EVALUATION OF ARRAY TYPE LAYOUTS FOR CELL LIBRARY GENERATION

by

Chi-Min Chu

Memorandum No. UCB/ERL M87/61

1 September 1987

EVALUATION OF ARRAY TYPE LAYOUTS FOR CELL LIBRARY GENERATION

by

Chi-Min Chu

Memorandum No. UCB/ERL M87/61

1 September 1987

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

TITLE PAGE

EVALUATION OF ARRAY TYPE LAYOUTS FOR CELL LIBRARY GENERATION

by

Chi-Min Chu

Memorandum No. UCB/ERL M87/61

1 September 1987

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

ACKNOWLEDGEMENTS

I would like to thank my research adviser, Professor Robert Brodersen, for his helpful suggestions and valuable comments on this work. I would also like to thank all my colleagues. Their useful discussions made my research progress more rapidly.

Research supported by DARPA Contract N00039-87-C-0182

ABSTRACT

The importance of the reduction of design time motivates a search for an alternative to traditional custom design. Array type layout creates a loss in circuit density, but greatly simplifies the effort for the designers. In addition, many desirable features are gained by using array type layout such as technology independence and fast turnaround for fabrication. In this report, we first set up the criteria for evaluation, then we investigate some prototype array layouts. These layouts include gate matrix, sea of gates, standard cell design, and some other new arrays. Basic design methodologies are discussed for each design. Simulations and test chips are made to verify the electrical performance. At the same time, we propose new design methods to improve the performance of each design. Relevant design problems, such as CAD tool usage and layout strategies, are also discussed. Finally, we make a comparison of all the array-type layouts and give recommendations based on the comparison. The study shows that sea of gates design has a number of advantages over the other designs. Also, the laser restructurable techniques are particularly interesting due to their very short turnaround time.

TABLE OF CONTENTS

EVALUATION OF ARRAY TYPE LAYOUTS FOR CELL LIBRARY GENERATION

by

Chi-Min Chu

Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
Berkeley, California 94720

CHAPTER 1

INTRODUCTION

1.1. Why Array-Type Layouts

Custom VLSI (Very Large Scale Integration) design achieves high density and high performance layouts, but at the cost of a long design time. Whenever there is a technology change, we have to redesign all the cells in the cell library. From the design experience of LAGER[1] cell library, we realize that we need to adopt a new layout style to attain the following goals:

(1) To simplify and unify layout procedure by using orderly structure.

(2) Whenever there is a technology change, the cell library can be easily updated.

(3) Adequate electrical performance.

(4) Short turnaround time.

(5) Reasonable circuit density.

Array-type layouts, including gate matrix, gate array, standard cell, and sea of gates designs, are investigated in this report. Compared with custom design, all these layout styles ease design procedures and thus help us to reach at least one of the above goals. As technology improves, we will see more and more designs using array type layouts because it is no longer necessary to spend as much time on custom design for the purpose of reducing area. This report discusses the kind of layout that is the most appropriate for a given application. When a new technology is available, we can follow the guidelines from this research and rapidly upgrade our design.

[1] LAGER is short for Layout Generator. It is a set of layout tools for ASICs (Application Specific Integrated Circuits) design. For more information, see [Raba85].

1.2. Array Type Layouts in Use

It is estimated that more than half of all semiconductors sold will be semicustom designs by 1990 [Gold85]. Among them, gate arrays are probably the most widely used layout style today. Over 70 vendors currently provide gate arrays, and improvements are being made to make gate array design more attractive. The sea of gates approach is basically a form of gate array. It makes use of multi-level of metal for routing so that all the routing can be built on top of cells. That is to say, there is no distinction between cell arrays and routing channels as are distinguished in gate arrays. With this feature, sea of gates design is more flexible and able to achieve denser layouts. Sea of gates design will be discussed in length in chapter 3. Design decisions are described and test results as well as simulation results are presented to support these conclusions.

Gate matrix design was first brought out by Lopez and Law in 1980 [Lope80]. The concept was to transpose the transistors in the logic cells of standard cell design onto the wiring channels. Due to the regular structure, the gate matrix design simplifies layout procedures and achieves the feature of technology independence. Chapter 2 will concentrate on the gate matrix design. Additional features will be elaborated and test results on a test chip will be presented.

Standard cell design is a structured type design. Though not strictly an array, it has many of the same characteristics. Compared with gate arrays, standard cell design has no predefined channels thus has more flexibility and a higher utilization of chip area. Compared with custom design, the complete and reusable standard cell library greatly shortens design time. Moreover, many CAD tools have been developed to aid standard cell design. That is why we regard standard cell design as another alternative. We are going to discuss more about standard cell design in Chapter 4 in which the basic design methodology will be described as well as simulation results.

New types of arrays are being developed. Even laser technology is being used in cell design to reduce turnaround time. In this report, we will cover a number of the basic array type layouts. Modifications of these layout styles may be made, but the essence would remain.

1.3. Prototype Datapath Used for Evaluation -- LAGER AAU

To understand the tradeoffs between each layout style, we built some test chips for evaluation. The prototype datapath we chose was LAGER Address Arithmetic Unit (AAU). The reasons for choosing this datapath are as follows:

(1) It is a real and useful datapath.

(2) It contains basic logic gates and latches, which allows measurements of the electrical performance.

(3) Routing of the AAU is not too complicated to handle.

The LAGER AAU designed by Pope [Pope85] contains three major parts: counters, decode sections, and adders. The schematic of a single bit slice for an AAU is shown in Figure 1.1 which includes an IX section and an IY section. Other possible organizations are IX-section-only structure and IY-section-only structure. IX sections are used for subprogram address calculation, while IY sections are used for main program address calculation. The differences between these two sections are, first, IX counters must be set to -1 during the main program, 0 during the first iteration of subprograms and so forth; IY counters must be reset to 0 when initialized or used as registers. Secondly, the control slices for IX sections and IY sections receive different control signals and have different functions. When there is no main program, an IX counter can also be reset to 0 to cause the subprogram to start immediately.

The basic structures of IX counters and IY counters are the same. Figure 1.2 shows the counter bit-slice schematic. In this figure, DIN is connected to GND when the counter is reset to 0, Vdd when the counter is reset to -1, and YINPUT when the counter is used as a register. For even bits, carry-in of the half adder is active low and carry-out is active high. For odd bits, carry-in is active high and carry-out is active low. The advantage of this design is that it saves one gate delay for each carry propagation. The same technique is also used in the adder and the decoder design. A two-phase non-overlapped clock is used for the AAU design. When clock1 is high, input data (DIN) or evaluation results of the half adders are loaded into the D latches, when clock2 is high, the data come out of the latches and the half adders start next evaluation. LOAD and COUNT signals are complement to each other. When LOAD is high, the counters are loaded into data or reset to the initial value. When COUNT is high, the counters count up 1 if CIN* of the least significant bit is active low, and thus perform a counter function.

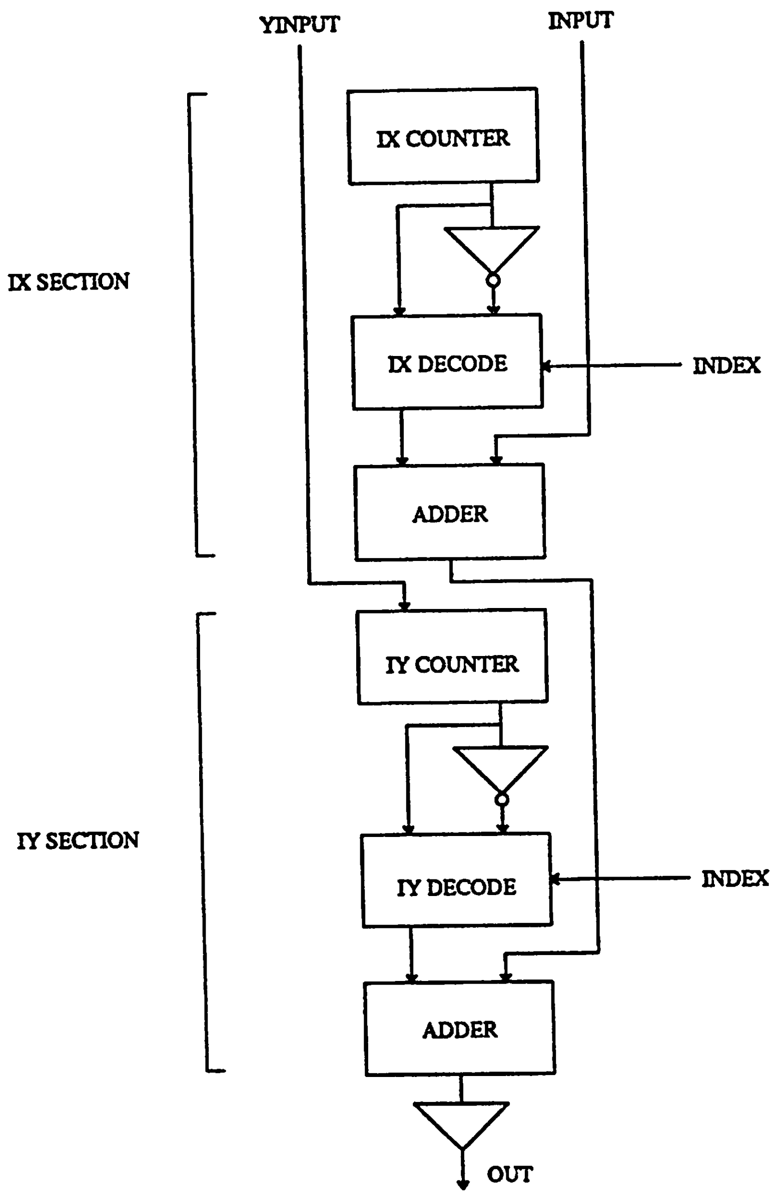


Figure 1.1 AAU bit-slice

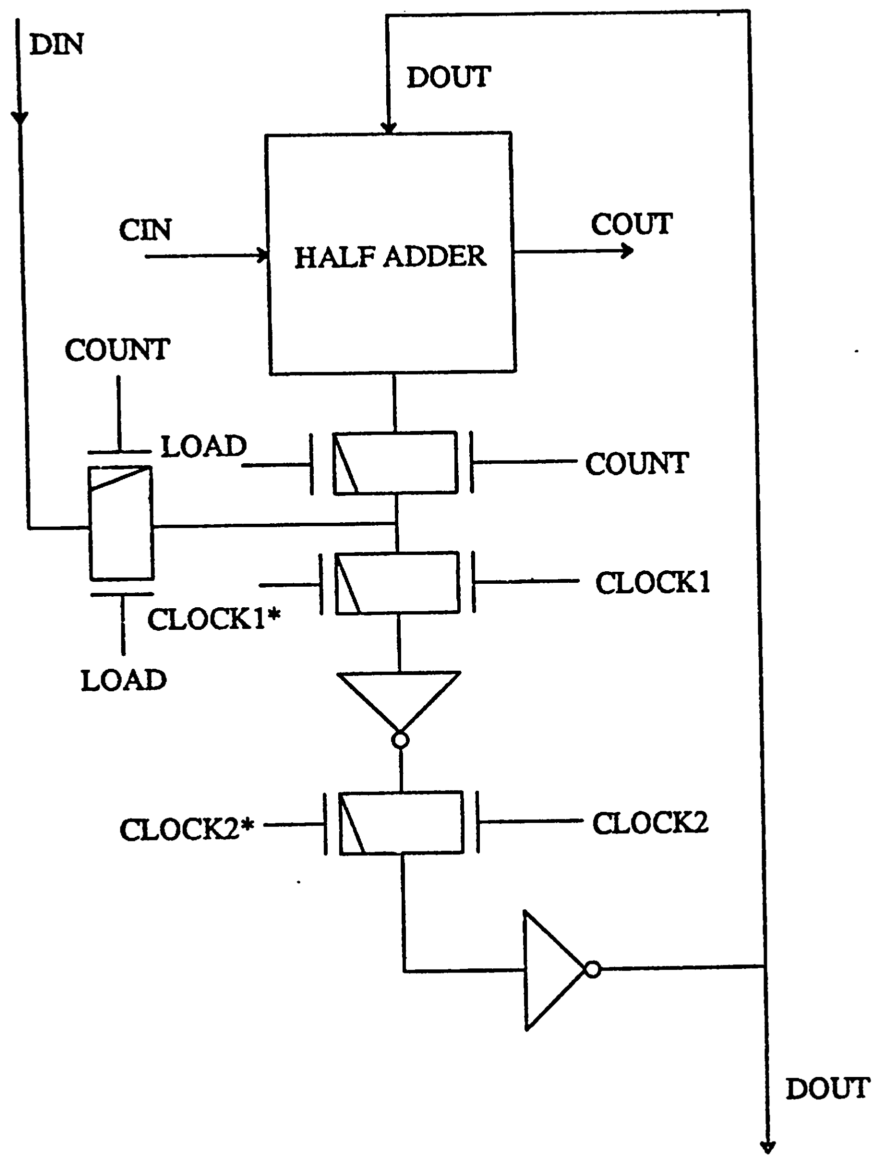


Figure 1.2 Counter bit-slice schematic

The decode section is empty unless a reference to a particular value of IX is made in the finite state machine definition. For each such reference, a row of decoder cells are included in the decode section. Basically, the decode section compares the counter value and the preset value in the decoder. If a match is found, a test signal is sent out to the finite state machine and the counter is reset. The decode section contains a NOR gate to decide whether the value from the counter is going to be added in the adder. If INDEX* equals to one, then the value is not added in; on the other hand, if INDEX* is equal to zero, the value will be added in the adder. The cells used in decode sections are shown in Figure 1.3.

We used carry propagate adders in the AAU design. The reasons are that we can easily parameterize the AAU by using carry propagate adders, and that in general, carry propagate adders are fast enough for normal AAU operation. Figure 1.4 is the circuit diagram for the adder cells.

The control slice contains many logic gates. The cells used are shown in Figure 1.5. The signal INC in ix.ctl cell comes from the on-chip ROM. When this signal goes high, the counter increments by one. The EOS signal in ix.ctl and the YCLOCK signal in iy.ctl are connected together. Both of them come from the Program Counter to decide whether the counter should load into data or keep on counting. The TEST signal is generated by the decoder cells in the decode section. TEST going high means that a match is met and the IY counter should load data at this moment.

The LAGER AAU can perform two addressing modes: indexed addressing and immediate addressing. Its detail operation can be found in [Pope85]. Starting from the next chapter, we will talk about how we use different array type layouts to design the cells of the AAU. Simulation is made to understand the electrical performance of layouts. Test chips will be made after satisfactory simulation results are obtained. From these results, we can easily find out the tradeoffs between these layout styles.,

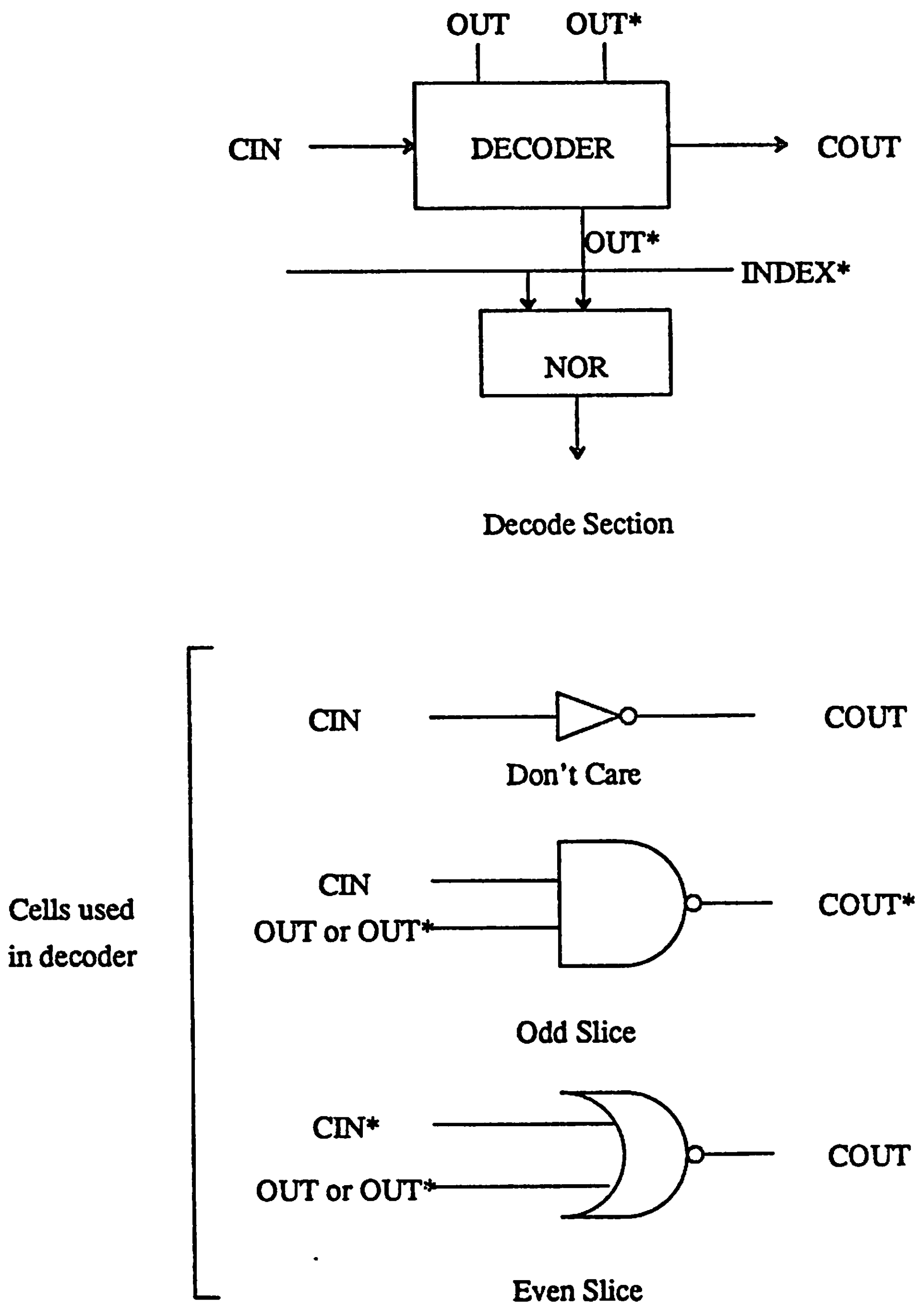


Figure 1.3 Schematics of cells in AAU decode section

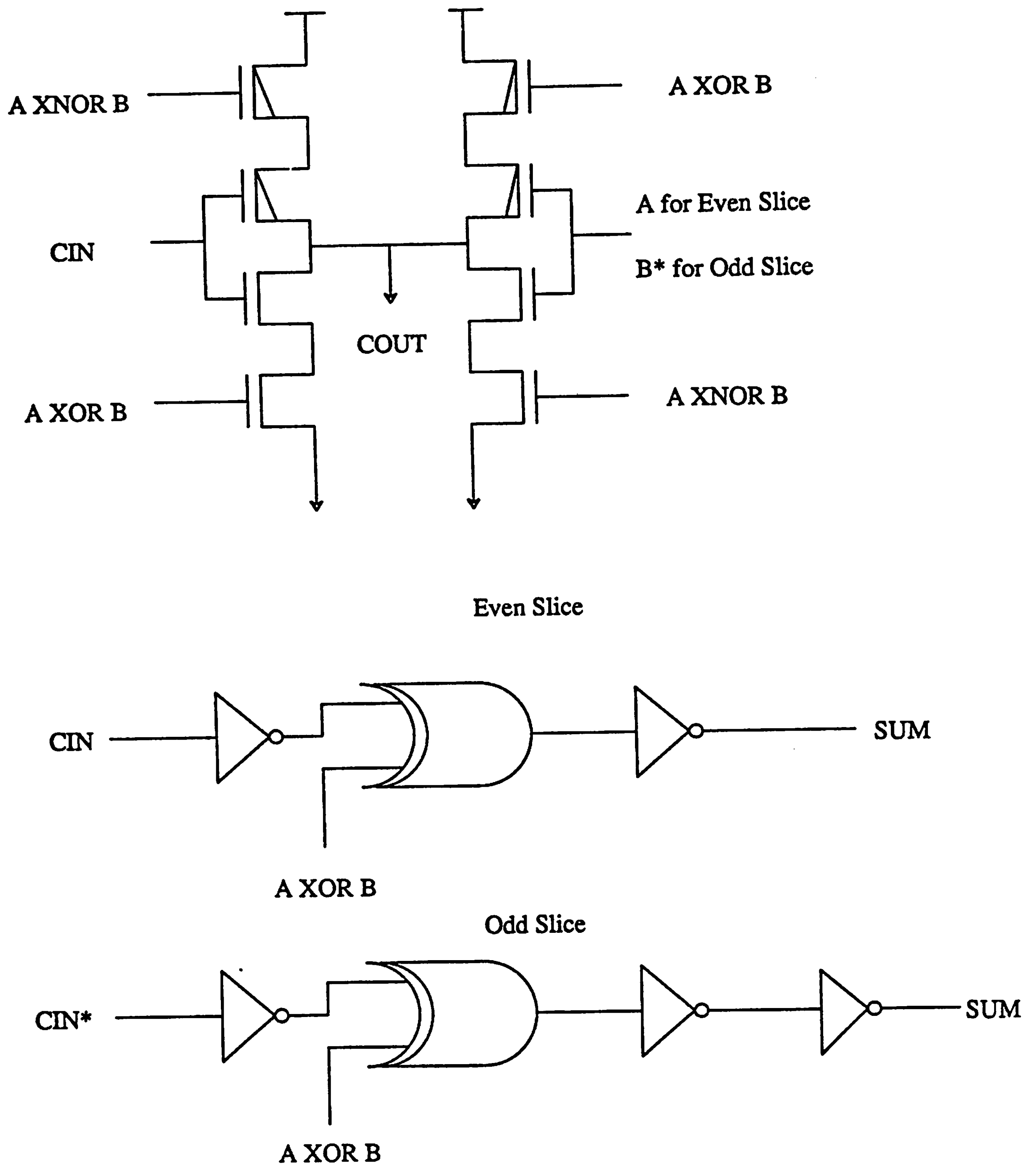


Figure 1.4 Adder cell

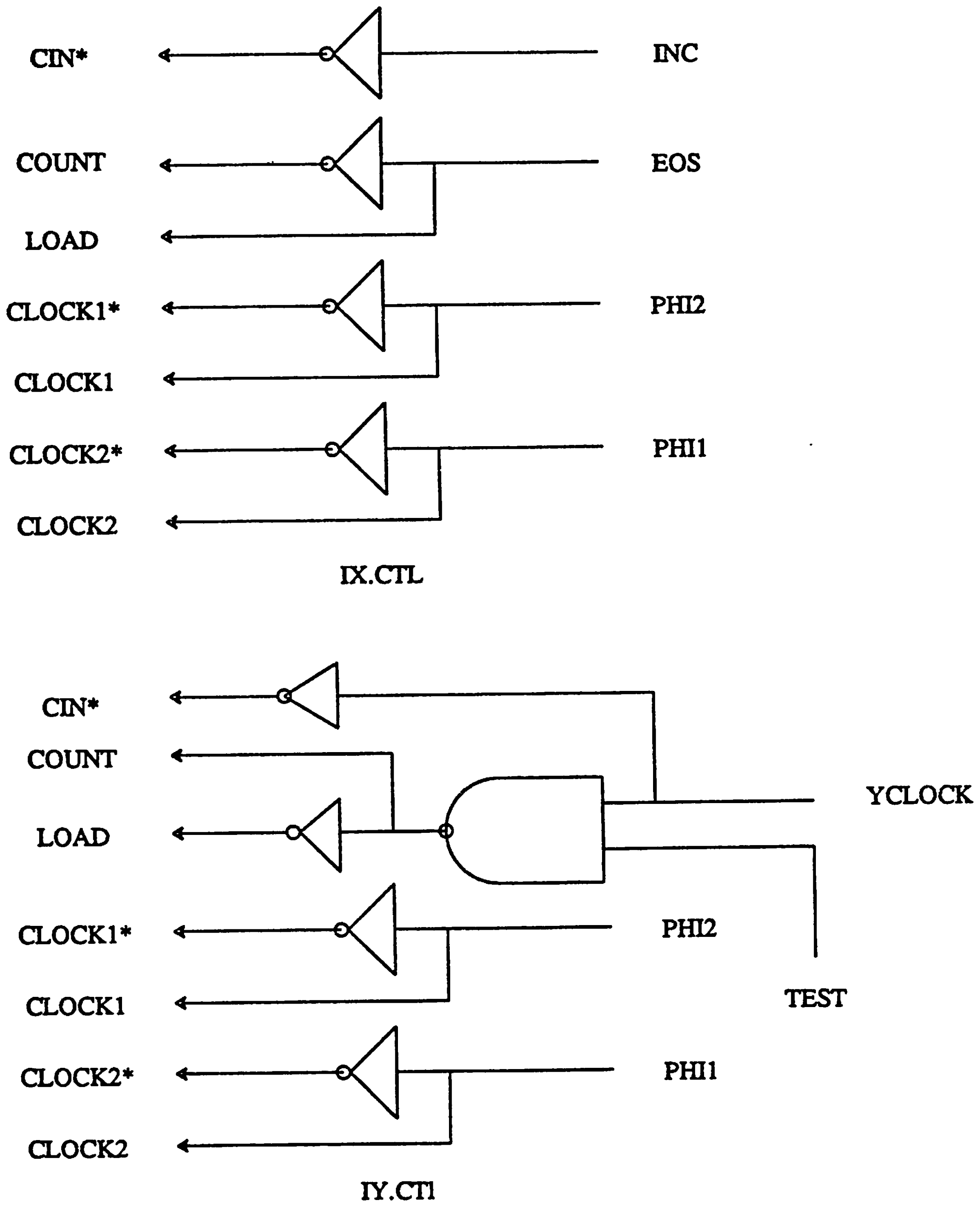


Figure 1.5 Schematics of AAU control cells

CHAPTER 2

GATE MATRIX

2.1. Basic Structure

The first array type layout we studied was gate matrix whose basic structure was originated by Lopez and Law in 1980. We made some modifications on their structure for the following reasons:

(1) Technologies used are different. The technology we use is MOSIS 3 micron scalable CMOS technology. We use two levels of metal for routing and different design rules.

(2) We need to interface these designs with other existing tools, such as the datapath compiler of LAGER III.

Figure 2.1 is the gate matrix layout of a D-latch cell, which was generated by a program called "game" [Sher86]. We will talk more about "game" in the next section. Basically, gate matrix structure involves polysilicon lines running vertically. These lines are equally spaced and parallel to one another. A transistor is formed if diffusion sits on top of polysilicon. Therefore, a polysilicon column serves as the gate of many transistors which lie on the line and the common connect among these transistors. Routing is simplified by these polysilicon lines since all the gate connections have been made.

Diffusion connections run between two polysilicon columns. It is not advisable to use diffusion for routing purposes due to its high resistance and capacitance. Therefore, a diffusion connection only exists when the source or drain of a transistor is connected to a power or ground bus. All the PMOS transistors are on one side, and all the NMOS transistors are on the other which reduces the overall area. When using the CMOS datapath compiler[1] to connect these cells, we will get a sandwich structure as shown in Figure 2.2. Note that all the cells must rotate 90 degrees to meet the specification of

[1] DataPath Compiler (DPC) is a tool to generate magic layouts of bit-sliced datapath starting from a structural description of the datapath in terms of interconnection of datapath functional blocks. For more information, see [Sriv87].

the datapath compiler.

Metal can run horizontally and vertically. For a cell design, we only use first level metal for interconnection. Metal two is reserved for global routing and providing feedthroughs. Metal one can overlap diffusion, polysilicon or transistors. There is no restriction on the metal one running.

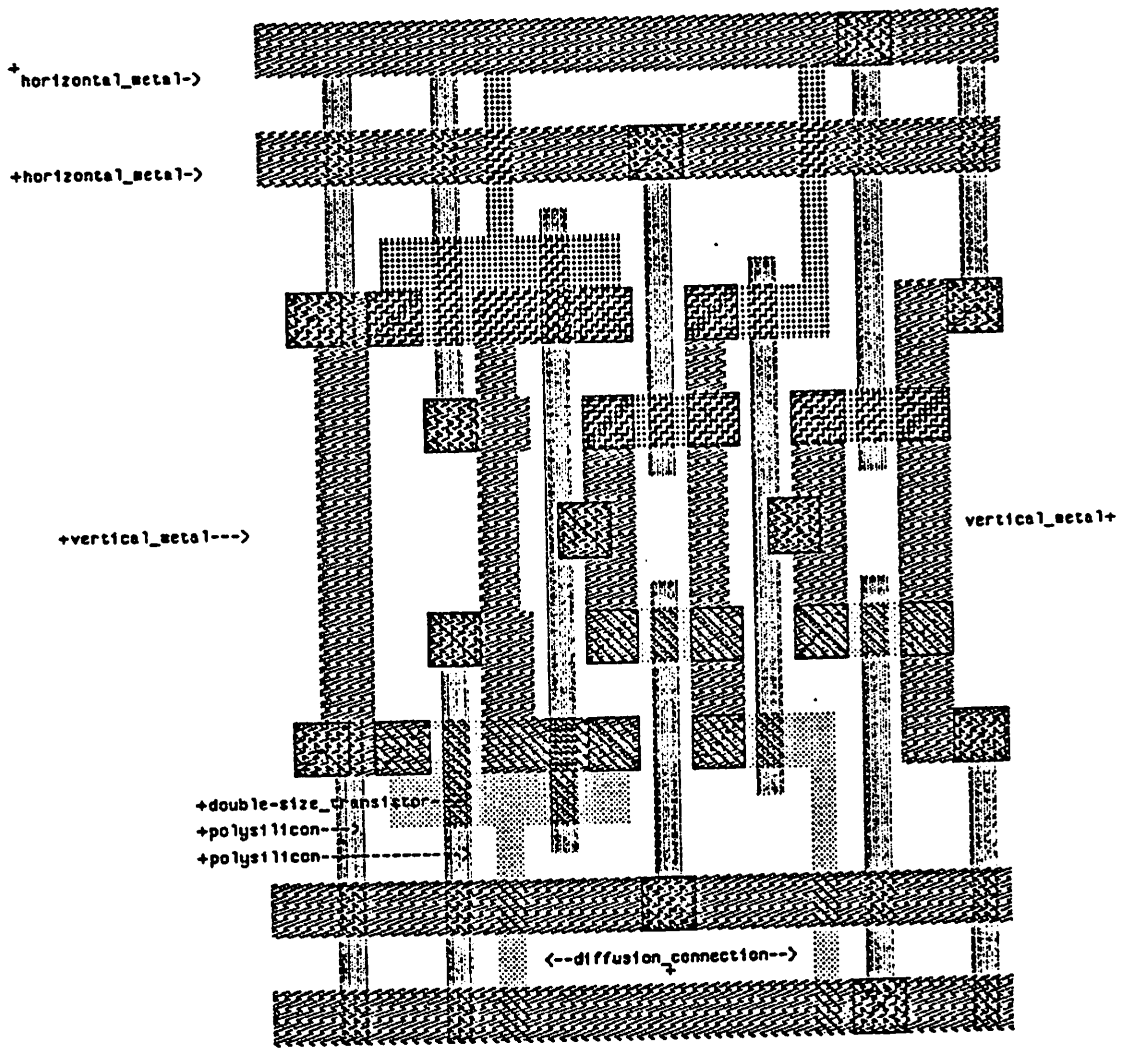


Figure 2.1 Layout of a gate matrix cell

To layout a cell, we should plan ahead with regard to the column position of polysilicon lines which may act as gates of transistors. The preparation stage of a layout consists of making a representational line drawing or a stick figure using the levels of interconnection available. These levels are polysilicon, diffusion and first level metal. The total number of polysilicon lines can never be less than the total number of discrete inputs to the circuit. The position of a polysilicon output is arbitrary and may be chosen for convenience. A transistor is formed by putting diffusion on the gating polysilicon column. Subsequent transistor placements will be determined by the input columns and the serial or parallel association among the transistors. Once the rows have been defined, further interconnection may be done by first level metal or diffusion links.

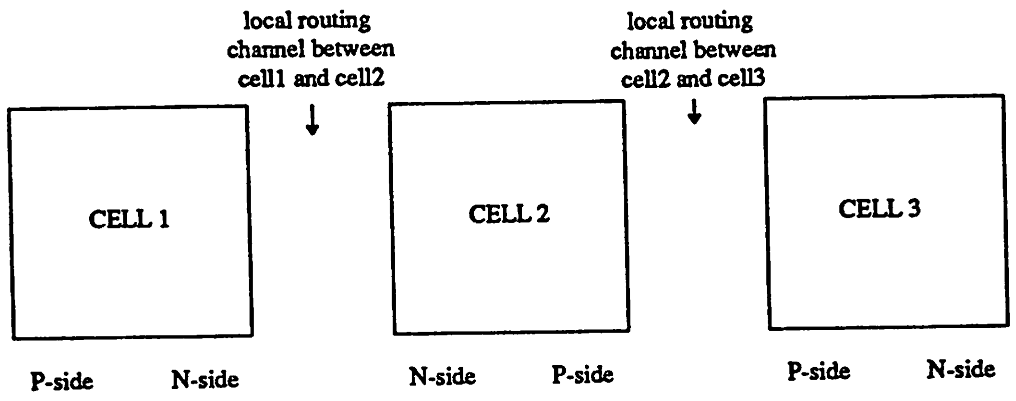

local routing channel between cell1 and cell2
local routing channel between cell2 and cell3
CELL 1
CELL 2
CELL 3
P-side
N-side
N-side
P-side
P-side
N-side

Figure 2.2 Connection of gate matrix cells using CMOS datapath compiler

2.2. Layout Tool -- GAME

A gate matrix layout translator called "game" has been installed. Basically, this program accepts symbolic input files and then translates them into layout files. Figure 2.3 shows a symbolic input file and the corresponding layout. Note that comments and notes can be put in the symbolic file. The symbolic input is technology independent. That is to say, if there is a technology change, we only need to modify the technology file referenced by "game". The symbolic input files are still applicable. Therefore, designers do not have to redesign the cells. It is reasonable to put remarks in the symbolic input files since these files will be kept regardless of technology changes. As shown in Figure 2.3(a), the text outside the brackets and braces is the comment of the cell.

The symbolic layout starts with "{" on a separate line and ends with "}" on another separate line. The legal symbols are listed as follows:

(1) ndiff: "n" (in horizontal and vertical directions)

(2) pdiff: "p" (in horizontal and vertical directions)

(3) nfet: "N" (can exist only on polysilicon channels)

(4) pfet: "P" (can exist only on polysilicon channels)

(5) metal1: "-" in horizontal and "|" in vertical direction

(6) metal2: forbidden inside the cell

(7) poly: "^" (only in vertical direction)

(8) crossover (no connection): "+" (all possibilities, except vertical metal1 over polysilicon or vertical diffusion in which cases "game" can not handle.)

(9) metal1 on top of nmos: "%"

(10) metal1 on top of pmos: "&"

(11) contact: "*" (all possibilities)

Compared with the original structure proposed by Lopez, "game" provides more symbols and flexibility. The layout generated is more compact and if compared with the custom design, the area loss is only about 20%[2].

"Game" provides another feature - labeling, which is not mentioned in the Lopez's paper. The text between brackets is the label description. For each label, designers must specify a label_name, a symbol, the y coordinate, and the x coordinate. Figure 2.3(a) shows some label usage. These labels are also shown in the layout of Figure 2.3(b).

To sum up, "game" provides a lot of desirable features:

(1) It can handle two kinds of technologies -- NMOS technology and SCMOS technology.

(2) Designers can specify the polysilicon pitch at will. There is a sparse option to provide a larger polysilicon pitch. The default polysilicon pitch is six lambda, which is the minimal possible

[2] The percentage is calculated from the ratio of AAU1 and the custom design of the LAGER AAU. If considering the leafcell design of AAU, the area loss ranges from 10% to 50%.

pitch for the 3 micron SCMOS technology.

(3) A new version of "game" provides a 90-degree rotation feature. To handle a cell which requires more than 80 columns for the symbolic input, designers can make use of this feature to relieve the restriction on the number of columns[3]. Since there is no limit on the number of rows, a rotated symbolic input can have any number of rows to represent the columns desired.

(4) "Game" will make vertical compaction on the layout automatically unless designers specify the other way around. The compaction can save the cell area up to about 15%, and thus makes the layout density comparable with custom design.

(5) "Game" provides automatic design rule checking. If a design rule violation is found, an error diagnosis will instruct the designers to make necessary modifications on the design. Error diagnoses will also be given if there are typing errors.

(6) Metal one usage is absolutely free if design rules are not violated. Metal one can overlap any kind of material and runs in both directions. To reduce resistance and capacitance, we can even replace polysilicon lines by metal one lines if possible. The free usage of first level metal provides a great flexibility for the users.

(7) Different sizes of transistors can be implemented by "game". To specify a wide transistor, designers will have to type several consecutive rows of N's (represents NMOS transistors) or P's (represents PMOS transistors). To design a weak transistor, designers must "snake" the transistor on a polysilicon column. This kind of design is not efficient from an area point of view. However, according to our design experience, weak transistors rarely appear in a datapath cell design.

(8) As mentioned above, "game" has a labeling capability. This feature is very desirable for the purposes of identifying terminals and routing.

[3] "Game" only allows 80 columns of symbolic input. This limitation is due to the width of text terminals.

This cell is the second part of an even-bit adder cell. When used with adr_evn1, we can get an full adder cell.

```
[
A ^ 0 1
SUM* ^ 0 5
AxnorB ^ 0 7
AxorB ^ 0 9
SUM ^ 0 11
Cin - 3 0
Cout* - 8 11
GND! n 0 4
GND! n 0 8
Vdd! p 27 4
Vdd! p 27 8
]

DSDSDSDSDSDSDSDS
{
 ^  n^ ^n^ ^
*+--+N*+++-*
n^  |^ ^n^ ^
++-*|+*^n^ ^
n^ ^ ^n^n^ ^
nN*N*++%+%*^
nN+Nn^nNnNn^
 ^|^ ^ ^ ^ ^
++++-+-+-+-+
|^ ^ ^ ^ ^ ^
|^ *-+-+-++^
|^ ^ ^ ^ ^|^
|^++-+-+-*|^
|^|^ ^ ^ ^|^
|^+Nn^ ^nN*^
|^*N**-+*N*^
|^ ^|^ ^ ^|^
|^*P*+*P*++^
|^+Pp^pPp^ ^
|^++-+-* ^ ^
|^ ^ ^ ^ ^ ^
+&*P*+*PpPp^
pPpPp^pPpPp^
*&+&+++&+&*^
 ^   ^ ^p^ ^
 ^  pPp^p^ ^
 ^  pP*+++-*
 ^  p^ ^p^ ^
}
```

Figure 2.3(a) Symbolic input of a logic cell for "game"

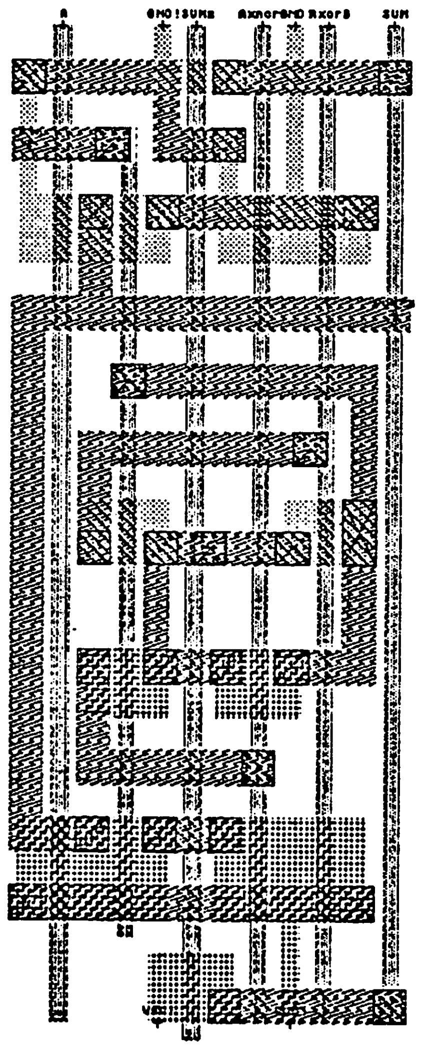


Figure 2.3(b) The corresponding layout generated by "game"

"Game" has many attractive features as described above, but "game" also has some weaknesses, which may be improved. First of all, its user interface is not good. Designers have to handle a lot of symbols when designing a cell. Graphic input is strongly urged to make "game" more user friendly. Secondly, the metal one width is fixed. Although "game" is able to chop off a piece of metal one when a design rule violation is detected, we still require different widths of metal one for power buses or for other routing requirements. Finally, well alignment will be a problem when it is incorporated with the

CMOS datapath compiler. For the current version of "game", there is still no way to specify the well boundary of a cell.

In the next section, we are going to use "game" to design all the cells we need for building the test chip AAU1. The layout strategy will be described in length to illustrate the tradeoffs of using gate matrix design.

2.3. Layout Strategy for Test Chip AAU1

LAGER AAU has a bit-slice structure which allows easy parameterization. For a complete datapath, the first slice is a ground slice which contains a ground bus and some simple interconnections. The last slice is a control slice which involves some control logic for generating the control signals of the cells in the same block. Power buses and system clock buses also run in the control slice. Between the ground slice and the control slice are the even and the odd bit slices of the datapath. See Figure 2.4 for the detailed structure.

We planned to use "game" to design each individual cell then used the CMOS datapath compiler to construct the bit slices and the whole datapath. Unfortunately, these two tools are not compatible. We need to make some modifications on the cells before we can use the datapath compiler to construct the whole structure. We will talk more about the incompatibility between these two programs in a later section.

Due to the bit-slice structure, we would like to make all the cells in a bit slice of approximately the same width. Therefore, we have broken larger cells into two pieces. The adder cell and the counter cell are the examples. For smaller cells, such as the decoder cell, we combined other logic inside the cell to make it have approximately the same width as the other cells. Using "game", we can easily predict the width of a cell; however, the height of a cell is unpredictable due to the compaction feature of this program. The well alignment for each cell in the same block is another problem since "game" does not provide with the feature of well specification. We will discuss more on these problems in a later section. In designing AAU1, all the interconnections were made by hand, no CAD tools were involved. Therefore, the above problems can be solved by proper placement of cells. In addition, routing area is minimized. The disadvantages are long design time and the higher possibility of routing errors.

AAU1 is a ten-bit address arithmetic unit. It includes both IX and IY sections. The preset value of the decode section is randomly chosen. The only criterion is the convenience of testing. Beside the ten-bit AAU, there is another one-bit single section AAU on the test chip. The reason for designing this one-bit AAU is that we can easily generate complete test vectors for it to insure the correctness of the AAU logic design. The 10-bit AAU testing, on the other hand, can provide us the information of propagation delay and the correctness of the function.

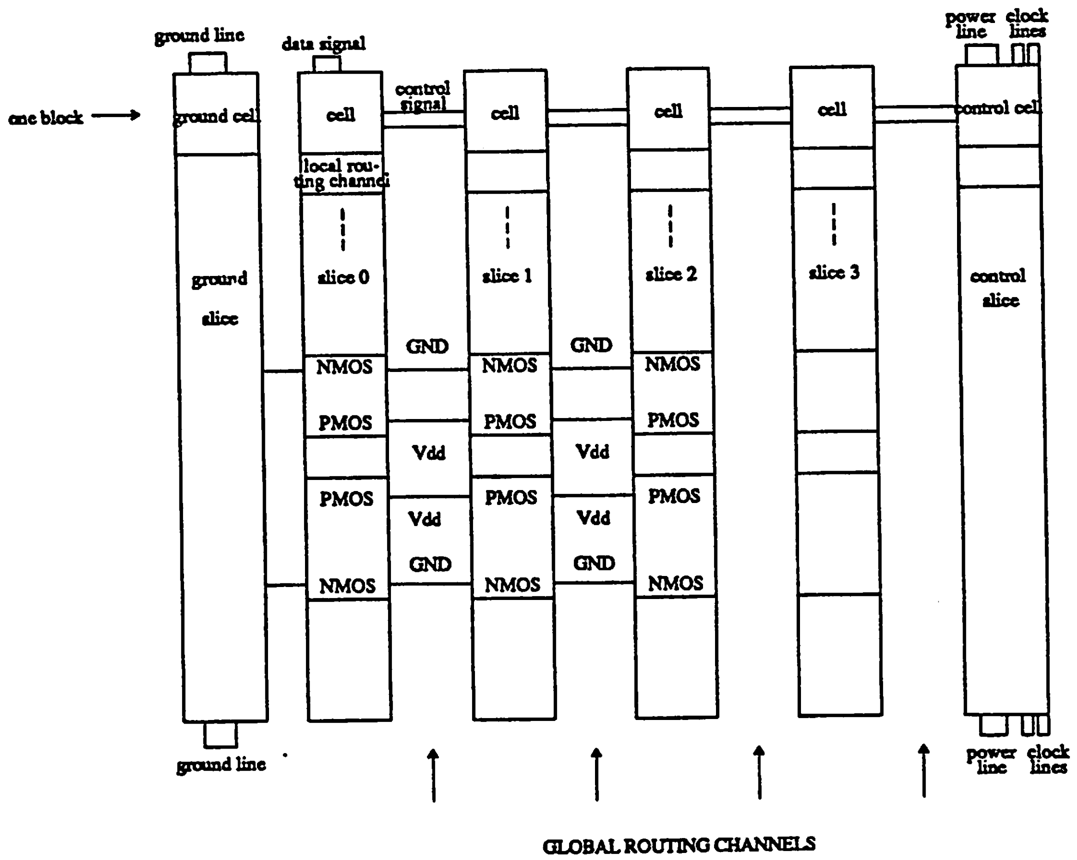


Figure 2.4 Bit-slice structure of the LAGER AAU

AAU1 has 48 pins in total. Figure 2.5(a) shows the whole layout of the chip. The 10-bit AAU has a height of 762 lambda and a width of 1083 lambda. Note that the datapath has been rotated 90 degrees to simulate the final result that will be generated by the datapath compiler. From Figure 2.5, we can see that the whole layout is very compact. The isolated part below the main datapath is the one-bit AAU. It shares the same power pins and clock pins with the 10-bit AAU.

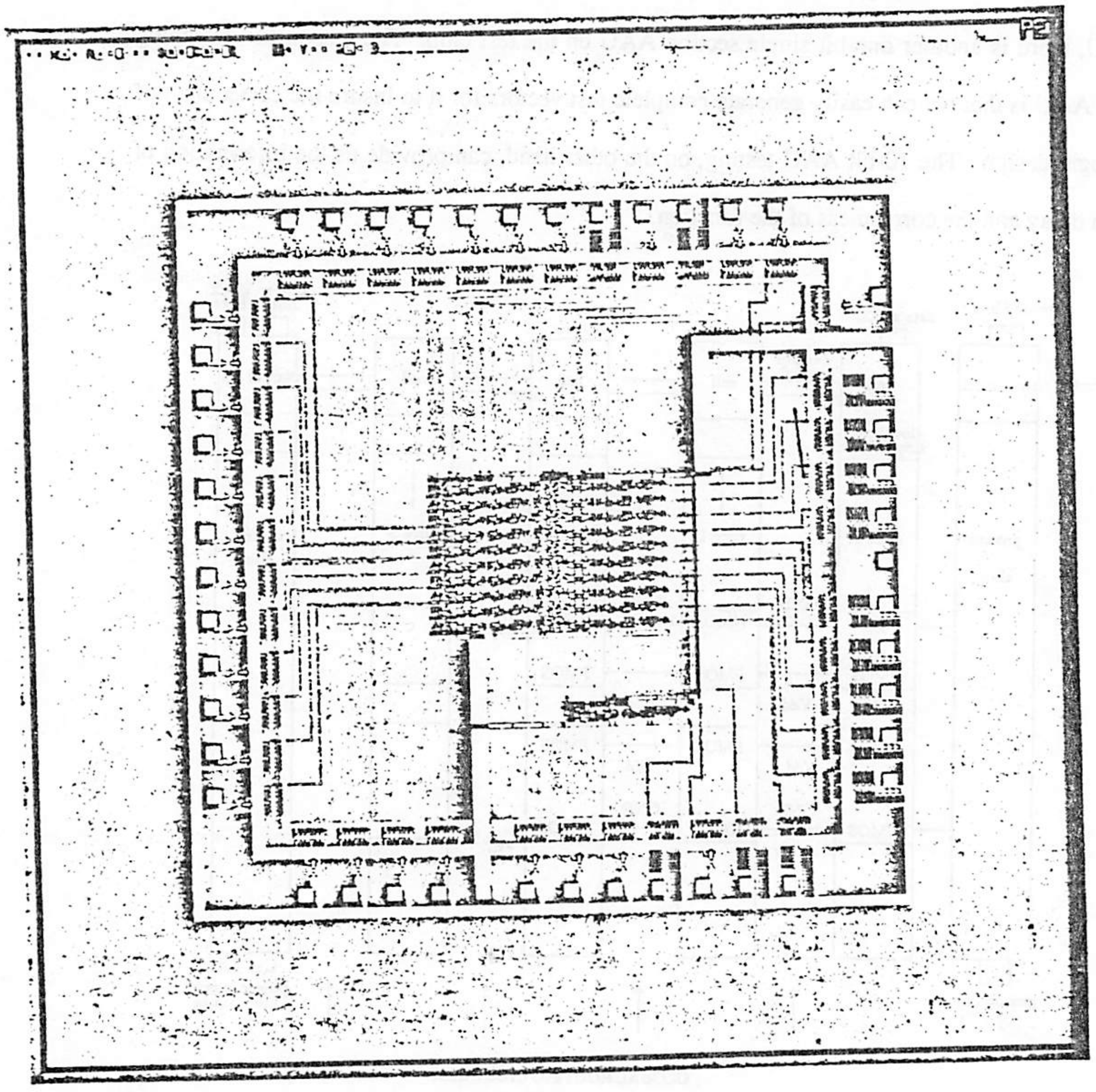


Figure 2.5(a) Die photo of AAU1

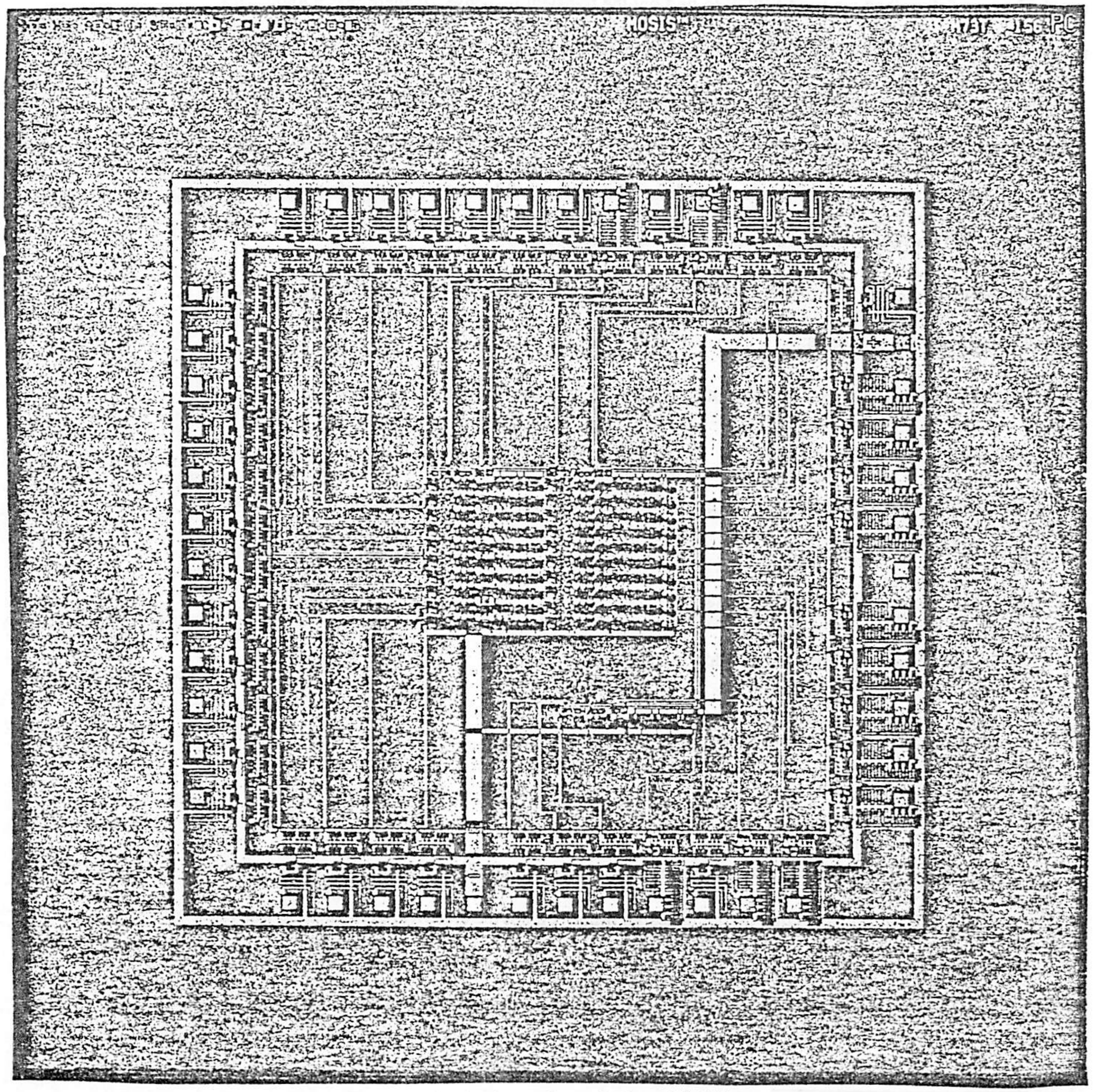


Figure 2.5(b) Die photo of new design of AAU1

2.4. Test Results of AAU1

Having designed all the cells, we used ESIM [Term82] to simulate the logic and crystal to find out the longest delay, then we used SPICE to find out the transient response. If there was a logic error or the delay was too long, then we redesigned the cell until the electrical performance met the specification. Our goal is to achieve a 10 MHz clock rate. Therefore, the total propagation delay can not be longer than 100 nsec. Table 2.1 lists the area and electrical performance of the final design. The longest propagation delay[4] of the adder is about 50 nsec for an 8-bit AAU. The performance of an 8-bit double-section AAU can approximately meet our specification.

To calculate the worst case delay of a 10-bit double-section AAU on a test chip, we need to consider at least the following terms:

(1) delay caused by input pads and output pads : 14 - 18 nsec according to test results.

(2) delay caused by clock signal driving : about 5 - 7 nsec according to simulation.

(3) delay caused by the counter, the inverter, and the decoder : about 15 nsec according to simulation.

(4) delay caused by a 10-bit adder : 55 nsec.

(5) delay caused by the second 10-bit adder : 10 nsec.

(6) delay caused by the output buffer : 5 nsec.

therefore, the total delay will be

$$18+7+15+55+10+5=110 nsec$$

The test chips were made by MOSIS on run M66V and M67Z. Testing equipment used was the Tektronix DAS. Generally speaking, the logic design is correct, but the electrical performance was not adequate. There is a significance difference between the simulation and test results. The reasons will be discussed later. The test results of AAU1 can be summarized as follows:

The worst case carry propagation delay of an adder is 4.5 nsec per stage, and the delay caused by sum evaluation is 10.5 nsec. Hence, the total delay of an 8-bit adder is $4.5 * 8+10.5=46.5$ nsec.

The one-bit single section AAU functions well. The test results confirm that the logic design of AAU1 is correct. To test the 10-bit double section AAU, we designed several groups of test vectors to test different cells. The results showed that all the counter cells, the decode section, and the adder cells worked fine. For the testing of the counter cells and the decode section, the clock rate can go up to more than 10 MHz. But for the testing of the adder cells, the clock rate can only go up to 5.6 MHz under the test of the most critical propagation delay. The difference between the simulation and the testing results are within 50% and there are three main reasons for the difference. First, the clock signal used for the testing was not good. The square waves generated by the signal generator included a lot of harmonics and noise when the frequency went up too high (about several MHz) and the two-phase non-overlapping clock was not exactly achieved. Second, the model used for the SPICE simulation were different from the real parameters due to process variations. Third, some distributed resistance and capacitance were ignored in the simulation. All these factors caused the difference between the simulation and the testing. A new test chip has been built and fabricated by MOSIS on run M73T. Figure 2.5(b) shows the die photo of the new design. In designing the new test chip, we used the specifications of the LAGER CMOS datapath compiler. Some of the cells have been redesigned to increase their driving capability[5]. The test results show that the new chip has better electrical performance than AAU1. The chip can work up to 6.5 MHz under the test of the most critical propagation delay. Again, this limitation is due to the adder cells. The counter cells and the decode section can go to frequencies higher than those of AAU1.

[5] The new design of AAU1 used a new logic to implement the adder cells. Some of the gate matrix rules are violated, but basically, it still belongs to the gate matrix design style.

Table 2.1 Area and Simulation Results of AAU Leafcells

cell name	area (lambda * lambda)	transistor density (# of transistors per unit area)	propagation delay (simulated by SPICE)
ctr.e	102*64	1/544	Cout: 4ns
			sum of half adder: 8ns
ctr.o	100*64	1/533	Cout: 4ns
			sum of half adder: 7ns
dff	76*56	1/423	Dout: 5.5ns
			Dout* : 7.5ns
dec.e.0	60*42	1/252	b : 7ns
			Cout : 2ns
dec.e.1	60*44	1/264	b : 8ns
			Cout : 2.5ns
dec.e.x	52*44	1/286	b : 6ns
			Cout : 2ns
dec.o.0	56*44	1/246	b : 8ns
			Cout : 1.5ns
dec.o.1	56*44	1/246	b : 6ns
			Cout : 2ns
adder_even	48*202	1/440	sum : 10ns
			Cout : 3.5ns
adder_odd	48*192	1/418	sum : 10.5ns
			Cout : 4.5ns
iy.ctl	46*28	1/214	7ns

2.5. Possibility of Interfacing GAME with CMOS Datapath Compiler

Although "game" has a lot of features, it was not popular because of the poor user interface. Another problem of "game" is its incompatibility with the CMOS datapath compiler. However, both of these problems can be solved by introducing a graphic input capability and a postprocessor to modify the layouts to meet the requirements of the datapath compiler. The postprocessor should be able to handle the following problems:

(1) Well alignment and cell height unification: The CMOS datapath compiler requires all the cells in the same block to have an equal height and an equal well boundary. This requirement can be achieved easily for custom design; nevertheless, "game", which has to take care of cell compaction, can not easily provide this feature. One possible solution to this problem is that designers specify all the cells which may appear in the same block in a single symbolic file. "Game" translates and makes compaction on all the cells at the same time. Thus all these cells will have the same height and well alignment automatically. If the sum of the columns of all the cells exceeds 80, designers can make use of the 90-degree rotation option provided by "game" to solve this problem. Another more natural solution is to modify "game" so that well boundary and cell sizes can be specified by the users. This feature is still not available now.

(2) Well contacts and power buses: The current version of "game" is unable to handle well contacts or power buses. However, the datapath compiler takes power lines as control lines, that is to say, cell designers must put power buses inside cells. To enable "game" to handle well contacts, we must introduce new symbols; to handle power buses, we should have a way to instruct "game" to adjust the widths of metal lines. Both problems can be solved, but the complexity of the symbolic input is even more unacceptable.

(3) Usage of metal two: Metal two is not allowed for cell design in "game". But for routing efficiency, the datapath compiler requires cell designers to provide feedthroughs on top of cells using metal two. To introduce metal two into "game", we will have to use a lot more symbols than it has now since metal two can overlap any kind of material and it should be able to run in both vertical and horizontal directions. We can not afford such a complicated symbolic input; therefore, a graphic input is the only solution to this problem. If "game" can provide a graphic user interface, it will be a very popular and useful tool for the gate matrix cell design.

2.6. Comments on Gate Matrix

Gate matrix design simplifies the layout procedure by its regular structure. Another attractive feature of gate matrix design is its technology independence. To make use of these features, better tools, such as "game" with a graphical user interface and compatibility with the CMOS datapath compiler, should be installed. In fact, a lot of designs are making use of gate matrix layout [Poon85] [Asad87] and better tools are being developed to facilitate these designs.

Gate matrix design can achieve approximately the same area efficiency as custom design. However, the electrical performance degrades due to the fact that it uses polysilicon or diffusion for interconnections. One solution to improve this problem is try to replace polysilicon or diffusion by metal as much as possible. The layout tool "game" provides users with the capability to do that and future tools have to provide users with the same capability.

The layout time for gate matrix design is shorter than that for custom design, but compared with gate array design, it takes longer. Gate matrix design provides designers a simple and unified procedure to layout cells, and the performance of the design can approximately reach the level of custom design without losing too much area. As long as proper tools can be supplied, gate matrix will be an important layout style.

CHAPTER 3

SEA OF GATES

3.1. Basic Idea of Sea of Gates Design

Conventional gate arrays [Holl87] have alternate rows of cell columns and routing channels of fixed widths to be used for first metal routing. This kind of structure has the following constraints on layout:

(1) The number of metal one tracks allocated must large enough to handle the routing of macrocells in the most congested area of the chip. But in reality, the channel width requirement varies from design to design and changes across the chip. More tracks have to be allocated than actually needed to ensure routability and thus results in wasted area.

(2) In many cases, the number of metal one tracks allocated is not enough in the adjacent routing channel and the metal lines have to detour through other channels. This causes undesirable congestion in adjacent channels and increases delay due to longer metal wires.

A double metal HCMOS Sea of Gates array [Hui85] and a channelless gate array [Hsu86] were brought out to tackle these constraints. The innovation of these designs is that the whole chip has no pre-defined first metal routing channels. Routing is built by running metal over cell columns. The number of tracks in each channel can be adjusted and it is easier to achieve 100% auto-routing with a high percentage of gate utilization. The authors claimed that a close to three fold improvement in internal gate density is achieved. In one particular circuit, 46 percent of the raw gates were utilized. With this approach, macrocells can grow in both directions. There will be no area wasted in predefined channels.

With this idea, Wong built up a high performance 129K-gate CMOS array [Wong86]. A Bit Map Controller was laid out on a 14.95 mm * 14.95 mm die. The operation frequency is over 20 MHz. Table 3.1 lists some of the key process parameters used by Wong's design. These parameters are quite different from the MOSIS SCMOS design rules which are available to us. Therefore, we will have to design our own Sea of Gates template, but the basic structure is the same as the Compacted Array.

Table 3.1 LSI and MOSIS Key Process Parameters

PARAMETERS	TYPES	LSI RULES	MOSIS RULES
GATE LENGTH	NMOS/PMOS	1.5/1.5 micron	2/2 lambda
EFFECTIVE CHANNEL LENGTH	NMOS/PMOS	0.9/0.9 micron	~1.77/1.64 micron*
GATE OXIDE THICKNESS		0.25 micron	0.0423 micron*
THRESHOLD VOLTAGE	NMOS/PMOS	0.8/0.8 V	0.82/0.78 V*
GRID SIZE	X/Y	5.6/6.0 micron	7/8 lambda
CONTACT HOLE SIZE		1.2 micron	2 lambda
FIRST METAL	LINE/SPACE	1.9/2.1 micron	4/3 lambda
VIA SIZE		1.6 micron	2 lambda
SECOND METAL	LINE/SPACE	2.5/2.1 micron	4/4 lambda

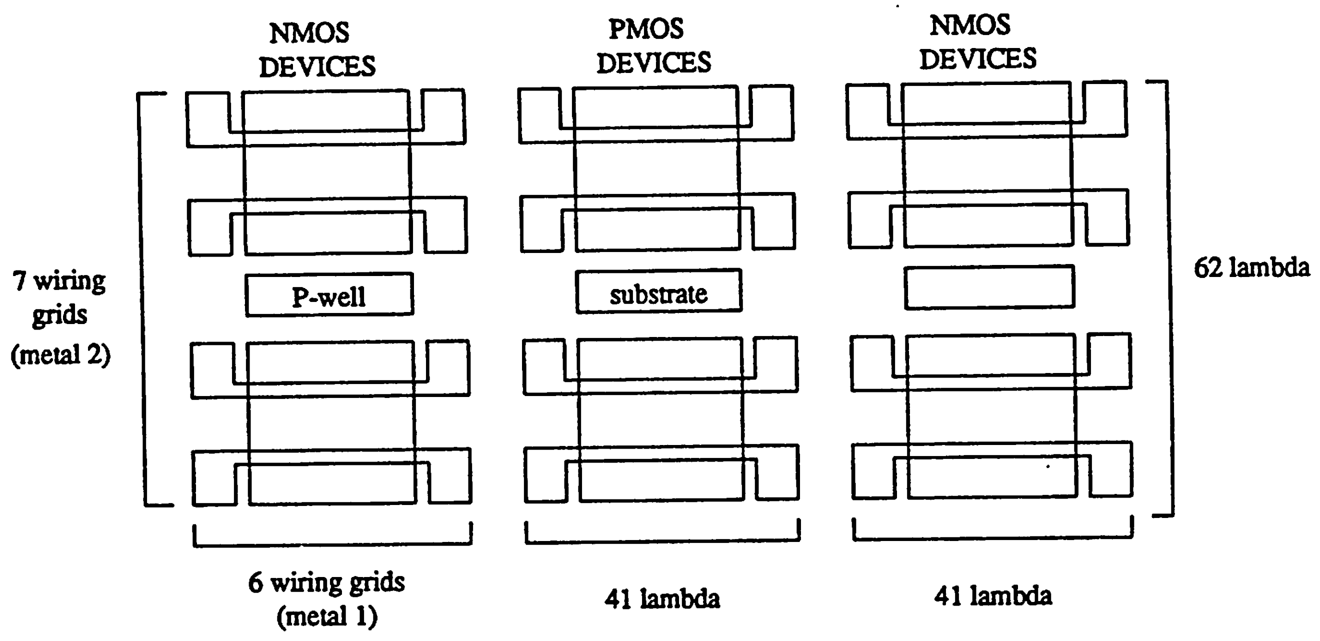
* Parameters are based on MOSIS 2 micron SCMOS technology.

3.2. Design of Sea-of-Gates Templates

The template of sea of gates design is similar to the gate array template except that there is no pre-defined channel. The whole chip is filled with cell columns. Because all the routing is built on top of the cells, the template has to be carefully designed to meet the requirement of grid-based routing. In another word, at any possible intersection of metal one and metal two, sufficient spacing should be allowed for a via to be placed.

Figure 3.1 shows the template designed for sea of gates layout. Basically, metal two runs horizontally and metal one runs vertically in the cell columns used for routing channels. Metal one is used for the interconnection in both directions inside the cells. The minimum size of a transistor is 2 lambda / 32 lambda. The reason for designing such wide transistors is to provide enough wiring grids for a cell column. In addition, wider transistors provide stronger driving capability. The power consumption is larger than gate matrix design but reasonable. In the following section, we will see from simulation results that the propagation delay is shorter than that of gate matrix design. Readers may worry about the area efficiency for wide-transistor templates. Compared with gate matrix design, the area loss for a cell is not serious (about 30% to 70%) even though the transistors are much wider.

The reason is that for gate matrix design, most cell area is devoted to interconnection. Although most transistors are of the minimum size, the total cell area can not be reduced due to the fixed routing area. For sea of gates design, all the routing is built on top of cells and will not cause extra area consumption if the transistors are properly sized to accommodate the routing. Cell area is decided by the number of transistors needed for each cell. In short, we can say that the cell area for gate matrix design is routing bound, while for sea of gates design is transistor bound.



* SYMMETRIC CELL STRUCTURE
* NO PREDEFINED ROUTING CHANNELS
* EVERY 2 CELLS HAVE P-WELL AND SUBSTRATE CONTACT

Figure 3.1 Internal array structure of Sea of Gates design

The possibility of designing weaker transistor templates is considered. According to the layout experience, 6 grids per cell column is about the optimal value. A weaker transistor template will degrade electrical performance. Also, it may cause more area consumption due to the fact that more cell columns are needed for completing interconnection. The optimal value certainly depends on the complexity of routing and cell design. For a simple cell design, smaller number of grids per cell column may be desired. Take the AAU design for an example, 5 grids per cell column is the optimal value when considering the area. If we use this template to design AAU2, the area of AAU2 can be reduced by 13%.

3.3. Layout Decisions Made for Test Chip AAU2

To investigate the performance of sea of gates design, we used the same logic as AAU1 to build a test chip AAU2. In laying out AAU2, we had to make some decisions on the datapath structure and the routing strategies. Should we keep the bit-slice structure for the sea of gates design? Should we use the CMOS datapath compiler to build up the datapath?

For the first problem, we need to consider the compatibility between bit-slice structure and sea of gates design. Basically, these two issues have no conflict, but would it be more efficient to use general purpose place-and-route algorithms than using the bit-slice structure? To use general purpose place-and-route algorithms for sea of gates design, we must have powerful CAD tools. Place-and-route algorithms for sea of gates design is different from conventional place-and-route algorithms in that the number of cells to handle is much larger. The conventional layout algorithms have difficulties in handling such large problems. For the time being, there is still no CAD tools available to support general purpose place-and-route design for very large sea of gates layouts. In addition, the bit-slice structure has many nice features such as parameterizability and regularity that will be lost if we adopt general purpose place-and-route design. Therefore, we decided to keep the bit-slice structure in designing AAU2.

Since we decided to keep the bit-slice structure for the AAU2 design, should we use the CMOS datapath compiler for building up the whole macrocell? The decision is no and the reasons can be summarized as follows:

(1) The CMOS datapath compiler does not use a grid-based router.

(2) The CMOS datapath compiler requires all the data terminals brought out to the top or to the bottom boundary and the control terminals brought out to the right or to the left boundary of a cell[1].

[1] The way the CMOS datapath compiler treats a cell is always 90-degree rotate of a usual cell. In another word, usually a

For a sea of gates design, this restriction will increase the layout difficulty because usually the width of a sea-of-gates cell is not wide enough to accommodate all the data terminals. Besides, unlike gate matrix design, the sea-of-gates cells can not make use of polysilicon or diffusion for intra-cell routing. This even increases the difficulties of bringing all the data terminals to the cell top/bottom boundary.

(3) The CMOS datapath compiler treats power and ground buses as control signals and makes them run perpendicularly to data signals. But for sea of gates design, it is natural to make the power and ground buses run in parallel with the data signals.

After thorough consideration, we decided on the bit-slice structure of AAU2 shown in Figure 3.2. Basically, it has the same structure as AAU1 except that the power lines run vertically instead of horizontally. Also, all the data signals were brought out to the right boundary and use global routing channels for connection if they can not be routed by local interconnections between the two cells. Figure 3.3 shows the whole layout of AAU2. The total area for the 10-bit AAU is 1413 lambda * 993 lambda. Compared with the gate matrix design, it is about 70 percent larger. The number is acceptable since the transistor sizes are so much bigger. If we use a weaker transistor template, say 5 grids per cell column, about 50% area loss can be achieved. In addition, the sea of gates design has many desirable features which are not achievable with the gate matrix design. We will talk more about the tradeoffs in the latter sections.

bit slice structure has data signals going vertically, control signals going horizontally. But for the datapath compiler, data signals go horizontally and control signals go vertically. To unify the directions, we take the direction of a normal bit slice as standard. What we see generated by the CMOS datapath compiler should rotate 90 degrees to get a correct direction.

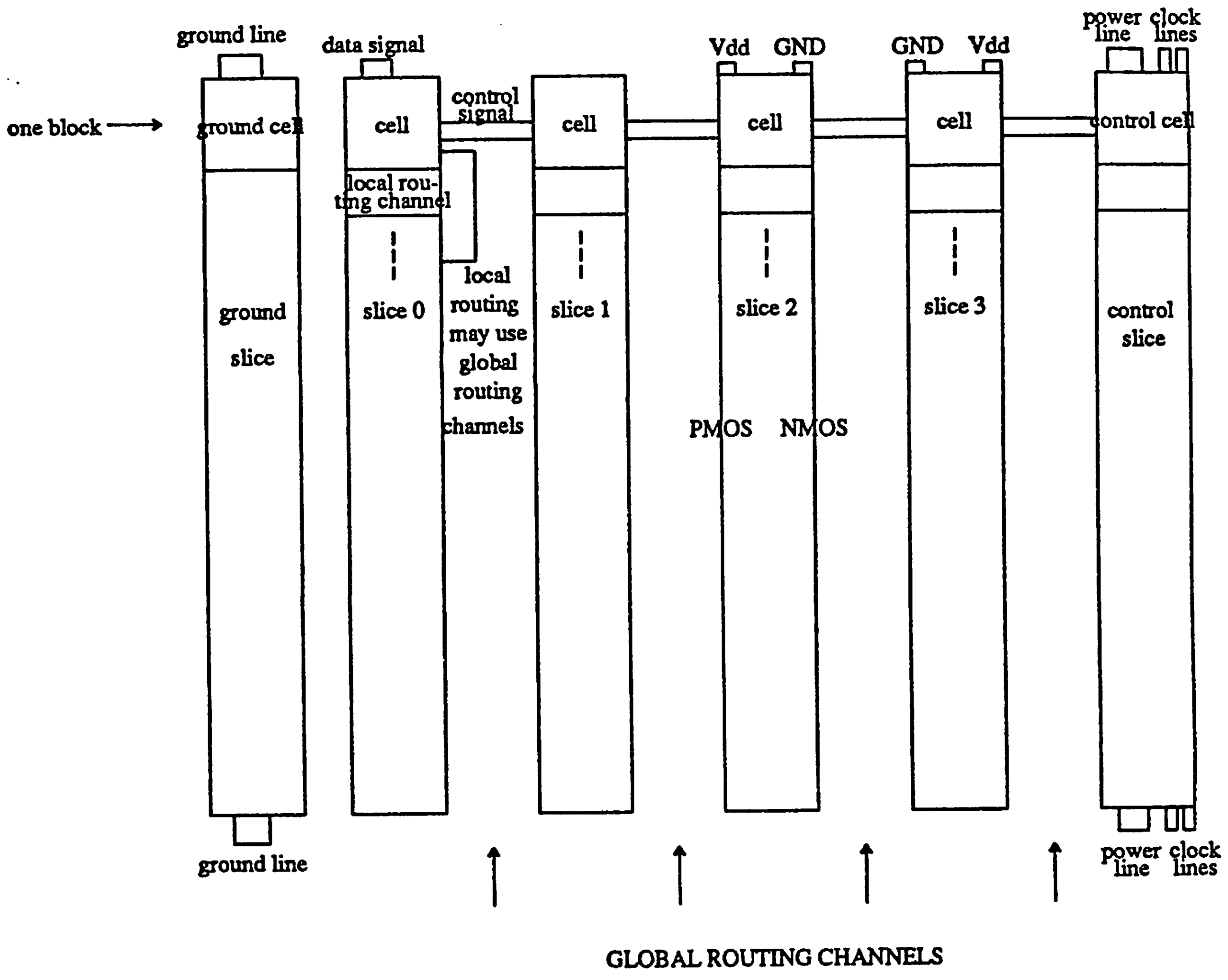


Figure 3.2 Bit-slice structure of AAU2

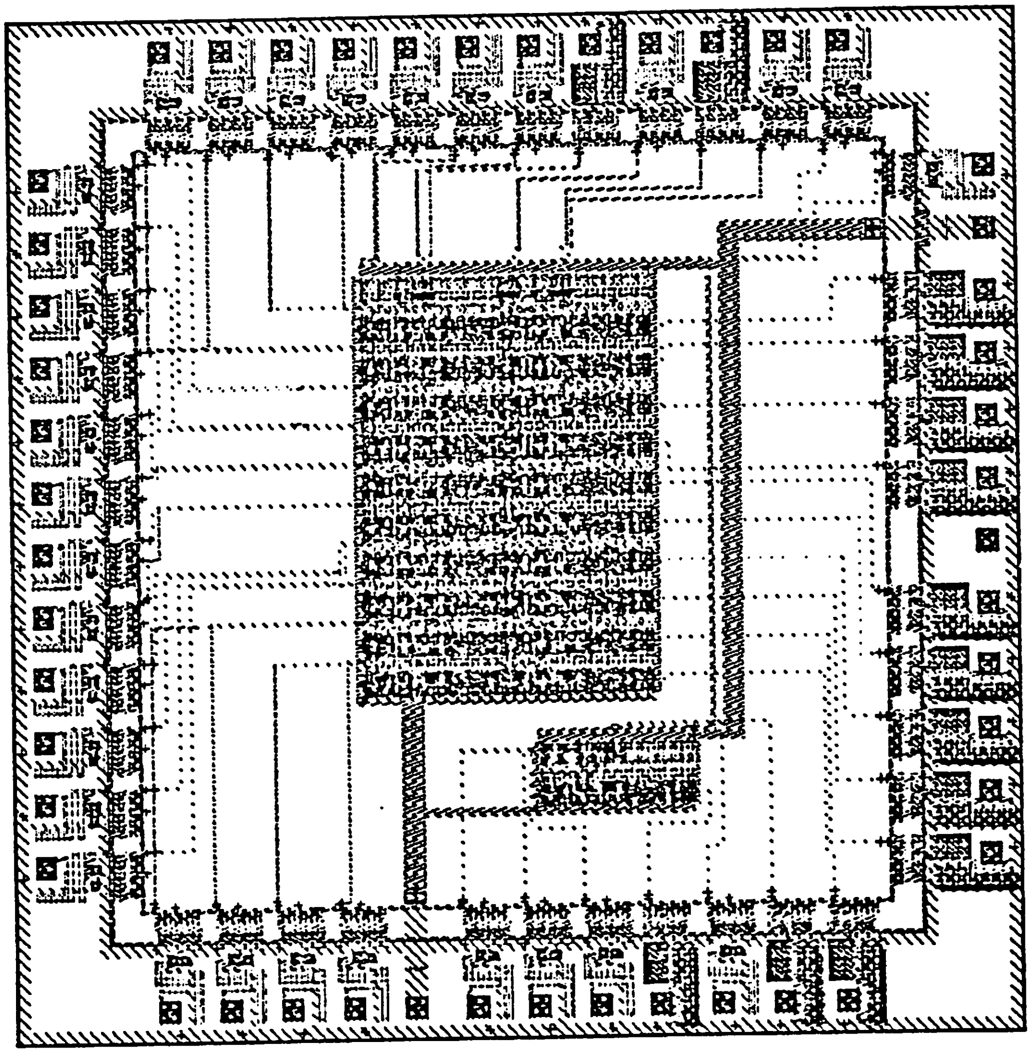


Figure 3.3 Chip layout of AAU2

3.4. Simulation and Test Results of AAU2

A basic cell layout is shown in Figure 3.4. Note that some of the transistors are not used due to the connection problems. Basically, most of the transistors are of the minimum size except those with different driving capability required. A double size transistor can be made by two minimum size transistors in parallel and a half size transistor can be made by two minimum size transistors in series. But either kind of transistors will need twice the area of the minimum size transistor.

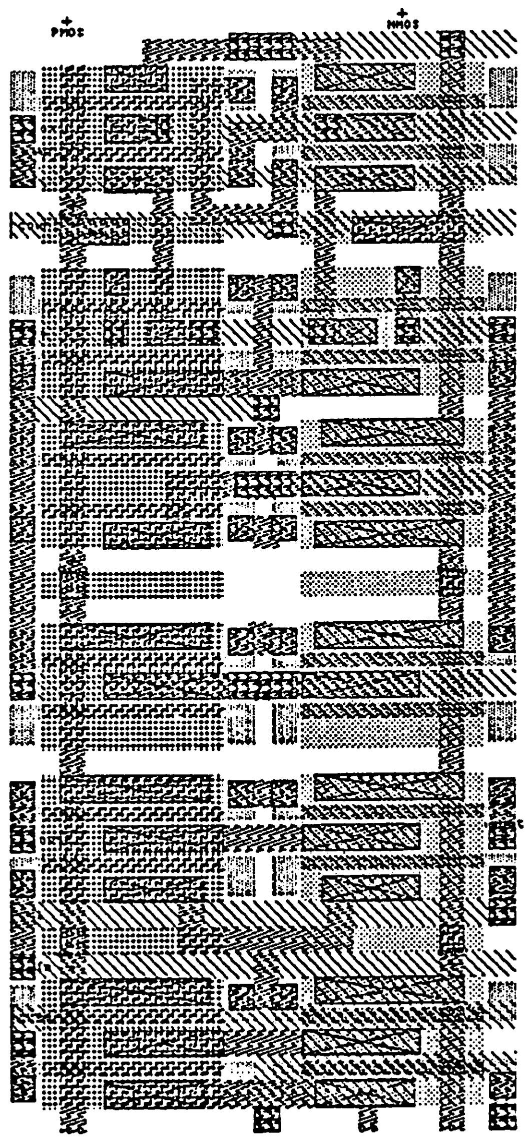


Figure 3.4 Counter cell laid out in sea of gates style

To design a cell, designers only need to perform the first and second level metal routing. Both the layout time and the turnaround time is greatly reduced. In addition, all the cells are technology independent. We don't need to redesign the cells in the cell library if technology changes. Table 3.2

lists the cell areas and the SPICE simulation results of some of the AAU2 leafcells. We expect that the electrical performance of AAU2 will be better than that of AAU1 from these simulation results. Note that the data of decoder cells are not listed in this table. The reason is in the difference between the designs of gate matrix and sea of gates. For the gate matrix design, we combined some other logic into the decoder cell to make the cell width approximately equal to those of the other cells in the same bit slice. For the sea of gates design, a decoder cell is simply a NOR or a NAND gate. It is unnecessary to combine other logic since all the cell widths are equal. Therefore, we did not list the simulation results of the decoder cells.

Table 3.2 Cell Area and Simulation Results of AAU2 Leafcells

cell name	area (lambda * lambda)	% larger than gate matrix	propagation delay simulated by SPICE
counter	113*81	25%	Cout: 2ns
			sum of half adder: 5ns
dff	70*84	30%	2ns
adder_even	190*83	70%	sum : 5.5ns
			Cout : 4.5ns
adder_odd	190*83	70%	sum : 8.5ns
			Cout : 4.5ns

A test chip of sea of gates design has been fabricated by MOSIS on run M73T. Figure 3.5 shows the die photo of AAU2. The same test procedure as AAU1 has been carried out and the test results show that AAU2 can work up to 6 MHz. The performance is slightly better than AAU1, but worse than the new design of AAU1. Nevertheless, the performance difference is not significant. Unlike the gate matrix designs, the performance limitation of AAU2 is due to the counter cells. The adder cells of AAU2 can work up to 7 MHz under the test of the most critical propagation delay, which is better than those of AAU1 and its improved version. However, the counter cells of AAU2 can only work up to 6 MHz. Above 6 MHz, a charge sharing problem shown on the oscilloscope will introduce some noise, and thus limits the performance.

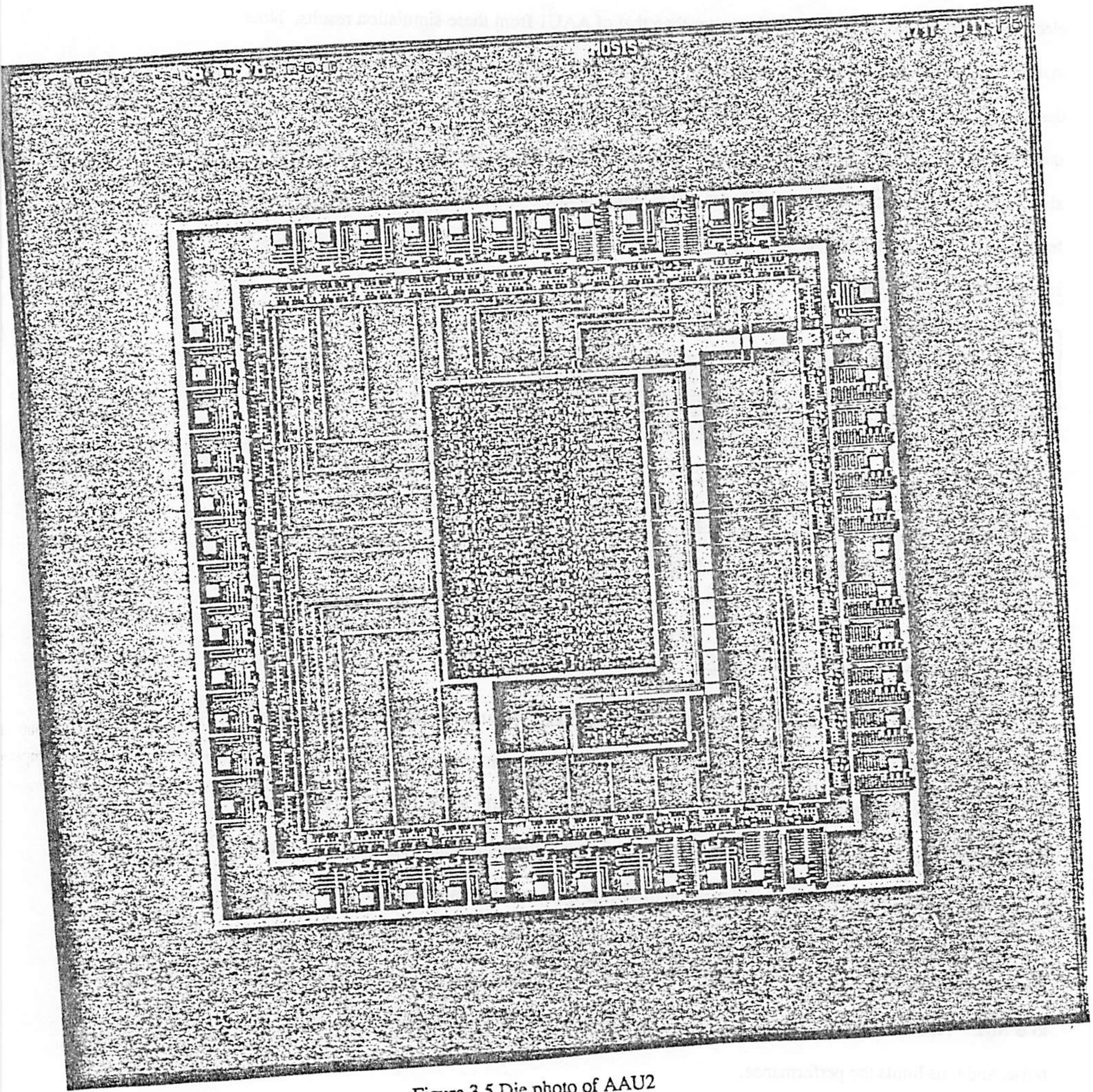


Figure 3.5 Die photo of AAU2

3.5. Comments on Sea of Gates Design

Sea of gates design meets most of the goals we set up in Chapter 1.

(1) It greatly simplifies and unifies layout procedures by its regular structure.

(2) All the cell design is technology independent. We only need to redesign the template accordingly when technology changes. The cell design which includes only the first and the second level metal routing will remain the same.

(3) From the simulation results, the electrical performance for the sea of gates design is better than that of the gate matrix design. The test results show that these two designs have approximately the same performance and AAU2 is slightly better than AAU1.

(4) Fast turnaround is an important advantage of sea of gates design over other array type designs. Wafers can be made just by going through the final stage of metalization; hence, significant savings in both cost and time can be attained.

The decreased layout density is an important handicap of sea of gates design. AAU2 is 70% larger than AAU1. Further optimization in area can be made by properly choosing the number of grids per cell column. For an ASIC chip, memory costs most of the chip area. A reasonable increase in datapath area is not as significant as that in memory cell design. With all these features, sea of gates design will become a very important design style for datapath design. ASICs with 13K usable gate arrays and 4Kb SRAM on a single chip based on the Compacted Array design [Chan87] have been built. This implies that the area problem can be solved when more advanced technology is available since the design even used stronger transistor templates. In addition, useful CAD tools have been provided to facilitate the design. Sea of gates design will replace gate array design in the future and become one of the most important layout styles since the weaknesses of gate array design have been substantially removed by this new array type design.

CHAPTER 4

STANDARD CELL DESIGN

4.1. Standard Cell Library From MSU

Standard cell design is one of the most important structured type layouts. Unlike gate array design, standard cell design has no predefined routing channels, and thus can make use of chip area more efficiently. Basically, standard cell design needs a standard cell library in which all the cells have the same height. Other characteristics of the cells in the library are that the power and the ground buses are connected automatically when cells are juxtaposed and that all the data and control terminals are brought out to the top or to the bottom boundaries of the cells for routing.

The chip layout of a standard cell design has interleaved cell rows and routing channels. The width of the cell rows is fixed and equal to the height of the cells in the library. On the contrary, the width of the routing channels is variable and depends on the number of tracks needed for routing. Currently, we have a standard cell library designed by Mississippi State University (MSU). This library contains some logic cells, latches, buffers, multiplexers, and pads. All these cells are ordered by a four-digit number. For examples, 1100 is a dual inverter cell, 1340 is a tri-state buffer etc. The heights of these cells are all 109 lambda. Most of the transistors inside the cells are large. A typical transistor has a width ranging from 20 to 60 lambda. The reason for this kind of design is to provide strong driving capability. Nevertheless, when driven by other circuitry, these cells may need a buffer stage in front of them to provide enough drive.

The cell library documentation provides the SPICE simulation results as well. Simulation runs are based on normal-case and worst-case delay respectively. Simulation results of various technologies are also available. The delay of a certain cell is calculated by assuming that the cell is driving the inverter cell of the library. Relevant delay information about driving other cells can be calculated from the delay provided. We can take those data as a reference since the information consists of simulation results. It is advised to build up test chips or do more precise simulation to obtain more reliable circuit performance in designing a datapath which has critical timing requirements.

4.2. CAD Tools Used for Standard Cell Design

With the standard cell library, we tried to build up a test chip AAU3 using the same logic as AAU1. The design involves a number of CAD tools. By using these tools, we can simply specify high level inputs and get the layout. Taking the IX section of AAU3 as an example, the design went through the following steps.

(1) Divide the datapath into different cells which are available in the cell library. Connect these cells properly and draw a block diagram for the datapath. This is just a preparation stage which does not need any CAD tools. Figure 4.1 shows the block diagram of the IX section which is going to be laid out using standard cell design.

(2) According to the block diagram, prepare a LISP like syntax input for a program called "eqn2sdl" which will generate sdl syntax as its output. "Eqn2sdl" provides many desirable features which enable users to prepare .sdl files without getting into the messy cell specification and net assignment business. Figure 4.2 is the input file to "eqn2sdl" corresponding to the block diagram of Figure 4.1. This file contains parent terminal declarations, internal terminal declarations, and some logic expressions which specify the connection of these terminals. After preparing this file, we can fire up "eqn2sdl" and get the .sdl file for the next step.

(3) Use design manager [Shun87] to run different layout generators. For our case, we need "wolfe" to do the standard cell placement and routing [Rude87] [Brau86]. Therefore, the design manager will send necessary information to "wolfe", then "wolfe" will make use of these information and "Oct" database to generate the layout. Other layout generators, such as Timlager and the CMOS datapath compiler, can also be called by the design manager to generate layouts. After this process, we got the layout as shown in Figure 4.3 which is a one-bit single section AAU in standard cell design.

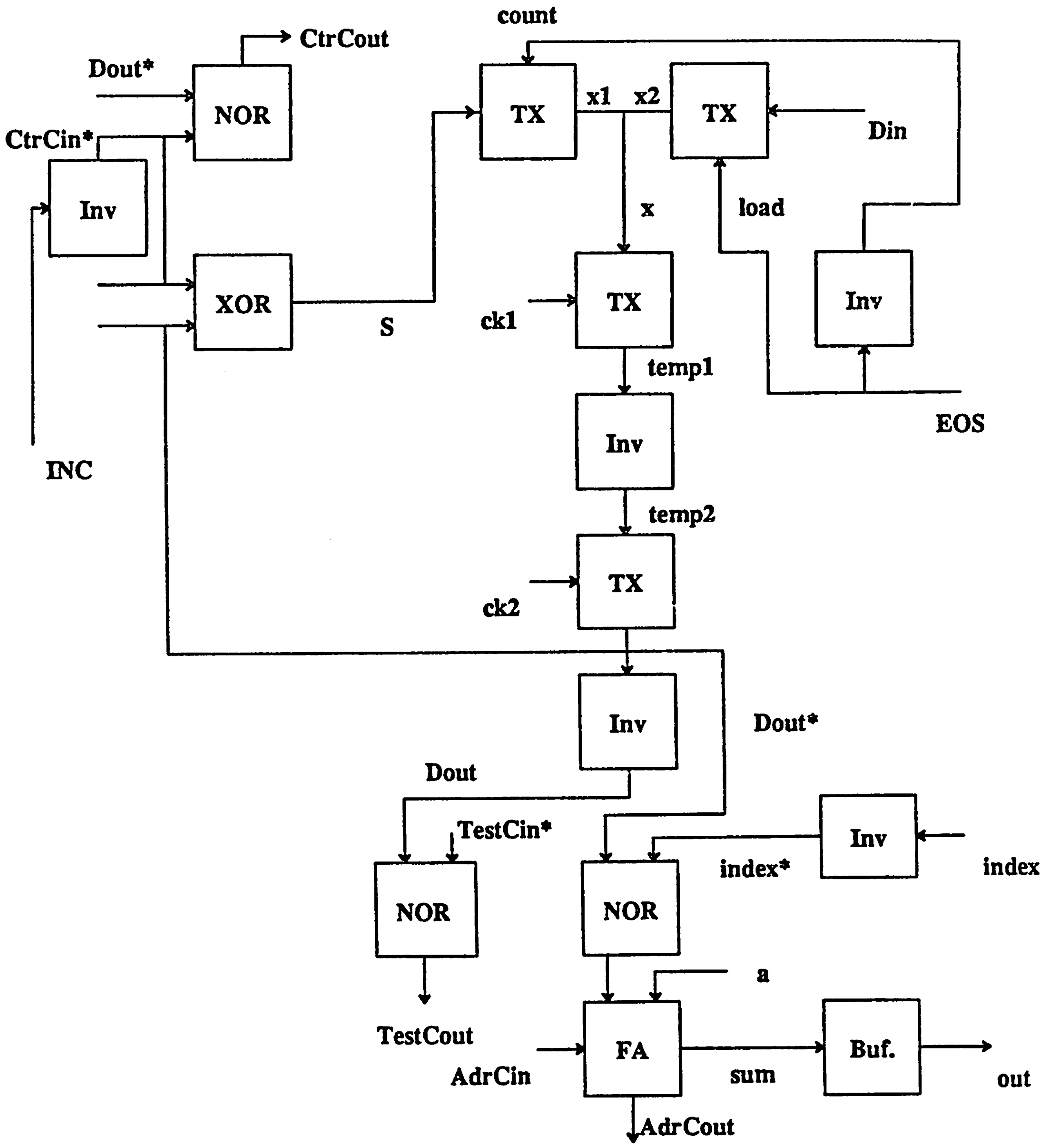


Figure 4.1 Block diagram of IX section

```
(parent! CtrCout INC EOS Din ck1 ck2 test
                index a AdrCout out)
(var! AdrCin CtrCin* load count TestCin* TestCout index*
                temp1 temp2 x1 x2 x s b Dout Dout* sum)
;declare useful functions
(defun del1! (x) (del! x PHI1))
(defun del2! (x) (del! x PHI2))

;Control slice
(set! index* (not! index))
(set! CtrCin* (not! INC))
(set! count (not! EOS))
(set! load (eql! EOS))
(set! test (eql! TestCout))

;Ground slice
(set! AdrCin (zero!))
(set! TestCin* (zero!))

;1 bit
(set! CtrCout (nor! CtrCin* Dout*))
(set! s (xor! CtrCin* Dout*))
(set! x1 (del! s count))
(set! x2 (del! Din load))
(set! x (merge! x1 x2))
(set! temp1 (del! x ck1))
(set! temp2 (not! temp1))
(set! Dout* (del! temp2 ck2))
(set! Dout (not! Dout*))
(set! TestCout (nor! TestCin* Dout))
(set! b (nor! index* Dout*))
(set! AdrCout (carry! a b AdrCin))
(set! sum (sum! a b AdrCin))
(set! out (eql! sum))
```

Figure 4.2 Input file of "eqn2sdl" for generating IX section

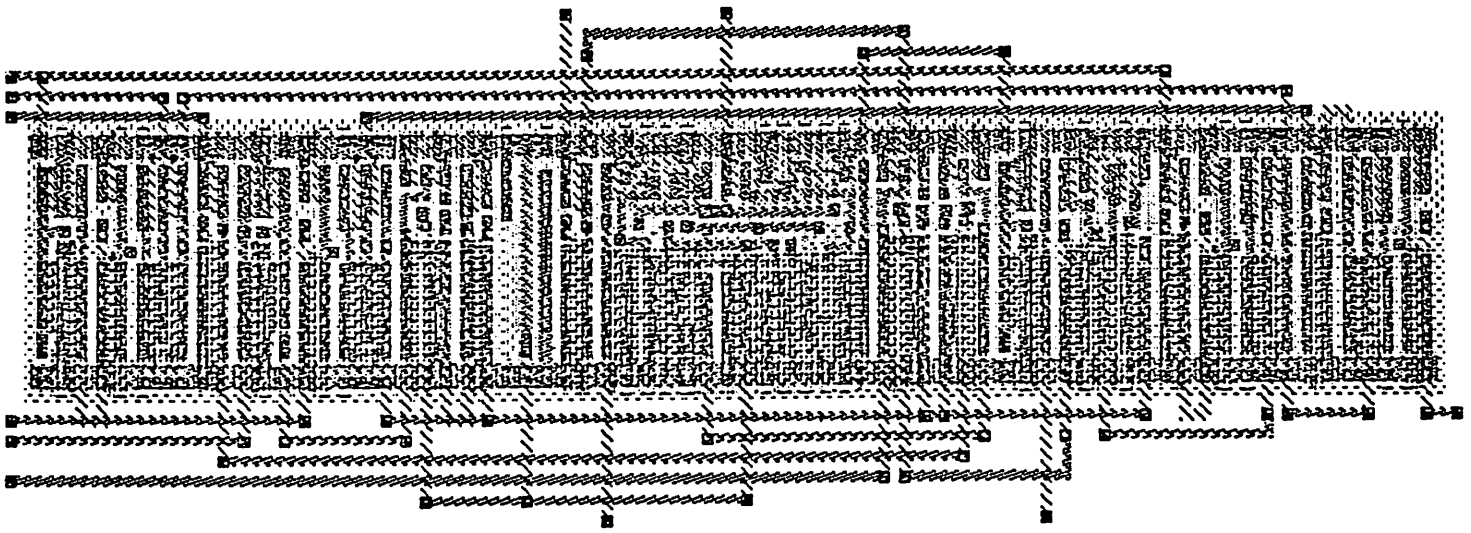


Figure 4.3 Standard cell layout of IX section

Using the same process, we designed AAU3 which requires a more complicated input and longer debugging time. The layout of AAU3 is shown in Figure 4.4. The area of the 10-bit double section AAU is 1628 lambda * 1220 lambda. Compared with AAU1 which has an area of 762 lambda * 1083 lambda and AAU2 which has area of 1413 lambda * 993 lambda, AAU3 is even more area consuming. The reasons are first, the transistors are much bigger compared with either the gate matrix design or the sea of gates design. Secondly, the placement and routing is not as efficient as that of the bit-slice structure because of the large number of cells. It is interesting to notice that the standard cell design which is closer to the custom design than the gate array design costs more area than the sea of gates design. This implies that the sea of gates design has greatly improved the area utilization of the original gate array design. If we properly reduce the transistor sizes by 40% of the cells in the library and assume that the placement and routing remains unchanged, we may reduce the area of AAU3 by about 18%. However, the area utilization is still worse than AAU2 by 15%.

After all the CAD tools needed are complete, standard cell design will be attractive because of the ease of the design. If we can further improve the cell library by introducing more cells, properly adjusting cell sizes, and reducing the propagation delay, then this design style will be even more useful.

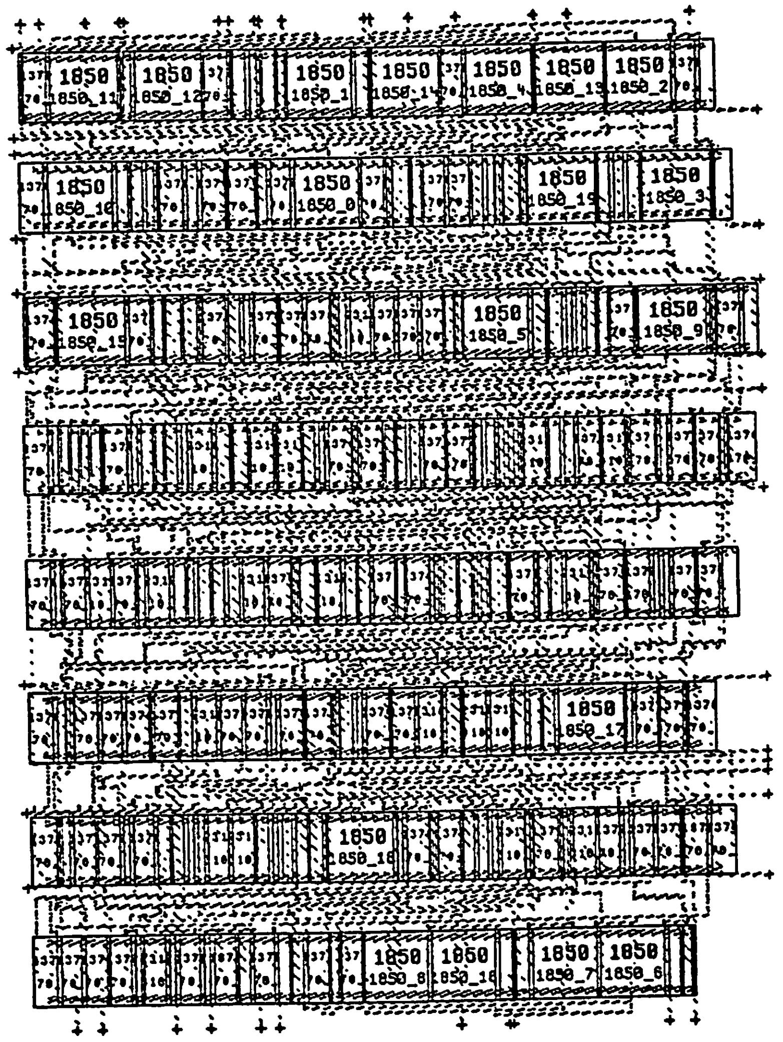


Figure 4.4 Chip layout of AAU3

4.3. Simulation Results

To check the correctness of the logic design of AAU3, we can use ESIM to do the simulation. Another possibility is to use the simulator DSIM built inside the design manager. DSIM can perform different levels of simulation, including function level, gate level, and switch level simulation. The commands to this simulator are very similar to those of ESIM. The main advantage of building a simulator inside the design manager is that we can simulate the operation of a datapath before we actually build it. In addition, we can use the same interconnections for both layout generation and simulation. With this facility, standard cell design is even more effective and attractive.

As described above, the SPICE simulation results for each individual cell are available in the cell library. These results provide designers with the basic information of a cell performance. If the performance does not meet the design requirements, designers will have to increase the transistor sizes or use more cells in parallel to increase the driving capability. According to the SPICE results provided, we expect AAU3 to work up to about 6MHz under nominal case and up to about 4MHz under the worst case. The main reason is that the carry chain of the 10-bit full adder, which is the critical data path of AAU3, takes about 13 nanosecond[1] per bit under the nominal case and about 20 nanosecond under the worst case. The performance is much worse than the simulation results of either the gate matrix design AAU1 or the sea of gates design AAU2. Therefore, we need to redesign the adder cell for a better performance. In fact, the adder cell in the cell library was not meant to be used for datapath design. To implement a datapath, designers should design their own cells for critical path usage.

For completeness, we want to point out that the SPICE simulation results provided in the cell library are dubious. Take the adder cell for an example, the worst case propagation delay 1 -> 0 of CI -> CO is 16 nanosecond for 3 micron technology and 1.2 nanosecond for 2 micron technology. SPICE simulation was done for this case and the results were 8 nanosecond for 3 micron technology and 3.5 nanosecond for 2 micron technology. Some other examples confirm us our belief that more precise

Notice that the carry-in signal of a full adder has to drive three inverters, i.e., the fanout is three. Therefore, the propagation delay provided in the cell library has to be properly modified to get the correct propagation delay for each stage.

simulation is necessary to complete the cell library.

4.4. Comments on Standard Cell Design

CAD tools are now available for standard cell design. The design time will be greatly reduced in the near future when all these tools are fully developed. Another even more important issue of standard cell design is a complete cell library. The cell library available now is not complete yet. More information, such as measured propagation delay for each cell, more precise simulation results, and documentation on logic diagrams, circuit diagrams, naming conventions etc. has to be included to make the cell library more useful. In addition, some of the cells must be redesigned to achieve better performance. Standard cell design will not be adopted until a fully functional cell library is achieved.

From the design experience of AAU3, we concluded that the current status of standard cell design is not appropriate for a datapath design for the following reasons:

(1) The electrical performance does not meet our goal of 10MHz operation frequency.

(2) The area used is much bigger than other array-type layouts.

(3) Turnaround time can not be reduced by standard cell design.

Although standard cell design is not a good choice for a full datapath design, it is suitable for some random logic design, such as the design of a control slice. When timing requirements and area consideration are not critical, standard cell design is still a good candidate for its ease of design. The impact of technology changes on standard cell design is not serious since we only need to update the cell library. All the placement and routing can be easily upgraded by rerunning "wolfe". With these advantages, standard cell design will still be a good choice on some random logic designs [Aldr87] in the near future.

CHAPTER 5

COMPARISON AND CONCLUSION

5.1. Laser Restructurable Techniques in Array Type Layouts

Laser Restructurable Techniques (LRT) were first used in wafer scale integration to improve yield rates [Garv83] [Raff85]. The idea was to make use of laser techniques to remove or form connections on fabricated wafers in order to interconnect working cells and wire around defective ones, thereby incorporating redundancy for improved yield on very large area circuits. A working system is being built in Lincoln Laboratory and a new "diode-link" has been developed that can be manufactured as part of a normal CMOS process. Some ASICs based on this technology have been designed, an example is a speech recognition system built by Lincoln Laboratory using Dynamic Time Warping Level-Building Algorithm [Mann86].

LRT can also be used in array type layouts to achieve very short turnaround time [Orba87]. A company called Western Microtechnology has used the laser techniques to provide customers with one day turnaround time laser programmed arrays [Bois86]. 1410 equivalent 2-input NAND gates are designed on a single chip to provide 80 selectable macrocell functions. Each chip has 92 bonding pads including 84 configurable I/O pads and 8 power pads. Figure 5.1 shows the primitive cell layout and Figure 5.2 shows how to use this cell to implement a 2-input NOR gate. There are 15 switches in a single cell with only four transistors. Therefore, the area utilization for transistors is very low. Much more restricted design rules for laser restructurable switches makes the utilization even worse. To improve the area utilization, we should try to reduce the number of switches per cell. Of course, technology changes, which allow less restricted design rules, can also improve the area utilization.

Using the technology developed by Lincoln Laboratory for wafer scale integration, we tried to layout the primitive cell of Figure 5.1. The cell size is 60 lambda * 62 lambda which can achieve about the same layout density as that designed by Western Microtechnology[1]. Figure 5.3 shows the layout. We can see that most of the area is occupied by the metal cuts or the diffusion links, active area is only a small part of the whole cell.

Another primitive cell design is proposed to improve the area utilization. Figure 5.4 is the new primitive cell design which makes better use of the cell area. A cell area of 58 lambda* 46 lambda which contains even more switches inside the cell is achieved. In addition, the cell has a much more regular structure which helps designers in dealing with the massive laser restructurable switches. Other advantages are that wider transistors are used to increase the driving capability, more switches are added to make the design more efficient and flexible, and like the sea of gates design, cell columns can also be used as routing channels, and thus greatly increase the chip area utilization.

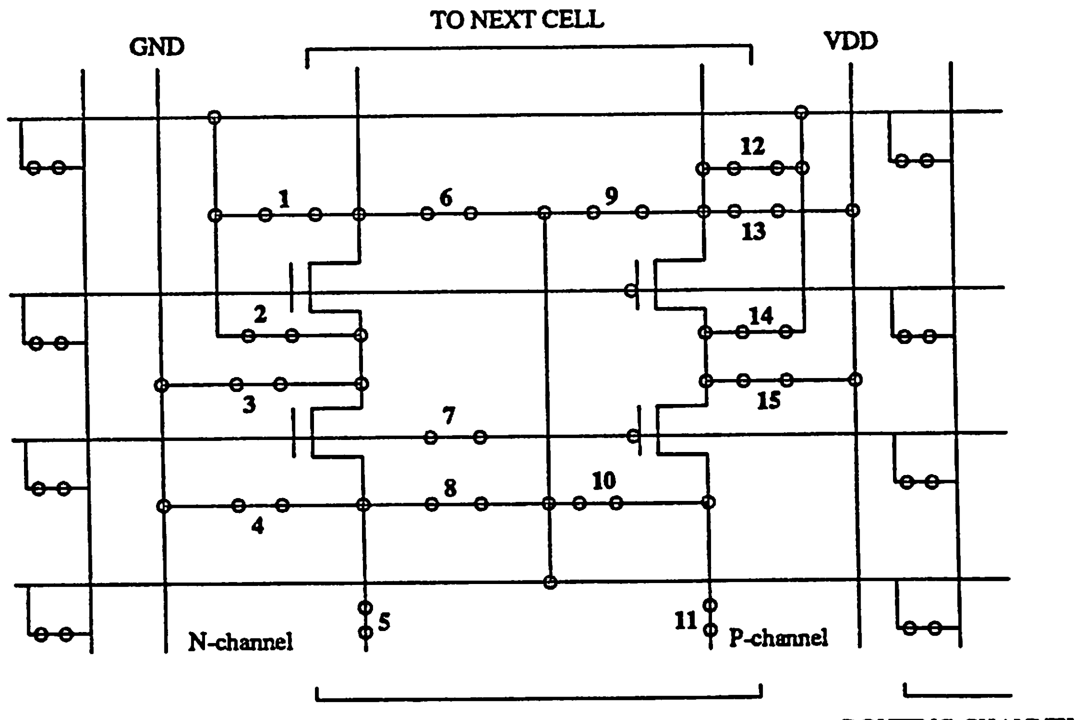


Figure 5.1 Primitive cell layout

[1] The laser technology used by Western Microtechnology can only cut metals. They do not have diffusion links.

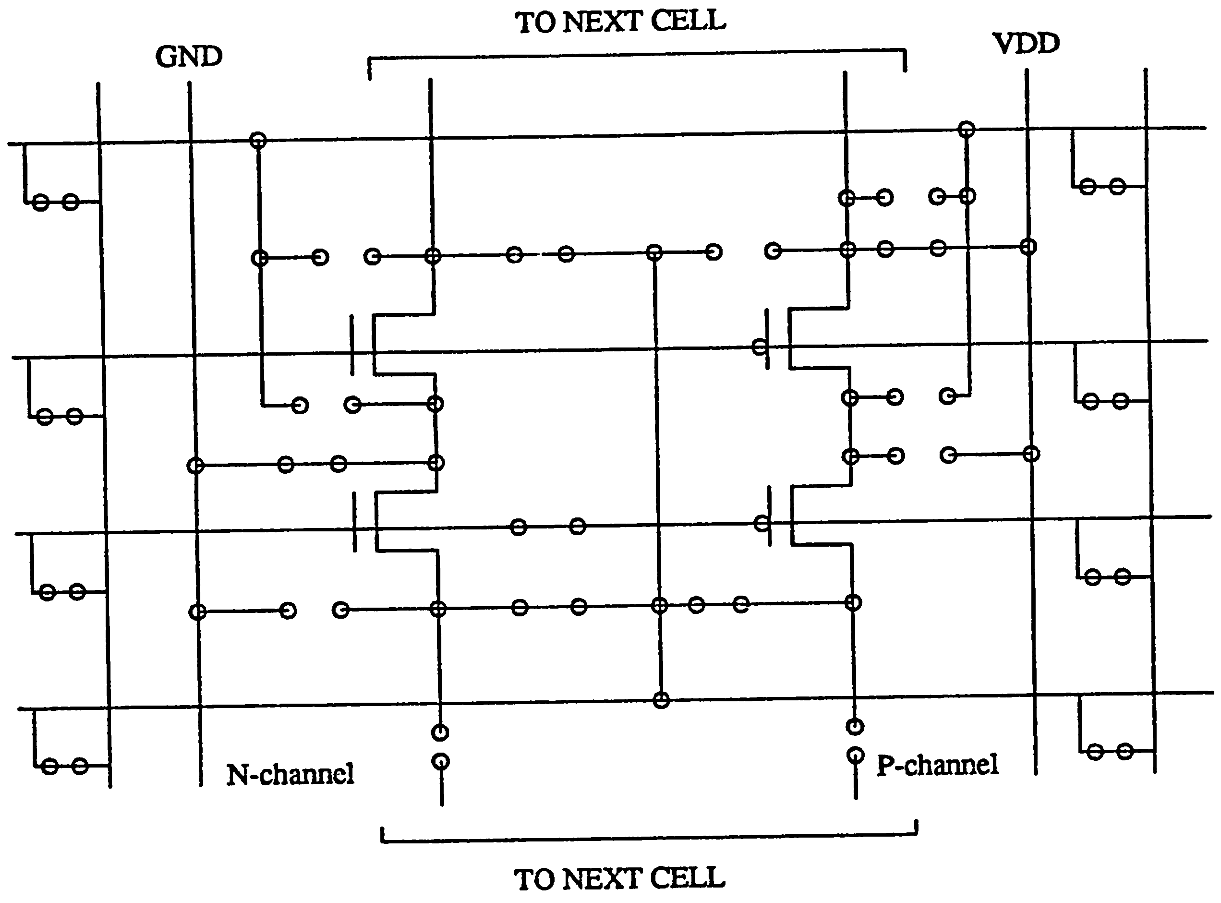


Figure 5.2 Laser cuts for a 2-input NOR gate

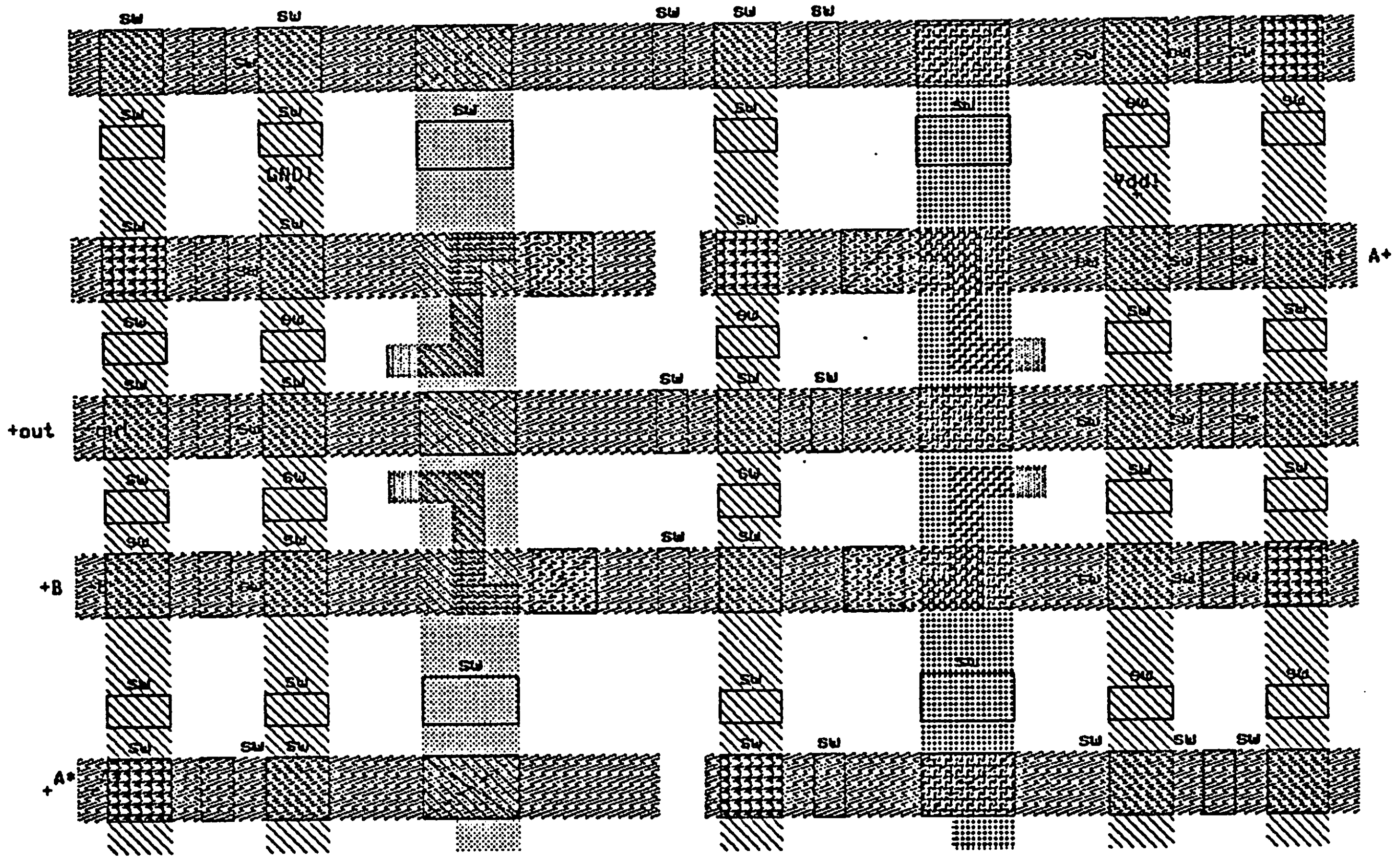


Figure 5.3 Layout of the primitive cell shown in Figure 5.1

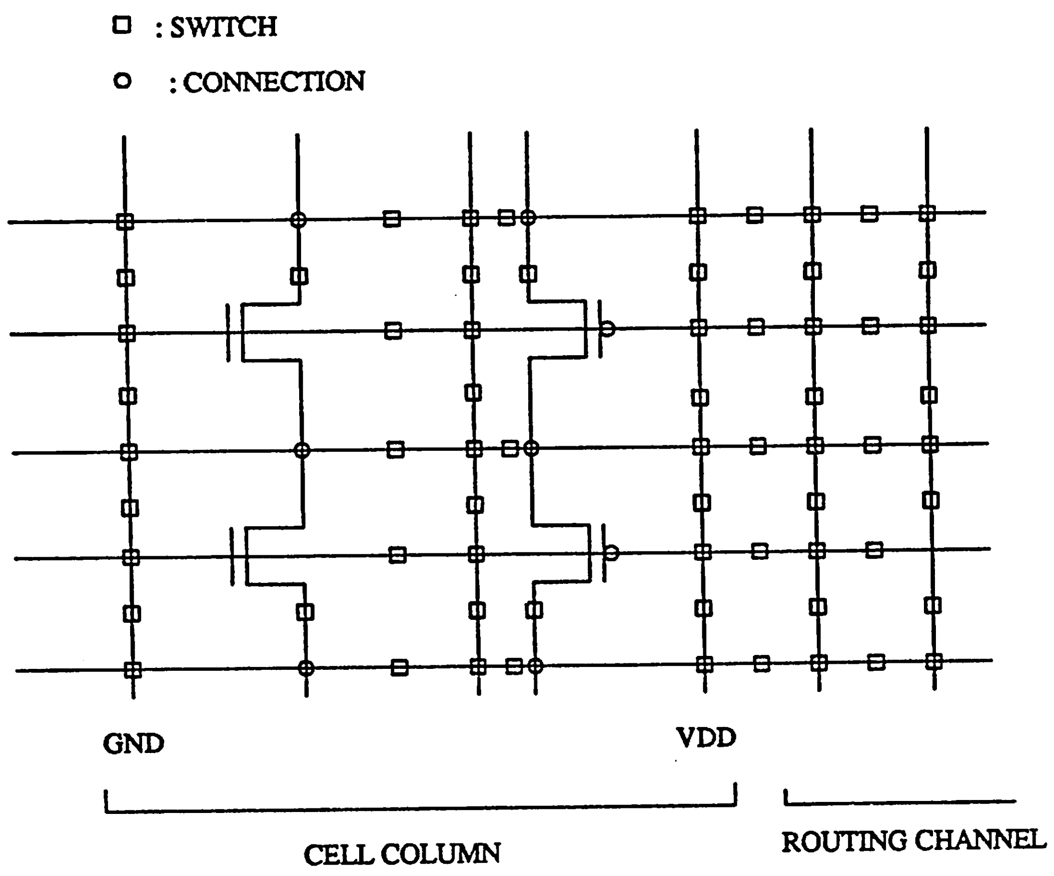


Figure 5.4 New primitive cell layout

Some cells have been laid out using the primitive cell. Figure 5.5 is an example which shows a 2-input XOR gate layout. More complicated cells, such as full adder cells and counter cells can also be laid out easily by diffusion links or metal cuts. The counter cell has an area of 188 lambda * 88 lambda. Compared with the gate matrix layout of the same counter cell which has an area of 102 lambda * 64 lambda or the sea of gates layout with a size of 113 lambda * 81 lambda, this counter cell has an acceptable area utilization. As to the electrical performance, since all the counter cells use the same logic, they have approximately the same propagation delay. According to the SPICE simulation results, the carry propagation delay of the counter cell using LRT is 4 nanosecond, the same as that of the gate matrix design.

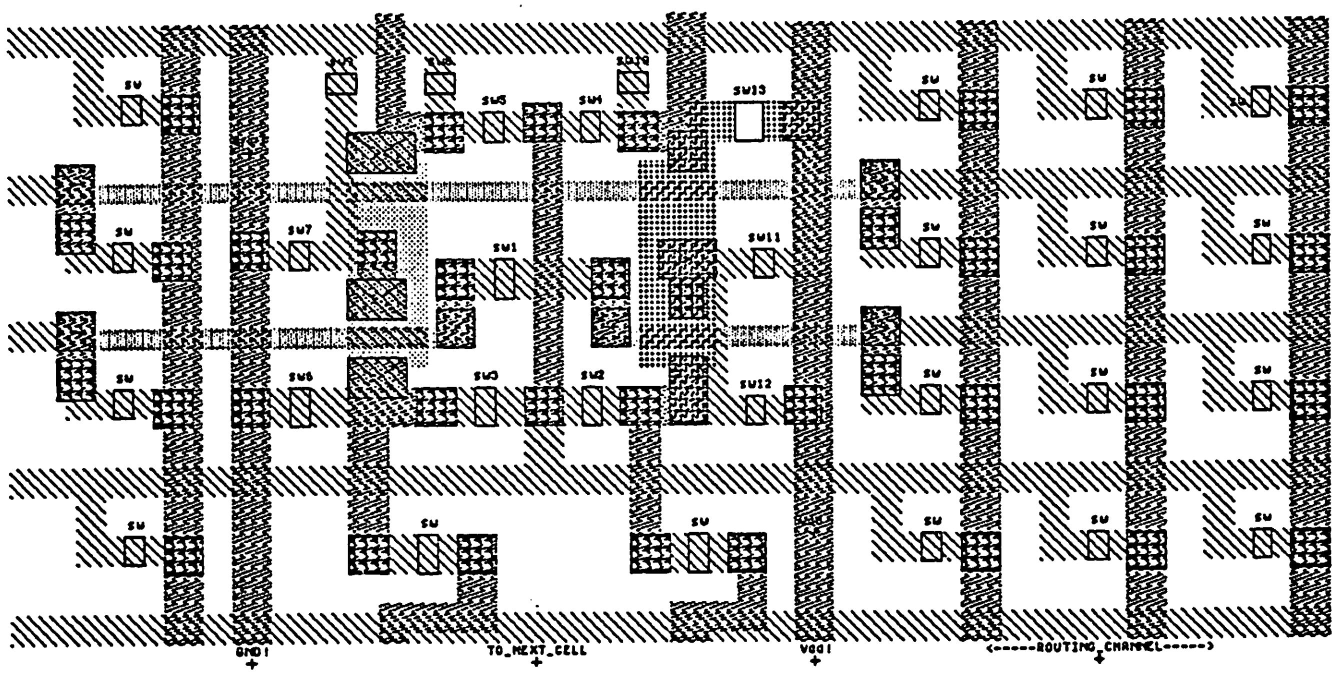


Figure 5.5 A 2-input XOR gate layout using LRT

Basically, this kind of design is very similar to gate array design. The only difference is that instead of using another mask for metal routing, laser restructurable techniques are used to cut metal or connect diffusion; therefore, no mask process is needed for routing and very short turnaround time can be achieved. The greatest problem of using LRT in array type layout is its low circuit density. However, as technology improves, we should be able to attain a higher area utilization.

All the features of gate array design can be obtained by the LRT design. The design process is greatly simplified compared to custom design. Designers only need to decide which pieces of metal to cut or which pieces of diffusion to connect, then a chip is made. No modification is necessary when technology changes since a chip is simply "a set of specifications on switches". The electrical performance is about the same as the other array type designs. In conclusion, using laser techniques in chip layout can achieve most of the goals we set up in Chapter 1 and is particularly interesting due to the feature of very fast turnaround.

5.2. Another Configurable Array Structure Design -- LCA

Logic Cell Array (LCA) is a user programmable reconfigurable logic array developed by Xilinx Incorporation [Cart86] [Hsie87]. Although it is also a restructurable array, it is different from the layout using LRT as described in the previous section. All the structures of LCA are electrically programmed; therefore, we can easily reprogram the array by specifying a different configuration. A graphic design system called XACT has been developed to specify the LCA designs. It contains some software and hardware packages including the LCA editor, a timing analyzer, a simulator, and the XACTOR development tools. The basic package runs on an IBM PC/XT or AT compatible computer with a color monitor and a mouse. Generally speaking, the whole system is very well developed and easy to use.

Three basic building blocks are contained in a LCA device: Configurable Logic Blocks (CLBs), Input/Output Blocks (IOBs), and Interconnects. A LCA device has 64 CLBs, arranged in an 8-by-8 matrix. Interconnects occupy the space between the rows and columns of CLBs and between the CLBs and the surrounding IOBs.

CLBs are the logic and storage circuitry. They are organized in a matrix and have outputs that implement the truth table or the Karnaugh map of their inputs. The logic element of a CLB can generate any combinational-logic function of its four inputs. The CLB also has a general purpose storage element that can be used to implement sequential functions.

IOBs provide the interface between the external pins and LCA internal resources. Any IOB can be defined as an input, an output or a bi-directional element with a tri-state control on the output. Each IOB also contains a flip-flop, which can capture input data and provide captured data as an alternative or direct-input data.

Interconnect wires connect CLBs and IOBs. When a function is configured, "switches" are used to connect interconnect segments to CLBs or IOBs. Different types of interconnect are available for different signal-routing requirements.

Configurations of LCA devices are specified with XACT design system, which produces configuration data. The configuration data is loaded into an LCA, enabling it to perform the functions. When power is removed from the LCA, the configuration is lost and the LCA returns to the unconfigured state. The configuration data can be passed to an LCA device from either an external memory using parallel stream or an external processor using a serial bit-stream. Multiple LCAs can be daisy-chained to be programmed simultaneously.

Figure 5.6, 5.7, and 5.8 show the three building blocks of LCAs. Three types of interconnect: direct interconnect, local interconnect, and long lines, as shown in Figure 5.8, perform connections under different situations. All interconnections are implemented by switches which can be distinguished as programmable interconnect points and switching matrices respectively.

An LCA device has a chip size of 7.92mm * 7.55mm under 2u CMOS double metal technology. LCAs have a lot of desirable features:

(1) The design process is very easy. To design a chip, designers only need to specify the logic functions of CLBs and IOBs and the interconnections between these blocks. In effect, there is no layout or circuit design involved. All the design belongs to logic level design. In addition, XACT system is well developed and easy to use. The timing analyzer, the simulator interface, and the design-rule checker greatly help designers to understand the circuit performance.

(2) LCA design is technology independent since all the design is logic level design. When new technology is available, new LCA devices will be used and the configuration data remain unchanged.

(3) Compared to EPROMs or EEPROMs, LCAs use dual port static memory, thus very short programming time is achieved. The ability of quick reconfiguration makes exhaustive testing practical.

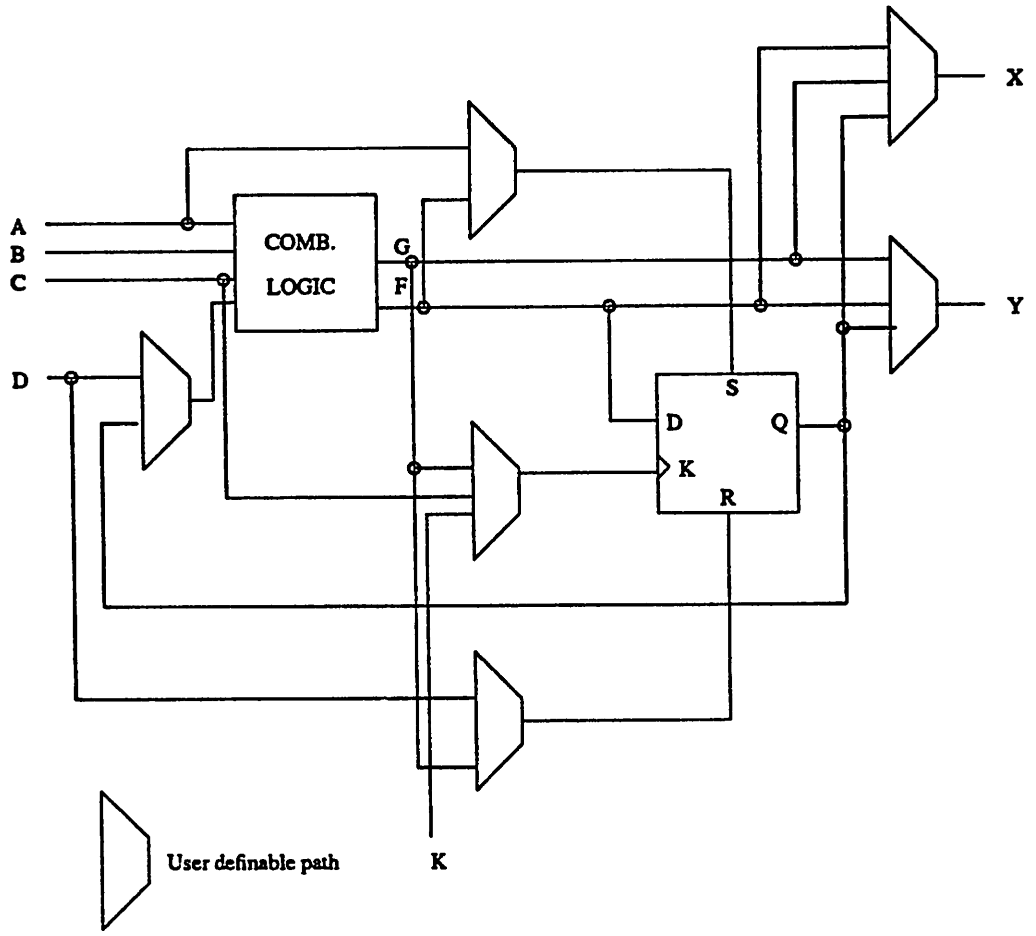


Figure 5.6 CLB block diagram

However, like laser restructurable layout, LCAs can not perform very complicated functions on a single chip. In order to achieve high flexibility, LCAs have to use crossbar switches to implement interconnections and use PLAs (Programmable Logic Arrays) like memory arrays to implement CLBs. In other words, RAM cells (5-T SRAM) are used in all possible interconnections and probably inside CLBs too. A lot of redundancy is provided as a tradeoff for programmability. Therefore, a very complicated chip only has the logic capability of a 1000 to 1600 unit cell gate array.

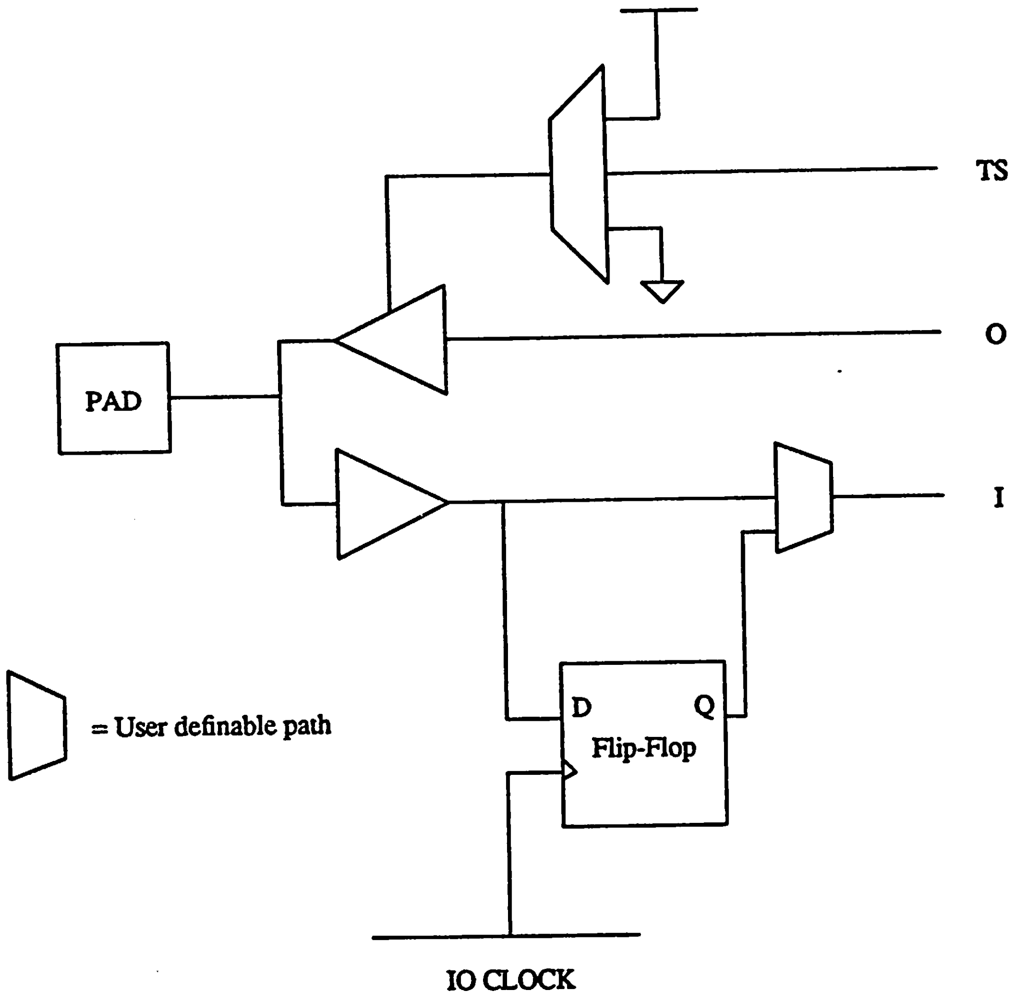


Figure 5.7 IOB block diagram

Another problem inherent in the LCA design is that its electrical performance is difficult to predict. Nevertheless, this is a natural consequence of logic design and can only be improved by use of better technology. The development of the LCA system involves new exploration of array type design. The easy learning capability of XACT tools, the idea of electrical programmability, and the high flexibility of chip usage are the greatest achievements in developing the LCA system.

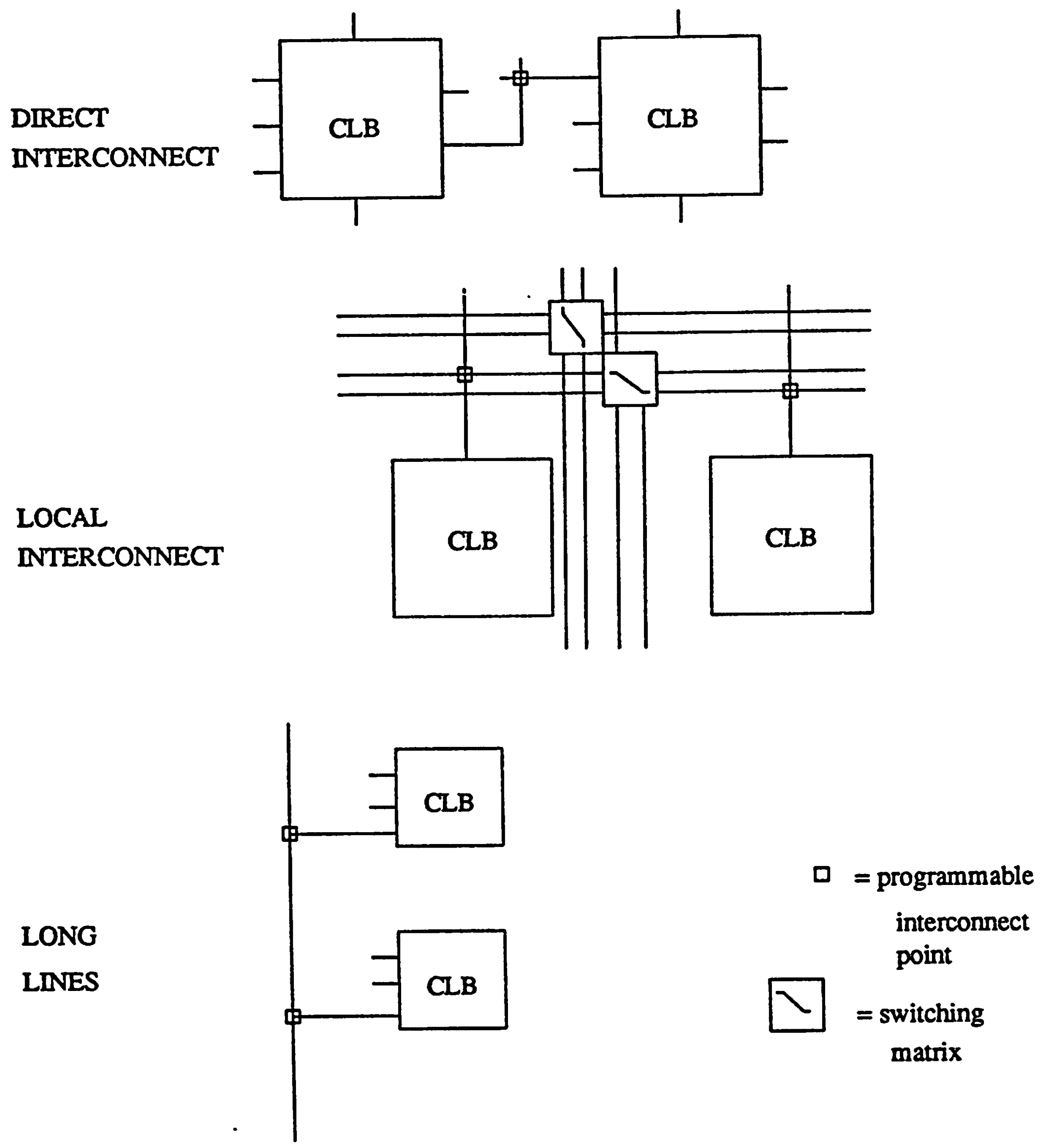


Figure 5.8 Types of interconnect and switches

5.3. Comparison of Layout Techniques

Using array type layouts is the key to simplifying the design procedure. All the array type layouts discussed so far have simplified layout procedure by using orderly structures. However, different types of layout have very diverse features and are suitable for various applications. Table 5.1 is the test results summary of the AAU designs in different layouts and Table 5.2 is a summary and comparison of the array type layouts described in this report. From these tables, we can easily make judgements on choosing a proper layout style.

Table 5.1 Summary of AAU Test Results

Chip Name	Layout Style	Chip Area (lambda * lambda)	Highest Working Frequency (MHz)	Performance Limit Factor
AAU1	gate matrix	762 * 1083	5.6	adder
AAU1 (new version)	gate matrix	741 *1089	6.5	adder
AAU2	sea of gates	1413 * 993	6.0	counter
AAU3*	standard cell	1628 * 1220	4 - 6	NA

* AAU3 is being fabricated by MOSIS on run M76G; the data listed are based on simulation results.

Table 5.2 Comparison of Array Type Layouts

Layout Style	Orderly Structure	Technology Change	Electrical Performance*	Turnaround Time	Design Time	Area Utilization**
Gate Matrix	yes	no change on symbolic inputs	slightly worse than custom design	same as custom design	shorter than custom design	~20% worse than custom design (1)
Standard Cell	yes	redesign cell library	depends on cell library	same as custom design	shorter than gate matrix	(3)
Gate Array	yes	redesign template	depends on template design	shorter than custom design	shorter than gate matrix	(4)
Sea of Gates	yes	redesign template	slightly better than gate matrix	same as gate array	same as gate array	~70% worse than gate matrix (2)
LRT	yes	redesign template	depends on template design	very fast turnaround ~1 day	same as gate array	~ (4)***
LCA****	yes	redesign LCA devices	NA	NA	programming time	~ (4)

* Electrical performance depends highly on logic and circuit design; readers should refer to simulation and test results for detail performance.

** 1 means best possible on this chart, 2 means second best, etc. The data given are based on the AAU designs of different layout styles.

*** Area utilization depends on template design. The rank here is based on the data given by Western Microtechnology.

**** In effect, LCA design is not a layout style, but a logic design.

Among all these layout styles, gate-array-type layouts, including gate arrays, sea of gates, and LRT design can be most easily upgraded when technology changes. Gate matrix design has the best area utilization since it is closer to custom design than any other array type layout. LRT design has the fastest turnaround through its use of laser technology. LCA design has the shortest design time, because it is basically a logic level design. However, taking all the characteristics into consideration, the sea of gates design, which has high performance in almost all respects, is the best layout style.

5.4. Final Remarks and Future Work

Much research is devoted to investigating various layout styles. All these new design methods aim for simplicity and regularity. Custom design which aims for electrical performance but takes long design times will only be adopted when area or timing considerations are very critical, such as in the memory cell design or very high frequency datapath design. Array type design which takes considerably less design time enables more and more ASICs to be designed to greatly improve system performance. Therefore, array type layout will be the main design methodology in the future.

It is important to survey the array type layouts presently available while investigating some new design methods. That was also the motivation of this research. Through all this research, we get to understand the features of different layouts; and, through our discussion, we are able to propose some solutions to make these array type layouts more useful.

Chip layout is highly susceptible to technology changes. But array type layout can be easily upgraded, so it is much more adaptable than custom design. New technology will also activate new array type design. Using laser techniques in chip design is a good example. With the same technology, the quality of array type design influences electrical performance. From this report, we understand the tradeoffs between different array type designs and their characteristics.

As to memory cell design, custom design of individual cells and then using module generators to generate a whole memory array is the most popular way. Sea of gates arrays and gate arrays usually have on-chip RAM to increase processor speed. Examples are LSA1500 series [Chan87] and [Kawa87]. Using sea of gates templates to design memory cells is also considered. A 3-transistor RAM cell is laid out with an area of 62 lambda * 27 lambda. Compared with the custom design of the RAM cell, the area is about twice as large and the speed is about the same. Since area consideration is especially important in memory cell design, custom design of memory cells will still be the major design methodology.

It is believed that more research will be directed toward array type design and laser restructurable techniques. New tools are being developed for array type design and more chips are being laid out in array styles. In particular, tools for standard cell design in LAGER III and tools for sea of gates design will be available in the near future. If the laser technology is available, we will also try to develop CAD tools for this kind of layout and design prototypes for evaluation. All the work will have great significance on the future ASIC design.

REFERENCE

[Aldr87] Aldridge, A. W., R. F. Keil, J. H. Panner, G. D. Pittman, and D. R. Thomas, "A 40K Equivalent Gate CMOS Standard Cell Chip," in Proceedings of the IEEE 1987 Custom Integraed Circuit Conference, pp. 248-252.

[Asad87] Asada, K. and J. Mavor, "A MOS Leaf-cell Generation System from Boolean Expressions," in Proceedings of the IEEE 1987 Custom Integraed Circuit Conference, pp. 21-24.

[Bois86] Boisvert, C. J., "One Day Prototype Laser Programmed Arrays," Product Report, Western Microtechnology, June 1986.

[Brau86] Braun, D., C. Sechen, and A. Sangiovanni-Vincentelli, " ThunderBird: A Complete Standard Cell Layout System," in Proceedings of the IEEE 1986 Custom Integraed Circuit Conference, pp. 276-280.

[Cart86] Carter, W. S., K. Duong, R. H. Freeman, H-C Hsieh, J. Y. Ja, J. E. Mahoney, L. T. Ngo, and S. L. Sze, "A User Programmable Reconfigurable Logic Array," in Proceedings of the IEEE 1986 Custom Integraed Circuit Conference, pp. 233-235.

[Chan87] Chan, T, A. Yuen, K. Knorpp, M. Hung, P. Tsao, M. Freie, Y. Chang, R. Rasmussen, A. Hui, and P. Yin, " Advanced Structured Arrays Combine High Density Memories with Channel-free Logic Array," in Proceedings of the IEEE 1987 Custom Integrated Circuits Conference, pp. 39-43.

[Garv83] Garverick, S. L. and E. A. Pierce, "A A Single Wafer 16-Point 16-MHz FFT Processor," in Proceedings of the IEEE 1983 Custom Integrated Circuits Conference, May 1983.

[Gold85] Gold, M., "Established Chip Vendors Shift Their Strategy to the Semicustom Market," Electronic Design, p. 29, Oct. 31, 1985.

[Holl87] Hollis, E. E., "Design of VLSI Gate Array ICs," Prentice-hall, Inc., 1987.

[Hsie87] Hsieh, H-C, K. Duong, J. Y. Ja, R. Kanazawa, L. T. Ngo, L. G. Tinkey, W. S. Carter, and R. H. Freeman, "A Second Generation User-programmable Gate Array," in Proceedings of the IEEE 1987 Custom Integraed Circuit Conference, pp. 515-521.

[Hui85] Hui, A., A. Wong, C. Dell'oca, D. Wong, and R. Szeto, "A 4.1K Gates Double Metal HCMOS Sea of Gates Array," in Proceedings of the IEEE 1985 Custom Integraed Circuit Conference, pp. 15-17.

[Hsu86] Hsu, C. P., R. A. Perry, S. C. Evans, J. Tang, and Y. Liu, "Automatic Layout of Channelless Gate Array," in Proceedings of the IEEE 1986 Custom Integraed Circuit Conference, pp. 281-284.

[Kawa87] Kawashima, M., M. Takechi, K. Ikuzaki, K. Itoh, M. Fujita, T. Nakao, and I. Masuda, "An 18K 1um CMOS Array with High Testability Structure," in Proceedings of the IEEE 1987 Custom Integraed Circuit Conference, pp. 52-55.

[Lope80] Lopez, A. D. and H. A. Law, "A Dense Gate Matrix Layout Method for MOS VLSI," IEEE Trans. on Electron Devices, vol. ED-27, pp. 1671-1675, Aug. 1980.

[Mann87] Mann, J. R. and F. M. Rhodes, "A Wafer Scale DTW Multiprocessor," in Proceedings of IEEE 1987 ICASSP, Apr. 1986.

[Orba87] Or-Bach, Z., K. Pierce, and S. Nance, "High Density Laser Programmable Gate Array Family," in Proceedings of the IEEE 1987 Custom Integraed Circuit Conference, pp. 526-528.

[Poon85] Poon, T. C., S. J. Hwang, D. C. Fowlis, G. P. Sampson, W. Y. Yang, M. Kerestes, R. F. Fischer, and M. L. Willis, "Design of a High Performance CMOS DRAM Controller Using a Modified Standard Cell Approach," in Proceedings of the IEEE 1985 Custom Integraed Circuit Conference, pp. 26-29.

[Pope85] Pope, S. P., "Automatic Generation of Signal Processing Integrated Circuits," Ph. D. Thesis, U. C. Berkeley, Feb. 22, 1985.

[Raba85] Rabaey, J. M., S. P. Pope, and R. W. Brodersen, "An Integrated Automated Layout Generation System for DSP Circuits," IEEE Trans. on Computer-Aided Design, vol. CAD-4, pp. 285-296, July 1985.

[Raff85] Raffel, J. I., A. H. Andersen, G. H. Chapman, K. H. Konkle, B. Mathur, A. M. Soares, and P. W. Wyatt, "A Wafer-Scale Digital Integrator Using Restructurable VLSI," IEEE Journal of Solid-State Circuits, vol. sc-20, pp. 399-406, Feb. 1985.

[Rude87] Rudell, R., " Wolfe - Oct Interface to the TimberWolf Standard Cell Placement Program," Berkeley CAD Tools User's Manual, 1987.

[Sher86] Sherstinsky, A., "Game - Gate Matrix Layout Translator," Berkeley CAD Tools User's Manual, 1986.

[Shun87] Shung, C-S, J. Rajeev, S. K. Azim, M. B. Srivastava, and R. W. Brodersen, "LAGER-III: A Framework for Algorithm-specific IC Design," submitted to ICCAD conference, 1987.

[Sriv87] Srivastava, M. B., "Automatic Layout Generation of CMOS Data Paths," MS Thesis, U. C. Berkeley, 1987.

[Term82] Terman, C., "Esim - Event Driven Switch Level Simulator," Berkeley CAD Tools User's Manual, 1982.

[Wong86] Wong, T., A. Hui, and D. Wong, "A High Performance 129K Gate CMOS Array," in Proceedings of the IEEE 1986 Custom Integraed Circuit Conference, pp. 568-571.