This is to certify that the

thesis entitled

Performance-Design Tradeoff of

Hierarchical VLSI Design Entry Points

presented by

Man-Kuan Vai

has been accepted towards fulfillment
of the requirements for

Masters degree in Elect. Engr.

Michael A. Shanblatt
Major professor

Date 5-8-85

Michael A. Shanblatt

PERFORMANCE-DESIGN TRADEOFF

OF

HIERARCHICAL VLSI DESIGN ENTRY POINTS

By

Man-Kuan Vai

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Department of Electrical Engineering and
Systems Science

1985

ABSTRACT

PERFORMANCE-DESIGN TRADEOFF

OF

HIERARCHICAL VLSI DESIGN ENTRY POINTS

By

Man-Kuan Vai

This research relates to VLSI design methodology, and especially to the performance versus design task tradeoff of specifying functionally identical circuits at various levels. The layouts of two examples, a ripple-carry adder and a Braun array multiplier, are designed with the assistance of a CAE system. The design entry points of transistor and gate level are considered.

The two case study circuits are designed independently at both levels and are evaluated with respect to their performance and design complexity. The comparative results indicate that better performance can be achieved by starting a design from the transistor level. However, the design complexity of the circuit is found to be lower in a gate level design.

To my parents and wife

Mr. and Mrs. Man-Kit Vai and Jin-Yu

ACKNOWLEDGEMENTS

I wish to acknowledge and thank my major advisor, Dr. Michael A. Shanblatt, who introduced me to this fascinating field and gave me numerous guidance and encouragement in the course of this research.

I also wish to thank the committee members, Dr. D. K. Reinhard and Dr. E. D. Goodman, for their valuable suggestions and comments in this work.

Finally, I owe many thanks to Jin-Yu for her emotional support.

TABLE OF CONTENTS

		Page
LIST OF TABLES		vi
LIST OF FIGURES		vii
I.	INTRODUCTION	1
	1.1 Problem Statement	3
	1.2 Approach	8
II.	BACKGROUND	9
	2.1 One-Bit Full Adder	9
	2.2 Ripple-Carry Adder	11
	2.3 Array Multiplier	13
	2.4 Implementation Technology	14
	2.5 Design Rules	17
	2.6 Delay Time Model	18
	2.7 Computer-Aided-Engineering of VLSI Circuits	19
III.	DEVELOPMENT OF DESIGN EXAMPLES	21
	3.1 Selection of Circuits	21
	3.2 Gate Library	23
	3.3 Full Adder Design	23
	3.3.1 Transistor Level Design	27
	3.3.2 Gate Level Design	29
	3.4 Ripple-Carry Adder	29
	3.5 Braun Array Multiplier	31
IV.	DESIGN EVALUATION	59
	4.1 Criteria of Evaluation	59
	4.2 Device Area Calculation	60
	4.3 Propagation Delay Calculation	62
	4.3.1 Delay Time Model	65
	4.3.2 Load Capacitance Estimation	69
	4.3.3 Results	73
	4.4 Time-Area Complexity	75
	4.5 Design Complexity	76
	4.6 Comparison	78

V. CONCLUSION 81
5.1 Summary 81
5.2 Contributions 85
5.3 Future Development 86

BIBLIOGRAPHY 88

LIST OF TABLES

Table		Page
3.1	Cell list for transistor level multiplier.	54
3.2	Cell list for gate level multiplier.	55
4.1	The dimensional data of the full adders.	61
4.2	The areas of ripple-carry adders.	61
4.3	The dimensional data of the cells for braun array multiplier.	63
4.4	The areas of Braun array multiplier.	64
4.5	The dimensional data of the transistors.	66
4.6	The typical physical parameters of the designs.	67
4.7	The delay times of the logic gates.	70
4.8	The effective capacitances of the gate inputs.	73
4.9	The propagation delays of ripple-carry adders.	74
4.10	The propagation delays of Braun array multipliers.	76
4.11	The time-area complexities of the design examples.	76
4.12	The design complexities of the design examples.	78

LIST OF FIGURES

Figure		Page
1.1	The procedure of integrated circuit design.	6
2.1	Logic gate diagram of a one-bit full adder with hard-wired logic.	10
2.2	Logic gate diagram of a one-bit full adder using only NOR gates.	12
2.3	Block diagram showing the construction of a ripple-carry adder.	13
2.4	Block diagram of a Braun array multiplier.	15
2.5	Schematic diagrams of basic NMOS logic gates.	16
3.1	The layout of an inverter cell.	24
3.2	The layout of a two-input NAND cell.	24
3.3	The layout of a two-input NOR cell.	25
3.4	The layout of a horizontal three-input NOR cell.	25
3.5	The layout of a vertical three-input NOR cell.	26
3.6	The layout of a four-input NOR cell.	26
3.7	The layout of a transistor level full adder cell.	28

Figure		Page
3.8	The layout of a gate level full adder cell.	30
3.9	The layout of a transistor level 4-bit ripple-carry adder.	32
3.10	The layout of a gate level 4-bit ripple-carry adder.	33
3.11	The rearranged block diagram of a Braun array multiplier.	35
3.12	The layout of cell 1 for multiplier design.	36
3.13	The layout of cell 2 for multiplier design.	37
3.14	The layout of cell 3 for multiplier design.	38
3.15	The layout of cell 4 for multiplier design.	39
3.16	The layout of cell 4A for multiplier design.	40
3.17	The layout of cell 4B for multiplier design.	41
3.18	The layout of cell 5 for multiplier design.	42
3.19	The layout of cell 5A for multiplier design.	43
3.20	The layout of cell 6 for multiplier design.	44
3.21	The layout of cell 7 for multiplier design.	45
3.22	The layout of cell 8 for multiplier design.	46
3.23	The layout of cell 9 for multiplier design.	47

Figure		Page
3.24	The layout of cell 10 for multiplier design.	48
3.25	The layout of cell 10A for multiplier design.	49
3.26	The layout of cell 10B for multiplier design.	50
3.27	The layout of cell 11 for multiplier design.	51
3.28	The layout of cell 11A for multiplier design.	52
3.29	Tesselation map for transistor level multiplier.	53
3.30	Tesselation map for gate level multiplier.	56
3.31	The layout of transistor level 5-by-5 Braun array multiplier.	57
3.32	The layout of gate level 5-by-5 Braun array multiplier.	58

CHAPTER I

INTRODUCTION

High speed VLSI (Very Large Scale Integration) has created a new challenge for circuit designers. Numerous algorithms and architectures have been proposed to take advantage of VLSI capabilities [1,2,3]. New design concepts, vastly different from those used in conventional design, have been and continue to be developed in order to facilitate efficient, manageable design.

Circuits and systems once requiring many individual chips can now be built on a single chip with VLSI technology. Various problems of reliability and performance that unavoidably arise from combining many discrete components have been eliminated or reduced. But the complicated process of VLSI design has introduced a new set of reliability and performance problems which are harder to visualize and more challenging to solve.

Due to the complexity of a VLSI design, not all algorithms and architectures are suitable for VLSI implementation. An architecture eligible for VLSI implementation must foremost possess a certain degree of design regularity. This is also called device modularity and relates to the capacity for device tessellation. A complicated VLSI design can be simplified into the design of several building blocks by taking advantage of its regularity. This is the "divide-and-conquer" philosophy of VLSI design.

Gate count was conventionally used in discrete component designs for the purpose of evaluation of the design complexity. However, the chip area and delay time represent a better measure of cost efficiency in VLSI designs. In fact, logic gates are cheap in a VLSI design. It is the interconnection or communication requirements which mainly contribute to the performance and cost effectiveness of a design. Interconnections not only use chip area but also play an important role in propagation delay.

It can be concluded from the above considerations that a good algorithm or architecture for VLSI implementation must possess the following properties [1]:

1. The architecture should be implemented by only a few different types of simple cells.

2. The architecture's data and control flow should be simple and regular, ideally connecting only nearest neighbors.

3. The architecture should use extensive pipelining and multiprocessing.

Many circuits meet with these requirements. A few examples include random access memory (RAM), read only memory (ROM), programmable logic array (PLA) and, of interest in this work, the systolic array.

Architectures which meet the above-mentioned properties tend to have a reduced design cost. Only a few simple cells have to be designed, and the cells on the chip are merely copies of these few basic ones. Regular interconnections imply that modularity and extensibility are achievable, so that a large chip can be formed by a tessellation of the basic cells. The characteristic of pipelining and multiprocessing means that a special-purpose chip can be implemented simply by including many identical cells on the chip in either a vertically parallel (pipeline) or horizontally parallel (multiprocessor) configuration. Ideally, a combination of both can be used. Finally, the regularity or modularity of an architecture enables a hierarchical design technique to be applied.

1.1 Problem Statement

The conglomerate process of integrated circuit design is traditionally regarded as a difficult field because of the necessity of expertise relating to solid state physics. But, the recent development of simplified design rules and computerized support tools provides a method for designers to experiment with circuit options without concern for the underlying physical phenomena. Naive VLSI designers can be successful after a minimum amount of practice with these new design tools.

Unfortunately, one of the drawbacks in VLSI technology is the high design cost. A designer must be able to use an efficient approach aimed at producing valid, working chip layouts at a reasonable cost in both time and dollars. At present, design costs dominate the whole cost of VLSI manufacture, and this trend will continue into the foreseeable future.

Design cost is directly related to the so-called turn-around time which is the time interval from the receipt of a device specification to final manufacturing output. Therefore, design time, which is a major contributing factor to turn-around time, plays an important role in design cost.

Two examples are provided here in order to demonstrate the seriousness of the problem of design time [4]. It is reported that the M68000, which is a 16-bit microprocessor by Motorola, required 52 man-years of design effort. Another popular 16-bit microprocessor, the Intel 8086, required 13 man-years of effort merely for layout. Obviously, few custom applications can support time-costs of this magnitude.

One reasonable way to reduce the design cost is by mass production so that the initial development cost is shared by many customers. This strategy is effective for general-purpose chips, especially in the area of standard SSI or MSI chips. However, other approaches must be used to keep the design cost low since many VLSI designs are for special-purpose applications and will not be made in quantities large enough to significantly reduce the design cost per chip.

Hierarchical design is one of the approaches that can be used to reduce the design cost. As mentioned above, the design task of a VLSI chip can be greatly simplified by architectural regularity which allows the entire chip to be constructed merely by tessellating some basic building blocks. This concept can be generalized to a lower level. No matter how complicated a VLSI circuit may be, the elements involved at the most bottom level of design are primarily only different types of transistors. Transistors can be connected in a hierarchical design to generate logic gates, such as NAND's, NOR's, etc., which constitute the basic elements at the next higher design level. The logic gates can be arranged to form processing elements or cells, such as full adders and shift registers. The processing elements can then be tessellated to implement the desired algorithm.

Elements or cells for various functions can be stored in a library from which they are accessed for later designs. For example, the layout of several general-purpose logic gates can be predesigned, evaluated, and stored in the library of a CAE system. A number of the stored logic gates can be recalled, placed, and routed, to form the function and interconnections required to implement a desired cell. The same idea also applies to the construction of a larger device by means of cells. A design can thus be started at the different levels of transistors, gates, cells, or more complex predefined elements. This design procedure is illustrated by a flow chart in Figure 1.1.

The designer has to determine the entry point of the design procedure after the architecture is defined. It is assumed that a

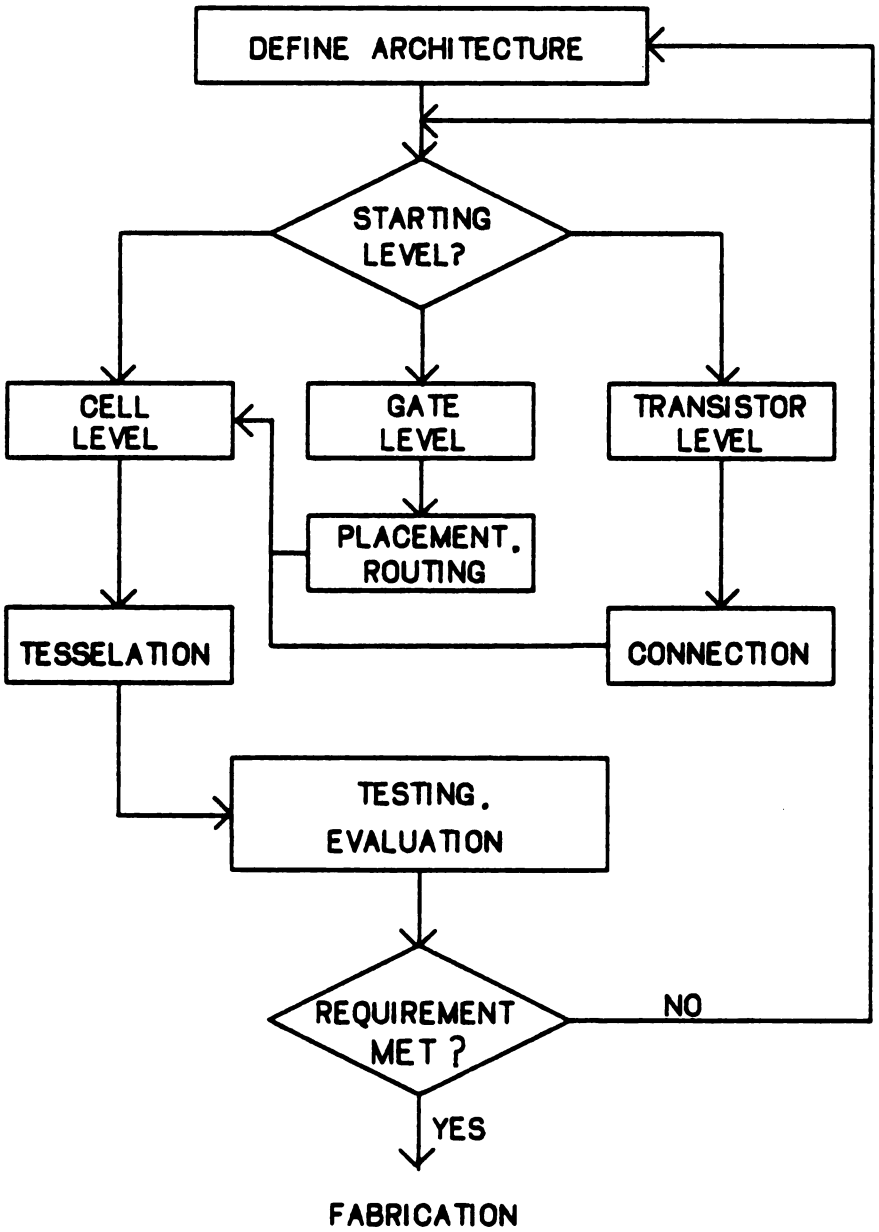

Figure 1.1 The procedure of integrated circuit design.

variety of gates or cells are predesigned and available for this procedure. The steps in the design procedure are highly interrelated. It can also be seen from Figure 1.1 that VLSI design is an iterative procedure.

The entry point in the design procedure will determine the performance and design cost of a chip. Generally speaking, the lower the level of the initial design, the better the performance of the final product in terms of a time-area parameter. But at the lower levels the design time increases. This tradeoff is reasonable since at a lower level the design has more flexibility and the possibility of obtaining a more optimal design is much higher. This advantage is obtained at the cost of more design work, which in turn makes the design time longer. In contrast, basic elements can be obtained from the library if the design is started at a higher level. However, the library elements at any level may not be best suited for a specific design specification since they were prepared without the knowledge of future tailored requirements. The performance of the final design may thus be affected.

A design may be started at the gate level or higher if the chip must be produced in a relatively short time and the performance is not a critical requirement. In contrast, the transistor level may be the required starting level if the performance is crucial and there is no rush for completion. Therefore, in addition to the chip performance, the time allowance for completion of a chip design must be also considered before the designer can make the decision as to the specific design level entry point. This decision is not always trivial due to

the complicated relationships in the time-area complexity resulting from designs started at different levels. Therefore, it is desirable to know, a priori, imformation about the performance-cost tradeoff of starting a design at different levels.

1.2 Approach

The goal of this research is to investigate the performance-cost tradeoff of different design starting levels. For this purpose, functionally identical circuits are designed at different levels to study their relative performance and design complexity. Two circuits, a ripple-carry adder and an array multiplier, are selected as working examples. These circuits are designed from both the transistor and gate levels.

The detailed layouts of both working examples are obtained with the aid of a CAE system so that a realistic evaluation can be done on the designs. Measures of time-area complexity and the design complexity are formulated for the purpose of comparison. The results of this research are intended to contribute to the development of a unified design methodology for VLSI.

CHAPTER II

BACKGROUND

The function and structure of a one-bit full adder, which is the basic building block for ripple-carry adders and array multipliers, are provided in this chapter. Then, the principles of ripple-carry adders and array multipliers are explained. Finally, the rules and computerized tools for VLSI design are described.

2.1 One-Bit Full Adder

The full adder is a basic functional unit which is used in many arithmetic devices. The operation of a full adder can be described in Boolean form as

$$S_i = A_i \oplus B_i \oplus C_i, \tag{2-1}$$

and

$$C_{i+1} = A_i B_i + B_i C_i + C_i A_i, \tag{2-2}$$

where A_i and B_i are inputs to the current stage and C_i is the carry from the previous stage.

This set of Boolean equations can be manipulated into many logically equivalent circuits [5]. The hardwired logic gate implementation of a full adder is shown in Figure 2.1 [6]. This design,

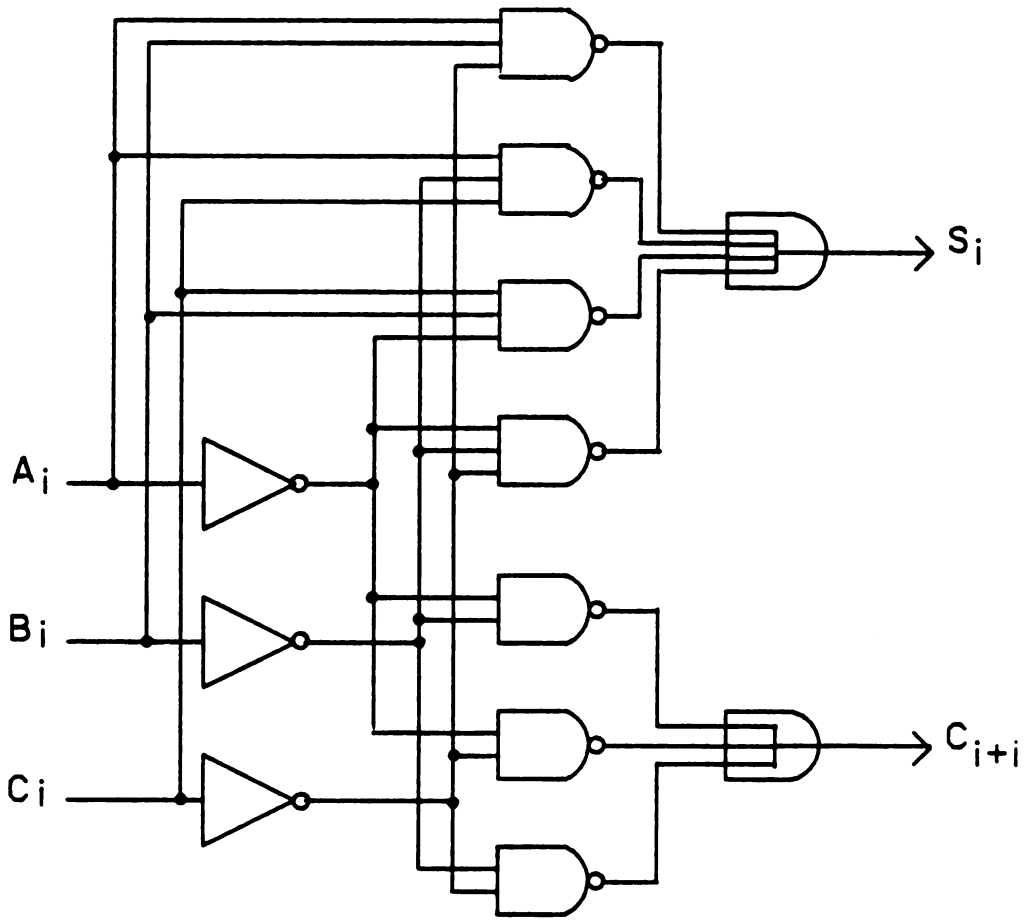

Figure 2.1 Logic gate diagram of a one-bit full adder with hard-wired logic [6].

with only two levels of gate delay, provides the minimum propagation time in the sense of gate delay. However, the hardwired logic gates in this circuit restrict the implemention to certain IC technologies, such as open collector circuitry and NMOS. Figure 2.2 provides another circuit version of a full adder using only NOR gates [5]. This circuit, having three levels of gate delay, is the optimal design without using hardwired logic gates [5].

The one-bit full adder is the basic building block in the examples of this research and thus its performance has a direct effect on that of the desired VLSI chips.

2.2 Ripple-Carry Adder

A ripple-carry adder is selected as a one-dimensional tesselation example in this research. A ripple-carry adder is formed by connecting one-bit full adders in a linear manner as illustrated in Figure 2.3.

The regularity and localized interconnectivity of this circuit makes it eligible for VLSI implementation and as a working example in this work even though it is considered to be a slow adder [6]. While faster adder circuits are known, such as carry lookahead and conditional sum adders, they lack the regularity and connectivity requirements described previously. The propagation time of an n-bit ripple-adder is nt_{FA}, where t_{FA} is the delay time of the one-bit full adder and n is the number of full adders in cascade. Ripple-carry addition is more likely

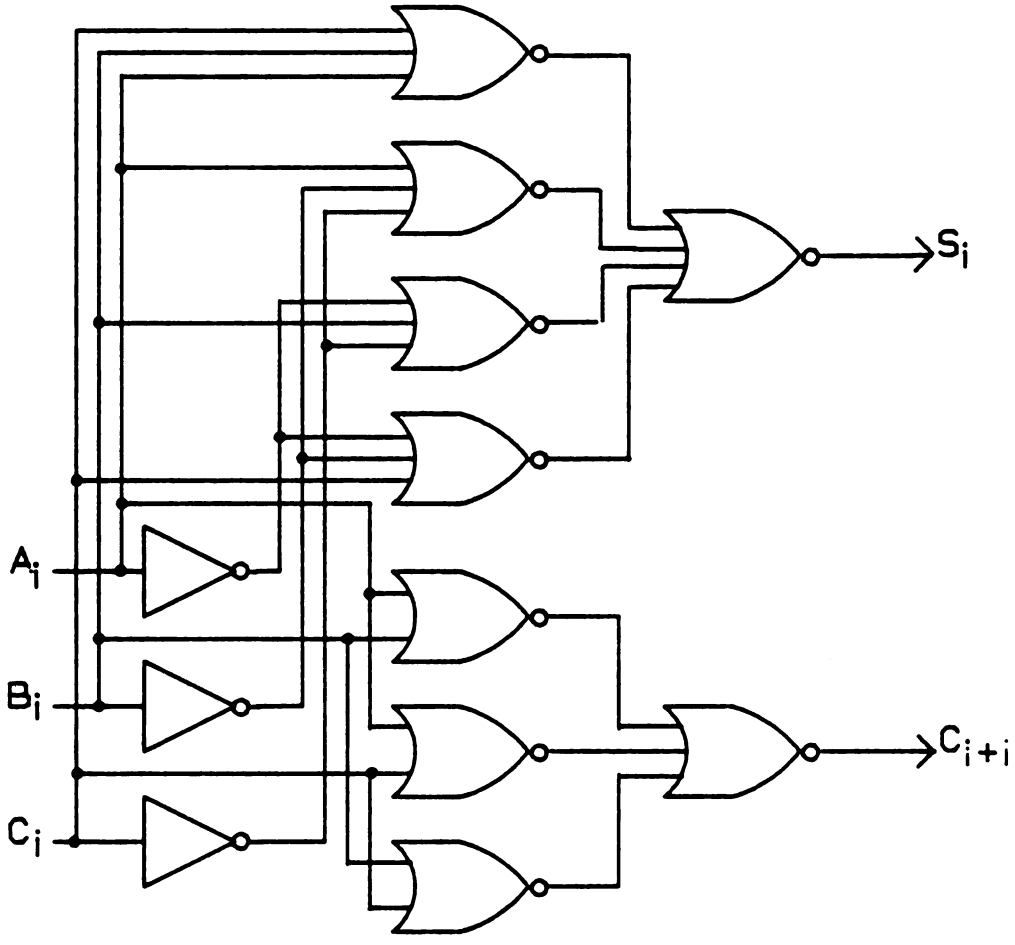

Figure 2.2 Logic gate diagram of a one-bit full adder using only NOR gates [5].

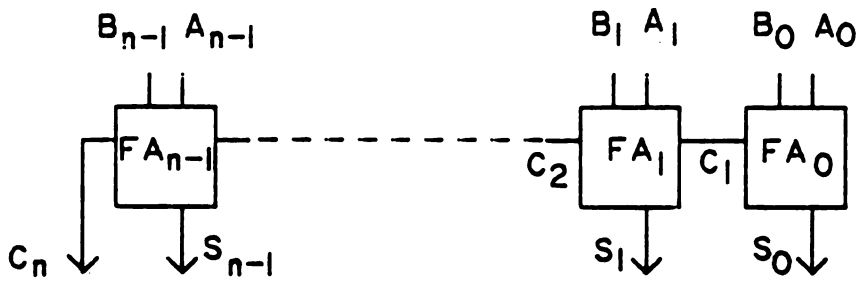

Figure 2.3 Block diagram showing the construction of a ripple-carry adder.

to be used as a functional block in a more complicated design rather than as a stand-alone chip.

2.3 Array Multiplier

High-speed multiplication is an essential function in many digital systems. The speed is important in various applications requiring real time arithmetic calculations. Conventional add-and-shift multipliers are less expensive in the sense of chip area requirements, however, they are too slow to satisfy many performance demands.

Both signed and unsigned array multipliers have been developed [6]. The main difference between these two types of array multipliers is that

the former can handle signed operands directly without the need for complementing circuitry. Neglecting this complementing circuitry for signed operands, the overall architectural characteristics of both types of array multipliers are quite similar. The Braun array, an unsigned array multiplier, is selected as a two-dimensional tesselation example in this research [6,7].

The circuit diagram of a 5-by-5 Braun array multiplier is shown in Figure 2.4. The partial product terms, $a_i b_j$, i = 0 to 4, j = 0 to 4, are called summands. The summands are generated in parallel by an appropriate number of AND gates. These summands are then fed to the full adders for operation. In general, an n-by-n multiplier needs $n(n-1)$ full adders and n^2 AND gates. The total delay time of an n-by-n multipler is $t_{AND}+2(n-1)t_{FA}$. This can be verified by tracing the worst case delay path in Figure 2.4, where t_{AND} and t_{FA} are the propagation delay times of an AND gate and a full adder, respectively.

2.4 Implementation Technology

Many IC technologies can be used for integrated circuits. Some examples of IC technologies include bipolar, NMOS, PMOS, CMOS. Each of these technologies has its advantages and disadvantages. Some of the factors that must be considered in choosing a technology include circuit density, richness of available circuit functions, performance per unit power, the topological properties of circuit interconnection paths,

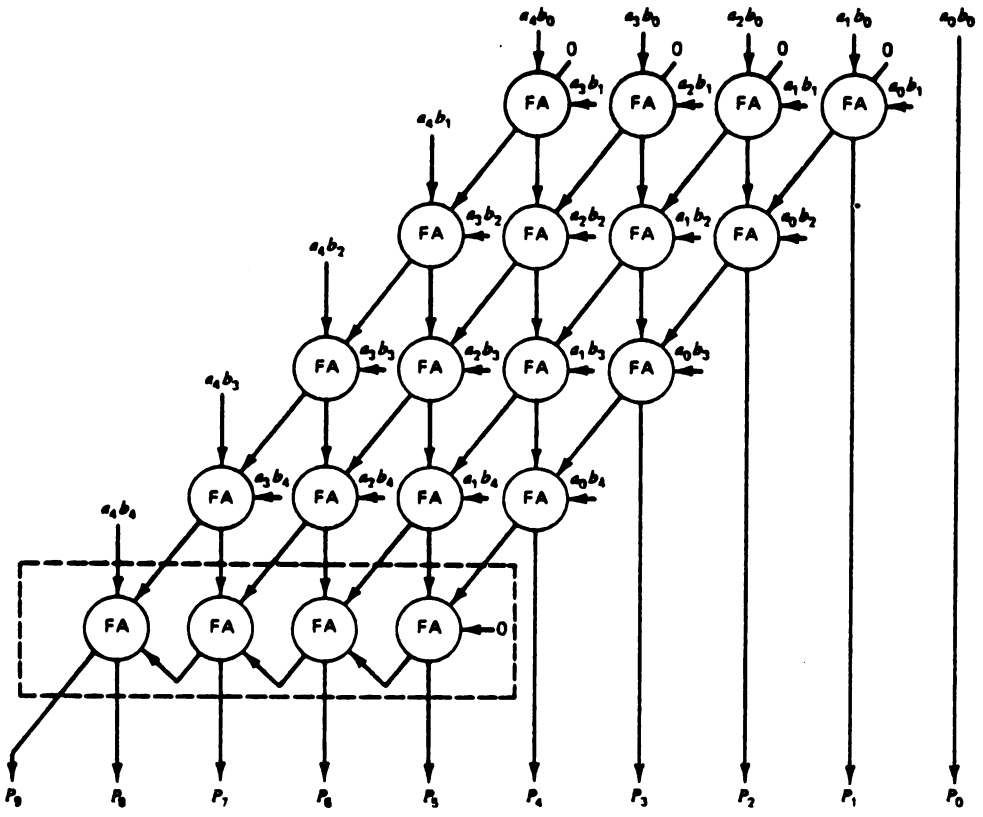

Figure 2.4 Block diagram of a Braun array multiplier [6].

suitability for total system implementation, and general availability of processing facilities [8]. Mead and Conway chose N-channel MOS technology for the reason that the layout prepared by NMOS technology can be easily scaled down as the technology advances [9]. This technology is also choosen in this work to implement the working examples.

The basic element in NMOS technology is an inverter, which is shown in Figure 2.5. The pull-down transistor is an enhancement mode device with a positive threshold voltage, and the pull-up transistor is a depletion mode device with a negative threshold voltage. The gate and source of the depletion mode device are connected together to provide a zero gate voltage so that it is always active and acts as a load

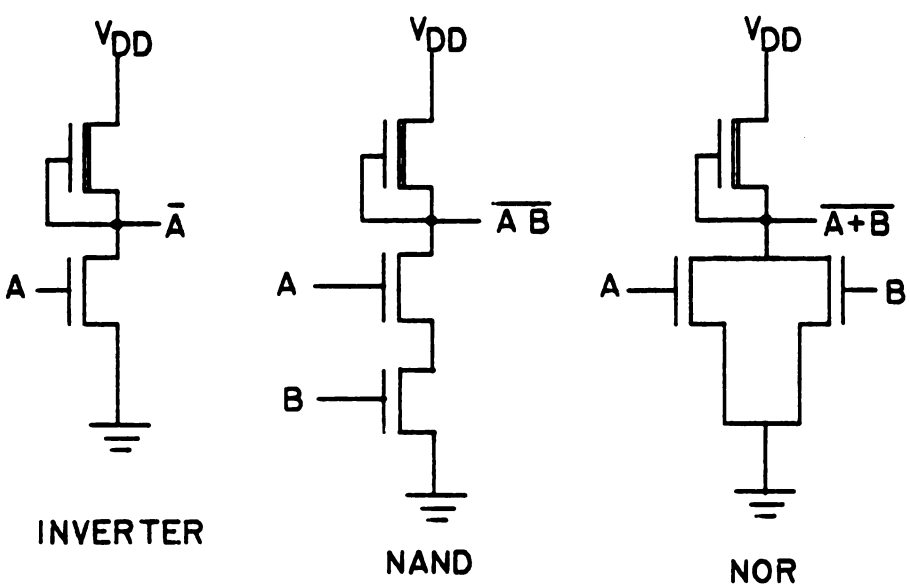

Figure 2.5 Schematic diagrams of basic NMOS logic gates [9].

resistor.

An enhancement mode transistor is formed in VLSI by crossing a polysilicon line over a diffusion line and a depletion mode transistor is formed by the same procedure plus the application of ion implantation to achieve a negative threshold voltage. The overlapped region of the polysilicon and diffusion lines determines the transistor channel. The channel length-to-width ratio, Z, of a transistor is an important parameter in NMOS technology. This ratio of the pull-up transistor, Z_{pu}, must be at least four times of that of the pull-down transistor, Z_{pd} [9]. Positive logic is used in NMOS circuits with the logic levels of approximately 0 and 5 volts. NAND and NOR gates can be constructed by simple modifications of the inverter circuit.

2.5 Design Rules

Design rules for VLSI are a set of rules stating the permissible geometries, including minimum allowable values for the widths, separations, extensions, overlaps, etc., that can be used by a designer in the integrated layout of a circuit. These rules assure that the patterns generated are within the resolution of the fabrication process and that they do not violate the device physics required for the proper operation of transistors and interconnections formed by the process.

Mead and Conway have developed a structured VLSI design method [9]. A set of design rules in dimensionless form is provided as constraints

on the allowable ratios of certain distances to a basic length unit. This length unit is approximately 1 micron for current research processes [10]. The pattern resolution of optical lithography is predicted to be about 0.5 microns by 1997 [11]. Moreover, new techniques including electron-beam and X-ray lithography have promised an even lower pattern resolution limitation [12].

The advantage of dimensionless design rules is that designs implemented accordingly can be easily scaled down as the fabrication process progresses. As the integrated circuit fabrication technology advances, the basic length unit decreases and thus the layout element geometries, which are a function of the basic length unit, also decrease. Therefore, the design may have a reasonable longevity.

2.6 Delay Time Model

Another contribution of Mead and Conway is a delay-time model which is used as a basic tool for determining the delay time of a logic gate [9]. This model recognizes that the delay time of a node depends on the total capacitance of that node together with the gate capacitance and transit time of the driving transistor. The transit time is defined as the average time required for an electron to move from source to drain. Assume that the inverter ratio, K, is the ratio of Z_{pu} to Z_{pd}. Then the falling and rising time for this inverter driving an identical inverter are τ and $k\tau$, respectively. An inverter with load capacitance

C_L requires $(C_L/C_g)\tau$ and $(C_L/C_g)K\tau$ of falling and rising time, respectively.

This model is easy to use. However, in a practical circuit the speed of NMOS device operation must be determined more realistically by the speed with which capacitors can be charged and discharged [13]. The model as described tends to give an underestimated delay time [14]. Therefore, a revised model which estimates the delay time by means of charging and discharging abilities of a logic gate is used in this research to evaluate the speed of the designs [14].

The charging/discharging model estimates the delay time of a logic gate by considering its ability for charging and discharging the effective load capacitance. The effective load capacitance comprises the input capacitances of the driven logic gates and the capacitances of the signal paths. The rising time of a logic gate is determined by how fast the load capacitance can be charged to the voltage level corresponding to logic 1 by the pull-up transistor. On the other hand, the falling time of a logic gate is determined by the speed with which the pull-down transistor discharges the load capacitance to the voltage level corresponding to logic 0.

2.7 Computer-Aided-Engineering of VLSI Circuits

The ability to bring new ideas to production faster is the key to economic success as the integrated circuit becomes increasingly

sophisticated. Until recently the design processes were relatively unautomated [4]. Designers manually created the drawings needed to implement the layout of a chip. Nowadays, VLSI design relies heavily on the support of special computer systems called Computer-Aided-Engineering (CAE) systems. The function of CAE, often lumped under the more general term CAD, is to provide the capability to produce better designs faster and with fewer errors. As with any automation, the primary goal of a CAE system is cost reduction. In the broadest sense, CAE implies the use of a computer system with specialized hardware and software to assist in everything from design, simulation, and testing, through ultimate manufacturing.

The Computervision CAD system in the Case Center of Michigan State University is used to support this research. The Computervision system provides a software package, CADDS2, specially developed for integrated circuit design. The capabilities of layout generation, design rule checking, cell tesselation, layout storage and other indispensible graphic manipulations are available on this system. The layouts generated can be converted into prescribed data formats which can be sent to the silicon foundry for fabrication. A computer language, Integrated Circuit Programming Language (ICPL), can be used for automatic tesselation and other design purposes.

All the layouts presented in this work are generated on this system. Programs written in ICPL are used to build the entire chip by the tesselation of building blocks. In addition, the system assists in the calculation of chip area and the estimation of design complexity.

CHAPTER III

DEVELOPMENT OF DESIGN EXAMPLES

This chapter describes the steps in designing the working examples. The layouts of the circuits, produced according to the Mead and Conway design rules, are then presented and described.

3.1 Selection of Circuits

The one-bit full adder, which is the basic functional block involved in this research, can be implemented in many logically equivalent, yet structurally different, circuits. The first step in the design process will thus be the selection of full adder circuits most conducive to the tessellation and VLSI constraints of this project.

The considerations applied in this research for selecting a circuit to implement a desired function are its gate count and number of delay levels. The gate count of a circuit gives a rough estimation of the area consumed on the chip and the delay levels provides information on its operation speed.

The full adder circuit, shown in Figure 2.1, has a gate count of ten and two delay levels if the hardwired AND gates are considered to consume negligible chip area and have sufficiently small delay time. This circuit is chosen for use in the transistor level design since

it is the best full adder in the sense of gate count and number of delay levels [6]. The operation of this circuit can be expressed by the Boolean equations

$$S_i = \overline{(\overline{A_i}\overline{B_i}\overline{C_i})}\ \overline{(\overline{A_i}B_iC_i)}\ \overline{(A_i\overline{B_i}C_i)}\ \overline{(A_iB_i\overline{C_i})}, \tag{3-1}$$

and

$$C_{i+1} = \overline{(\overline{A_i}\overline{B_i})}\ \overline{(\overline{A_i}\overline{C_i})}\ \overline{(\overline{B_i}\overline{C_i})}. \tag{3-2}$$

A special restriction must be considered in the selection of a full adder circuit for the gate level design. The ratio, Z_{pu}/Z_{pd}, of a complex gate formed of hardwired individual logic gates will be changed since all the pull-down transistors are now connected in parallel. The layout has then to be further modified to maintain the required ratio. The logic gates stored in the gate library are not supposed to be modified in this research so that the advantages of gate level design can be fully utilized. This consideration prohibits the two-level full adder in Figure 2.1 from being used in the gate level designs.

The circuit, shown in Figure 2.2, having a gate count of twelve and three delay levels, is selected for the use in the gate level designs. The reason for this choice is that this circuit is optimal in the sense of delay time and gate count if hardwired logic can not be used [5]. The operation of this circuit can be described by the Boolean equations

$$S_i = \overline{\overline{(A_i+B_i+C_i)} + \overline{(A_i+\overline{B_i}+\overline{C_i})} + \overline{(\overline{A_i}+B_i+\overline{C_i})} + \overline{(\overline{A_i}+\overline{B_i}+C_i)}}, \tag{3-3}$$

and

$$C_{i+1} = \overline{\overline{(A_i+B_i)} + \overline{(A_i+C_i)} + \overline{(B_i+C_i)}}. \tag{3-4}$$

3.2 Gate Library

The design layouts of general logic gates are stored in the Computervision CAD system to form a gate library. The logic gates needed in this research include inverters, two-input NAND, two-input NOR, three-input NOR and four-input NOR gates. Two different configurations are provided for the three-input NOR gates. All three inputs are oriented horizontally in the first version and vertically in the second version.

The layouts of these gates are designed to be universal so that they can be readily pulled from the library and connected to form the desired functional blocks. The ratio Z_{pu}/Z_{pd} is four for all the library gates. This is the minimum requirement. The layouts of these library gates are shown in Figures 3.1 to 3.6.

These and other layout figures in this thesis are shown in color so as to represent different layers more clearly. The diffusion layer is shown in green, the polysilicon layer is shown in red, the metal layer is shown in blue and both the ion implantation and contact cut layers are shown in black.

3.3 Full Adder Design

The full adders designed at both transistor and gate levels are described in this section.

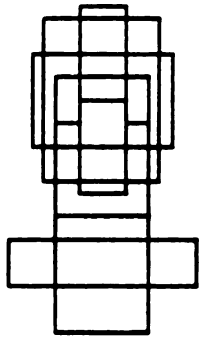

Figure 3.1 The layout of an inverter cell.

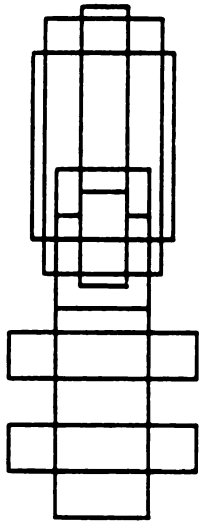

Figure 3.2 The layout of a two-input NAND cell.

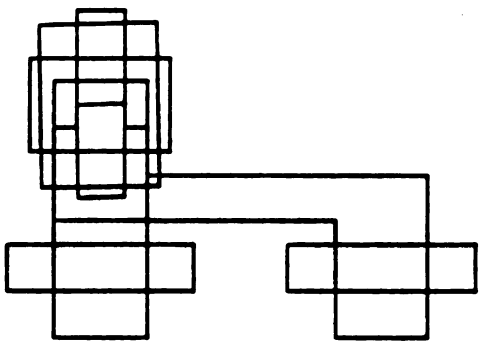

Figure 3.3 The layout of a two-input NOR cell.

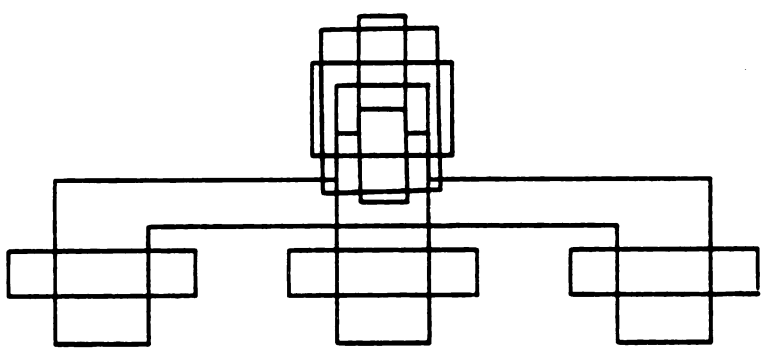

Figure 3.4 The layout of a horizontal three-input NOR cell.

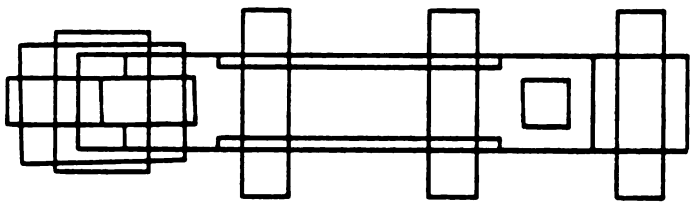

Figure 3.5 The layout of a vertical three-input NOR cell.

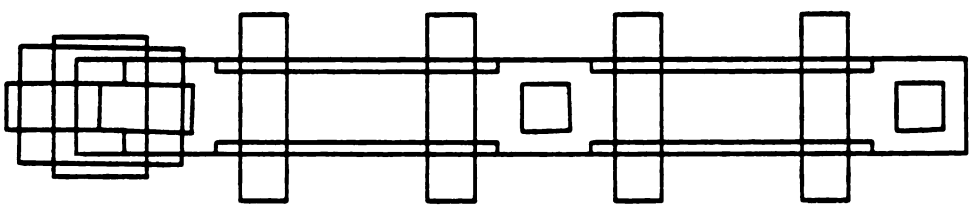

Figure 3.6 The layout of a four-input NOR cell.

3.3.1 Transistor Level Design

The layout of the full adder of Figure 2.1 when designed at the transistor level is shown in Figure 3.7. Two metal lines, required for power supply, run horizontally across the top and bottom of the full adder cell and all transistors are placed within the space defined between them. A third metal line for ground connection passes through the cell and divides it into two regions. The upper region contains the transistors for generating the input complements. The inputs and their complements are fed to the lower region by means of polysilicon lines which form transistors with the diffusion lines. Crossover problems are incurred whenever it is required to run a polysilicon line across a diffusion line without forming an undesired transistor. Crossovers of this type are solved by using metal lines. The transistors contained in the lower region of the cell generate the sum and carry-out of the full adder.

The external communication lines of the full adder cells are located to meet the objective that individual cells can be readily connected side-by-side and the carry-out of a full adder will be connected to the carry-in of the full adder at next stage.

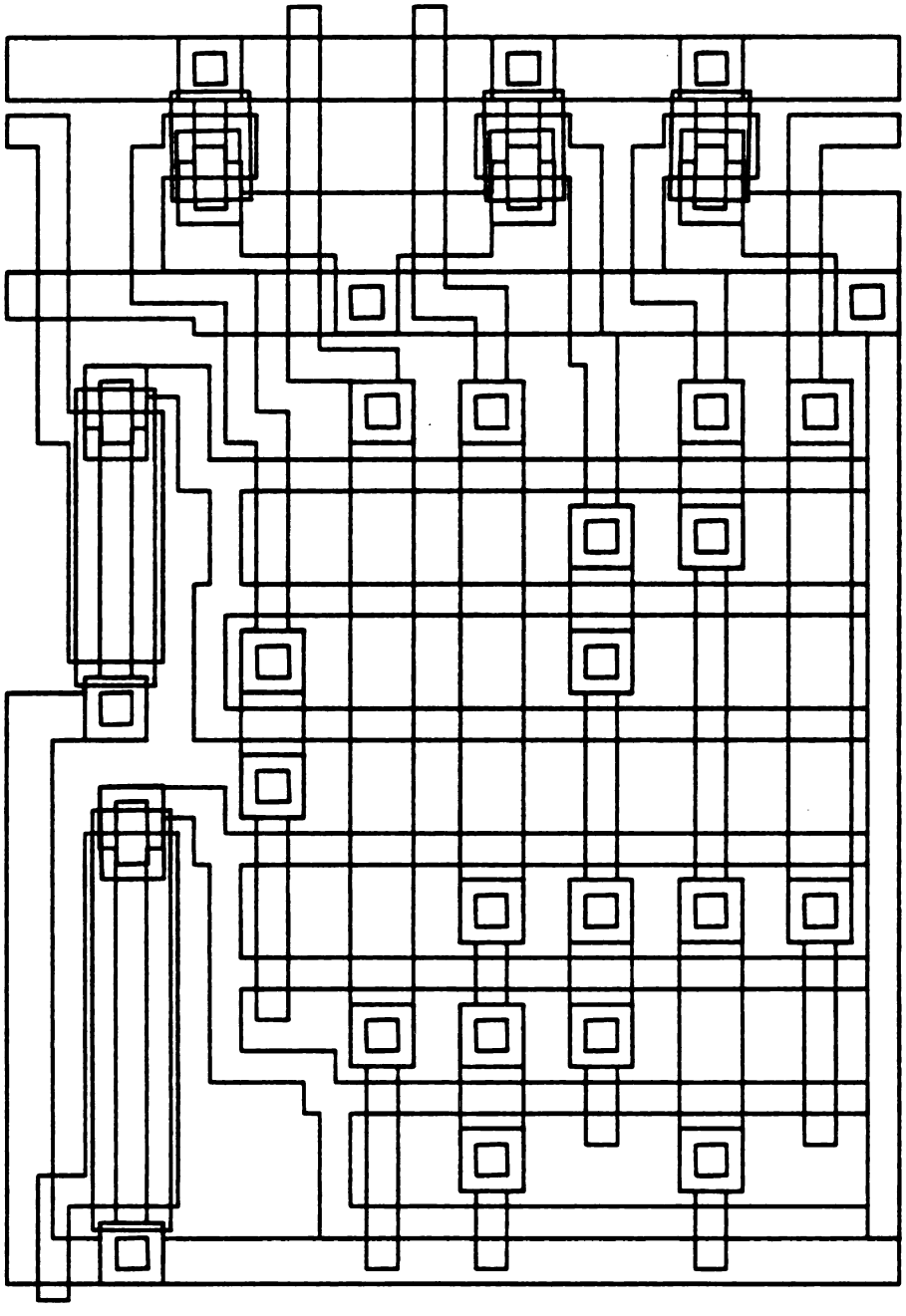

Figure 3.7 The layout of a transistor level full adder cell.

3.3.2 Gate Level Design

The layout of the full adder cell corresponding to Figure 2.2 and designed at gate level is shown in Figure 3.8. Four metal lines run horizontally across the cell. The top and bottom metal lines are provided for power supply and the other two are used for ground connection. In this manner, the cell is divided into three regions. The top region of the cell contains the inverters for input complements and three two-input NOR gates cooperating with a three-input NOR gate to produce the carry-out of the full adder. The bottom region of the cell accomodates four three-input NOR gates and a four-input NOR gate to generate the sum output. The middle region provides space for the routing between logic gates.

All the logic gates are available from the library. The major work at this level of design involves the placement and routing of the appropriate gates.

3.4 Ripple-Carry Adder

The full adder cells described in the above sections are ready for use in the construction of ripple-carry adders since the intercell connections, considered as they were designed, were configured to support linear tessellation.

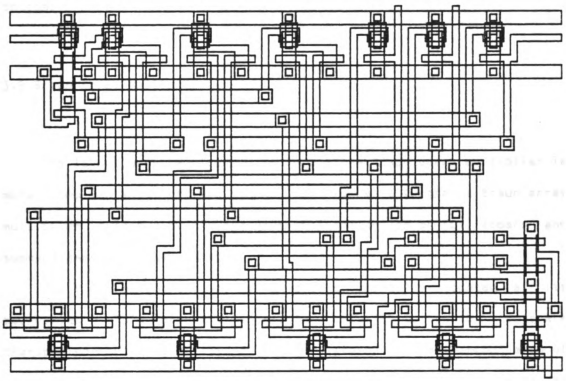

Figure 3.8 The layout of a gate level full adder cell.

The appropriate number of full adder cells are tessellated in a linear manner by means of a program written in ICPL. Figures 3.9 and 3.10 show the layout of 4-bit ripple-carry adders designed at transistor and gate levels, respectively. In both cases, the operands are fed from the top of the ripple-carry adder and the result is obtained at the bottom.

3.5 Braun Array Multiplier

The layout work required to implement the Braun array multiplier is more complicated. It was shown in Figure 2.4 that a Braun array multiplier is formed by a two-dimensional connection of full adders and summand-generating AND gates.

The I/O connections of a chip are typically located at its boundaries. This presents a problem relating to how the inputs are transported to the respective full adders placed in the internal area of the array. The full adders can be placed on the chip according to the configuration implied in Figure 2.4. However, almost half of the chip area will be wasted if a square or rectangular chip, which is typical in a commercial process, is used to implement the Braun array multiplier as it appears in Figure 2.4. The full adder array is thus rearranged to a square or rectangular configuration before it is implemented to solve this problem.

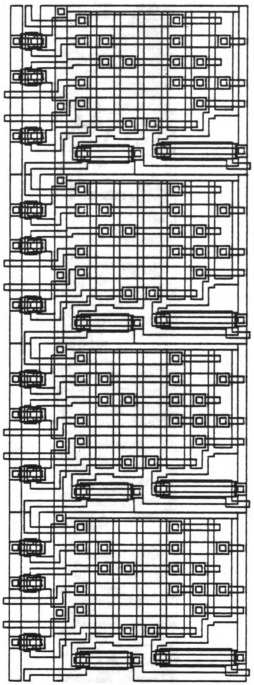

Figure 3.9 The layout of a transistor level 4-bit ripple-carry adder.

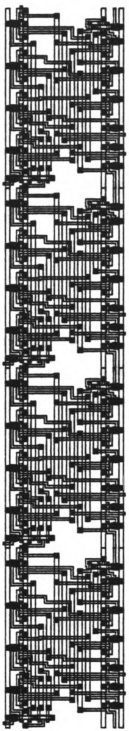

Figure 3.10 The layout of a gate level 4-bit ripple-carry adder.

Another problem is the implementation of AND gates for the generation of summands. The AND function cannot be obtained directly in NMOS and has to be formed by a NAND gate followed by an inverter. This implies that there will be a two level delay. This problem is solved by using NAND gates to replace the AND gates and exchanging the roles of the related inputs and their complements in the circuit.

The array is rearranged to a rectangular configuration shown in Figure 3.11 and the NAND gates are incorporated as part of the full adder cells instead of being treated as separate cells. This also helps to solve the input problem since, after the rearrangement, the cells in the same row will require the same b_i and those in the same column will require the same a_i. The input operands now go in the vertical and horizontal directions as opposed to the diagonal and horizontal directions of the original version.

The required inputs of the various full adder cells in the array are not the same. The cells at the top row of the array need two summand inputs and a zero input. The cells at the bottom row perform a ripple-carry addition. Other full adders have some of their inputs connected to neighbor cells. Several full adder cells, different in the sense of input requirements, are thus prepared for tessellation.

The layouts of all the cells prepared for the tessellation to form Braun array multipliers are shown in Figures 3.12 to 3.28. Most of the cells are basically a combination of a full adder and one or more NAND gates for preparing the summands. The tessellation map in Figure 3.29 shows the relationship between different cells and their locations

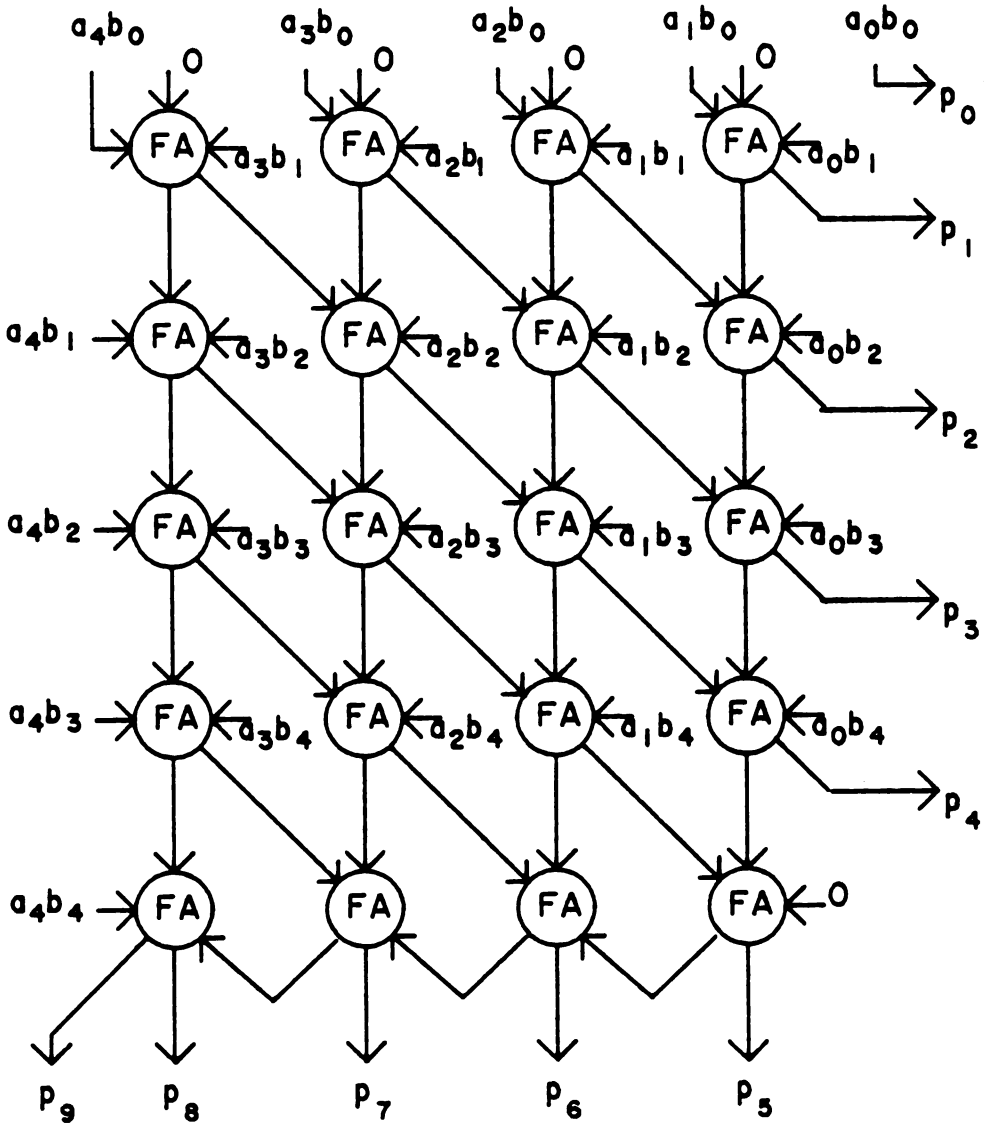

Figure 3.11 The rearranged block diagram of a Braun array multiplier.

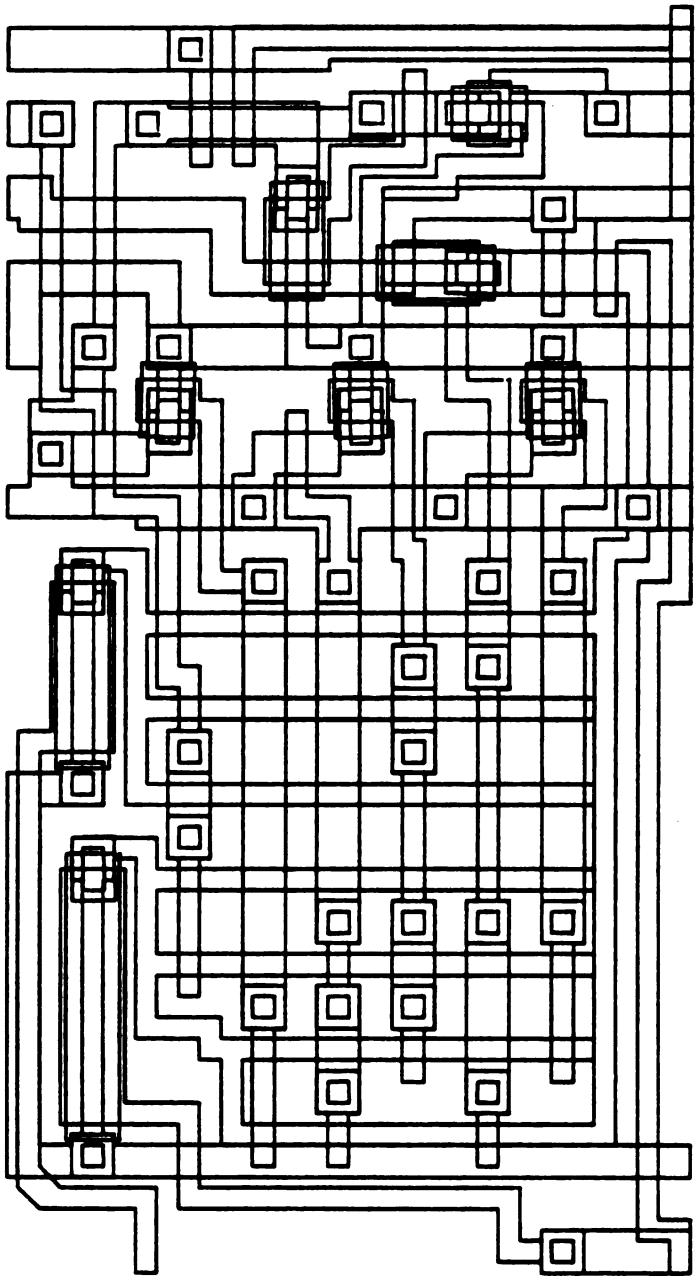

Figure 3.12 The layout of cell 1 for multiplier design.

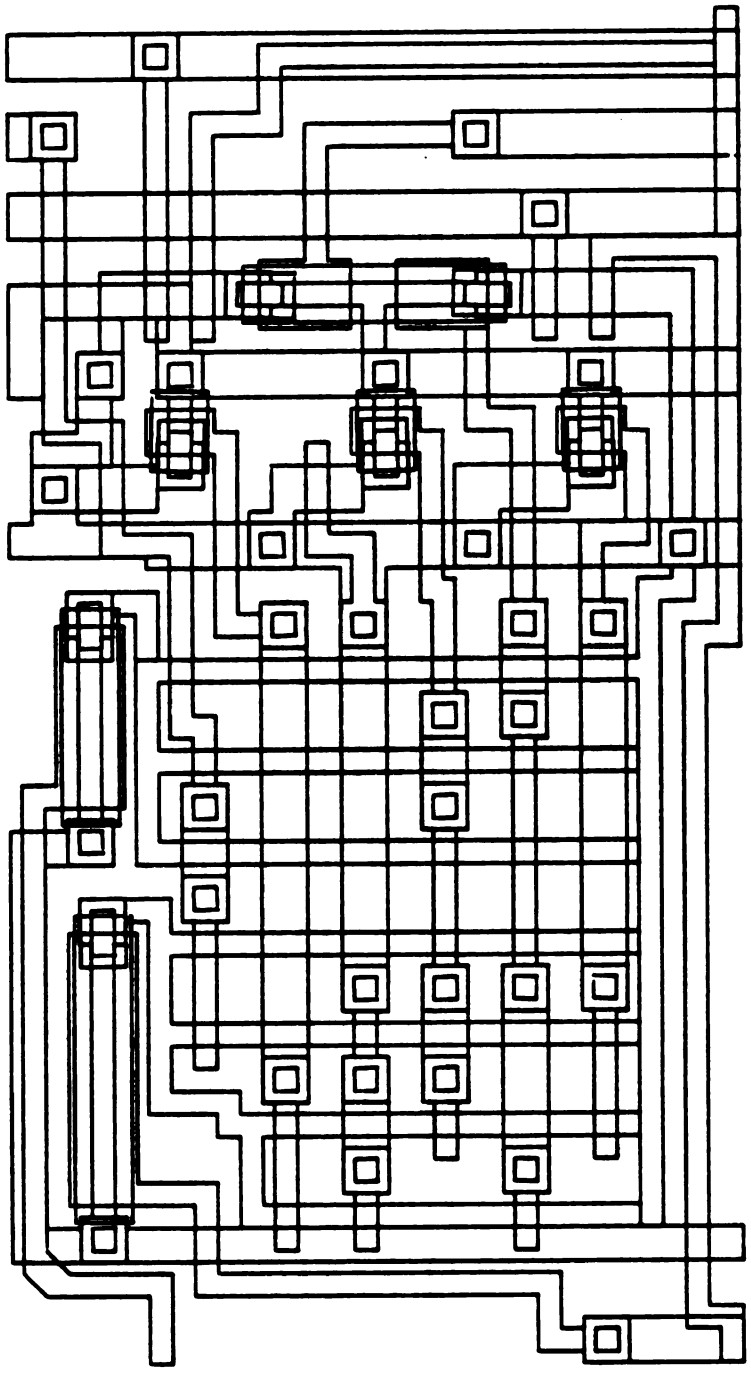

Figure 3.13 The.layout of cell 2 for multiplier design.

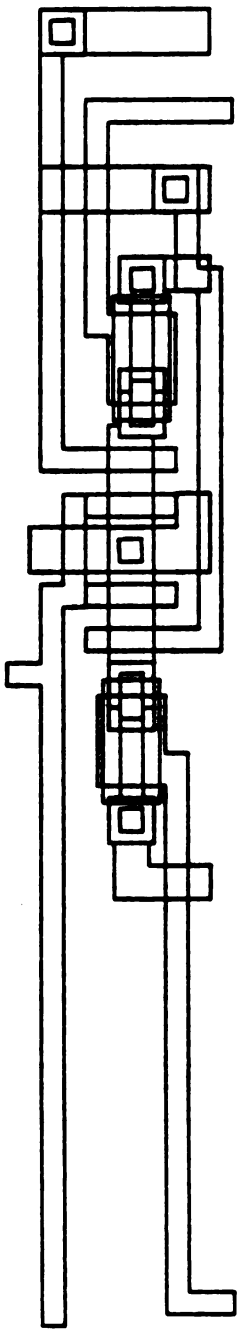

Figure 3.14 The layout of cell 3 for multiplier design.

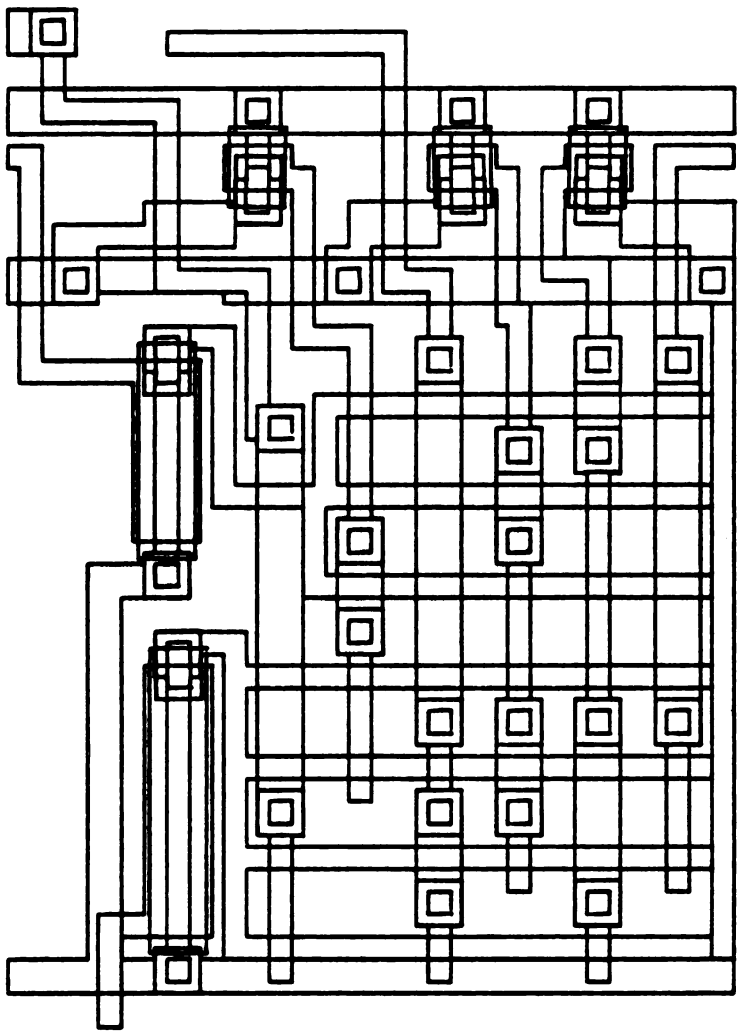

Figure 3.15 The layout of cell 4 for multiplier design.

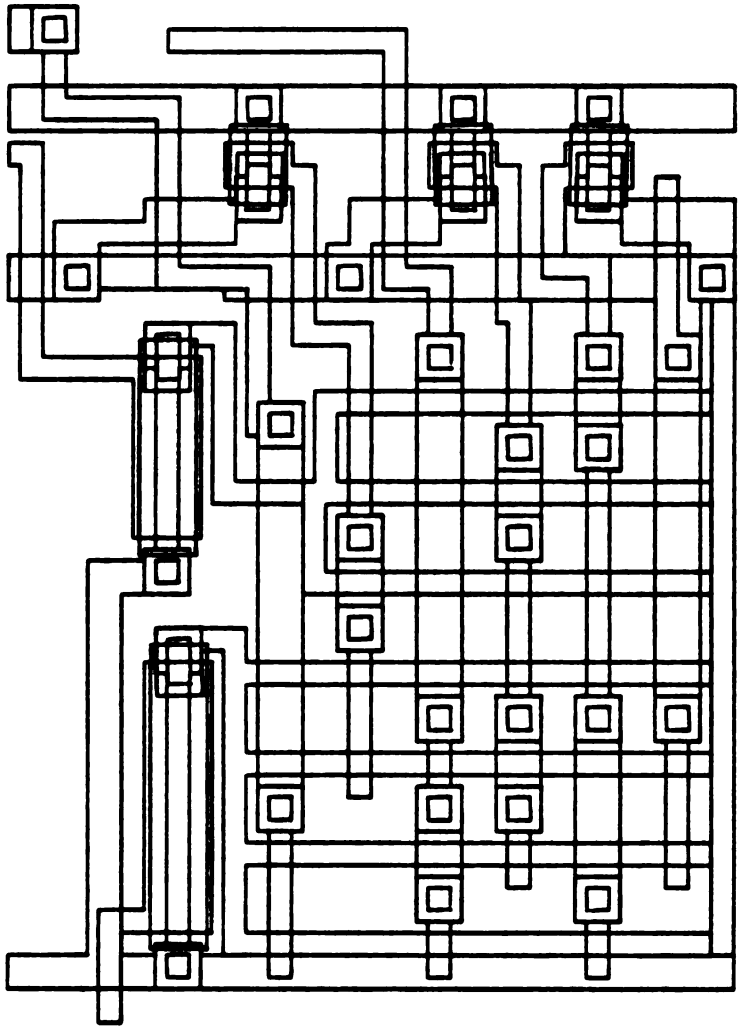

Figure 3.16 The layout of cell 4A for multiplier design.

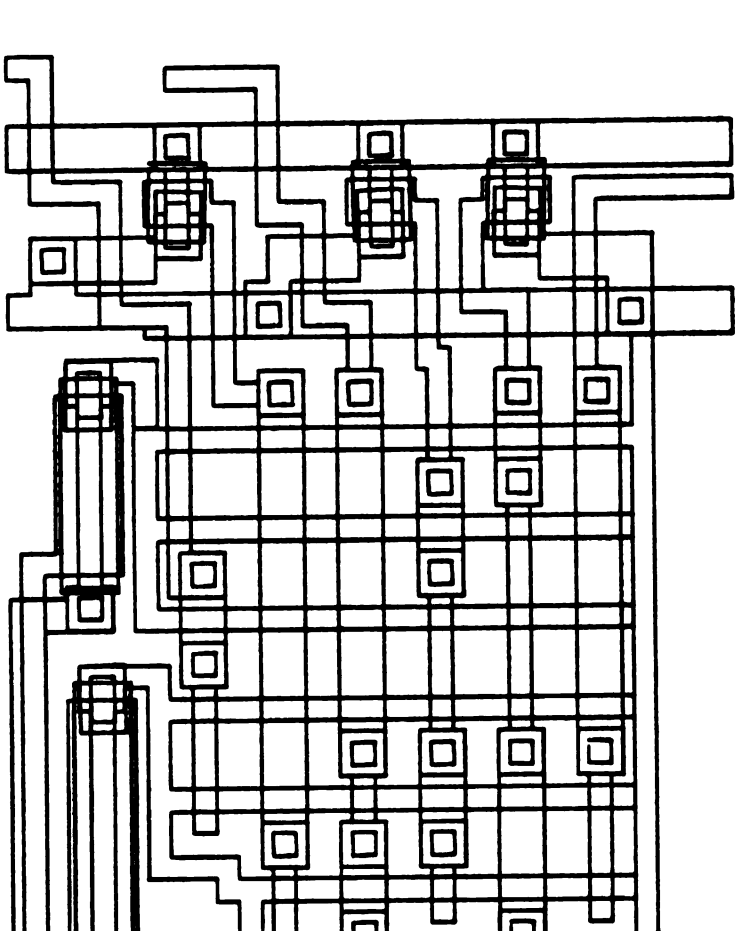

Figure 3.17 The layout of cell 4B for multiplier design.

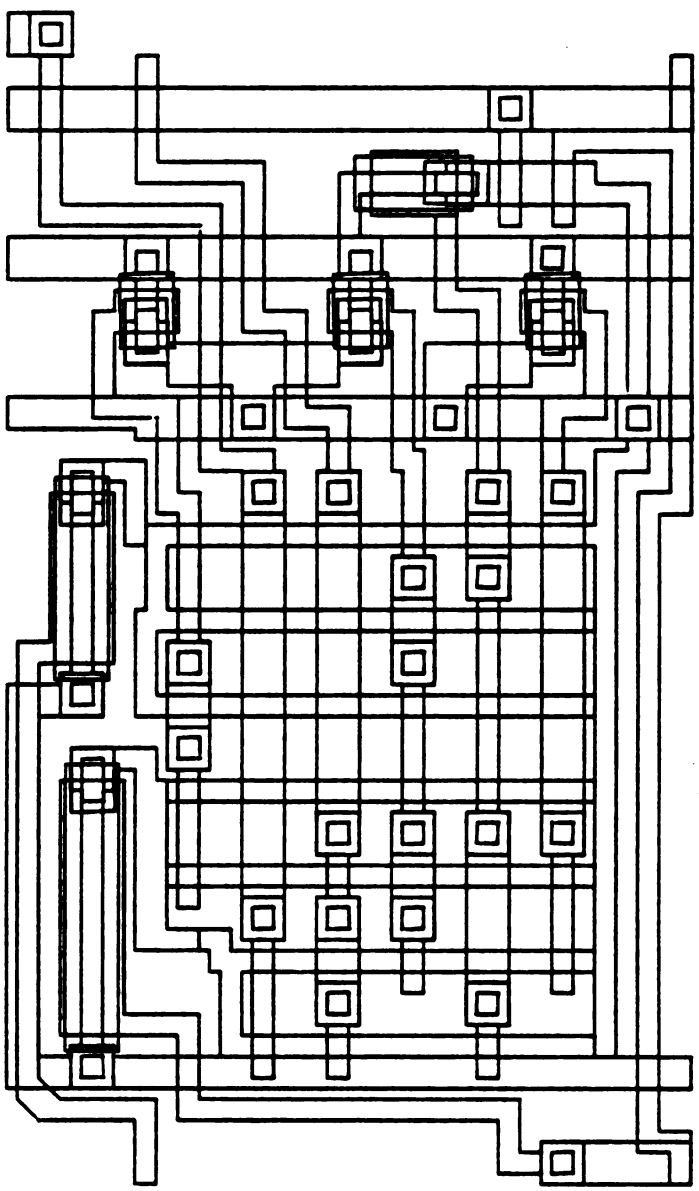

Figure 3.18 The layout of cell 5 for multiplier design.

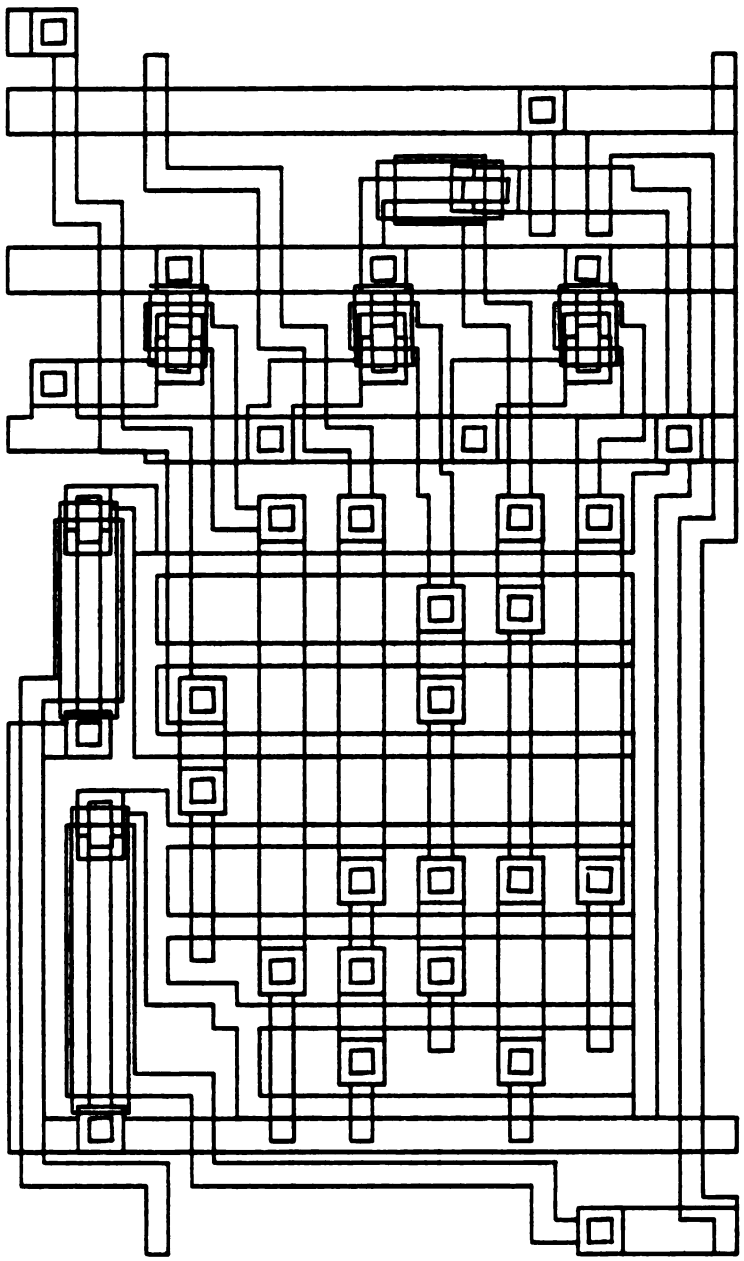

Figure 3.19 The layout of cell 5A for multiplier design.

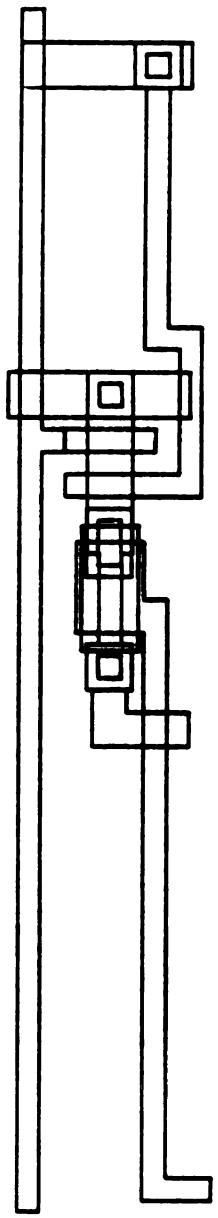

Figure 3.20 The layout of cell 6 for multiplier design.

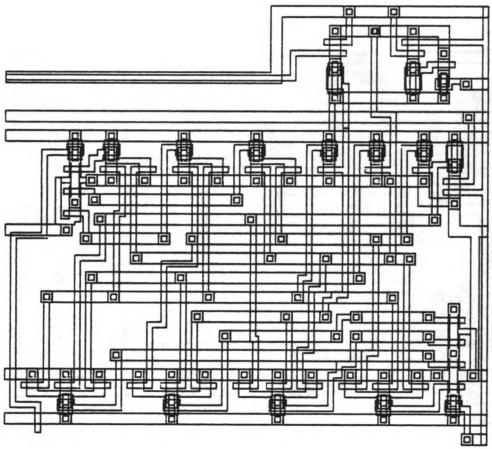

Figure 3.21 The layout of cell 7 for multiplier design.

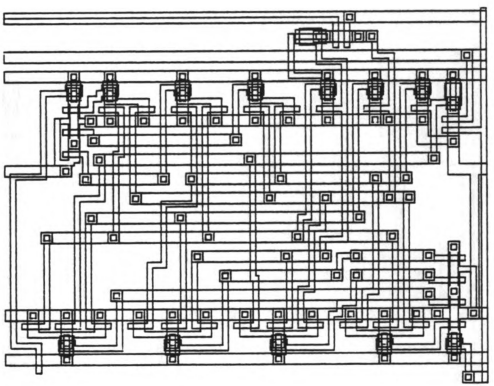

Figure 3.22 The layout of cell 8 for multiplier design.

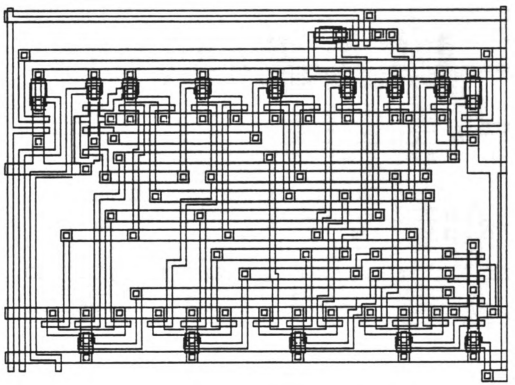

Figure 3.23 The layout of cell 9 for multiplier design.

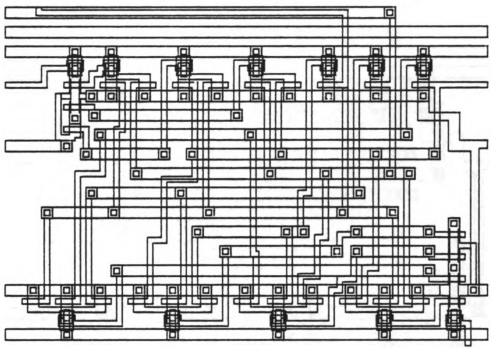

Figure 3.24 The layout of cell 10 for multiplier design.

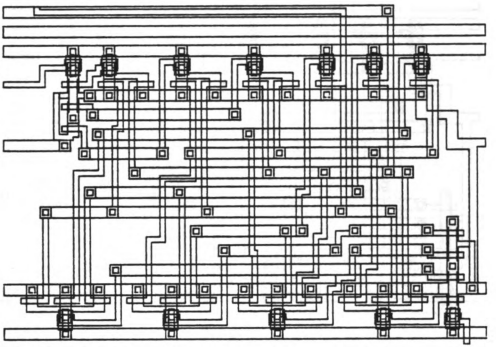

Figure 3.25 The layout of cell 10A for multiplier design.

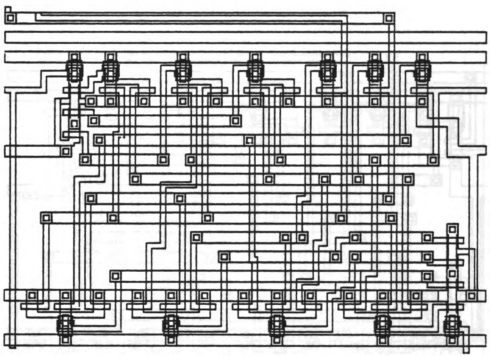

Figure 3.26 The layout of cell 10B for multiplier design.

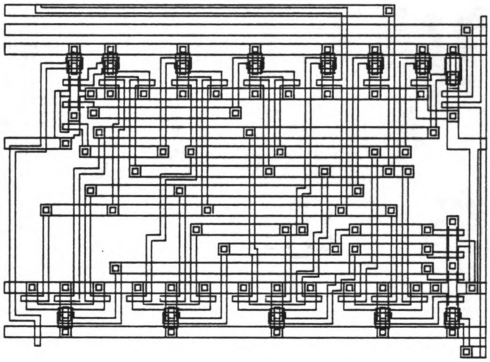

Figure 3.27 The layout of cell 11 for multiplier design.

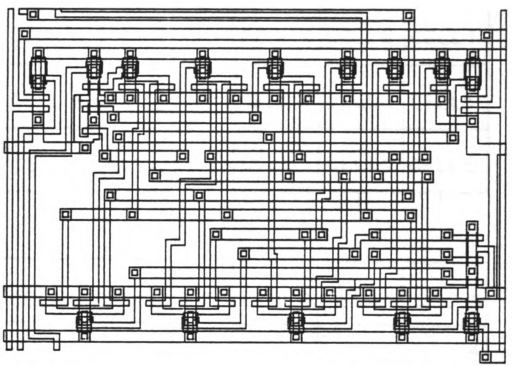

Figure 3.28 The layout of cell 11A for multiplier design.

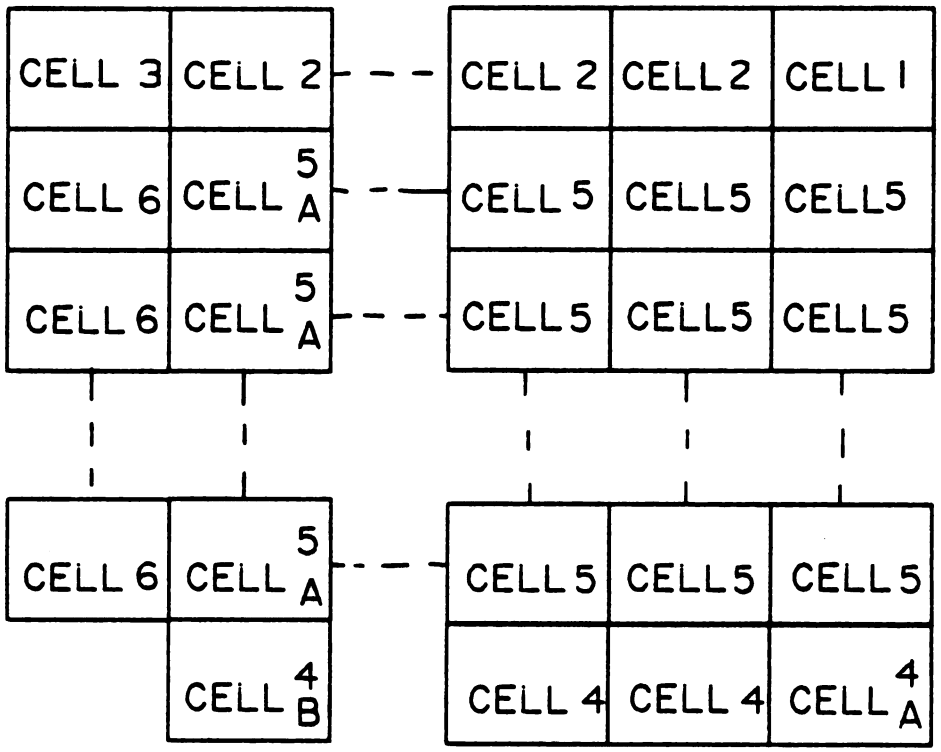

Figure 3.29 Tessellation map for transistor level multiplier.

in the transistor level multiplier array design. Figure 3.30 is a similar tessellation map provided for the gate level multiplier design.

The number of different cells required for the tessellation of a transistor level n-by-n Braun array multiplier is listed in Table 3.1, while table 3.2 is a similar list for gate level multiplier design. Programs written in ICPL are used to produce the layout of n-by-n Braun array multipliers designed by tessellation of the appropriate building blocks at both levels. The tessellations are carried out according to the above maps. The results of designing a 5-by-5 Braun array

Table 3.1 Cell list for transistor level multiplier.

Cell	Quantity
Cell 1	1
Cell 2	n-2
Cell 3	1
Cell 4	n-3
Cell 4A	1
Cell 4B	1
Cell 5	$(n-2)^2$
Cell 5A	n-2
Cell 6	n-2

Table 3.2 Cell list for gate level multiplier.

Cell	Quantity
Cell 7	1
Cell 8	n-3
Cell 9	1
Cell 10	n-3
Cell 10A	1
Cell 10B	1
Cell 11	$(n-2)^2$
Cell 11A	n-2

multiplier at the transistor and gate levels are shown in Figures 3.31 and 3.32 to demonstrate the operations of these tessellation programs.

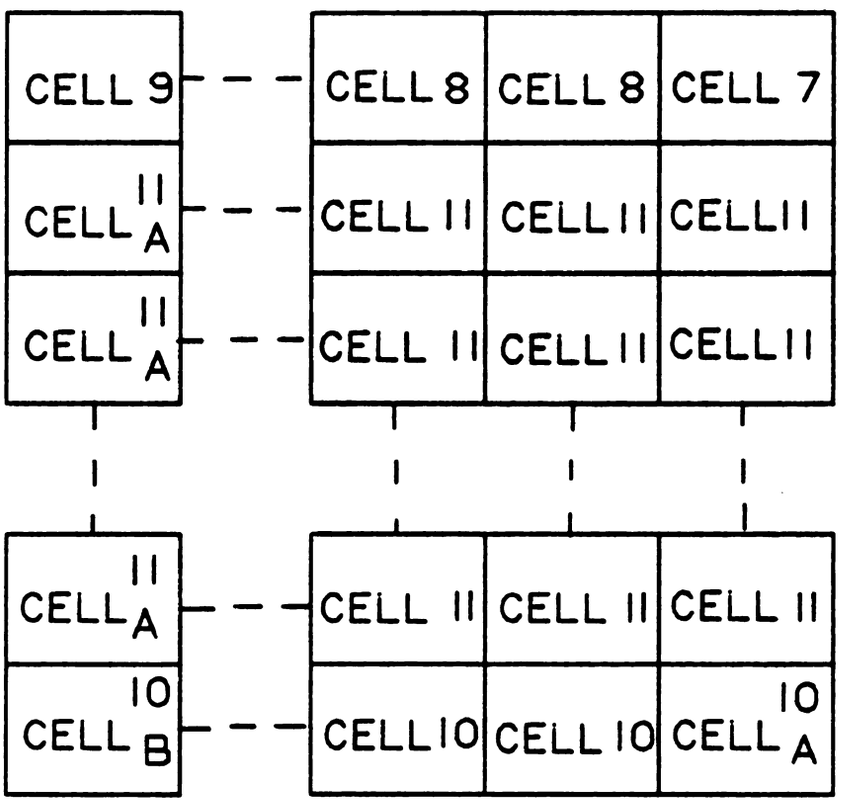

Figure 3.30 Tessellation map for gate level multiplier.

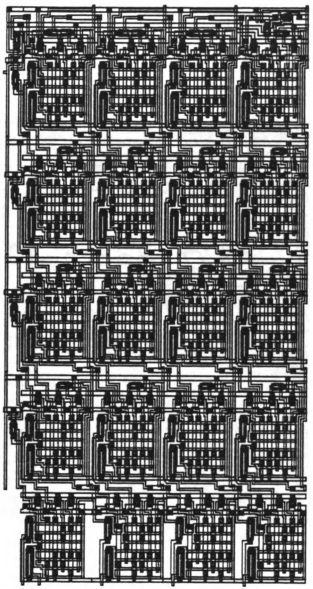

Figure 3.31 The layout of transistor level 5-by-5 Braun array multiplier.

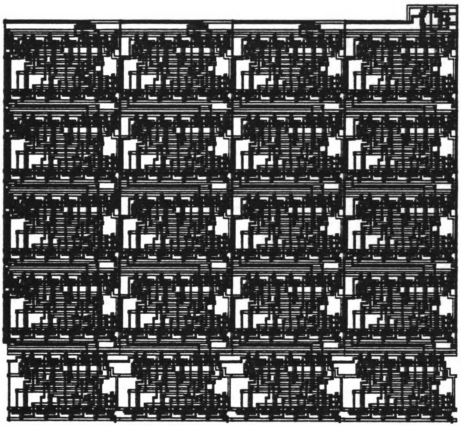

Figure 3.32 The layout of gate level
5-by-5 Braun array multiplier.

CHAPTER IV

DESIGN EVALUATION

The main objective of this research is to determine the performance tradeoffs of designing VLSI chips from different starting levels. The criteria for evaluating designs are defined and the circuits produced in this research are then evaluated and compared in this chapter.

4.1 Criteria of Evaluation

Chip area and propagation time are conventionally used to evaluate the performance of an IC chip. In most cases, these two parameters oppose each other and speed is often achieved at the cost of chip area. Chip area and propagation time are combined in this research to define a term called time-area complexity. The time-area complexity is used to measure the performance of the examples in this work.

The absolute design time, counted in weeks or months spent by the designer, is a subjective parameter. This design time does not provide useful information for comparison since it depends on many human factors including the designer's experience. A measure of design complexity, indicating the degree of difficulty encounterd in the design, will be used in this thesis to estimate the relative design time. The design complexity of a chip will be measured by means of a count of its

components and the density of its layout.

4.2 Device Area Calculation

The area of a cell is readily calculated from its layout diagram. The cells produced in this research are substantially rectangular in shape. The area of a cell is thus the product of its length and width. The total area of a device can then be computed by summing the areas of its building blocks. The fact that all the layouts in this thesis are produced with the basic length unit, λ, of 1 micron enables the following data to be interpreted in either way. The use of this basic length unit has the advantage that the results are technology independent. This implies that these data are still valid even when the layouts are scaled down.

The dimensional data, including the lengths, widths and areas, of the full adder cells provided for ripple-carry adders are listed in Table 4.1.

A total of n one-bit full adders are required for an n-bit ripple-carry adder (RCA). The total area of an n-bit ripple carry adder can thus be calculated by

$$A_{RCA} = nA_{FA}, \tag{4-1}$$

where A_{FA} is the area of the full adder used for tessellation in the

Table 4.1 The dimensional data of the full adders.

Cell	Length (micron)	Width (micron)	Area ($micron^2$)
Transistor level full adder	80	57	4,560
Gate level full adder	162	106	17,172

ripple-carry adder. The areas of various sizes of ripple-carry adder are listed in Table 4.2.

Table 4.2 The areas of ripple-carry adders.

Size	Transistor level design ($micron^2$)	Gate level design ($micron^2$)
4-bit	18,240	68,688
8-bit	36,480	137,376
16-bit	72,960	274,752

The area calculation of the Braun array multiplier (BAM) is slightly more complicated since several types of cells are involved. In addition, there are cell area overlaps in the tessellation procedure due to the intercell communications.

The dimensional data of the cells prepared for the Braun array multiplier are listed in Table 4.3.

The tessellation maps (Figures 3.29 and 3.30) and the cell lists (Tables 3.1 and 3.2) of the Braun array multiplier are used to develop equations for calculating the area of an n-by-n Braun array multiplier. The total area of an n-by-n Braun array multiplier (A_{BAMT}), designed at transistor level, is given in micron^2 by

$$A_{BAMT} = ((n-1)\,64+15)\,(117+106\,(n-2)+83)\,. \qquad (4\text{-}2)$$

The total area of an n-by-n Braun array multiplier (A_{BAMG}), designed at gate level, is given in micron^2 by

$$A_{BAMG} = ((n-2)\,173+180)\,(130+123\,(n-2)+120)\,. \qquad (4\text{-}3)$$

The areas of various sizes of Braun array multipliers as calculated by these equations are listed in Table 4.4.

4.3 Propagation Delay Calculations

Practical logic gates and circuits unavoidably take time to generate valid outputs from the applied inputs. This time delay is an

Table 4.3 The dimensional data of the cells for Braun array multiplier.

Cell	Length (micron)	Width (micron)	Area ($micron^2$)
Cell 1	117	64	7,488
Cell 2	117	64	7,488
Cell 3	117	17	1,989
Cell 4	87	64	5,568
Cell 4A	87	64	5,568
Cell 4B	86	64	5,504
Cell 5	110	64	7,040
Cell 5A	110	64	7,040
Cell 6	106	17	1,802
Cell 7	157	173	27,161
Cell 8	134	173	23,182
Cell 9	134	180	24,120
Cell 10	120	173	20,760
Cell 10A	120	173	20,760
Cell 10B	120	173	20,760
Cell 11	127	173	21,971
Cell 11A	127	180	22,860

Table 4.4 The areas of Braun array multiplier.

Size	Transistor level design (micron^2)	Gate level design (micron^2)
5-by-5	140,378	432,681
8-by-8	387,068	1,203,384
16-by-16	1,641,900	5,131,144

important parameter in IC design since it must be used for determining the clocking period of a system formed by logic circuits.

The output of a logic gate may, at any time, be switched to either one (high-level) or zero (low-level). The unequal configurations of pull-up and pull-down transistors used in NMOS logic gates result in an inherent feature of asymmetric pull-up (rising) and pull-down (falling) times [9]. Usually, the longer of these two is the pull-up time and it is thus considered as the worst case delay time of the logic gate. Thus, in the following description, the term "delay time of a logic gate" refers to its pull-up time.

The computation of propagation time involves finding the worst case signal path, in the sense of delay time, between the inputs and outputs of the circuit. The propagation time of a circuit is then estimated by summing up the delay times of the active gates constituting this worst case path.

Due to the circuit complexity, it is not always easy to locate the worst case signal path. Fortunately, complex hardware usually can be partitioned into functional blocks. Therefore, the worst case signal path may be defined in terms of functional blocks instead of logic gates. Thus, the propagation time of a chip can be calculated by adding up the delay times of the functional blocks in the worst case signal path.

The sizes of the pull-up and pull-down transistors used in the designs are listed in Table 4.5 and will be applied in the following propagation delay calculations.

In addition, since these calculations relate to the physical parameters of the transistors, such as doping concentration, threshold voltages, oxide thickness, etc., reasonable assumptions using typical values are listed in Table 4.6 [9,14].

The charging/discharging model for the calculation of delay time is described in the following section. Finally, the results of calculation using this model are provided.

4.3.1 Delay Time Model

One way to compute the speed of a logic circuit is to define a unit gate delay time corresponding to one level of logic [6]. A good measure of this unit gate delay time would be the delay time of a NAND or NOR

Table 4.5 The dimensional data of the transistor.

Design level	Gate	L/W (pull-up) (micron)	L/W (pull-down) (micron)
Transistor	Inverter	4/2	2/4
	Sum-generating hardwired 3-input NAND	24/2	2/2 (each input)
	Carry-generating hardwired 2-input NAND	16/2	2/2 (each input)
	2-input NAND	8/2	2/4 (each input)
Gate	Inverter	4/2	2/4
	2-input NOR	4/2	2/4 (each input)
	3-input NOR	4/2	2/4 (each input)
	4-input NOR	4/2	2/4 (each input)
	2-input NAND	8/2	2/2 (each input)

Table 4.6 The typical physical parameters of the designs.

Threshold voltage of depletion mode transistor	-4 V
Threshold voltage of enhancement mode transistor	1 V
N-type impurity doping concentration	10^{+16} cm^{-3}
Electron mobility at $300^{o}K$	1000 cm^2/V-sec
Oxide thickness between transistor gate and channel	250 A
Voltage supply (V_{dd})	5 V
High level gate output	5 V
Low level gate output	0 V
Dielectric constant of oxide (e_{ox})	$3.4515X10^{-11}$ F/m

gate. The delay time in a multilevel logic circuit can then be determined by the number of equivalent NAND gate delay levels or the number of levels.

This is probably the most simplistic way to estimate the propagation time of a logic circuit, but it does not take into consideration the loading effect of the logic gates and can only be used as a very rough estimate. It is useful at the beginning of the design process in order to decide which algorithm or architecture is to be chosen for the implementation of a desired function. Actually, the result from this can only be obtained in an ideal case and is a lower bound on the propagation time. In order to fully evaluate a design, a

more precise model which takes loading effect into consideration must be used [14].

The delay time of a logic gate is directly related to the driving capability of its transistors. The pull-up time is limited by the effective load capacitance and the charging current provided by the pull-up transistor. The pull-down time is determined by the effective load capacitance and the discharging current drained by the pull-down transistor. The pull-up time, T_{pu}, and pull-down time, T_{pd}, can be estimated by

$$T_{pu} = C_L V_H / I_{pu}, \tag{4-4}$$

and

$$T_{pd} = C_L V_H / I_{pd}, \tag{4-5}$$

where I_{pu} and I_{pd} are the average pull-up and pull-down currents, respectively, C_L is the effective load capacitance and V_H is the high state output voltage. The average current through the load capacitance may be calculated by taking the average of the currents supplied or drained by a transistor over its active (saturation and triode) regions [14]. The average pull-up and pull-down currents can be expressed by

$$I_{pu} = u_n C_{ox} V_{th}^2 (V_{dd}+V_{th}/3) / 2Z_{pu} V_{dd}, \tag{4-6}$$

and

$$I_{pd} = u_n C_{ox} (V_{dd}-V_{th})^2 (2V_{dd}+V_{th}) / 6Z_{pd} V_{dd}, \tag{4-7}$$

where u_n is the n-type impurity mobility, C_{ox} is the capacitance produced by the transistor gate oxide, V_{dd} is the supply voltage, V_{th} is

the threshold voltage, Z_{pu} and Z_{pd} are the length-width ratios of the pull-up and pull-down transistors, respectively.

Substituting the average currents, I_{pu} and I_{pd}, into equations 4-4 and 4-5, respectively, will give

$$T_{pu} = 2Z_{pu}(V_{dd})^2 C_L/u_n C_{ox}[(V_{th})^2(V_{dd}+V_{th}/3)], \quad (4\text{-}8)$$

and

$$T_{pd} = 6Z_{pd}(V_{dd})^2 C_L/u_n C_{ox}[(V_{dd}-V_{th})^2(2V_{dd}+V_{th})]. \quad (3\text{-}9)$$

According to this model, the delay times of various logic gates involved in this research in terms of C_L, the effective load capacitance, are listed in Table 4.7.

4.3.2 Load Capacitance Estimation

It is essential to estimate the effective load capacitance C_L before the charging/discharging model can be used. Consider the case where the input of a logic gate is connected to the output of another logic gate. The total capacitance, C_L, appearing at the output of the driving gate will be calculated as follows.

The major capacitance related to the load capacitance is the transistor gate capacitance, C_{ox}, due to the oxide interposed between the gate and substrate of the pull-down transistor in the loading logic gate.

Table 4.7 The delay times of the logic gates (in terms of C_L).

Design level	Gate	Delay time (nsec/F)
Transistor	Inverter	$1.24X10^4C_L$
	Sum-generating hardwired 3-input NAND	$7.42X10^4C_L$
	Carry-generating hardwired 2-input NAND	$4.95X10^4C_L$
	2-input NAND	$2.47X10^4C_L$
Gate	Inverter	$1.24X10^4C_L$
	2-input NOR	$1.24X10^4C_L$
	3-input NOR	$1.24X10^4C_L$
	4-input NOR	$1.24X10^4C_L$
	2-input NAND	$2.47X10^4C_L$

The gate capacitance, C_{ox}, of the pull-down transistor in a loading gate may be estimated by

$$C_{ox} = e_{ox}LW/D, \tag{4-10}$$

where e_{ox} is the dielectric constant of the oxide layer, L and W are the length and width of the transistor channel, and D is the thickness of the oxide layer interposed between the gate and the channel [14]. This

gate capacitance can be approximately divided equally into gate-to-source capacitance, C_{gs}, and gate-to-drain capacitance, C_{gd} [15]. The gate-to-source capacitance may be directly accounted for in the effective loading capacitance. However, the gate-to-drain capacitance will be charged in one direction for one polarity of input and in the opposite direction for the opposite polarity input. Thus, its effect on the system is twice that of an equivalent parasitic capacitance to ground. The gate-to-drain capacitance should be approximately doubled and added to the gate-to-source capacitance [9,15]. There are other minor parasitic capacitances, lumped as C_{stray}, associated with the transistor. These are assumed as one-tenth of the input capacitance in this model [16].

In integrated circuits, the capacitances of circuit nodes are due not only to the gate input capacitances but also to the capacitances to ground of the signal paths connected to the nodes [9]. This type of capacitance is not negligible. While gate input capacitances are typically an order of magnitude greater per unit area than the capacitances of the signal paths, the signal paths are often much larger in area than the associated gate regions. Therefore, a substantial fraction of the delay encountered may be accounted for by the communication paths.

It is impractical to calculate these capacitances by considering the signal paths piece by piece. The worse case signal path capacitance, C_{path}, estimated by considering the communication path which produces the largest capacitance in the cell, is used for the

propagation time calculation in this research. Only the capacitances between immediately adjacent lines on the same layer are considered. The other capacitances due to far apart lines or lines on different layers are neglected because of their relatively smaller values.

The path capacitance is estimated for the transistor and gate level full adder cells as $4X10^{-15}$ F and $9X10^{-15}$ F , respectively [15]. The path capacitance for the cells in a Braun array multiplier designed at transistor level is the same as that in the transistor level full adder cell. However, this capacitance is estimated as $2X10^{-14}$F for the cells in gate level Braun array multiplier, due to the much longer worst case communication path involved.

The total effective load capacitance due to one loading gate is defined as the sum of the gate-to-source capacitance, the doubled gate-to-drain capacitance and the stray capacitance of its input transistor and the associated path capacitance. This relationship is expressed mathematically by

$$C_L = C_{gs}+2C_{gd}+C_{stray}+C_{path}. \qquad (4-11)$$

The effective load capacitances associated with the inputs of the logic gates used in this research are listed in Table 4.8.

Table 4.8 The effective capacitances of the gate inputs.

Design level	Gate	Effective input capacitance
Transistor	Inverter	$1.77X10^{-14}$ F
	Sum-generating hardwired 3-input NAND	$8.83X10^{-15}$ F
	Carry-generating hardwired 2-input NAND	$8.83X10^{-15}$ F
	2-input NAND	$8.83X10^{-15}$ F
Gate	Inverter	$1.77X10^{-14}$ F
	2-input NOR	$1.77X10^{-14}$ F
	3-input NOR	$1.77X10^{-14}$ F
	4-input NOR	$1.77X10^{-14}$ F
	2-input NAND	$8.83X10^{-15}$ F

4.3.3 Results

The delay time of a full adder cell must be considered in two cases depending on the nature of any intercell connection. The worst case signal path of a full adder cell can be considered as the path from operand inputs to the carry-out bit if only the carry-out bit is used for intercell connection. The delay times of the full adder cells

designed at transistor and gate levels are calculated according to this signal path as 2.438 and 2.090 nsec, respectively. The worst case signal path of the full adder cell must also be considered as the path from operand inputs to its sum bit output if the sum bit is for communication between cells. The delay times of the cells designed at transistor and gate levels are then 3.408 and 2.181 nsec, respectively.

The worst case signal path of a full adder cell used in a ripple-carry adder is defined from operand input to carry output. This is because the intercell connections are achieved by means of connecting the nearest neighbor carry-out/carry-in lines. The propagation delay of the ripple-carry adder is thus the sum of the carry-out delays of the individual cells. Some representative values are listed in Table 4.9.

Table 4.9 The propagation delays of ripple-carry adders.

Size	Transistor level design (nsec)	Gate level design (nsec)
4-bit	9.752	8.36
8-bit	19.504	16.72
16-bit	39.008	33.44

The full adder cells used in a Braun array multiplier are divided into two groups, those which use preformed summand inputs and those

which perform ripple-carry addition at the bottom of the array. The worst case signal path in the first group of cells must be taken as from their operand inputs to their sum bit outputs while in the second group of cells it is considered as from their operand inputs to their carry-out outputs. The worst case signal path of the entire Braun array includes the NAND gates for generating the summands. The propagation delay of an n-by-n Braun array multiplier is given by the equation

$$T_{BAM} = T_{NAND} + (n-1)T_{FA1} + T_{RCA} \qquad (4\text{-}12)$$

where T_{NAND} is the delay time of a two-input NAND gate, T_{FA1} is the delay time of a one-bit full adder using summand inputs and T_{RCA} is the delay time of the (n-1)-bit ripple-carry adder at the bottom of the array. The results of some representative array sizes are listed in Table 4.10.

4.4 Time-Area Complexity

The time-area complexity is defined as the product of propagation time and chip area. This parameter is used as a measure of the performance of the designs proposed in this research. This figure should be as small as possible for performance optimization. The time-area complexity of the design examples are listed in Table 4.11.

Table 4.10 The propagation delays of Braun array multipliers.

Size	Transistor level design (nsec)	Gate level design (nsec)
5-by-5	24.354	18.614
8-by-8	41.892	31.427
16-by-16	88.660	66.025

4.5 Design Complexity

Table 4.11 The time-area complexities of the design examples.

Circuit	Transistor level design ($micron^2$-nsec)	Gate level design ($micron^2$-nsec)
4-bit ripple-carry adder	177,876	574,232
8-bit ripple-carry adder	711,506	2,296,927
16-bit ripple-carry adder	2,846,024	9,187,707
5-by-5 Braun array	3,418,766	8,053,924
8-by-8 Braun array	16,215,053	37,818,749
16-by-16 Braun array	145,570,854	333,652,639

The design time is also one of the important factors in the determination of the starting level of a particular design. One factor determining the complexity of a design is the number of transistors used. The number of transistors contributes to the difficulty measure of placement, routing and interconnection problems which will eventually determine the required design time.

Design compactness, which pertains to the density of elements on a unit chip area, is achieved by spending more design time. This design time is spent in either man-hours, for a hand layout design, or in a combination of man-hours and CPU time, for a computer-assisted layout. The number of transistors per unit chip area is thus an additional index of design complexity.

As a comparative figure of merit, the design complexity of a certain chip is defined in this research as the product of the number of transistors used and the density of the device.

The design complexity of ripple-carry adders and Braun array multipliers is related to the building cells used in the tessellation. The design complexity of a ripple-carry adder will be the same as that of the full adder cell used. It should be noted that the design complexity of a Braun array multiplier does not depend on its size since any size of Braun array multiplier can be readily generated by means of the same tessellation program with a well-defined set of cells. The cells used in the formation of Braun array multiplier are based on the modification of the full adder cells and have a similar degree of design complexity. Since a comparative result is desired here, it is

sufficient to consider the most complicated cell in a Braun array multiplier to evaluate its design complexity. The design complexity of the working examples are listed in Table 4.12.

Table 4.12 The design complexities of the design examples.

Circuit	Transistor level Design	Gate level Design
Ripple-carry adder	0.148	0.093
Braun array multiplier	0.154	0.102

4.6 Comparison

The designs produced in this research are compared according to the above parameters. The results show that the area of a circuit designed at transistor level is much smaller than that of a functionally identical circuit designed at the gate level. This is reasonable since more flexibility is available in a lower design level. With respect to propagation time, the gate level designs are found to be slightly faster than their transistor counterparts according to the charging/discharging delay time model. However, the performance index, calculated by taking the product of the propagation time and the chip area of a circuit,

shows that a better performance can be achieved by starting the design at the transistor level. These results were anticipated intuitively and come as no surprise.

The result of the delay time comparison, however, was not anticipated. The selection of circuits was based on the criteria of minimum gate count and number of delay levels. The transistor level designs are supposed to be faster than their gate level counterparts under this consideration. However, the results of this research show that there is no strict relationship between the delay levels and the propagation time.

The reason for this result is that, in NMOS technology, the actual gate delay time heavily depends on the size of its pull-up transistor and the load capacitance. The standard assumptions, such as the speed of a NAND gate being the same as that of a NOR gate, are not correct in this IC technology. This is because the channel length of the pull-up transistor in a NAND gate has to be increased to maintain the appropriate pull-up/pull-down ratio while a NOR gate does not have the same problem. The fan-out of a logic gate can also slow down the speed of the driving gate since a larger fan-out implies that the load capacitance of the driving gate is larger. These situations have not been considered in the delay time estimation using gate level delays. In conclusion, this supports the assertion that the gate delay model can only be used as a very rough estimation of the propagation time. The types of logic gates and the fan-out conditions must also be considered in the final selection of a circuit.

A known source of inaccuracy in the propagation delay calculation is the estimation of load capacitances. For example, the path capacitance is estimated roughly by considering the worst case delay path. This may result in a longer than necessary delay time.

The comparison of design complexity unsurprisingly verifies the intuitive notion that higher design complexity is involved in the circuits designed at the lower level.

In summary, this research demonstrates that, in a VLSI design environment, better circuit performance can be achieved by starting the design at a lower level. However, the design complexity encountered at this level will be higher.

CHAPTER V

CONCLUSION

In this work, the tradeoff of designing VLSI circuits at two different starting levels has been studied. The layouts of two examples, a ripple-carry adder and a Braun array multiplier, have been produced and compared for this objective.

5.1 Summary

Advances in VLSI technology have provided new promise for the custom implementation of dedicated circuits. The number of discrete components required in a circuit is greatly reduced by the utilization of VLSI chips. This eliminates the connections between many individual components and the associated problems.

Even though thorough understanding of the entire VLSI technology is rather difficult, an emerging set of simplified design rules and computerized design tools enables an engineer with little knowledge of solid state physics to successfully experiment with VLSI designs.

However, there are some drawbacks in this newly established field. The major drawback is the high design cost which dominates the entire cost of a VLSI project. New design methodologies must continue to be developed and applied to reduce this cost. Hierarchical design is one

way to meet this end.

The basic elements used in a VLSI circuit are different types of transistors. The transistors can be connected in a certain manner to form logic gates. Cells or functional units can then be constructed with the logic gates. The cells are connected to form the desired circuits.

The logic gates or functional blocks can be predesigned and stored to form a library. A design can then be started at different levels. The lowest design level is the transistor level, which implies that the designer has the highest flexibility in the design. This normally results in a better design which is achieved at the cost of a longer design time. The predesigned library gates or cells can be used in a higher level design, and the work is thus simplified into the placement and routing of logic gates or cells. The performance of the resultant system may be affected due to the fact that the library gates or cells are designed to be universal, usually without knowledge of the exact requirement of end designs in which they will be used.

The objective of this research is to investigate the performance tradeoffs resulting from starting representative designs at two distinct design levels, the transistor and gate levels. The layout and analysis of two working examples, a ripple-carry adder and a Braun array multiplier, are independently designed with the assistance of a CAE system. The circuits produced are then evaluated and compared.

The criteria used in this research for evaluation are time-area complexity and design complexity. Time-area complexity, defined as the

product of propagation time and chip area, provides a comparative measure of the performance of a chip. Design complexity is defined as the product of the transistor quantity used in the design and the density of elements on a unit chip area. This provides a comparative measure of the design complexity which is related to the design time.

The various candidate circuits for the implementation of a full adder, which is the major building block in both working examples, are chosen according to their gate counts and gate delay levels. The full adder circuit selected for the transistor level design is a hard-wired circuit with two delay levels. Another circuit having three delay levels is chosen for the gate level design. The selection for gate level design is based on the assumption that the library gates are fixed and cannot be user modified so as to take full advantage of design at this level. A complex gate generated by hard-wiring several individual NMOS gates will have an inadequate pull-up/pull-down ratio.

The circuit layouts, which are based on the Mead and Conway design rules for NMOS IC technology, are presented in Chapter III. This includes the library logic gates, the full adders, the ripple carry adders, the cells for Braun array multipliers and the multipliers themselves. All of these were uniquely designed and parameterized for the purpose of this study.

The ripple-carry adders are formed using an ICPL program tessellating its full adder cells in a linear manner. The array of a Braun array multiplier is first rearranged to solve the problems of signal communication and efficient use of chip area. A program written

in ICPL is used to tessellate the basic cells according to predefined tessellation maps to form the Braun array multipliers. In both the ripple-carry adder and Braun array multiplier cases the ICPL tessellation program enables a designer to produce the layouts, and thus the major portion of the design specifications for ultimate fabrication, by merely specifying the desired array size in bits.

The chip areas and propagation times of the working examples are calculated and presented in Chapter IV. The time-area complexity is calculated and the parameter design complexity for both examples is also obtained.

The charging/discharging model is used in the calculation of propagation delay. This model is based on the charging/discharging ability of a logic gate and its effective loading capacitance. The delay time of a logic gate is defined as the time needed by the driving gate to charge-up the load capacitance to the voltage level corresponding to logic 1. The load capacitance includes the input capacitance of the logic gates at next stage and the communication path capacitance.

The design examples are then compared in terms of the chip performance and design complexity parameters. The results demonstrate that, for functionally identical circuits, the design started at lower level will have better circuit performance which is obtained at the cost of higher design complexity.

5.2 Contributions

This research can be viewed as one of the many steps in establishing a unified VLSI design methodology of which the final goal is total design automation.

Hierarchical design is an efficient approach to reduce the high design cost of a VLSI circuit. In a hierarchical design, the design procedure can be entered at different starting points. In other words, the designer has to decide the entry point of the design procedure, which will eventually determine the performance and design cost of the desired chip. Even though it is commonly anticipated that better performance can be obtained at the cost of longer design time if the design is started at a lower level, the decision of which starting level should be chosen is not trivial.

This research studied the performance and design complexity tradeoff of two different entry points in the hierarchical design procedure. The results of designing two circuits, a ripple-carry adder and a Braun array multiplier, showed that the lower level design had a better performance and encounters higher design complexity.

The complete design procedure, from the selection of circuit for a desired function to the generation of layout diagrams, has been performed in this research. As mentioned above, the comparison of propagation delay disclosed a result which was not anticipated at the beginning of the design procedure. The gate level design has a delay time which is less than that of the transistor level design. This

demonstrates that the commonly used consideration of choosing a circuit according to its gate count and delay levels is not always appropriate. This is because the basic assumption in this consideration, that each gate level has the same contribution to the propagation delay of the entire circuit, is incorrect in NMOS technology.

5.3 Future Development

One of the important performance factors in this project, the propagation time, is calculated without the assistance of the CAE system. The fact that the delay time model used for this purpose can only provide a rough estimation is regrettable but is a function of the available tools.

The normal practice of VLSI design requires that the chip be fabricated before precise measurement of the propagation time is possible. The future tasks for this research would be to do a simulation on the circuits designed and then send the mask information to a silicon foundry for prototype fabrication. More meaningful results can then be obtained.

The gate library can be refined by including more gates with different configurations and pull-up/pull-down ratios to meet the purpose of various applications. A more informative method would be to define a level somewhere between the transistor and gate levels by storing different sizes of depletion and enhancement mode transistors in

the library. The hybrid advantages of both transistor and gate level designs can then be incorporated into the circuits.

Computer programs can be written to generate transistors or gates automatically on a CAE system according to the design rules. Many human errors can thus be eliminated.

The necessity of defining a more complete set of criteria for selecting a circuit to implement a desired function is demonstrated in the above description. One of the future development tasks must be in this direction.

BIBLIOGRAPHY

1. Foster, M. J. and Kung, H. T., "The Design of Special-Purpose VLSI Chips," Computer, Vol. 13 (January 1980), pp. 26-40.

2. Kung, H. T., "Let's Design Algorithms for VLSI Systems," Proc. Caltech. Conf. Very Large Scale Integration (January 1979), pp. 75-90.

3. Fairbairn, D. G., "VLSI: A New Frontier for Systems Designers," Computer, Vol. 15 (January 1982), pp. 87-96.

4. Losleben, P., "Computer Aided Design for VLSI," Very Large Scale Integration VLSI: Fundamentals and Applications, edited by Barbe, D. F., Springer-Verlag Berlin Heidelberg, New York (1982), pp. 89-125.

5. Liu, T. K., Hohulin, K. R., Shiau, L. E., and Muroga, S., "Optimal One-Bit Full Adders with Different Types of Gates," IEEE Trans. on Computers, Vol. C-23, No. 1 (January 1974), pp. 63-70.

6. Hwang, K., Computer Arithmetic, John Wiley and Sons, New York (1979).

7. Braun, E. L., Digital Computer Design, Academic Press, New York (1963).

8. Tobias, J. R., "LSI/VLSI Building Blocks," Computer, Vol. 14 (August 1981), pp. 83-101.

9. Mead, C. and Conway, L., Introduction to VLSI Systems, Addison-Wesley Pub. Co., Reading, Massachusetts (1980).

10. Beyers, J. W. et al., "A 32-Bit VLSI CPU Chip," IEEE Journal of Solid-State Circuits, Vol SC-16, No. 5 (October 1981), pp. 537-547.

11. Keyes, R. W., "Physical Limits in Semiconductor Electronics," Science, Vol. 195 (March 1977), pp. 1230-1235.

12. Edison, J. C., "Fast Electron-Beam Lithography," IEEE Spectrum (July 1981), pp. 24-28.

13. Taub, H. and Schilling, D., Digital Integrated Electronics, McGraw-Hill Inc. (1977).

14. Reinhard, D. K., Integrated Circuit Engineering Notes, Dept. of Electrical Engineering and Systems Science, Michigan State University (1985).

15. Barna, A., VHSIC Technologies and Tradeoffs, John Wiley and Sons, New York (1981).

16. Personal communication with Reinhard, D. K., Dept. of Electrical Engineering and System Science, Michigan State University (December, 1984).