2 116 2 20n

** Load of Vin
C100 100 0 500F

Vdd 1 0 DC 5v
Vgt 124 145 0v
Vvt 104 146 0v
Vt 2 0 DC 5v pulse(0v 5v 1ns 100ps 100ps 20ns 40ns)
.TRAN 100ps 40ns
.END
```

```
** MOS model: 1.2um technology
**
.MODEL nfet NMOS LEVEL=3 PHI=0.6 TOX=2.25E-8 XJ=0.2U TPG=1
+ VTO=0.8186 DELTA=1.757 LD=1.134E-7 KP=9.1547E-5
+ UO=596.5 THETA=0.1085 GAMMA=0.5266 NSUB=1.968E+16
+ NFS=5.5E+12 VMAX=1.942E+5 ETA=6.654E-2 KAPPA=0.1121
+ CGDO=2.6106E-10 CGSO=2.6106E-10 CGBO=6.3402E-10
+ CJ=3.1146E-4 MJ=1.0667 CJSW=4.3777E-10 MJSW=0.15423 PB=0.8
* Weff = Wdrawn - Delta_W
* The suggested Delta_W is 1.997E-7
.MODEL pfet PMOS LEVEL=3 PHI=0.6 TOX=2.25E-8 XJ=0.2U TPG=-1
+ VTO=-0.9456 DELTA=1.552 LD=1.172E-8 KP=3.1646E-5
+ UO=206.2 THETA=1.69E-1 GAMMA=0.4619 NSUB=1.514E+16
+ NFS=4.999E+12 VMAX=4.441E+5 ETA=1.635E-1 KAPPA=10
+ CGDO=2.6981E-11 CGSO=2.6981E-11 CGBO=8.6508E-10
+ CJ=4.7864E-4 MJ=0.4973 CJSW=1.4771E-10 MJSW=0.190593 PB=0.85
* Weff = Wdrawn - Delta_W
* The suggested Delta_W is 3.128E-7

** SPICE file created for circuit pad6bo
** Technology: scmos
**

** NODE: 0 = GND
** NODE: 1 = Vdd
** NODE: 2 = Error
RLUMP0 100 101 3738.5
RLUMP1 102 103 182.5
RLUMP2 104 105 5085.0
M0 101 103 105 143 pfet L=1.2U W=4.8U
RLUMP3 106 107 4966.5
RLUMP4 100 108 3738.5
RLUMP5 104 109 5085.0
M1 107 108 109 143 pfet L=1.2U W=24.0U
RLUMP6 100 110 3738.5
RLUMP7 102 111 182.5
RLUMP8 112 113 2802.0
M2 110 111 113 117 nfet L=1.2U W=2.4U
RLUMP9 106 114 4966.5
RLUMP10 100 115 3738.5
RLUMP11 112 116 2802.0
M3 114 115 116 117 nfet L=1.2U W=9.6U
RLUMP12 117 118 7076.0
RLUMP13 100 119 3738.5
RLUMP14 120 121 3683.0
M4 118 119 121 117 nfet L=1.2U W=30.0U
RLUMP15 120 122 3683.0
RLUMP16 100 123 3738.5
RLUMP17 117 124 7076.0
M5 122 123 124 117 nfet L=1.2U W=30.0U
RLUMP18 117 125 7076.0
RLUMP19 100 126 3738.5
RLUMP20 120 127 3683.0
M6 125 126 127 117 nfet L=1.2U W=30.0U
RLUMP21 120 128 3683.0
RLUMP22 100 129 3738.5
RLUMP23 117 130 7076.0
M7 128 129 130 117 nfet L=1.2U W=30.0U
RLUMP24 117 131 7076.0
RLUMP25 100 132 3738.5
RLUMP26 120 133 3683.0
M8 131 132 133 117 nfet L=1.2U W=30.0U
RLUMP27 120 134 3683.0
RLUMP28 100 135 3738.5
RLUMP29 117 136 7076.0
M9 134 135 136 117 nfet L=1.2U W=30.0U
RLUMP30 117 137 7076.0
RLUMP31 100 138 3738.5
RLUMP32 120 139 3683.0
M10 137 138 139 117 nfet L=1.2U W=30.0U
RLUMP33 120 140 3683.0
RLUMP34 100 141 3738.5
RLUMP35 117 142 7076.0
M11 140 141 142 117 nfet L=1.2U W=30.0U
RLUMP36 143 144 2827.0
RLUMP37 106 145 4966.5
RLUMP38 120 146 3683.0
M12 144 145 146 117 nfet L=1.2U W=45.0U
RLUMP39 120 147 3683.0
RLUMP40 106 148 4966.5
RLUMP41 143 149 2827.0
M13 147 148 149 117 nfet L=1.2U W=45.0U
RLUMP42 143 150 2827.0
RLUMP43 106 151 4966.5
RLUMP44 120 152 3683.0
M14 150 151 152 117 nfet L=1.2U W=45.0U
RLUMP45 120 153 3683.0
RLUMP46 106 154 4966.5
RLUMP47 143 155 2827.0
M15 153 154 155 117 nfet L=1.2U W=45.0U
RLUMP48 143 156 2827.0
RLUMP49 106 157 4966.5
RLUMP50 120 158 3683.0
M16 156 157 158 117 nfet L=1.2U W=45.0U
RLUMP51 120 159 3683.0
RLUMP52 106 160 4966.5
RLUMP53 143 161 2827.0
M17 159 160 161 117 nfet L=1.2U W=45.0U
RLUMP54 143 162 2827.0
RLUMP55 106 163 4966.5
RLUMP56 120 164 3683.0
M18 162 163 164 117 nfet L=1.2U W=45.0U
RLUMP57 120 165 3683.0
RLUMP58 106 166 4966.5
RLUMP59 143 167 2827.0
M19 165 166 167 117 nfet L=1.2U W=45.0U
** NODE: 168 = 10_114_16# ===FLOATING===
C0 143 117 516F
** NODE: 143 = Vdd_br
C1 120 117 850F
** NODE: 120 = pad
C2 117 117 436F
** NODE: 117 = GND_br
C3 112 117 72F
** NODE: 112 = GND_tr
C4 106 117 93F
** NODE: 106 = 6_264_486#
C5 100 117 62F
** NODE: 100 = 6_102_492#
** NODE: 102 = out
C6 104 117 115F
** NODE: 104 = Vdd_tr
** NODE: 0 = GND!
** NODE: 1 = Vdd!

** Bonding wires
RB0 117 0 1
CB0 117 0 25f
```

```
LB0 117 0 20n
RB1 143 1 1
CB1 143 0 25f
LB1 143 1 20n
RB2 120 3 1
CB2 120 0 25f
LB2 120 3 20n

** PCB wirings and 5pF load
RW0 3 4 4
CW0 3 0 4p
CL 4 0 5p

Vdd 1 0 DC 5v
Vvt 104 143 0v
Vgt 112 117 0v
Vt 102 117 DC 5v pulse(0v 5v 1ns 100ps 100ps 50ns 100ns)
.TRAN 100ps 100ns
.END
```

```
** MOS model: 1.2um technology
**
.MODEL nfet NMOS LEVEL=3 PHI=0.6 TOX=2.25E-8 XJ=0.2U TPG=1
+ VTO=0.8186 DELTA=1.757 LD=1.134E-7 KP=9.1547E-5
+ UO=596.5 THETA=0.1085 GAMMA=0.5266 NSUB=1.968E+16
+ NFS=5.5E+12 VMAX=1.942E+5 ETA=6.654E-2 KAPPA=0.1121
+ CGDO=2.6106E-10 CGSO=2.6106E-10 CGBO=6.3402E-10
+ CJ=3.1146E-4 MJ=1.0667 CJSW=4.3777E-10 MJSW=0.15423 PB=0.8
* Weff = Wdrawn - Delta_W
* The suggested Delta_W is 1.997E-7
.MODEL pfet PMOS LEVEL=3 PHI=0.6 TOX=2.25E-8 XJ=0.2U TPG=-1
+ VTO=-0.9456 DELTA=1.552 LD=1.172E-8 KP=3.1646E-5
+ UO=206.2 THETA=1.69E-1 GAMMA=0.4619 NSUB=1.514E+16
+ NFS=4.999E+12 VMAX=4.441E+5 ETA=1.635E-1 KAPPA=10
+ CGDO=2.6981E-11 CGSO=2.6981E-11 CGBO=8.6508E-10
+ CJ=4.7864E-4 MJ=0.4973 CJSW=1.4771E-10 MJSW=0.190593 PB=0.85
* Weff = Wdrawn - Delta_W
* The suggested Delta_W is 3.128E-7

** input pad subcircuit
**
.SUBCKT inpad 1 145 2 100
RLUMP0 100 101 1149.5
RLUMP1 102 103 2333.0
RLUMP2 104 105 7813.5
M0 101 103 105 146 pfet L=1.2U W=22.2U
RLUMP3 104 106 7813.5
RLUMP4 102 107 2333.0
RLUMP5 100 108 1149.5
M1 106 107 108 146 pfet L=1.2U W=22.2U
RLUMP6 100 109 1149.5
RLUMP7 102 110 2333.0
RLUMP8 104 111 7813.5
M2 109 110 111 146 pfet L=1.2U W=22.8U
RLUMP9 104 112 7813.5
RLUMP10 102 113 2333.0
RLUMP11 100 114 1149.5
M3 112 113 114 146 pfet L=1.2U W=22.8U
RLUMP12 102 115 2333.0
RLUMP13 116 117 2433.0
RLUMP14 104 118 7813.5
M4 115 117 118 146 pfet L=1.2U W=15.0U
RLUMP15 104 119 7813.5
RLUMP16 116 120 2433.0
RLUMP17 102 121 2333.0
M5 119 120 121 146 pfet L=1.2U W=15.0U
RLUMP18 100 122 1149.5
RLUMP19 102 123 2333.0
RLUMP20 124 125 3708.0
M6 122 123 125 145 nfet L=1.2U W=7.2U
RLUMP21 124 126 3708.0
RLUMP22 102 127 2333.0
RLUMP23 100 128 1149.5
M7 126 127 128 145 nfet L=1.2U W=7.2U
RLUMP24 100 129 1149.5
RLUMP25 102 130 2333.0
RLUMP26 124 131 3708.0
M8 129 130 131 145 nfet L=1.2U W=7.8U
RLUMP27 124 132 3708.0
RLUMP28 102 133 2333.0
RLUMP29 100 134 1149.5
M9 132 133 134 145 nfet L=1.2U W=7.8U
RLUMP30 102 135 2333.0
RLUMP31 116 136 2433.0
RLUMP32 124 137 3708.0
M10 135 136 137 145 nfet L=1.2U W=9.6U
RLUMP33 124 138 3708.0
RLUMP34 116 139 2433.0
RLUMP35 102 140 2333.0
M11 138 139 140 145 nfet L=1.2U W=9.6U
RLUMP36 102 141 2333.0
RLUMP37 116 142 2433.0
RLUMP38 124 143 3708.0
M12 141 142 143 145 nfet L=1.2U W=9.6U
** NODE: 144 = 10_114_16# ===FLOATING===
C0 145 145 229F
** NODE: 145 = GND_br ===FLOATING===
C1 146 145 238F
** NODE: 146 = Vdd_br ===FLOATING===
C2 124 145 122F
** NODE: 124 = GND_tr
C3 116 145 2336F
** NODE: 116 = pad
C4 100 145 130F
** NODE: 100 = in
C5 102 145 63F
** NODE: 102 = 6_308_974#
C6 104 145 274F
** NODE: 104 = Vdd_tr
** NODE: 0 = GND!
** NODE: 1 = Vdd!

** Bonding wires
RB0 145 0 1
CB0 145 0 25f
LB0 145 0 20n
RB1 146 1 1
CB1 146 0 25f
LB1 146 1 20n
RB2 116 2 1
CB2 116 0 25f
LB2 116 2 20n

Vgt 124 145 0v
Vvt 104 146 0v
.ENDS inpad

** output pad subcircuit
**
.SUBCKT outpad 1 117 102 3
RLUMP0 100 101 3738.5
RLUMP1 102 103 182.5
RLUMP2 104 105 5085.0
M0 101 103 105 143 pfet L=1.2U W=4.8U
RLUMP3 106 107 4966.5
RLUMP4 100 108 3738.5
RLUMP5 104 109 5085.0
M1 107 108 109 143 pfet L=1.2U W=24.0U
RLUMP6 100 110 3738.5
RLUMP7 102 111 182.5
RLUMP8 112 113 2802.0
M2 110 111 113 117 nfet L=1.2U W=2.4U
RLUMP9 106 114 4966.5
RLUMP10 100 115 3738.5
RLUMP11 112 116 2802.0
M3 114 115 116 117 nfet L=1.2U W=9.6U
RLUMP12 117 118 7076.0
RLUMP13 100 119 3738.5
```

```
RLUMP14 120 121 3683.0
M4 118 119 121 117 nfet L=1.2U W=30.0U
RLUMP15 120 122 3683.0
RLUMP16 100 123 3738.5
RLUMP17 117 124 7076.0
M5 122 123 124 117 nfet L=1.2U W=30.0U
RLUMP18 117 125 7076.0
RLUMP19 100 126 3738.5
RLUMP20 120 127 3683.0
M6 125 126 127 117 nfet L=1.2U W=30.0U
RLUMP21 120 128 3683.0
RLUMP22 100 129 3738.5
RLUMP23 117 130 7076.0
M7 128 129 130 117 nfet L=1.2U W=30.0U
RLUMP24 117 131 7076.0
RLUMP25 100 132 3738.5
RLUMP26 120 133 3683.0
M8 131 132 133 117 nfet L=1.2U W=30.0U
RLUMP27 120 134 3683.0
RLUMP28 100 135 3738.5
RLUMP29 117 136 7076.0
M9 134 135 136 117 nfet L=1.2U W=30.0U
RLUMP30 117 137 7076.0
RLUMP31 100 138 3738.5
RLUMP32 120 139 3683.0
M10 137 138 139 117 nfet L=1.2U W=30.0U
RLUMP33 120 140 3683.0
RLUMP34 100 141 3738.5
RLUMP35 117 142 7076.0
M11 140 141 142 117 nfet L=1.2U W=30.0U
RLUMP36 143 144 2827.0
RLUMP37 106 145 4966.5
RLUMP38 120 146 3683.0
M12 144 145 146 117 nfet L=1.2U W=45.0U
RLUMP39 120 147 3683.0
RLUMP40 106 148 4966.5
RLUMP41 143 149 2827.0
M13 147 148 149 117 nfet L=1.2U W=45.0U
RLUMP42 143 150 2827.0
RLUMP43 106 151 4966.5
RLUMP44 120 152 3683.0
M14 150 151 152 117 nfet L=1.2U W=45.0U
RLUMP45 120 153 3683.0
RLUMP46 106 154 4966.5
RLUMP47 143 155 2827.0
M15 153 154 155 117 nfet L=1.2U W=45.0U
RLUMP48 143 156 2827.0
RLUMP49 106 157 4966.5
RLUMP50 120 158 3683.0
M16 156 157 158 117 nfet L=1.2U W=45.0U
RLUMP51 120 159 3683.0
RLUMP52 106 160 4966.5
RLUMP53 143 161 2827.0
M17 159 160 161 117 nfet L=1.2U W=45.0U
RLUMP54 143 162 2827.0
RLUMP55 106 163 4966.5
RLUMP56 120 164 3683.0
M18 162 163 164 117 nfet L=1.2U W=45.0U
RLUMP57 120 165 3683.0
RLUMP58 106 166 4966.5
RLUMP59 143 167 2827.0
M19 165 166 167 117 nfet L=1.2U W=45.0U
** NODE: 168 = 10_114_16# ---FLOATING---
C0 143 117 516F
** NODE: 143 = Vdd_br
C1 120 117 850F
** NODE: 120 = pad
C2 117 117 436F
** NODE: 117 = GND_br
C3 112 117 72F
** NODE: 112 = GND_tr
C4 106 117 93F
** NODE: 106 = 6_264_486#
C5 100 117 62F
** NODE: 100 = 6_102_492#
** NODE: 102 = out
C6 104 117 115F
** NODE: 104 = Vdd_tr
** NODE: 0 = GND!
** NODE: 1 = Vdd!

** Bonding wires
RB0 117 0 1
CB0 117 0 25f
LB0 117 0 20n
RB1 143 1 1
CB1 143 0 25f
LB1 143 1 20n
RB2 120 3 1
CB2 120 0 25f
LB2 120 3 20n

Vvt 104 143 0v
Vgt 112 117 0v
.ENDS outpad

X1 1 10 2 3 outpad
X2 1 20 4 5 inpad

** PCB wirings
RW 3 4 4
CW 3 0 4p

** Load of Vin
CL 5 20 500F

Vdd 1 0 DC 5v
Vt 2 10 DC 5v pulse(0v 5v 1ns 100ps 100ps 40ns 80ns)
.TRAN 100ps 80ns
.END
```

```
** MOS model: 0.5um technology
**
.MODEL nfet NMOS LEVEL=3 PHI=0.6 TOX=1.1E-8 XJ=0.2U TPG=1
+ VTO=0.8186 DELTA=1.757 LD=0 KP=9.1547E-5
+ UO=596.5 THETA=0.1085 GAMMA=0.5266 NSUB=1.968E+16
+ NFS=5.5E+12 VMAX=1.942E+5 ETA=6.654E-2 KAPPA=0.1121
+ CGDO=2.6106E-10 CGSO=2.6106E-10 CGBO=6.3402E-10
+ CJ=3.1146E-4 MJ=1.0667 CJSW=4.3777E-10 MJSW=0.15423 PB=0.8
* Weff = Wdrawn - Delta_W
* The suggested Delta_W is 1.997E-7
.MODEL pfet PMOS LEVEL=3 PHI=0.6 TOX=1.1E-8 XJ=0.2U TPG=-1
+ VTO=-0.9456 DELTA=1.552 LD=0 KP=3.1646E-5
+ UO=206.2 THETA=1.69E-1 GAMMA=0.4619 NSUB=1.514E+16
+ NFS=4.999E+12 VMAX=4.441E+5 ETA=1.635E-1 KAPPA=10
+ CGDO=2.6981E-11 CGSO=2.6981E-11 CGBO=8.6508E-10
+ CJ=4.7864E-4 MJ=0.4973 CJSW=1.4771E-10 MJSW=0.190593 PB=0.85
* Weff = Wdrawn - Delta_W
* The suggested Delta_W is 3.128E-7

** DIODE model
**
.MODEL diode D IS=1E-14 RS=10 N=1 TT=1E-10 CJO=2E-12
+ VJ=0.6 M=0.5 EG=1.11 XTI=3 BV=40 TNOM=50

M0 12 13 11 11 pfet l=0.5u w=37.5u
M1 13 14 11 11 pfet l=0.5u w=12.5u
M2 12 13 10 10 nfet l=0.5u w=12.5u
M3 13 14 10 10 nfet l=0.5u w=12u
R1 14 15 200
D1 14 11 DIODE 308p
D2 10 14 DIODE 308p

** Bonding wires
RB0 10 0 1
CB0 10 0 25f
LB0 10 0 20n
RB1 11 1 1
CB1 11 0 25f
LB1 11 1 20n
RB2 15 16 1
CB2 15 0 25f
LB2 15 16 20n

** Load of Vin
CL 12 10 500f

Vdd 1 0 DC 5v
Vt 16 0 DC 5v pulse(0v 5v 1ns 100ps 100ps 20ns 40ns)
.TRAN 100ps 40ns
.END
```

```
** MOS model: 0.5um technology
**
.MODEL nfet NMOS LEVEL=3 PHI=0.6 TOX=1.1E-8 XJ=0.2U TPG=1
+ VTO=0.8186 DELTA=1.757 LD=0 KP=9.1547E-5
+ UO=596.5 THETA=0.1085 GAMMA=0.5266 NSUB=1.968E+16
+ NFS=5.5E+12 VMAX=1.942E+5 ETA=6.654E-2 KAPPA=0.1121
+ CGDO=2.6106E-10 CGSO=2.6106E-10 CGBO=6.3402E-10
+ CJ=3.1146E-4 MJ=1.0667 CJSW=4.3777E-10 MJSW=0.15423 PB=0.8
* Weff = Wdrawn - Delta_W
* The suggested Delta_W is 1.997E-7
.MODEL pfet PMOS LEVEL=3 PHI=0.6 TOX=1.1E-8 XJ=0.2U TPG=-1
+ VTO=-0.9456 DELTA=1.552 LD=0 KP=3.1646E-5
+ UO=206.2 THETA=1.69E-1 GAMMA=0.4619 NSUB=1.514E+16
+ NFS=4.999E+12 VMAX=4.441E+5 ETA=1.635E-1 KAPPA=10
+ CGDO=2.6981E-11 CGSO=2.6981E-11 CGBO=8.6508E-10
+ CJ=4.7864E-4 MJ=0.4973 CJSW=1.4771E-10 MJSW=0.190593 PB=0.85
* Weff = Wdrawn - Delta_W
* The suggested Delta_W is 3.128E-7

M0 13 12 11 11 pfet l=0.5u w=2u
M1 13 12 10 10 nfet l=0.5u w=1u
M2 14 13 11 11 pfet l=0.5u w=10u
M3 14 13 10 10 nfet l=0.5u w=4u
M4 15 14 11 10 nfet l=0.5u w=150u
M5 15 13 10 10 nfet l=0.5u w=100u
R15 15 16 1
C15 15 10 1p

** Bonding wires
RB0 10 0 1
CB0 10 0 25f
LB0 10 0 20n
RB1 11 1 1
CB1 11 0 25f
LB1 11 1 20n
RB2 16 17 1
CB2 16 0 25f
LB2 16 17 20n

** PCB wiring and 5pF load
RW 17 18 4
CW 17 0 4p
CL 18 0 5p

Vdd 1 0 DC 5v
Vt 12 10 DC 5v pulse(0v 5v 1ns 100ps 100ps 20ns 40ns)
.TRAN 100ps 40ns
.END
```

```
** MOS model: 0.5um technology
**
.MODEL nfet NMOS LEVEL=3 PHI=0.6 TOX=1.1E-8 XJ=0.2U TPG=1
+ VTO=0.8186 DELTA=1.757 LD=0 KP=9.1547E-5
+ UO=596.5 THETA=0.1085 GAMMA=0.5266 NSUB=1.968E+16
+ NFS=5.5E+12 VMAX=1.942E+5 ETA=6.654E-2 KAPPA=0.1121
+ CGDO=2.6106E-10 CGSO=2.6106E-10 CGBO=6.3402E-10
+ CJ=3.1146E-4 MJ=1.0667 CJSW=4.3777E-10 MJSW=0.15423 PB=0.8
* Weff = Wdrawn - Delta_W
* The suggested Delta_W is 1.997E-7
.MODEL pfet PMOS LEVEL=3 PHI=0.6 TOX=1.1E-8 XJ=0.2U TPG=-1
+ VTO=-0.9456 DELTA=1.552 LD=0 KP=3.1646E-5
+ UO=206.2 THETA=1.69E-1 GAMMA=0.4619 NSUB=1.514E+16
+ NFS=4.999E+12 VMAX=4.441E+5 ETA=1.635E-1 KAPPA=10
+ CGDO=2.6981E-11 CGSO=2.6981E-11 CGBO=8.6508E-10
+ CJ=4.7864E-4 MJ=0.4973 CJSW=1.4771E-10 MJSW=0.190593 PB=0.85
* Weff = Wdrawn - Delta_W
* The suggested Delta_W is 3.128E-7

** DIODE model
**
.MODEL diode D IS=1E-14 RS=10 N=1 TT=1E-10 CJO=2E-12
+ VJ=0.6 M=0.5 EG=1.11 XTI=3 BV=40 TNOM=50

** input pad subcircuit
**
.SUBCKT inpad 1 10 16 12
M0 12 13 11 11 pfet l=0.5u w=37.5u
M1 13 14 11 11 pfet l=0.5u w=12.5u
M2 12 13 10 10 nfet l=0.5u w=12.5u
M3 13 14 10 10 nfet l=0.5u w=12u
R1 14 15 200
D1 14 11 DIODE 308p
D2 10 14 DIODE 308p

** Bonding wires
RB0 10 0 1
CB0 10 0 25f
LB0 10 0 20n
RB1 11 1 1
CB1 11 0 25f
LB1 11 1 20n
RB2 15 16 1
CB2 15 0 25f
LB2 15 16 20n
.ENDS inpad

** output pad subcircuit
**
.SUBCKT outpad 1 10 12 17
M0 13 12 11 11 pfet l=0.5u w=2u
M1 13 12 10 10 nfet l=0.5u w=1u
M2 14 13 11 11 pfet l=0.5u w=10u
M3 14 13 10 10 nfet l=0.5u w=4u
M4 15 14 11 10 nfet l=0.5u w=150u
M5 15 13 10 10 nfet l=0.5u w=100u
R15 15 16 1
C15 15 10 1p

** Bonding wires
RB0 10 0 1
CB0 10 0 25f
LB0 10 0 20n
RB1 11 1 1
CB1 11 0 25f
LB1 11 1 20n
RB2 16 17 1
CB2 16 0 25f
LB2 16 17 20n
.ENDS outpad

X1 1 10 2 3 outpad
X2 1 20 4 5 inpad

** PCB wiring
RW 3 4 4
CW 3 0 4p

** Load of Vin
CL 5 20 500f

Vdd 1 0 DC 5v
Vt 2 10 DC 5v pulse(0v 5v 1ns 100ps 100ps 20ns 40ns)
.TRAN 100ps 40ns
.END
```

```
** MOS model: 0.5um technology
**
.MODEL nfet NMOS LEVEL=3 PHI=0.6 TOX=1.1E-8 XJ=0.2U TPG=1
+ VTO=0.8186 DELTA=1.757 LD=0 KP=9.1547E-5
+ UO=596.5 THETA=0.1085 GAMMA=0.5266 NSUB=1.968E+16
+ NFS=5.5E+12 VMAX=1.942E+5 ETA=6.654E-2 KAPPA=0.1121
+ CGDO=2.6106E-10 CGSO=2.6106E-10 CGBO=6.3402E-10
+ CJ=3.1146E-4 MJ=1.0667 CJSW=4.3777E-10 MJSW=0.15423 PB=0.8
* Weff = Wdrawn - Delta_W
* The suggested Delta_W is 1.997E-7
.MODEL pfet PMOS LEVEL=3 PHI=0.6 TOX=1.1E-8 XJ=0.2U TPG=-1
+ VTO=-0.9456 DELTA=1.552 LD=0 KP=3.1646E-5
+ UO=206.2 THETA=1.69E-1 GAMMA=0.4619 NSUB=1.514E+16
+ NFS=4.999E+12 VMAX=4.441E+5 ETA=1.635E-1 KAPPA=10
+ CGDO=2.6981E-11 CGSO=2.6981E-11 CGBO=8.6508E-10
+ CJ=4.7864E-4 MJ=0.4973 CJSW=1.4771E-10 MJSW=0.190593 PB=0.85
* Weff = Wdrawn - Delta_W
* The suggested Delta_W is 3.128E-7


** SPICE file created for circuit pad6i
** Technology: scmos
**

RLUMP0 100 101 1149.5
RLUMP1 102 103 2333.0
RLUMP2 104 105 7813.5
M0 101 103 105 146 pfet L=0.5U W=9.25U
RLUMP3 104 106 7813.5
RLUMP4 102 107 2333.0
RLUMP5 100 108 1149.5
M1 106 107 108 146 pfet L=0.5U W=9.25U
RLUMP6 100 109 1149.5
RLUMP7 102 110 2333.0
RLUMP8 104 111 7813.5
M2 109 110 111 146 pfet L=0.5U W=9.25U
RLUMP9 104 112 7813.5
RLUMP10 102 113 2333.0
RLUMP11 100 114 1149.5
M3 112 113 114 146 pfet L=0.5U W=9.25U
RLUMP12 102 115 2333.0
RLUMP13 116 117 2433.0
RLUMP14 104 118 7813.5
M4 115 117 118 146 pfet L=0.5U W=6.25U
RLUMP15 104 119 7813.5
RLUMP16 116 120 2433.0
RLUMP17 102 121 2333.0
M5 119 120 121 146 pfet L=0.5U W=6.25U
RLUMP18 100 122 1149.5
RLUMP19 102 123 2333.0
RLUMP20 124 125 3708.0
M6 122 123 125 145 nfet L=0.5U W=3U
RLUMP21 124 126 3708.0
RLUMP22 102 127 2333.0
RLUMP23 100 128 1149.5
M7 126 127 128 145 nfet L=0.5U W=3U
RLUMP24 100 129 1149.5
RLUMP25 102 130 2333.0
RLUMP26 124 131 3708.0
M8 129 130 131 145 nfet L=0.5U W=3.25U
RLUMP27 124 132 3708.0
RLUMP28 102 133 2333.0
RLUMP29 100 134 1149.5
M9 132 133 134 145 nfet L=0.5U W=3.25U
RLUMP30 102 135 2333.0
RLUMP31 116 136 2433.0
RLUMP32 124 137 3708.0
M10 135 136 137 145 nfet L=0.5U W=4U
RLUMP33 124 138 3708.0
RLUMP34 116 139 2433.0
RLUMP35 102 140 2333.0
M11 138 139 140 145 nfet L=0.5U W=4U
RLUMP36 102 141 2333.0
RLUMP37 116 142 2433.0
RLUMP38 124 143 3708.0
M12 141 142 143 145 nfet L=0.5U W=4U
** NODE: 144 = 10_114_16# ===FLOATING===
C0 145 0 39.76F
** NODE: 145 = GND_br ===FLOATING===
C1 146 0 41.32F
** NODE: 146 = Vdd_br ===FLOATING===
C2 124 0 21.18F
** NODE: 124 = GND_tr
C3 116 0 405.56F
** NODE: 116 = pad
C4 100 0 22.57F
** NODE: 100 = in
C5 102 0 14.41F
** NODE: 102 = 6_308_974#
C6 104 0 47.57F
** NODE: 104 = Vdd_tr
** NODE: 0 = GND!
** NODE: 1 = Vdd!

** Bonding wires
RB0 145 0 1
CB0 145 0 25f
LB0 145 0 20n
RB1 146 1 1
CB1 146 0 25f
LB1 146 1 20n
RB2 116 2 1
CB2 116 0 25f
LB2 116 2 20n

** Load of Vin
C100 100 0 500F

Vdd 1 0 DC 5v
Vgt 124 145 0v
Vvt 104 146 0v
Vt 2 0 DC 5v pulse(0v 5v 1ns 100ps 100ps 20ns 40ns)
.TRAN 100ps 40ns
.END
```

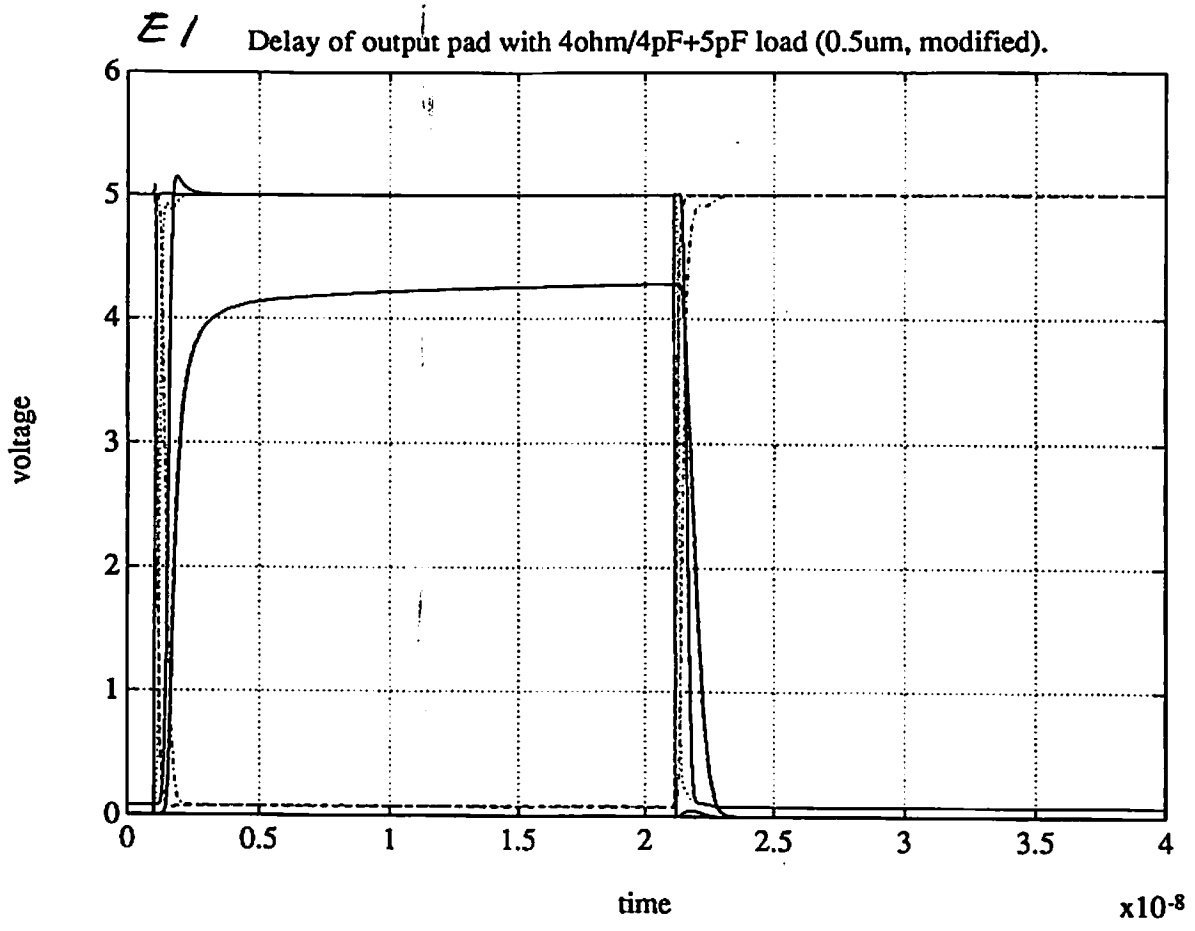
```
** MOS model: 0.5um technology
**
.MODEL nfet NMOS LEVEL=3 PHI=0.6 TOX=1.1E-8 XJ=0.2U TPG=1
+ VTO=0.8186 DELTA=1.757 LD=0 KP=9.1547E-5
+ UO=596.5 THETA=0.1085 GAMMA=0.5266 NSUB=1.968E+16
+ NFS=5.5E+12 VMAX=1.942E+5 ETA=6.654E-2 KAPPA=0.1121
+ CGDO=2.6106E-10 CGSO=2.6106E-10 CGBO=6.3402E-10
+ CJ=3.1146E-4 MJ=1.0667 CJSW=4.3777E-10 MJSW=0.15423 PB=0.8
* Weff = Wdrawn - Delta_W
* The suggested Delta_W is 1.997E-7
.MODEL pfet PMOS LEVEL=3 PHI=0.6 TOX=1.1E-8 XJ=0.2U TPG=-1
+ VTO=-0.9456 DELTA=1.552 LD=0 KP=3.1646E-5
+ UO=206.2 THETA=1.69E-1 GAMMA=0.4619 NSUB=1.514E+16
+ NFS=4.999E+12 VMAX=4.441E+5 ETA=1.635E-1 KAPPA=10
+ CGDO=2.6981E-11 CGSO=2.6981E-11 CGBO=8.6508E-10
+ CJ=4.7864E-4 MJ=0.4973 CJSW=1.4771E-10 MJSW=0.190593 PB=0.85
* Weff = Wdrawn - Delta_W
* The suggested Delta_W is 3.128E-7

** SPICE file created for circuit pad6o
** Technology: scmos
**

RLUMP0 100 101 3738.5
RLUMP1 102 103 182.5
RLUMP2 104 105 5085.0
M0 101 103 105 11 pfet L=0.5U W=2U
RLUMP3 106 107 4966.5
RLUMP4 100 108 3738.5
RLUMP5 104 109 5085.0
M1 107 108 109 11 pfet L=0.5U W=10U
RLUMP6 100 110 3738.5
RLUMP7 102 111 182.5
RLUMP8 112 113 2802.0
M2 110 111 113 10 nfet L=0.5U W=1U
RLUMP9 106 114 4966.5
RLUMP10 100 115 3738.5
RLUMP11 112 116 2802.0
M3 114 115 116 10 nfet L=0.5U W=4U
RLUMP12 117 118 7076.0
RLUMP13 100 119 3738.5
RLUMP14 120 121 3683.0
M4 118 119 121 10 nfet L=0.5U W=12.5U
RLUMP15 120 122 3683.0
RLUMP16 100 123 3738.5
RLUMP17 117 124 7076.0
M5 122 123 124 10 nfet L=0.5U W=12.5U
RLUMP18 117 125 7076.0
RLUMP19 100 126 3738.5
RLUMP20 120 127 3683.0
M6 125 126 127 10 nfet L=0.5U W=12.5U
RLUMP21 120 128 3683.0
RLUMP22 100 129 3738.5
RLUMP23 117 130 7076.0
M7 128 129 130 10 nfet L=0.5U W=12.5U
RLUMP24 117 131 7076.0
RLUMP25 100 132 3738.5
RLUMP26 120 133 3683.0
M8 131 132 133 10 nfet L=0.5U W=12.5U
RLUMP27 120 134 3683.0
RLUMP28 100 135 3738.5
RLUMP29 117 136 7076.0
M9 134 135 136 10 nfet L=0.5U W=12.5U
RLUMP30 117 137 7076.0
RLUMP31 100 138 3738.5
RLUMP32 120 139 3683.0
M10 137 138 139 10 nfet L=0.5U W=12.5U
RLUMP33 120 140 3683.0
RLUMP34 100 141 3738.5
RLUMP35 117 142 7076.0
M11 140 141 142 10 nfet L=0.5U W=12.5U
RLUMP36 143 144 2827.0
RLUMP37 106 145 4966.5
RLUMP38 120 146 3683.0
M12 144 145 146 10 nfet L=0.5U W=18.75U
RLUMP39 120 147 3683.0
RLUMP40 106 148 4966.5
RLUMP41 143 149 2827.0
M13 147 148 149 10 nfet L=0.5U W=18.75U
RLUMP42 143 150 2827.0
RLUMP43 106 151 4966.5
RLUMP44 120 152 3683.0
M14 150 151 152 10 nfet L=0.5U W=18.75U
RLUMP45 120 153 3683.0
RLUMP46 106 154 4966.5
RLUMP47 143 155 2827.0
M15 153 154 155 10 nfet L=0.5U W=18.75U
RLUMP48 143 156 2827.0
RLUMP49 106 157 4966.5
RLUMP50 120 158 3683.0
M16 156 157 158 10 nfet L=0.5U W=18.75U
RLUMP51 120 159 3683.0
RLUMP52 106 160 4966.5
RLUMP53 143 161 2827.0
M17 159 160 161 10 nfet L=0.5U W=18.75U
RLUMP54 143 162 2827.0
RLUMP55 106 163 4966.5
RLUMP56 120 164 3683.0
M18 162 163 164 10 nfet L=0.5U W=18.75U
RLUMP57 120 165 3683.0
RLUMP58 106 166 4966.5
RLUMP59 143 167 2827.0
M19 165 166 167 10 nfet L=0.5U W=18.75U
** NODE: 168 = 10_114_16# ===FLOATING===
C0 143 10 89.58F
** NODE: 143 = Vdd_br
C1 120 10 147.57F
** NODE: 120 = pad
C2 117 10 75.69F
** NODE: 117 = GND_br
C3 112 10 12.50F
** NODE: 112 = GND_tr
C4 106 10 16.15F
** NODE: 106 = 6_264_486#
C5 100 10 10.76F
** NODE: 100 = 6_102_492#
** NODE: 102 = out
C6 104 10 19.97F
** NODE: 104 = Vdd_tr
** NODE: 0 = GND!
** NODE: 1 = Vdd!

** Bonding wires
RB0 117 0 1
CB0 117 0 25f
LB0 117 0 20n
RB1 143 1 1
CB1 143 0 25f
```

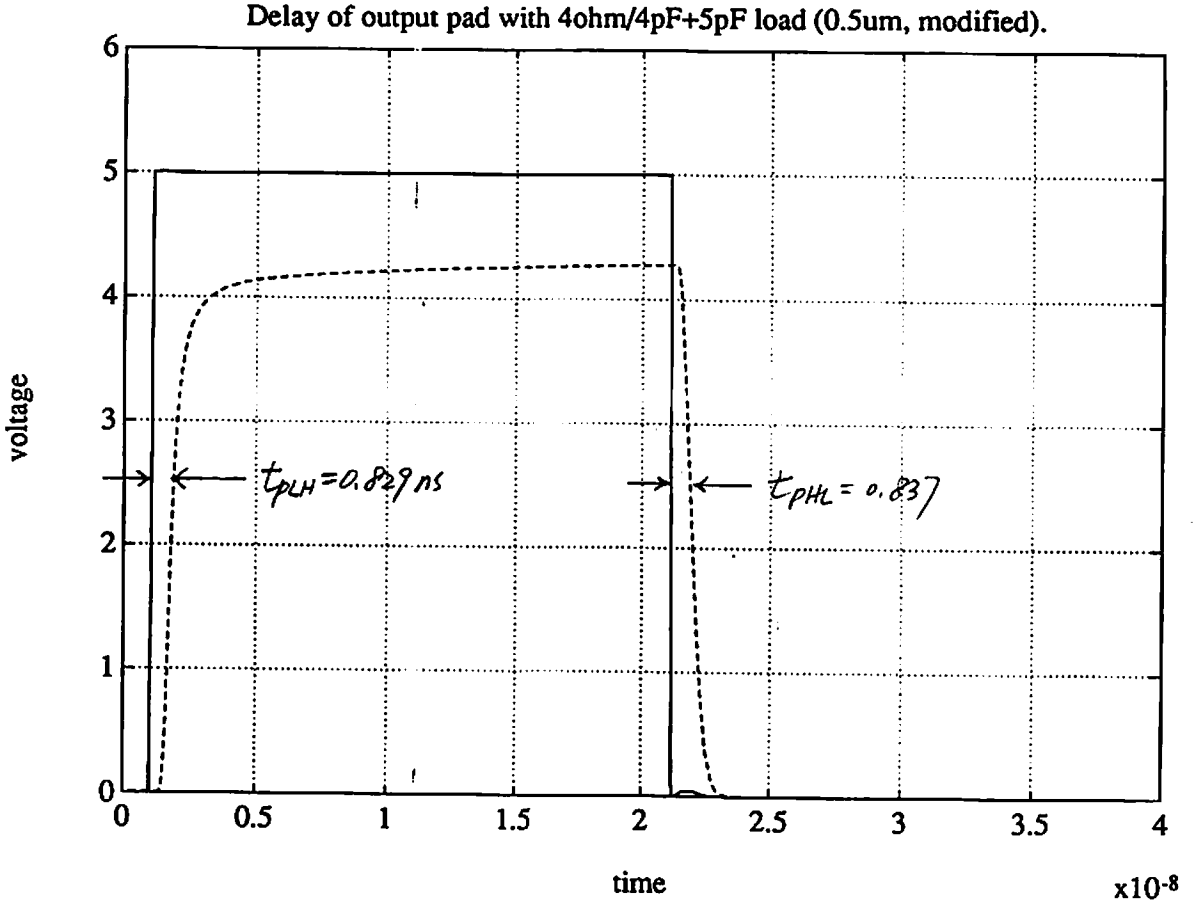
```
LB1 143 1 20n
RB2 120 3 1
CB2 120 0 25f
LB2 120 3 20n

** PCB wirings and 5pF load
RW0 3 4 4
CW0 3 0 4p
CL 4 0 5p

Vdd 1 0 DC 5v
Vvt 104 143 0v
Vgt 112 117 0v
Vt 102 117 DC 5v pulse(0v 5v 1ns 100ps 100ps 50ns 100ns)
.TRAN 100ps 100ns
.END
```

```
** MOS model: 0.5um technology
**
.MODEL nfet NMOS LEVEL=3 PHI=0.6 TOX=1.1E-8 XJ=0.2U TPG=1
+ VTO=0.8186 DELTA=1.757 LD=0 KP=9.1547E-5
+ UO=596.5 THETA=0.1085 GAMMA=0.5266 NSUB=1.968E+16
+ NFS=5.5E+12 VMAX=1.942E+5 ETA=6.654E-2 KAPPA=0.1121
+ CGDO=2.6106E-10 CGSO=2.6106E-10 CGBO=6.3402E-10
+ CJ=3.1146E-4 MJ=1.0667 CJSW=4.3777E-10 MJSW=0.15423 PB=0.8
* Weff = Wdrawn - Delta_W
* The suggested Delta_W is 1.997E-7
.MODEL pfet PMOS LEVEL=3 PHI=0.6 TOX=1.1E-8 XJ=0.2U TPG=-1
+ VTO=-0.9456 DELTA=1.552 LD=0 KP=3.1646E-5
+ UO=206.2 THETA=1.69E-1 GAMMA=0.4619 NSUB=1.514E+16
+ NFS=4.999E+12 VMAX=4.441E+5 ETA=1.635E-1 KAPPA=10
+ CGDO=2.6981E-11 CGSO=2.6981E-11 CGBO=8.6508E-10
+ CJ=4.7864E-4 MJ=0.4973 CJSW=1.4771E-10 MJSW=0.190593 PB=0.85
* Weff = Wdrawn - Delta_W
* The suggested Delta_W is 3.128E-7

** input pad subcircuit
**
.SUBCKT inpad 1 145 2 100
RLUMP0 100 101 1149.5
RLUMP1 102 103 2333.0
RLUMP2 104 105 7813.5
M0 101 103 105 146 pfet L=0.5U W=9.25U
RLUMP3 104 106 7813.5
RLUMP4 102 107 2333.0
RLUMP5 100 108 1149.5
M1 106 107 108 146 pfet L=0.5U W=9.25U
RLUMP6 100 109 1149.5
RLUMP7 102 110 2333.0
RLUMP8 104 111 7813.5
M2 109 110 111 146 pfet L=0.5U W=9.25U
RLUMP9 104 112 7813.5
RLUMP10 102 113 2333.0
RLUMP11 100 114 1149.5
M3 112 113 114 146 pfet L=0.5U W=9.25U
RLUMP12 102 115 2333.0
RLUMP13 116 117 2433.0
RLUMP14 104 118 7813.5
M4 115 117 118 146 pfet L=0.5U W=6.25U
RLUMP15 104 119 7813.5
RLUMP16 116 120 2433.0
RLUMP17 102 121 2333.0
M5 119 120 121 146 pfet L=0.5U W=6.25U
RLUMP18 100 122 1149.5
RLUMP19 102 123 2333.0
RLUMP20 124 125 3708.0
M6 122 123 125 145 nfet L=0.5U W=3U
RLUMP21 124 126 3708.0
RLUMP22 102 127 2333.0
RLUMP23 100 128 1149.5
M7 126 127 128 145 nfet L=0.5U W=3U
RLUMP24 100 129 1149.5
RLUMP25 102 130 2333.0
RLUMP26 124 131 3708.0
M8 129 130 131 145 nfet L=0.5U W=3.25U
RLUMP27 124 132 3708.0
RLUMP28 102 133 2333.0
RLUMP29 100 134 1149.5
M9 132 133 134 145 nfet L=0.5U W=3.25U
RLUMP30 102 135 2333.0
RLUMP31 116 136 2433.0
RLUMP32 124 137 3708.0
M10 135 136 137 145 nfet L=0.5U W=4U
RLUMP33 124 138 3708.0
RLUMP34 116 139 2433.0
RLUMP35 102 140 2333.0
M11 138 139 140 145 nfet L=0.5U W=4U
RLUMP36 102 141 2333.0
RLUMP37 116 142 2433.0
RLUMP38 124 143 3708.0
M12 141 142 143 145 nfet L=0.5U W=4U
** NODE: 144 = 10_114_16# ===FLOATING===
C0 145 0 39.76F
** NODE: 145 = GND_br ===FLOATING===
C1 146 0 41.32F
** NODE: 146 = Vdd_br ===FLOATING===
C2 124 0 21.18F
** NODE: 124 = GND_tr
C3 116 0 405.56F
** NODE: 116 = pad
C4 100 0 22.57F
** NODE: 100 = in
C5 102 0 14.41F
** NODE: 102 = 6_308_974#
C6 104 0 47.57F
** NODE: 104 = Vdd_tr
** NODE: 0 = GND!
** NODE: 1 = Vdd!

** Bonding wires
RB0 145 0 1
CB0 145 0 25f
LB0 145 0 20n
RB1 146 1 1
CB1 146 0 25f
LB1 146 1 20n
RB2 116 2 1
CB2 116 0 25f
LB2 116 2 20n

Vgt 124 145 0v
Vvt 104 146 0v
.ENDS inpad

** output pad subcircuit
**
.SUBCKT outpad 1 117 102 3
RLUMP0 100 101 3738.5
RLUMP1 102 103 182.5
RLUMP2 104 105 5085.0
M0 101 103 105 11 pfet L=0.5U W=2U
RLUMP3 106 107 4966.5
RLUMP4 100 108 3738.5
RLUMP5 104 109 5085.0
M1 107 108 109 11 pfet L=0.5U W=10U
RLUMP6 100 110 3738.5
RLUMP7 102 111 182.5
RLUMP8 112 113 2802.0
M2 110 111 113 10 nfet L=0.5U W=1U
RLUMP9 106 114 4966.5
RLUMP10 100 115 3738.5
RLUMP11 112 116 2802.0
M3 114 115 116 10 nfet L=0.5U W=4U
RLUMP12 117 118 7076.0
RLUMP13 100 119 3738.5
```

```
RLUMP14 120 121 3683.0
M4 118 119 121 10 nfet L=0.5U W=12.5U
RLUMP15 120 122 3683.0
RLUMP16 100 123 3738.5
RLUMP17 117 124 7076.0
M5 122 123 124 10 nfet L=0.5U W=12.5U
RLUMP18 117 125 7076.0
RLUMP19 100 126 3738.5
RLUMP20 120 127 3683.0
M6 125 126 127 10 nfet L=0.5U W=12.5U
RLUMP21 120 128 3683.0
RLUMP22 100 129 3738.5
RLUMP23 117 130 7076.0
M7 128 129 130 10 nfet L=0.5U W=12.5U
RLUMP24 117 131 7076.0
RLUMP25 100 132 3738.5
RLUMP26 120 133 3683.0
M8 131 132 133 10 nfet L=0.5U W=12.5U
RLUMP27 120 134 3683.0
RLUMP28 100 135 3738.5
RLUMP29 117 136 7076.0
M9 134 135 136 10 nfet L=0.5U W=12.5U
RLUMP30 117 137 7076.0
RLUMP31 100 138 3738.5
RLUMP32 120 139 3683.0
M10 137 138 139 10 nfet L=0.5U W=12.5U
RLUMP33 120 140 3683.0
RLUMP34 100 141 3738.5
RLUMP35 117 142 7076.0
M11 140 141 142 10 nfet L=0.5U W=12.5U
RLUMP36 143 144 2827.0
RLUMP37 106 145 4966.5
RLUMP38 120 146 3683.0
M12 144 145 146 10 nfet L=0.5U W=18.75U
RLUMP39 120 147 3683.0
RLUMP40 106 148 4966.5
RLUMP41 143 149 2827.0
M13 147 148 149 10 nfet L=0.5U W=18.75U
RLUMP42 143 150 2827.0
RLUMP43 106 151 4966.5
RLUMP44 120 152 3683.0
M14 150 151 152 10 nfet L=0.5U W=18.75U
RLUMP45 120 153 3683.0
RLUMP46 106 154 4966.5
RLUMP47 143 155 2827.0
M15 153 154 155 10 nfet L=0.5U W=18.75U
RLUMP48 143 156 2827.0
RLUMP49 106 157 4966.5
RLUMP50 120 158 3683.0
M16 156 157 158 10 nfet L=0.5U W=18.75U
RLUMP51 120 159 3683.0
RLUMP52 106 160 4966.5
RLUMP53 143 161 2827.0
M17 159 160 161 10 nfet L=0.5U W=18.75U
RLUMP54 143 162 2827.0
RLUMP55 106 163 4966.5
RLUMP56 120 164 3683.0
M18 162 163 164 10 nfet L=0.5U W=18.75U
RLUMP57 120 165 3683.0
RLUMP58 106 166 4966.5
RLUMP59 143 167 2827.0
M19 165 166 167 10 nfet L=0.5U W=18.75U
** NODE: 168 = 10_114_16# ===FLOATING===
C0 143 10 89.58F
** NODE: 143 = Vdd_br
C1 120 10 147.57F
** NODE: 120 = pad
C2 117 10 75.69F
** NODE: 117 = GND_br
C3 112 10 12.50F
** NODE: 112 = GND_tr
C4 106 10 16.15F
** NODE: 106 = 6_264_486#
C5 100 10 10.76F
** NODE: 100 = 6_102_492#
** NODE: 102 = out
C6 104 10 19.97F
** NODE: 104 = Vdd_tr
** NODE: 0 = GND!
** NODE: 1 = Vdd!

** Bonding wires
RB0 117 0 1
CB0 117 0 25f
LB0 117 0 20n
RB1 143 1 1
CB1 143 0 25f
LB1 143 1 20n
RB2 120 3 1
CB2 120 0 25f
LB2 120 3 20n

Vvt 104 143 0v
Vgt 112 117 0v
.ENDS outpad

X1 1 10 2 3 outpad
X2 1 20 4 5 inpad

** PCB wirings
RW0 3 4 4
CW0 3 0 4p

** Load of Vin
CL 5 20 500f

Vdd 1 0 DC 5v
Vt 2 10 DC 5v pulse(0v 5v 1ns 100ps 100ps 20ns 40ns)
.TRAN 100ps 40ns
.END
```

E1


Delay of output pad with 4ohm/4pF+5pF load (0.5um, modified).
voltage
time
x10-8
0
1
2
3
4
5
6
0.5
1.5
2.5
3.5


Delay of output pad with 4ohm/4pF+5pF load (0.5um, modified).
voltage
time
x10-8
tpLH=0.829 ns
tpHL = 0.837
0
1
2
3
4
5
6
0.5
1.5
2.5
3.5

```
** MOS model: 0.5um technology
**
.MODEL nfet NMOS LEVEL=3 PHI=0.6 TOX=1.1E-8 XJ=0.2U TPG=1
+ VTO=0.8186 DELTA=1.757 LD=0 KP=9.1547E-5
+ UO=596.5 THETA=0.1085 GAMMA=0.5266 NSUB=1.968E+16
+ NFS=5.5E+12 VMAX=1.942E+5 ETA=6.654E-2 KAPPA=0.1121
+ CGDO=2.6106E-10 CGSO=2.6106E-10 CGBO=6.3402E-10
+ CJ=3.1146E-4 MJ=1.0667 CJSW=4.3777E-10 MJSW=0.15423 PB=0.8
* Weff = Wdrawn - Delta_W
* The suggested Delta_W is 1.997E-7
.MODEL pfet PMOS LEVEL=3 PHI=0.6 TOX=1.1E-8 XJ=0.2U TPG=-1
+ VTO=-0.9456 DELTA=1.552 LD=0 KP=3.1646E-5
+ UO=206.2 THETA=1.69E-1 GAMMA=0.4619 NSUB=1.514E+16
+ NFS=4.999E+12 VMAX=4.441E+5 ETA=1.635E-1 KAPPA=10
+ CGDO=2.6981E-11 CGSO=2.6981E-11 CGBO=8.6508E-10
+ CJ=4.7864E-4 MJ=0.4973 CJSW=1.4771E-10 MJSW=0.190593 PB=0.85
* Weff = Wdrawn - Delta_W
* The suggested Delta_W is 3.128E-7

M0 13 12 11 11 pfet l=0.5u w=2u
M1 13 12 10 10 nfet l=0.5u w=1u
M2 14 13 11 11 pfet l=0.5u w=6u
M3 14 13 10 10 nfet l=0.5u w=3u
M4 19 14 11 11 pfet l=0.5u w=18u
M5 19 14 10 10 nfet l=0.5u w=9u
M6 20 19 11 11 pfet l=0.5u w=54u
M7 20 19 10 10 nfet l=0.5u w=27u
M8 15 20 11 10 nfet l=0.5u w=300u
M9 15 19 10 10 nfet l=0.5u w=100u
R15 15 16 1
C15 15 10 174f

** Bonding wires
RB0 10 0 1
CB0 10 0 25f
LB0 10 0 20n
RB1 11 1 1
CB1 11 0 25f
LB1 11 1 20n
RB2 16 17 1
CB2 16 0 25f
LB2 16 17 20n

** PCB wiring and 5pF load
RW 17 18 4
CW 17 0 4p
CL 18 0 5p

Vdd 1 0 DC 5v
Vt 12 10 DC 5v pulse(0v 5v 1ns 100ps 100ps 20ns 40ns)
.TRAN 100ps 40ns
.END
```

200 MHz Queue

888-03-4696 Ta-Chuan Hsu

1. Introduction

This report is about a queue which is a critical component in multi-multi and should be able to work under 200 MHz. The speed is extremely high, so there is no commercial product available. In the report, I will illustrate how the queue was implemented by TRW 0.5 um technology to achieve the fantastic high speed.

2. Logic function

The block diagram of the queue is shown in fig. A1. We keep two pointers, one for read and one for write, pointing to the memory words. Once the outside world wants to read, it must check the EMPTY flag first to see if there are items in the queue. After read, the read pointer increments by one, thus pointing to the next word. Similarly, before write, we check the FULL flag to see if there is still free space in the queue to be used for the coming data. If so, we write data into queue and increments write pointer by one and also pointing to the next position. Fig. A2 shows a clock period and the jobs to be done in one period. We will follow the sequence of these jobs to illustrate the implementation of each block in the queue.

3. Control Logic

The control logic was implemented by using MPLA program. The layout is in fig. B1. Since this is a PLA, the output delay for each signal shold be about the same. So I only show one signal delay diagram in fig. C1. The delay time is 617 ps.

4. Read and Write Pointers

These two pointers was constructed by a series of cascaded D flip-flops (logic diagram in fig. A4 layouts in fig.B2-3 and simulation results in C2-6). The numerical data is in table 1-2. From the tables, we can see that the delay for passing through the transmission gates controlled by output enable signal is no longer than 49 ps and the propogation dalay for a D flip-flop is 198 ps at worst.

5. Memory

The memory size is 16 by 32. The logic diagram for one bitline is shown in fig. A3 and the layout is in fig. B4. The simulation results are shown in fig. C7-10. Also, from table 3, we know that the worst case delay for a memory operation (either read or write) is 677 ps.

6. Comparator

The comparator is also a PLA. therefore the delay shold be the same as that of control logic, which is 617 ps. Its layout is in fig. B6.

7. Flag Logic

The logic diagram is shown in fig. A5 and the layout in fig. B7. The delay is one D flip-flop plus one nor gate, which should be less than 300 ps (198 ps + 100ps.)

8. Conclusion

The overall delay is 617 (control) + 49 (output enable delay of shift registers) + 677 (memory)

+198 (one shift operation) + 617 (comparison) + 300 (flag generation) ps. which is 2458 ps. This delay is about a half period of 200MHz. Thus, we can fulfill the requirement elegantly.

Table 1: Edge-triggered D flip-flop (in picosecs)

t_{SETUP}	t_{HOLD}	t_{PLH}	t_{PHL}	t_{SET}	t_{RESET}
300	0	83	198	131	267

* All delays are evaluated from 50% to 50%.

Table 2: D flip-flop with output enable, transmission gate delay (in picosec)

t_P	t_P
49	46
LOGIC 1	LOGIC 0

* The delay for logic 1 is evaluated from 50% output enable signal (OE) to 50% Q signal (from 2.5V to 3.75V.)
* The delay for logic 0 is evaluated from 50% output enable signal (OE) to 50% Q signal (from 2.5V to 1.25V.)

Table 3: Memory (in picosecs)

t_{SETUP}	t_{HOLD}	t_{PLH}	t_{PHL}	t_P	t_P
0	700	470	151	597	677
WRITE	WRITE	WRITE	WRITE	READ 0	READ 1

* All delays for WRITE operations are evaluated from 50% word select signal to 50% cell signal.
* The delay for READ 0 operation is evaluated from 50% sense enable signal (SAE) to 50% dataout signal (dataout from 2.83V to 1.415V.)
* The delay for READ 1 operation is evaluated from 50% sense enable signal (SAE) to 50% dataout signal (dataout from 2.83V to 3.91V.)

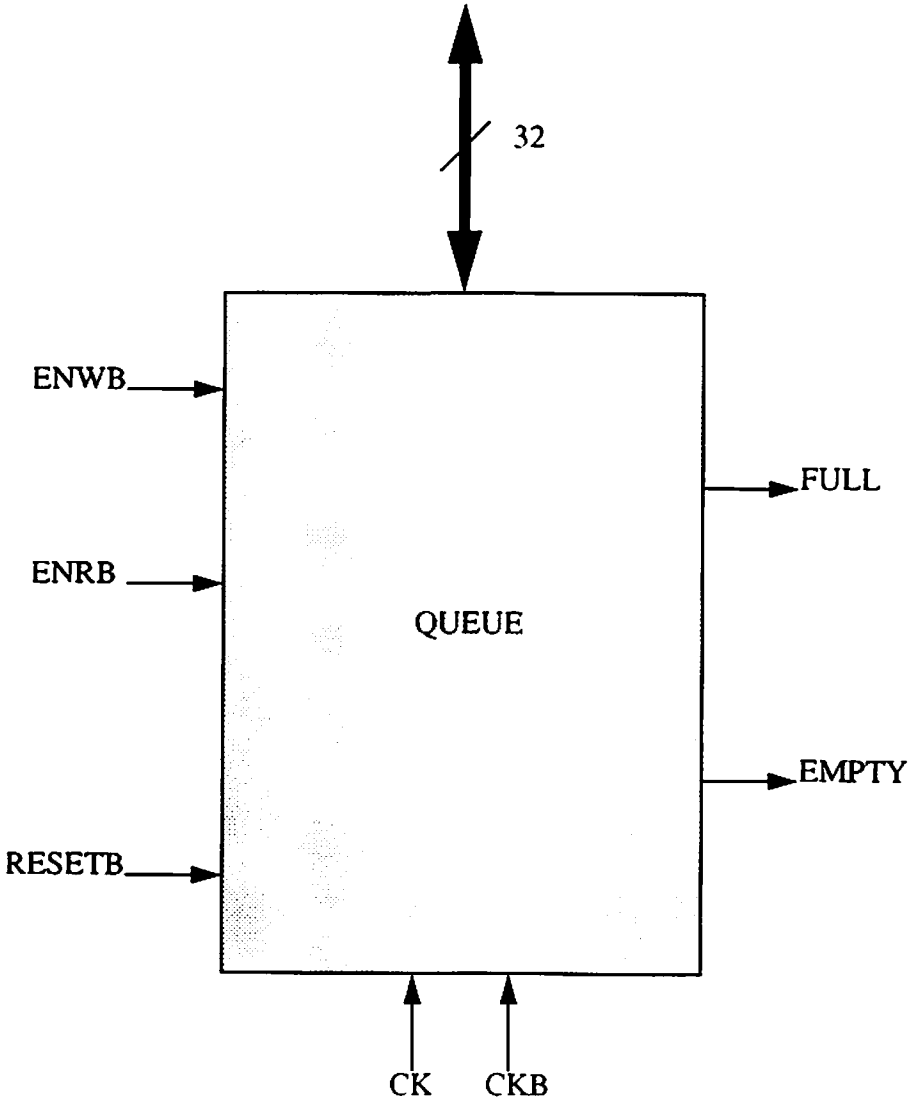


Fig. A1. Block diagram of Queue (to be continued)

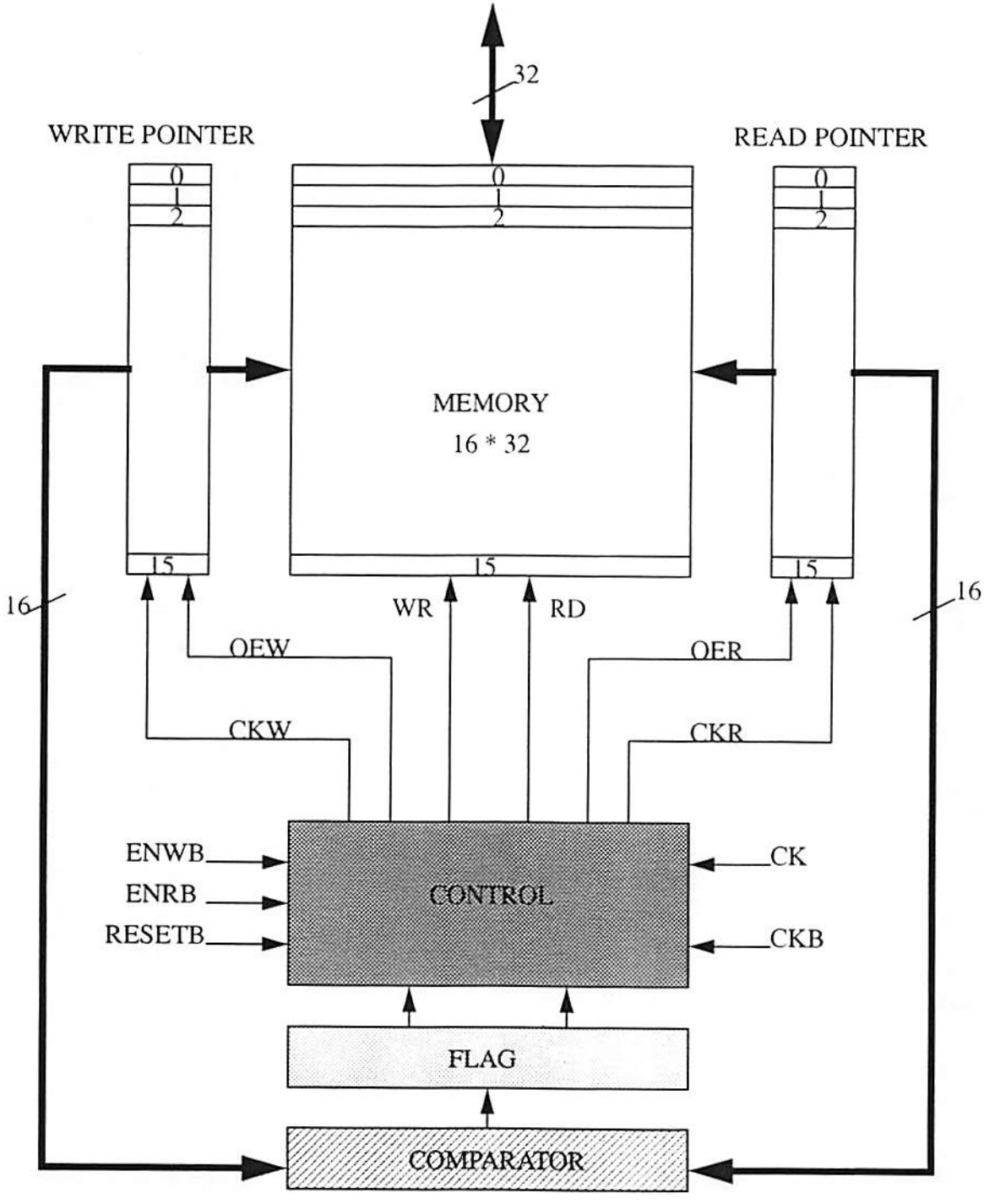


Fig. A1. Block Diagram of Queue (continued)

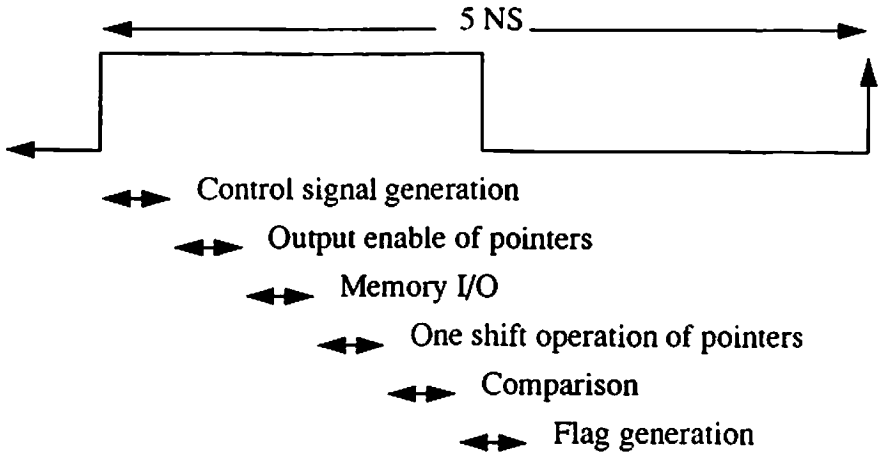


Fig. A2. The Jobs to be Done in One Clock Period

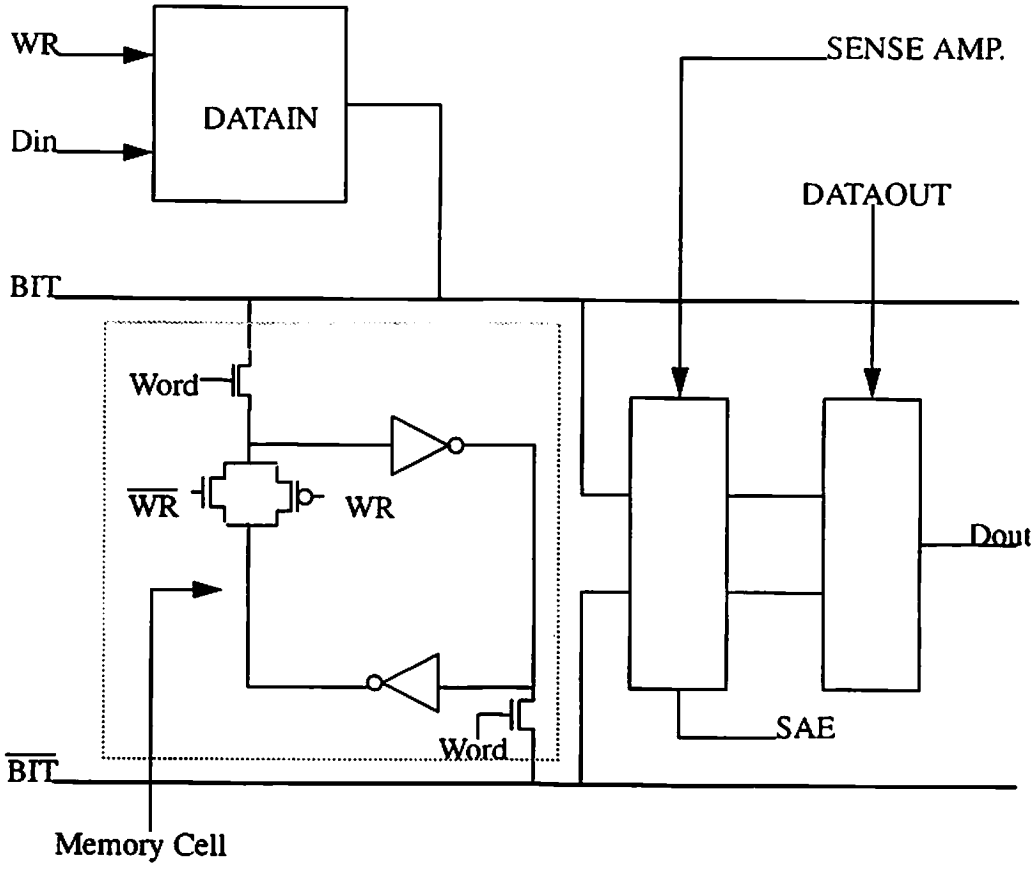
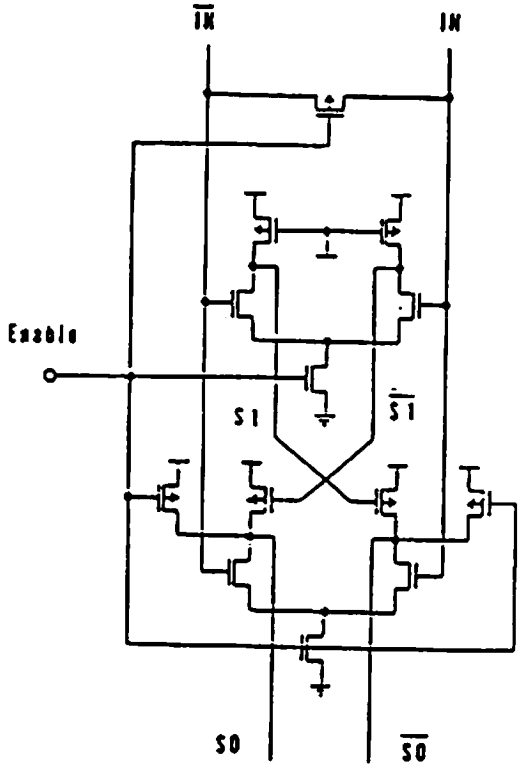
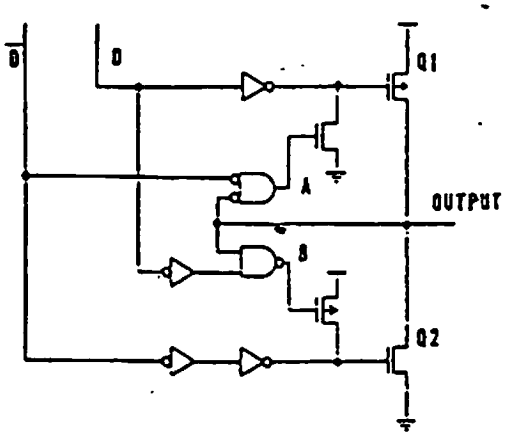


Figure A3 Bitline (to be continued)



Sense-amplifier Circuitry



Data-output Circuitry

Fig. A3 Bitline (continued)

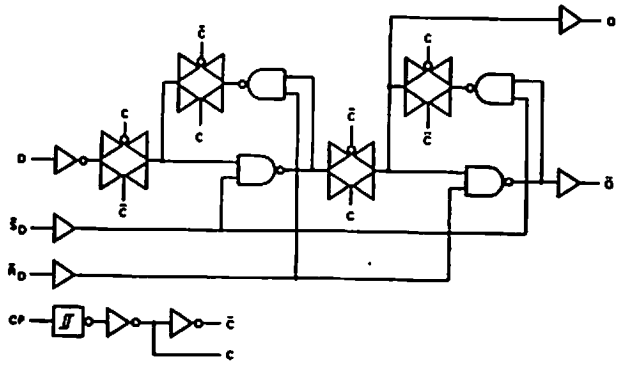


Fig. A4 Logic diagram of a D flip-flop

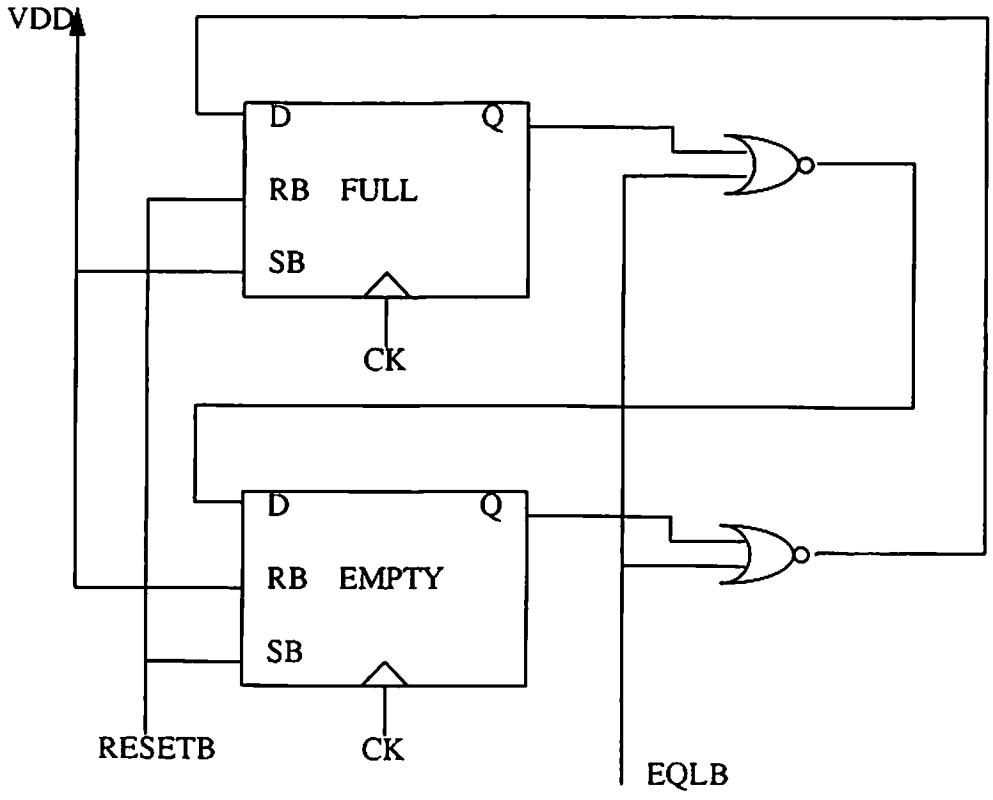


Fig. A5 Flag Logic

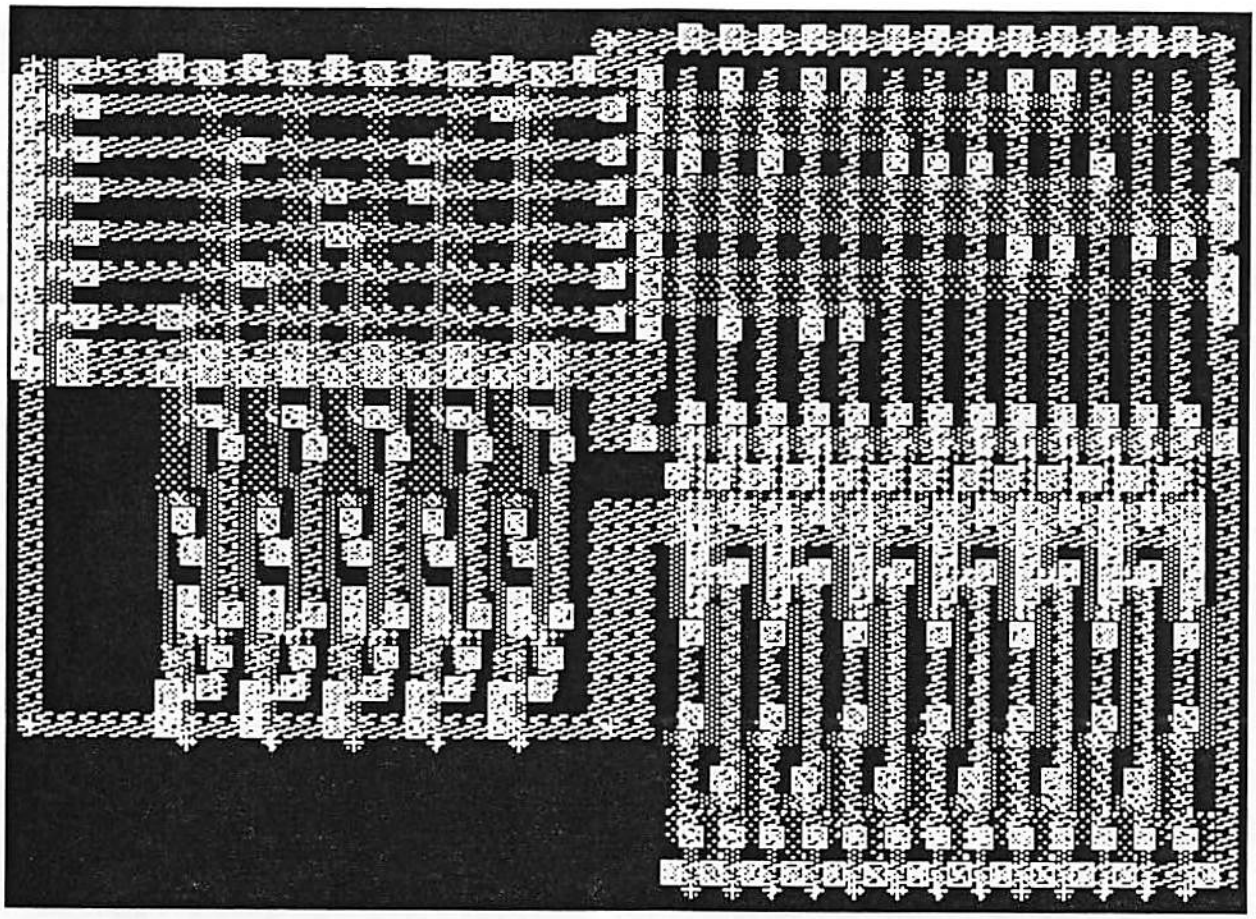


Fig. B1. Control Logic (166 *238 lambda^2)

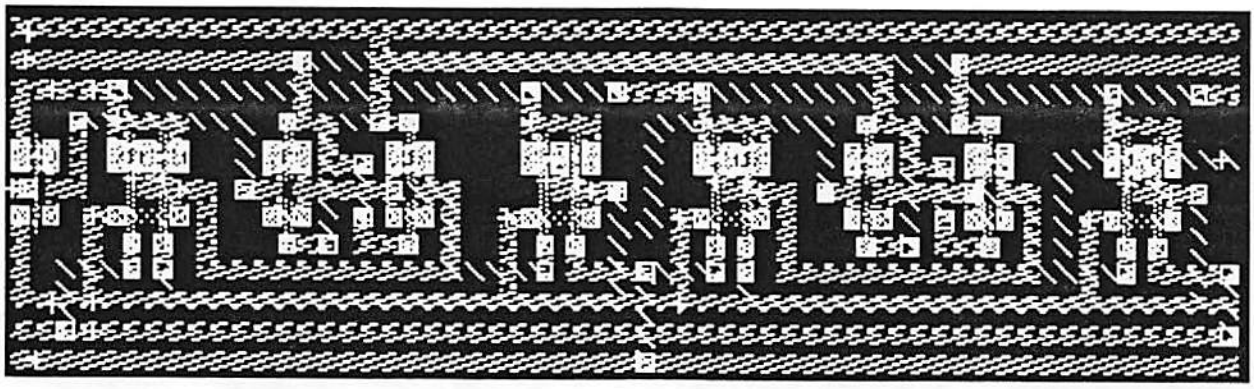


Fig. B2. D flip-flop (91 * 322 lambda^2)

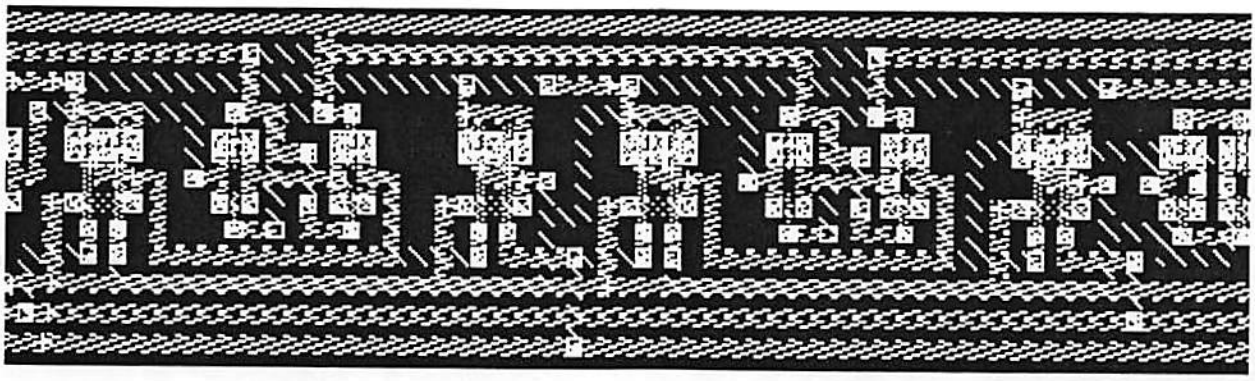
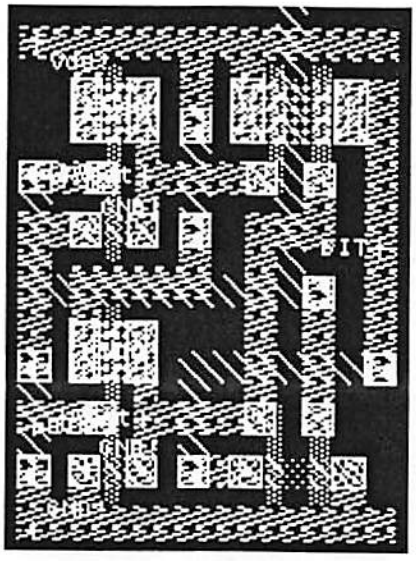
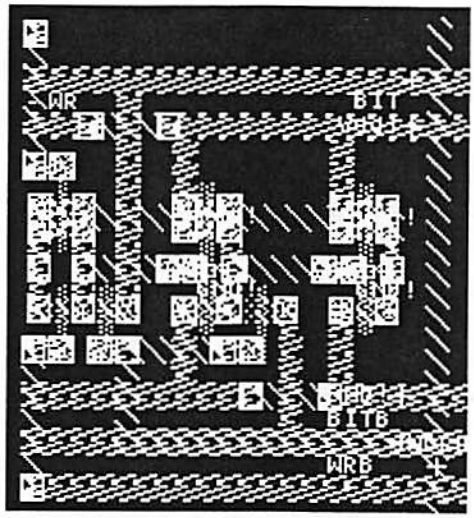


Fig. B3. D flip-flop with output enable (91 * 367 lambda^2)

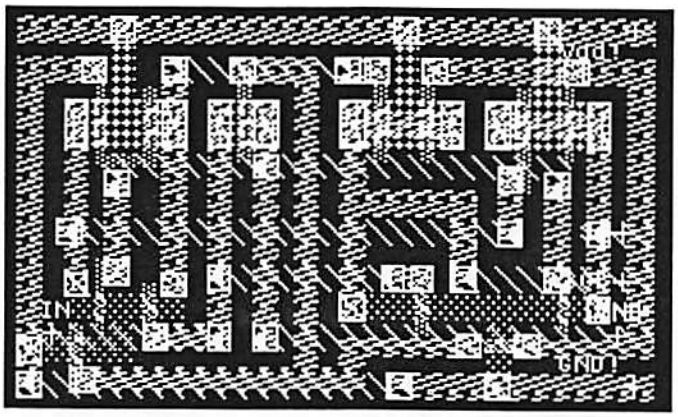


Datain (72 * 51 lambda^2)

Fig. B4. Bitline (to be continued)

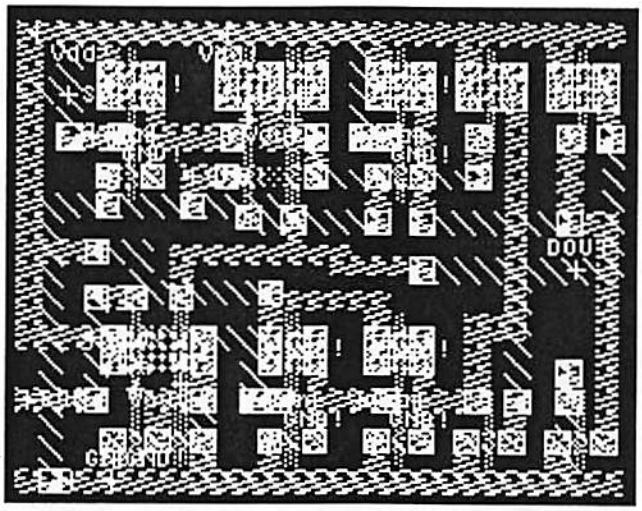


Cell (96 * 80 lambda^2)



Sense Amplifier (64 * 109 lambda^2)

Fig. B4. Bitline (to be continued)



Dataout (82 * 107 lambda^2)

Unexpanded Bitline

Expanded Bitline (96 * 1581 lambda^2)

Fig. B4. Bitline (continued)

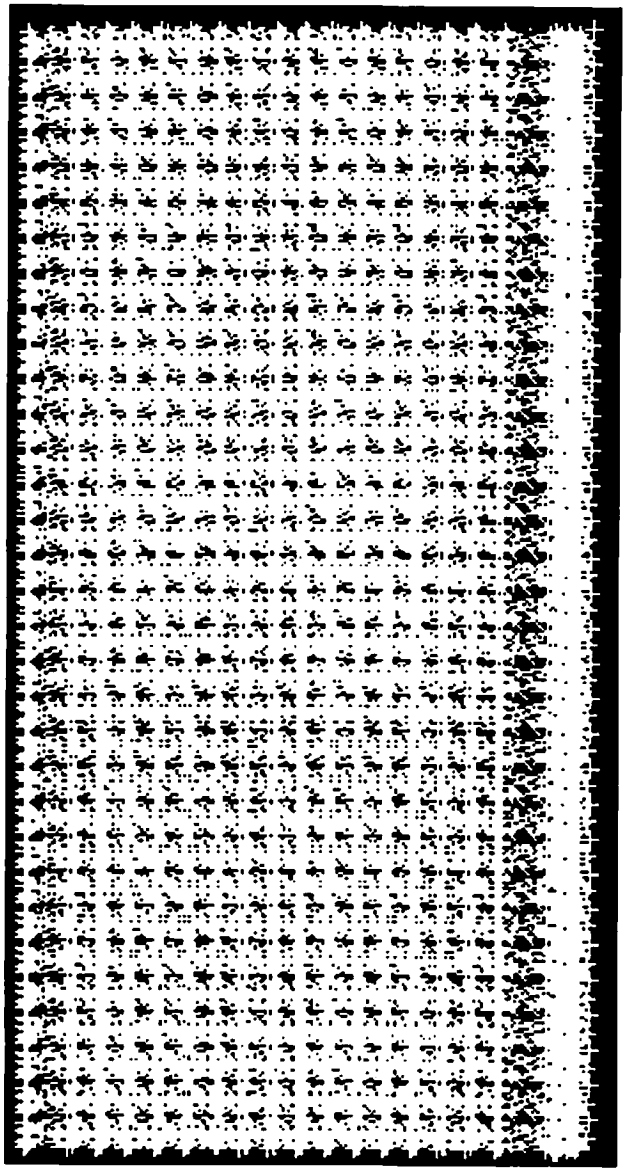


Fig. B5. Memory. 16 by 32 (3073 * 1581 lambda^2)

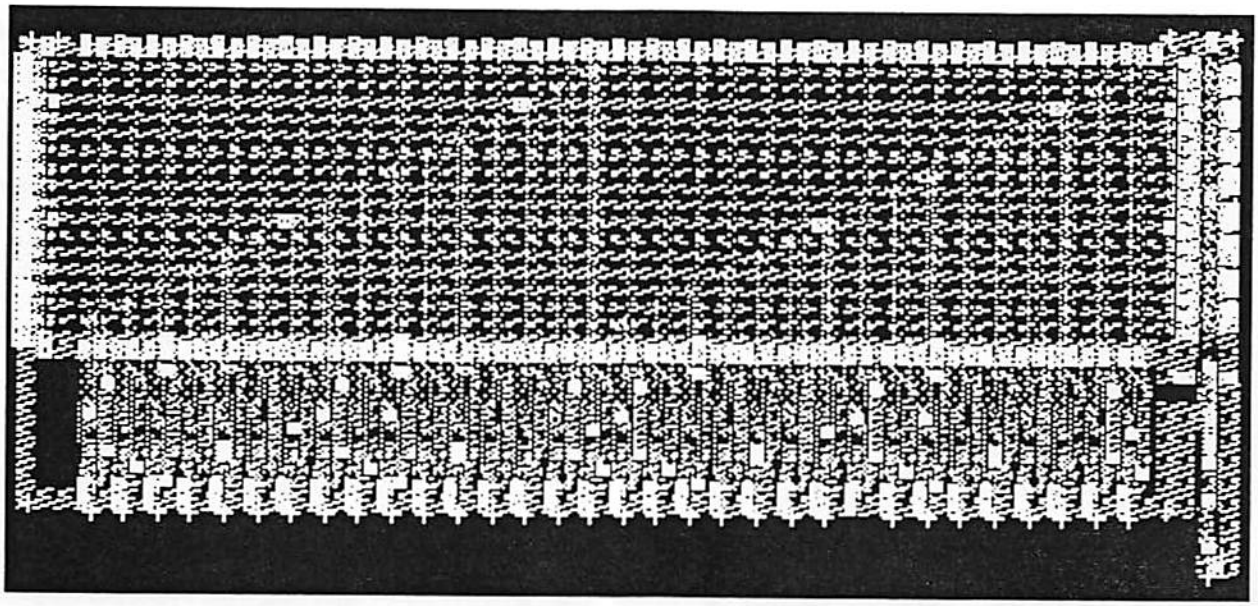


Fig. B6. Comparator (258 * 586 lambda^2)

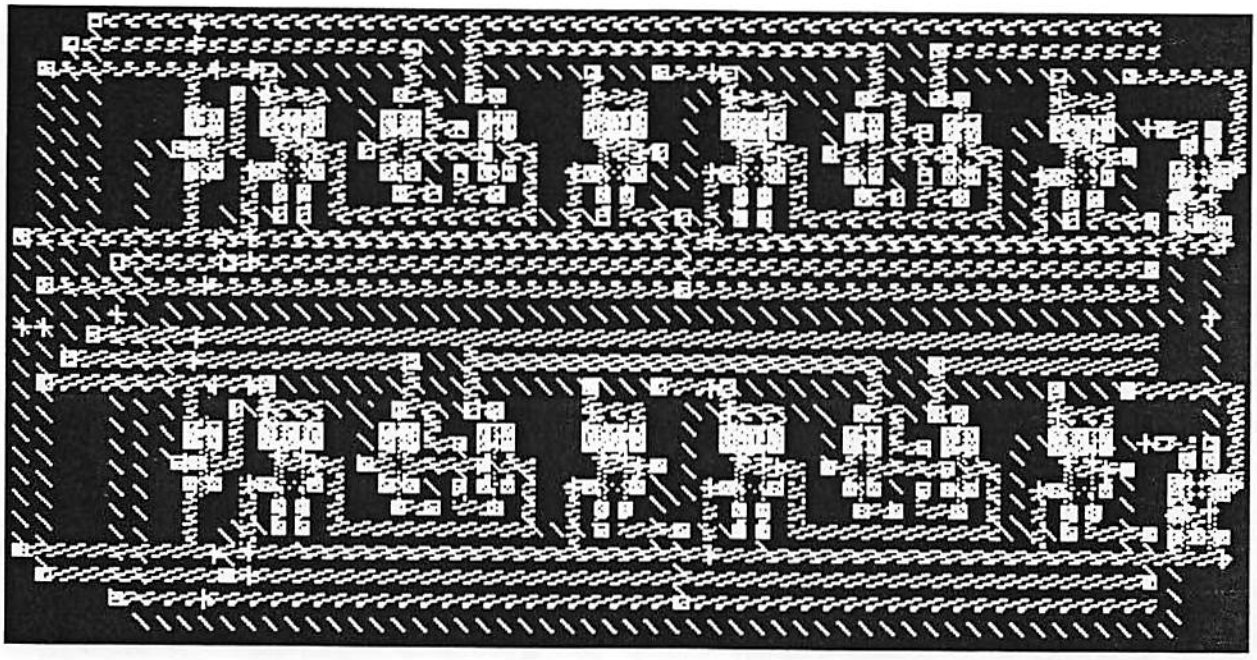


Fig. B7. Flag Logic (202 * 407 lambda^2)

Delay of Control Logic

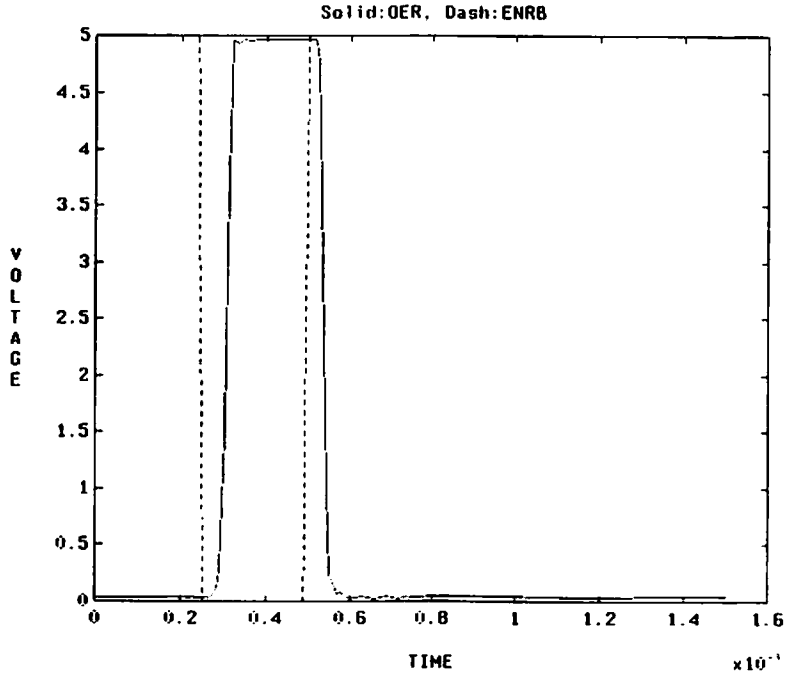


Fig. C1. ENRB is a signal from outside world to request reading
OER is generated to enable output of read pointer

Delay of D flip-flop

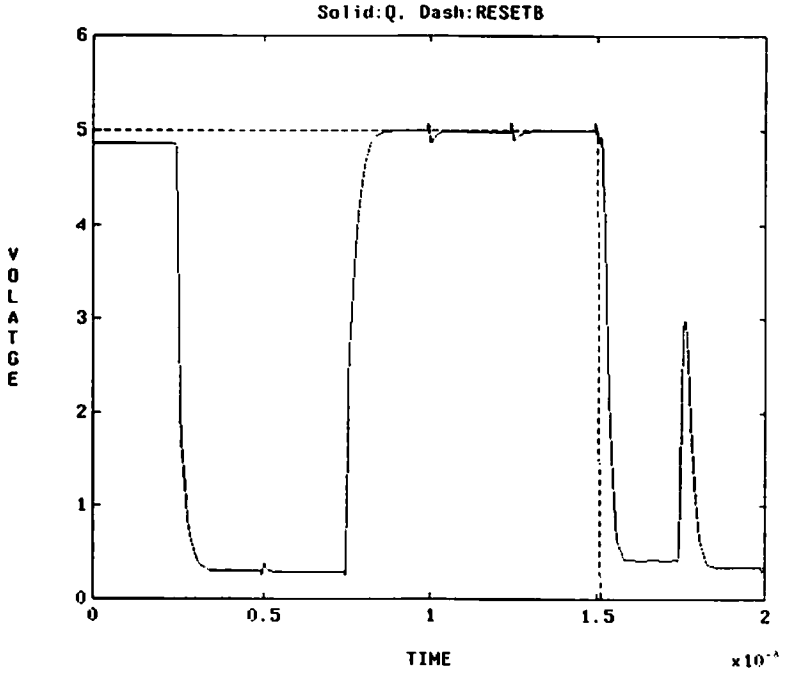


Fig. C2. t_{RESET}

Delay of D flip-flop

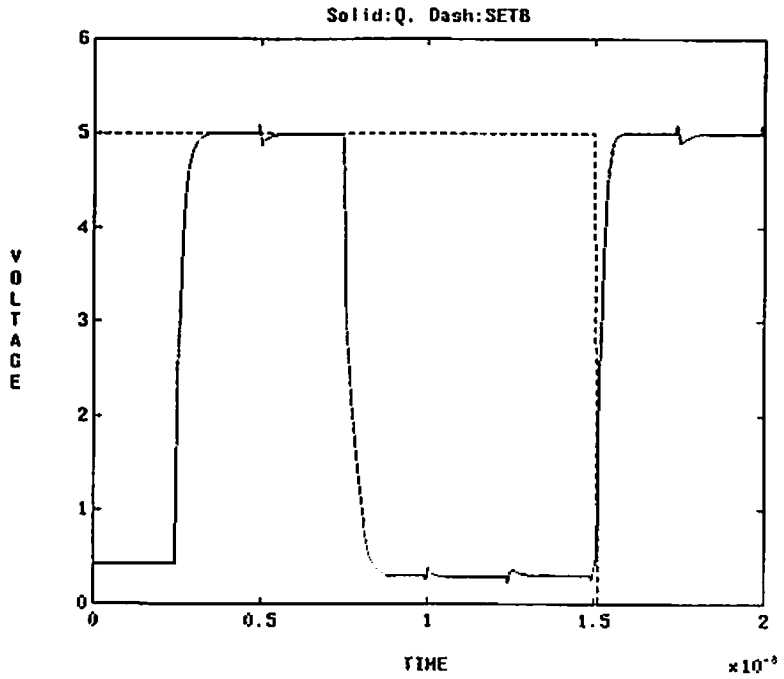
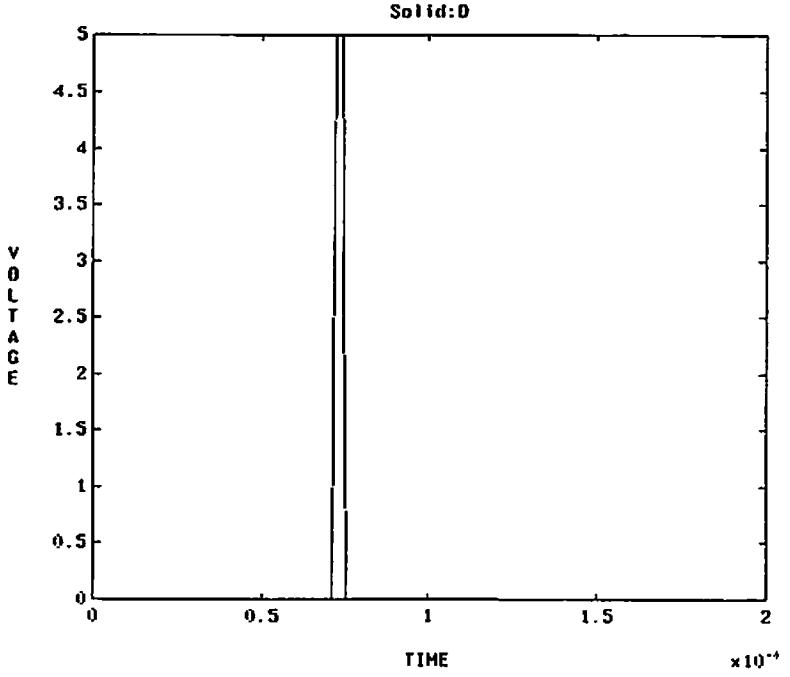


Fig. C3. t_{SET}

Delay of D flip-flop



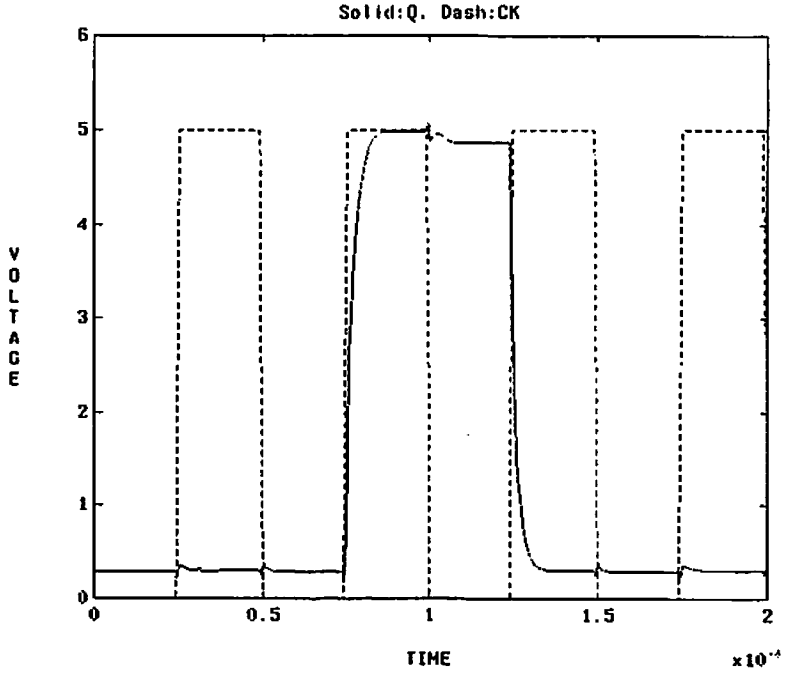


Fig. C4. t_{PLH}

Delay of D flip flop

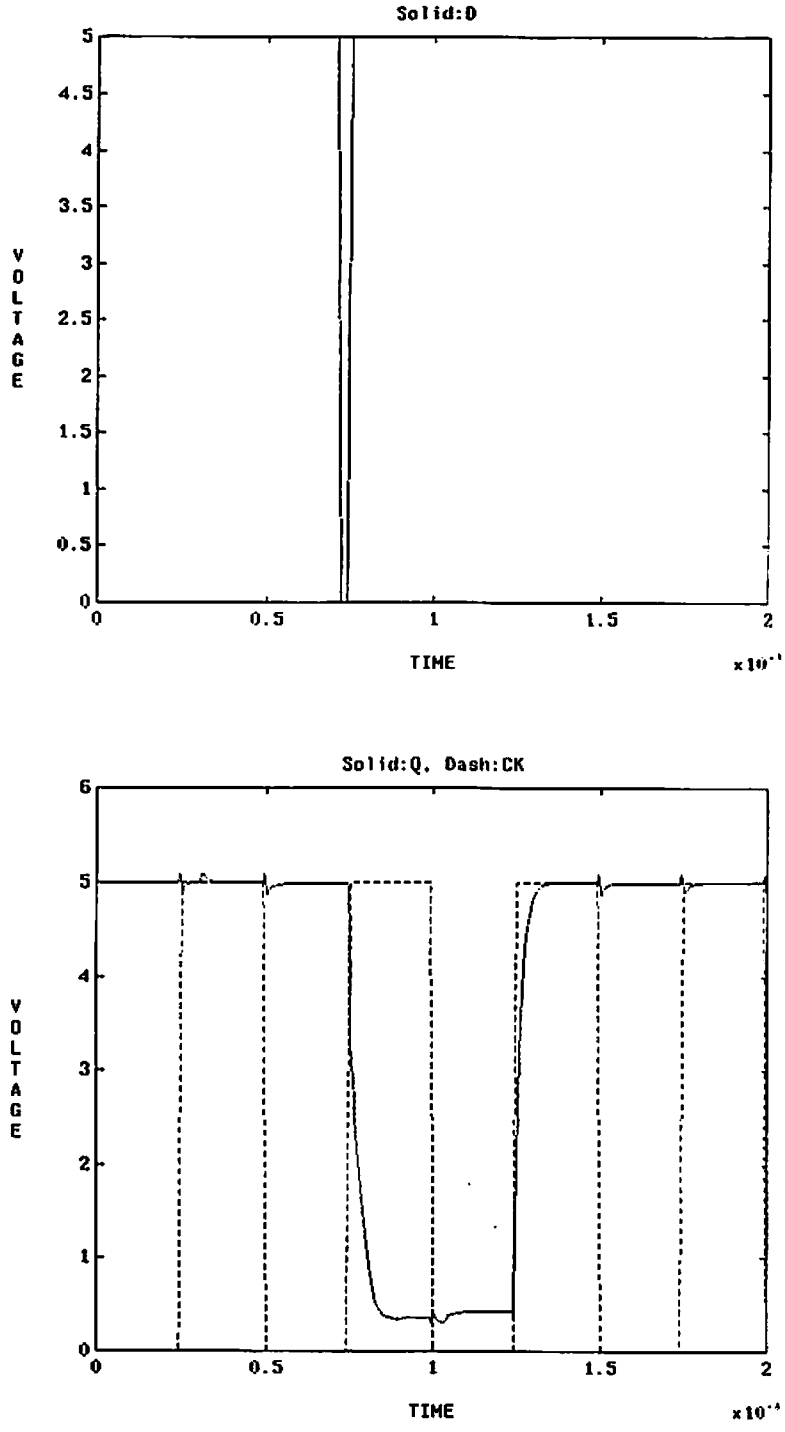
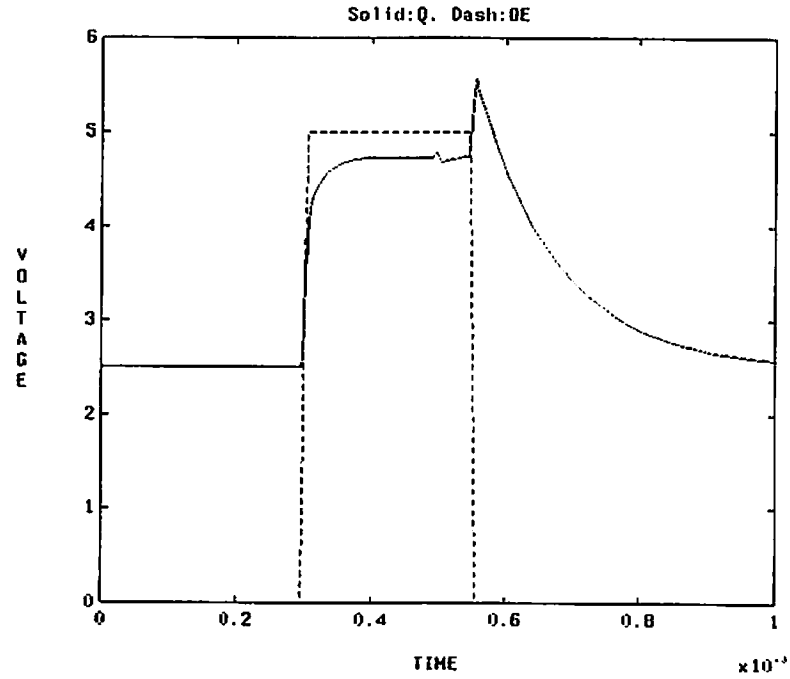
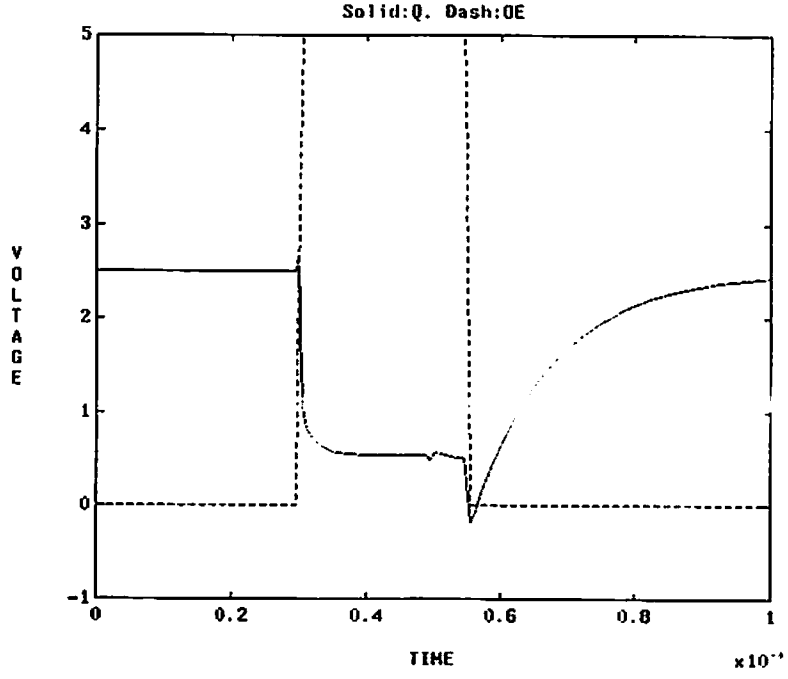


Fig. C5. t_{PHL}

Delay of D flip-flop with output enable



Delay for logic 1



Delay for logic 0

Fig. C6. Delay of a transmission gate in D flip-flop

Write delay of a memory cell

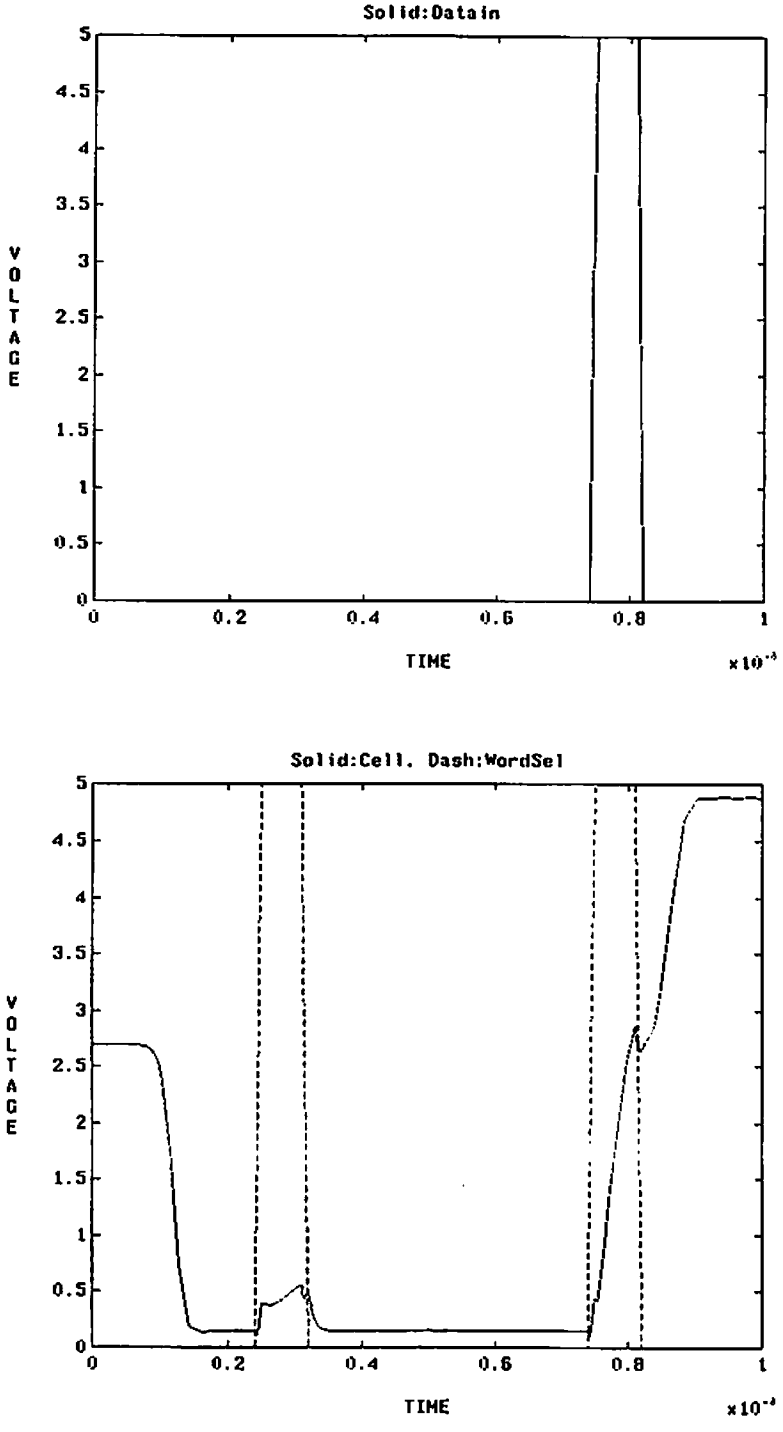
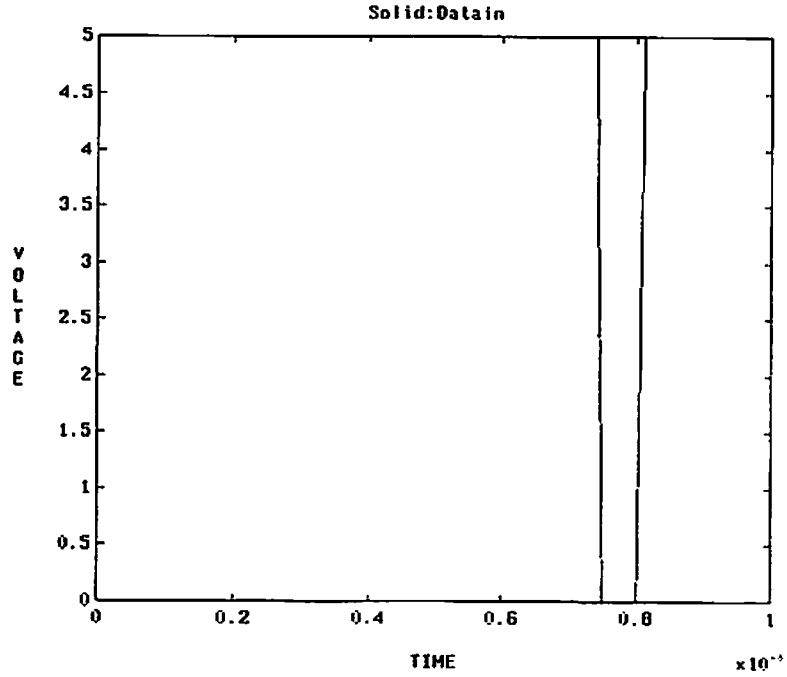


Fig. C7. t_{PLH}

Write delay of a memory cell



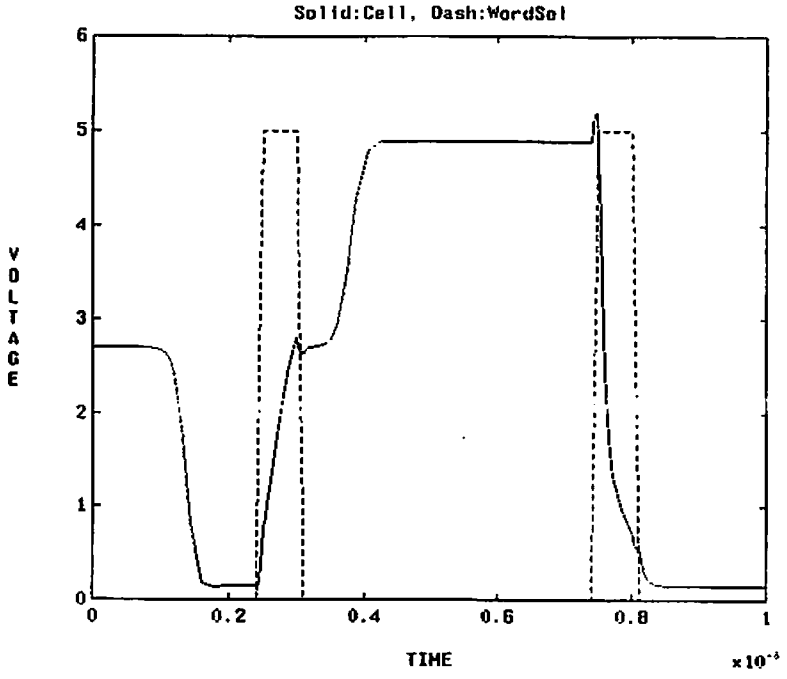


Fig. C8. t_{PHL}

Delay for reading logic 0 from a memory cell

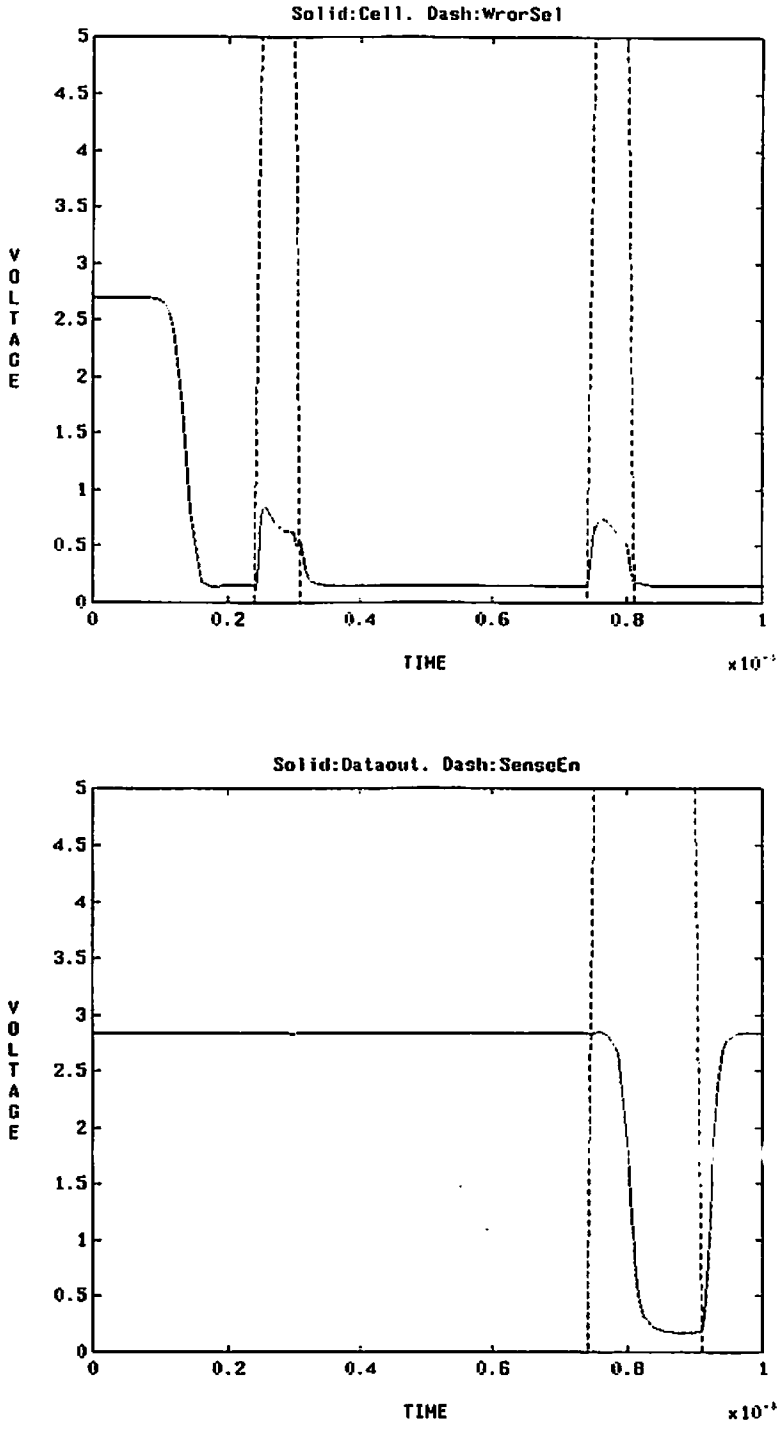


Fig. C9. $t_{P,READ\,0}$

Delay for reading logic 1 from a memory cell

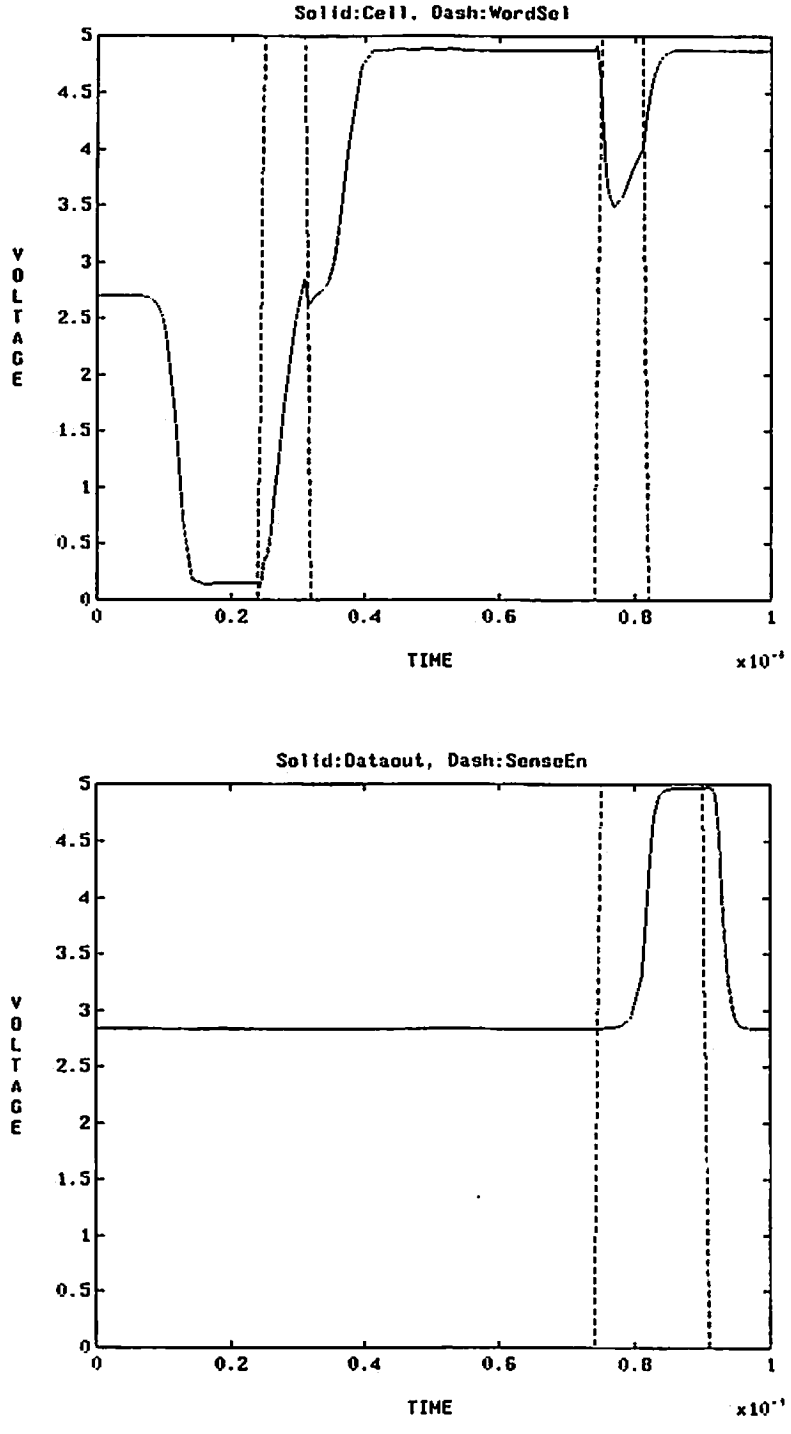


Fig. C10. $t_{P, READ\ 1}$