

## **INFORMATION TO USERS**

**This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.**

**The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.**

**In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.**

**Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.**

**Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.**

**ProQuest Information and Learning  
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA  
800-521-0600**

**UMI<sup>®</sup>**



**THE UNIVERSITY OF OKLAHOMA  
GRADUATE COLLEGE**

**POWER-SPEED TRADE-OFF  
IN  
PARALLEL PREFIX CIRCUITS**

**A Dissertation**

**SUBMITTED TO THE GRADUATE FACULTY**

**in partial fulfillment of the requirements for the**

**degree of**

**Doctor of Philosophy**

**By**

**SIRIRUT VANICHAYOBON  
Norman, Oklahoma  
2002**

**UMI Number: 3038980**

**UMI<sup>®</sup>**

---

**UMI Microform 3038980**

**Copyright 2002 by ProQuest Information and Learning Company.**

**All rights reserved. This microform edition is protected against  
unauthorized copying under Title 17, United States Code.**

---

**ProQuest Information and Learning Company  
300 North Zeeb Road  
P.O. Box 1346  
Ann Arbor, MI 48106-1346**


**© Copyright by SIRIRUT VANICHAYOBON 2002  
All Rights Reserved.**

**POWER-SPEED TRADE-OFF  
IN  
PARALLEL PREFIX CIRCUITS**

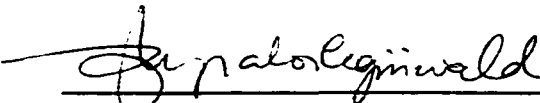
**A Dissertation APPROVED FOR THE  
SCHOOL OF COMPUTER SCIENCE**

**BY**

  
\_\_\_\_\_  
**Sudarshan K. Dhall, Committee Chair**

  
\_\_\_\_\_  
**S. Lakshmivaran**

  
\_\_\_\_\_  
**John K. Antonio**

  
\_\_\_\_\_  
**Le Gruenwald**

  
\_\_\_\_\_  
**Albert B. Schwarzkopf**

## **Acknowledgements**

**“No sweat, no gain” were the first words my professor said in the first class I attended at OU. I always remember them whenever I am down, and keep them as my encouragement.**

**I would like to express my deepest thanks and appreciation to my advisor, Prof. Sudarshan K. Dhall, and my co-advisor, Prof. S. Lakshmiarahan, for their confidence in my work and me. They both have guided me through the dissertation and provided support and encouragement. Without them, I would never have finished my work on this project. They are the wind beneath my wings. I have learned a lot from them both personally and professionally.**

**I am greatly indebted to Prof. John K. Antonio for his financial support, his kindness, and his guidance during the work as well as for his comments, suggestions, and support on my dissertation. I am very grateful to have had the opportunity to work on his project.**

**I extend my gratitude, appreciation and sincere thanks to Dr. Le Gruenwald for her valuable guidance, and support through my study, and for serving on my committee. I also would like to thank Dr. Albert Schwarzkopf for his time and support while serving on my committee.**

**I would like to thank the Thai government for bringing me to the USA, and for supporting me financially and motivating me. I would not dare to be in the States all by myself.**

**I would like to thank the Oklahoma Climatological Survey for giving me an opportunity to work with them.**

**All my life, I have met many people and have many good friends. I would like to thank all of them for touching me and giving me their love and support. In particular, I would like to thank Supawadee Poompuang for guiding me the way to live in the States. I would like to thank Sridhar Kulasekharan for his encouragement and helping me with everything I have asked for. I would like to thank Nathan Phillips for being my good friend, sending me encouragement and helping me with English. I would like to thank Vinod Choyi for being my good friend. I also would like to thank my officemates and my colleagues, Hongping Li for sharing invaluable discussion, Wang Jun, Manoj Suresh Kumar, Ming-Shan Su, Leonard Brown, and Brian Veale for giving me many laughs. I would like to thank Anurat Wisitsora, who destroys all my electronic doubts. I would like to thank Kemming Zhang for taking care of my sister when I am busy with my studies. Friends in Thailand and the USA, Sakaowrat Modthongkum, Nutharin Phusitphoykai, Chantarin Titawiriya, Aurawan Wiratanapokin, Maneerat Maneewong, Wimon Wongcharoen, Kitsiri Kaewpipat, Siriporn Laopiriyawong, Rossukon Laopaiboon,**

**Kulwadee L. Pigott, Charnnarong Saikaew and everyone, I would like to thank you all for being good listeners and for your tireless encouragement. I also want to thank my friends from Internet world who answer my many questions.**

**I would like to thank the School of Computer Science department's administrative staff and The University of Oklahoma for providing such a positive study environment.**

**I am grateful to learn Vipassana meditation technique from Teacher S. N. Goenka and to read the book, "Living, Loving and Learning" by Dr. Leo Buscaglia. They teach me to see through life and to live my life with joy.**

**I would like to thank my younger sister, Phanarat Vanitchyobol, for supporting and being with me whenever I need her.**

**I truly thank my brothers and sister in Thailand for their love and support, and for helping and taking care of my parents while I am away.**

**Finally, I would like to thank my very beloved mother and father, to whom this work is dedicated. They both work tirelessly to give their children an opportunity for education that they didn't have. I can't wait to be with them and make them proud.**

**May everyone be happy.**



# TABLE OF CONTENTS

|  |           |
|--|-----------|
| <b>CHAPTER 1. INTRODUCTION</b>                             | <b>1</b>  |
| <b>CHAPTER 2. PREFIX COMPUTATION</b>                       | <b>4</b>  |
| 2.1 Prefix Computation Model .....                         | 4         |
| 2.2 Prefix Circuit: An Overview .....                      | 11        |
| 2.2.1 The Serial Prefix Circuit .....                      | 12        |
| 2.2.2 The Divide-and-Conquer Parallel Prefix Circuit ..... | 13        |
| 2.2.3 The Ladner-Fischer Parallel Prefix Circuit .....     | 15        |
| 2.2.4 The Brent-Kung Parallel Prefix Circuit .....         | 17        |
| 2.2.5 The Snir Parallel Prefix Circuit .....               | 19        |
| 2.2.6 The LYD Parallel Prefix Circuit .....                | 25        |
| 2.2.7 The Shih-Lin Parallel Prefix Circuit .....           | 29        |
| 2.3 Comparison .....                                       | 31        |
| <b>CHAPTER 3 SOURCES OF POWER CONSUMPTION</b>              | <b>33</b> |
| 3.1 CMOS .....   | 33        |
| 3.2 Power Consumption .....                                | 35        |
| 3.2.1 Sources of Power Consumption .....                   | 35        |
| 3.2.2 Power Consumption and Fan-out .....                  | 42        |
| 3.3 The Circuit-level Simulation: PSpice .....             | 43        |
| <b>CHAPTER 4 POWER MODELING OF PREFIX CIRCUITS</b>         | <b>45</b> |
| 4.1 Step 1 - The Constant Output Capacitance .....         | 46        |

|   |   |            |
|---|---|------------|
| 4.1.1   | The Serial Prefix Circuit .....   | 47         |
| 4.1.2   | The Divide-and-Conquer Parallel Prefix Circuit .....                      | 48         |
| 4.1.3   | The Brent-Kung Parallel Prefix Circuit .....                              | 51         |
| 4.1.4   | The Ladner-Fisher Parallel Prefix Circuit .....                           | 56         |
| 4.1.5   | The Snir Parallel Prefix Circuit .....                                    | 57         |
| 4.1.6   | The Shih-Lin Parallel Prefix Circuit .....                                | 58         |
| 4.1.7   | The LYD Parallel Prefix Circuit .....                                     | 60         |
| 4.2   | Step 2 - Capacitance of Residual Circuit .....                            | 63         |
| 4.2.1   | The Serial Prefix Circuit .....   | 64         |
| 4.2.2   | The Divide-and-Conquer Parallel Prefix Circuit .....                      | 66         |
| 4.2.3   | The Brent-Kung Parallel Prefix Circuit .....                              | 69         |
| 4.2.4   | The Ladner-Fisher Parallel Prefix Circuit .....                           | 74         |
| 4.2.5   | The Snir Parallel Prefix Circuit and Shih-Lin Parallel Prefix Circuit ... | 74         |
| 4.2.6   | The LYD Parallel Prefix Circuit .....                                     | 76         |
| 4.3   | Comparison .....  | 79         |
| <br><b>CHAPTER 5 POWER-SPEED TRADE-OFF IN PREFIX CIRCUITS</b> |   | <b>81</b>  |
| 5.1   | Prefix Circuits at Fixed Voltage .....                                    | 82         |
| 5.2   | Effects of Voltage Scaling on Prefix Circuits.....                        | 84         |
| 5.3   | Summary .....   | 92         |
| <br><b>CHAPTER 6 ADDITION CIRCUITS</b>                        |   | <b>94</b>  |
| 6.1   | Adder: Theory .....   | 95         |
| 6.2   | Parallel Addition .....   | 99         |
| 6.2.1   | Binary Addition as a Prefix Problem .....                                 | 101        |
| <br><b>CHAPTER 7 SIMULATION RESULT</b>                        |   | <b>110</b> |
| 7.1   | Effect of Block Size on Adder Implementation .....                        | 110        |

|                             |  |            |
|-----------------------------|--|------------|
| 7.2                         | Effect of Prefix Circuit on Adder Implementation ..... | 114        |
| 7.3                         | Summary .....  | 117        |
| <b>CHAPTER 8 CONCLUSION</b> |  | <b>118</b> |
| <b>BIBLIOGRAPHY</b>         |  | <b>121</b> |
| <b>APPENDICES</b>           |  | <b>126</b> |
|                             | Appendix A .....                                       | 126        |
|                             | Appendix B .....                                       | 134        |
|                             | Appendix C .....                                       | 136        |
|                             | Appendix D .....                                       | 143        |
|                             | Appendix E .....                                       | 152        |

## LIST OF TABLES

|     |  |     |
|-----|--|-----|
| 2.1 | A comparison of the seven prefix circuits illustrated in this chapter .....  | 32  |
| 4.1 | The constant output capacitance table for $BK(N)$ .....  | 53  |
| 4.2 | Expression of the constant output capacitance .....  | 63  |
| 4.3 | The residual circuit table for $BK(N)$ .....   | 71  |
| 4.4 | The effective circuit capacitance of prefix circuits.....  | 80  |
| 5.1 | Estimated power consumption based on Eq. 4.1 for various prefix circuits<br>for $N = 64$ .....   | 89  |
| 6.1 | Adder truth table .....  | 96  |
| 6.2 | Gate count of a $N$ -bit adder .....   | 109 |
| 7.1 | Gate count, delay time, power consumption, and power-delay-product of<br>different design of 8-bit, 16-bit, 32-bit, and 64-bit adders using the<br>divide-and-conquer prefix circuit ..... | 111 |
| E.1 | A comparison of the exact capacitance values and the estimated capacitance<br>values of the Brent-Kung prefix circuit .....  | 152 |

## LIST OF FIGURES

|      |   |    |
|------|---|----|
| 2.1  | An illustration of the prefix computation model .....   | 5  |
| 2.2  | An illustration of an operation mode and a repeater node .....  | 6  |
| 2.3  | An illustration of the prefix circuit's layout .....  | 6  |
| 2.4  | The prefix circuit with 4 inputs, size=4, depth=2 .....   | 8  |
| 2.5  | The prefix circuit <i>A</i> with 4 inputs, size=3, depth=3 .....  | 9  |
| 2.6  | The prefix circuit <i>B</i> with 4 inputs, size=4, depth=2 .....  | 9  |
| 2.7  | An illustration of the serial prefix circuit, $S(N)$ , derived from [LD94] .....  | 12 |
| 2.8  | The serial circuit with 10 inputs, $S(10)$ , size=9, depth=9 .....  | 13 |
| 2.9  | An illustration of the divide-and-conquer prefix circuit, $DC(N)$ , derived from [LD94] .....                                       | 14 |
| 2.10 | The divide-and-conquer parallel prefix circuit with 8 inputs, $DC(8)$ , size=12, depth=3 .....                                      | 15 |
| 2.11 | An illustration of the Ladner-Fischer parallel prefix circuit when $k=0$ , $LF_0(N)$ , derived from [LF80].....                     | 16 |
| 2.12 | An illustration of the Ladner-Fischer parallel prefix circuit when $k \neq 0$ , $LF_k(N)$ , derived from [LF80].....                | 16 |
| 2.13 | Examples of Ladner-Fisher parallel prefix circuits with 8 inputs .....  | 17 |
| 2.14 | A Brent-Kung parallel prefix circuit, $BK(N)$ based on divide-and-conquer strategy ( $o$ =odd, $e$ =even), derived from [LD94]..... | 18 |
| 2.15 | An illustration of the Brent-Kung parallel prefix circuit, $BK(8)$ , size=11, depth=4 .....   | 19 |
| 2.16 | The Snir prefix circuit, $SN(N) = CR(N_1) \cdot S(N_2)$ .....   | 20 |
| 2.17 | An illustration of the uncompressed layered prefix circuit, size=11, depth=5 .....  | 22 |
| 2.18 | An illustration of the compressed layered prefix circuit, size=1, depth=4 .....   | 23 |
| 2.19 | The Snir prefix circuit, $SN(19)$ , size=28, depth=8 .....  | 24 |
| 2.20 | The $Q(7)$ prefix circuit .....   | 26 |

|      |   |    |
|------|---|----|
| 2.21 | The structure of $LYD(N)$ , derived from [LD94] .....   | 28 |
| 2.22 | The $LYD(19)$ prefix circuit with size 31 and depth 5 .....   | 29 |
| 2.23 | The $SL(N)$ prefix circuit, $SL(N) = CR(N_1) \cdot S(N_2)$ .....  | 30 |
| 2.24 | The $SL(19)$ prefix circuit, size=30 and depth=6 .....  | 31 |
|      |   |    |
| 3.1  | $P$ -type and $N$ -type transistor and their characteristics .....  | 34 |
| 3.2  | CMOS inverter .....   | 34 |
| 3.3  | CMOS NAND gate .....  | 35 |
| 3.4  | CMOS NOR gate .....   | 35 |
| 3.5  | The leakage current from the gate to the drain of a transistor .....  | 36 |
| 3.6  | An illustration of short-circuit when both $P$ -type and $N$ -type transistor<br>being in the on state at the same time ..... | 36 |
| 3.7  | An illustration of capacitance charging .....   | 37 |
| 3.8  | Plots of normalized delay vs. supply voltage for a variety of different logic<br>circuits, derived from [CB95] .....          | 38 |
| 3.9  | An illustration of the glitching behavior of a chain of eight NAND gates<br>[RCN01] .....                                     | 40 |
| 3.10 | An illustration of extra transaction activity, derived from [CB95] .....  | 41 |
| 3.11 | Effect of fan-out on power consumption of a 2-input XOR gate .....  | 43 |
|      |   |    |
| 4.1  | An illustration of the serial prefix circuit, $S(N)$ .....  | 47 |
| 4.2  | An illustration of the serial prefix circuit, $S(N)$ , built from $S(N-1)$ .....  | 47 |
| 4.3  | An illustration of the divide-and-conquer prefix circuit, $DC(N)$ , built from<br>$DC(N/2)$ , derived from [LD94] .....       | 49 |
| 4.4  | A Brent-Kung parallel prefix circuit, $BK(N)$ , divided into three parts<br>( $o$ =odd, $e$ =even), derived from [LD94] ..... | 51 |
| 4.5  | Part C, the distribution of $N/2 - 1$ nodes .....   | 54 |
| 4.6  | The $SN(N)$ circuit, $SN(N) = CR(N_1) \cdot S(N_2)$ .....   | 57 |
| 4.7  | The $SL(N)$ circuit, $SL(N) = CR(N_1) \cdot S(N_2)$ .....   | 59 |
| 4.8  | The structure of $LYD(N)$ , derived from [LD94] .....   | 61 |

|      |  |    |
|------|--|----|
| 4.9  | The serial prefix circuit for $N$ inputs with fan-out shown in solid lines .....   | 64 |
| 4.10 | The residual circuit of the serial prefix circuit, shown in solid lines .....  | 65 |
| 4.11 | The illustration of the residual circuit of the $S(N)$ , built from $S(N-1)$ .....   | 65 |
| 4.12 | The divide-and-conquer prefix circuit, $DC(N)$ , with fan-outs shown in solid lines, derived from [LD94] .....   | 67 |
| 4.13 | The residual circuit of the divide-and-conquer prefix circuit, $DC(N)$ , shown in solid lines .....  | 67 |
| 4.14 | The residual network of the Brent-Kung parallel prefix circuit, $BK(N)$ , divided in to 3 parts .....  | 70 |
| 4.15 | Part C, the Distribution of $N/2 - 1$ nodes .....  | 72 |
| 4.16 | The $SN(N)$ , and $SL(N)$ circuit .....  | 75 |
| 4.17 | The structure of $LYD(N)$ , derived from [LD94] .....  | 77 |
| -    |  |    |
| 5.1  | Power consumption of the 32-bit XOR parallel prefix circuits, obtained through PSpice simulation .....   | 82 |
| 5.2  | Estimated power consumption of prefix circuits when $N=32$ bits .....  | 82 |
| 5.3  | Comparison between simulation results and modified estimation results for $N=32$ bits. The modified estimation enhances the original estimation by including a component of power proportionally to circuit size ..... | 83 |
| 5.4  | Power consumption of the XOR parallel prefix circuits at fixed voltage, obtained through PSpice simulation .....   | 84 |
| 5.5  | Estimated power consumption of prefix circuits with fixed voltage .....  | 84 |
| 5.6  | Plot of supply voltage vs. normalized delay from [10] .....  | 85 |
| 5.7  | Estimated delay of parallel prefix circuits when $N=64$ .....  | 87 |
| 5.8  | Estimated power consumption of parallel prefix circuits when $N=64$ .....  | 87 |
| 5.9  | Estimated power-delay product of parallel prefix circuits when $N=64$ .....  | 87 |
| 5.10 | Delay of the 64-bit XOR parallel prefix circuits, obtained through PSpice simulation .....   | 90 |
| 5.11 | Power consumption of the 64-bit XOR parallel prefix circuits, obtained through PSpice simulation .....   | 90 |

|      |  |     |
|------|--|-----|
| 5.12 | Power-delay product of the 64-bit XOR parallel prefix circuits, obtained through PSpice simulation .....   | 90  |
| 6.1  | The Block diagram of the full-adder circuit .....  | 96  |
| 6.2  | The K-Maps for the full-adder-circuit .....  | 96  |
| 6.3  | A chain of $N$ full-adders .....   | 98  |
| 6.4  | A full-adder circuit .....   | 98  |
| 6.5  | Three stages of the implementation of the fast adder .....   | 100 |
| 6.6  | The partition of all $p_i$ and $g_i$ into $r$ group with $q$ members each .....  | 101 |
| 6.7  | A parallel scheme for computing the carry, derived from [LD90] .....   | 102 |
| 7.1  | The plots of delay, power consumption, and power-delay-product of different design of 8-, 16, 32-, and 64-bit adders using the divide-and-conquer prefix circuit ..... | 112 |
| 7.2  | The illustration of the effect of the block size on other factors .....  | 113 |
| 7.3  | The plot of power-delay-product of the divide-and-conquer, the LYD, and the Shih-Lin prefix circuits .....   | 115 |
| 7.4  | The plot of power consumption of four prefix circuits using in carry calculation in 64-bit adder implementing with block size of sixteen .....                         | 116 |
| A.1  | Inverter .....   | 126 |
| A.2  | Switch model of CMOS transistor .....  | 127 |
| A.3  | The equivalent action of an inverting gate when a step input charges and discharges the capacitor .....  | 128 |
| A.4  | Charging and discharging exponential curves for an $RC$ network .....  | 129 |
| A.5  | The plot of the delay vs. $V_{DD}$ .....   | 130 |
| A.6  | CMOS inverter's input and output waveforms .....   | 132 |
| A.7  | CMOS inverter's power and energy waveforms .....   | 133 |
| B.1  | A CMOS inverter .....  | 134 |



|     |  |     |
|-----|--|-----|
| B.2 | A CMOS AND gate .....  | 134 |
| B.3 | A CMOS OR gate .....   | 135 |
| B.4 | A CMOS XOR gate .....  | 135 |
| C.1 | The worst case input of XOR gate (i.e., the first input is equal to 0 and the other inputs are 0 $\rightarrow$ 1), causing the output to ripple the most ..... | 136 |
| C.2 | The 8-bit XOR gate implemented with the serial prefix circuit .....  | 137 |
| C.3 | The outputs of 8-bit XOR gates implemented with the serial prefix circuit, showing the longest ripple (the maximum number of switching) .....                  | 138 |
| C.4 | Delay of 8-bit XOR gates implemented with the serial prefix circuit from PSpice simulation .....   | 139 |
| C.5 | Energy of 8-bit XOR gates implemented with the serial prefix circuit from PSpice simulation .....  | 139 |
| C.6 | The 8-bit XOR gate implemented with the divide-and-conquer prefix circuit .....  | 140 |
| C.7 | The outputs of 8-bit XOR gates implemented with the divide-and-conquer prefix circuit, showing the longest ripple (the maximum number of switching) .....      | 141 |
| C.8 | Delay of 8-bit XOR gates implemented with the divide-and-conquer prefix circuit from PSpice simulation .....   | 142 |
| C.9 | Energy of 8-bit XOR gates implemented with the divide-and-conquer prefix circuit from PSpice simulation .....  | 142 |
| D.1 | Preprocessing: carry propagate bits and carry generate bits .....  | 143 |
| D.2 | Postprocessing: $s_i = a_i \oplus b_i \oplus c_{i-1}$ .....  | 143 |
| D.3 | The implementation of $E_i$ , $i = 1$ .....  | 144 |
| D.4 | The implementation of carry bits .....   | 145 |
| D.5 | The implementation of $E_i$ , $1 \leq i \leq 2$ .....  | 145 |
| D.6 | The implementation of prefix circuit with 4 inputs .....   | 146 |
| D.7 | The implementation of carry bits .....   | 147 |

|             |   |            |
|-------------|---|------------|
| <b>D.8</b>  | <b>The implementation of <math>E_i</math>, <math>1 \leq i \leq 4</math></b> | <b>148</b> |
| <b>D.9</b>  | <b>The implementation of prefix circuit</b>                                 | <b>148</b> |
| <b>D.10</b> | <b>The implementation of carry bits</b>                                     | <b>149</b> |
| <b>D.11</b> | <b>The implementation of <math>E_i</math>, <math>1 \leq i \leq 8</math></b> | <b>150</b> |
| <b>D.12</b> | <b>The implementation of prefix circuit with 4 inputs</b>                   | <b>150</b> |
| <b>D.13</b> | <b>The implementation of carry bits</b>                                     | <b>151</b> |

## **Abstract**

Optimizing area and speed in parallel prefix circuits have been considered important for long time. The issue of power consumption in these circuits, however, has not been addressed. This dissertation presents a comparative study of different parallel prefix circuits from the point of view of power-speed trade-off. The power consumption and the power-delay product of seven parallel prefix circuits were compared. A linear output capacitance assumption, combined with PSpice simulations, is used to investigate the power consumption in the circuits. The degrees of freedom studied include different parallel prefix algorithms and voltage scaling. The results show that the use of the linear output capacitance assumption provides results that are consistent with those obtained using PSpice simulations. Because of the size-depth trade-off characteristic of prefix circuits, the results also show that parallelism of prefix circuits at a certain level coupled with the use of low supply voltage can be used to reduce the power-delay product to attain a desired throughput beyond the minimum possible. The study enables us to understand the power consumption behavior of prefix circuits, and to pick the suitable prefix circuit for the acceptable power consumption in the prefix with a given throughput. Circuit designers can then choose the best prefix circuit for a particular application.

# **CHAPTER 1**

## **INTRODUCTION**

The three most widely accepted metrics for measuring the quality of an integrated circuit are its area, speed, and power consumption. Optimizing area and speed have been considered important for long time, but minimizing power consumption has been gaining prominence only recently [Bel01, BM00, CB95, GNHF01, Hub00, Mil00, RP00, RP96]. One important reason for minimizing power consumption of a circuit is the proliferation of portable electronic systems, such as laptops, mobile phones and wireless devices, where maximizing battery life is important. Since it is desirable to minimize the size and weight of batteries in such devices, while increasing the time between battery recharges, finding methods of reducing power consumption has assumed considerable importance recently.

In this dissertation, we study power-speed trade-off for prefix circuits. The prefix circuits play an important role in many applications. It appears in a number of areas such as the carry-look-ahead adder, ranking, packing, radix sort, etc. [LD94]. Many new approaches for prefix circuits with the goal of optimizing depth (i.e., speed) and size (i.e., area) have been proposed [BK82, LF80, LD94, LS99, Snir86]. As a result, performance in terms of the speed and area has improved. The issue of power consumption in these circuits, however, has not been addressed. Therefore, our goal is to make a comparative study of different prefix circuits from the point of view of power-speed trade-off in order

to facilitate the design choices, specifications, and resource limitations. In this study, we use the power-delay product as a quality measure for the prefix circuits. The power-delay product is the product of the circuit's power consumption and propagation delay, which represents the energy consumed by the circuit per operation.

Two issues have been addressed in this dissertation. The first deals with our proposed power modeling of prefix circuits. Then, the model, combined with PSpice simulations, is used to investigate the power consumption in the circuits considered. The simulations were carried out on both fixed and scaled supply voltage. It is found that amongst the parallel prefix circuits the circuit having the shortest depth (the divide-and-conquer prefix circuit) consumes the most power. Also according to PSpice simulations, the power-delay product of the LYD prefix circuit seems to be the best (lowest) amongst the circuits considered while the power-delay product of the divide-and-conquer is the highest. The second issue deals with an investigation of the binary adders using selected prefix algorithms. A parameter in the implementation of these circuits is the choice of block size for computing carries in parallel. The 8-, 16-, 32-, and 64-bit binary adders were implemented and simulated on PSpice. The performance was measured and compared. In regard of power-delay product, we have found that an optimum block size falls somewhere around the middle among the various possible block sizes.

The rest of this dissertation is divided into seven chapters. Chapter II presents a literature survey on the various prefix circuits and discusses the current state of the art in this field. Chapter III reviews the sources of power consumption in CMOS circuits and presents strategies to estimate power consumption of the circuits. In addition, Chapter III briefly introduces the circuit simulation tool called PSpice. Chapter IV focuses on

**modeling the power consumption of the prefix circuits. The analysis of the power-speed trade-off of various prefix circuits is described in Chapter V. Chapter VI introduces the basic addition principle and structure as well as the formulation of carry propagation as a prefix problem. The simulation studies of adders are given in Chapter VII. Finally, the main results of the dissertation are summarized in Chapter VIII.**

-

## CHAPTER 2

### PREFIX COMPUTATION

As parallel-processing computers have proliferated, the notion of prefix computation has gained considerable attention in the literature and it played an important role in parallel algorithms. In 1963, Ofman, a Russian Mathematician, was a pioneer in introducing the use of prefix computation for fast binary adder circuits. The prefix computation appears in a number of areas such as the carry-look-ahead adder, the ranking, the packing, the radix sort, the finite state transducers, and the solutions of linear recurrences [LD94]. In this chapter, the prefix computation model is introduced. Then a survey of the seven well-known prefix circuits is presented.

#### 2.1 Prefix Computation Model

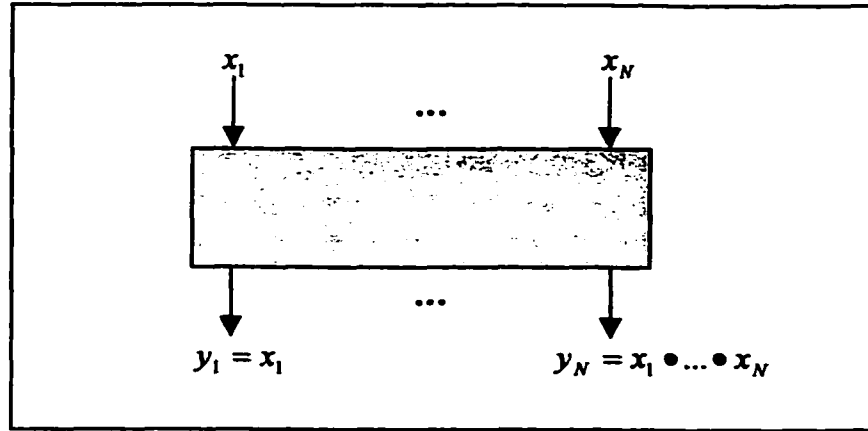
A *prefix computation* [LD94], or simply the *prefix circuit*, is the process of taking  $N$  inputs values  $x_1, x_2, \dots, x_{N-1}, x_N$  and producing  $N$  output values  $y_1, y_2, \dots, y_{N-1}, y_N$  such that

$$y_1 = x_1,$$

$$y_i = y_{i-1} \bullet x_i = x_1 \bullet x_2 \bullet \dots \bullet x_{i-1} \bullet x_i, \quad \text{for } 2 \leq i \leq N$$

and  $\bullet$  is an associative binary operation as shown in Figure 2.1. In other words, each  $y_i$  is obtained by “operating” together the first  $i$  elements of the sequence of  $x_i$  --hence, the term “prefix.” As an example, suppose that  $x_i = 1$  for  $1 \leq i \leq N$ , and let  $\bullet$  be the

ordinary addition. Then,  $y_1 = x_1 = 1$ ,  $y_2 = y_1 + x_2 = 2$ , and so on. Therefore, the prefix circuit produces  $y_i = i$  for  $1 \leq i \leq N$ .



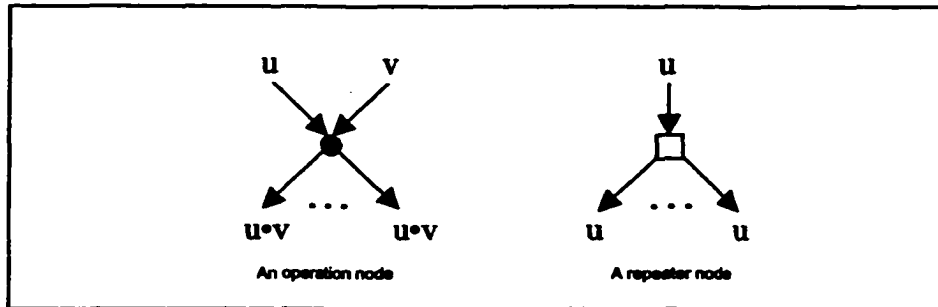
**Figure 2.1:** An illustration of the prefix computation model.

The inputs of the prefix circuit,  $x_i$ 's, can be anything depending on its application. If the input is either an integer, real number, or complex number and its operation is one of the two arithmetic operators (i.e.,  $+$ , and  $\times$ ), we call the circuit as an *arithmetic circuit*. If the input is a Boolean element (for example,  $\{0, 1\}$  or  $\{true, false\}$ ) associated with a Boolean operator we call it as a *Boolean circuit*.

A prefix circuit with  $N$  inputs can also be viewed as a directed acyclic graph  $G = (V, E)$  with  $N$  input vertices,  $N$  output vertices, and at least  $N-1$  internal vertices. These vertices will be referred to as input nodes, output nodes, and internal nodes, respectively. An internal node is neither an input nor an output node. There are two types of internal nodes: operation nodes and repeater nodes. An  $N$ -input prefix circuit has at least  $N-1$  operation nodes and has zero or more repeater nodes. An illustration of an operation node and a repeater node is shown in Figure 2.2. An operation node shown as a

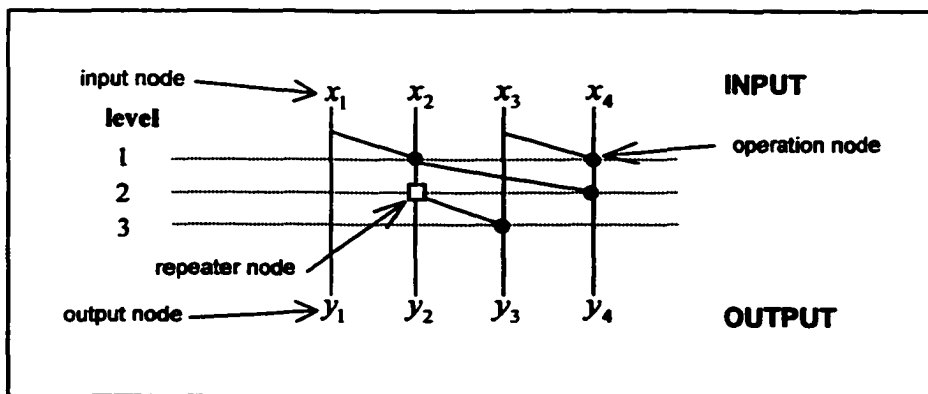


black dot, ●, takes two inputs and produces one output. A repeater node shown as a small square, □, takes one input and produces as output one or more copies of its input.



**Figure 2.2:** An illustration of an operation node and a repeater node.

In the prefix circuit's layout, vertical lines identify the inputs and outputs. The inputs are the lines leading from the top while the outputs are the lines leading to the bottom. As an example, Figure 2.3 illustrates the layout and the components of a prefix circuit. The numbers along the left-hand side of the layout give the depth (level) of the operation nodes on the right. Note that the first output node in the prefix circuit is from the first input node and the other outputs are from the internal nodes.



**Figure 2.3:** An illustration of the prefix circuit's layout.

The metrics for measuring the performance of a prefix circuit are the circuit size, depth, fan-in, and fan-out. These are explained in detail in the following.

### **Circuit Size**

The *size* of a prefix circuit,  $size(N)$ , is the total number of operation nodes in the circuit. The size represents the amount of space required for the circuit. The circuit with smaller size occupies less chip area in VLSI implementation [WE93]. One of the design aims may be minimizing the size of the circuit.

### **Circuit Depth**

The *depth* of a prefix circuit,  $depth(N)$ , is the length of the longest path measured in terms of the number of operations along the path in the circuit from its input nodes to its output nodes. If a prefix circuit produces its outputs at depths  $d_1, d_2, \dots, d_k$ , the depth of a circuit is the maximum of  $\{d_1, d_2, \dots, d_k\}$ . In other words, the depth of a prefix circuit is the maximum depth of its outputs. The circuit depth is related to its computation time. In VLSI implementation, a circuit with smaller depth is generally faster than one with greater depth when the fan-out of most nodes in the two circuits is similar [WE93]. A prefix circuit is *depth-optimal* if the circuit has the smallest depth among all possible circuits.

### **Circuit Fan-in and Fan-out**

The *fan-in* of a prefix circuit is the maximum fan-in of all nodes in the circuit. The fan-in

of a node is the number of inputs the node has in the path being exercised. Thus, the fan-in of a node is defined as the node's indegree. The fan-in of a node except the input nodes can be either bounded or unbounded. A node has unbounded fan-in if the fan-in is not fixed. In this study, unless otherwise stated, we are interested in the prefix circuit with the fan-in of two, which represents a binary operation.

The *fan-out* of a prefix circuit is the maximum fan-out of all nodes in the circuit. The fan-out of a node is the number of outputs the node produces to drive the other nodes. The fan-out of a node is defined as the node's outdegree. A node has unbounded fan-out if the fan-out is not fixed. In the circuit shown in Figure 2.4, the nodes have fan-out of three, and one, respectively. In the following, unless otherwise stated, we assume that the fan-out of the prefix circuit is a function of  $N$ .

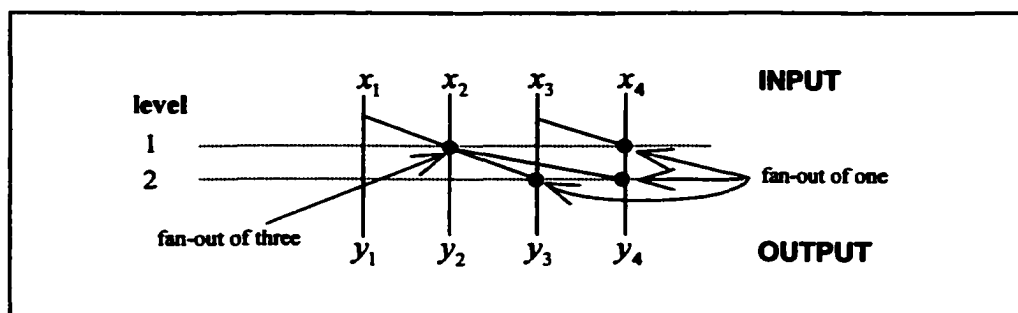


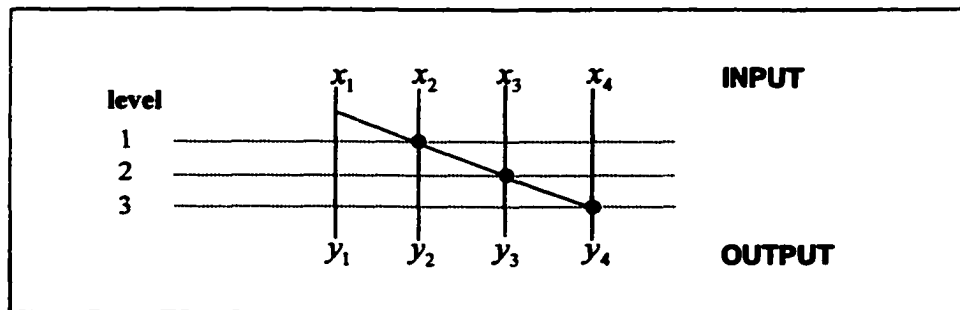
Figure 2.4: The prefix circuit with 4 inputs, size=4, depth=2.

### Size-depth trade-off

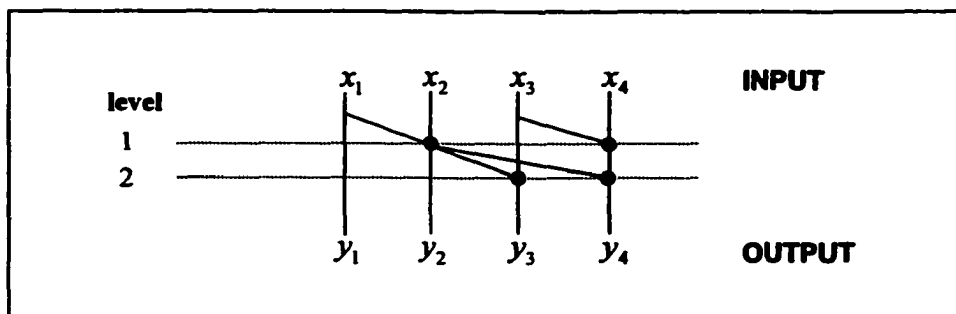
Ladner and Fisher [LF80] were the first to introduce the important property of the prefix circuit, the size-depth trade-off. They showed that a decrease in the circuit depth can be achieved by an increase in the circuit size and *vice versa*. Snir [Sni86] further strengthened this notion by proving the following result:

**Theorem [Sni86]** The sum of the size and depth of a prefix circuit,  $G(N)$ , is lower bounded by  $2N - 2$ , i.e.,  $size(G(N)) + depth(G(N)) \geq 2N - 2$ .

This bound is tight in the sense that there are prefix circuits which actually achieve this bound. Figure 2.5 and Figure 2.6 show the size-depth trade-off of prefix circuits. The circuit  $A$  and circuit  $B$  produce the same outputs,  $y_i$  where  $1 \leq i \leq 4$ .



**Figure 2.5:** The prefix circuit  $A$  with 4 inputs,  $size = 3, depth = 3$ .



**Figure 2.6:** The prefix circuit  $B$  with 4 inputs,  $size = 4, depth = 2$ .

The circuit  $A$  in Figure 2.5 has size 3 and depth 3 while the circuit  $B$  in Figure 2.6 has larger size but smaller depth (i.e., size is 4 and depth is 2). Hence the circuit  $B$  is faster but has to do more work than the circuit  $A$ . Both circuits are (size, depth)-optimal.

The *deficiency* of a prefix circuit [Sni86] is defined as

$$deficiency = size + depth - (2N - 2).$$

Since  $2N - 2$  is the lower bound on the sum of size and depth, clearly, if *deficiency* = 0, then the prefix circuit is said to be *(size, depth)-optimal*.

In this study, all inputs are assumed to be at level zero. Unless otherwise stated, we assume the number of inputs is  $N$  which need not to be a power of two. The input nodes will be denoted as  $x_1, x_2, \dots, x_{N-1}, x_N$ . For integers  $i$  and  $j$  in the range  $1 \leq i \leq j \leq N$ , we define

$$i : j = x_i \bullet x_{i+1} \bullet \dots \bullet x_j.$$

Thus, for  $i = 1, 2, \dots, N$ , we have  $i : i = x_i$ , since the composition of just one input  $x_i$  is itself. For  $i, j$ , and  $k$  satisfying  $1 \leq i \leq j \leq k \leq N$ , we also have the identity

$$i : k = i : j - 1 \bullet j : k,$$

since the  $\bullet$  operator is associative [LD94]. For purposes of notational convenience, the input values  $x_i$ 's are labeled with the integer  $i$ , and the output values  $y_i$ 's are labeled with  $1 : i$ , where  $1 : i = x_1 \bullet x_2 \bullet \dots \bullet x_{i-1} \bullet x_i$  for  $1 \leq i \leq N$ . All input nodes have zero fan-in and a fan-out of at most two. The output nodes have at most one fan-in and zero fan-out. For the internal nodes, the operation nodes have two fan-ins while the repeater nodes have only one fan-in. However, both have unbounded fan-out. We will use this structure to represent a prefix circuit for the rest of the study. This type of prefix circuit is termed as a *conservative circuit* [Sni86]. If a prefix circuit produces its last output (i.e.,  $1 : N$ ) at level  $\lceil \lg N \rceil$ , we call such a circuit as a *restricted prefix circuit*. We will see in the next section that the restricted prefix circuit plays a major role in many parallel prefix circuits.

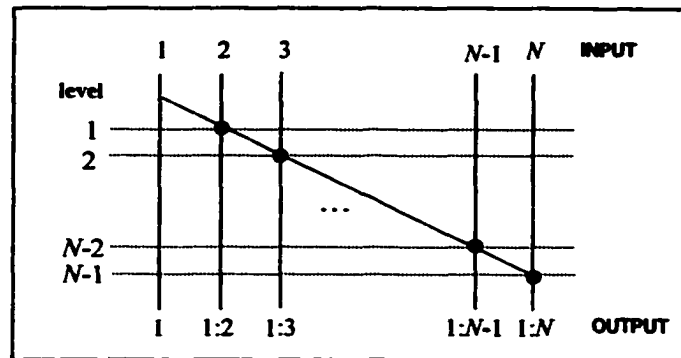
## 2.2. Prefix Circuits: An Overview

In this section, we review the design of prefix circuits commonly found in literature. We first introduce the serial prefix circuit. The size and depth complexity of this circuit is  $O(N)$ . Then the parallel prefix circuits based on the divide-and-conquer approach are presented. These circuits are known as the divide-and-conquer prefix circuit, the Ladner-Fischer prefix circuit [LF80], and the Brent-Kung prefix circuit [BK82]. By way of comparison with the serial prefix circuit, the size of the divide-and-conquer prefix circuit increases to  $O(N \lg N)$  whereas the size of the Brent-Kung prefix circuit is  $O(N)$ . However, the computation time of all three circuits is improved to  $O(\lg N)$ . Ladner-Fischer prefix circuit is the first circuit that shows the trade-off between the circuit size and circuit depth. Finally, the prefix circuits that are (size, depth)-optimal and are based on the combination of two or more prefix circuits are presented. Each circuit has its own methodology to divide inputs into two or more parts, intending to reduce the circuit depth. For example, the Snir prefix circuit [Sni86] and the Shih-Lin prefix circuit [LS99] are composed of two parts. The first part is the non-optimal prefix circuit called the compressed layered prefix circuit and the second part is the serial prefix circuit. The Lakshmivaranhan, Yang and Dhall's prefix circuit (LYD prefix circuit) is composed of four parts and has the shortest circuit depth among all (size, depth)-optimal prefix circuits. Note that all of the circuits, except the serial prefix circuit, have unbounded fan-out and operate in parallel: more than one operations are performed at a time. Instead of producing the outputs one by one at a time as in the serial prefix circuit, they produce

outputs  $y_1, y_2, \dots, y_{N-1}, y_N$  more quickly. In the following, unless specified otherwise, all the logarithms are to the base 2.

### 2.2.1 The Serial Prefix Circuit

The serial prefix circuit,  $S(N)$ , produces the outputs one by one at a time. It is straightforward to construct the serial prefix circuit. The layout of the circuit for  $N$  inputs is illustrated in Figure 2.7. The  $S(N)$  circuit is formed by cascading  $N - 1$  operation nodes, and feeding the output of the previous level directly into the input of the current level. Each operation node has a fan-in of exactly two and a fan-out of two except the last operation which has only one fan-out.



**Figure 2.7:** An illustration of the serial prefix circuit,  $S(N)$ , derived from [LD94].

The last output is produced at depth  $N - 1$ . There is one operation node at each level so the size of the circuit is  $N - 1$ , which is the smallest size among all other circuits. Moreover, the serial prefix circuit is a (size, depth)-optimal prefix circuit since the sum of its size and depth is  $2N - 2$ . However, the circuit is neither depth-optimal nor a restricted circuit. Due to the size-depth trade-off rule, the serial prefix circuit has the longest depth

among all other circuits (i.e., slowest circuit). Thus, all other faster circuits must have sizes larger than  $N - 1$ . Figure 2.8 shows the serial prefix circuit for  $N = 10$ . The output from the  $i^{\text{th}}$  level is the input of the  $(i + 1)^{\text{th}}$  level. For example, the output of node labeled 1:2 is the input of node labeled 1:3. The circuit size and depth are 9. Even though the serial prefix circuit uses only  $N - 1$  operations, the time taken is also  $N - 1$ . Hence we have to look at other alternatives for better performance.

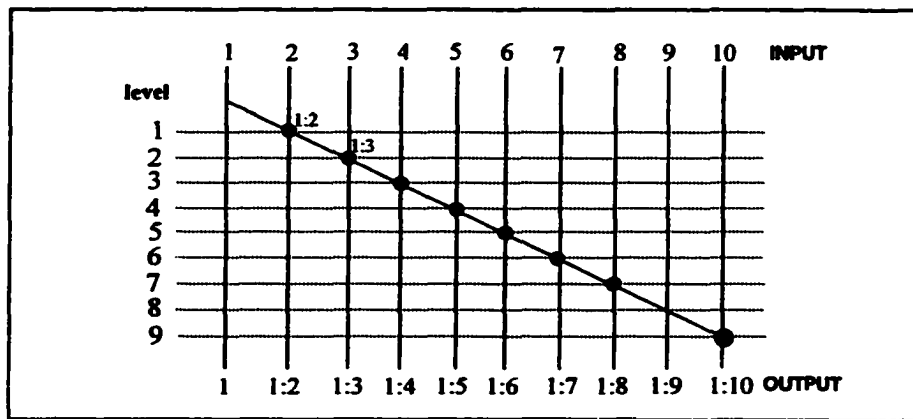


Figure 2.8: The serial circuit with 10 inputs,  $S(10)$ ,  $size = 9$ ,  $depth = 9$ .

### 2.2.2 The Divide-and-Conquer Parallel Prefix Circuit

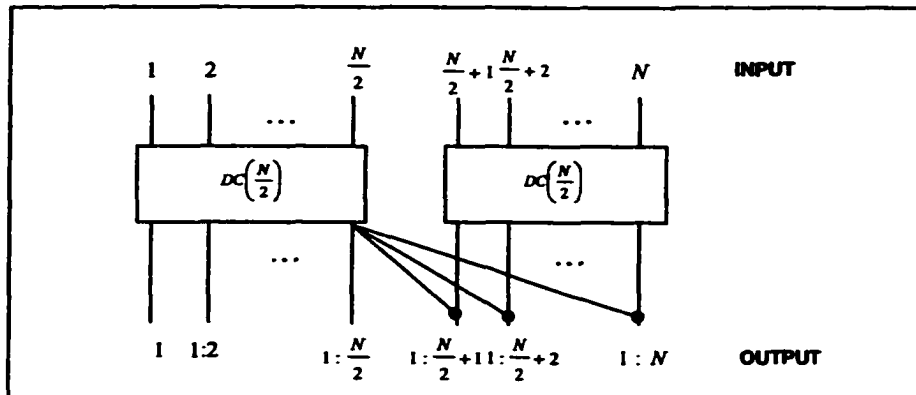
The divide-and-conquer prefix circuit reduces the depth to  $\lg N$ , as opposed to  $N - 1$  needed by the serial prefix circuit, by using parallel operations and the well-known divide-and-conquer strategy. The construction of the divide-and-conquer parallel prefix circuit with  $N$  inputs, denoted as  $DC(N)$ , is illustrated in Figure 2.9. The  $DC(N)$  circuit can be built from two  $DC(N/2)$  circuits, recursively. Thus the  $size(DC(N))$  is the size of two  $DC(N/2)$  circuits plus additional connection nodes, which are  $N/2$  in number.



Similarly, the depth of  $DC(N)$  is one more than the depth of  $DC(N/2)$ . Therefore, the following recurrences for the size and depth of this circuit are immediate:

$$size(DC(N)) = 2size(DC(\frac{N}{2})) + \frac{N}{2}, \quad \text{with} \quad size(DC(2)) = 1.$$

$$depth(DC(N)) = depth(DC(\frac{N}{2})) + 1, \quad \text{with} \quad depth(DC(2)) = 1.$$



**Figure 2.9:** An illustration of the divide-and-conquer prefix circuit,  $DC(N)$ , derived from [LD94].

Solving, for  $size(DC(N))$ , we get

$$size(DC(N)) = \frac{N}{2} \lg N = O(N \lg N)$$

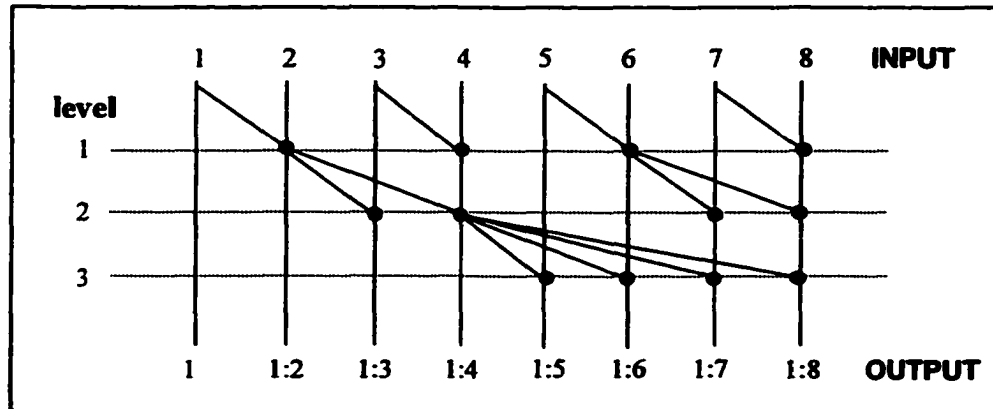
Solving, for  $depth(DC(N))$ , we get

$$depth(DC(N)) = \lg N = O(\lg N)$$

Thus, the  $DC(N)$  circuit takes only  $\lg N$  time. This circuit is, therefore, depth optimal.

However, the circuit size is much bigger than the serial prefix circuit, increasing to  $O(N \lg N)$ . However, the circuit is not (size, depth)-optimal because the sum of the size and depth of the circuit is much more than the lower bound  $2N - 2$ , for  $N > 4$ . Figure

2.10 shows the circuit  $DC(N)$  for  $N = 8$ . The circuit size and depth are 12 and 3, respectively. In this case we have reduced the depth to  $\lg N$  but the number of operations increases to  $(N/2)\lg N$ .



**Figure 2.10:** The divide-and-conquer parallel prefix circuit with 8 inputs,  $DC(8)$ , size = 12, depth = 3.

### 2.2.3 The Ladner-Fischer Parallel Prefix Circuit

From the above description, we see that the serial circuit has longer depth but smaller size whereas the divide-and-conquer parallel prefix circuit has smaller depth but larger size. Ladner and Fischer [LF80] were the first to discuss the size-depth trade off in prefix circuits -- a reduction of the circuit depth is achieved at the cost of an increase in the number of operations. They introduced a family of circuits,  $LF_k(N)$ , where  $k$  denotes the depth above  $\lceil \lg N \rceil$ , with  $0 \leq k \leq \lceil \lg N \rceil$ . Based on the divide-and-conquer strategy,  $LF_0(N)$  and  $LF_k(N)$  (when  $k \neq 0$ ) are defined recursively as shown in Figure 2.11 and 2.12, respectively.

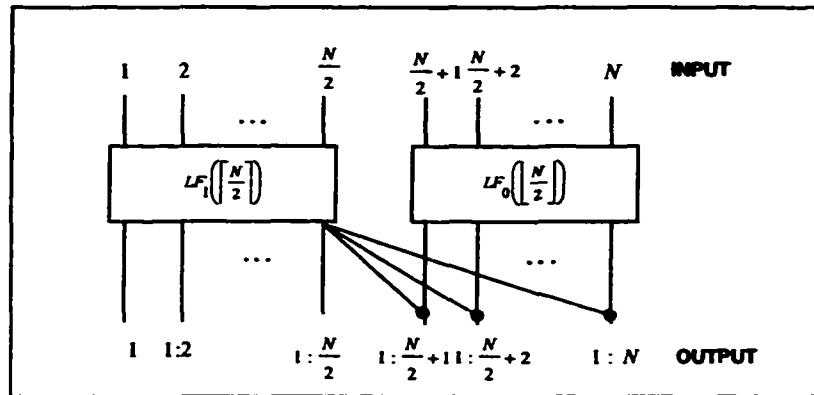
The last output,  $1:N$ , of the  $LF_k(N)$  circuit for all  $N$  and  $k$  is available in  $\lceil \lg N \rceil$

units of time so the circuit is a restricted parallel prefix circuit. The circuit size depends on the value of  $k$  such that

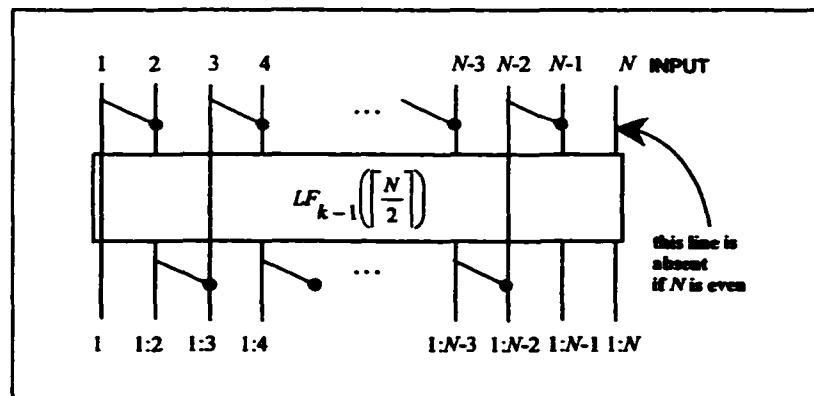
$$size_0(N) = 4N - F(5 + \lg N) + 1,$$

$$size_k(N) = 2N(1 + \frac{1}{2^k}) - F(5 + \lg N - k) - k + 1,$$

where  $F(N)$  is the  $N^{\text{th}}$  Fibonacci number and for  $k \geq 1$ . Note that when  $N$  is not a power of 2, this solution is not precise. The circuit depth by construction is  $\lceil \lg N \rceil \leq depth(LF_k(N)) \leq 2\lceil \lg N \rceil - 2$ .



**Figure 2.11:** An illustration of the Ladner-Fischer parallel prefix circuit when  $k = 0$ ,  $LF_0(N)$ , derived from [LF80].



**Figure 2.12:** An illustration of the Ladner-Fischer parallel prefix circuit when  $k \neq 0$ ,  $LF_k(N)$ , derived from [LF80].

The Ladner-Fischer circuit is depth-optimal when  $k = 0$ . The circuit is not (size, depth)-optimal because  $size(LF_k(N)) + depth(LF_k(N)) > 2N - 2$ , for  $k \geq 0$  and  $N > 4$ . Therefore, the  $LF_k(N)$  circuit has  $O(N)$  size and  $O(\lg N)$  depth. Figure 2.13 illustrates the  $LF_k(N)$  circuits, for  $0 \leq k \leq 1$ . The circuit size and depth vary with the value of  $k$ . As the value of  $k$  increases, the circuit size decreases but the circuit depth increases. This algorithm allows us to trade the size for depth and *vice-versa*.

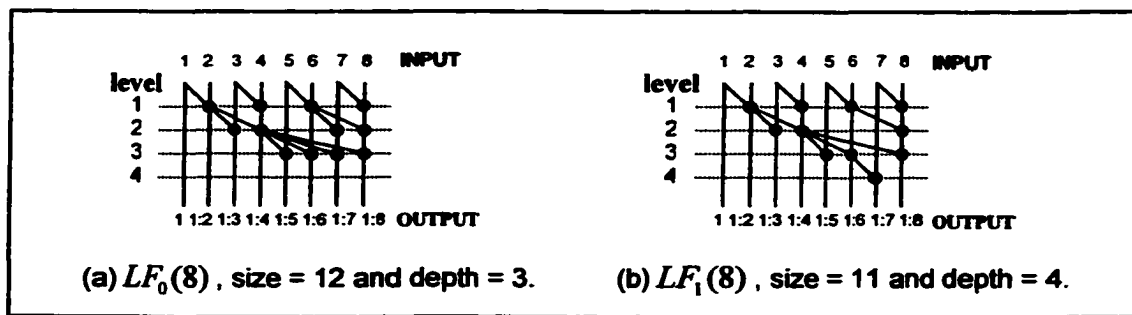


Figure 2.13: Examples of Ladner-Fischer parallel prefix circuits with 8 inputs.

### 2.2.4 The Brent-Kung Parallel Prefix Circuit

The Brent-Kung prefix circuit [BK82],  $BK(N)$ , is another circuit which is based on the divide-and-conquer strategy. This circuit has smaller size than that of the  $LF_k$  ( $k < \lceil \lg N \rceil - 2$ ) circuits, but its depth is greater than that of these circuits. This described circuit can be as follows. Let  $N = 2^n$ . The  $BK(N)$  is divided into three levels -- the first level with  $N/2$  operation nodes, the second level with  $BK(N/2)$ , and the last level with  $(N/2 - 1)$  operation nodes. According to Figure 2.14, we can build  $BK(N)$  from  $BK(N/2)$  recursively. The following recurrences for the size and depth of this circuit are

immediate:

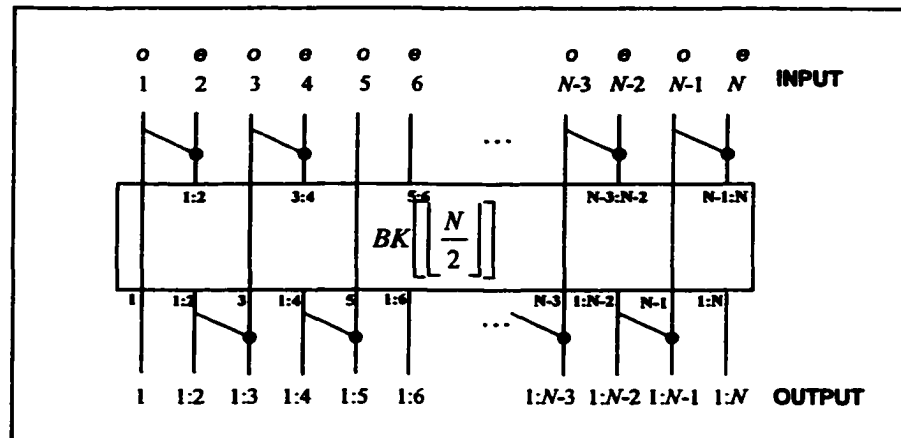
$$size(BK(N)) = size(BK(\frac{N}{2})) + N - 1, \quad \text{with} \quad size(BK(4)) = 4.$$

$$depth(BK(N)) = depth(BK(\frac{N}{2})) + 2, \quad \text{with} \quad depth(BK(4)) = 2.$$

When  $N = 2^n$ , we can solve these recurrences easily, as follows.

$$\begin{aligned} size(BK(N)) &= size(BK(\frac{N}{2})) + N - 1 \\ &= 2N - \lg N - 2 = O(N) \end{aligned}$$

Similarly,

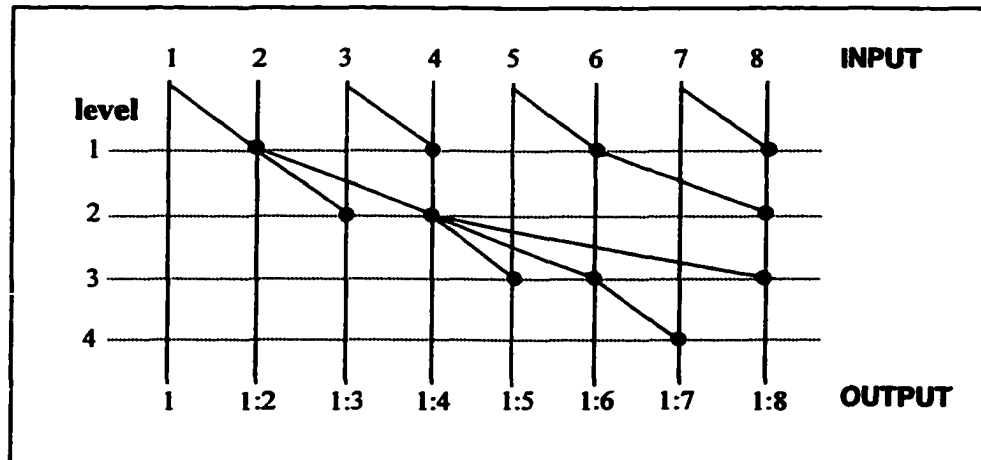


**Figure 2.14:** A Brent-Kung parallel prefix circuit,  $BK(N)$  based on divide-and-conquer strategy (o = odd, e = even), derived from [LD94].

$$\begin{aligned} depth(BK(N)) &= depth(BK(\frac{N}{2})) + 2 \\ &= 2 \lg N - 2 = O(\lg N) \end{aligned}$$

The  $BK(N)$  circuit takes  $O(\lg N)$  time like the  $DC(N)$  circuit. However, the circuit size, which is  $O(N)$ , is smaller than that of the  $DC(N)$  circuit. The circuit is not depth-

optimal, and because  $size(BK(N)) + depth(BK(N)) > 2N - 2$  for  $N > 4$ ,  $BK(N)$  is not (size, depth)-optimal either. Figure 2.15 shows the  $BK(N)$  circuit for  $N = 8$ . The circuit size and depth are 11 and 4, respectively. This is a compromise between serial prefix circuit and the divide-and-conquer algorithms. In this case the number of operations is  $2N - \lg N - 2$  and the depth is  $2 \lg N - 2$ .



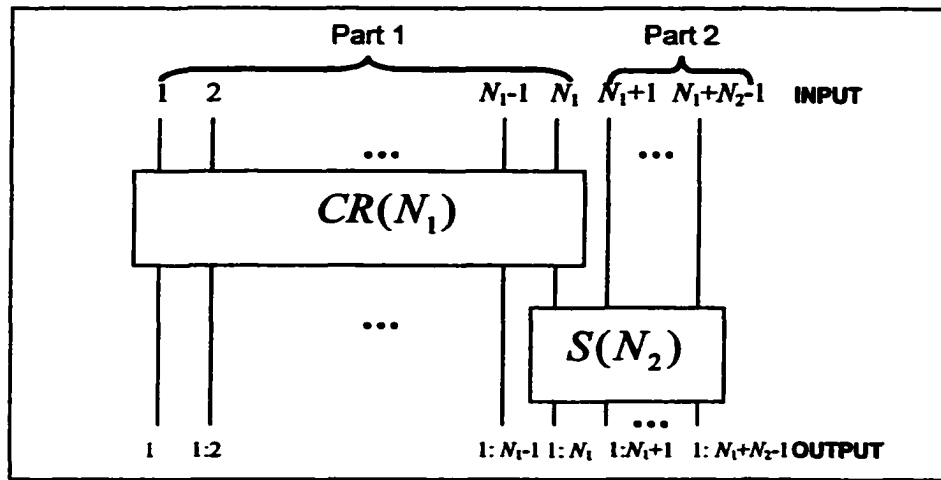
**Figure 2.15:** An illustration of the Brent-Kung parallel prefix circuit,  $BK(8)$ ,  $size = 11$ ,  $depth = 4$ .

### 2.2.5 The Snir Parallel Prefix Circuit

Snir [Sni86] showed that the sum of the circuit depth and circuit size of any prefix circuit with  $N$  inputs is lower bounded by  $2N - 2$  (that is  $depth(N) + size(N) \geq 2N - 2$ ). He also introduced an algorithm to construct the (size, depth)-optimal prefix circuits for any  $N$  with the depth in the range  $\max(\lceil \lg N \rceil, 2\lceil \lg N \rceil - 2) \leq depth(SN(N)) \leq N - 1$ . The deficiency of a prefix circuit is defined as

$$deficiency = size + depth - (2N - 2).$$

A circuit with zero deficiency is said to be (size, depth)-optimal. The Snir prefix circuit,  $SN(N)$ , is the combination of two prefix circuits: the compressed layered prefix circuit,  $CR(N_1)$ , and the serial prefix circuit,  $S(N_2)$ , where  $N = N_1 + N_2 - 1$ . The circuit's layout is shown in Figure 2.16.  $SN(N)$  is constructed by feeding the last output of  $CR(N_1)$  as the first input of  $S(N_2)$ .



**Figure 2.16:** The Snir prefix circuit,  $SN(N) = CR(N_1) \cdot S(N_2)$ .

### Compressed Layered Prefix Circuits [LD94, Sni86]

The compressed layered prefix circuit,  $CR(N)$ , is obtained by compressing the layered parallel prefix circuit. The compression involves moving same nodes to their actual level as determined by the path from the input nodes. The design of the layered parallel prefix circuit [Sni86] is based on the divide-and-conquer strategy. The design specifies the operations level by level as follows. Let  $g_\alpha$  be a set of a pair of inputs such that

$$g_\alpha = \{(i, j) \mid \text{a node at level } \alpha \text{ is fed by lines } i \text{ and } j\}.$$

Now, given  $N$ , let  $m = \lceil \lg N \rceil$ . For each level, let

$$g_t = \left\{ (k2^t - 2^{t-1}, \min(N, k2^t)) \mid k = \left\lfloor \frac{N-1}{2^t} + \frac{1}{2} \right\rfloor, \dots, 2, 1 \right\}$$

be the set of operations at level  $t$  for,  $t = 1, \dots, m$ , and

$$g_{m+t} = \left\{ (k2^{m-t}, k2^{m-t} + 2^{m-t-1}) \mid k = \left\lfloor \frac{N-1}{2^{m-t}} - \frac{1}{2} \right\rfloor, \dots, 2, 1 \right\}$$

be the set of operations at level  $m+t$  for,  $t = 1, \dots, m-1$ .

The depth of the circuit as defined above is  $2\lceil \lg N \rceil - 1$  (i.e.,  $m + m - 1$ ). The first  $\lceil \lg N \rceil$  levels construct a complete binary tree rooted at  $1:2^{\lceil \lg N \rceil}$ . Including the leaves of the binary tree, which are all inputs, the tree depth is  $\lceil \lg N \rceil + 1$ . Therefore, the tree clearly has  $(N-1)$  internal nodes,  $\lceil \lg N \rceil + 1$  of which are output nodes (i.e., nodes labeled  $1:2^x$  for  $x = 0, 1, \dots, \lg N$ ). A prefix circuit with  $N$  inputs must have  $N$  outputs. Therefore, the remainder of  $\lceil \lg N \rceil - 1$  levels contain  $N - \lceil \lg N \rceil - 1$  nodes. Thus, the total size of the circuit is  $2N - \lceil \lg N \rceil - 2$ . Also the last output  $y_N$  is available at depth  $\lceil \lg N \rceil$ . Hence, the circuit is a restricted prefix circuit. In this layered design definition, there are operation nodes at level  $> m$  where inputs do not exactly come from the immediately preceding level. Such nodes are then moved to the appropriate level. After all such nodes are moved to the appropriate level the layered circuit is compressed to yield a circuit with depth as follows:

$$\text{depth}(\text{CR}(N)) = \begin{cases} \lceil \lg N \rceil & \text{if } N \leq 5, \\ 2r - 3 & \text{if } 3 \times 2^{r-2} \leq N < 2^r \text{ for } r \geq 3, \\ 2r - 2 & \text{if } 2^r \leq N \leq 3 \times 2^{r-1} \text{ for } r \geq 3. \end{cases}$$



As an example, let  $N = 8$ . Then  $m = \lceil \lg 8 \rceil = 3$  and we obtain  $g_t$ , where  $t = 1, 2, \dots, 5$ , as follows.

$$g_1 = \{(1,2), (3,4), (5,6), (7,8)\}$$

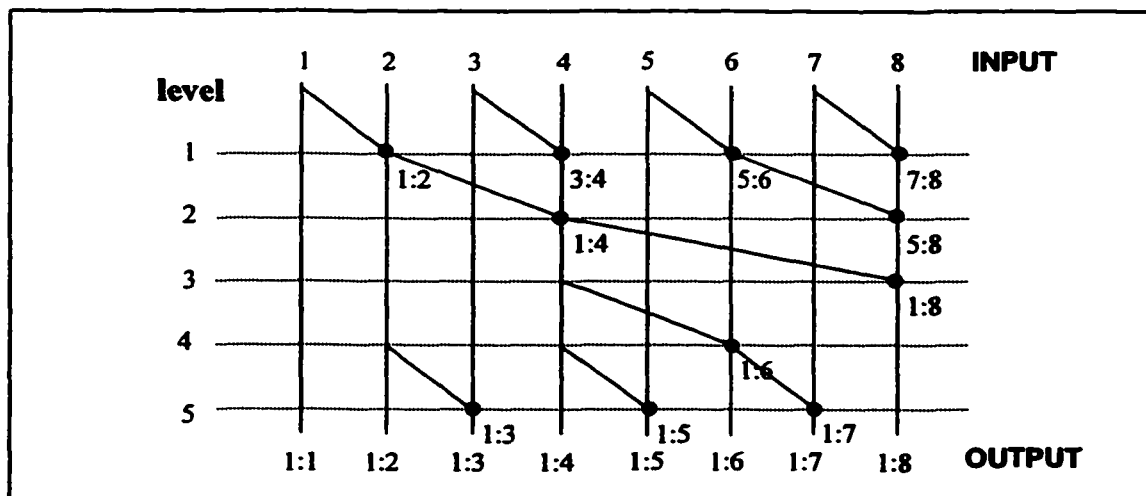
$$g_2 = \{(2,4), (6,8)\}$$

$$g_3 = \{(4,8)\}$$

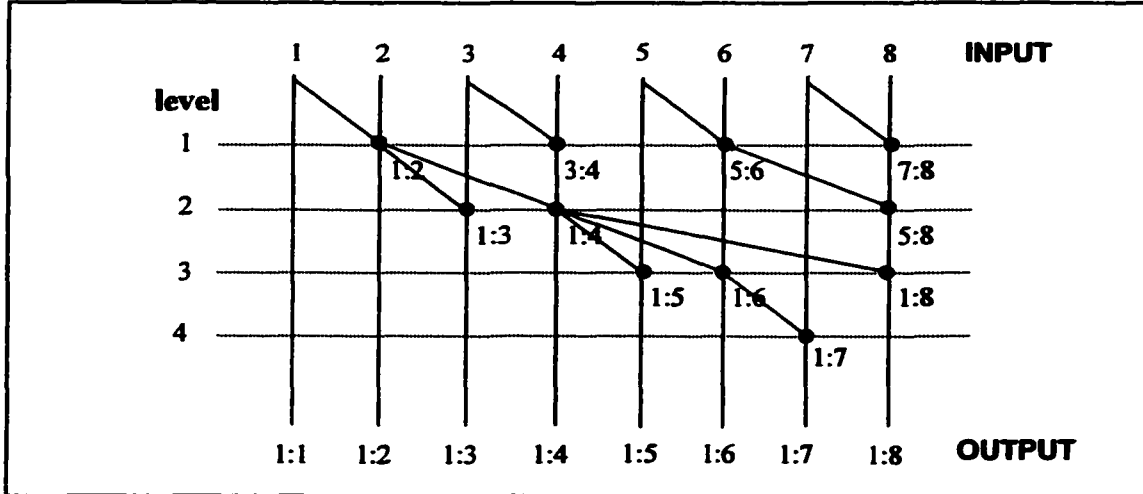
$$g_4 = \{(4,6)\}$$

$$g_5 = \{(2,3), (4,5), (6,7)\}$$

The layout of the layered prefix circuit before compression is shown in Figure 2.17. Its depth is 5 and its size is 11. However, the circuit can be compressed by moving the node labeled 1:6 at level 4 to level 3 and the nodes labeled 1:3, 1:5, and 1:7 at level 5 to level 2, 3, and 4, respectively. As shown in Figure 2.18, the depth of the compressed circuit is reduced by one level. Thus, the new circuit depth is 4. The compressed circuit is not a (size, depth)-optimal circuit since the circuit's deficiency is greater than zero as follows.



**Figure 2.17:** An illustration of the layered parallel prefix circuit,  $size = 11$ ,  $depth = 5$ .



**Figure 2.18:** An illustration of the compressed layered prefix circuit,  $size = 11$ ,  $depth = 4$ .

$$\begin{aligned}
 \text{deficiency} &= \text{size}(CR(N)) + \text{depth}(CR(N)) - (2N - 2) \\
 &\geq (2N - 2 - \lceil \lg N \rceil) + (2\lceil \lg N \rceil - 3) - (2N - 2) \\
 &\geq \lceil \lg N \rceil - 3 \\
 &\geq 0
 \end{aligned}$$

As in the previous discussion, the Snir's circuit,  $SN(N)$ , is composed of two prefix circuits: the compressed layered prefix circuit and the serial prefix circuit. Therefore, the circuit size and depth are defined as

$$\text{size}(SN(N)) = \text{size}(CR(N_1)) + \text{size}(S(N_2))$$

$$\text{depth}(SN(N)) = \max\{\text{depth}(CR(N_1)), \lceil \lg N_1 \rceil + \text{depth}(S(N_2))\}$$

Although the  $SN(N)$  circuit is a combination of a (size-depth)-non-optimal prefix circuit (that is  $CR(N_1)$ ) and the (size, depth)-optimal prefix circuit (that is  $S(N_2)$ ), it is a (size, depth)-optimal prefix circuit [Sni86]. If the given input value  $N$  satisfy the inequality

$$2\lceil \lg N \rceil - 2 \leq \text{depth}(SN(N)) < 2\lg(N-1) - 1,$$

then the design recursively defines (size, depth)-optimal circuit with depth  $\text{depth}(SN(N))$ . Otherwise, a circuit with

$$N - 2 \geq \text{depth}(SN(N)) \geq 2\lg(N-1) - 1$$

is given.

As an example of the  $SN(N)$  circuit, let  $N = 19$ . Then  $r = 4$ ,  $N_2 = r + 1 = 5$  and  $N_1 = N - N_2 + 1 = 15$ . The  $SN(19)$  circuit is given in Figure 2.19, which is composed of  $CR(15)$  and  $S(5)$ . Clearly, the circuit depth is 8, the circuit size is 28 and their sum is 36, which is equal to  $(2 \times 19 - 2)$ . Hence,  $SN(19)$  is (size, depth)-optimal. However, Snir parallel prefix circuit is not depth-optimal, and also not a restricted prefix circuit.

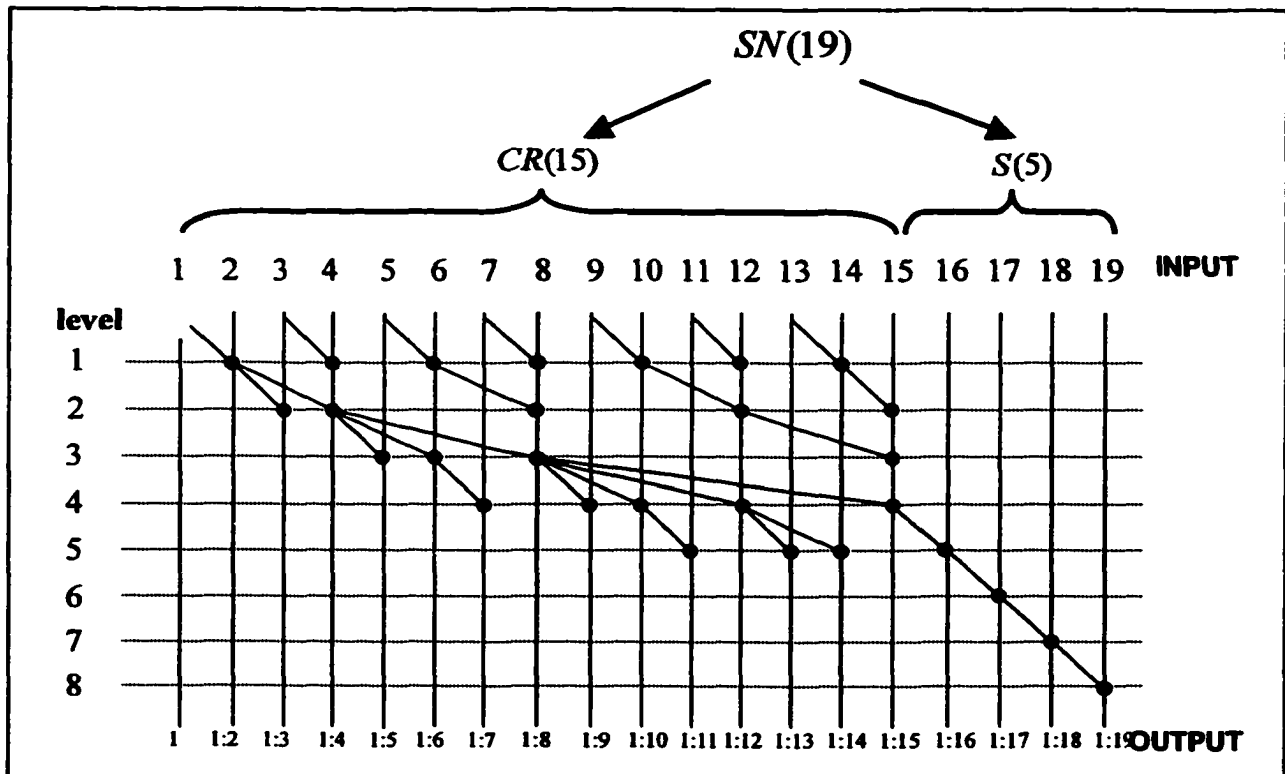


Figure 2.19: The Snir prefix circuit,  $SN(19)$ , size = 28 and depth = 8.

## 2.2.6 The LYD Parallel Prefix Circuit

Lakshmivarahan, Yang, and Dhall [LYD87] were the first to introduce the algorithm to design a (size, depth)-optimal prefix circuit, having the smallest depth among all other circuits, for  $N = 9$  to  $12$ ,  $N = 17$  to  $20$ , and  $N = 33$ . Their discovery proves that there is (size, depth)-optimal prefix circuit with depth in the range  $\lceil \lg N \rceil \leq d(N) \leq \max(\lceil \lg N \rceil, 2\lceil \lg N \rceil - 3)$ . Moreover, their algorithm gives the depth-optimal prefix circuits for some inputs. The algorithm distributes all  $N$  inputs in to four parts properly. Like the  $SN(N)$  prefix circuit, Part 1 corresponds to the compressed layered prefix circuit. Part 2 is a new optimal prefix circuit,  $Q(N)$ , proposed by the group [LYD87, LD94]. Part 3 and Part 4 are the serial prefix circuits.

### New optimal prefix circuit, $Q(N)$

$Q(N)$  is a new class of (size, depth)-optimal prefix circuits with condition

$$N = \frac{t(t+1)}{2} + 1, \quad \text{for } t > 0.$$

Let  $g_{i,j}$  denote the  $j^{\text{th}}$  node at level  $i$  and be represented with an ordered pair  $(a,b)$ ; where  $a = \text{left}(g_{i,j})$  and  $b = \text{right}(g_{i,j})$ , refer to the left and right inputs of the node  $g_{i,j}$ , respectively.

The  $Q(N)$  circuit is constructed as follows.

1. At level 1,  $g_{1,1} = (1,2)$ ,  $g_{1,2} = (3,4)$ , and

$$g_{1,j} = (\text{left}(g_{1,j-1}) + (j-1), \text{right}(g_{1,j-1}) + (j-1)), \text{ for } j = 3, 4, \dots, t.$$

2. For levels  $i = 2$  to  $t$ ,

$$g_{i,1} = (\text{right}(g_{i-1,1}), \text{right}(g_{i-1,2})), \text{ and}$$

$$g_{i,j} = (\text{right}(g_{i-1,j+1}), \text{right}(g_{i-1,j+1}) + 1), \text{ for } j = 2, 3, \dots, t + 1 - i.$$

3. The nodes at level  $(t + 1)$  are given by

$$g_{t+1} = \{(\text{right}(g_{i,1}), \text{right}(g_{i,1}) + j) \mid i = 1, 2, \dots, t - 1, j = 1, 2, \dots, i\}$$

The  $Q(N)$  circuit has unique properties: The circuit depth is equal to the circuit width and the circuit size is equal to the square of the circuit depth.

Let  $N = 7$ . Thus  $t = 3$ . We obtain  $g_{i,j}$  as follows.

$$g_{1,1} = (1,2) \qquad g_{1,2} = (3,4) \qquad g_{1,3} = (5,6)$$

$$g_{2,1} = (2,4) \qquad g_{2,2} = (6,7) \qquad g_{3,1} = (4,7)$$

$$g_4 = (2,3), (4,5), (4,6)$$

The  $Q(7)$  circuit is illustrated in Figure 2.20. As seen, the  $Q(N)$  circuit consists of blocks of the serial prefix circuits with block sizes increasing in an arithmetic sequence.

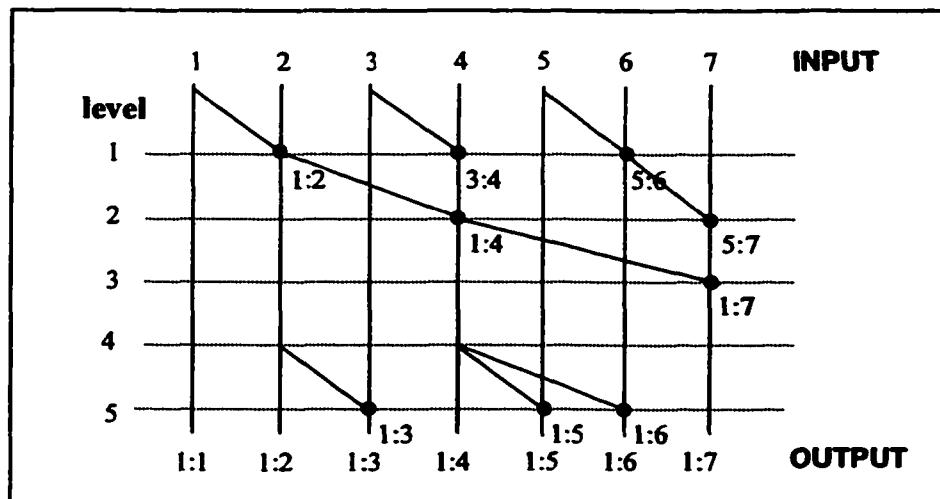


Figure 2.20: The  $Q(7)$  prefix circuit.

The *LYD* prefix circuit,  $LYD(N)$ , is composed of 4 parts (see Figure 2.21) as follows.

*Part 1*: the compressed layered prefix circuit,  $CR(N_1)$ ,

$$depth(Part1) \leq depth(N)$$

$$size(Part1) = 2N_1 - \lceil \lg N_1 \rceil - 2$$

The last output,  $1 : N_1$ , is available at level  $t = \lceil \lg N_1 \rceil \leq depth(N) - 2$ .

*Part 2*: the new (size, depth)-optimal prefix circuit,  $Q(N_2)$ ,

$$N_2 = \frac{\lceil \lg N_1 \rceil (\lceil \lg N_1 \rceil + 1)}{2} + 1$$

$$depth(Part2) = \lceil \lg N_1 \rceil + 2 \leq depth(N)$$

The size after combining with Part 1 is  $size(Part2) = 2N_2 - 1$ . The last output,  $1 : N_1 + N_2$ ,

is available at level  $\lceil \lg N_1 \rceil + 1$ .

*Part 3*: the serial prefix circuit,  $S(N_3)$ ,

$$depth(Part3) = \lceil \lg N_1 \rceil + 1 + N_3 \leq depth(N)$$

$$size(Part3) = N_3$$

*Part 4*: the serial prefix circuit,  $S(N_4)$ ,

$$depth(Part4) = \lceil \lg N_1 \rceil + 2 + N_3 = depth(N)$$

$$size(Part4) = 2N_4 - 1$$

where  $N = N_1 + N_2 + N_3 + N_4$ ;  $N_1, N_2$ , and  $N_4 \geq 1$ ; and  $N_3 \geq 0$ .

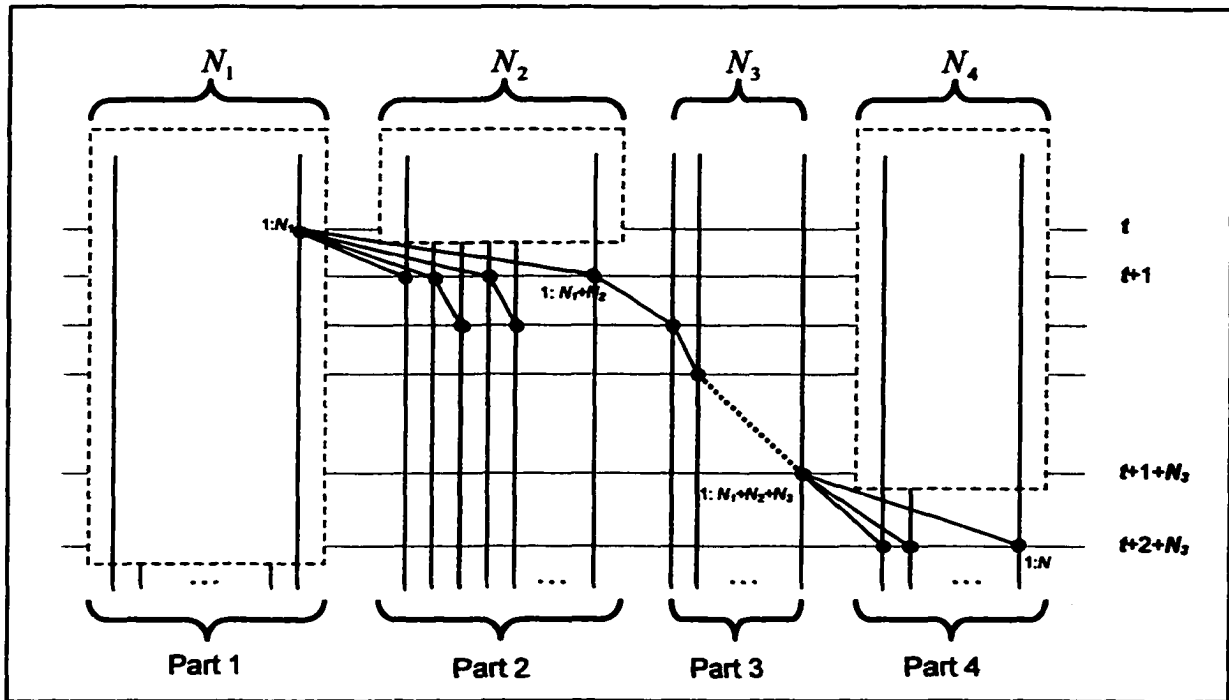


Figure 2.21: The structure of  $LYD(N)$ , derived from [LD94].

Thus, the circuit depth is  $\lceil \lg N_1 \rceil + 2 + N_3$  and the circuit size is  $size(Part1) + size(Part2) + size(Part3) + size(Part4)$ , which is  $(2N - 2) - depth(N)$ . It is easy to see that the circuit  $LYD(N)$  is (size, depth)-optimal. For any integer  $N$ , there exists a (size, depth)-optimal prefix circuit,  $LYD(N)$ , such that  $2\lceil \lg N \rceil - 6 \leq depth(LYD(N)) \leq 2\lceil \lg N \rceil - 3$ . However, the circuit is not restricted prefix circuit and not size optimal. But for many  $N$ 's, the circuit yields the optimal depth. As an example, Figure 2.22 shows the circuit  $LYD(19)$ , which is a combination of  $CR(8) \cdot Q(7) \cdot S(0) \cdot S(4)$ .

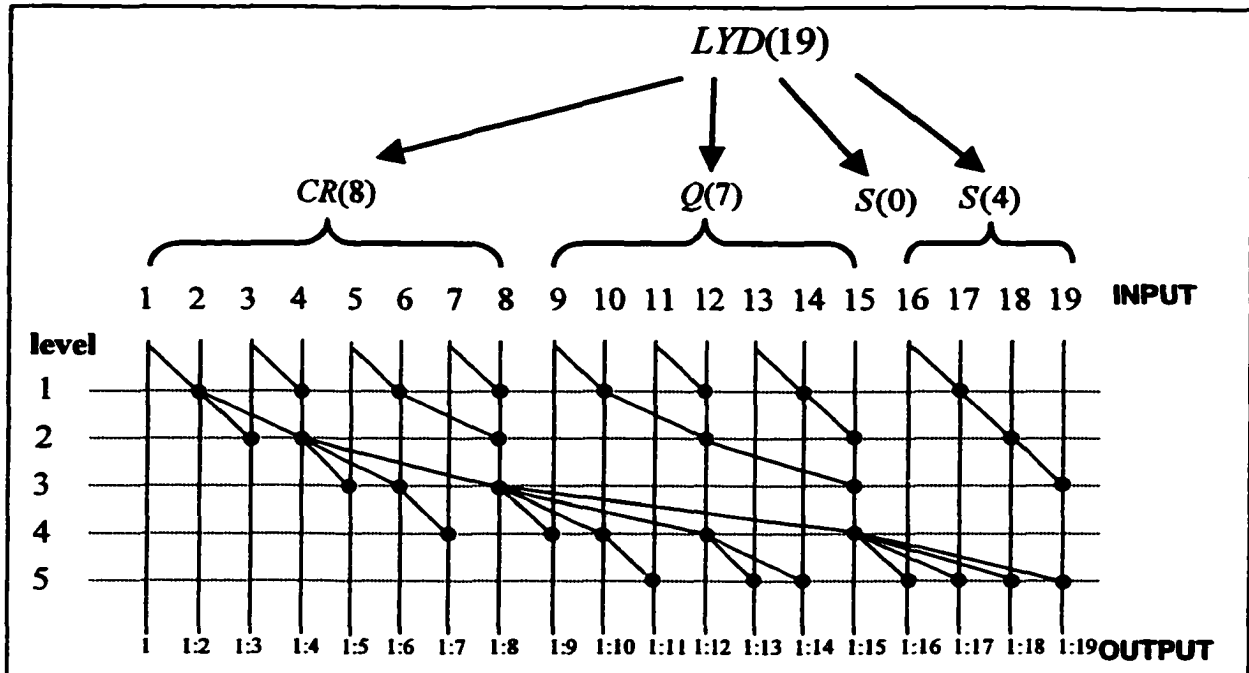


Figure 2.22: The  $LYD(19)$  prefix circuit with size 31 and depth 5.

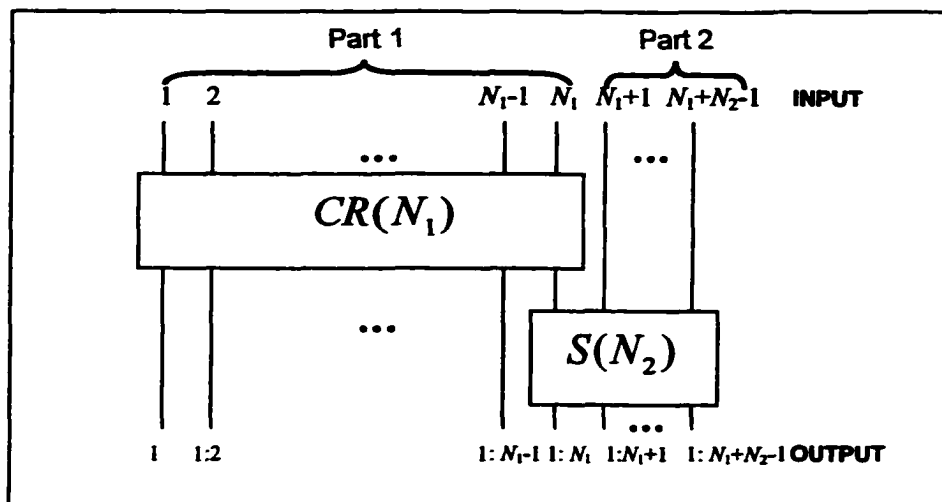
### 2.2.7 The Shih-Lin Parallel Prefix Circuit

Recently, Lin and Shih [LS99] have proposed a new (size, depth)-optimal prefix circuit,  $SL(N)$ , with the depth in the range

$$2\lceil \lg N \rceil - 5 \leq \text{depth}(SL(N)) \leq N - 1, \quad \text{for } N \geq 12.$$

The structure of the  $SL(N)$  circuit is similar to the  $SN(N)$  circuit but differs in the partitions of the circuit. The  $SL(N)$  circuit is also composed of two parts: the compressed layered prefix circuit and the serial prefix circuit as shown in Figure 2.23. In other words,  $SL(N) = CR(N_1) \cdot S(N_2)$ , where  $N = N_1 + N_2 - 1$ . This algorithm offers the same or equivalent performance as that of  $LYD$  but it is easier to implement.





**Figure 2.23:** The  $SL(N)$  circuit,  $SL(N) = CR(N_1) \cdot S(N_2)$ .

Let  $depth(SL(N))$  be the depth of the  $SL(N)$  circuit, defined above. Then

$$depth(SL(N)) = \begin{cases} 2\lceil \lg N \rceil - 5 & \text{if } 2^{r-1} < N < 2^{r-1} + r - 4 \quad \text{for } r \geq 6, \\ 2\lceil \lg N \rceil - 4 & \text{if } 2^{r-1} + r - 4 \leq N < 3 \times 2^{r-2} \quad \text{for } r \geq 5, \\ 2\lceil \lg N \rceil - 3 & \text{if } 3 \times 2^{r-2} \leq N \leq 2^r \quad \text{for } r \geq 4. \end{cases}$$

The following are the conditions to choose  $N_2$  [LS99].

If  $r \geq 4$  and  $3 \times 2^{r-2} \leq N \leq 2^r$ , then  $N_2 = r - 2$ .

If  $r \geq 6$  and  $2^{r-1} < N < 2^{r-1} + r - 4$ , then  $N_2 = r - 3$ .

If  $r \geq 5$  and  $N = 2^{r-1} + r - 4$ , then  $N_2 = r - 2$ .

If  $r \geq 5$  and  $2^{r-1} + r - 4 < N < 3 \times 2^{r-2}$ , then  $N_2 = r - 3$ .

Since  $depth(SL(N)) + size(SL(N)) = 2N - 2$ , the  $SL(N)$  circuit is a (size, depth)-optimal prefix circuit [LS99]. Like the Snir prefix circuit, the  $SL(N)$  circuit is neither depth-optimal nor restricted prefix circuit. As an example of  $SL(N)$ , let  $N = 19$ . Then  $r = 5$  and  $2^{r-1} + r - 4 < N \leq 3 \times 2^{r-2}$ . The layout of the  $SL(19)$  circuit is given in Figure 2.24, which

is composed of  $CR(18)$  and  $S(2)$ . Clearly, the circuit depth is 6, the circuit size is 30, and  $size(SL(19)) + depth(SL(19)) = 2N - 2 = 36$ . Comparing the  $SL(19)$  circuit with the  $LYD(19)$  circuit, the  $SL(19)$  circuit's depth is longer while its size is smaller.

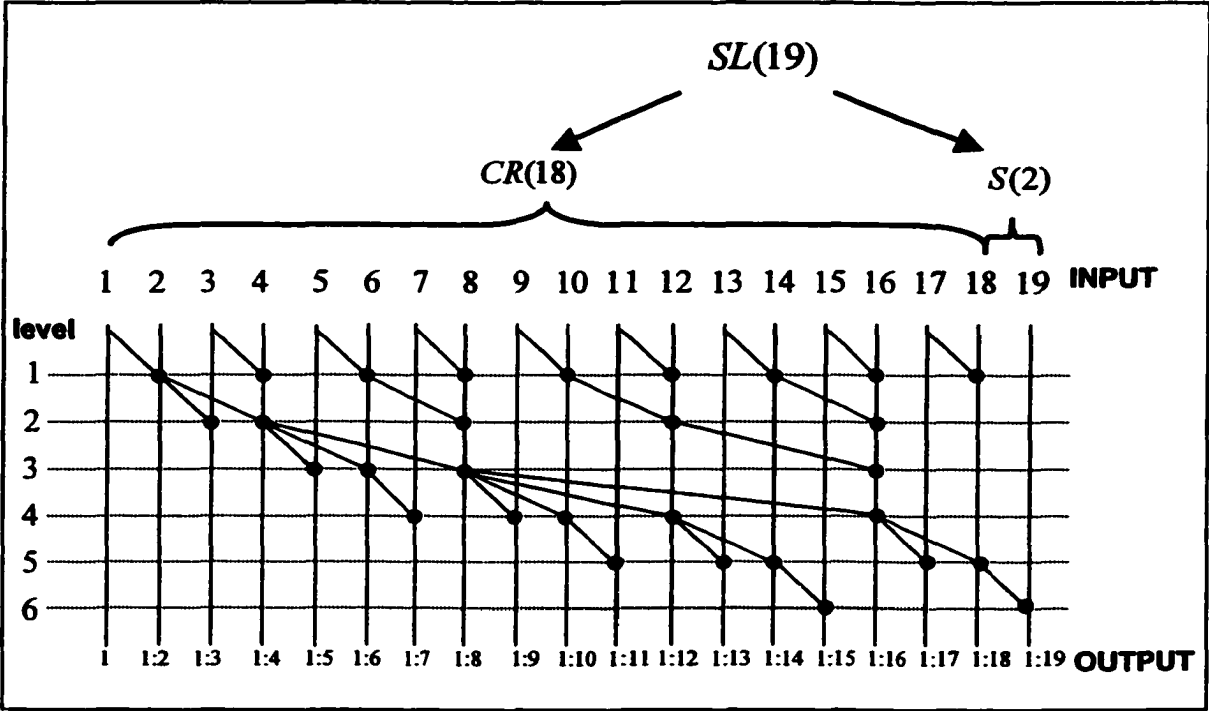


Figure 2.24: The  $SL(19)$  prefix circuit, size = 30 and depth = 6.

### 2.3 Comparison

Table 2.1 provides a comparison of the prefix circuits illustrated in this chapter. While the parallel prefix circuits have desirable depths, which are  $O(\lg N)$ , they differ widely in the number of operations performed. Only four prefix circuits (i.e., serial, Snir, Shih-Lin, and LYD prefix circuits) are (size, depth)-optimal. The divide-and-conquer prefix circuit and the  $LF_0$  prefix circuit have the shortest depth and the serial prefix circuit has the smallest size.

The size-depth trade-off does apply to any prefix circuit. For example, the serial prefix circuit performs fewest operations (i.e., smallest size) compared to the others, but has the longest depth while the divide-and-conquer prefix circuit has the largest size, but has the smallest depth. Although the Shih-Lin prefix circuit and the Snir prefix circuit have similar circuit layouts, the Shih-Lin prefix circuit has a smaller depth than the Snir prefix circuit. All circuits have unbounded fan-out except the serial prefix circuit that has a constant fan-out of two. The divide-and-conquer prefix circuit and the  $LF_0$  prefix circuit have the largest fan-out  $((N/2)+1)$ . The Brent-Kung, Shih-Lin and Snir prefix circuits have the same fan-out  $(\lceil \lg N \rceil + 1)$ , which is smaller than that of the LYD prefix circuit  $(2\lceil \lg N \rceil - 2)$ .

**Table 2.1:** A Comparison of the seven prefix circuits illustrated in this chapter.

| Prefix Circuit   | Size   | Depth   | Fan-out           | (size, depth)-optimal              |
|--|--|---|-------------------|------------------------------------|
| Serial   | $N - 1$                                      | $N - 1$   | 2                 | Yes<br><small>size-optimal</small> |
| Divide-and-Conquer   | $(N/2)\lg N$                                 | $\lg N$   | $(N/2)+1$         | No<br><small>depth-optimal</small> |
| $LF_0$   | $4N - F(5 + \lg N) + 1$                      | $\lg N + k$   | $(N/2^{k+1}) + k$ | No                                 |
| $LF_k$<br><small>when <math>0 &lt; k &lt; \lg N - 2</math></small> | $2N(1 + (1/2^k)) - F(5 + \lg N - k) - k + 1$ |   |                   |                                    |
| $LF_k$<br><small>when <math>k \geq \lg N - 2</math></small>        | $2N - \lg N - 2$                             | $2\lg N - 2$  | $\lg N + 1$       |                                    |
| Brent-Kung   | $2N - \lg N - 2$                             | $2\lg N - 2$  | $\lg N + 1$       | No                                 |
| Snir   | $2N - 2 - \text{depth}$                      | $\max(\lg N, 2\lg N - 2)$<br>$\leq \text{depth} \leq N - 1$ | $\lg N + 1$       | Yes                                |
| LYD  | $2N - 2 - \text{depth}$                      | $2\lg N - 6 \leq \text{depth} \leq 2\lg N - 3$              | $2\lg N - 2$      | Yes                                |
| Shih-Lin   | $2N - 2 - \text{depth}$                      | $2\lg N - 5 \leq \text{depth} \leq 2\lg N - 3$              | $\lg N + 1$       | Yes                                |

## **CHAPTER 3**

### **SOURCES OF POWER CONSUMPTION**

In the previous chapter we examined size and depth trade-offs of various prefix circuit designs. We want to examine the power consumption characteristics of these circuits. In this chapter, the sources of power consumption in circuits are reviewed and the strategies to estimate the power consumption of the various prefix circuits are presented. This should help us to better understand the power consumption characteristics of the circuits. We also introduce the circuit simulation tool called PSpice in brief.

#### **3.1 CMOS**

Presently, CMOS (*Complementary-symmetry Metal-Oxide Semiconductor*) technology is the most popular technology used by the digital IC (Integrated Circuit) industry because of its low power consumption, its good scalability and its speed [CB95, RCN01, WE93]. CMOS technology uses two types of transistors: a *P*-type transistor and an *N*-type transistor realizing logic functions. Figure 3.1 shows the *P*-type and *N*-type transistors, and their characteristics. The *P*-type transistor has a bubble on its symbol indicating that the transistor is conducting when its input is 0. The *N*-type transistor is conducting when its input is 1. The input has been labeled with the signal *s*.

#### **Examples of CMOS Logic**

The CMOS inverter is the heart of all digital designs. Each complex design (for example, NAND gate) can be clearly explained if the inverter's characteristics are understood. It consists of two transistors, one *P*-type and one *N*-type transistor. Figure 3.2 shows the CMOS inverter and its truth table.

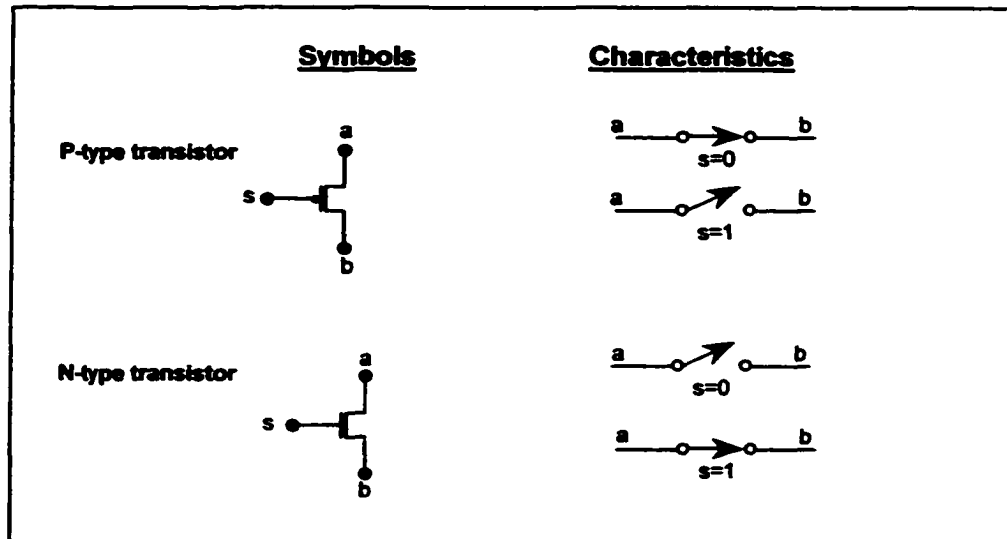


Figure 3.1: *P*-type and *N*-type transistor and their characteristics.

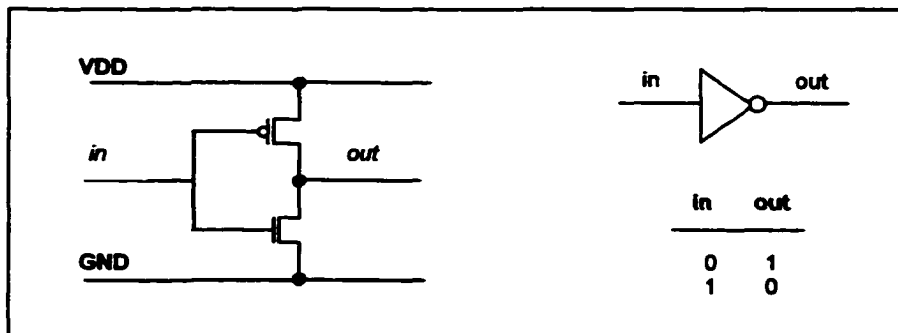


Figure 3.2: CMOS inverter.

The CMOS NAND gate, CMOS NOR gate and their truth tables are illustrated in Figures 3.3 and 3.4, respectively. Both gates consist of four transistors, two *P*-type and two *N*-type transistors.

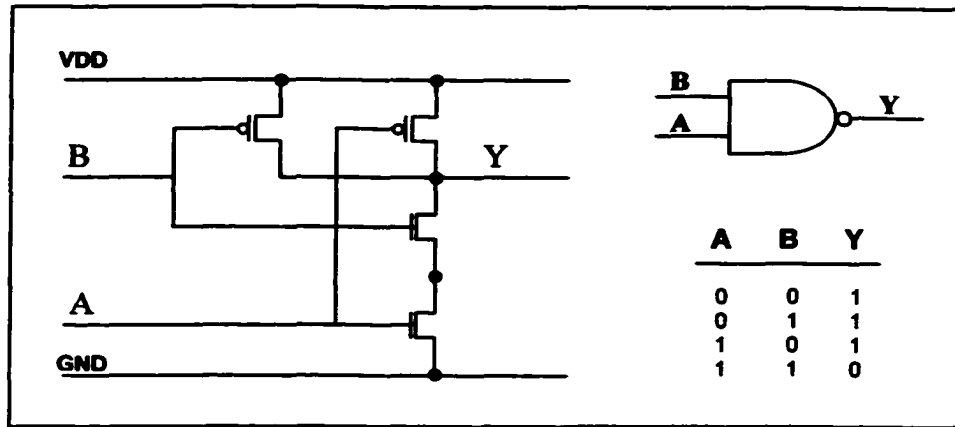


Figure 3.3: CMOS NAND gate.

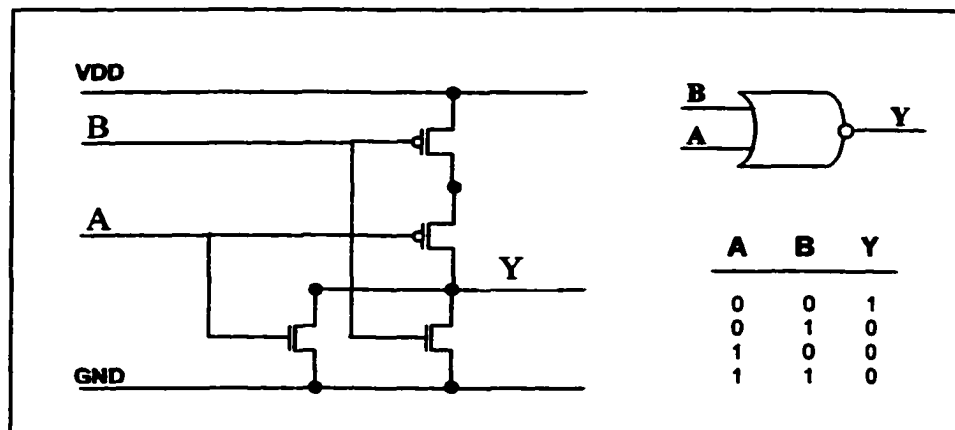


Figure 3.4: CMOS NOR gate.

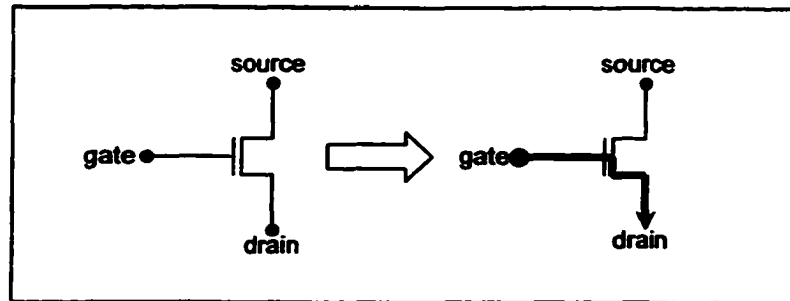
## 3.2 Power Consumption

### 3.2.1 Sources of Power Consumption

In CMOS circuits, power consumption is due to the following three types of current flow [WE93]:

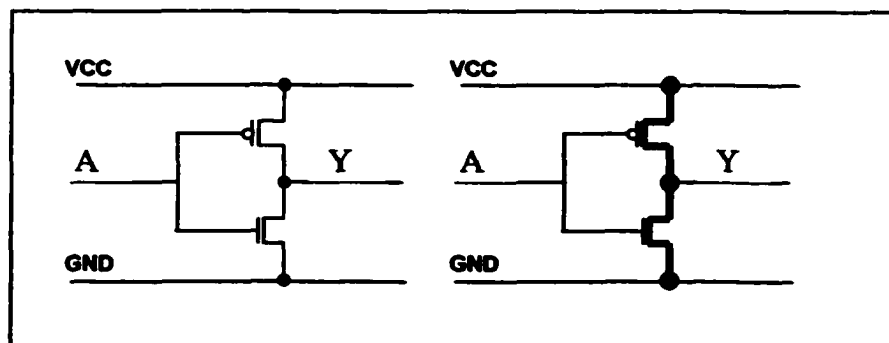
1. Static power consumption due to leakage currents. The static power consumption occurs when some current leaks through to other parts of the transistor (i.e., the

leakage current from the gate to the drain as shown in Figure 3.5), resulting in power loss. The power loss due to leakage current in CMOS is usually insignificant compared to the dynamic power consumption [CB95, RCN01].



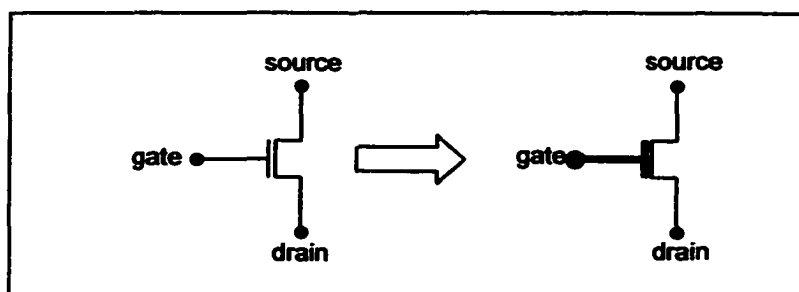
**Figure 3.5:** The leakage current from the gate to the drain of a transistor.

2. Dynamic power consumption due to short-circuit currents. The short-circuit occurs when both *P*-type and *N*-type transistors are momentarily *on* at the same time (see Figure 3.6). Although there is some dynamic power consumption from the short circuit, this power loss is usually insignificant compared to the power dissipated from the switching [CB95, RCN01].



**Figure 3.6:** An illustration of short-circuit when both *P*-type and *N*-type transistor being in the on state at the same time.

3. Dynamic power consumption due to switching currents from repetitively charging and discharging the parasitic capacitances at the transistor's gate (see Figure 3.7). The currents must flow through the transistor's gate to reach the capacitances (i.e., charging the capacitance). During the switching transient, the power is dissipated (i.e., discharging the capacitance). The charging and discharging of the parasitic capacitances are the dominant form of power consumption in CMOS circuits [WE93].



**Figure 3.7:** An illustration of capacitance charging.

Therefore, two components establish the amount of power consumption in a CMOS circuit. They are static and dynamic. Static power consumption is due to imperfect transistors while dynamic power consumption is due to the process of switching transistors on and off. However, in properly designed CMOS circuits, the major portion of the power consumption is from dynamic switching. As a result, in this study, we focus on the dynamic component due to the repetitive charging and discharging of the capacitive loads.

The average power consumption in a CMOS gate or module (e.g., an adder) due to switching can be written as [CB95, WE93]:

$$P_{switching} = C_{eff} V_{DD}^2 f, \quad (3.1)$$



where  $C_{eff}$  is the effective capacitance switched,  $V_{DD}$  is the supply voltage, and  $f$  is the clock frequency.  $C_{eff}$  has two components, the switching activity (signal transition activity) per clock cycle,  $p_f$ , and the load capacitance,  $C_L$ . Thus, for a given circuit running at a given speed (i.e.,  $C_L$  and  $f$  constant), power consumption is a function of the supply voltage and switching activity. Therefore, power reduction can be achieved by either operating the circuit at a lower voltage or by choosing an architecture that reduces the switching activity of the circuit's signals.

### Effect of Voltage Scaling

Due to the quadratic relationship between the supply voltage and the power consumption, lowering supply voltage can be an effective way to achieve dramatic power savings. However, as the supply voltage is decreased, the circuit delay generally increases relatively independent of the logic function and style; see Figure 3.8. Thus, reducing

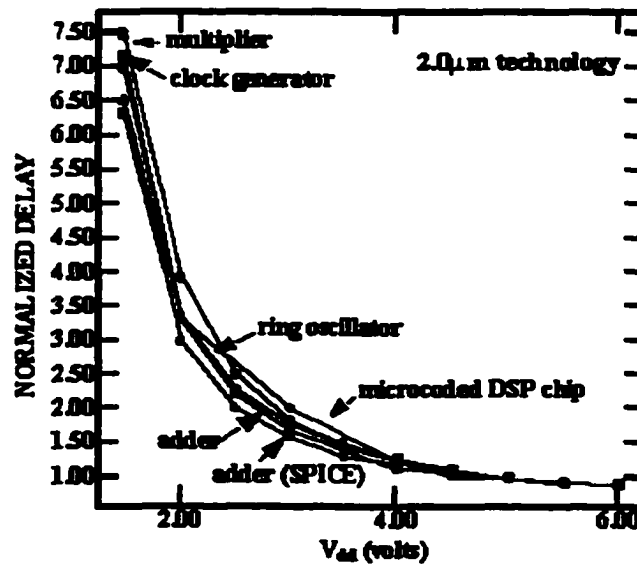


Figure 3.8: Plots of normalized delay vs. supply voltage for a variety of different logic circuits, derived from [CB95].

supply voltage unfortunately reduces the system throughput. This loss in throughput can be recovered in some cases by applying architectural techniques to compensate for the additional delay (e.g., utilization of parallelism and pipeline). Reference [CB95] shows that by changing circuit architecture (i.e., using parallelism and pipelining) it is possible to gain significant speed improvements with only a slight increase in power, hence enabling some voltage down-scaling while maintaining the throughput.

### **Effect of Switching Activity**

The power in CMOS circuits is dissipated when the signals in the circuit switch (i.e., change values). As a result, the amount of switching activity is an indicator of the power consumption. The manner in which the nodes in a circuit are interconnected can have a strong influence on the overall switching activity [CB95]. Some architectures induce extra transition activity at the operation nodes called glitching transitions or dynamic hazards, which consume extra power. Glitching is a major problem that increases the effective switching activity, causing a circuit node to undergo several rapid transitions in a single clock cycle [CB95, RCN01].

Figure 3.9 illustrates an example of the glitching behavior for a chain of eight NAND gates [RCN01] by using a PSpice<sup>®</sup> simulation [Cad00]. In the simulation, all bits of the first input were set to logic 'one' and all bits of second input transition from logic 'zero' to 'one'. For an ideal circuit without propagation delays, the resultant outputs VOUT2, 4, 6 and 8 would stay logic 'one' all the time. However, due to the presence of delays, these outputs switch to low temporarily. This glitching causes extra power to be consumed. Outputs VOUT1, 3, 5 and 7 do not glitch; they just have some propagation delay. It is

noted that the degree of glitching depends on the switching pattern of the input signals [RCN01].

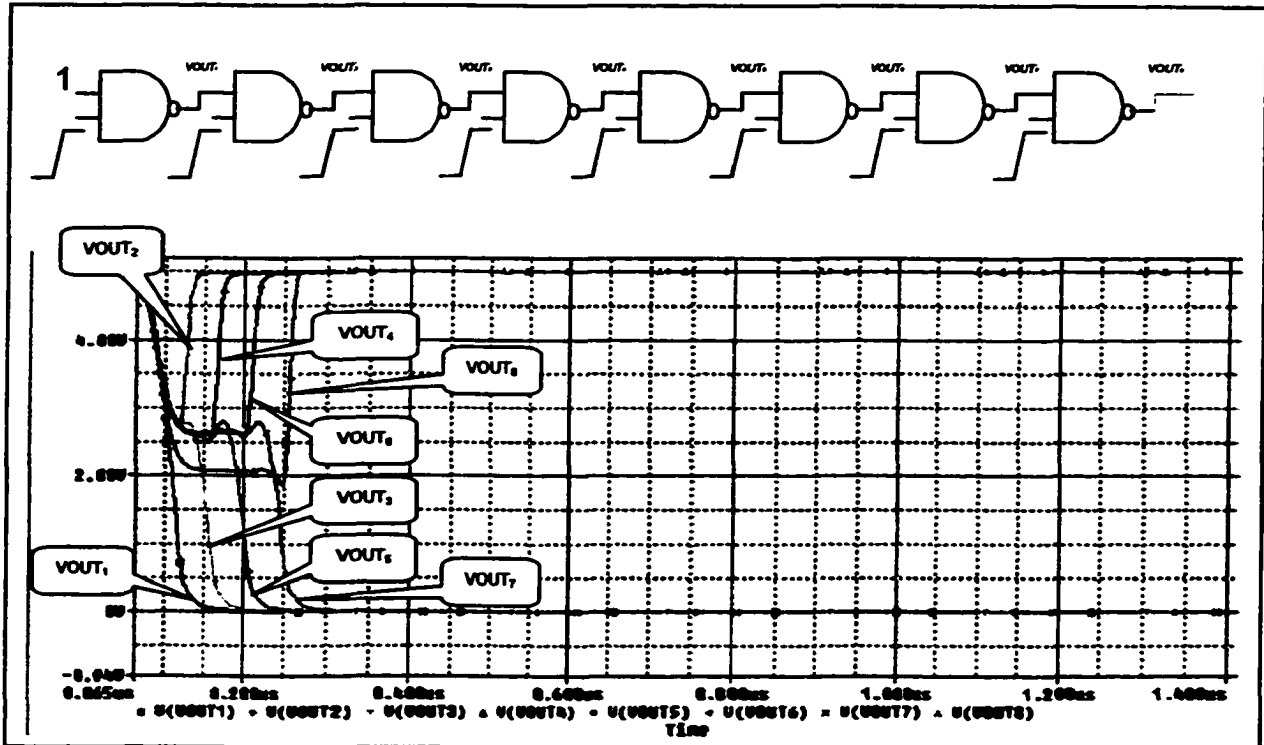


Figure 3.9: An illustration of the glitching behavior of a chain of eight NAND gates [RCN01].

To reduce glitching activity, the depth of the signal paths in the circuit should be balanced. The following is an illustration of two different circuit architectures of a 4-input adder. In Figure 3.10(a), assume that all primary inputs (A, B, C, and D) arrive at the time  $t_0$  and the implementation is non-pipelined. While the first adder makes one transition by computing  $A+B$ , the second adder also makes one transition based on C and the previous (i.e., initial) value of  $A+B$ . After the correct value of  $A+B$  has propagated through the first adder at time say  $t_0 + t_p$ , the second adder re-evaluates  $(A+B)+C$ , which is complete at time  $t_0 + 2t_p$ . Thus, there is a second transition at the second adder.

Similarly, there will be three transitions at the third adder. With a path-balancing approach of Figure 3.10 (b), while the first and second adders make one transition the third adder will make only two transitions to produce the same output as in Figure 3.10 (a). In [CB95], the “total switched capacitance” of the circuit layout in Figures 3.10(a) and 3.10(b) has been simulated by using a switch-level simulator over random input patterns. The results show that the switched capacitance of the circuit layout in Figure 3.10(a) is larger than that of the circuit layout in Figure 3.10(b) by a factor of 1.5 for a four input addition, and 2.5 for an eight input addition. Hence, increasing circuit depth generally increases the total switched capacitance due to glitching and thus increases power consumption [CB95]. As a consequence, the amount of transition activity (switching activity) for a layered and non-pipelined circuit can be a function of depth  $d$  and the number of nodes at each level  $i$ ,  $w_i$ , as [CB95]

$$\sum_{i=1}^d iw_i . \tag{3.2}$$

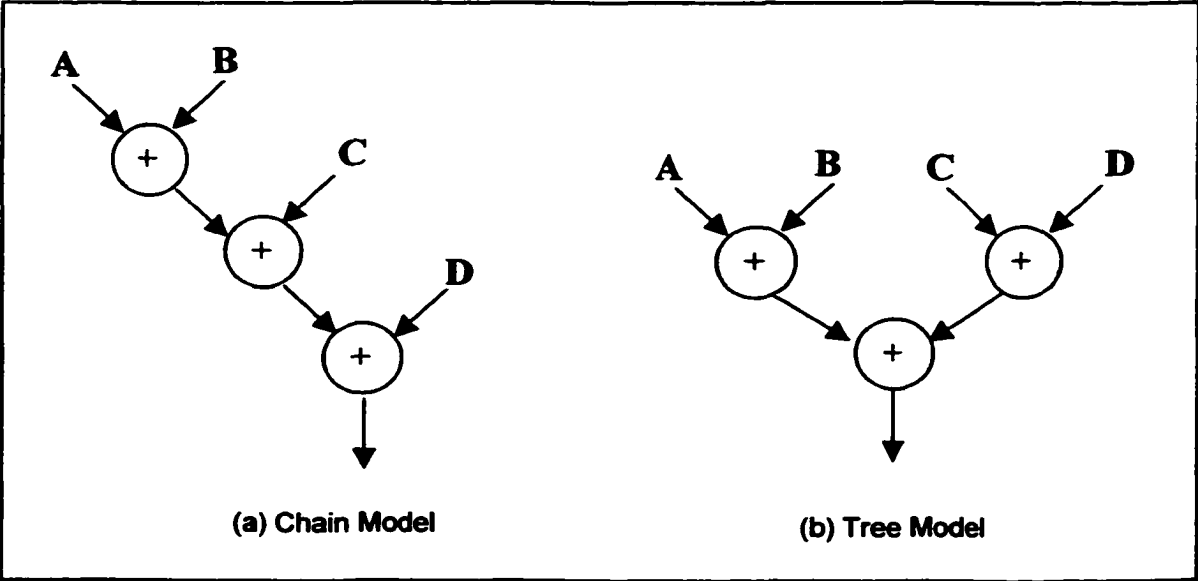


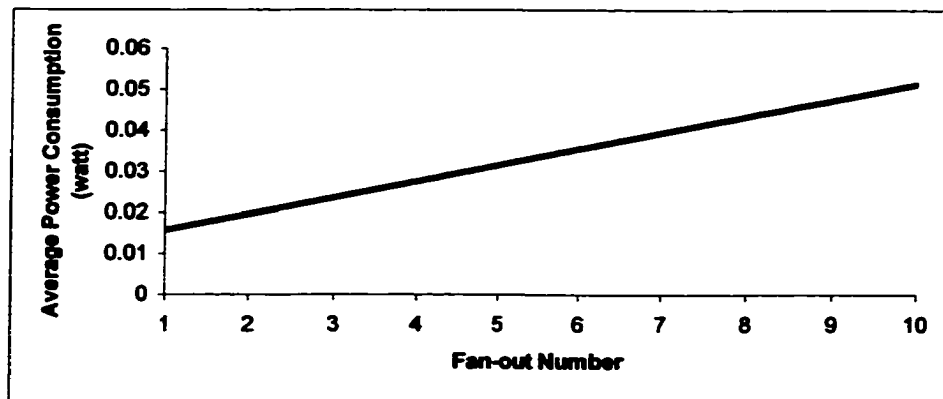
Figure 3.10: An illustration of extra transition activity, derived from [CB95].

From this, it follows that in the worst case estimate for the switching activity of such a circuit can grow according to  $O(d^2)$ , assuming a constant number of nodes at each level.

From the previous discussion and the example of Figure 3.10, we have seen that different circuit architectures for performing the same function can consume different amounts of power. Therefore, the implementation of the various prefix circuits in an application will have different power consumption as well. However, in the prefix circuits, we cannot say with certainty that the circuit with the longer depth will consume more power than one with shorter depth. The reason is that both depth and the number of operation nodes among the candidate prefix circuits differ. In prefix circuits, when the depth decreases, the number of operation nodes (i.e., size) generally increases and vice versa. This is known as the size-depth trade-off [LF80, LD94]. As a result, the switching activity in a prefix circuit not only depends on its logic depth but also on the number of operation nodes at each level. The circuit with shorter depth and more nodes might have more switching activity than the one with longer depth and fewer nodes.

### **3.2.2 Power Consumption and Fan-out**

Besides the switching activity at an operation node, the node's fan-out also has an effect on power consumption in a circuit design in VLSI [Ca196, WE93]: the larger the fan-out, the more power the circuit consumes because there are more signals. For example, by using the PSpice over random input patterns, the power consumed by a 2-input XOR gate is dependent upon the fan-out and the relationship is linear (Figure 3.11). Hence, fan-out should be taken into account when a power consumption estimate is made for the prefix circuit.



**Figure 3.11:** Effect of fan-out on power consumption of a 2-input XOR gate.

### **3.3 The Circuit-level Simulation: PSpice**

The circuit-level simulation called SPICE (Simulation Program for Integrated Circuit Emphasis) is a powerful general purpose analog and digital circuit simulator that is used to verify circuit design and to predict the circuit behavior under a variety of different circumstances. The program SPICE was originally developed at the Electronics Research Laboratory of the University of California at Berkeley in early 1970's and has become a *de facto* standard in the area of analog and digital simulation. SPICE is often used to characterize logic cells. The software performs a simulation of the design and monitors the power supply current waveform. This technique gives accurate power consumption. But it is very time-consuming.

In this study, we use a PC version of SPICE called PSpice [Cad00]. PSpice is registered trademark of Orcad Corporation and it is the most popular circuit simulation software on the market today. PSpice offers a large library of models obtained from files of standard components, semiconductor manufactures, and user inputs so that users can run simulations with confidence and get accurate results. Circuits are entered using a

schematic capture editor, which can access the component and symbol libraries. The simulation results take the form of textual, tabular, and graphical output, depending on the analysis performed and the Probe post-processor displays output data in the form of graphs.

In the next chapter, we will analyze switching activity and fan-out for each prefix circuit considered. We then use this to further estimate and investigate the power-speed trade-off between various types of prefix circuits.

## CHAPTER 4

### POWER MODELING OF PREFIX CIRCUITS

Having seen the various sources of power consumption in general circuits we now focus on analytical model for predicting the average power consumption of a prefix circuit. As mentioned previously, the signal switching activity has a major influence on the power consumption. Therefore, the switching activity will be used as a basis to determine power consumption of prefix circuits. Further, as mentioned in Section 3.2.2, the power consumption of an operation node is a linear function of fan-out [Cal96]. Therefore, to take into account the effect of fan-out on the output load capacitance of an operation node, we assume that the load capacitance of a node with fan-out  $k$  is equal to  $C_0 + C'(k - 1)$ , where  $C_0$  is the load capacitance of a node with fan-out 1, and  $C'$  is the load capacitance for each additional fan-out [Smi97].

The *effective circuit capacitance* of a prefix circuit,  $cap_{eff}(N)$ , is the effective load capacitance of all nodes in the circuit. As defined here, the effective circuit capacitance depends on input signal patterns and the effects of signal glitching. Thus if a node output experiences two transitions due to glitching, its effective capacitance is twice that of the physical capacitance. Because the degree of glitching depends on input signal patterns, we consider derivations of the worst case scenario in which glitching at the nodes are assumed to be the maximum possible. By scaling the effective circuit capacitance by the circuit clock frequency and  $V_{DD}^2$ , we arrive at our power estimate.



$$P = cap_{eff}(N)V_{DD}^2 f. \quad (4.1)$$

The capacitance evaluation for various circuits according to our model is made in two steps. As a first step, in Section 4.1, we assume that the load capacitance for each operation node is independent of the fan-out, i.e., the load capacitance in the constant  $C_0$ . In the second step we first compute the residual network by deleting one output of each operation node with fan-out  $\geq 1$ . We then compute the load capacitance of the residual circuit assuming that the load capacitance of each node is  $C'$ , independent of the fan-out. This step is repeated  $k - 1$  times where  $k$  is the fan-out of the given circuit. This step is performed in Section 4.2. The total capacitance is the sum of the values obtained in step 1 and step 2.

#### 4.1 Step 1 - The Constant Output Capacitance

In this step, we assume that the physical output capacitance of each operation node is constant. Let  $Kcap_{eff}(N)$  be the effective circuit capacitance under the constant output capacitance assumption,  $depth(N)$  be the depth of the circuit,  $w_i$  be the number of operation nodes in the circuit at level  $i$ , and  $C_0$  as the assumed constant load capacitance of one node. Then from Eq. 3.2,

$$Kcap_{eff}(N) = \left( \sum_{i=1}^{depth(N)} iw_i \right) C_0 \quad (4.2)$$

In the following, we use this equation to derive  $Kcap_{eff}(N)$  of the various prefix circuits.

#### 4.1.1. The Serial Prefix Circuit

From the layout of the serial prefix circuit in Figure 4.1, we see that each level contains one operation node and each operation node has exactly two fan-ins and two fan-outs. The size and depth of this circuit is  $(N-1)$ . As shown in Figure 4.2, the  $S(N)$  circuit can be built from the  $S(N-1)$  circuit by adding a new input into the  $S(N-1)$  circuit at the  $depth(S(N))^{th}$  level (i.e., at level  $N-1$ ). Thus, we can determine the recurrence for the constant output capacitance of the serial prefix circuit for  $N$  inputs as the sum of the capacitance of  $N-1$  inputs and the capacitance of the new input at the  $depth(S(N))^{th}$  level as follows

$$Kcap_{eff}(N) = Kcap_{eff}(N-1) + depth(S(N)) \cdot 1, \quad \text{with} \quad Kcap_{eff}(2) = 1.$$

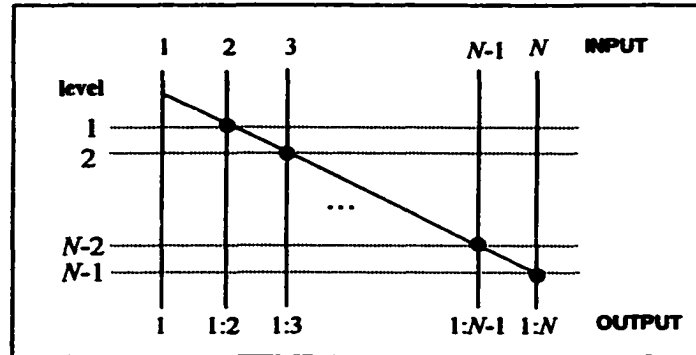


Figure 4.1: An illustration of the serial prefix circuit,  $S(N)$ .

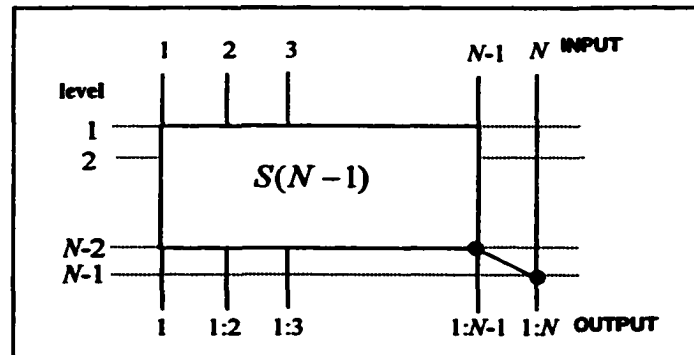


Figure 4.2: An illustration of the serial prefix circuit,  $S(N)$ , built from  $S(N-1)$ .

Therefore, the recurrence can be solved as

$$\begin{aligned}
 Kcap_{eff}(N) &= Kcap_{eff}(N-1) + depth(S(N)) \cdot 1 \\
 &= Kcap_{eff}(N-1) + (N-1) \\
 &= Kcap_{eff}(N-2) + (N-2) + (N-1) \\
 &\quad \vdots \\
 &= Kcap_{eff}(2) + (2) + \dots + (N-3) + (N-2) + (N-1) \\
 &= 1 + \sum_{i=2}^{N-1} i \\
 &= \frac{N(N-1)}{2} \\
 &= O(N^2)
 \end{aligned}$$

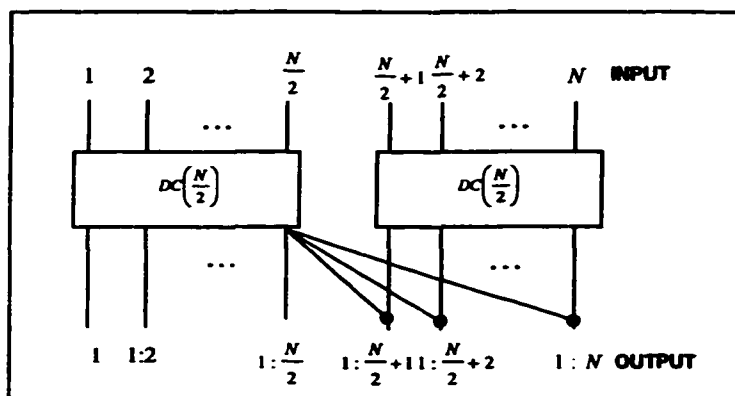
The size and depth of the serial prefix circuit is  $(N-1)$ . Therefore,  $Kcap_{eff}(N)$  can be written as a function of the circuit's size ( $s$ ) and depth ( $d$ ) as follows

$$\begin{aligned}
 Kcap_{eff}(N) &= \frac{N(N-1)}{2} \\
 &= \frac{1}{2}s(d+1) \\
 &= \frac{1}{2}(sd+s) \\
 &= \frac{sd+s}{2}
 \end{aligned}$$

Obviously, the serial prefix circuit has  $O(N)$  size,  $O(N)$  depth and  $O(N^2)$  effective circuit capacitance under the constant output capacitance.

#### 4.1.2. The Divide-and-Conquer Parallel Prefix Circuit

Let  $N = 2^n$ . Using the well-known divide-and-conquer strategy, the divide-and-conquer parallel prefix circuit,  $DC(N)$ , is designed according to the principle illustrated in Figure 4.3. That is,  $DC(N)$  is built from two  $DC(N/2)$  circuits and by connecting output  $1:N/2$  from the first  $DC(N/2)$  to each of the output of the second  $DC(N/2)$  at level  $depth(DC(N/2)) + 1$ . Therefore, the circuit's,  $Kcap_{eff}(N)$ , can be derived from that of  $DC(N/2)$ , according to the following recurrence relation,



**Figure 4.3:** An illustration of the divide-and-conquer prefix circuit,  $DC(N)$ , built from  $DC(N/2)$ , derived from [LD94].

$$Kcap_{eff}(N) = 2Kcap_{eff}\left(\frac{N}{2}\right) + \left(depth(DC\left(\frac{N}{2}\right)) + 1\right) \frac{N}{2}, \quad \text{with} \quad Kcap_{eff}(2) = 1.$$

The first part of  $Kcap_{eff}(N)$  is the constant output capacitance from the two circuits with  $N/2$  inputs while the second part is the capacitance from the last level of  $DC(N)$ . Since there are  $N/2$  operation nodes at the last level, the circuit depth is  $depth(DC(N/2)) + 1$ .

Recall that  $depth$  of  $DC(N) = \lg N$ . Therefore, we have

$$Kcap_{eff}(N) = 2Kcap_{eff}\left(\frac{N}{2}\right) + \left(\lg \frac{N}{2} + 1\right) \frac{N}{2}$$

$$\begin{aligned}
&= 2^2 Kcap_{eff} \left( \frac{N}{2^2} \right) + 2^1 \left( \lg \frac{N}{2^2} + 1 \right) \frac{N}{2^2} + 2^0 \left( \lg \frac{N}{2^1} + 1 \right) \frac{N}{2^1} \\
&\quad \vdots \\
&= 2^{n-1} Kcap_{eff} \left( \frac{N}{2^{n-1}} \right) + 2^{n-2} \left( \lg \frac{N}{2^{n-1}} + 1 \right) \frac{N}{2^{n-1}} + \dots + 2^1 \left( \lg \frac{N}{2^2} + 1 \right) \frac{N}{2^2} + 2^0 \left( \lg \frac{N}{2^1} + 1 \right) \frac{N}{2^1} \\
&= 2^{n-1} Kcap_{eff}(2) + 2^{n-2} (\lg 2^1 + 1) 2^1 + \dots + 2^1 (\lg 2^{n-2} + 1) 2^{n-2} + 2^0 (\lg 2^{n-1} + 1) 2^{n-1} \\
&= 2^{n-1} (1) + 2^{n-1} (2) + \dots + 2^{n-1} (n-1) + 2^{n-1} (n-1+1) \\
&= 2^{n-1} \sum_{i=1}^n i \\
&= 2^{n-1} \frac{n(n+1)}{2} \\
&= \frac{N}{4} (\lg N)^2 + \lg N \\
&= O(N(\lg N)^2)
\end{aligned}$$

We also can write  $Kcap_{eff}(N)$  in terms of circuit size (i.e.,  $s = \frac{N}{2} \lg N$ ) and circuit depth

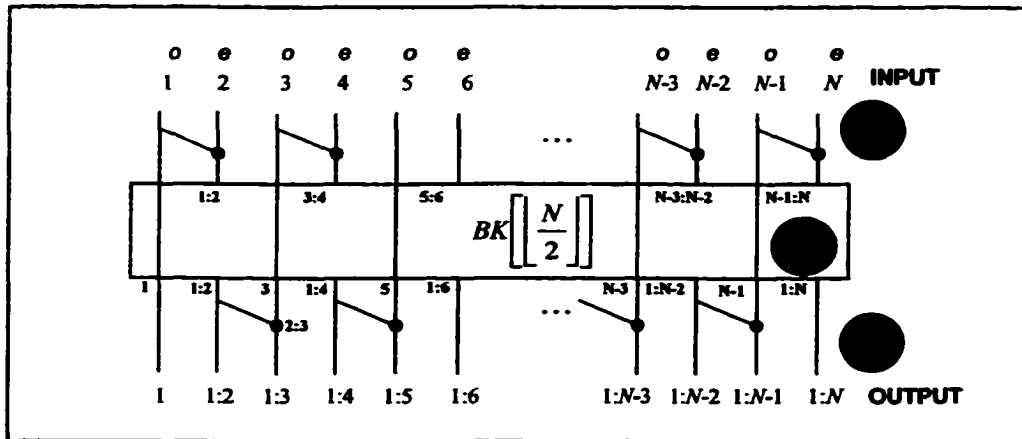
(i.e.,  $d = \lg N$ ), as follows.

$$\begin{aligned}
Kcap_{eff}(N) &= \frac{N}{4} (\lg N)^2 + \lg N \\
&= \frac{N}{4} (\lg N)^2 + \frac{N}{4} \lg N + \frac{N}{4} (\lg N)^2 - \frac{N}{4} (\lg N)^2 \\
&= \frac{N}{2} (\lg N)^2 + \frac{1}{2} \cdot \frac{N}{2} \lg N - \frac{1}{2} \cdot \frac{N}{2} (\lg N)^2 \\
&= sd + \frac{1}{2}s - \frac{1}{2}sd \\
&= \frac{sd + s}{2}
\end{aligned}$$

Thus, the divide-and-conquer prefix circuit has  $O(N \lg N)$  size,  $O(\lg N)$  depth and  $O(N(\lg N)^2)$  effective circuit capacitance under the constant output capacitance assumption

#### 4.1.3 The Brent-Kung Parallel Prefix Circuit

Let  $N = 2^n$ . The Brent-Kung prefix circuit for  $N$  inputs,  $BK(N)$ , is also built from the Brent-Kung circuit for  $N/2$  inputs, as shown in Figure 4.4. The recurrence relation for this circuit is, however, not as straightforward as the previous two circuits. The part of the problem arises because  $BK(N/2)$  occupies the middle level, which causes the level of all nodes in  $BK(N/2)$  to increase. This requires taking into account the number of nodes at



**Figure 4.4:** A Brent-Kung parallel prefix circuit,  $BK(N)$ , divided into three parts ( $o$  = odd,  $e$  = even), derived from [LD94].

each level of  $BK(N/2)$ , and is not at all difficult to overcome. However, the major problem is the last step where output of  $BK(N/2)$  is combined with half of the inputs as illustrated in part C of Figure 4.4. Although all these nodes appear at the last level of the

circuit, in fact, some of them are at lower level. To determine level of each node, we construct a table (Table 4.1) for  $BK(N)$  corresponding to the circuit layout. The table is divided into three parts,  $A$ ,  $B$ , and  $C$ , corresponding to the circuit layout in Figure 4.4. The entries of the form  $(x_i, i)$  in the table represent the fact that level  $i$  has  $x_i$  nodes. The first row is divided into two parts – column 1 corresponding to part  $B$ , and column 2 corresponding to part  $C$  while the second row is represented by part  $A$ . Computation for capacitance corresponding to part  $B$  is simple. In this part there are  $N/2$  operation nodes – all at level 1. Hence, capacitance of part  $B$  is equal to  $N/2$ . Computation for part  $A$  can also be achieved easily by observing that all inputs to  $BK(N/2)$  are at level 1, which cause the level of each node in  $BK(N/2)$  to increase by 1. Let  $w_i$  be the number of nodes at level  $i$  in  $BK(N/2)$ .

Then,

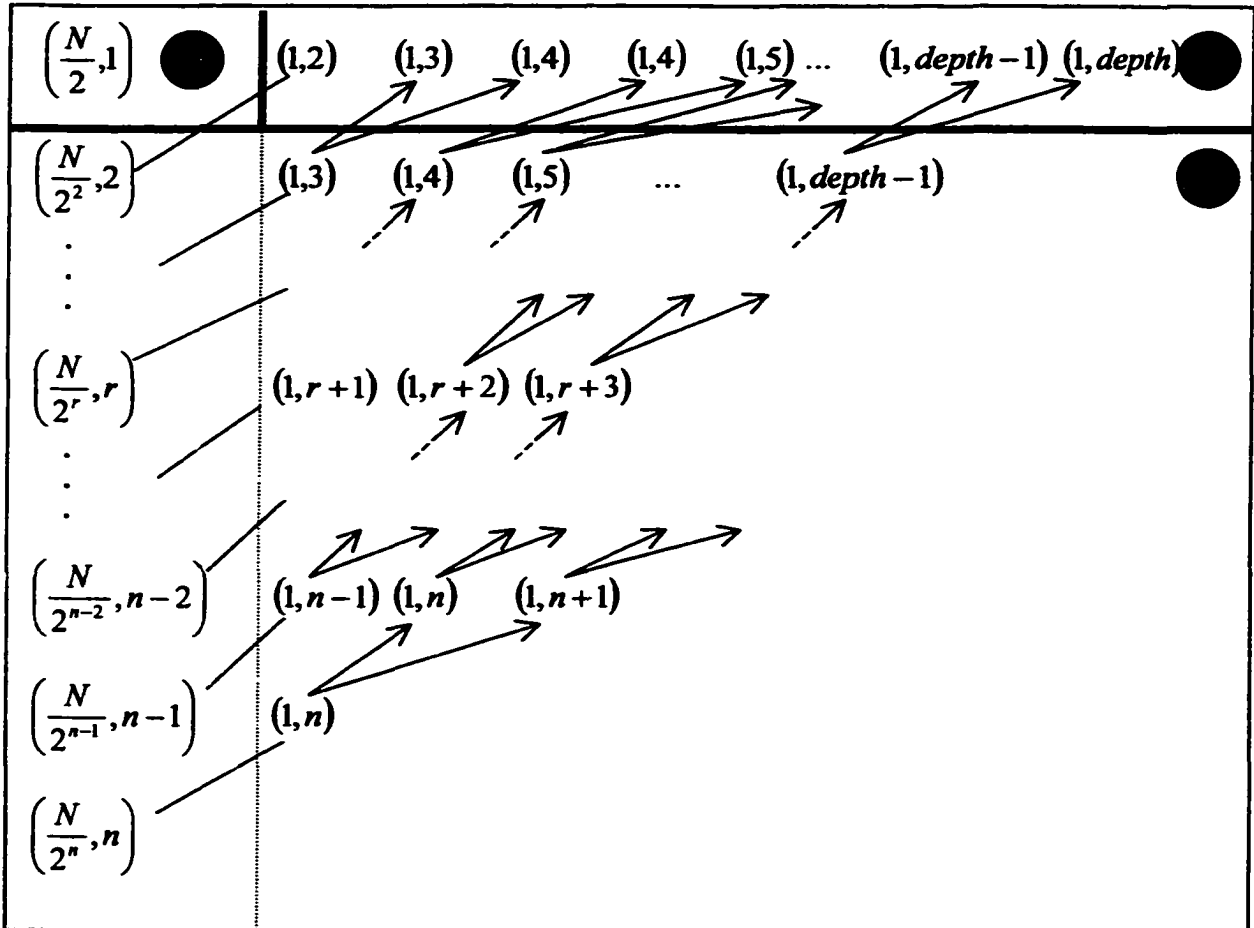
$$Kcap_{eff}(PartA) = \sum_{i=1}^{depth(BK(N/2))} w_i(i+1) = \sum_{i=1}^{depth(BK(N/2))} iw_i + \sum_{i=1}^{depth(BK(N/2))} w_i = Kcap_{eff}\left(\frac{N}{2}\right) + N - \lg \frac{N}{2} - 2.$$

Note that part  $C$  has  $(N/2 - 1)$  operation nodes. Though in the circuit diagram they appear to be at the last level, in fact they are distributed at different levels of the circuit. To compute the capacitance for part  $C$ , let capacitance of part  $C$  be denoted as  $K(N)$ .

A row in Table 4.1 represents the first level (i.e., column 1) and the last level (i.e., column 2) of  $BK(N/2^k)$  circuit, for  $0 \leq k \leq \lg N - 1$ , after distributing all nodes of the last level to the lower level. Let row  $i$  of column 2 in Table 4.1 be  $K(N/2^k)$ , where  $1 \leq i \leq \lg N$  and  $0 \leq k \leq \lg N - 1$ . For example, the first row (i.e., part  $C$  in Table 4.1) represents the nodes used to be at the last level of  $BK(N)$  circuit (i.e., part  $C$  in Figure

4.4). The second row represents the nodes used to be the last level of  $BK(N/2)$  circuit (i.e., the subpart  $C$  of part  $A$  in Figure 4.4). The relationship of each row in the table is as follows:

**Table 4.1:** The constant output capacitance table for  $BK(N)$ .



- The first entry of column 2 at row  $i$  is generated from the entry at row  $(i+1)$ , locating at row  $i$ 's diagonal in column 1, as one operation node having the same circuit depth as  $(i+1)$ 's entry (see the line |). For example, the entry  $(1,n)$  at row  $\lg N - 1$  is generated from the entry  $(N/2^n, n)$  at row



$\lg N$ . Then this new output entry at row  $i$  produces two entries at row  $(i-1)$ : one operation node having the same circuit depth as  $i$ 's entry and one operation node having one more circuit depth than the  $i$ 's entry (see the arrow $\uparrow$ ). For example, the entry  $(1, n)$  at row  $\lg N - 1$  produces the entry  $(1, n)$  and the entry  $(1, n+1)$  at row  $\lg N - 2$ .

Therefore, in column 2, the first row,  $K(N)$ , (i.e., part C in Table 4.1) is derived from the second row,  $K(N/2)$  (see Figure 4.5). The  $K(N)$  is written as follows.

$$\begin{aligned}
 K(N) &= 2 + (k_1 + 1) + (k_1 + 2) + (k_2 + 1) + (k_2 + 2) + \dots + (k_{\frac{N}{4}-1} + 1) + (k_{\frac{N}{4}-1} + 2) \\
 &= 2 + 2(k_1 + k_2 + \dots + k_{\frac{N}{4}-1}) + \left(\frac{N}{4} - 1\right) + 2\left(\frac{N}{4} - 1\right) \\
 &= 2K\left(\frac{N}{2}\right) + \frac{3N}{4} - 1
 \end{aligned}$$

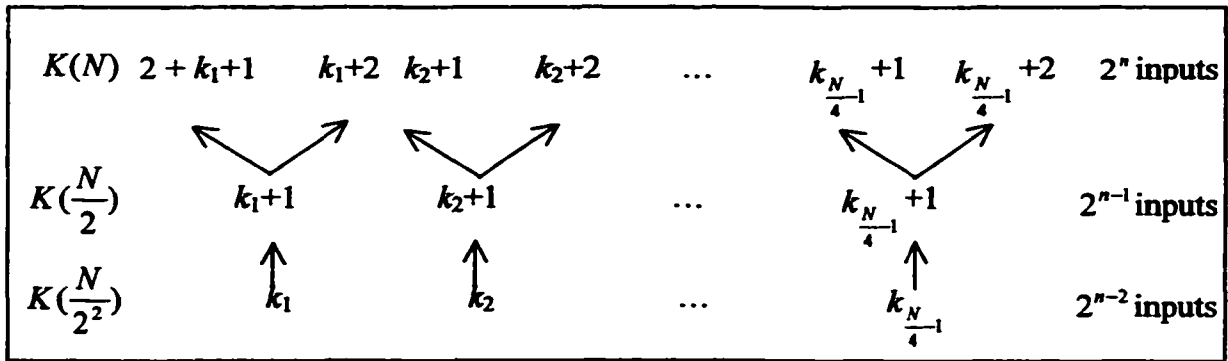


Figure 4.5: Part C, the distribution of  $N/2 - 1$  nodes.

Solving for  $K(N)$ , we obtain  $K(N) = (3/4)N \lg N - 5N/4 + 1$ . Thus  $Kcap_{eff}(N)$ , which is the sum of capacitances of Part A, B and C, can be written as follows:

$$Kcap_{eff}(N) = Kcap_{eff}\left(\frac{N}{2}\right) + \left[ \frac{3}{2} \cdot \frac{N}{2} \cdot \left(\lg \frac{N}{2}\right) + 2 \cdot \frac{N}{2} - \left(\lg \frac{N}{2}\right) - 1 \right]$$

$$\begin{aligned}
&= Kcap_{eff}\left(\frac{N}{2^2}\right) + \left[ \frac{3}{2} \cdot \frac{N}{2^2} \cdot (\lg \frac{N}{2^2}) + 2 \cdot \frac{N}{2^2} - (\lg \frac{N}{2^2}) - 1 \right] + \left[ \frac{3}{2} \cdot \frac{N}{2} \cdot (\lg \frac{N}{2}) + 2 \cdot \frac{N}{2} - (\lg \frac{N}{2}) - 1 \right] \\
&\quad \vdots \\
&= Kcap_{eff}\left(\frac{N}{2^{n-1}}\right) + \left[ \frac{3}{2} \cdot \frac{N}{2^{n-1}} \cdot \lg \frac{N}{2^{n-1}} + 2 \cdot \frac{N}{2^{n-1}} - \lg \frac{N}{2^{n-1}} - 1 \right] + \dots \\
&\quad + \left[ \frac{3}{2} \cdot \frac{N}{2^1} \cdot \lg \frac{N}{2^1} + 2 \cdot \frac{N}{2^1} - \lg \frac{N}{2^1} - 1 \right] \\
&= Kcap_{eff}(2) + \left[ \frac{3}{2} \cdot \frac{N}{2^{n-1}} \cdot (\lg \frac{N}{2^{n-1}}) + 2 \cdot \frac{N}{2^{n-1}} - (\lg \frac{N}{2^{n-1}}) - 1 \right] + \dots \\
&\quad + \left[ \frac{3}{2} \cdot \frac{N}{2^1} \cdot (\lg \frac{N}{2^1}) + 2 \cdot \frac{N}{2^1} - \lg \frac{N}{2^1} - 1 \right] \\
&= 1 + \left[ \frac{3}{2} \cdot \frac{N}{2^{n-1}} \cdot \lg \frac{N}{2^{n-1}} + 2 \cdot \frac{N}{2^{n-1}} - \lg \frac{N}{2^{n-1}} - 1 \right] + \dots + \left[ \frac{3}{2} \cdot \frac{N}{2^1} \cdot \lg \frac{N}{2^1} + 2 \cdot \frac{N}{2^1} - \lg \frac{N}{2^1} - 1 \right] \\
&= 1 + \frac{3}{2} \left[ \left( \frac{N}{2^{n-1}} \cdot \lg \frac{N}{2^{n-1}} \right) + \left( \frac{N}{2^{n-2}} \cdot \lg \frac{N}{2^{n-2}} \right) + \dots + \left( \frac{N}{2^1} \cdot \lg \frac{N}{2^1} \right) \right] \\
&\quad + 2 \left[ \frac{N}{2^{n-1}} + \frac{N}{2^{n-2}} + \dots + \frac{N}{2^1} \right] - \left[ \lg \frac{N}{2^{n-1}} + \lg \frac{N}{2^{n-2}} + \dots + \lg \frac{N}{2^1} \right] - (n-1) \\
&= \frac{3}{2} \sum_{i=0}^{n-1} i 2^i + 2 \sum_{i=0}^{n-1} 2^i - \sum_{i=0}^{n-1} i - n \\
&= \frac{3}{2} [2 + n \cdot 2^n - 2 \cdot 2^n] + 2[2^n - 1] - \frac{n(n-1)}{2} - n \\
&= \left[ 3 + \frac{3}{2} n \cdot 2^n - 3 \cdot 2^n \right] + [2 \cdot 2^n - 2] - \frac{n^2}{2} + \frac{n}{2} - n \\
&= 1 - N + \frac{3}{2} N \lg N - \frac{(\lg N)^2}{2} - \frac{\lg N}{2} \\
&= \left[ 1 + \frac{3}{2} N \lg N \right] - \frac{1}{2} [2N + (\lg N)^2 + \lg N] \\
&= O(N \lg N)
\end{aligned}$$

We can also write  $Kcap_{eff}(N)$  in terms of size and depth. Thus,

$$\begin{aligned}
Kcap_{eff}(N) &= \left[1 + \frac{3}{2}N \lg N\right] - \frac{1}{2}[2N + (\lg N)^2 + \lg N] \\
&= \left[2N \lg N - (\lg N)^2 - \frac{3}{2}\lg N - N + 1\right] - \frac{1}{2}\lg N[N - \lg N - 2] \\
&= \left[\frac{sd + s}{2}\right] - \left[\left(\frac{d}{4} + \frac{1}{2}\right) \cdot \left(\frac{s}{2} - \frac{d}{4} - \frac{3}{2}\right)\right] \\
&= \frac{3sd}{8} + \frac{d^2}{16} + \frac{s}{4} + \frac{d}{2} + \frac{3}{4}
\end{aligned}$$

Similar to the previous circuits, the constant output capacitance is  $O(sd)$  ( $O(N \lg N)$ ).

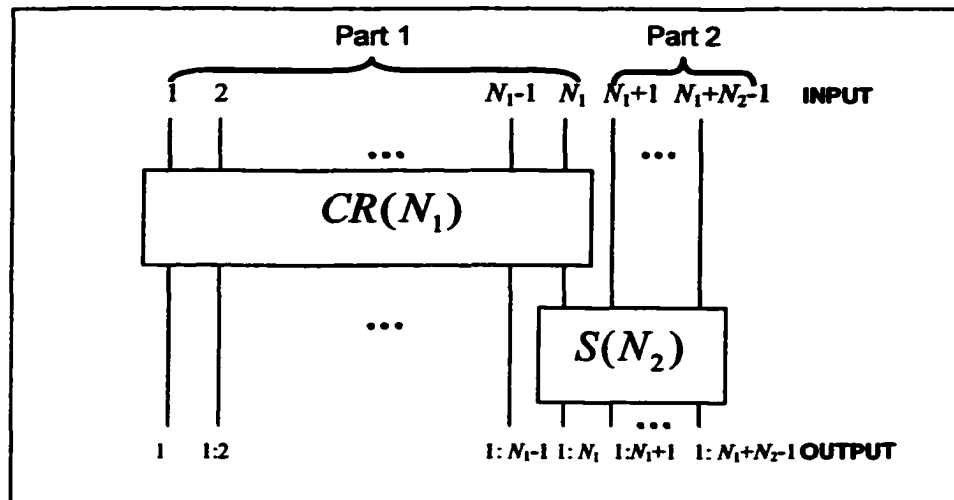
#### 4.1.4 The Ladner-Fischer Parallel Prefix Circuit

As described in Chapter 2, Ladner and Fischer [LF80] introduced the family of circuits  $LF_k(N)$  when  $0 \leq k \leq \lceil \lg N \rceil$ . Different values of  $k$  give different prefix circuits' structures. However,  $LF_k(N)$ s are bounded by the divide-and-conquer prefix circuit and the Brent-Kung prefix circuit. The  $LF_0(N)$  prefix circuit has the shortest depth and biggest sizes among the family of circuits  $LF_k(N)$ . Also the  $LF_0(N)$  prefix circuit has the same depth as the divide-and-conquer circuit. Both circuits' structures are similar with small input  $N$ . But the size of the  $LF_0(N)$  circuit is smaller than that of the divide-and-conquer circuit when  $N$  is larger. Thus, the constant output capacitance of the  $LF_k(N)$  circuit is lower bounded by the constant output capacitance of the divide-and-conquer circuit. The  $LF_k(N)$  prefix circuit behaves like the Brent-Kung prefix circuit when  $k \geq \lg N - 2$ . Therefore, the upper bound of the  $LF_k(N)$  prefix circuit is the constant output capacitance of the Brent-Kung circuit.

To summarize, the effective capacitance under the constant output capacitance assumption is in the range  $Kcap_{eff}(DC(N)) \leq Kcap_{eff}(LF_k(N)) \leq Kcap_{eff}(BK(N))$ . That is  $N((\lg N)^2 + \lg N)/4 \leq Kcap_{eff}(LF_k(N)) \leq [1 + (3N \lg N)/2] - [2N + (\lg N)^2 + \lg N]/2$ .

#### 4.1.5 The Snir Parallel Prefix Circuit

The Snir parallel prefix circuit,  $SN(N)$ , is composed of two parts as shown in Figure 4.6. The two parts consist of the compressed layered prefix circuit,  $CR(N_1)$ , and the serial prefix circuit,  $S(N_2)$ , where  $N = N_1 + N_2 - 1$ . Therefore, the capacitance is computed by summing the capacitance of these two parts.



**Figure 4.6:** The  $SN(N)$  circuit,  $SN(N) = CR(N_1) \cdot S(N_2)$ .

When  $N = 2^n$ , the Brent-Kung prefix circuit is the compressed layered prefix circuit. Therefore, we can use the Brent-Kung prefix circuit's capacitance formula for the compressed layered prefix circuit's capacitance formula. When  $N \neq 2^n$ , this formula overestimates the capacitance by less than 7% (see Appendix E).

In the Snir prefix circuit, the capacitance of the first part can be computed by using the formula from the Brent-Kung parallel prefix circuit described in Section 4.1.3 whereas the capacitance of the second part can be computed by starting from the level of the last output of Part 1 (i.e.,  $\lceil \lg N_1 \rceil$ ). There are  $N_2 - 1$  operations. Each operation is at a succeeding level. Therefore, the capacitance of the second part is given by

$$\begin{aligned} Kcap_{eff}(S(N_2)) &= \sum_{i=1}^{N_2-1} (\lceil \lg N_1 \rceil + i) \\ &= \left( (N_2 - 1) \lceil \lg N_1 \rceil + \sum_{i=1}^{N_2-1} i \right) \\ &= \left( N_2 \lceil \lg N_1 \rceil - \lceil \lg N_1 \rceil + \frac{N_2^2 - N_2}{2} \right) \end{aligned}$$

The constant output capacitance of the circuit  $SN(N)$  is given by

$$\begin{aligned} Kcap_{eff}(SN(N)) &= Kcap_{eff}(CR(N_1)) + Kcap_{eff}(S(N_2)) \\ &= \left[ 1 + \frac{3}{2} N_1 \lceil \lg N_1 \rceil \right] - \frac{1}{2} [2N_1 + (\lceil \lg N_1 \rceil)^2 + \lceil \lg N_1 \rceil] + \\ &\quad N_2 \lceil \lg N_1 \rceil - \lceil \lg N_1 \rceil + \frac{N_2^2 - N_2}{2} \end{aligned}$$

Clearly, capacitance of the circuit  $SN(N)$  is  $O(N \lg N)$ .

#### 4.1.6 The Shih-Lin Parallel Prefix Circuit

The  $SL(N)$  parallel prefix circuit [LS99] is composed of two parts consisting of the layered prefix circuit,  $CR(N_1)$ , and the serial prefix circuit  $S(N_2)$ , where  $N = N_1 + N_2 - 1$  (see Figure 4.7). Therefore, the capacitance is computed by summing the capacitance from these two parts. As discussed in the previous section, the

capacitance of the first part can be computed by using the formula from the Brent-Kung parallel prefix circuit described in Section 4.1.3 whereas the constant capacitance of the second part can be computed by computing the capacitance of the serial prefix circuit and the capacitance of the connecting nodes starting from the last output of Part 1 (i.e.,  $\lceil \lg N_1 \rceil$ ). Therefore, the constant capacitance of the second part as before is

$$N_2 \lceil \lg N_1 \rceil - \lceil \lg N_1 \rceil + (N_2^2 - N_2)/2.$$

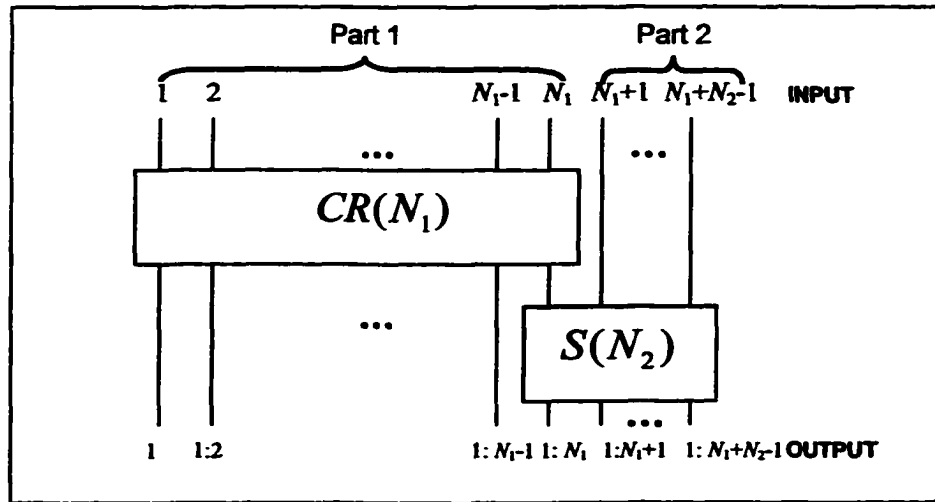


Figure 4.7: The  $SL(N)$  circuit,  $SL(N) = CR(N_1) \cdot S(N_2)$ .

The constant output capacitance of the circuit  $SL(N)$  is

$$\begin{aligned} Kcap_{eff}(SL(N)) &= Kcap_{eff}(CR(N_1)) + Kcap_{eff}(S(N_2)) \\ &= \left[ 1 + \frac{3}{2} N_1 \lceil \lg N_1 \rceil \right] - \frac{1}{2} [2N_1 + (\lceil \lg N_1 \rceil)^2 + \lceil \lg N_1 \rceil] + \\ &\quad N_2 \lceil \lg N_1 \rceil - \lceil \lg N_1 \rceil + \frac{N_2^2 - N_2}{2} \end{aligned}$$

Clearly, the effective capacitance of the circuit  $SL(N)$  under the constant output capacitance assumption is also  $O(N \lg N)$ .

#### 4.1.7 The LYD Parallel Prefix Circuit

The LYD parallel prefix circuit,  $LYD(N)$ , is composed of four parts including the layered prefix circuit,  $CR(N_1)$ ,  $Q(N_2)$ ,  $S(N_3)$ , and  $S(N_4)$  where  $N = N_1 + N_2 + N_3 + N_4$  (see Figure 4.8). Therefore, the capacitance is computed by summing the capacitance from these four parts. The capacitance of Part 1 can be computed by using the formula from the Brent-Kung parallel prefix circuit described in Section 4.1.3. The capacitance of Part 2, Part 3, and Part 4 can be computed by starting from the level of the last output of the first part, second part and the third part, respectively.

**Part 1** is the  $CR(N_1)$  circuit. Thus, the capacitance of Part 1 is

$$= \left\{ \left[ 1 + \frac{3}{2} N_1 (\lg N_1) \right] - \left[ \frac{1}{2} [2N_1 + (\lg N_1)^2 + (\lg N_1)] \right] \right\}$$

**Part 2** is the  $Q(N_2)$  circuit. Let  $t$  be  $\lceil \lg N_1 \rceil$ . For level  $i = 1$  to  $t$ , level  $i$  has

$(t - i + 1)$  operation nodes. Thus, the capacitance is  $\sum_{i=1}^t i \cdot (t - i + 1)$ .

At level  $(t + 1)$ , there are  $(t + 1)$  operation nodes.

At level  $(t + 2)$ , there are  $t(t - 1)/2$  operation nodes.

Thus, the capacitance of Part 2 is

$$\begin{aligned}
&= \sum_{i=1}^t i \cdot (t-i+1) + (t+1) \cdot (\lceil \lg N_1 \rceil + 1) + \frac{t(t-1)}{2} \cdot (\lceil \lg N_1 \rceil + 2) \\
&= \left( t \sum_{i=1}^t i - \sum_{i=1}^t i^2 + \sum_{i=1}^t i \right) + (t \cdot \lceil \lg N_1 \rceil + t + \lceil \lg N_1 \rceil + 1) + \frac{1}{2} (t^2 \lceil \lg N_1 \rceil + 2t^2 - t \lceil \lg N_1 \rceil - 2t) \\
&= \left( \frac{t^3 + 3t^2 + 2t}{6} \right) + (t \cdot \lceil \lg N_1 \rceil + t + \lceil \lg N_1 \rceil + 1) + \frac{1}{2} (t^2 \lceil \lg N_1 \rceil + 2t^2 - t \lceil \lg N_1 \rceil - 2t) \\
&= \frac{4}{6} t^3 + 2t^2 + \frac{4}{3} t + 1 \\
&= \frac{2}{3} \lceil \lg N_1 \rceil^3 + 2 \lceil \lg N_1 \rceil^2 + \frac{4}{3} \lceil \lg N_1 \rceil + 1
\end{aligned}$$

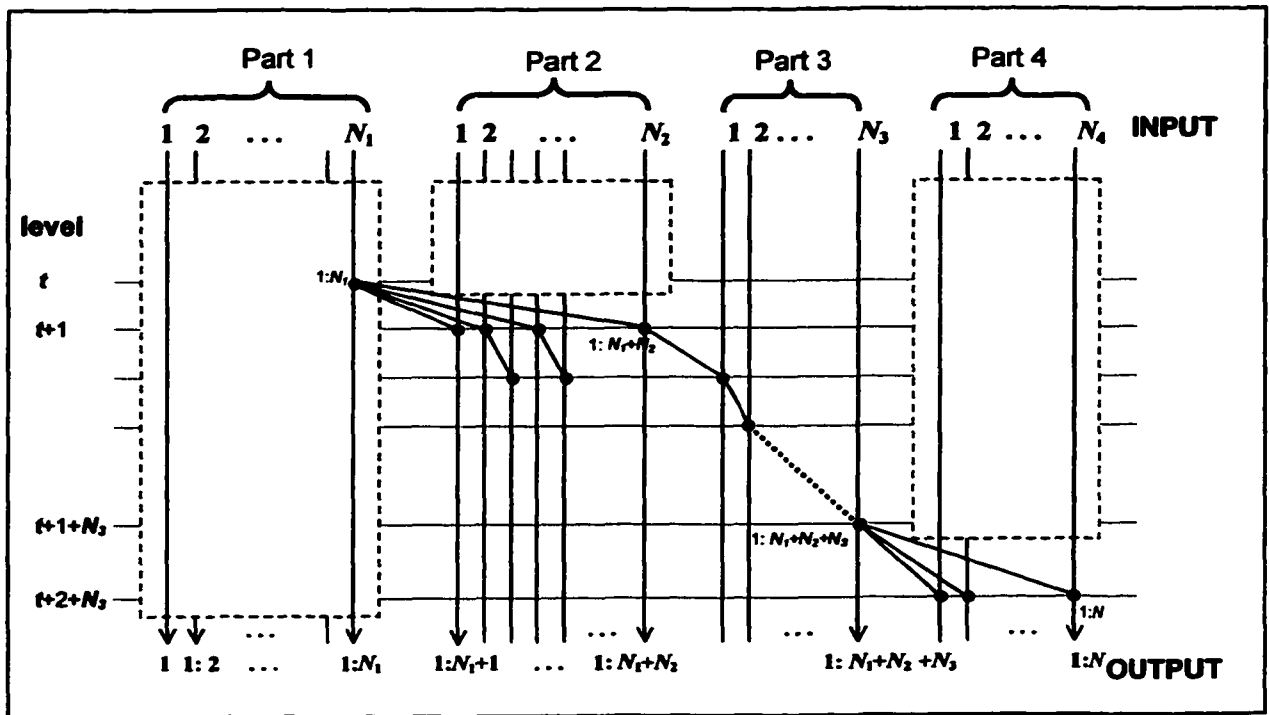


Figure 4.8: The structure of  $LYD(N)$ , derived from [LD94].

Part 3 is the  $S(N_3)$  circuit. Thus, the capacitance of Part 3 is

$$= \sum_{i=1}^{N_3} (\lceil \lg N_1 \rceil + 1 + i)$$



$$= \left( N_3 \lceil \lg N_1 \rceil + N_3 + \frac{N_3(N_3 + 1)}{2} \right)$$

**Part 4** is the  $S(N_4)$  circuit. Thus, the capacitance of Part 4 is

$$= \left( \sum_{i=1}^{N_4-1} i + N_4 (\lceil \lg N_1 \rceil + 2 + N_3) \right)$$

$$= \left( \frac{N_4(N_4 - 1)}{2} + N_4 (\lceil \lg N_1 \rceil + 2 + N_3) \right)$$

Therefore,  $LYD(N)$ 's capacitance is calculated from the sum of **Part 1**, **Part 2**, **Part 3**, and **Part 4** derived as follows

$$Kcap_{eff}(N) = \left[ 1 + \frac{3}{2} N_1 (\lg N_1) \right] - \left[ \frac{1}{2} [2N_1 + (\lg N_1)^2 + (\lg N_1)] \right] +$$

$$\left[ \frac{2}{3} \lceil \lg N_1 \rceil^3 + 2 \lceil \lg N_1 \rceil^2 + \frac{4}{3} \lceil \lg N_1 \rceil + 1 \right] +$$

$$\left[ (N_3 + N_4) \left( \lceil \lg N_1 \rceil + \frac{3}{2} \right) + \frac{1}{2} (N_3^2 + N_4^2) + (N_3 \cdot N_4) \right]$$

Clearly, capacitance of the circuit  $LYD(N)$  is also  $O(N \lceil \lg N \rceil)$ .

Table 4.2 shows all the expressions for the effective circuit capacitance for prefix circuits considered assuming the constant output capacitance assumption.

**Table 4.2.** Expression of the constant output capacitance.

| Prefix Circuit     | $Kcap_{eff}(N)$  |
|--------------------|--|
| Serial             | $\left\{ \frac{N(N-1)}{2} \right\} C_0$  |
| Divide-and-Conquer | $\left\{ \frac{N}{4} (\lg N)^2 + \lg N \right\} C_0$   |
| Brent-Kung         | $\left\{ 1 + \frac{3}{2} N \lg N - \frac{1}{2} [2N + (\lg N)^2 + \lg N] \right\} C_0$  |
| $LF_k$             | $\left\{ \frac{N}{4} (\lg N)^2 + \lg N \right\} C \leq LF_k \leq \left\{ 1 + \frac{3}{2} N \lg N - \frac{1}{2} [2N + (\lg N)^2 + \lg N] \right\} C_0$  |
| Snir               | $\left\{ \left[ 1 + \frac{3}{2} N_i (\lg N_i) \right] - \left[ \frac{1}{2} [2N_i + (\lg N_i)^2 + (\lg N_i)] \right] + \left[ N_i \lceil \lg N_i \rceil - \lceil \lg N_i \rceil + \left( \frac{N_i^2 - N_i}{2} \right) \right] \right\} C_0$  |
| Shih-Lin           | $\left\{ \left[ 1 + \frac{3}{2} N_i (\lg N_i) \right] - \left[ \frac{1}{2} [2N_i + (\lg N_i)^2 + (\lg N_i)] \right] + \left[ N_i \lceil \lg N_i \rceil - \lceil \lg N_i \rceil + \left( \frac{N_i^2 - N_i}{2} \right) \right] \right\} C_0$  |
| LYD                | $\left\{ \left[ 1 + \frac{3}{2} N_i \lg N_i \right] - \left[ \frac{1}{2} [2N_i + (\lg N_i)^2 + \lg N_i] \right] + \left[ \frac{2 \lceil \lg N_i \rceil}{3} + \lceil \lg N_i \rceil + \frac{4 \lceil \lg N_i \rceil}{3} + 1 \right] + \left[ (N_i + N_e) \left( \lceil \lg N_i \rceil + \frac{3}{2} \right) + \frac{1}{2} (N_i^2 + N_e^2) + (N_i N_e) \right] \right\} C_0$ |

#### 4.2. Step2 – Capacitance of Residual Circuit

We have assumed that a node with fan-out  $k \geq 1$ , has a physical output capacitance given as  $C_0 + (k - 1)C'$ . However, the capacitances computed in Section 4.1 for various circuits is based on the assumption that the capacitance of each node is  $C_0$  irrespective of the fan-out of the node. We still need to account for the component  $(k - 1)C'$  for a node with fan-out  $k$ ,  $k > 1$ . To get this value, we introduce the concept of the *residual circuit*. The residual circuit of a prefix circuit is the circuit obtained by eliminating one of the fan-outs from each operation node of the given prefix circuit. For example, Figure 4.13 shows the residual circuit of the divide-and-conquer prefix circuit. This residual circuit is the result of removing one of the fan-outs (i.e., the vertical fan-out) from each operation node of the circuit in Figure 4.12. We can compute the capacitance of this residual circuit,

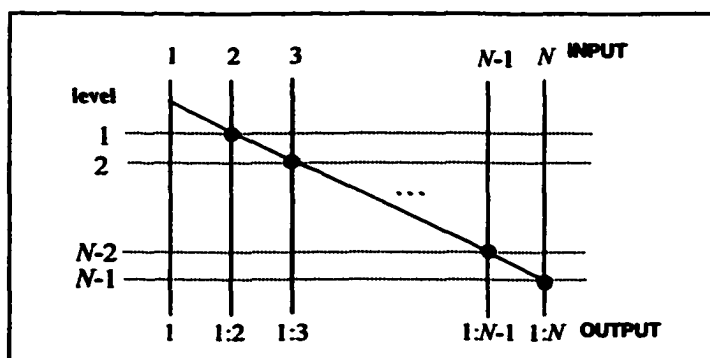
$Rcap_{eff}(N)$ , by assuming constant output capacitance (i.e.,  $C'$ ) for all operation nodes. We then construct the residual circuit of the current residual circuit by removing one fan-out from each operation node and compute its residual output capacitance. We continue accumulating the capacitances after every reduction until there are no more links to remove. Thus, the total effective circuit capacitance of the prefix circuit using the linear output capacitance assumption is given by

$$cap_{eff}(N) = Kcap_{eff}(N)C_0 + Rcap_{eff}(N)C'.$$

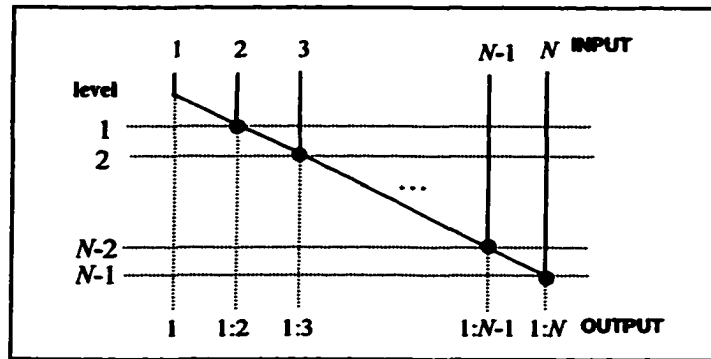
#### 4.2.1. The Serial Prefix Circuit

The layout of the serial prefix circuit for  $N$  inputs,  $S(N)$ , with fan-out shown in solid lines is illustrated in Figure 4.9. Each operation node has a fan-out of exactly two. The residual circuit obtained after removing the vertical fan-out is shown in Figure 4.10. Each operation node, except the last one, has exactly one fan-out. As shown in Figure 4.11, the residual circuit with  $N$  inputs can be built from the residual circuit with  $N-1$  inputs with the following recurrence relation:

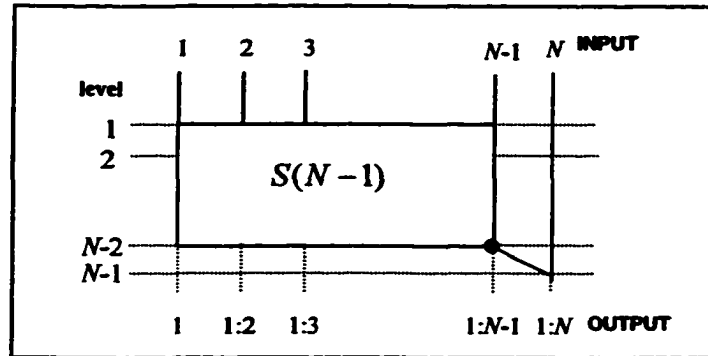
$$Rcap_{eff}(N) = Rcap_{eff}(N-1) + depth(S(N-1)) \cdot 1, \quad \text{with} \quad Rcap_{eff}(2) = 0.$$



**Figure 4.9:** The serial prefix circuit for  $N$  inputs with fan-out shown in solid lines.



**Figure 4.10:** The residual circuit of the serial prefix circuit,  $S(N)$ , shown in solid lines.



**Figure 4.11:** An illustration of the residual circuit of the  $S(N)$ , built from  $S(N-1)$ .

Since  $depth(S(N-1)) = N-2$ , solving this recurrence, we get

$$\begin{aligned}
 Rcap_{eff}(N) &= Rcap_{eff}(N-1) + (N-2) \\
 &= Rcap_{eff}(N-2) + (N-3) + (N-2) \\
 &\quad \vdots \\
 &= Rcap_{eff}(N-N+2) + (N-N+1) + \dots + (N-4) + (N-3) + (N-2) \\
 &= Rcap_{eff}(2) + (1) + \dots + (N-4) + (N-3) + (N-2)
 \end{aligned}$$

$$\begin{aligned}
&= 0 + \sum_i^{N-2} i \\
&= \frac{(N-1)(N-2)}{2}
\end{aligned}$$

The size and depth of the serial circuit are  $(N-1)$ . Therefore,  $Rcap_{eff}(N)$  can be written as,

$$\begin{aligned}
Rcap_{eff}(N) &= \frac{(N-1)(N-2)}{2} \\
&= \frac{s-(d-1)}{2} \\
&= \frac{(sd-s)}{2}
\end{aligned}$$

Thus using the linear output capacitance assumption, the effective capacitance for the serial prefix circuit is as follows.

$$\begin{aligned}
cap_{eff}(N) &= \left\{ \frac{N(N-1)}{2} \right\} C_0 + \left\{ \frac{(N-1)(N-2)}{2} \right\} C', \text{ or} \\
cap_{eff}(N) &= \left\{ \frac{sd+s}{2} \right\} C_0 + \left\{ \frac{sd-s}{2} \right\} C',
\end{aligned}$$

where  $s$  and  $d$  are the size and depth of the circuit, respectively. Both values are equal to  $N-1$ . Hence, the serial prefix circuit has  $O(N)$  size,  $O(N)$  depth, and  $O(N)^2$  effective circuit capacitance under the linear output capacitance assumption.

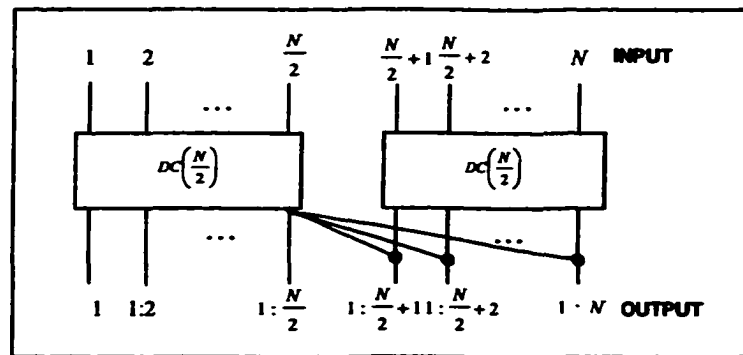
#### 4.2.2. The Divide-and-Conquer Parallel Prefix Circuit

The layout of the divide-and-conquer prefix circuit for  $N$  inputs,  $DC(N)$ , with fan-outs shown in solid lines is illustrated in Figure 4.12. The operation node at level  $depth(DC(N/2))$  has the maximum fan-out, which is  $(N/2+1)$ . After removing the

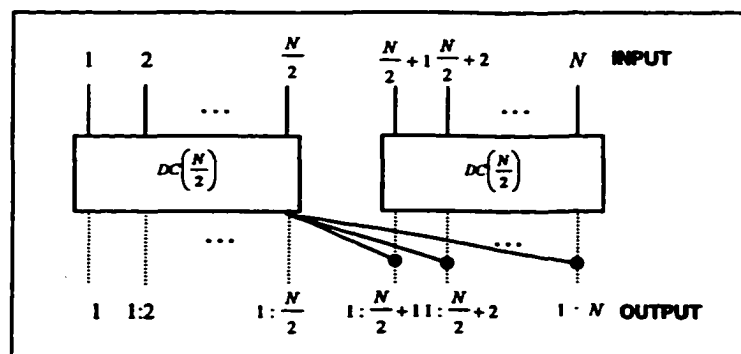
vertical fan-outs, the residual circuit is derived as shown in Figure 4.13. The operation node at level  $depth(DC(N/2))$  has the maximum fan-out, which is  $(N/2)$ .

Let  $N = 2^n$ . Using the divide-and-conquer strategy, the residual circuit of  $DC(N)$  with  $N$  inputs can be computed from the circuit of  $N/2$  inputs. Hence we can write the recurrences to compute the capacitance of the residual circuit from the parallel prefix circuit  $DC(N)$  according to the principle illustrated in Figure 4.13 as follows:

$$Rcap_{eff}(N) = 2Rcap_{eff}\left(\frac{N}{2}\right) + \frac{N}{2} \lg \frac{N}{2}, \quad \text{with} \quad Rcap_{eff}(2) = 0.$$



**Figure 4.12:** The divide-and-conquer prefix circuit,  $DC(N)$ , with fan-outs shown in solid lines, derived from [LD94].



**Figure 4.13:** The residual circuit of the divide-and-conquer prefix circuit,  $DC(N)$ , shown in solid lines.

Note that the first part of  $Rcap_{eff}(N)$  is the load capacitance of the two circuits with  $(N/2)$  inputs while the second part is the cost of merging cost of these two circuits.

Solving the recurrence, we get

$$\begin{aligned}
Rcap_{eff}(N) &= 2Rcap_{eff}\left(\frac{N}{2}\right) + \frac{N}{2} \lg \frac{N}{2} \\
&= 2^2 Rcap_{eff}\left(\frac{N}{2^2}\right) + 2^1 \frac{N}{2^2} \lg \frac{N}{2^2} + 2^0 \frac{N}{2^1} \lg \frac{N}{2^1} \\
&\quad \cdot \\
&\quad \cdot \\
&\quad \cdot \\
&= 2^{n-1} Rcap_{eff}\left(\frac{N}{2^{n-1}}\right) + 2^{n-2} \frac{N}{2^{n-1}} \lg \frac{N}{2^{n-1}} + \dots + 2^1 \frac{N}{2^2} \lg \frac{N}{2^2} + 2^0 \frac{N}{2^1} \lg \frac{N}{2^1} \\
&= 2^{n-1} Rcap_{eff}(2) + 2^{n-2} 2^1 \lg 2^1 + \dots + 2^1 2^{n-2} \lg 2^{n-2} + 2^0 2^{n-1} \lg 2^{n-1} \\
&= 2^{n-1} (1) + 2^{n-1} (2) + \dots + 2^{n-1} (n-2) + 2^{n-1} (n-1) \\
&= 2^{n-1} \sum_{i=1}^{n-1} i \\
&= 2^{n-1} \frac{n(n-1)}{2} \\
&= \frac{N}{4} ((\lg N)^2 - \lg N) \\
&= O(N(\lg N)^2)
\end{aligned}$$

We can also write the capacitance of the residual circuit as a function of size and depth as follows.

$$\begin{aligned}
Rcap_{eff}(N) &= \frac{N}{4} ((\lg N)^2 - \lg N) \\
&= \frac{N}{4} (\lg N)^2 - \frac{N}{4} \lg N + \frac{N}{4} (\lg N)^2 - \frac{N}{4} (\lg N)^2 \\
&= \frac{N}{2} (\lg N)^2 - \frac{1}{2} \cdot \frac{N}{2} \lg N - \frac{1}{2} \cdot \frac{N}{2} (\lg N)^2
\end{aligned}$$

$$\begin{aligned}
&= sd - \frac{1}{2}s - \frac{1}{2}sd \\
&= \frac{sd - s}{2}
\end{aligned}$$

Thus, using the linear output capacitance assumption, the effective capacitance for the divide-and-conquer prefix circuit is as follows.

$$\begin{aligned}
cap_{eff}(N) &= \left\{ \frac{N}{4} ((\lg N)^2 + \lg N) \right\} C_0 + \left\{ \frac{N}{4} ((\lg N)^2 - \lg N) \right\} C' \text{ or} \\
cap_{eff}(N) &= \left\{ \frac{sd + s}{2} \right\} C_0 + \left\{ \frac{sd - s}{2} \right\} C',
\end{aligned}$$

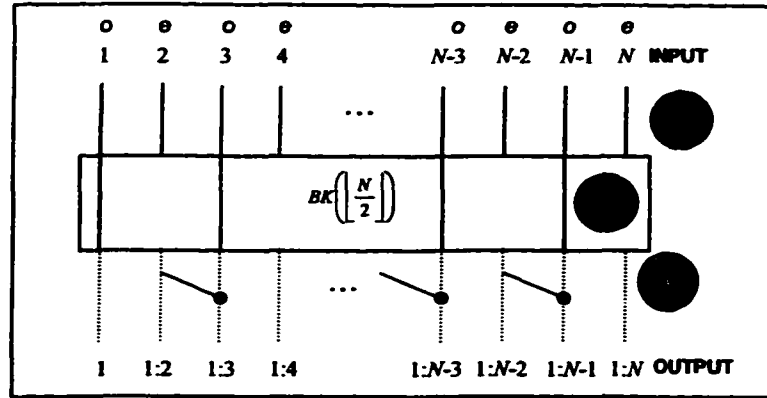
where  $s = \frac{N}{2} \lg N$  and  $d = \lg N$ .

To summarize, the divide-and-conquer prefix circuit has  $O(N \lg N)$  size,  $O(\lg N)$  depth, and  $O(N(\lg N)^2)$  effective circuit capacitance under the linear output capacitance assumption.

### 4.2.3 The Brent-Kung Parallel Prefix Circuit

Let  $N = 2^n$ . After removing the vertical fan-out from the  $BK(N)$  circuit, the residual circuit of  $BK(N)$  for  $N$  inputs can be computed from the circuit of  $N/2$  inputs, as shown in Figure 4.14. The recurrence relation for this residual circuit is, however, not as straight forward as for the previous two circuits. The number of the remaining fan-out of a node at level  $i$  is the number of the connection links to the node at level  $i+1$ . Thus, the problem is reduced to the  $BK(N)$  circuit with being shifted up one level (i.e., the  $i^{\text{th}}$  level is the





**Figure 4.14:** The residual network of the Brent-Kung parallel prefix circuit,  $BK(N)$ , divided into 3 parts.

$(i+1)^{\text{th}}$  level). The process to compute the residual circuit is like the process to compute the  $BK(N)$  circuit in Section 4.1.3. Similarly, the major problem is the last step where output of  $BK(N/2)$  is combined with half of the inputs as illustrated in part *C* of Figure 4.14. To determine exactly the level of each remaining fan-out of a node in Part *C*, we construct a table (Table 4.3) for the residual circuit of  $BK(N)$  corresponding to the residual circuit layout. The table is divided into three parts, *A*, *B*, and *C*, corresponding to the residual circuit layout in Figure 4.14.

As in Section 4.1.3, the entries of the form  $(x_i, i)$  in the table represent the fact that the node at level  $i$  has  $x_i$  fan-outs. The first row is divided into two parts – column 1 corresponds to part *B*, and column 2 corresponds to part *C*. Computation for residual network corresponding to part *B* is zero since, after shifting up, all nodes at level 1 are removed. Computation for part *A* can be achieved easily by observing that all operation nodes of the residual circuit of  $BK(N/2)$  are at one lower level, meaning that the level of each node in the residual circuit of  $BK(N/2)$  is the same as the level of  $BK(N/2)$  circuit.

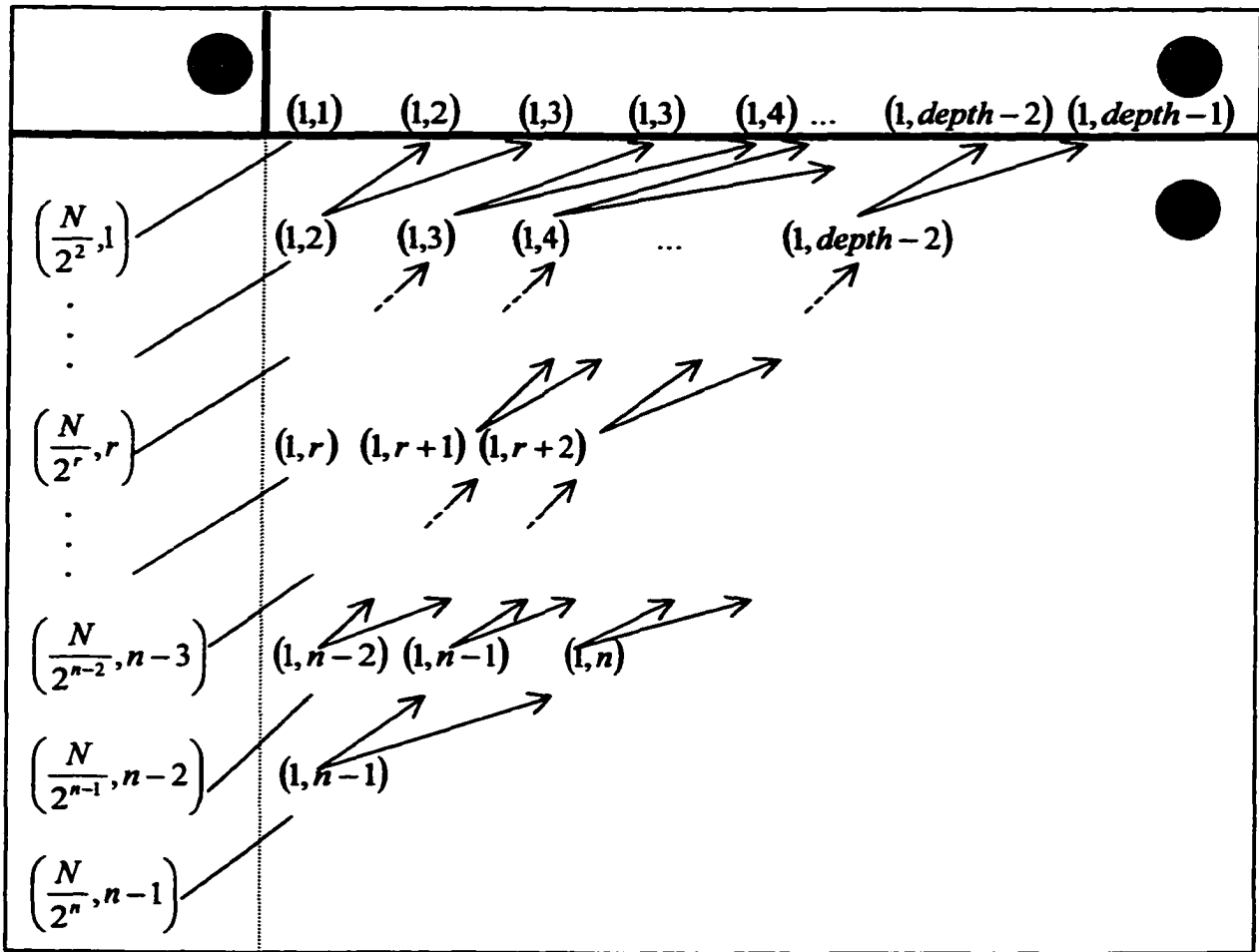
Let  $l_i$  be the number of nodes at level  $i$  in  $BK(N/2)$ . Then, the residual network of

$$Part A = \sum_{i=1}^{depth(N/2)} il_i = \sum_{i=1}^{depth(N/2)} il_i = Kcap_{eff}(N/2). \text{ To compute the capacitance for part } C, \text{ note}$$

that part  $C$  has  $\frac{N}{2} - 1$  operation nodes distributed at different levels of the circuit. Let

capitance of part  $C$  be denoted as  $K(N)$ .

**Table 4.3:** The residual circuit table for  $BK(N)$



Let row  $i$  of column 2 in Table 4.3 be  $K(N/2^k)$ , where  $0 \leq i \leq \lg N - 1$  and  $0 \leq k \leq \lg N - 1$ . Each row represents the last level of the residual circuit of

$BK(N/2^k)$  circuit, for  $0 \leq k \leq \lg N - 1$ , after level adjustment. For example, the first row (i.e., part C in Table 4.3) represents the last level of the residual circuit of  $BK(N)$  circuit (i.e., part C in Figure 4.14). The second row represents the last level of the residual circuit of  $BK(N/2)$  circuit (i.e., the subpart C of part A in Figure 4.14). The relationship of each row in the table is as follow:

- The first entry of column 2 at row  $i$  is generated from the entry at row  $(i+1)$ , located at  $i$ 's diagonal in column 1 as one operation node having the same circuit depth as  $(i+1)$ 's entry (see the line |). For example, the entry  $(1, n-1)$  at row  $\lg N - 2$  is generated from the entry  $(N/2^n, n-1)$  at row  $\lg N - 1$ . Then this new output entry at row  $i$  produces two entries at row  $(i-1)$ : one operation node having the same circuit depth as  $i$ 's entry and one operation node having one more circuit depth than the  $i$ 's entry (see the arrow  $\uparrow$ ). For example, the entry  $(1, n-1)$  at row  $\lg N - 2$  produces the entry  $(1, n-1)$  and the entry  $(1, n)$  at row  $\lg N - 3$ .

Therefore, in column 2, the first row,  $K(N)$  (i.e., part C in Table 4.3), is derived from the second row,  $K(N/2)$  (see Figure 4.15) and can be written as follow.

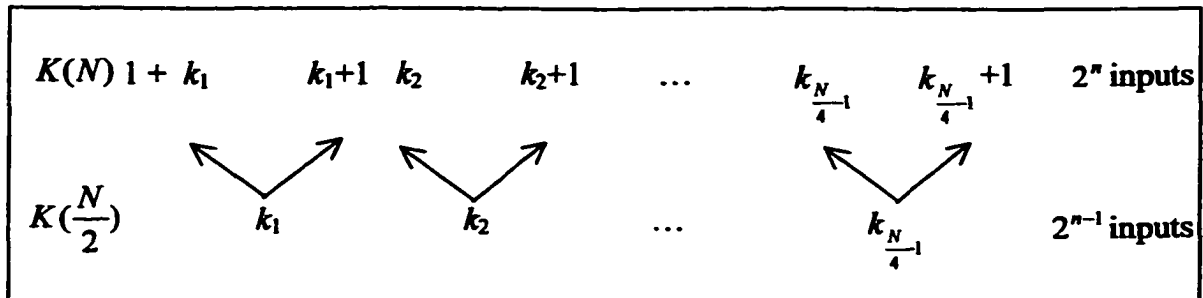


Figure 4.15: Part C, the distribution of  $N/2 - 1$  nodes.

$$\begin{aligned}
\text{Part } C &= 1 + k_1 + (k_1 + 1) + k_2 + (k_2 + 1) + \dots + k_{\frac{N}{4}-1} + (k_{\frac{N}{4}-1} + 1) \\
&= 1 + 2(k_1 + k_2 + \dots + k_{\frac{N}{4}-1}) + \left(\frac{N}{4} - 1\right) \\
&= 2K\left(\frac{N}{2}\right) + \frac{N}{4}
\end{aligned}$$

From Section 4.1.3, we know that

$$K(N) = \frac{3}{4}N \lg N - \frac{5N}{4} + 1, \text{ and}$$

$$Kcap_{\text{eff}}(N) = \left[1 + \frac{3}{2}N \lg N\right] - \frac{1}{2}[2N + (\lg N)^2 + \lg N].$$

Thus,  $Rcap_{\text{eff}}(N)$ , which is the residual circuit, can be written as follows:

$$\begin{aligned}
Rcap_{\text{eff}}(N) &= Kcap_{\text{eff}}\left(\frac{N}{2}\right) + 2K\left(\frac{N}{2}\right) + \frac{N}{4} \\
&= \left(1 + \frac{3}{2} \frac{N}{2} \lg \frac{N}{2}\right) - \frac{1}{2} \left(2 \frac{N}{2} + (\lg \frac{N}{2})^2 + \lg \frac{N}{2}\right) + 2 \left(\frac{3}{4} \frac{N}{2} \lg \frac{N}{2} - \frac{5}{4} \cdot \frac{N}{2} + 1\right) + \frac{N}{4} \\
&= (1 + 2) + \left(\frac{3}{2} + \frac{3}{2}\right) \frac{N}{2} \lg \frac{N}{2} - \frac{1}{2} \left((\lg \frac{N}{2})^2\right) - \left(\frac{1}{2} \lg \frac{N}{2}\right) - \left(\frac{5}{2} + 1\right) \frac{N}{2} + \frac{N}{4} \\
&= 3 + 3 \left(\frac{N}{2} \lg \frac{N}{2}\right) - \frac{1}{2} \left((\lg \frac{N}{2})^2\right) - \frac{1}{2} \left(\lg \frac{N}{2}\right) - \frac{3N}{2} \\
&= 3 \left(1 + \frac{N}{2} \lg \frac{N}{2}\right) - \frac{1}{2} \left(3N + (\lg \frac{N}{2})^2 + \lg \frac{N}{2}\right)
\end{aligned}$$

with gives the effective capacitance as:

$$\begin{aligned}
cap_{\text{eff}}(N) &= \left\{1 + \frac{3}{2}N \lg N - \frac{1}{2}[2N + (\lg N)^2 + \lg N]\right\} C_0 + \\
&\quad \left\{3 \left(1 + \frac{N}{2} \lg \frac{N}{2}\right) - \frac{1}{2} \left(3N + (\lg \frac{N}{2})^2 + \lg \frac{N}{2}\right)\right\} C'.
\end{aligned}$$

The effective circuit capacitance under linear output capacitance assumption of the Brent-Kung prefix circuit is  $O(N \lg N)$ .

#### 4.2.4 The Ladner-Fisher Parallel Prefix Circuit

As described in Section 4.1.4 that the effective capacitance of the family of the  $LF_k(N)$  prefix circuit is bounded by the divide-and-conquer prefix circuit and the Brent-Kung prefix circuit. Thus, the  $LF_k(N)$  circuit's effective capacitance is such that

$$\left\{ \left( \frac{N}{4} \right) (\lg N)^2 + \lg N \right\} C_0 + \left\{ \left( \frac{N}{4} \right) (\lg N)^2 - \lg N \right\} C' \leq \text{cap}_{\text{eff}}(N) \leq \left\{ 1 + \frac{3}{2} N \lg N - \frac{1}{2} [2N + (\lg N)^2 + \lg N] \right\} C_0 + \left\{ 3 \left( 1 + \frac{N}{2} \lg \frac{N}{2} \right) - \frac{1}{2} \left( 3N + (\lg \frac{N}{2})^2 + \lg \frac{N}{2} \right) \right\} C'.$$

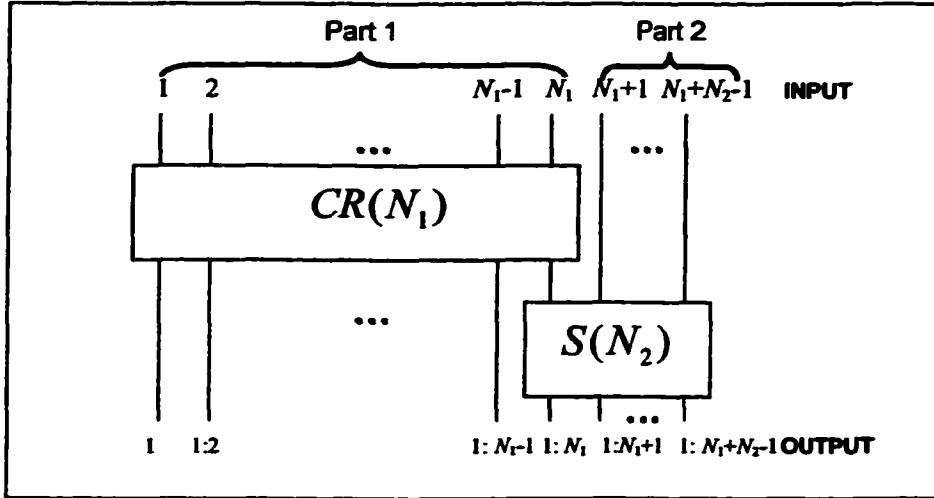
#### 4.2.5 The Snir Parallel Prefix Circuit and The Shih-Lin Parallel Prefix Circuit

As in Chapter 2, the  $SN(N)$  and  $SL(N)$  parallel prefix circuit are composed of two parts which are the compressed layered prefix circuit,  $CR(N_1)$ , and the serial prefix circuit  $S(N_2)$ , where  $N = N_1 + N_2 - 1$  (see Figure 4.16). Note that only the residual circuit is computed here. Therefore, the capacitance of the residual circuit is given by

$$R\text{cap}_{\text{eff}}(N) = R\text{cap}_{\text{eff}}(\text{Part1}) + R\text{cap}_{\text{eff}}(\text{Part2})$$

We can use the formula from the Brent-Kung parallel prefix circuit in Section 4.2.3 to compute  $R\text{cap}_{\text{eff}}(\text{Part1})$ . Thus,

$$R\text{cap}_{\text{eff}}(\text{Part1}) = 3 \left( 1 + \frac{N_1}{2} \lg \frac{N_1}{2} \right) - \frac{1}{2} \left( 3N_1 + (\lg \frac{N_1}{2})^2 + \lg \frac{N_1}{2} \right).$$



**Figure 4.16:** The  $SN(N)$ , and  $SL(N)$  prefix circuits.

The  $Rcap_{eff}(Part2)$  (i.e., the serial circuit) can be computed by starting at the level of the last output of Part 1 which is  $\lceil \lg N_1 \rceil$ . Since there are  $N_2 - 1$  operations, and the circuit depth is also  $N_2 - 1$ , the  $Rcap_{eff}(Part2)$  is given by:

$$\begin{aligned}
 Rcap_{eff}(Part2) &= \sum_{i=0}^{N_2-2} (\lceil \lg N_1 \rceil + i) \\
 &= \sum_{i=0}^{N_2-2} \lceil \lg N_1 \rceil + \sum_{i=0}^{N_2-2} i \\
 &= \lceil \lg N_1 \rceil (N_2 - 2 + 1) + \frac{(N_2 - 2)(N_2 - 1)}{2} \\
 &= \lceil \lg N_1 \rceil (N_2 - 1) + \frac{1}{2}(N_2^2 - 3N_2 + 2)
 \end{aligned}$$

The effective capacitance under linear output capacitance assumption is

$$cap_{eff}(N) = Kcap_{eff}(N) + Rcap_{eff}(N)$$

$$\begin{aligned}
 cap_{eff}(N) &= \left\{ \left[ 1 + \frac{3}{2} N_1 (\lg N_1) \right] - \left[ \frac{1}{2} [2N_1 + (\lg N_1)^2 + (\lg N_1)] \right] + \right. \\
 &\quad \left. \left[ N_2 \lceil \lg N_1 \rceil - \lceil \lg N_1 \rceil + \left( \frac{N_2^2 - N_2}{2} \right) \right] \right\} C_0 +
 \end{aligned}$$

$$\left\{ \left[ 3 \left( 1 + \frac{N_1}{2} \lg \frac{N_1}{2} \right) - \frac{1}{2} \left( 3N_1 + \left( \lg \frac{N_1}{2} \right)^2 + \lg \frac{N_1}{2} \right) \right] + \left[ \lg N_1 \left[ (N_2 - 1) + \frac{1}{2} (N_2^2 - 3N_2 + 2) \right] \right] \right\} C'$$

This implies that the  $SN(N)$  and  $SL(N)$  prefix circuit take  $O(N \lceil \lg N \rceil)$  to compute linear capacitance.

#### 4.2.6 The LYD Parallel Prefix Circuit

The LYD parallel prefix circuit,  $LYD(N)$ , is composed of four parts which are the layered prefix circuit,  $CR(N_1)$ ,  $Q(N_2)$ ,  $S(N_3)$ , and  $S(N_4)$ , where  $N = N_1 + N_2 + N_3 + N_4$  (see Figure 4.17). Therefore, the residual circuit is computed by summing the capacitance of the residual circuit from these four parts. The capacitance of Part 1 can be computed by using the formula from the Brent-Kung parallel prefix circuit described in Section 4.2.3. The capacitance of Parts 2, 3 and 4 can be computed by starting at the level of the last output of the first part, second part and the third part, respectively.

**Part 1** represents the circuit  $CR(N_1)$ . Thus the capacitance of Part 1 is equal to

$$3 \left( 1 + \frac{N_1}{2} \lg \frac{N_1}{2} \right) - \frac{1}{2} \left( 3N_1 + \left( \lg \frac{N_1}{2} \right)^2 + \lg \frac{N_1}{2} \right).$$

**Part 2** represents the circuit  $Q(N_2)$ . For level  $i = 1$  to  $t$ , level  $i$  has  $(t - i + 1)$  operation nodes.

$Rcap_{eff}(Part2) = Rcap_{eff}(A) + Rcap_{eff}(B) + Rcap_{eff}(ConnectionCircuit)$ , where

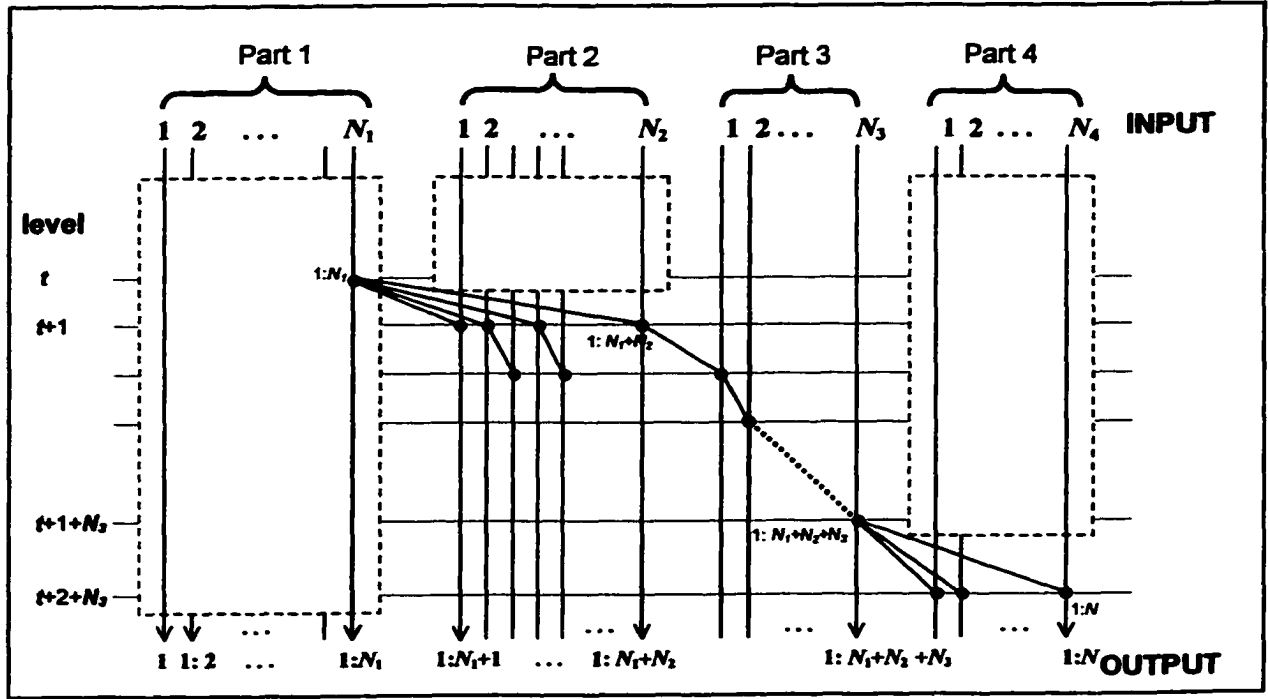


Figure 4.17: The structure of  $LYD(N)$ , derived from [LD94].

$$\begin{aligned}
 Rcap_{eff}(A) &= \sum_{i=2}^t (ti + 2i - i^2 - t - 1) \\
 &= t \sum_{i=2}^t i + 2 \sum_{i=2}^t i - \sum_{i=2}^t i^2 - \sum_{i=2}^t t - \sum_{i=2}^t 1 \\
 &= \frac{t \cdot t(t+1)}{2} + t(t+1) - \frac{t(t+1)(2t+1)}{6} - t^2 + t - t + 1 - t - 2 + 1 \\
 &= \frac{1}{6}(t^3 - t).
 \end{aligned}$$

$$Rcap_{eff}(B) = t \lceil \lg N_1 \rceil + \lceil \lg N_1 \rceil.$$

$$Rcap(\text{ConnectionCircuit}) = \frac{1}{2} [t^2 \lceil \lg N_1 \rceil + t^2 - t \lceil \lg N_1 \rceil - t].$$

Hence,

$$Rcap_{eff}(\text{Part2}) = \frac{t^3}{6} + \frac{1}{2} [t^2 + t^2 \lceil \lg N_1 \rceil + t \lceil \lg N_1 \rceil] + \lceil \lg N_1 \rceil - \frac{2}{3}t$$



Since  $t = \lceil \lg N_1 \rceil$ ,

$$Rcap_{eff}(Part2) = \frac{2}{3} \lceil \lg N_1 \rceil + \lceil \lg N_1 \rceil^2 + \frac{\lceil \lg N_1 \rceil}{3}$$

**Part 3** represents the circuit  $S(N_3)$ .  $Rcap_{eff}(Part3)$  is given by

$$\begin{aligned} Rcap_{eff}(Part3) &= \sum_{i=1}^{N_3} (\lceil \lg N_1 \rceil + i) \\ &= \sum_{i=1}^{N_3} \lceil \lg N_1 \rceil + \sum_{i=1}^{N_3} i \\ &= N_3 \lceil \lg N_1 \rceil + \frac{(N_3 + 1)N_3}{2} \\ &= \frac{N_3}{2} (2 \lceil \lg N_1 \rceil + N_3 + 1) \end{aligned}$$

**Part 4** represents the circuit  $S(N_4)$ .  $Rcap_{eff}(Part4)$  is given by

$$\begin{aligned} Rcap_{eff}(Part4) &= \sum_{i=1}^{N_4-2} i + N_4 (\lceil \lg N_1 \rceil + N_3 + 1) \\ &= \frac{(N_4 - 2)(N_4 - 1)}{2} + N_4 \lceil \lg N_1 \rceil + N_4 N_3 + N_4 \\ &= \frac{N_4^2}{2} - \frac{N_4}{2} + N_4 \lceil \lg N_1 \rceil + N_4 N_3 + 1 \end{aligned}$$

Thus the effective capacitance of  $LYD(N)$  circuit is given by

$$\begin{aligned} cap_{eff}(N) &= \left\{ \left[ 1 + \frac{3}{2} N_1 \lg N_1 \right] - \left[ \frac{1}{2} [2N_1 + (\lg N)^2 + \lg N_1] \right] + \right. \\ &\quad \left[ \frac{2 \lceil \lg N_1 \rceil^2}{3} + \lceil \lg N_1 \rceil^2 + \frac{4 \lceil \lg N_1 \rceil}{3} + 1 \right] + \\ &\quad \left. \left[ (N_3 + N_4) \left( \lceil \lg N_1 \rceil + \frac{3}{2} \right) + \frac{1}{2} (N_3^2 + N_4^2) + (N_3 \cdot N_4) \right] \right\} C_0 + \end{aligned}$$

$$\left\{ \left[ 3 \left( 1 + \frac{N_1}{2} \lg \frac{N_1}{2} \right) - \frac{1}{2} \left( 3N_1 + \left( \lg \frac{N_1}{2} \right)^2 + \lg \frac{N_1}{2} \right) \right] + \left[ \frac{2}{3} \lceil \lg N_1 \rceil + \lceil \lg N_1 \rceil + \frac{\lceil \lg N_1 \rceil}{3} \right] + \left[ \frac{N_3}{2} (2 \lceil \lg N_1 \rceil + N_3 + 1) + \frac{N_4^2}{2} + N_4 \lceil \lg N_1 \rceil + N_3 N_4 + 1 - \frac{N_4}{2} \right] \right\} C'$$

Again, the effective capacitance under linear output capacitance of the  $LYD(N)$  circuit is  $O(N \lceil \lg N \rceil)$ .

### 4.3. Comparison

Table 4.4 provides a comparison of the effective circuit capacitance of the different prefix circuits considered here. The serial prefix circuit has the largest effective circuit capacitance,  $O(N^2)$ . All parallel prefix circuits have  $O(N \lg N)$  effective circuit capacitance, except the divide-and-conquer prefix circuit and  $LF_0$  prefix circuit whose values are  $O(N(\lg N)^2)$ .

**Table 4.4:** The effective circuit capacitance of prefix circuits.

| Prefix Circuit     | $cap_{eff}(N)$   |
|--------------------|--|
| Serial             | $\left\{ \frac{N(N-1)}{2} \right\} C_0 + \left\{ \frac{(N-1)(N-2)}{2} \right\} C'$   |
| Divide-and-Conquer | $\left\{ \frac{N}{4} (\lg N)^2 + \lg N \right\} C_0 + \left\{ \frac{N}{4} (\lg N)^2 - \lg N \right\} C'$   |
| Brent-Kung         | $\left\{ 1 + \frac{3}{2} N \lg N - \frac{1}{2} [2N + (\lg N)^2 + \lg N] \right\} C_0 + \left\{ 3 \left( 1 + \frac{N}{2} \lg \frac{N}{2} \right) - \frac{1}{2} \left( 3N + (\lg \frac{N}{2})^2 + \lg \frac{N}{2} \right) \right\} C'$   |
| $LF_k$             | $\left\{ \frac{N}{4} (\lg N)^2 + \lg N \right\} C_0 + \left\{ \frac{N}{4} (\lg N)^2 - \lg N \right\} C' \leq LF_k \leq \left\{ 1 + \frac{3}{2} N \lg N - \frac{1}{2} [2N + (\lg N)^2 + \lg N] \right\} C_0 + \left\{ 3 \left( 1 + \frac{N}{2} \lg \frac{N}{2} \right) - \frac{1}{2} \left( 3N + (\lg \frac{N}{2})^2 + \lg \frac{N}{2} \right) \right\} C'$   |
| Snir               | $\left\{ \left[ 1 + \frac{3}{2} N_i (\lg N_i) \right] - \left[ \frac{1}{2} [2N_i + (\lg N_i)^2 + \lg N_i] \right] + \left[ N_i \lceil \lg N_i \rceil - \lceil \lg N_i \rceil + \left( \frac{N_i^2 - N_i}{2} \right) \right] \right\} C_0 + \left\{ \left[ 3 \left( 1 + \frac{N_i}{2} \lg \frac{N_i}{2} \right) - \frac{1}{2} \left( 3N_i + (\lg \frac{N_i}{2})^2 + \lg \frac{N_i}{2} \right) \right] + \left[ \lceil \lg N_i \rceil (N_i - 1) + \frac{1}{2} (N_i^2 - 3N_i + 2) \right] \right\} C'$  |
| Shih-Lin           | $\left\{ \left[ 1 + \frac{3}{2} N_i (\lg N_i) \right] - \left[ \frac{1}{2} [2N_i + (\lg N_i)^2 + \lg N_i] \right] + \left[ N_i \lceil \lg N_i \rceil - \lceil \lg N_i \rceil + \left( \frac{N_i^2 - N_i}{2} \right) \right] \right\} C_0 + \left\{ \left[ 3 \left( 1 + \frac{N_i}{2} \lg \frac{N_i}{2} \right) - \frac{1}{2} \left( 3N_i + (\lg \frac{N_i}{2})^2 + \lg \frac{N_i}{2} \right) \right] + \left[ \lceil \lg N_i \rceil (N_i - 1) + \frac{1}{2} (N_i^2 - 3N_i + 2) \right] \right\} C'$  |
| LYD                | $\left\{ \left[ 1 + \frac{3}{2} N_i \lceil \lg N_i \rceil \right] - \left[ \frac{1}{2} [2N_i + (\lg N_i)^2 + \lg N_i] \right] + \left[ \frac{2 \lceil \lg N_i \rceil}{3} + 2 \lceil \lg N_i \rceil + \frac{4 \lceil \lg N_i \rceil}{3} + 1 \right] + \left[ (N_i + N_i) \left( \lceil \lg N_i \rceil + \frac{3}{2} \right) + \frac{1}{2} (N_i^2 + N_i^2) + (N_i N_i) \right] \right\} C_0 + \left\{ \left[ 3 \left( 1 + \frac{N_i}{2} \lg \frac{N_i}{2} \right) - \frac{1}{2} \left( 3N_i + (\lg \frac{N_i}{2})^2 + \lg \frac{N_i}{2} \right) \right] + \left[ \frac{2}{3} \lceil \lg N_i \rceil + \lceil \lg N_i \rceil + \frac{\lceil \lg N_i \rceil}{3} \right] + \left[ \frac{N_i}{2} (2 \lceil \lg N_i \rceil + N_i + 1) + \frac{N_i^2}{2} + N_i \lceil \lg N_i \rceil + N_i N_i + 1 - \frac{N_i}{2} \right] \right\} C'$ |

## **CHAPTER 5**

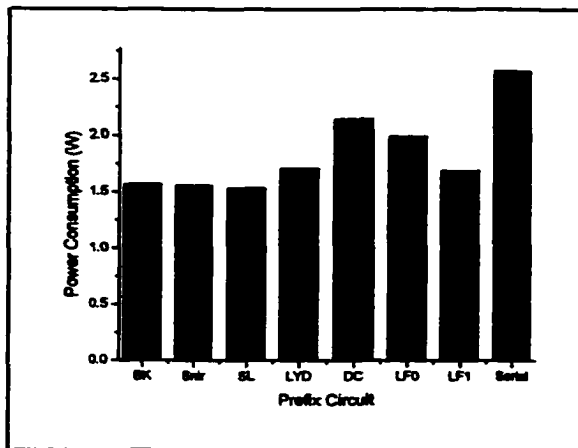
### **POWER-SPEED TRADE-OFF IN PREFIX CIRCUITS**

In Chapter 4, the power modeling for prefix circuits was proposed. One way to validate the model is to use simulation. This Chapter deals with the circuit simulations we conducted to investigate the prefix circuits' behavior to match with the prediction of our linear output capacitance assumption. These simulations allow the circuit designers to choose the best prefix circuit for a particular application. The degrees of freedom studied include different prefix circuit designs and voltage scaling. Voltage scaling is used because power consumption is a quadratic function of the voltage.

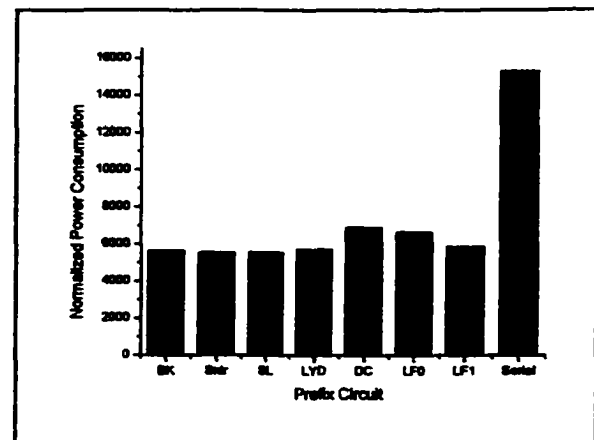
For purpose of investigating the linear output capacitance assumption, we implemented XOR gates under various prefix circuits at fixed supply voltage using PSpice. The power consumption of these circuits was measured and then compared with the estimated power consumption using the linear output capacitance assumption. After observing the behavior of power consumption of these prefix circuits, simulation was extended to study the effect of voltage reduction on power consumption. The 64-bit XOR gates implemented with different prefix circuits were simulated starting at power supply voltage 2.8 V and then scaling down to 1.4V. The range for the supply voltage is based on current technology and market trends [RCN01]. The possible decrease in power consumption under different circuit constraints has also been investigated to see which circuit will be more appropriate for a desired throughput.

## 5.1 Prefix Circuits at Fixed Voltage

In this section, we present simulation results for the 8-bit, 16-bit, 32-bit, and 64-bit XOR gates of seven prefix circuits ( $DC(N)$ ,  $BK(N)$ ,  $LF_0(N)$ ,  $LF_1(N)$ ,  $SN(N)$ ,  $SL(N)$ , and  $LYD(N)$ ). Simulations were first carried out using PSpice at a power supply voltage of 2.8V to investigate the effect of various prefix circuits on the circuit power consumption. The simulation results for 32-bit XOR with different prefix circuits using the worst case input for serial prefix circuit (i.e., the first input is equal to 0 and the other inputs are  $0 \rightarrow 1$ .), is shown in Figure 5.1. The result in Figure 5.2 is calculated from the formula  $P(normalized) = cap_{eff}(N)V_{DD}^2 f / (C' f)$ . Commensurate with our model analysis in Chapter 4, amongst all the prefix circuits, the serial prefix circuit consumes the maximum power due to the longest ripple (the maximum number of switching); power consumption is much larger compared to other circuits. Amongst the remaining circuits, results obtained from simulations (Figure 5.1) and the theoretical model (Figure 5.2) show that the divide-and-conquer prefix circuit consumes the most power, followed by the  $LF_0$  and the LYD prefix circuits.

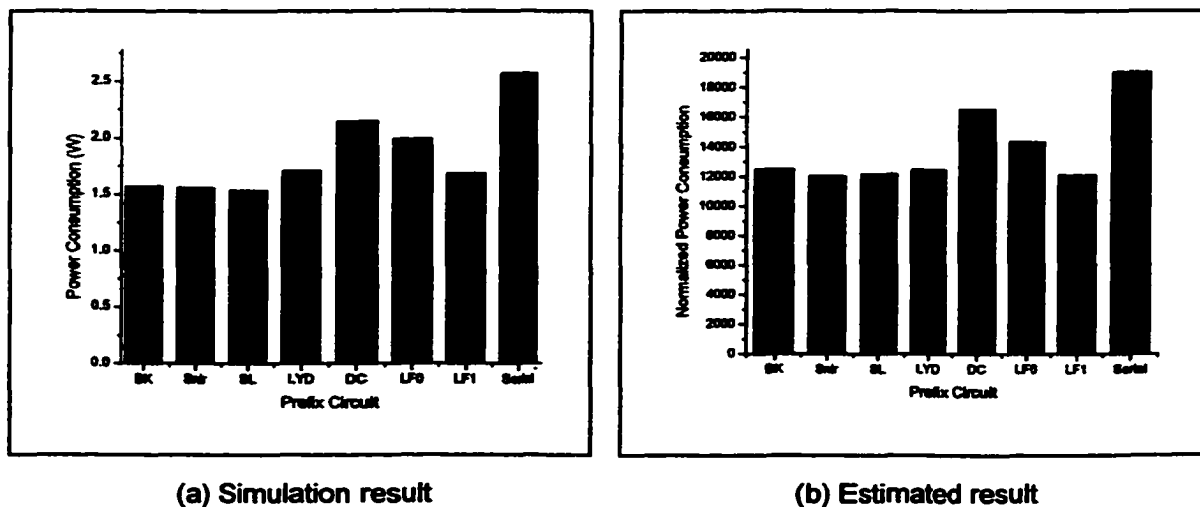


**Figure 5.1:** Power consumption of the 32-bit XOR parallel prefix circuits, obtained through PSpice simulation.



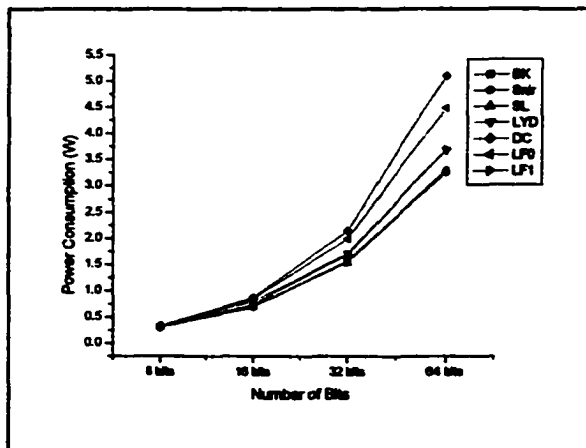
**Figure 5.2:** Estimated power consumption of prefix circuits when  $N = 32$  bits.

Comparing Figures 5.1 and 5.2, we find that our estimated values have the same distribution as the values obtained by simulation. There is, however, one discrepancy – the power consumption value of the serial circuit to the other parallel prefix circuits according to the model estimate is much greater than simulation results. This may be due to the fact that we did not consider static power consumption in our estimation, which depends on the gate technology and circuit size. The size of parallel prefix circuits is almost two times as large as that of the serial circuit. Thus, in simulation, the static power consumption component is more pronounced for parallel prefix circuits than for the serial circuit. This reduces the power consumption ratio between serial and parallel prefix circuits in simulation. Figures 5.3(a) and 5.3(b) plot the simulation result and a modified estimation by adding static power consumption component to the original estimation. We see that the simulation result in this case corroborate with the estimated values for parallel prefix circuits.

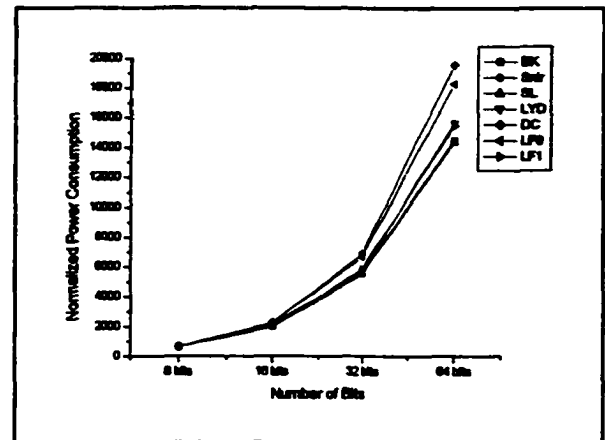


**Figure 5.3:** Comparison between simulation results and modified estimation results for  $N = 32$  bit. The modified estimation enhances the original estimation by including a component of power proportionally to circuit size.

The simulation and theoretical results for the 8-bit, 16-bit, 32-bit and 64-bit XOR parallel prefix circuits with different designs are shown in Figures 5.4 and 5.5, respectively. Amongst the parallel prefix circuits, the divide-and-conquer prefix circuit consumes the most power, followed by the  $LF_0$  prefix circuit and the LYD prefix circuit. The Shih-Lin and the Snir prefix circuits' power consumption is similar to the power consumption of the Brent-Kung prefix circuit. Comparing the simulation result (Figure 5.4) with the theoretical results (Figures 5.5), it is easily seen that the linear output capacitance assumption could be used reliably to predict power consumption of prefix circuits.



**Figure 5.4:** Power consumption of the XOR parallel prefix circuits at fixed voltage, obtained through Pspice simulation.



**Figure 5.5:** Estimated power consumption of parallel prefix circuits with fixed voltage.

## 5.2 Effects of Voltage Scaling on Prefix Circuits

To study the effect of voltage scaling on power consumption, while aiming at circuit design for reduction on delay, the following experiment was conducted. The 64-bit XOR gate under seven parallel prefix circuit implementations (divide-and-conquer, Brent-

Kung,  $LF_0$ ,  $LF_1$ , Snir, Shih-Lin, and LYD) introduced in Chapter 2, were carefully simulated to measure the power consumption and the circuit delay under supply voltage ranging from 2.8V to 1.4V [RCN01] to see the effect of speed on low power consumption under different circuit constraints.

Before presenting the results of simulation studies, an overview of the effects of scaling on supply voltage is given. As noted in Chapter 3, the average power consumption in a CMOS module can be written as follows:

$$P_{switching} = C_{eff} V_{DD}^2 f . \quad (5.1)$$

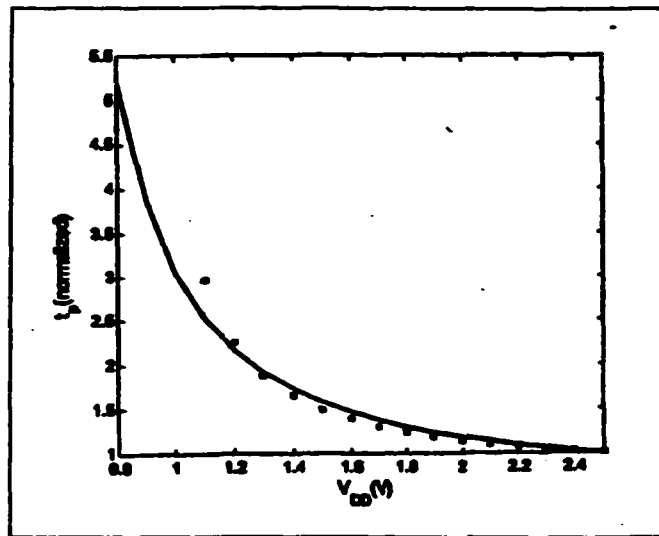


Figure 5.6: Plot of supply voltage vs. normalized delay [CB95].

This equation indicates that the most effective way to reduce power consumption is by operating the circuit at a lower  $V_{DD}$ , allowing a quadratic reduction in power. However, as seen from Figure 5.6, as  $V_{DD}$  decreases, the circuit delay generally increases. Hence, the system throughput reduces. The relationship between circuit delay,  $T_p$ , and supply



voltage,  $V_{DD}$ , is modeled as follow [Mac96]

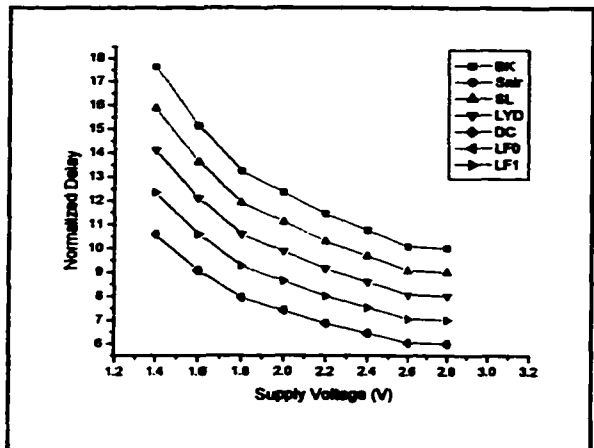
$$T_p = \frac{C_L V_{DD}}{k(W/L)(V_{DD} - V_t)^2} \quad (5.2)$$

where  $C_L$  is the gate capacitance,  $V_t$  is the threshold voltage,  $k$  is the technology-dependent parameter, and  $W$  and  $L$  are the channel width and length of the transistors, respectively. According to Eq. 5.2,  $T_p$  increases as  $V_{DD}$  approaches  $V_t$ . A sharp increase in delay can be observed if  $V_{DD} \leq 2V_t$  [RCN01].

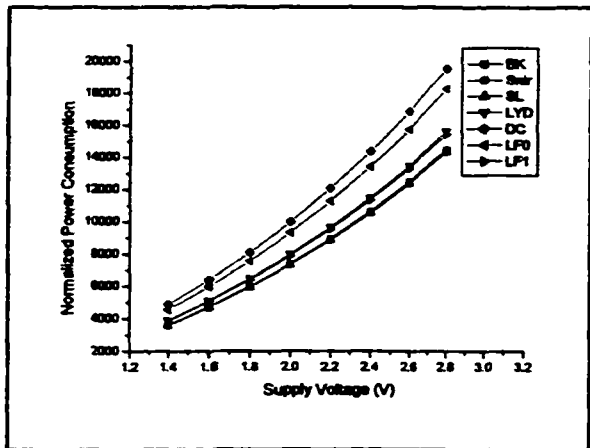
Thus where on one hand lowering power supply reduces the power consumption, on the other, it reduces the throughput. Looking closely at Eq. 5.1 and Eq. 5.2, we observe that though power consumption decreases quadratically with decrease in power supply, it only increases the time-delay inversely with the power reduction. Therefore, a commonly used technique to reduce power consumption without loss in throughput is to introduce parallelism [CB95]. Using parallelism will reduce the time-delay relative to the effective degree of parallelism. Hence we can use lower supply voltage proportionally (according to Figure 5.6) to maintain the same level of throughput with overall lower power consumption. Unfortunately, introduction of parallelism increases the number of computation nodes in many circuits, which, in turn, increases the power consumption. Because of the size-depth trade-off characteristic of the prefix circuits (Chapter 2), we can take advantage of parallelism only to the extent that parallelism reduces circuit depth.

### **Theoretical Results**

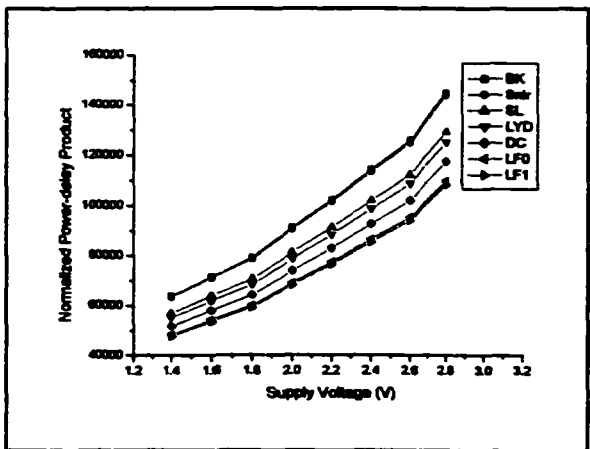
Figures 5.7, 5.8, and 5.9 give estimated delay, power consumption, and power-delay



**Figure 5.7:** Estimated delay of parallel prefix circuits when  $N=64$ .



**Figure 5.8:** Estimated power consumption of parallel prefix circuits when  $N=64$ .



**Figure 5.9:** Estimated power-delay product of parallel prefix circuits when  $N=64$ .

product for the 64-bit parallel prefix circuits obtained from the theoretical model. Figure 5.7 shows the result obtained by assuming the circuits' delay to be proportional to the circuits' depth and applying the normalized delay from Figure 5.6 in order to take the effect of the supply voltage on the delay. The estimated power consumptions for the circuit considered are shown in Figure 5.8. The divide-and-conquer circuit that has the shortest depth and largest size consumes the maximum power. The Brent-Kung prefix circuit has the highest power-delay product while the divide-and-conquer prefix circuit and the  $LF_0$  prefix circuits have the power-delay product lower than that of the Brent-Kung prefix circuit, the Snir prefix circuit, the Shih-Lin prefix circuit and the LYD prefix circuit.

Table 5.1 shows the estimated power consumption of the different prefix circuits at fixed and reduced supply voltage when  $N = 64$ . The power is estimated using the formula of Eq. 4.1,  $P = cap_{eff}(N)V_{DD}^2f$ , where  $cap_{eff}(N)$  is the effective circuit capacitance. For this study we used  $C_0 = 0.9$  and  $C' = 0.3$  [Smi97]. When the supply voltage is fixed at 2.8V, the serial prefix circuit consumes more power than any other circuit.

To lower power consumption by reducing the supply voltage, let us assume a fixed acceptable delay. Further, assume that time-delay is proportional to depth and that a delay proportional to a depth of 10 with  $V_{DD} = 2.8$  volts is acceptable. Thus the voltage of the Brent-Kung and Snir circuits cannot be lowered, and the delay of the serial circuits is not acceptable. The supply voltages of the other five prefix circuits can be dropped from 2.8V and still achieve the acceptable delay. For example, because the delay for the

divide-and-conquer prefix circuit is proportional to 6 at 2.8V, the voltage can be dropped from 2.8V to 1.48V to obtain a time-delay proportional to a depth of 10. The operating frequency can be decreased by a factor of 0.6. Thus the normalized power consumption of the divide-and-conquer prefix circuit is:

$$P(\text{normalized}) = \text{cap}_{\text{eff}}(N)V_{DD}^2 f = (2,496C')(1.48)^2(0.6f)/(C'f) \approx 3,280.$$

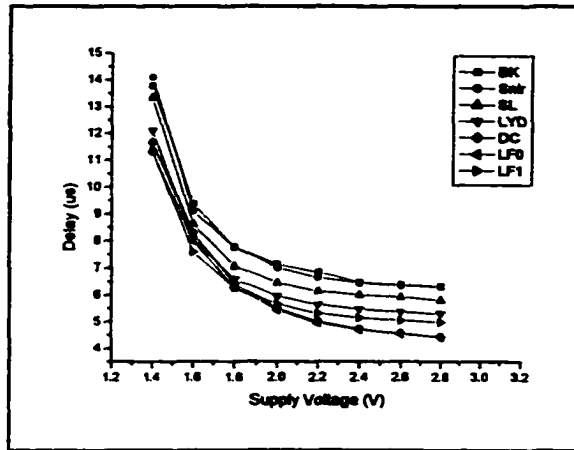
**Table 5.1:** Estimated power consumption based on Eq. 4.1 for various prefix circuits for  $N = 64$ .

| Prefix Circuit     | Depth | $\text{cap}_{\text{eff}}(64)$ | Power<br>(normalized)<br>$V_{DD} = 2.8V$ | NewPower<br>(normalized)<br>after reducing $V_{DD}$ |
|--------------------|-------|-------------------------------|--|---|
| Serial             | 63    | $2016C_0 + 1953C'$            | 62,728                                   | -   |
| Divide-and-Conquer | 6     | $672C_0 + 480C'$              | 19,569                                   | 3,280<br>$V_{DD} = 1.48V$                           |
| Brent-Kung         | 10    | $492C_0 + 372C'$              | 14,488                                   | 14,488<br>$V_{DD} = 2.6V$                           |
| $LF_0$             | 6     | $625C_0 + 457C'$              | 18,283                                   | 3,065<br>$V_{DD} = 1.48V$                           |
| $LF_1$             | 7     | $527C_0 + 390C'$              | 15,453                                   | 3,987<br>$V_{DD} = 1.7V$                            |
| Snir               | 10    | $487C_0 + 371C'$              | 14,363                                   | 14,363<br>$V_{DD} = 2.6V$                           |
| Shih-Lin           | 9     | $487C_0 + 370C'$              | 14,355                                   | 9,491<br>$V_{DD} = 2.4V$                            |
| LYD                | 8     | $528C_0 + 410C'$              | 15,633                                   | 6,381<br>$V_{DD} = 2V$                              |

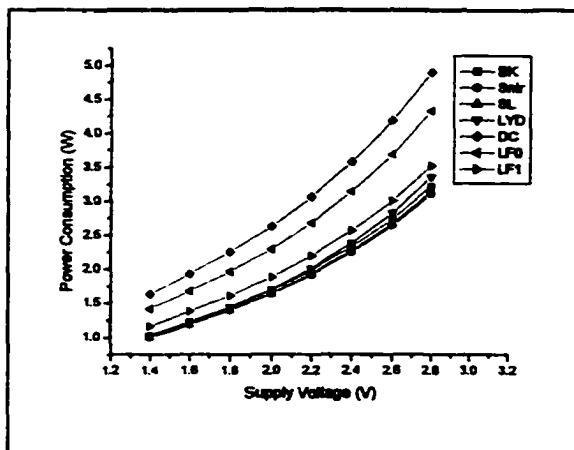
After scaling the supply voltage, there is a power improvement in the circuits having depth shorter than 10. Among these circuits, the  $LF_0$  prefix circuit has a major reduction in power due to its shortest depth.

### Simulation Results

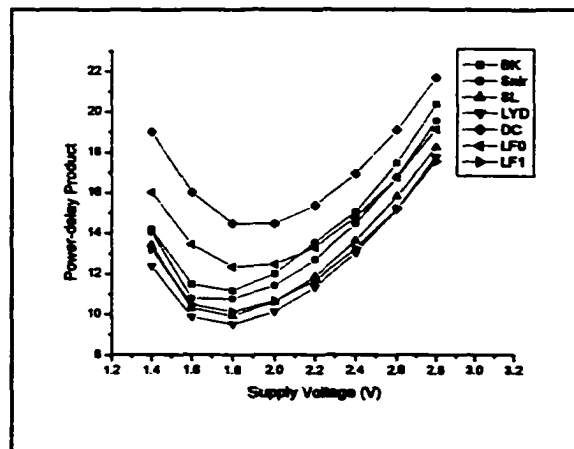
Figures 5.10, 5.11, and 5.12 give delay, power consumption, and power-delay product for



**Figure 5.10:** Delay of the 64-bit XOR parallel prefix circuits, obtained through PSpice simulation.



**Figure 5.11:** Power consumption of the 64-bit XOR parallel prefix circuits, obtained through PSpice simulation.



**Figure 5.12:** Power-delay product of the 64-bit XOR parallel prefix circuits, obtained through PSpice simulation.

the 64-bit XOR parallel prefix circuits obtained through PSpice simulation over random inputs. As expected, amongst the parallel prefix circuits considered, the divide-and-conquer prefix circuit consumes the most power. Also, though the delay of the divide-and-conquer prefix circuit is the least for some values of the voltage supply, it is not so for lower voltages. This may be due to its very high fan-out compared to others ( $O(N)$  vs.  $O(\lg N)$ ). As the supply voltage is reduced, power consumption is also reduced. Comparing the model predictions (Figures 5.8) to simulation result (Figures 5.11), it was found that the use of the linear output capacitance assumption gives similar results as PSpice simulation.

From the point of view of the power-delay product metric, the LYD prefix circuit is found to be the best across the entire voltage scaling. This means that the circuit provides the best trade-off between power and delay. Another result of simulation studies shows that the power-delay product of the divide-and-conquer prefix circuit is the highest, followed by that of the  $LF_0$  prefix circuit. This is at variance with our model prediction and may be due to the fact that these circuits have a very high fan-out (see Table 2.1 for fan-out). In our model, we do not take into account the effect of fan-out on the delay.

Also according to the simulation, with voltage-scaling technique, the LYD prefix circuit has the least power consumption compared to other circuits. For example, let us assume the maximum acceptable delay is  $6.4 \mu\text{s}$ . From Figures 5.10 and 5.11, to achieve this time-delay, the supply voltage of the divide-and-conquer,  $LF_0$ ,  $LF_1$ , Shih-Lin, and LYD prefix circuits can be 1.8V, 1.78V, 1.78V, 2V, and 1.8V, respectively. Therefore, the power consumption of the divide-and-conquer,  $LF_0$ ,  $LF_1$ , Shih-Lin, and LYD prefix circuits is 2.25, 1.94, 1.59, 1.64, and 1.44 W, respectively. This shows that power

reduction of about 1.6 times can be obtained without speed loss by using the LYD prefix circuit compared with using the divide-and-conquer prefix circuit by using appropriately chosen supply voltage.

### **5.3 Summary**

This chapter presented a comparative study of different parallel prefix circuits from the point of view of power-speed trade-off. The power consumption and the power-delay product of seven parallel prefix circuits were compared. We have shown that the use of the linear output capacitance assumption provides results that are consistent with those obtained by using PSpice simulation. The model enables us to understand the power consumption behavior of prefix circuits, and to pick the suitable prefix circuit for the acceptable power consumption and/or time-delay. We have also shown that parallelism at a certain level coupled with the use of low supply voltage can be used to reduce the power consumption in the prefix circuit without throughput loss. Our analysis, combined with PSpice simulations, shows that amongst the parallel prefix circuits the divide-and-conquer prefix circuit consumes the most power in spite of having the shortest depth and the highest parallelism. Also according to PSpice simulation, the trade-off between power and delay of the LYD prefix circuit seems to be the best of all the circuits considered.

The main discrepancy between the model and the simulation result is the power-delay product metric. This may be due to the fact that the fan-out of the divide-and-conquer prefix circuit and the  $LF_0$  prefix circuit is very high as compared to other prefix circuits. In this analysis, we have assumed that the delay is uniquely determined by the depth of the circuit. The results of the simulation of the divide-and-conquer prefix circuit in

particular indicate that large fan-out in addition to contributing to larger power consumption may also indirectly affect the time-delay. Modeling this interaction between high fan-out and time-delay is an interesting problem.



## **CHAPTER 6**

### **ADDITION CIRCUITS**

In this chapter we study the application of prefix circuits in adders and look at the power consumption characteristics when different prefix circuits are used. An addition of two binary numbers is of great interest to digital designers since it is the most commonly used operation in many other operations (e.g., counting, multiplication, division, etc.). Many researchers have investigated the various implementations of adder circuits. Examples are [BD01, BL01, FB01, FL00, Lin81]. For a general introduction refer to [HP90, Hwa79, Kor93, Omo94]. There are a number of ways of formulating the process of binary addition. Each way provides different insight and thus suggests different implementation. Although each implementation is available to serve different requirements, the focus of various implementations is on the calculation of all carry bits quickly, since the key to fast addition is the fast calculation of all the carries. In one of these implementations, the addition of two binary numbers is expressed as a prefix problem by transforming the computation of all carry bits to prefix computation. The adder using this technique is called a prefix adder.

Prefix adder has been addressed in various papers. For example, [JS95] investigated 32-bit Ladner-Fischer prefix adder on speed and area. [AD98] introduced a new irregular parallel prefix adder. [BL01, ZS01] introduce an algorithm to construct low area-delay product parallel prefix adders. A similar study [Kno01] explored space used and speed of

two parallel prefix adder designs. Previous work on low power adder can be found in many studies. By using the same CMOS technology, [Cal96, NIO96] compared power consumption of various adder architectures (i.e., ripple carry adder, carry look-ahead adder (CLA), etc.). [KBL95] studied different technologies (i.e., full static CMOS, complementary pass-transistor logic, double pass-transistor logic, etc.) with a given adder architecture for low-power adder design. Besides considering different technologies and adder architectures, another approach employs transistor sizing for a low power full adder design [BAW00, Rad01]. Although we will measure power consumption of the various implementations of adders, our objective is different from theirs. In our study, we concentrate on investigating and comparing power consumption of prefix adder based on Brent's algorithm [Bre70], using some of the prefix circuits from Chapter 2, with different sizes. Brent's algorithm transforms the carry computations to a prefix problem and hence is an ideal candidate for studying prefix circuits.

In the following, the basics of an adder are given in detail. Then, the method of implementing a fast parallel prefix addition based on the Brent's algorithm [Bre70, LD90] is explored and details of how the computation of all carry bits is transformed into the prefix computation are given.

## 6.1 Adder: Theory

A circuit for adding two binary digits and a carry-bit is called a *full-adder* (FA). Figure 6.1 shows the block diagram of the full-adder. The FA adder circuit takes  $x$ ,  $y$ , and  $z$  as inputs and produces  $s$  and  $c$  as outputs. Table 6.1 is the truth table for the full adder. When all input bits are 0, both the outputs are 0. When an odd number of inputs equals 1,

the  $s$  output will be 1. The  $c$  output has a value of 1 if two or three inputs equal 1. Following is a possible set of algebraic expressions for the two output variables derived from the K-map (Figure 6.2)

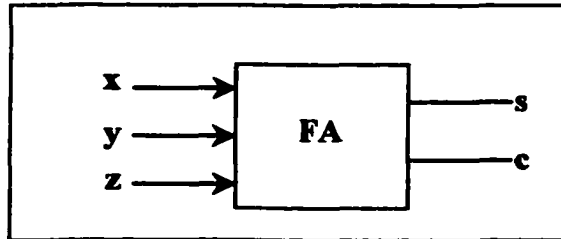


Figure 6.1: The Block diagram of the full-adder circuit.

Table 6.1: Adder truth table.

| Inputs |   |   | Outputs |   |
|--------|---|---|---------|---|
| x      | y | z | c       | s |
| 0      | 0 | 0 | 0       | 0 |
| 0      | 0 | 1 | 0       | 1 |
| 0      | 1 | 0 | 0       | 1 |
| 0      | 1 | 1 | 1       | 0 |
| 1      | 0 | 0 | 0       | 1 |
| 1      | 0 | 1 | 1       | 0 |
| 1      | 1 | 0 | 1       | 0 |
| 1      | 1 | 1 | 1       | 1 |

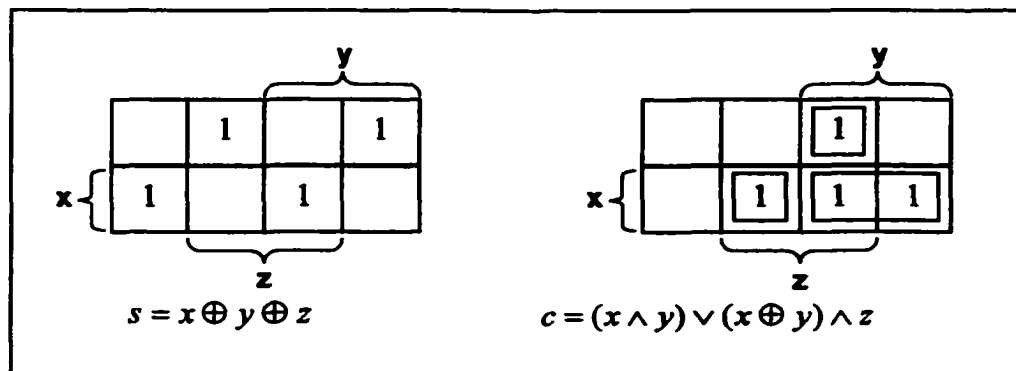


Figure 6.2: The K-Maps for the full-adder circuit.

$$\left. \begin{aligned} s &= x \oplus y \oplus z \\ c &= (x \wedge y) \vee (x \oplus y) \wedge z \end{aligned} \right\} (6.1.1)$$

where  $\oplus$ ,  $\vee$ , and  $\wedge$  are logical exclusive-OR, (inclusive) OR, and AND operations, respectively. Note that the output  $c$  can be expressed in any one of the following forms:

- $c = (x \wedge y) \vee (x \vee b) \wedge z$ ,
- $c = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$ , and
- $c = (x \vee y) \wedge ((x \wedge y) \vee z)$ .

One addition circuit of  $N$ -bit integers is the chain of  $N$  full-adders as shown in Figure 6.3. Let  $a = a_N a_{N-1} \dots a_2 a_1$ ,  $b = b_N b_{N-1} \dots b_2 b_1$ , and  $s = s_N \dots s_2 s_1$  be  $N$ -bit integers. Let  $s$  be the sum of  $a$  and  $b$ . We sum the binary bits from right to left, propagating any carry from  $FA_i$  to  $FA_{i+1}$ , for  $1 \leq i \leq N$  (see Figure 6.3). In the  $i^{\text{th}}$  FA, we take as inputs bits  $a_i$  and  $b_i$  and the carry-in bit  $c_{i-1}$  to produce the sum bit  $s_i$  and the carry-out bit  $c_i$ . The carry-out bit  $c_i$  from the  $i^{\text{th}}$  FA is the carry-in bit into the  $(i+1)^{\text{th}}$  FA. Since there is no carry-in for position 0, we assume that  $c_0 = 0$ . The carry-out  $c_N$  is the sum bit  $s_{N+1}$ . Therefore, in general, for  $1 \leq i \leq N$ ,

$$\left. \begin{aligned} s_i &= a_i \oplus b_i \oplus c_{i-1} \\ c_i &= (a_i \wedge b_i) \vee (a_i \oplus b_i) \wedge c_{i-1} \end{aligned} \right\} (6.1.2)$$

where  $c_0 = 0$ . This circuit is called the ripple-carry adder. Each full-adder takes three stages and five logical elements (Figure 6.4). For the chain of  $N$  full-adders,  $5N - 3$  logic elements and  $2N - 1$  stages are needed to compute the output  $s$ . Hence, the circuit has  $O(N)$  size and  $O(N)$  time. As seen in (6.1.2), to compute the sum bits,  $s_i$  ( $1 \leq i \leq N$ ) in parallel, we need the carry bit  $c_i$  ( $1 \leq i < N$ ). Therefore, the faster the carry bit  $c_i$  is

known, the faster the addition circuit is. In other words, the key to parallel addition is parallelizing the computation of all the carry bits.

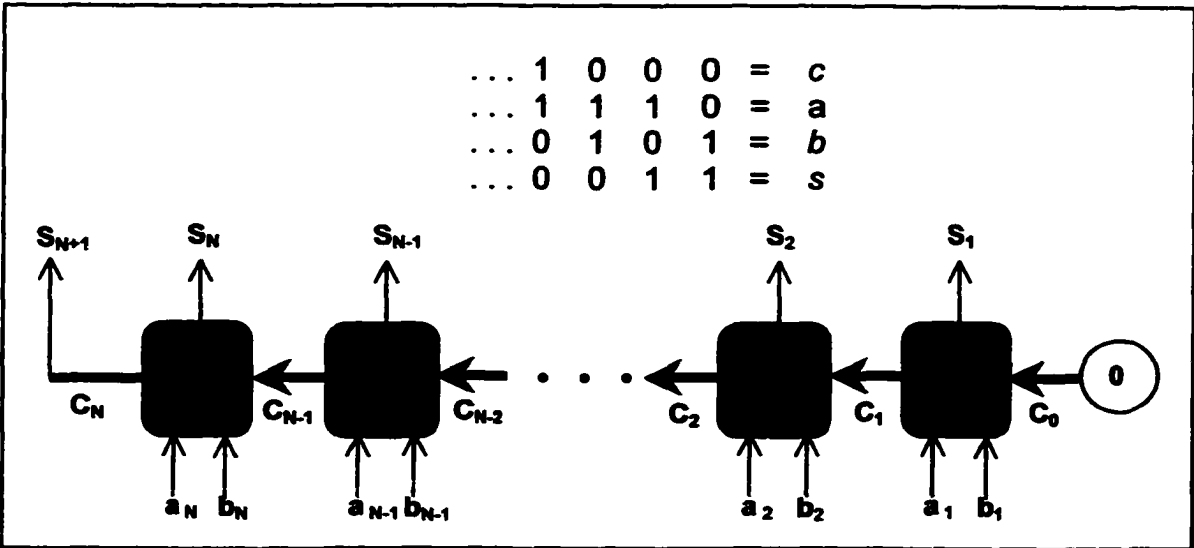


Figure 6.3: A chain of  $N$  full-adders.

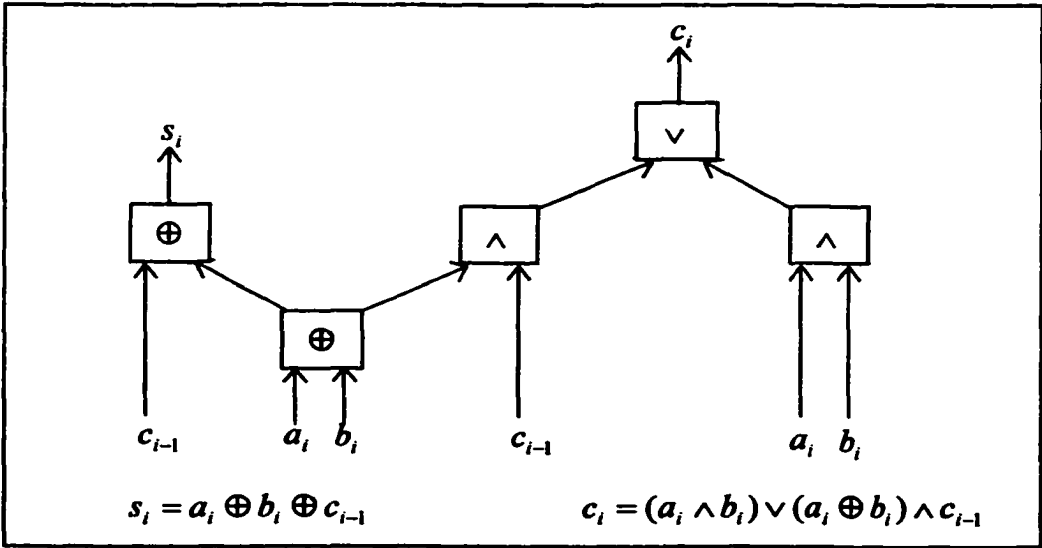


Figure 6.4: A full-adder circuit.

## 6.2 Parallel Addition

Let  $a = a_N a_{N-1} \dots a_2 a_1$ , and  $b = b_N b_{N-1} \dots b_2 b_1$  be two integers to be added. Let  $s = (a + b) \bmod 2^N$ , where  $s = s_N \dots s_2 s_1$ . Therefore,

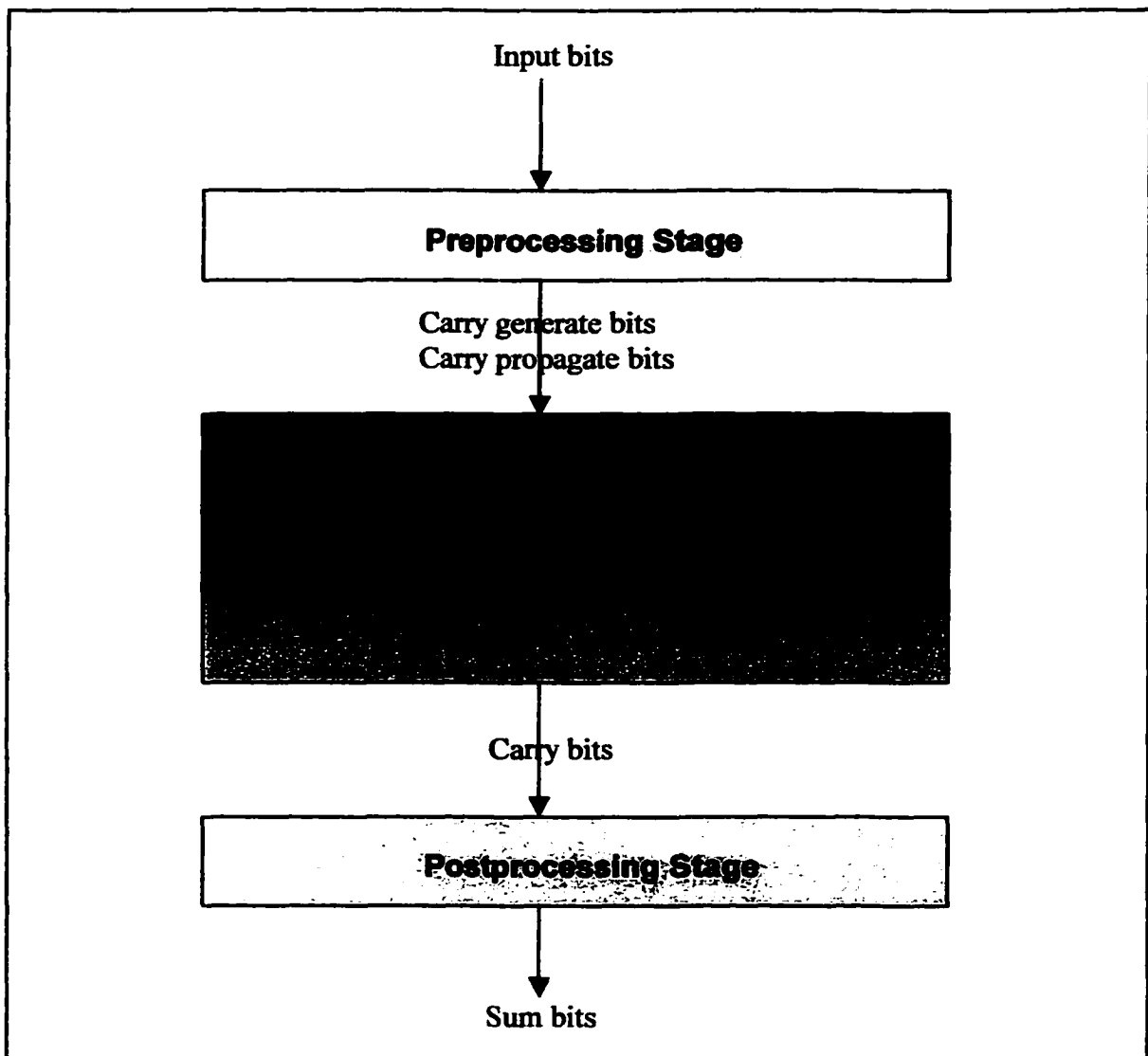
$$\left. \begin{array}{l} \text{where} \\ s_i = a_i \oplus b_i \oplus c_{i-1} \\ c_0 = 0, \\ c_i = p_i \wedge (g_i \vee c_{i-1}), \\ p_i = a_i \vee b_i, \\ \text{and} \\ g_i = a_i \wedge b_i, \end{array} \right\} (6.2.1)$$

for  $(1 \leq i \leq N)$ . The carry bit  $c_i$  is the carry from the  $i^{\text{th}}$  bit position,  $p_i$  is a *carry propagate* condition, and  $g_i$  is a *carry generate* condition. As discussed before, we need the carry bit  $c_{i-1}$  ( $1 \leq i \leq N$ ) for computing the sum bit  $s_i$ . By distributing the propagate bit  $p_i$  and the generate bit  $g_i$  to  $c_i = p_i \wedge (g_i \vee c_{i-1})$  in (6.2.1), we obtain

$$\left. \begin{array}{l} c_1 = p_1 \wedge g_1 \\ c_2 = p_2 \wedge (g_2 \vee (p_1 \wedge g_1)) \\ \vdots \\ c_i = p_i \wedge (g_i \vee (p_{i-1} \wedge (g_{i-1} \vee \dots \vee (p_1 \wedge g_1) \dots))). \end{array} \right\} (6.2.2)$$

The implementation of the fast addition is carried out in three stages: the preprocessing stage, the carry computation stage and the postprocessing stage (see Figure 6.5). The preprocessing stage computes the carry propagate bit  $p_i$  and the carry generate bit  $g_i$  in parallel in just one unit step (that is  $p_i = a_i \vee b_i$  and  $g_i = a_i \wedge b_i$ ; for  $1 \leq i \leq N$ ). In the carry computation stage, the calculation of all carry bits is converted into the prefix circuit problem, which is discussed later in this section. The inputs of the prefix circuit

for carry calculation are the carry propagate bits and the carry generate bits from the preprocessing stage. Once all the carry bits are known, the postprocessing stage produces the sum bits in two steps (that is  $s_i = a_i \oplus b_i \oplus c_{i-1}$ ; for  $1 \leq i \leq N$ ). The time in preprocessing and postprocessing stages is negligible compared to the computation time of the carry. As a result, computing all carry bits quickly is the key to high-speed addition. Therefore, we will concentrate on the carry computation for the rest of this chapter.



**Figure 6.5:** Three stages of the implementation of the fast adder.

### 6.2.1 Binary Addition as a Prefix Problem

Brent [Bre70] has presented the upper bound on the computation of the carry for the parallel addition of two  $N$ -bit integers. The following discussions are derived from [Bre70, LD90]. Let  $T_A(N)$  be the time required to add two  $N$ -bit binary numbers. Then from the above discussion

$$T_A(N) = T_C(N-1) + 3.$$

### Carry Computation

A schematic diagram of the Brent's algorithm for computing  $c_N$  is given in Figure 6.7.

To expedite the carry computation, Brent uses the following strategy to compute  $c_N$ :

- Compute all  $p_i$  ( $p_i = a_i \vee b_i$ ) and  $g_i$  ( $g_i = a_i \wedge b_i$ ).
- Partition all  $p_i$  and  $g_i$  into  $r$  groups; each group has  $q$  members, where  $N = rq$  (see Figure 6.6).

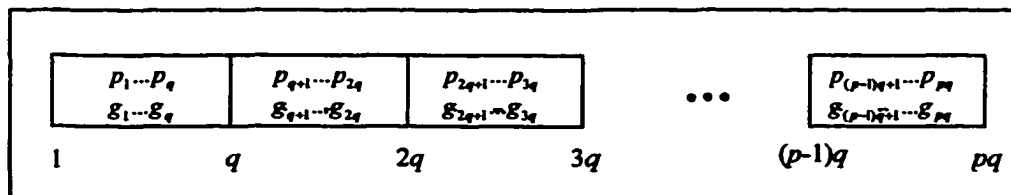


Figure 6.6: The partition of all  $p_i$  and  $g_i$  into  $r$  groups with  $q$  members each.

- Brent's algorithm to express the carry calculation  $c_N$  is as follow.

Let  $r \geq 1$  and  $q \geq 1$  be integers such that  $N = rq$ .

Let 
$$P_i = p_{iq} \wedge \dots \wedge p_{(i-1)q+1},$$



$$D_i = P_r \wedge \dots \wedge P_{i+1}, \quad D_r = 1,$$

$$E_i = p_{iq} \wedge (g_{iq} \vee \dots (p_{(i-1)q+1} \wedge g_{(i-1)q+1}) \dots),$$

and

$$F_i = D_i \wedge E_i,$$

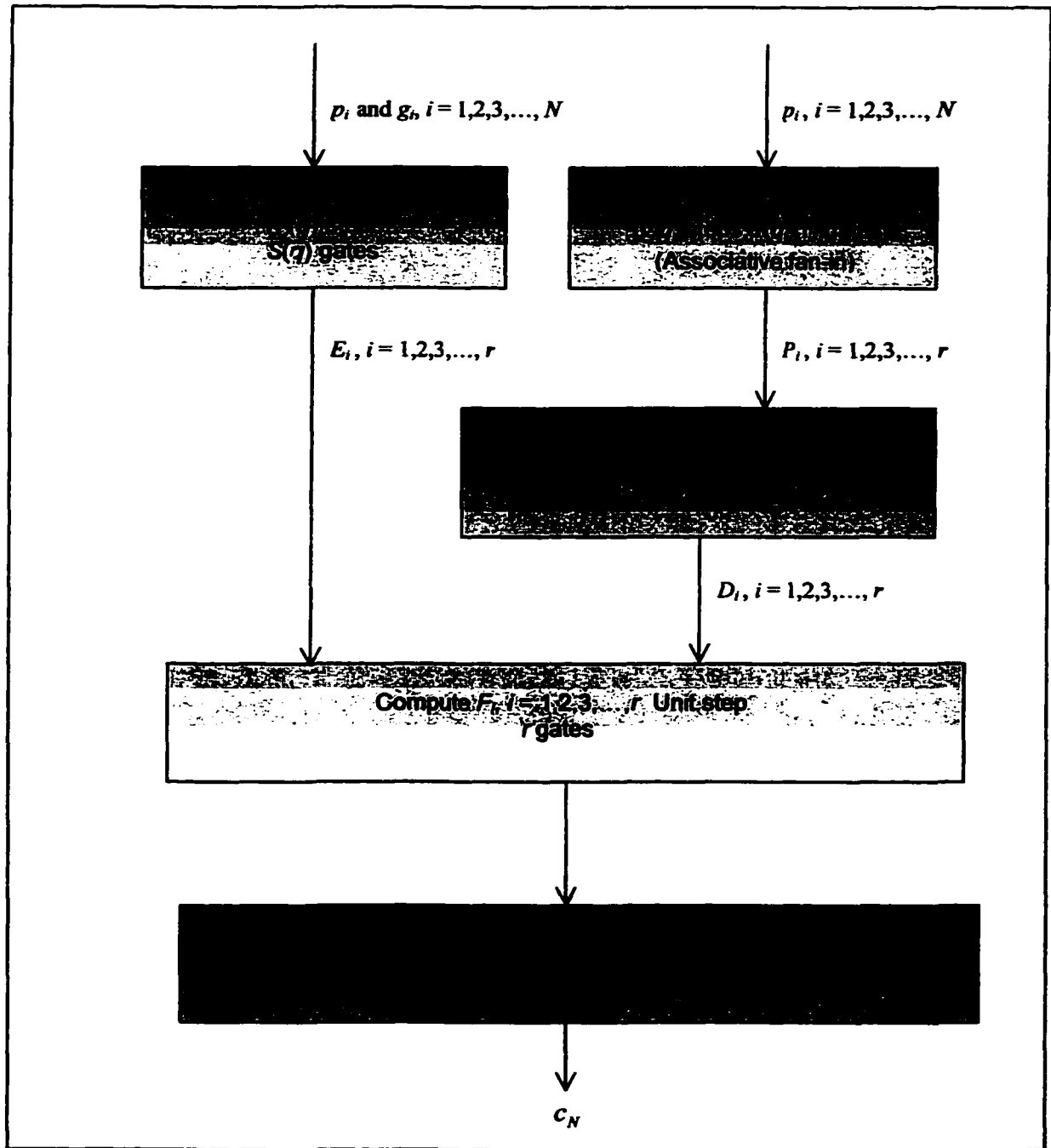


Figure 6.7: A parallel scheme for computing the carry, derived from [LD90].

then, by associativity, commutativity, and distributivity,

$$c_N = F_r \vee F_{r-1} \vee \dots \vee F_2 \vee F_1.$$

During the time the value of  $c_N$  is obtained, all other  $c_i$ 's are also obtained. Note that  $P_i$  is the product of carry propagates of group  $i$ ,  $D_i$  is prefix circuit (i.e.,  $P_{i+1} : P_r$ ), and  $E_i$  is the carry out of block  $i$  ( $1 \leq i \leq r$ ). The upper bound on the computation time of the last carry bit,  $T_c(N)$ , is given by

$$T_c(N) \leq 1 + \lceil \lg r \rceil + \max\{T_c(q), \lceil \lg q \rceil + \text{PrefixComputationTime}\}.$$

As can be seen from Figure 6.7, the running time of the algorithm depends on the number of blocks, the block size, and the choice of the prefix circuit. The detailed proof is given in [LD90].

The following example illustrates these quantities. Let  $N = 8$ . There are four possible choices to partition all  $p_i$  and  $g_i$ :

- $r = 1$  and  $q = 8$
- $r = 2$  and  $q = 4$
- $r = 4$  and  $q = 2$
- $r = 8$  and  $q = 1$

**Case 1:**  $r = 1$  and  $q = 8$ .

Then,

$$P_1 = p_8 \wedge p_7 \wedge p_6 \wedge p_5 \wedge p_4 \wedge p_3 \wedge p_2 \wedge p_1$$

$$D_1 = 1$$

$$E_1 = p_8 \wedge (g_8 \vee (p_7 \wedge (g_7 \vee (p_6 \wedge (g_6 \vee (p_5 \wedge (g_5 \vee (p_4 \wedge (g_4 \vee (p_3 \wedge (g_3 \vee (p_2 \wedge (g_2 \vee (p_1 \wedge g_1))...))$$

$$F_1 = D_1 \wedge E_1$$

$$= p_8 \wedge (g_8 \vee (p_7 \wedge (g_7 \vee (p_6 \wedge (g_6 \vee (p_5 \wedge (g_5 \vee (p_4 \wedge (g_4 \vee (p_3 \wedge (g_3 \vee (p_2 \wedge (g_2 \vee (p_1 \wedge g_1))...))$$

Then,

$$c_8 = F_1 = E_1$$

All other  $c_i$ , for  $1 \leq i \leq 7$ , are also available when  $c_8$  is completed.

$$c_7 = p_7 \wedge (g_7 \vee (p_6 \wedge (g_6 \vee (p_5 \wedge (g_5 \vee (p_4 \wedge (g_4 \vee (p_3 \wedge (g_3 \vee (p_2 \wedge (g_2 \vee (p_1 \wedge g_1))...))$$

$$c_6 = p_6 \wedge (g_6 \vee (p_5 \wedge (g_5 \vee (p_4 \wedge (g_4 \vee (p_3 \wedge (g_3 \vee (p_2 \wedge (g_2 \vee (p_1 \wedge g_1))...))$$

$$c_5 = p_5 \wedge (g_5 \vee (p_4 \wedge (g_4 \vee (p_3 \wedge (g_3 \vee (p_2 \wedge (g_2 \vee (p_1 \wedge g_1))...))$$

$$c_4 = p_4 \wedge (g_4 \vee (p_3 \wedge (g_3 \vee (p_2 \wedge (g_2 \vee (p_1 \wedge g_1))...))$$

$$c_3 = p_3 \wedge (g_3 \vee (p_2 \wedge (g_2 \vee (p_1 \wedge g_1))))$$

$$c_2 = p_2 \wedge (g_2 \vee (p_1 \wedge g_1))$$

$$c_1 = p_1 \wedge g_1$$

Note that Case 1 is just a serial computation of  $c_1, c_2, \dots$ , and  $c_8$ .

**Case 2:**  $r = 2$  and  $q = 4$ .

Then,

$$P_1 = p_4 \wedge p_3 \wedge p_2 \wedge p_1, \quad P_2 = p_8 \wedge p_7 \wedge p_6 \wedge p_5$$

$$D_1 = P_2, \quad D_2 = 1$$

$$E_1 = p_4 \wedge (g_4 \vee (p_3 \wedge (g_3 \vee (p_2 \wedge (g_2 \vee (p_1 \wedge g_1)))))$$

$$E_2 = p_8 \wedge (g_8 \vee (p_7 \wedge (g_7 \vee (p_6 \wedge (g_6 \vee (p_5 \wedge g_5)))))$$

and

$$F_1 = D_1 \wedge E_1 = [P_2] \wedge E_1$$

$$F_2 = D_2 \wedge E_2 = E_2.$$

Then,

$$\begin{aligned} c_8 &= F_2 \vee F_1 \\ &= E_2 \qquad \qquad \qquad \vee \quad [P_2] \wedge E_1 \end{aligned}$$

All other  $c_i$ , for  $1 \leq i \leq 7$ , are also obtained as a byproduct of the  $c_8$  computation.

$$c_7 = [p_7 \wedge (g_7 \vee (p_6 \wedge (g_6 \vee (p_5 \wedge g_5))))] \quad \vee \quad [p_7 \wedge p_6 \wedge p_5] \wedge E_1$$

$$c_6 = [p_6 \wedge (g_6 \vee (p_5 \wedge g_5))] \quad \vee \quad [p_6 \wedge p_5] \wedge E_1$$

$$c_5 = [p_5 \wedge g_5] \quad \vee \quad [p_5] \wedge E_1$$

$$c_4 = E_1$$

$$c_3 = [p_3 \wedge (g_3 \vee (p_2 \wedge (g_2 \vee (p_1 \wedge g_1))))]$$

$$c_2 = [p_2 \wedge (g_2 \vee (p_1 \wedge g_1))]$$

$$c_1 = [p_1 \wedge g_1]$$

**Case3:**  $r = 4$  and  $q = 2$ .

$$P_1 = p_2 \wedge p_1, \quad P_2 = p_4 \wedge p_3, \quad P_3 = p_6 \wedge p_5, \quad P_4 = p_8 \wedge p_7$$

$$D_1 = P_4 \wedge P_3 \wedge P_2, \quad D_2 = P_4 \wedge P_3, \quad D_3 = P_4, \quad D_4 = 1$$

$$E_1 = p_2 \wedge (g_2 \vee (p_1 \wedge g_1)) \quad E_2 = p_4 \wedge (g_4 \vee (p_3 \wedge g_3))$$

$$E_3 = p_6 \wedge (g_6 \vee (p_5 \wedge g_5)) \quad E_4 = p_8 \wedge (g_8 \vee (p_7 \wedge g_7))$$

and

$$F_1 = D_1 \wedge E_1 = [P_4 \wedge P_3 \wedge P_2] \wedge E_1$$

$$F_2 = D_2 \wedge E_2 = [P_4 \wedge P_3] \wedge E_2$$

$$F_3 = D_3 \wedge E_3 = [P_4] \wedge E_3$$

$$F_4 = D_4 \wedge E_4 = E_4.$$

Then,

$$\begin{aligned} c_8 &= F_4 \vee F_3 \vee F_2 \vee F_1 \\ &= E_4 \quad \vee \quad [P_4] \wedge E_3 \quad \vee \quad [P_4 \wedge P_3] \wedge E_2 \quad \vee \quad [P_4 \wedge P_3 \wedge P_2] \wedge E_1 \end{aligned}$$

All other  $c_i$ , for  $1 \leq i \leq 7$ , are also obtained as a byproduct of the  $c_8$  computation.

$$\begin{aligned} c_7 &= [p_7 \wedge g_7] \quad \vee \quad [p_7] \wedge E_3 \quad \vee \quad [p_7 \wedge P_3] \wedge E_2 \quad \vee \\ &\quad [p_7 \wedge P_3 \wedge P_2] \wedge E_1 \end{aligned}$$

$$c_6 = E_3 \quad \vee \quad [P_3] \wedge E_2 \quad \vee \quad [P_3 \wedge P_2] \wedge E_1$$

$$c_5 = [p_5 \wedge g_5] \quad \vee \quad [p_5] \wedge E_2 \quad \vee \quad [p_5 \wedge P_2] \wedge E_1$$

$$c_4 = E_2 \quad \vee \quad [P_2] \wedge E_1$$

$$c_3 = [p_3 \wedge g_3] \quad \vee \quad [p_3] \wedge E_1$$

$$c_2 = E_1$$

$$c_1 = [p_1 \wedge g_1]$$

**Case4:**  $r = 8$  and  $q = 1$ .

Then,

$$P_1 = p_1, \quad P_2 = p_2, \quad P_3 = p_3, \quad P_4 = p_4,$$

$$P_5 = p_5, \quad P_6 = p_6, \quad P_7 = p_7, \quad P_8 = p_8$$

$$D_1 = P_8 \wedge P_7 \wedge P_6 \wedge P_5 \wedge P_4 \wedge P_3 \wedge P_2, \quad D_2 = P_8 \wedge P_7 \wedge P_6 \wedge P_5 \wedge P_4 \wedge P_3,$$

$$D_3 = P_8 \wedge P_7 \wedge P_6 \wedge P_5 \wedge P_4, \quad D_4 = P_8 \wedge P_7 \wedge P_6 \wedge P_5,$$

$$D_5 = P_4 \wedge P_3 \wedge P_2, \quad D_6 = P_8 \wedge P_7,$$

$$D_7 = P_8, \quad D_8 = 1$$

$$E_1 = p_1 \wedge g_1 \quad E_2 = p_2 \wedge g_2 \quad E_3 = p_3 \wedge g_3 \quad E_4 = p_4 \wedge g_4$$

$$E_5 = p_5 \wedge g_5 \quad E_6 = p_6 \wedge g_6 \quad E_7 = p_7 \wedge g_7 \quad E_8 = p_8 \wedge g_8$$

and

$$F_1 = D_1 \wedge E_1 = [P_8 \wedge P_7 \wedge P_6 \wedge P_5 \wedge P_4 \wedge P_3 \wedge P_2] \wedge E_1$$

$$F_2 = D_2 \wedge E_2 = [P_8 \wedge P_7 \wedge P_6 \wedge P_5 \wedge P_4 \wedge P_3] \wedge E_2$$

$$F_3 = D_3 \wedge E_3 = [P_8 \wedge P_7 \wedge P_6 \wedge P_5 \wedge P_4] \wedge E_3$$

$$F_4 = D_4 \wedge E_4 = [P_8 \wedge P_7 \wedge P_6 \wedge P_5] \wedge E_4$$

$$F_5 = D_5 \wedge E_5 = [P_8 \wedge P_7 \wedge P_6] \wedge E_5$$

$$F_6 = D_6 \wedge E_6 = [P_8 \wedge P_7] \wedge E_6$$

$$F_7 = D_7 \wedge E_7 = [P_8] \wedge E_7$$

$$F_8 = D_8 \wedge E_8 = E_8$$

Then,

$$c_8 = F_8 \vee F_7 \vee F_6 \vee F_5 \vee F_4 \vee F_3 \vee F_2 \vee F_1$$

$$\begin{aligned} c_8 = E_8 & \quad \vee \quad [P_8] \wedge E_7 & \quad \vee \\ & \quad \vee \quad [P_8 \wedge P_7] \wedge E_6 & \quad \vee \\ & \quad \vee \quad [P_8 \wedge P_7 \wedge P_6] \wedge E_5 & \quad \vee \\ & \quad \vee \quad [P_8 \wedge P_7 \wedge P_6 \wedge P_5] \wedge E_4 & \quad \vee \end{aligned}$$

$$[P_8 \wedge P_7 \wedge P_6 \wedge P_5 \wedge P_4 \wedge P_3] \wedge E_2 \quad \vee \quad [P_8 \wedge P_7 \wedge P_6 \wedge P_5 \wedge P_4 \wedge P_3 \wedge P_2] \wedge E_1$$

All other  $c_i$ , for  $1 \leq i \leq 7$ , are also obtained as a byproduct of the  $c_8$  computation.

$$c_7 = E_7 \quad \vee \quad [P_7] \wedge E_6 \quad \vee$$

$$[P_7 \wedge P_6] \wedge E_5 \quad \vee \quad [P_7 \wedge P_6 \wedge P_5] \wedge E_4 \quad \vee$$

$$[P_7 \wedge P_6 \wedge P_5 \wedge P_4] \wedge E_3 \quad \vee \quad [P_7 \wedge P_6 \wedge P_5 \wedge P_4 \wedge P_3] \wedge E_2 \quad \vee$$

$$[P_7 \wedge P_6 \wedge P_5 \wedge P_4 \wedge P_3 \wedge P_2] \wedge E_1$$

$$c_6 = E_6 \quad \vee \quad [P_6] \wedge E_5 \quad \vee$$

$$[P_6 \wedge P_5] \wedge E_4 \quad \vee \quad [P_6 \wedge P_5 \wedge P_4] \wedge E_1 \quad \vee$$

$$[P_6 \wedge P_5 \wedge P_4 \wedge P_3] \wedge E_2 \quad \vee \quad [P_6 \wedge P_5 \wedge P_4 \wedge P_3 \wedge P_2] \wedge E_1$$

$$c_5 = E_5 \quad \vee \quad [P_5] \wedge E_4 \quad \vee \quad [P_5 \wedge P_4] \wedge E_3 \quad \vee$$

$$[P_5 \wedge P_4 \wedge P_3] \wedge E_2 \quad \vee \quad [P_5 \wedge P_4 \wedge P_3 \wedge P_2] \wedge E_1$$

$$c_4 = E_4 \quad \vee \quad [P_4] \wedge E_3 \quad \vee \quad [P_4 \wedge P_3] \wedge E_2 \quad \vee \quad [P_4 \wedge P_3 \wedge P_2] \wedge E_1$$

$$c_3 = E_3 \quad \vee \quad [P_3] \wedge E_2 \quad \vee \quad [P_3 \wedge P_2] \wedge E_1$$

$$c_2 = E_2 \quad \vee \quad [P_2] \wedge E_1$$

$$c_1 = E_1$$

In general, given  $N = 2^n$ , let  $r \geq 1$  and  $q \geq 1$  be integers such that  $N = rq$ . The following holds.

- Decomposition:  $r$  blocks, each has  $q$  elements.

$$r = 2^i \text{ and } q = 2^{n-i} \quad \text{for} \quad 0 \leq i \leq n.$$

➤ Number of possible parallel implementations =  $n$ .

- Family of prefix circuit:

- For each  $r > 1$ , number of prefix circuits built =  $r - 1$ .

The computation time for an  $N$ -bit adder is at least

$$4 + \lceil \lg r \rceil + \max\{T_c(q), \lceil \lg q \rceil + \text{PrefixComputationTime}\}$$

Table 6.1 lists all the operations used in the calculation of a  $N$ -bit adder. The total number of AND gates used depends on the prefix circuit. The degree of parallelism depends on the size of the block and the number of blocks: the bigger the block size, the lower the degree of parallelism. When the size of the block is  $N$ , it is a serial computation of all carry bits.

**Table 6.2: Gate count of a  $N$ -bit adder.**

| Gate Count                                      | AND                 | OR                  | XOR  |
|---|---------------------|---------------------|------|
| Computing $p_i$ and $g_i$ for $1 \leq i \leq N$ | $N$                 | $N$                 | -    |
| Computing $E_i$ for $1 \leq i \leq r$           | $N$                 | $r(q-1)$            | -    |
| Prefix Circuit                                  | vary                | vary                | -    |
| Combining $E_i$ and Prefix Circuit              | $\frac{q(r-1)r}{2}$ | -                   | -    |
| Calculating $c_i$ for $1 \leq i \leq N$         | -                   | $\frac{q(r-1)r}{2}$ | -    |
| Calculating $s_i$ for $1 \leq i \leq N$         | -                   | -                   | $2N$ |



## **CHAPTER 7**

### **SIMULATION RESULTS**

In Chapter 5, the performance in term of time-delay, power consumption, and power-delay product of parallel prefix circuits described in Chapter 2 was investigated. In this chapter, we extend the investigation of their application in prefix adder. Binary adder using the prefix circuits of varying sizes was proposed by Brent [Bre70]. In our investigation, we use Brent's algorithm. We compare number of operations used, time-delay and power consumption as well as the power-delay product in order to find out the effect of varying the size of different parallel prefix circuits on speed and power consumption.

#### **7.1 Effect of Block Size on Adder Implementation**

The 8-, 16-, 32-, and 64-bit adders with varying block sizes for computation of carries were simulated. The simulation was carried out using PSpice at a power supply voltage of 3V. The divide-and-conquer prefix circuit is the candidate circuit for the study. The simulation results are shown in Table 7.1 and Figure 7.1.

Table 7.1 compares the exact gate count, time-delay, power consumption, and power-delay product for all prefix adders studied. Figure 7.1 illustrates the graphical representation of the comparisons. The comparison results obtained from PSpice simulation allow us to conclude that there is a difference in speed and power between

**Table 7.1: Gate count, delay time, power consumption, and power-delay-product of different design of 8-, 16-, 32-, and 64-bit adders using the divide-conquer prefix circuit.**

**8-bit adder**

| Type of Implementations |            | Gate Count |    |     | Delay (us) | Power Consumption (uW) | Power-delay Product |
|-------------------------|------------|------------|----|-----|------------|------------------------|---------------------|
| Number of Blocks        | Block Size | AND        | OR | XOR |            |                        |                     |
| 1                       | 8          | 16         | 15 | 16  | 8.14       | 0.99                   | 8.05                |
| 2                       | 4          | 24         | 18 | 16  | 6.01       | 1.15                   | 6.90                |
| 4                       | 2          | 37         | 24 | 16  | 4.50       | 1.46                   | 6.56                |
| 8                       | 1          | 65         | 36 | 16  | 4.02       | 2.10                   | 8.42                |

**16-bit adder**

| Type of Implementations |            | Gate Count |     |     | Delay (us) | Power Consumption (uW) | Power-delay Product |
|-------------------------|------------|------------|-----|-----|------------|------------------------|---------------------|
| Number of Blocks        | Block Size | AND        | OR  | XOR |            |                        |                     |
| 1                       | 16         | 32         | 31  | 32  | 16.43      | 1.99                   | 32.65               |
| 2                       | 8          | 52         | 38  | 32  | 10.20      | 2.39                   | 24.31               |
| 4                       | 4          | 80         | 52  | 32  | 6.60       | 3.07                   | 20.22               |
| 8                       | 2          | 137        | 80  | 32  | 5.08       | 4.32                   | 21.9                |
| 16                      | 1          | 257        | 136 | 32  | 4.51       | 5.08                   | 22.88               |

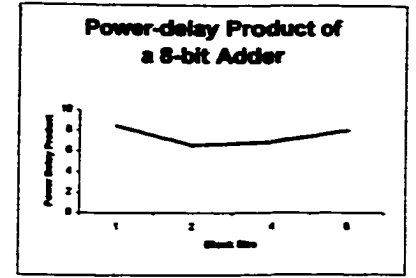
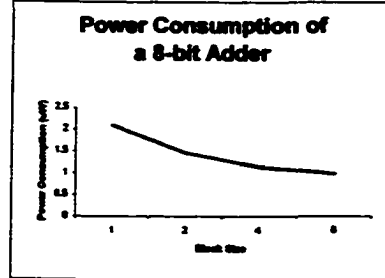
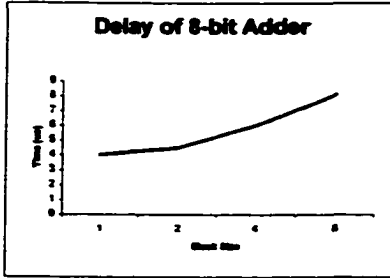
**32-bit adder**

| Type of Implementations |            | Gate Count |     |     | Delay (us) | Power Consumption (uW) | Power-delay Product |
|-------------------------|------------|------------|-----|-----|------------|------------------------|---------------------|
| Number of Blocks        | Block Size | AND        | OR  | XOR |            |                        |                     |
| 1                       | 32         | 64         | 63  | 64  | 33.12      | 3.96                   | 131.17              |
| 2                       | 16         | 112        | 78  | 64  | 18.55      | 4.85                   | 89.91               |
| 4                       | 8          | 172        | 108 | 64  | 10.79      | 6.30                   | 68.01               |
| 8                       | 4          | 288        | 168 | 64  | 7.19       | 9.13                   | 65.66               |
| 16                      | 2          | 529        | 288 | 64  | 5.68       | 14.84                  | 84.33               |

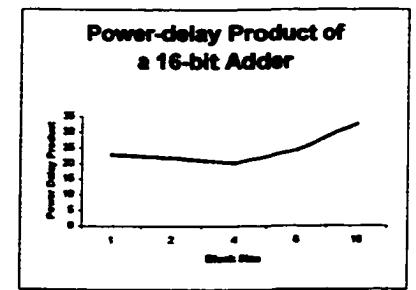
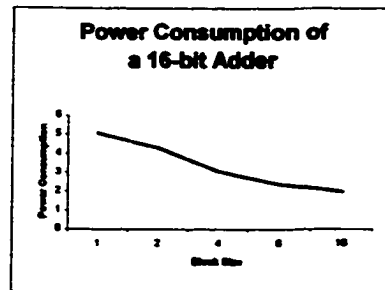
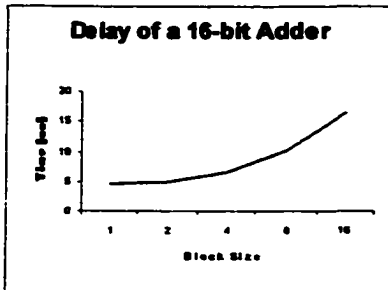
**64-bit adder**

| Type of Implementations |            | Gate Count |     |     | Delay (us) | Power Consumption (uW) | Power-delay Product |
|-------------------------|------------|------------|-----|-----|------------|------------------------|---------------------|
| Number of Blocks        | Block Size | AND        | OR  | XOR |            |                        |                     |
| 4                       | 16         | 368        | 220 | 128 | 19.19      | 13.57                  | 260.30              |
| 8                       | 8          | 604        | 334 | 128 | 11.41      | 19.65                  | 224.17              |
| 16                      | 4          | 1088       | 592 | 128 | 7.81       | 31.36                  | 244.78              |

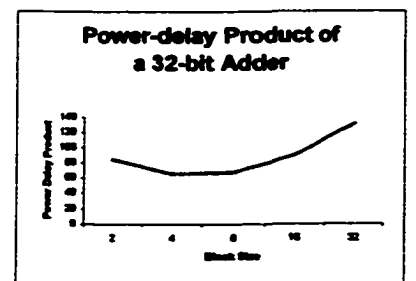
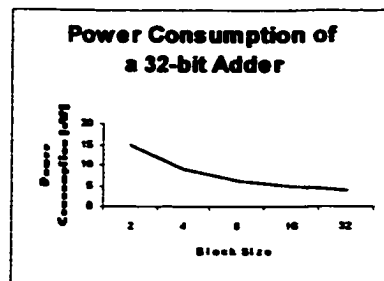
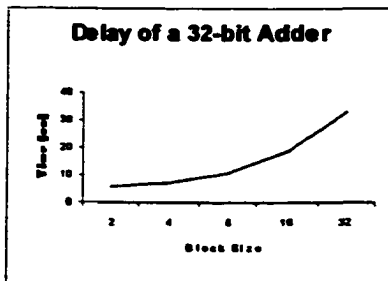
### 8-bit adder



### 16-bit adder



### 32-bit adder



### 64-bit adder

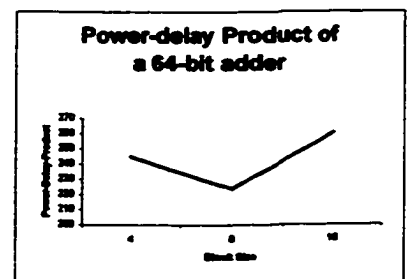
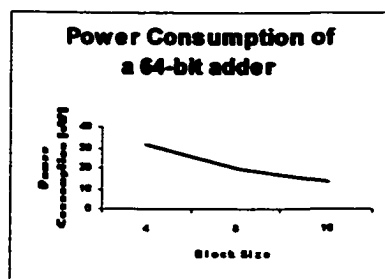
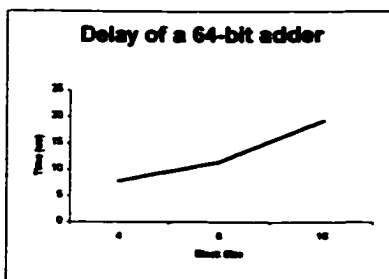
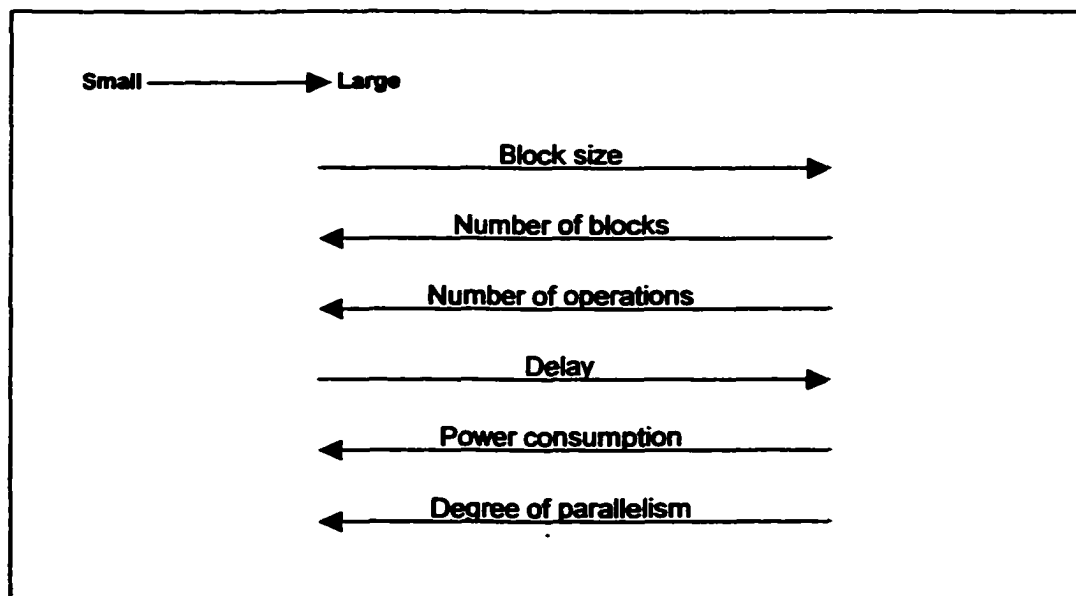


Figure 7.1: The plots of delay time, power consumption, and power-delay-product of different design of 8-, 16-, 32-, and 64-bit adder using the divide-conquer prefix circuit.

different blocking schemes. In regard to the power-delay product, it is interesting to observe that the best performance of the implementation lies somewhat in the middle value of the various choices for block size. The optimum block size of 8-bit adder is two. The block size of four is optimum for 16-bit and 32-bit adders while the block size of eight is optimum for 64-bit adder.

The effect of the blocking schemes can be summarized as in Figure 7.2. As the block size increases, it takes longer to complete the computation, but it consumes less power. The bigger block size performs fewer operations and has less degree of parallelism. For example, an  $N$ -bit adder implemented with a block size of  $N$  has the smallest number of operations and the lowest power consumption compared to implementations using smaller block sizes.



**Figure 7.2:** The illustration of the effect of the block size on other factors.

Unfortunately, it is also the slowest. The implementation with the smallest block size gives the fastest adder for every input length at the cost of a large number of operations and power consumption. From the power consumption point of view, the implementation with biggest block size is, therefore, the most efficient one. The opposite holds true if the circuit delay is important. On the other hand, the biggest and smallest block sizes show poor power-delay product.

## **7.2 Effect of Prefix Circuit on Adder Implementation**

In this section, three different prefix circuits, the divide-and-conquer, the Shih-Lin, and the LYD prefix circuits, were chosen to be candidates for carry computation in our simulation study. This is because, in Chapter 5, we found that the divide-and-conquer prefix circuit is the fastest prefix circuit while the LYD prefix circuit gives the best performance in terms of power-delay product. The Shih-Lin prefix circuit is a (size, depth)-optimal prefix circuit [LS99]. Recall that there are two issues to be considered in the process of carry computation. The first issue is the computation of the prefix circuit inside the block and another is the computation of a family of prefix circuits.

In the simulation, three best block schemes (that is four, eight, and sixteen) in term of power-delay product are considered. Figure 7.3 shows the simulation result of the 64-bit adder with three different block size schemes implemented with three different prefix circuits. The optimum block size in terms of the power-delay product turns out to be eight for each of the three prefix circuits. The power-delay-products of the LYD and Shih-Lin prefix circuits are similar. The divide-and-conquer prefix circuit has the highest power-delay product when the block size is eight and sixteen. However, when the block size

reduces to four, the power-delay product of the divide-and-conquer becomes closer to the LYD and Shih-Lin prefix circuits. This is due to the strong impact of using the divide-and-conquer prefix circuit in the computation of the family of prefix circuit on power consumption. The computation benefits from the divide-and-conquer prefix circuit's well-organized structure, which allows efficient implementation.

The size of the family of prefix circuits depends on the block size; the smaller the block size, the bigger the family of prefix circuits. For example, a block size of four has fifteen prefix circuits in the family. On the other hand, a block size of sixteen has only three prefix circuits in the family. When the family of prefix circuits is larger, there is a large possibility of its members sharing the structure with other members. This will result in lower power consumption due to reduced number of computation nodes. On the other hand, with the smaller family of prefix circuit, the sharing is small.

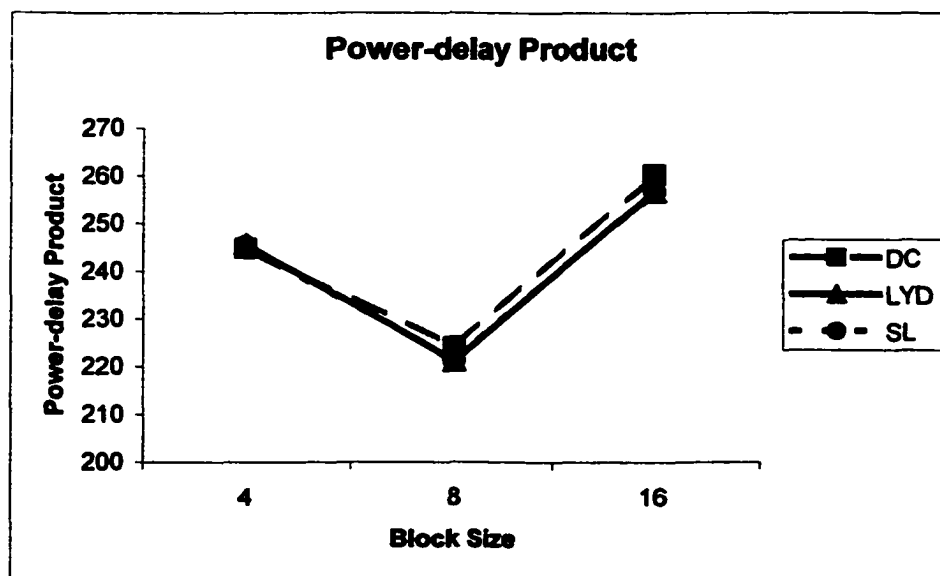
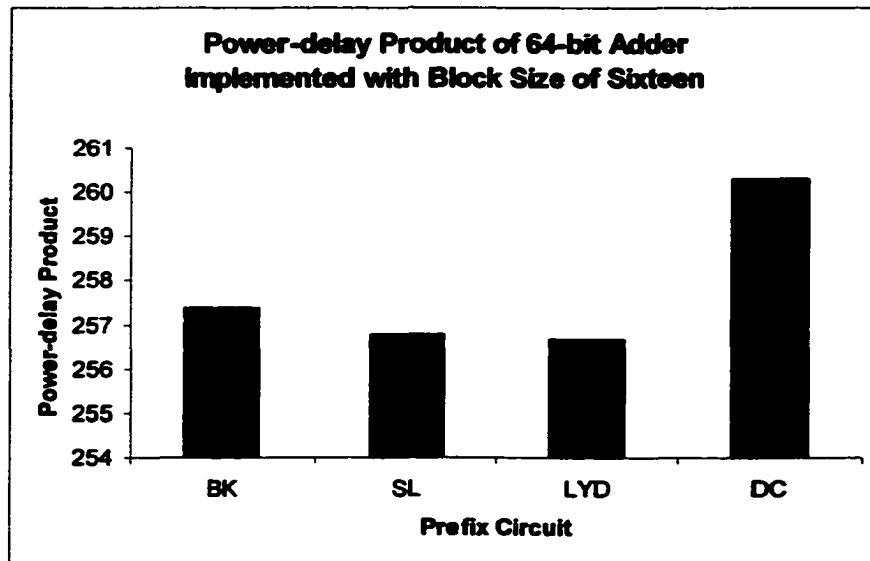


Figure 7.3: The plot of power-delay product of the divide-and-conquer, the LYD, and the Shih-Lin prefix circuits.

The choice of the prefix circuit inside the block also dominates the power consumed especially when the block size is large. To see its effect on power-delay product, let us consider power-delay product of the 64-bit adder implemented with block size of sixteen in four different prefix circuits in Figure 7.4. The figure shows that the (size, depth)-optimal prefix circuits (i.e., the Shih-Lin and the LYD prefix circuits) have smaller power-delay product than (size, depth)-non-optimal prefix circuits (i.e., the divide-and-conquer and the Brent-Kung prefix circuits). Like the simulation results in Chapter 5, the divide-and-conquer prefix circuit has the highest power-delay product followed by the Brent-Kung, the Shih-Lin, and the LYD prefix circuits.



**Figure 7.4:** The plot of power-delay product of four prefix circuits using in carry calculation in 64-bit adder implementing with block size of sixteen.

### **7.3 Summary**

The binary adder implemented with different block schemes consumes different levels of power. According to the Brent's algorithm [Bre70], there are  $\lg N$  ways to implement parallel  $N$ -bit adders. In terms of power-delay product, our simulation results show that the optimum block size falls somewhere in the middle of all the block sizes. In order to implement a low-power prefix adder [Bre70], the LYD prefix circuit is a good candidate for implementing prefix circuit inside the block while the prefix circuit with well-organized structure is a good candidate for implementing the family of prefix circuits.



## **CHAPTER 8**

### **CONCLUSIONS**

The three most widely accepted metrics for measuring the quality of a circuit are its area, speed, and power consumption. Optimizing area and speed have been considered important for long time, but minimizing power consumption has been gaining prominence only recently. Power consumption is an important issue in both portable and non-portable systems.

The dominant source of power consumption is dependent on supply voltage and switching activity when capacitance and operating frequency are fixed. Therefore, the reduction in voltage and switching activity means the reduction in power. However, a reduction in voltage may result in longer delays, and reduced throughput. However, reduction in throughput can be overcome by parallelism. Because of the size-depth trade-off characteristic of prefix circuits, parallelism can be increased only up to a certain level.

Different circuit structures induce different switching activity. As a result, different circuit architectures for performing the same function can consume different amount of power. Therefore, the implementation of the various prefix circuits in an application will have different power consumptions as well. Usually, the circuit architecture with longer depth will consume more power than one with shorter depth. However, due to the size-depth trade-off characteristic of prefix circuits, the switching activity in a prefix circuit not only depends on its logic depth but also on the number of operation nodes at each

level. The circuit with shorter depth and more nodes might have more switching activity than the one with longer depth and less nodes.

In this dissertation we conducted a comparative study of various prefix circuits from the point of view of power-speed trade-off. The dissertation presented the linear output capacitance model for the estimation of power consumption in seven families of prefix circuits. The proposed linear output capacitance model allows us to estimate power consumption in prefix circuits considered. This model helps direct the design at the high level. Results obtained by the model and simulations refute several commonly held beliefs about the consumption of power in prefix circuits (i.e., a circuit with shorter depth consumes less power than a circuit with longer depth), and also lend insight into possible prefix circuits for future power-prediction prefix circuit applications. Besides, based on the model and simulations, we have investigated the possible decrease in power consumption with the use of low supply voltages while maintaining the original performance level under different prefix circuits. For example, the simulation results have shown that power reductions of about 1.6 times can be obtained without throughput loss by using the LYD prefix circuit compared with using the divide-and-conquer prefix circuit. Finally, the 8-, 16-, 32- and 64-bit prefix adders were implemented and simulated under different blocking schemes. In regard to power-delay product, we found that an optimum block size falls somewhere around the middle among the various possible block sizes. For example, the result shows that the optimum block size is two for 8-bit adder, four for 16-bit and 32-bit adders, and eight for 64-bit adder. In order to implement a low-power prefix adder based on Brent's algorithm [Bre70], the (size, depth)-optimal prefix circuit is a good candidate for implementing prefix circuit inside the block while the

**prefix circuit with well-organized structure is a good candidate for implementing the family of prefix circuits needed in the prefix adder.**

**There are several open questions for future work.**

- **Power modeling of prefix circuits with bounded fan-out.**
- **The effect of fan-out on time-delay.**
- **New prefix circuit that has a structure that benefits the computation of a family of prefix circuits.**
- **Pipelining implementation for low-power prefix circuit.**

## BIBLIOGRAPHY

- [AD98] C. Arjhan, and R. G. Deshmukh, "A Novel Fault-Model For Regular And Irregular Parallel-prefix Adders, *Proceedings, IEEE on Southeastcon*, pp. 397-400, 1998.
- [BAW00] H. T. Bui, A. Al-Sheraidha, and Y. Wang, "Design and Analysis of 10-Transistor Full Adders Using Novel XOR-XNOR Gates", *IEEE Proceedings of ICSP*, pp. 619-622, 2000.
- [BD01] V. A. Bartlett, and A. G. Dempster, "Using Carry-save Adders in Low-power Multiplier Blocks", *The 2001 IEEE International Symposium on Circuits and Systems*, Vol. 4, pp. 222 –225, 2001.
- [Bel01] C. Belady, "Cooling and Power Consideration for Semiconductors Into the Next Century", *Proceedings of the 2001 International Symposium on Low Power Electronics and Design*, pp.100-105, 2001.
- [BL01] A. Beaumont-Smith, and C. C. Lim, "Parallel Prefix Adder Design", *Proceedings The 15<sup>th</sup> IEEE Symposium on Computer Arithmetic*, pp. 218-225, 2001.
- [Boy99] R. L. Boylestad, *Introductory Circuit Analysis*, Prentice Hall, 1999.
- [Bre70] R. P. Brent, "On the Addition of Binary Numbers", *IEEE Transactions on Computers*, Vol. 19, pp. 758-759, 1970.
- [BK82] R. P. Brent, and H. T. Kung, "A Regular Layout for Parallel Adders", *IEEE Transactions on Computers*, Vol. 31, pp. 260-264, 1982.

- [BM00] L. Benini, and G. Micheli, "System-level Power Optimization: Techniques and Tolls", *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, Vol. 5(2), April 2000.
- [Cad00] Cadence Design Systems, Inc., *PSpice User's Guide Manual*, Version 9.2, San Jose, CA, January 2000.
- [Cal96] T. K. Callaway, *Area, Delay, and Power Modeling of CMOS Adder and Multipliers*, Ph.D. Dissertation, The University of Texas at Austin, 1996.
- [CB95] A. P. Chandrakasan, and R. W. Brodersen, *Low Power Digital CMOS Design*, Kluwer Academic Publishers, Norwell, MA, 1995.
- [FB01] A. A. Fayed, and M. A. Bayoumi, "A Low Power 10-transistor Full Adder Cell For Embedded Architectures", *The 2001 IEEE International Symposium on Circuits and Systems*, Vol. 4, pp. 226 –229, 2001.
- [FL00] S. B. Furber, and J. Liu, "A Novel Area-efficient Binary Adder", Conference Record of the Thirty-Fourth Asilomar Conference on Signals, Systems and Computers, Vol. 1, pp. 119 –123, 2000.
- [GNHF01] K. I. Geisler, T. D. Nielsen, D. F. Hall, and R. Frowd, "The Rise of Energy Delivery Management Systems", *IEEE Transmission and Distribution Conference and Exposition*, Vol. 2, pp. 895-900, 2001.
- [HP90] J. Hennessy, and D. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann Publishers, 1990.
- [Hub00] P. Huber, "Why 99.9 percent is not good enough", ACM: Ubiquity, New York, 2000, [http://www.acm.org/ubiquity/interviews/p\\_huber\\_1.html](http://www.acm.org/ubiquity/interviews/p_huber_1.html).

- [Hwa79] K. Hwang, *Computer Arithmetic: Principles, Architecture, and Design*, New York, John Wiley and Sons, 1979.
- [JS95] K. J. Janik, and L. Shih-Lien, "VLSI Implementation of a 32-bit Kozen Formulation Ladner/Fischer Parallel Prefix Adder", *Proceedings of the 8<sup>th</sup> Annual IEEE International*, pp. 57-59, 1995.
- [Kno01] S. Knowles, "A Family of Adders". *Proceedings. 15<sup>th</sup> IEEE Symposium on Computer Arithmetic*, pp. 277 –281, 2001.
- [KBL95] U. Ko, P. T. Balsara, and W. Lee, "Low-Power Design Techniques for High-Performance CMOS Adders", *IEEE Transactions on Very Large Scale Integration Systems*, Vol. 3(2), 1995.
- [Kor93] I. Koren, *Computer Arithmetic Algorithms*, Englewood Cliffs, N.J., Prentice Hall, 1993.
- [LF80] R. E. Ladner, and M. J. Fischer, "Parallel Prefix Computation", *Journal of ACM*, Vol. 27, pp. 831-838, 1980.
- [LYD87] S Lakshmivarahan, C. M. Yang, and S. K. Dhall, "Optimal Parallel Prefix Circuits with  $(size, depth) = 2N - 2$  and  $\lceil \log N \rceil \leq depth \leq \lceil 2 \log N \rceil - 3$ ", *Proceedings of the International Conference on Parallel Processing*, pp. 58-65, 1987.
- [LD90] S. Lakshmivarahan, and S. K. Dhall, *Analysis and Design of Parallel Algorithms: Arithmetic and Matrix Problems*, McGraw Hill, New York, NY, 1990.
- [LD94] S. Lakshmivarahan, and S. K. Dhall, *Parallel Computing Using the Prefix Problem*. Oxford University Press, New York, NY, 1994.

- [LS99] Y. M. Lin, and C. C. Shih, "A New Class of Depth-Size Optimal Parallel Prefix Circuits", *Journal of Supercomputing*, Vol. 14, pp. 39-52, 1999.
- [Lin81] H. Ling, "High-Speed Binary Adder", *IBM J. Research and Development*, Vol. 25, pp. 156-166, May 1981.
- [Mac96] E. Macii, "RT and Algorithmic-Level Optimization for Low Power", *Low Power Design in Deep Submicron Electronics*, Kluwer Academic Publishers, pp. 355-379, 1996.
- [Mil00] M. Mills, "Kyto and the Internet: The Energy Implications of the Digital Economy", Testimony before the Subcommittee on national Economic Growth, Natural, and Regulatory Affairs, U.S. House of Representatives, Washington, DC., February 2000,  
<http://www.house.gov/reform/neg/hearings/020200/mills.htm>.
- [NIO96] C. Nagendra, M J. Irwin, and R. M. Owens, "Area-Time-Power Tradeoffs in Parallel Adders", *IEEE Transactions on Circuits and Systems*, Vol. 43(10), pp. 689-702, 1996.
- [Omo94] A. R. Omondi, *Computer Arithmetic Systems, Algorithms, Architecture and Implementations*, Prentice Hall, 1994.
- [RCN01] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits A Design Perspective*, early draft of the 2<sup>nd</sup> edition, April 2001,  
<http://bwrc.eecs.berkeley.edu/Classes/IcBook/2ndEdition.html>.
- [Rad01] D. Radhakrishnan, "Low-voltage Low-power CMOS Full Adder", *IEEE Proceedings on Circuits, Devices and Systems*, Vol. 148, pp. 19 -24, 2001.

- [RP96] J. M. Rabaey, and M. Pedram, *Low Power Design Methodologies*, Kluwer Academic Publishers, Boston, 1996.
- [RP00] K. Roy, and S. Prasad, "Low-power CMOS VLSI Circuit Design", John Wiley, New York, 2000.
- [Smi97] M. Smith, *Application-Specific Integrated Circuits*, Addison Wesley, Menlo Park, CA, 1997.
- [Sni86] M. Snir, "Depth-Size Tradeoffs for Parallel Prefix Computation", *Journal of Algorithms*, Vol. 17, pp. 185-201, 1986.
- [WE93] N. H. E. Weste, and K. Eshraghian, *Principles of CMOS VLSI Design: A System Perspective*, Addison-Wesley, MA, 1993.
- [ZS01] M. Ziegler, and M. Stan, "Optimal logarithmic adder structures with a fan-out of two for minimizing the area-delay product", *The 2001 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 657-660, Vol. 2, 2001.



## APPENDIX A

### RC network

Recall that there are three sources of power dissipation in a digital CMOS circuit. The majority source of the power dissipation is due to the logic transitions. As the nodes in a digital CMOS circuit transition back and forth between the two logic levels, the parasitic capacitances are charged and discharged.

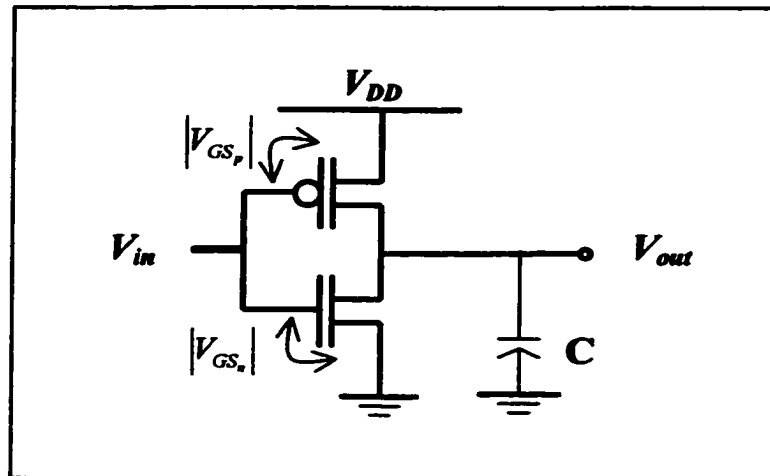
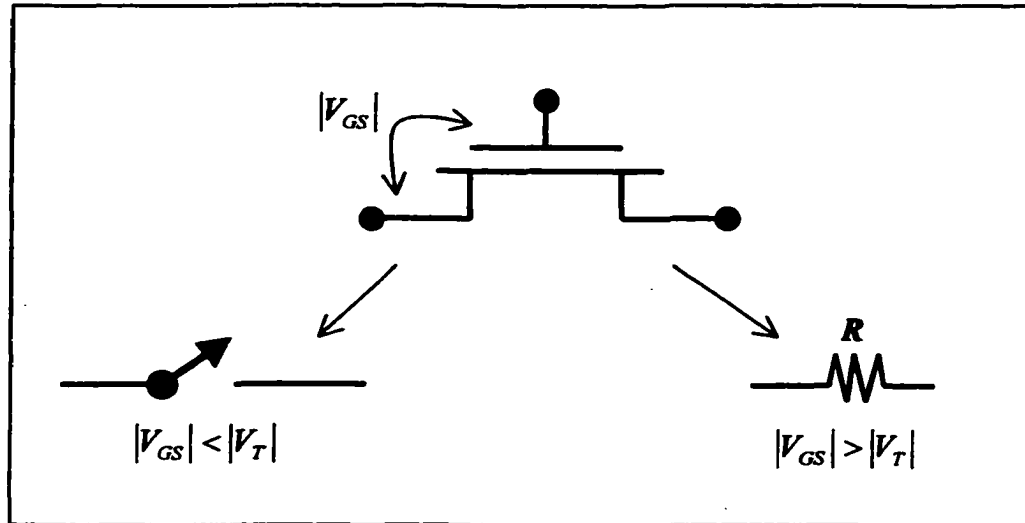


Figure A.1: Inverter.

Most of the models used to explain the power consumption behavior of ICs are based on the equations derived from the analysis of the CMOS inverter (see Figure A.1) [RCN01]. Understanding its behaviors can be extended to explain the behaviors of more complicated designs such as NAND gates, adders, etc. Hence, an overview of the CMOS inverter is presented. Refer to [RCN01, WE93], for more detail.

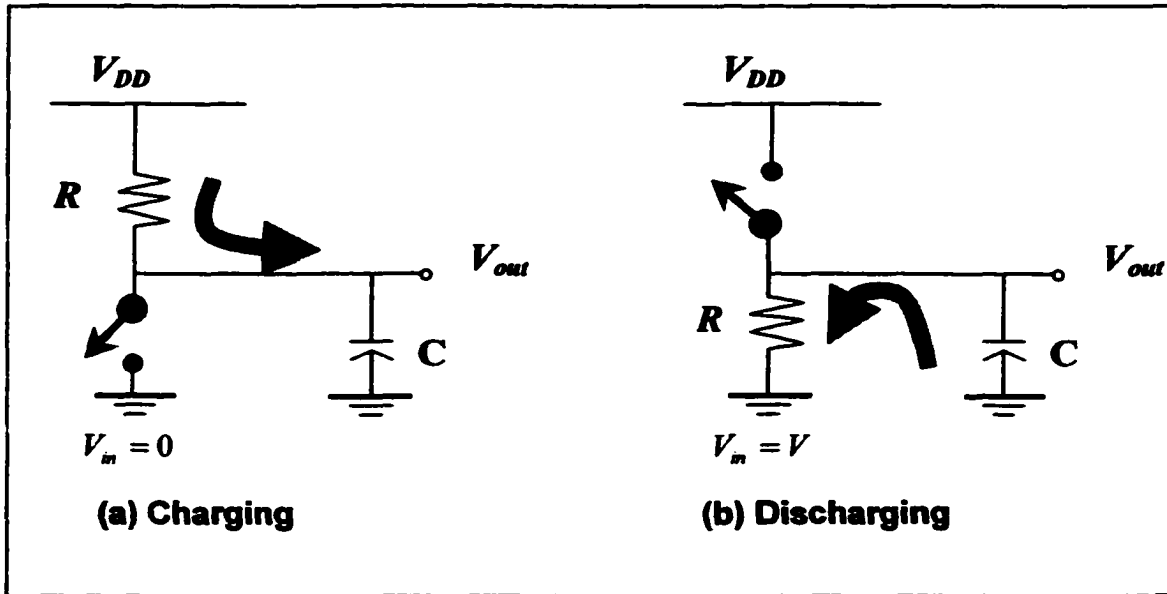
In the switch model [RCN01], a transistor could be either a switch or a resistor, depending on the value of its gate-to-source voltage,  $V_{GS}$ , and its threshold voltage,  $V_T$ . The transistor acts like the switch when  $|V_{GS}| < |V_T|$  and acts like the resistor when  $|V_{GS}| > |V_T|$  (see Figure A.2).



**Figure A.2:** Switch model of CMOS transistor.

When applying a step input, the capacitor will charge and discharge in response to the input to an inverting gate. When the input goes from its low level to its high level ( $V_{in}$  going from 0 to  $V$ ), the  $P$ -type transistor acts like the resistor and the  $N$ -type transistor acts like the switch (see Figure A.3(a)). An  $RC$  network is formed. The capacitor charges toward the high level of the input through the resistor and  $V_{out} = V$ . This action is analogous to connecting a supply voltage to the  $RC$  network as illustrated in Figure A.3(a). When the input goes from its high level back to its low level ( $V_{in}$  going from  $V$  to 0), the  $P$ -type transistor acts like the switch and the  $N$ -type transistor acts like the resistor (see Figure A.3(b)). An  $RC$  network is formed. The  $N$ -type transistor provides a

current path to the ground. The capacitor discharges back through the ground and  $V_{out} = 0$ . This action is analogous to replacing the NMOS with the  $RC$  network, as illustrated in Figure A.3(b).



**Figure A.3:** The equivalent action of an inverting gate when a step input charges and discharges the capacitor.

When a capacitor charges or discharges through a resistor  $R$ , a certain time is required for the capacitor to charge/discharge fully. The rate at which the capacitor charges or discharges is determined by the time constant  $RC$  (i.e., it is the first-order analysis of digital circuits). The  $RC$  is derived and has the units of time as follows:

$$RC = \left(\frac{V}{I}\right)\left(\frac{Q}{V}\right) = \left(\frac{V}{Q/t}\right)\left(\frac{Q}{V}\right) = t$$

Its symbol is  $\tau$  (Greek letter tau). Thus,

$$\tau = RC.$$

Its unit of measure is the second. Note that the value of  $RC$  will never be greater than a few seconds because  $C$  is very small (i.e., it is usually found in microfarads or picofarads), unless  $R$  is very large.

A capacitor charges and discharges following an exponential curve, as shown in Figure A.4. In the charging phase, the charging curve is an increasing exponential while in the discharging phase, the discharging curve is a decreasing exponential (see Figure A.4). The charging voltage of a capacitor at any instant of time is given as follow:

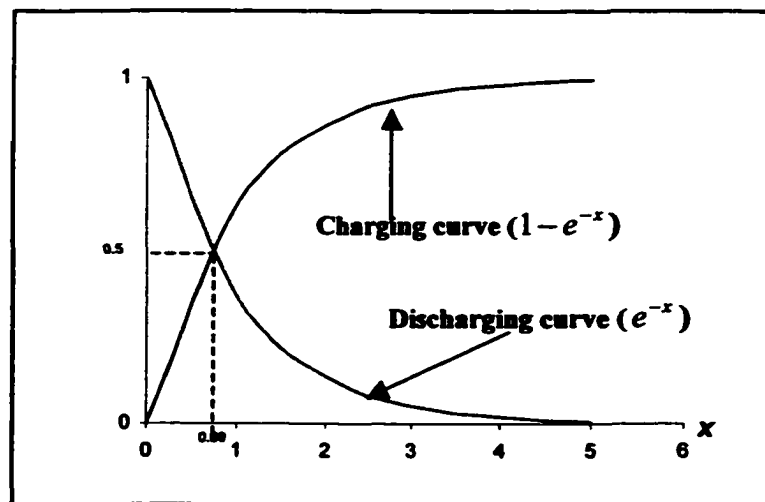
$$V_{out}(t) = (1 - e^{-t/\tau})V.$$

In the discharging phase, the voltage across the capacitor would be the following:

$$V_{out}(t) = e^{-t/\tau}V.$$

It takes five time constants to approximately fully charge/discharge a capacitor [Boy99].

Considering charging phase, the factor  $(1 - e^{-t/\tau})$  is exponential function of the form  $(1 - e^{-x})$ , where  $x = t/\tau$  and  $e = 2.71828\dots$ . A plot of  $(1 - e^{-x})$  for  $x \geq 0$  appears in Figure A.4. The time to reach the 50% point, the propagation delay ( $t_p$ ), is  $0.69\tau$ .



**Figure A.4:** Charging and discharging exponential curves for an  $RC$  network.

A simple first-order derivation of the  $t_p$  is given by [CB95, RCN01]

$$t_p = 0.69C_L R \approx \frac{C_L V_{DD}}{I} = \frac{C_L V_{DD}}{k(W/L)(V_{DD} - V_T)^2},$$

where  $C_L$  is the gate capacitance,  $V_{DD}$  is the supply voltage,  $V_T$  is the threshold voltage,  $k$  is a technology-dependent parameter, and  $W$  and  $L$  are the channel width and length of the transistors, respectively. Clearly, the circuit delay is a function of supply

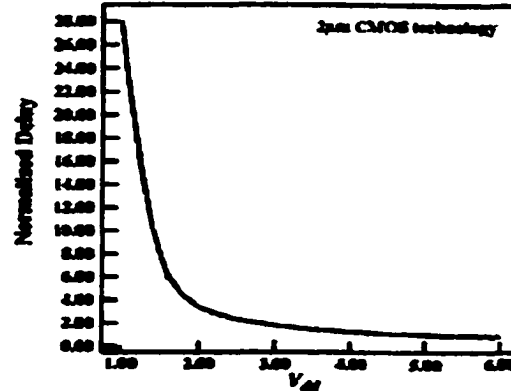


Figure A.5: The plot of the delay vs.  $V_{DD}$ .

voltage. Figure A.5 shows the plot of the circuit delay versus supply voltage and the figure suggests that there is the monotonic dependency of the propagation delay versus supply voltage. As the supply voltage is reduced, the delay of CMOS circuits increases monotonically.

Note that is the  $t_p$  an artificial gate quality metric, which is used to compare different semiconductor technologies or logic design styles [RCN01]. The  $t_p$  of a gate defines how quickly it responds to an input signal when passing through the gate.

## Power Consumption due to Switching

As the previous discussion, the power consumption in a CMOS circuit is from the dynamic source due to switching that can be calculated from the product of the current flows and voltage different. Considering a CMOS inverter, when the  $P$ -type transistor is charging a capacitance,  $C_{eff}$ , at a frequency,  $f = 1/T$ , with supply voltage,  $V_{DD}$ , the current through the transistor is  $C_{eff}(dV/dt)$ . The power consumption is thus  $C_{eff}V(dV/dt)$  for one-half the period of the input,  $1/2f$ . The power consumed at the  $P$ -Type can be calculated by the equation

$$\frac{1}{T} \int_0^{1/2f} C_{eff} V \left( \frac{dV}{dt} \right) dt = \frac{C_{eff}}{T} \int_0^{V_{DD}} V dV = \frac{1}{2T} C_{eff} V_{DD}^2$$

When  $N$ -type transistor discharges the capacitor, the power dissipation is equal, and making the total power consumption [CB95, WE93]

$$P_{switching} = C_{eff} V_{DD}^2 f$$

Let  $p_f$  be the switching activity per one clock cycle and  $C_L$  be the amount of load capacitance switched. Thus,  $C_{eff} = p_f C_L$ . The average dynamic power consumption of a CMOS gate due to the switching current is equal to

$$P_{switching} = p_f C_L V_{DD}^2 f . \quad (1)$$

Since the time integral of power is energy ( $\Delta E$ ), it follows that

$$\Delta E = P_{switching} \times T$$

As an example of finding power consumption of the circuit, the following plot shows the waveforms of the CMOS inverter, simulated by using the circuit simulator called PSpice (see Figure A.6). From top to bottom, the waveforms represent the inverter input voltage, the inverter output voltage and the inverter dynamic power consumption. CMOS is very power efficient because it consumes power during the brief periods of switching (see Figure A.6). We can conclude that the power consumption of CMOS logic is directly proportional to switching frequency. Figure A.7 shows the inverter's energy. The time integral of power is energy that can be computed as follow.

$$P_{switching} = \int_0^T IV_{DD} dt / T.$$

Thus, the power consumed by the inverter in Figure A.7 is equal to  $\frac{8.817u}{2ms} \left( \frac{\Delta E}{T} \right)$  watts.

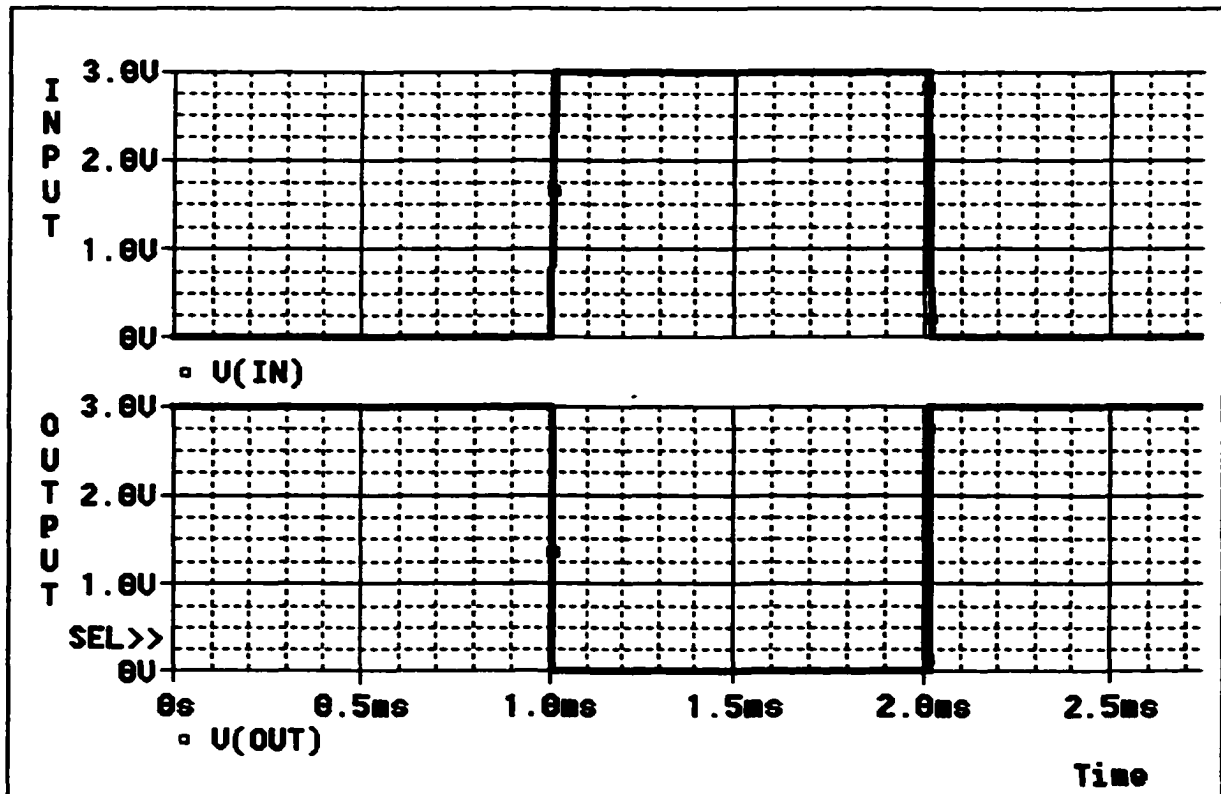


Figure A.6: CMOS inverter's input and output waveforms.

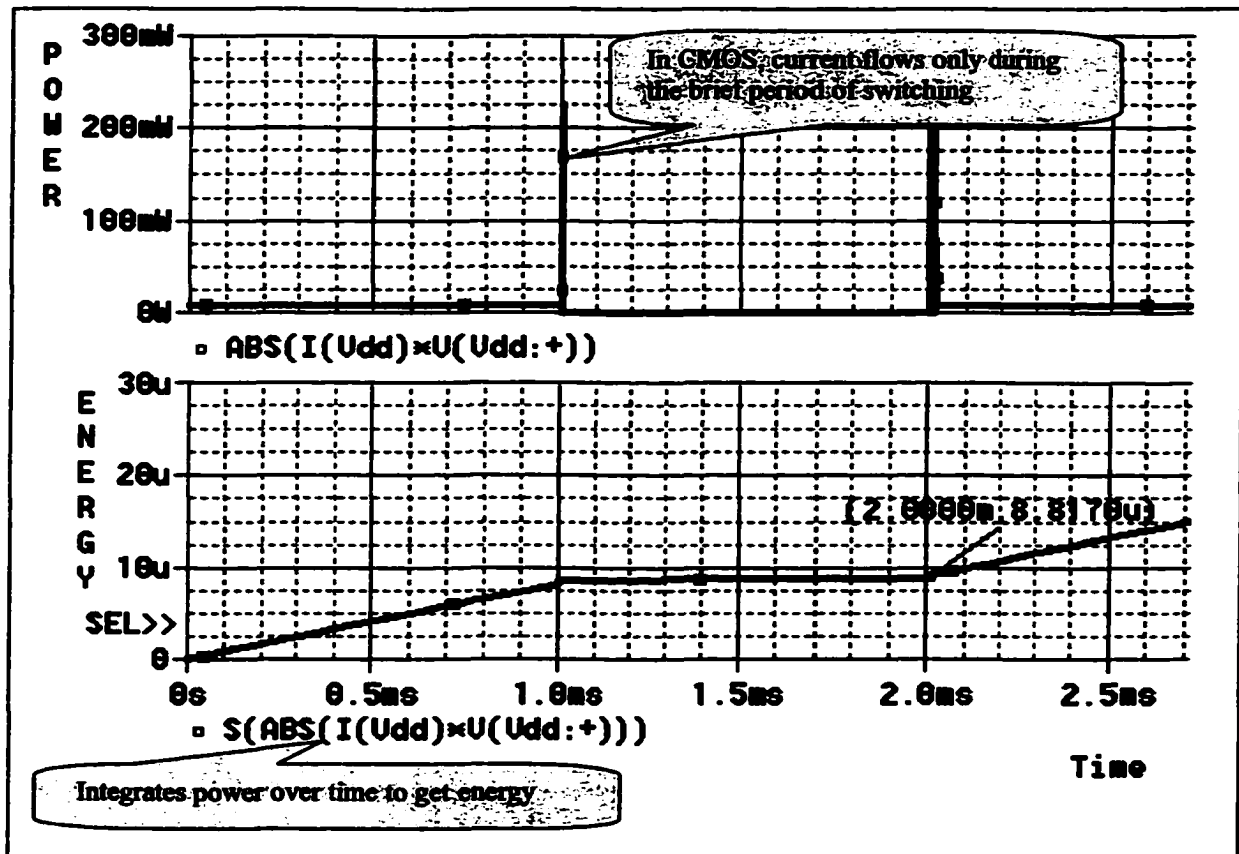


Figure A.7: CMOS inverter's power and energy waveforms.



# APPENDIX B

## CMOS Gates

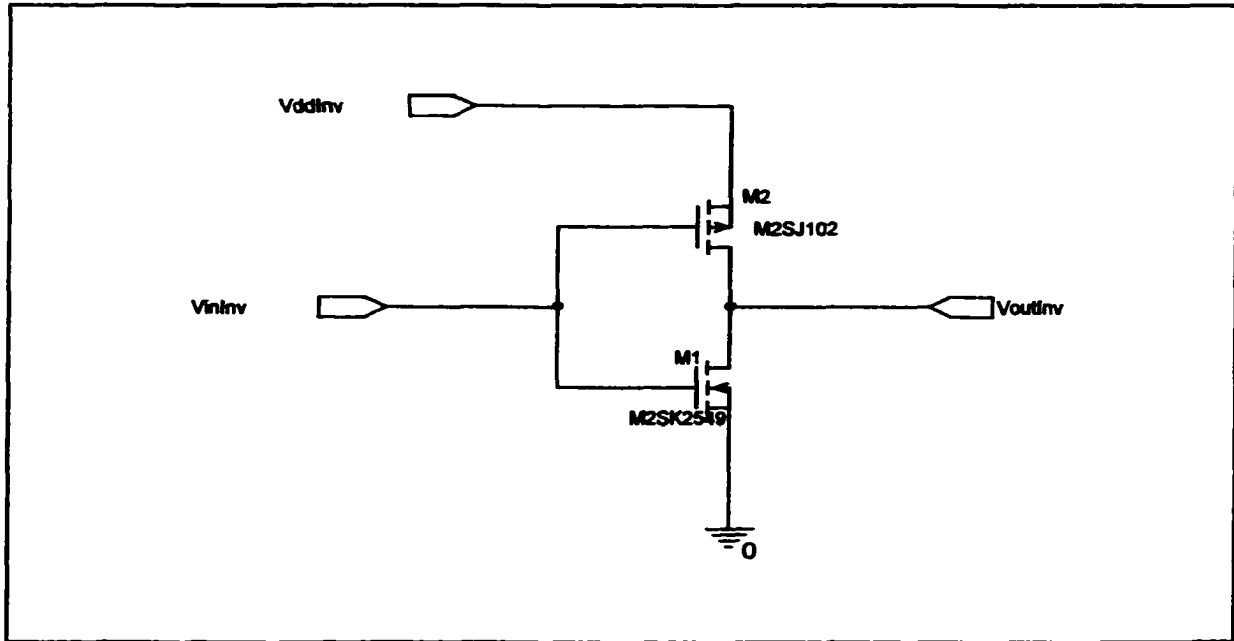


Figure B.1: A CMOS inverter

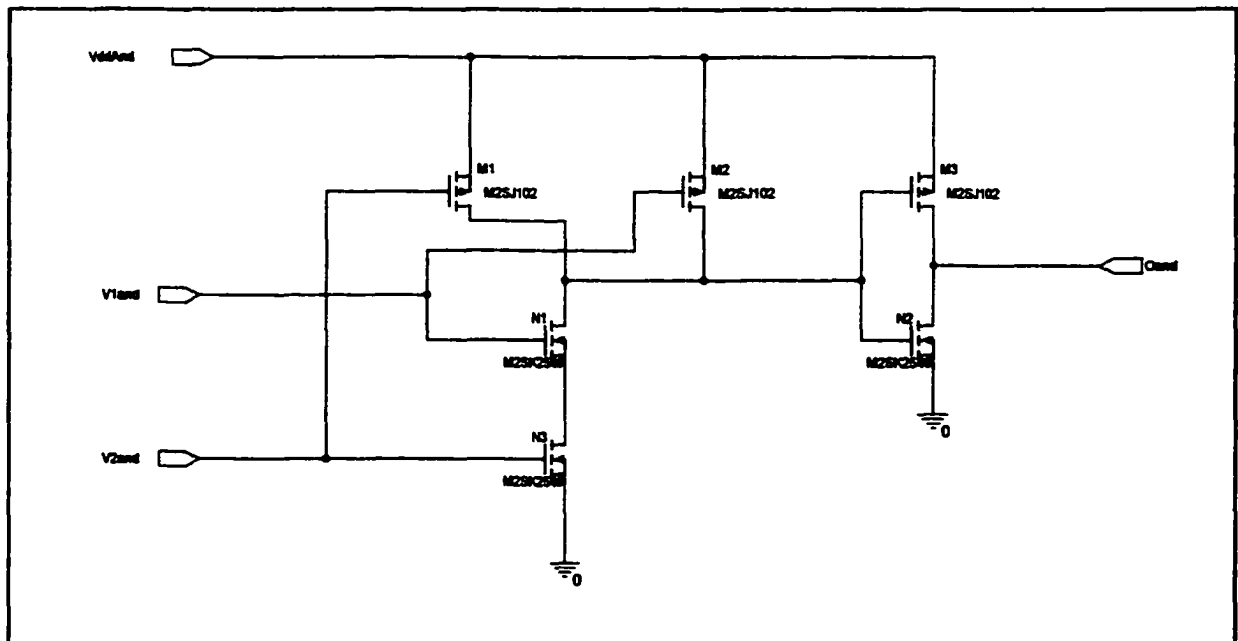
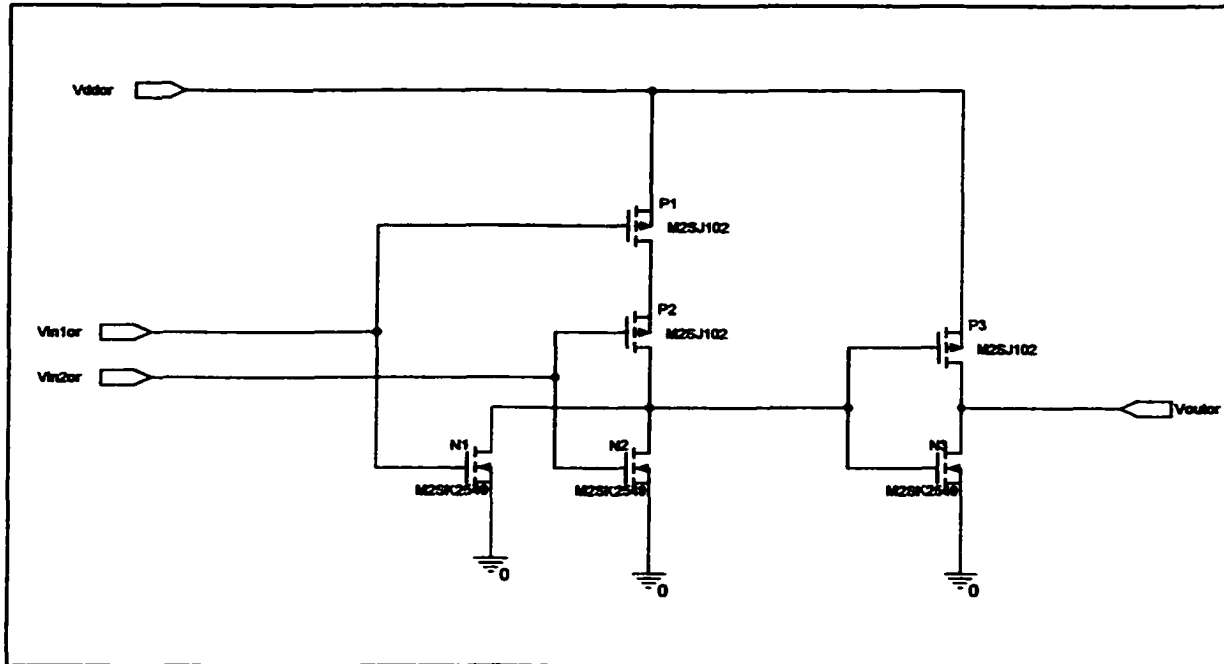
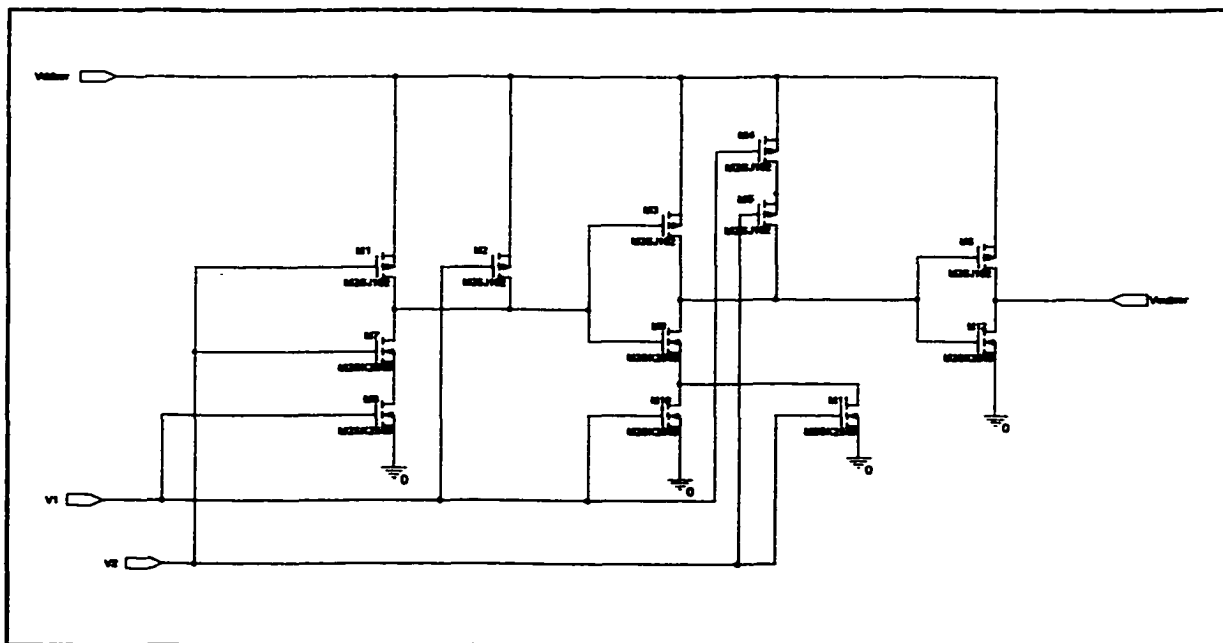


Figure B.2: A CMOS AND gate



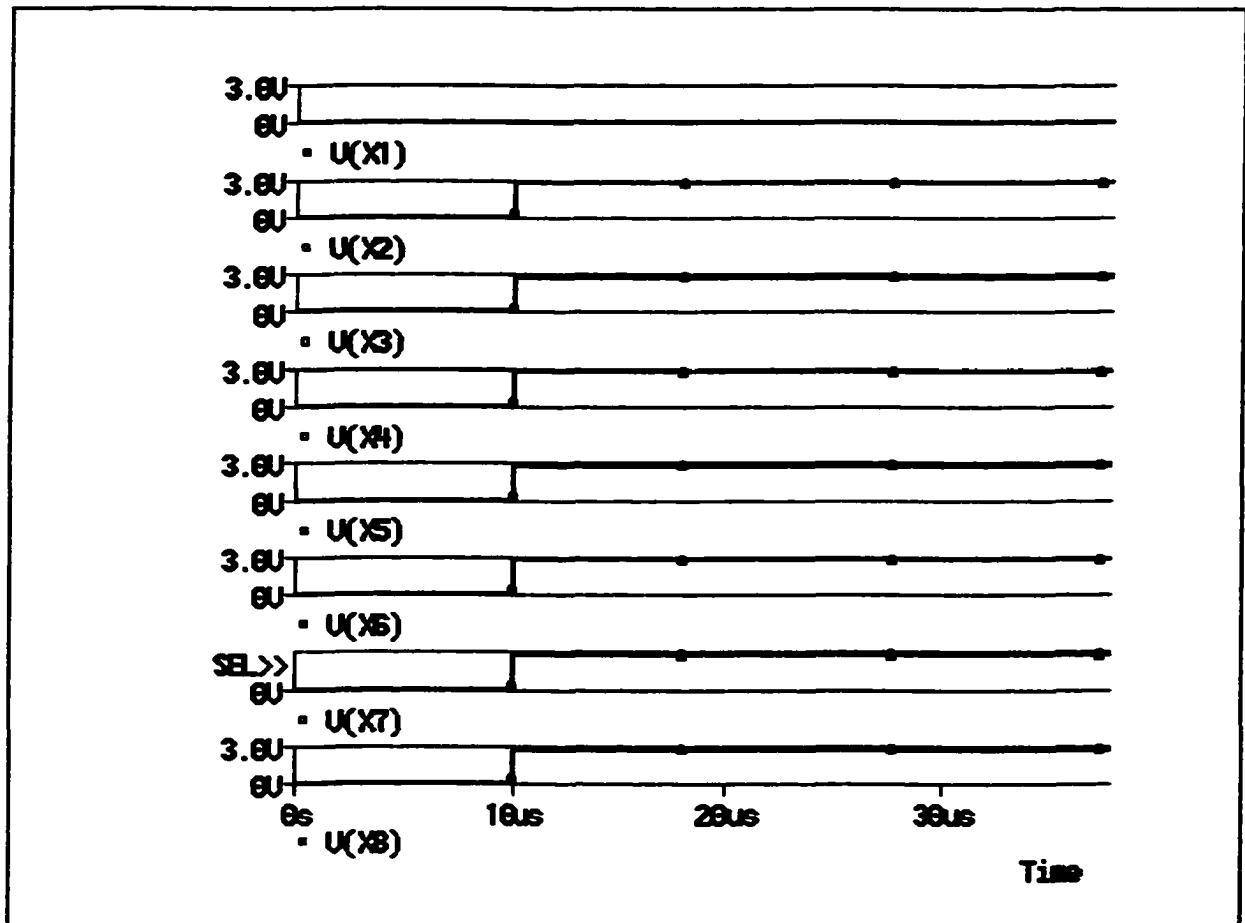
**Figure B.3: A CMOS OR gate**



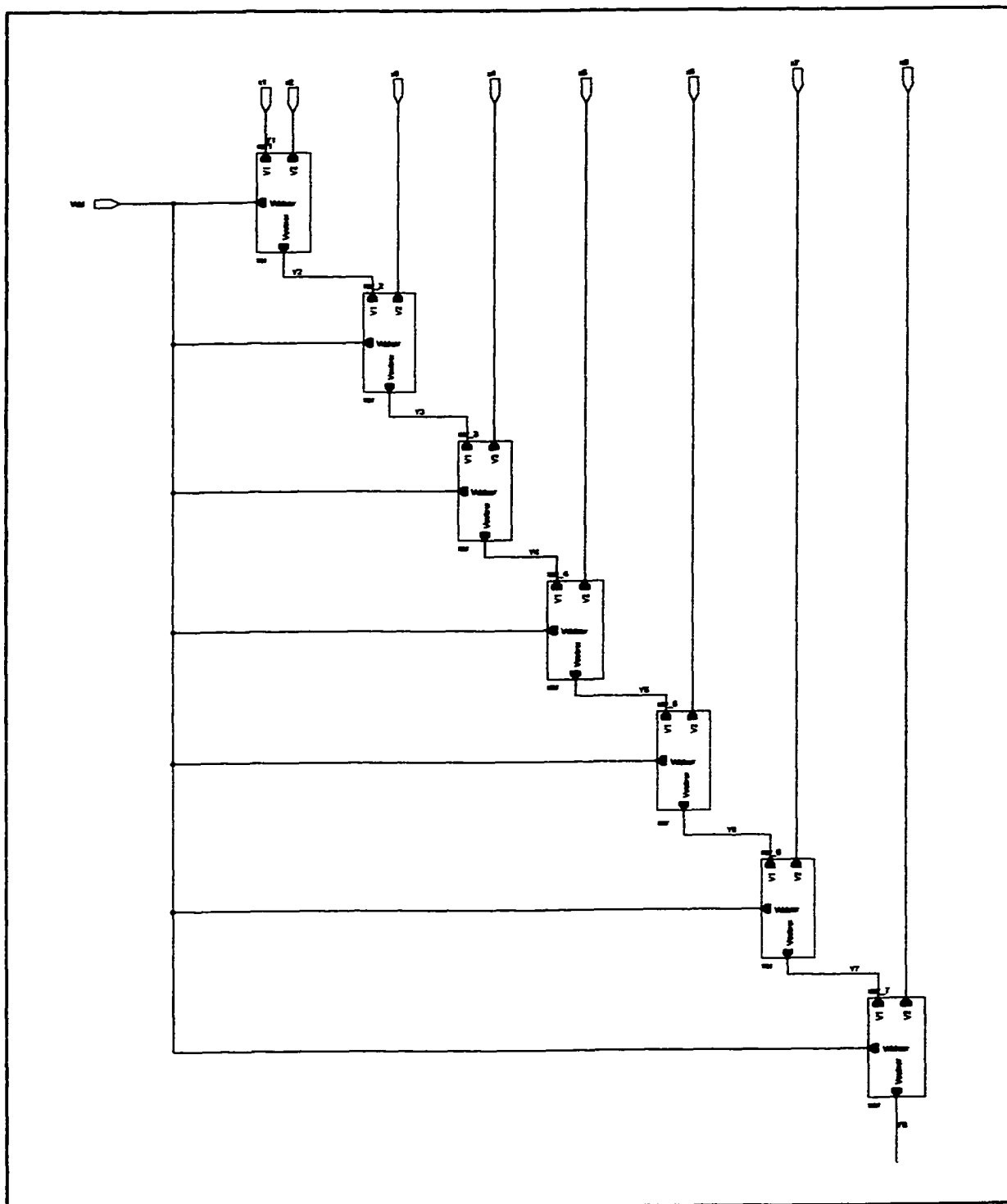
**Figure B.4: A CMOS XOR gate**

## APPENDIX C

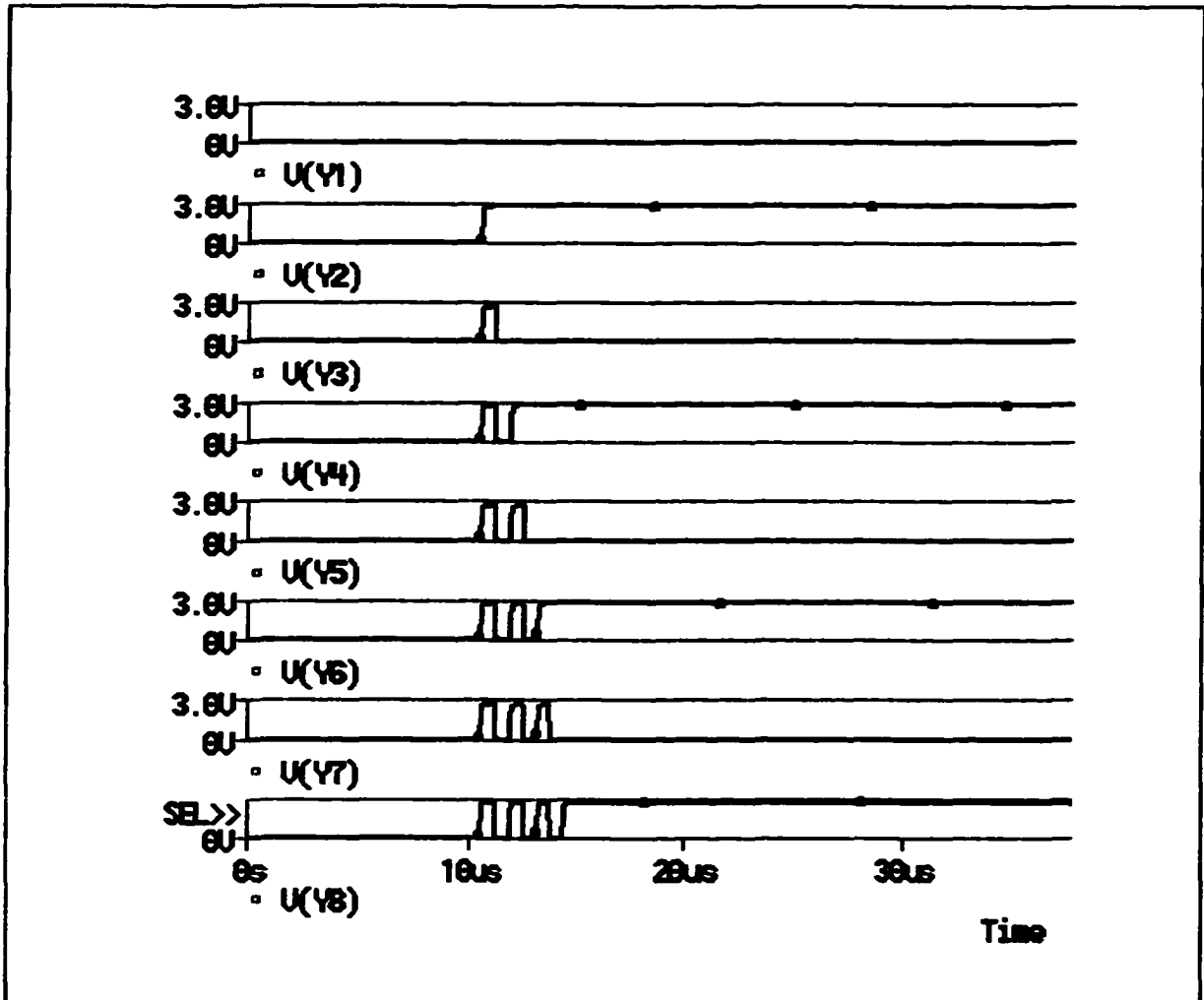
Due to the size, the examples of the 8-bit XOR gates implemented are shown.



**Figure C.1:** The worst case input of XOR gate (i.e., the first input is equal to 0 and the other inputs are  $0 \rightarrow 1$ ), causing the output to ripple the most.



**Figure C.2: The 8-bit XOR gate implemented with the serial prefix circuit.**



**Figure C.3:** The outputs of 8-bit XOR gates implemented with the serial prefix circuit, showing the longest ripple (the maximum number of switching).

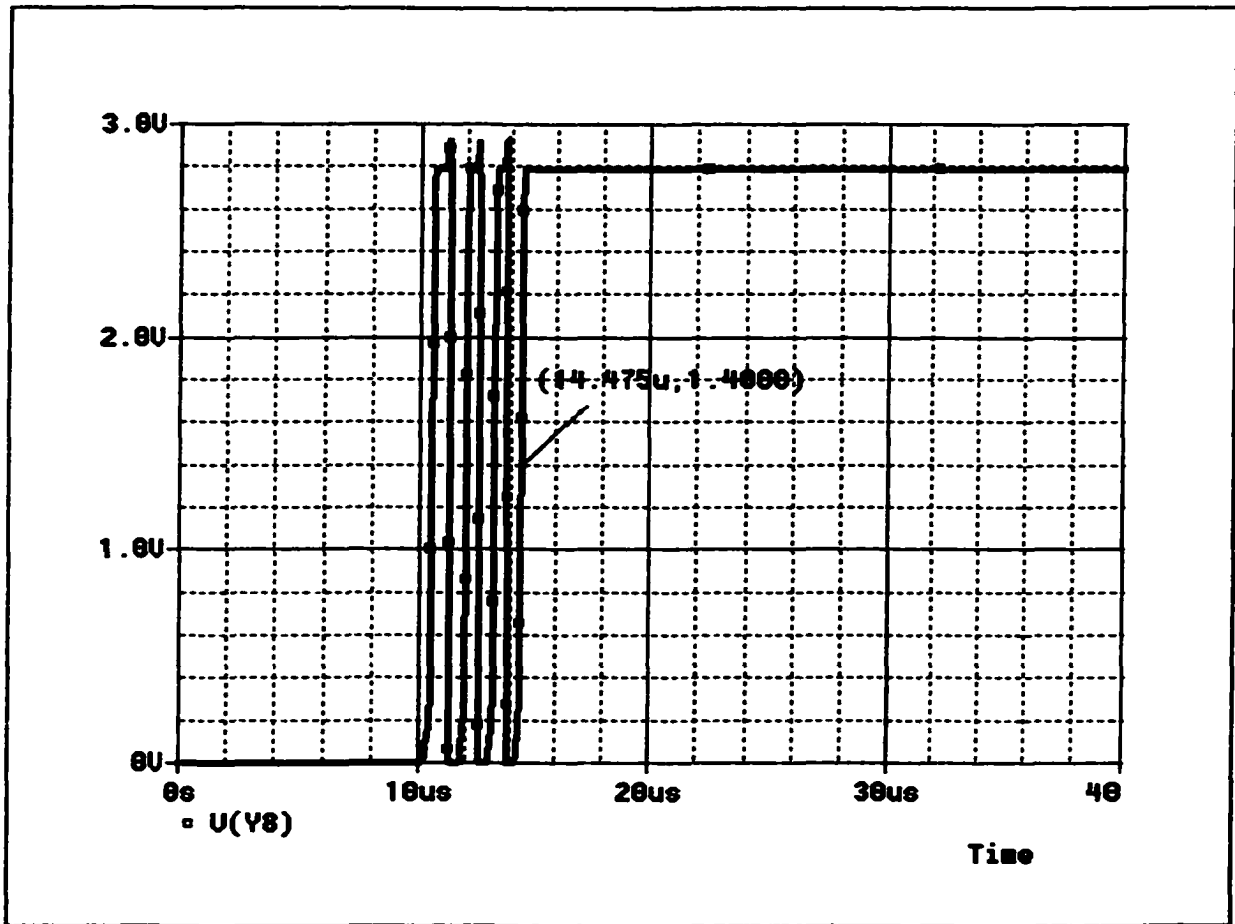


Figure C.4: Delay of 8-bit XOR gates implemented with the serial prefix circuit from PSpice simulation.

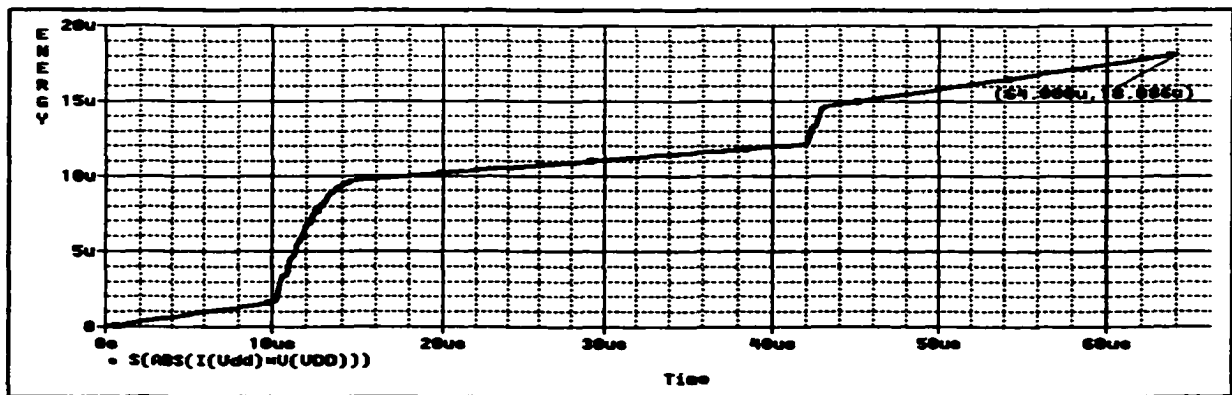
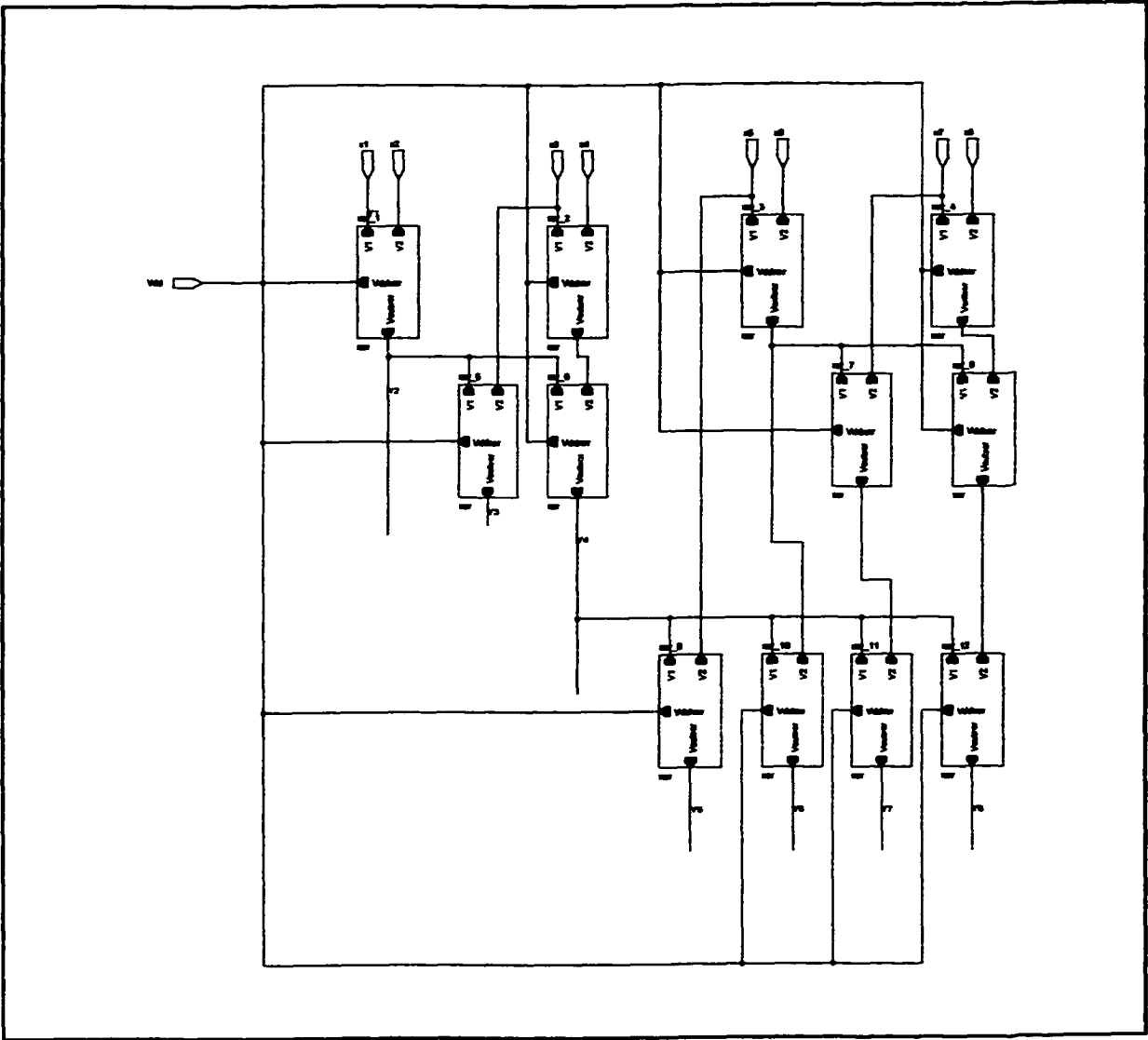


Figure C.5: Energy of 8-bit XOR gates implemented with the serial prefix circuit from PSpice simulation.



**Figure C.6:** The 8-bit XOR gate implemented with the divide-and-conquer prefix circuit.

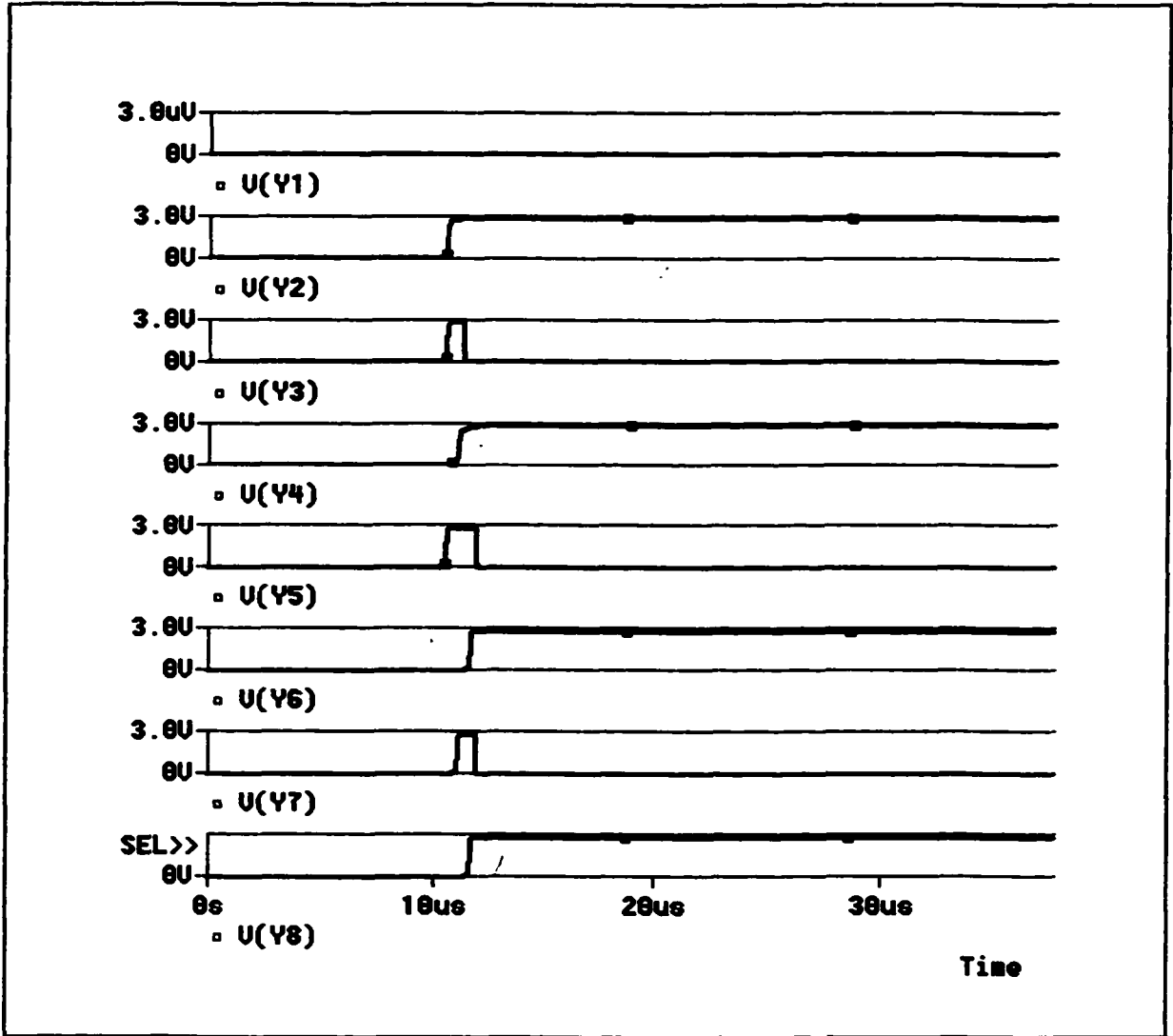
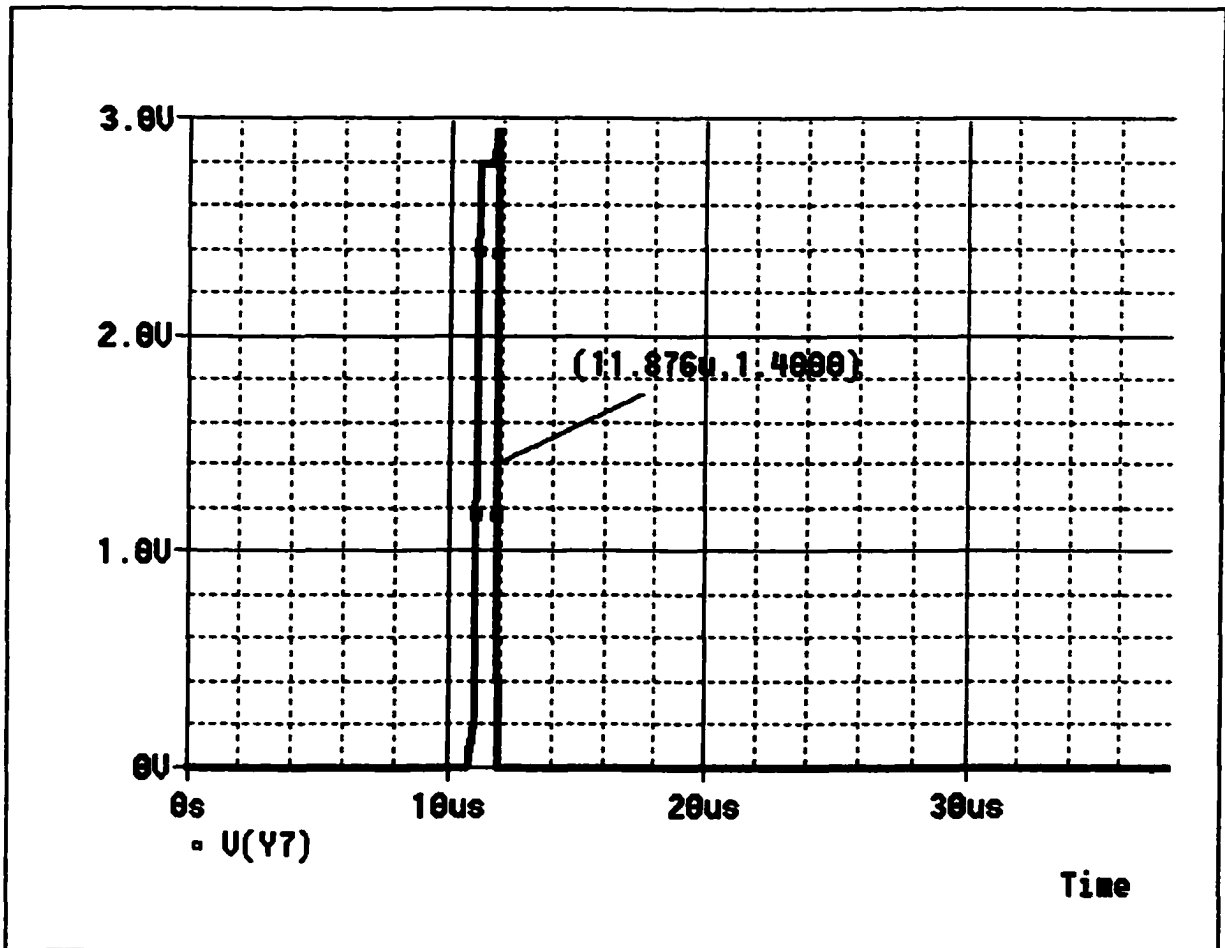
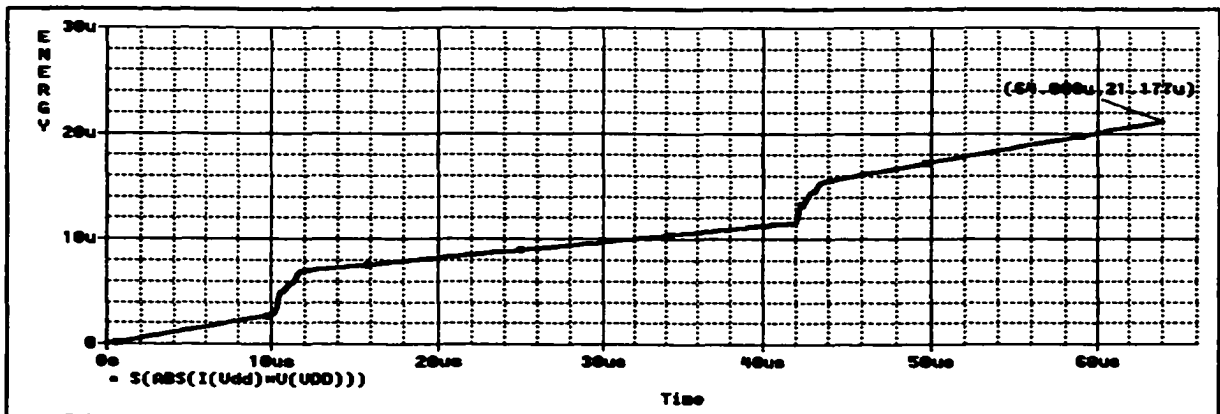


Figure C.7: The outputs of 8-bit XOR gates implemented with the divide-and-conquer prefix circuit.





**Figure C.8:** Delay of 8-bit XOR gates implemented with the divide-and-conquer prefix circuit from PSpice simulation.



**Figure C.9:** Energy of 8-bit XOR gates implemented with the divide-and-conquer prefix circuit from PSpice simulation.

## APPENDIX D

The implementation of a prefix adder is divided into 3 parts; preprocessing, carry computation, and postprocessing. Preprocessing and postprocessing are shown first. Then carry computation with different block sizes (i.e., 4 possible ways) is shown. Due to the big size, the 8-bit prefix adder implementations are shown.

### Preprocessing and Postprocessing

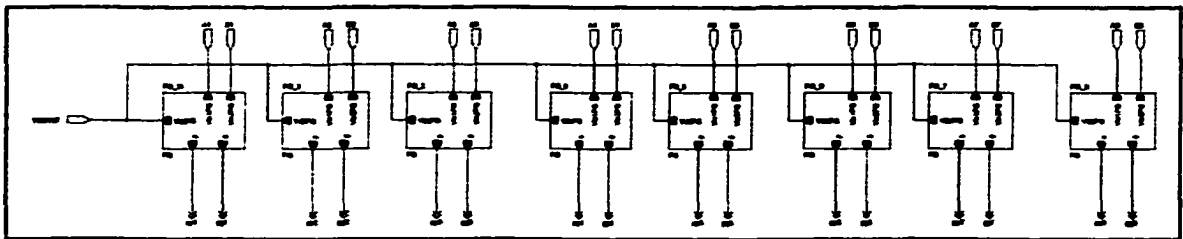


Figure D.1: Preprocessing: carry propagate bits and carry generate bits

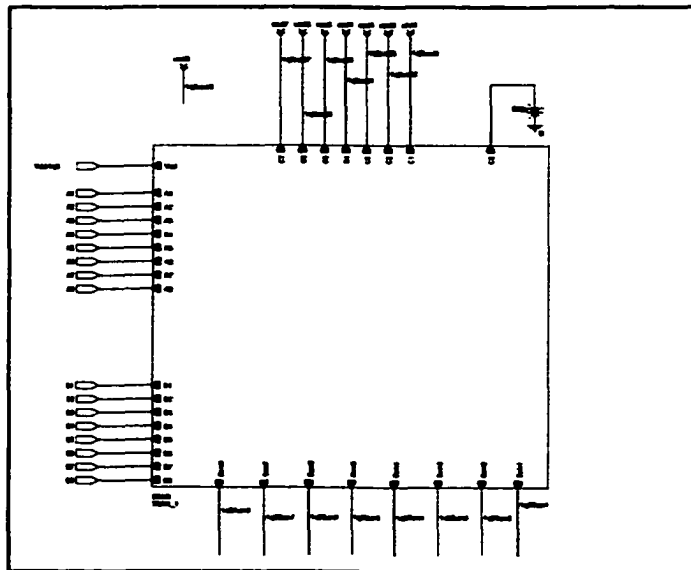


Figure D.2: Postprocessing:  $s_i = a_i \oplus b_i \oplus c_{i-1}$ .

# Carry Computation

## 1. *RIQ8*

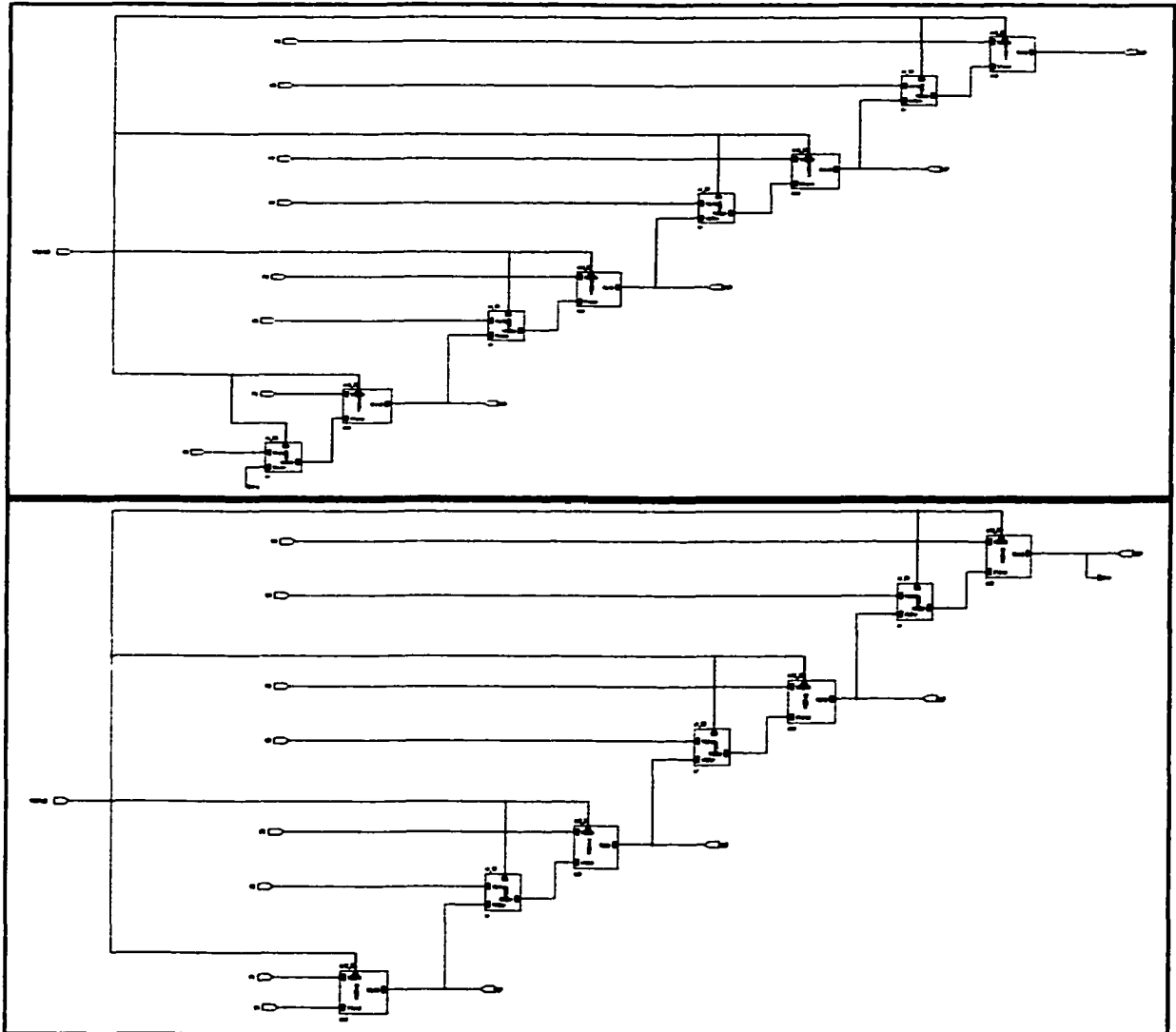


Figure D.3: The implementation of  $E_i$ ,  $i = 1$ .

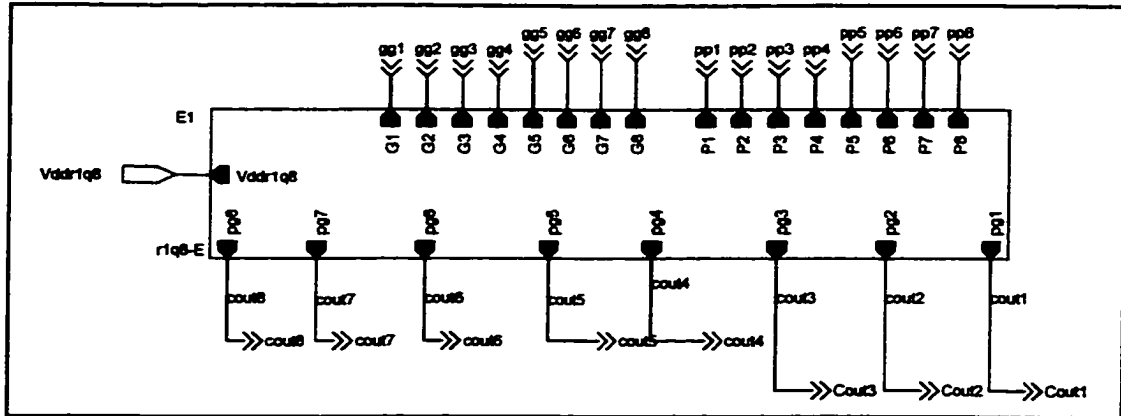


Figure D.4: The implementation of carry bits.

## 2. R2Q4

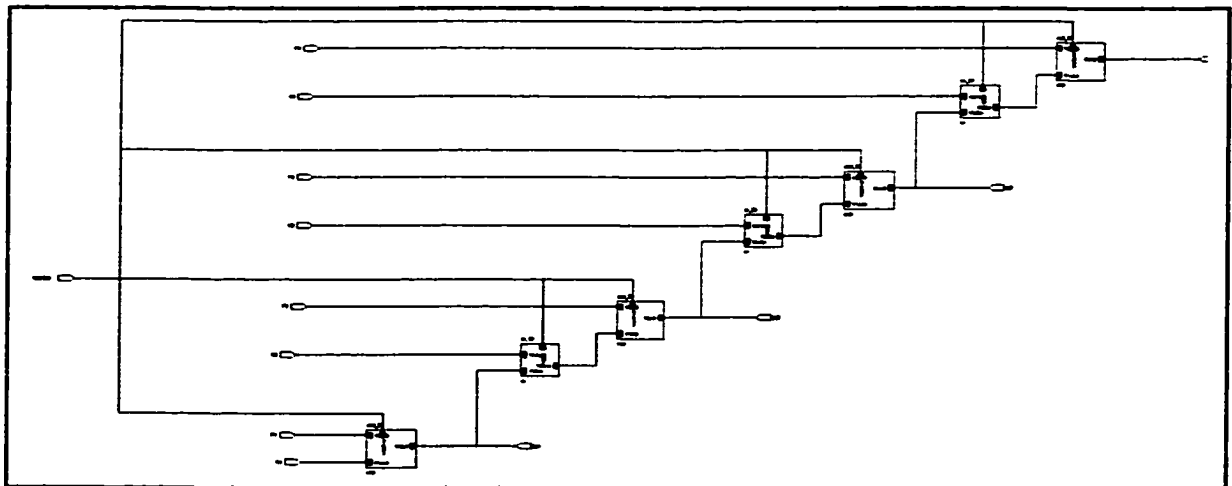
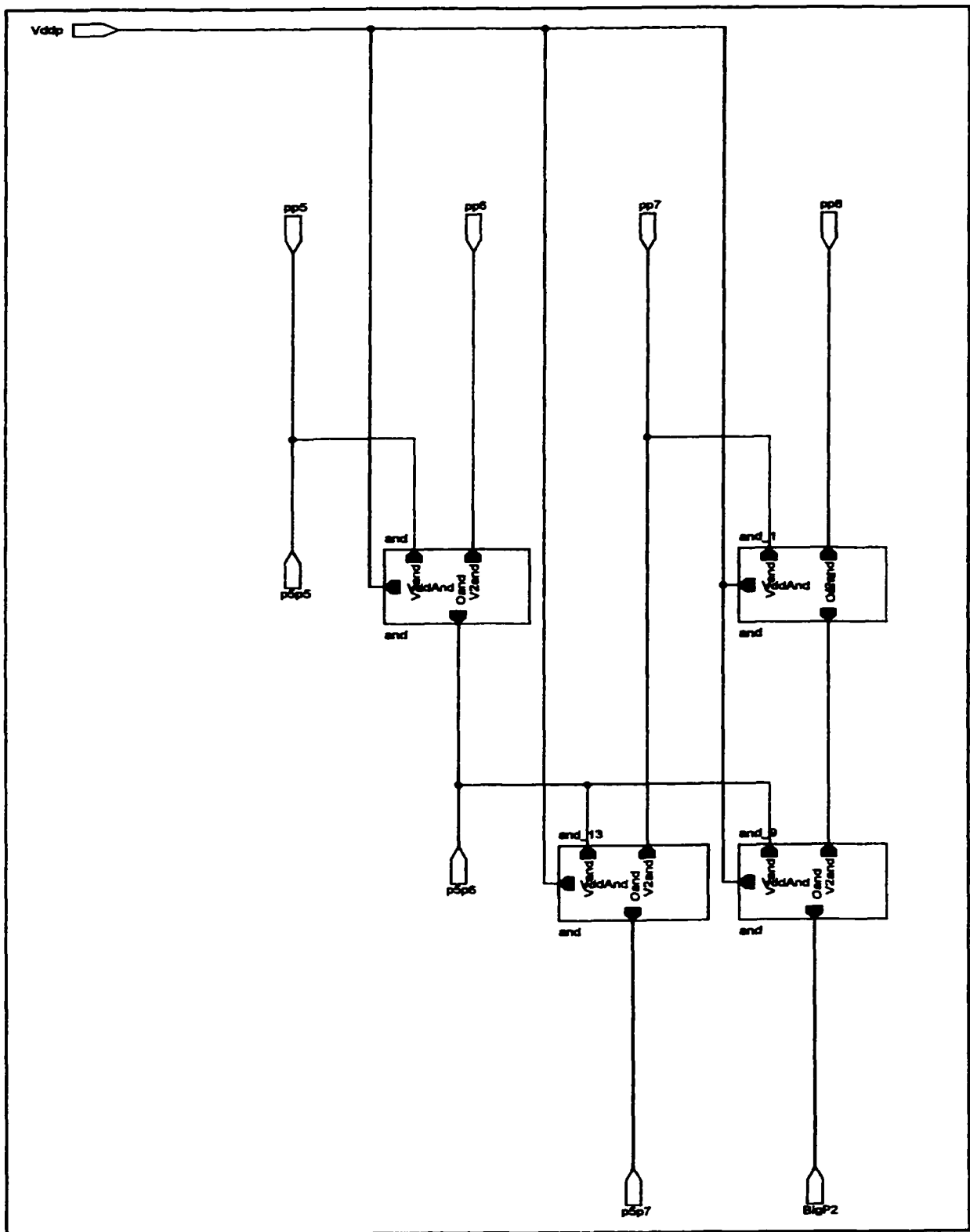
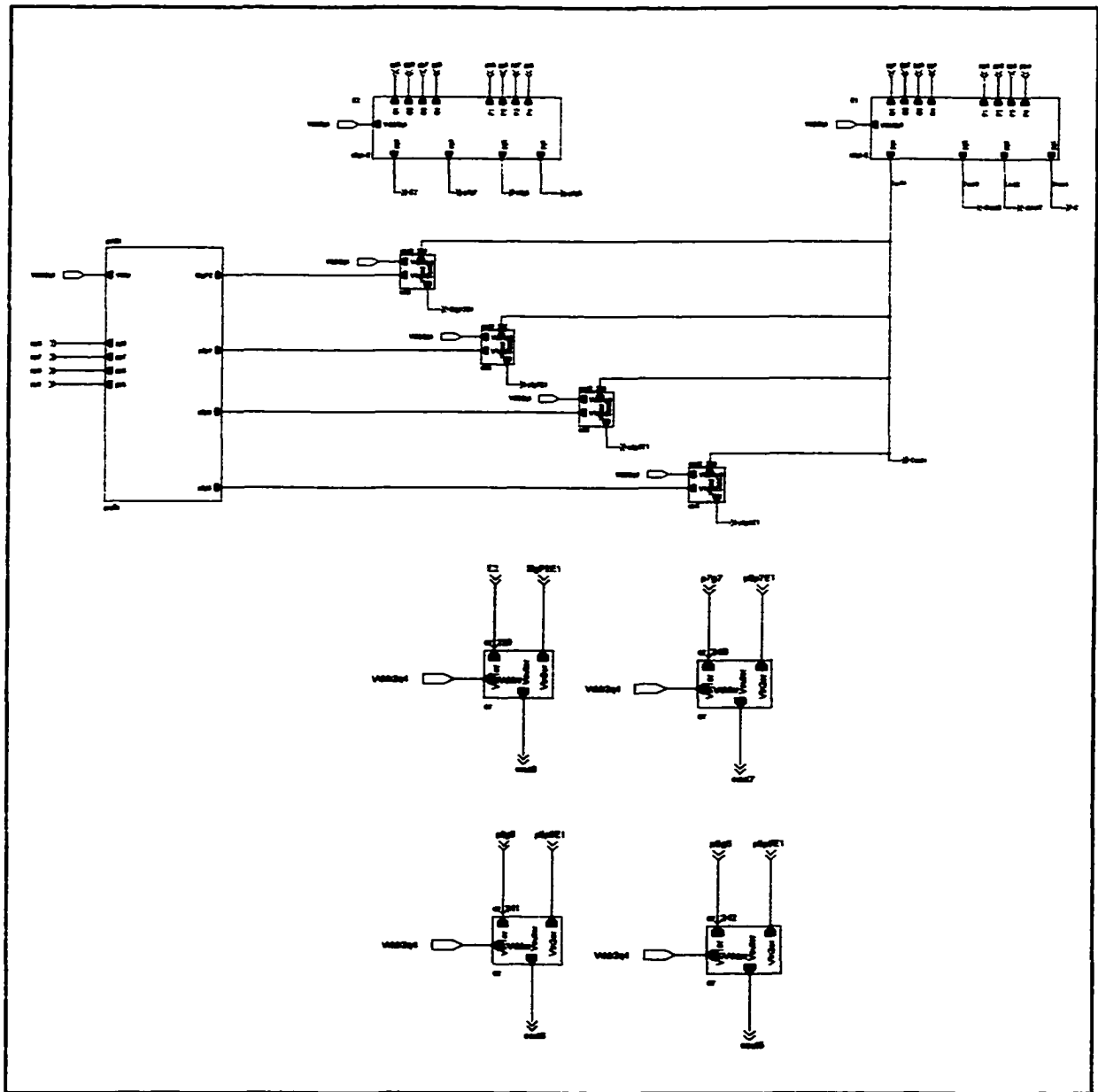


Figure D.5: The implementation of  $E_i, 1 \leq i \leq 2$ .



**Figure D.6:** The implementation of prefix circuit with 4 inputs.



**Figure D.7: The implementation of carry bits.**

### 3. R4Q2

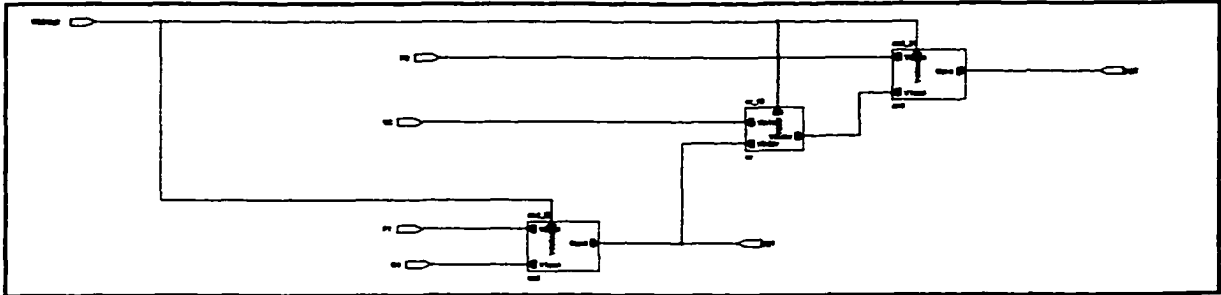


Figure D.8: The implementation of  $E_i$ ,  $1 \leq i \leq 4$ .

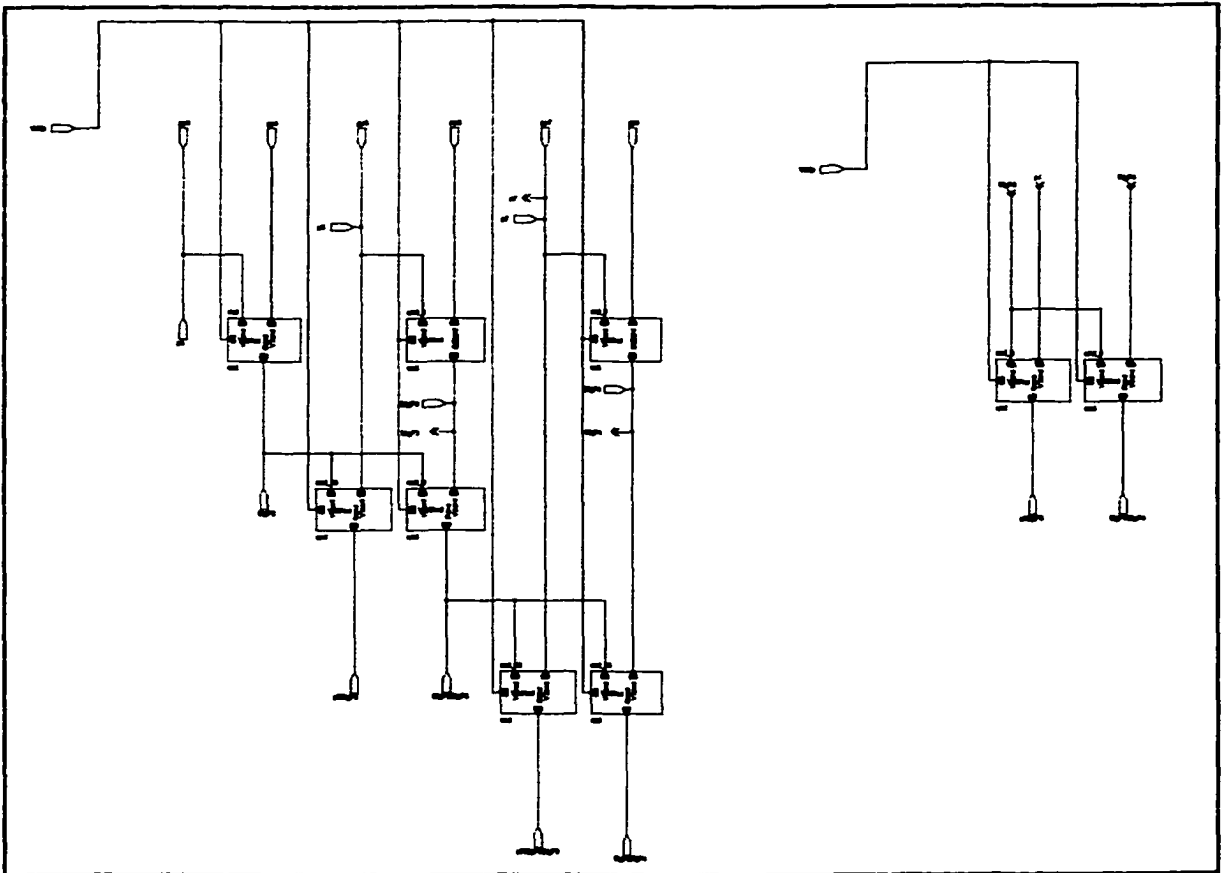
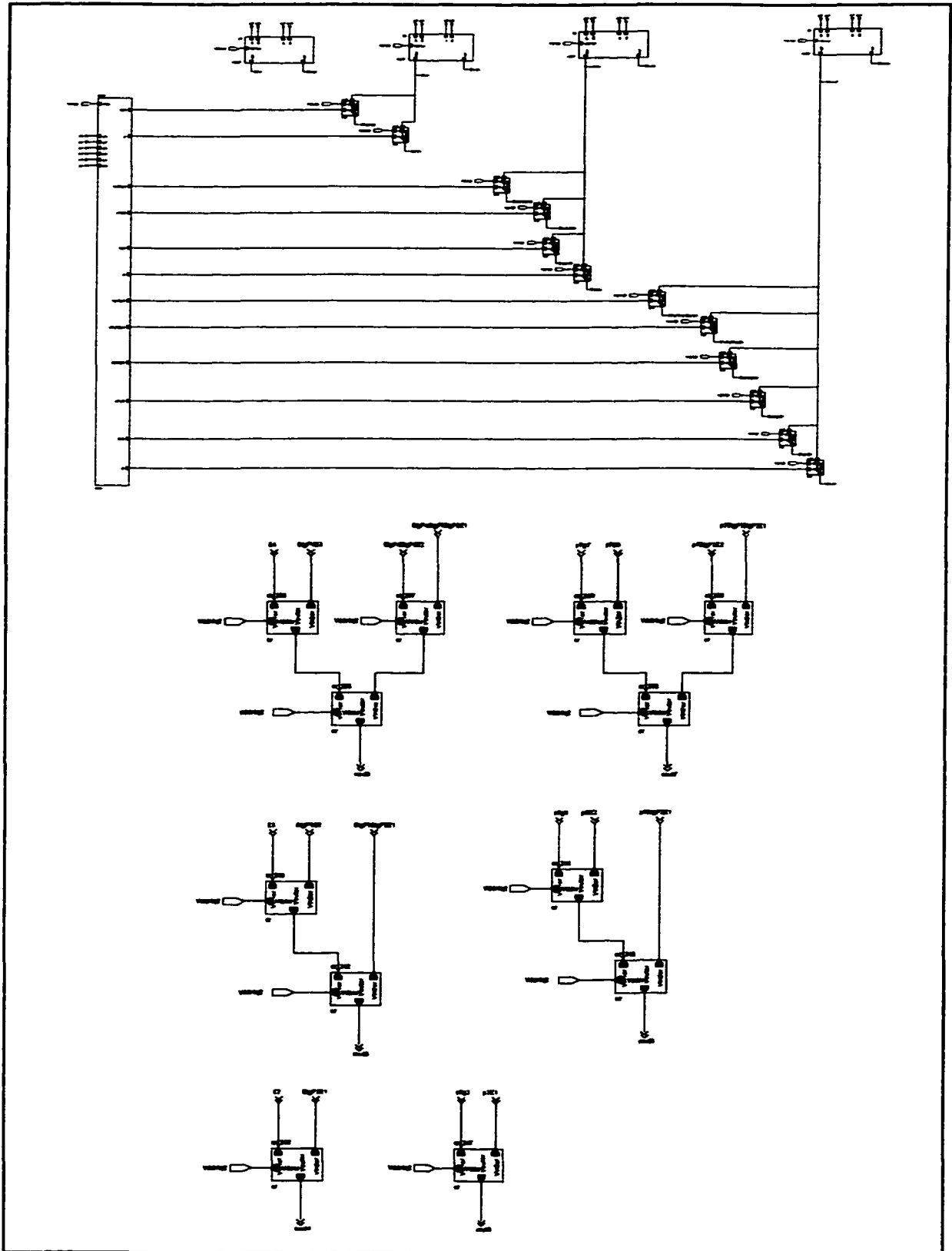


Figure D.9: The implementation of prefix circuit.



**Figure D.10: The implementation of carry bits.**



#### 4. RIQ8

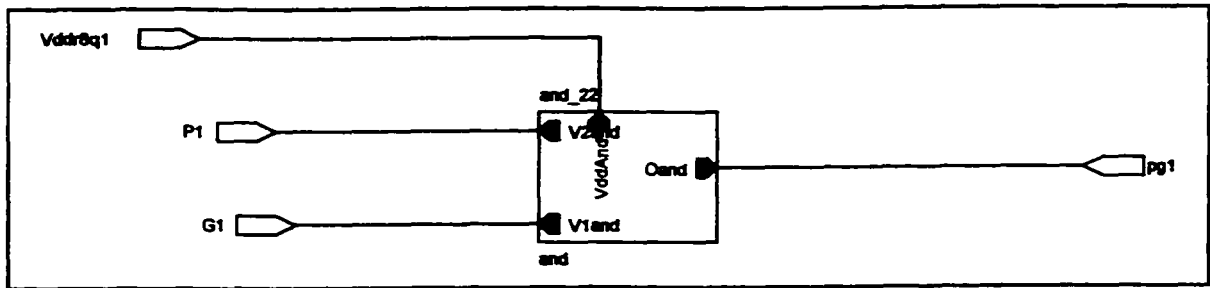


Figure D.11: The implementation of  $E_i$ ,  $1 \leq i \leq 8$ .

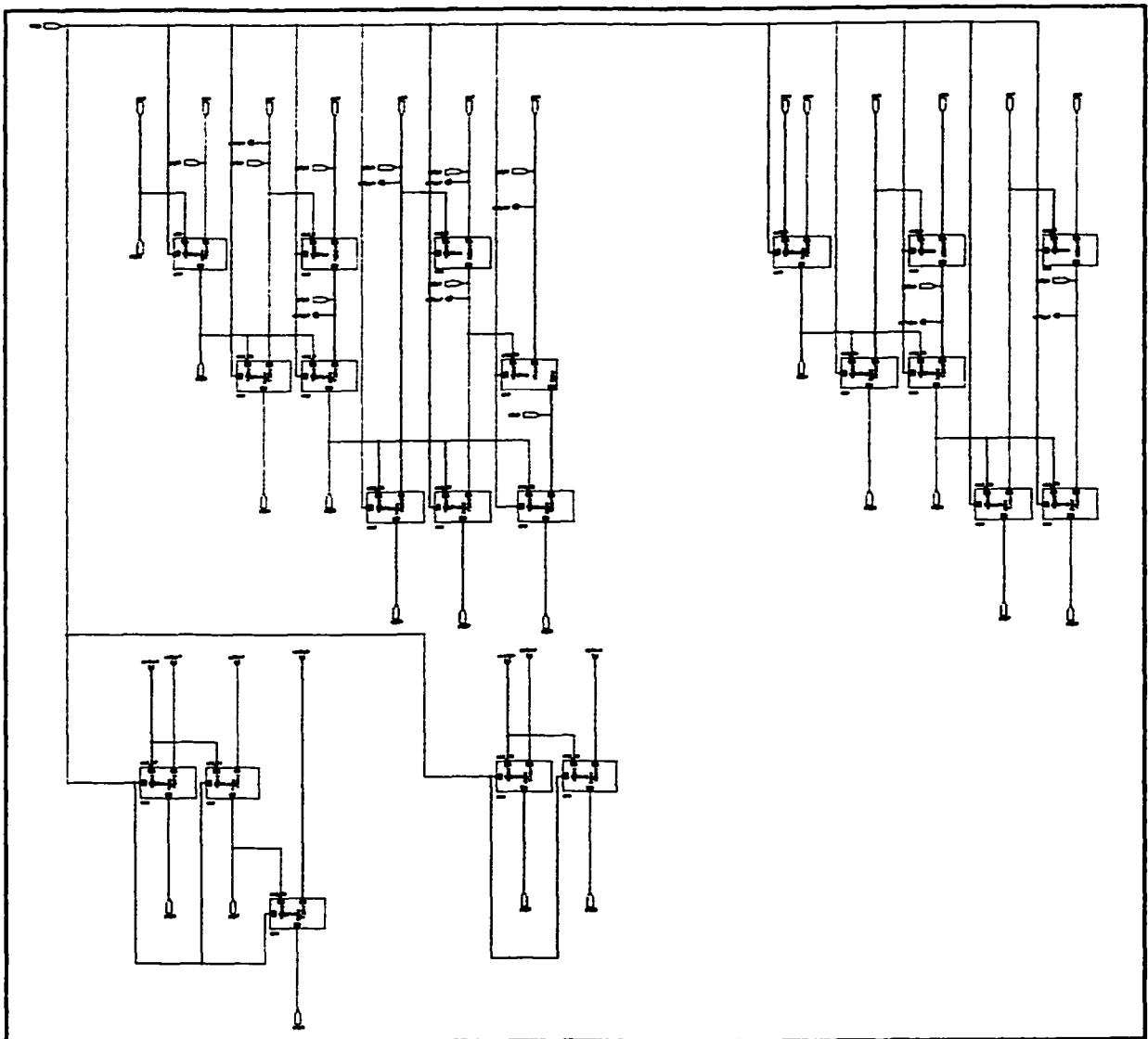


Figure D.12: The implementation of prefix circuit.

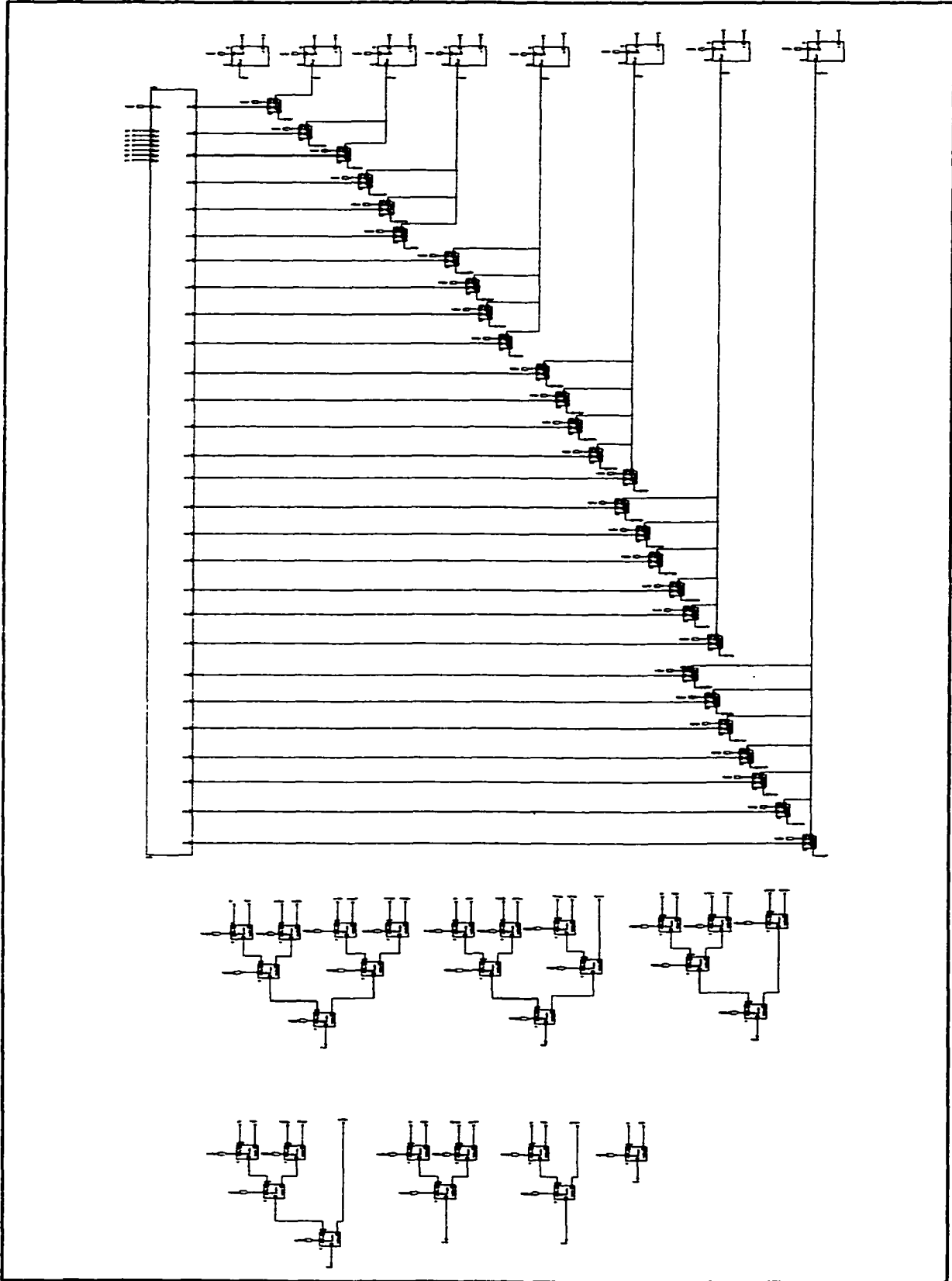


Figure D.13: The implementation of carry bits.

## APPENDIX E

**Table E.1:** A comparison of the exact capacitance values and the estimated capacitance values of the Brent-Kung prefix circuit.

| <b>N</b> | <b>Estimate(<math>C_0</math>)</b> | <b>Exact(<math>C_0</math>)</b> | <b>%Error</b> |
|----------|-----------------------------------|--------------------------------|---------------|
| 2        | 1                                 | 1                              | 0%            |
| 3        | 3.0838                            | 3                              | 2.79%         |
| 4        | 6                                 | 6                              | 0%            |
| 5        | 9.5578                            | 9                              | 6.20%         |
| 6        | 13.6312                           | 13                             | 4.86%         |
| 7        | 18.1329                           | 18                             | 0.74%         |
| 8        | 23                                | 23                             | 0.00%         |
| 9        | 28.1848                           | 27                             | 4.39%         |
| 10       | 33.6504                           | 32                             | 5.16%         |
| 11       | 39.3671                           | 38                             | 3.60%         |
| 12       | 45.3109                           | 44                             | 2.98%         |
| 13       | 51.4617                           | 51                             | 0.91%         |
| 14       | 57.8028                           | 57                             | 1.41%         |
| 15       | 64.3197                           | 64                             | 0.50%         |
| 16       | 71                                | 71                             | 0%            |
| 17       | 77.8329                           | 76                             | 2.41%         |
| 18       | 84.8089                           | 82                             | 3.43%         |
| 19       | 91.9195                           | 89                             | 3.28%         |
| 20       | 99.1573                           | 96                             | 3.29%         |
| 21       | 106.5156                          | 104                            | 2.42%         |
| 22       | 113.9883                          | 111                            | 2.69%         |
| 23       | 121.5698                          | 119                            | 2.16%         |
| 24       | 129.2552                          | 127                            | 1.78%         |
| 25       | 137.0400                          | 136                            | 0.76%         |
| 26       | 144.9199                          | 143                            | 1.34%         |
| 27       | 152.8910                          | 151                            | 1.25%         |
| 28       | 160.9499                          | 159                            | 1.23%         |
| 29       | 169.0932                          | 168                            | 0.65%         |
| 30       | 177.3178                          | 176                            | 0.75%         |
| 31       | 185.6210                          | 185                            | 0.34%         |
| 32       | 194                               | 194                            | 0%            |
| 33       | 202.4524                          | 200                            | 1.23%         |
| 34       | 210.9757                          | 207                            | 1.92%         |
| 35       | 219.5679                          | 215                            | 2.12%         |
| 36       | 228.2269                          | 223                            | 2.34%         |
| 37       | 236.9507                          | 232                            | 2.13%         |
| 38       | 245.7375                          | 240                            | 2.39%         |

| <b>N</b> | <b>Estimate(C<sub>0</sub>)</b> | <b>Exact(C<sub>0</sub>)</b> | <b>%Error</b> |
|----------|--------------------------------|-----------------------------|---------------|
| 39       | 254.5856                       | 249                         | 2.24%         |
| 40       | 263.4933                       | 258                         | 2.13%         |
| 41       | 272.4590                       | 268                         | 1.66%         |
| 42       | 281.4813                       | 276                         | 1.99%         |
| 43       | 290.5588                       | 285                         | 1.95%         |
| 44       | 299.6901                       | 294                         | 1.94%         |
| 45       | 308.8739                       | 304                         | 1.60%         |
| 46       | 318.1091                       | 313                         | 1.63%         |
| 47       | 327.3945                       | 323                         | 1.36%         |
| 48       | 336.7289                       | 333                         | 1.12%         |
| 49       | 346.1113                       | 344                         | 0.61%         |
| 50       | 355.5407                       | 352                         | 1.01%         |
| 51       | 365.0161                       | 361                         | 1.11%         |
| 52       | 374.5366                       | 370                         | 1.23%         |
| 53       | 384.1012                       | 380                         | 1.08%         |
| 54       | 393.7091                       | 389                         | 1.21%         |
| 55       | 403.3594                       | 399                         | 1.09%         |
| 56       | 413.0515                       | 409                         | 0.99%         |
| 57       | 422.7843                       | 420                         | 0.66%         |
| 58       | 432.5574                       | 429                         | 0.83%         |
| 59       | 442.3698                       | 439                         | 0.77%         |
| 60       | 452.2210                       | 449                         | 0.72%         |
| 61       | 462.1103                       | 460                         | 0.46%         |
| 62       | 472.0369                       | 470                         | 0.43%         |
| 63       | 482.0004                       | 481                         | 0.21%         |
| 64       | 492                            | 492                         | 0%            |
| 65       | 502.0352208                    | 499                         | 0.61%         |
| 66       | 512.1054706                    | 507                         | 1.01%         |
| 67       | 522.2102                       | 516                         | 1.20%         |
| 68       | 532.3488765                    | 525                         | 1.40%         |
| 69       | 542.5209835                    | 535                         | 1.41%         |
| 70       | 552.7260201                    | 544                         | 1.60%         |
| 71       | 562.9634999                    | 554                         | 1.62%         |
| 72       | 573.2329504                    | 564                         | 1.64%         |
| 73       | 583.5339129                    | 575                         | 1.48%         |
| 74       | 593.8659414                    | 584                         | 1.69%         |
| 75       | 604.2286022                    | 594                         | 1.72%         |
| 76       | 614.6214737                    | 604                         | 1.76%         |
| 77       | 625.0441454                    | 615                         | 1.63%         |
| 78       | 635.496218                     | 625                         | 1.68%         |
| 79       | 645.9773024                    | 636                         | 1.57%         |
| 80       | 656.4870199                    | 647                         | 1.47%         |
| 81       | 667.0250013                    | 659                         | 1.22%         |
| 82       | 677.5908868                    | 668                         | 1.44%         |
| 83       | 688.1843256                    | 678                         | 1.50%         |
| 84       | 698.8049755                    | 688                         | 1.57%         |
| 85       | 709.4525028                    | 699                         | 1.50%         |

| <b>N</b> | <b>Estimate(C<sub>0</sub>)</b> | <b>Exact(C<sub>0</sub>)</b> | <b>%Error</b> |
|----------|--------------------------------|-----------------------------|---------------|
| 86       | 720.1265816                    | 709                         | 1.57%         |
| 87       | 730.826894                     | 720                         | 1.50%         |
| 88       | 741.5531294                    | 731                         | 1.44%         |
| 89       | 752.3049846                    | 743                         | 1.25%         |
| 90       | 763.0821631                    | 753                         | 1.34%         |
| 91       | 773.8843755                    | 764                         | 1.29%         |
| 92       | 784.7113387                    | 775                         | 1.25%         |
| 93       | 795.5627758                    | 787                         | 1.09%         |
| 94       | 806.4384162                    | 798                         | 1.06%         |
| 95       | 817.337995                     | 810                         | 0.91%         |
| 96       | 828.2612533                    | 822                         | 0.76%         |
| 97       | 839.2079374                    | 835                         | 0.50%         |
| 98       | 850.177799                     | 844                         | 0.73%         |
| 99       | 861.1705952                    | 854                         | 0.84%         |
| 100      | 872.1860878                    | 864                         | 0.95%         |
| 101      | 883.2240438                    | 875                         | 0.94%         |
| 102      | 894.2842347                    | 885                         | 1.05%         |
| 103      | 905.3664365                    | 896                         | 1.05%         |
| 104      | 916.47043                      | 907                         | 1.04%         |
| 105      | 927.5959998                    | 919                         | 0.94%         |
| 106      | 938.7429352                    | 929                         | 1.05%         |
| 107      | 949.9110293                    | 940                         | 1.05%         |
| 108      | 961.100079                     | 951                         | 1.06%         |
| 109      | 972.3098854                    | 963                         | 0.97%         |
| 110      | 983.5402531                    | 974                         | 0.98%         |
| 111      | 994.7909903                    | 986                         | 0.89%         |
| 112      | 1006.061909                    | 998                         | 0.81%         |
| 113      | 1017.352824                    | 1011                        | 0.63%         |
| 114      | 1028.663554                    | 1021                        | 0.75%         |
| 115      | 1039.993922                    | 1032                        | 0.77%         |
| 116      | 1051.343751                    | 1043                        | 0.80%         |
| 117      | 1062.71287                     | 1055                        | 0.73%         |
| 118      | 1074.101111                    | 1066                        | 0.76%         |
| 119      | 1085.508306                    | 1078                        | 0.70%         |
| 120      | 1096.934293                    | 1090                        | 0.64%         |
| 121      | 1108.378912                    | 1103                        | 0.49%         |
| 122      | 1119.842004                    | 1114                        | 0.52%         |
| 123      | 1131.323415                    | 1126                        | 0.47%         |
| 124      | 1142.822992                    | 1138                        | 0.42%         |
| 125      | 1154.340586                    | 1151                        | 0.29%         |
| 126      | 1165.876048                    | 1163                        | 0.25%         |
| 127      | 1177.429234                    | 1176                        | 0.12%         |
| 128      | 1189                           | 1189                        | 0.00%         |