ANALYSIS OF SEA-OF-GATES TEMPLATE AND CELL LIBRARY DESIGN ISSUES

by

Lorraine S. Layer

Memorandum No. UCB/ERL M88/8

15 January 1988

ANALYSIS OF SEA-OF-GATES TEMPLATE AND CELL LIBRARY DESIGN ISSUES

by

Lorraine S. Layer

Memorandum No. UCB/ERL M88/8

15 January 1988

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

TITLE PAGE

ANALYSIS OF SEA-OF-GATES TEMPLATE AND CELL LIBRARY DESIGN ISSUES

by

Lorraine S. Layer

Memorandum No. UCB/ERL M88/8

15 January 1988

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Acknowledgements

I would like to thank Wayne Christopher, my very good friend, who first encouraged me to take on this project. Wayne helped me think through the many details of the project. Always questioning and willing to help, Wayne brought me into the world of computer programming gently and provided invaluable guidance when I tried to make my ideas compile. His helpfulness both in and out of the office is immeasurable.

I am indebted to Rick Spickelmier, David Harrison, Peter Moore, Rick Rudell, Tom Laidig, Jeff Burns, Ellen Sentovich, and still others in the CAD group for taking time to answer my questions, listen to my dilemmas be they technical or otherwise, and especially for always being there to "fix the Layer problem." David and Rick S. were especially helpful in helping me work with the software being developed at Berkeley, and I thank them.

I would like to thank my advisor, Richard Newton for having great enthusiasm toward the Sea-of-Gates design style and my project in particular. Richard's firm guidance throughout the project kept me on the right track until completion.

I also wish to thank all those who participated in the EE290h Fall '87 course for probing deeply at the many issues covered in this report. Especially Richard Newton and Carlo Sequin who prodded me to work harder and think thoroughly by openly challenging the things that I presented. Their input about my work helped increase my productivity and their openness made me feel comfortable and finally confident.

I greatly appreciate the efforts of Alberto Sangiovanni-Vincentelli and Richard Newton of creating the very active and open research atmosphere of the CAD group which I thoroughly enjoyed being a part of.

Finally I would like to thank my parents, Bill and Isabell, for all of their love and confidence over the past twenty-four years.

TABLE OF CONTENTS

CHAPTER 1

VLSI Design Styles - Advantages and Disadvantages

1.1. Introduction

Integrated circuit complexity is growing steadily. Present day designs contain hundreds of thousands of transistors and future designs are predicted to have a million or more devices on a single chip. As chip size and the number of reliable layers of interconnect increases so does the need for fast, cheap, reliable prototyping of large circuits.

The design style chosen to implement an ASIC (Application Specific Integrated Circuit) is governed by tradeoffs between cost, turn-around time, packing density and performance. A full custom ASIC design usually has a higher packing density and better performance than its semicustom counterpart but the cost is greater and the turn-around time longer. Semicustom designs can be on a programmable array with some layers predefined or standard or macro cell which require customization of all the mask layers. Cell arrangements for several semicustom design styles are shown in Figure 1.1. The advantages and disadvantages of the various types of semicustom design are presented in the following sections.

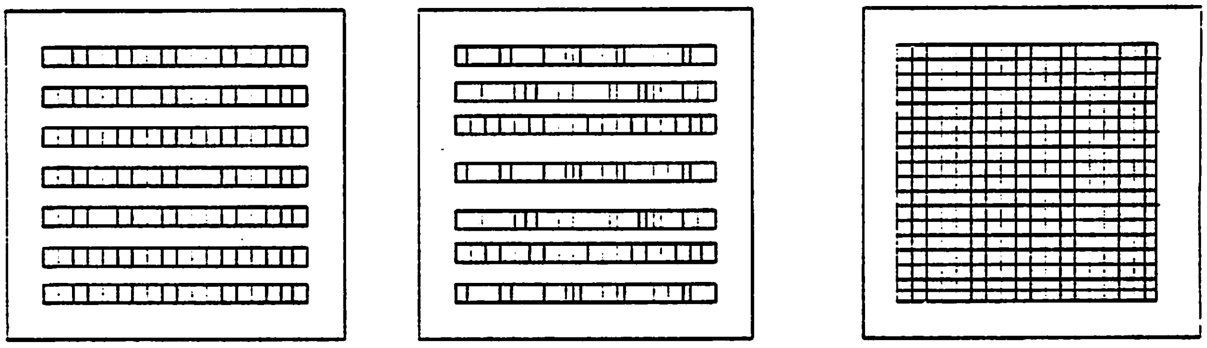

a) Gate array b) Standard cell c) Channelless array

Figure 1.1 Cell arrangements for several semicustom design styles.

1.2. The Gate Array

The *gate-array* (also referred to as master-slice, or uncommitted logic array),Figure 1a, is by far the most common programmable array used for ASIC designs. In this approach, a two-dimensional array of replicated transistors is fabricated to a point just prior to the interconnection levels. A particular circuit function is then implemented by customizing the connections within each local group of transistors, to define its characteristics as a basic cell, and by customizing the interconnections between cells in the array to define the overall circuit. Generally a two-level interconnection scheme is used for signals and, in some approaches, a third, more coarsely defined layer of interconnections is provided for power and ground connections. The interconnections are implemented on a rectilinear grid in the *channels* between the cells.

1.3. Standard Cell Design

The *standard cell* (or polycell) approach refers to a design method where a library of custom-designed cells is used to implement a logic function. These cells are generally of the complexity of simple logic gates or flip-flops and may be restricted to constant height and/or width to aid packing and ease of power distribution. Unlike the programmable array approach, standard cell layout involves the customization of all mask layers. This additional freedom permits variable width channels to be used. While most standard cell systems only permit inter-cell wiring in the channels between rows of cells or through cells via pre-determined "feed-through" cells, some systems permit over-cell routing.

1.4. Macro Cell Design

It is often relatively inefficient to implement all classes of logic functions in a single design approach. For example, a standard cell approach is inefficient for memory circuits such as RAM and stack. In the *macro-cell* method, large circuit blocks, customized to a certain type of logic function, are available in a circuit library. These blocks are of irregular size and shape and may allow functional customization via interconnect, such as a PLA or ROM macro[1], or they can be parameterized with respect to topology as well[2-4]. With the parameterized cell, the number of inputs and outputs may be parameters of the cell. In some systems macro cells may also be embedded in gate-array or standard-cell designs.

1.5. Advantages and Disadvantages of Gate Array Design

The major advantages of gate arrays over design styles which require customization of all mask layers are much lower non-recurring engineering costs and faster turn-around times. The first is very important for low volume parts and both are very important for prototype parts which can change often as the system design progresses.

Conventional gate arrays with fixed routing channels between rows of active devices have distinct disadvantages to other methods of semicustom integrated circuit (IC) design. The most obvious problem with predefined routing tracks is that more tracks must be allocated than will be utilized in most designs in order to ensure routability of congested areas of dense circuits. If the width of routing channels can be increased or decreased as needed, there would be no wasted tracks and the total number or routing tracks would be significantly reduced compared to the total number needed in conventional gate arrays. [5]

Another problem with conventional gate arrays having fixed routing channels is that channels toward the center of the chip may fill up before all signals have been routed causing the remaining signals to use feedthroughs which results in slower signals, or the signals may not be routable at all. Furthermore, unused channel space in less dense sections of the chip is wasted silicon area. Although some conventional gate arrays have wider channels in the middle of the chip where the channel widths are selected based on a statistical argument[6,7], the channel width could still be insufficient for some designs and too much for other designs.

Another drawback with fixed channels between devices is that macrocell implementation such as RAM, ROM, or PLA is extremely costly and impractical.

The standard cell design style relieves some of the drawbacks of conventional gate arrays. Channel widths are variable so that congested areas can have wider channels and signals are guaranteed to be routable. Since all masks are usually customized, it is possible to design dense macrocells that can grow both horizontally and vertically and place them in the circuit with the standard cells. It is also possible to increase transistor size in non-integer multiples as is not the case in gate array where a large transistor is either two, three, or more times as wide as the smallest transistor.

As processing technology is improved and more layers of high quality interconnect are provided, the concept of routing channels becomes less and less important; standard cell designs will map to macrocell and the conventional gate array becomes channelless as in the Sea-of-Gates design style. The channelless gate array retains the advantages of fewer customized masks and therefore faster turn around times and lower cost than standard or macrocell designs. Since circuits implemented in Sea-of-Gates have performance and density close to that of standard cell designs the advantages of customizing a prefabricated array can be exploited without the usual compromises. This is especially significant since recent trends in the ASIC community indicate that gate arrays are satisfactory for high volume parts.

In order for Sea-of-Gates designs to have the density and performance of their standard cell counterparts the core cell of the array must be well designed with a variety of factors in mind. Part of the purpose of this research was to identify the important considerations for the Sea-of-Gates core cell, to review existing architectures, and then to compare several styles within a set of criteria which will be described in next two sections.

CHAPTER 2

Sea-of-Gates Design Considerations and Existing Implementations

As previously stated, the core cell design is crucial for Sea-of-Gates designs to be as dense and as fast as their standard cell and macrocell counterparts. At the same time the core cell must be versatile enough to accommodate many different types of library components efficiently. First it is necessary to identify the important design considerations for both the core cell architecture and the chip size array. In this section the various considerations and tradeoffs that contribute to both the design of the core cell(s) and how they should be organized to form the full array are presented. Examples of existing channelless array architectures are given in order to exemplify the tradeoffs described below.

2.1. Core Cell Design Considerations

The architecture of the basic cells can be distinguished by several factors such as:

- the number, ratio, and relative sizes of n- and p- type transistors
- the isolation techniques used
- the number of routing tracks available

- the number of possible connections to the gates
- whether or not the gates of transistor pairs are connected with polysilicon
- whether the gates run perpendicular to the diffusion or at some angle
- whether the cell architecture is tailored to accommodate a special cell such as RAM, ROM, PLA, or dynamic logic and to what degree, if at all does the specialization cause wasted space or other undesired features
- the degree of symmetry in the core cell

2.2. Full Array Design Considerations

In terms of the array architecture, some distinguishing features include:

- the number of core cell types and how they are distributed
- power and ground bus widths and organization
- whether or not to have prefabricated special blocks such as RAM or ROM

2.3. Existing Implementations

A simple core cell has one n- and one p-transistor of the same size. An example of such a cell is the LSI Logic core cell[8] with n- and p-devices which are 28 microns wide and gate lengths of 1.5 microns as shown in Figure 2.1. With two connections to each gate and four tracks over each transistor, this cell has 12 vertical routing tracks. There are four horizontal tracks; three over the transistor pair and one for contact to the substrate or well. Two of the vertical tracks are used for power and ground busses if the array location is used to implement a

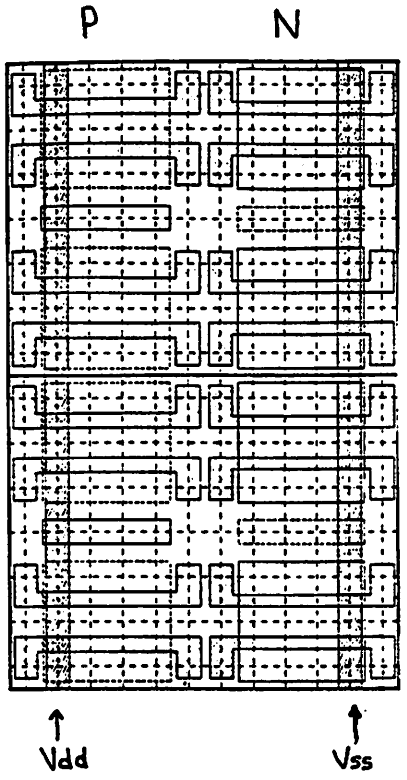


Figure 2.1 The LSI Sea-of-Gates core cell

library cell (as opposed to being part of a channel). More specialized chip templates may have prefabricated specialized block surrounded by Sea-of-Gates basic cells as does the LSI Logic Structured Array series which has four different masterslices with various size 1-,3- or 5-port SRAMs as illustrated in Figure 2.2 [5].

Figure 2.3 reveals that Hughes Aircraft Company has a similar style core cell in two micron technology [9], except that power and ground run down the center of the transistors and the cell contains eight transistors because the well or sub-

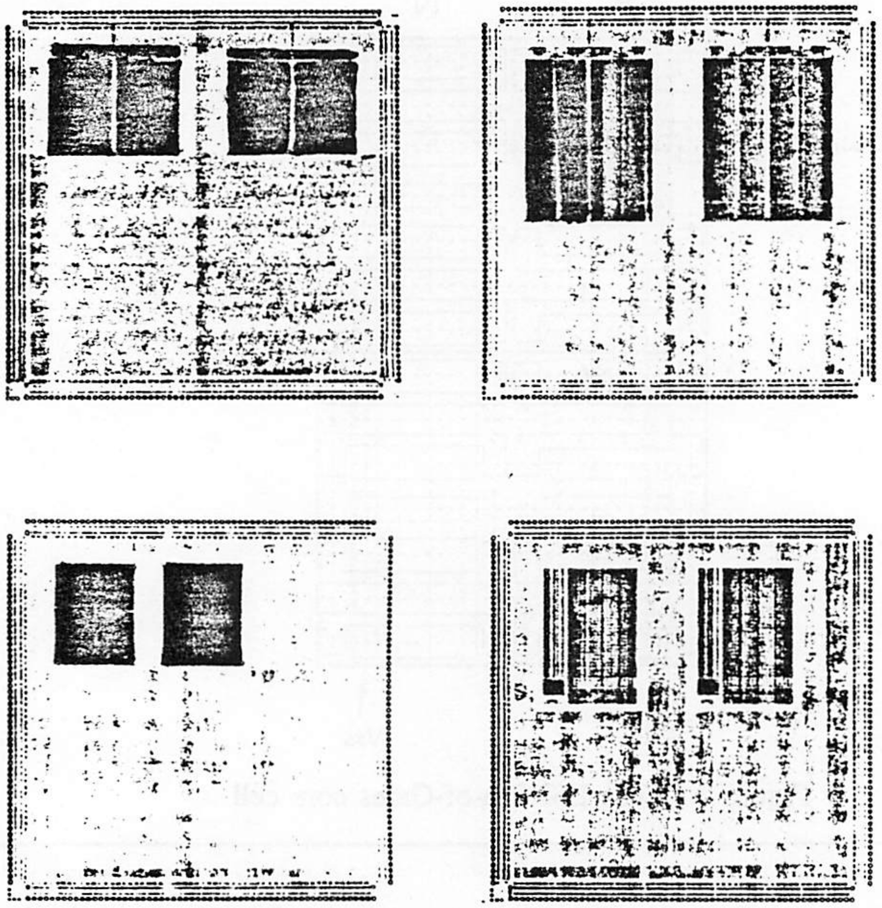


Figure 2.2 LSI Logic offers prefabricated arrays with memory blocks embedded in the Sea-of-Gates array.

strate contacts are spaced every other diffusion block. In both cases the cells are tiled so that the final array has an n-, p-, n-, p-type transistor pattern (NPNP) with two transistors per diffusion block and no gates connected with polysilicon.

Inherently CMOS logic requires the gates of the n- and p-devices to be tied together. Thus, several existing core cells have gate pairs connected in polysilicon

- a non-programmable layer, which saves a blockage of one or more tracks on first-layer metal. The CMOS transmission gate shown in Figure 2.4 is a case where the gates of the n- and p-transistors need to be connected to different signals for correct operation. Since latch devices are used extensively in most circuits it is impractical to connect the gates of all the transistor pairs in the array. Given in Figure 2.5 is the Hitachi core cell as presented in 1987[10] in which three out of six transistor pairs are connected at the gates in poly. In the Hitachi method the smaller transistors are used for transmission gates where needed so that flip flop size is minimized [10] and where they are not needed in the circuitry the area is used for wiring channel space. This cell is repeated so that the chip size template has small n-, large n-, large p-, small p- transistor (nNPp) arrangement. While the nNPp organization allows for sharing power busses between adjacent same type transistors, this does not seem to be the case since the small transistors are not used for cells without transmission gates and since the poly contact spots for the smaller transistors are on the side of the large n- transistors.

Another cell which has both small and large n- and p-transistors is the core cell used by Fijitsu [11]. Pictured in Figure 2.6 this cell has one large n- and one large p-transistor per diffusion strip and two smaller devices per diffusion block with gates that run perpendicular to the gates of the larger devices. The smaller transistors are used when SRAM, ROM, or other specialized cells are present in the circuit. When tiled to form the full array, this cell also creates a nNPp architecture; however it is unclear as to how power and ground are routed, i.e, whether

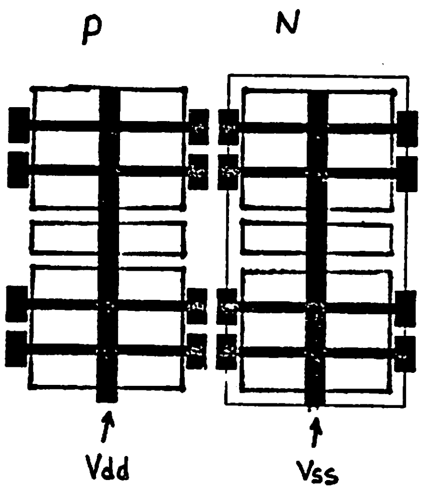


Figure 2.3 The Hughes Aircraft Corporation's Sea-of-Gates core cell

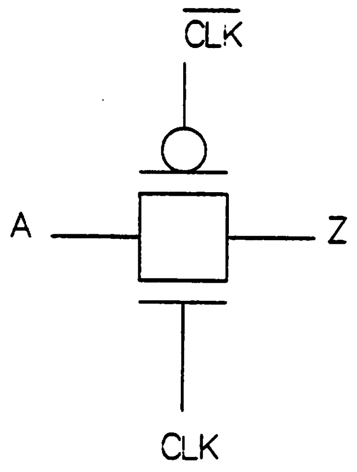


Figure 2.4 The CMOS transmission gate - the n- and p-gates must be tied to different signals for correct operation.

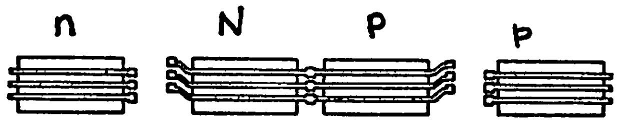


Figure 2.5 The core cell developed at Hitachi. Cells without transmission gates use only the six center transistors.

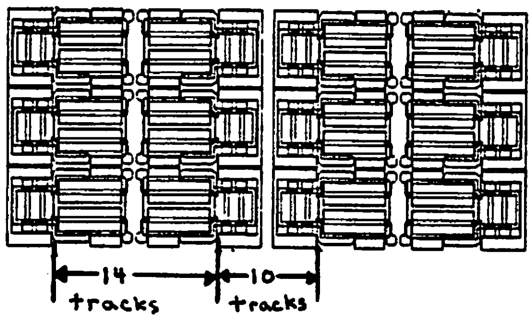


Figure 2.6 Fijitsu core cell.

or not power busses are shared and how wide they are.

Power and ground bus widths and organization may affect the global routing policy in terms of preferred directions for the interconnect layers. The advantage of sharing power busses with adjacent transistors is that the space between metal lines can be filled as part of the solid power bus. Motorola's core cell [12] as illustrated in Figure 2.7 is mirrored in the Y direction so that power lines are shared between two cells. This cell has eight transistors per diffusion block with two small "fake" transistors which are always off for isolation purposes. The p- and n-devices are sized differently with 15 and 11 microns respectively; the gate length is 0.5 micron for both. It is interesting to note that this cell has sixteen first

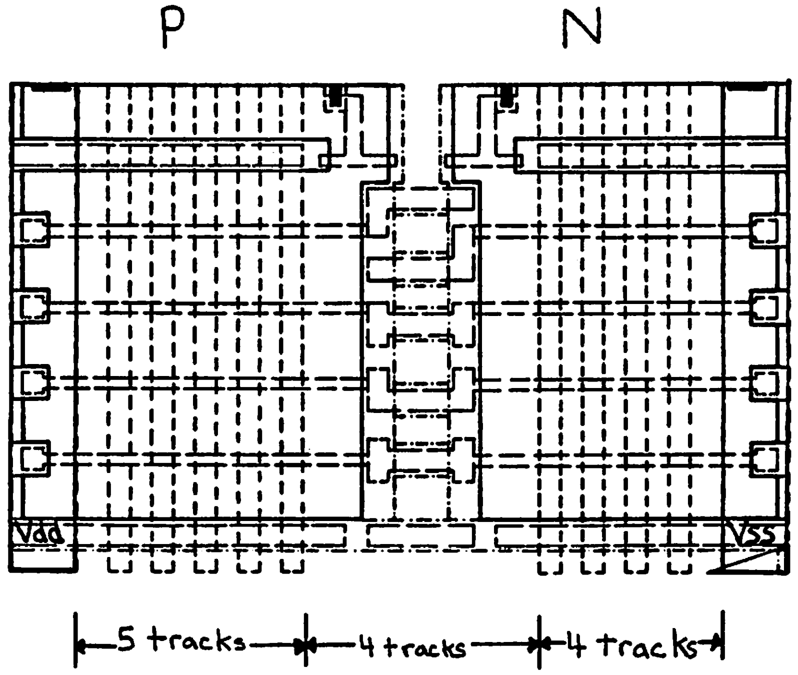


Figure 2.7 The Motorola core cell has 14 tracks between power and ground.

metal tracks running perpendicular to the gates so that simple library cells will have many feedthroughs. For example, a three input NAND gate is implemented as shown in Figure 2.8 and has nine feedthroughs on first-layer metal and all of the six second metal tracks are also free for intercell routing. A more complicated cell, a level sensitive scan flip flop uses 64 transistors and still has seven feedthroughs on first layer metal [12].

Another cell which shares power and ground busses between adjacent cells is the core cell developed at Siemens [13], see Figure 2.9. This cell is designed to accommodate a single port SRAM cell efficiently. It has ten tracks between the

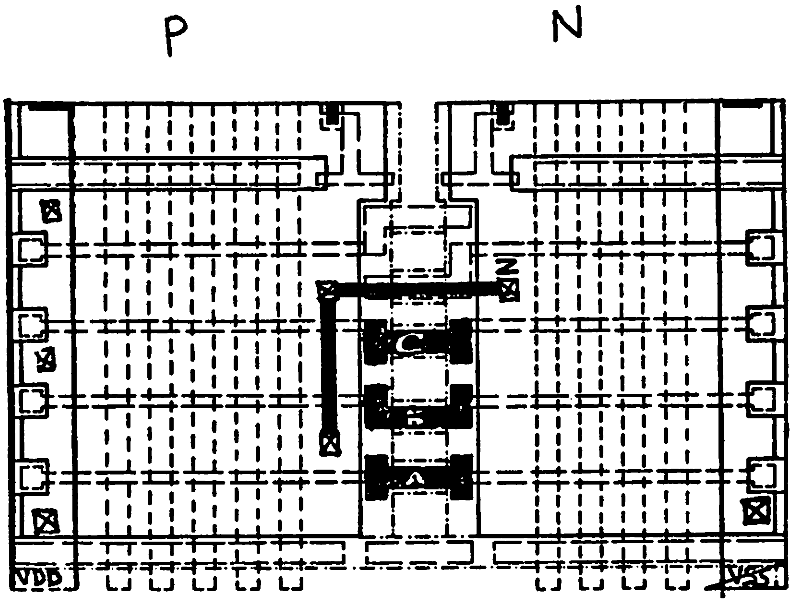


Figure 2.8 A three-input NAND implemented on the Motorola cell has 9 feedthroughs on first layer metal.

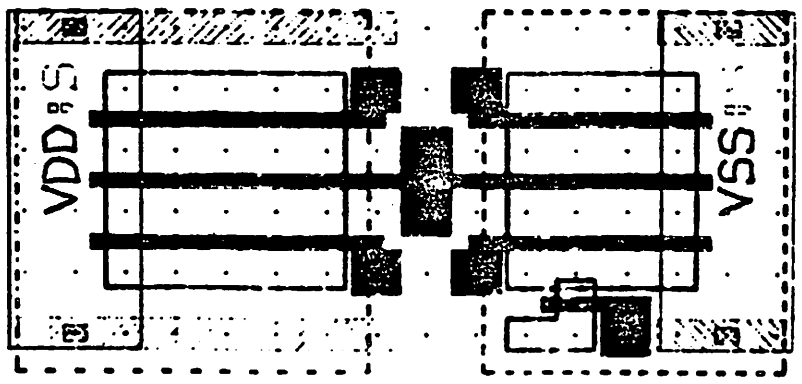


Figure 2.9 Siemen's core cell layout.

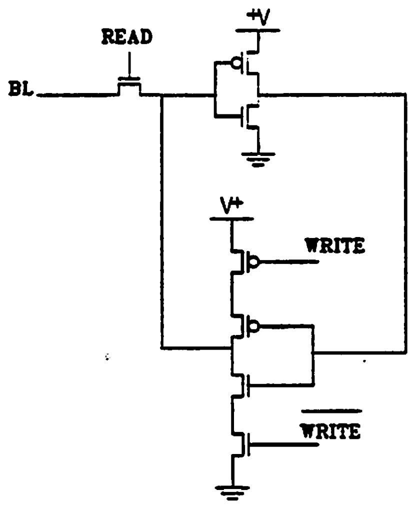


Figure 2.10 Schematic of the single-port SRAM cell that the Siemen's cell is tailored for.

rails over n- and p-type transistors which are 24.4 and 28.2 microns wide respectively, both with gate length of 1.5 microns. There are six routing tracks parallel to the gates and contact to the well or substate is made every diffusion block, i.e, every three transistors. A small n-transistor is in series with the other n-transistors and is used to implement the pass transistor of the seven transistor memory cell which is shown in Figure 2.10.

The core cell of a channelless gate array can be optimized to accommodate any given design style. The ones described so far have been meant for use primarily with static CMOS logic with the same number of n- and p-devices except for the Siemens cell which was optimized for the implementation of a static single port RAM cell. The Institut Fur Mikroelektrik Stuttgart (IMS) considers dynamic CMOS logic to be very crucial to the design of dense, high speed circuits, thus they have developed a microarchitecture which is specialized to fit dynamic logic designs of several types such as cascode voltage logic, domino logic, etc[14]. A transistor diagram of the cell, given in Figure 2.11, reveals six n-type transistors for every two p-transistors. Two of the six n-transistors are smaller as is usually needed to implement dynamic logic. The exact ratio of transistor widths is 4:2:1 for p-,n-, and small n-transistors. A characteristic of this cell that sets it apart from the others is that the p- and large n-transistors' gates make a 45 degree angle with the edges of the underlying diffusions so that there can be a track change without blocking a feed-through track. (Figure 2.12) Another distinguishing characteristic of this cell is that is uses two types of isolation: gate isolation for the p- and large

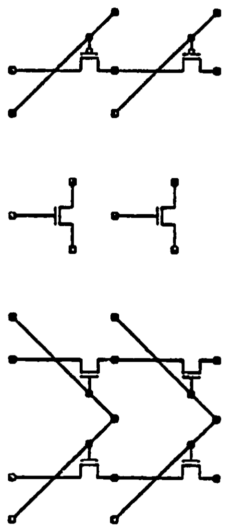
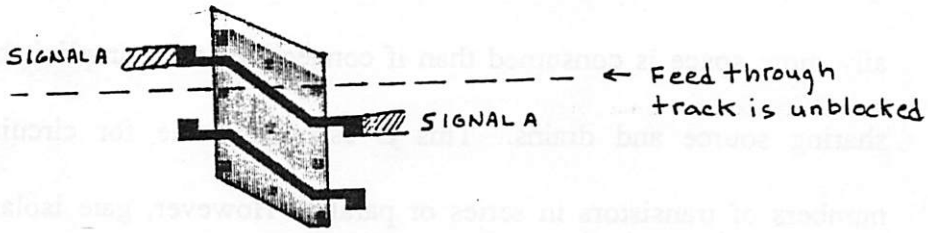
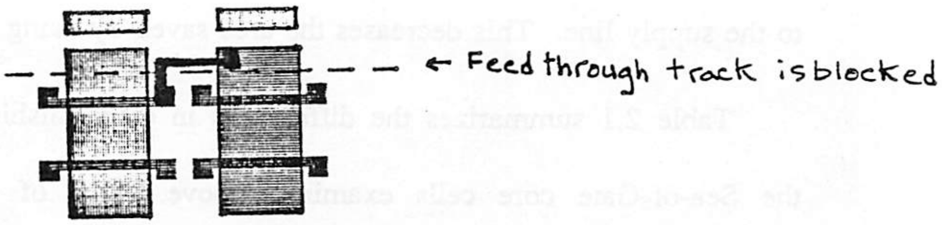


Figure 2.11 Schematic of the IMS core cell.



a) 45 degree gate orientation with respect to diffusion

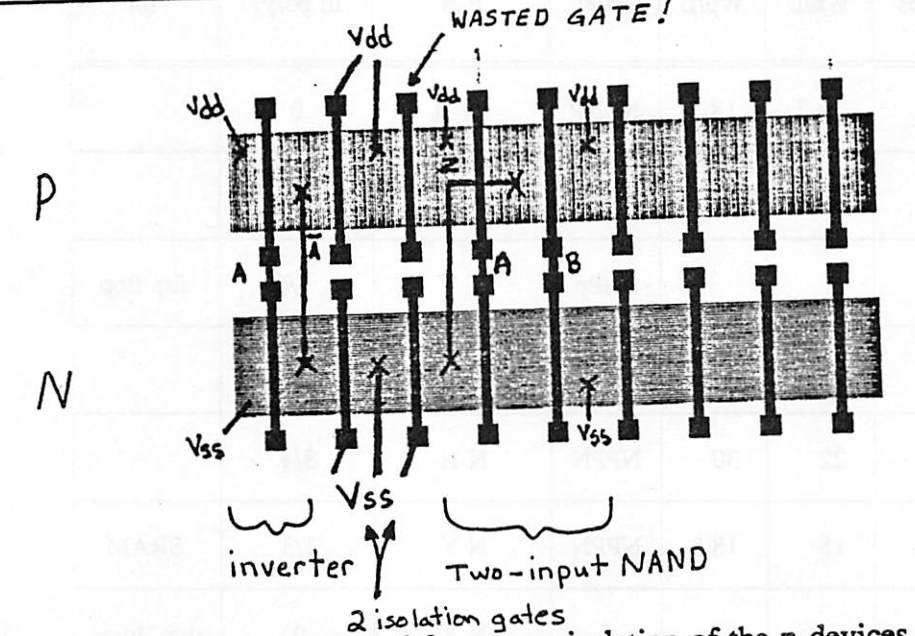


b) Gate perpendicular to diffusion edge
Figure 2.12

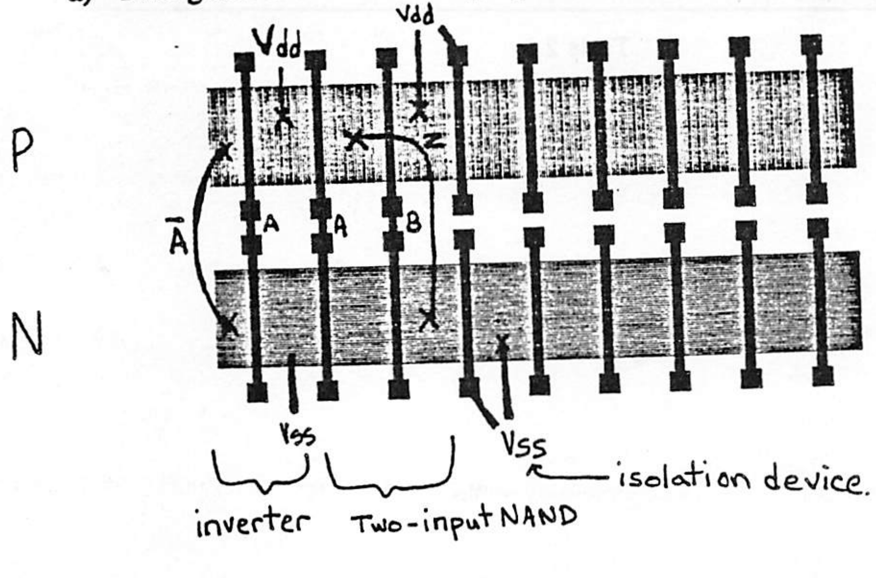
n-transistors and oxide isolation for the small n-transistors.

Gate isolation uses "off" transistors rather than field oxide to isolate neighboring transistors, that is, when an n-diffusion strip has more than two gates, two devices can be separated by connecting the gate between them to ground. The advantage of using this technique is that it allows for high packing density of transistors, especially if the power supply connections needed to ensure absolute isolation can be shared between the isolation device and an adjacent logic cell. For example, Figure 2.13 shows two ways that an inverter and a two-input NAND can be isolated; the first way requires two isolation gates while the second way only uses one. When oxide isolation is used, i.e, only one transistor per diffusion block, usually more space is consumed than if connections were implicit by adjacent devices sharing source and drains. This is especially true for circuits that have large numbers of transistors in series or parallel. However, gate isolation does have the disadvantage that if only the gate is tied to the supply line that turns it off, then spikes on the line could turn on the isolating device. To overcome this possibility, the source or drain as well as the gate of the isolating device should be connected to the supply line. This decreases the area saved by using gate isolation.

Table 2.1 summarizes the differences in distinguishing characteristics among the Sea-of-Gate core cells examined above. Most of the designs have equal numbers of p- and n-devices which is to be expected since they are targeted for primarily CMOS designs. Three of the seven cells have one or more gates connected with poly. In the extreme case, the Hitachi cell has all three of its large transistor pairs connected with poly. Besides not being able to turn these transistors off by tying the gates to the supply lines, it is also impossible to parallel one device of the pair and not the other. An example of when it is advantageous to parallel p-devices and not n-devices is the timing optimization of a three-input NOR gate, Figure 2.14. Since the n-devices are in parallel and the p-devices are in series, this gate will have a faster fall time than rise time (assuming the mobility of the n's is greater than that of the p's, which is usually the case). If the delay of this gate is critical it can be improved by making the widths of the p-devices twice



a) Two gates are needed for proper isolation of the n-devices



b) By sharing connections to the power busses only one gate is needed to isolate the n-devices

Figure 2.13 Some examples of gate isolation

Cell designed by	#N : #P devices	For largest devices Wn/L	Wp/L	order	Different size devices P N	Gates connected in poly?	Design tailored for
LSI	1 : 1	18.7	18.7	NPNP	N N	0	
Hughes	1 : 1			NPNP	N N	0	
Hitatchi	1 : 1			nNPp	Y Y	3/6	flip flop
Fijitsu	1 :1			nNPp	Y Y	0	
Motorola	1 : 1	22	30	NPPN	N N	3/4	
Siemens	7 : 6	15	18.8	NPPN	N Y	1/3	SRAM
IMS	6 : 2				N Y	0	dyn. logic

Table 2.1

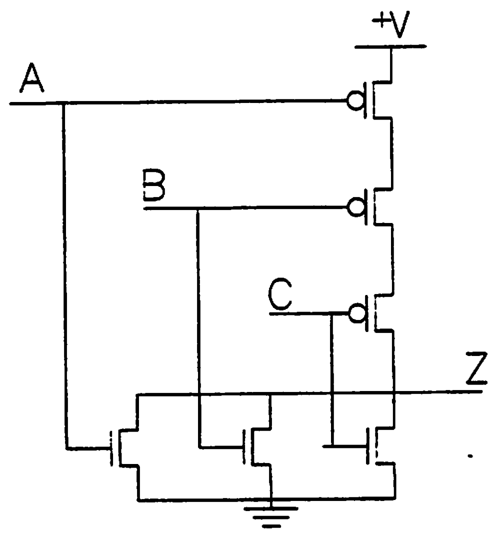


Figure 2.14 Transistor diagram for a three-input NOR gate.

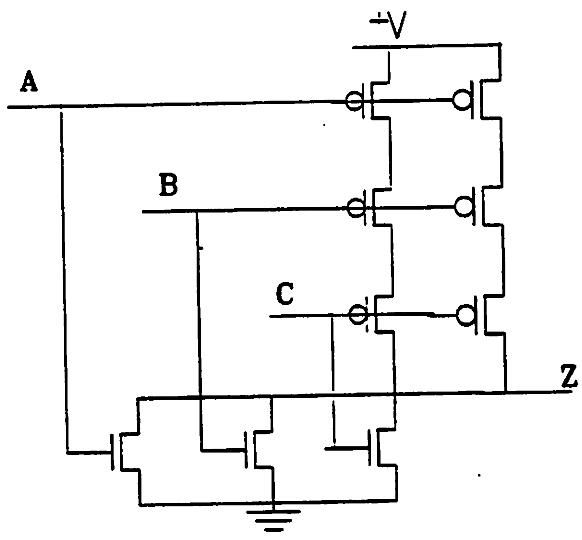


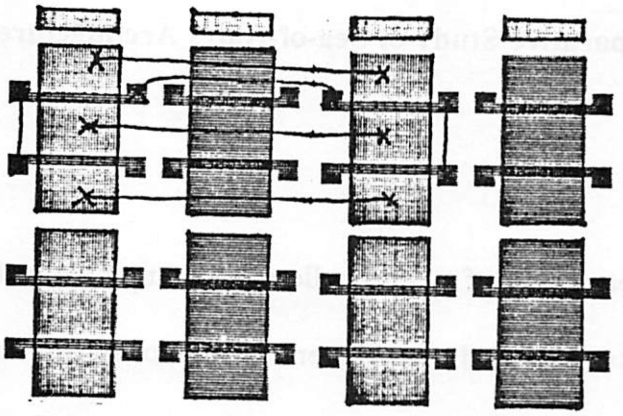
Figure 2.15 A three-input NOR gate with p-devices connected in parallel to improve performance.

as wide by substituting each p-transistor in Figure 2.14. with two p-transistors in parallel as shown in Figure 2.15. Although paralleling devices may not always be the best way to increase performance it is a method that is used frequently.

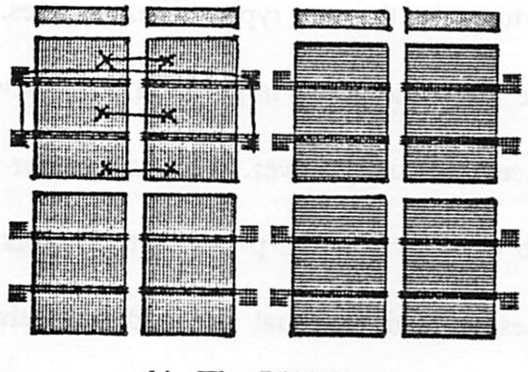
Half of the cells with no gates connected with poly also don't have different size devices of the same type. This makes sense since any device can be used as part of a transmission gate or a parallel structure as described above.

When only the large transistors in the core cells are considered, four out of six (not including the IMS cell), have a NPNP architecture when tiled to form a chip size array. The advantage of the NNPP arrangement for allowing power busses to be shared among adjacent devices of the same type has already been described. The other advantage of the NNPP organization is that parallelling devices is more easily accomplished as shown in Figure 2.16. It may be argued that it is better to use the transistors of the same type which are in the same column for parallel devices, such an approach would work when both the n- and p-devices are to be paralleled but if only one side is as was the case for the three-input NOR gate described above then it is desirable to use the devices in another column.

Symmetry in the core cell and their organization are issues because it can make library cell design simplier since cells can be mirrored along axes of symmetry. After examining several existing core cells and how they are used to form a full array it becomes clear that all of the methods studied have a single core cell which is reflected or repeated. Furthermore, there is only one example of a masterslice array with a prefabricated RAM block. Although this is a viable solution for memory-intensive ASICs requiring fast turn around time, it violates the



a) The PNPN case



b) The PPNN case

Figure 2.16 Four p-transistors in parallel.

potential Sea-of-Gates feature that large special blocks can be realized on the core itself rather than in prefabricated blocks which may be much to large or one bit to small.[13]

Based on this comparative study, representative cell architectures were selected for detailed analysis. These cells are the subject of the next chapter.

CHAPTER 3

A Comparative Study of Sea-of-Gates Architectures

Various implementations of a channelless gate array have been studied in terms of basic transistor size and arrangement. Traditional gate arrays contain p- and n-type transistors of the same size since they may be configured in series or parallel it is not desirable to make the two types different sizes. It can be argued that the p-devices in a gate array should be larger than the n-devices since the carrier mobility of the p-devices is usually lower. This is true for the simple case of an inverter where the ratio of widths of the p- and n-transistors should be inverse the ratio of their mobilities to achieve equal rise and fall delays. However this simple ratio does not hold for arbitrary circuit configurations with various series and parallel combinations of n- and p-transistors. This is because the effective width-to-length ratio (W/L) of parallel transistors is the sum of the widths over the length while the effective W/L of a series combination is the width over the sum of the lengths. The W/L for a gate array transistor can be chosen by determining the average gate load, defining an operating speed, and running SPICE simulations to determine what W/L is optimum and should be used for the transistors of the array. This method of determining the gate array transistor size ignores a very important consideration in ASICs today - the need to minimize capacitive loads on high fanout signals such as clock lines. In the Sea-of-Gates scenario, this consideration is extremely important since the channelless array is very well suited for implementing large macro blocks such as ROM, RAM, or PLA. In RAM, for example, the capacitive load of the bit lines is a critical parameter for determining the maximum speed of the circuit. With this in mind, it is desirable to have small transistors in an array where the architecture allows for transistors to be connected in parallel easily.

Another consideration in choosing a channelless array architecture is its ability to be routed. If it is desirable to have inner-cell routing accomplished over only the transistor area needed to implement the cell then large transistors are favored. However, as the transistors become bigger a higher penalty is paid if routing over an unused transistor renders it unusable. It has been shown that if the width of routing channels can be varied, as in the standard cell case, there will be no unused tracks and the total number of tracks will be minimized. Thus, the smaller the track increment the fewer unused tracks there will be.

For intercell routing it is advantageous to have blockages on as few layers as possible, e.g, given two layers of interconnection, it is desirable to have an architecture which has just the "right" number of routing tracks per basic unit so that most cells can be routed using primarily one layer of interconnect. In keeping with this objective, it is advantageous to have power lines on one layer which are wide enough to supply adequate current for most circuits and only need to use another layer for power rails if the circuit is very large or expected to run at higher speeds.

Three different implementations for the array were chosen and a basic tiling unit (BTU -the smallest repetitive unit that doesn't need to be rotated to create a full array) was created for each. Figure 3.1 shows the three BTU's used. The first style, shown in Figure 3.1a, has alternating n-, p-, n-, (NPNP) transistors with two connections to each gate, gate width of 16 microns, and a first level metal power busses which are the minimum metal width. This style is similar to the LSI Logic and Hughes Aircraft Corp. cells as described in the previous section. The second style has n-, n-, p-, p-type, (NNPP) transistors so that power and ground busses can be shared by adjacent columns. This style was implemented in two ways, first the transistors were made as small as possible as shown in Figure 3.1b with six first metal routing tracks between power and ground. The second implementation also has six routing tracks between rails but has the power lines running over the transistors which results in wider transistors as shown in Figure 3.1c. All styles have transistor gate length of 2 microns.

A three-input NAND gate and the D-flip flop used in the Mississippi State University (MSU) library, were implemented on BTUs of each design style. Figure 3.2 gives the D-flip flop transistor diagram. The cells were compared in terms of size, porosity and drive capability. Figures 3.3-8 show the implementations of the three-input NAND and the D-flip flop on the three templates as labeled. Routability was considered in that the amount of second layer metal required to implement the cell was compared for the same cell type implemented on the various array styles. Table 3.1 shows the width-to-length ratio of the basic tiling unit

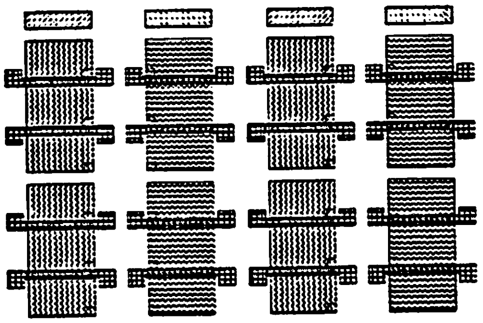


Figure 1a. NPNP array style.

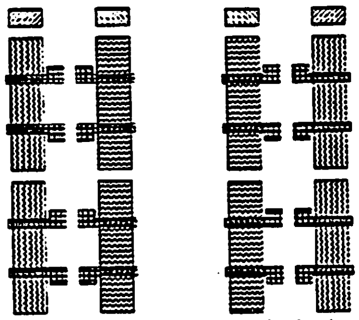


Figure 1b. Minimum width devices
in an NNPP style array.

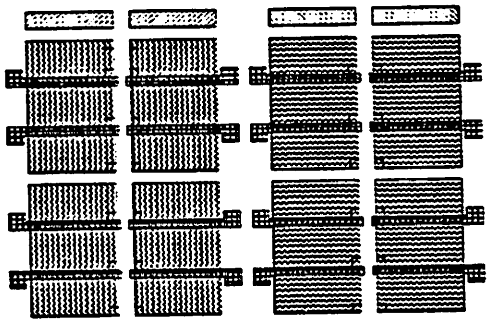


Figure 1c. NNPP style array

Figure 3.1

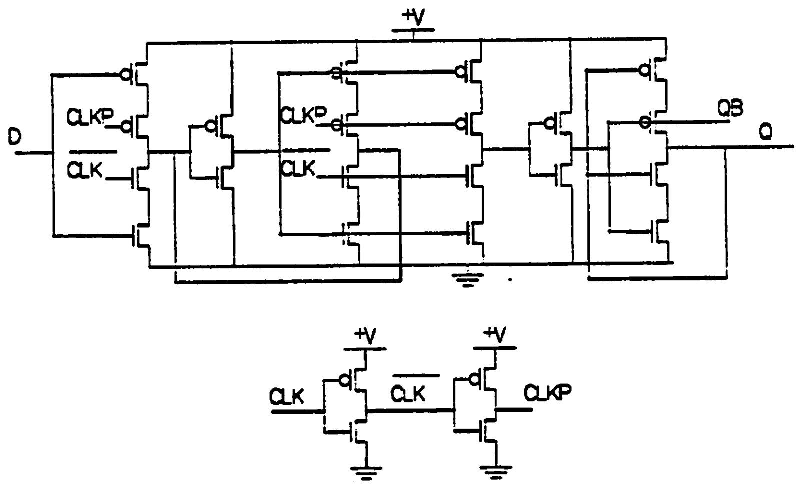


Figure 3.2 The transistor diagram of the MSU D-flip flop.

tiling unit transistors, resulting cell size, and delay of the three-input NAND-for driving loads of 0.5pF, 1pF, and 2pF. SPICE3 was used with the same transistor models for all simulations but the W/L values and corresponding source and drain areas were adjusted to reflect the size of the transistors used. SPICE decks and simulation results are given in Appendix A.

As expected, the three-input NAND implemented with the minimum size transistors was the slowest with the smallest cell area but only by 8.5 percent below the other two. In order to obtain better drive from the minimum size three-input NAND the cell was changed to be a three-input AND by including a double

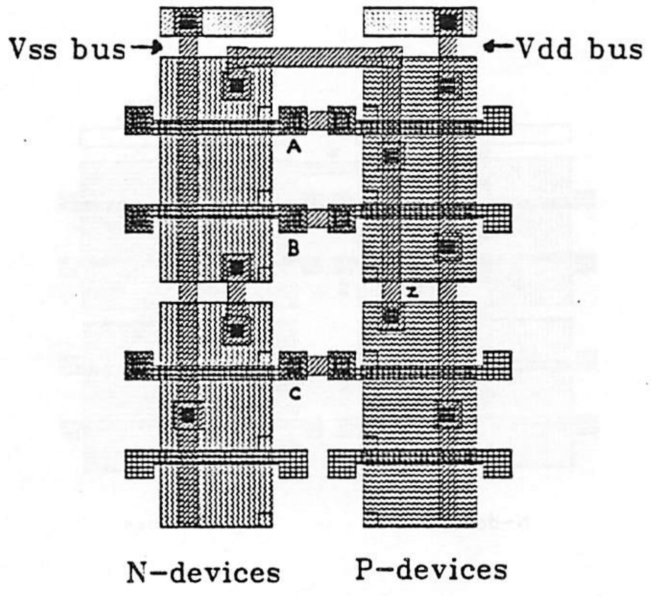


Figure 3.3 A three-input NAND on the NPNP core cell

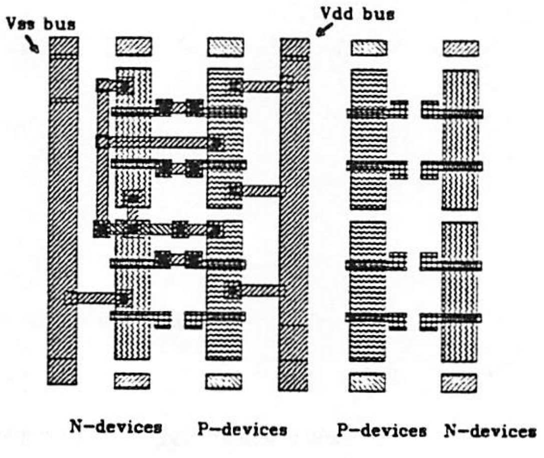


Figure 3.4 A three-input NAND on the NNPP core cell using minimum width transistors

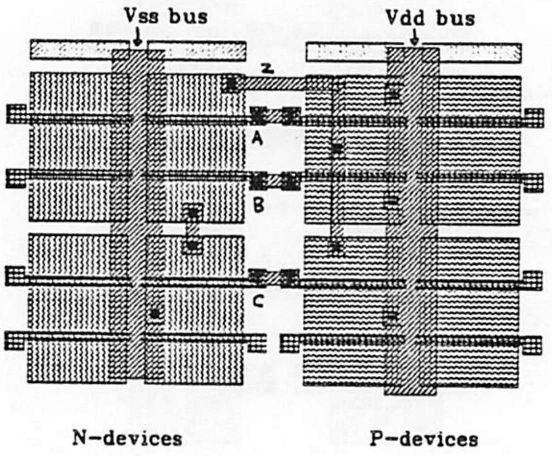


Figure 3.5 A three-input NAND on the NNPP core cell with solid power busses running over the transistors

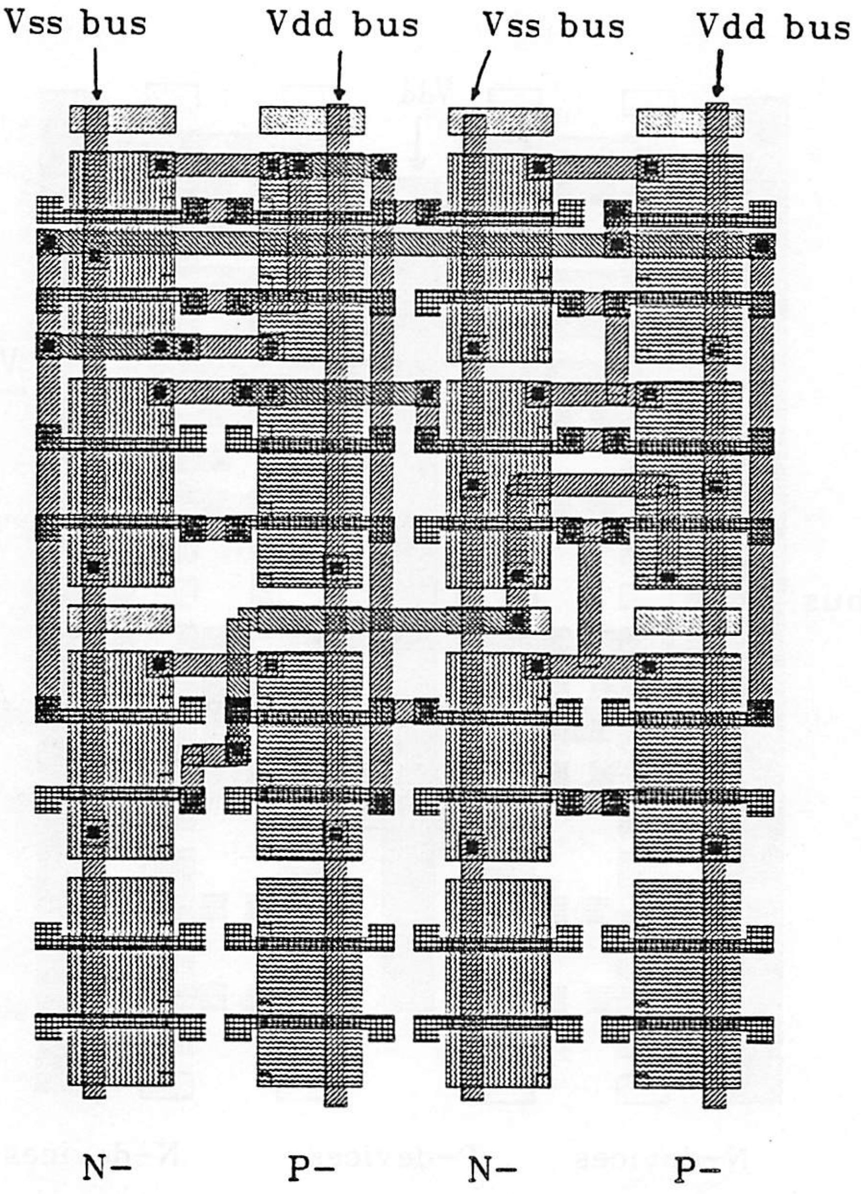


Figure 3.6 The MSU D-flip flop as implemented on the NPNP array

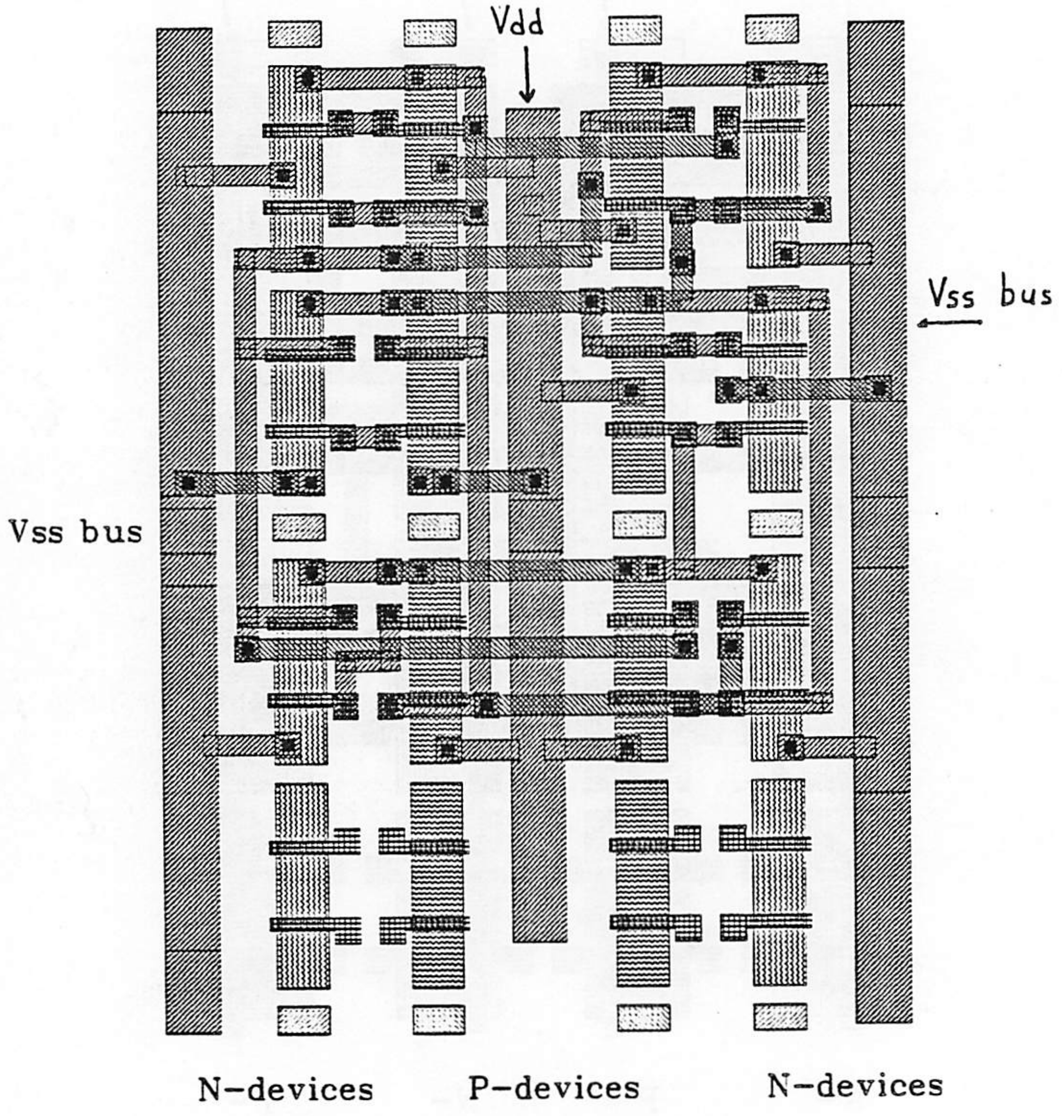


Figure 3.7 The MSU D-flip flop as implemented on the NNPP array with minimum size devices

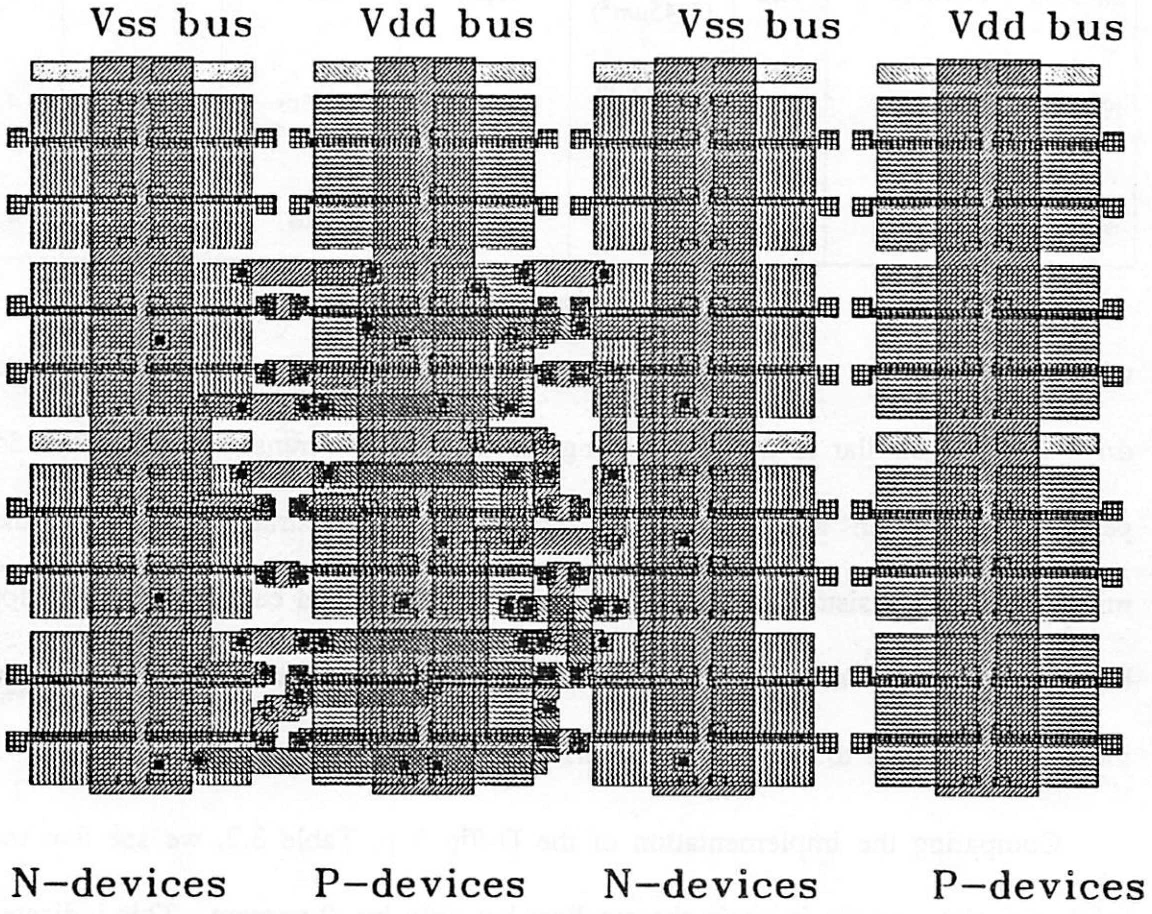


Figure 3.8 The MSU D-flip flop as implemented on the NNPP array

Cell	Architecture Style	W/L	Cell Size	Power Line Width	$t_p = \frac{t_{pHL} + t_{pLH}}{2}$		
					$C_L = 0.5pF$	$C_L = 1pF$	$C_L = 2pF$
NAND3	Skinny P-N-N-P	8/2	53x56μm (2968μm^2)	8μm	2.95ns	5.3ns	9.4ns
NAND3	P-N-P-N	16/2	59x55μm (3245μm^2)	3μm	2ns	3ns	5.45ns
NAND3	P-N-N-P	21/2	59x55μm (3245μm^2)	12μm	1.55ns	2.4ns	4.25ns
AND3	P-N-N-P	8/2	53x94μm (4982μm^2)	8μm	2.1ns	3ns	4.7ns

Table 3.1 Three-input NAND results.

wide inverter as the output stage as shown in Figure 3.9. The resulting cell had drive that was similar to the cells having the much larger transistor widths but 54 percent larger than the other cells. This example illustrates that while the minimum size transistors are favorable for keeping the load capacitance small for high fanout signals such as clock lines, the performance penalty is high and the increase in cell size to achieve better performance is significant.

Comparing the implementation of the D-flip flop, Table 3.2, we see that the minimum size version is again the smallest but only by .9 percent. This indicates that the larger the cell to be implemented the more the size is dominated by routing requirements rather than transistor size. For instance, the skinny D-flip flop uses 9 second layer metal routing tracks, has no feedthroughs on first metal, and only 4 on second metal. The D-flip flip layout on the other two templates have 7 second layer metal tracks used resulting in in 6 and 7 feedthroughs for the NPNP and the

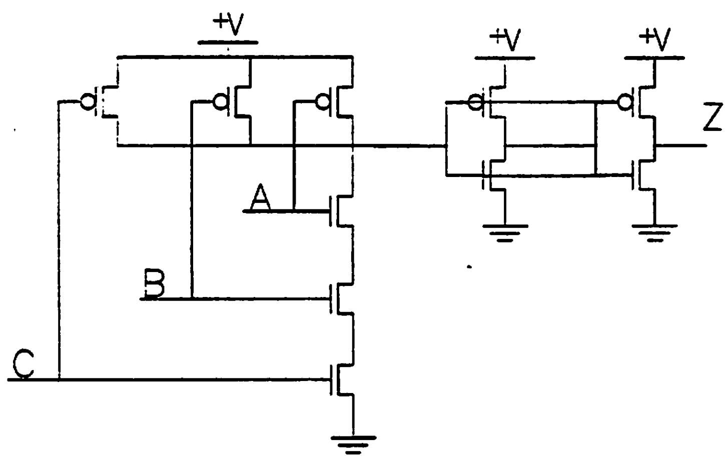


Figure 3.9 A three-input NAND gate with a double wide inverter on the output stage in order to improve drive capability.

NPPN styles respectively. However the NPNP cell probably will need extra connections to a wider power bus at some point since the power busses in this cell are minimum size wires.

An advantage of the NPPN style over the NPNP is that it is easier to connect many p-type transistors in parallel so that a single transistor can be small. Another advantage has to do with sharing the power busses between adjacent columns of the same type diffusion. Specifically, since a first metal power line has to span the minimum separation between similar diffusion and twice the minimum contact cut

Cell	Architecture Style	W/L	cell size	power line width	# Feedthroughs on 2nd metal
MSUDFF	P-N-N-P	8/2	108x113μm (12,204μm^2)	8μm	5
MSUDFF	P-N-P-N	16/2	113x112μm (12,656μm^2)	3μm	8
MSUDFF	P-N-N-P	21/2	114x108μm (12,312μm^2)	12μm	7
MSUDFF	Standard Cell		112x88μm (9,856μm^2)		

Table 3.2 D-flip flop results.

plus four times the minimum overlap contact, in most cases the width will be sufficient so that power and ground routing can be in one layer only.

Another means by which to compare channelless gate array architecture is to see which is more suited for use as RAM, ROM, PLA, etc. as macrocells on the generic template. Since the design and implementation of such macroblocks is beyond the scope of this research it was decided to fix the basic architecture of the array but allow transistor sizes, number of horizontal routing tracks between power rails and the power and ground widths etc. to be varied. This is accomplished by using the array/library generator program which is described in the next section.

CHAPTER 4

Skipper: A Program to Generate Sea-of-Gates Templates

Since the Sea-of-Gates design style is relatively new to the ASIC community, it is not at all clear what are the optimal values of the parameters introduced in Chapter 2, such as transistor sizes and number of feedthroughs for any given array architecture. For this reason, a *template generator* has been written which allows the user to decide many of these values dynamically and, therefore, to experiment with a variety of arrays before making a final choice. Of course, reasonable defaults have been included for all user specified parameters so that first time users can use the generator easily by simply replying to its questions with carriage returns. This program is intended for use during the development phase of the base array. Once a base array is chosen, it would not be changed during use for chip design.

4.1. Overview of The Skipper Program

Skipper is the array generator program to be used for with the Mariner design system. The array architecture is the P-N-N-P style with power lines that are shared by adjacent columns as was described in Chapter 3. Transistor sizes, however are variable and are user defined. Skipper creates templates for both the basic tiling unit and a chip size template with input and output (I/O) pads, power and ground pads, corner pads, and spacers.

First Skipper asks if a new template for the basic tiling unit is needed, if so Skipper queries the user for needed information. As the program asks questions about the template it also asks some things about the chip size template. This is necessary so that default parameters can be calculated as closely as possible to what will be needed for the chip size template (see section 4.3). If the user doesn't want to make a new BTU he/she can give the name of an already existing BTU and Skipper will use that to create a chip size template, warning if parameters such as power and ground widths are insufficient.

4.2. Questions, defaults, and calculations

This section describes how the default parameters are arrived at and what the minimum or maximum allowable answers are determined. The default values for the user-supplied parameters are given in Table 4.1. The current default for the number of transistors in a chip is 50,000 which is around the mean for commercially available channelless array templates. The default number of pads per chip is arrived at similarly. The maximum expected frequency of operation, f, is used in power calculations. The number of metal tracks between rails has a minimum value of six since this is the number needed for the library cells that can be generated to fit on the template. The number of metal tracks between rails, N_{metal1} is used to determine the available width (w_{avail}) for the p- and n-transistors combined as will be described later. The number of transistor pairs between well contacts

Parameter	Definition	Default Value
N_{trans}	Number of transistors on a chip	50,000
N_{pads}	Number of I/O pads on a chip	100
f	Maximum expected frequency of operation	10Mhz
N_{metal1}	Number of tracks between power and ground	6
N_{plugs}	Number of transistor pairs between well contacts	2
C	Average load capacitance as seen by a driving device	1pF
ΔV	Voltage swing for CMOS devices	5 volts
a	Average activity factor	0.5

Table 4.1 Parameters used by Skipper and the default values

(N_{plugs})

is only permitted to be one or two since two is assumed to be the minimum required for electrical reasons to bias the substrate or wells to prevent stray currents that could lead to latchup or other circuit malfunctions.

Power width, w_{pwr}, is calculated using the standard equation for CMOS driving CMOS:

$$w_{pwr} = C \ \Delta V \ f \ i \ a \qquad (4.1)$$

where C is the average load capacitance as seen by a driving device. ΔV is the voltage swing for CMOS devices which is generally the difference between power and ground. Frequency is as specified by the user in megahertz. a is the average activity factor which represents the number of gates that are switching at any one time and its default is set to 0.5. (Such a low activity factor is chosen since while most transistors will not be switching at the same time, some of the devices will not be used as part of the circuitry at all, rather they will be used for routing over and therefore don't draw current from the power busses.) t is an approximation to the number of transistors per two columns and is determined differently according to whether the number of transistors (N_{trans}) or the number of pads (N_{pads}) was specified. In the first case it is assumed that the transistor is as high as it is wide so that:

$$t = 2 * \sqrt{N_{trans}} \tag{4.2}$$

In the second case the height of the chip or column (h_c) can be found by Equation 4.3 where X_{pad} is the width of the pad cell.

$$h_c = (N_{pads} / 4) * X_{pad} \tag{4.3}$$

The number of transistors per column is simply h_c/h_t where h_t is the height of a transistor and is determined as follows. Since there are two transistors per diffusion block and the number of diffusion blocks is given by the user as N_{plugs} the number of transistors per plug ($N_{trans\_per\_plug}$) is simply:

$$N_{trans\_per\_plug} = 2 * N_{plugs} \tag{4.4}$$

The height per plug ($h_{plug\_to\_plug}$) is determined by Equation 4.5 since the number of horizontal tracks per diffusion block (t_{per_block}) and the width of a horizontal track (w_{track}) are known.

$$h_{plug\_to\_plug} = ((N_{plugs} * t_{per\_block}) + 1) * w_{track} \tag{4.5}$$

$h_{plug\_to\_plug}$ divided by $N_{trans\_per\_plug}$ is h_t so that:

$$i = 2 * h_c / h_t \tag{4.6}$$

The factor of two arises since i is an approximation to the number of transistors per two columns since the power busses are shared by adjacent columns. Although the minimum power width required is calculated it is not given as the default unless it is greater than the minimum power width needed to allow contact to two columns as shown in Figure 4.1.

The power width, number of metal one tracks, and several design rules combine to determine the total available width for the p and n transistors as follows. See Figure 4.2.

$$w_{avail} = (N_{metal1} - 2) * (w_{wire} + w_{space}) + w_{pwr} - w_{minsep} - 2 * w_{gateoa} \tag{4.7}$$

w_{gateoa} is the gate overlap of active distance, w_{wire} is the minimum width needed to be able to fit a contact and the overlap on both sides of the cut, w_{space} is the minimum space between wires, and w_{minsep} is the minimum horizontal separation between transistor diffusions of the same type, see Figure 4.3. w_{minsep} is calculated by:

$$w_{minsep} = max((poly2p + 2 * polyoa), act2act) \tag{4.8}$$

where *poly2p* is the minimum poly spacing, and *act2act* is the minimum active to active spacing.

The default transistor width is simply half of the total available. If the user specifies transistor widths such that the sum is greater than w_{avail}, Skipper figures out how many metal tracks there will be between power and ground and if there

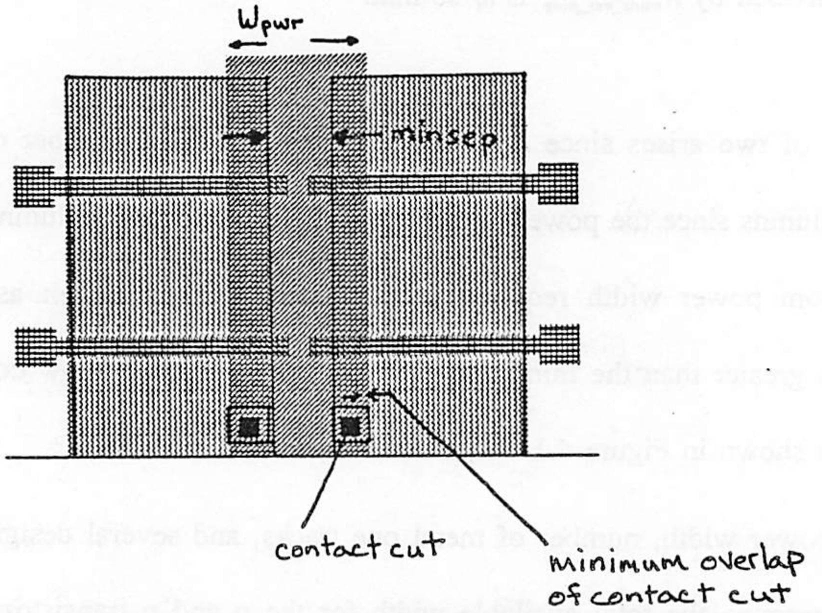


Figure 4.1 The minimum metal one power bus width needed to contact to both transistor columns that the bus serves

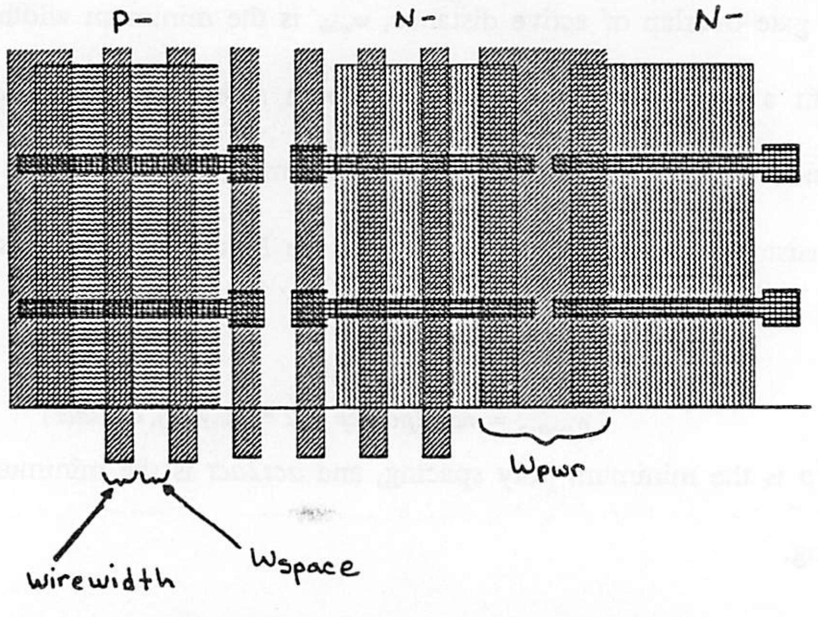


Figure 4.2 w_{avail} is the total available width for the p- and n- transistors

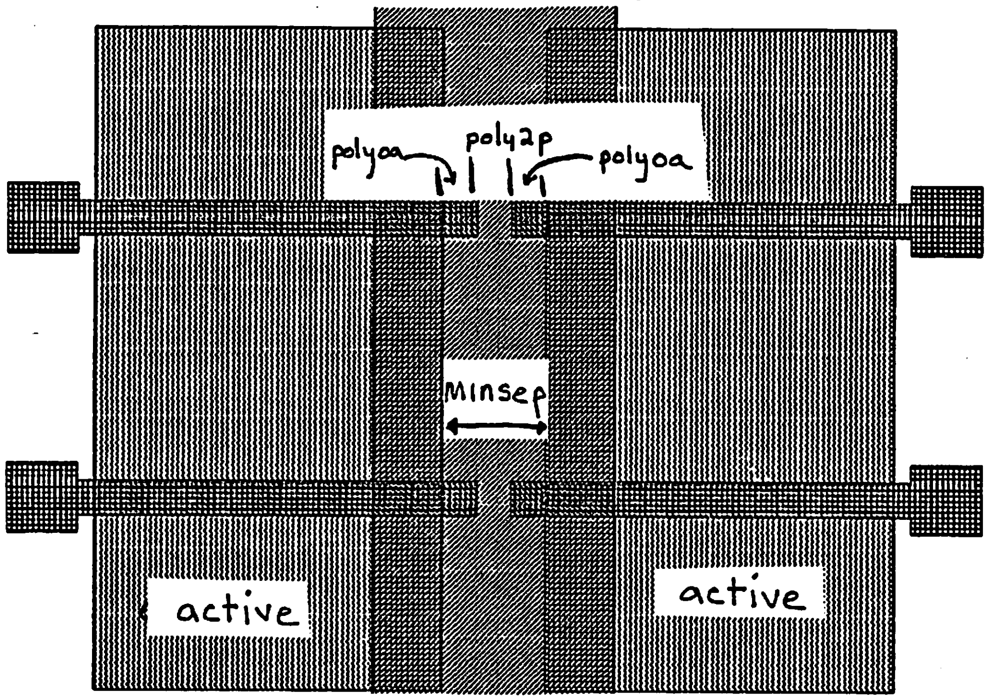


Figure 4.3 The minimum horizontal sepatation between transistors of the same type

are any extra microns needed in order to produce the device widths specified but don't contribute to part of a routing track. Specifically, extra microns are the remainder of the width rail to rail (w_{r2r}) and a routing track ($w_{wire} + w_{space}$). Next the user is informed of the new number of metal routing tracks and asked if he wants to continue or go back and change his choices for the transistor widths. If the specified widths sum to less than the available width the the basic tiling unit will have extra routing tracks as shown in Figure 4.4. The extra tracks are located on the outside of the active area so that the extra track will be totally free from cell routing if the cell is only half a tiling unit wide. See Appendix B for examples of

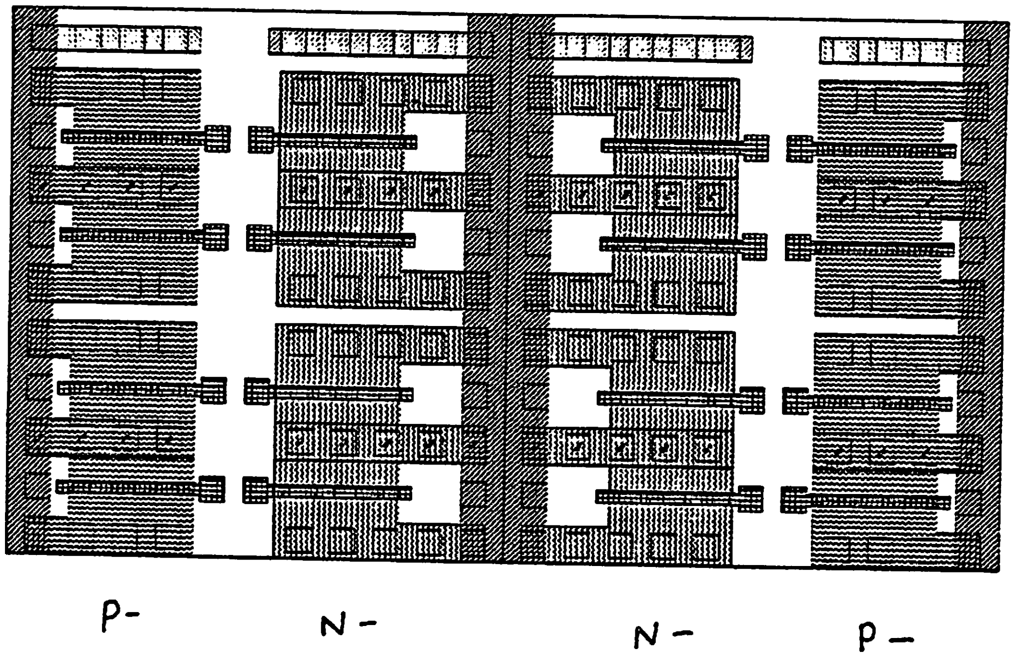


Figure 4.4 The number of tracks between rails is 9; the widths of the p- and n-transistors were specified to be smaller than w_{avail}

core cell generation using Skipper.

4.2. Determining chip dimensions

When the user specifies the number of I/O pads as the determining parameter for chip size, the chip dimensions are obtained by dividing the total number of pads by four, multiplying by the pad width and adding twice the pad height. This results in a square chip. The chip dimensions are depicted in Figure 4.5. The number of transistors is determined by how many BTUs will fit inside the area

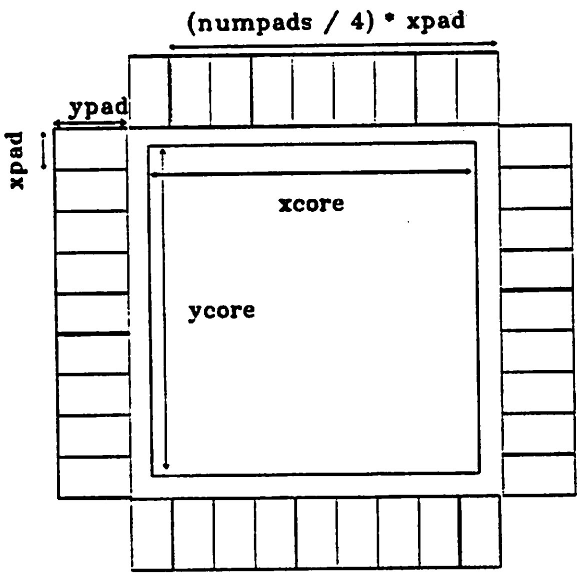


Figure 4.5 Chip dimensions when the number of I/O pads is specified by the user

defined by X_{core} and Y_{core} which are determined by Equations 4.9 and 4.10 respectively.

$$X_{core} = (N_{pads}/4) * X_{pad} - 2*(w_{wire} + w_{space}) \tag{4.9}$$

$$Y_{core} = (N_{pads}/4) * X_{pad} - 2*(w_{wire} + w_{space}) \tag{4.10}$$

If h_{p2p} is the height of the BTU, w_{r2r} is the width from center of the rails, then the number of rows of BTUs, r, and c, the number of columns, are determined by Equations 4.11 and 4.12.

$$r = Y_{core} \; / \; h_{p2p} \tag{4.11}$$

$$c = X_{core} / (2 * w_{r2r}) \tag{4.12}$$

The resulting number of transistors in the chip will be:

$$N_{trans} = r * c * 8 * N_{plugs} \tag{4.13}$$

where the factor of 8 arises because the smallest BTU has four n- and four p-transistors.

If the number of transistors is specified as the determining factor for chip size then the number of rows and columns of the BTU are determined from Equations 4.13 and 4.14 which imply that the chip will be square.

$$r * h_{p2p} = c * w_{r2r} \tag{4.14}$$

The resulting expressions for *r* and *c* are:

$$r = \sqrt{((N_{trans} * w_{r2r}) / (8 * N_{plugs} * h_{p2p}))} \tag{4.15}$$

$$c = \sqrt{((N_{trans} * h_{p2p}) / (8 * N_{plugs} * w_{r2r}))} \tag{4.16}$$

Next X_{core} and Y_{core} are calculated:

$$X_{core} = c * 2 * w_{r2r} + 2 * (w_{wire} + w_{space}) \tag{4.17}$$

$$Y_{core} = r * 2 * h_{p2p} + 2 * (w_{wire} + w_{space}) \tag{4.18}$$

so that the number of pads in the x and y directions, N_{xpads}, N_{ypads}, can be determined as follows:

$$N_{xpads} = X_{core} / X_{pad} + .5 \tag{4.19}$$

$$N_{ypads} = Y_{core} / X_{pad} + .5 \tag{4.20}$$

4.4. Power busses in second metal

As presented so far, Skipper assumes that the power and ground busses are implemented on first metal. However, this may not be the best solution. Therefore

Skipper asks the user whether power busses should be in first or second metal. The program flow is exactly the same for the second case except that the minimum power width and the minimum horizontal separation between active areas of the same type is calculated differently. With second layer metal power lines w_{minsep} is the maximum of the previous expression and the minimum via size plus twice the greater between the metal two overlap of via and the via to active space, as shown in Figure 4.6.

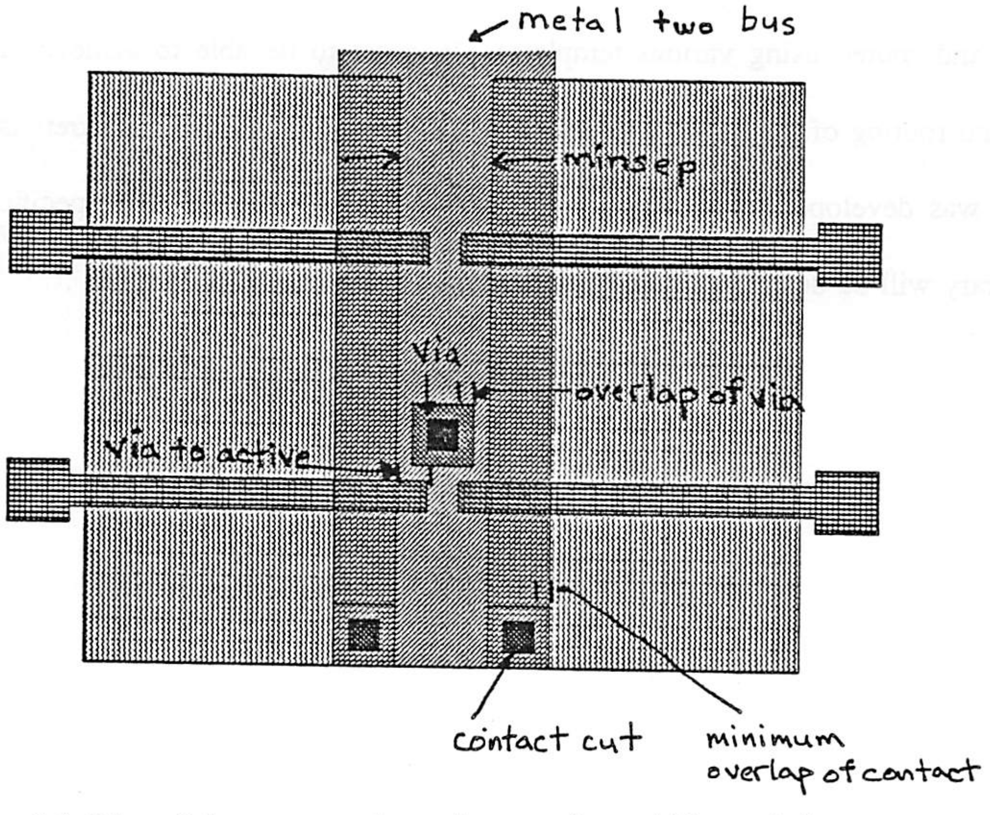


Figure 4.6 The minimum second metal power bus width needed to contact to both transistor columns that the bus serves

4.5. Changing the design rule set

Skipper reads the necessary design-rule information from TAP[19], a Technology Access Program so that the design rule set can change and Skipper will not need to be altered. TAP accesses a technology facet, which is an OCT[20] facet that uses a bag and property structure to hold all the information that describes a technology. Thus any design rule set can be used with Skipper to generate the core cell and the chip size template.

The advantage of being able to vary the number of vertical routing tracks between power and ground will become obvious when large, complex examples are placed and routed using various templates. In order to be able to achieve placement and routing of the same examples with different template sizes a "stretchable" library was developed to accompany the various size templates. The specifics of the library will be described in the next section.

CHAPTER 5

The Skipper Library - Scalable and Stretchable

5.1. Introduction

While the basic architecture of a Sea-of-Gates template has been decided in terms of the number of p- and n-transistors, the type of isolation used, the number of connection points to each gate, whether or not gates are connected in poly, and the angle the gates make with the diffusion, other template parameters remain variable as was discussed in the explanation of Skipper, the template generator program. By fixing a minimum value for the number of tracks between power rails and a maximum value for the number of diffusion blocks per substrate or well contacts, library cells can be captured and later stretched to fit on larger templates.

5.2. Initial Library Cell Design

The minimum number of vertical first layer metal tracks between power and ground was set to six, i.e, two over each diffusion block and one over each poly tab. Six was chosen so that most basic cells such as CMOS static logic cells, can be implemented primarily with one layer of interconnect within the cell transistor area, and more complicated cells use both first and second layer metal for intracell routing. The transistor sizes and the power and ground widths remain variable since they don't affect the available routing area.

The 30 cells that are listed in Table 5.1 were designed manually on the minimum size (default) template, (six vertical first layer metal tracks, five horizontal second layer metal tracks per diffusion block plus one horizontal track over the

Cell Name	Logic Function	Area (lambda)
and3	z = ab	59 x 88 = 5192
ao21	z = ab + c	59 x 88 = 5192
aoi21	z = !(ab + c)	59 x 88 = 5192
aoi22	z = !(ab + cd)	59 x 88 = 5192
impnd3	z = !(!abc)	59 x 88 = 5192
impnor3	z = !(!a + b + c)	59 x 88 = 5192
inv	z = !a	59 x 39 = 2301
inv2	z = !a	59 x 39 = 2301
inv4	z = !a	59 x 88 = 5192
inv8	z = !a	118 x 88 = 10,384
inv+xms	z = !a	59 x 39 = 2301
muxinv2	z = !(as + b!s)	59 x 39 = 230
nd2	z = !(ab)	59 x 39 = 2301
nd3	z = !(abc)	59 x 88 = 5192
nd4	z = !(abcd)	59 x 88 = 5192
nor2	z = !(a + b)	59 x 39 = 2301
nor3	z = !(a + b + c)	59 x 88 = 5192
nor4	z = !(a + b + c + d)	59 x 88 = 5192
oai21	z = !(a (b + c))	59 x 88 = 5192
oa21	z = (a (b + c))	59 x 88 = 5192
oai22	z = !(a + b)(c + d)	59 x 88 = 5192
or3	z = a + b + c	59 x 88 = 5192
dynscanlatch	see Appendix E	59 x 88 = 5192
dyndff	see Appendix E	59 x 176 = 10,384
dyntestdff	see Appendix E	59 x 264 = 15,576
dyntstendff	see Appendix E	59 x 352 = 20,768
staticdff	see Appendix E	118 x 127 = 14,986
xmsgate	z = a	59 x 39 = 2301
xnor2	z = !(a!b + !ab)	59 x 88 = 5192
xor2	z = (a!b + !ab)	59 x 88 = 5192

Table 5.1 Parameters used by Skipper and the default values

substrate contact area.) Figure 5.1 shows the default template and its wiring grid. Library cells contain one or more instances of the template and paths on the metal layers, and contact and via instances. The template has labelled terminals at each possible contact locations on the wiring grid as shown in Figure 5.2. By making connections only at the labelled terminals the nets can easily be regenerated or transferred to another cell which implements the same function but has a larger basic tiling unit. Clearly this is not true for smaller templates. In this way the library is said to be stretchable.

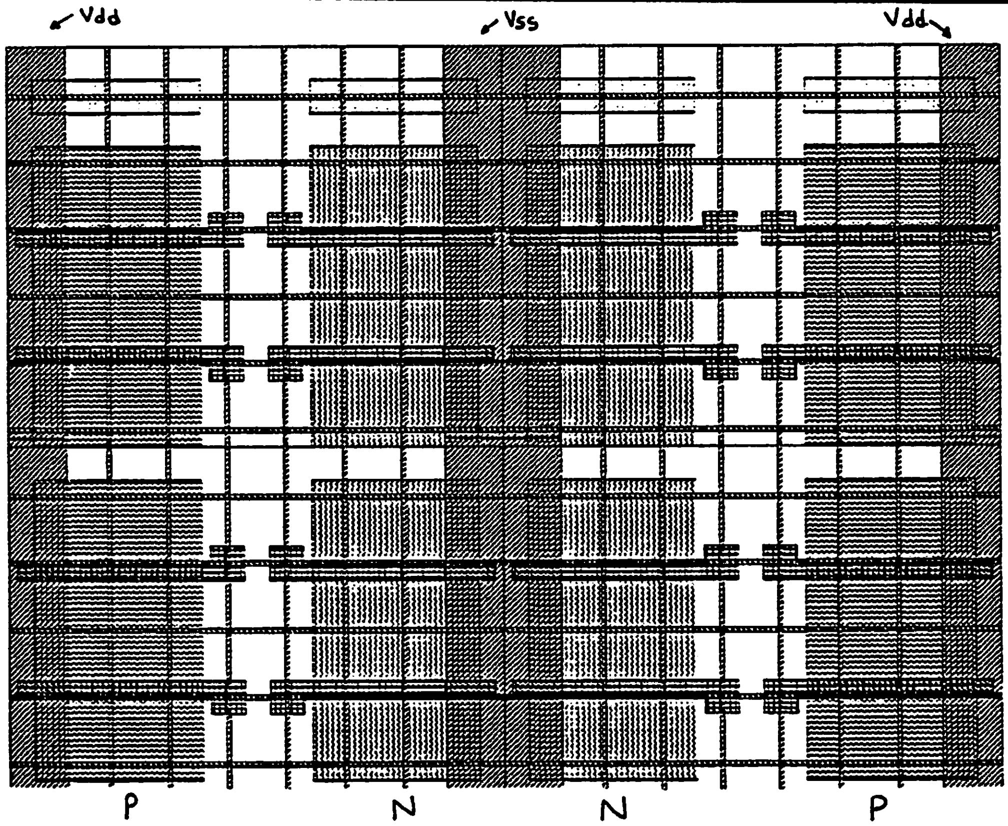


Figure 5.1 The default template and its wiring grid

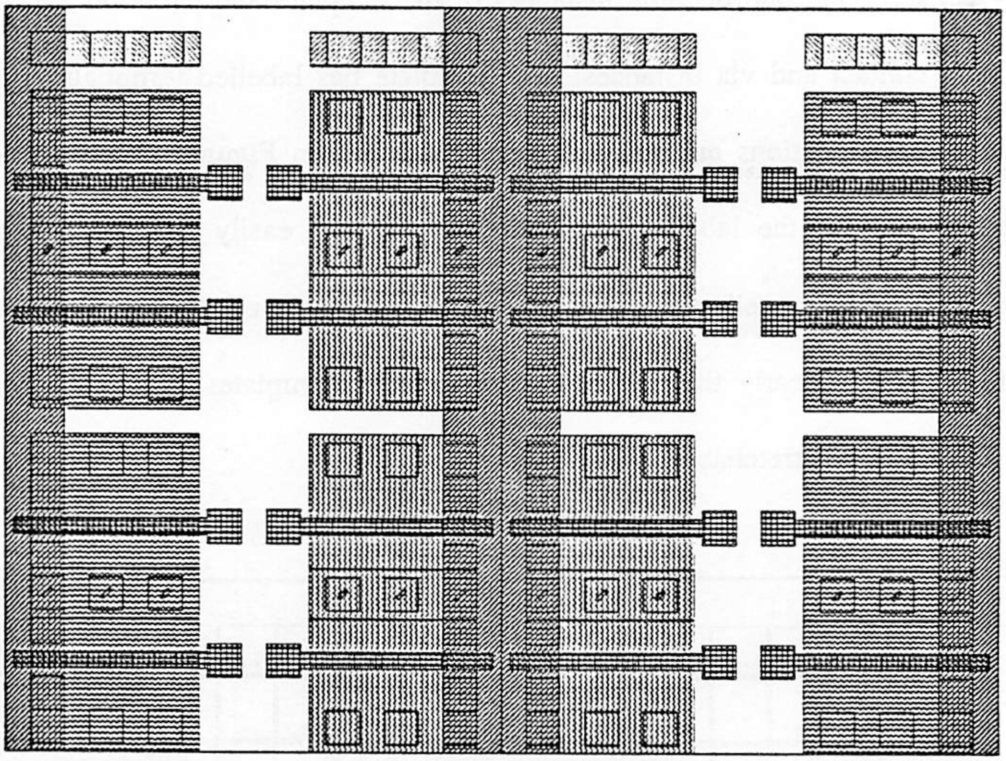


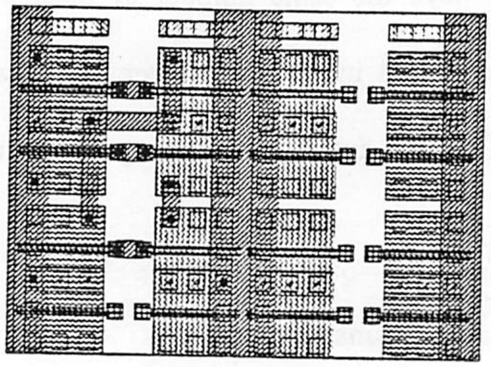
Figure 5.2 The default template with labelled terminals at each possible contact location on the wiring grid

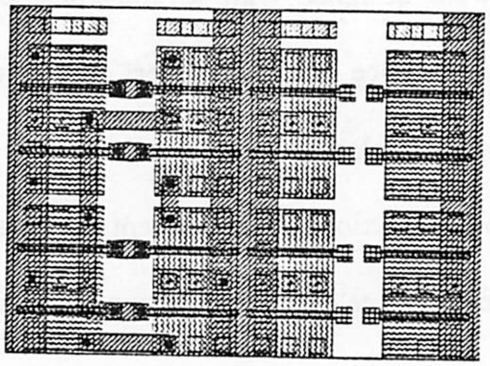
5.3. Stretching The Library Cells

The program that regenerates the library cells is called Regen. Regen does the mapping by building up a symbolic grid for both the old and the new templates by looking at the positions of the terminals on the template, with the assumption that any terminal will fall on grid lines in the X- and Y-direction. Then Regen maps the wiring in the old cell onto the grid of the old template and from there maps it to the grid of the new template. Regen then creates the new cell, makes the appropriate number of basic tiling units needed to implement the cell, copies the important properties, and creates the wiring. The old template and the new template must have the same number of transistors per block and must have power and ground routed in the same layer. Otherwise the topology of the cell would be different and the automatic mapping couldn't be done in such a simple manner.

5.4. Library Cell Considerations

The library cells were designed with several considerations particular to the Sea-of-Gates design style. Since only two layers of interconnect are available in the process system chosen, implementing the cells in first layer metal only was a high priority. Which logic functions to implement as library cells was influenced by whether or not the resulting cell contained any unused transistors. For example, a three-input NAND gate uses six devices as shown in Figure 5.3a. In this case it is does not cost any thing to design a three-input AND gate and a three-input NAND gate with one of the inputs inverted as shown in Figures 5.3b and 5.3c. It can be argued that the three-input NAND is not as large as the others because it can share half of the second diffusion block with another cell in some cases. However being able to determine this would introduce complexity in both the library cells and a placer program that would have to determine that cells could overlap legally. It is much easier to implement both the cells with wasted devices and the combined functions that use the same area. The resulting library can be used with MIS [21], a multi-level logic synthesis and minimization system, which, among other things can map a boolean network into a specified cell library with the total


a) NAND3


b) AND3

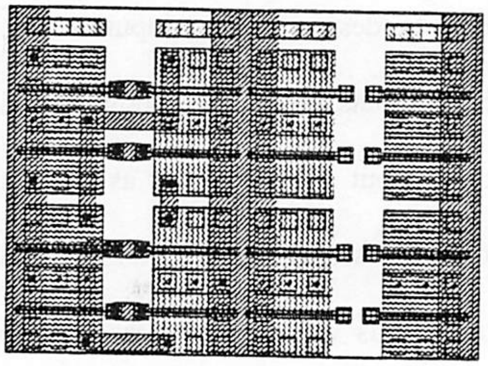

c) Z = !(!a b c)

Figure 5.3

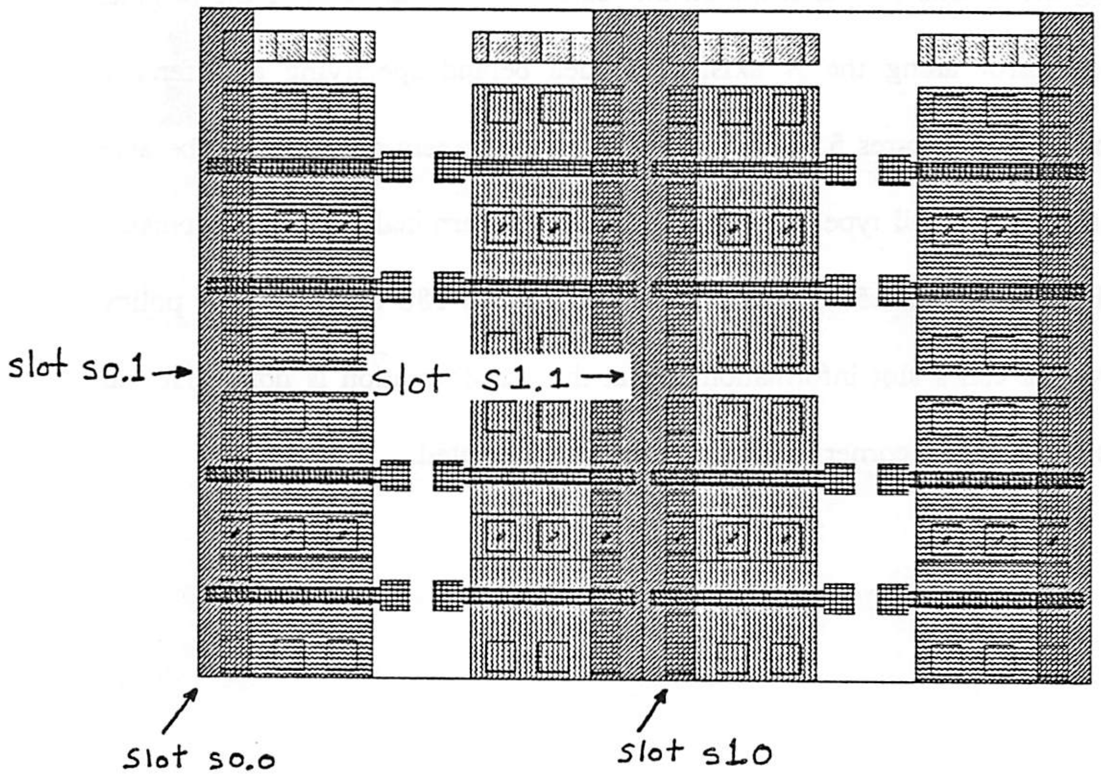
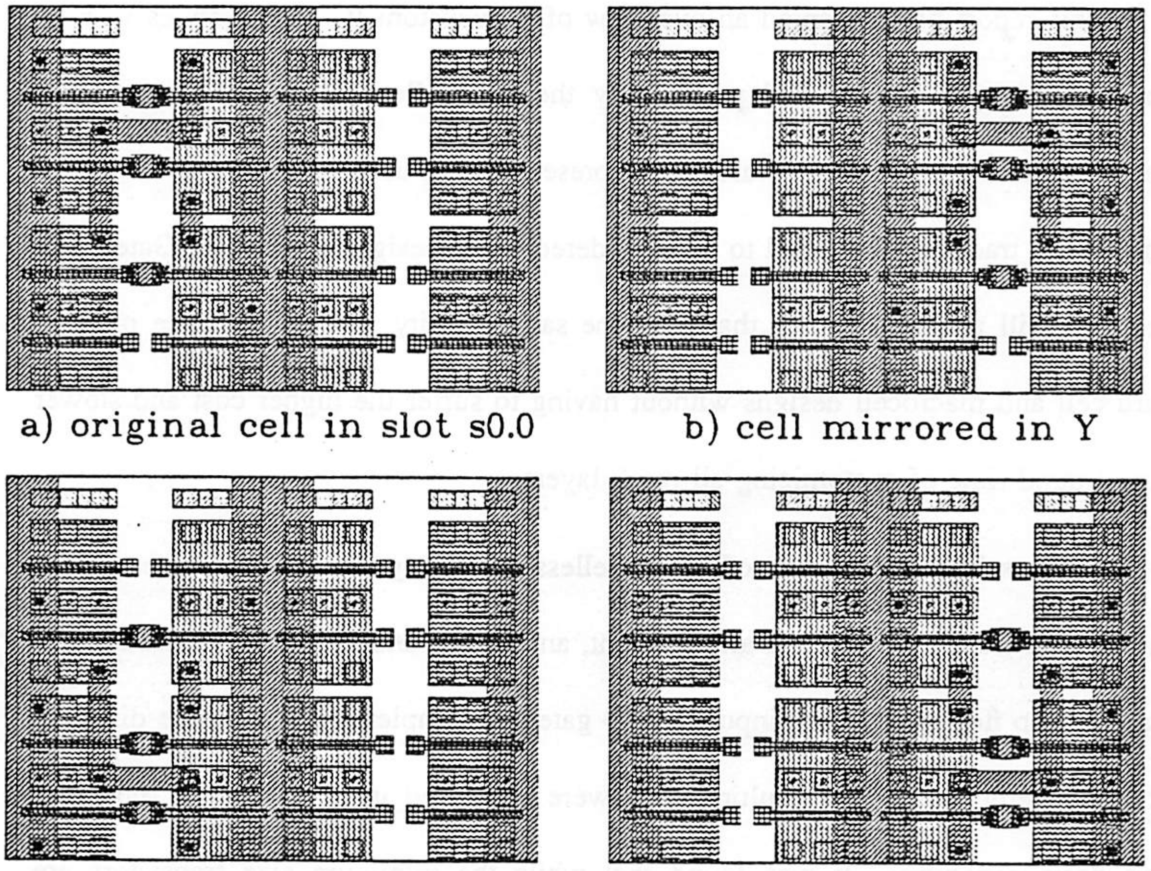


Figure 5.4

cost with respect to area or delay being minimized.

Another consideration in library cell design is where the cells can be placed on the template. Figure 5.4 shows the default template produced by Skipper with the legal locations for the lower left corner of a library cell. These locations are called *slots*. Each library cell contains information that indicates which slots the cell can fit into and the necessary transformation that needs to be performed on the metal paths and contact and via instances with respect to center for the cell to fit in that slot. An example of how the slot information is used for a three-input NAND is given in Figure 5.5. Figure 5.5a has the NAND gate in slot **s0.0** , which is the default so that no transform need be specified. Figure 5.5b is the same cell mirrorred along the Y axis so that it fits on the **s1.0** slot. Figure 5.5c is the original cell with a mirror along the X axis. The idea behind specifying a difference between the cells in Figures 5.5a and 5.5c is that a placement program will be able to choose the correct cell type according to where the terminals connect to outside the cell. Finally Figure 5.5d is the original cell rotated 180 degrees. The policy for expressing a cell's slot information is that the transformation is done first and then the cells lower left corner is placed at the slot indicated.


a) original cell in slot s0.0

b) cell mirrored in Y

c) cell mirrored in X

d) cell mirrored in X and Y

Figure 5.5

CHAPTER 6

Conclusions

This report has presented an overview of semicustom IC design styles with an emphasis on gate arrays and particularly the channelless or Sea-of-Gates array. Several existing core architectures were presented and used to exemplify the various design tradeoffs that need to be considered when designing a Sea-of-Gates core cell that will result in ASICs that have the same density and performance of standard cell and macrocell designs without having to suffer the higher cost and slower turn-around time of customizing all mask layers.

Various implementations of a channelless gate array architecture were studied in terms of transistor size and arrangement, and power and ground organization. A static D-flip flop and a three-input NAND gate were implemented on three different core cell templates. The resulting cells were compared in terms of size, porosity, and drive capability. It was found that while the minimum size transistors are favorable for keeping the load capacitance small for high fanout signals such as clock lines, the performance penalty is high and the increase in cell size to achieve better performance is significant. Furthermore, the D-flip flop cells indicated that the larger the cell to be implemented the more the size is dominated by routing requirement ts rather than transistor size.

It was decided to fix the architecture of the Sea-of-Gates core cell to be used with the Mariner system being developed at U. C. Berkeley to have an NPPN transistor arrangement with solid power busses which are shared between adjacent columns of the same type. A template generator was presented which allows transistor sizes, number of horizontal routing tracks between power rails and the power and ground widths etc. to be varied. In order to be able to achieve placement and routing of the same examples with different template sizes a "stretchable" library was developed to accompany the various size templates.

30 cells were designed manually on the minimum size template which has 6 wire tracks between power and ground. These cells can be regenerated to fit on larger templates by using the Regen program which copies the paths and connectors that are on labelled terminals on the template. The old template and the new template must have the same number of transistors per block and must have power and ground routed in the same layer in order for the automatic mapping to be done in such a simple manner.

Since the Sea-of-Gates design style is relatively new to the ASIC community, it is not at all clear what are the optimal values of the parameters introduced such as transistor sizes and number of feedthroughs for any given array architecture. The template generator allows the user to decide many of these values dynamically and, therefore, to experiment with a variety of arrays before making a final choice for the core cell to be used for actual chip design. Still other issues related to Sea-of-Gates designs are open. For example, evaluating the impact of changes in process technology such as denser design rules or adding a second poly layer. Also, the effect of the isolation technique used, e.g, gate isolation or field isolation, on on circuit performance needs to be investigated and might entail lab work to determine how much noise on the power lines affect circuit operation when gate isolation is used.

The advantage of being able to vary the number of vertical routing tracks between power and ground will become obvious when large, complex examples are placed and routed using various templates. The density and performance of the resulting circuits will help to decide the optimum values of the parameters that have been described in this report.

Appendix A

SPICE Decks Used For NAND3

The following are SPICE decks used to simulate the three-input NAND gates described in Chapter 3. The models for the n- and p-transistors are the same for each simulation, however the source and drain areas reflect the size of the devices used to implement the gate. For each case CLOAD was varied and the simulations were rerun. Similarly, two runs were performed to determine the high to low and low to high delays for each value of load capacitance. The input and output nodes are specified by comments in the decks. The current version of SPICE3 being developed at Berkeley was used.

The following deck was used for the three-input NAND on a template with NPPN transistor arrangement and solid power busses shared by adjacent columns of the same type.

```
OCT2SPICE spice input deck from i3nd3
+ 21 31 22 32 23 24 33 25 34 26 27 35 28 36 29 210 37 211 38 212 55
*
M1 1 41 2 0 CMOSN L=2U W=16U AS 80P AD 144P PS 26U PD 34U
M2 3 42 2 0 CMOSN L=2U W=16U AS 80P AD 144P PS 26U PD 34U
M3 4 43 5 0 CMOSN L=2U W=16U AS 80P AD 144P PS 26U PD 34U
M4 6 44 5 0 CMOSN L=2U W=16U AS 80P AD 144P PS 26U PD 34U
M5 7 45 8 0 CMOSN L=2U W=16U AS 80P AD 144P PS 26U PD 34U
M6 9 46 8 0 CMOSN L=2U W=16U AS 80P AD 144P PS 26U PD 34U
M7 10 47 0 11 CMOSN L=2U W=16U AS 80P AD 144P PS 26U PD 34U
M8 12 48 0 11 CMOSN L=2U W=16U AS 80P AD 144P PS 26U PD 34U
* node 0 should be ground
M21 21 31 22 55 CMOSP L=2U W=16U AS 80P AD 144P PS 26U PD 34U
M22 23 32 22 55 CMOSP L=2U W=16U AS 80P AD 144P PS 26U PD 34U
M23 24 33 25 55 CMOSP L=2U W=16U AS 80P AD 144P PS 26U PD 34U
M24 26 34 25 55 CMOSP L=2U W=16U AS 80P AD 144P PS 26U PD 34U
M25 27 35 28 55 CMOSP L=2U W=16U AS 80P AD 144P PS 26U PD 34U
M26 29 36 28 55 CMOSP L=2U W=16U AS 80P AD 144P PS 26U PD 34U
M27 210 37 211 55 CMOSP L=2U W=16U AS 80P AD 144P PS 26U PD 34U
M28 212 38 211 55 CMOSP L=2U W=16U AS 80P AD 144P PS 26U PD 34U
```

```
* node 55 should be vdd
+ VTO=-0.777    KP=19.17E-6       GAMMA=0.52      LAMBDA=4.92E-2
+ TOX=423E-10  CGSO=1.469E-10     CGDO=1.469E-10      CJ=2.4E-4
+ MJ=0.5       CJSW=3.62E-10    MJSW=0.29        XJ=0.4E-6
+ TPG=-1       LD=0.18E-6       NSUB=1.42E15    NFS=4.74E12
+ NEFF=1.001E-2       NSS=0            DELTA=1.0E-6       VMAX=34600
+ UO=235        UEXP=0.142       UCRIT=20967)
*
+ VTO=0.82      KP=48.9E-6       GAMMA=0.47      LAMBDA=1.99E-2
+ TOX=423E-10  CGSO=0.939E-10    CGDO=0.939E-10    CJ=1.4563E-4
+ MJ=0.6       CJSW=6.6E-10      MJSW=0.31        XJ=0.55E-6
+ TPG=1                 LD=0.115E-6      NSUB=1E16       NFS=5.667E12
+ NEFF=1.001E-2       NSS=0            DELTA=3.4E-5       VMAX=65466
+ UO=600        UEXP=5.325E-3   UCRIT=12714)
*
*
x0 SUPPLY 0 SUPPLY 0 SUPPLY SUPPLY 0 SUPPLY 0 SUPPLY NET1493 NET1327
+ UNC1 NET1325 NET1352 NET1352 NET1384 0 0 SUPPLY 0 SUPPLY SUPPLY
+ NET1384 NET1493 SUPPLY NET1325 NET1347 NET1327
+ SUPPLY 0 SUPPLY 0 SUPPLY 0 0 SUPPLY 0 SUPPLY 0  SUPPLY SGA
VDD SUPPLY  0 DC 5.0
*add load capacitance
CLOAD NET1493 0 1PF
*
*high to low delay  ***** NO load cap added
*VA NET1327 0 PULSE( 0 5.0 2NS 2NS 2NS 20NS 50NS)
*VB NET1325 0 PULSE( 0 5.0 2NS 2NS 2NS 20NS 50NS)
*VC NET1384 0 PULSE( 0 5.0 2NS 2NS 2NS 20NS 50NS)
*j
*low to high delay  ***** NO load cap added
VA NET1327 0 PULSE( 5.0 0  02NS 2NS 2NS 20NS 50NS)
VB NET1325 0 PULSE( 5.0 0 2NS 2NS 2NS 20NS 50NS)
VC NET1384 0 PULSE( 5.0 0 2NS 2NS 2NS 20NS 50NS)
*
*the output node is NET1493
*
```

The following deck was used for the three-input NAND on a template with NPNP transistor arrangement.

```
OCT2SPICE spice input deck from lsind3sim
+ 21 31 22 32 23 24 33 25 34 26 55
M1 1 41 2 0 CMOSN L=2U W=21U AS 105P AD 189P PS 31U PD 39U
M2 3 42 2 0 CMOSN L=2U W=21U AS 105P AD 189P PS 31U PD 39U
M3 4 43 5 0 CMOSN L=2U W=21U AS 105P AD 189P PS 31U PD 39U
M4 6 44 5 0 CMOSN L=2U W=21U AS 105P AD 189P PS 31U PD 39U
*
* node 55 should be vdd
* node 0 should be ground
M21 21 31 22 55 CMOSP L=2U W=21U AS 105P AD 189P PS 31U PD 39U
M22 23 32 22 55 CMOSP L=2U W=21U AS 105P AD 189P PS 31U PD 39U
M23 24 33 25 55 CMOSP L=2U W=21U AS 105P AD 189P PS 31U PD 39U
M24 26 34 25 55 CMOSP L=2U W=21U AS 105P AD 189P PS 31U PD 39U
+ VTO=-0.777    KP=19.17E-6      GAMMA=0.52     LAMBDA=4.92E-2
+ TOX=423E-10  CGSO=1.469E-10    CGDO=1.469E-10    CJ=2.4E-4
+ MJ=0.5       CJSW=3.62E-10   MJSW=0.29       XJ=0.4E-6
+ TPG=-1       LD=0.18E-6      NSUB=1.42E15   NFS=4.74E12
+ NEFF=1.001E-2       NSS=0          DELTA=1.0E-6      VMAX=34600
+ UO=235       UEXP=0.142      UCRIT=20967)
*
+ VTO=0.82      KP=48.9E-6       GAMMA=0.47      LAMBDA=1.99E-2
+ TOX=423E-10  CGSO=0.939E-10   CGDO=0.939E-10    CJ=1.4563E-4
+ MJ=0.6       CJSW=6.6E-10     MJSW=0.31       XJ=0.55E-6
+ TPG=1              LD=0.115E-6      NSUB=1E16       NFS=5.667E12
+ NEFF=1.001E-2       NSS=0          DELTA=3.4E-5      VMAX=65466
+ UO=600       UEXP=5.325E-3   UCRIT=12714)
*
*
x0 SUPPLY 0 0 NET1232 NET1287 NET1287 NET1222 UNC1 NET1220
+ NET1558 0 SUPPLY SUPPLY NET1232 NET1558 SUPPLY NET1222
+ NET1558 NET1220 SUPPLY SUPPLY SGB
x2 SUPPLY 0 SUPPLY 0 SUPPLY SUPPLY 0 SUPPLY 0 SUPPLY 0 SUPPLY
+ 0 SUPPLY 0 0 SUPPLY 0 SUPPLY 0  SUPPLY SGB
x4 SUPPLY 0 SUPPLY 0 SUPPLY SUPPLY 0 SUPPLY 0 SUPPLY 0 SUPPLY
+ 0 SUPPLY 0 0 SUPPLY 0 SUPPLY 0  SUPPLY SGB
VDD SUPPLY  0 DC 5.0
*add load capacitance
CLOAD NET1558 0 1PF
*
*high to low delay  ***** NO load cap added
*VA NET1220 0 PULSE( 0 5.0 2NS 2NS 2NS 20NS 50NS)
*VB NET1222 0 PULSE( 0 5.0 2NS 2NS 2NS 20NS 50NS)
*VC NET1232 0 PULSE( 0 5.0 2NS 2NS 2NS 20NS 50NS)
*j
```

```
*low to high delay  ***** NO load cap added
VA NET1220 0 PULSE( 5.0 0  02NS 2NS 2NS 20NS 50NS)
VB NET1222 0 PULSE( 5.0 0 2NS 2NS 2NS 20NS 50NS)
VC NET1232 0 PULSE( 5.0 0 2NS 2NS 2NS 20NS 50NS)
*
*the output node is NET1558
*
```

The following deck was used for the three-input NAND on a template with NPPN transistor arrangement with minimum size transistors.

```
OCT2SPICE spice input deck from i3nd3
+ 21 31 22 32 23 24 33 25 34 26 27 35 28 36 29 210 37 211 38 212 55
M1 1 41 2 0 CMOSN L=2U W=8U AS 40P AD 72P PS 10U PD 26U
M2 3 42 2 0 CMOSN L=2U W=8U AS 40P AD 72P PS 10U PD 26U
M3 4 43 5 0 CMOSN L=2U W=8U AS 40P AD 72P PS 10U PD 26U
M4 6 44 5 0 CMOSN L=2U W=8U AS 40P AD 72P PS 10U PD 26U
M5 7 45 8 0 CMOSN L=2U W=8U AS 40P AD 72P PS 10U PD 26U
M6 9 46 8 0 CMOSN L=2U W=8U AS 40P AD 72P PS 10U PD 26U
M7 10 47 0 11 CMOSN L=2U W=8U AS 40P AD 72P PS 10U PD 26U
M8 12 48 0 11 CMOSN L=2U W=8U AS 40P AD 72P PS 10U PD 26U
*
* node 55 should be vdd
* node 0 should be ground
M21 21 31 22 55 CMOSP L=2U W=8U AS 40P AD 72P PS 10U PD 26U
M22 23 32 22 55 CMOSP L=2U W=8U AS 40P AD 72P PS 10U PD 26U
M23 24 33 25 55 CMOSP L=2U W=8U AS 40P AD 72P PS 10U PD 26U
M24 26 34 25 55 CMOSP L=2U W=8U AS 40P AD 72P PS 10U PD 26U
M25 27 35 28 55 CMOSP L=2U W=8U AS 40P AD 72P PS 10U PD 26U
M26 29 36 28 55 CMOSP L=2U W=8U AS 40P AD 72P PS 10U PD 26U
M27 210 37 211 55 CMOSP L=2U W=8U AS 40P AD 72P PS 10U PD 26U
M28 212 38 211 55 CMOSP L=2U W=8U AS 40P AD 72P PS 10U PD 26U
+ VTO=-0.777     KP=19.17E-6       GAMMA=0.52     LAMBDA=4.92E-2
+ TOX=423E-10  CGSO=1.469E-10     CGDO=1.469E-10     CJ=2.4E-4
+ MJ=0.5      CJSW=3.62E-10    MJSW=0.29      XJ=0.4E-6
+ TPG=-1      LD=0.18E-6      NSUB=1.42E15    NFS=4.74E12
+ NEFF=1.001E-2       NSS=0          DELTA=1.0E-6       VMAX=34600
+ UO=235       UEXP=0.142      UCRIT=20967)
*
+ VTO=0.82      KP=48.9E-6       GAMMA=0.47      LAMBDA=1.99E-2
+ TOX=423E-10  CGSO=0.939E-10    CGDO=0.939E-10    CJ=1.4563E-4
+ MJ=0.6       CJSW=6.6E-10     MJSW=0.31       XJ=0.55E-6
+ TPG=1               LD=0.115E-6      NSUB=1E16       NFS=5.667E12
+ NEFF=1.001E-2       NSS=0            DELTA=3.4E-5      VMAX=65466
+ UO=600       UEXP=5.325E-3   UCRIT=12714)
*
*
x0 SUPPLY 0 SUPPLY 0 SUPPLY SUPPLY 0 SUPPLY 0 SUPPLY NET1493
+ NET1327 UNC1 NET1325 NET1352 NET1352 NET1384 0 0 SUPPLY 0
+ SUPPLY SUPPLY NET1384 NET1493 SUPPLY NET1325 NET1347 NET1327
+ SUPPLY 0 SUPPLY 0 SUPPLY 0 0 SUPPLY 0 SUPPLY 0  SUPPLY SGA
VDD SUPPLY  0 DC 5.0
*add load capacitance
CLOAD NET1493 0 .5PF
*
*high to low delay  ***** NO load cap added
*VA NET1327 0 PULSE( 0 5.0 2NS 2NS 2NS 20NS 50NS)
*VB NET1325 0 PULSE( 0 5.0 2NS 2NS 2NS 20NS 50NS)
*VC NET1384 0 PULSE( 0 5.0 2NS 2NS 2NS 20NS 50NS)
*j
*low to high delay  ***** NO load cap added
VA NET1327 0 PULSE( 5.0 0  02NS 2NS 2NS 20NS 50NS)
VB NET1325 0 PULSE( 5.0 0 2NS 2NS 2NS 20NS 50NS)
VC NET1384 0 PULSE( 5.0 0 2NS 2NS 2NS 20NS 50NS)
*
*the output node is NET1493
*
```

The following is the SPICE deck used to simulate a three-input AND on a template with NPPN transistor arrangement with minimum size transistors.

```
OCT2SPICE spice input deck from sknd3
+ 21 31 22 32 23 24 33 25 34 26 27 35 28 36 29 210 37 211 38 212 55
M1 1 41 2 0 CMOSN L=2U W=8U AS 40P AD 72P PS 10U PD 26U
M2 3 42 2 0 CMOSN L=2U W=8U AS 40P AD 72P PS 10U PD 26U
M3 4 43 5 0 CMOSN L=2U W=8U AS 40P AD 72P PS 10U PD 26U
M4 6 44 5 0 CMOSN L=2U W=8U AS 40P AD 72P PS 10U PD 26U
M5 7 45 8 0 CMOSN L=2U W=8U AS 40P AD 72P PS 10U PD 26U
M6 9 46 8 0 CMOSN L=2U W=8U AS 40P AD 72P PS 10U PD 26U
M7 10 47 0 11 CMOSN L=2U W=8U AS 40P AD 72P PS 10U PD 26U
M8 12 48 0 11 CMOSN L=2U W=8U AS 40P AD 72P PS 10U PD 26U
*
```

```
* node 55 should be vdd
* node 0 should be ground
M21 21 31 22 55 CMOSP L=2U W=8U AS 40P AD 72P PS 10U PD 26U
M22 23 32 22 55 CMOSP L=2U W=8U AS 40P AD 72P PS 10U PD 26U
M23 24 33 25 55 CMOSP L=2U W=8U AS 40P AD 72P PS 10U PD 26U
M24 26 34 25 55 CMOSP L=2U W=8U AS 40P AD 72P PS 10U PD 26U
M25 27 35 28 55 CMOSP L=2U W=8U AS 40P AD 72P PS 10U PD 26U
M26 29 36 28 55 CMOSP L=2U W=8U AS 40P AD 72P PS 10U PD 26U
M27 210 37 211 55 CMOSP L=2U W=8U AS 40P AD 72P PS 10U PD 26U
M28 212 38 211 55 CMOSP L=2U W=8U AS 40P AD 72P PS 10U PD 26U
+ VTO=-0.777     KP=19.17E-6      GAMMA=0.52     LAMBDA=4.92E-2
+ TOX=423E-10  CGSO=1.469E-10     CGDO=1.469E-10     CJ=2.4E-4
+ MJ=0.5       CJSW=3.62E-10   MJSW=0.29      XJ=0.4E-6
+ TPG=-1       LD=0.18E-6      NSUB=1.42E15    NFS=4.74E12
+ NEFF=1.001E-2       NSS=0          DELTA=1.0E-6      VMAX=34600
+ UO=235       UEXP=0.142      UCRIT=20967)
*
+ VTO=0.82      KP=48.9E-6      GAMMA=0.47     LAMBDA=1.99E-2
+ TOX=423E-10  CGSO=0.939E-10   CGDO=0.939E-10    CJ=1.4563E-4
+ MJ=0.6       CJSW=6.6E-10     MJSW=0.31      XJ=0.55E-6
+ TPG=1               LD=0.115E-6     NSUB=1E16      NFS=5.667E12
+ NEFF=1.001E-2       NSS=0          DELTA=3.4E-5      VMAX=65466
+ UO=600       UEXP=5.325E-3   UCRIT=12714)
*
*
*
x0 SUPPLY NET2882 SUPPLY NET2882 SUPPLY SUPPLY NET2882 SUPPLY
+ NET2882 SUPPLY NET2152 NET2153 NET2278 NET2150 NET2276 NET2276
+ NET2151 NET2882 NET2152 NET2160 NET2160 NET2152 SUPPLY NET2151
+ NET2152 SUPPLY NET2150 NET2152 NET2153 SUPPLY NET2882 SUPPLY
+ NET2882 SUPPLY NET2882 NET2882 SUPPLY NET2882 SUPPLY NET2882 SUPP
*
x1 SUPPLY NET2882 SUPPLY NET2882 SUPPLY SUPPLY NET2882 SUPPLY
+ NET2882 SUPPLY NET2271 NET2160 NET2882 NET2882 SUPPLY SUPPLY
+ NET2882 SUPPLY NET2882 SUPPLY NET2882 SUPPLY NET2882 SUPPLY
+ NET2882 NET2882 SUPPLY SUPPLY NET2160 NET2271 NET2882 SUPPLY
+ NET2882 SUPPLY NET2882 NET2882 SUPPLY NET2882 SUPPLY NET2882 SUPP
*
VDD SUPPLY  0  5.0
*add load capacitance
*CLOAD NET2152 0 .5PF
*
*high to low delay  ***** NO load cap added
*VA NET2153 0 PULSE( 0 5.0 2NS 2NS 2NS 20NS 50NS)
```

```
*VB NET2150 0 PULSE( 0 5.0 2NS 2NS 2NS 20NS 50NS)
*VC NET2151 0 PULSE( 0 5.0 2NS 2NS 2NS 20NS 50NS)
*j
*low to high delay  ***** NO load cap added
VA NET2153 0 PULSE( 5.0 0  02NS 2NS 2NS 20NS 50NS)
VB NET2150 0 PULSE( 5.0 0 2NS 2NS 2NS 20NS 50NS)
VC NET2151 0 PULSE( 5.0 0 2NS 2NS 2NS 20NS 50NS)
*
*z1 is net2152
*zbar is 216
*the output node is NET2271
*
```

Appendix B

Examples of Core Cell Generation

The following is an example of the default template generated by the skipper program. Figure B.1 shows the resulting template.

```
prompt > skipper
Do you want power in metal 1? [yes]
Do you want to give the number of transistors (as opposed to pads)? [yes]
How many transistors? [50000]
Maximum expected frequency of operation in MHz? [10]
How many blocks between plugs? [2]
How many transistors per block? [2]
How many vertical wires between power and ground? [6]
How many tracks over the n transistor? [3]
That leaves 3 tracks for the p transistor.
Power rail width? [14]
Width of N is 20, width of P is 20
Want to make these less? [no]
The template will have 50000 transistors, 54 pads.
Name of the new template? [template] default
All done.
```

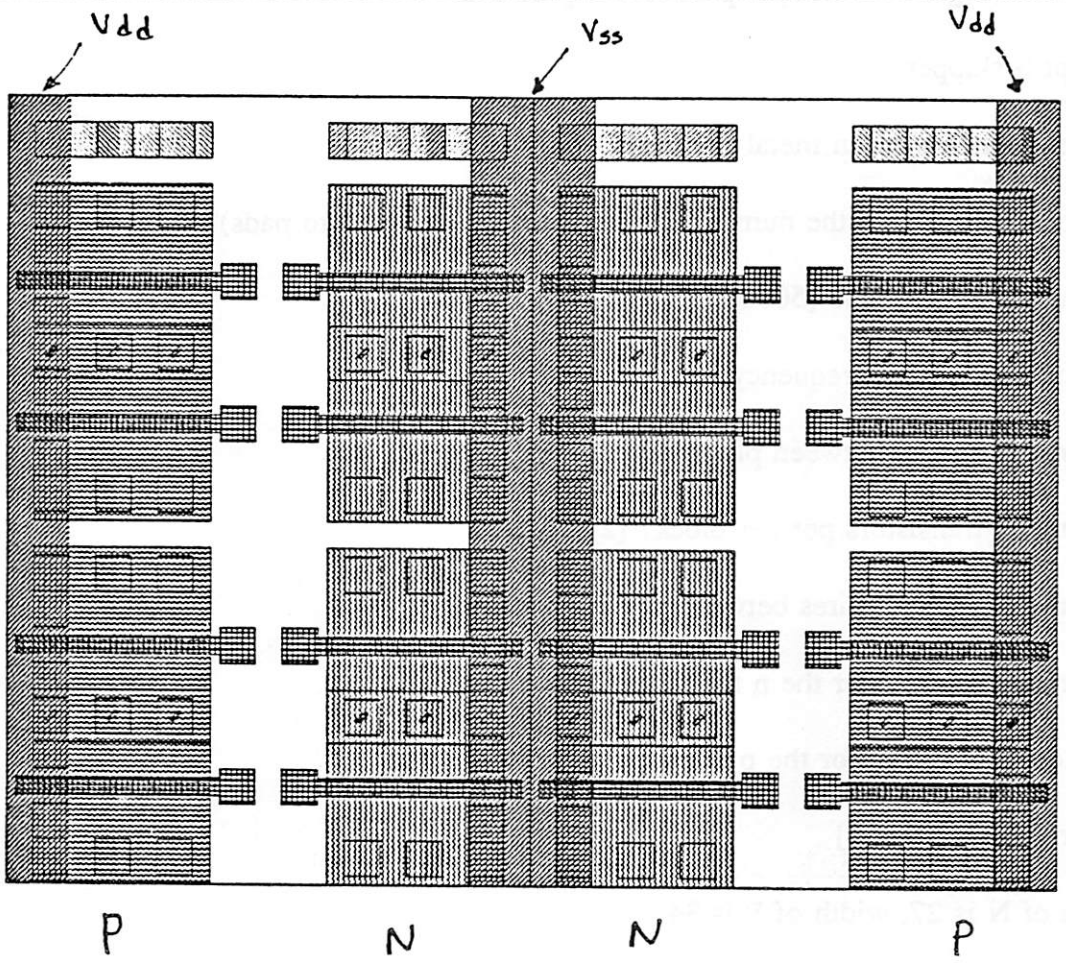


Figure B.1 The default template

The following is an example produced Figure B.2.

```
prompt > skipper

Do you want power in metal 1? [yes]

Do you want to give the number of transistors (as opposed to pads)? [yes]

How many transistors? [50000]

Maximum expected frequency of operation in MHz? [10]

How many blocks between plugs? [2]

How many transistors per per block? [2]

How many vertical wires between power and ground? [6] 9

How many tracks over the n transistor? [4]

That leaves 5 tracks for the p transistor.

Power rail width? [14]

Width of N is 27, width of P is 34

ant to make these less? [no]

The template will have 50000 transistors, 64 pads.

Name of the new template? [template] bigp

All done.
```

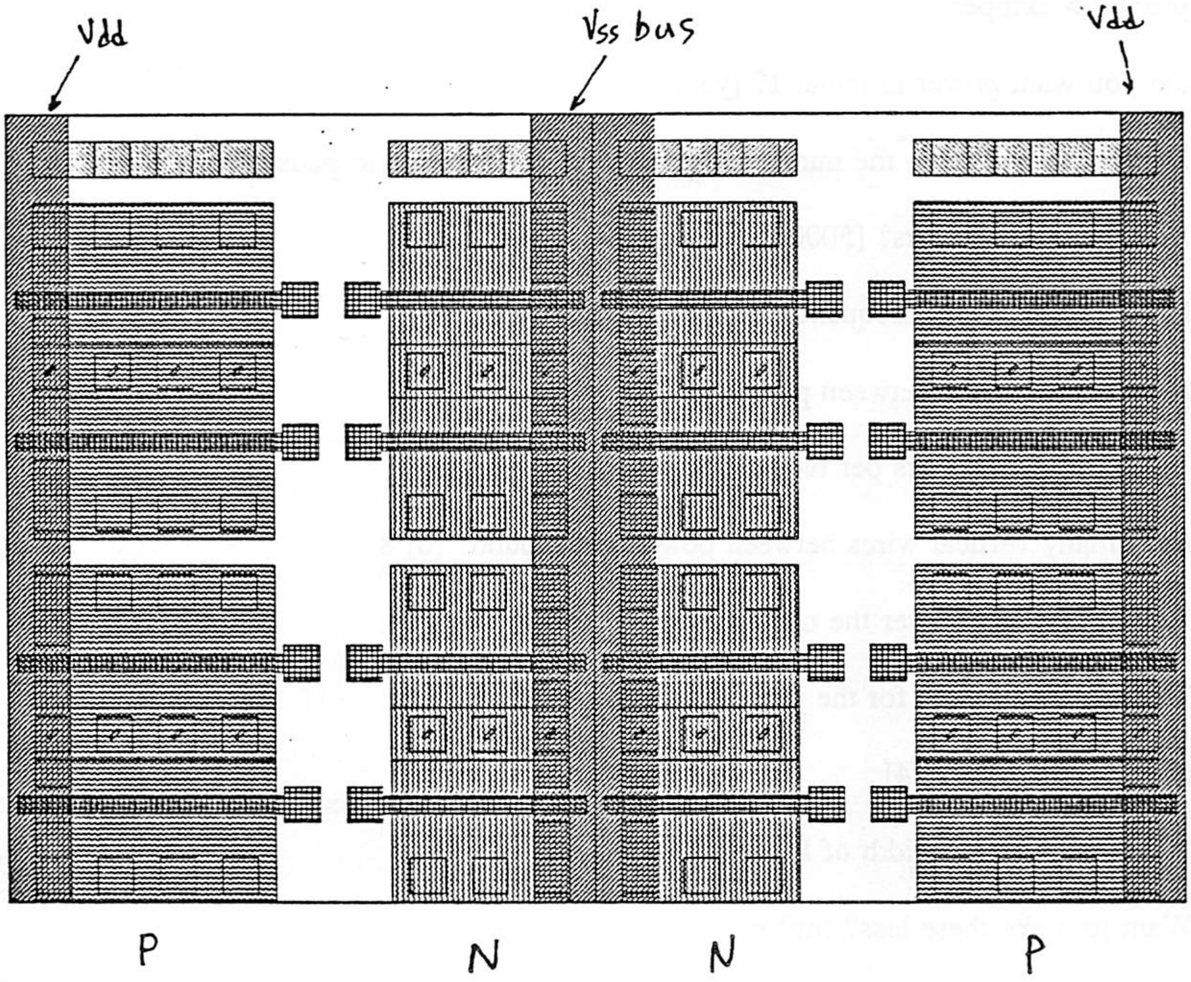


Figure B.2

The following produced Figure B.3.

```
prompt > skipper

Do you want power in metal 1? [yes]

Do you want to give the number of transistors (as opposed to pads)? [yes]

How many transistors? [50000]

Maximum expected frequency of operation in MHz? [10]

How many blocks between plugs? [2]

How many transistors per block? [2]

How many vertical wires between power and ground? [6] 8

How many tracks over the n transistor? [4]

That leaves 4 tracks for the p transistor.

Power rail width? [14]

Width of N is 27, width of P is 27

Want to make these less? [no] y

Width of an N transistor? [27] 20

Width of a P transistor? [27] 20

The template will have 50000 transistors, 62 pads.

Name of the new template? [template] fingers

All done.
```

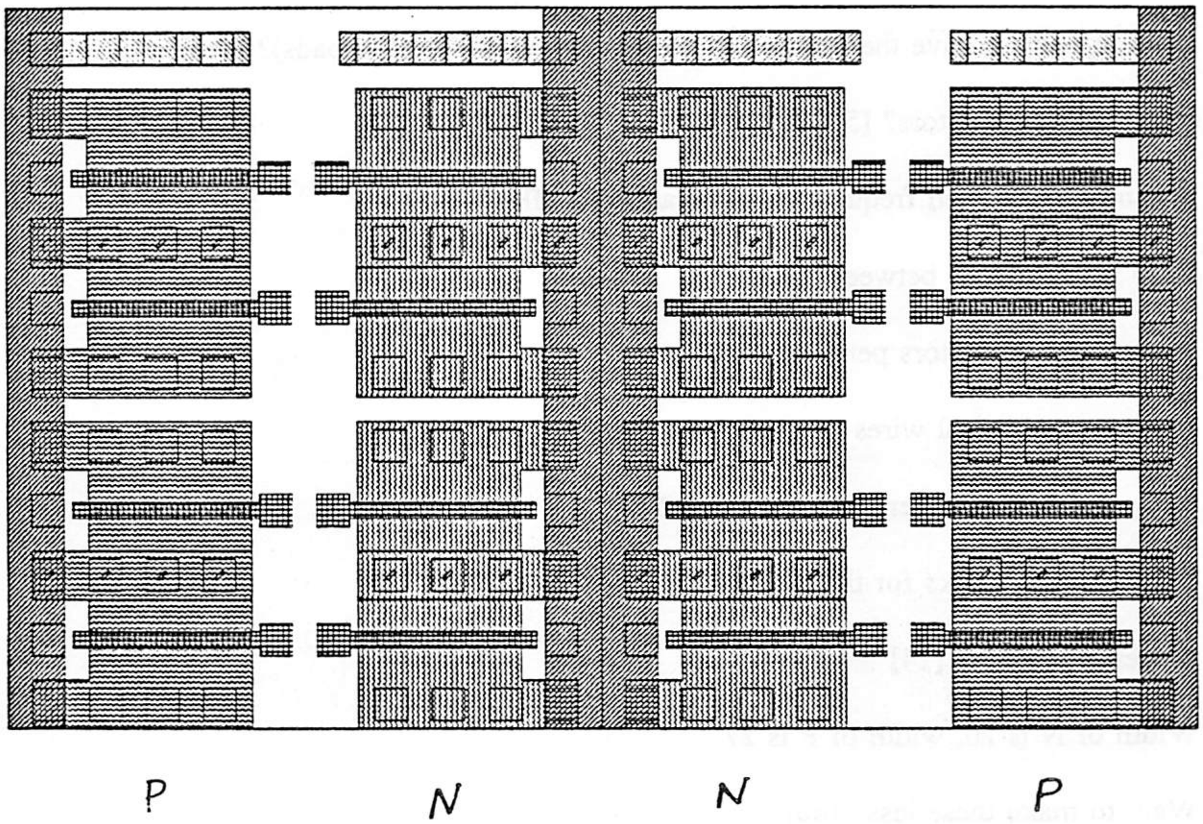


Figure B.3

The following skipper session produced Figure B.4.

```
prompt > skipper

Do you want power in metal 1? [yes]

Do you want to give the number of transistors (as opposed to pads)? [yes]

How many transistors? [50000]

Maximum expected frequency of operation in MHz? [10]

How many blocks between plugs? [2]

How many transistors per block? [2]

How many vertical wires between power and ground? [6] 7

How many tracks over the n transistor? [3]

That leaves 4 tracks for the p transistor.

Power rail width? [14]

Width of N is 20, width of P is 27

Want to make these less? [no]

The template will have 50000 transistors, 58 pads.

Name of the new template? [template] odd7

All done.
```

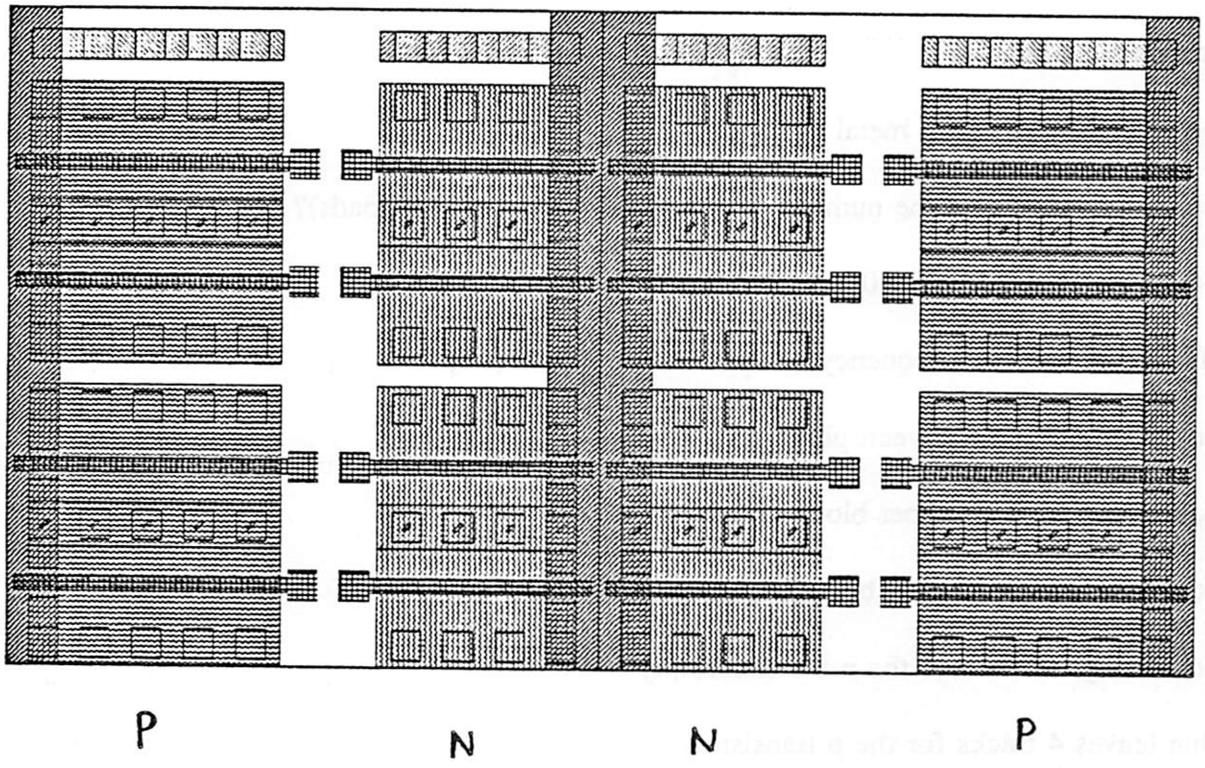


Figure B.4

The following skipper session produced Figure B.5.

```
prompt > skipper
Do you want power in metal 1? [yes]
Do you want to give the number of transistors (as opposed to pads)? [yes]
How many transistors? [50000]
Maximum expected frequency of operation in MHz? [10]
How many blocks between plugs? [2]
How many transistors per block? [2]
How many vertical wires between power and ground? [6] 8
How many tracks over the n transistor? [4]
That leaves 4 tracks for the p transistor.
Want to make these less? [no] y
Width of an N transistor? [27] 20
Width of a P transistor? [27] 25
The template will have 50000 transistors, 62 pads.
Name of the new template? [template] smfingers
All done.
```

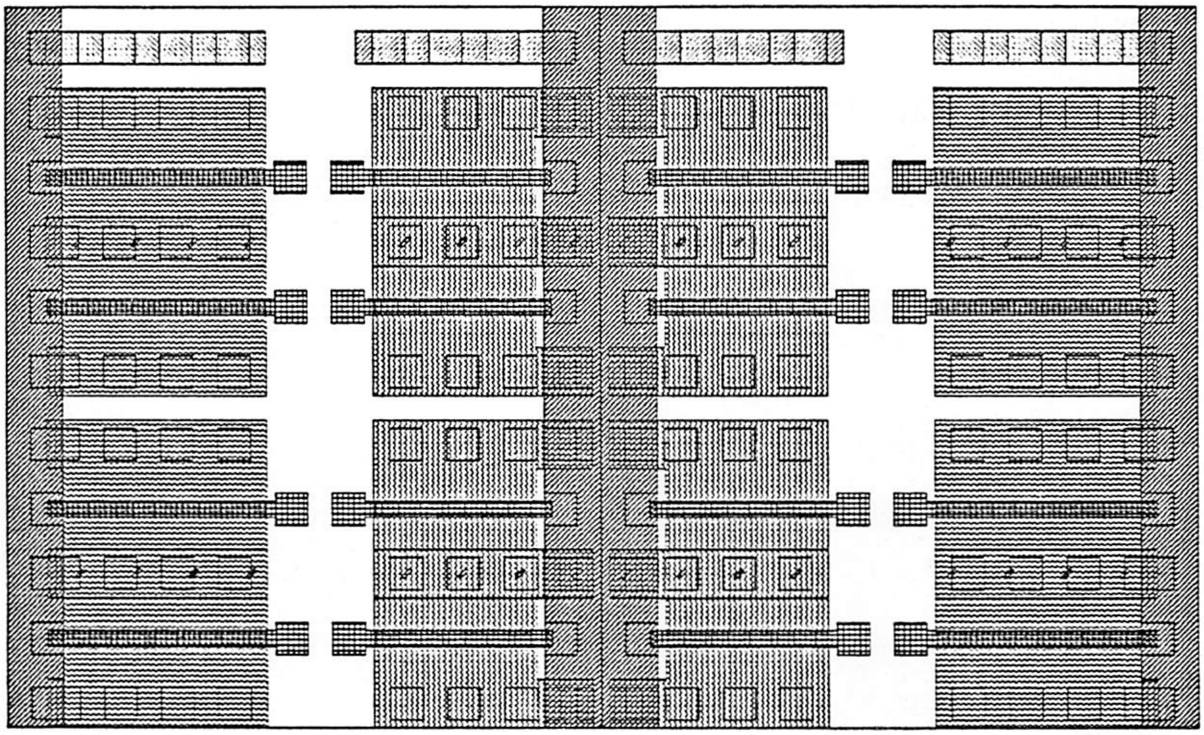


Figure B.4

Appendix C

Examples of Library Cell Generation

The following cells were regenerated from cells designed manually to fit on the default template. The manual page for the Regen program is given in Appendix F.

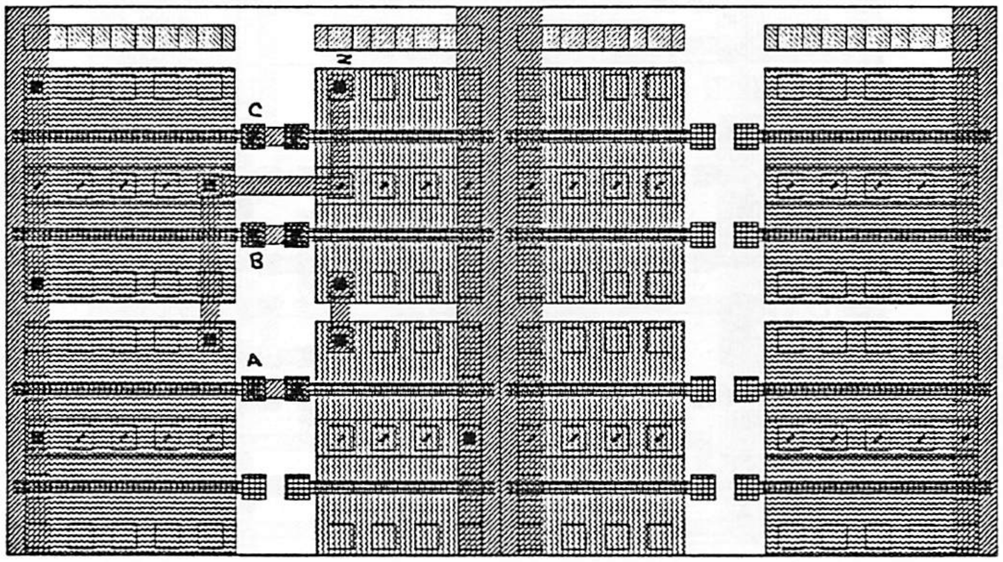


Figure C.1 A three-input NAND regenerated to fit on the template given in Figure B.2

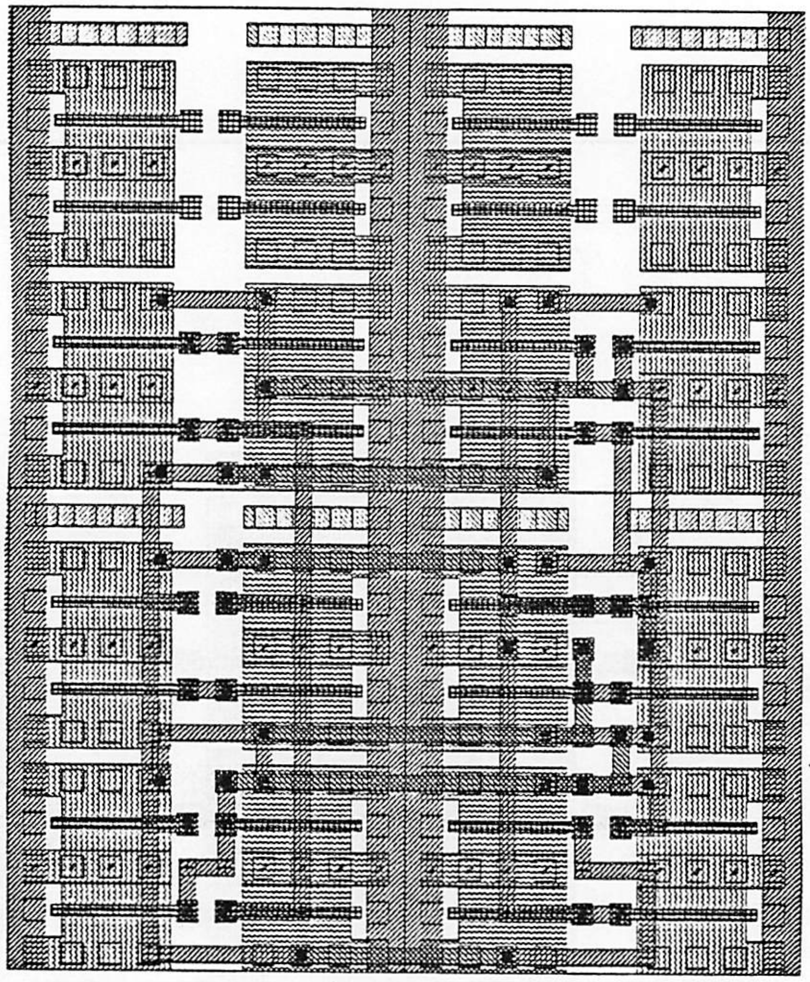
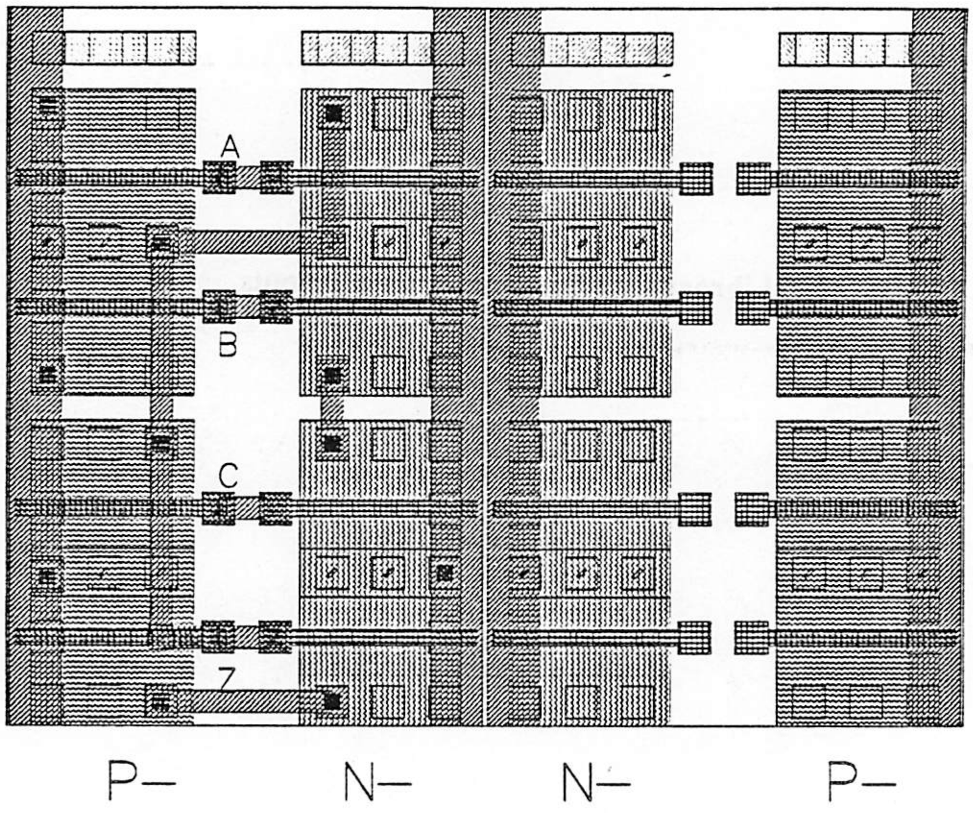


Figure C.2 A static D-flip flop regenerated to fit on the template given in Figure B.3

Appendix D

Library Cell Schematics and Layouts



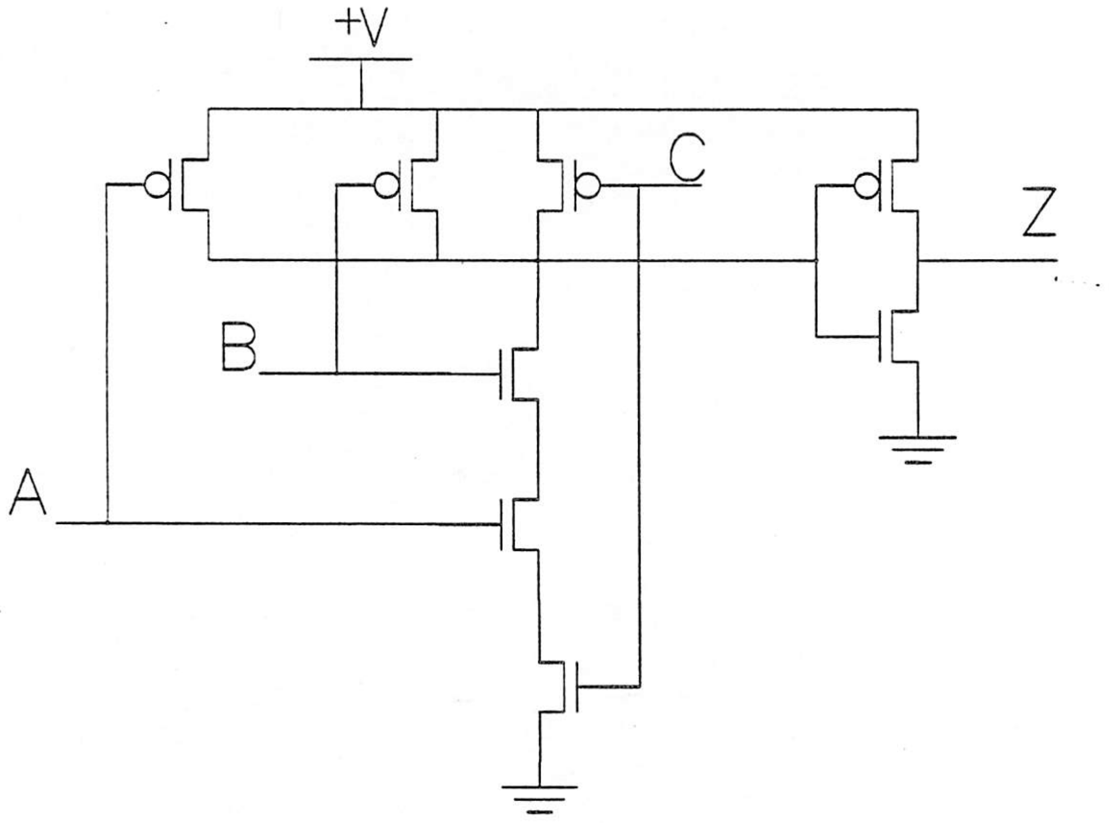
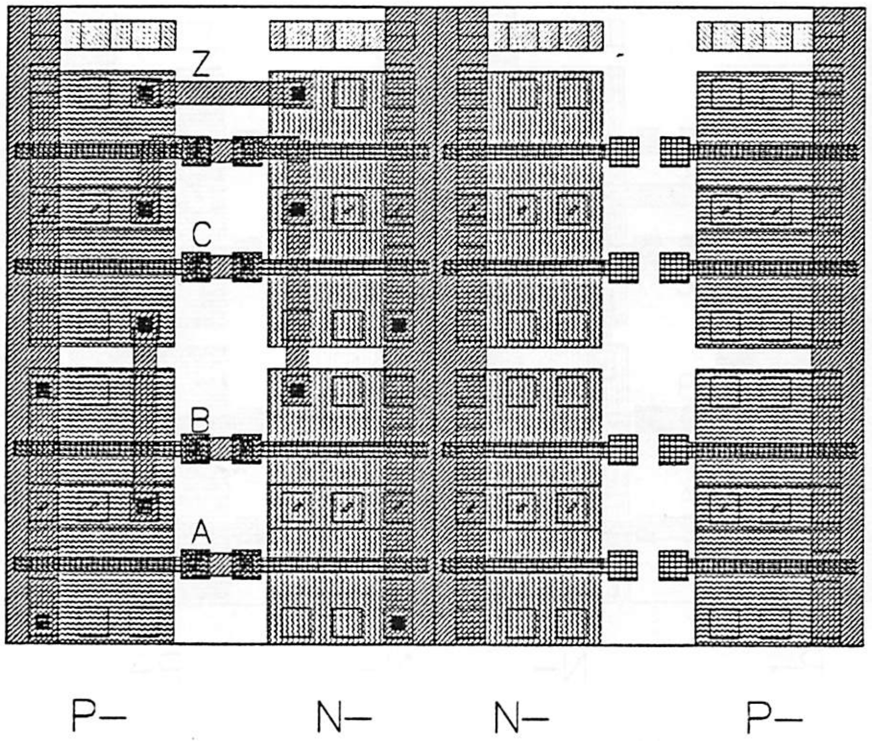


Figure D.1 and3 z = ab



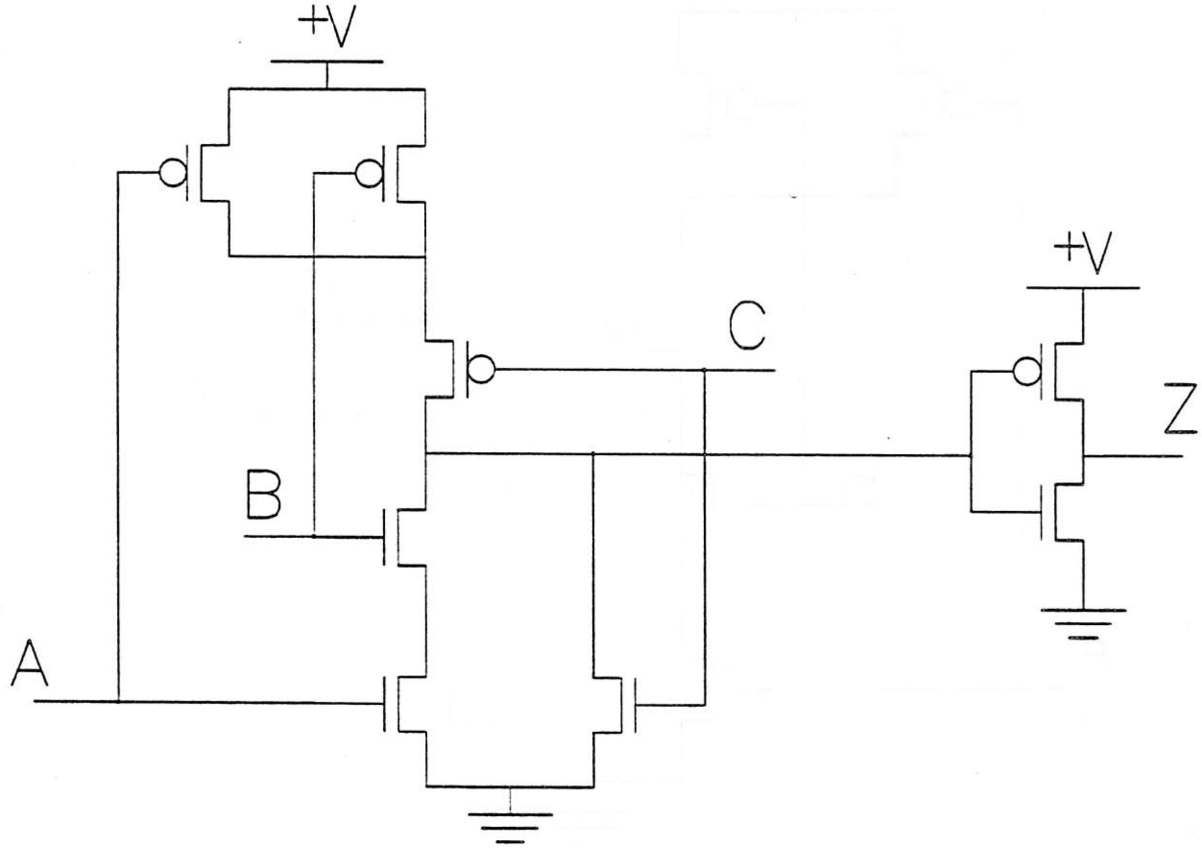
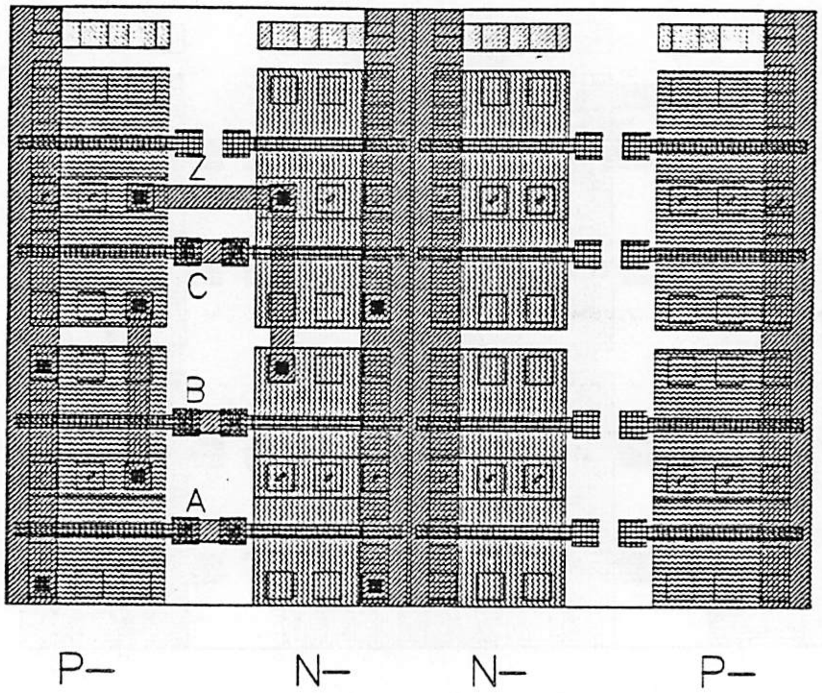


Figure D.2 ao21 z = ab + c



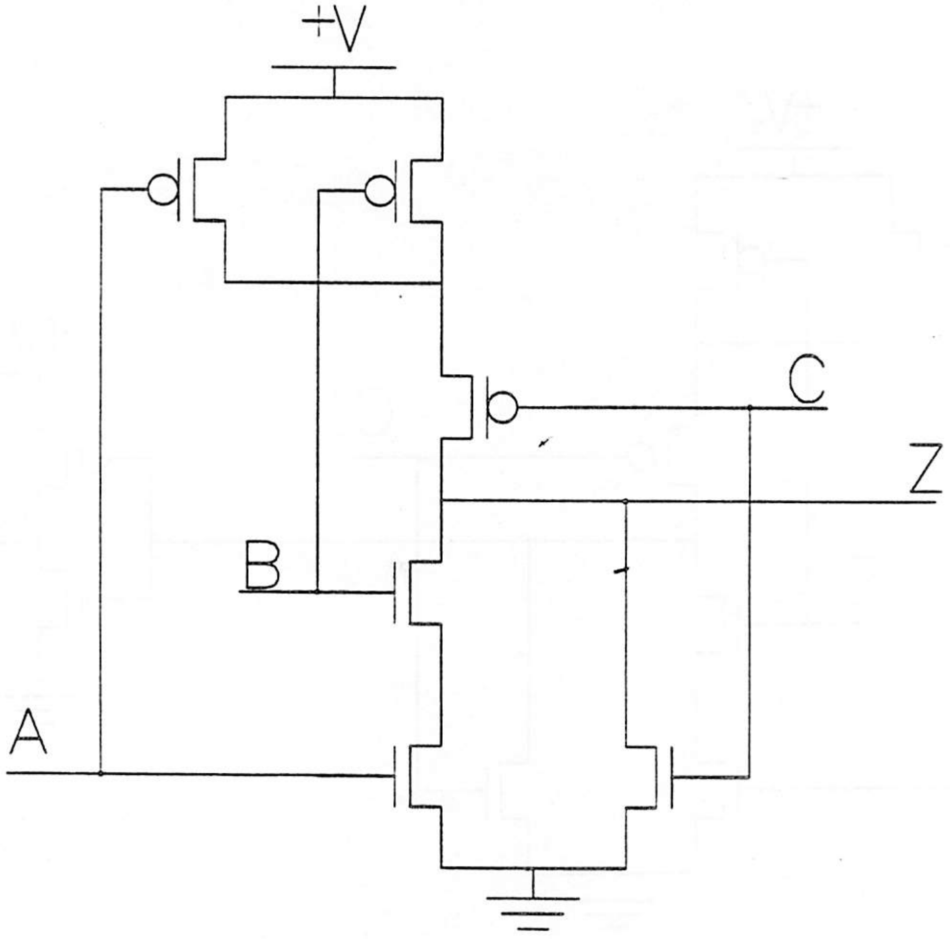
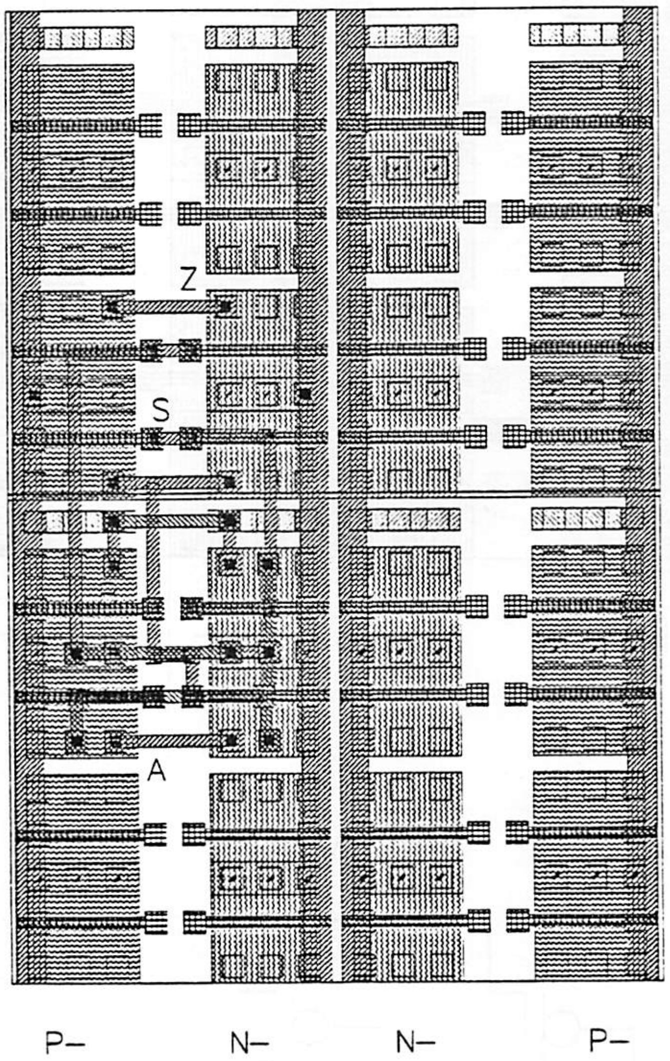


Figure D.3 aoi21 z = !(ab + c)



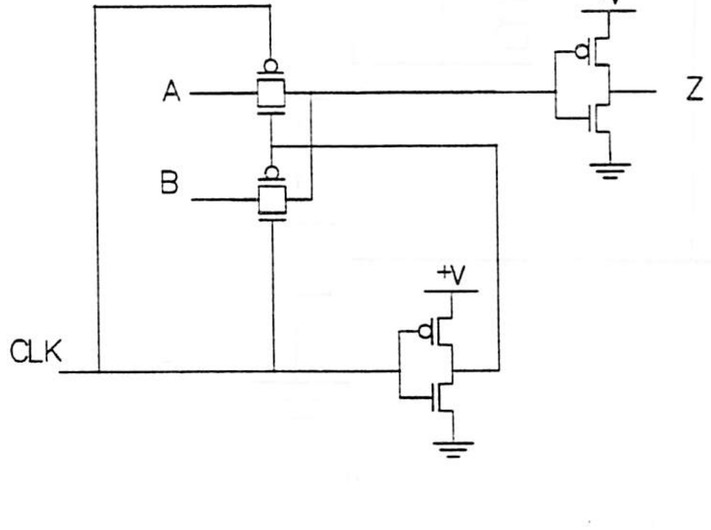
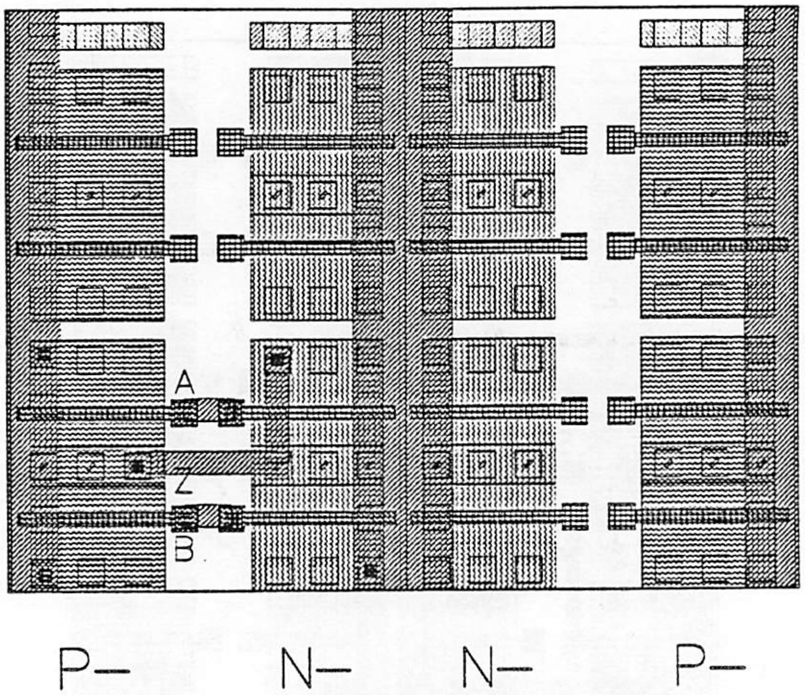


Figure D.12 muxinv2 z = !(as + b!s)



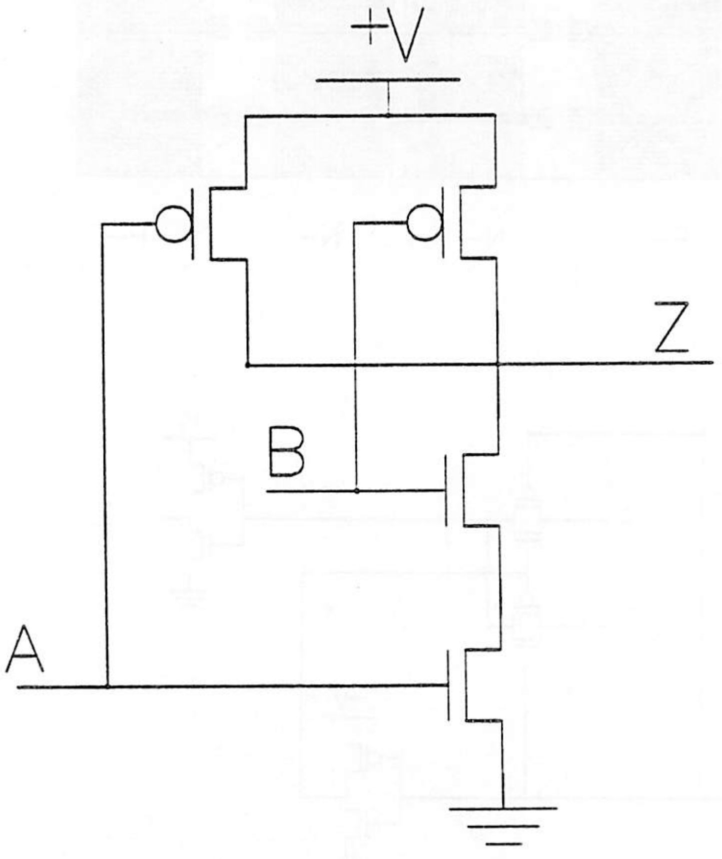
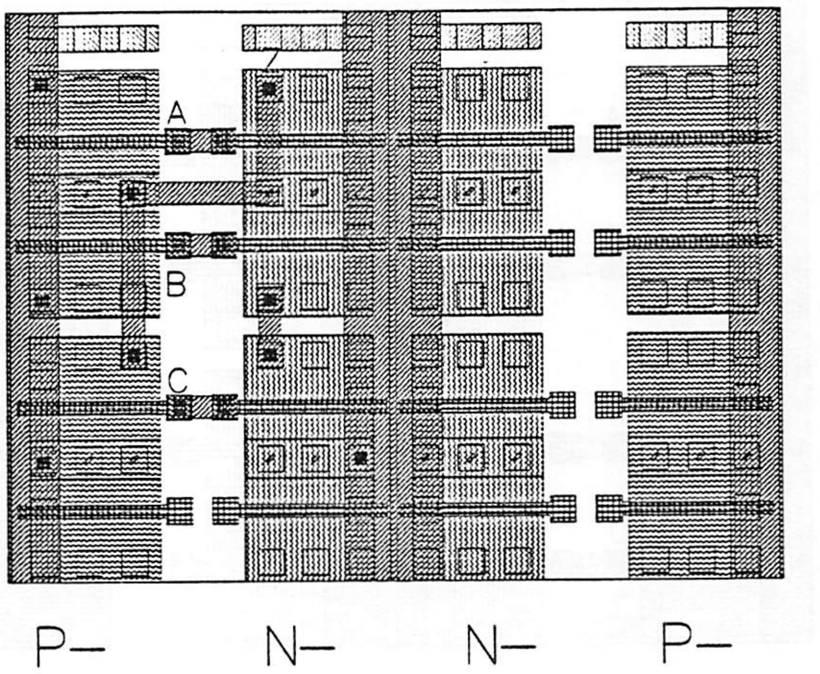


Figure D.13 nd2 z = !(ab)



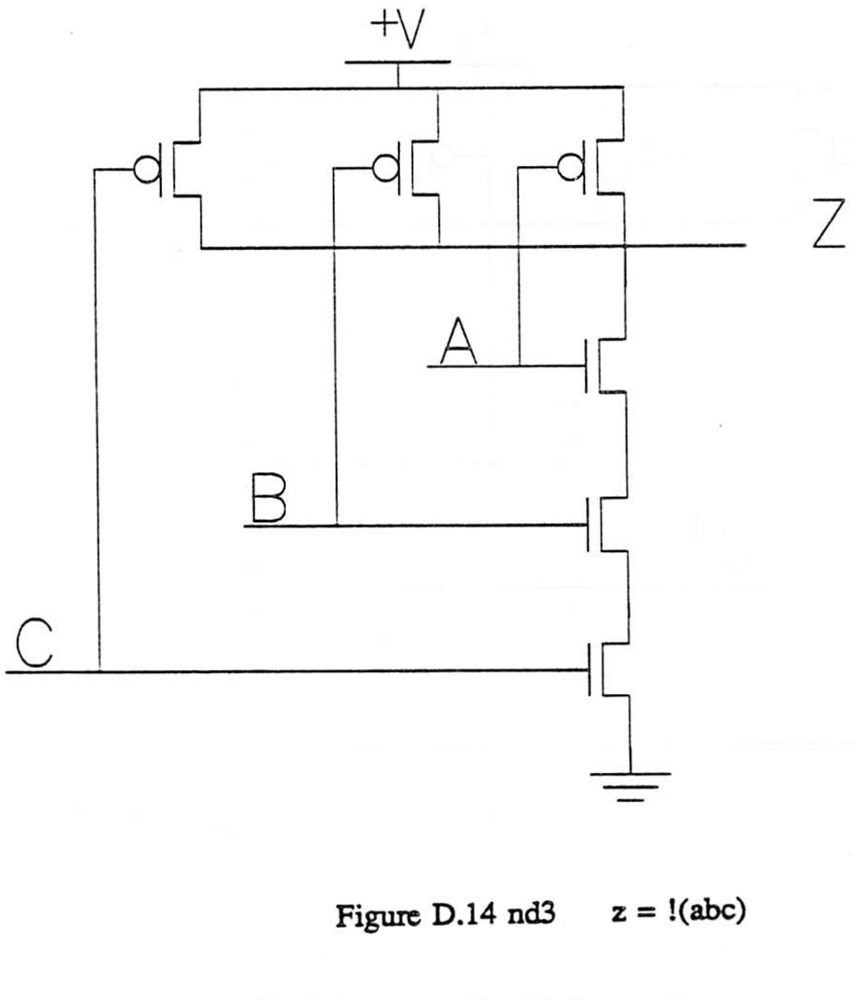
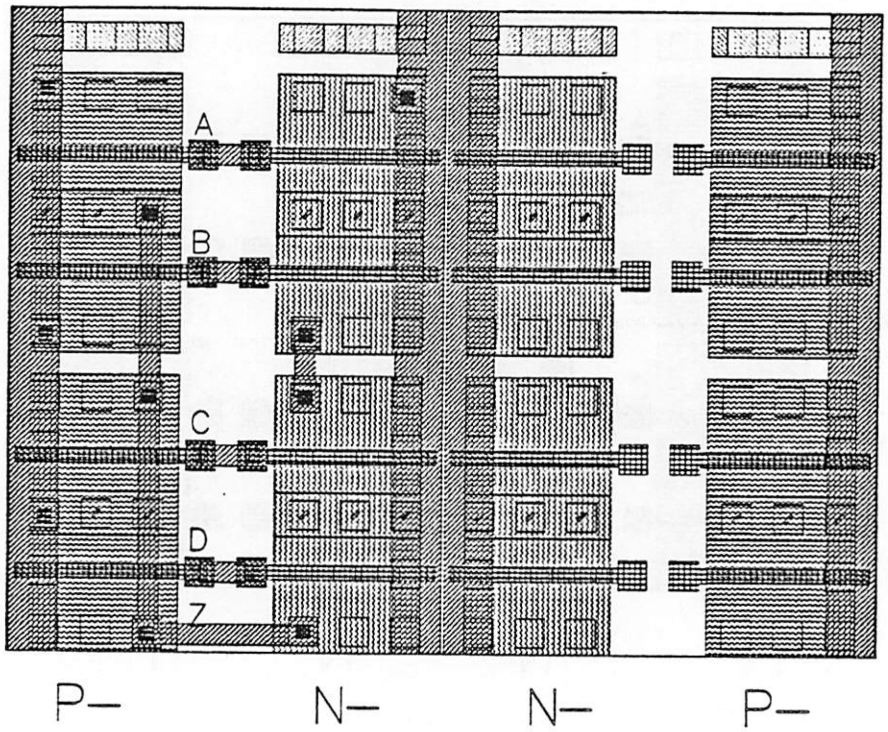


Figure D.14 nd3 z = !(abc)



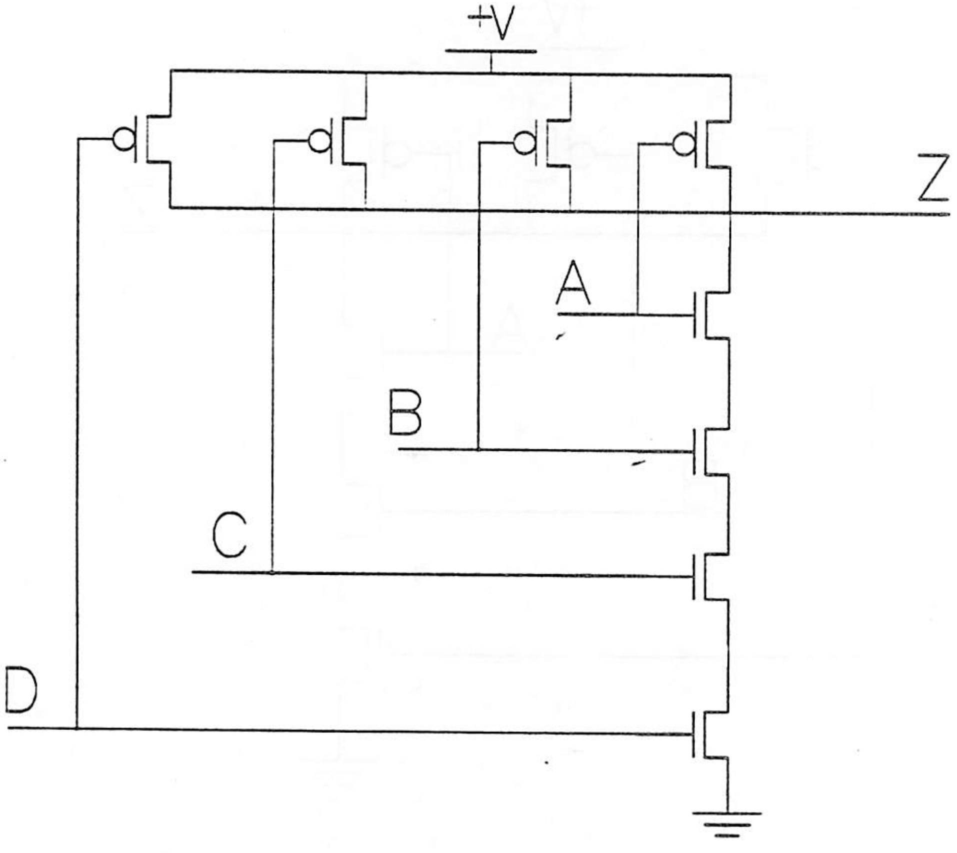
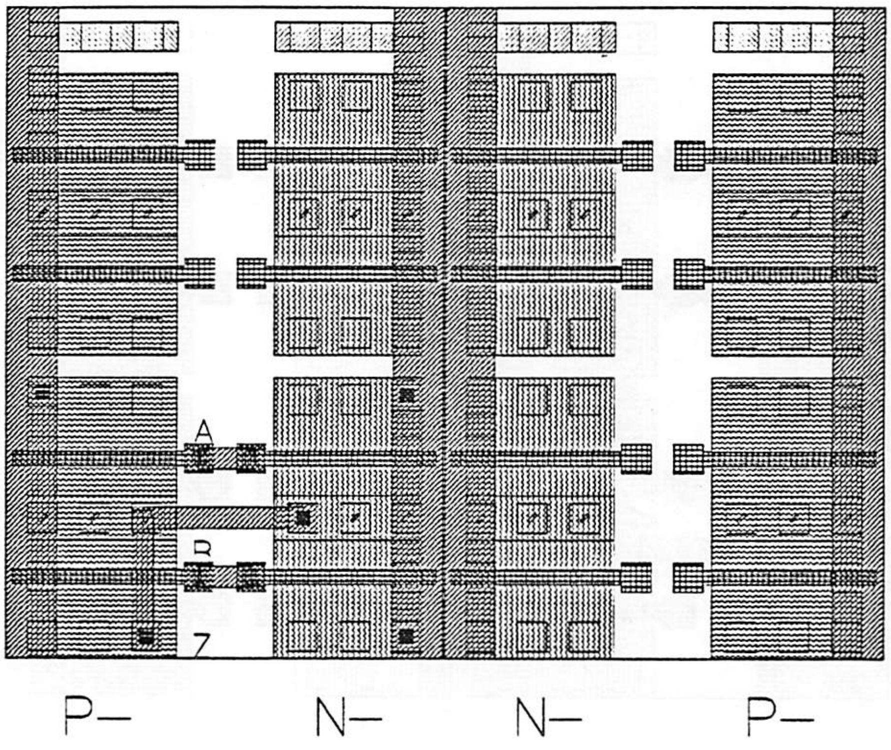


Figure D.15 nd4 z = !(abcd)



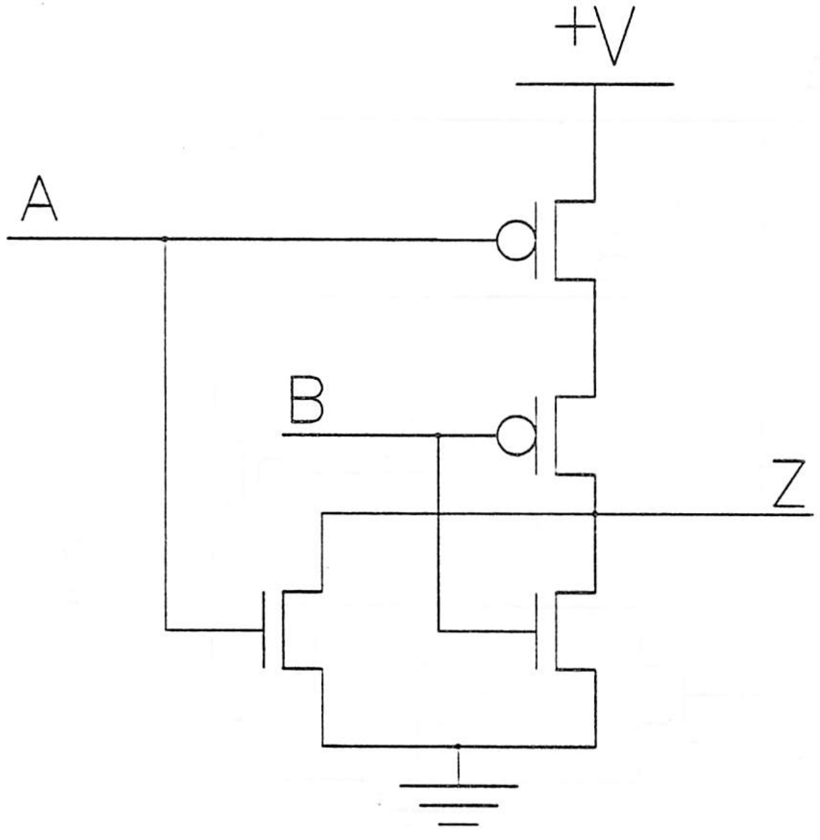
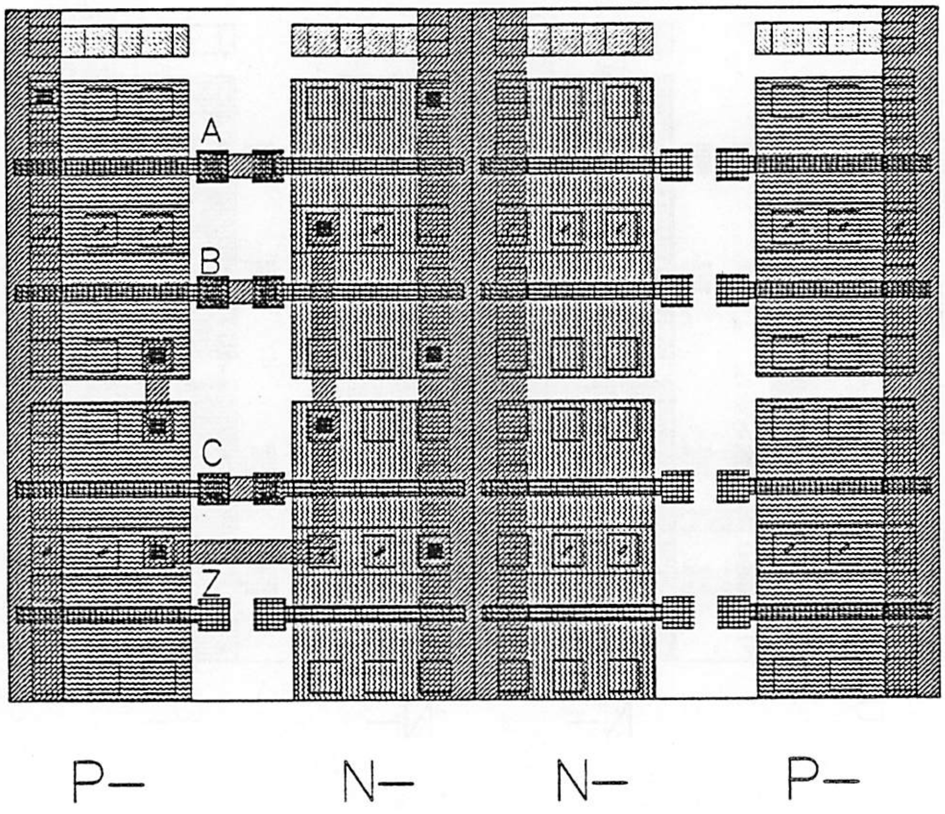


Figure D.16 nor2 z = !(a + b)



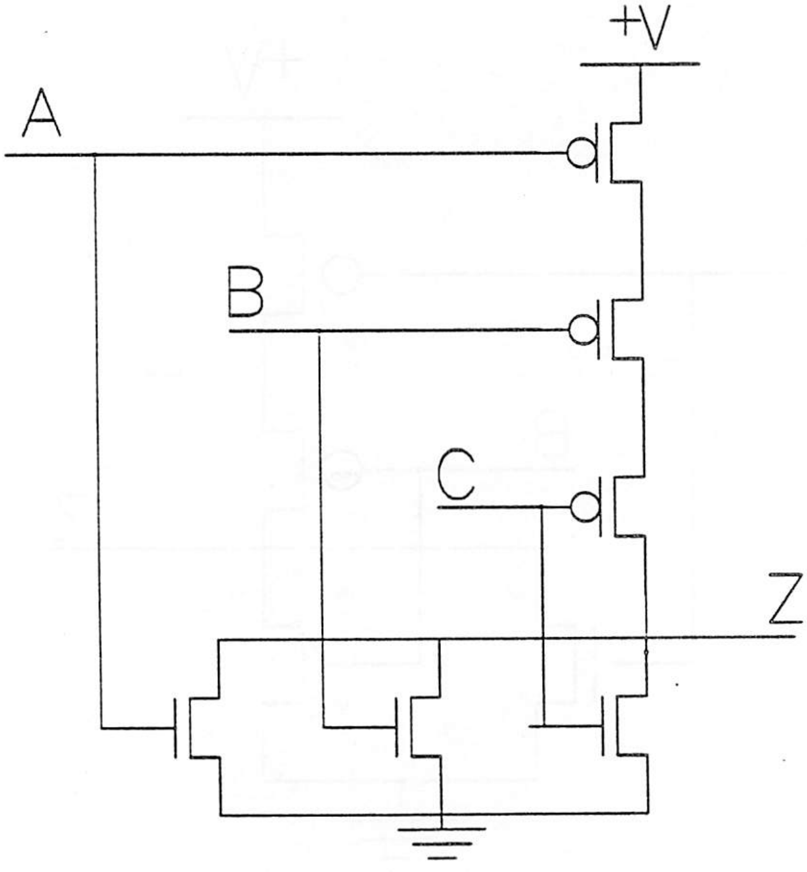
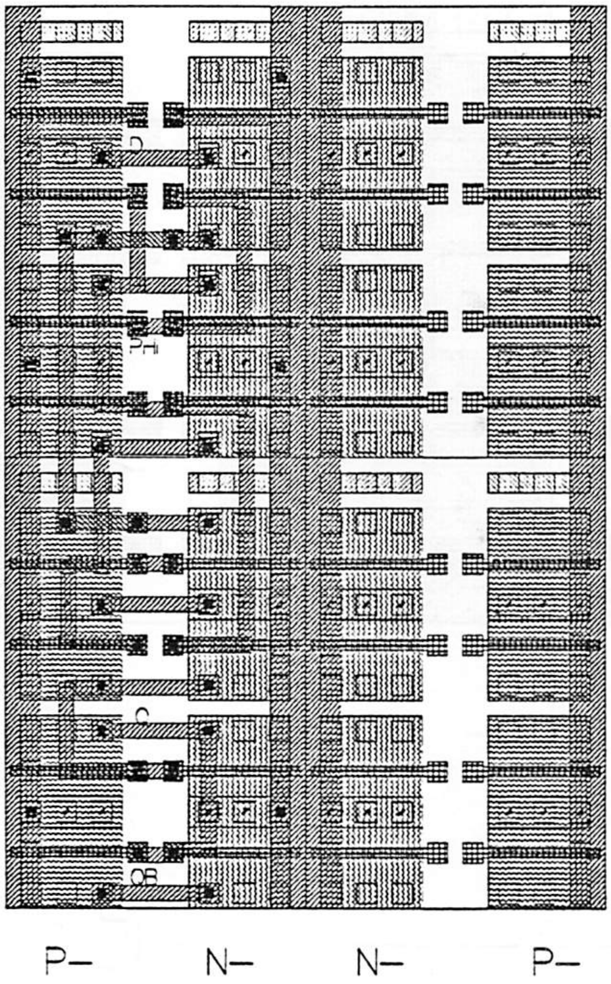


Figure D.17 nor3   z = !(a + b + c)



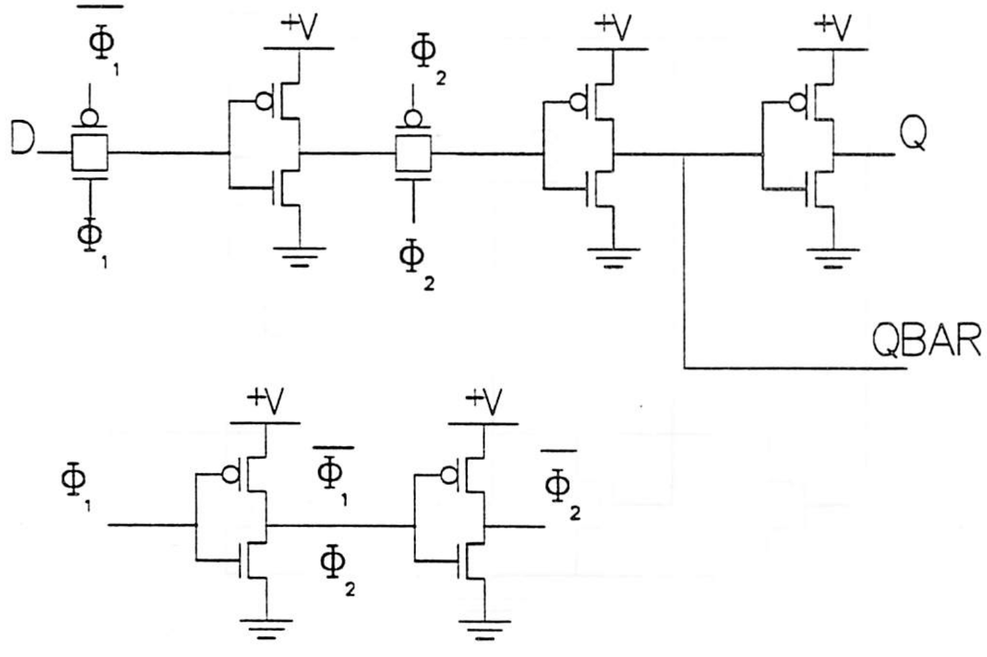
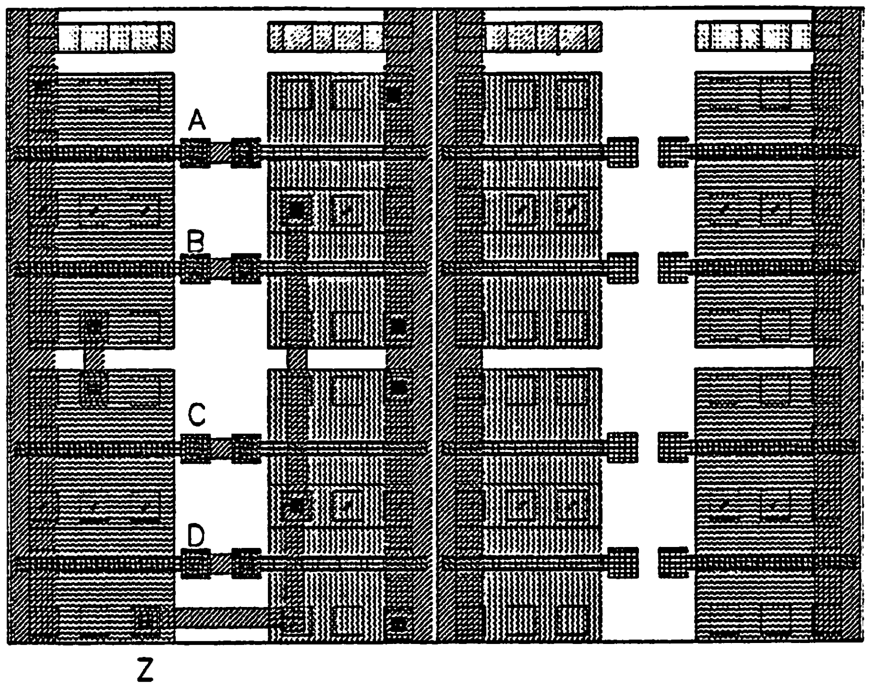


Figure D.18 dyndff



P– N– N– P–

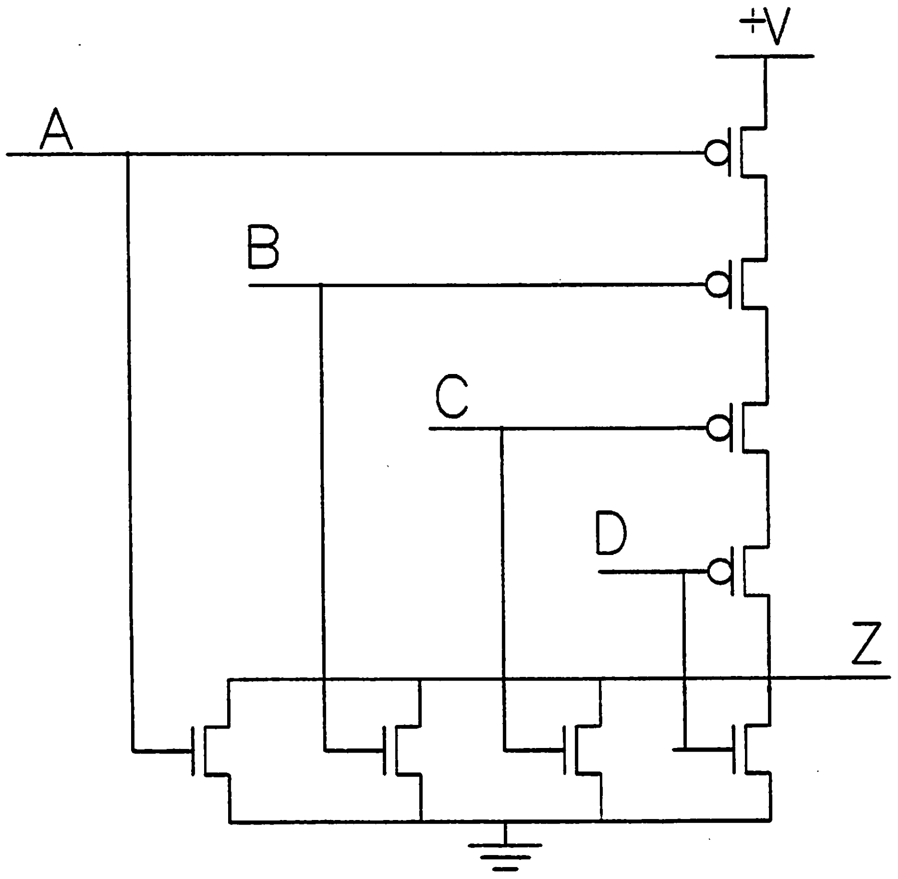
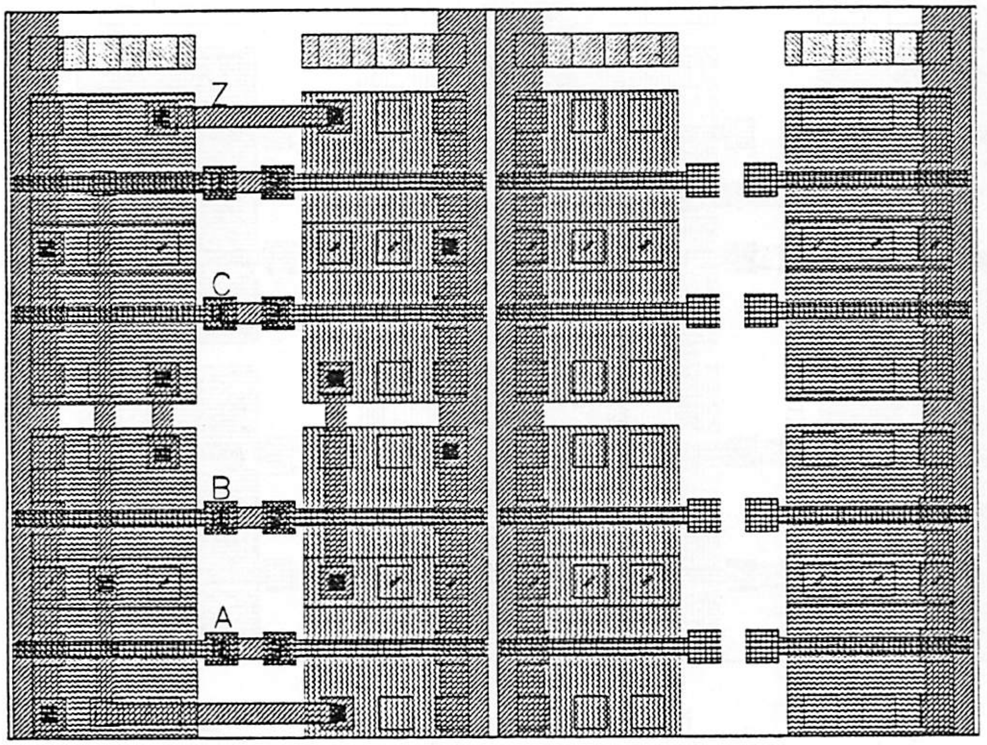


Figure D.19 nor4 $z = !(a + b + c + d)$



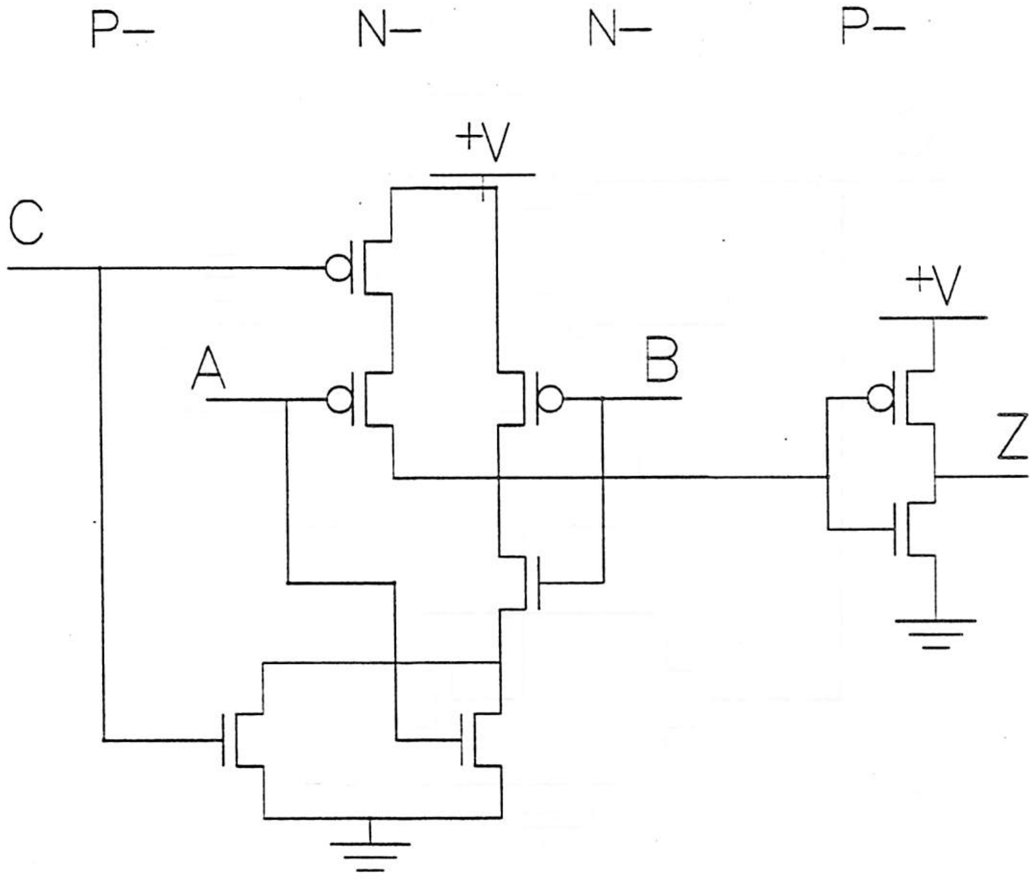
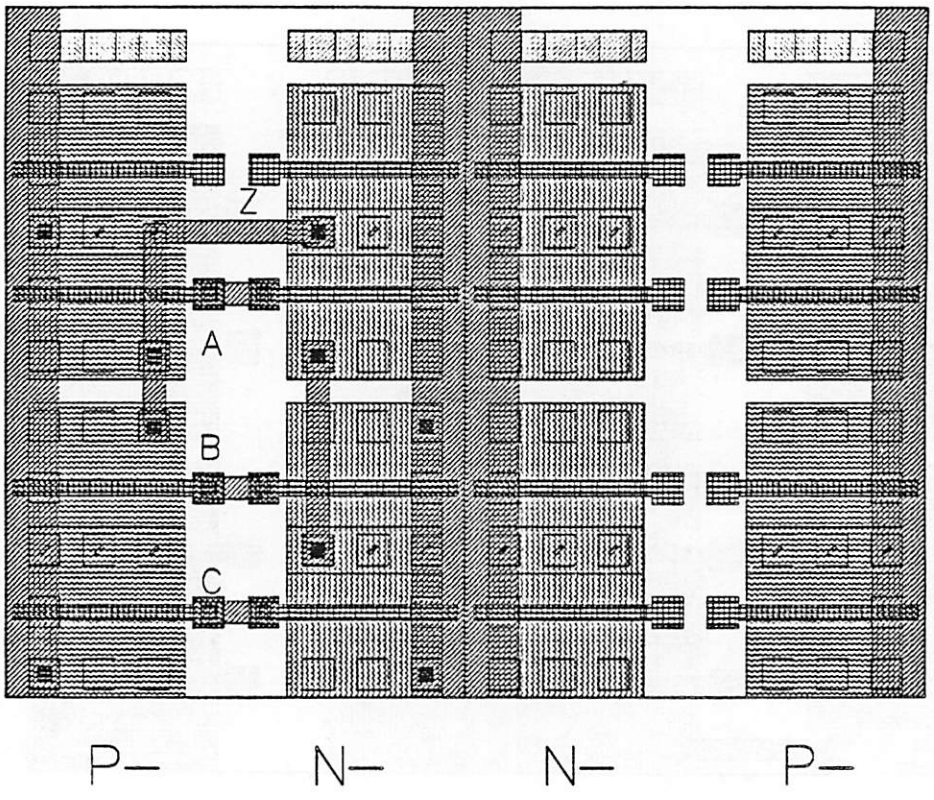


Figure D.20 oa21   z = (a (b + c))



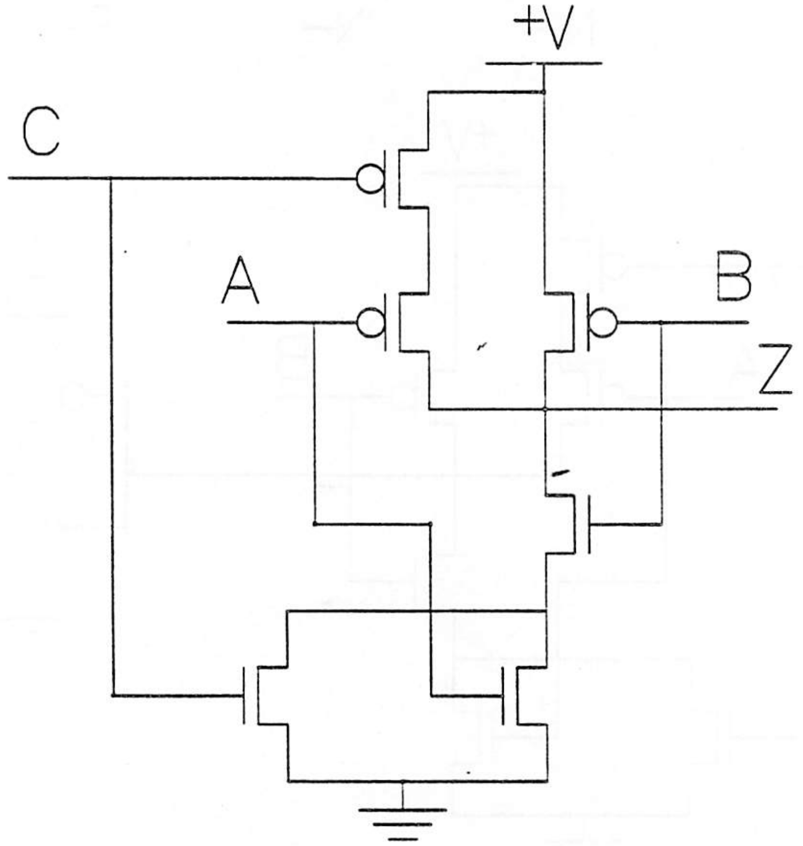
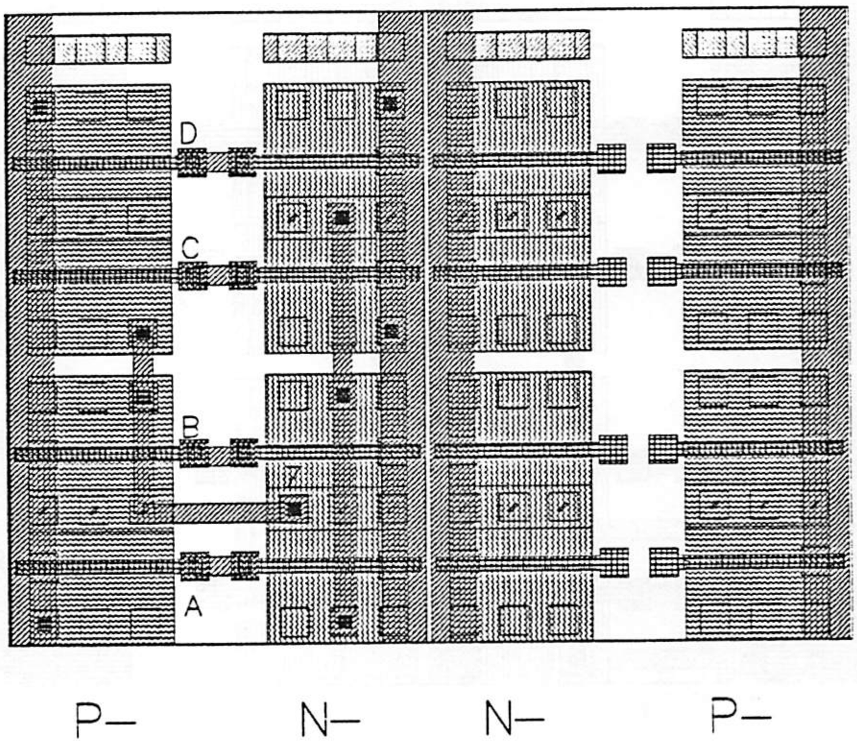


Figure D.21 oai21 z = !(a (b + c))



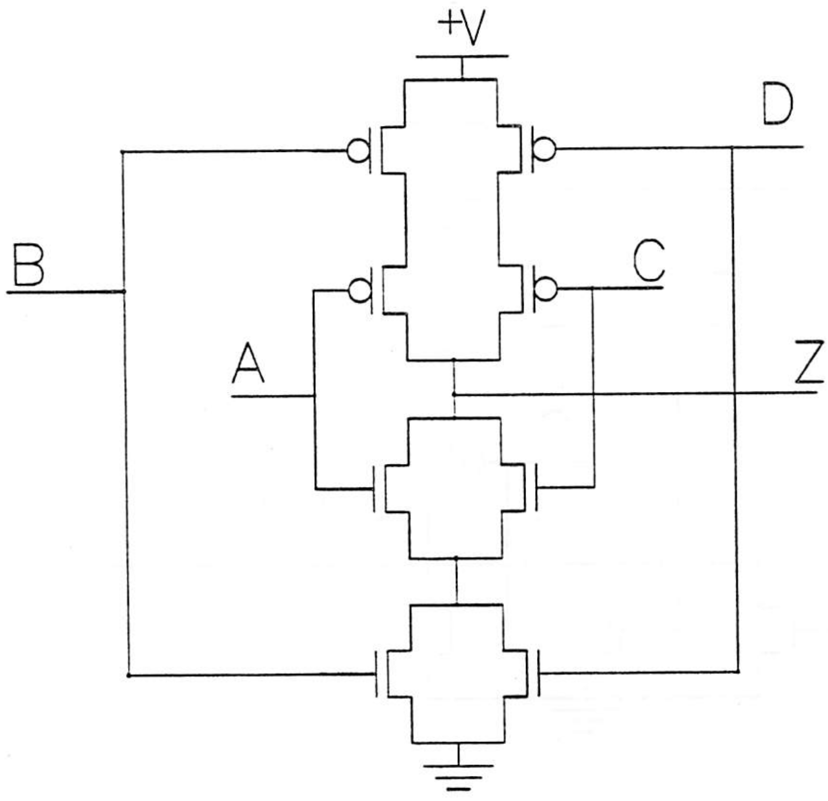
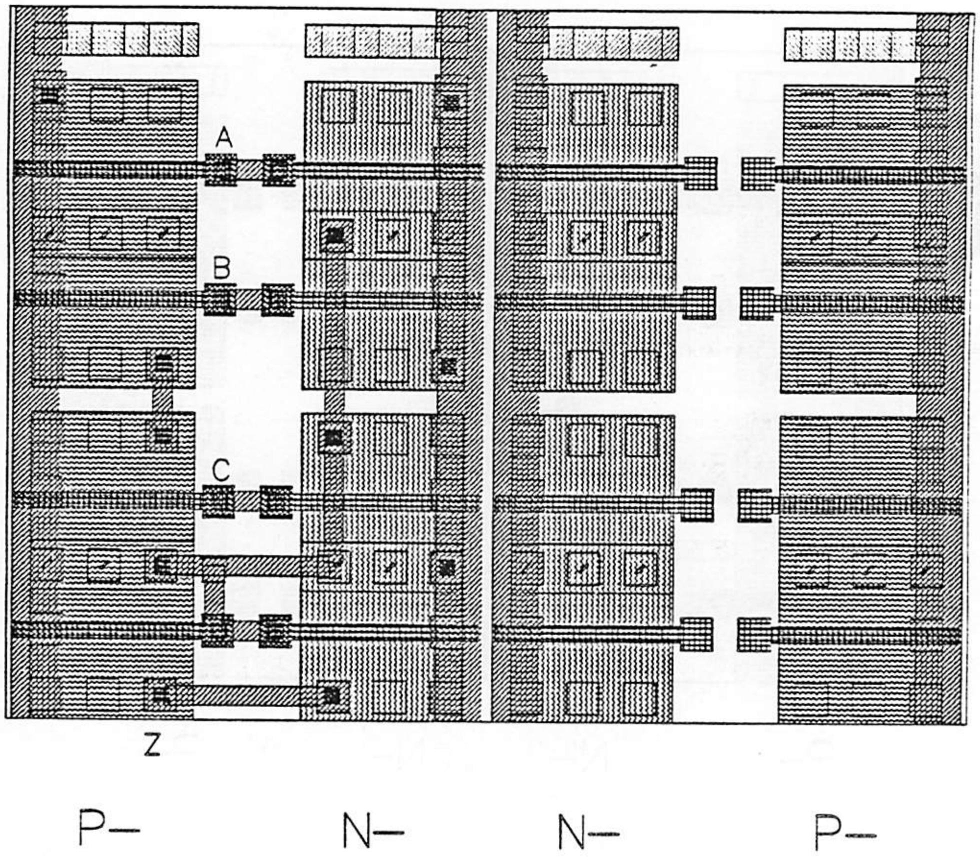


Figure D.22 oai22   z = !(a + b)(c + d)



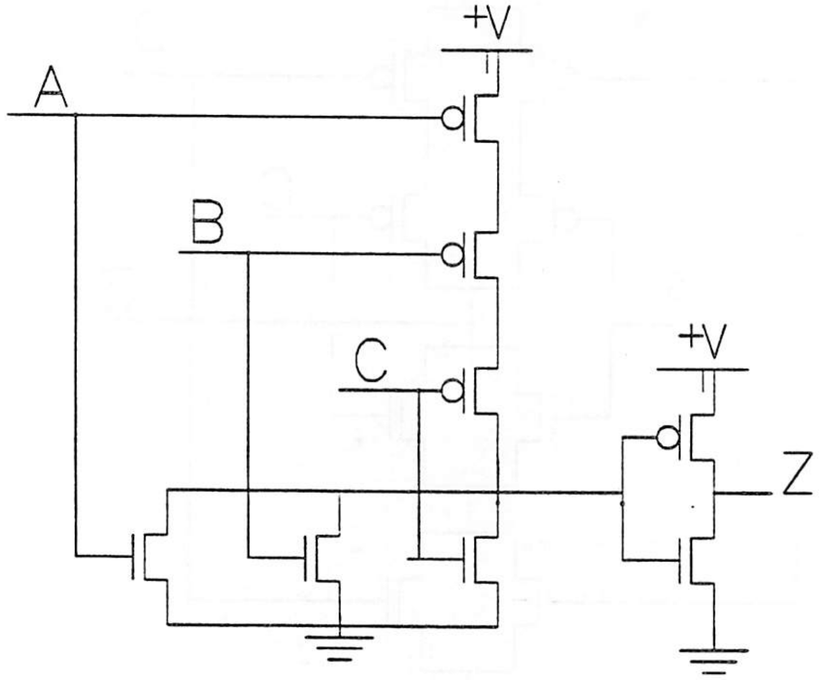
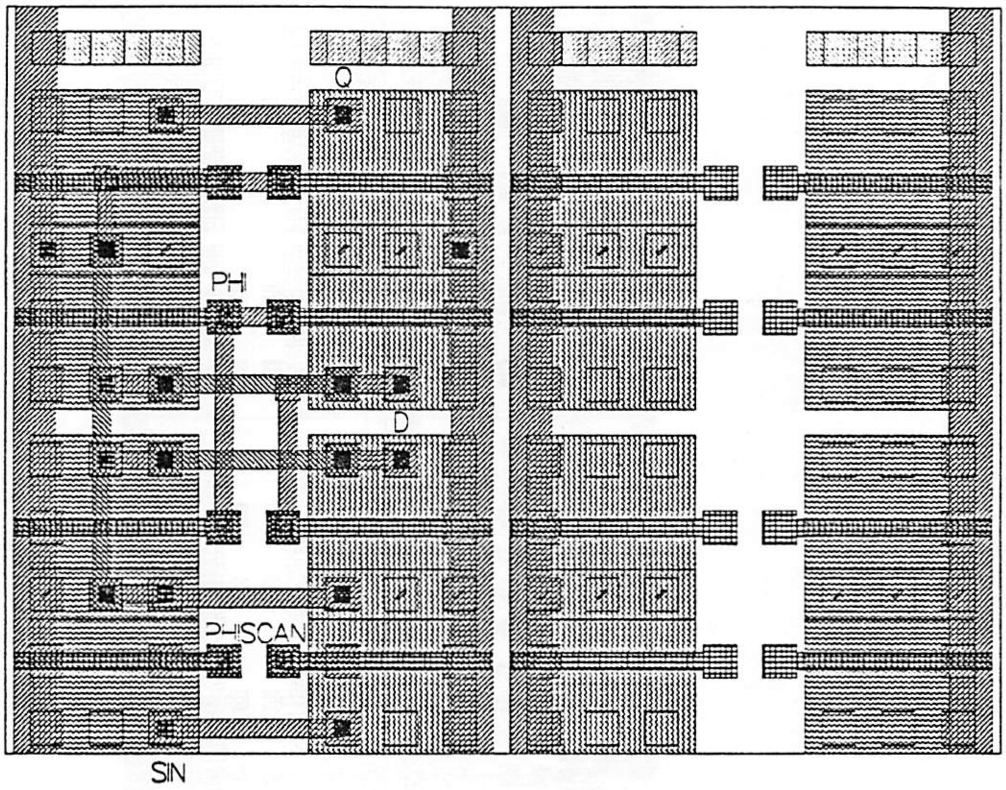


Figure D.23 or3 z = a + b + c



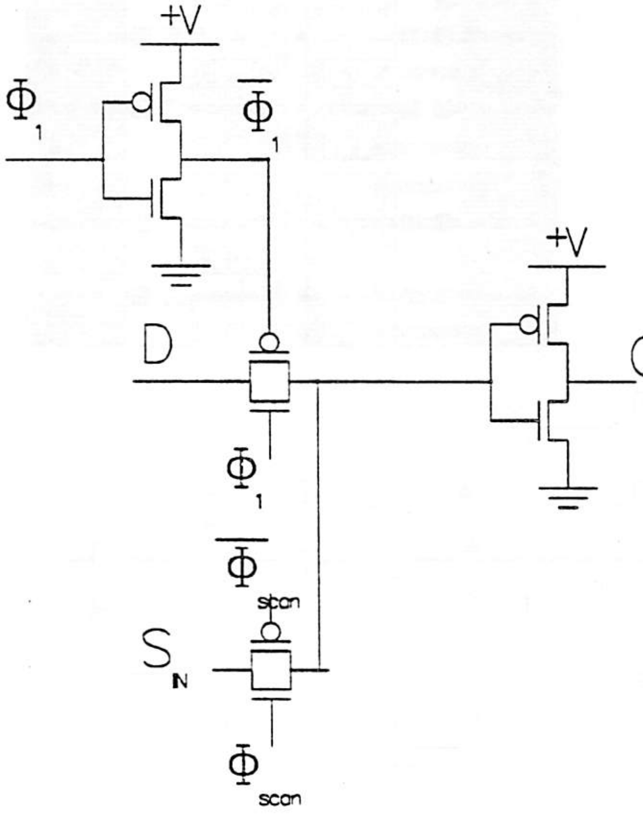


Figure D.24 dynscanlatch

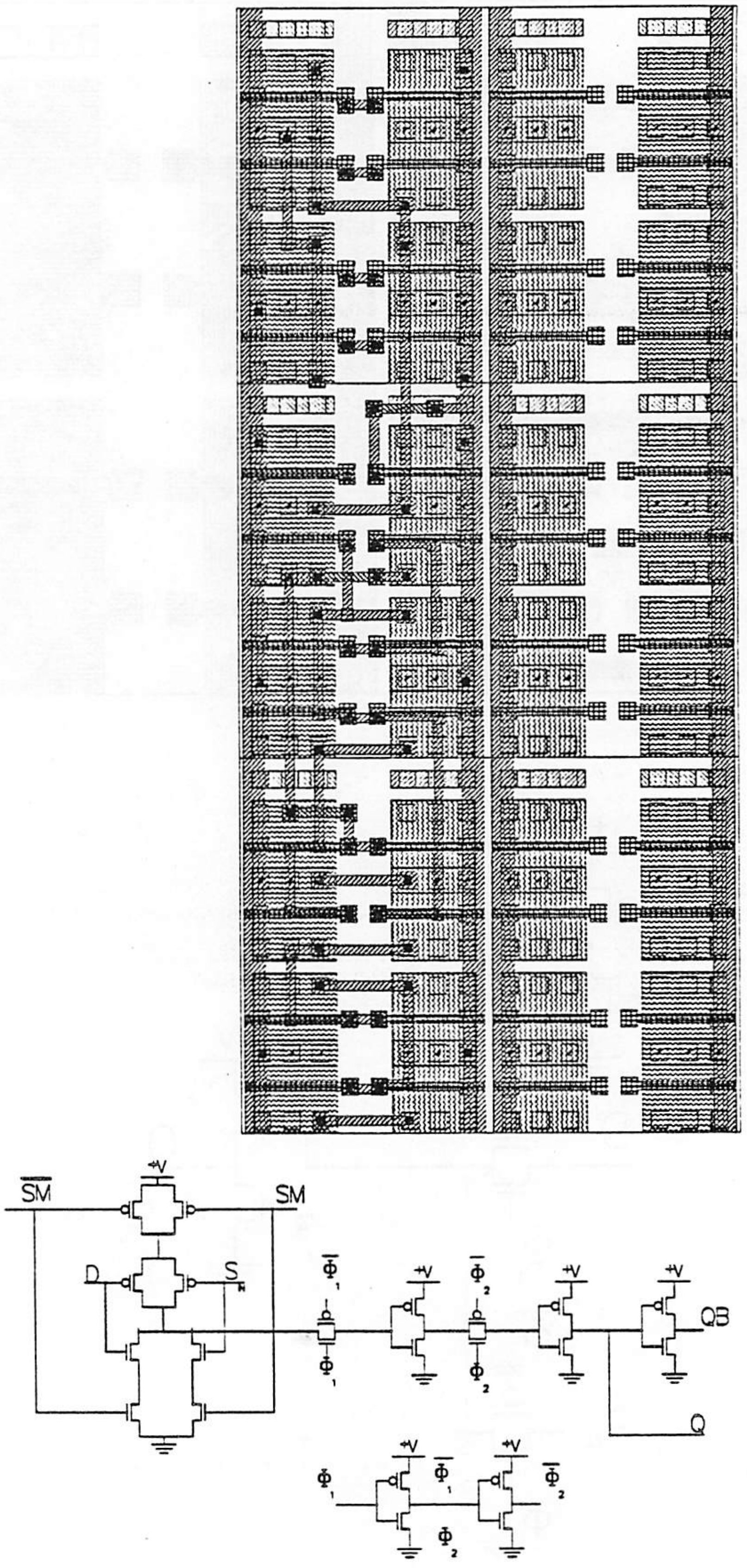
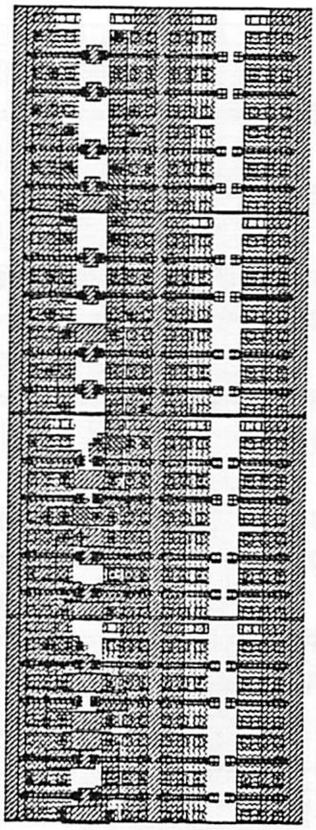


Figure D.25 dyntestdff



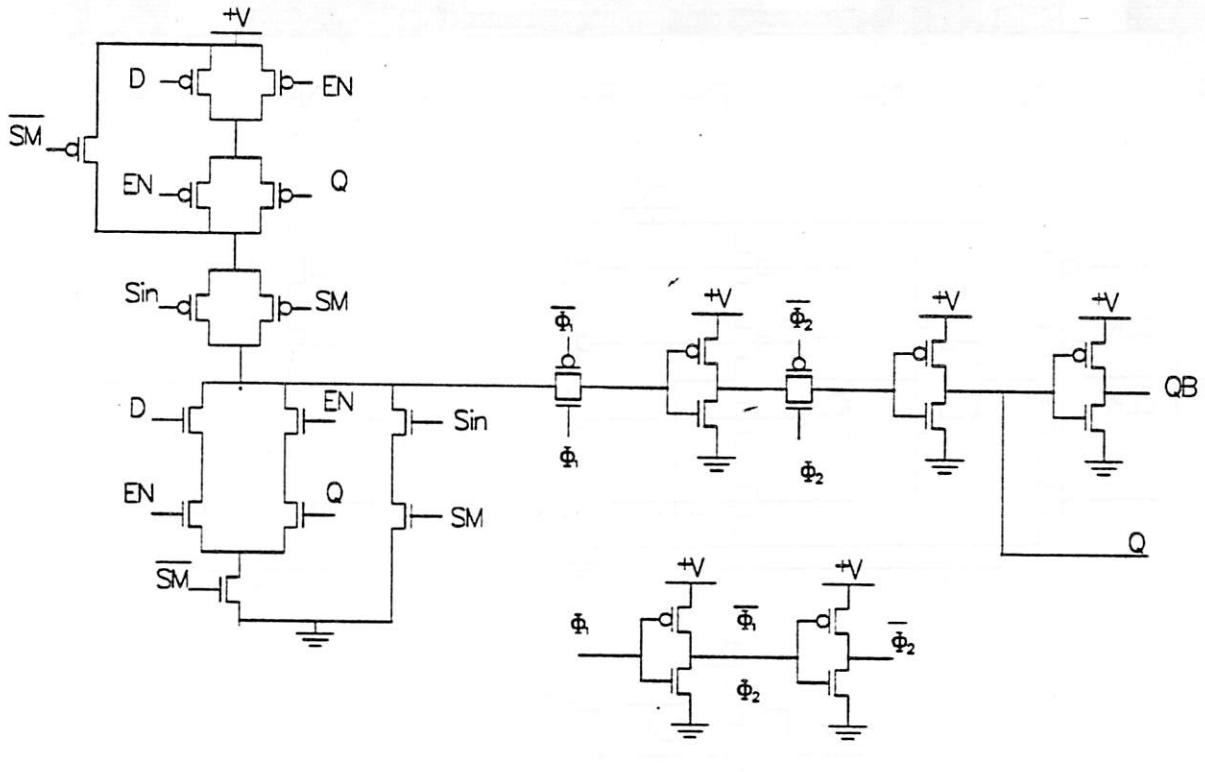
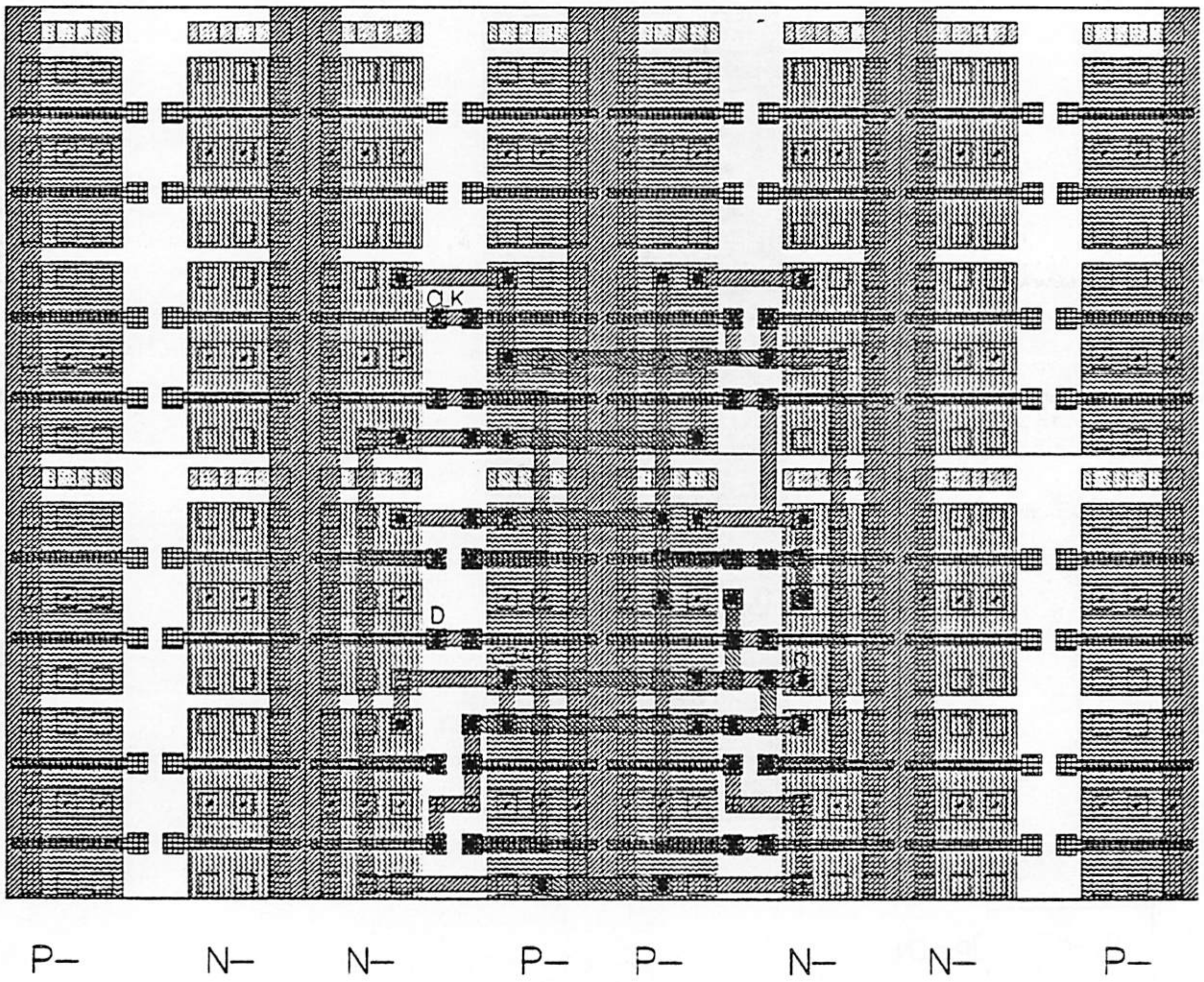
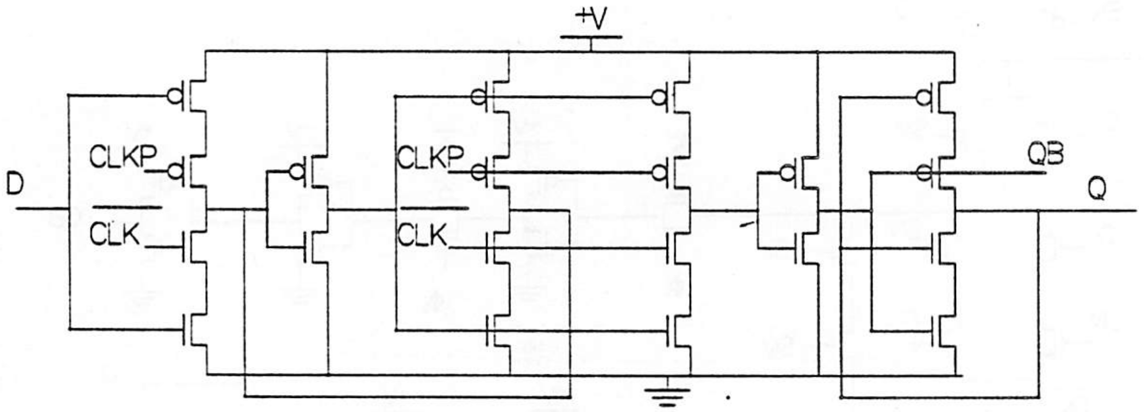


Figure D.26 dyntstendff





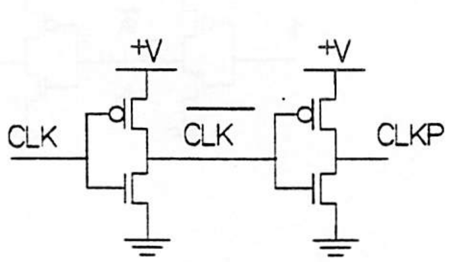
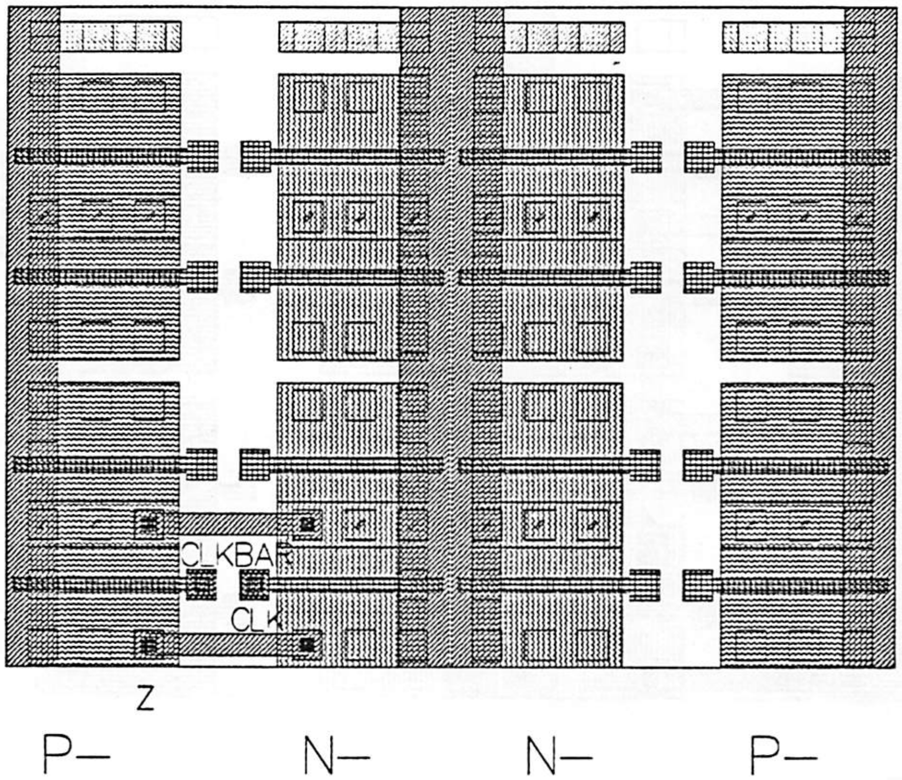


Figure D.27 staticdff



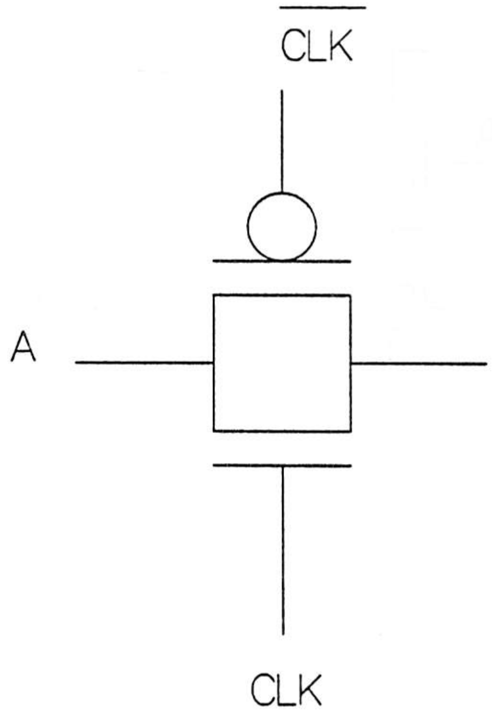
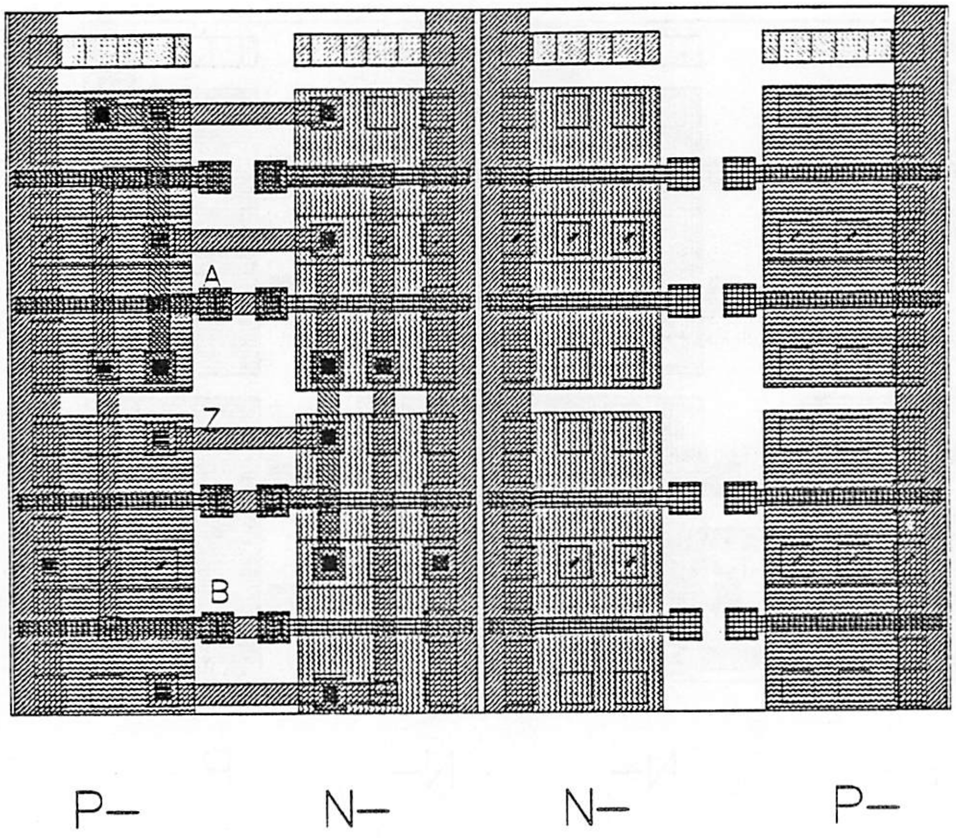


Figure D.28 xmsgate z = a



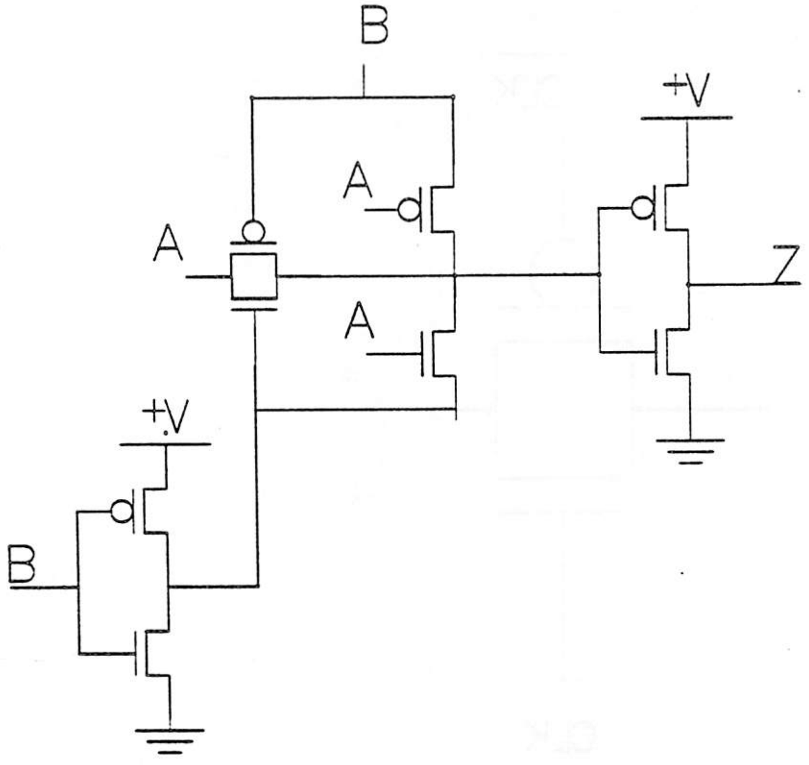
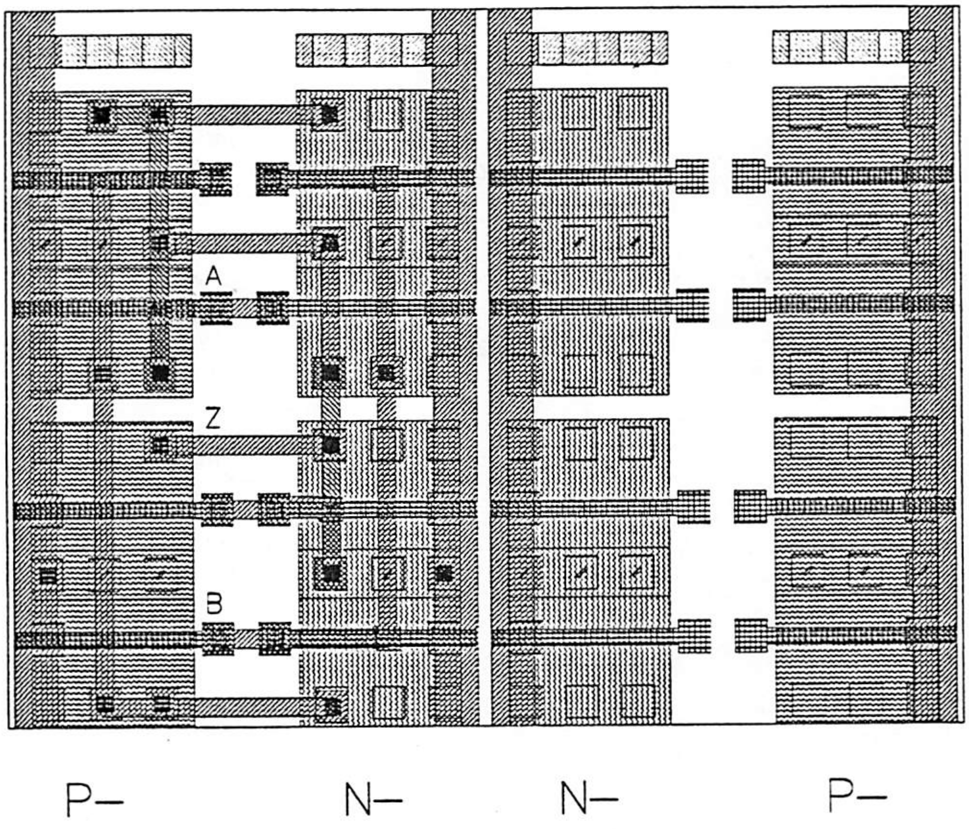


Figure D.29 xnor2 z = !(a!b + !ab)



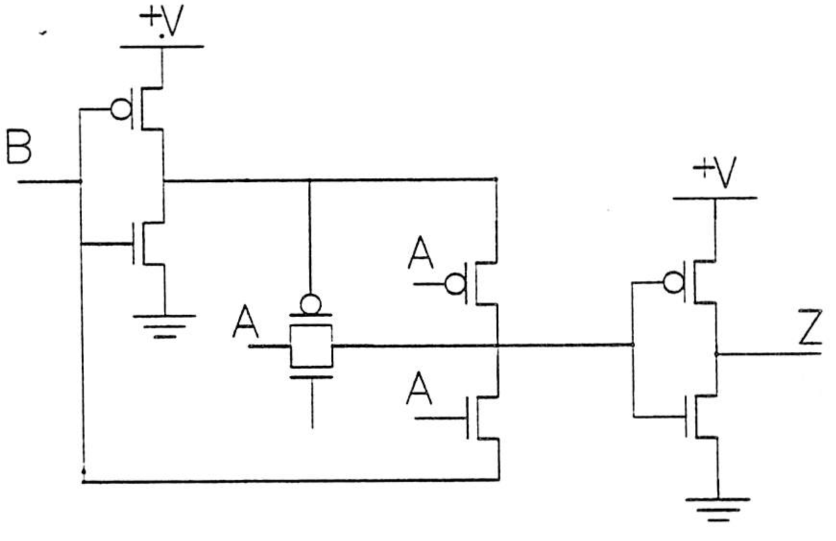


Figure D.30 xor2   z = (a!b + !ab)

Appendix E

Manual Pages

NAME

skipper – A sea-of-gates template generator program

SYNOPSIS

skipper

DESCRIPTION

Skipper is an interactive program which generates sea-of-gates templates for both the basic tiling unit (BTU), (the smallest repetitive unit that doesn't need to be rotated to create a full array), and/or a full chip size array with I/O pads, corner, and spacer pads. The architecture of the BTU has n-, n-, p-, p-type, (NNPP) transistors with power and ground buses that run over the diffusion of the transistors and are shared by adjacent columns.

In making the BTU template, skipper allows the user to decide the N and P transistor sizes, the number of metal1 tracks between power and ground, power and ground widths, and the number of transistor pairs between substrate contacts. The user is prompted for all variables over which he/she has control and for each is provided a reasonable default which either has been calculated from previously given information or is the current reasonable default.

HOW IT WORKS

First Skipper asks if you want to make a new template for the basic tiling unit, if the answer is yes it queries the user for needed information. As it asks questions about the template it also asks some things about the chip size template. This is done so that default parameters can be calculated as closely as possible to what will be needed for the chip size template. If the user doesn't want to make a new BTU he/she can give the name of an already existing one and skipper will use that to create a chip size template, warning if parameters such as power and ground widths are insufficient.

HOW TO RUN SKIPPER AND LOOK AT THE RESULTS

The best way to become familiar with what skipper does is to run it. First time users can easily become familiar with skipper by typing returns for all of the questions so that the template and chip produced will be the result of the default parameters. The default BTU template can be viewed using vem by opening the cell template:physical. The default chip size template is chip:unplaced.

SEE ALSO

vem(1), oct(3), admiral(1)

AUTHORS

Lorraine Layer, Wayne Christopher

BUGS

Send bug reports to "mariner@eros".

NAME

regen – re-create a Sea of Gates library for a new template

SYNOPSIS

regen [**–o** old_template] [**–n** new_template] [**–d** new_directory] cell:view ...

DESCRIPTION

Regen maps a cell library from one *template* (basic transistor structure) to another. The template that is currently used by the cells should be given as *old_template*, the one desired for the new cells should be *new_template*, and the directory that the new cells should be placed should be given as *new_directory*. *Regen* will then re-create each of the specified cells with the new template instead of the old one. If the old cell name was .../cellname:viewname, the new name will be newdir/cellname:viewname (i.e, the leading components of the pathname will be stripped off).

The cells and the templates should conform to the Sea of Gates symbolic policy (as well as the general *oct* symbolic policy). The program does the mapping by building up a symbolic grid for both templates by looking at the positions of the terminals on the template, with the assumption that any terminal will fall on grid lines in the X- and Y-direction, and any grid line will have at least one terminal falling on it. Then it maps the wiring in the old cell onto the grid for the old template and from there maps it to the grid for the new template. It then creates the new cell, makes the appropriate number of copies of the template, copies the important properties, and creates the wiring.

The templates must each have a property called "CENTER", which tells *regen* where the middle of the transistors are. If the new template has more tracks on the outside of the transistors, the grid numbers of the locations near the center should stay fixed.

RESTRICTIONS

The old template and the new template must have the same number of transistors per block and blocks per plug, and must have power and ground routed in the same layer. Otherwise the topology of the cell would be different and the automatic mapping wouldn't make sense.

AUTHOR

Wayne Christopher (faustus@ic.Berkeley.EDU)

SEE ALSO

/cad/doc/mariner/*, ckplace(1), stitch(1)

BUGS

REFERENCES

[1] J. W. Jones, "Array Logic Macros," *IBM J. Res. Develop.*, pp.98-109, March 1975.

[2] D. Johansen, "Bristle Blocks: A Silicon Compiler," *Proc. 16th Design Automation Conference*, pp.310-313, June 1979.

[3] J.B.Brinton, "CHAS seeks title of global CAD system," February 10, 1981, pp.100-102.

[4] G.B.Goates, "ABLE: A LISP-based layout modeling language with user-definable procedural models for Storage/Logic Array design," M.S. Report, University of Utah, December 1980.

[5] T. Chan et al., "Advanced Structured Arrays Combine High Density Memories With Channel-free Logic Array," proceedings of the Custom Integrated Circuits Conference, pp. 39-43, May 1987.

[6] W.R.Heller, W.F.Mikhail, and W.E.Donath, "Prediction of wiring space requirement s for LSI," Journal of Design Automation and Fault Tolerant Computing, vol.2 , 1978, pp.117-144. A.A.El Gamal, "Two-dimensional estimation model for interconnections in master-slice integrated circuits," *IEEE Trans. Circ. and Syst.*, Vol.CAS-28 , No.2, February 1981, pp.127-137.

[8] A. Hui et al., "A 4.1K Gates Double Metal HCMOS Sea Of Gates Array," proceedings of the Custom Integrated Circuits Conference, pp. 15-17, May 1985.

[9] C. P. Hsu, "Automatic Layout Of Channelless Gate Array," proceedings of the Custom Integrated Circuits Conference, pp. 281-284, May 1986.

[10] M. Kawashima et al., "An 18K 1um CMOS Gate Array With High Testability Structure," proceedings of the Custom Integrated Circuits Conference, pp. 52-55, May 1987.

[11] H. Kubosawa et al., "Layout Approaches To High-Density Channelles Masterslice," proceedings of the Custom Integrated Circuits Conference, pp. 48-51, May 1987.

[12] F. Anderson and J. Ford, "A 0.5 Micron 150K Channelless Gate Array," proceedings of the Custom Integrated Circuits Conference, pp. 35-38, May 1987.

[13] P. Birzele and P. Michel, "Reduced Design and Manufacturing Time by New Semicustom Standards and VLSI-Design Methodologies," *Siemens Internal Document* Mar. 1987.

[14] M. Beunder, J. Kernhof, and B. Hoefflinger, "Effective Implementation Of Complex and Dynamic CMOS Logic In A Gate Forest Environment," proceedings of the Custom Integrated Circuits Conference, pp. 44-47, May 1987.

[15] T. Wong et al., "A High Performance 129K Gate CMOS Array," proceedings of the Custom Integrated Circuits Conference, pp. 568-571, May 1986.

[16] S. Atiq et al., "Channelles Architecture: A New Approach For CMOS Standard Cell Design," proceedings of the Custom Integrated Circuits Conference, pp. 12-14, May 1985.

[17] R. Lipp, "Advanced Architecture (Channel-less) Dual Layer Metal CMOS Gate Array," proceedings of the Custom Integrated Circuits Conference, pp. 71-73, May 1983.

[18] M. R. Chin, C. P. Hsu and K. Y. Liao,, "Triple-Level Metal Gate Array Using Channelless Architecture," proceedings of the Custom Integrated Circuits

Conference, pp. 725-728, May 1987.

[19] T. Laidig, *TAP Users Manual*, U. C. Berkeley Internal Report, Feb. 1986.

[20] P. Moore, *The General Structure Of OCT*, U. C. Berkeley Internal Report, Feb. 1986.

[21] R. Brayton etal., "Multiple Level Logic Optimization System." *Digest of Technical Papers ICCAD-86*, Santa Clara, pp.356-359, Nov. 1986.

[22] D. S. Harrison, P. Moore, R. L. Spickelmier and A. R. Newton, "Data Management And Graphics Editing in The Berkeley Design Environment," *Digest of Technical Papers ICCAD-86*, Santa Clara, pp. 24-27, Nov. 1986.

[23] E. E. Hollis, *Design of VLSI Gate Array ICs*, Prentice Hall, 1987.

[24] D. A. Hodges and H. G. Jackson, *Analysis And Design of Digital Integrated Circuits*, Mcgraw-Hill, 1983.