AN ABSTRACT OF THE THESIS OF

Bo Zhou for the degree of Master of Science in Electrical & Computer Engineering presented on December 2, 1996.

Title: High Speed Digital FIR Filter Design

Abstract approved: Redacted for Privacy

Shih_Lien Lu

The objective of this thesis is to design a high speed digital FIR filter. The inputs of the system come from a Delta-Sigma modulator. This FIR filter takes 1024 inputs, multiplies them with their coefficients and adds the results. The main design task is to take the input data, which are unweighted single-bit binary numbers at 156MHz, multiply each bit with the corresponding coefficient and add them to get a weighted multi-bit output at 20MHz.

High Speed Digital FIR Filter Design

by

Bo Zhou

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Completed December 2, 1996
Commencement March 1997

Master of Science thesis of Bo Zhou presented on December 2, 1996

APPROVED:

Redacted for Privacy

Major Professor, representing Electrical & Computer Engineering

Redacted for Privacy

Chair of Department of Electrical & Computer Engineering

Redacted for Privacy

Dean of Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Redacted for Privacy

Bo Zhou, Author

ACKNOWLEDGEMENTS

With utmost respect and gratitude, I want to thank Dr. Shih_Lien Lu, my supervisor. His broad knowledge and sharp thinking gave the initial idea for this project, and his kindness and patience encouraged me through the project. I greatly appreciate his the support on this project and on my whole master study.

Many thanks to Dr. Jack Kenney, Dr. David J. Allstot and Dr. James Welty for taking their precious time out of their busy schedules to serve on my graduate committee and to read the manuscript of this thesis.

Thanks to Dr. Richard Schreier, Mr. Bo Zhang, Mr. Haiqing Lin, Mr. Bo Wang and Mr. Wenjun Su for their kind help with the thesis.

Thanks to Ms. Rita Wells for her help with all the tedious paper work that makes this thesis possible.

Thanks to Mr. John Seredich at Silicon Systems Incorporated for taking the effort to help me fix the format and present the thesis in such an elegant way.

Thanks to Westinghouse Electric Co. for their financial support on the project.

And finally, I want to say thank you to my husband and my little boy for all the precious things you give to me.

TABLE OF CONTENTS

Page

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

Page

LIST OF FIGURES

Figure **Page**

LIST OF FIGURES (Continued)

Figure Page

LIST OF TABLES

LIST OF APPENDIX FIGURES

LIST OF APPENDIX FIGURES (Continued)

LIST OF APPENDIX FIGURES (Continued)

Figure **Page**

LIST OF APPENDIX FIGURES (Continued)

Figure **Page**

High Speed Digital FIR Filter Design

Chapter 1. Introduction

In the last two decades, Digital Signal Processing (DSP) has made enormous progress both in theory and practice. With the advancement of Very Large Scale Integrated Circuit (VLSI) technology, more and more applications are using DSP as the primary solution due to the reliability, reproducibility, compactness and efficiency of the digital technology. The objective of this thesis is to design a high speed digital Finite Impulse Response (FIR) filter, which is part of a Delta-Sigma Analog to Digital Converter. This filter will be part of a digital decimator which is used to convert the high-speed bit-stream output of the Delta-Sigma modulator into Nyquist-rate PCM data. Conceptually, this operation consists of two parts: lowpass filtering and down-sampling. For the sake of economy, the over-sampling ratio can be reduced in stages. Since a single-stage filtering down-sampling block requires more than 10,000 taps to achieve our specification requirement, we employed the design of a multiple-stage decimator. This work describes the design and implementation of the first-stage decimator. The inputs of the system come from a Delta-Sigma modulator. This Finite Impulse Response filter takes 1024 inputs, multiplies them with their coefficients and adds the results. The main design task is to take the input data, which is unweighted single bit binary numbers at 156MHz, multiply each bit with the corresponding coefficient and add them to get a weighted multibit output at 20MHz.

1.1. FIR Filter System

The so-called FIR filter is a widely used filter in DSP. Mathematically, a sampled data FIR filter is represented by:

$$y(n) = \sum_{i=0}^{m-1} c_i x(n-i)$$

Where

y(n) = the output of the filter

x(i) = the input signal stream

c_i = the ith coefficient of the filter

m-1 = the order or length of the filter

In general, we can view the equation as a computational procedure (an algorithm) for determining the output sequence y(n) of the system from the input sequence x(n).

There are a number of well-known forms for the sampled data FIR filter. One form is shown in Figure 1.1. It is the so called Direct-Form Structure FIR.

The system of a FIR filter is mainly composed of registers, adders and multipliers when implemented with hardware. The precision of the adder and multiplier depend on the precision of the coefficient, the length of the filter, and the desired precision or result. The multipliers can be a costly component. If the filter response is fixed and the coefficients are therefore fixed, the multipliers may be simplified to contain only the product terms required.

In fixed FIR filters, a great deal of signal-processing expertise goes into designing the coefficients that reduce the number of additions efficiently.

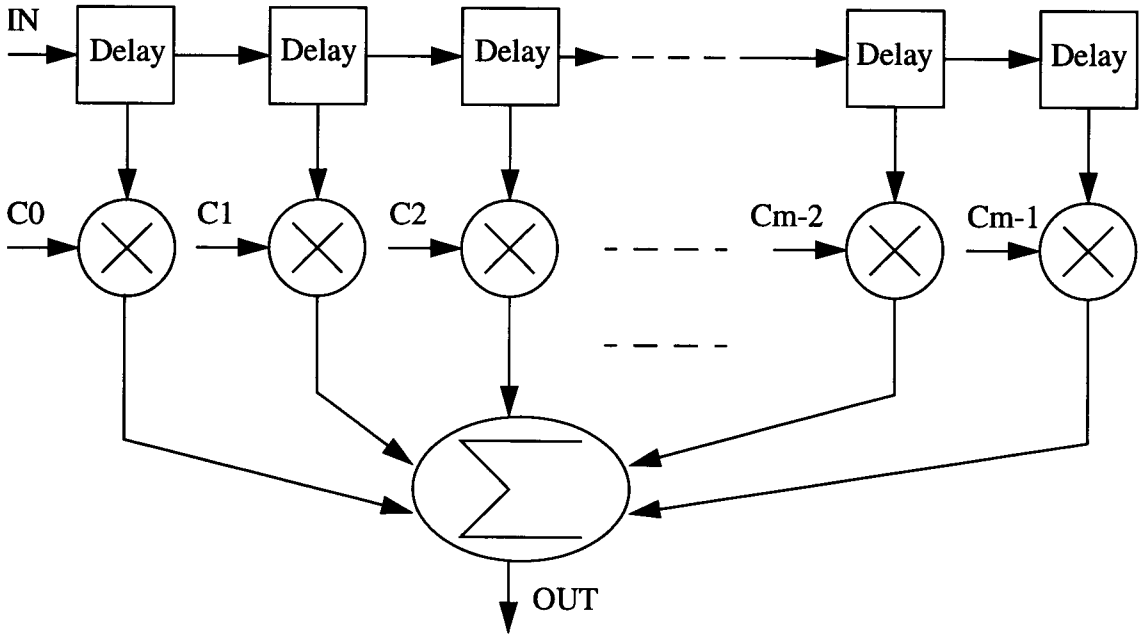


Figure 1.1 Direct-Form Realization of FIR Filter Structure

1.2 Linear Phase FIR

An FIR filter has linear phase if its unit sample response satisfies the condition

$$c(n) = \pm c(M-1-n) \qquad n = 0, 1, \ldots, M-1$$

The system we are dealing with is a linear phase FIR filter, so we can use this symmetry or antisymmetry characteristic to simplified the system. So the structure shown in Figure 1.2 is used for the design.

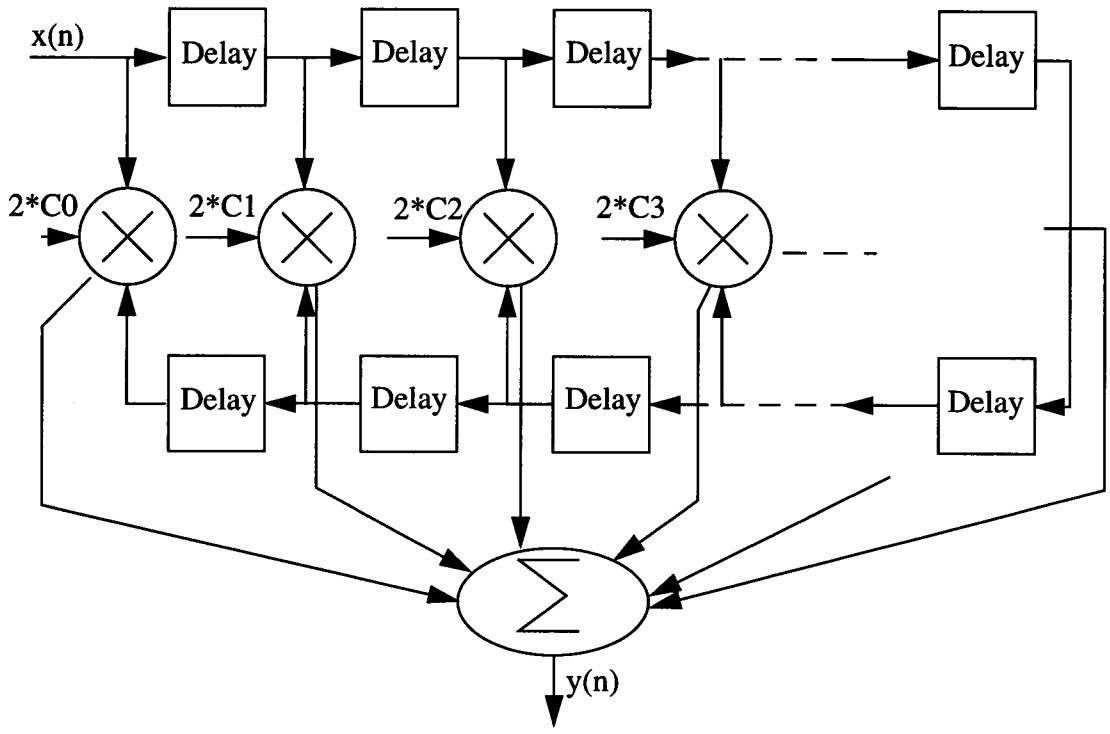


Figure 1.2 Direct-Form Realization of Linear-Phase FIR System

1.3 System Outline

The oversampled bit-stream is clocked at 10 GHz. An intermediate multiplexer reduces the one-bit-stream at 10 GHz into 64-bit parallel data at 156 MHz. The final output of the FIR filter will be clocked at 20MHz. As a result the system will have to work at a high speed.

There are two main obstacles for the design. One is to finish the large amount of multiplication and addition within a relatively short period of time, the other is managing the large amount of data flow. Several main design methods are taken here to achieve the above mentioned goals. They are:

1. XOR reduces the number of additions at the first step.

2. *AND* gates implement the multiplications.
3. *Wallace Tree* structure reduces the large number of additions.
4. *Combination of carry select and carry lookahead* does the final addition.
5. *Pipelined* structure manages the data flows.

There are three basic circuit elements used in the design. They are: D flip-flop, XOR, and full adder. Besides these three repeatedly used elements, there are some other circuit macros, such as the carry lookahead and carryselect macro, which are also used in the implementation of the filter. In this design, since the whole circuit consists numerous number of repeating macros, the optimally of each basic cell can contribute greatly to the overall performance of the circuit. Using the minimum number of transistors for the basic macros can reduce the parasitic capacitance for the circuit, which will contribute not only to lower power consumption, but also benefit the delay. Spice simulation and first step layout have been done for the basic elements of the circuit in order to estimate the overall chip area and circuit performance. Powerview was used to set up the schematic structure for the design. Logic simulation has been done for the macros of the system.

1.4 Organization of the Document

Chapter 2 presents the overall structure of the system. Chapter 3 explains the design of the D flip-flop and shows the simulation results of its main characteristics. Chapter 4 explains the design of Full Adder and discusses the affect of temperature variation, power supply variation and transistor size optimization on its delay. Chapter 5 presents all the function blocks of the system and discusses the different structures of Wallace Tree. Chapter 6, gives the conclusion of the design and some suggestions about the future work.

Chapter 2. Overall Structure of the System

A general electronic system is a black box that performs a desired input/output transformation. A digital system has its input and output coded in binary format. Like any other digital system, our design accepts certain binary inputs and produces certain outputs. The inputs for the system are 64 unweighted bits in parallel at 156MHz. The output for the system is one multi-bit word at 20MHz. The transformation of this system involves taking these $64\left(\frac{156M}{20M}\right) = 1024$ inputs, multiplying them with the corresponding 20-bit 1024 coefficients, and adding the results together. The final output is a 30 bit words. Many times digital systems are advantageous because they can be partitioned into modules. Each module can be built with cells which act as the building blocks. The main modules of the system are: the input part, including the shift register block and the latch register block; the pre-processor part, including the XOR block and the AND block; and the addition part, including Wallace Tree and carry-look-ahead & carry-select block.

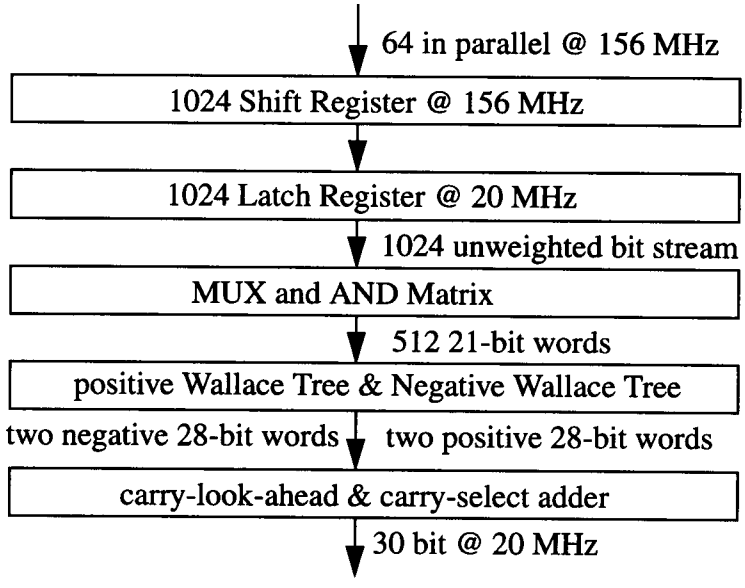


Figure 2.1 Overall Structure for the System

Figure 2.1 is the block diagram for the overall structure of the system. These blocks can be divided into three parts: the inputs part, the pre-processing part, the processing part. This chapter will describe the blocks in each part individually.

2.1 Input Part

The input part reads in the input data and converts them to certain data form that is appropriate for the data-processor of the system to deal with. Shift register block and latch register block are used in this part to achieve the goal.

2.1.1 Shift Register.

The inputs of the system are 64 unweighted bits at 156MHz in parallel. The system needs to take in 1024 input data, then do the data processing every 50ns (@20MHz). The shift register block reads in the 64 parallel inputs every 6.4ns (@156MHz), shifts them down and stores them in 64 register levels. So the shift register block is composed of 64 levels of register cells. Each cell contains 64 DFF's in parallel. All the registers in this block work at 156MHz.

2.1.2 Latch Register

Since the clock for the output is 20MHz, the 1024 156MHz input needs to be latched at 20MHz and sent to the next stage in parallel. The latch register reads the outputs of the 64×64 matrix shift register every 50ns, and sends them to the next stage in parallel.

2.2 Pre-Processor Part

The large number of data is a big obstacle for the design. Pre-processor makes use of the characteristic of the system and reduces the number of inputs before the main data-processor. By this, the size of the addition part can be reduced dramatically. The blocks in this part are the XOR block and the AND block. The XOR block reduces the number of input data entering the filter, the AND block does the multiplication for the filter.

2.2.1 XOR

The coefficients of FIR filter are symmetric, i.e. $C(i) = C(n-i)$ or antisymmetric, i.e. $C(i) = -C(n-i)$. We can use this characteristic to try to reduce the number of data that the processing part. If the inputs are symmetric, i.e. $IN(i) \oplus IN(n-i) = 0$, the multiplications and the addition of the two multiplications will give $2 \times C(i)$ or $2 \times C(n-i)$. If they are anti-symmetric, i.e. $IN(i) \oplus IN(n-i) = 1$, the addition of the two multiplications will produce a zero.

Based on this, we add a XOR block before the filter itself to find if the coefficient is symmetric or anti-symmetric. This XOR block compares coefficients in parallel and drives the AND block.

2.2.2 AND

The multiplication is done by bit to bit AND-gates, which are equivalent to 2-to-1 muxes. The input of the mux are individual bits of the $coefficience \times 2$, which are ready for the chip and the outputs of the XOR's.

2.3 Data Processor Part

The Data Processor Part is the main part of the FIR filter. It contains the addition of all the outputs from the multiplications. It begins with the Wallace Tree structure, and uses the carry-look-ahead and carry-select adder to do the final addition of the output from the Wallace Tree.

2.3.1 Wallace Tree

Wallace Tree structure provides the structure for additions. Based on the manageability and speed trade-off, two basic counters (15,4) [15-to-4] and (3,2) [3-to-2] are used. It is assumed that the coefficient of the filter are evenly distributed positive and negative numbers.

Wallace Tree can not deal with the sign-bit. So the data is grouped into positive and negative group numbers, the sign bit information will be kept for the final addition. But practically, the sign of the coefficient of the filter is random. Duplicatation of this design and some minor changes can be done to achieve this. Section 5.4 discusses this in detail.

The Wallace Tree compose several levels that can convert 1024 20-bit number to 4 28-bit numbers gradually.

2.3.2 Carry-Select and Carry-Look-Ahead Adder

The Carry-Select and Carry-Look-Ahead Adder takes four 28-bit output from Wallace Tree, along with the sign bits, and calculates the sum of these four 29-bit

numbers. Carry-select and carry-look-ahead are used in this block to achieve higher speed.

2.4 Summary

The chapter briefly describes the overall structure of the digital filter. The detail functionary and performance will be described in later chapters.

Chapter 3. Full Adder

3.1 Full Adder

A full adder is a combinational circuit that forms the arithmetic sum of three input bits. It consists of three inputs and two outputs. Two of the input variables are the two bits needed to be added. The third input is the carry out from the previous lower weighted addition. These three inputs are equally weighted. Because the arithmetic sum of the three binary digits ranges in value from 0 to 3, two binary bits are needed for the outputs. One output has the same weigh as that of the three input bits, the other has a higher weight than that of the inputs. The truth table of the full adder is shown in Table 3.1.

Table 3.1 Truth Table for Full Adder

Inputs			Outputs	
A	B	C	S	COUT
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

The eight rows under the input variables designate all possible combinations of 1's and 0's that these variables may have. The 1's and 0's for the output variables are determined from the arithmetic sum of the input bits. When all input bits are 0's, the

output is 0. The S output is equal to 1 when only one input is equal to 1 or when all three inputs are equal to 1. The C output has a carry of 1 if two or three inputs are equal to 1.

So the truth table gives one simple definition about full adder: a full adder finds the number of ones in the inputs, and gives that number in binary form. Thus, it is actually a 3-to-2 counter. The schematic of a full adder using static CMOS complementary gate is given in Figure 3.1

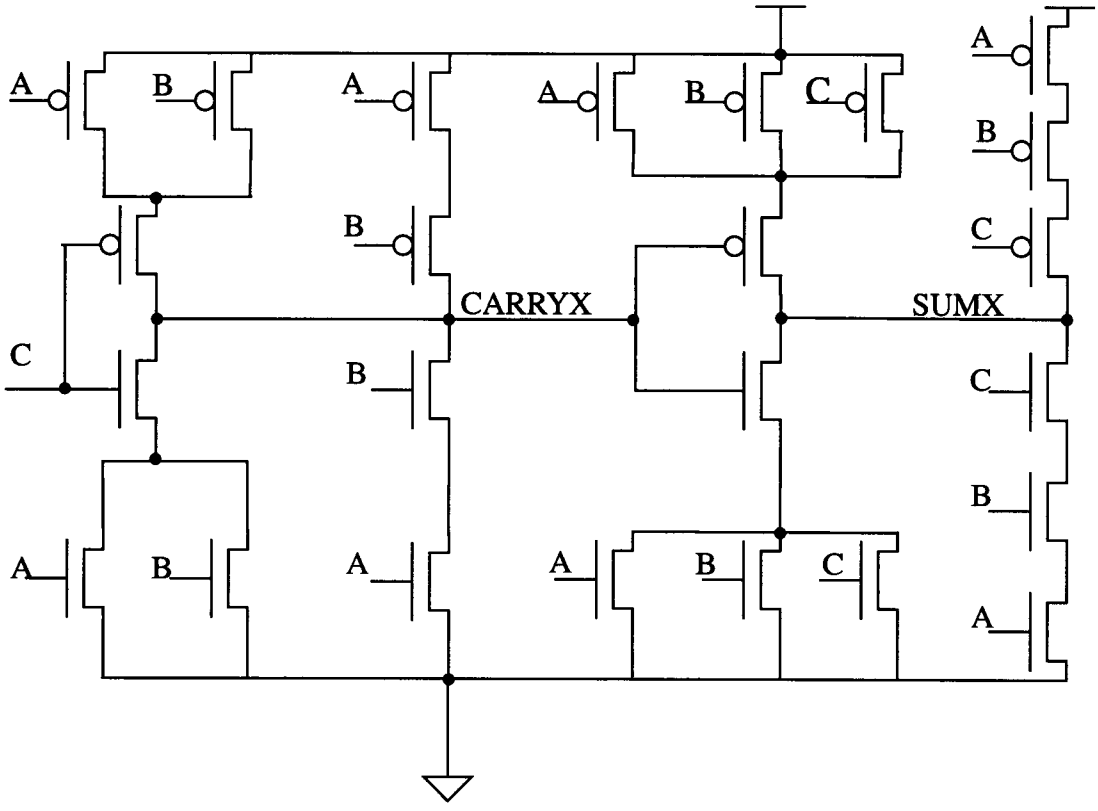


Figure 3.1 Complementary CMOS Full Adder

3.2 The Comparison of Static CMOS Adder and Transmission Gate Adder

Besides the static CMOS full adder, a different implementation of adder uses a novel exclusive-or (XOR) gate. The schematic for this XOR gate is shown in Figure 3.2.

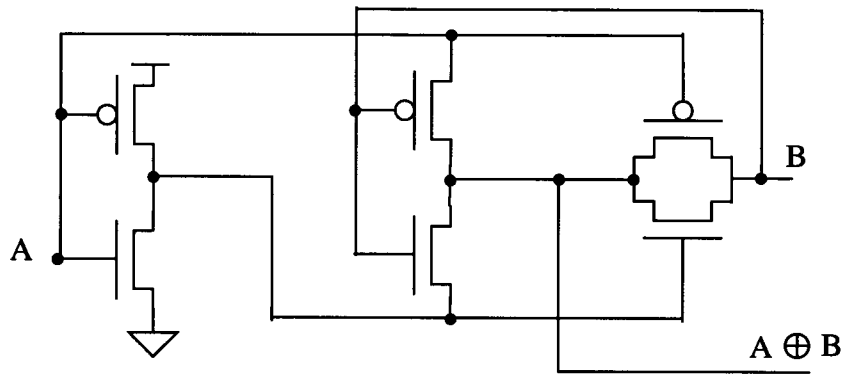


Figure 3.2 Transmission Gate XOR

With XOR, inverters and transmission gate, a full adder may be implemented as the Figure 3.3. The SUM ($A \oplus B \oplus C$) is formed by a multiplexer controlled by $A \oplus B$. If a second thought is given to the truth table of full adder, it can be found that Carry=Cin when $A \oplus B = 1$, and Carry=A (or B) when $A \oplus B = 0$.

This adder has 24 transistors, the same as the complementary one, but the Carry and Sum have the same delay. In addition, the Sum and Carry signals are noninverted. The disadvantage of this structure is that the delay is much worse than the static CMOS design according to *10*.

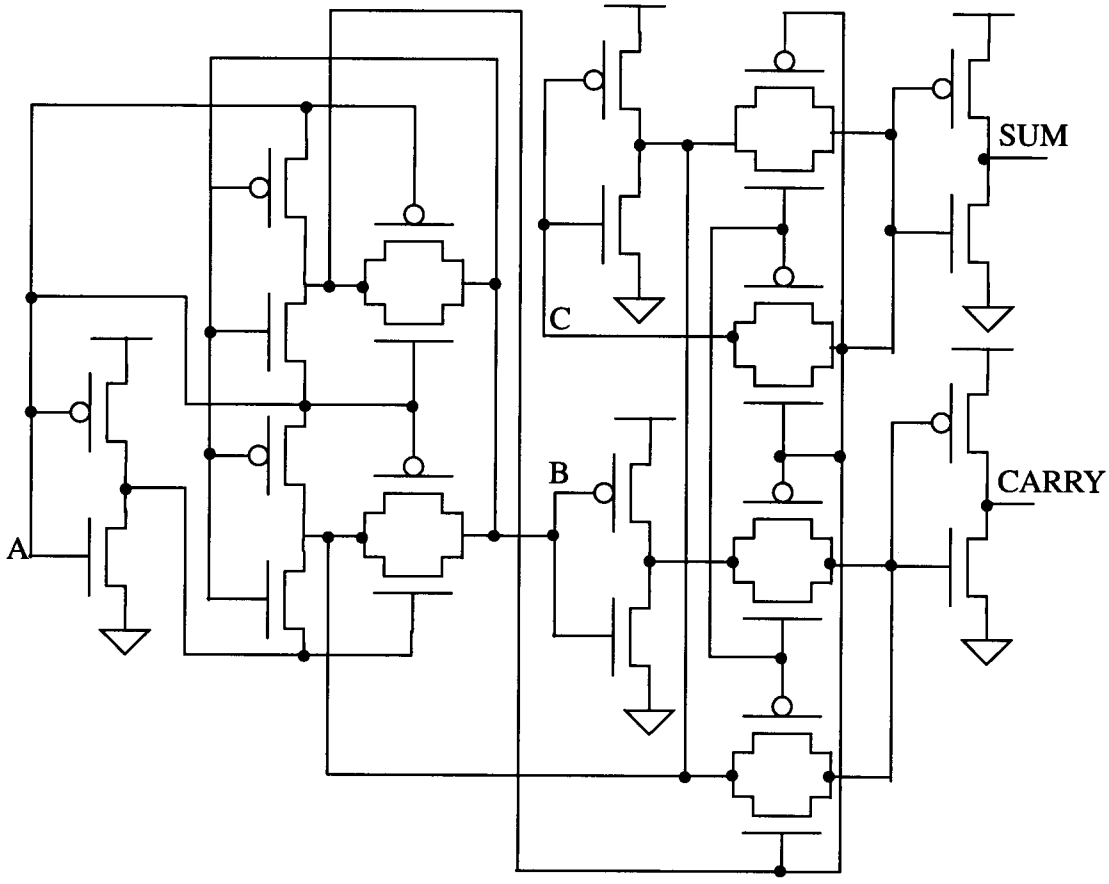


Figure 3.3 Transmission Gate Full Adder

3.3 Static CMOS Full Adder Simulation

There are several factors that affect the circuit performance. Simulations are done with regard to the effect of temperature, transistor size and power supply variations.

3.3.1 The Affect of Temperature on Circuit Performance

The circuit performance is largely decided by the I-V characteristic of the transistors used in the circuit. The delay, for instance, is decided by the charging or

discharging current of the output node and the parasitic or real capacitance of that node. For the MOSFET used in this design, the I-V characteristic is:

$$0 < V_{ds} < V_{gs} - V_t \qquad I_d = K\frac{W}{L}\left[(V_{gs} - V_t)\cdot V_{ds} - \frac{1}{2}\cdot V_{ds}^2\right]$$
$$V_{gs} > V_t$$

$$I_d = \frac{1}{2}K\frac{W}{L}(V_{gs} - V_t)^2 \qquad V_{ds} > V_{gs} - V_t \qquad V_{gs} > V_t$$

$$I_d = 0 \qquad V_{ds} \geq 0 \qquad V_{gs} < V_t$$

In the above equations, $K = \mu C_{ox}$, where μ is the mobility of the carrier of the MOSFET, C_{ox} is the capacitance density (capacitance per unit area). Both μ and V_t are heavily temperature dependent, while they have opposite affect on the I-V characteristic. μ is the mobility of the channel. There are two collision or scattering mechanisms that dominate in a semiconductor and affect the carrier mobility: lattice scattering and ionized impurity scattering. For the channel mobility of MOSFET, lattice scattering is the main mechanism because of the low impurity concentration. The lattice scattering is related to the thermal motion of atoms.

To the first order approximation,

$$\mu \alpha T^{\left(-\frac{3}{2}\right)}$$

So the temperature coefficient for the channel mobility μ is negative, i.e. the mobility increases as the temperature decreases.

On the other hand, the threshold voltage is a function of temperature too. The absolute value of the threshold voltage decreases with an increase in temperature. This variation is approximately -4mv/$°C$ for high substrate doping levels, and -2mv/$°C$ for low doping levels.

So the affect of temperature on the circuit depends on the combination of the two factors. In oder to get a rough feeling about the affect of these two factors, the following calculation is done to compare the time for a inverter's output node drop from 5v to 4v under 25$°C$ and -220$°C$.When the output of a inverter drops from 5v to 4v, the pmos is in cutoff region while the nmos is in its saturation region. The saturation current of the nmos provides the discharge current for the node.

$$I_{sat} = (1/2) \cdot (W/L) \cdot K \cdot (V_{gs} - V_t)^2$$

Where: $W/L = 3/2$

$$K = \mu \cdot C_{ox} = 84(\mu/V^2)$$

$$(V_{gs} - V_t) = 5 - 0.7 = 4.3$$

So the $I_{sat} \approx 1200\mu$.

For discharge current, $\Delta t = (\Delta Q)/I = (\Delta V \cdot C)/I$, if the load capacitance is 50fF (a typical gate capacitance for a inverter), the discharge time is approximately 0.04 ns under room temperature. If the temperature is -220$°C$, according to the above discussion of temperature affect on V_t and μ, the V_t will be approximately 1.5v, while

the K will be $\frac{84}{300^{(-3/2)}} \cdot 53^{(-3/2)} = 1234(\mu/V^2)$, so the resulting $I_{sat} = 11000\mu$, which will result in a decrease in discharging time about 10 times.

From the above discussion, it is obvious that the temperature drop will speed up the circuit. Simulations use the following stimulus under different temperatures to verify the effect.

Table 3.2 Stimulus to Simulate the Affect of Temperature on FA Delay

ain	bin	cin	time point	sum	cout
0	0	0	10 ns	0	0
1	0	0	20 ns	1	0
1	1	0	30 ns	0	1
1	1	1	40 ns	1	1
1	1	0	50 ns	0	1
0	1	0	60 ns	1	0
0	1	1	70 ns	0	1
0	1	0	80 ns	1	0

Table 3.3 Delay of FA with Minimized Transistor Sizes

		T=25 C		T=-220 C	
		a (b)	cin	a (b)	cin
sumb	up edge	0.97 ns	0.80 ns	0.68 ns	0.80 ns
	down edge	0.21 ns	0.28 ns	0.11 ns	0.09 ns

Table 3.3 Delay of FA with Minimized Transistor Sizes

		T=25 C		T=-220 C	
coutb	up edge	1.38 ns	1.56 ns	1.40 ns	1.60 ns
	down edge	0.29 ns	0.35 ns	0.16 ns	0.18 ns

From Table 3.3, the delay of the circuit does decrease while the temperature goes down.

3.3.2 The Affect of Transistor Size on Circuit Performance

The sizes of the transistors in the circuit affect the circuit performance because they affect the I-V characteristic of the MOSFET. Generally speaking, the large size gives larger current under the same nodes' voltages, which means a stronger driving ability. But a larger size transistor means a larger load for the stages that drive it. So the affect of the sizes for the transistors on the circuit performance is not a straight forward relation.

Table 3.4 Delay of FA with Optimum Transistor Sizes

		T=25 C		T=-220 C	
		a (b)	cin	a (b)	cin
sumb	up edge	0.76 ns	0.65ns	0.68 ns	0.60 ns
	down edge	0.14 ns	0.25 ns	0.10 ns	0.09 ns
coutb	up edge	1.12 ns	1.22 ns	1.08 ns	1.30 ns
	down edge	0.29 ns	0.25 ns	0.16 ns	0.11 ns

The simulation results show that the speed of the full adder relies on the sizes of the transistors in the circuit. The "optimum size" taken here is from *1*. The optimum may not be suitable for the circuit, but it does give some idea that the size of the transistor affects the circuit performance. A variety of software packages have been developed to aid in the optimization of transistor sizes.

3.3.3 The Effect of Power Supply Variation on Circuit Performance

In an IC chip, the variation of power supply is very normal and usually within the range of 10%.

Generally speaking, the increase of power supply means large current for the same transistor, so that will make the circuit faster. Simulation is done using the same stimulus shown in Table 3.2 on page 17 while the VDD is changed from 4.5 to 5.5v. The simulation results are shown in Table 3.5.

Table 3.5 The Delay of FA with Vdd Variation

Out-put	Edge	Input A or B			Carry Input		
Sb		4.5 V	5.0 V	5.5 V	4.5 V	5.0 V	5.5 V
	Up	1.13 ns	0.98 ns	0.85 ns	0.98 ns	0.80 ns	0.65 ns
	Down	0.24 ns	0.22 ns	0.17 ns	0.29 ns	0.28 ns	0.27 ns
Cb	Up	1.75 ns	1.39 ns	1.13 ns	1.95 ns	1.58 ns	1.30 ns
	Down	0.30 ns	0.29 ns	0.27 ns	0.28 ns	0.35 ns	0.30 ns

Simulation results verify that the delay of the circuit decreases as the power supply goes up.

3.4 Summary

From the simulation and discussions in this chapter, the complementary CMOS full adder is the choice for the design. The average for this full adder is approximately 0.8 ns using 0.8 um CK processing in SSI. The delay of the circuit decreases as the temperature goes down. The optimization of the sizes for the transistors in the circuit will increase the performance of the circuit, which can be achieved by some software simulation. The power supply variation affects the circuit performance: as the power supply goes up, the delay of the circuit drops. But with all the variations, the full adder used in the design can have a delay less than 1.2 ns, which is equivalent to 800 MHz.

Chapter 4. Design and Simulation of D Flip Flop

4.1 Sequential Circuits

Sequential circuits have memories so that the output signals can be function not only of the present input signals but also of past ones. Often, in sequential circuits, output signals are fed back. Thus, an output signal can be a function, not only of past input signals, but also of the past output signals. So a sequential circuit can be considered to consist of the interconnection of a combinational circuit and a memory.

Sequential circuits are so named because they allow operations to be performed in sequence. Sequential circuits are usually slower than combinational circuits because the operations have to be performed in sequence. However, the modern large digital computer, or even most small computer applications must have memories to function properly. Thus, sequential circuits are of prime importance in modern digital devices.

Sequential circuits are classified into two types, synchronous and asynchronous. In synchronous sequential circuits, the signals only change their values at discrete times, that is, they all change in synchronism. Pulses are generated by a device called a master clock. The master clock pulses synchronize the operation of all the devices within the digital device. In general, different types of digital circuits respond at different rates. The rate at which the master clock generates pulses must be slow enough to permit the slower circuit to respond. This then limits the speed of all circuits.

In an asynchronous sequential circuit, each device responds at its own states. Therefore, in general, asynchronous circuits are considerably faster than sequential ones.

The system designed here is a synchronous sequential circuit with the input speed at 156MHz, while the speed for the overall circuit is 20 MHz.

4.2 Flip-Flops

A very basic sequential circuit is called a flip-flop or latch. This is a digital device whose output remains constant (i.e., either a 0 or 1) until it is switched in response to its input signal. This accounts for its name: that is, it flips or flops from one possible output to the other and remains there until flipped back, or, equivalently, latches at one output until changed by the input.

There are different kinds of flip-flops. A general block diagram representation is shown in Figure 4.1.

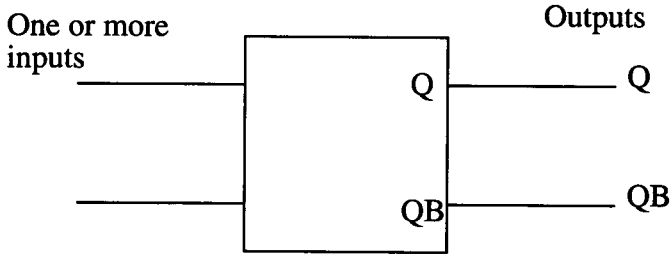


Figure 4.1 Block Diagram for a Flip-Flop.

The value of the output marked Q is called the state of the flip-flop. That is, when Q=1, then the state is 1 and when Q=0, the state is 0. Flip-flops usually have two outputs, one equals the state and the other equals its complement. The complement output is marked as QB.

Once the state of a flip-flop is set, it remains this way until changed by the input. Thus, a flip-flop "remembers" its inputs.

4.3 D Flip-Flop

The D (delay) flip-flop has only one input. The levels of a clock, CLK, are used to drive the D flip-flop (DFF) to either the storage state or the input state. If D is the input signal, Q and Q' are the CLK and $\overline{CLK}$, the state equations for positive and negative level-sensitive latch can be expressed as

$$Q' = D \cdot CLK + Q \cdot \overline{CLK}$$

$$\text{and} \quad Q' = D \cdot \overline{CLK} + Q \cdot CLK$$

The first equation describes a latch which passes the input data when $CLK = 1$ and stores it when $CLK = 0$. Inversely, the second equation describes a complementary latch, which receives input data at $CLK = 0$ and stores it at $CLK = 1$.

If two complementary latches are connected in series, one will be in the storage state while other is in the input state and a 'non-transparent' edge triggered flip-flop is formed. One well-known structure is called the master-slave D flip flop. The master stage reads in the input state when the clock is low (high), while the slave stage passes the input state to the output when the clock is high (low).

One structure that makes use of the transmission gate is shown in Figure 4.2. As can be seen from the circuit, the master and slave stage will remain isolated as long as

the clk1 and clk2 are not high simultaneously. The negative feedback loop keeps the state of the master and slave stage latched as long as the clock does not have another up-edge or down-edge. At the up-edge of the clock, the master stage reads the input, while the slave stage passes the input state to the output at the down-edge of the clock. So this kind of flip-flop is called edge-trigger DFF.

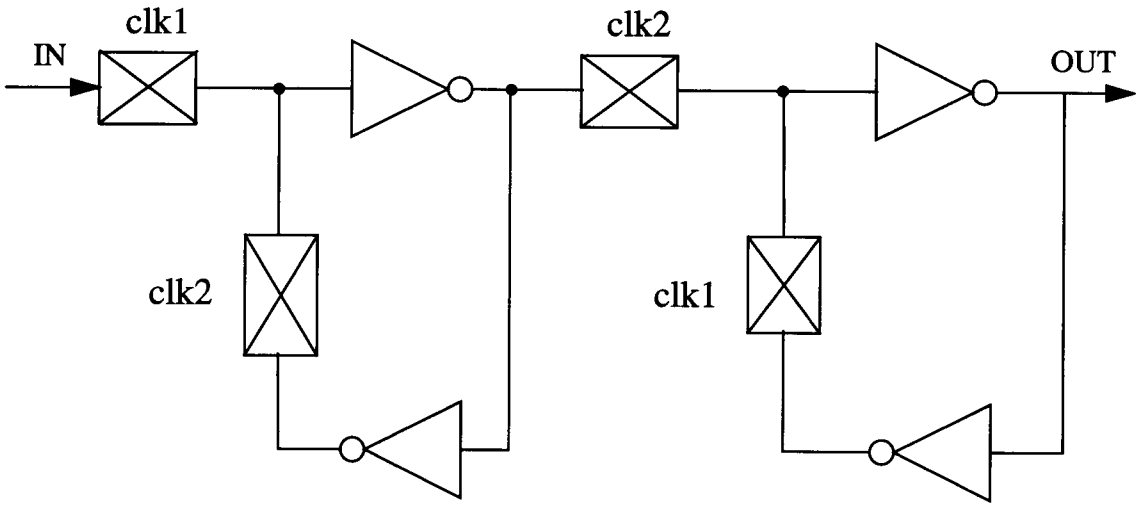


Figure 4.2 Static CMOS D FLIP FLOP

4.4 CMOS Static DFF Simulation

4.4.1 The Comparison of Three Kinds of Transmission Gate DFF

The transmission gate in the above structure can be a single NMOS or single PMOS transistor or CMOS transmission gate. Due to the working characteristics of these three kinds of gates, the working performance and the highest working frequency for them will be very different.

In order to find out the working characteristics for NMOS, PMOS and CMOS transmission, the following setups are used.

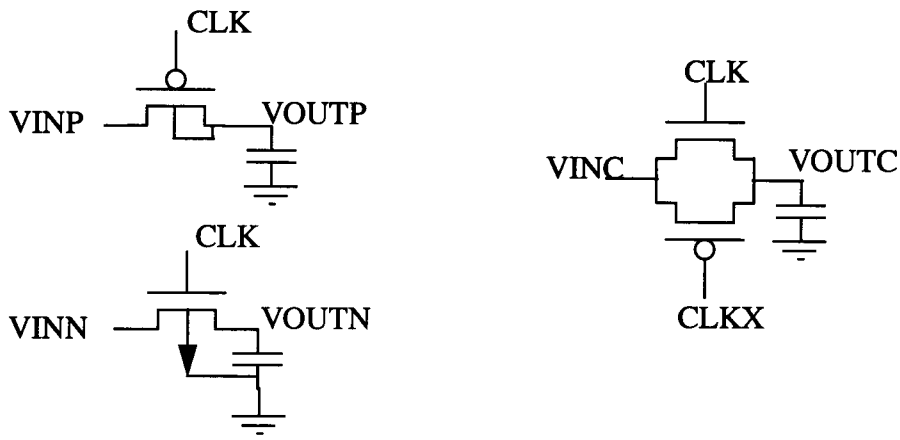


Figure 4.3 Test Circuit for Different Kinds of Transmission Gates

For NMOS transmission gate, when the $CLK = 0$, the gate is open, the VOUTN keeps its old state. When $CLK = VDD$, the gate is closed, the stable voltage for VOUTN depends on the voltage level of VINN. If $VINN = 0$, the resulting $VOUTN = 0$; if $VINN = VDD$, the resulting $VOUTN = VDD - V_T$. Most of the processing are n-well processing, so the bulk node for NMOS can only tied to 0v, which means the $V_{BS} = VDD - V_t = 4.3v$, this V_{BS} will form a negative feedback on the VINN (body-effect). $V_t = V_{t0} + 1/2 \cdot (\sqrt{V_{bs} + 2 \cdot \phi_s} - \sqrt{2 \cdot \phi_s})$, where $V_{t0} = 0.7v$, $\phi_s = 0.3v$. So the resulting $V_t = 1.5v$, this will pull the VINN=3.5v. So the NMOS transmission gate can only transfer voltage between 0 to 3.5v. If this kind of transmission gate is used in the circuit, the noise margin for the next stage will be largely reduced. An even worse thing is that if these kinds of transmission gates were

used in series, this body-effect would become worse and worse along the transmission gate chain, and ultimately kill the initial signal.

For PMOS transmission gate, this body-effect can be eliminated by connecting the bulk node of PMOS with its source node. This is because most of the processing are n-well process, so the bulk node for each PMOS can be tied to its own voltage instead of tied together to certain voltage. So the PMOS transmission gate has better voltage performance than NMOS.

The second factor that must be taken into consideration when choosing the transmission gate is to find out the highest working frequencies for them. In order to do this, different kinds of clocks are put into the DFF's composed by each of these three kinds of transmission gates, increase the clock frequency gradually until it fails, while the frequency ratio of the clock to input signal is kept at 2:1.

The result of simulation is summarized in Table 4.1

Table 4.1 Speeds of DFF With Different Transmission Gate

Gate Type	CMOS	NMOS	PMOS
Highest Frequency	358M	357M	100M

The simuation shows that only the CMOS and NMOS transmission gates have the required working frequency. Considering the voltage level problem of the NMOS transmission gate, the CMOS transmission gate is the best choice for the design.

4.4.2 Working Performances with Different Sizes of Transistors

The working performance of the circuit is largely affected by the sizes of the transistors. In CMOS static circuits, a larger size transistor gives a larger current under the same gate voltages, which means stronger driving ability. But a large size transistor also means large load capacitance for the previous gate. Simulation is done with the design using minimized size transistors. Based on the waveform of the internal nodes, we increase the sizes of the driver transistors for the slower nodes, while keeping the faster node driver transistor at the minimum size. The optimum sized DFF can work up to 516MHz, while the minimum sized DFF can work up to 356 MHz.

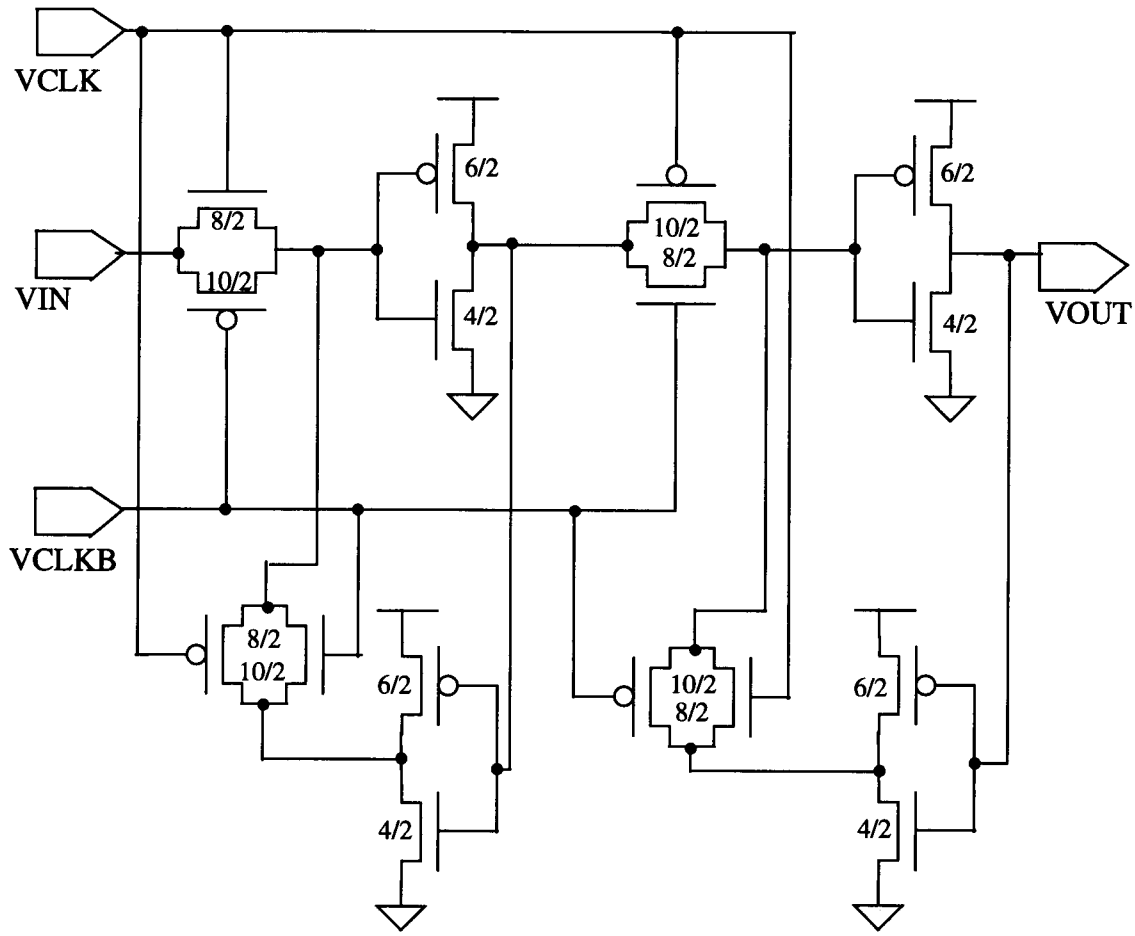


Figure 4.4 CMOS Static DFF with Improved Size Transistors

4.4.3 Timing Simulation About DFF

When using DFF, it is necessary to understand the timing of the various signals used in DFF. This is especially true when different components are interconnected.

If gates are to function properly, then the clock signal must meet certain requirements. There are two such important specs about DFF, the *setup time* and the *hold time*.

The setup and hold time of a register are the deviation from an ideal register caused by finite circuit delay.

The setup time is the delay between the 50% whole voltage point of the incoming signal and the 50% whole voltage point of reading edge of the clock, i.e. the input must be set up at the 'setup time' before the reading clock edge.

The hold time is the delay between the 50% whole voltage point of the reading edge of the clock and the 50% voltage point of the changing edge of the input, in order to get the input read in the DFF correctly.

From the view of charge and discharge, setup time is the time to chargeup (or discharge) the input node to the right input state. If the input driver is an ideal voltage source, there should be no time needed to charge (or discharge) that node up. But in real circuits, this is the time for the driver which drives the DFF to charge (or discharge) it output node to the right state before the reading edge of the clock for the DFF. Hold time is the time to charge (or discharge) the internal node of the DFF to the right state, so it is

determined by the charge (or discharge) current and the parasitic capacitance of the internal nodes within the DFF.

If the data of a register does not obey the setup and hold time constraints, a potential clock race problem may occur. This race results in erroneous data being stored in the register.

In order to simulate the setup time for the DFF, the input and clock in Figure 4.5 are used:

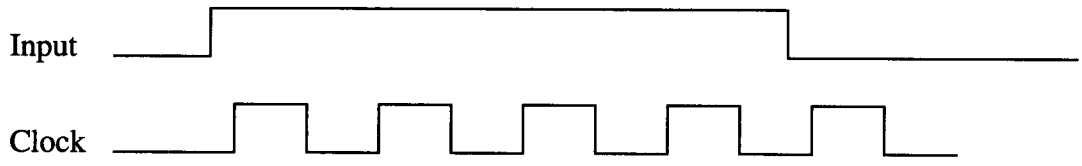


Figure 4.5 DFF Clock & Input Relation

The simulation is done by moving the up-edge of the input step-by-step towards the up-edge of the clock, until the operation fails.

The simulation shows the setup time for the circuit is 0ns for both T=25$^{\circ}C$ and T=-220$^{\circ}C$, assuming that the node capacitance at the input point is 50 fF.

In order to get the hold time for the circuit, the down-edge of the input is pushed backwards to the up-edge until the circuit fails to read in the right input.

Simulation gives a hold time to be 2.3ns for T=25$^\circ C$ and 2.4ns for T=-220$^\circ C$, assuming the output node load is 50 fF.

4.5 Summary

This design uses CMOS static DFF. It has 0ns setup time and 2.3ns hold time. The DFF with PMOS transmission gate has a much lower speed than that with NMOS or CMOS transmission gate. The DFF with NMOS and CMOS can work up to 350MHz using 1.1 micro CMOS process, but the NMOS transmission gate has a very serious body-effect that can't be eliminated in n-well processing. So the CMOS transmission gate is chosen for the design. The size of the transistors in the DFF affects the performance of the circuit. But the optimum size isn't available due the limitation of tools.

Consider a digital system containing many master-slave flip-flops, with the outputs of some flip-flops going to the inputs of other flip-flops. Assume that the clock pulse inputs to all flip-flops at the same time. At the beginning of each clock pulse, some of the master elements change states, but all the flip-flop outputs remain at their previous values. After the clock pulse returns to 0, some of the outputs change state, but none of these new states has any affect on any of the master elements until the next clock pulse. Thus the states of flip-flops in the system can be changed simultaneously during the same clock pulse, even though outputs of flip-flops are connected to the inputs of flip-flops. This is possible because the new states appear at the output terminals only after the clock pulse has returned to 0. Therefore, the binary content of the second is transferred to the first, and both transfers can occur during the same clock pulse.

Chapter 5. Function Blocks

The system is composed of the following function blocks: Shift Register, Latch Register, Wallace Tree and Carry-look-ahead & Carry-select Adder.

5.1 Shift Register and Latch Register

The first block for the system is 1024 shift registers. This block latches the 64 unweighted bits in parallel for 8 times. So it comes as the matrix of 64×8 form. The inputs of the DFF work at 156MHz.

The block that follows the shift register block is the latch register block. This block takes the 8 rows of 64 register output, converts them to 1024 bits in parallel. The input frequency for this block is 20MHz.

The schematic of this block is shown on Figure 5.1.

5.2 Wallace Tree

When three or more operands are to be added together, the speed of the traditional carry-ripple adder is restricted by the carry ripples between bits. When the number of bits is large, the traditional carry-ripple adder becomes so slow that it can not be used in a speed sensitive system. Several techniques for these kinds of multiple operand addition that attempt to lower the carry-propagation penalty have been proposed and implemented. The technique that is most commonly used is carry-save addition. In a carry-save adder (CSA), the carry propagation is only allowed in the last step, while in all the other steps a partial sum and a sequence of carries are generated.

Therefore, a CSA is capable of reducing the number of operands to be added from 3 to 2, without any carry propagation.

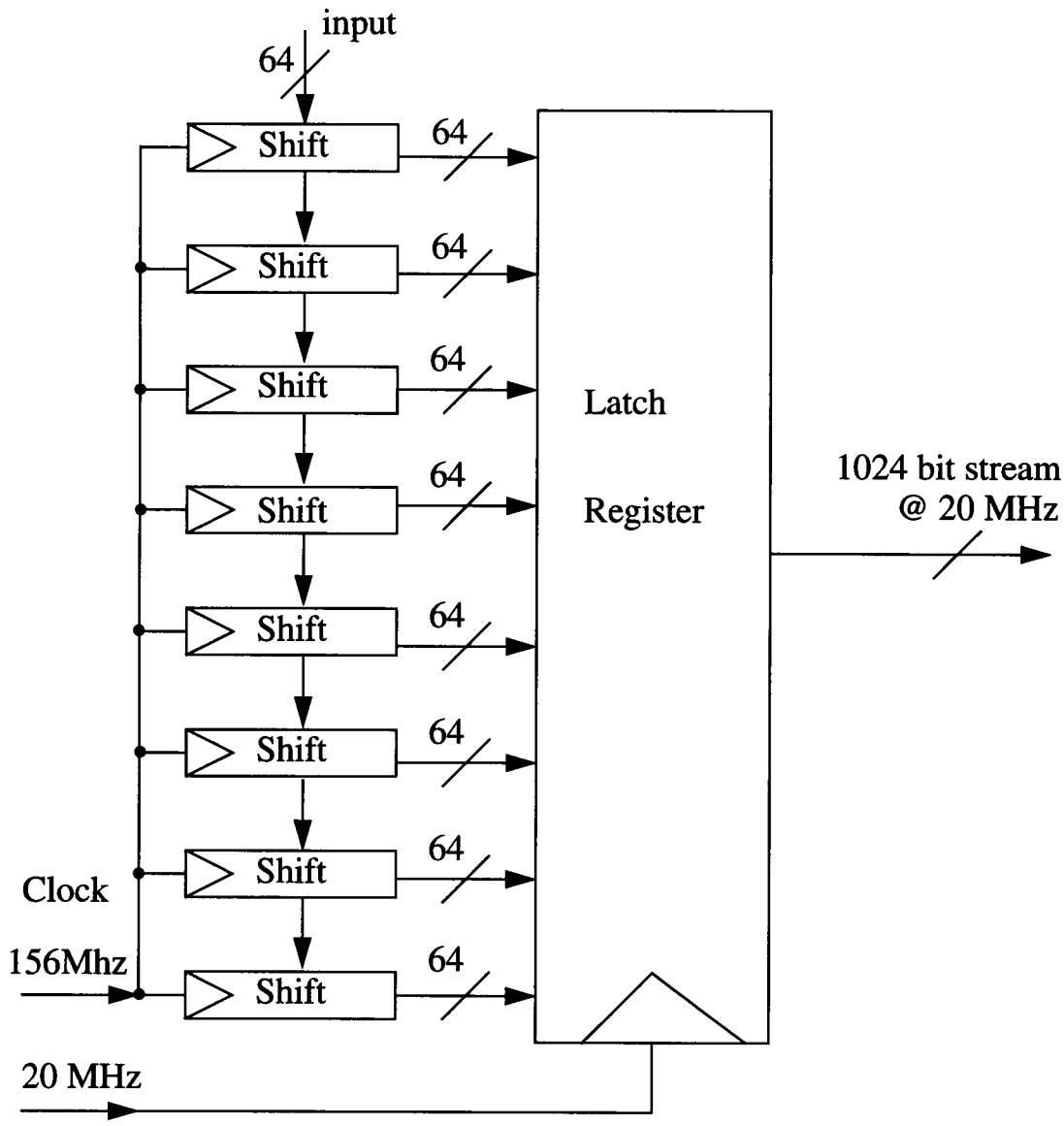


Figure 5.1 Shift Register and Latch Register Block

5.2.1 Carry-Save Adders

A carry-save adder can be implemented in several different ways. In the simplest implementation, the basic element of the carry-save adder is a full adder with three inputs, x, y, and z, whose arithmetic operation can be described by:

$$x \cdot 2^i + y \cdot 2^i + z \cdot 2^i = c \cdot 2^{i+1} + s \cdot 2^i$$

where

x, y and z are the inputs of CSA with 2^i weight

c and s are the outputs of CSA with 2^{i+1} and 2^i weight

It is obvious that the full adder is a (3,2) CSA or counter. Besides (3,2) counter, there are (7,3) and (15,4),... counters. CSA can also be put in another way, that is CSA counts the number of ones in the same weighted multi input and output the counted number in binary data form.

Figure 5.2 gives some real number examples for these kinds of CSA.

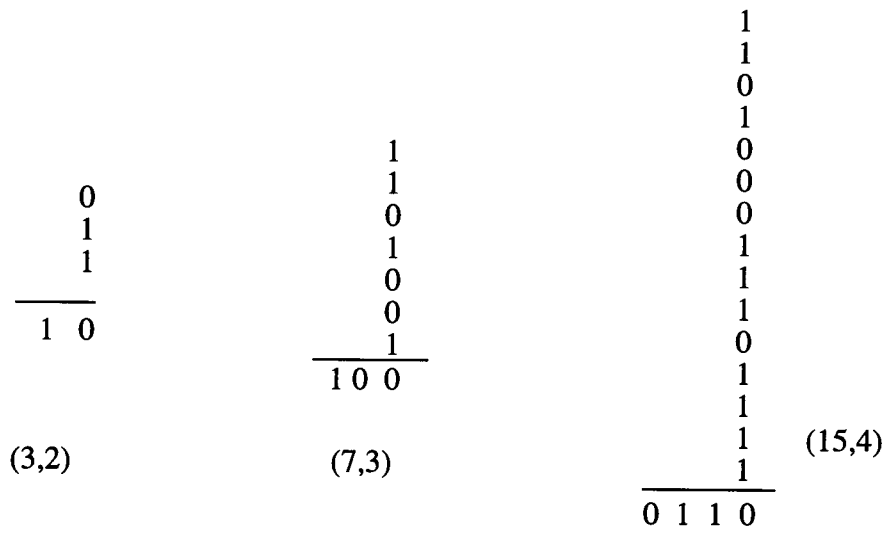


Figure 5.2 Examples of Single Bit CSA

The structures for these basic counters are shown in Figure 5.3. The real implementations for these counters are shown in Figure 5.4 and Figure 5.5.

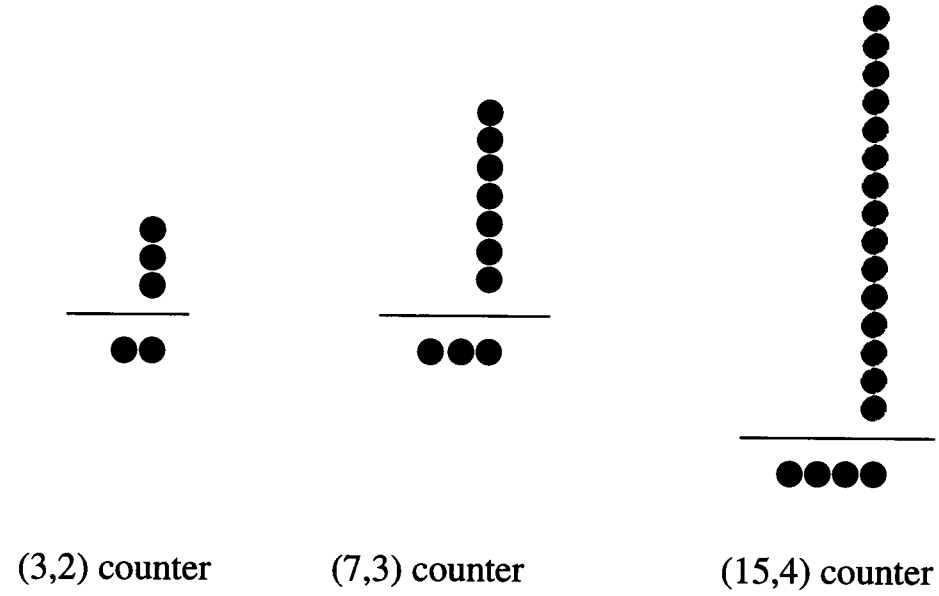


Figure 5.3 Counters

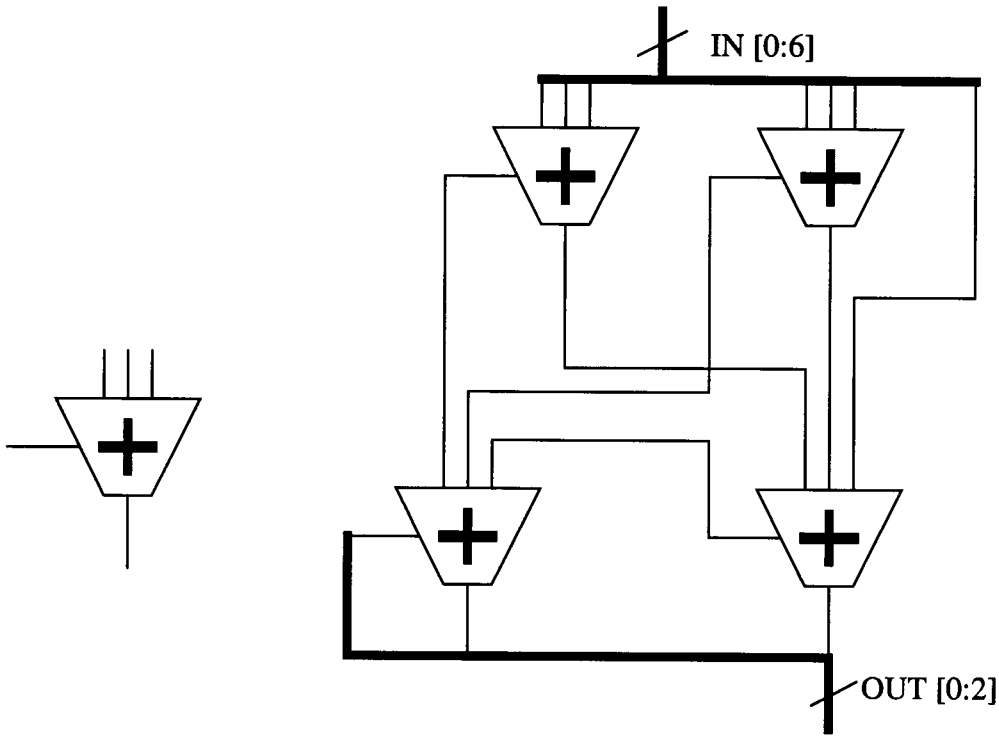
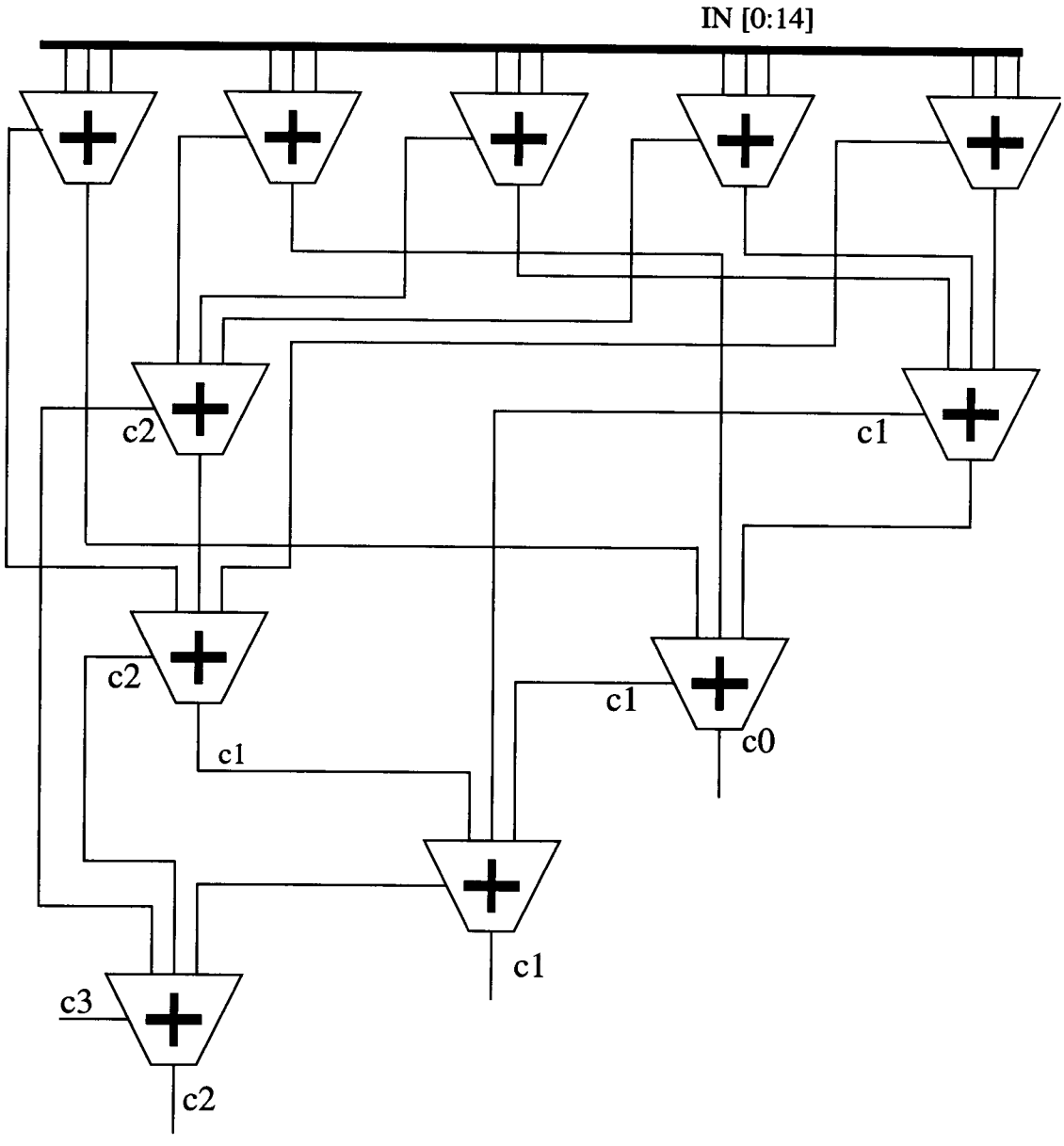


Figure 5.4 (3,2) and (7,3) Counter Structures



C i: ith weight
Delay: Five FA Delays

Figure 5.5 (15,4) Counter Structure

5.2.2 Wallace Tree

If the operands for the CSA or counters are multi-bit words, a way to organize the operations is a tree commonly called Wallace Tree. Similar to the single bit CSA, the multibit CSA counts the number of ones in the same weight bits separately without caring about the carry ripple from the lower weighted bits. Figure 5.6 is an example of Wallace Tree dealing with real multi-bit numbers.

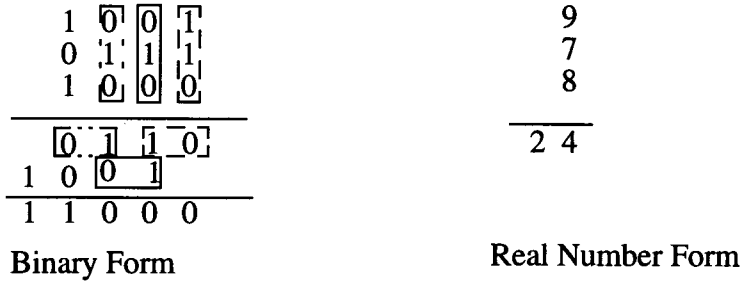


Figure 5.6 Example of Multi-Bit Wallace Tree

Figure 5.7 gives a three 20-bit Wallace Tree, three 20-bit number can be reduced to two 21-bit numbers in one full-adder delay. Figure 5.8 shows a 20-bit (7,3) Wallace Tree, seven 20-bit numbers can be reduced to two 22-bit numbers in four full-adder delays, Figure 5.9 shows a 20-bit (15,4) Wallace Tree, fifteen 20-bit numbers can be reduced to two 22-bit numbers in seven full-adder delays.

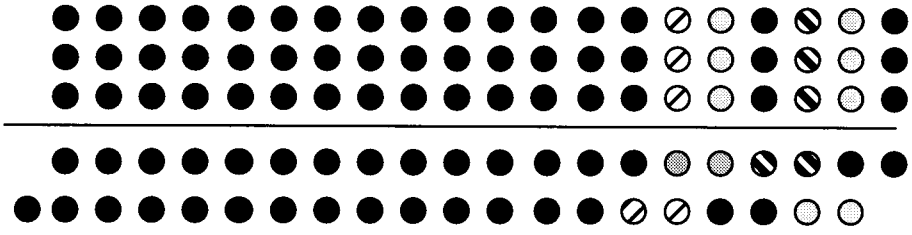
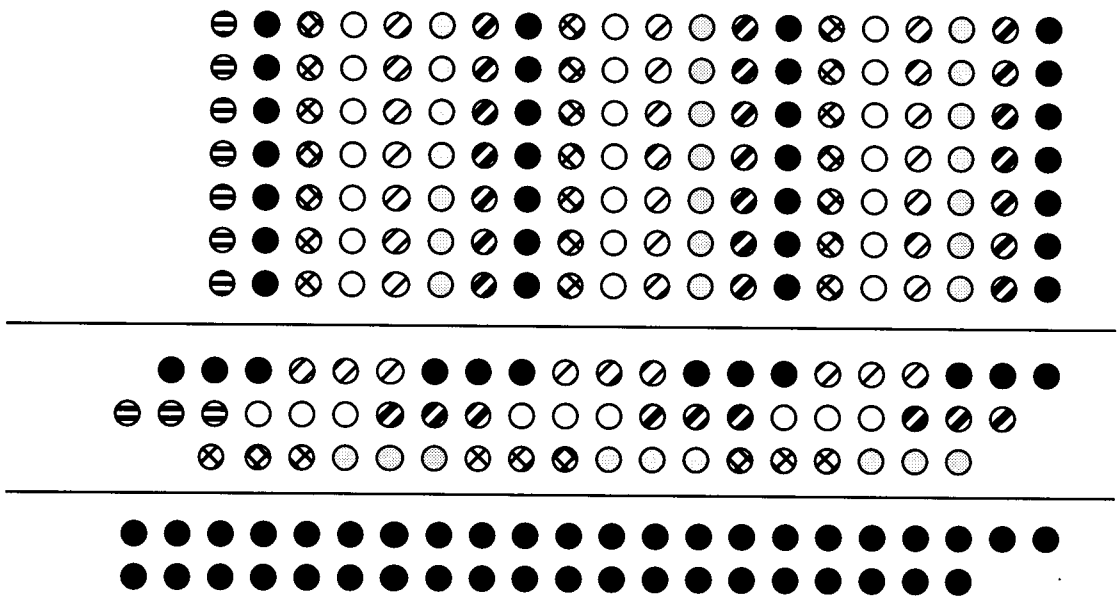


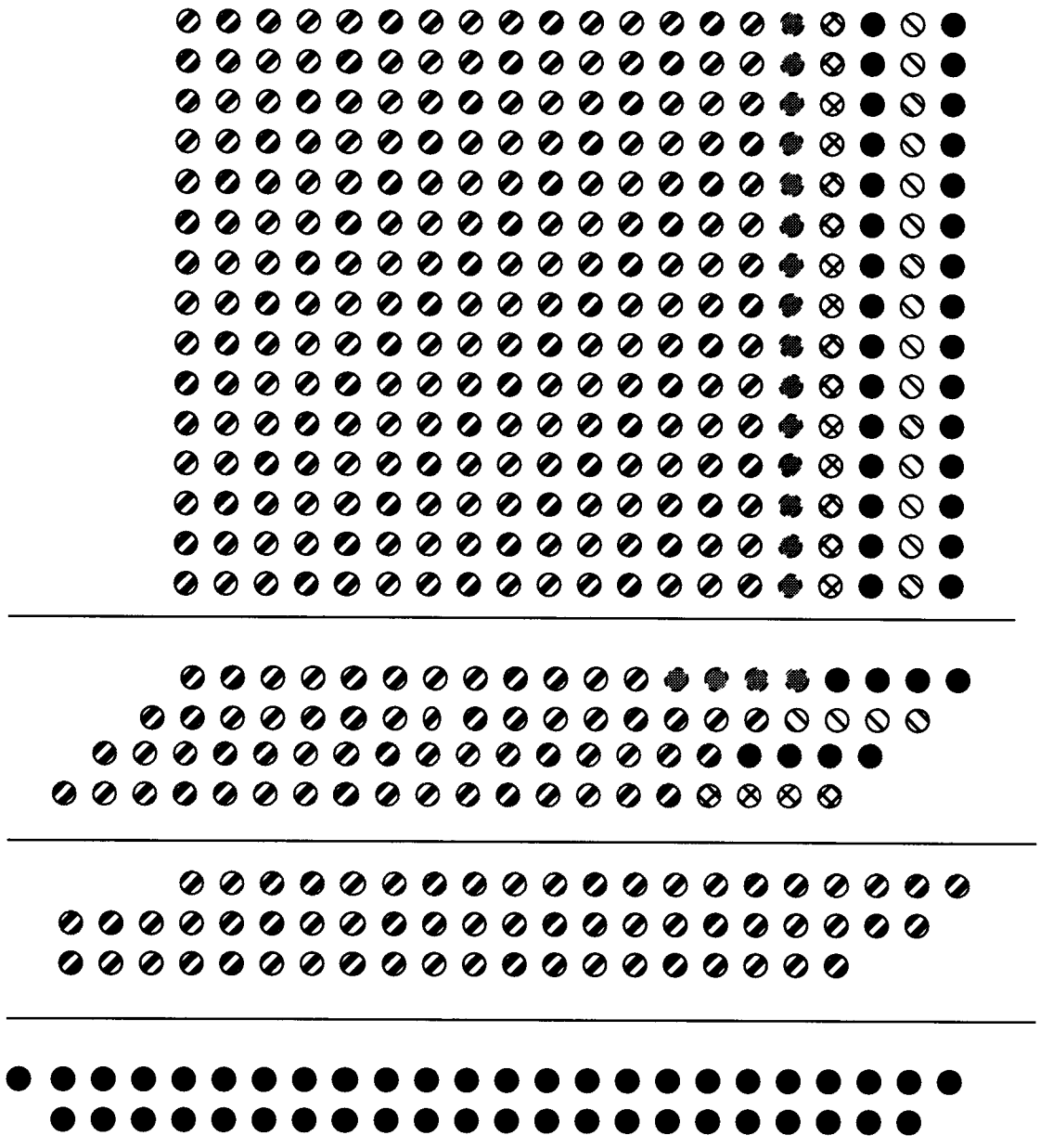
Figure 5.7 Three 20_bit Wallace Tree Structure



Seven 2-bit words ► One 20-bit word and one 22-bit word

Total delay is four full adder delays

Figure 5.8 Seven 20-bit Wallace Tree Structure



Fifteen 20-bit Words ⟶ Two 24-bit Words

Total delay: 5+1+1=7 Seven Full Adder Delays

Figure 5.9 Fifteen 20-bit Wallace Tree Structure

In Wallace Tree, the number of operands is reduced by a factor of 2/3 at each level if the basic (3,2) counter is used. Consequently,

$$\text{Number of levels} \approx \frac{\log(k/2)}{\log(3/2)}$$

This equation only provides an estimation of the number of levels. If different kind of basic counters is used, the level number may vary.

5.2.3 Sign-Bit in Wallace Tree

Wallace Tree is very efficient to reduce the number of additions. The basic element for this structure is the counter, which causes some problems. As indicated by the name, counters only count the number of ones in the inputs, so they won't take care of the sign bits.

So if the counters are used to deal with signed numbers, some pre-separation has to be done to separate the positive and negative numbers. Then the counter can be used to deal with each group separately.

The results are four numbers, two of them are positive, two of them are negative. These numbers should be reunited with their sign bit information to recover the right numbers.

For this design, the inputs of the Wallace Tree are signed binary numbers. For simplicity, it is assumed that the inputs are evenly positive and negative. So the Wallace Tree is divided evenly into two parts, positive part and negative part. But this is seldom the case. The worst case is all the numbers are positive or all are negative.

The following adjustments can be done in order to make the circuit work in that case:

1. Use XOR at each of the inputs to decide which block, positive or negative block, the input should be sent to. Meanwhile send a zeros to the non-chosen block.

2. Duplicate the structure in both positive and negative blocks to make sure the circuit in each block can deal with 512 numbers instead of 256 ones.

Due to the duplication, the Wallace Tree will need two more three 20-bit to two 21-bit layers at the end of the positive and negative blocks.

So Wallace Tree can be used for the addition of signed numbers at the price of larger hardware implementation.

5.2.4 Structure Trade-Off for Wallace Tree

When parallel additions are done, there are several basic circuit elements to choose from, (3,2), (7,3), (15,4), etc. How to choose the basic element is a trade-off between the simplicity of the circuit and the total time required to finish the calculation. As shown in Figure 5.10, if the (3,2) counter is used instead of the (15,4) counter as the basic element, the total delay is only 6 full adder delays instead of 7 full adder delays. But as also seen from the two structures, using (15,4) as the basic element makes the structure simpler and easier to manage.

In this design, two kinds of basic elements are used. (15,4) is used in the top level of the Wallace Tree to reduce the large number of inputs, while (3,2) is used for the following levels when the number of additions is reduced to a more manageable number.

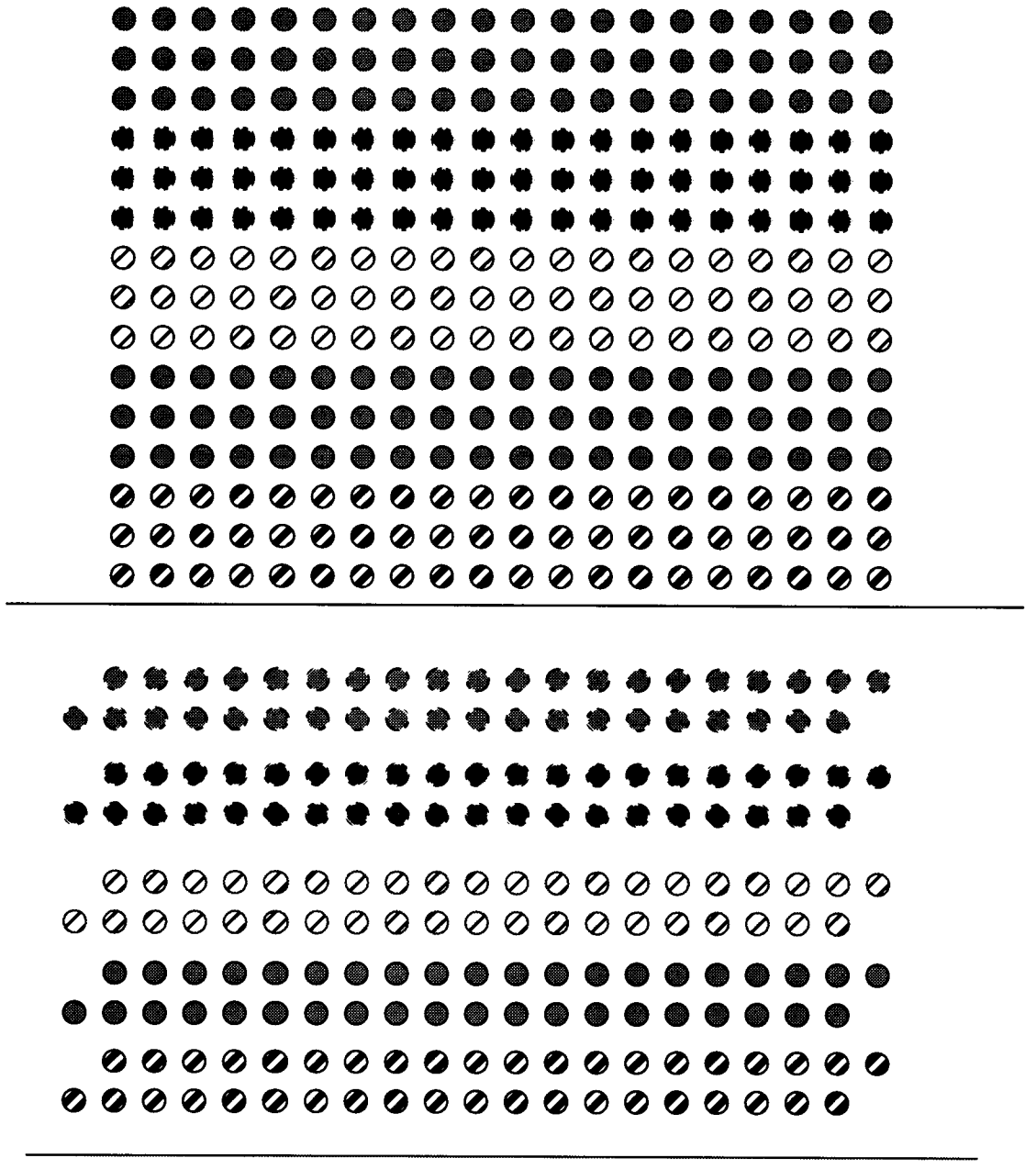
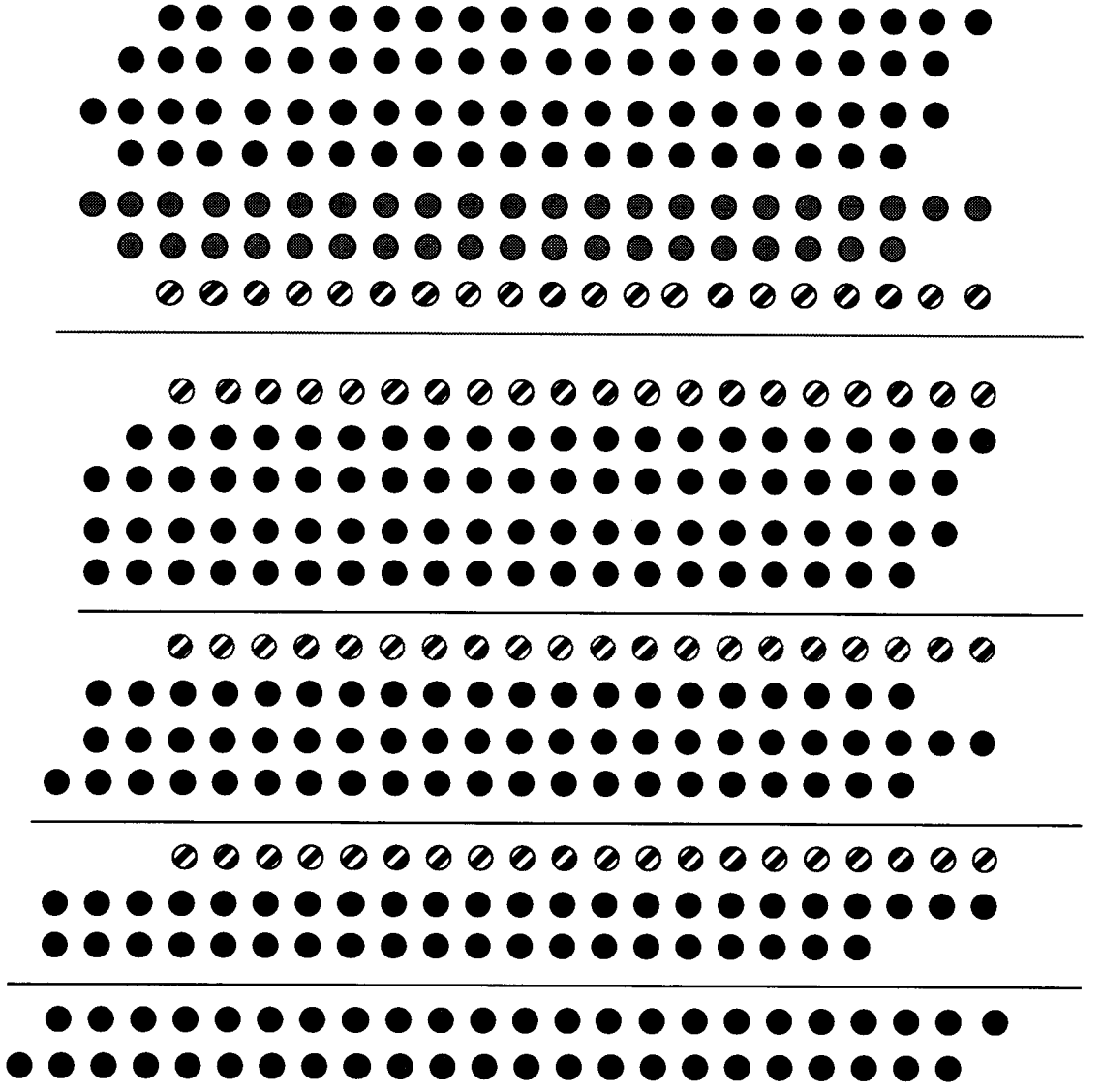


Figure 5.10 Fifteen 20-bit Wallace Tree with (3,2) to be the Basic Element



Fifteen 20-bit words ⟶ One 21-bit word and one 22-bit words

Total delay is six full adder delays

Figure 5.10 Continued

5.3 Carry-Look-Ahead and Carry-Select Adder

The outputs of the Wallace Tree are two positive 28-bit numbers and two negative 28-bit numbers. In order to add them to one 30-bit number, a 4-bit carry look-ahead and carry-select adder is used.

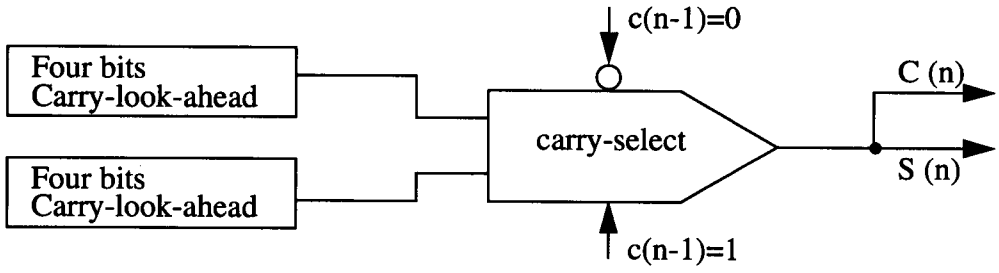


Figure 5.11 Carry-Look-Ahead & Carry-Select Adder

In Figure 5.11, one of the 'Four bits Carry_look_ahead' does the addition assuming the carry input from the previous bit is 0, the other one does the addition assuming the carry input from the previous bit is 1, then the carry-select selects the right output when the previous carry reaches this block.

5.4 Summary

The input part of the system: shift-register and the latch-register blocks, converts the inputs to the data form that can be dealt by the system. Wallace Tree does the large amount of additions for the system, with the help of the pre-processing part to deal with the sign bit. The cost of this may be twice the hardwares for the realization. Traditional adder is still needed at the end of the Wallace Tree to get the final single multi-bit word.

Chapter 6. Conclusion

6.1 Conclusion

From the design and the simulation, 1024 data points enter the designed FIR filter in the form of 64 bits in parallel at 156 MHz, and the system transfers the information into one single 30-bit word at 20MHz.

Since the data flow for the system is all pipelined, the highest speed the system can attain is 1/(one FA delay), which is about 300MHz.

There are about 2100 DFF, 1024 XOR, 1024 AND, 13000 FA and 32 4-bit carry-look-ahead adder. The estimated area for the gate layout is about 65 mm^2. If considering the pads and the routing area to be twice the area of the transistor area, the total die size will be around 100 mm^2.

6.2 Future Work

This design is based on the assumption that all the coefficients are un-signed, if the coefficients are signed binary numbers, the whole structure can be duplicated to do that.

This design uses some (7,3) (15,4) to do the Wallace Tree; the speed can be further increased by using (3,2) to do the Wallace Tree. This will require more detailed design of more circuit levels.

Based on the tree structure of Wallace Tree, the basic floorplan can take the square form, with the input at the outerside of the chip and have the output generated at the center of the chip. The chip area for the successive levels shrink in accordance with the shrinkage in the amount of data or adding for each levels.

Bibliography

1 West, Neil H. E. and Eshraghian, Kamran. 1992. *Principles of CMOS VLSI Design*, Addison-Wesley Publishing Company.

2 Waser, Shlomo and Flynn, Michael J. 1982. *Introduction to Arithmetic for Digital Systems Designers,* Holt, Rinehart and Winston.

3 Geiger, Randall L., Allen, Phillip E. and Strader, Noel R. 1990. *VLSI Design Techniques for Analog and Digital Circuits*, McGraw-Hill.

4 Neamen, Donald A. 1992. *Semiconductor Physics and Devices Basic Principles*, IRWIN.

5 Mano, M.Morris, 1992. *Digital Design*, Englewood Cliffs, N.J.:Prentice Hall

6 Proakis, John G. and Manolakis, Dimitris G., 1996. *Digital Signal Processing Principles, Algorithms and Applications,* Englewood Cliffs, N.J.:Prentice Hall.

7 Lu, Shih_Lien and Ercegovac, M., Aug. 1990. "A novel CMOS implementation of double-edge-triggered flip-flops," *IEEE J.Solid-State Circuit, vol.25,* pp.1008-1010.

8 Koren, Israel, 1993. *Computer Arithmetic Algorithms*, Englewood Cliffs, N.J.: Prentice Hall.

9 Chirlian, Paul M., 1987. *Analysis and Design of Integrated Electronic Circuits*, second edition, New York.: Harper & Row, Publishers.

10 Lu, Shih_Lien, ECE Department, OSU, *Comment on "A New Design of the CMOS Full Adder"*, to be published.

11 Swartzlander, Jr. Earle E., 1992. *Parallel Counters*, IEEE Computer Arithmetic, vol. 1, pp 90-93.

Appendices

Appendix A Hspice Simulation Results

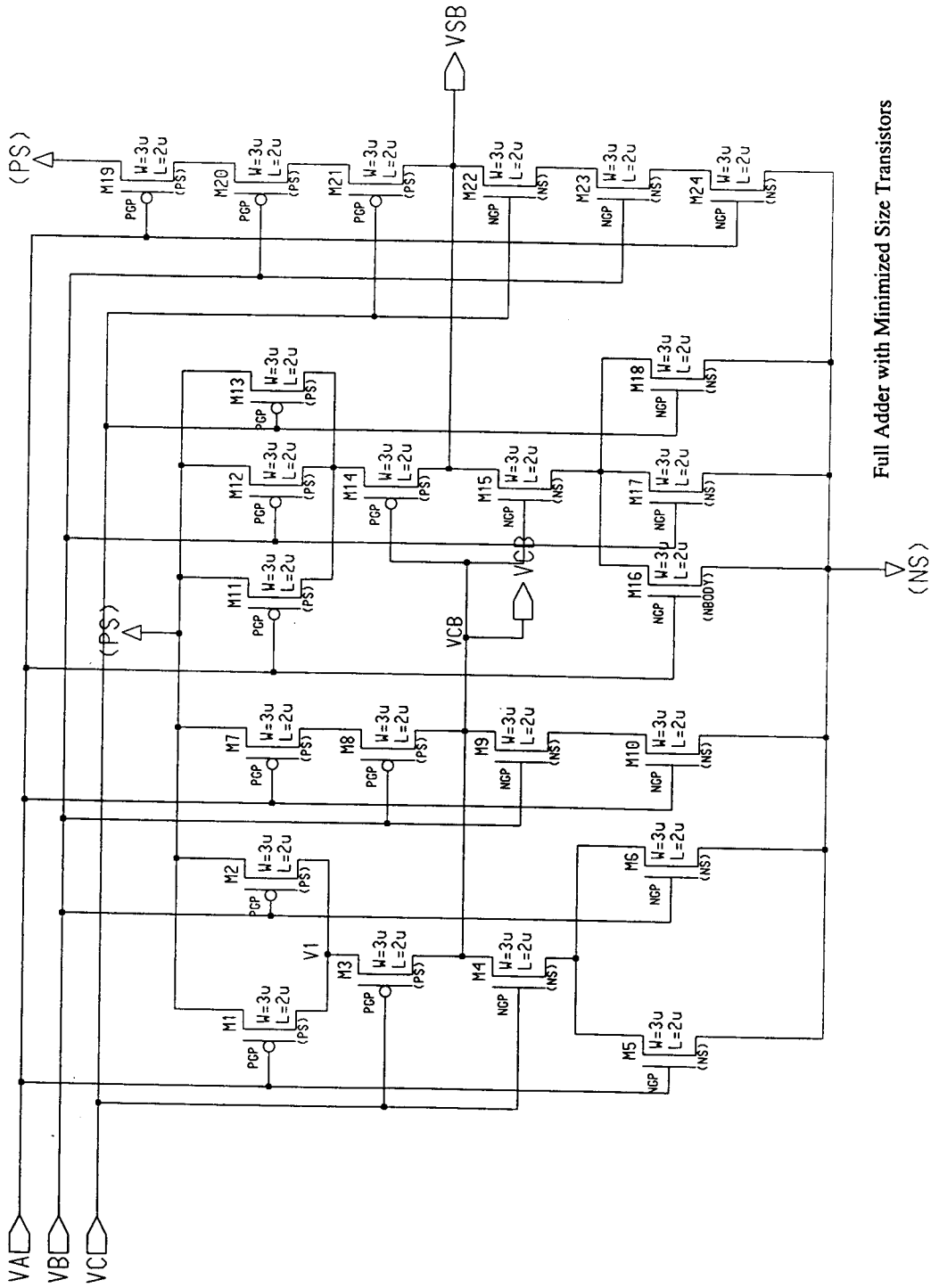


Figure A.1 Full Adder with Minimum Sized Transistors

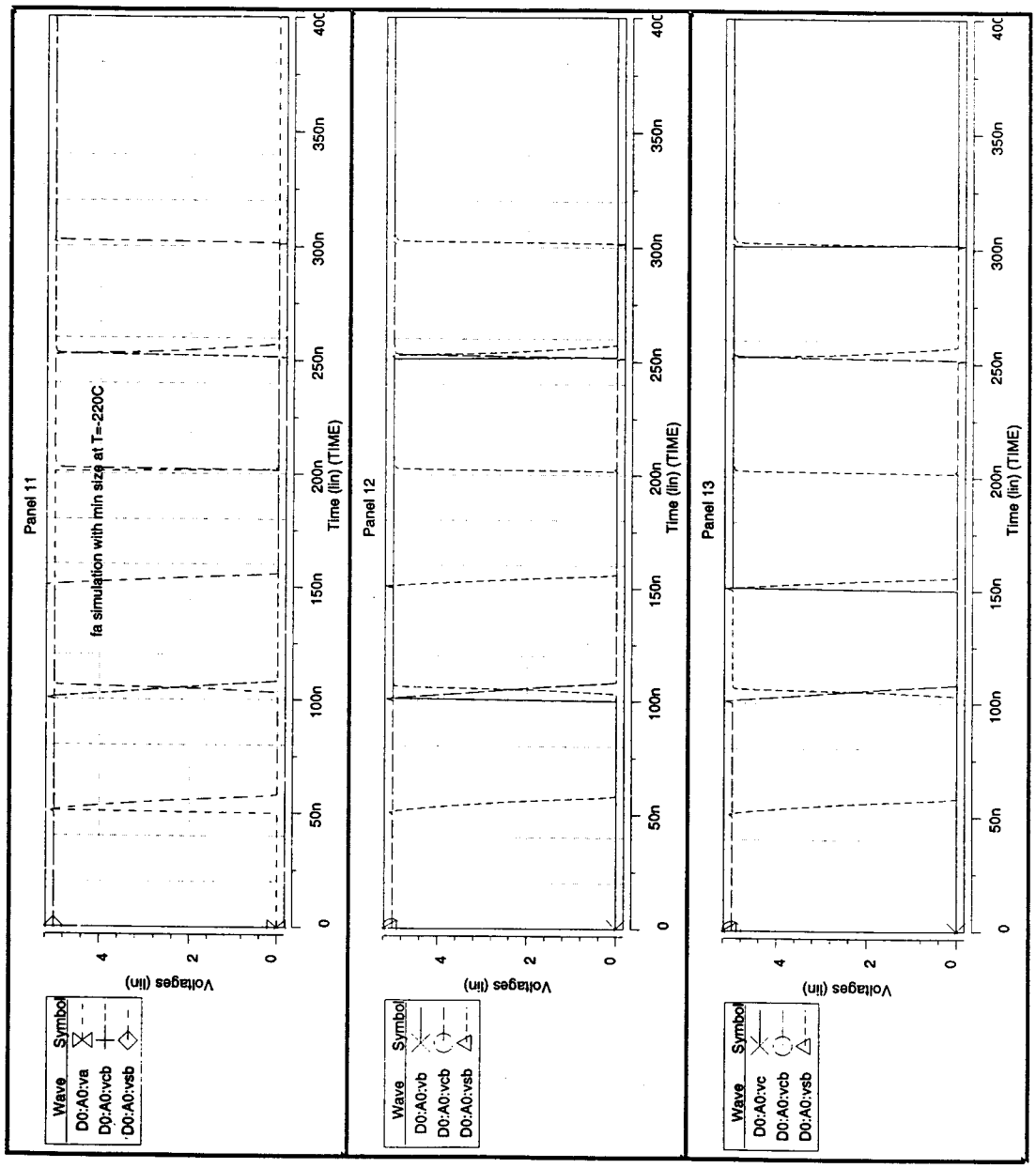


Figure A.2 Full Adder Simulation with Minimum Size Transistors (T=-220)

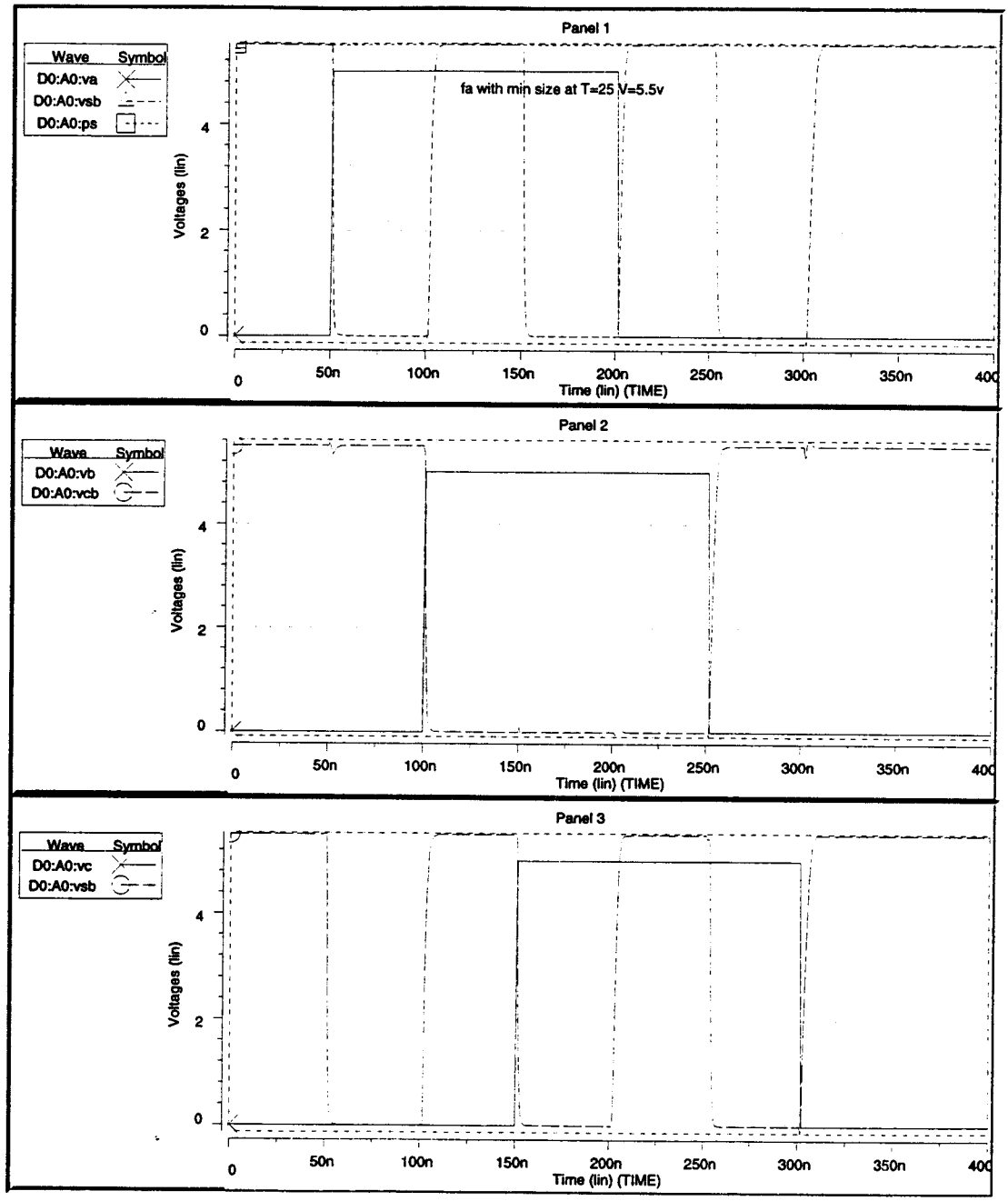


Figure A.3 Full Adder Simulation with Minimum Sized Transistors(25C, 5.5V)

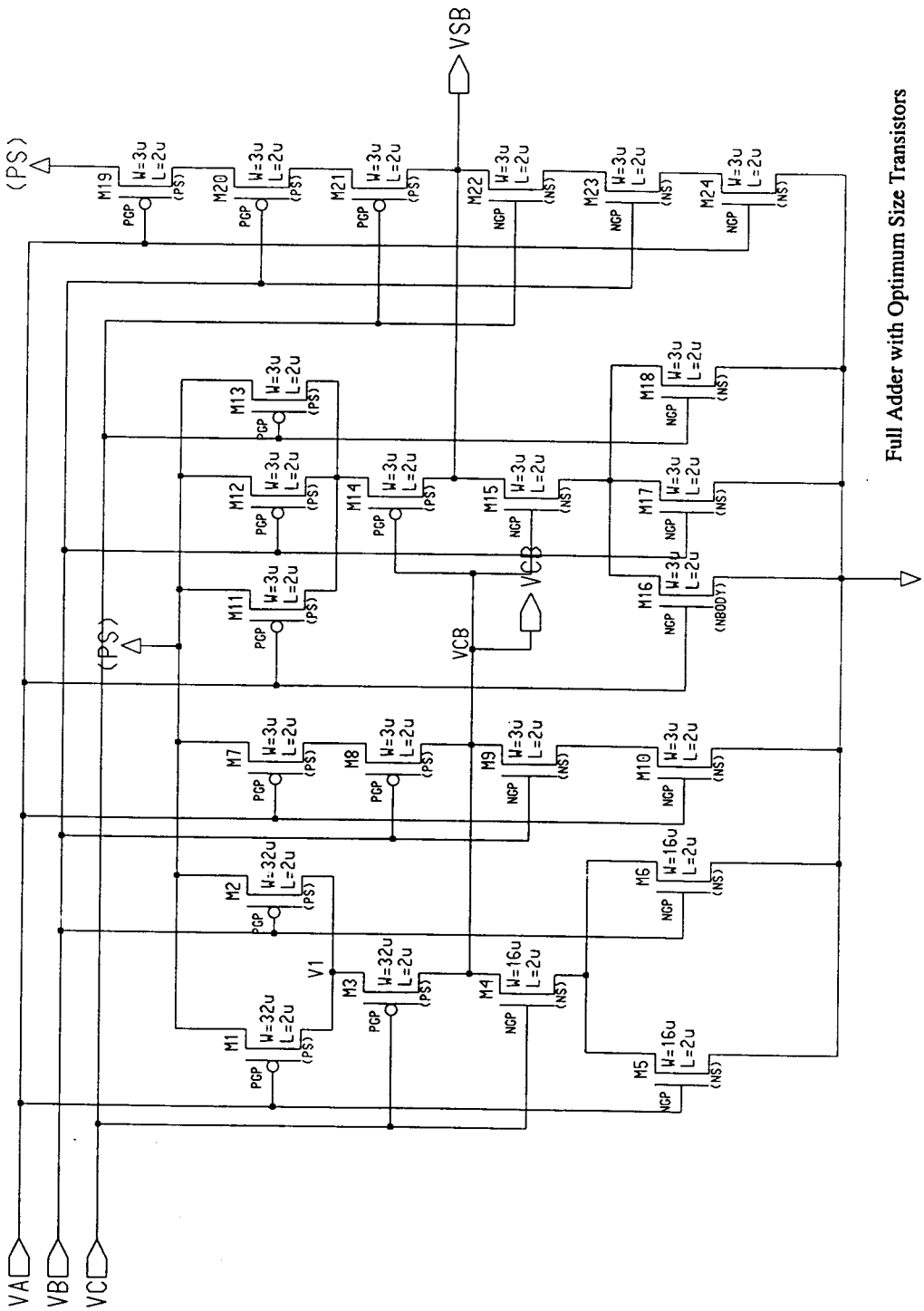


Figure A.4 Full Adder with Optimum Sized Transistors

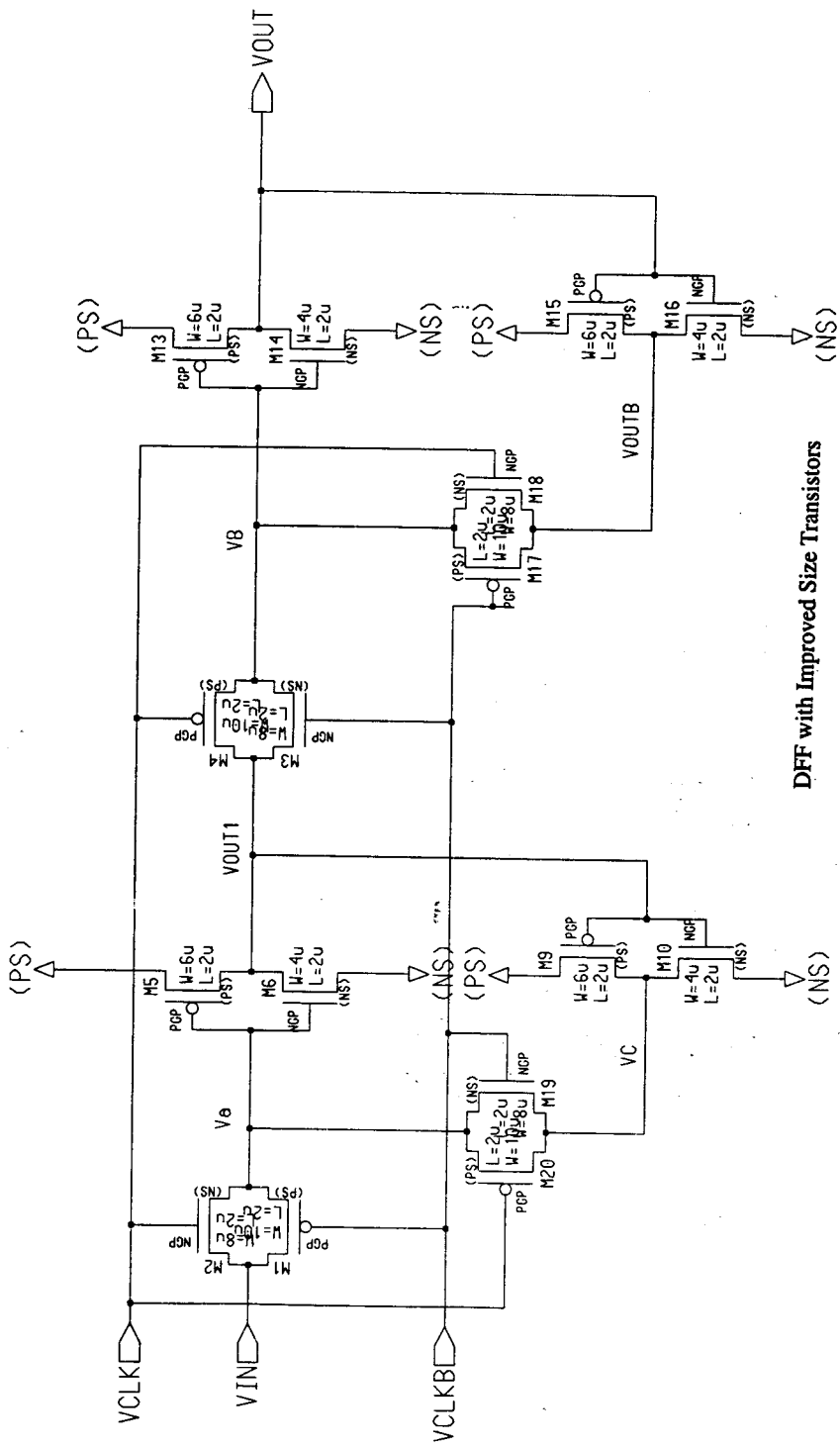


Figure A.4 Full Adder with Optimum Sized Transistors

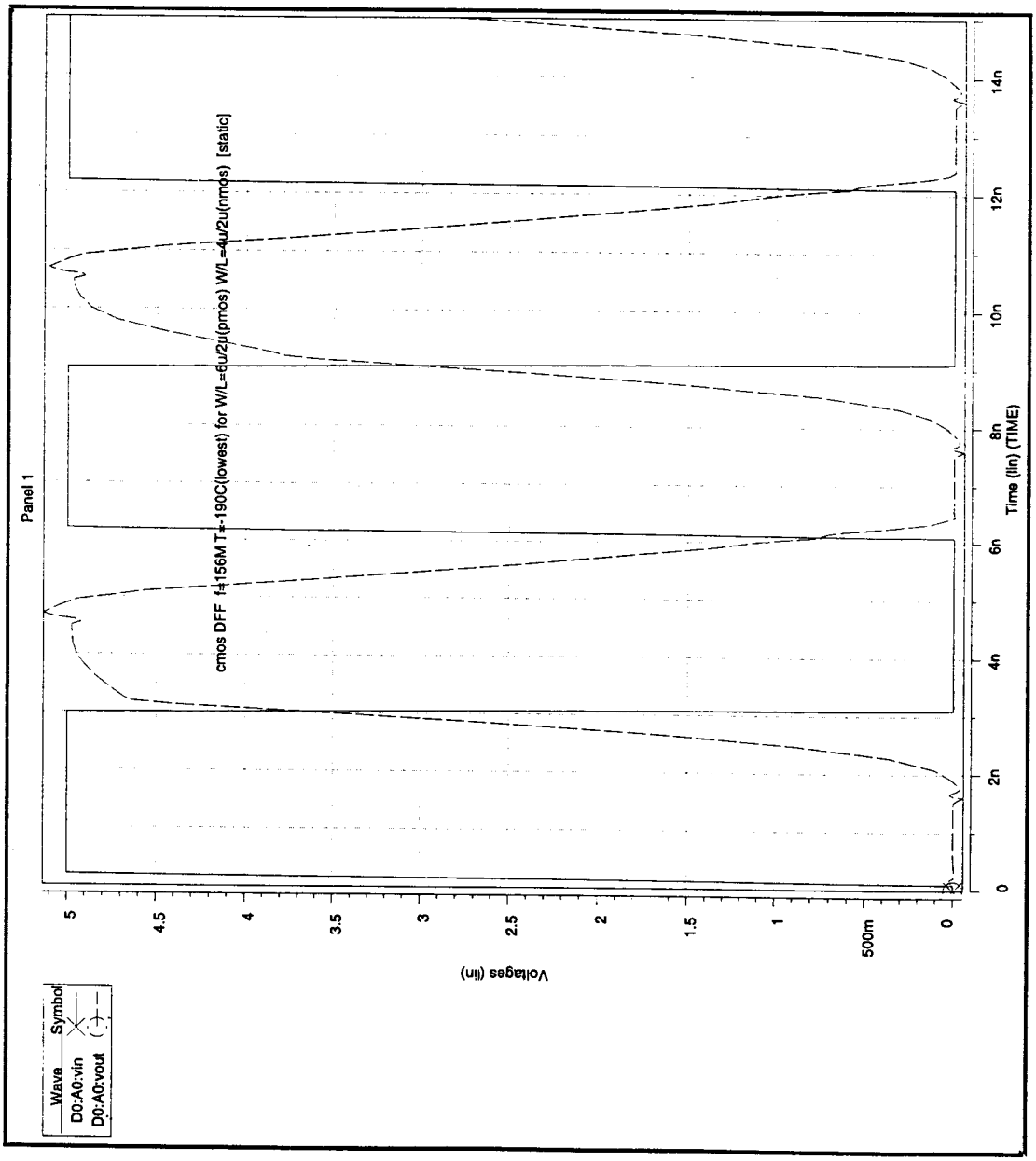


Figure A.6 Static DFF Simulation (1)

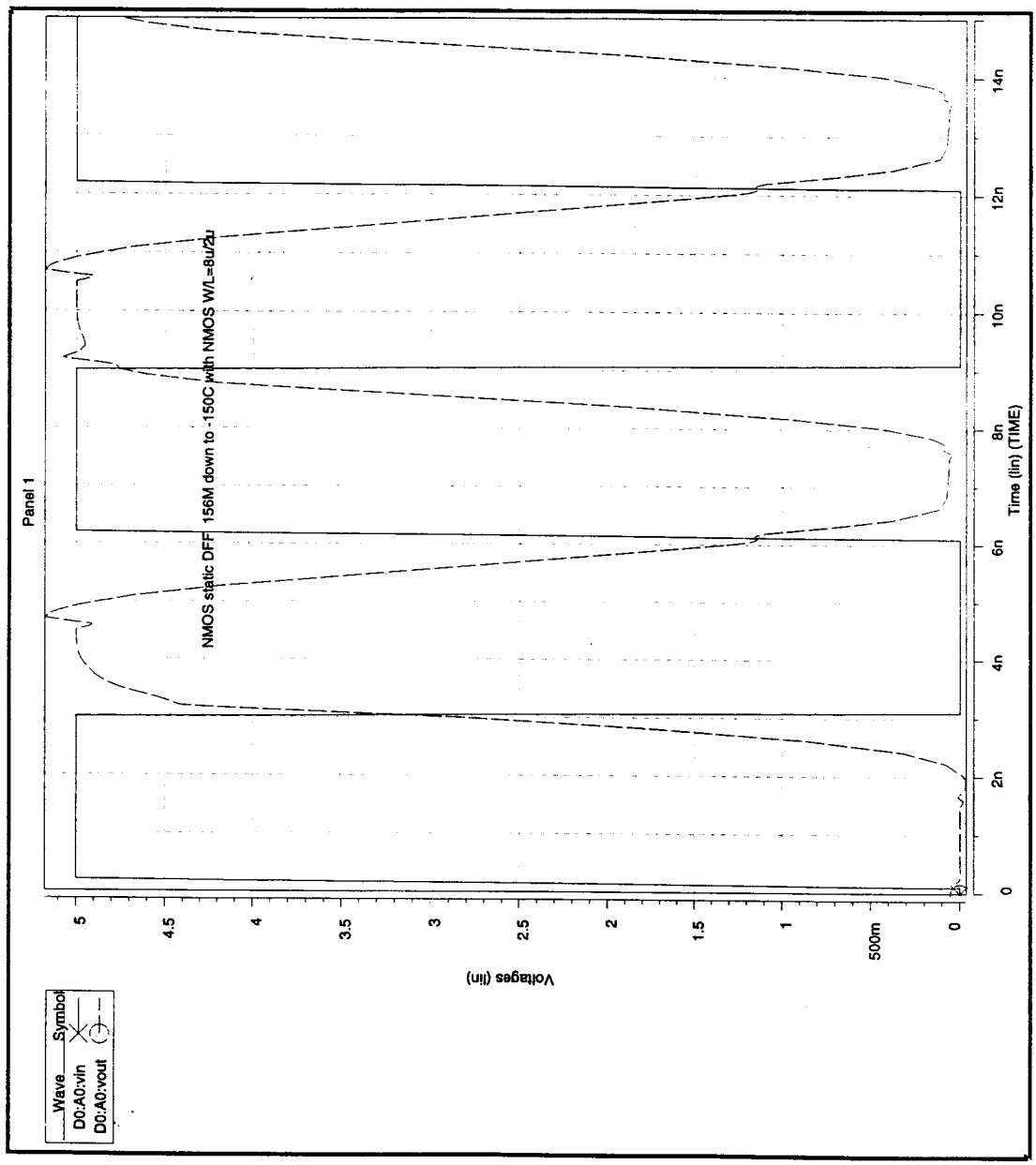


Figure A.7 DFF Simulation (2)

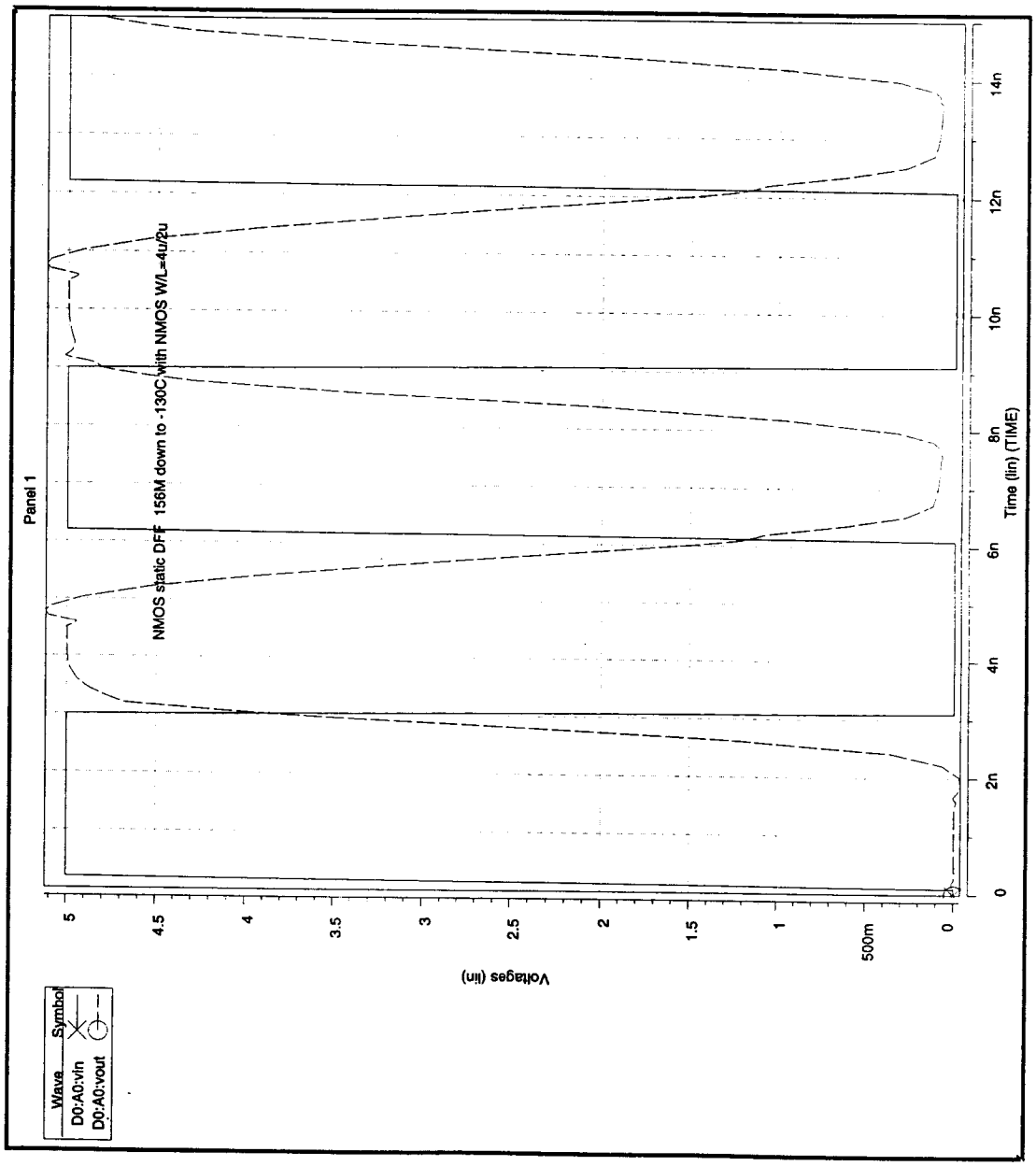


Figure A.8 DFF Simulation (3)

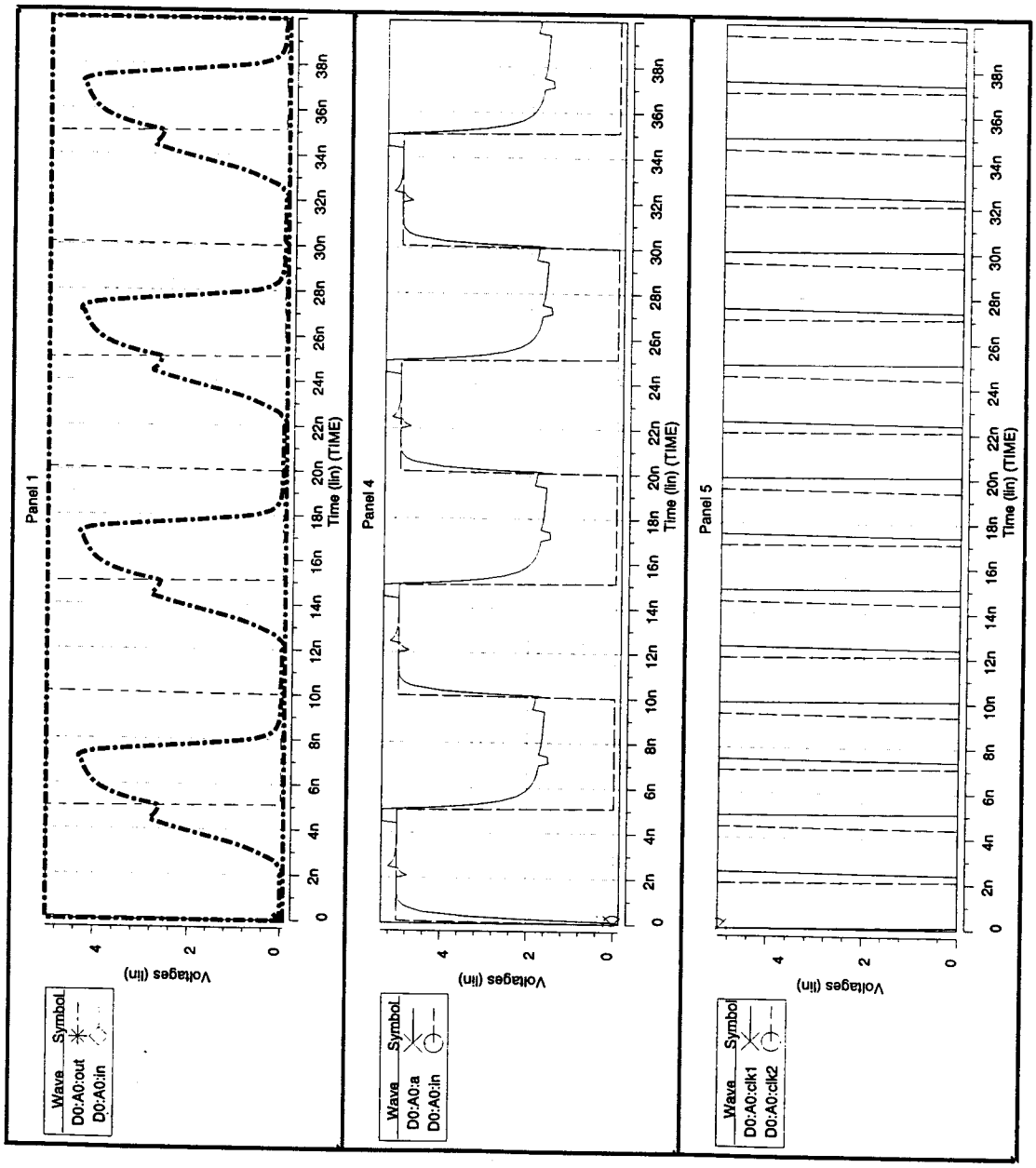


Figure A. 9 DFF Simulation (4)

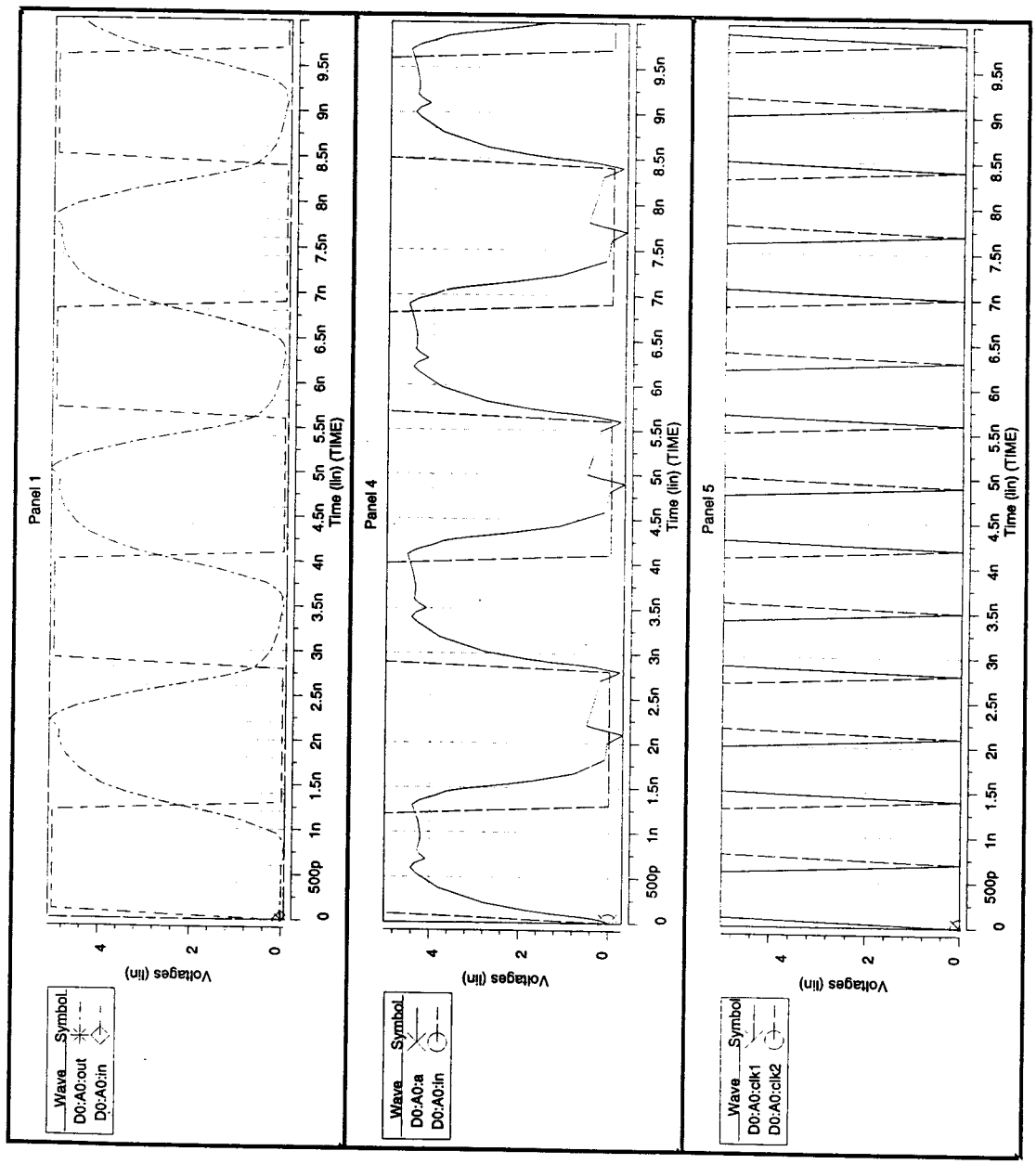


Figure A. 10 DFF Simulation (5)

Appendix B Wallace Tree Structures

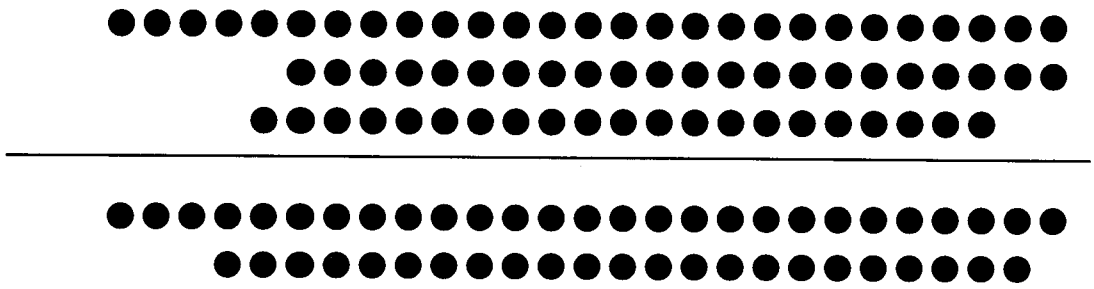


Figure B.1 MAC 01

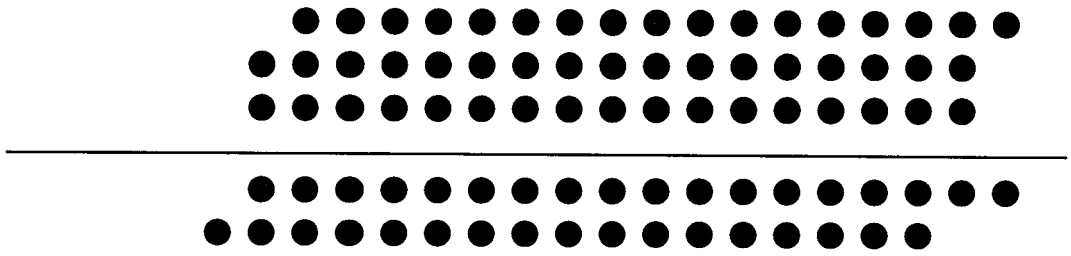


Figure B.2 MAC 02

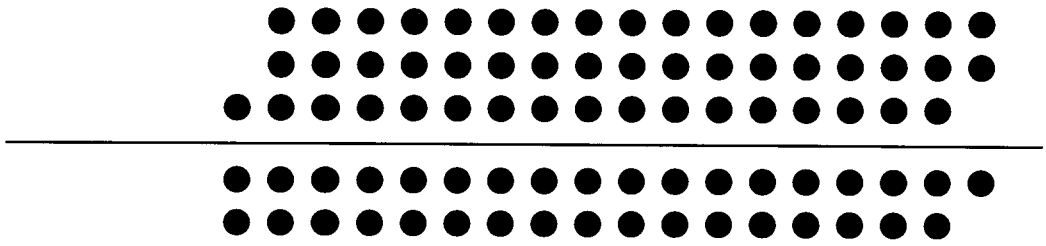


Figure B.3 MAC 03

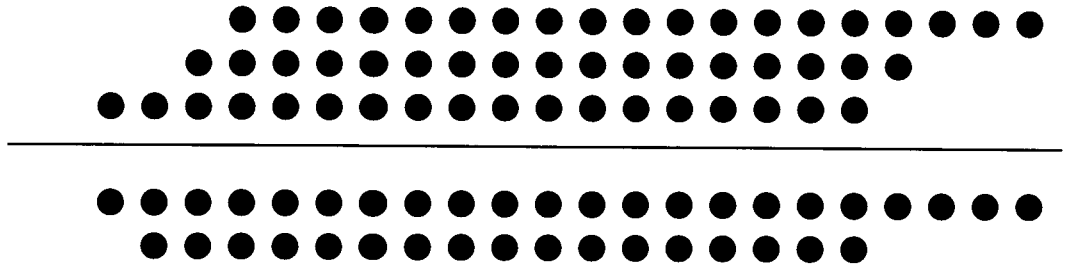


Figure B.4 MAC 04

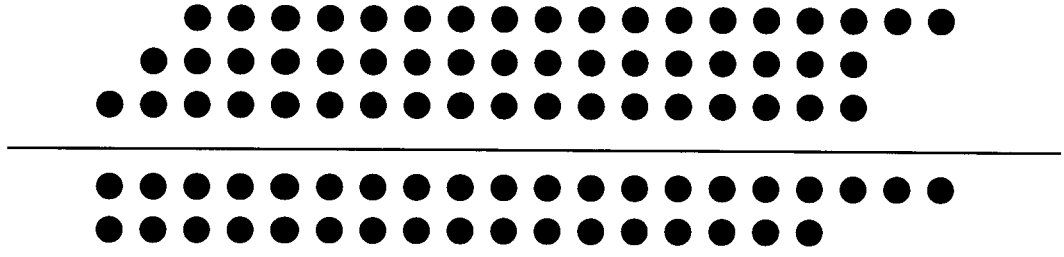


Figure B.5 MAC 11

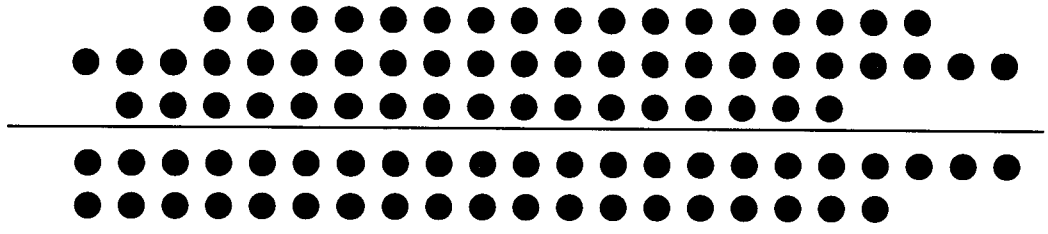


Figure B.6 MAC 12

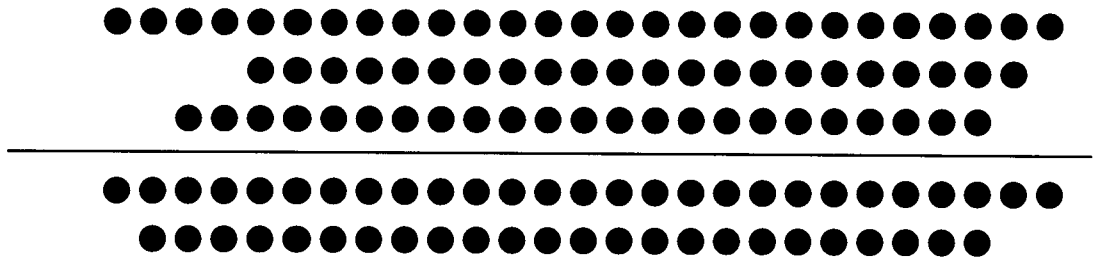


Figure B.7 MAC 21

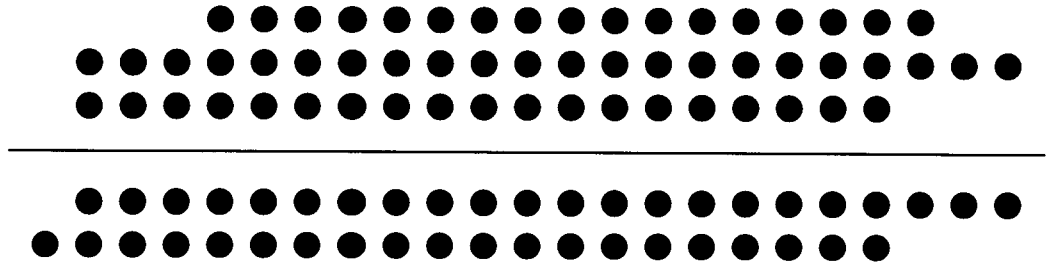


Figure B.8 MAC 22

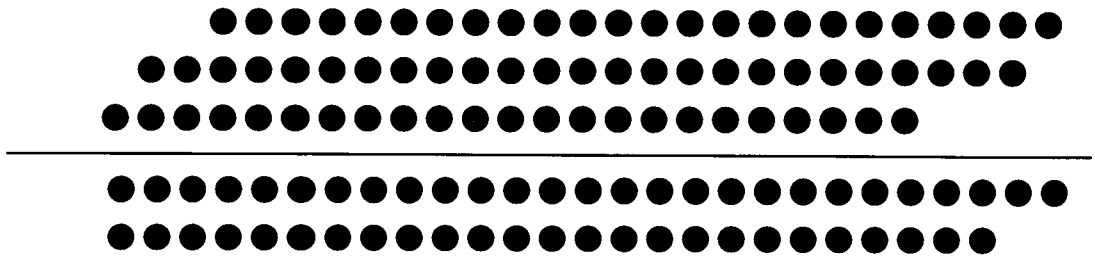


Figure B.9 MACRO 31

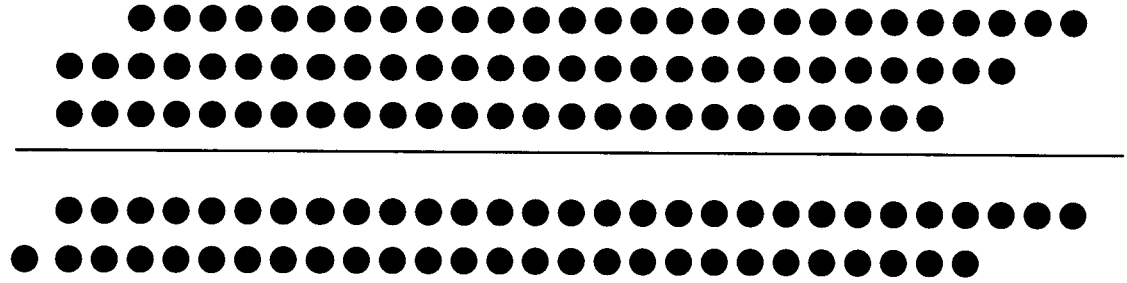


Figure B.10 MACRO 41

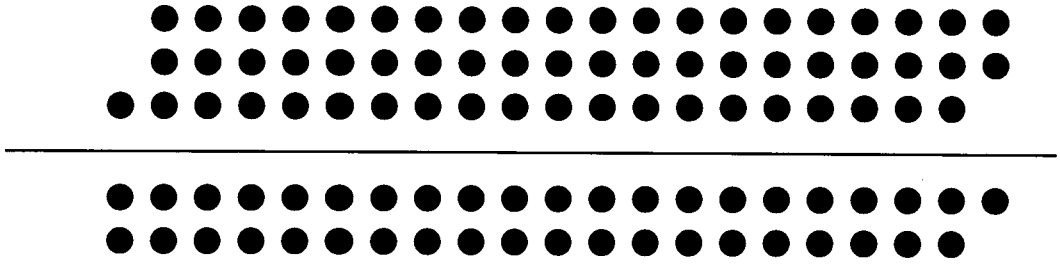


Figure B.11 MAC 3

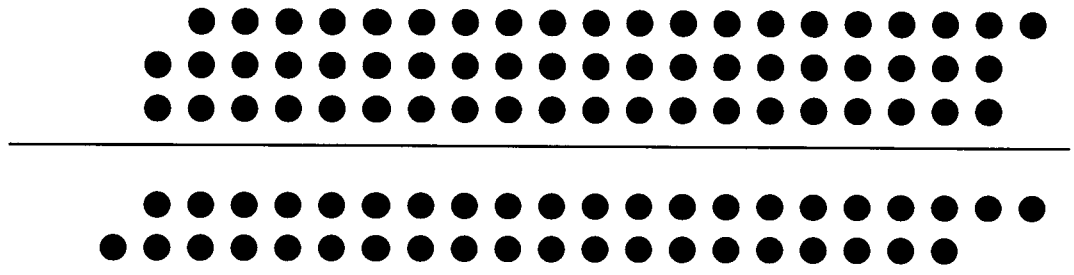


Figure B.12 MAC 4

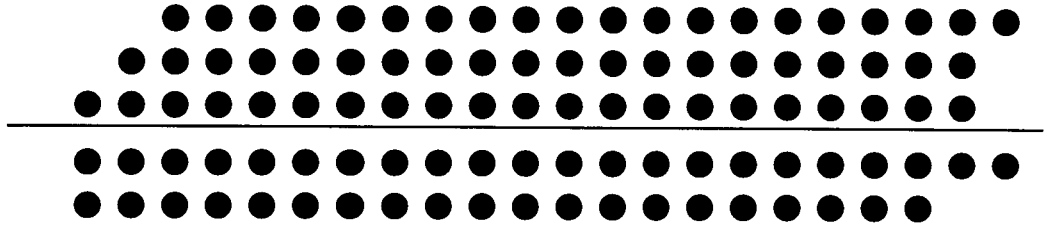


Figure B.13 MAC 5

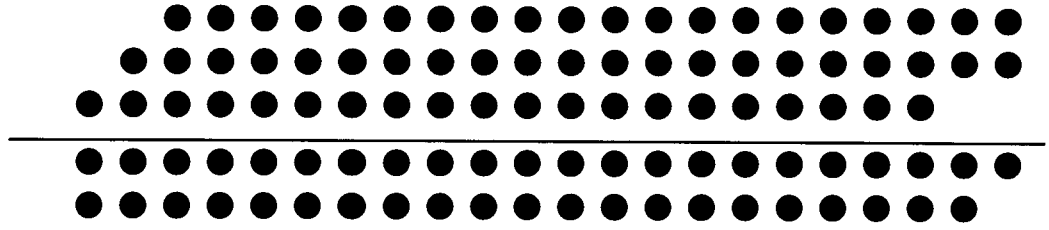


Figure B.14 MAC 6

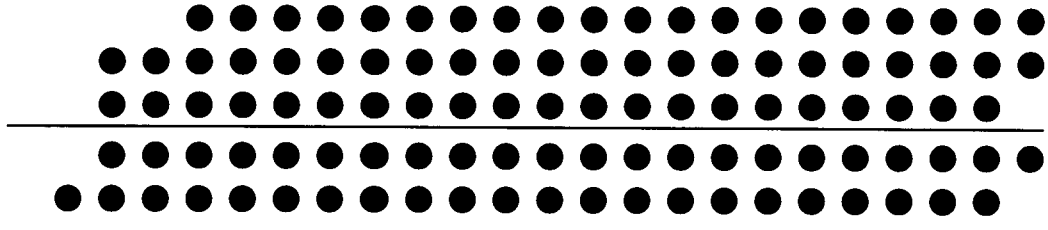


Figure B.15 MAC 7

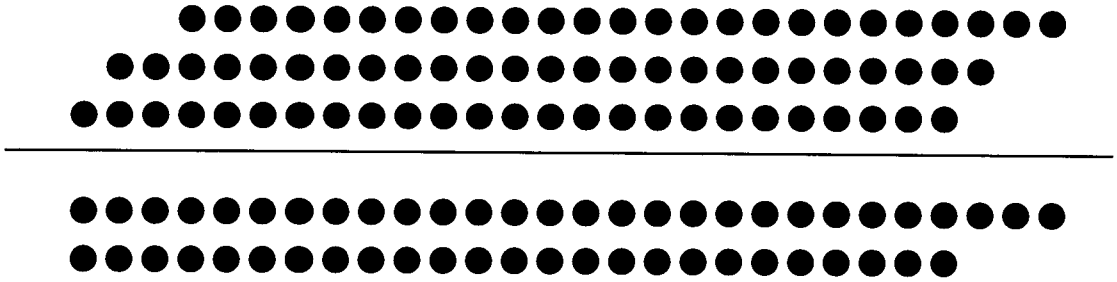


Figure B.16 MAC 8

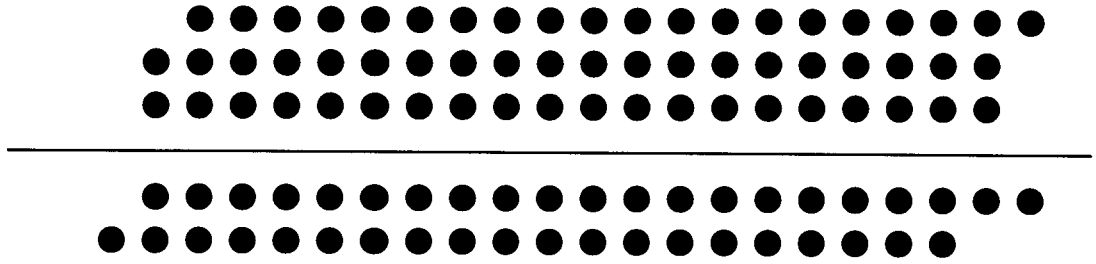


Figure B.17 W3N20L20

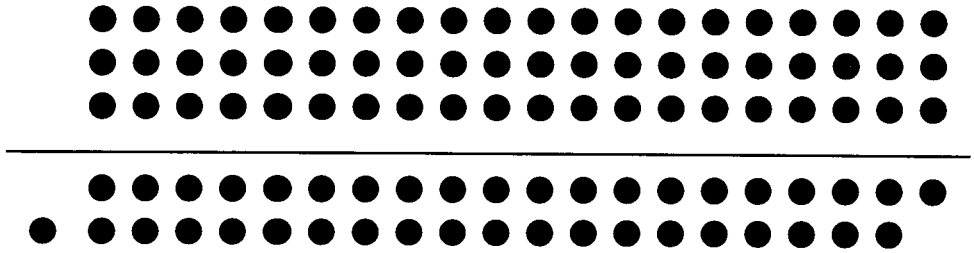
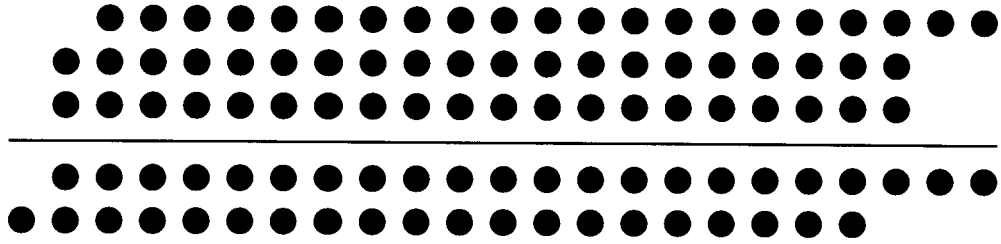
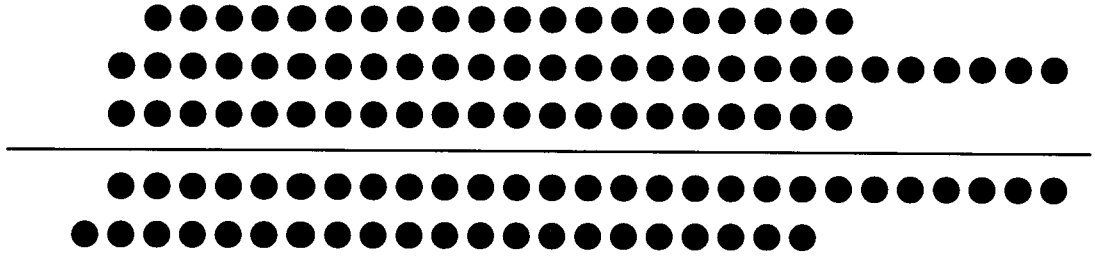


Figure B.18 W3N20B



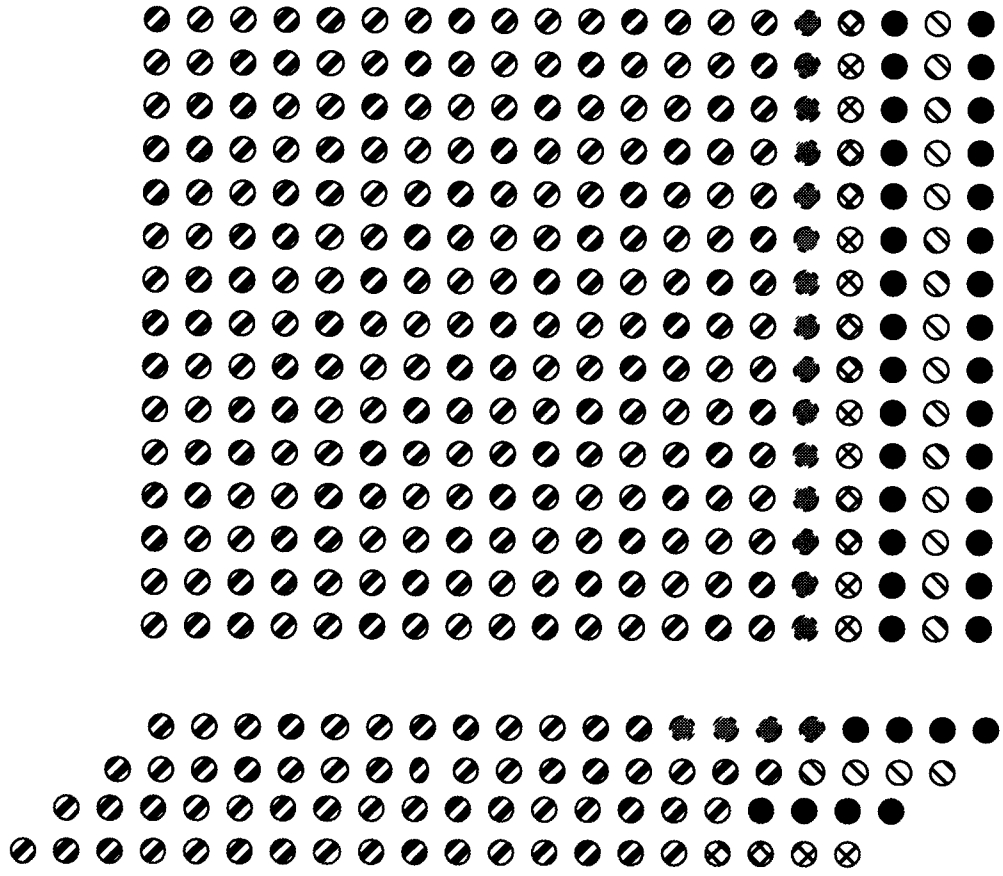
W3N21L20

Figure B.19 W3N21L20



W3N26L20

Figure B.20 W3N26L20



W15N20B
15 20-bit Wallace Tree

Figure B.21 W15N20B

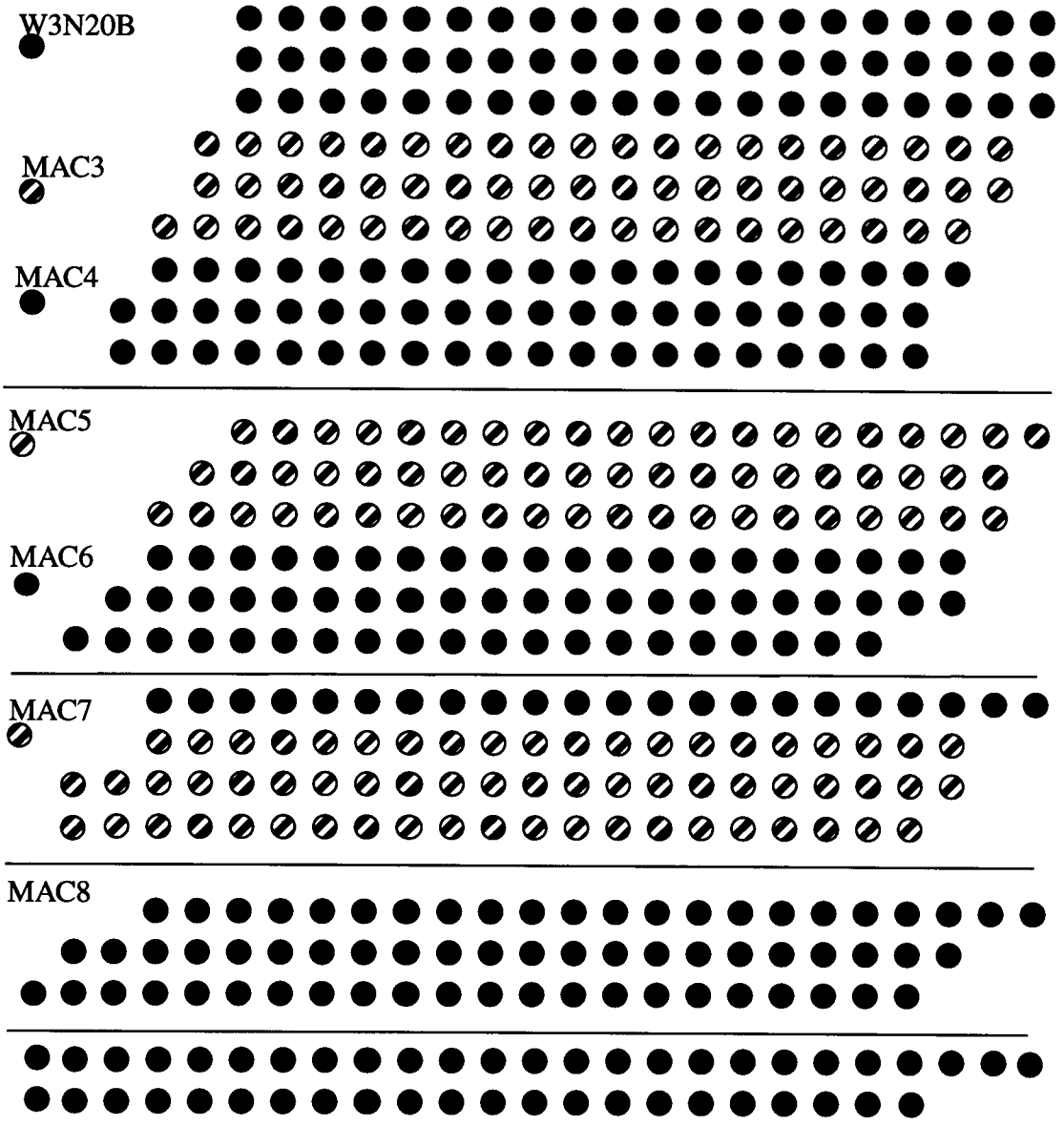


Figure B.22 W9N20B

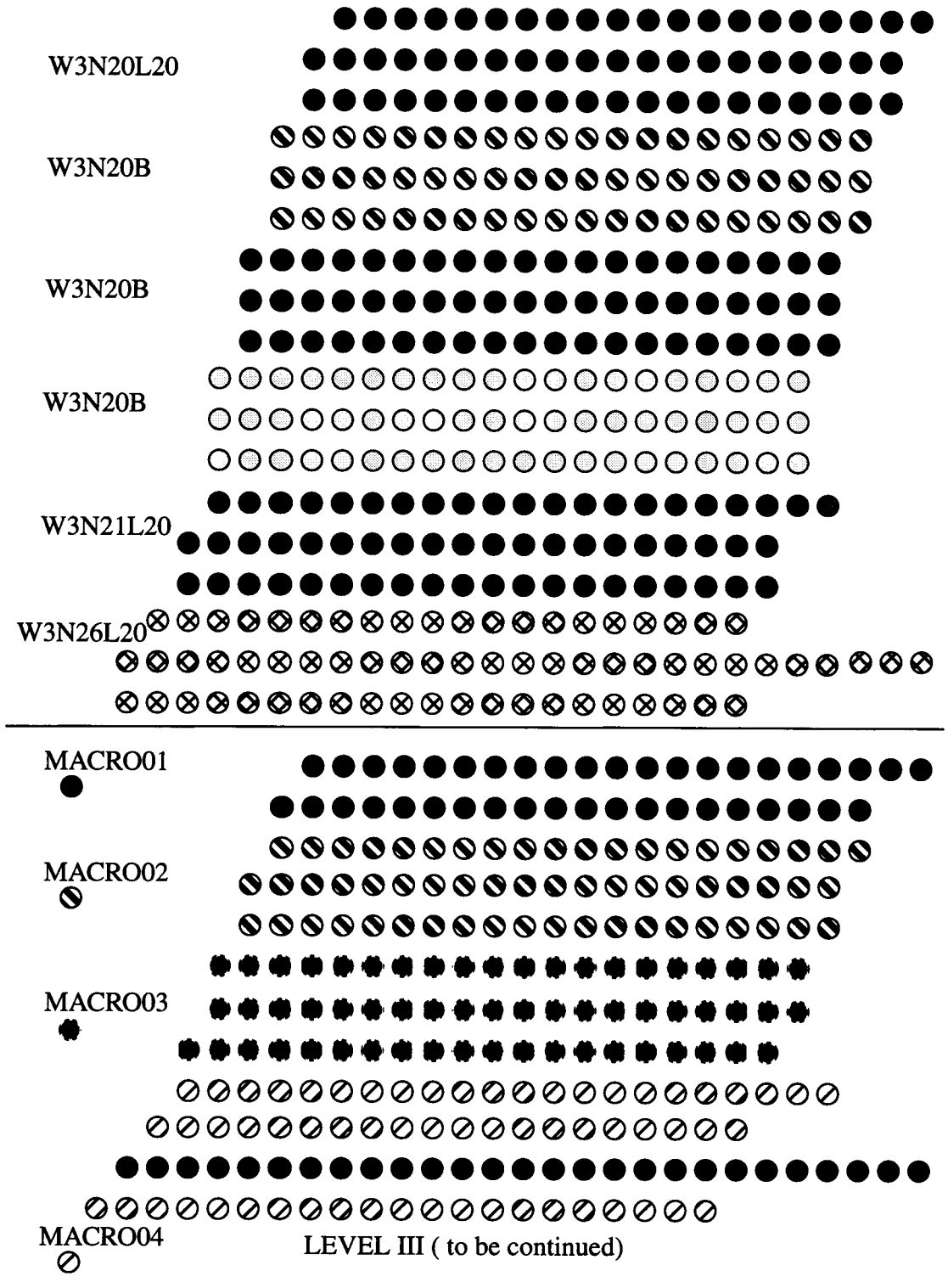


Figure B.23 Level III

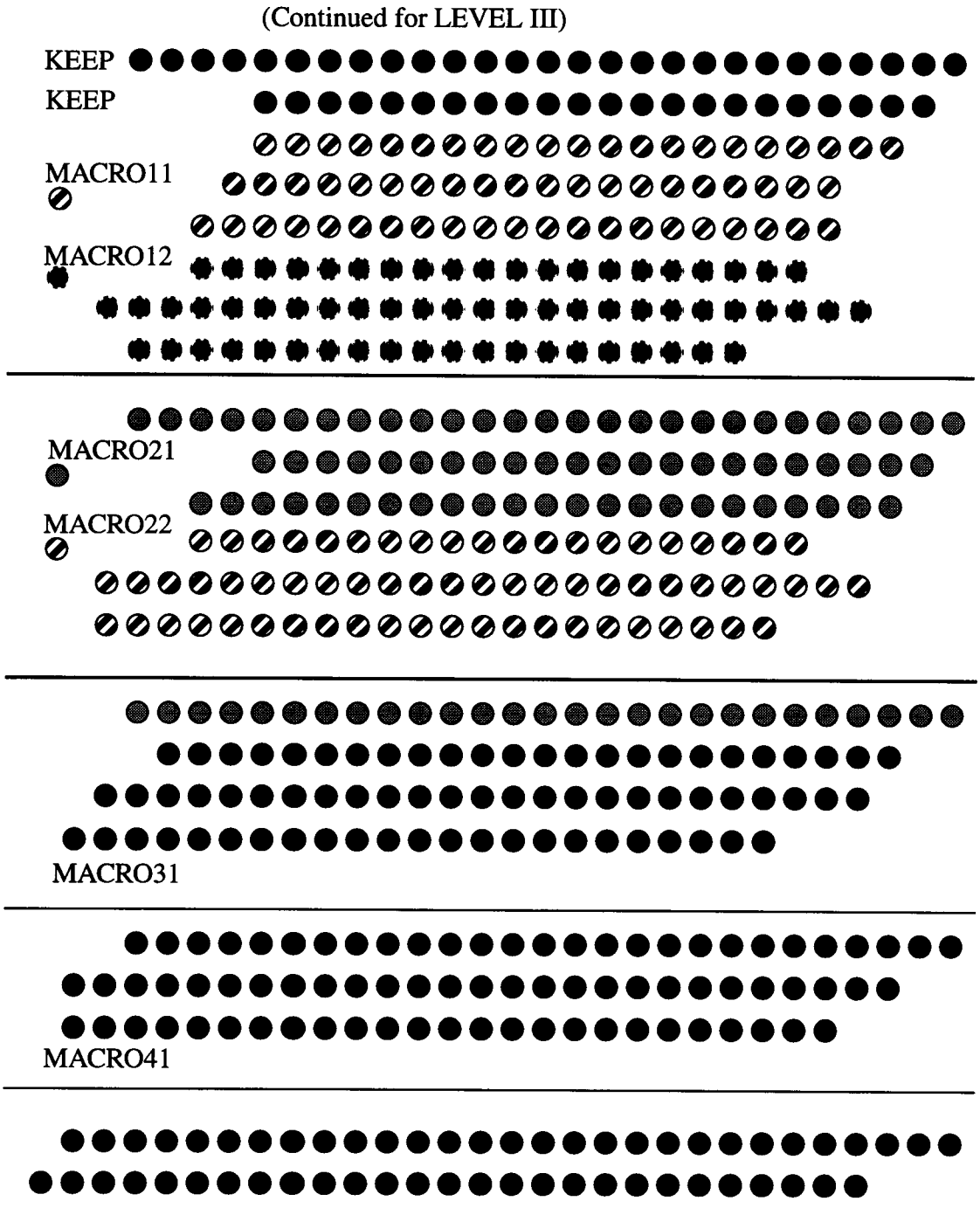


Figure B.23 Level III (continued)

Appendix C Schematics of the Wallace Trees

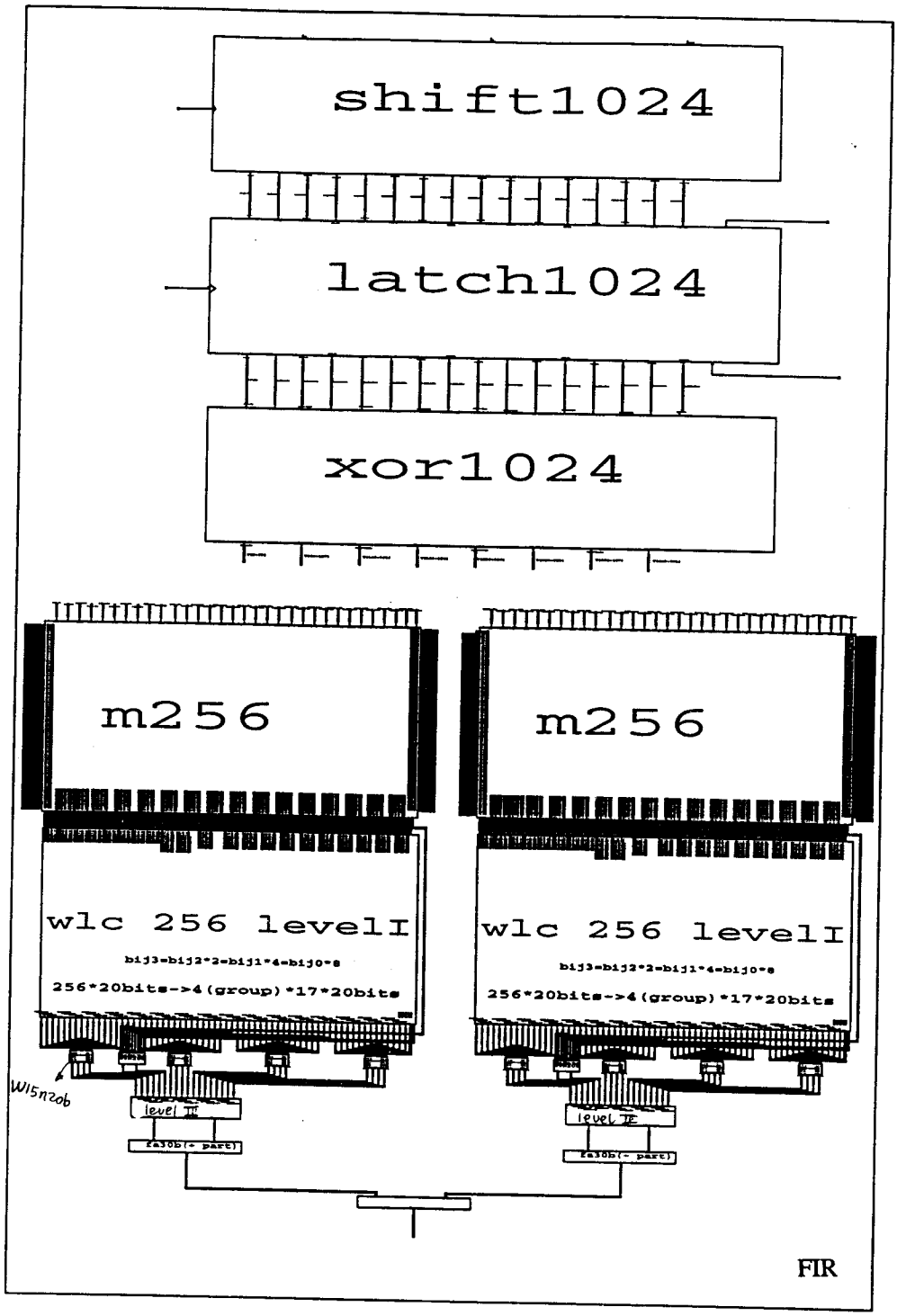


Figure C.1 FIR Top Level Schematic

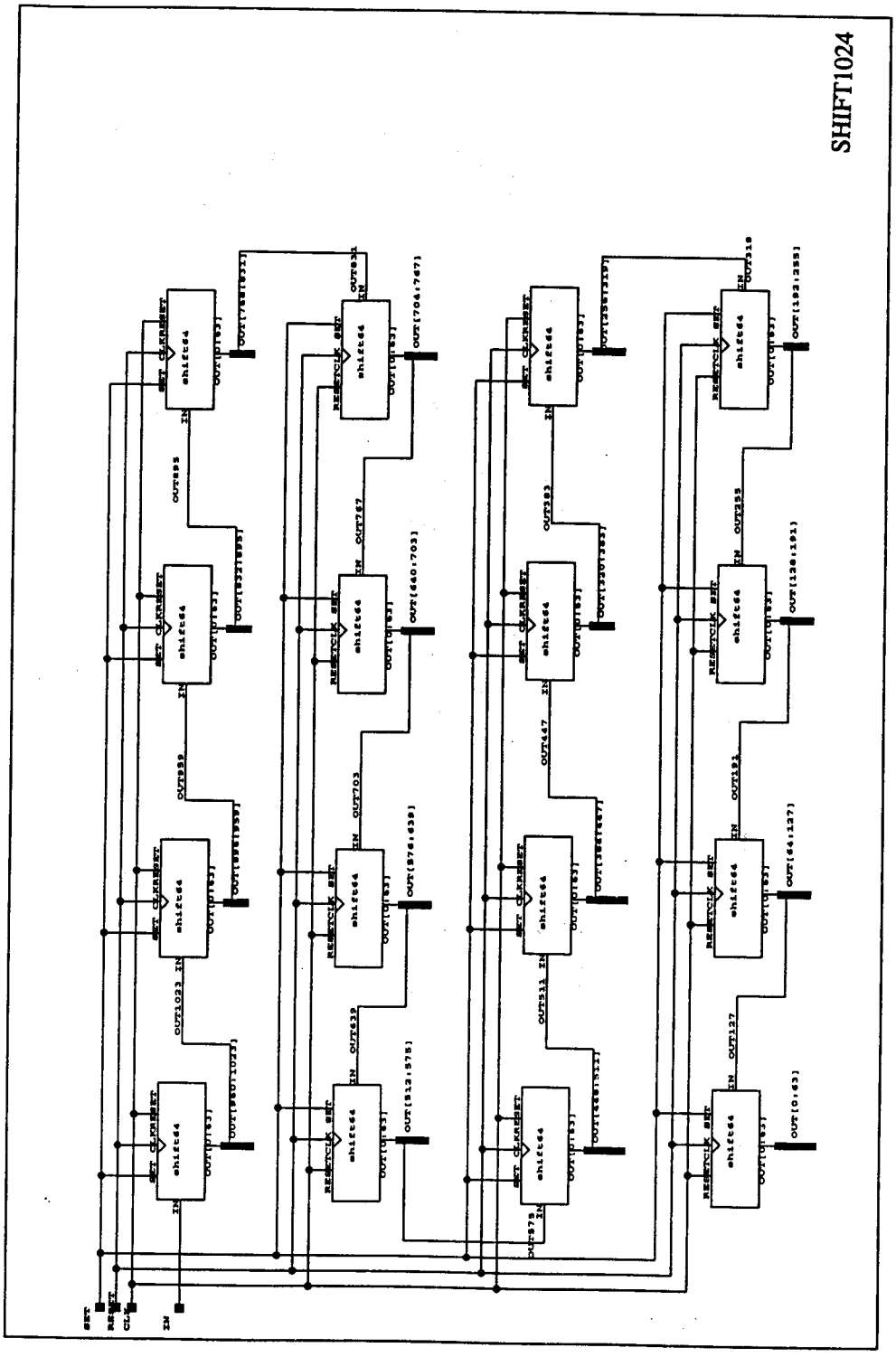


Figure C.2 Shift 1024

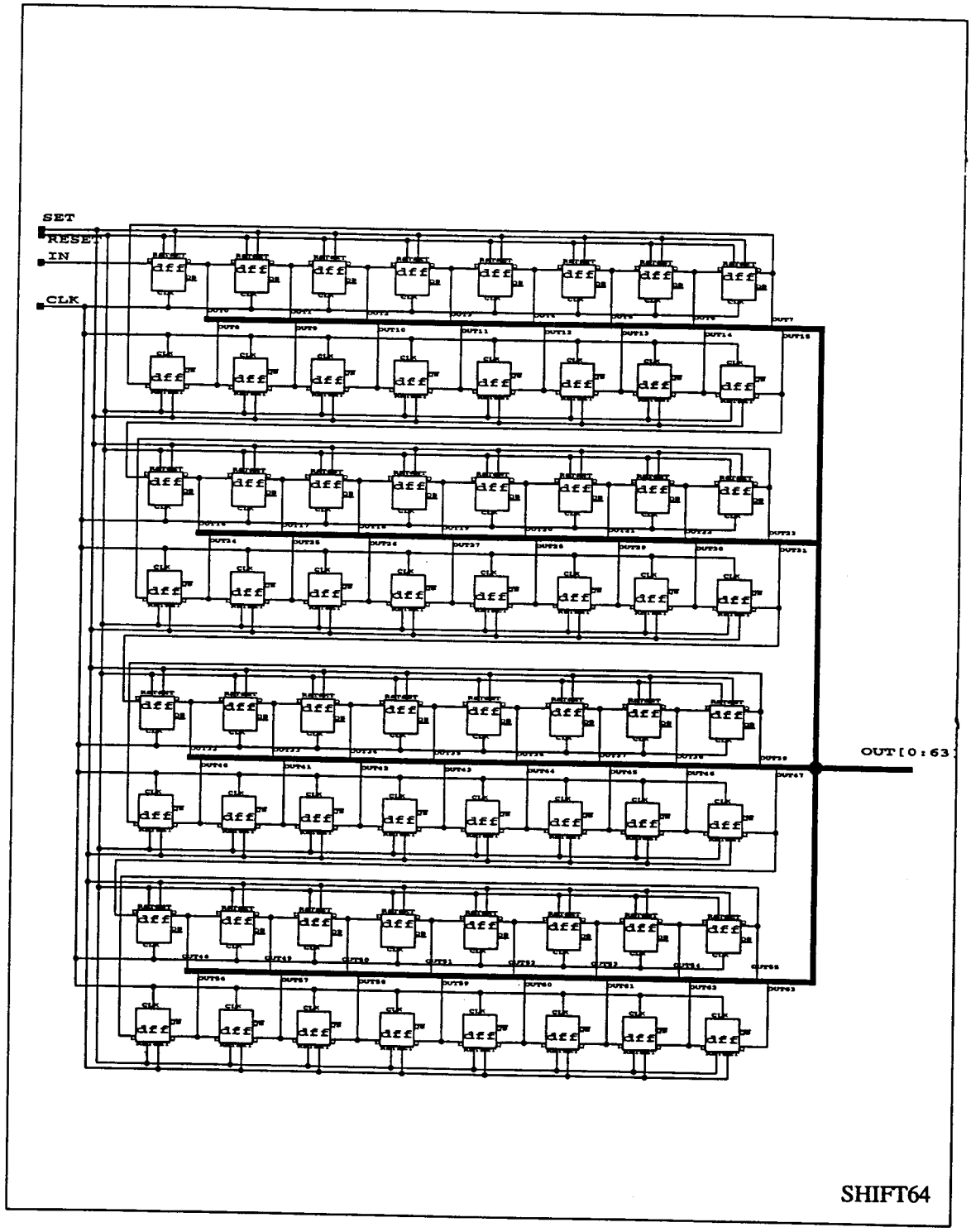


Figure C.3 Shift 64

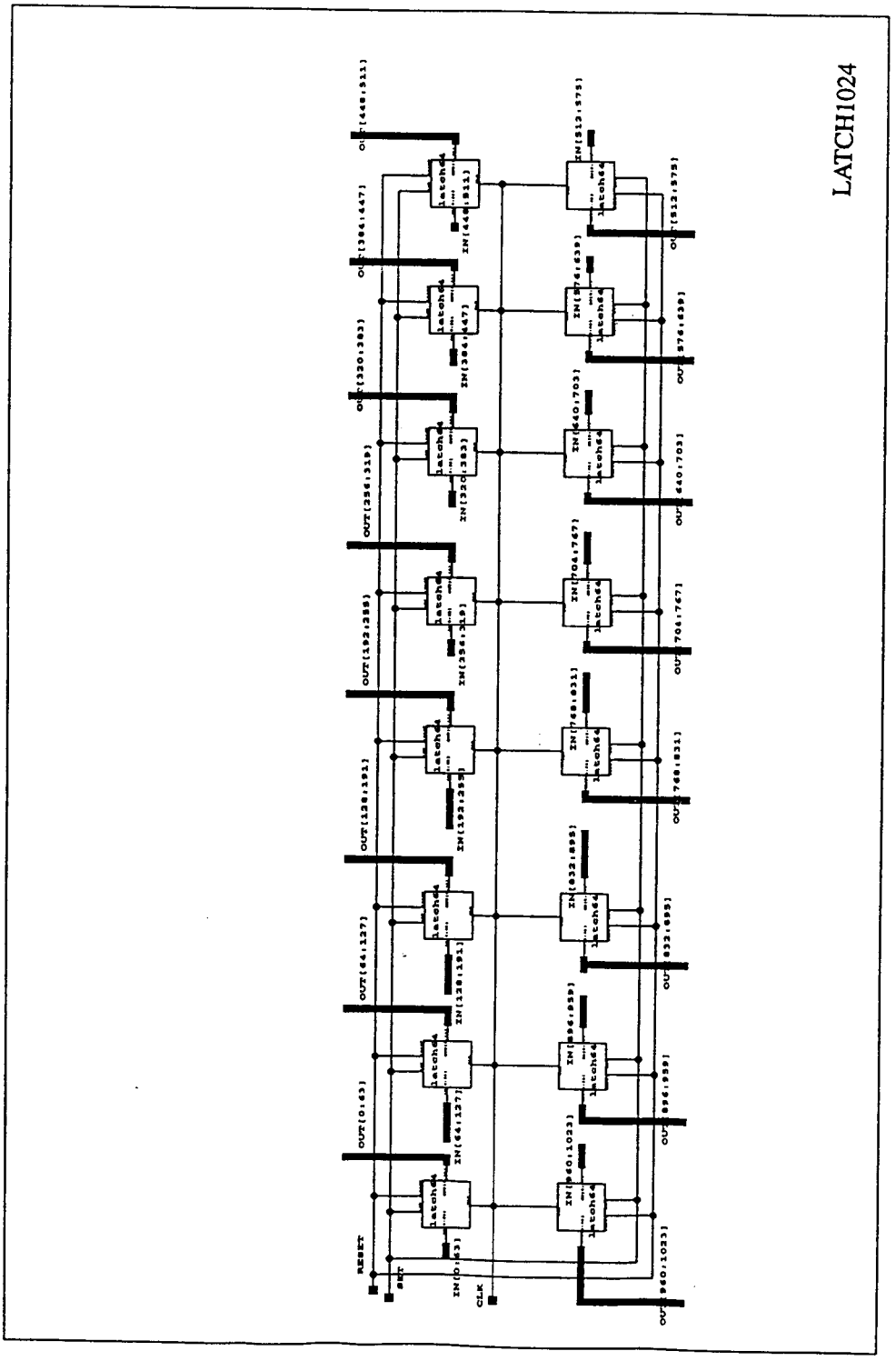


Figure C.4 Latch 1024

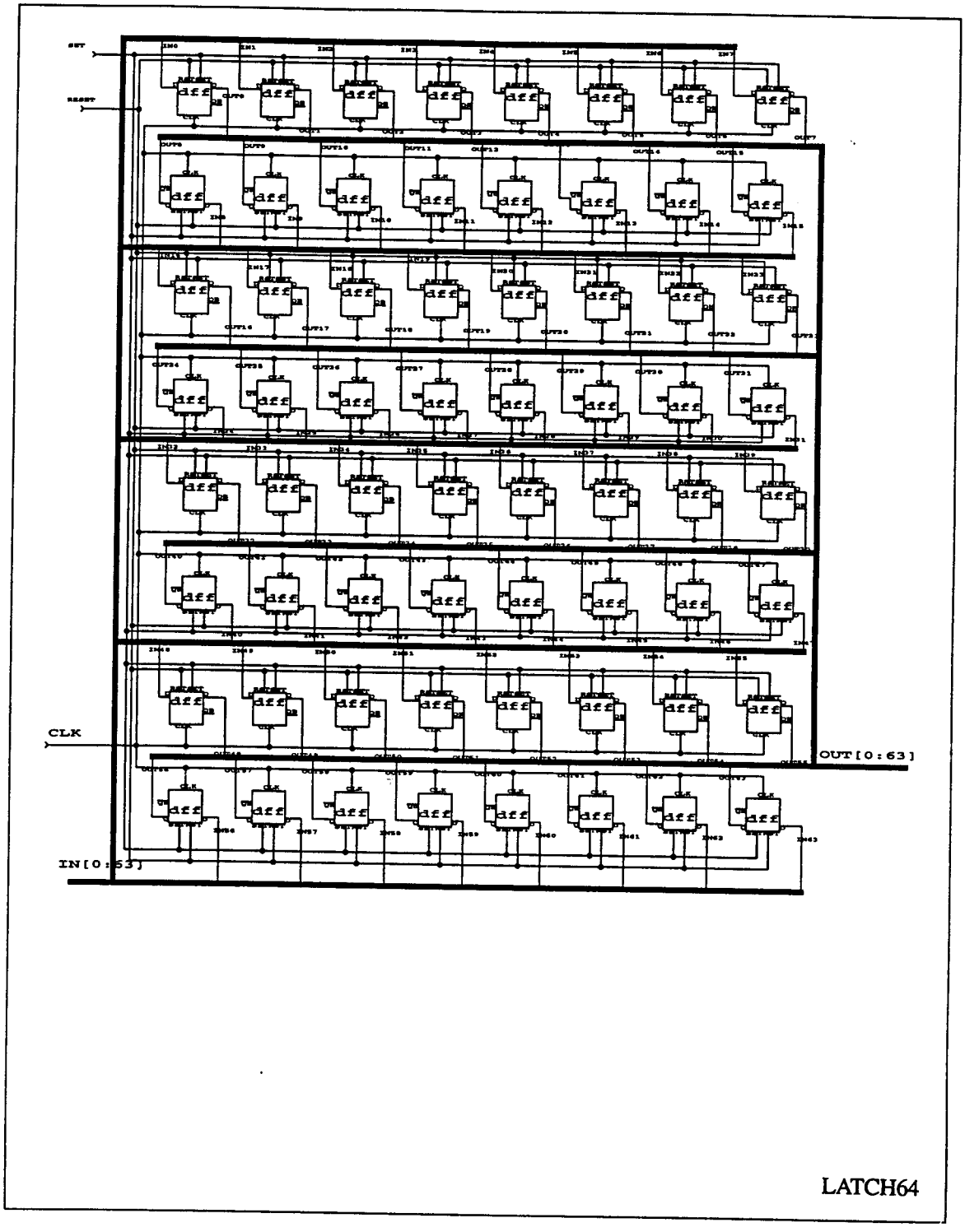


Figure C.5 Latch 64

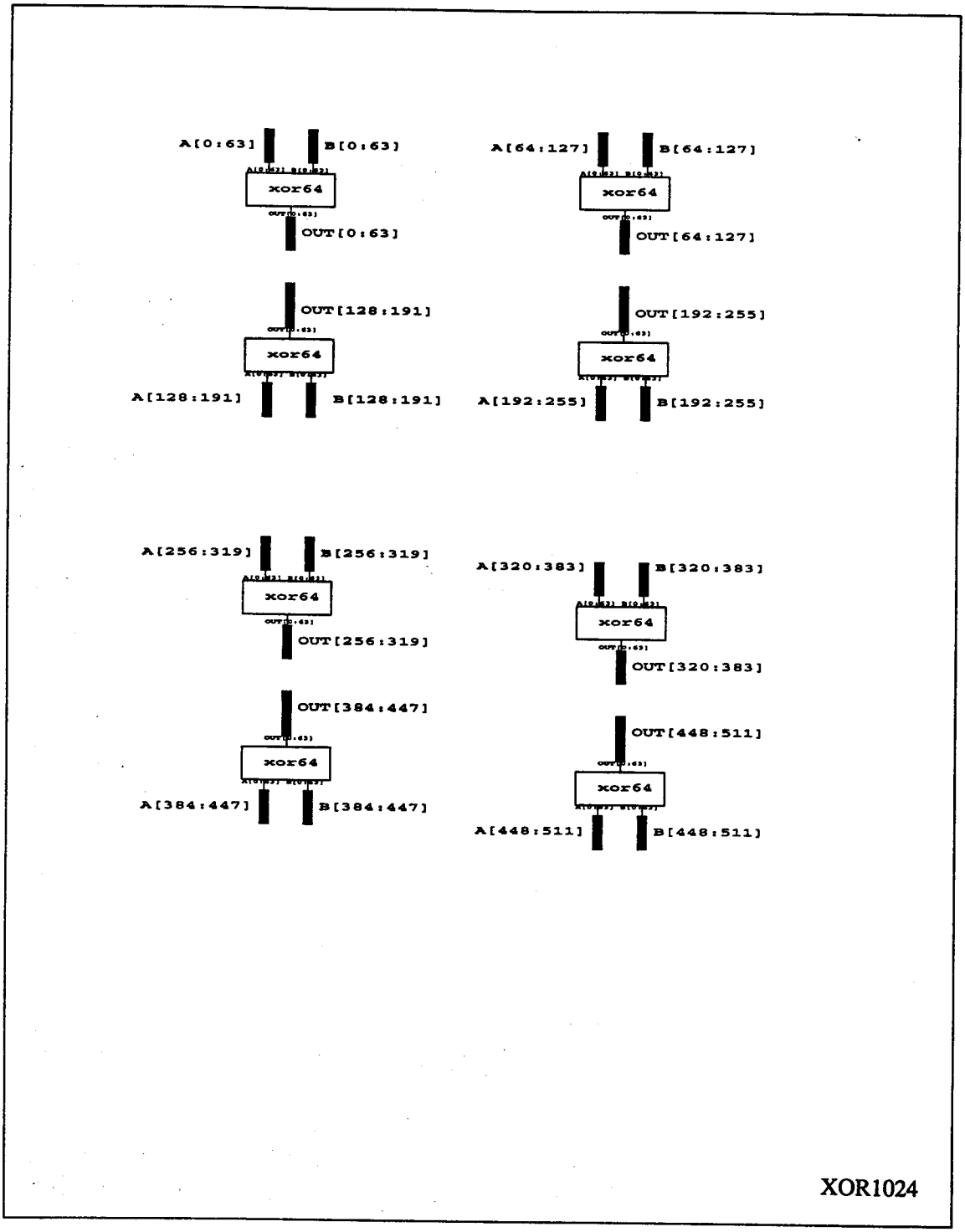


Figure C.6 XOR 512

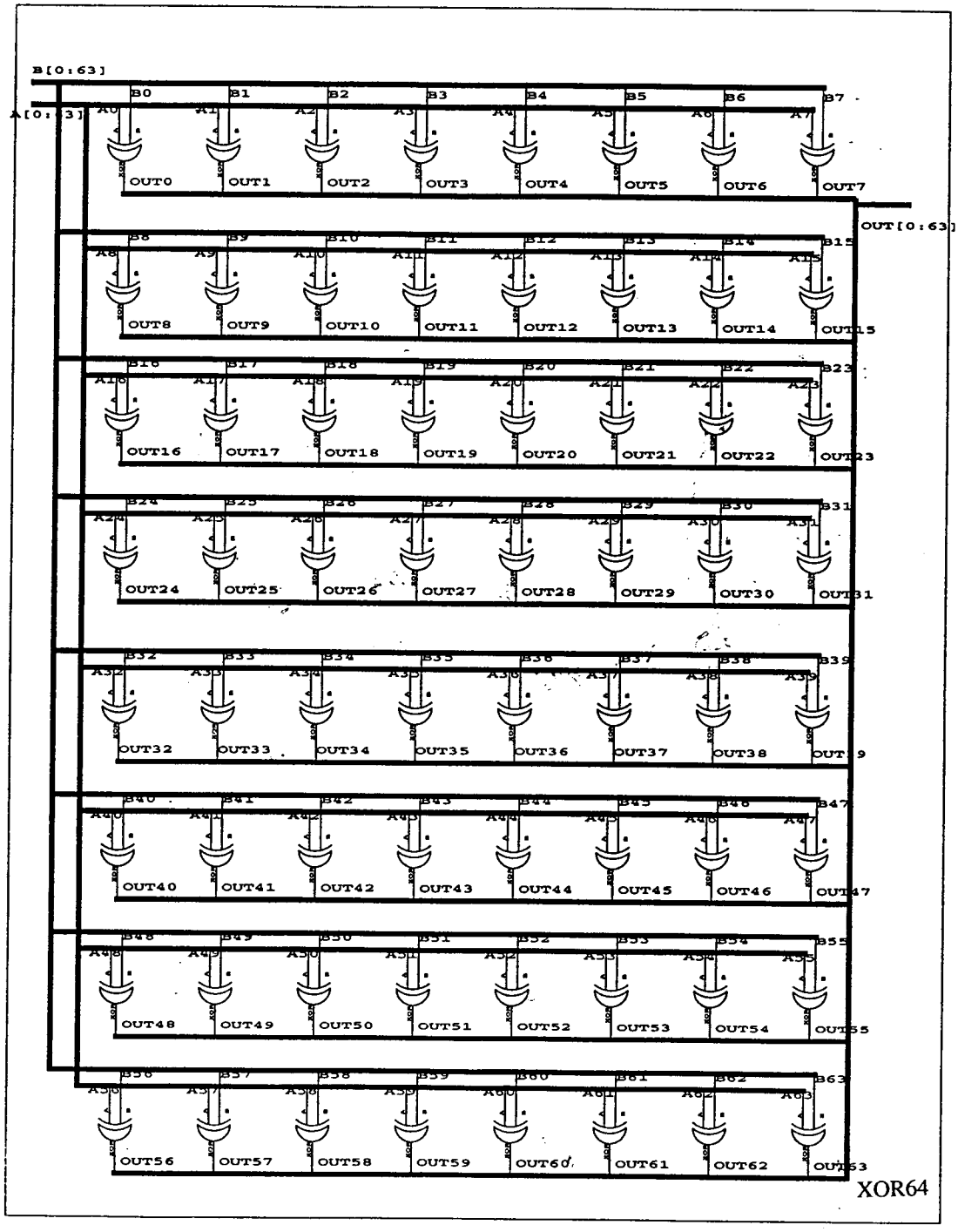


Figure C.7 XOR 64

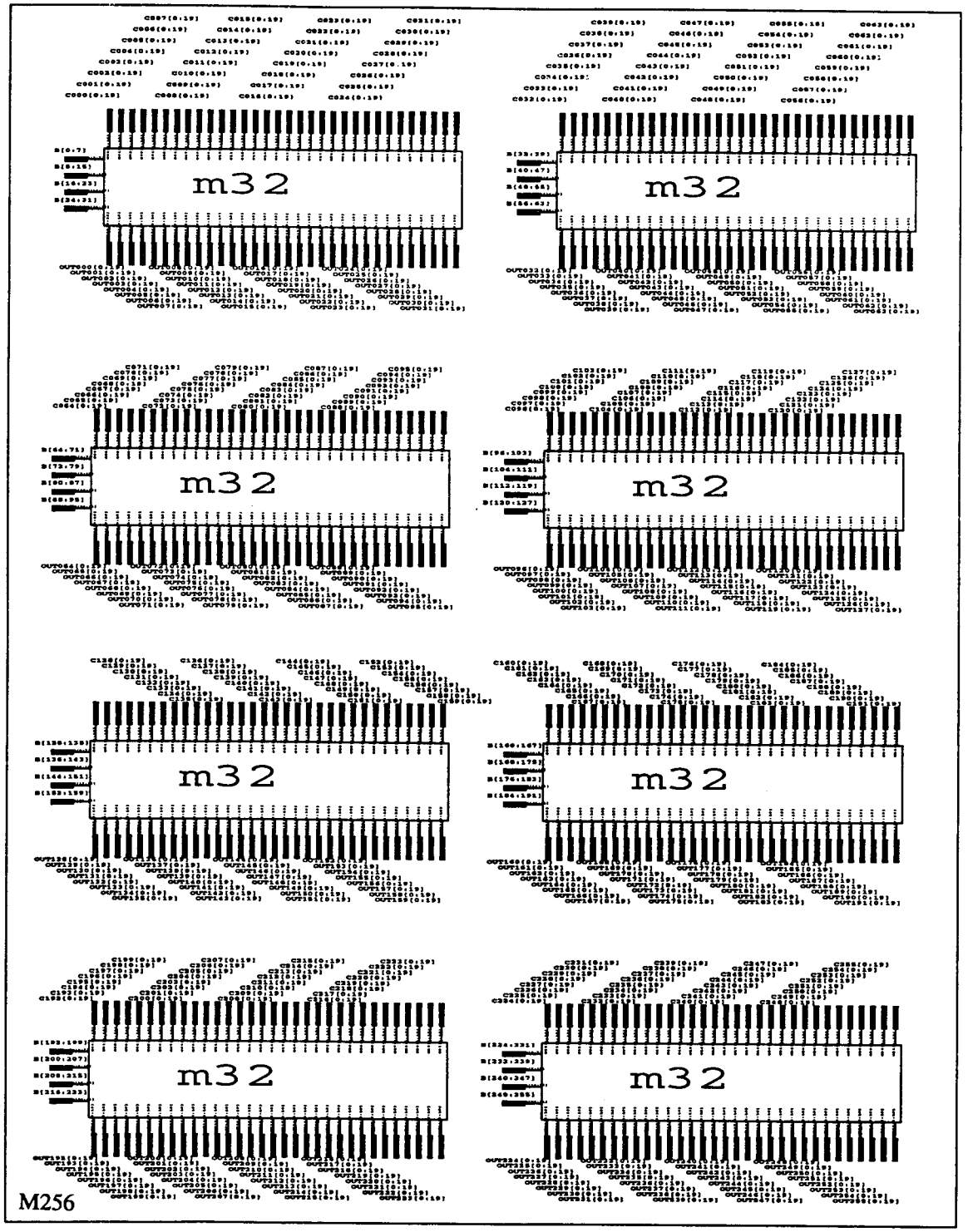


Figure C.8 M256

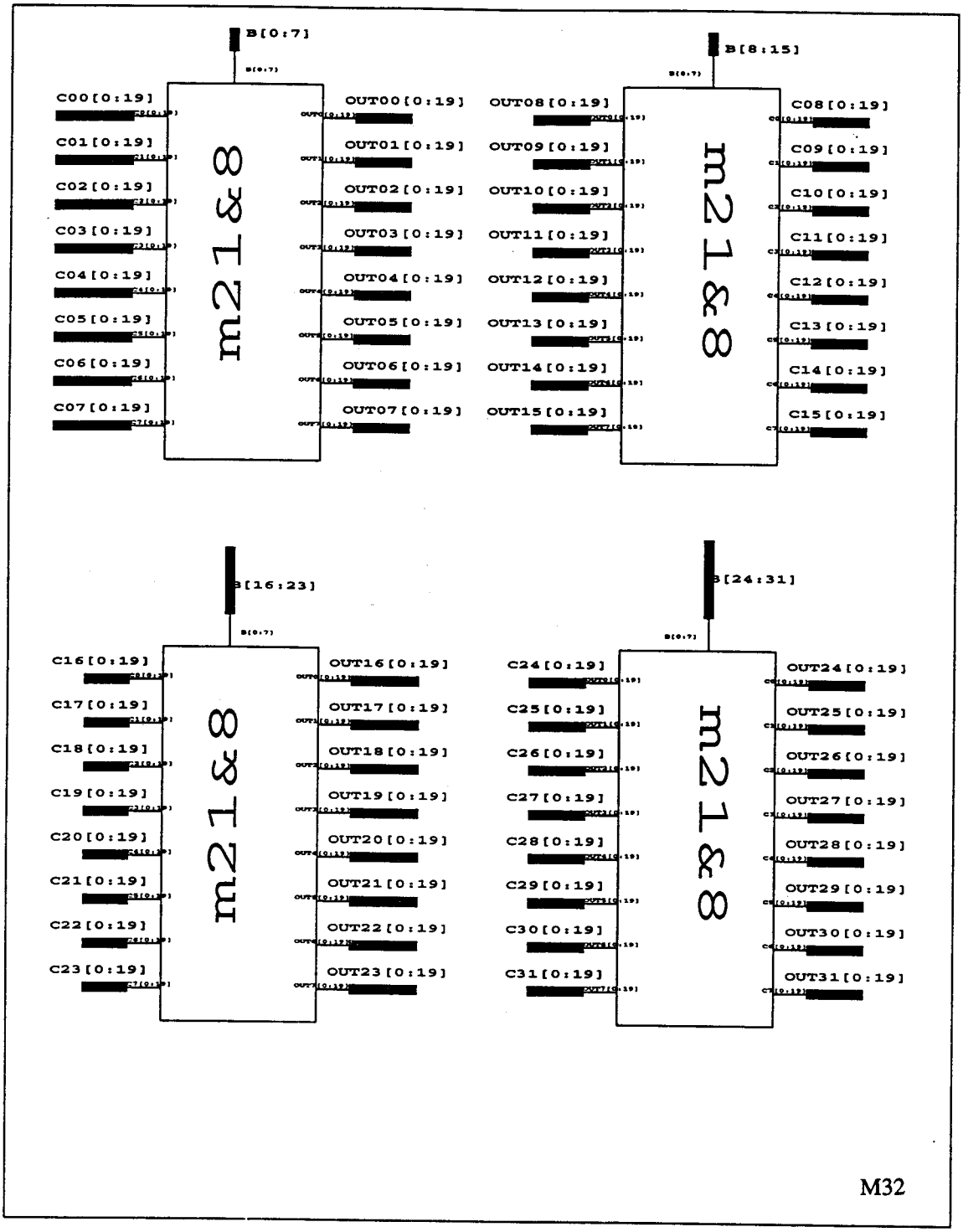


Figure C.9 M32

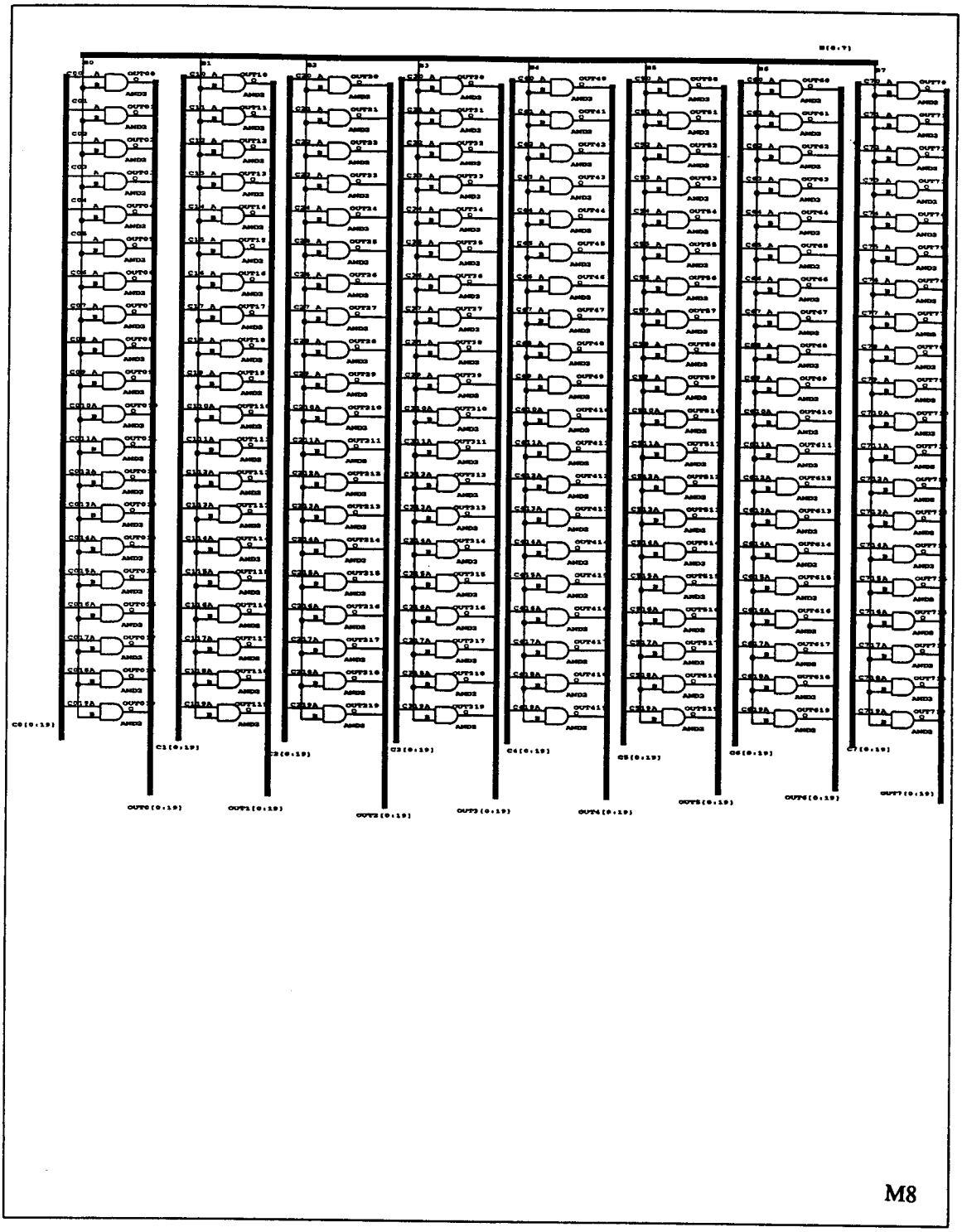


Figure C.10 M8

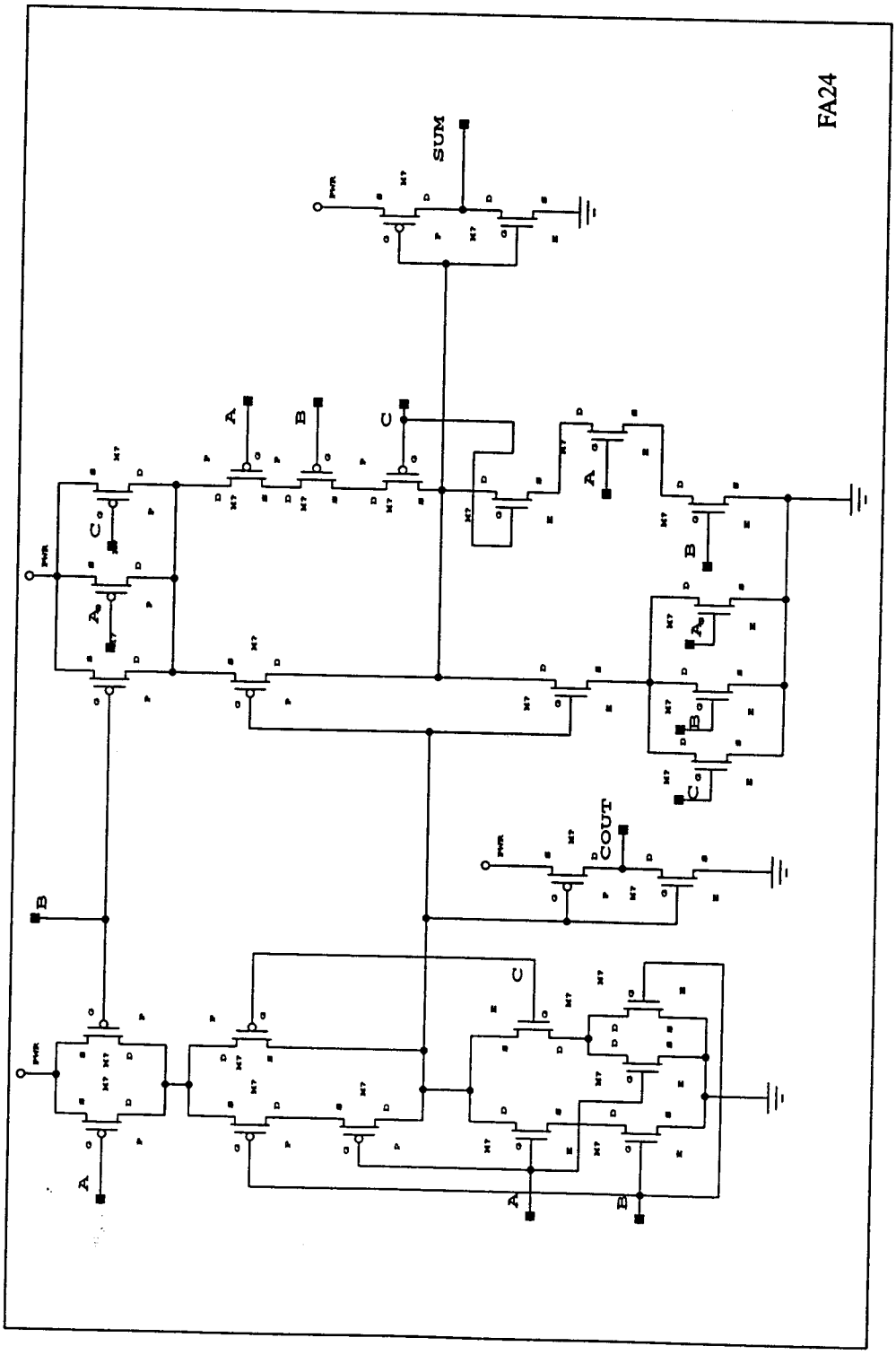


Figure C.11 FA 24 or Counter (3,2)

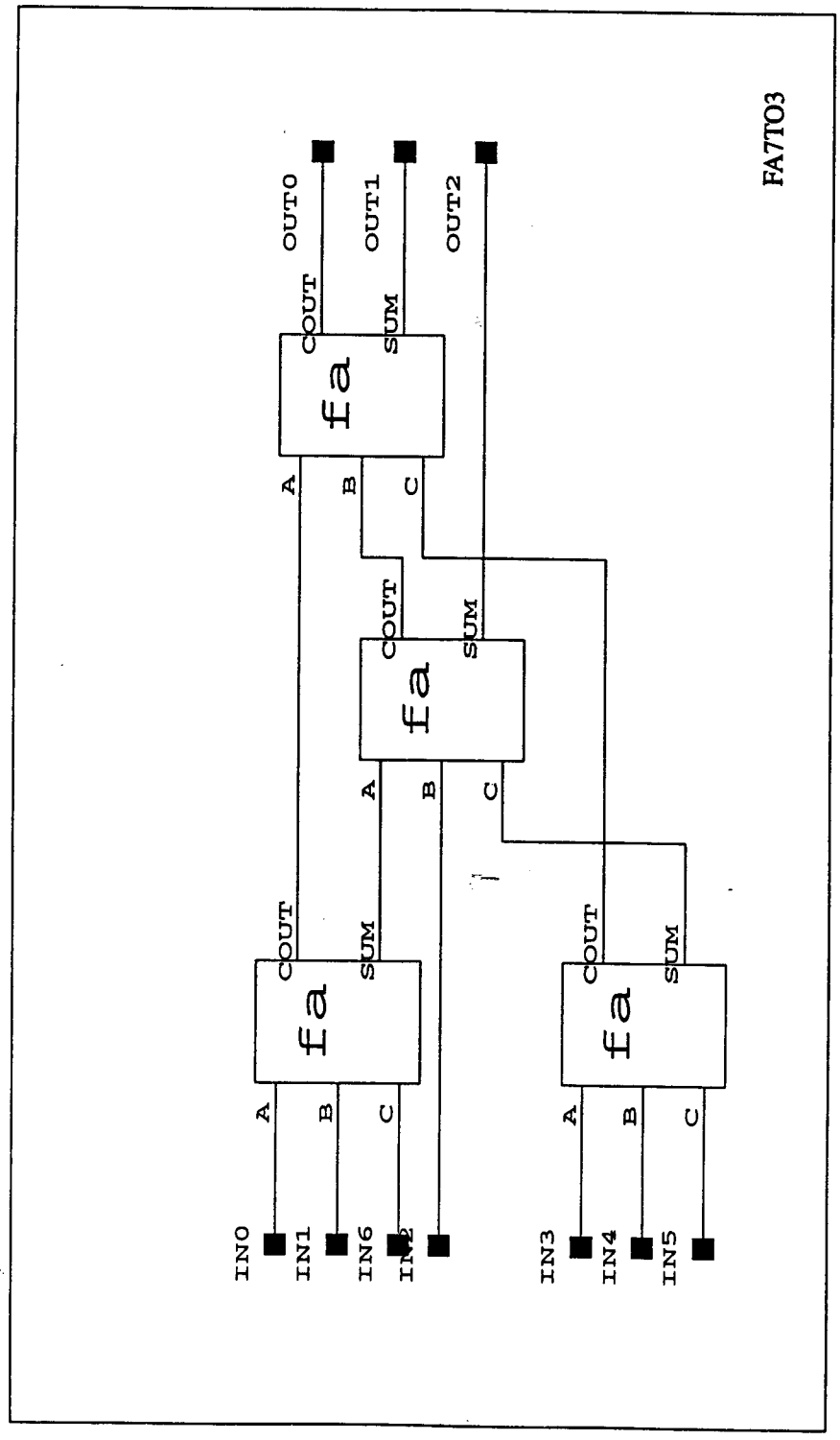


Figure C.12 Counter (7,3)

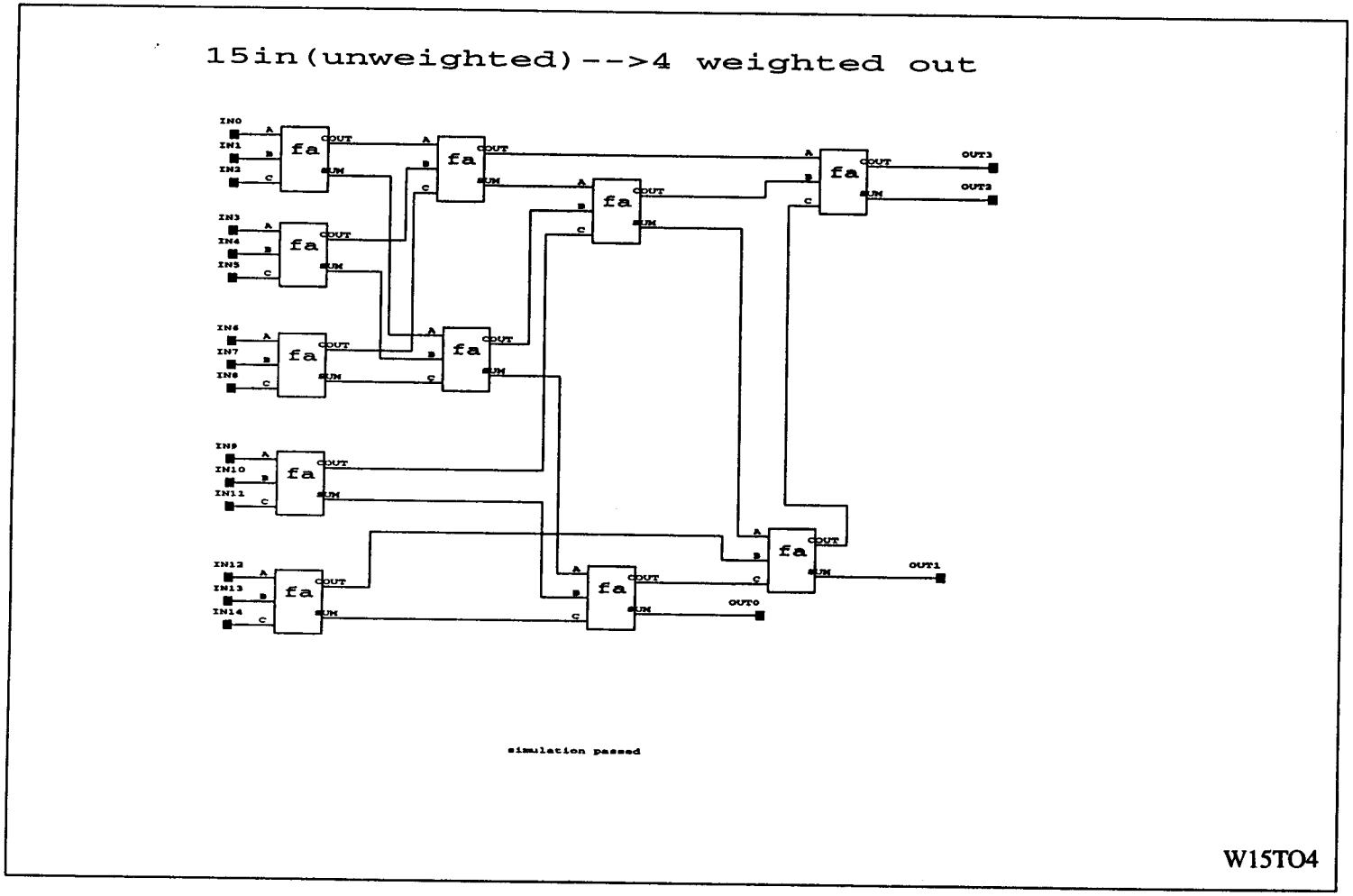


Figure C.13 Counter (15,4)

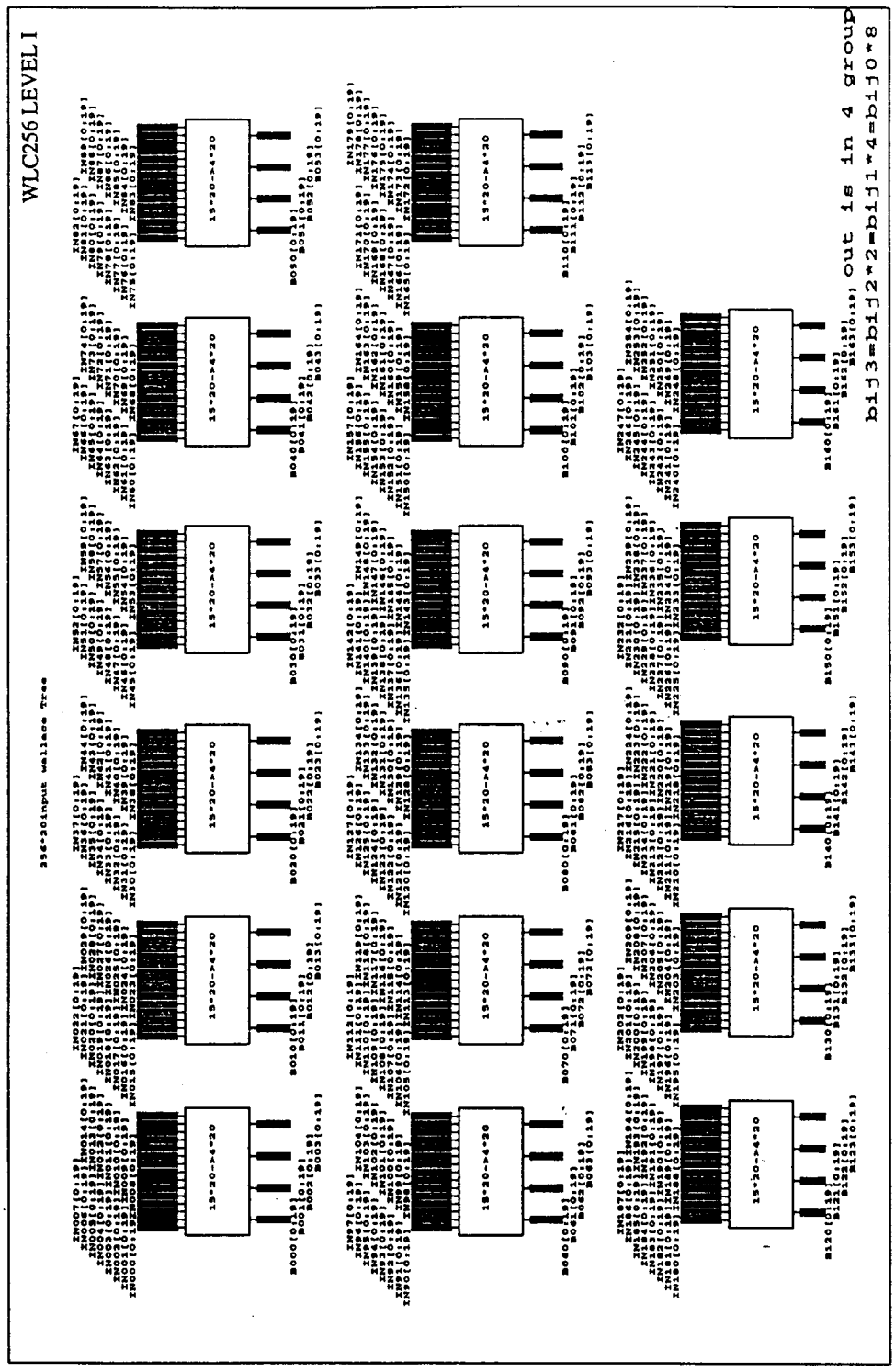


Figure C.14 Level I

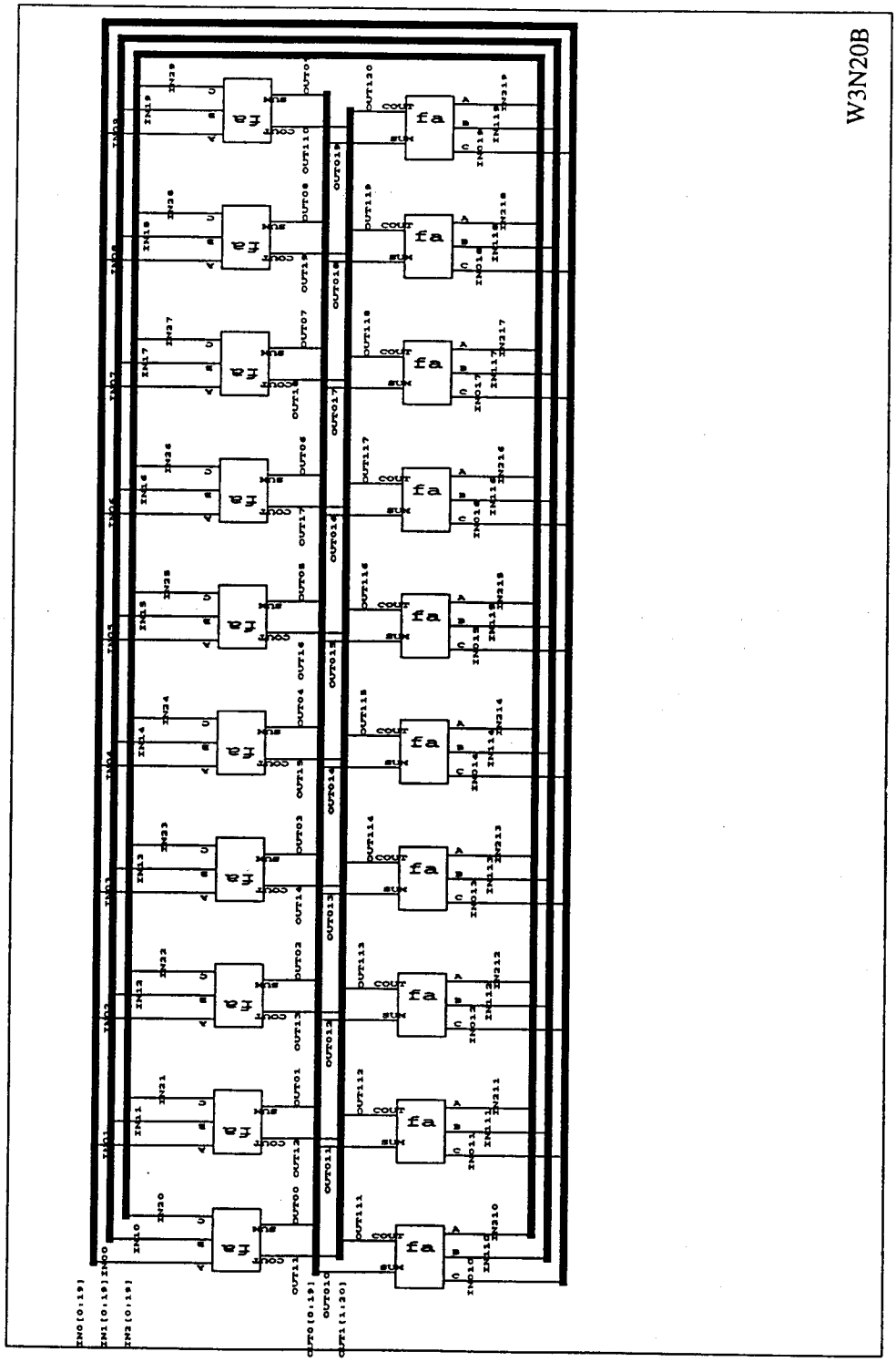


Figure C.15 W3N20B

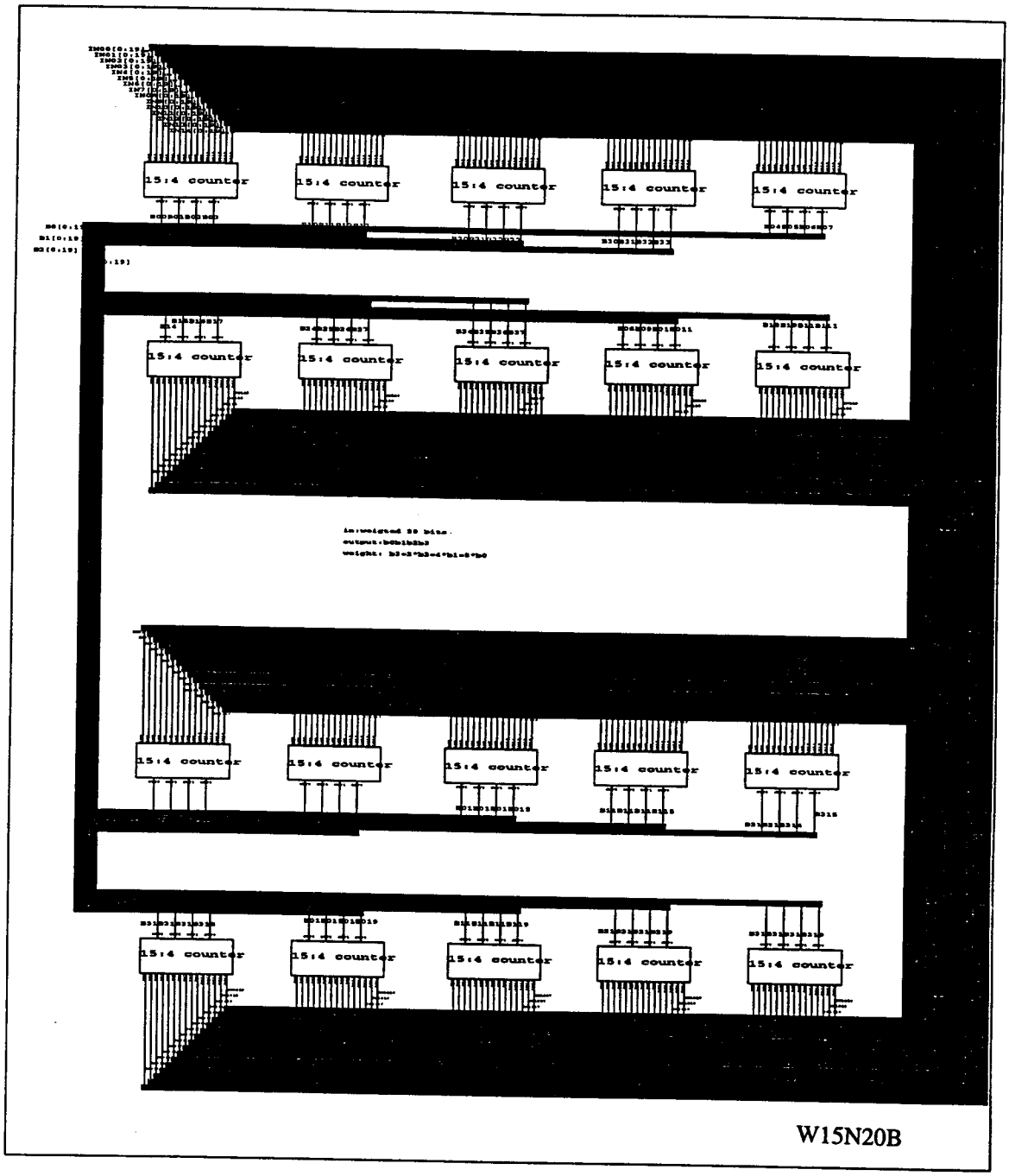


Figure C.16 W15N20B

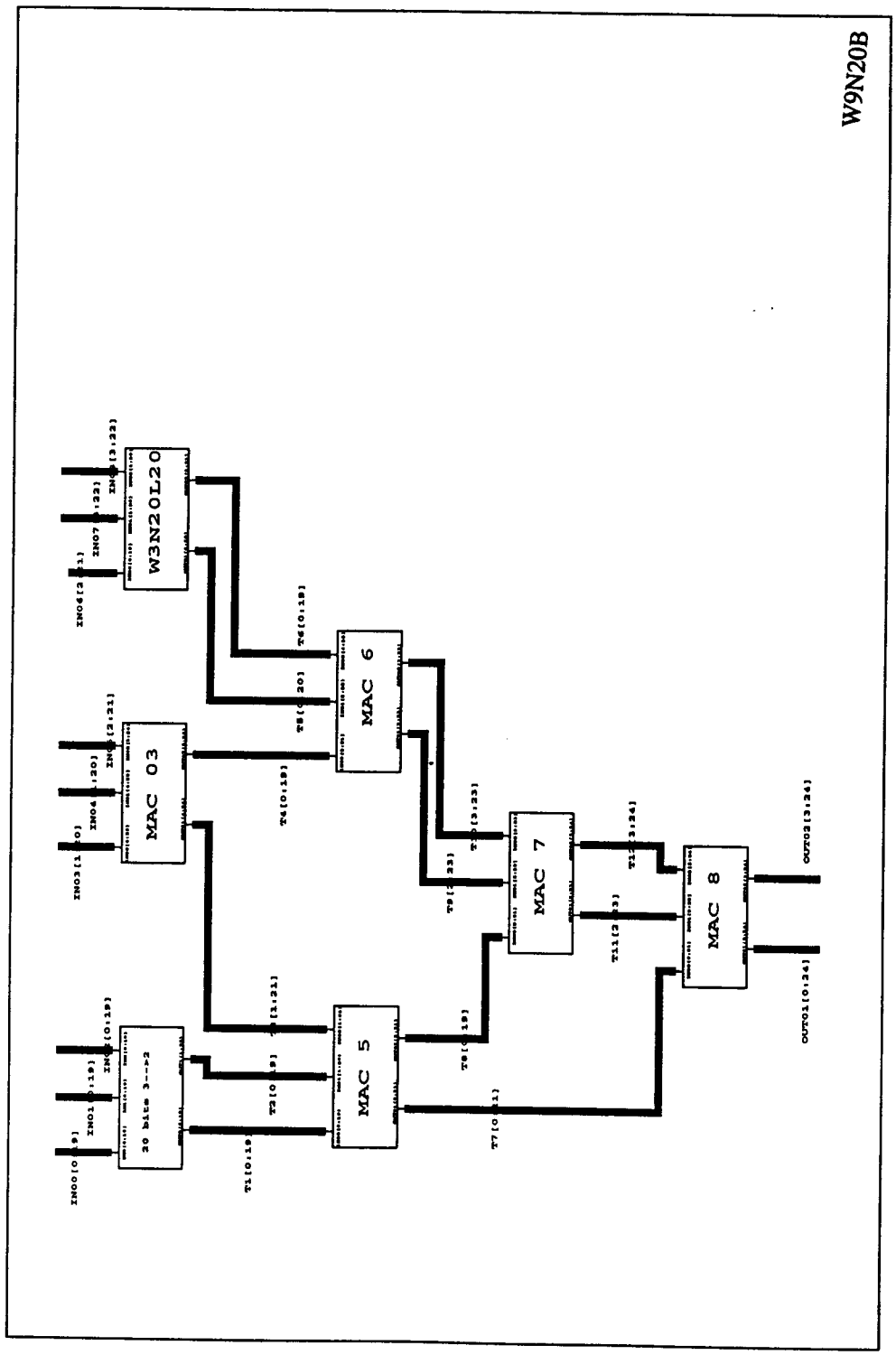


Figure C.17 W9N20B

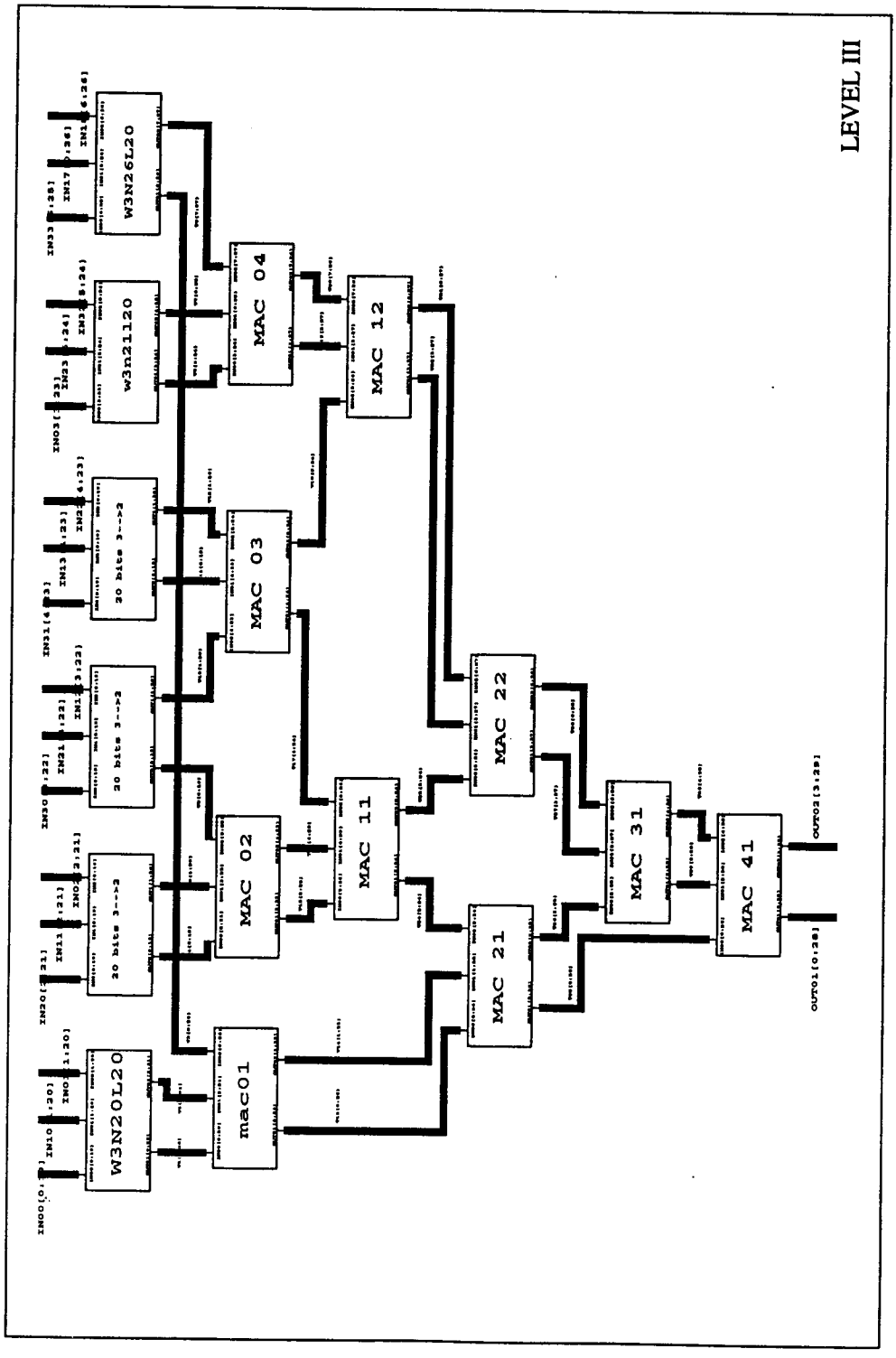


Figure C.18 Level III

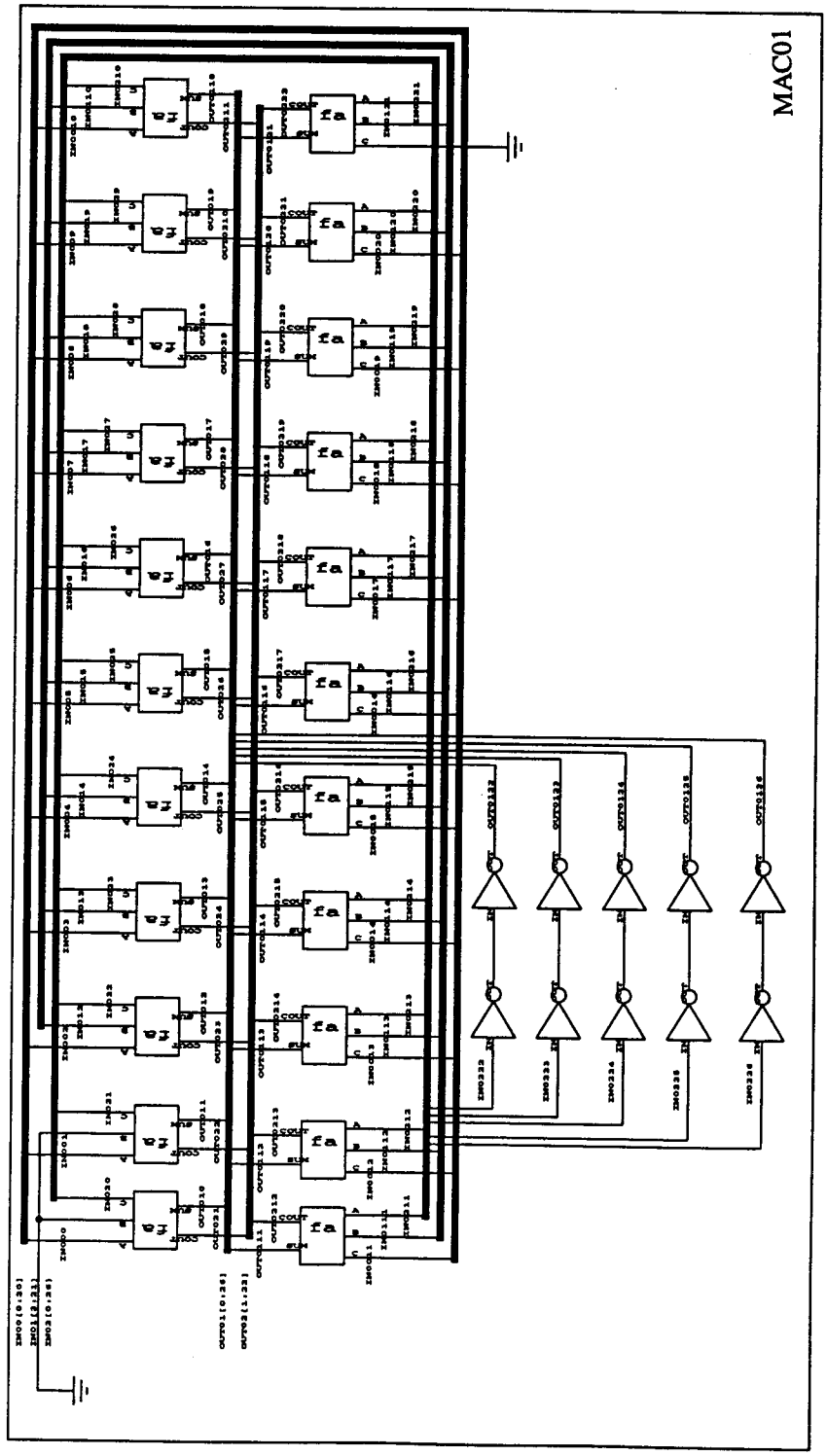


Figure C.19 MAC 01

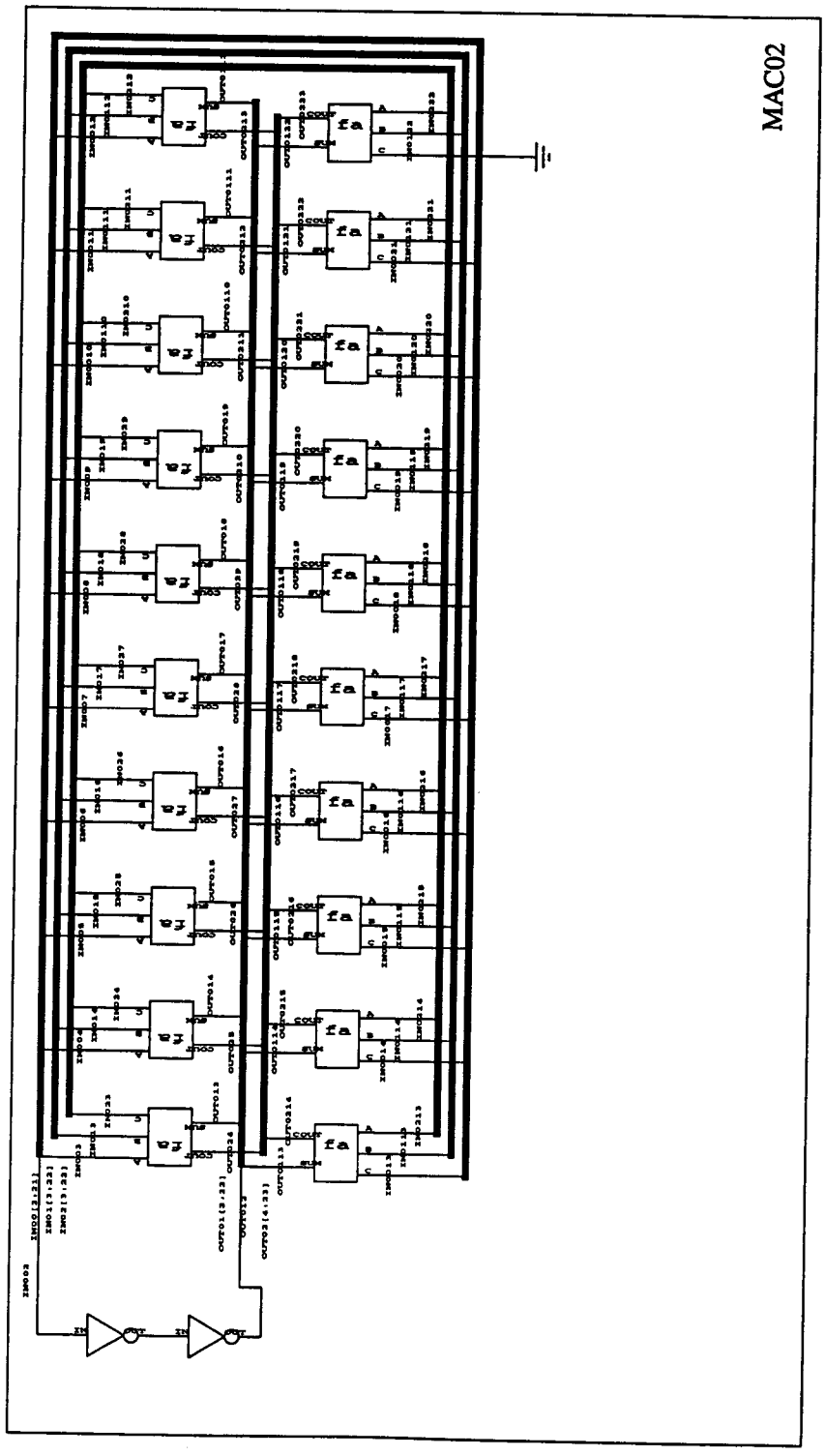


Figure C.20 MAC 02

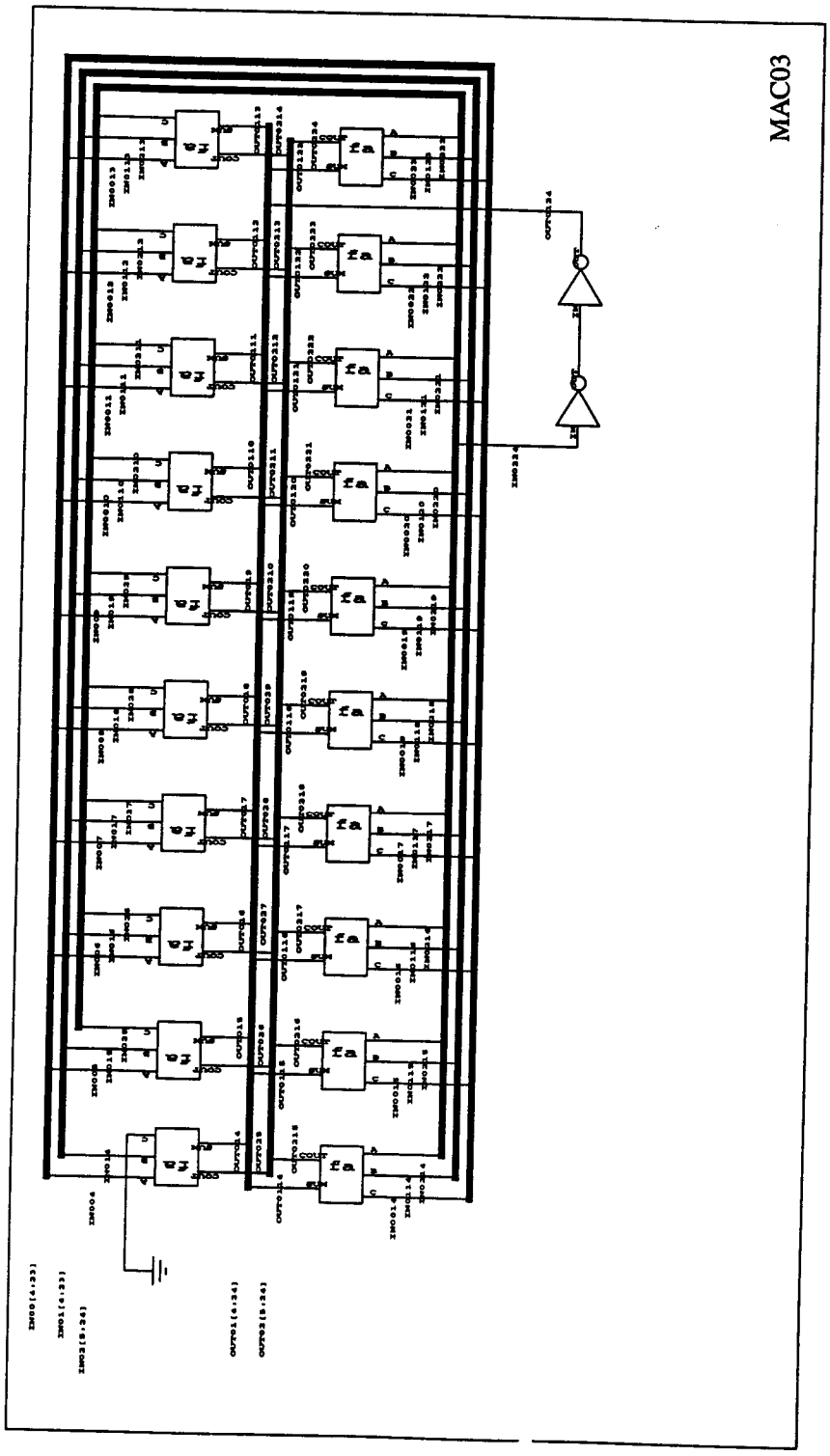


Figure C.21 MAC 03

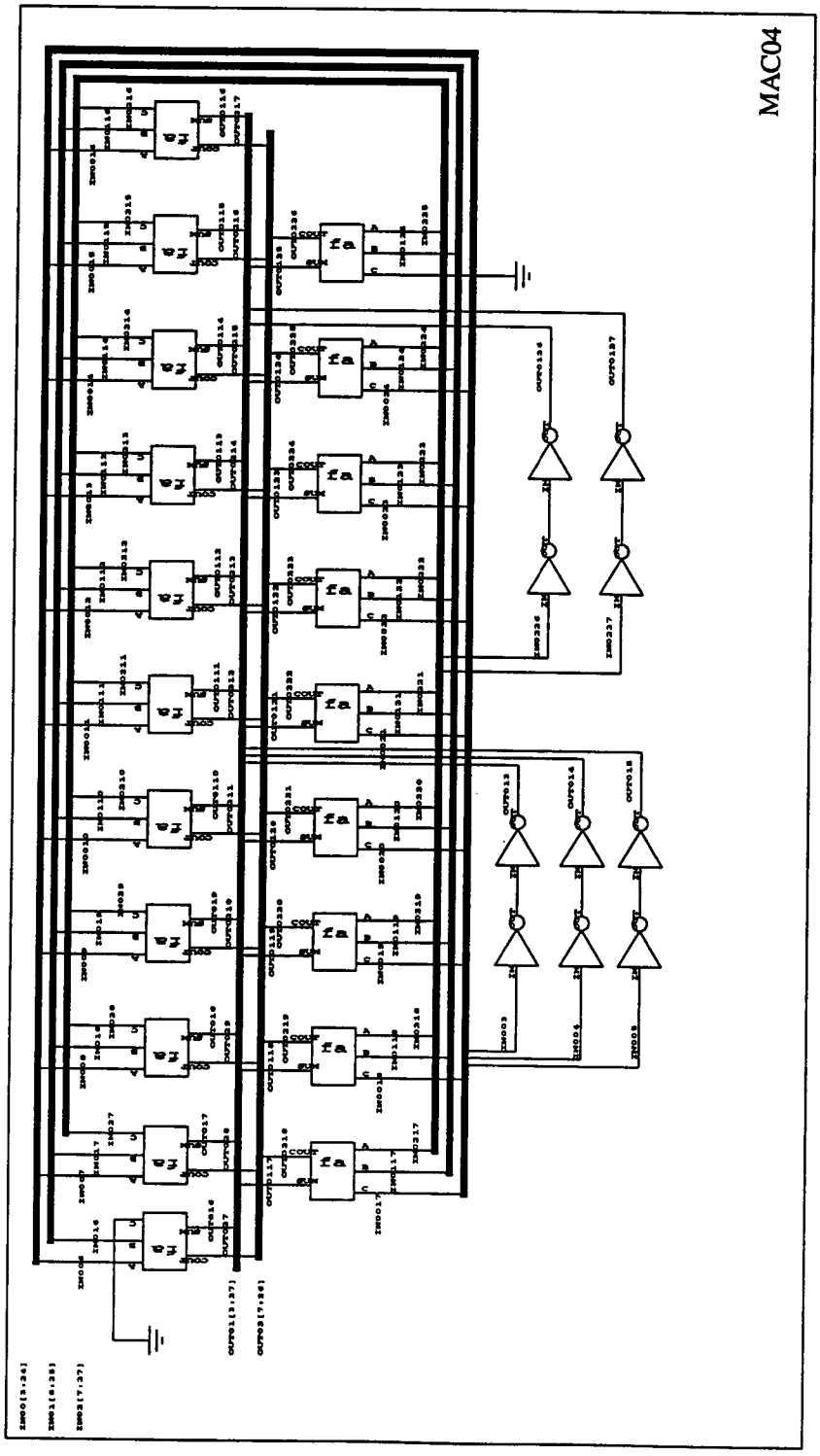


Figure C.22 MAC 04

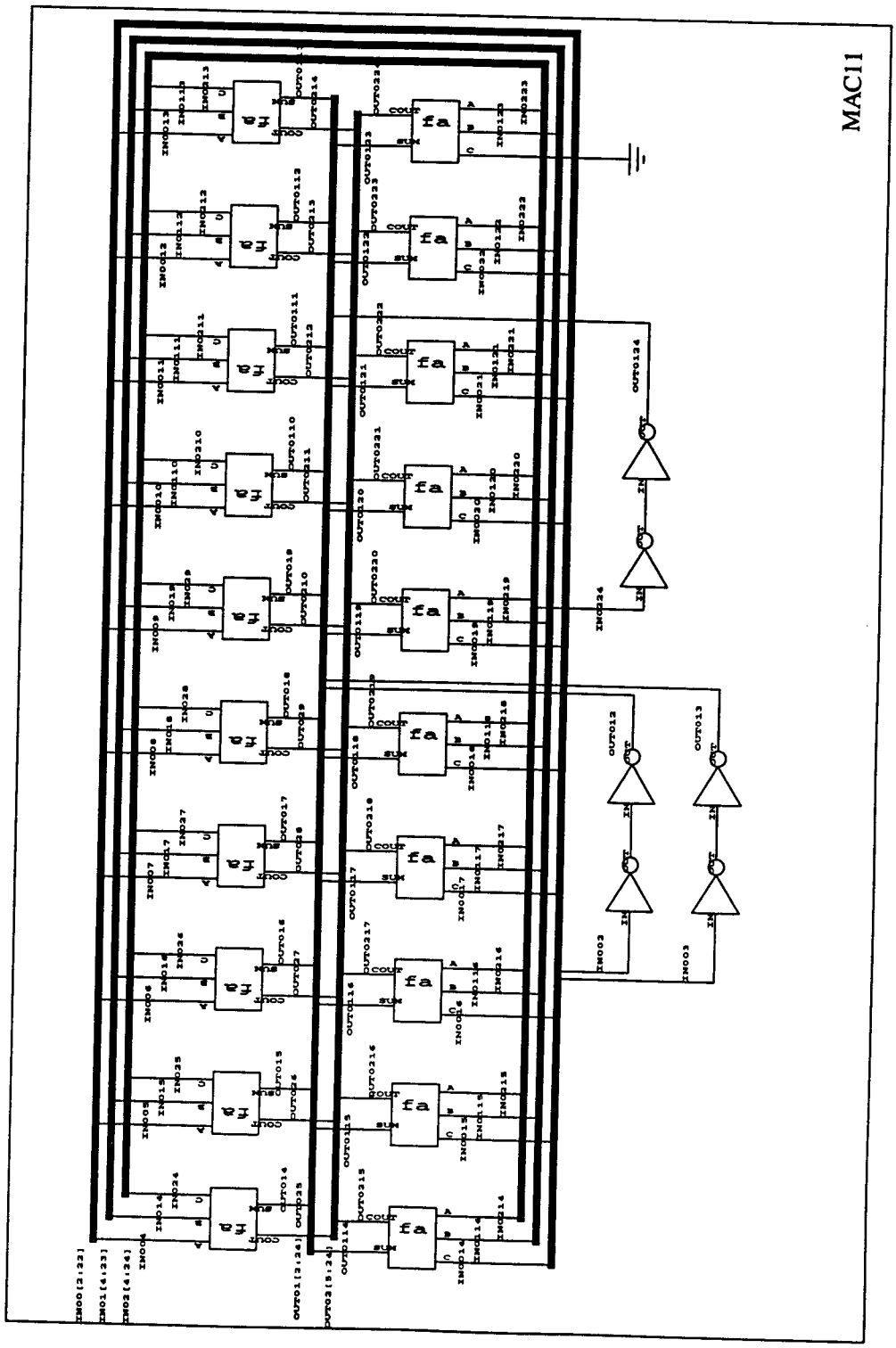


Figure C.23 MAC 11

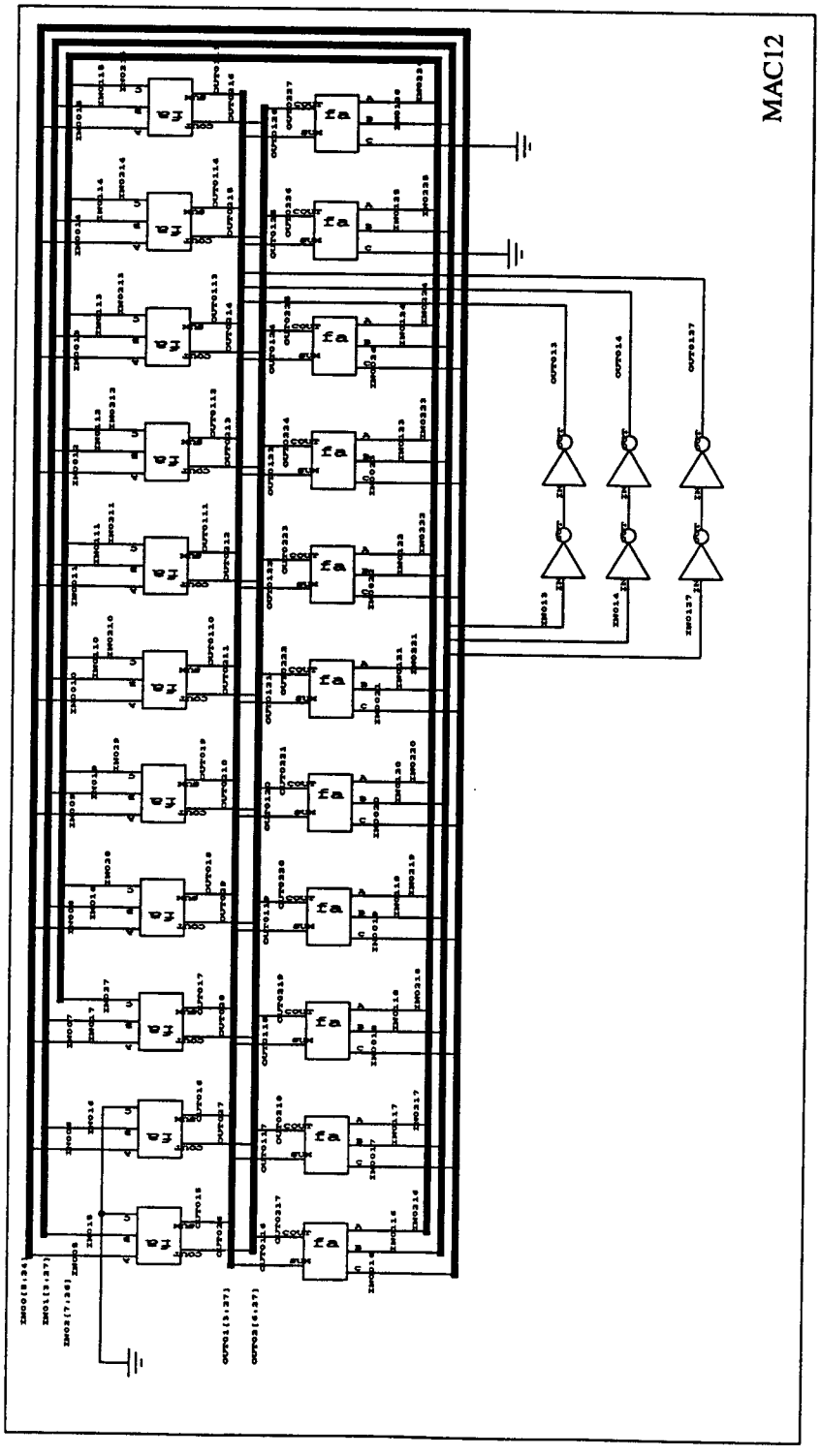


Figure C.24 MAC 12

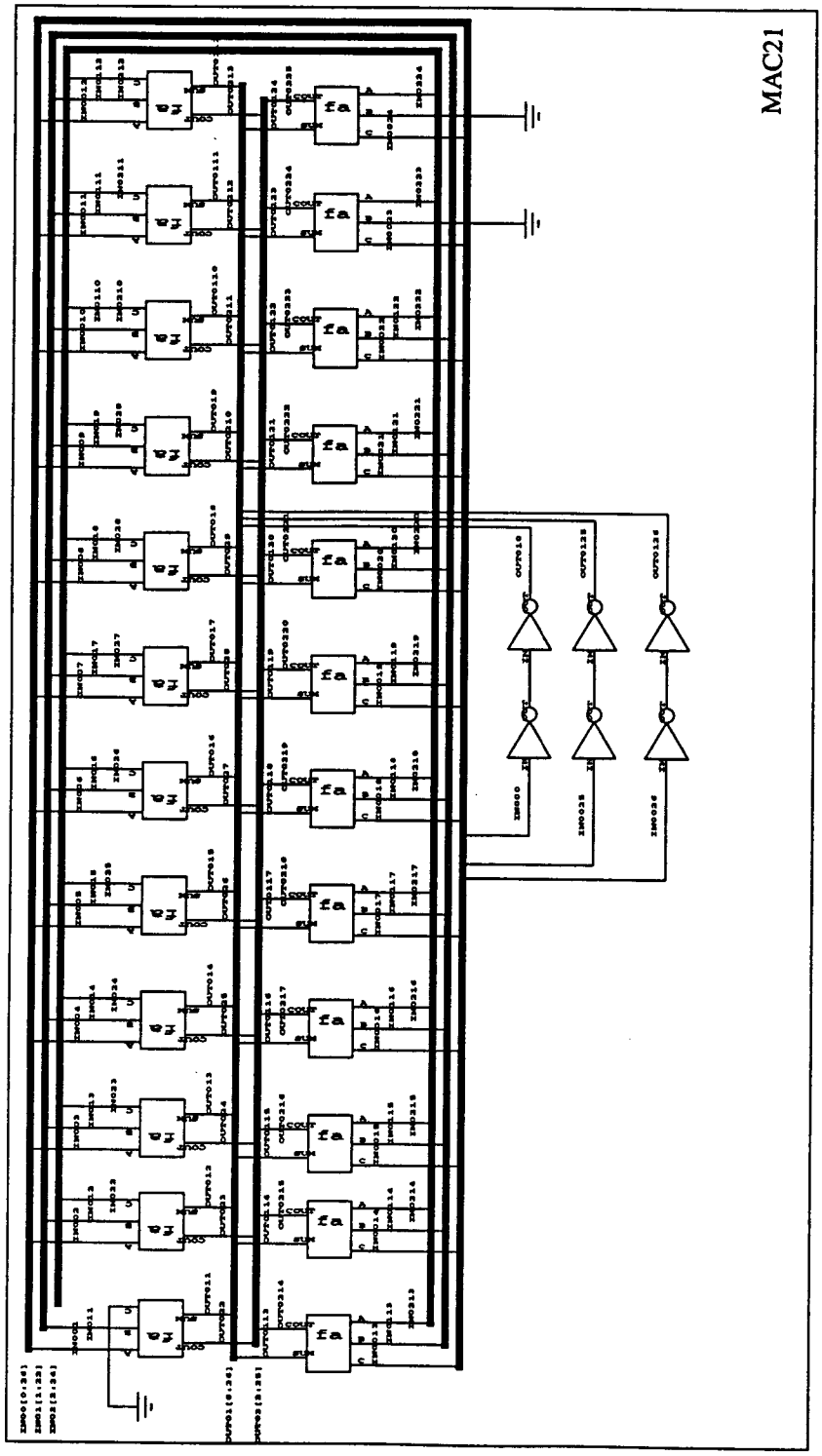


Figure C.25 MAC 21

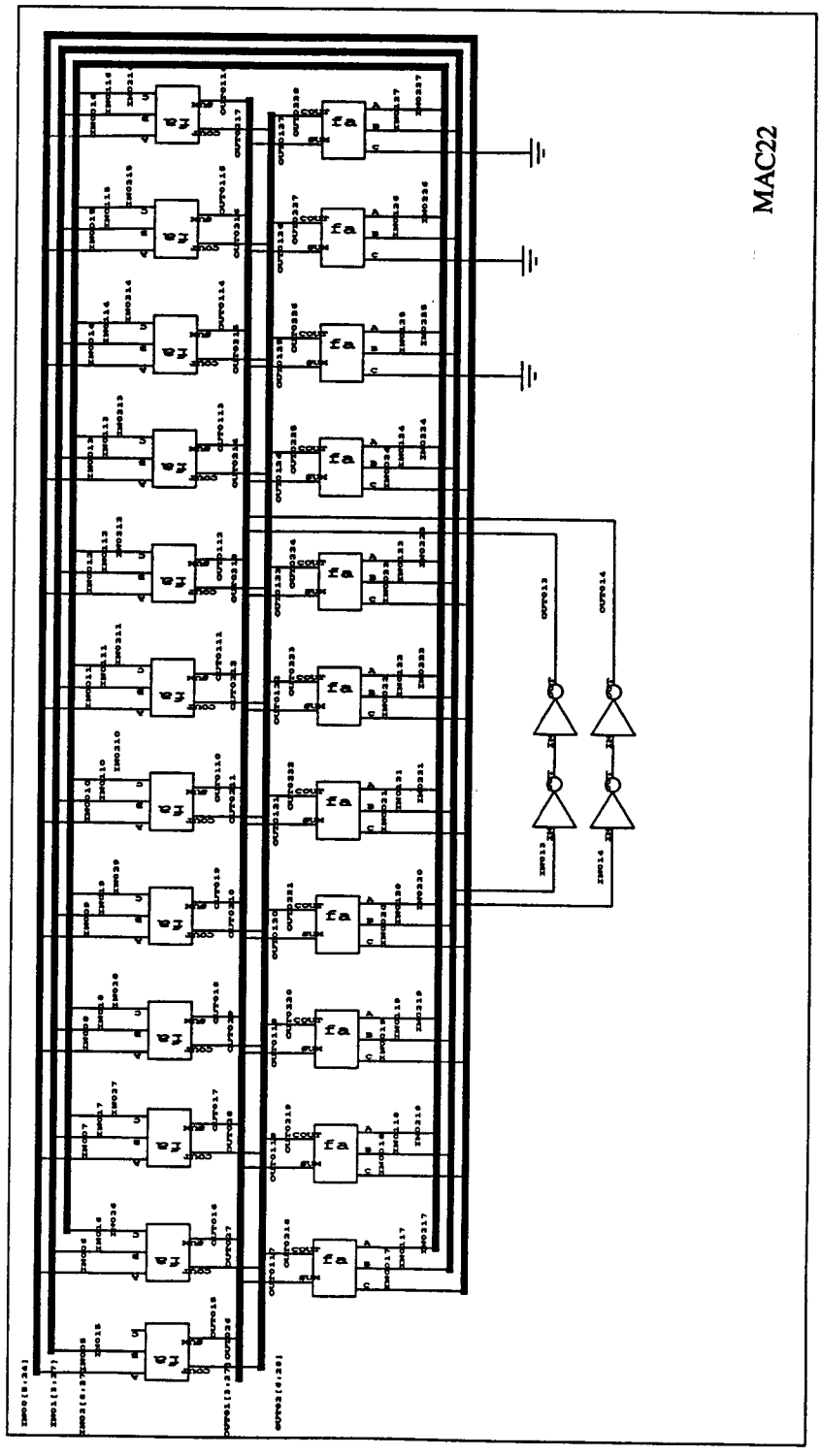


Figure C.26 MAC 22

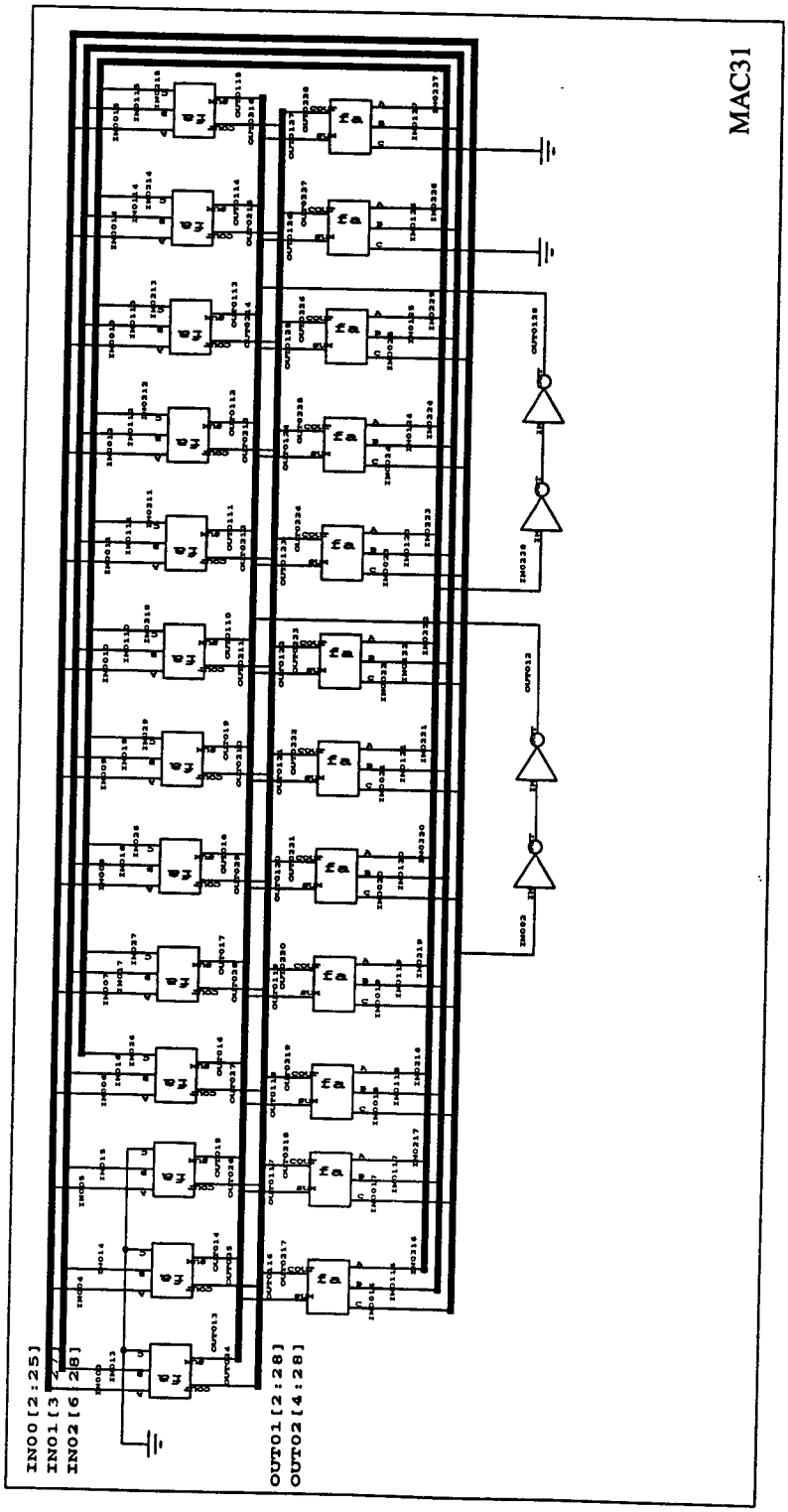


Figure C.27 MAC 31

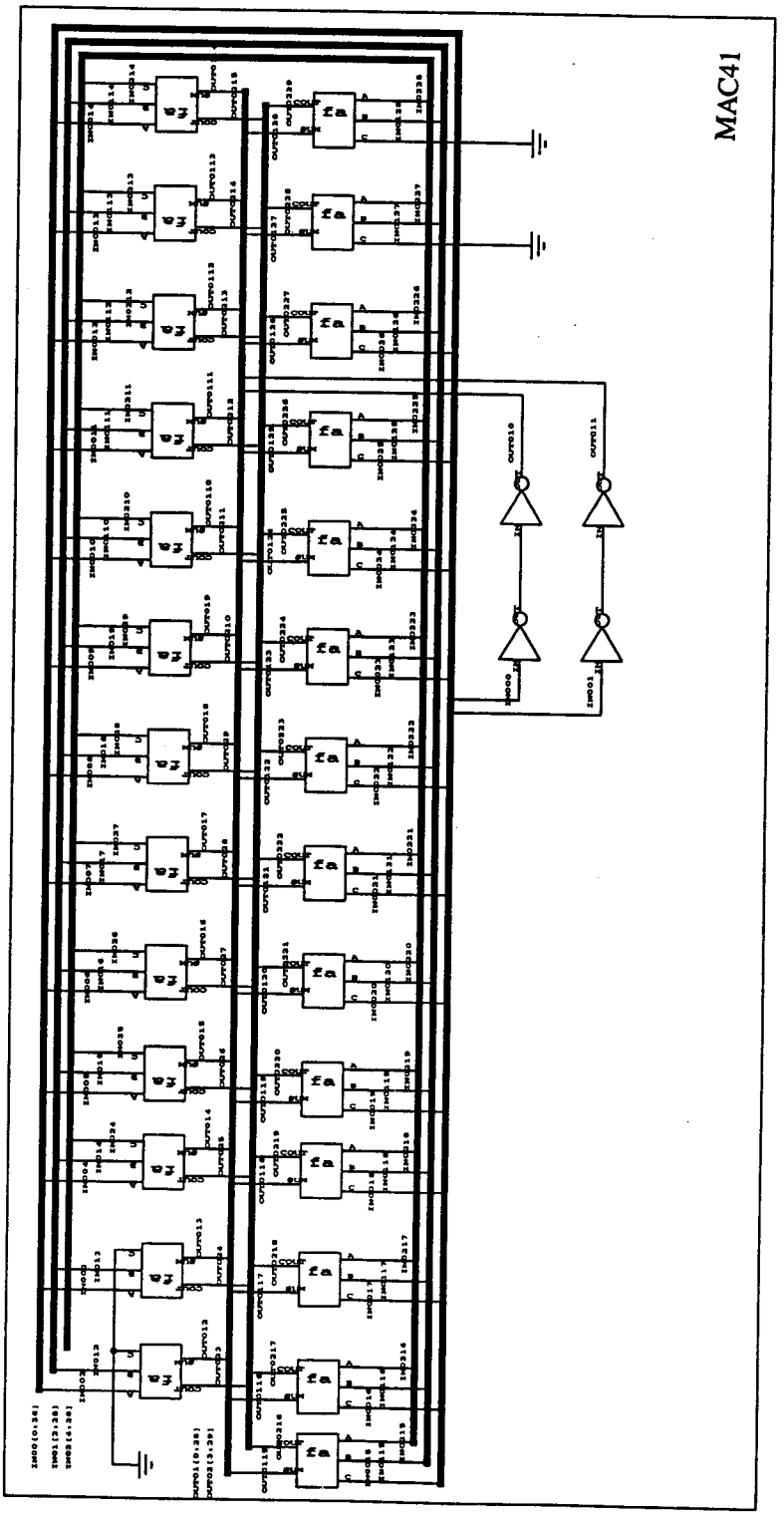


Figure C.28 MAC 41

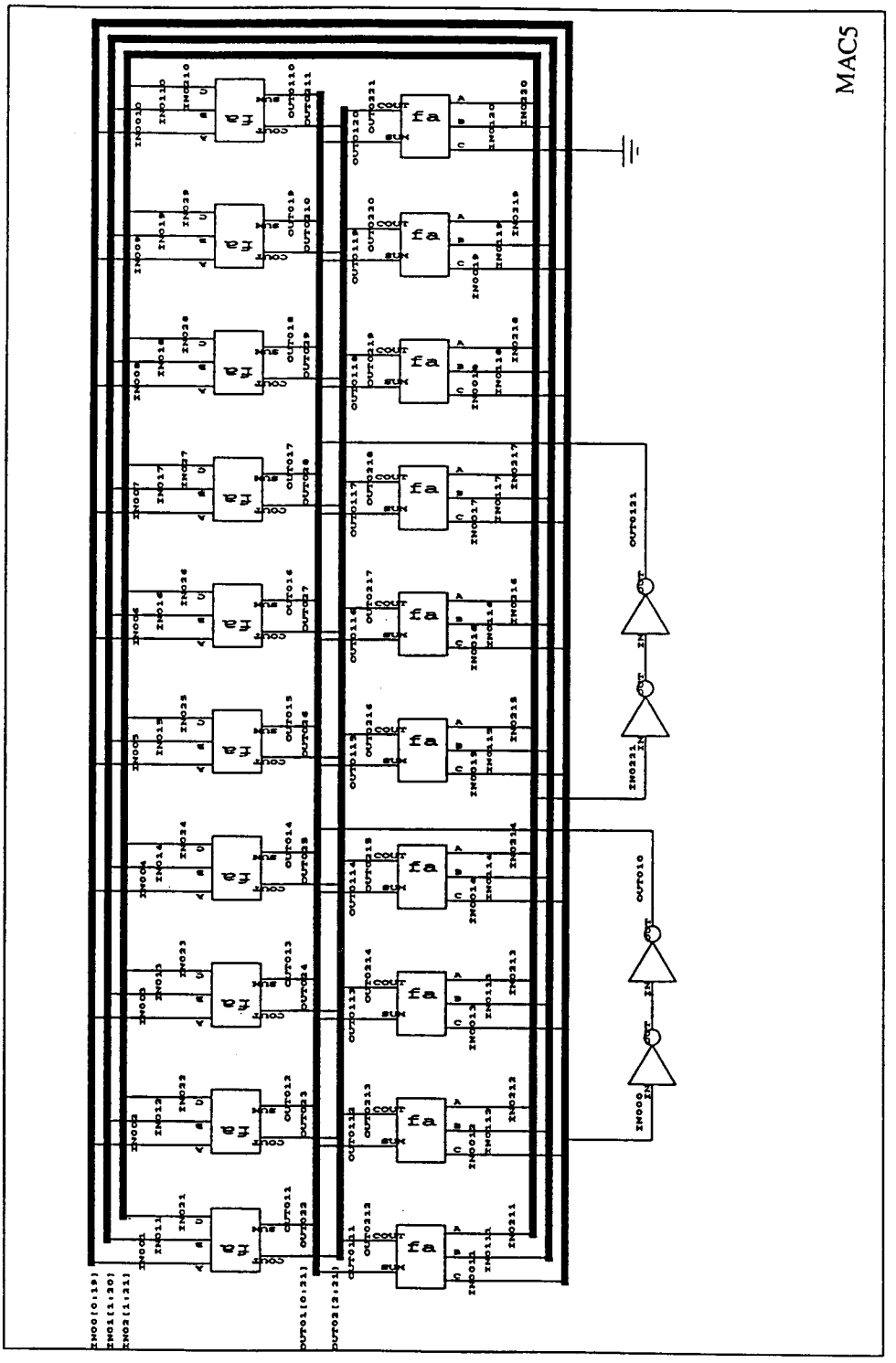


Figure C.29 MAC 5

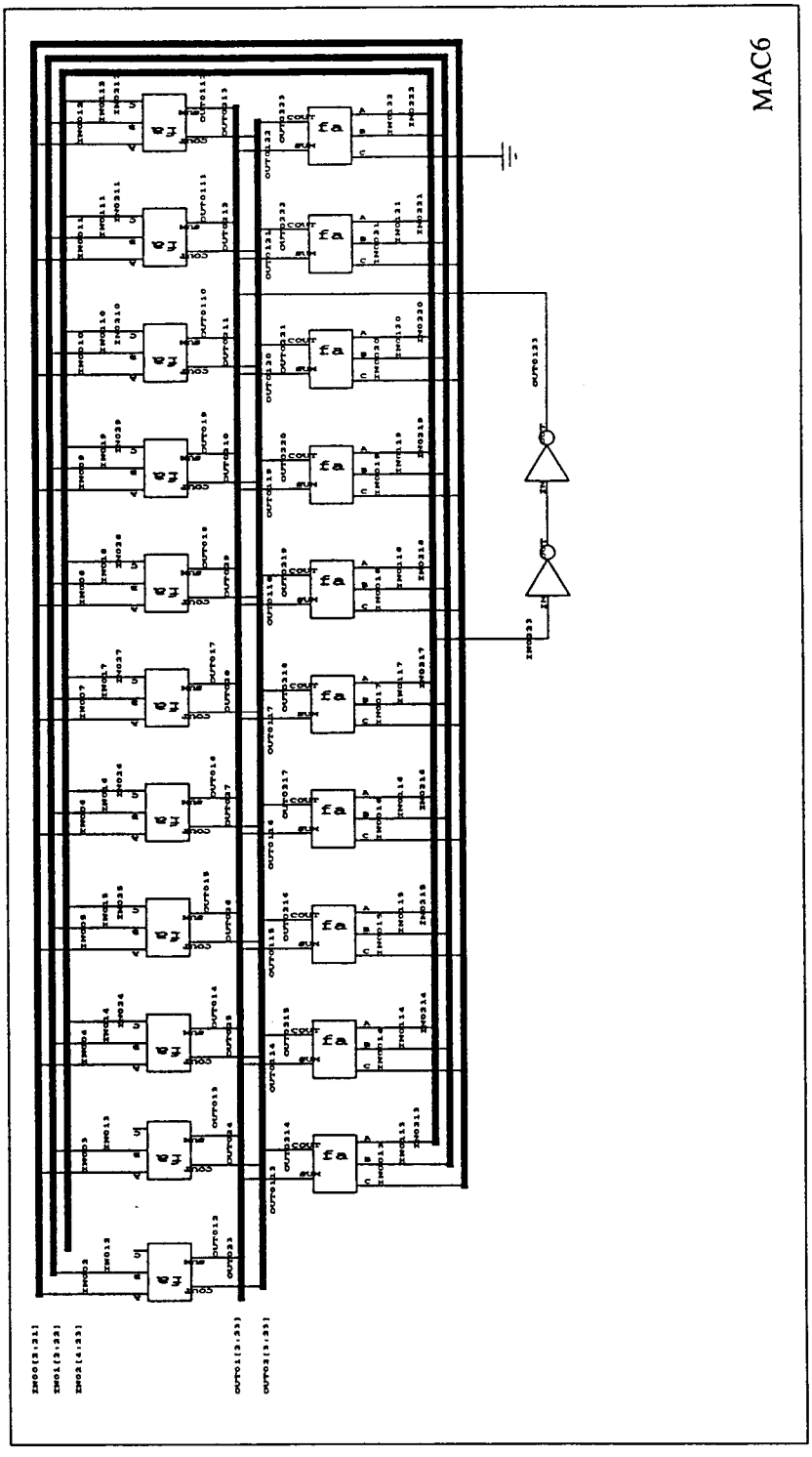


Figure C.30 MAC 6

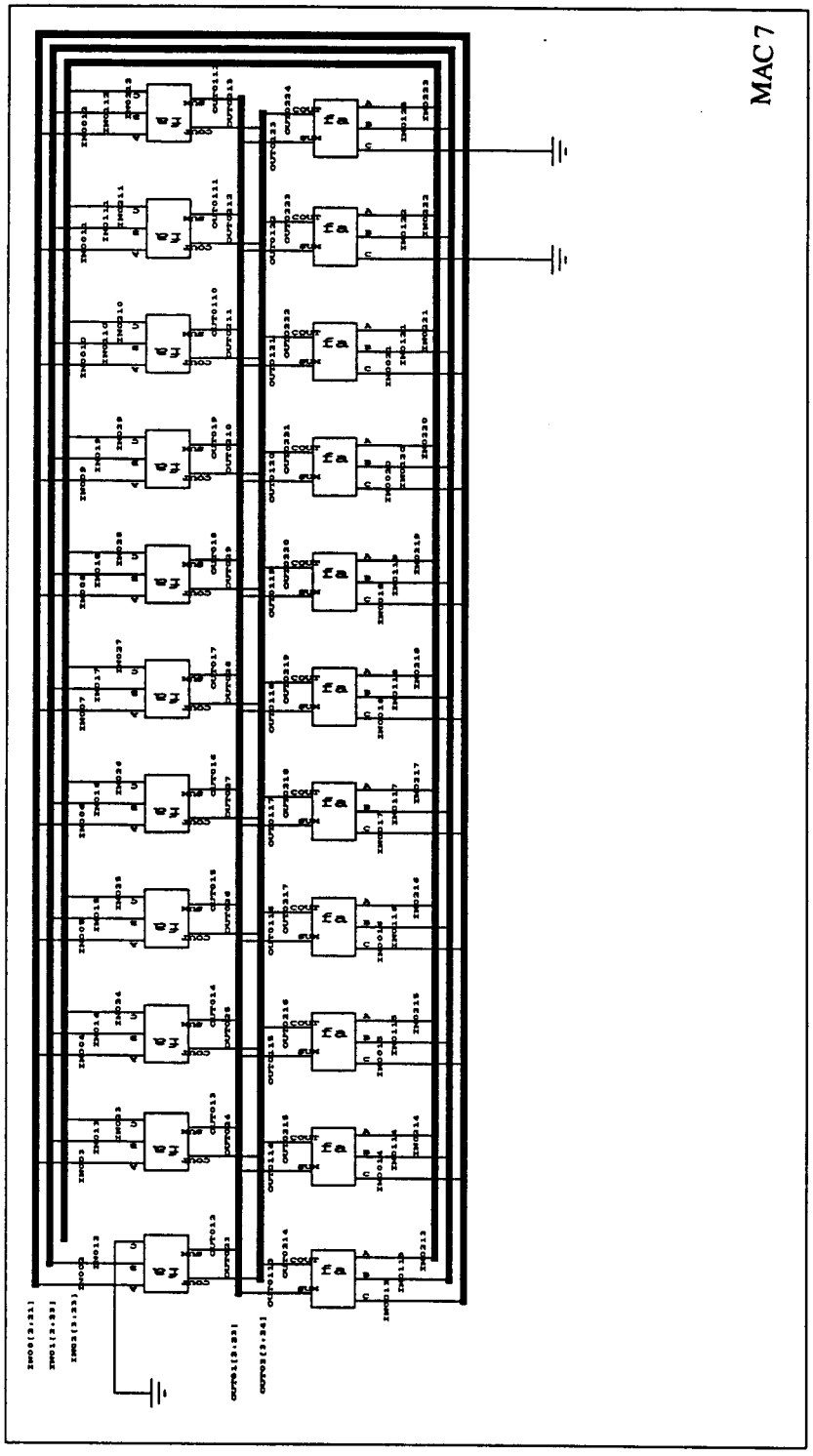


Figure C.31 MAC 7

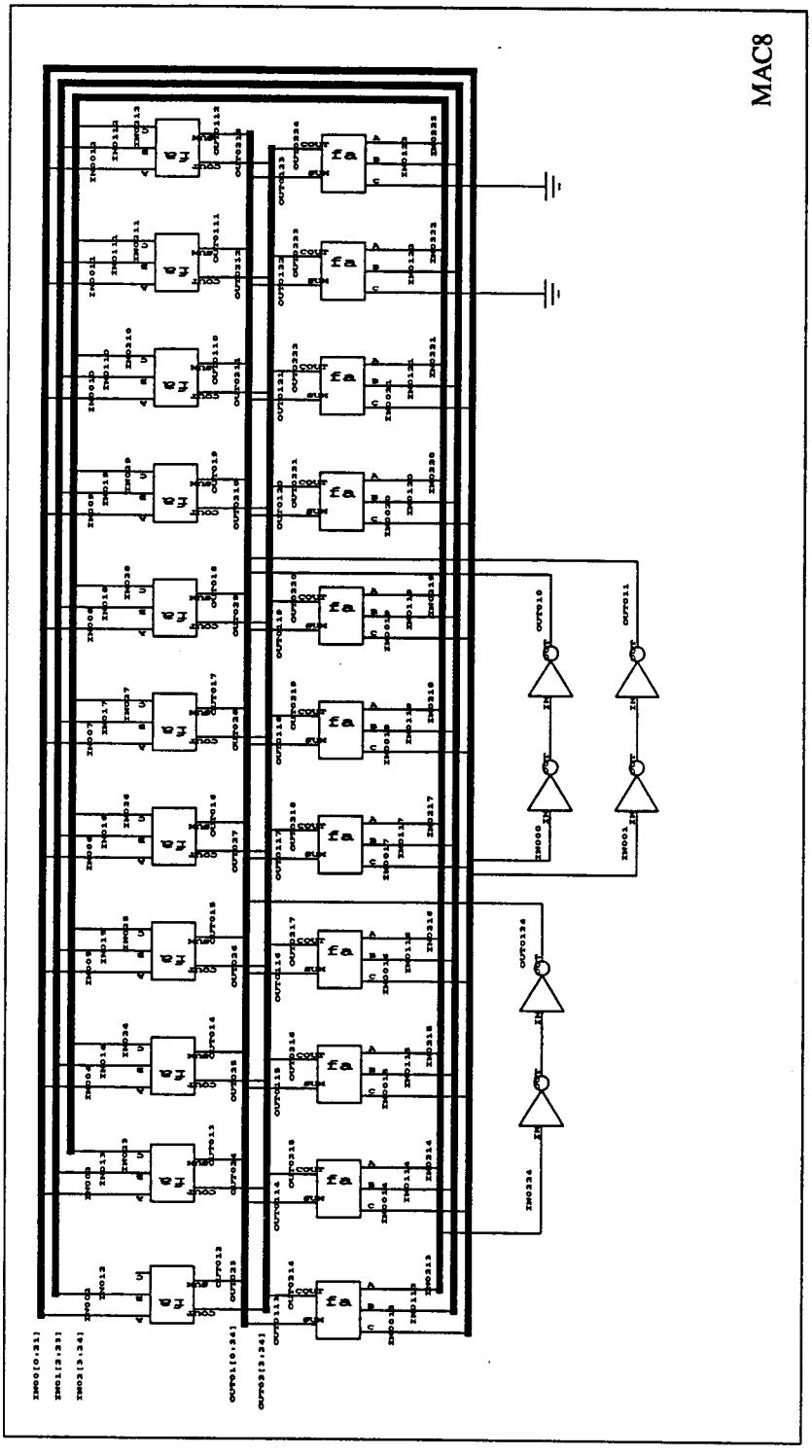


Figure C.32 MAC 8

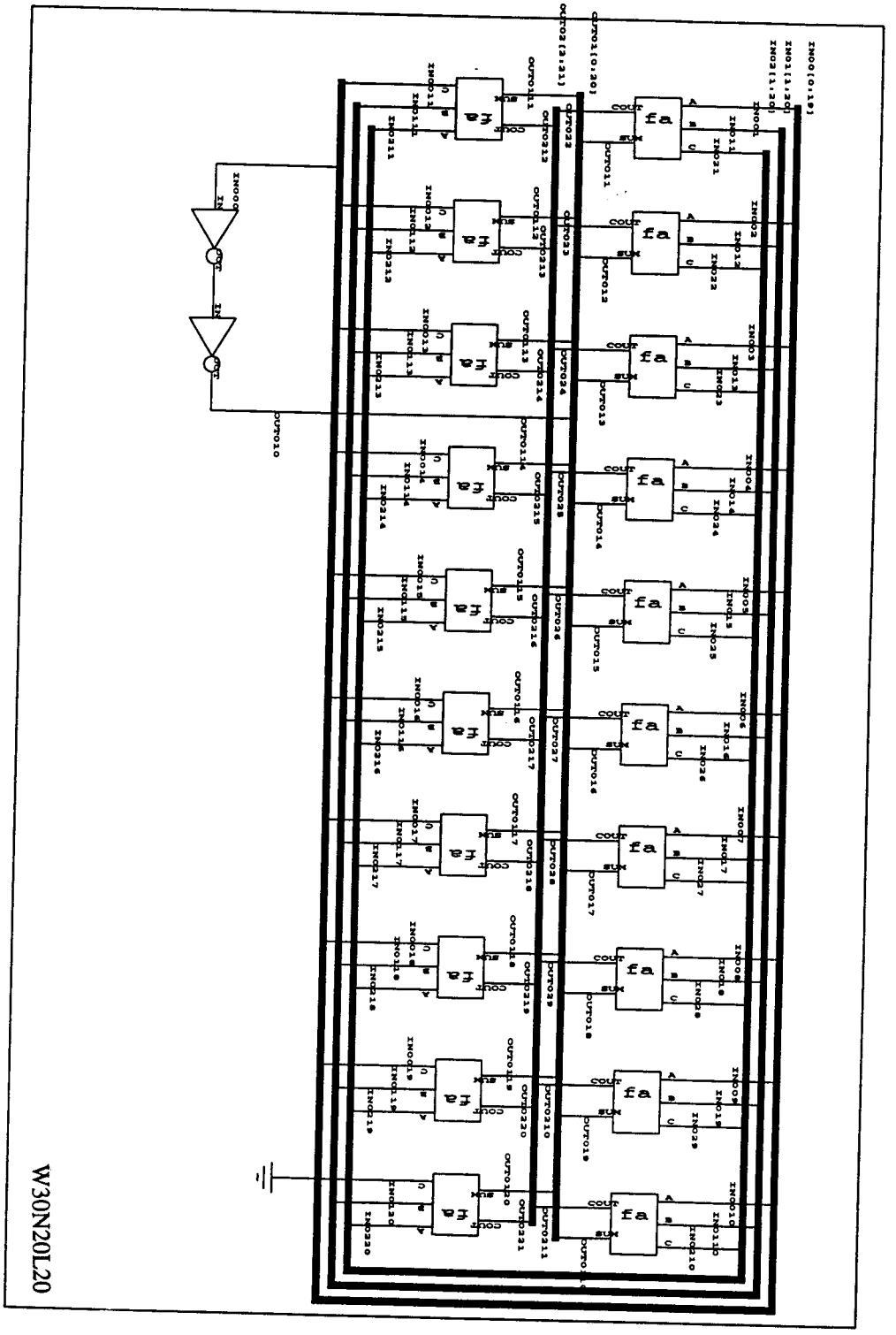


Figure C.33 W30N20L20

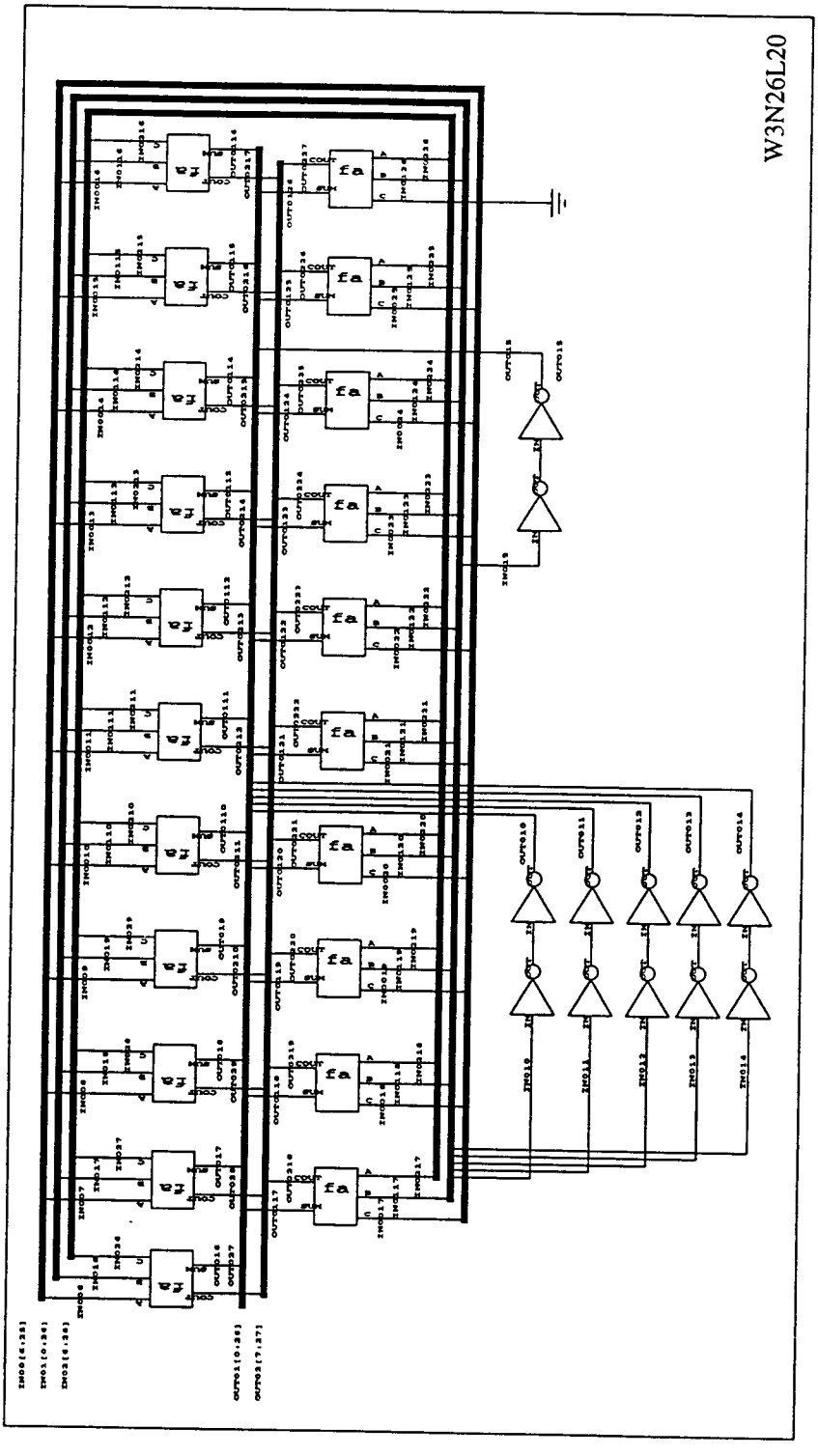


Figure C.34 W3N26L20

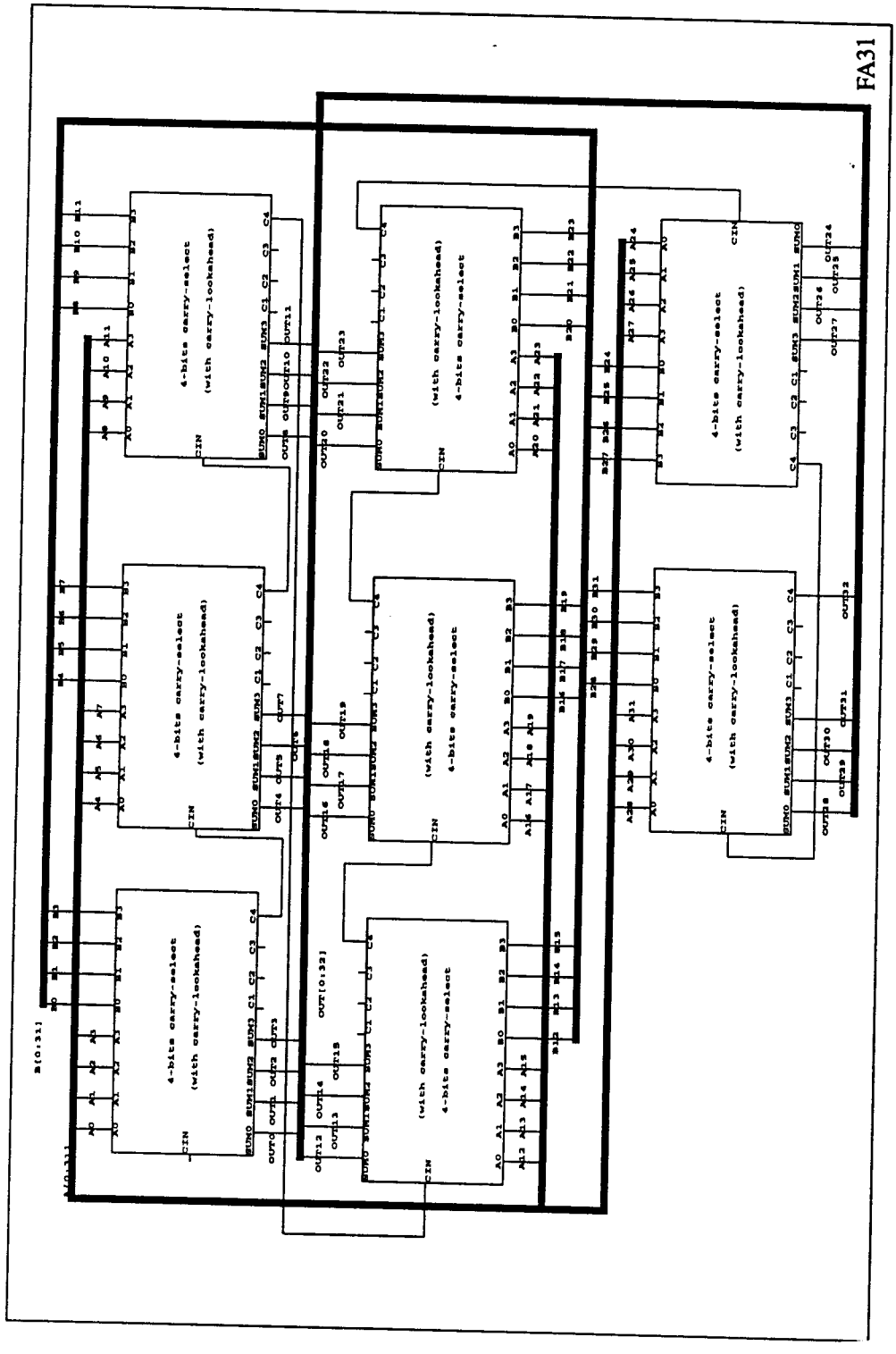


Figure C.35 28-bit Carry-Look-Ahead & Carry-Select Adder

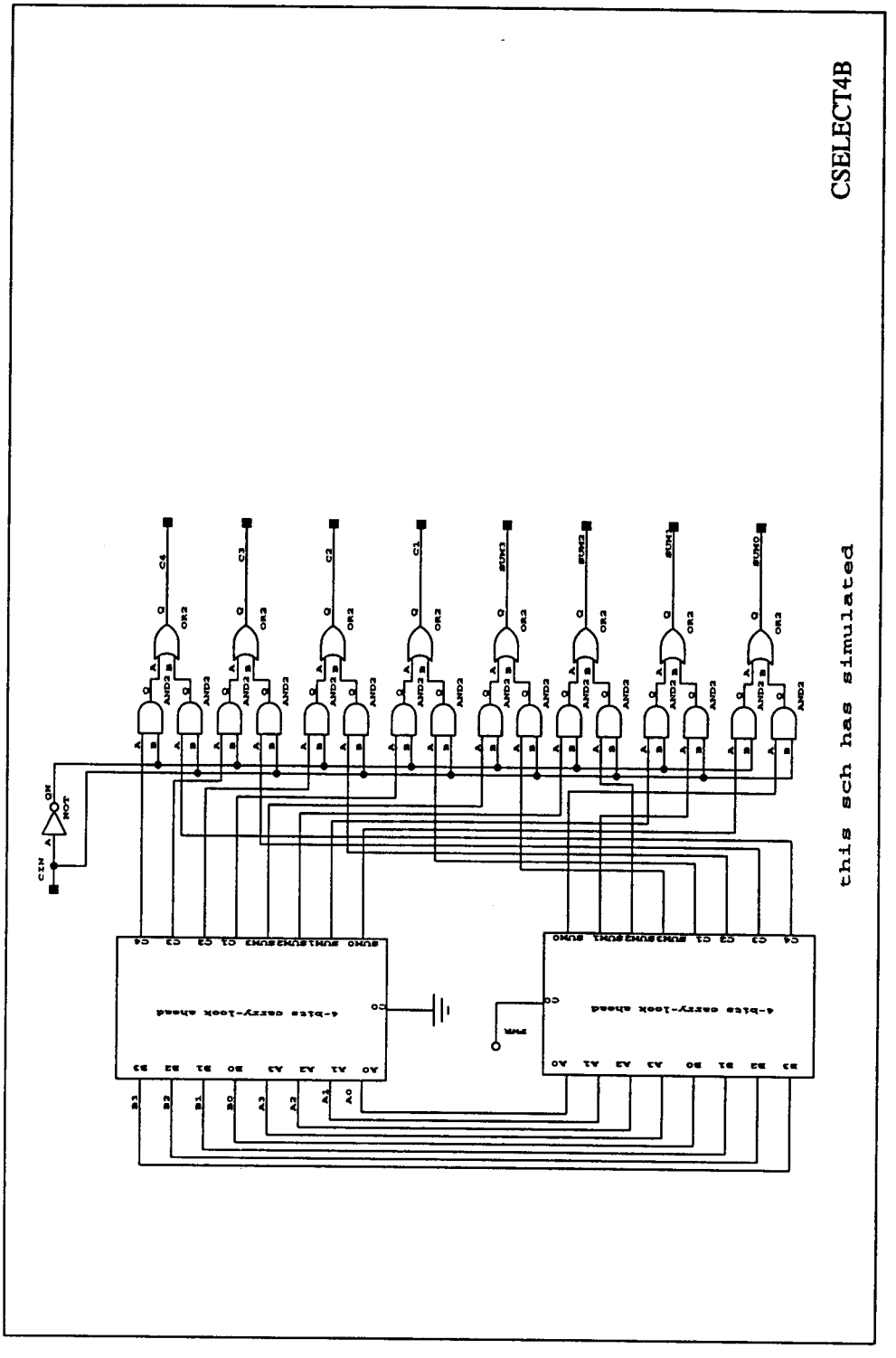


Figure C.36 4-bit Carry-Select Adder

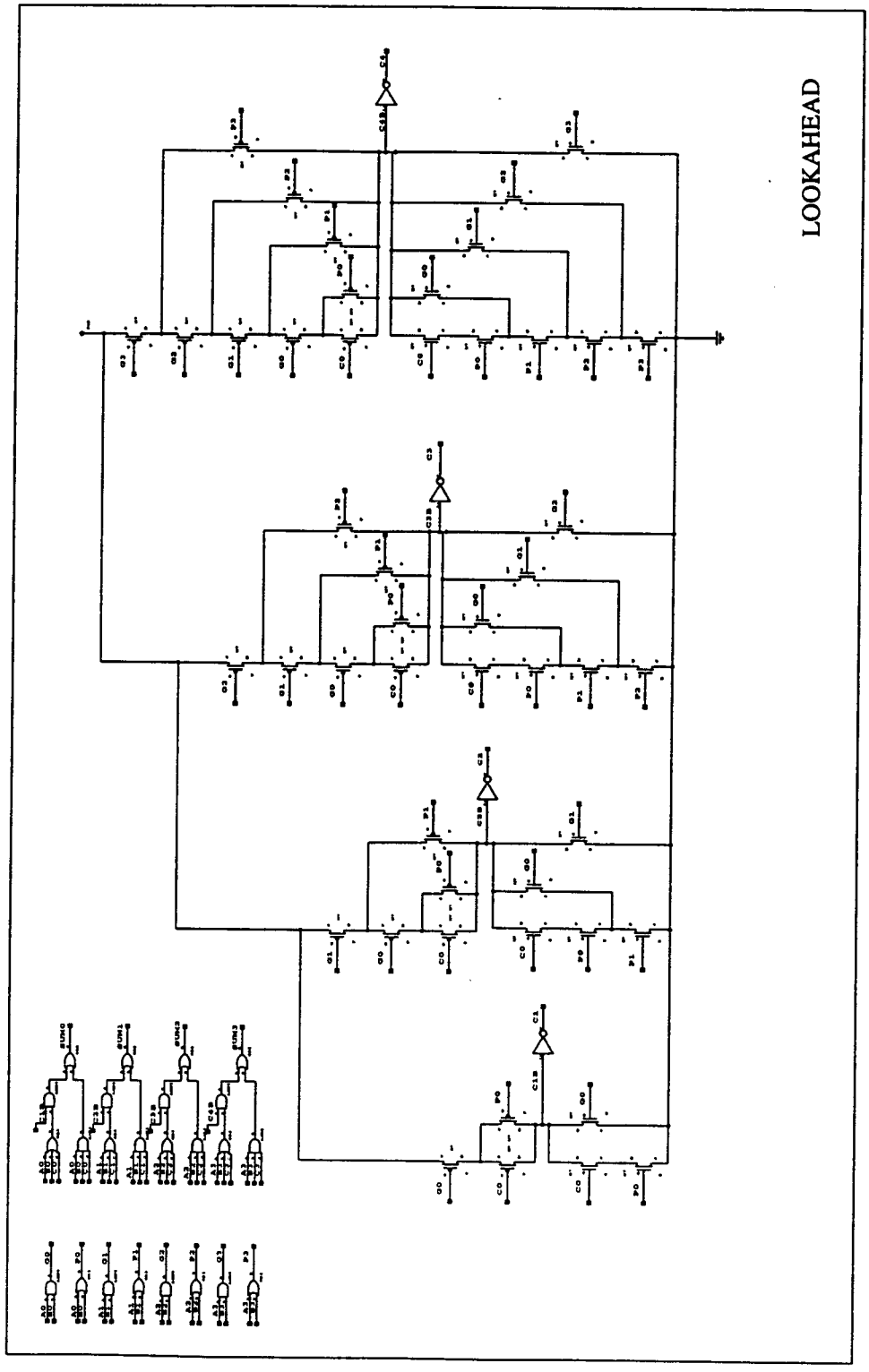


Figure C.37 4-bit Carry-Look-Ahead Adder