

TIMING AND AREA OPTIMIZATION
OF CMOS BARREL SHIFTER

By

BYUNGHA JOO

Bachelor of Science

Yonsei University

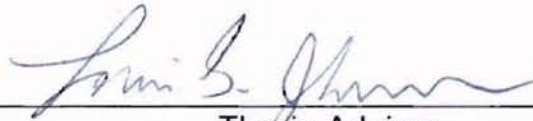
Seoul, Korea

1988

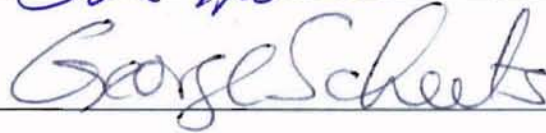
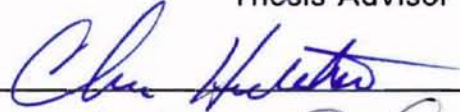
Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May, 1996

TIMING AND AREA OPTIMIZATION
OF CMOS BARREL SHIFTER

Thesis Approved:



Thesis Advisor



Dean of the Graduate College

PREFACE

I wish to express sincere appreciation to my adviser, Dr. Louis G. Johnson. His guidance throughout this research has been precious. My sincere appreciation extends to my other committee members Dr. George Scheets, and Dr. Chriswell Hutchens.

I would also like to give my special appreciation to my parents, Jongheon Joo and Kyungja Cho; my parents-in-law, Jaenam Lee and Teaksung Yang for their support and encouragement. The strong encouragement of my brother-in-law, Sunyoung Lee is also gratefully appreciated. I want to share this joy with my younger sister, Hyunyeon, my brave son, Myungjoon and my pretty daughter, Yujin.

Finally, thanks also go to my wife, Jeeyoung Lee for her love, many sacrifices, and precious care for us.

TABLE OF CONTENTS

| Chapter | Page |
|---|------|
| I. INTRODUCTION | 1 |
| Design Automation | 1 |
| Chapter Description | 3 |
| II. LITERATURE REVIEW | 5 |
| Introduction | 5 |
| Switch Level CMOS Model | 5 |
| CMOS Circuit Optimization | 7 |
| III. DELAY ESTIMATION | 10 |
| Introduction | 10 |
| Transistor Model | 10 |
| RC Delay Estimation | 12 |
| IV. TIMING AND AREA OPTIMIZATION OF SERIES INVERTING BUFFERS | 15 |
| Introduction | 15 |
| Timing Estimation | 15 |
| Area Estimation | 18 |
| Simulation Results | 20 |
| V. TIMING AND AREA OPTIMIZATION OF BARREL SHIFTER | 26 |
| Introduction | 26 |
| Overview of Barrel Shifter | 26 |
| Layout Parameters | 28 |
| Circuit Optimization | 31 |
| Optimization Results | 37 |
| Comparison to others work | 39 |

| Chapter | Page |
|------------------------------------|------|
| VI. SUMMARY and FUTURE WORKS | 48 |
| LITERATURE CITED | 50 |
| APPENDIX I | 53 |
| APPENDIX II | 61 |

LIST OF TABLES

| Table | Page |
|--|------|
| I. MOS Channel Resistance Classification | 11 |
| II. Normalized Parameters | 15 |
| III. 3 Stage Inverter Optimization | 21 |
| IV. 7 Stage Inverter Optimization | 23 |
| V. 8 bit Barrel Shifter transize Result | 37 |
| VI. transize Process Parameters | 39 |
| VII. transize Result of 32 bit Barrel Shifter | 40 |
| VIII. transize Result of 8 bit Barrel Shifter | 41 |
| IX. iCONTRAST Result of 8 bit Barrel Shifter | 42 |
| X. Clocked Inverter Type 8 bit Barrel Shifter Result | 45 |
| XI. Computing Time to Optimize 8 bit Barrel Shifter | 47 |

LIST OF FIGURES

| Figure | Page |
|--|------|
| 1. Series Inverters | 1 |
| 2. Transistor Model | 11 |
| 3. MOS Capacitance Model | 12 |
| 4. RC Tree | 13 |
| 5. Capacitance Origin | 16 |
| 6. Series Inverters for Optimization | 16 |
| 7. Area Dependency of an Inverter Layout | 18 |
| 8. Transistor Size of 3 Stage Series Inverters | 22 |
| 9. Delay and Area Relationship of 3 Stage Series Inverters | 23 |
| 10. Optimal Point Evaluation of 7 Stage Series Inverters | 24 |
| 11. Variation of Number of Stage for Series Inverters | 25 |
| 12. Layout Scheme of Barrel Shifter | 27 |
| 13. Simplified Schematic of Barrel Shifter | 27 |
| 14. Function of Standard Layout Blocks | 28 |
| 15. RC Equivalent Circuit for Barrel Shifter | 29 |
| 16. Worst Case Data Path in Barrel Shifter | 31 |
| 17. Schematic of 8 bit Barrel Shifter | 38 |

| Figure | Page |
|---|------|
| 18. Area and Timing Relationship of 32 bit Barrel Shifter after transize Optimization | 41 |
| 19. Area and Timing Relationship of 8 bit Barrel Shifter after transize Optimization | 42 |
| 20. Area and Timing Relationship of 8 bit Barrel Shifter after iCONTRAST Optimization | 43 |
| 21. Schematic of 8 bit Barrel Shifter with iCONTRAST Result | 44 |
| 22. Area and Timing Relationship of Clocked Inverter Type Barrel Shifter after iCONTRAST Optimization | 45 |
| 23. Schematic of Clocked Inverter Type 8 bit Barrel Shifter with iCONTRAST Result | 46 |

CHAPTER I

INTRODUCTION

Design Automation

In a CMOS integrated circuit design, it is time consuming procedure to determine the proper transistor sizes for the design with circuit simulator. A lot of iterative simulations are needed to get the best performance of the design. Making a design faster does not mean just making transistors bigger to provide more current driving capability.

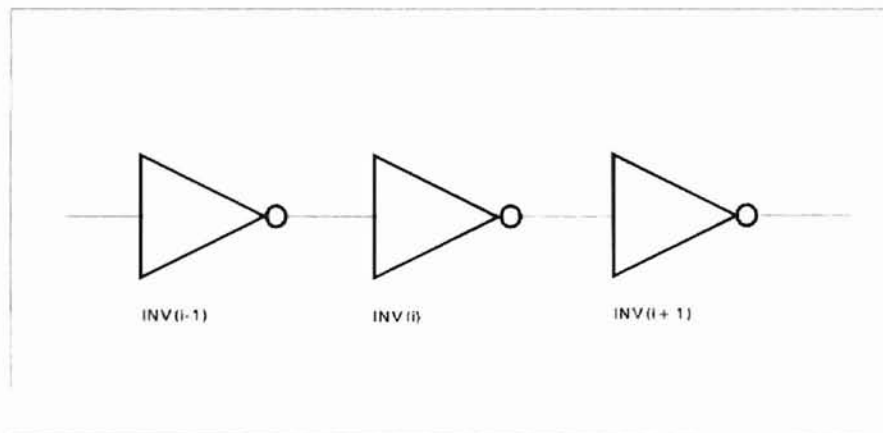


Figure 1. Series Inverters

For a CMOS series inverter in figure 1, a bigger size of $inv(i)$ reduces the delay to $inv(i+1)$. However, from the $inv(i-1)$'s standpoint, $inv(i-1)$ has bigger load. So, $inv(i-1)$ consumes more time than before.

Enlargement of all transistors is not an economical solution and the whole chip size is restricted by process equipment in fabrication. Also, it may not improve delay characteristics. The best solution for the designer is to determine the most effective size for each transistor within a given area.

First of all, size and delay optimization is not the problem of gate level design but the problem of transistor level. However, SPICE like full transistor modeling is not effective, because finding the optimal size in big circuits which have many transistors is important. Full transistor modeling takes too much time and resource for practical use in very large circuits. Therefore, a switch level model should be used. In other words, among the various kinds of transistor characteristics, the simplified model with selected characteristics should be employed.

From the literature, the following contributions are of interest. The transistor is regarded as a current source and its current characteristic is piece wise linear [Ousterhout, 1985] [Hedenstierna, 1987] [Sakurai, 1990] [Dutta, 1995]. The signal transferred among transistors is not a step input but ramp input which has non-zero rise and fall time [Ousterhout, 1985] [Hedenstierna, 1987] [Sakurai, 1990] [Dutta, 1995]. If capacitors of several type are represented by a lumped capacitor, some degree of error rate should be included. A distributed capacitor model has better accuracy [Ousterhout, 1985] [Hedenstierna, 1987] [Sakurai, 1990] [Sapatnekar, 1993]. Resistance of poly gates in wide channel transistors is not negligible for wide or long

channel MOSFETs [Sakurai, 1985]. Series or parallel connected transistors as in nand or nor gates also affect the performance of a circuit [Caisso, 1991].

There were a lot of efforts to get the optimal value of transistor size. The law of diminishing returns was used [Lin, 1975]. Some people solved the problem with lagrangian multipliers [Cirit, 1987] [Hedlund, 1987] [Marple, 1989]. Some papers dealt with a two step iteration method. First, timing is optimized and then area is optimized with the determined timing restriction [Chen, 1991] [Dai, 1989]. Recently, convex optimization was applied [Sapatnekar, 1993].

Previous papers concerned general circuits which have combinational gates and sequential logic circuits.

The objective of this paper is to determine whether fast solution finding of timing and area optimization with a simplified transistor model in a barrel shifter design which has many inputs and outputs and a repeated structure can be practical.

Chapter Description

In Chapter II, a literature review is presented which covers switch level CMOS transistor model and CMOS circuit optimization. In Chapter III, the basic idea of delay estimation is presented which covers transistor modeling

and RC circuit analysis. In Chapter IV, a series of inverters is optimized. In Chapter V, the barrel shifter is optimized and compared to other work. In Chapter VI, a summary and conclusion is given.

CHAPTER II

LITERATURE SURVEY

Introduction

This chapter describes the switch level CMOS model and CMOS circuit optimization.

Switch Level CMOS Model

Delay and area optimization of MOS circuits can be done with a switch level transistor model, and delay estimation of RC circuits. Ousterhout [Ousterhout, 1985] classified three simplified transistor models -- lumped RC model, lumped slope model, and distributed slope model. The lumped RC model is a well-known and simple model. All transistors are substituted with an equivalent resistor, and all loads are replaced by a single capacitor. The lumped RC model is the least accurate model among the three models. The lumped slope model is basically the lumped RC model, but it takes care of non-zero rise and fall times in the input signal. Effective equivalent resistance is calculated through interpolation of two resistance values. One is high-pass equivalent resistance and the other is low-pass equivalent resistance. The distributed slope model considers distributed capacitance. In

the Penfield and Rubinstein method [Rubinstein, 1983], capacitors are considered separately.

Hedenstierna et al. [Hedenstierna, 1987] studied CMOS inverter delay in response to a ramp input. From their paper, the input signal is defined as follows.

$$V_{IN} = \begin{cases} 0, & t < 0 \\ st, & 0 < t < \tau \\ 1, & t > \tau \end{cases}$$

where $\tau = 1/s$ is the rise time of input voltage ramp.

Sakurai et al. [Sakurai, 1990] introduced an α -power law model. In this model, the transistor is a piece wise linear current source.

$$I_D = \begin{cases} 0, & (V_{GS} \leq V_{TH}: \text{cut-off}) \\ \left(\frac{I'_{D0}}{V'_{D0}} \right) V_{DS}, & (V_{DS} < V_{D0}: \text{linear}) \\ I'_{D0}, & (V_{DS} \geq V_{D0}: \text{saturation}) \end{cases}$$

In this model, I'_{D0} and V'_{D0} are modeled in α -powered form. From this model, they extracted propagation delay estimation. The α -power law has good accuracy according to their report.

Dutta et al. [Dutta, 1995] improved the α -power law transistor model. The α -power law transistor model does not give accurate delay estimation when the input slope is very fast or slow. They also took care of negative

delay. Negative delay happens when the input slope is slow and load capacitance is small so that the output switches faster than the input.

Rubinstein et al. [Rubinstein, 1983] proposed a method that can provide upper and lower bounds on path delay in RC tree circuits. Lin et al. [Lin, 1985] extended Rubinstein et al.'s method to a general RC network.

Sakurai et al. [Sakurai, 1985] suggested a degradation factor that accounts for the effect of poly gate sheet resistance on transistor propagation delay. To make a transistor work properly, the transistor should have a low degradation factor.

CMOS Circuit Optimization

Lin et al. [Lin, 1975] tried to find optimal transistor size values with a normalized delay and area equation,

$$F = (\text{normalized area})(\text{normalized delay})^k$$

where k is weight factor. Optimal values are located at the point where F is minimum.

Hedlund [Hedlund, 1987] was looking for delay, area and power optimization. He used an RC transistor model and applied lagrangian multipliers to solve the non-linear problem. He minimized the scaling factor between transistors in adjacent stages. Therefore, the initial P-N transistor ratio within each stage is unchanged.

Cirit [Cirit, 1987] used an RC transistor model and defined area as the sum of all transistor widths. He built an equation to minimize with lagrangian multipliers. And then, he tried to get optimized values with a bisection method.

A two step iteration method was proposed by Dai et al. [Dai, 1989]. At the first step, the program determines optimum timing for the target circuit, and then, within the optimal timing, the program minimizes the circuit size.

Marple [Marple, 1989] developed an optimized layout generator. His program reads a layout database, and represents the circuit in a delay graph to find the critical path. With lagrangian multipliers, an optimized circuit is extracted. Finally, the program generates the optimized layout.

Cheng et al. [Cheng, 1991] developed **iCOACH** that is a poly cell based optimized layout generator. **iCOACH** considers charge sharing and noise margin. The circuit is optimized through a two step iterative process.

Sapatnekar et al. [Sapatnekar, 1993] developed **iCONTRAST** that used a convex optimization method. The transistor model of the program is a non-RC delay model. Originally, the delay model proposed by Hedenstierna et al.'s model was applied. Later, the delay model was changed to the α -power law. **iCONTRAST** minimizes area under a given timing specification. **iCONTRAST** consumes a lot of time. According to their paper, to optimize a

32 bit adder, it takes about eight hours of execution time on a Sun SparcStation I.

The purpose of this thesis is to determine whether a simplified automatic optimization tool can give effective value with significantly small time compared to **iCONTRAST**. Results of this study may provide an alternative to the problem of timing and area optimization in large and repeated pattern circuits like the barrel shifter.

CHAPTER III

DELAY ESTIMATION

Introduction

Delay calculation for a certain path in a CMOS circuit can be divided into two parts. One is the transistor model and the other is the RC tree delay estimation. For fast calculation, the transistor is modeled as a fixed resistor. The RC tree delay is considered in the method that was proposed by Rubinstein et al. [Rubinstein, 1983].

Transistor Model

The simplest model of MOSFET is a fixed resistor that is controlled by its gate voltage. When the transistor is off, there is an open circuit. When the transistor is on, there is a current path with fixed resistance value. This model has about 25% error compared to SPICE results [Ousterhout, 1985], but still gives a benefit on the calculation time.

The resistance of a MOS transistor is proportional to the length and inversely proportional to its channel width. This is one of the factors that can make calculation fast. Generally in digital circuits, the minimum channel length of a process is used. Therefore, the on resistance of a MOS transistor is only inversely proportional to its width.

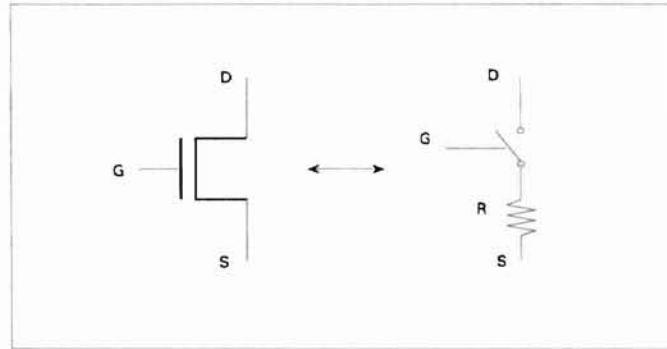


Figure 2. Transistor Model

A MOS transistor has different resistance values depending upon the signal value which passes between drain and source. In this paper, the on resistance of a MOS transistor has four different cases.

| | |
|-----|-------------------------------------|
| Rnl | low pass resistance of NMOS |
| Rnh | high pass resistance of NMOS |
| Rpl | low pass resistance of PMOS |
| Rph | high pass resistance of PMOS |

Table I. MOS Channel Resistance Classification

The capacitance of a MOSFET is also simplified. A MOS transistor has capacitance on the gate, drain, and source. These capacitances have nonlinear characteristics. For the convenience of calculation, all capacitances are replaced by fixed capacitances.

The whole gate capacitance is located between gate and ground when the transistor is off. When the transistor is on, halves of the gate capacitance are applied between gate and drain, and gate and source.

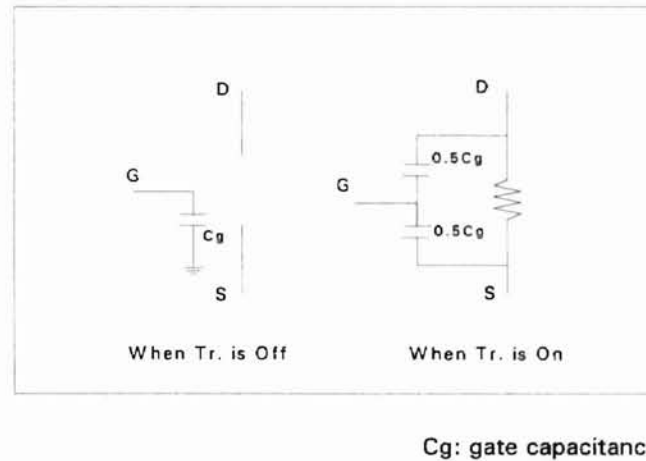


Figure 3. MOS Capacitance Model

RC Delay Estimation

A CMOS circuit can be replaced by the previous RC model. In the equivalent circuit, the gate voltage determines the on or off state of the MOS transistor. Drain and source voltage determine high or low pass resistance value. From the RC equivalent circuit, delay can be produced in this way [Chu, 1987].

$$T_{delay} = \sum_k R_{ke} C_k$$

Figure 4 shows sample of RC tree.

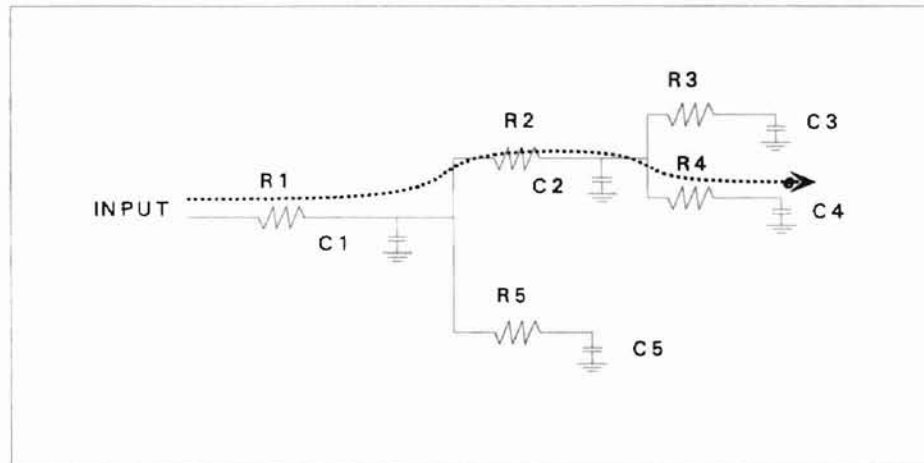


Figure 4. RC Tree

RC time constants between a certain node e and the input are all added up at each stage. Then, from the rest of the branches, RC time constants consist of the resistor which is directly shared between the path to e and the branch, and the capacitor for the branch node. Total time delay for the RC tree is the sum of all RC time constants. This method is valid only if all nodes have same initial voltages.

$$\tau_{INPUT \rightarrow e} = R_1 \cdot C_1 + (R_1 + R_2) \cdot C_2 + (R_1 + R_2 + R_4) \cdot C_4 \quad : \text{time constants of RC path (input to } e)$$

$$+ R_1 \cdot C_5 + (R_1 + R_2) \cdot C_3 \quad : \text{time constants of the rest of the branches}$$

In circuits with transmission gates, both P and N type MOSFET are used in parallel for improvement of signal transfer. But, because of this structure, there is a resistor loop, so that the RC circuit is no longer a tree. When there is only one transmission gate, capacitances from the P and N transistor can be combined by adding them together and the equivalent resistance is a

simple parallel combination of the two resistances. This usually makes the equivalent circuit a tree again.

When there are two or more transmission gates, equivalent capacitance and resistance are not easily available. Generally when there are two or more transmission gates, for layout simplicity, there is no interconnection between N and P MOS transistors. In this way, the area of diffusion contacts can be saved. To solve this case, the following assumptions are needed [Lin, 1984]. The final capacitance can be divided for each path according the path's current driving capability such that the delay timings for two path's are the same.

For example, if portion a of the final capacitance is driven by the P type MOS transistor, then portion $(1-a)$ of the final capacitance is driven by the N type MOS transistor. The delay equation for each path can be formed. Then, since delay timings for both paths are the same, a can be obtained in terms of R and C. As a result, delay timing for transmission gates is available.

CHAPTER IV

TIMING AND AREA OPTIMIZATION OF SERIES INVERTING BUFFERS

Introduction

Timing and area optimization of series CMOS inverters was performed. The MOS transistor is represented as a resistor and capacitor. The optimization process used the law of diminishing returns. The law of diminishing returns was applied to proper level finding of demand and supply in Economics. Transistor size on both ends of the series inverters was fixed. Parameter values from a $2\mu\text{m}$ N-well process was used. The programming language was **MATLAB**.

Timing Estimation

Timing and area optimization of series inverters in CMOS technology was investigated. During the optimization process, resistance and capacitance values are used in normalized form.

| | |
|--------|--|
| R_p' | normalized PMOS resistance, $R_p' = R_p \cdot L$ where R_p is PMOS channel sheet resistance per unit area |
| R_n' | normalized NMOS resistance, $R_n' = R_n \cdot L$ where R_n is NMOS channel sheet resistance per unit area |
| C_D' | normalized diffusion capacitance |
| C_G' | normalized gate capacitance |
| C_I | interconnection capacitance |

Table II. Normalized Parameters

C_i is independent of channel width of MOS transistor, so it is regarded as a constant. Capacitances are extracted from diffusion, gate, and interconnection in layout. Figure 5 shows where each capacitance comes from.

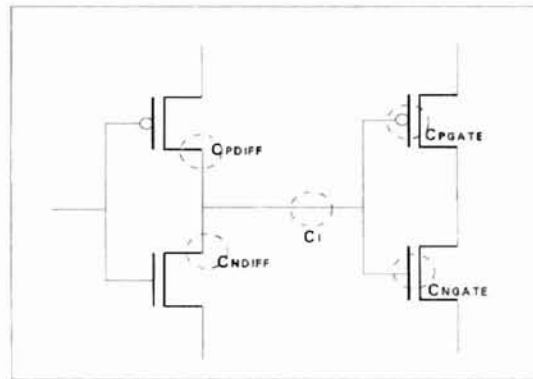


Figure 5. Capacitance Origin

Diffusion capacitances (C_{PDIFF} , C_{NDIFF}) are functions of the driving MOS transistor width and Gate capacitances (C_{PGATE} , C_{NGATE}) are functions of the driven MOS transistor width.

The series inverters for optimization are shown in figure 6.

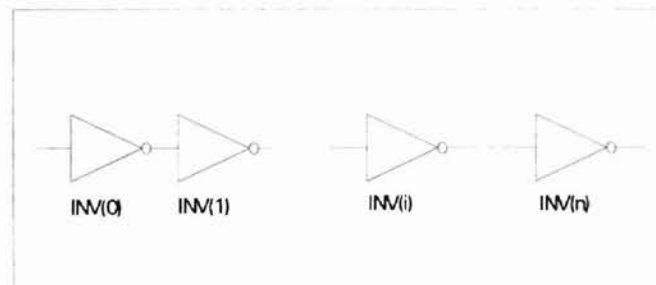


Figure 6. Series Inverters for Optimization

When a MOS transistor is replaced by a resistor and capacitor, the resistor and capacitor are represented as follows.

$$R_i = \frac{R'}{W_i}$$

$$C_i = C_{D'} \cdot W_i + C_l + C_{G'} \cdot W_{i+1}$$

Total path delay can be represented as follows.

$$\begin{aligned} \tau &= \sum_{k=0}^n \tau_k \\ &= \sum_{k=0}^n R_k \cdot C_k \\ &= \sum_{k=0}^n \frac{R'}{W_k} (C_{D'} \cdot W_k + C_l + C_{G'} \cdot W_{k+1}) \\ &= \dots + \frac{R'}{W_{i-1}} (C_{D'} \cdot W_{i-1} + C_l + C_{G'} \cdot W_i) + \frac{R'}{W_i} (C_{D'} \cdot W_i + C_l + C_{G'} \cdot W_{i+1}) + \dots \end{aligned} \quad (1)$$

Differentiate both side by W_i to find

$$\frac{\partial \tau}{\partial W_i} = \frac{R' \cdot C_{G'}}{W_{i-1}} - \frac{R'}{W_i^2} (C_l + C_{G'} \cdot W_{i+1}) \quad , 0 < i < n \quad (2)$$

From (2), the minimum delay point can be derived without any area consideration.

$$\text{Let } \frac{\partial \tau}{\partial W_i} = 0$$

$$\begin{aligned} W_{i-1} &= \frac{C_{G'} \cdot W_i^2}{C_l + C_{G'} \cdot W_{i+1}} \\ &= \frac{W_i^2}{\frac{C_l}{C_{G'}} + W_{i+1}} \end{aligned}$$

$$W_i = \sqrt{W_{i-1} \cdot \left(\frac{C_I}{C_{G'}} + W_{i+1} \right)}$$

where only a positive value of W_i has meaning.

However, this value is not economical, because this W_i does not take into consideration how much area is used.

Area Estimation

The area of an inverter can be divided into two parts. One is constant area that is essential area. The other is dependent area that varies according to the width of the MOS transistor.

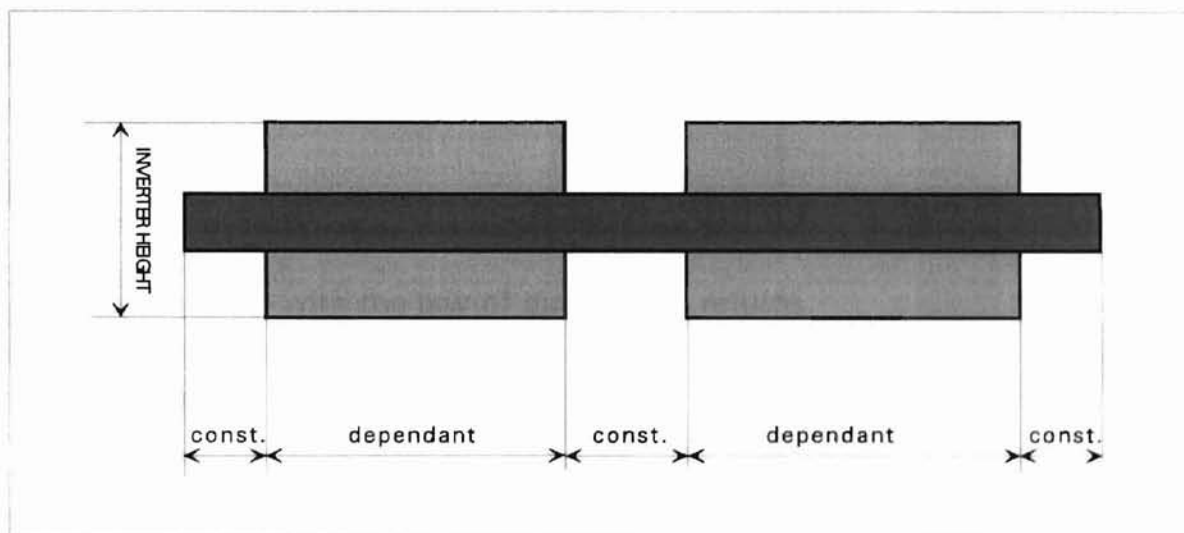


Figure 7. Area Dependency of an Inverter Layout

If the constant area of an inverter is always same for every inverter, the size of each inverter can be represented as follows.

$$A_i = A_0 + A' \cdot W_i$$

where, A_i : size of the i-th inverter

A_0 : constant area of the inverter

A' : inverter height

W_i : channel width of the i-th inverter

The total size of the series inverters is

$$\begin{aligned} A &= \sum_{k=0}^n A_k \\ &= \sum_{k=0}^n (A_0 + A' \cdot W_k) \end{aligned} \quad (3)$$

Differentiate both sides with respect to W_i

$$\frac{\partial A}{\partial W_i} = A' \quad (4)$$

From the two differential equations for area and delay, optimization process will be processed with the law of diminishing returns.

$$\begin{aligned} \frac{d\tau}{\tau} &= -K \cdot \frac{dA}{A} \\ d\tau &= -K \cdot \frac{\tau}{A} \cdot dA \\ \frac{\partial \tau}{\partial W_i} &= -K \cdot \frac{\tau}{A} \cdot \frac{\partial A}{\partial W_i} \end{aligned} \quad (5)$$

$$(2),(4) \Rightarrow (5)$$

$$\frac{R' \cdot C_G'}{W_{i-1}} - \frac{R'}{W_i^2} (C_l + C_G' \cdot W_{i+1}) = -K \cdot \tau \cdot \frac{A'}{A} \quad (6)$$

where K is an arbitrary weight factor.

In (6), A is function of W_i and has Σ inside. For calculation convenience, (6) is divided.

$$M = \tau \cdot \frac{A'}{A}$$

From (1) and (3)

$$M = \frac{A'}{\sum_{k=0}^n (A_0 + A' \cdot W_k)} \cdot \sum_{k=0}^n \frac{R'}{W_k} (C_D' \cdot W_k + C_l + C_G' \cdot W_{k+1}) \quad (7)$$

and,

$$\frac{R' \cdot C_G'}{W_{i-1}} - \frac{R'}{W_i^2} (C_l + C_G' \cdot W_{i+1}) = -K \cdot M$$

$$W_i = \sqrt{\frac{R' \cdot (C_l + C_G' \cdot W_{i+1})}{\frac{R' \cdot C_G'}{W_{i-1}} + K \cdot M}} \quad (8)$$

(7) and (8) are solved by making a guess at W_i and using the guess in (7) and the right side of (8) to produce a new guess. This procedure is iterated until the guesses for W_i converge.

Simulation Results

Using equations (7) and (8), the optimization process was performed in a 2 μ m CMOS N-well process. Typical values for the simulation are as follows. Normalized gate capacitance (C_G') was 1.68×10^{-9} F/meter. Normalized diffusion capacitance (C_D') was 0.95×10^{-9} F/meter. Interconnection capacitance (C_l) is the sum of diffusion contact capacitance, metal

capacitance and poly capacitance. In this optimization, Interconnection capacitance was assumed to be constant and typical value is 18fF. Normalized channel resistance (R') was 0.084 Ω -meter. In this optimization, N and P-type MOS transistors were assumed to have the same channel resistance. The height of the inverter (A') was set to 25 μm . The essential area for an inverter was $625 \times 10^{-12} \text{meter}^2$. The weight factor (K) for the law of diminishing returns was set to 1. Three stage series inverters were optimized. The MOS transistor size of first and last inverter were fixed to 10 μm and 300 μm each. The result is shown in table III.

| Transistor Name | Size (unit: μm) |
|-----------------|-----------------------------|
| W0 | 10 (fixed) |
| W1 | 48.4357 |
| W2 | 300 (fixed) |

Table III. 3 stage inverter optimization

The size of the MOS transistors in the middle inverter was traced with various sizes of both end inverters. In figure 8, W0 is size of the first inverter, W1 is size of the middle inverter, and W2 is size of the last inverter.

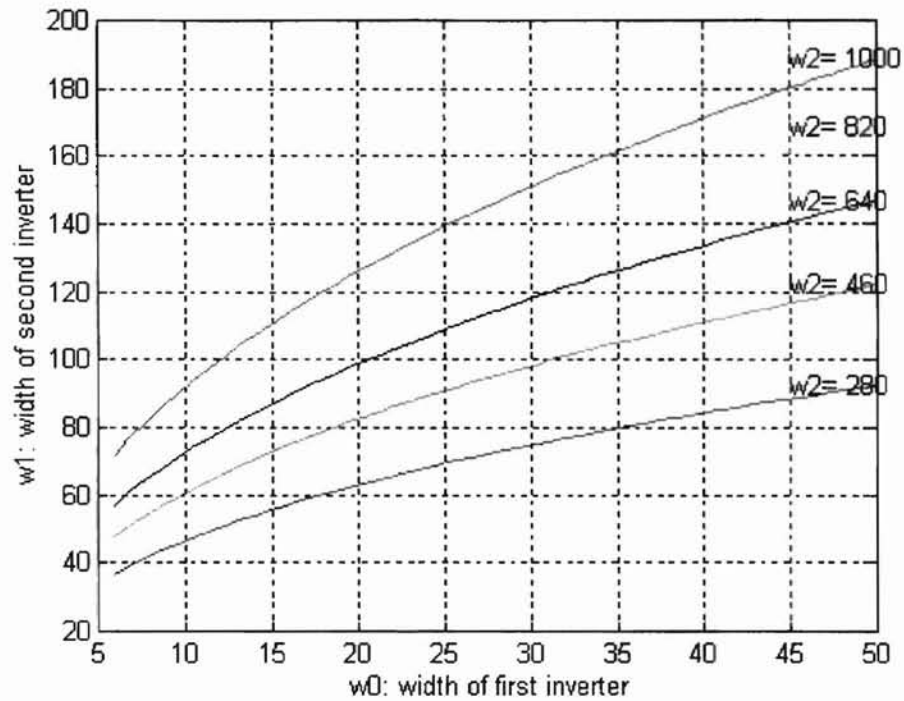


Figure 8. Transistor Sizes of 3 Stage Series Inverters (unit: μm)

In figure 9, the MOS transistor size of middle inverter was traced with several MOS transistor sizes of the last inverter and one MOS transistor size of the first inverter. '*' marks show the optimal point for each combination. The curves show delay and area relationship when the MOS transistor size of the middle inverter was changed.

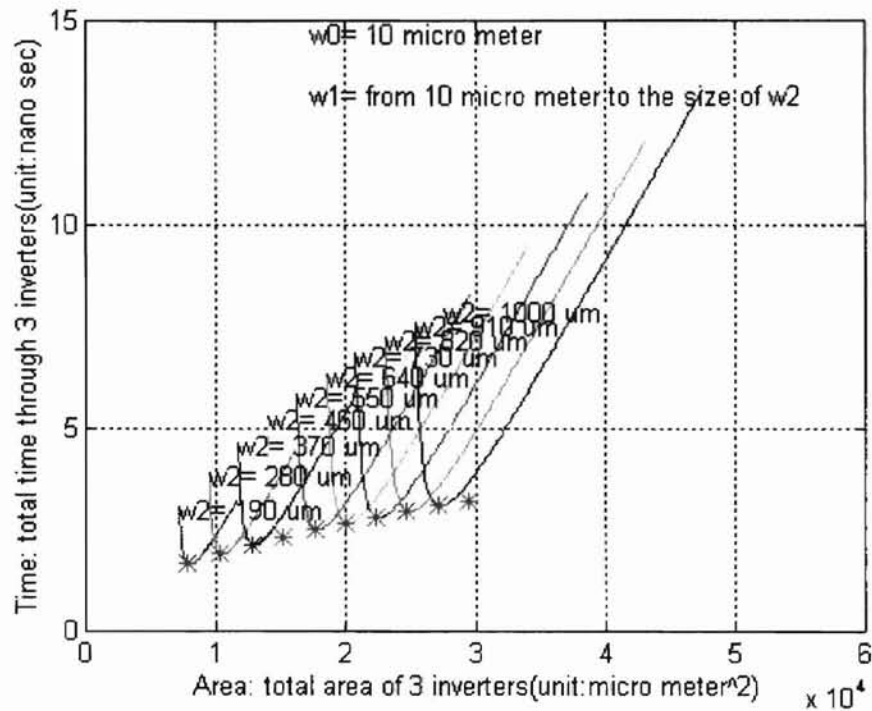


Figure 9. Delay and Area Relationship for 3 Stage Series Inverters

For a seven stage series inverters, the MOS transistor size of the first and last inverters are $10\mu\text{m}$ and $500\mu\text{m}$ respectively. Table IV shows the optimization result.

| Transistor Name | Size |
|-----------------|-----------------|
| W0 | 10 (fixed) |
| W1 | 17.6032 |
| W2 | 25.0262 |
| W3 | 34.4724 |
| W4 | 54.9984 |
| W5 | 123.4313 |
| W6 | 500.000 (fixed) |

Table IV. 7 Stage Inverter Optimization (unit: μm)

To verify that the result is optimal, the sizes of all middle inverters were forced to be changed up to $\pm 50\%$ of the optimized value, then delay and area relationship was observed.

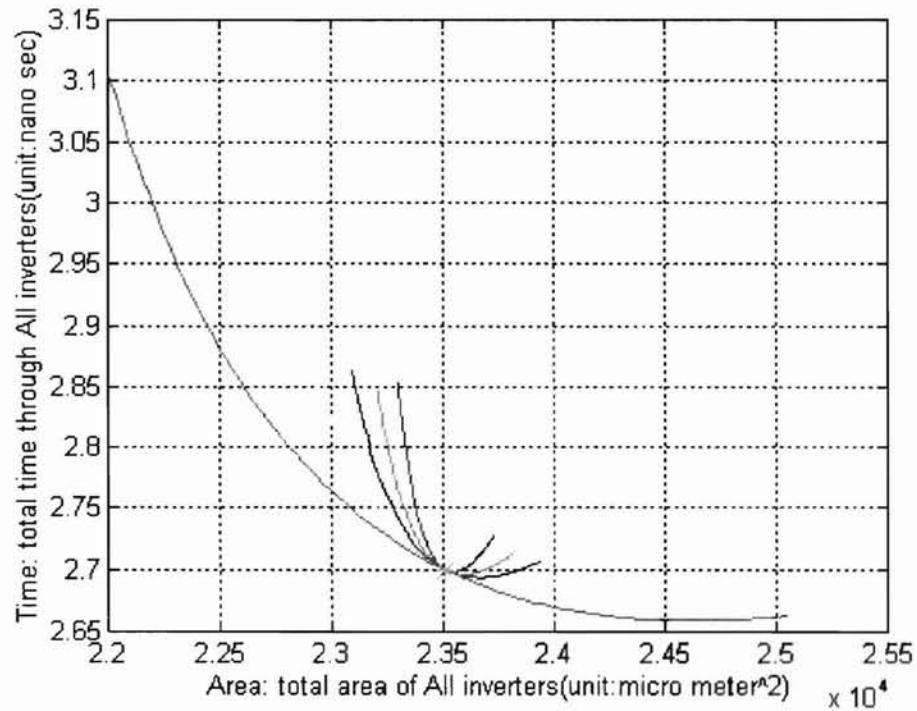


Figure 10. Optimal Point Evaluation of 7 Stage Series Inverters

In figure 10, all curves are focusing to one point.

The optimal number of stages was also investigated. With the same MOS transistor sizes on the ends, number of stages was varied from 3 to 20.

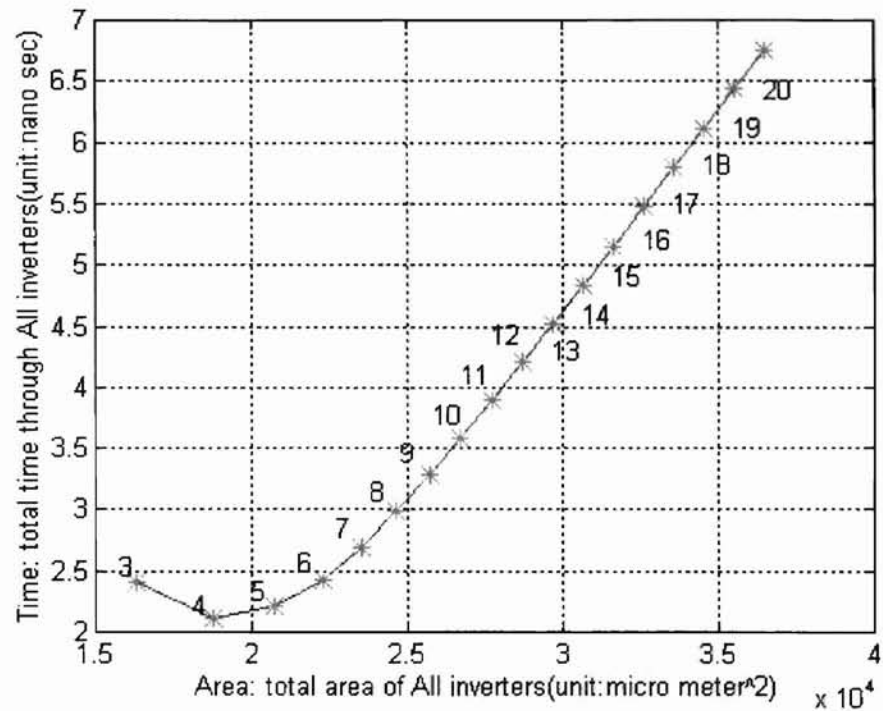


Figure 11. Variation of Number of Stage for Series Inverters

When the number of stages is four, delay is minimum. And when the number of stages is three, area is minimum. Though total area is bigger, five stage series inverters shows faster response than three stage series inverters.

CHAPTER V

TIMING AND AREA OPTIMIZATION OF BARREL SHIFTER

Introduction

With lagrangian multipliers, timing and area optimization of a barrel shifter was performed. In this chapter, the layout of the barrel shifter is divided into several standard layout blocks. That makes it easy to change the configuration as needed. The program was named **transize** and written in **C** language. Final results are compared to the results from the program **iCONTRAST** which was developed by Sapatnekar et al.[Sapatnekar, 1993].

Overview of Barrel Shifter

The basic component of the barrel shifter is a 2 to 1 multiplexer. These multiplexers are arrayed as shown in figure 12 and 13. In the layout, inputs are located at the left side, outputs are at the right side, and control signals are at the top side of the layout. All transistors are placed with their widths in the horizontal direction, so changing the transistor size of each stage is easy to accomplish.

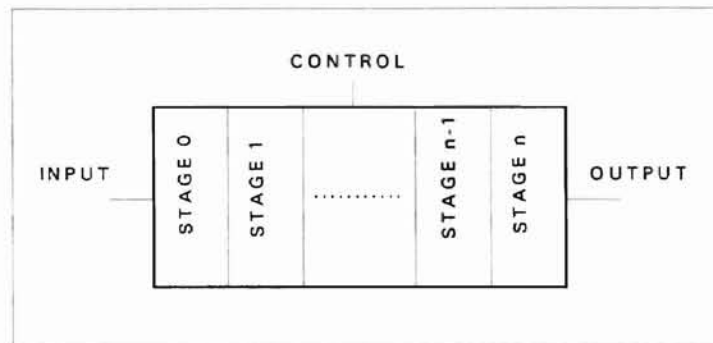


Figure 12. Layout Scheme of Barrel Shifter

The schematic of a barrel shifter is shown in simplified form in figure 13. A transmission gate does not have signal driving capability. It is only a resistive switch. Therefore, the signal should be amplified with proper buffer insertion between multiplexers.

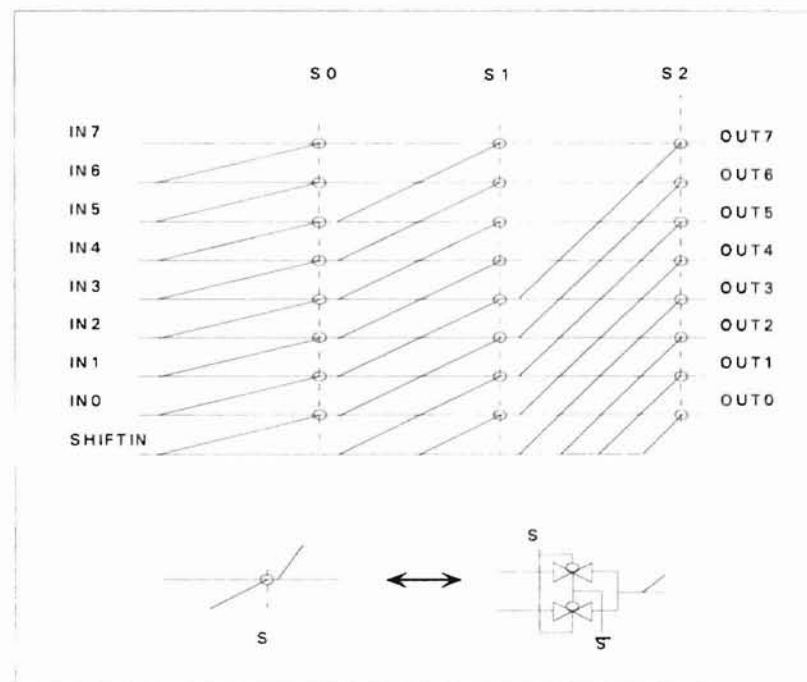
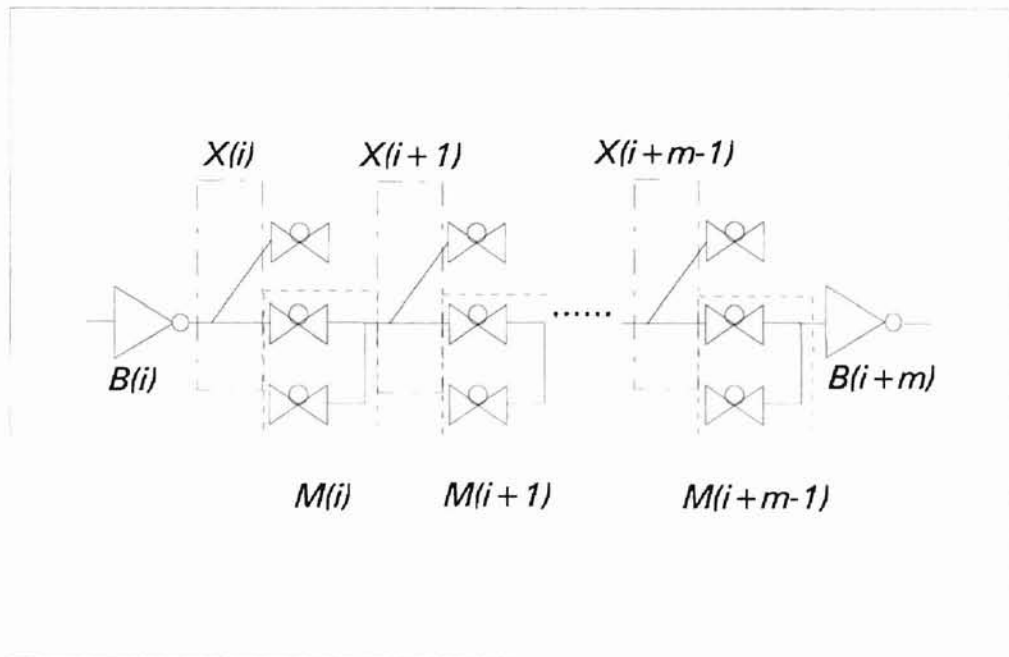


Figure 13. Simplified Schematic of Barrel Shifter

Layout Parameters

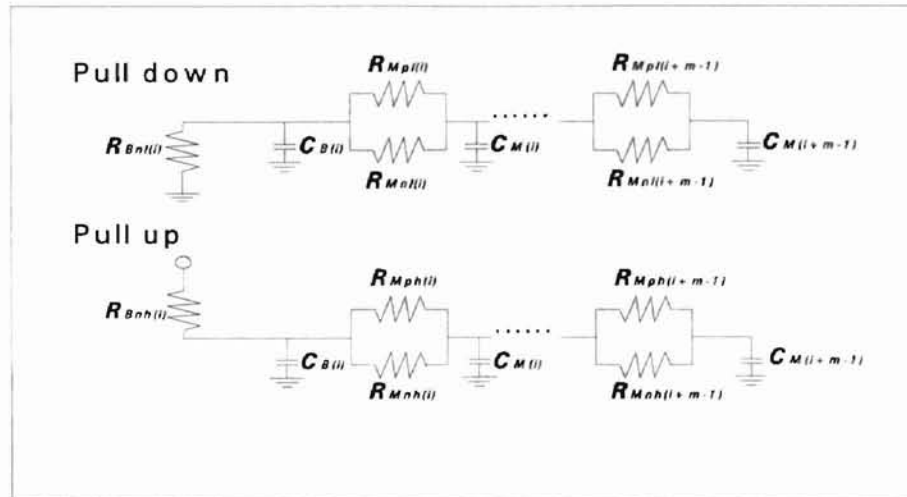
A $2\mu\text{m}$ CMOS N-well process was used. Standard layout blocks are an inverting buffer, a transmission gate multiplexer, and cross over cells which connect multiplexers. Figure 14 shows the function of the standard layout blocks. $B(i)$ is the i -th block of inverting buffers, $X(i)$ is the i -th block of cross over cells, and $M(i)$ is the i -th block of multiplexers.



m : number of multiplexers

Figure 14. Function of Standard Layout Blocks

For each multiplexer, only one transmission gate is turned on. The RC equivalent circuit is showed in figure 15.



| | |
|--------------|--|
| $R_{Bnl}(i)$ | n-tr on-resistance of inverting buffer |
| $R_{Bph}(i)$ | p-tr on-resistance of inverting buffer |
| $R_{Mpl}(i)$ | p-tr on-resistance of multiplexer when low data passing |
| $R_{Mph}(i)$ | p-tr on-resistance of multiplexer when high data passing |
| $R_{Mnl}(i)$ | n-tr on-resistance of multiplexer when low data passing |
| $R_{Mnh}(i)$ | n-tr on-resistance of multiplexer when high data passing |
| $C_B(i)$ | total capacitance connected to inverting buffer output |
| $C_M(i)$ | total capacitance connected to multiplexer output |

Figure 15. RC Equivalent Circuit for Barrel Shifter

The delay of each case can be estimated with the Elmore delay formula,

$$T_{delay} = \sum_k R_{ke} C_k .$$

$$\tau_{rise} = R_{Bph}(i) \cdot C_B(i) + (R_{Bph}(i) + R_{Mnh}(i) || R_{Mph}(i)) \cdot C_M(i) + \dots$$

$$+ (R_{Bph}(i) + R_{Mnh}(i) || R_{Mph}(i) + \dots + R_{Mnh}(i+m-1) || R_{Mph}(i+m-1)) \cdot C_M(i+m-1)$$

$$\tau_{fall} = R_{Bph}(i) \cdot C_B(i) + (R_{Bph}(i) + R_{Mnh}(i) || R_{Mph}(i)) \cdot C_M(i) + \dots$$

$$+ (R_{Bph}(i) + R_{Mnh}(i) || R_{Mph}(i) + \dots + R_{Mnh}(i+m-1) || R_{Mph}(i+m-1)) \cdot C_M(i+m-1)$$

The capacitance driven by buffer is calculated in this equation.

$$C_B(i) = C_{OB}(i) + C_X(i) + C_{IOM}(i) + C_{IM}(i)$$

where $C_{OB}(i)$: inverting buffer output capacitance
mainly diffusion capacitance

$C_X(i)$: cross over capacitance which is metal capacitance between inverting buffer and multiplexer
 $C_{IOM}(i), C_{IIM}(i)$: input capacitance of multiplexer
 diffusion capacitance and interconnection capacitance
 difference of two parameters is interconnection capacitance.
 $C_{IOM}(i) = C_{IOM} + C'_{IMn} \cdot W_{Mn}(i) + C'_{IMp} \cdot W_{Mp}(i)$
 $C_{IIM}(i) = C_{IIM} + C'_{IMn} \cdot W_{Mn}(i) + C'_{IMp} \cdot W_{Mp}(i)$

Each multiplexer drives this amount of capacitance.

$$C_M(i) = C_{OM}(i) + C_X(i+1) + C_{IOM}(i+1) + C_{IIM}(i+1)$$

where $C_{OM}(i)$: output capacitance of multiplexer.

The last multiplexer of each stage drives this amount of capacitance.

$$C_M(i+m-1) = C_{OM}(i+m-1) + C_{IB}(i+m)$$

where $C_{IB}(i)$: gate capacitance of inverting buffer and interconnection capacitance

The area equation for barrel shifter is as follows

$$\begin{aligned}
 A &= \sum A(i) \\
 &= \sum (A_B(i) + A_X(i) + A_M(i))
 \end{aligned}$$

where $A_B(i)$: Inverting buffer area
 $A_B(i) = A_{BI} + A' \cdot (W_{Bn}(i) + W_{Bp}(i))$
 $A_M(i)$: multiplexer area
 $A_M(i) = A_{MI} + A' \cdot (W_{Mn}(i) + W_{Mp}(i))$
 $A_X(i)$: cross over area, independent of transistor size. different cross over cells are needed for each stage.

Circuit Optimization

There are many signal paths in the barrel shifter. Among them, the data path is the main concern. Though data can flow through different paths, the capacitive and resistive loads are always the same. From the low order bit to the high order bit, the capacitive loads decrease. But, this slight difference can be ignored. Therefore, the worst case data path can be chosen from low order bit. The optimization results with the worst data path can be applied to the whole circuit.

For a sample data path, equations for optimization will be derived. In this optimization, the first and last inverting buffer have fixed transistor size. For calculation simplicity, all outputs of multiplexers are amplified by an inverting buffer. Figure 16 shows a sample worst case data path of the barrel shifter.

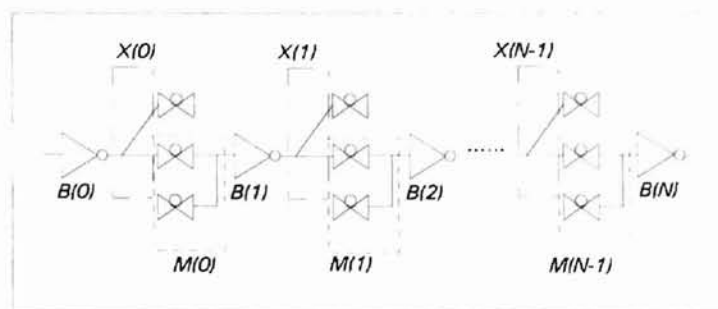


Figure 16. Worst Case Data Path in Barrel Shifter

There are N multiplexers and $N + 1$ inverting buffers. The top and bottom transmission gates are off and the middle transmission gates are on. The top

and bottom transmission gates are considered as a capacitive load for their diffusion capacitance. The sizes of the top and bottom transmission gates are equal to size of the middle transmission gates.

Delay between the input and output of the path must be determined for two uses.

The input rising delay is

$$\begin{aligned}\tau_r &= \tau_{f0} + \tau_{r1} + \tau_{f2} + \tau_{r3} + \tau_{f4} + \tau_{r5} + \tau_{f6} + \tau_{r7} + \dots \\ &= \sum_{i=odd} \tau_r(i) + \sum_{i=even} \tau_f(i)\end{aligned}$$

The input falling delay is

$$\begin{aligned}\tau_f &= \tau_{r0} + \tau_{f1} + \tau_{r2} + \tau_{f3} + \tau_{r4} + \tau_{f5} + \tau_{r6} + \tau_{f7} + \dots \\ &= \sum_{i=odd} \tau_f(i) + \sum_{i=even} \tau_r(i)\end{aligned}$$

For each stage, rise and fall delays are represented as follows.

$$\tau_r(i) = R_{Bph}(i) \cdot C_B(i) + (R_{Bph}(i) + R_{Mnh}(i) || R_{Mph}(i)) \cdot C_M(i)$$

$$\tau_f(i) = R_{Bnl}(i) \cdot C_B(i) + (R_{Bnl}(i) + R_{Mnl}(i) || R_{Mpl}(i)) \cdot C_M(i)$$

where $C_B(i) = C_{OB}(i) + C_X(i) + C_{IOM}(i) + C_{IM}(i)$

$$C_M(i) = C_{OM}(i) + C_{IB}(i+1)$$

Using a lagrangian multiplier,

$$f = \tau_r + \alpha \cdot (\tau_f - \tau_r) = (1 - \alpha) \cdot \tau_r + \alpha \cdot \tau_f$$

where $0 < \alpha < 1$ to avoid maximizing delays when minimizing f .

$$f = \sum_{i=odd} [(1-\alpha) \cdot \tau_r(i) + \alpha \cdot \tau_f(i)] + \sum_{i=even} [\alpha \cdot \tau_r(i) + (1-\alpha) \cdot \tau_f(i)]$$

$$= \sum_i [(1 - q(i)) \cdot \tau_r(i) + q(i) \cdot \tau_f(i)]$$

$$\text{where } q(i) = \begin{cases} \alpha & i = \text{odd} \\ 1 - \alpha & i = \text{even} \end{cases}$$

Minimizing f gives

$$0 = \frac{\partial f}{\partial W_{Bn}(i)} = (1 - q(i-1)) \cdot (R_{Bph}(i-1) + R_{Mnh}(i-1) \| R_{Mph}(i-1)) \cdot C'_{IBn} +$$

$$q(i-1) \cdot (R_{Bnl}(i-1) + R_{Mnl}(i-1) \| R_{Mpl}(i-1)) \cdot C'_{IBn} +$$

$$(1 - q(i)) \cdot R_{Bph}(i-1) \cdot C'_{OBn} +$$

$$q(i) \cdot \left[R_{Bnl}(i) \cdot C'_{IBn} - \frac{R_{Bnl}(i)^2}{R'_{nl}} (C_B(i) + C_M(i)) \right]$$

$$0 = \frac{\partial f}{\partial W_{Bp}(i)} = (1 - q(i-1)) \cdot (R_{Bph}(i-1) + R_{Mnh}(i-1) \| R_{Mph}(i-1)) \cdot C'_{IBp} +$$

$$q(i-1) \cdot (R_{Bnl}(i-1) + R_{Mnl}(i-1) \| R_{Mpl}(i-1)) \cdot C'_{IBp} +$$

$$q(i) \cdot R_{Bnl}(i-1) \cdot C'_{OBp} +$$

$$(1 - q(i)) \cdot \left[R_{Bph}(i) \cdot C'_{IBp} - \frac{R_{Bph}(i)^2}{R'_{ph}} (C_B(i) + C_M(i)) \right]$$

$$0 = \frac{\partial f}{\partial W_{Mn}(i)} = (1 - q(i)) \cdot \left[R_{Bph}(i) \cdot (2C'_{IMn} + C'_{OMn}) + \right.$$

$$\left. (R_{Mnh}(i) \| R_{Mph}(i)) \cdot C'_{OMn} - \frac{(R_{Mnh}(i) \| R_{Mph}(i))^2}{R'_{nh}} \cdot C_M(i) \right] +$$

$$q(i) \cdot \left[R_{Bnl}(i) \cdot (2C'_{IMn} + C'_{OMn}) + \right.$$

$$\begin{aligned}
& (R_{Mnl}(i) \parallel R_{Mpl}(i)) \cdot C'_{OMn} - \frac{(R_{Mnl}(i) \parallel R_{Mpl}(i))^2}{R'_{nl}} \cdot C_M(i) \\
0 = \frac{\partial f}{\partial W_{Mp}(i)} = & (1 - q(i)) \cdot \left[R_{Bph}(i) \cdot (2C'_{lMp} + C'_{oMp}) + \right. \\
& \left. (R_{Mnh}(i) \parallel R_{Mph}(i)) \cdot C'_{oMp} - \frac{(R_{Mnh}(i) \parallel R_{Mph}(i))^2}{R'_{ph}} \cdot C_M(i) \right] + \\
& q(i) \cdot \left[R_{Bnl}(i) \cdot (2C'_{lMp} + C'_{oMp}) + \right. \\
& \left. (R_{Mnl}(i) \parallel R_{Mpl}(i)) \cdot C'_{oMp} - \frac{(R_{Mnl}(i) \parallel R_{Mpl}(i))^2}{R'_{pl}} \cdot C_M(i) \right] \\
0 = \frac{\partial f}{\partial \alpha} = & \tau_f - \tau_r
\end{aligned}$$

These equations are for delay minimization only. Area equations need to be included. In this optimization, delay will be minimized within specified area A_0 . Using lagrangian multipliers,

$$g = \tau_r + \alpha_t \cdot (\tau_f - \tau_r) + \alpha_A \cdot (A - A_0)$$

where

α_t is lagrangian multiplier for timing optimization

α_A is lagrangian multiplier for area optimization

A is total area

A_0 is target area

Differentiate both sides with respect to $W(i)$

$$0 = \frac{\partial g}{\partial W(i)} = \frac{\partial f}{\partial W(i)} + \alpha_A \cdot \frac{\partial A}{\partial W(i)}$$

$$0 = \frac{\partial g}{\partial \alpha_f} = \tau_f - \tau_r$$

$$0 = \frac{\partial g}{\partial \alpha_A} = A - A_0$$

Area equation is

$$A = \sum A(i) = \sum (A_B(i) + A_X(i) + A_M(i)) = A_{constant} + A' \cdot \sum W(i)$$

Differentiate both sides with respect to $W(i)$

$$\frac{\partial A}{\partial W(i)} = \frac{\partial}{\partial W(i)} (A_{constant} + A' \cdot \sum W(i)) = A'$$

With this equation,

$$\begin{aligned} 0 = \frac{\partial g}{\partial W_{Bn}(i)} = & (1 - q(i-1)) \cdot (R_{Bph}(i-1) + R_{Mnh}(i-1) || R_{Mph}(i-1)) \cdot C'_{IBn} + \\ & q(i-1) \cdot (R_{Bnl}(i-1) + R_{Mnl}(i-1) || R_{Mpl}(i-1)) \cdot C'_{IBn} + \\ & (1 - q(i)) \cdot R_{Bph}(i-1) \cdot C'_{OBn} + \\ & q(i) \cdot \left[R_{Bnl}(i) \cdot C'_{IBn} - \frac{R_{Bnl}(i)^2}{R'_{nl}} (C_B(i) + C_M(i)) \right] + \alpha_A \cdot A' \end{aligned}$$

$$\begin{aligned} 0 = \frac{\partial g}{\partial W_{Bp}(i)} = & (1 - q(i-1)) \cdot (R_{Bph}(i-1) + R_{Mnh}(i-1) || R_{Mph}(i-1)) \cdot C'_{IBp} + \\ & q(i-1) \cdot (R_{Bnl}(i-1) + R_{Mnl}(i-1) || R_{Mpl}(i-1)) \cdot C'_{IBp} + \\ & q(i) \cdot R_{Bnl}(i-1) \cdot C'_{OBp} + \\ & (1 - q(i)) \cdot \left[R_{Bph}(i) \cdot C'_{IBp} - \frac{R_{Bph}(i)^2}{R'_{ph}} (C_B(i) + C_M(i)) \right] + \alpha_A \cdot A' \end{aligned}$$

$$0 = \frac{\partial g}{\partial W_{Mn}(i)} = (1 - q(i)) \cdot \left[R_{Bph}(i) \cdot (2C'_{IMn} + C'_{OMn}) + \right. \\ \left. (R_{Mnh}(i) || R_{Mph}(i)) \cdot C'_{OMn} - \frac{(R_{Mnh}(i) || R_{Mph}(i))^2}{R'_{nh}} \cdot C_M(i) \right] + \\ q(i) \cdot \left[R_{Bnl}(i) \cdot (2C'_{IMn} + C'_{OMn}) + \right. \\ \left. (R_{Mnl}(i) || R_{Mpl}(i)) \cdot C'_{OMn} - \frac{(R_{Mnl}(i) || R_{Mpl}(i))^2}{R'_{nl}} \cdot C_M(i) \right] +$$

$$\alpha_A \cdot A'$$

$$0 = \frac{\partial g}{\partial W_{Mp}(i)} = (1 - q(i)) \cdot \left[R_{Bph}(i) \cdot (2C'_{IMp} + C'_{OMP}) + \right. \\ \left. (R_{Mnh}(i) || R_{Mph}(i)) \cdot C'_{OMP} - \frac{(R_{Mnh}(i) || R_{Mph}(i))^2}{R'_{ph}} \cdot C_M(i) \right] + \\ q(i) \cdot \left[R_{Bnl}(i) \cdot (2C'_{IMp} + C'_{OMP}) + \right. \\ \left. (R_{Mnl}(i) || R_{Mpl}(i)) \cdot C'_{OMP} - \frac{(R_{Mnl}(i) || R_{Mpl}(i))^2}{R'_{pl}} \cdot C_M(i) \right] +$$

$$\alpha_A \cdot A'$$

When α_A is zero, the area is not constrained and the delay is a minimum, The fixed size for the first and last inverter keep the area finite. If α_A is negative, that means the target area is bigger than the minimum delay area. In that case, the target area may be reduced or set α_A to zero. When α_A is positive, the area is less than the minimum delay area and the delay is

increased. Thus, a positive α_A allows a trade-off between increased delay and reduced area.

Optimization Results

In this optimization, the barrel shifter has full CMOS transmission gates and inverting buffers for every two stages of transmission gates to amplify the data signal. Layout was made with standard layout blocks. That makes it easy to handle different barrel shifter configurations. All transistors are oriented in one direction. Therefore, variation of transistor sizes affects the size of only one side of barrel shifter. Figure 17 shows the schematic of an 8 bit barrel shifter.

For each group, the same value of optimized width is used. In other words, WBp(0) and WBn(0) in TABLE V are the PMOS and NMOS transistor widths of the B(0) group transistors.

| | | | | | | | |
|---------------------|-------|--------|-------|-----------------------------|---|--------|-------|
| WBp(0) | 4 | WBn(0) | 4 | WMp(1) | 4 | WMn(1) | 5.307 |
| WBp(1) | 9.261 | WBn(1) | 6.712 | WMp(2) | 4 | WMn(2) | 4 |
| WBp(2) | 8.612 | WBn(2) | 5.543 | WMp(3) | 4 | WMn(3) | 4 |
| WBp(3) | 30 | WBn(3) | 20 | | | | |
| Total Area | | | | 6732.883789 μm^2 | | | |
| Total Rising Delay | | | | 5.811049 nano seconds | | | |
| Total Falling Delay | | | | 5.806156 nano seconds | | | |

Table V. 8 bit Barrel Shifter **transize** Result (unit: μm)

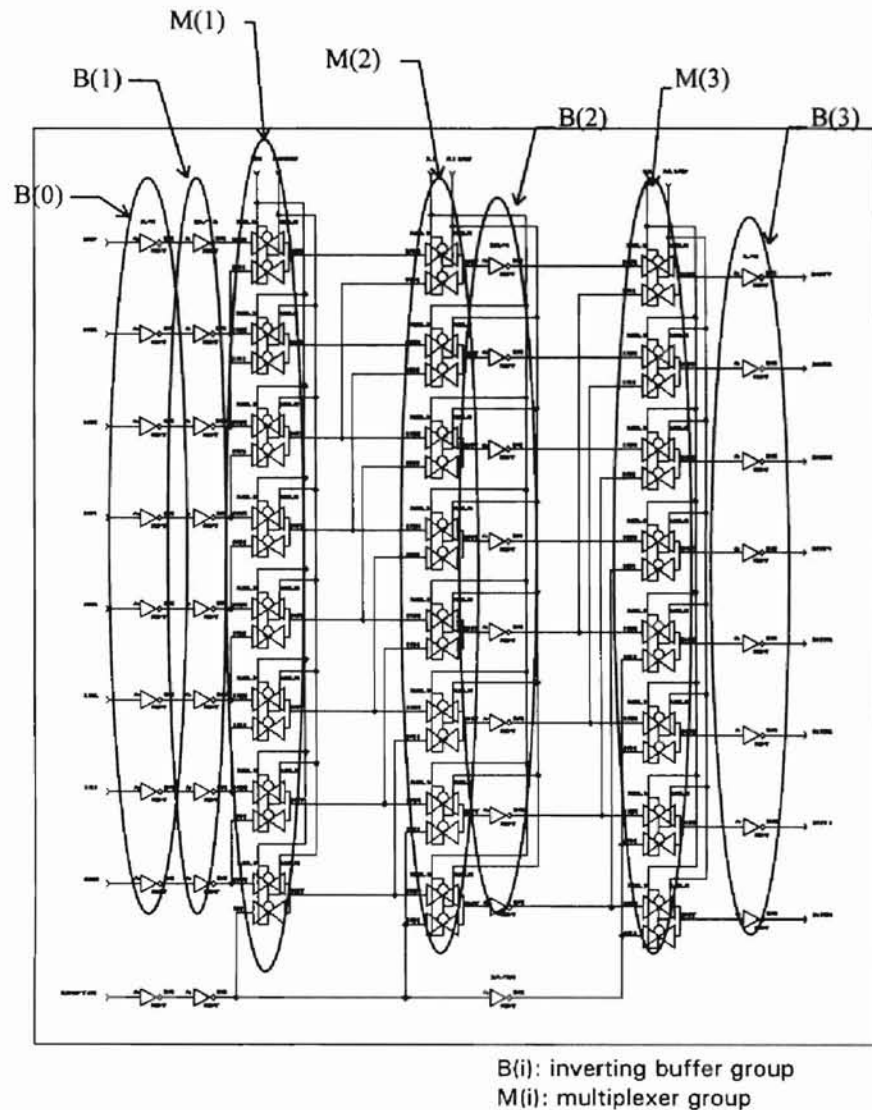


Figure 17. Schematic of 8 bit Barrel Shifter

Within a given area, optimization process is looking for the fastest combination of transistor sizes. Process parameters for program **transize** are as follows. These values are from 2 μ m CMOS N-well process parameters and the standard layout blocks for the barrel shifter design. These parameters depend on both the process and the layout style.

| | | |
|-------|---|--------|
| CIB | buffer input constant cap | 0.005 |
| CIBn | buffer input cap coefficient for Wn | 0.0018 |
| CIBp | buffer input cap coefficient for Wp | 0.0018 |
| COB | buffer output constant cap | 0.012 |
| COBn | buffer output cap coefficient for Wn | 0.0026 |
| COBp | buffer output cap coefficient for Wp | 0.0027 |
| CI0M | mux input 0 constant cap | 0.012 |
| CI1M | mux input 1 constant cap | 0.011 |
| CIMn | mux input cap coefficient for Wn | 0.0026 |
| CIMp | mux input cap coefficient for Wp | 0.0027 |
| COM | mux output constant cap | 0.006 |
| COMn | mux output cap coefficient for Wn | 0.0022 |
| COMp | mux output cap coefficient for Wp | 0.0025 |
| Rnl | low signal resistance coefficient for 1/Wn | 26 |
| Rnh | high signal resistance coefficient for 1/Wn | 104 |
| Rph | low signal resistance coefficient for 1/Wp | 56 |
| Rpl | high signal resistance coefficient for 1/Wp | 156 |
| AIB | buffer constant area | 480 |
| AIM | mux constant area | 552 |
| H | cell height | 24 |
| Cx(0) | cross over cap | 0 |
| Cx(1) | cross over cap | 0.006 |
| Cx(2) | cross over cap | 0.013 |
| Cx(3) | cross over cap | 0.025 |
| Ax(0) | cross over area | 0 |
| Ax(1) | cross over area | 384 |
| Ax(2) | cross over area | 576 |
| Ax(3) | cross over area | 1152 |

Table VI. **transize** Process Parameters

Comparison to other work

A program developed at the University of Illinois, **iCONTRAST** finds critical delay paths using **PERT** and the delay graph, and uses the convex optimization method. **PERT** was originally developed for schedule management of big projects. Circuit designers employed this technique to

delay estimation [Kirkpatrick, 1966]. **iCONTRAST** converts the given CMOS circuit to undirected graph. Gate channel of each MOS transistor is replaced by a net, and the connections of MOS transistors are considered as nodes. Using this graph, **PERT** finds overall delay estimation like schedule management. **iCONTRAST** is a general circuit optimization tool, but **transize** is dedicated to barrel shifter optimization. Therefore, **iCONTRAST** gives individual transistor sizes. In designs with the standard layout blocks, **iCONTRAST** may have lower efficiency. Most of timing and area optimization tools can not consider layout skill or variation of layout scheme, so direct comparison between the two tools is difficult. Through the comparisons of computing time and tendency of results, relative comparison is shown.

32 bit barrel shifter with **transize** is shown in table VII and figure 18.

| Area | Rising | Falling |
|----------|---------|---------|
| 15512.17 | 11.0652 | 10.9000 |
| 15857.31 | 9.1670 | 9.1417 |
| 16620.21 | 7.6333 | 7.7724 |
| 17151.37 | 7.2467 | 7.3332 |
| 17303.75 | 7.2155 | 7.2070 |
| 18157.72 | 6.8295 | 7.0140 |
| 18648.52 | 6.8045 | 6.8497 |
| 19031.86 | 6.7091 | 6.8406 |

Table VII. **transize** Result of 32 bit Barrel Shifter

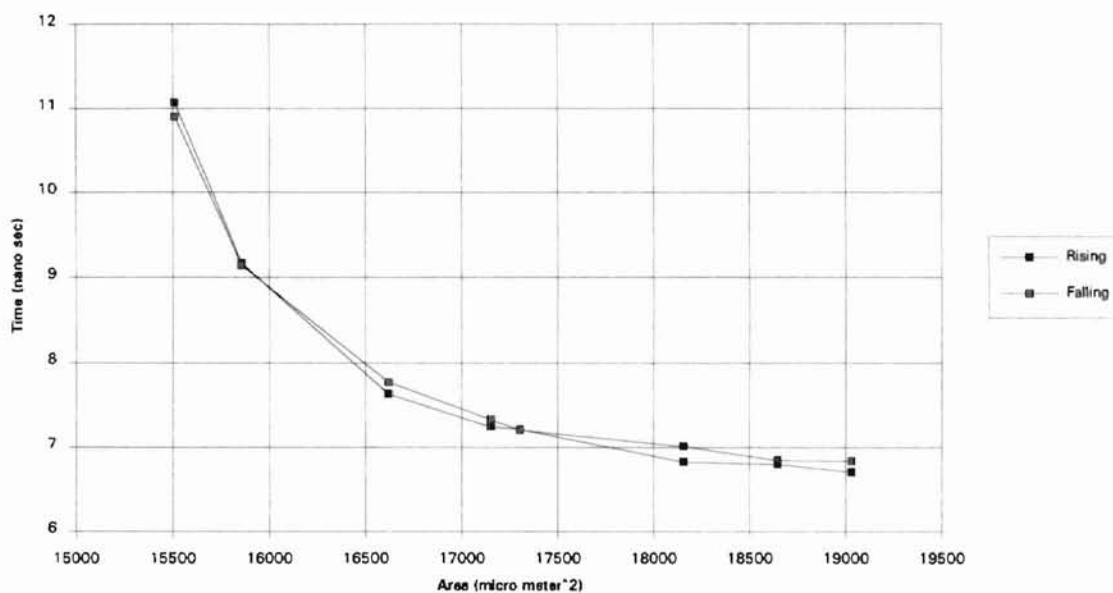


Figure 18. Area and Timing Relationship of 32 bit Barrel Shifter after **transize** Optimization

8 bit barrel shifter with **transize** is shown in table VIII and figure 19.

| Area | Rising | Falling |
|---------|--------|---------|
| 6383.54 | 6.5415 | 6.5248 |
| 6383.54 | 6.5415 | 6.5248 |
| 6406.63 | 6.3008 | 6.3413 |
| 6712.27 | 5.1399 | 5.1377 |
| 6861.72 | 4.8605 | 4.8876 |
| 7050.78 | 4.6667 | 4.6477 |
| 7317.75 | 4.4769 | 4.4842 |
| 7418.19 | 4.4467 | 4.4315 |
| 7972.75 | 4.3000 | 4.3349 |
| 7942.09 | 4.2875 | 4.3517 |
| 8116.04 | 4.3295 | 4.2883 |

Table VIII. **transize** Result of 8 bit Barrel Shifter

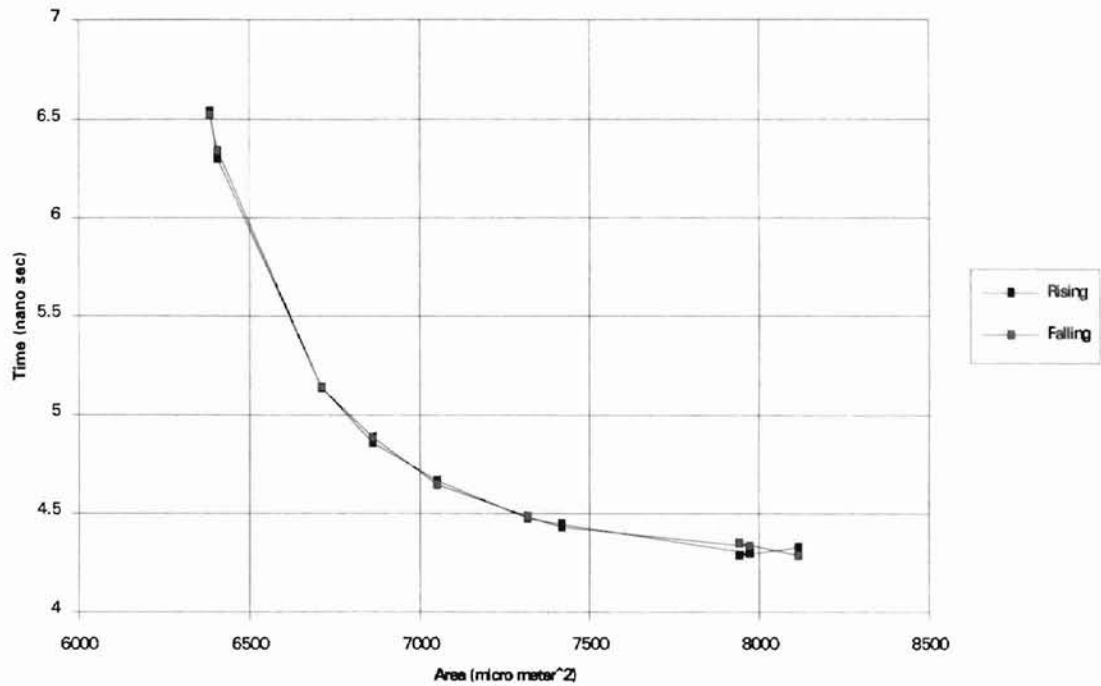


Figure 19. Area and Timing Relationship of 8 bit Barrel Shifter after **transize** Optimization

The 8 bit barrel shifter with **iCONTRAST** is shown in table IX and figure 20.

| Area | Td |
|---------|-------|
| 1E + 10 | 26.90 |
| 1E + 10 | 12.00 |
| 2227.96 | 12.20 |
| 876.94 | 15.50 |
| 694.63 | 17.20 |
| 691.93 | 19.40 |
| 664.85 | 23.10 |
| 638.33 | 22.70 |
| 625.97 | 25.50 |

Table IX. **iCONTRAST** Result of 8 bit Barrel Shifter

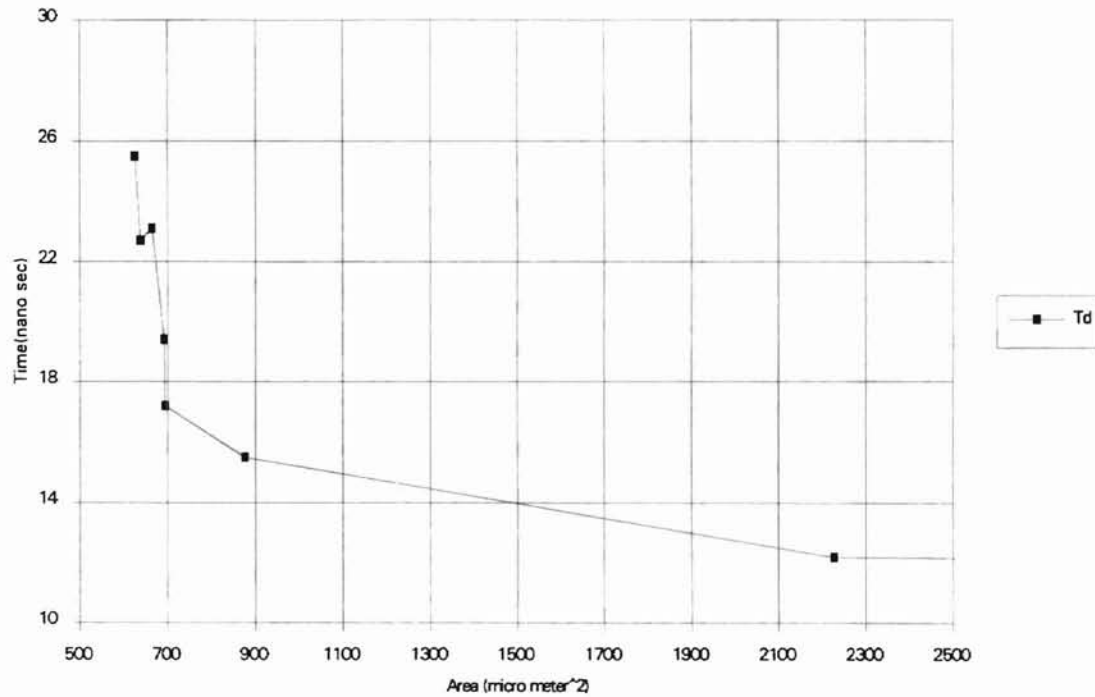


Figure 20. Area and Timing Relationship of 8 bit Barrel Shifter after
iCONTRAST Optimization

After optimization, **iCONTRAST** produced irregular size like 29 μm , 13 μm and 20 μm when most of the other transistor sizes are 4 μm as shown in figure 21. It is not an efficient layout. In each stage, the size of the load is almost the same for every transistor. Therefore, almost the same sizes of transistors for each stage are expected. One possible reason this happens is the wrong critical delay path searching on transmission gate circuit. Actually, the transmission gate has a bi-directional characteristic, but, in this barrel shifter, only one of two transmission gates is on at any time and data signals flow only one direction.

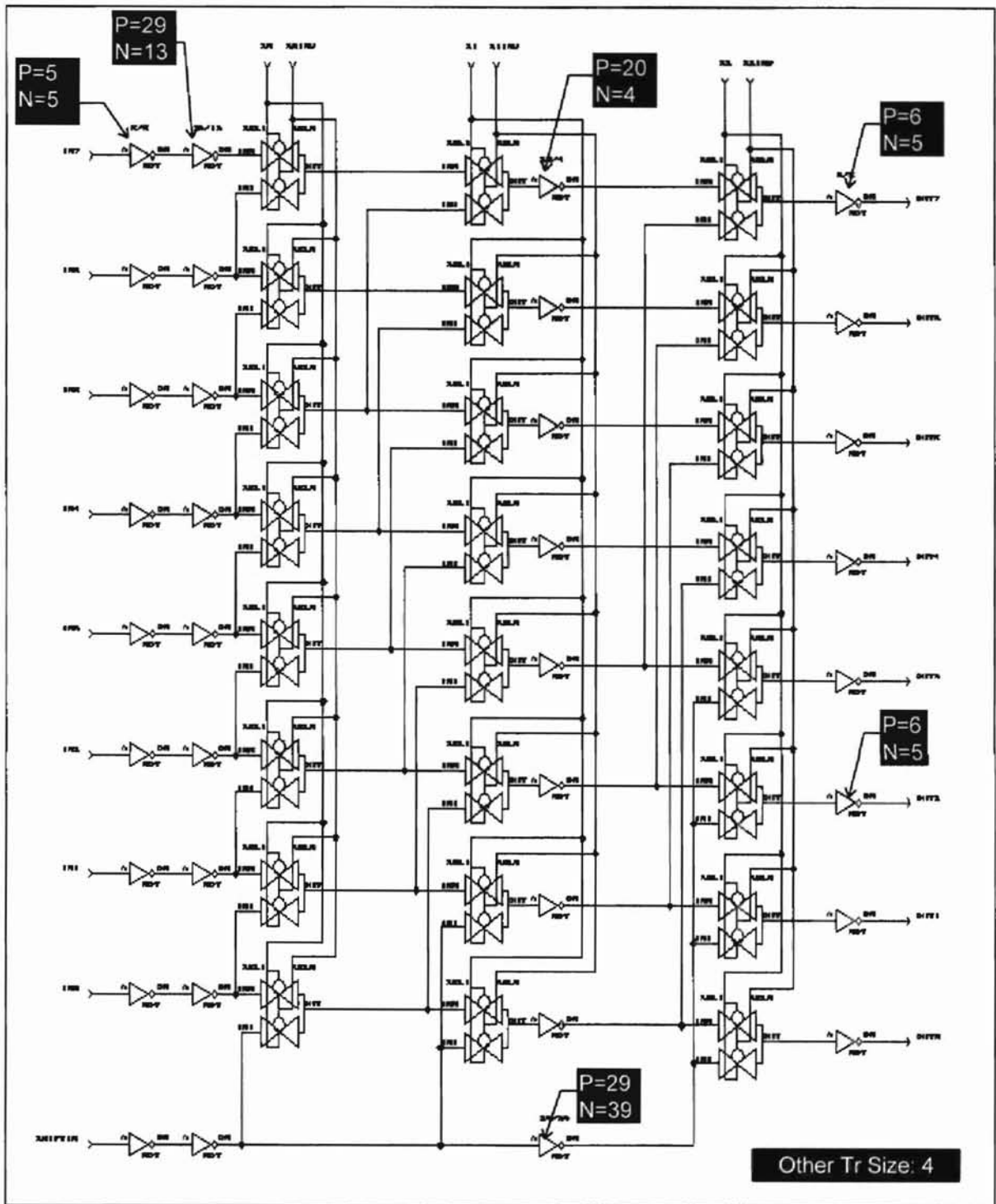


Figure 21. Schematic of 8 bit Barrel Shifter with iCONTRAST Result

iCONTRAST might consider backward signal passing on transmission gate. To verify this assumption, the transmission gates are replaced by clocked inverters.

| Area | Td |
|---------|-------|
| 2738.88 | 7.94 |
| 1363.37 | 9.99 |
| 1037.46 | 12.00 |
| 960.39 | 14.00 |
| 934.85 | 15.90 |
| 928.89 | 16.80 |
| 928.89 | 16.80 |
| 928.89 | 16.80 |
| 928.89 | 16.80 |

Table X. Clocked Inverter type 8 bit Barrel Shifter Result

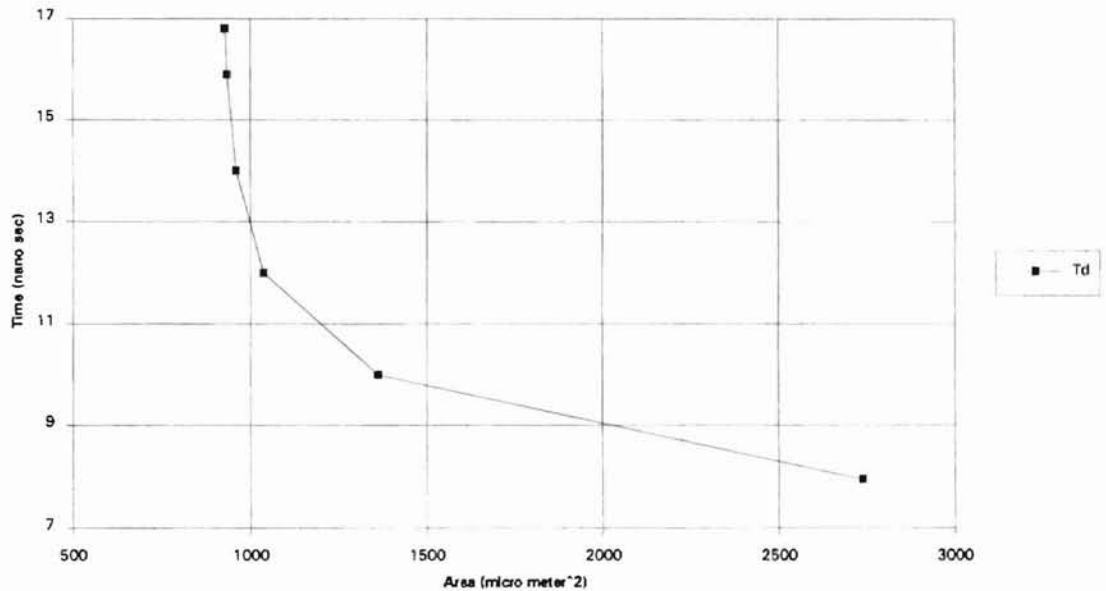


Figure 22. Area and Timing Relationship of Clocked Inverter Type Barrel Shifter after **iCONTRAST** Optimization

From the result of clocked inverter circuit, transistor sizes are almost regular for each stage shown in figure 23.

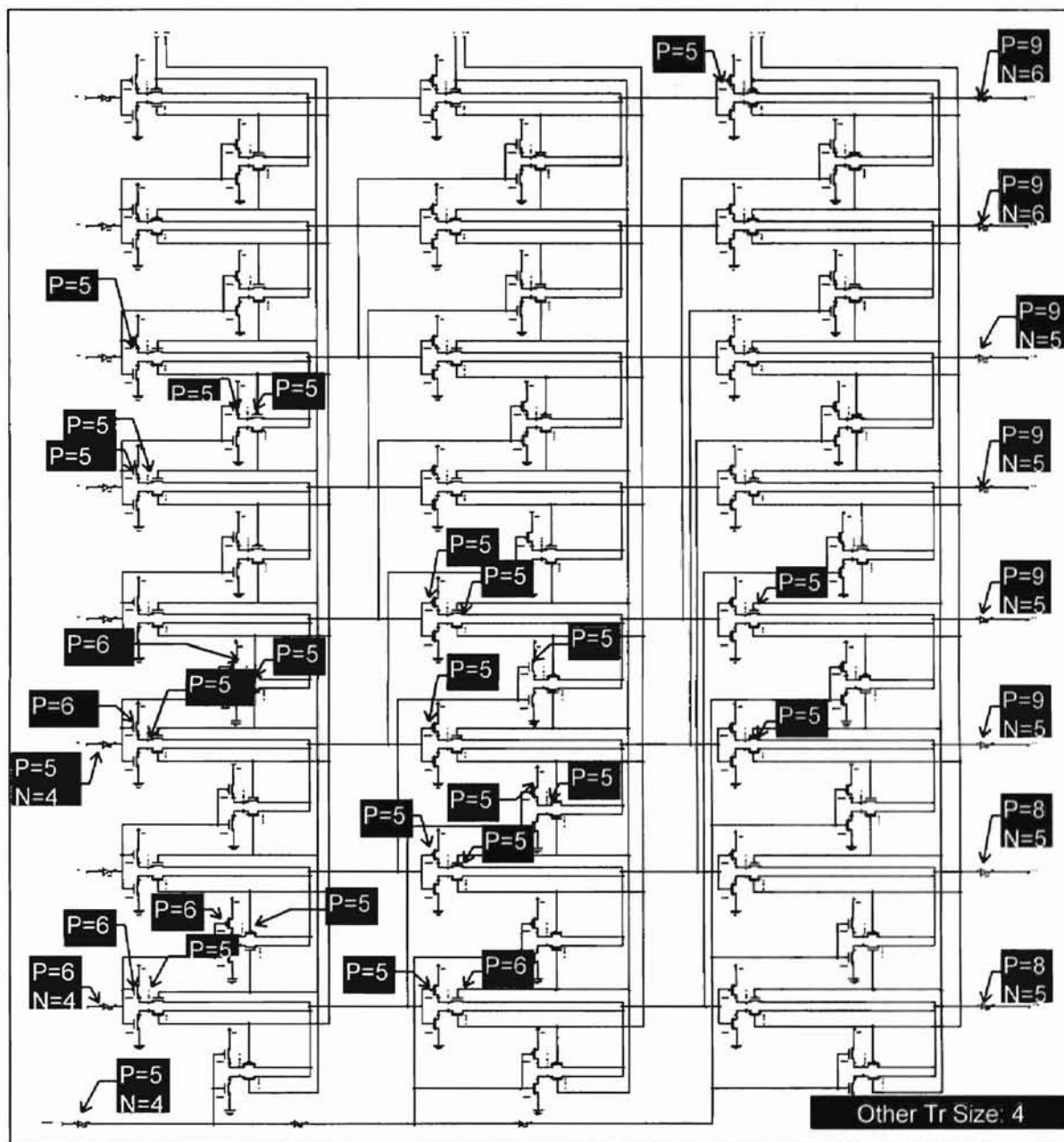


Figure 23. Schematic of Clocked Inverter Type 8 bit Barrel Shifter with

iCONTRAST Result

It happens that **iCONTRAST** finds impractical critical delay paths that do not consider signal direction in transmission gates. From the result in figure 23, **iCONTRAST** produced the expected output with signal direction control. A CMOS circuit is represent in graph by **iCONTRAST** [Sapatnekar, 1993]. **PERT** uses a delay graph of MOS circuit [Chen, 1991]. In the delay graph, **iCONTRAST** does not consider the signal direction or complementary relationship of the two transmission gates in a multiplexer. Therefore, the both transmission gates of a 2 to 1 multiplexer could be turned on at the same time. Thus, there is a signal path from an input to another input. However, this signal path does not exist in real circuit and is very long and large resistive path.

There is a big difference of computing time between **transize** and **iCONTRAST**. Table XI shows computing times for a barrel shifter executed at a Sun SparcStation 20.

| | | |
|------------------|------|--------|
| transize | real | 0.4 |
| | user | 0.0 |
| | sys | 0.1 |
| iCONTRAST | real | 5:30.4 |
| | user | 5:28.0 |
| | sys | 0.7 |

Table XI. Computing Time to Optimize 8 bit Barrel Shifter

CHAPTER VI

SUMMARY AND FUTURE WORKS

Area and timing optimizations of barrel shifters have been investigated using lagrangian multipliers within reasonable computing time. In order to find optimal solution values quickly, MOS transistors have been replaced by resistors and capacitors. The critical delay path was chosen by a heuristic method for simple calculation. Standard layout blocks were used in the layout.

Using the law of diminishing returns, basic behaviors of area and time trade-offs for MOS transistor were observed. With lagrangian multipliers, the barrel shifter was optimized. The results were compared to **iCONTRAST**'s. **transize** computes the optimal sizes faster than **iCONTRAST** does. Second, the critical delay path searching in **iCONTRAST** found an impractical critical delay path. The RC transistor model, lagrangian multiplier, and circuit simplification of repeated structure contributed to the quick response time of **transize**.

The optimized barrel shifter using **iCONTRAST** has an abnormally big transistor. That is because **iCONTRAST** is looking for critical delay paths without consideration of signal flowing direction in the transmission gates. The barrel shifter with clocked inverters does not have that kind of irregularity in MOS transistor size. There were some variations of transistor

size because that transistor had a slightly different load than others. Also, circuits with repeated structure such as the barrel shifter can be more easily layouted and verified when simplified circuit was used for optimization.

This study only dealt with optimization of a repeated structure circuit such as the barrel shifter. Also, these results may not be applicable for general circuits because the standard layout block was used. The RC transistor model has less accuracy than newly developed model that take into account non-linearities.

This study gives a basic idea of why dedicated optimization tools are needed. With better transistor models, improvement of critical delay path finding rules, and linking to layout editor, better optimization tools could be developed.

REFERENCES

- Caisso, J., E. Cerny and N. C. Rumin. (May 1991). "A recursive technique for computing delays in series-parallel MOS transistor circuits," *IEEE Trans. on Computer-Aided Design*, vol.10(no.5): pp.589-595.
- Chen, H. Y. and S. M. Kang. (Jan. 1991). "iCOACH: A circuit optimization aid for CMOS high-performance circuits," *Integration*, vol.10: pp.185-212.
- Chu, C. and M. A. Horowitz. (Nov. 1987). "Charge-sharing models for switch-level simulation," *IEEE Trans. on Computer-Aided Design*, vol. CAD-6(no.6): pp.1053-1061.
- Cirit, M. A. (June 1987). "Transistor sizing in CMOS circuits," *Proc. of 24th ACM/IEEE Design Automation Conference*:pp.121-124.
- Dai, Z. and K. Asada. (May 1989). "MOSIZ: A two-step transistor sizing algorithm based on optimal timing assignment method for multi-stage complex gates," *Proc. of IEEE 1989 Custom Integrated Circuit Conference*: pp.17.3.1-17.3.4.
- Dutta, S., S. S. M. Shetti and S. L. Lusky. (Aug. 1995). "A comprehensive delay model for CMOS inverters," *IEEE J. of Solid-State Circuits*, vol.30(no.8): pp.864-871.
- Kirkpatrick, T. I. and N. R. Clark. (March 1966). "PERT as an aid to logic design," *IBM Journal of Res. and Devel.*, vol.10: pp.135-141.

- Hedenstierna, N. and K. O. Jeppson. (March 1987). "CMOS circuit speed and buffer optimization," IEEE Trans. on Computer-Aided Design, vol. CAD-6(no.2): pp.270-281.
- Hedlund, K. S. (June 1987). "Aesop: A tool for automated transistor sizing," Proceedings of 24th ACM/IEEE Design Automation Conf.: pp.114-120.
- Lin, H. C. and L. W. Linholm. (April 1975). "An optimized output stage for MOS integrated circuits," IEEE J. of Solid-State Circuits, vol.SC-10(no.2): pp.106-109.
- Lin, T. M. and C. A. Mead. (Oct. 1984). "Signal delay in general RC networks," IEEE Trans. on Computer-Aided Design, vol.CAD-3(no.4): pp.331-349.
- Marple, D. (June 1989). "Transistor size optimization in the tailor layout system," Proc. of 26th ACM/IEEE Design Automation Conference: pp.43-48.
- Ousterhout, J. K. (July 1985). "A switch-level timing verifier for digital MOS VLSI," IEEE Trans. on Computer-Aided Design, vol.CAD-4(no.3): pp.336-349.
- Rubinstein, J., P. Penfield and M. Horowitz. (July 1983). "Signal delays in RC tree network," IEEE Trans. on Computer-aided Design, vol.CAD-2: pp.202-211.

- Sakurai, T. and A. R. Newton. (April 1990). "Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas," IEEE J. of Solid-State Circuits, vol.25(no.2): pp.584-594.
- Sakurai, T. and T. Iizuka. (Feb. 1985). "Gate electrode RC delay effects in VLSI's," IEEE J. of Solid-State Circuits, vol.SC-20(no.1): pp.290-294.
- Sapatnekar, S. S. et al. (Nov. 1993). "An exact solution to the transistor sizing problem for CMOS circuits using convex optimization," IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol.12(no.11): pp.1621-1634.

APPENDIX I

1. MATLAB Program 1

```
% This program finds optimal transistor size
% of 3 stage inverter series
clear
clc
% definitions
% w0 : gate width of first inverter
% w1 : gate width of second inverter
% w2 : gate width of third inverter
% Ci : interconnection capacitance
% Cgp : normalized gate capacitance Cg'
% Rp : normalized gate resistance R'
% Mtaup: differatiation of tau M
% w1=sqrt( Rp*w0*(Ci+Cgp*w2)/(Rp*Cgp+taup*w0) )
% Ap : height of inverter
% A0 : minimum size of inverter
% Cdp : normalized drain capacitance Cd'
% Mtaup=Ap/(3*A0+Ap*(w0+w1+w2))*(Rp/w0*(Cdp*w0+Ci+Cgp*w1)
+Rp/w1*(Cdp*w1+Ci+Cgp*w2)+Rp/w2*(Cdp*w2+Ci) )
% initialization
w0=10e-6;
w1=1e-6
w2=300e-6;
Ci=18e-15;
Cgp=1.68e-9;
Rp=0.084;
Ap=25e-6;
A0=625e-12;
Cdp=0.95e-9;
Wdiff=100;
sprintf(' w1(micro) Wdiff(micro) ')
while abs(Wdiff)>1e-10,
w1bef=w1;
Mtaup=Ap/(3*A0+Ap*(w0+w1+w2))*(Rp/w0*(Cdp*w0+Ci+Cgp*w1)
+Rp/w1*(Cdp*w1+Ci+Cgp*w2)+Rp/w2*(Cdp*w2+Ci) );
w1aft=sqrt( Rp*w0*(Ci+Cgp*w2)/(Rp*Cgp+Mtaup*w0) );
Wdiff=w1bef-w1aft;
w1=w1aft;
sprintf(' %g %g', w1*1e6, Wdiff*1e6)
end
sprintf(' w0(micro) w1(micro) w2(micro) ')
sprintf(' %g %g %g %g', w0*1e6, w1*1e6, w2*1e6)
end
```

2. MATLAB Program 2

```
% This program shows transistor size relationship among
% 3 stage inverter series
clear
```

```

clc
clg
% definitions
% w0 : gate width of first inverter
% w1 : gate width of second inverter
% w2 : gate width of third inverter
% Ci : interconnection capacitance
% Cgp : normalized gate capacitance Cg'
% Rp : normalized gate resistance R'
% Mtaup: differatiation of tau M
% w1=sqrt( Rp*w0*(Ci+Cgp*w2)/(Rp*Cgp+Mtaup*w0) )
% Ap : height of inverter
% A0 : minimum size of inverter
% Cdp : normalized drain capacitance Cd'
% Mtaup=Ap/(3*A0+Ap*(w0+w1+w2))*(Rp/w0*(Cdp*w0+Ci+Cgp*w1)
+Rp/w1*(Cdp*w1+Ci+Cgp*w2)+Rp/w2*(Cdp*w2+Ci) )
% initialization
Ci=18e-15;
Cgp=1.68e-9;
Rp=0.084;
Ap=25e-6;
A0=625e-12;
Cdp=0.95e-9;
N=5;
w2i=100e-6;
w2f=1000e-6;
w2d=(w2f-w2i)/N;
for j=1:N,
w2=w2d*j+w2i;
M=50;
w0i=5e-6;
w0f=50e-6;
w0d=(w0f-w0i)/M;
for i=1:M,
w0(i)=w0d*i+w0i;
wlc=1e-6;
Wdiff=100;
while abs(Wdiff)>1e-10,
wlbef=wlc;
Mtaup=Ap/(3*A0+Ap*(w0(i)+wlc+w2))*(Rp/w0(i)*(Cdp*w0(i)
+Ci+Cgp*wlc)+Rp/wlc*(Cdp*wlc+Ci+Cgp*w2)
+Rp/w2*(Cdp*w2+Ci) );
wlaft=sqrt(Rp*w0(i)*(Ci+Cgp*w2)/(Rp*Cgp+Mtaup*w0(i)));
Wdiff=wlbef-wlaft;
wlc=wlaft;
end
w1(i,j)=wlc;
end
end
plot(w0*1e6,w1*1e6)
xlabel('w0: width of first inverter')
ylabel('w1: width of second inverter')

```

```

%title(' w0, w1, and w2 (unit: micron)')
    for i=1:N,
        w2=w2d*i+w2i;
        mm=sprintf('w2= %g',w2*1e6);
        text(0.9*w0f*1e6,w1(M,i)*1e6,mm)
    end
end

```

3. MATLAB program 3

```

% This program shows optimal size of middle inverter
clear
clc
clg
% definitions
% w0 : gate width of first inverter
% w1 : gate width of second inverter
% w2 : gate width of third inverter
% Ci : interconnection capacitance
% Cgp : normalized gate capacitance Cg'
% Rp : normalized gate resistance R'
% Mtaup: differatiation of tau M
% w1=sqrt( Rp*w0*(Ci+Cgp*w2)/(Rp*Cgp+Mtaup*w0) )
% Ap : height of inverter
% A0 : minimum size of inverter
% Cdp : normalized drain capacitance Cd'
% Mtaup=Ap/(3*A0+Ap*(w0+w1+w2))*(Rp/w0*(Cdp*w0+Ci+Cgp*w1)
+Rp/w1*(Cdp*w1+Ci+Cgp*w2)+Rp/w2*(Cdp*w2+Ci) )
% Total delay
% tau=Rp/w0*(Cdp*w0+Ci+Cgp*w1)+Rp/w1*(Cdp*w1+Ci+Cgp*w2)
+Rp/w2*(Cdp*w2+Ci)
% Total area
% A=3*A0+Ap*(w0+w1+w2)
% initialization
Ci=18e-15;
Cgp=1.68e-9;
Rp=0.084;
Ap=25e-6;
A0=625e-12;
Cdp=0.95e-9;
w0=10e-6;
M=10;
w2i=100e-6;
w2f=1000e-6;
w2d=(w2f-w2i)/M;
    for j=1:M,
        w2=w2d*j+w2i;
        N=100;
        w1d=(w2-w0)/N;
            for i=1:N,
                w1=w1d*i+w0;
                tau(i,j)=Rp/w0*(Cdp*w0+Ci+Cgp*w1)
            end
        end
    end

```

```

+Rp/w1*(Cdp*w1+Ci+Cgp*w2)+Rp/w2*(Cdp*w2+Ci);
A(i,j)=3*A0+Ap*(w0+w1+w2);
end
Wdiff=100;
while abs(Wdiff)>1e-10,
    wlbef=w1;
    Mtaup=Ap/(3*A0+Ap*(w0+w1+w2))*(Rp/w0*(Cdp*w0+Ci+Cgp*w1)
        +Rp/w1*(Cdp*w1+Ci+Cgp*w2)+Rp/w2*(Cdp*w2+Ci));
    wlaft=sqrt(Rp*w0*(Ci+Cgp*w2)/(Rp*Cgp+Mtaup*w0));
    Wdiff=wlbef-wlaft;
    w1=wlaft;
end
tauo(j)=Rp/w0*(Cdp*w0+Ci+Cgp*w1)+Rp/w1*(Cdp*w1+Ci+Cgp*w2)
    +Rp/w2*(Cdp*w2+Ci);
Aopt(j)=3*A0+Ap*(w0+w1+w2);
end
plot(A*1e12,tau*1e9,Aopt*1e12,tauo*1e9,'*')
xlabel('Area: total area of 3 inverters(unit:micro meter^2)')
ylabel('Time: total time through 3 inverters(unit:nano sec)')
for i=1:M,
    w2=w2d*i+w2i;
    mm=sprintf('w2= %g um',w2*1e6);
    text(A(1,i)*1e12,tau(1,i)*1e9,mm)
end
mm=sprintf('w0= %g micro meter',w0*1e6);
text(A(N,M)/3*1e12,tau(N,M)*1e9,mm)
mm=sprintf('w1= from %g micro meter to the size of w2',w0*1e6);
text(A(N,M)/3*1e12,tau(N,M)*1e9*9/10,mm)
grid
end

```

4. MATLAB Program 4

```

% This program finds optimal transistor size of
% multiple stages of inverter series.
clear
clc
% definitions
% W(i): gate width of i-th inverter
% Ci : interconnection capacitance
% Cgp : normalized gate capacitance Cg'
% Rp : normalized gate resistance R'
% Mtaup: differatiation of tau M
% W(i)=sqrt(Rp*W(i-1)*(Ci+Cgp*W(i+1))/(Rp*Cgp+Mtaup*W(i-1)))
% Ap : height of inverter
% A0 : minimum size of inverter
% Cdp : normalized drain capacitance Cd'
% Mtaup=Ap/(m*A0+Ap*(w0+w1+w2))*(Rp/w0*(Cdp*w0+Ci+Cgp*w1)
    +Rp/w1*(Cdp*w1+Ci+Cgp*w2)+Rp/w2*(Cdp*w2+Ci))
% M : Number of Inverter Stages
% initialization
M=7;

```



```

% size of the first inverter
W(1)=10e-6;
% size of the last inverter
W(M)=500e-6;
% set other inverter's initial value
    for i=2:M-1,
        W(i)=W(M);
    end
Ci=18e-15;
Cgp=1.68e-9;
Rp=0.084;
Ap=25e-6;
A0=625e-12;
Cdp=0.95e-9;
Wdiff=100;
Wold=W;
    while abs(Wdiff)>1e-10,
        Area=M*A0+Ap*sum(W);
        Mtaup=0;
        for i=2:M-1,
            Mtaup=Mtaup+Rp/W(i-1)*(Cdp*W(i-1)+Ci+Cgp*W(i));
        end
        Mtaup=Mtaup+Rp/W(M)*(Cdp*W(M)+Ci);
        Mtaup=Ap/Area*Mtaup;
        for i=2:M-1,
            W(i)=sqrt(Rp*W(i-1)*(Ci+Cgp*W(i+1))/(Rp*Cgp+Mtaup*W(i-1)));
        end
        Wdiff=0;
        for i=2:M-1,
            Wdiff=Wdiff+abs(W(i)-Wold(i));
        end
        Wold=W;
    end
W*1e6
end
W*1e6
end

```

5. MATLAB Program 5

```

% This program evaluates optimal transistor size.
clear
clc
clg
% definitions
% W(i): gate width of i-th inverter
% Ci : interconnection capacitance
% Cgp : normalized gate capacitance Cg'
% Rp : normalized gate resistance R'
% Ap : height of inverter
% A0 : minimum size of inverter
% Cdp : normalized drain capacitance Cd'
% M : Number of Inverter Stages

```

```

% initialization
Ci=18e-15;
Cgp=1.68e-9;
Rp=0.084;
Ap=25e-6;
A0=625e-12;
Cdp=0.95e-9;
M=7;
W(1)=10e-6;
W(M)=500e-6;
% N : total number of forced error points
N=100;
% set optimized value of inverters
for i=2:M-1,
    if i==2,
        mm=sprintf('Size of %g nd inverter in micron : ',i);
    elseif i==3,
        mm=sprintf('Size of %g rd inverter in micron : ',i);
    else
        mm=sprintf('Size of %g th inverter in micron : ',i);
    end
    xx=input(mm);
    W(i)=xx*1e-6;
end
Worg=W;
tau=0;
for i=1:M-1,
    tau=tau+Rp/W(i)*(Cdp*W(i)+Ci+Cgp*W(i+1));
end
tau=tau+Rp/W(M)*(Cdp*W(M)+Ci);
Area=0;
for i=1:M,
    Area=Area+Ap*W(i);
end
Area=Area+M*A0;
Topt=tau;
Aopt=Area;
% set initial value of other inverter
for j=2:M-1,
    W=Worg;
    for i=1:N,
        W(j)=0.5*Worg(j)+Worg(j)*i/N;
        tau=0;
        for k=1:M-1,
            tau=tau+Rp/W(k)*(Cdp*W(k)+Ci+Cgp*W(k+1));
        end
        tau=tau+Rp/W(M)*(Cdp*W(M)+Ci);
        Area=0;
        for k=1:M,
            Area=Area+Ap*W(k);
        end
        Area=Area+M*A0;
    end
end

```

```

        T(i,j-1)=tau;
        A(i,j-1)=Area;
    end
end
plot(A*1e12,T*1e9,Aopt*1e12,Topt*1e9,'*')
xlabel('Area: total area of All inverters(unit:micro meter^2)')
ylabel('Time: total time through All inverters(unit:nano sec)')
grid
end

```

6. MATLAB Program 6

```

% This program finds optimal number of stages
clear
clc
clg
% definitions
% W(i): gate width of i-th inverter
% Ci : interconnection capacitance
% Cgp : normalized gate capacitance Cg'
% Rp : normalized gate resistance R'
% Mtaup: differatiation of tau M
% w(i)=sqrt( Rp*w(i-1)*(Ci+Cgp*w(i+1))/(Rp*Cgp+Mtaup*w(i-1)) )
% Ap : height of inverter
% A0 : minimum size of inverter
% Cdp : normalized drain capacitance Cd'
% Mtaup=Ap/(m*A0+Ap*(w0+w1+w2))*(Rp/w0*(Cdp*w0+Ci+Cgp*w1)
+Rp/w1*(Cdp*w1+Ci+Cgp*w2)+Rp/w2*(Cdp*w2+Ci) )
% M : Number of Inverter Stages
% initialization
for M=3:20,
% size of the first inverter
W(1)=10e-6;
% size of the last inverter
W(M)=500e-6;
% set other inverter's initial value
for i=2:M-1,
W(i)=W(M);
end
Ci=18e-15;
Cgp=1.68e-9;
Rp=0.084;
Ap=25e-6;
A0=625e-12;
Cdp=0.95e-9;
Wdiff=100;
Wold=W;
while abs(Wdiff)>1e-10,
Area=M*A0+Ap*sum(W);
Mtaup=0;
for i=2:M-1,
Mtaup=Mtaup+Rp/W(i-1)*(Cdp*W(i-1)+Ci+Cgp*W(i));

```

```

        end
        Mtaup=Mtaup+Rp/W(M) * (Cdp*W(M) +Ci) ;
        Mtaup=Ap/Area*Mtaup;
        for i=2:M-1,
            W(i)=sqrt( Rp*W(i-1)
                *(Ci+Cgp*W(i+1)) / (Rp*Cgp+Mtaup*W(i-1)) );
        end
        Wdiff=0;
        for i=1:M,
            Wdiff=Wdiff+abs(W(i)-Wold(i));
        end
        Wold=W;
        W*1e6;
        end
M;
W*1e6;
    for i=1:M,
        Wopt(M-2,i)=W(i);
    end
tau=0;
    for i=1:M-1,
        tau=tau+Rp/W(i) * (Cdp*W(i) +Ci+Cgp*W(i+1));
    end
tau=tau+Rp/W(M) * (Cdp*W(M) +Ci) ;
Area=0;
    for i=1:M,
        Area=Area+Ap*W(i);
    end
Area=Area+M*A0;
Topt(M-2)=tau;
Aopt(M-2)=Area;
end
plot(Aopt*1e12,Topt*1e9,Aopt*1e12,Topt*1e9,'*')
for i=1:10,
mm=sprintf('%g',i+2);
text(Aopt(i)*1e12*29/30,Topt(i)*1e9*21/20,mm);
end
for i=11:18,
mm=sprintf('%g',i+2);
text(Aopt(i)*1e12,Topt(i)*1e9*19/20,mm);
end
xlabel('Area: total area of All inverters(unit:micro meter^2)')
ylabel('Time: total time through All inverters(unit:nano sec)')
grid
end

```

APPENDIX II

This program is the barrel shifter optimizer, **transize**.

1. Input Data File Sample

```
0.06 0.014 0.014
0.1 0.05 0.05
0.2 0.25 0.05 0.05
0.2 0.05 0.05
50 300
100 600
408
480
24
3
1 1 1 1
1 1 1
0.1 0.2 0.4
480 576 1152
4 8 8 20
4 8 8 20
4 8 8
4 8 8
0.5
0.1
0.0
0.0
7500
0.5
5
50
100
```

2. Makefile

```
HOME =

O_FILES = main.o \
          readinit.o \
          update.o \
          neww.o \
          iterate.o \
          delay.o \
          area.o

BINARY = transize

LIBS    = -lm

IFLAGS  =
```

```

CFLAGS =

CC      = cc

$(BINARY):      $(O_FILES)
                $(CC) $(CFLAGS) $(IFLAGS) -o $(BINARY) $(O_FILES) $(GENLIB)
$(LIBS)

.c.o:
        $(CC) $(CFLAGS) $(IFLAGS) -c $<

$(O_FILES):      transize.h

install:
        cp $(BINARY) $(HOME)/bin

```

3. transize.h

```

#define MAXSTAGES 10

/* capacitance coefficients */
float CIB; /*buffer input constant cap*/
float CIBn; /*buffer input cap coeff for Wn*/
float CIBp; /*buffer input cap coeff for Wp*/
float COB; /*buffer output constant cap*/
float COBn; /*buffer output cap coeff for Wn*/
float COBp; /*buffer output cap coeff for Wp*/
float CI0M; /*MUX input 0 constant cap*/
float CI1M; /*MUX input 1 constant cap*/
float CIMn; /*MUX input cap coeff for Wn*/
float CIMp; /*MUX input cap coeff for Wp*/
float COM; /*MUX output constant cap*/
float COMn; /*MUX output cap coeff for Wn*/
float COMP; /*MUX output cap coeff for Wp*/

/* resistance coefficients */
float Rnl; /*low signal resistance coeff for 1/Wn*/
float Rnh; /*high signal resistance coeff for 1/Wn*/
float Rpl; /*low signal resistance coeff for 1/Wp*/
float Rph; /*high signal resistance coeff for 1/Wp*/

/* computed quantities */
float CB[MAXSTAGES]; /*buffer output cap for each stage*/
float CM[MAXSTAGES]; /*MUX output cap for each stage*/
float Cx[MAXSTAGES]; /*crossover cap for each stage*/
float RBph[MAXSTAGES]; /*buffer pFET high signal resistance for each
stage*/
float RBnl[MAXSTAGES]; /*buffer nFET low signal resistance for each
stage*/
float RMh[MAXSTAGES]; /*MUX parallel FET high signal resistance for each
stage*/

```

```

float RMl[MAXSTAGES]; /*MUX parallel FET low signal resistance for each
stage*/
float alphaA; /*Lagrange multiplier for area constraint*/
float alphas; /*Lagrange multiplier for equal time delays*/
float Wbn[MAXSTAGES+1]; /*nFET widths for each buffer stage*/
float Wbp[MAXSTAGES+1]; /*pFET widths for each buffer stage*/
float Wmn[MAXSTAGES]; /*nFET widths for each MUX stage*/
float Wmp[MAXSTAGES]; /*pFET widths for each MUX stage*/
float dalphas; /*change in Lagrange multiplier for area constraint*/
float dalphas; /*change in Lagrange multiplier for equal time delays*/
float dWbn[MAXSTAGES]; /*change in nFET widths for each buffer stage*/
float dWbp[MAXSTAGES]; /*change in pFET widths for each buffer stage*/
float dWmn[MAXSTAGES]; /*change in nFET widths for each MUX stage*/
float dWmp[MAXSTAGES]; /*change in pFET widths for each MUX stage*/
float AItot; /* total constant area */

/* circuit structure */
int N; /*actual number of stages*/
int parity[MAXSTAGES]; /* number of inversions from input */
int B[MAXSTAGES]; /* 0 = no inverter buffer, 1 = inverter buffer present
*/
int M[MAXSTAGES]; /* 0 = no MUX, 1 = MUX present */
float A0; /* desired circuit area for one bit path*/
float H; /* cell height */

/* convergence parameters */
float dWmax; /*max change in W's in an iteration*/
float Wtol; /*tolerance for W's: dWmax < Wtol */
float ttol; /*tolerance for diff between tr and tf */
float Atol; /*tolerance for area*/
int maxiter; /*maximum number of iterations allowed */
float dWs; /*convergence solving*/
int dWf; /*convergence solving flag*/

```

4. main.c

```

#include <stdio.h>
#include <math.h>
#include "transize.h"

main(argc, argv)
int argc;
char *argv[];
{
    void readint();
    void update();
    void iterate();
    void delay();
    float area();

    FILE *fp;
    int i, iterW, itera, iterA;

```

```

char ans[10];
float tr,tf; /* rise and fall time delay */
float A; /* area */
float alphasatnew,alphatold,delt,deltold;
float alphaAnew,alphaAold,delA,delAold;
int been0;
float delA0;

if (argc > 2) {
    fprintf(stderr,"usage: %s \n", argv[0]);
    fprintf(stderr,"  or %s filename\n", argv[0]);
    exit(1);
}
if (argc == 1) {
    fp = stdin;
}
else {
    if ( (fp=fopen(argv[1],"r")) == NULL) {
        fprintf(stderr,"cannot open file: %s\n",argv[1]);
        exit(1);
    }
}

readinit(fp);
update();
delay(&tr,&tf);
A = area();
printf("\ninitial rise time = %f, initial fall time = %f\ninitial area
= %f\n",tr,tf,A);
delA = Atol + Atol;
iterA = 0;
been0 = 0;
while ( (fabs(delA) > Atol) && (iterA < maxiter) ){
#ifdef DEBUG
    printf("\niterA = %d, alphaA = %f, delA = %f, alphaAold = %f,
delAold =%f\n",iterA,alphaA,delA,alphaAold,delAold);
#endif
    if (iterA == 1) {
#ifdef DEBUG
        printf("iterA = 1 branch\n");
#endif
        if (dalphaA == 0.0) exit(0);
        alphaAold = alphaA;
        alphaA = alphaA + dalphaA;
        delAold = delA;
    }
    if (iterA > 1) {
#ifdef DEBUG
        printf("iterA > 1 branch\n");
#endif
        if (delA == delAold) {
            fprintf(stderr,"error: infinite alphaA!\n");

```



```

    exit(1);
}
alphaAnew = (alphaAold*delA - alphaA*delAold)/(delA - delAold);
if (alphaAnew <= 0.0) {
    if (been0) {
        alphaAnew = (alphaA*delA0)/(delA0 - delA);
        if (alphaAnew <= 0.0) {
            fprintf(stderr,"error: negative alphaA, choose smaller area
or use min delay area.\n");
            exit(1);
        }
    }
    else alphaAnew = 0.0;
}
alphaAold = alphaA;
alphaA = alphaAnew;
delAold = delA;
}
#ifdef DEBUG
    printf("\niterA = %d, alphaA = %f, delA = %f, alphaAold = %f,
delAold = %f\n",iterA,alphaA,delA,alphaAold,delAold);
#endif
    iterA++;
    printf("\n*****\n\nalphaA(%d) =
%f\n",iterA,alphaA);

    itera = 0;
    delt = ttol + ttol;
    while ( (fabs(delt) > ttol) && (itera < maxiter) ) {
#ifdef DEBUG
        printf("\nitera = %d, alphas = %f, delt = %f, alphasold = %f,
deltold = %f\n",itera,alphat,delt,alphatold,deltold);
#endif
        if (itera == 1) {
#ifdef DEBUG
            printf("itera = 1 branch\n");
#endif
            if (dalphas == 0.0) break;
            alphasold = alphas;
            alphas = alphas + dalphas;
            deltold = delt;
        }
        if (itera > 1) {
#ifdef DEBUG
            printf("itera > 1 branch\n");
#endif
            if (delt == deltold) {
                fprintf(stderr,"error: infinite alphas!\n");
                exit(1);
            }
            alphasnew = (alphasold*delt - alphas*deltold)/(delt - deltold);
            alphasold = alphas;

```



```

printf("\n buffer pFET widths are: \n");
for (i=0;i<=N;i++) printf(" %f",WBp[i]);
printf("\n MUX nFET widths are: \n");
for (i=0;i<N;i++) printf(" %f",WMn[i]);
printf("\n MUX pFET widths are: \n");
for (i=0;i<N;i++) printf(" %f",WMP[i]);
delay(&tr,&tf);
printf("\nrising input delay = %f, falling input delay =
%f\n",tr,tf);
A = area();
printf("\nArea = %f\n",A);
delt = tf - tr;
if (dWmax > Wtol) {
    printf("\n\nerror: convergence failed. Increase iterW or
Wtol.\n");
    exit(1);
}
}
if ( (dalphat != 0.0) && (fabs(delt) > ttol) ) {
    printf("\n\nerror: convergence failed. Increase itera or
ttol.\n");
    exit(1);
}
delA = A - A0;
if ( (alphaA == 0.0) && (been0 == 0) ) {
    delA0 = delA;
    been0 = 1;
}
}
if ( (dalphat != 0.0) && (fabs(delA) > Atol) ) {
    printf("\n\nerror: convergence failed. Increase iterA or Atol.\n");
    exit(1);
}
}
}

```

5. area.c

```

#include "transize.h"

float area()
{
    float A;
    int i;

    A = AItot;
    for (i=0;i<N;i++) {
        if (B[i]) {
            A += H*(WBn[i] + WBp[i]);
        }
        if (M[i]) {
            A += H*(WMn[i] + WMP[i]);
        }
    }
}

```

```

    }
#ifdef DEBUG
    printf("\nA = %f\n",A);
#endif
    return(A);
}

```

6. delay.c

```

#include "transize.h"

void delay(tr,tf)
float *tr, *tf;
{
    float tri,tfi;
    int i,j;
    float f,p,q;
    float Rh,Rl;

    f = 0.0;
    *tr = 0.0;
    *tf = 0.0;
    for (i=0;i<N;i++) {
        tri = 0.0;
        tfi = 0.0;
        if (B[i]) {
            tri = RBph[i]*CB[i];
            tfi = RBnl[i]*CB[i];
        }
        Rh = 0.0;
        Rl = 0.0;
        for (j=i;;j--) {
            if (M[j]){
                Rh += RMh[j];
                Rl += Rml[j];
            }
            if (B[j]){
                Rh += RBph[j];
                Rl += RBnl[j];
                break;
            }
        }
#ifdef DEBUG
        printf("i = %d, Rh = %f, Rl = %f\n",i,Rh,Rl);
#endif
        if (M[i]) {
            tri += Rh*CM[i];
            tfi += Rl*CM[i];
        }
        if (parity[i]) { /* odd */
            *tr = *tr + tfi;
            *tf = *tf + tri;
        }
    }
}

```

```

        p = alphas;
        q = 1.0 - alphas;
    }
    else { /* even */
        *tr = *tr + tri;
        *tf = *tf + tfi;
        p = 1.0 - alphas;
        q = alphas;
    }
#ifdef DEBUG
    f += p*tri + q*tfi;
    printf("delay partial f = %f\n",f);
#endif
}
#ifdef DEBUG
    printf("\nf = %f, tr = %f, tf = %f\n",f,*tr,*tf);
#endif
}

```

7. iterate.c

```

#include "transize.h"

void iterate()
{
    int i,j;
    float p,q;
    float a,b; /* d2f/dw2 and df/dw */
    float f;
    float CL,Rh,Rl;
    float alphaxht;

    alphaxht = alphaA*H;
#ifdef DEBUG
    f = alphas*RBph[0]*CB[0] + (1.0 - alphas)*RBnl[0]*CB[0];
#endif
    for (i=0;i<N;i++) {
#ifdef DEBUG
        printf("\n i = %d\n",i);
#endif
        if (parity[i]) { /* odd */
            p = alphas;
            q = 1.0 - alphas;
        } else { /* even */
            p = 1.0 - alphas;
            q = alphas;
        }
#ifdef DEBUG
        printf("p = %f, q = %f\n",p,q);
#endif
    }

    /* inverter buffer width changes */

```

```

if ( (i==0) || (B[i] == 0) ) {
    dWBn[i] = 0;
    dWBp[i] = 0;
}
else {
    CL = CB[i] + CM[i];
    for (j=i+1;;j++) {
        if (B[j]) break;
        CL += CM[j];
    }
    Rh = 0.0;
    Rl = 0.0;
    for (j=i-1;;j--) {
        if (M[j]){
            Rh += RMh[j];
            Rl += Rml[j];
        }
        if (B[j]){
            Rh += RBph[j];
            Rl += RBnl[j];
            break;
        }
    }
}
#ifdef DEBUG
    printf("Rh = %f, Rl = %f\n",Rh,Rl);
#endif
a = (q+q)*RBnl[i]*RBnl[i]/Rnl*(RBnl[i]/Rnl*CL - COBn);
b = alphaxht + q*Rh*CIBn + p*Rl*CIBn +
    p*RBph[i]*COBn + q*( RBnl[i]*COBn - RBnl[i]*RBnl[i]*CL/Rnl );
#ifdef DEBUG
    printf("df/dWBn = %f, d2f/dWBn2 =%f\n",b,a);
#endif
dWBn[i] = -b/a;
a = (p+p)*RBph[i]*RBph[i]/Rph*(RBph[i]/Rph*CL - COBp);
b = alphaxht + q*Rh*CIBp + p*Rl*CIBp +
    p*( RBph[i]*COBp - RBph[i]*RBph[i]*CL/Rph ) + q*RBnl[i]*COBp;
#ifdef DEBUG
    printf("df/dWBp = %f, d2f/dWBp2 =%f\n",b,a);
#endif
dWBp[i] = -b/a;
#ifdef DEBUG
    f += p*RBph[i]*CB[i] + q*RBnl[i]*CB[i];
    printf("partial f = %f\n",f);
#endif
}

/* MUX width changes */
if ( M[i] == 0 ) {
    dWMn[i] = 0;
    dWMP[i] = 0;
}
else {

```

```

    CL = CM[i];
    for (j=i+1;;j++) {
        if (B[j]) break;
        CL += CM[j];
    }
    Rh = 0.0;
    Rl = 0.0;
    for (j=i;;j--) {
        if ( (M[j] != 0) && (j != i) ) {
            Rh += RMh[j];
            Rl += Rml[j];
        }
        if (B[j]){
            Rh += RBph[j];
            Rl += RBnl[j];
            break;
        }
    }
}

#ifdef DEBUG
    printf("Rh = %f, Rl = %f\n",Rh,Rl);
#endif
a = (p+p)*RMh[i]*RMh[i]/Rnh*(RMh[i]/Rnh*CL - COMn) +
    (q+q)*Rml[i]*Rml[i]/Rnl*(Rml[i]/Rnl*CL - COMn);
b = alphaxht + p*( Rh*(CIMn + CIMn + COMn) + RMh[i]*COMn -
RMh[i]*RMh[i]/Rnh*CL ) +
    q*( Rl*(CIMn + CIMn + COMn) + Rml[i]*COMn -
Rml[i]*Rml[i]/Rnl*CL );
#ifdef DEBUG
    printf("df/dWMn = %f, d2f/dWMn2 =%f\n",b,a);
#endif
dWMn[i] = -b/a;
a = (p+p)*RMh[i]*RMh[i]/Rph*(RMh[i]/Rph*CL - COMp) +
    (q+q)*Rml[i]*Rml[i]/Rpl*(Rml[i]/Rpl*CL - COMp);
b = alphaxht + p*( Rh*(CIMp + CIMp + COMp) + RMh[i]*COMp -
RMh[i]*RMh[i]/Rph*CL ) +
    q*( Rl*(CIMp + CIMp + COMp) + Rml[i]*COMp -
Rml[i]*Rml[i]/Rpl*CL );
#ifdef DEBUG
    printf("df/dWmp = %f, d2f/dWmp2 =%f\n",b,a);
#endif
dWmp[i] = -b/a;
#ifdef DEBUG
    f += p*(Rh + RMh[i])*CM[i] + q*(Rl + Rml[i])*CM[i];
    printf("partial f = %f\n",f);
#endif
}
}
#ifdef DEBUG
    printf("f = %f\n",f);
#endif
}

```

8. neww.c

```

#include <math.h>
#include "transize.h"

float neww(W,dW)
float W,dW;
{
    float Wmin = 4.0;

    W = W + dW;
    if (W < Wmin) {
        dW = dW + Wmin -W;
        W = Wmin;
    }
    dW = fabs(dW);
    if (dW > dWmax) dWmax = dW;
    return(W);
}

```

9. readinit.c

```

/* reads constants and initial values for variables*/
#include <stdio.h>
#include "transize.h"

readinit(fp)
FILE *fp;
{
    int i;
    float AIB,AIM,Ax[MAXSTAGES];

    printf("buffer input cap coeffs CIB, CIBn, CIBp: ");
    fscanf(fp,"%f %f %f",&CIB,&CIBn,&CIBp);
    printf("CIB = %f, CIBn = %f, CIBp = %f\n",CIB,CIBn,CIBp);

    printf("buffer output cap coeffs COB, COBn, COBp: ");
    fscanf(fp,"%f %f %f",&COB,&COBn,&COBp);
    printf("COB = %f, COBn = %f, COBp = %f\n",COB,COBn,COBp);

    printf("MUX input cap coeffs CI0M, CI1M, CIMn, CIMp: ");
    fscanf(fp,"%f %f %f %f",&CI0M,&CI1M,&CIMn,&CIMp);
    printf("CI0M = %f, CI1M = %f, CIMn = %f, CIBp = %f\n",CI0M,CI1M,CIMn,CIMp);

    printf("MUX output cap coeffs COM, COMn, COMP: ");
    fscanf(fp,"%f %f %f",&COM,&COMn,&COMP);
    printf("COM = %f, COMn = %f, COMP = %f\n",COM,COMn,COMP);

    printf("nFET res coeffs Rnl, Rnh: ");
    fscanf(fp,"%f %f",&Rnl,&Rnh);
    printf("Rnl = %f, Rnh = %f\n",Rnl,Rnh);
}

```



```

    for (i=0;i<=N;i++) fscanf(fp,"%f",&WBp[i]);
    printf("WBp = "); for (i=0;i<=N;i++) printf(" %f",WBp[i]);
    printf("\n");

    printf("initial widths for MUX nFETs: \n");
    for (i=0;i<N;i++) fscanf(fp,"%f",&WMn[i]);
    printf("WMn = "); for (i=0;i<N;i++) printf(" %f",WMn[i]);
    printf("\n");

    printf("initial widths for MUX pFETs: \n");
    for (i=0;i<N;i++) fscanf(fp,"%f",&WMp[i]);
    printf("WMp = "); for (i=0;i<N;i++) printf(" %f",WMp[i]);
    printf("\n");

    printf("initial lagrange multiplier for propagation time: ");
    fscanf(fp,"%f",&alphan);
    printf("alphan = %f\n",alphan);

    printf("initial change in lagrange multiplier for propagation time:
");
    fscanf(fp,"%f",&dalphan);
    printf("dalphan = %f\n",dalphan);

    printf("initial lagrange multiplier for area: ");
    fscanf(fp,"%f",&alphaA);
    printf("alphaA = %f\n",alphaA);

    printf("initial change in lagrange multiplier for area: ");
    fscanf(fp,"%f",&dalphaA);
    printf("dalphaA = %f\n",dalphaA);

    printf("desired circuit area: \n");
    fscanf(fp,"%f",&A0);
    printf("A0 = %f\n",A0);

    printf("W tolerance: \n");
    fscanf(fp,"%f",&Wtol);
    printf("Wtol = %f\n",Wtol);

    printf("tr,tf diff tolerance: \n");
    fscanf(fp,"%f",&ttol);
    printf("ttol = %f\n",ttol);

    printf("area tolerance: \n");
    fscanf(fp,"%f",&Atol);
    printf("Atol = %f\n",Atol);

    printf("max iterations: \n");
    fscanf(fp,"%d",&maxiter);
    printf("maxiter = %d\n",maxiter);

    AItot = 0.0;

```

```

for (i=0;i<N;i++) {
    dWBn[i] = 0.0;
    dWBp[i] = 0.0;
    dWMn[i] = 0.0;
    dWMp[i] = 0.0;
    CB[i] = 0.0;
    CM[i] = 0.0;
    if (i) {
        if (B[i]) parity[i] = 1 - parity[i-1];
        else parity[i] = parity[i-1];
    }
    else parity[0] = 1;
    if (B[i]) AItot += AIB;
    AItot += Ax[i];
    if (M[i]) AItot += AIM;
}

#ifdef DEBUG
    printf("parity = "); for (i=0;i<N;i++) printf(" %d",parity[i]);
    printf("\n");
    printf("AItot = %f\n",AItot);
#endif
}

```

10. update.c

```

#include "transize.h"

void update()
{
    float neww();
    int i;

    /*alphanat = alphanat + dalphanat;*/
    dWmax = 0.0;
    for (i=0;i<N;i++) {
        WBn[i] = neww(WBn[i],dWBn[i]);
        WBp[i] = neww(WBp[i],dWBp[i]);
        WMn[i] = neww(WMn[i],dWMn[i]);
        WMp[i] = neww(WMp[i],dWMp[i]);
    }
    for (i=0;i<N;i++) {
        RBph[i] = Rph/WBp[i];
        RBnl[i] = Rnl/WBn[i];
        RMh[i] = 1.0/(WMn[i]/Rnh + WMp[i]/Rph);
        Rml[i] = 1.0/(WMn[i]/Rnl + WMp[i]/Rpl);
        if (B[i]) {
            CB[i] = COB + COBn*WBn[i] + COBp*WBp[i] + Cx[i];
            if (M[i]) CB[i] += CIOM + CI1M + 2.0*CIMn*WMn[i] +
2.0*CIMp*WMp[i];
            else CB[i] += CIB + CIBn*WBn[i+1] + CIBp*WBp[i+1];
        }
    }
}

```

```
    if (M[i]) {  
        CM[i] = COM + COMn*WMn[i] + COMp*WMP[i];  
        if (B[i+1]) CM[i] += CIB + CIBn*WBn[i+1] + CIBp*WBp[i+1];  
        else CM[i] += Cx[i] + CIOM + CIIM + 2.0*CIMn*WMn[i+1] +  
2.0*CIMP*WMP[i+1];  
    }  
}  
}
```

VITA 2

Byungha Joo

Candidate for the Degree of

Master of Science

Thesis: TIMING AND AREA OPTIMIZATION OF CMOS BARREL SHIFTER

Major Field: Electrical Engineering

Biographical:

Personal Data: Born in Busan, Korea, November 19, 1965, son of Mr. Jongheon Joo and Mrs. Kyungja Cho

Educational: Graduated from Posung High School, Seoul, Korea in February 1984; received Bachelor of Science from Yonsei University, Seoul, Korea in February 1988; completed requirements for Master of Science at Oklahoma State University in May, 1996.

Professional Experience: Researcher, (1988-1989) SAMSUNG Semiconductor and Telecommunication, Kyungki-Do, Korea
Researcher, (1989-1991) SAMSUNG Electronics, Kyungki-Do, Korea
Associate Staff Researcher, (1991-1993) SAMSUNG Electronics, Kyungki-Do, Korea