AN ABSTRACT OF THE THESIS OF

Parag Shantu Shah for the degree of Master of Science in Electrical and Computer Engineering presented on July 8, 1998. Title: Low-Power High-Performance 32-Bit 0.5μm CMOS Adder.

Abstract approved: 

Shih-Lien Lu

Currently, the two most critical factors of microprocessor design are performance and power. The optimum balance of these two factors is reflected in the speed-power product (SPP). 32-bit CMOS adders are used as representative circuits to investigate a method of reducing the SPP. The purpose of this thesis is to show that sizing gates according to fan-out and removing buffer drivers can reduce the SPP. This thesis presents a method for sizing gates in large fan-out parallel prefix circuits to reduce the SPP and compares it to other methods. Three different parallel prefix adders are used to compare propagation delay and SPP. The first adder uses the depth-optimal prefix circuit. The second adder is based on Wei, Thompson, and Chen’s time-optimal adder. The third adder uses a recursive doubling formation where all cells have minimum transistor width dimensions. The component cells in the adders are static CMOS as described by Brent and Kung. For all circuits, the smallest propagation delay occurs when the highest voltage supply is applied. The smallest SPP occurs when the lowest voltage supply is applied, but with the lowest performance. The Recursive Doubling Adder always has the lowest propagation delay for a particular set of parameters. However, its SPP is nearly equal to the Brent-Kung Adders and lower than Wei’s Adder. The power-frequency analysis reveals that a decrease in Vt causes higher power consumption due to leakage.
Low-Power High-Performance 32-Bit 0.5\(\mu\)m CMOS Adder

by

Parag Shantu Shah

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented July 8, 1998
Commencement June 1999
Master of Science thesis of Parag Shantu Shah presented on July 8, 1998

APPROVED:

Redacted for Privacy

Major Professor, representing Electrical and Computer Engineering

Redacted for Privacy

Head of Department of Electrical and Computer Engineering

Redacted for Privacy

Dean of Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Redacted for Privacy

Parag Shantu Shah, Author
ACKNOWLEDGEMENT

This thesis acknowledges the following people for their support and assistance:

Shih-Lien Lu
Çetin Kaya Koç
Roger Traylor
John F. Wager
David J. Griffiths
Isadore Netto
Vaishali Phatak
Deepika Sharma
Ravi Puri
Kysse Siala
TABLE OF CONTENTS

1. INTRODUCTION ........................................................................................................ 1

   1.1 Problem Definition......................................................................................... 1

   1.2 Statement of Purpose..................................................................................... 2

2. REVIEW OF LITERATURE .................................................................................... 3

   2.1 Parallel Prefix Computation ........................................................................ 3

   2.2 Circuit Formulations ..................................................................................... 7

   2.3 Propagation Delay ......................................................................................... 10

   2.4 Power Consumption ...................................................................................... 11

   2.5 Summary of Review ..................................................................................... 12

3. DESIGNS .................................................................................................................. 13

   3.1 Brent-Kung Adder ......................................................................................... 13

   3.2 Wei, Thompson, and Chen Adder .................................................................. 17

   3.3 Recursive Doubling Adder ............................................................................ 19

   3.4 Summary of Designs ..................................................................................... 21

4. EXPERIMENTAL RESULTS ................................................................................ 23

   4.1 Simulation Variations ................................................................................... 23

   4.2 Propagation Delay ......................................................................................... 24

   4.3 Speed-Power Product ................................................................................... 29

   4.4 Power Leakage .............................................................................................. 36

   4.5 Interconnect Wires ...................................................................................... 38
TABLE OF CONTENTS (Continued)

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.6 Experimental Results Summary</td>
<td>39</td>
</tr>
<tr>
<td>5. CONCLUSIONS</td>
<td>40</td>
</tr>
<tr>
<td>5.1 Summary</td>
<td>40</td>
</tr>
<tr>
<td>5.2 Future Research</td>
<td>41</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>42</td>
</tr>
<tr>
<td>APPENDICES</td>
<td>43</td>
</tr>
<tr>
<td>Appendix A HSPICE Command File</td>
<td>44</td>
</tr>
<tr>
<td>Appendix B Level 3 HSPICE Parameters</td>
<td>45</td>
</tr>
<tr>
<td>Appendix C HSPICE Adder Netlist</td>
<td>46</td>
</tr>
<tr>
<td>Appendix D Random Input Generator</td>
<td>57</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Sample Product Circuit</td>
<td>4</td>
</tr>
<tr>
<td>2.2</td>
<td>Recursive Construction of $P_0(n)$</td>
<td>5</td>
</tr>
<tr>
<td>2.3</td>
<td>Recursive Construction of $P_k(n)$, $k \geq 1$</td>
<td>5</td>
</tr>
<tr>
<td>2.4</td>
<td>Drivers Used in Recursive Construction of $R(n)$</td>
<td>6</td>
</tr>
<tr>
<td>2.5</td>
<td>Base $P_k(n)$ Circuits for $1 \leq n \leq 5$</td>
<td>7</td>
</tr>
<tr>
<td>2.6</td>
<td>Static CMOS Implementations of $G$ and $P$ Subcells (positive weighted inputs)</td>
<td>9</td>
</tr>
<tr>
<td>2.7</td>
<td>Static CMOS Implementations of $G$ and $P$ Subcells (negative weighted inputs)</td>
<td>9</td>
</tr>
<tr>
<td>3.1</td>
<td>Brent-Kung Adder</td>
<td>14</td>
</tr>
<tr>
<td>3.2</td>
<td>Wei’s Adder</td>
<td>18</td>
</tr>
<tr>
<td>3.3</td>
<td>Recursive Doubling Adder</td>
<td>20</td>
</tr>
<tr>
<td>4.1</td>
<td>Propagation Delay ($V_{dd} = 3.0$ Volts)</td>
<td>25</td>
</tr>
<tr>
<td>4.2</td>
<td>Propagation Delay ($V_{dd} = 2.5$ Volts)</td>
<td>26</td>
</tr>
<tr>
<td>4.3</td>
<td>Propagation Delay ($V_{dd} = 2.0$ Volts)</td>
<td>27</td>
</tr>
<tr>
<td>4.4</td>
<td>Propagation Delay ($V_{dd} = 1.5$ Volts)</td>
<td>28</td>
</tr>
<tr>
<td>4.5</td>
<td>Speed-Power Product ($V_{dd} = 3.0$ Volts)</td>
<td>30</td>
</tr>
<tr>
<td>4.6</td>
<td>Speed-Power Product ($V_{dd} = 2.5$ Volts)</td>
<td>31</td>
</tr>
<tr>
<td>4.7</td>
<td>Speed-Power Product ($V_{dd} = 2.0$ Volts)</td>
<td>32</td>
</tr>
<tr>
<td>4.8</td>
<td>Speed-Power Product ($V_{dd} = 1.5$ Volts)</td>
<td>33</td>
</tr>
<tr>
<td>4.9</td>
<td>Power Leakage Effects of Decreasing $V_t$</td>
<td>37</td>
</tr>
<tr>
<td>Table</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>3.1</td>
<td>Depth and Size Variations Between the Three Designs</td>
<td>22</td>
</tr>
<tr>
<td>4.1</td>
<td>Propagation Delay ( (V_{dd} = 3.0 \text{ Volts}) )</td>
<td>25</td>
</tr>
<tr>
<td>4.2</td>
<td>Propagation Delay ( (V_{dd} = 2.5 \text{ Volts}) )</td>
<td>26</td>
</tr>
<tr>
<td>4.3</td>
<td>Propagation Delay ( (V_{dd} = 2.0 \text{ Volts}) )</td>
<td>27</td>
</tr>
<tr>
<td>4.4</td>
<td>Propagation Delay ( (V_{dd} = 1.5 \text{ Volts}) )</td>
<td>28</td>
</tr>
<tr>
<td>4.5</td>
<td>Actual MOSFET Count</td>
<td>35</td>
</tr>
<tr>
<td>4.6</td>
<td>Effective MOSFET Count</td>
<td>35</td>
</tr>
<tr>
<td>4.7</td>
<td>Power Leakage Effects of Decreasing ( V_t )</td>
<td>38</td>
</tr>
<tr>
<td>4.8</td>
<td>Total Number of Interconnects Between CLA Nodes</td>
<td>39</td>
</tr>
</tbody>
</table>
DEDICATION

This thesis is dedicated to my loving Father, Mother, Bhai, and Bhabi.
1. INTRODUCTION

In microprocessors, the adder is one of the most frequently used hardware structures for arithmetic operations. The adder is used extensively by multiplication and division algorithms. Besides arithmetic instructions, it is also used for address calculations. If the adder is not designed correctly, it can be a major bottleneck to a microprocessor. A frequently used, improperly optimized, adder will consume high amounts of power in comparison with other logic.

1.1 Problem Definition

There are many different ways of designing an adder both at the gate (or logic) level and the transistor (or circuit) level. At the gate level, different implementations for an adder include the carry-look-ahead adder, conditional sum adder, carry-skip adder, and the carry-save adder [4]. Some implementations at the circuit level include the static CMOS full adder, mirror adder, dynamic adder, and Manchester carry-chain adder [5].

Previously, the speed or area of an adder was the primary concern, with little attention being paid to power consumption. To create a high performance adder, the speed-power product (SPP) must be minimized, while reducing area. The SPP is equal to
the product of the adder’s propagation delay and the average power consumption. This thesis presents a method to produce a static CMOS circuit that has an optimal SPP.

1.2 Statement of Purpose

The purpose of this thesis is to show that by sizing gates according to fan-out and removing buffer drivers then the SPP can be reduced. Adder circuits are used to test the transistor sizing method. The adder scheme used is the carry look-ahead adder. The main research began from Wei, Thompson, and Chen’s “Time-Optimal Design of a CMOS Adder.” [3] Their paper focused only on finding the fastest adder. This thesis examines different designs, and finds the SPP-optimal design. This thesis presents circuit simulations done in HSPICE. The adders were stimulated with several hundred random numbers. A method of sizing the transistors similar to [3] was used. This thesis intends to show that this new method is more energy efficient.
2. REVIEW OF LITERATURE

Before looking at the simulations and results, base concepts related to addition in computer arithmetic must be reviewed. To meet today's high performance requirements, the primary adder used for many microprocessors is the carry look-ahead adder (CLA). In an n-bit adder, the sum value for the \( i^{th} \) bit is a function of the \( i^{th} \) bits of the two operands and the carry out from the previous bit position. In a CLA, the core of the logic calculates the carry values for each bit position in a parallel fashion. The parallel prefix computation uses an associative operation to calculate the carry values. We begin with a discussion of parallel prefix computation. Following the description of parallel prefix computation is the description of associative operation used in adders. The chapter concludes with a brief discussion of power consumption in CMOS digital design.

2.1 Parallel Prefix Computation

The prefix problem occurs when given a particular associative operation \((x_1 \cdot x_2 \cdot x_3 \cdot \ldots \cdot x_k)\) the product of \( k \) different numbers must be calculated. Ladner and Fisher presented a solution to the prefix problem in [1]. A recursive solution can solve the problem. The recursive solution from [1] allows multiple solutions that have different depths and sizes.

The parallel prefix computation circuit is represented by a directed acyclic graph where each node is a separate operation. An example of a graph where the depth is three
and the size is four is shown in Figure 2.1. It computes the end product $x_1 \cdot x_2 \cdot x_2 \cdot x_3 \cdot x_4$ along with some other sub-products such as $x_1 \cdot x_2$ and $x_2 \cdot x_3$.

![Sample Product Circuit](image)

**Figure 2.1 Sample Product Circuit**

The depth is the number of nodes on the graph in the longest path (from top to bottom). Depth affects the propagation delay of the circuit. The size is the total number of nodes in the graph. Size determines the area of the circuit.

$P_k(n)$ denotes the family of circuits that solves the prefix problem for $n$ inputs. The parameter $k$ ranges from 0 to $\lceil \log_2 n \rceil$. The depth is less than $k - \lceil \log_2 n \rceil$ and the size is bounded below $2(1 - 2^k)n - 4$. For $k=0$, the reverse construction of $P_0(n)$ is shown in Figure 2.2. For $k>0$, the reverse construction of $P_k(n)$ is shown in Figure 2.3. For the recursive solutions shown, exact solutions exist when $n$ is a power of two.

$S_0(n) = 4n - F(5 + \log_2 n) + 1,$

$S_1(n) = 3n - F(4 + \log_2 n),$ and

$S_k = 2(1 + 2^k)n - F(5 + \log_2 n - k) + 1 - k.$
Figure 2.2 Recursive Construction of $P_0(n)$

Figure 2.3 Recursive Construction of $P_k(n), k \geq 1$
The function $F(m)$ is the $m^{th}$ Fibonacci number. $F(m) = (\phi^m - \phi^{-m}) \sqrt{5}$, where $\phi = (1 + \sqrt{5})/2$ and $\phi^{-m} = (1 - \sqrt{5})/2$.

Wei, Thompson, and Chen introduced a new family of circuits, $R(n)$, where the blocks have an arbitrary size (Figure 2.4) instead of a reduction of half. [3] Therefore, $R(n)$ is a super set of $P_0(n)$. The most significant bit (MSB) of the right block (bit $m$) has a large fan-out. When the depth of $R(n - m)$ is larger than the depth of $R(m)$, a multi-stage driver is placed on the MSB of $R(m)$ as shown in Figure 2.4. The main objective for the new construction was to find the time-optimal adder. The multi-stage driver was used to solve the problem of delays associated with certain nodes having large fan-out.

![Figure 2.4 Drivers Used in Recursive Construction of $R(n)$](image-url)
In Figure 2.5, the construction of $P_k(n)$ for the basic graphs are shown, where $1 \leq n \leq 5$. The basic graphs shown are used to construct more complicated graphs for larger values of $n$.

![Figure 2.5 Base $P_k(n)$ Circuits for $1 \leq n \leq 5$](image)

2.2 Circuit Formulations

Now that the concept of parallel prefix computation has been reviewed, the formulations of the nodes in the graphs for an adder can be introduced. The typical method used to compute the sum $(s_n s_{n-1} \ldots s_0)$ of two $n$-bit operands $(a_n a_{n-1} \ldots a_0$ and $b_n b_{n-1} \ldots b_0$) is given by the equations
\[ c_0 = 0. \]
\[ c_i = a_i b_i + a_i c_{i-1} + b_i c_{i-1}, \text{ and} \]
\[ s_i = a_i \oplus b_i \oplus c_{i-1}. \]

The symbol \( \oplus \) is the exclusive-or operation, and \( c_i \) is the carry generated by the \( i^{th} \) bit position. The generate and propagate values \((g_i, p_i)\) are calculated for each bit position to offer an alternate way to determine \( c_i \). The equations are
\[ g_i = a_i b_i, \]
\[ p_i = a_i + b_i, \]
\[ c_0 = 0, \text{ and} \]
\[ c_i = g_i + p_i c_{i-1}. \]

The latter equation is based upon the fact that a carry is generated if both input bits are 1 (high), or if at least one of the bits is high and there is an incoming carry from the previous bit. The equations above can be grouped into an associative operation, \( \bullet \), such that
\[(g, p) \bullet (g', p') = (g + p g', p p'), \]
\[(G_i, P_i) = (g_i, p_i) \text{ if } i = 1, \]
\[(G_i, P_i) = (g_i, p_i) \bullet (G_{i-1}, P_{i-1}) \text{ if } 2 \leq i \leq n, \text{ and} \]
\[ c_i = G_i, \]

where \( g, p, g', \) and \( p' \) are binary variables. The first equation represents the logic that each node in the prefix array evaluates. The left portion of the output is the \( G \) subcell and the right portion of the output is the \( P \) subcell. To make the nodes more efficient there are two types of circuits for each subcell, one with positive weighted inputs and one with negative weighted inputs. The static CMOS implementations of the various cells are shown in Figure 2.6 and Figure 2.7.
The subcells in Figure 2.6 take in positive signals and output complemented signals. The subcells in Figure 2.7 take in complemented signals and output positive signals. The P and G subcells produce the output signals $pout$ and $gout$, respectively. For
Typically, one uses minimum-length and minimum-width transistors for the pull-down transistors (NMOS). To produce a similar circuit response for the pull-up transistors (PMOS), minimum-length is used; however, the width needs to be increased by a factor of two. The minimum width for PMOS transistors is double the minimum width of NMOS transistors because the effective resistivity for PMOS is double the effective resistivity for NMOS. The higher resistivity slows response time, so the width must be increased to compensate. [5,8]

2.3 Propagation Delay

In the adder circuits, propagation delay is a function of fan-in, fan-out, and the depth. Here the main focus is propagation delay of the transistors with large loads. The propagation delay of a transistor is often calculated by using a simple first-order RC network. The resistance is dependent upon the width and length of the transistor’s n-channel (NMOS) or p-channel (PMOS). Increased widths decrease the resistivity of the device, while decreasing the width causes a larger resistance. The capacitance is dependent upon the output capacitance of the driving transistor and the input capacitance of the load transistors. Typically, the gate capacitance, $C_g$, dominates all other capacitances by a factor of five to ten. The total resistance multiplied by the total load capacitance is the time constant, $\tau$, of the network. The goal of minimum propagation
delay is to minimize $t$ by sizing transistors to provide minimum resistance and minimum capacitance. However, decreasing the width of a transistor increases resistance and decreases capacitance, while increasing its width reduces resistance and increases capacitance. [8]

2.4 Power Consumption

The power consumption in a circuit is just as critical as the propagation delay. To optimize SPP, both must be reduced. In a CMOS circuit, power consumption is affected by several factors which include dynamic switching, short-circuit current, and static leakage current.

Dynamic switching is the dominant factor of power consumption. A capacitor is either charged or discharged any time an input changes and an output node voltage changes. The power dissipated in doing so is given by $P_{\text{dyn}} = C_L V_{\text{dd}}^2 f$. The power is dependent upon the size of the capacitor, the voltage supply, and the frequency. Larger capacitors store more energy. If the output switches from 0 V to $V_{\text{dd}}$, then one or more PMOS transistors will dissipate energy while the output is charged up to $V_{\text{dd}}$. If the output switches from $V_{\text{dd}}$ to 0 V, then a capacitor is being discharged and the energy is dissipated through one or more NMOS transistors. Frequency is also a factor since increased activity of a circuit will require more power.
While dynamic switching contributes to most of the power usage, other factors are more apparent with lower supply and threshold voltages. Power consumption due to a short-circuit current occurs due to non-zero rise and fall times. There exists a direct path from \( V_{dd} \) to ground for a brief time when inputs are changing. This short circuit current occurs when both PMOS and NMOS transistors are on at the same time. The power due to direct-path currents is given by 

\[
P_{sc} = \frac{1}{2}(t_r + t_f) V_{dd} V_{peak} f.
\]

The third factor, static leakage, is partially caused by leakage current through the reversed-biased diode junctions in the transistors. The junction is either the source-substrate junction or the drain substrate junction. Another cause of static leakage is the subthreshold current. Smaller threshold voltages cause higher leakage currents. The static power, \( P_{stat} \), is equal to the product \( I_{leakage} V_{dd} \).

2.5 Summary of Review

At the beginning of this chapter, parallel prefix computation was described. There are two main families of parallel prefix circuits. The first is known as the \( P_k(n) \) family and the second is the \( R(n) \) family. Furthermore, the \( R(n) \) family is a superset of the \( P_2(n) \) family. Typical CMOS circuits used to evaluate logic in the nodes of a prefix circuit were also discussed. The third concept discussed was propagation delay. It is dependent upon a transistor’s effective resistance and the total capacitive load on the output. The final topic of this chapter was power consumption. We described the major contributing factors including dynamic switching, short-circuit current, and static leakage current.
3. DESIGNS

Having reviewed the concepts of parallel prefix computation, propagation delay, and power consumption, different designs may now be examined in detail. The circuit simulator HSPICE was used to simulate three different parallel prefix designs. The first adder is based on the $P_k(n)$ family[1][2]. The second design is from the $R(n)$ family[3]. The final and third design is based on the basic concept of recursive doubling. All of the designs simulated are 32-bit adders. Besides having two 32-bit operands and an output sum, they each have a $C_{in}$ and a $C_{out}$. The carry propagation with look-ahead portion of the circuit is the main difference among the three designs. Some XOR and XNOR gates are interchanged in the summation portion to maintain polarity correctness.

3.1 Brent-Kung Adder

The first design is based on the $P_k(n)$ family from Ladner and Fisher's work [1]. It is labeled as Brent-Kung because of the formulation of each node in the carry look-ahead array. The CMOS circuits for the node formulations are in Section 2.2. Brent and Kung used these CMOS circuits for their adder in [2]. They used a variation of recursive doubling for their carry look-ahead circuit. Unlike the Brent-Kung(BK) Adder, the other adders are named according to the carry look-ahead implementation. The particular design examined here is $P_0(32)$. This design has the smallest depth in the family and it has the largest number of nodes in its graph. This would imply that this circuit has the smallest delay and will use the most power.
Figure 3.1 Brent-Kung Adder
The $P_0(32)$ circuit used in the simulations is shown in Figure 3.1. For symmetry and simplicity, $P_0(n)$ is recursively constructed from two sets of $P_0(n/2)$ instead of $P_0(n/2)$ and $P_1(n/2)$ (Figure 2.2). The top row of XOR gates and other combinational logic are for the pre-calculations of generate and propagate terms. Inverters are included in selective places because the signal changes polarity between positive weight and negative weight after each level of nodes. The bottom row of XOR and XNOR enable the final summation of the operands and the carries that were calculated in the parallel prefix computation. The bottom node for each column in the carry look-ahead (CLA) is different from the regular nodes since they only need to evaluate the $G$ subcell.

There is a relationship between the capacitive load of a node and the delay that a node will experience when evaluating a new value. The nodes that act as loads to other nodes increase the capacitive load. Since some nodes are driving multiple nodes at the same time, there must be an increase in drive capability to offset the increased delay.

Many of the nodes near the bottom of the CLA have large fan-out. The bottom node in sum bit 15 has a load of 16 other nodes (to the left) plus an XOR gate. This large fan-out creates a large delay, which hinders fast propagation of the signals. A common solution is to put in a multi-stage driver between the node and its load. However, in this type of circuit it would not be feasible since it would actually increase the overall logic depth. The minimum depth of this design was the reason it was chosen. There are other CLA designs where multi-stage drivers are more appropriate. The multi-stage driver will be presented in the next section where Wei[3] found the time-optimal design.
A progressive sizing approach to find a power and time optimal design is used for this adder. Instead of putting in a multi-stage driver, the transistors in the node will have widths that are a fraction of the load. For instance, if the load is 16 nodes and the ratio is two, then the loaded node is given a relative size of eight. A loaded node is a node that has a fan-out of two or more. A fan-out of one is equivalent to the load of one inverter. For the circuit in Figure 3.1, the bottom loaded nodes are not minimum size (sum bits 15, 7, 3, and 1). Also, the second-to-last loaded nodes in the left half of the circuit are not minimum size (sum bits 23, 19, and 17). The intention is to show that circuits are more effective with re-sized nodes than those with larger depth and multi-stage drivers.

The node size ratio equations begin with the load of the \( C_{out} \) node and they go back along the bottom row of nodes, among other nodes mentioned previously. The equations are included in Appendix C and are repeated here in a more readable form.

\[
\begin{align*}
W_Ca &= W_{min} \\
W_{Cbara} &= W_{min} \\
W_Cb &= (3*W_{min})^*r' \\
W_{Cbarb} &= (17*W_{min} + W_Cb)^*r' \\
W_Cc &= (9*W_{min} + W_{Cbarb})^*r' \\
W_{Cbarc} &= (5*W_{min} + W_Cc)^*r' \\
W_Cd &= (3*W_{min} + W_{Cbarc})^*r' \\
W_{GPa} &= W_{min} \\
W_{GPbara} &= W_{min}
\end{align*}
\]
\[ \text{WGPb} = (9 \times \text{Wmin}) \times r \]
\[ \text{WGPbarb} = (4 \times \text{Wmin} - \text{WGPb}) \times r \]
\[ \text{WGPc} = (2 \times \text{Wmin} + \text{WGPbarb}) \times r \]

The "W" stands for transistor width. A "C" following the "W" stands for a partial node that is in the bottom generating the final carry for its respective bit. When a "GP" follows the name, then that is a full node with both outputs. The small letter at the end is used to differentiate the different loaded nodes. An "a" is a minimum sized node. A "b" is the first loaded node at the bottom-left portion of the carry generator, and so on until the last loaded node. The "bar" in some of the names refers to a node that inputs and outputs signals opposite in polarity to that of the regular node (refer to Section 2.2). The nodes whose sizes are being increased are on the critical delay paths.

### 3.2 Wei, Thompson, and Chen Adder

The second design, part of the \( R(n) \) family [3], is henceforth referred to as Wei’s Adder. The adder has the same type of pre-computation circuit and sum circuit, while the carry generator is different. Since the block size can vary greatly (refer to Figure 2.4), Wei created a model to find out which combination of block sizes works the best. He included the use of multi-stage drivers for loaded nodes. The time-optimal circuit is shown in Figure 3.2.

There are no published results on the power usage optimization for this design. Wei’s goal was to find a circuit that was optimized only for time delay. Once again,
Figure 3.2 Wei's Adder
minimum-sized inverters are placed to keep the signal polarity correct. Here, the depth of the carry generator graph is eight, two levels more than the BK adder. The extra multi-stage drivers also affect the size of the adder. There are 115 nodes in Wei’s Adder versus 81 nodes in the BK adder.

A simple ratio was not used to find the proper ratio of the multi-stage driver. For the cascaded driver, propagation time is a function of the fan-out, \( f \), the ratio between stages, \( r \), and the number of stages, \( s \). The driver ratio is \( r = f^{(s+1)} \) for minimum propagation delay [3]. As one saw earlier in Section 2.1 and Figure 2.3, the drivers are placed between each of the recursive blocks. From Figure 3.2, it can be seen that the number of stages in a driver ranges from one to three.

The intent of the design is that it be the fastest with the multi-stage drivers to compensate for the large fan-out. However, the BK Adder will be faster if specific nodes are increased in size as described in Section 3.1. The power consumption for Wei’s Adder will be higher than the BK Adder because Wei’s Adder requires more transistors.

### 3.3 Recursive Doubling Adder

The third design is the Recursive Doubling Adder (Figure 3.3). The maximum fan-out in this adder is two. For the top row, signals go to a node on the current bit and to the node on the next bit over to the left. For the second row, signals go to the current
Figure 3.3 Recursive Doubling Adder
node and two nodes over to the left. The number of nodes that the signals skip doubles each time. For the last row (not including the Cout node), signals skip over 16 bits.

The Recursive Doubling Adder has a total of 130 nodes and its depth is six. Since there are no nodes with large fan-out, all of the nodes are minimum size. The power usage for this adder will be greater than or equal to the power usage of Wei's Adder (Section 3.2) because it requires more nodes than Wei's Adder. The Recursive Doubling Adder is the fastest of the group because it has the minimum depth possible for a 32-bit adder. Signals propagate quickly through all parts of the circuit due to the low fan-out.

3.4 Summary of Designs

The three different 32-bit adder designs simulated in HSPICE were presented. The first design was the BK Adder, part of the \( P_k(n) \) family (Figure 3.1). The BK Adder uses the least amount of transistor devices and its carry look-ahead circuit has a minimum logical depth of six levels. Wei's Adder was the second adder, and it is part of the \( R(n) \) family (Figure 3.2). This adder's carry look-ahead portion has a logical depth of eight levels and has a bigger size. The third adder was the Recursive Doubling Adder. Its carry look-ahead is six levels, and it uses the highest number of transistors.

Table 3.1 summarizes all three designs with respect to size and depth without accounting for the re-sized nodes. The Depth column shows the total number of logic levels for each adder's carry look-ahead circuit. The Size column shows the total number of nodes in each adder's carry look-ahead circuit. Since depth and size roughly
approximate delay and power, respectively, the Depth*Size column estimates the speed-power product for each adder. From Table 3.1, one should expect the BK Adder to have the best overall performance.

<table>
<thead>
<tr>
<th></th>
<th>Depth (levels)</th>
<th>Size (nodes)</th>
<th>Depth*Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. BK</td>
<td>6</td>
<td>81</td>
<td>486</td>
</tr>
<tr>
<td>2. Wei</td>
<td>8</td>
<td>115</td>
<td>920</td>
</tr>
<tr>
<td>3. Recursive Doubling</td>
<td>6</td>
<td>130</td>
<td>780</td>
</tr>
</tbody>
</table>

Table 3.1 Depth and Size Variations Between the Three Designs
4. EXPERIMENTAL RESULTS

There are four separate sets of propagation delay and power data from the designs described in Chapter 3. The product of the propagation delay ($t_p$) and power was computed to present the speed-power product (SPP). Wei’s Adder and the Recursive Doubling Adder each has one set of data, while the Brent-Kung (BK) adder has two sets of data. The difference between the two sets of data for the BK adders is that one used a fan-out size ratio of two ($r=2$), and the other one used a fan-out size ratio of three ($r=3$).

Section 4.1 describes the variations for the simulations that were run. Section 4.2 presents the propagation delays from each simulation, and Section 4.3 presents the SPP results. Section 4.4 describes simulations run to compare power leakage for various values of $V_t$. Interconnect is briefly discussed in Section 4.5. The HSPICE simulation files are in Appendix A, Appendix B, and Appendix C. Please note that the parameter $r$ (driver size ratio) has a multiplicative inverse meaning in the netlist file in Appendix C.

4.1 Simulation Variations

For each of the four different adders, simulations were run with two different variations using a constant input frequency. The two variations were voltage supply, $V_{dd}$, and change in threshold voltage, $dV_t$. $V_{dd}$ was varied from 1.5 V to 3.0 V in steps of 0.5 V, and $dV_t$ was varied from -0.2 V to 0.6 V in steps of 0.1 V. A positive value for $dV_t$ means a decrease in the threshold voltage, $V_t$, and a negative value means an increase in
Vt. A simple C-program created a file for random input vectors to the adders (Appendix D). There were 500 vectors simulated for each design. Since HSPICE integrates over the entire time interval of the 500 random vectors, the power measurement calculation represents a good average.

4.2 Propagation Delay

The propagation delay measurements taken were a measurement of the 50% point on the input rise (or fall) to the 50% point on Cout's rise (or fall). Slightly over half of the 500 random input vectors caused a voltage swing in the Cout bit line. Cout has the largest theoretical delay since it is the bit line with the largest depth for each design. However, the actual worst case delay may differ due to transistors in series for each of the gates of a particular path. So, the delay data for each particular set of voltage parameters is the worst case of about 250 raw measurements from SPICE.

The following four pages contain the propagation delay data in graphical and in table format. The four different tables and the four different figures account for the four different Vdd values beginning with 3.0 V and ending with 1.5 V. The tables contain delay, power, and SPP values, so the data will not be repeated in the Section 4.3.

In the top graph of each figure, the delays are graphed with respect to Vt. In the bottom graph of each figure, the delays are graphed with respect to Vdd/Vt. In every
<table>
<thead>
<tr>
<th>Type of Adder</th>
<th>Brent-Kung r=2</th>
<th>Brent-Kung r=3</th>
<th>Wei</th>
<th>Recursive Doubling</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Vdd/Vt</td>
<td>tp</td>
<td>Power</td>
<td>SPP</td>
</tr>
<tr>
<td>-0.2</td>
<td>3.5</td>
<td>3.6</td>
<td>5.5</td>
<td>19.7</td>
</tr>
<tr>
<td>-0.1</td>
<td>3.9</td>
<td>3.3</td>
<td>5.6</td>
<td>18.6</td>
</tr>
<tr>
<td>0.0</td>
<td>4.5</td>
<td>3.1</td>
<td>5.7</td>
<td>17.7</td>
</tr>
<tr>
<td>0.1</td>
<td>5.3</td>
<td>2.9</td>
<td>5.9</td>
<td>17.0</td>
</tr>
<tr>
<td>0.2</td>
<td>6.4</td>
<td>2.7</td>
<td>6.1</td>
<td>16.4</td>
</tr>
<tr>
<td>0.3</td>
<td>8.2</td>
<td>2.5</td>
<td>6.3</td>
<td>16.2</td>
</tr>
<tr>
<td>0.4</td>
<td>11.2</td>
<td>2.4</td>
<td>6.9</td>
<td>16.8</td>
</tr>
<tr>
<td>0.5</td>
<td>17.9</td>
<td>2.3</td>
<td>9.4</td>
<td>21.9</td>
</tr>
<tr>
<td>0.6</td>
<td>44.5</td>
<td>2.2</td>
<td>14.8</td>
<td>33.2</td>
</tr>
</tbody>
</table>

* dVt (Volts), Vt (Volts), Vdd/Vt (Volts/Volts), tp (ns), Power (mW), SPP (pJ)

Table 4.1 Propagation Delay ($V_{dd} = 3.0$ Volts)
<table>
<thead>
<tr>
<th>Type of Adder</th>
<th>Brent-Kung ( r=2 )</th>
<th>Brent-Kung ( r=3 )</th>
<th>Wei</th>
<th>Recursive Doubling</th>
</tr>
</thead>
<tbody>
<tr>
<td>( dV_t )</td>
<td>( V_t )</td>
<td>( Vdd/V_t )</td>
<td>( tp )</td>
<td>( Power )</td>
</tr>
<tr>
<td>-0.2</td>
<td>0.9</td>
<td>2.9</td>
<td>5.0</td>
<td>3.8</td>
</tr>
<tr>
<td>-0.1</td>
<td>0.8</td>
<td>3.3</td>
<td>4.5</td>
<td>3.8</td>
</tr>
<tr>
<td>0.0</td>
<td>0.7</td>
<td>3.7</td>
<td>4.1</td>
<td>3.9</td>
</tr>
<tr>
<td>0.1</td>
<td>0.6</td>
<td>4.4</td>
<td>3.7</td>
<td>4.0</td>
</tr>
<tr>
<td>0.2</td>
<td>0.5</td>
<td>5.3</td>
<td>3.4</td>
<td>4.1</td>
</tr>
<tr>
<td>0.3</td>
<td>0.4</td>
<td>6.8</td>
<td>3.2</td>
<td>4.3</td>
</tr>
<tr>
<td>0.4</td>
<td>0.3</td>
<td>9.3</td>
<td>2.9</td>
<td>4.6</td>
</tr>
<tr>
<td>0.5</td>
<td>0.2</td>
<td>14.9</td>
<td>2.8</td>
<td>5.8</td>
</tr>
<tr>
<td>0.6</td>
<td>0.1</td>
<td>37.1</td>
<td>2.7</td>
<td>9.1</td>
</tr>
</tbody>
</table>

* \( dV_t \) (Volts), \( V_t \) (Volts), \( Vdd/V_t \) (Volts/Volts), \( tp \) (ns), Power (mW), SPP (pJ)

**Table 4.2** Propagation Delay (\( V_{dd} = 2.5 \) Volts)

![Figure 4.2 Propagation Delay (\( V_{dd} = 2.5 \) Volts)](image-url)
<table>
<thead>
<tr>
<th>Type of Adder</th>
<th>Brent-Kung r=2</th>
<th>Brent-Kung r=3</th>
<th>Wei</th>
<th>Recursive Doubling</th>
</tr>
</thead>
<tbody>
<tr>
<td>dVt</td>
<td>Vt</td>
<td>Vdd/Vt</td>
<td>tp</td>
<td>Power</td>
</tr>
<tr>
<td>-0.2</td>
<td>0.9</td>
<td>2.3</td>
<td>8.4</td>
<td>2.4</td>
</tr>
<tr>
<td>-0.1</td>
<td>0.8</td>
<td>2.6</td>
<td>7.1</td>
<td>2.4</td>
</tr>
<tr>
<td>0.0</td>
<td>0.7</td>
<td>3.0</td>
<td>6.2</td>
<td>2.5</td>
</tr>
<tr>
<td>0.1</td>
<td>0.6</td>
<td>3.5</td>
<td>5.4</td>
<td>2.5</td>
</tr>
<tr>
<td>0.2</td>
<td>0.5</td>
<td>4.3</td>
<td>4.8</td>
<td>2.6</td>
</tr>
<tr>
<td>0.3</td>
<td>0.4</td>
<td>5.4</td>
<td>4.3</td>
<td>2.7</td>
</tr>
<tr>
<td>0.4</td>
<td>0.3</td>
<td>7.5</td>
<td>3.8</td>
<td>2.9</td>
</tr>
<tr>
<td>0.5</td>
<td>0.2</td>
<td>11.9</td>
<td>3.5</td>
<td>3.3</td>
</tr>
<tr>
<td>0.6</td>
<td>0.1</td>
<td>29.7</td>
<td>3.3</td>
<td>5.2</td>
</tr>
</tbody>
</table>

* dVt (Volts), Vt (Volts), Vdd/Vt (Volts/Volts), tp (ns), Power (mW), SPP (pJ)

Table 4.3 Propagation Delay (V_{dd} = 2.0 Volts)

![Figure 4.3 Propagation Delay (V_{dd} = 2.0 Volts)](image-url)
Table 4.4 Propagation Delay ($V_{dd} = 1.5$ Volts)

![Graphs showing propagation delay vs. $V_t$ and $V_{dd}/V_t$ for different adders.](image)

**Figure 4.4** Propagation Delay ($V_{dd} = 1.5$ Volts)
case, the Recursive Doubling adder outperforms the other designs with respect to propagation delay. Wei’s Adder, designed to be time-optimal, is actually the slowest. The two BK Adders have delays very close to each other. As expected, the BK Adder with \( r=2 \) has a slightly better propagation delay than the adder with \( r=3 \).

Each adder follows an interesting trend on every graph. One will notice that as \( V_t \) decreases, the slope of the delay line approaches zero. In other words, there is a critical point where the performance stops improving with the reduction of \( V_t \). This is mainly because \( V_t \) is only one of the many factors that affect propagation delay such as \( V_{dd} \), line resistance, line capacitance, load capacitance, and circuit design. Propagation delay is also squarely dependent upon the difference between \( V_{dd} \) and \( V_t \). Increasing \( V_t \) causes an increase in the propagation delay. On the lower half of each figure, one will notice that higher values of \( V_t \) cause time delay to increase squarely. The lower values of \( V_{dd} \) are more susceptible to a change in \( V_t \). For example, when \( V_{dd}=3.0 \) V and \( V_t \) is lowered to 0.1 V (Figure 4.1), \( t_p \) decreases by 1/3. Yet, when \( V_{dd}=1.5 \) V and \( V_t \) is lowered to 0.1 V, \( t_p \) decreases by 2/3. A higher voltage supply causes a faster design. For two different figures where the \( V_{dd}/V_t \) ratio is equivalent, the one with a higher \( V_{dd} \) is faster.

4.3 Speed-Power Product

The speed-power product (SPP) is equal to the propagation delay (in ns) times the power consumption (in mW). SPP is a measure of a circuit’s energy efficiency. In microprocessors, energy consumption is important in circuit design.
Figure 4.5 Speed-Power Product ($V_{dd} = 3.0$ Volts)
Figure 4.6 Speed-Power Product ($V_{dd} = 2.5$ Volts)
Figure 4.7 Speed-Power Product (V_{dd} = 2.0 Volts)
Figure 4.8 Speed-Power Product (V_{dd} = 1.5 Volts)
Similar to the propagation delay figures, the SPP graphs are also split up into four separate figures. The SPP data from Tables 4.1 through 4.4 are graphed in Figures 4.5 through 4.8. Also, the top half of each figure shows SPP versus \( V_t \), and the bottom half of each figure shows SPP versus \( V_{dd}/V_t \).

The first observation to make from the SPP graphs is that there exists a positive curvature for each of the sets of data. In other words, there exists a minimum SPP at a particular threshold value in between the highest and lowest values of \( V_t \). The SPP data does not plateau as the delay did. The trend of the data of different designs for the SPP graphs is similar for each particular voltage. The minimum SPP occurs at a lower \( V_t \) for lower voltage supplies. At \( V_{dd}=3.0 \) V (Figure 4.5), the minimum SPP is around \( V_t=0.4 \) V, and at \( V_{dd}=1.5 \) V (Figure 4.8), the minimum SPP is around \( V_t=0.2 \) V.

The rate at which the SPP increases as \( V_t \) increases or decreases is different for different voltage supplies. For rising values of \( V_t \), SPP rises slowly at higher voltages and quicker at lower voltages. For decreasing values of \( V_t \), SPP rises slower at lower voltages than at the higher voltages. Both propagation delay and power consumption are dependent upon the voltage supply, \( V_{dd} \), and the transistor currents. An increase in \( V_{dd} \) decreases delay and increases power. A decrease in \( V_{dd} \) increases delay and decreases power. As mentioned previously, the current in a transistor, which affects delay and power, is directly proportional to the difference between \( V_{dd} \) and \( V_t \).

In the previous section, one saw that all of the designs ran faster at higher voltages. However, when the power is included, the best design is not the fastest. Power
consumption is much greater at higher voltages. The smallest values of SPP occur at lower supply voltages. For example, the lowest SPP at $V_{dd}=3.0$ V is 15.7 pJ (Table 4.1) while the lowest SPP at $V_{dd}=1.5$ V is 8.3 pJ (Table 4.4).

The device count is an important factor that affects the power consumption of a circuit since it contributes directly toward the overall capacitive load. Table 4.5 contains the total MOSFET count for each of the designs. Also, Table 4.6 lists the effective transistor count by taking into account that some designs have various transistor sizes. Designs that use more transistors consume more power. The BK Adder has fewer MOSFET devices than Wei's Adder and the Recursive Doubling adder. Remember from Chapter 3 that the BK Adder uses the least amount of nodes in the prefix carry.

<table>
<thead>
<tr>
<th>Adder Design</th>
<th>Total Number of Transistors</th>
</tr>
</thead>
<tbody>
<tr>
<td>BK</td>
<td>1930</td>
</tr>
<tr>
<td>Wei</td>
<td>2342</td>
</tr>
<tr>
<td>Recursive Doubling</td>
<td>2358</td>
</tr>
</tbody>
</table>

Table 4.5 Actual MOSFET Count

<table>
<thead>
<tr>
<th>Adder Design</th>
<th>Effective Number of Transistors</th>
</tr>
</thead>
<tbody>
<tr>
<td>BK (r=2)</td>
<td>2181</td>
</tr>
<tr>
<td>BK (r=3)</td>
<td>2043</td>
</tr>
<tr>
<td>Wei</td>
<td>2389</td>
</tr>
<tr>
<td>Recursive Doubling</td>
<td>2358</td>
</tr>
</tbody>
</table>

Table 4.6 Effective MOSFET Count
computation. Although the Recursive Doubling Adder has more nodes in its design, the multi-stage drivers make the effective MOSFET count in Wei's Adder greater than the effective MOSFET count for the Recursive Doubling Adder.

The total number of transistors were counted for 32-bit adders. The size of $n$-bit adders can be predicted by deriving equations for the total number of nodes, $S(n)$, in the carry look-ahead portion of the adder. The sizes of 4-bit, 8-bit, 16-bit and 32-bit adders were used to derive approximate equations. $S_{BK}(n) \approx (n/2)\log_2 n$ for BK Adders, and $S_{RD}(n) \approx n\log_2(n/2)$ for Recursive Doubling Adders. It is difficult to predict the size of an $n$-bit Wei’s Adder, $S_{Wei}(n)$, because it is designed from dynamic programming[3]. $S_{Wei}(n)$ is close to $S_{BK}(n)$ for small $n$, and it increases towards $S_{RD}(n)$ for larger $n$.

4.4 Power Leakage

For the BK Adder, where the ratio between the load and driving node is two, simulations with different input frequencies were run. Here, $V_{dd}$ was constant at 3 V and $dV_t$ had values of 0.0 V, 0.2 V, 0.4 V and 0.6 V. The power leakage data is shown in Figure 4.9 and Table 4.7. The two far right columns in Table 4.11 contain power consumption values at $f=0$ MHz that were extrapolated from the data at the other frequencies. The calculated values are estimated power consumption due to current leakage (see Section 2.3).
Figure 4.9 Power Leakage Effects of Decreasing $V_t$.
The top graph in Figure 4.9 shows what happens when one decreases the threshold voltage. There is little change in power consumption when $V_t$ is decreased by 0.4 V. Yet, when $dV_t = 0.6$ V (a decrease of 0.6 V), power consumption increases significantly by about 9 mW for each particular frequency. By extending the lines to the 0 MHz axis, one can get the excess power consumption that is not caused by transistor switching. The lower part of Figure 4.9 graphically shows the extrapolated values. There is very little power leakage until $V_t$ is close to zero.

4.5 Interconnect Wires

The simulations did not take interconnect delays into account. However, the total number of interconnects between CLA nodes are totaled in Table 4.8. Also, a spreadsheet program was used to determine the trend for $n$-bit adders based on 4-bit, 8-bit, 16-bit, and 32-bit adders. The BK Adder requires fewer interconnecting wires than Wei’s Adder and the Recursive Doubling Adder. Also note that interconnect affects the Recursive

<table>
<thead>
<tr>
<th>Freq. (MHz)</th>
<th>$I_{vdd}$ (mA)</th>
<th>Power (mW)</th>
<th>$I_{vdd}$ (mA)</th>
<th>Power (mW)</th>
<th>$I_{vdd}$ (mA)</th>
<th>Power (mW)</th>
<th>$I_{vdd}$ (mA)</th>
<th>Power (mW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>1.0</td>
<td>2.9</td>
<td>1.2</td>
<td>3.6</td>
<td>3.7</td>
<td>11.2</td>
<td>0.0</td>
<td>0.04</td>
</tr>
<tr>
<td>100</td>
<td>1.9</td>
<td>5.7</td>
<td>2.3</td>
<td>6.9</td>
<td>4.9</td>
<td>14.8</td>
<td>0.2</td>
<td>0.04</td>
</tr>
<tr>
<td>200</td>
<td>3.8</td>
<td>11.4</td>
<td>4.5</td>
<td>13.5</td>
<td>7.3</td>
<td>22.0</td>
<td>0.4</td>
<td>0.33</td>
</tr>
<tr>
<td>300</td>
<td>5.7</td>
<td>17.1</td>
<td>6.7</td>
<td>20.1</td>
<td>9.7</td>
<td>29.2</td>
<td>0.6</td>
<td>7.58</td>
</tr>
</tbody>
</table>

Table 4.7 Power Leakage Effects of Decreasing $V_t$
Doubling Adder more since the average wire length in the Recursive Doubling Adder is greater than in the other two adders.

<table>
<thead>
<tr>
<th></th>
<th>BK</th>
<th>Wei</th>
<th>Recursive Doubling</th>
</tr>
</thead>
<tbody>
<tr>
<td>32-bit</td>
<td>243</td>
<td>416</td>
<td>363</td>
</tr>
<tr>
<td>(n)-bit</td>
<td>0.34(n^{1.02})</td>
<td>0.24(n^{1.57})</td>
<td>0.27(n^{1.45})</td>
</tr>
</tbody>
</table>

Table 4.8 Total Number of Interconnects Between CLA Nodes

4.6 Experimental Results Summary

In this chapter, the variations in the HSPICE simulations run on the designs described in Chapter 3 have been explained. Section 4.2 presented the experimental data for propagation delay. For most variations, the Recursive Doubling Adder was the fastest. Section 4.3 presented the experimental data for power consumption as the speed-power product (SPP). Except for Wei's Adder, all of the adders had similar SPP values when the parameters were varied. The SPP of Wei's Adder was higher than the other adders. Also in Section 4.3, we found that the BK Adder (\(r=2\)) had the lowest effective MOSFET transistor count and Wei's Adder had the highest effective MOSFET transistor count. In Section 4.4, it was seen that the power leakage is not significant until \(V_T\) is close to 0 V (within 0.1 V). Finally, the trend of wires interconnecting CLA nodes for the designs were compared. The BK Adder was found to have the fewest number of interconnects for an \(n\)-bit adder.
5. CONCLUSIONS

Conclusions reached are presented below. Section 5.1 provides a summary of the thesis. It includes the simulated designs and results after simulations, where different parameters were adjusted. Following the summary, Section 5.2 presents ideas for further research on this subject.

5.1 Summary

In the preceding chapters, three different adder designs were presented as well as the results of simulating the designs under several different voltage supplies, threshold voltages, and frequencies.

For all of the adder designs, the smallest propagation delay occurred with $V_{dd}=3.0$ V, while the lowest SPP occurred with $V_{dd}=1.5$ V. The Recursive Doubling Adder was always the fastest for any specific set of parameters.

Considering the speed-power product, the Recursive Doubling Adder and the BK Adders were nearly equal, and they were also all lower than Wei's Adder. Each set of data had a minimum SPP. The threshold voltage at which the minimum SPP occurred increased as $V_{dd}$ was increased ranging from 0.2 Volts to 0.4 Volts. Also, the Brent-Kung Adders were the smallest in terms of the total number of MOSFET devices.
When $V_t$ is decreased, power consumption increases. The power is linearly dependent on frequency. The increase in power is not significant until $dV_t=0.6$ V ($V_t$ is close to zero). The power-frequency data provided the power leakage in each circuit. The power leakage difference between $dV_t=0.0$ V and $dV_t=0.4$ V is insignificant relative to the 7.6 mW of leakage that occurred at $dV_t=0.6$ V.

Looking at each of the results for the various parameters, an adder suitable to particular criteria may be chosen. If one is looking for just a fast adder, the Recursive Doubling Adder is best. However, considering size and area, the best alternative would be the BK Adder ($r=2$). The optimum speed-power product occurs at a supply voltage of 1.5 V and a threshold voltage of 0.2 V.

5.2 Future Research

The design approach for reducing the effects of fan-out can be applied to other circuits besides an adder. There are portions of a microprocessor where it may also be more suitable to size the driving gates or cells accordingly instead of placing multi-stage drivers. The need to use greater depth and include multi-stage drivers is not necessary when the design approach is used with the BK Adder. The threshold value must also be varied to get the optimum efficiency of a circuit if $V_{dd}$ changes. Further studies should include the RC delays caused by interconnects.
REFERENCES


APPENDICES
APPENDIX A HSPICE Command File

* 32-bit Adder using Brent-Kung *
* Parag Shah *

.include "/nfs/stak/u1/s/shahp/mentor/class/ECE503/include/level3.param"
.include "/nfs/stak/u1/s/shahp/mentor/class/ECE503/brentKung/adder5.nl"
.include "/nfs/stak/u1/s/shahp/mentor/class/ECE503/include/200mhz2500ns30v.wav"

* increase Vp to get better speed
* decrease Vn to get better speed
Vp pwell 0 0
Vn nwell 0 Vd
VDD VDD 0 Vd DC
.GLOBAL VDD pwell nwell

.param Vd=3
.param dVt=0.0

.tran In 2500n
.meas tran avg_power avg power
.meas tran avg_ivdd avg i(vdd)
.meas tran avg_ipwell avg i(vp)
.meas tran avg_inwell avg i(vn)

.probe v(cout)
.plot v(cout)
.option post ingold=2 probe

xadder A0 A1 A10 A11 A12 A13 A14 A15 A16 A17 A18 A19 A2
+ A4 A5 A6 A7 A8 A9 B0 B1 B10 B11 B12 B13 B14 B15
+ B28 B29 B3 B30 B31 B4 B5 B6 B7 B8 B9 CIN Cout SUM0
+ SUM1 SUM10 SUM11 SUM12 SUM13 SUM14 SUM15 SUM16 SUM17
+ SUM18 SUM19 SUM2 SUM20 SUM21 SUM22 SUM23 SUM24 SUM25
+ SUM26 SUM27 SUM28 SUM29 SUM3 SUM30 SUM31 SUM4 SUM5 SUM6
+ SUM7 SUM8 SUM9 adder1a

.END
APPENDIX B  Level 3 HSPICE Parameters

```
.MODEL N NMOS LEVEL=3 PHI=0.700000 TOX=9.5000E-09 XJ=0.200000U TPG=1
+ VTO=0.6674 DELTA=1.4270E+00 LD=6.3300E-08 KP=1.7146E-04
+ UO=471.7 THETA=1.6690E-01 RSH=3.3470E+01 GAMMA=0.5219
+ NSUB=1.0840E+17 NFS=5.9080E+11 VMAX=2.2650E+05 ETA=2.0550E-02
+ KAPP=2.1270E-01 CGDO=9.0000E-11 CGSO=9.0000E-11
+ CGBO=3.6007E-10 CJ=5.69E-04 MJ=0.661 CJSW=2.00E-11
+ MJSW=0.609 PB=0.99
* Weff = Wdrawn - Delta_W
* The suggested Delta_W is 3.3260E-07

.MODEL P PMOS LEVEL=3 PHI=0.700000 TOX=9.5000E-09 XJ=0.200000U TPG=-1
+ VTO=-0.9188 DELTA=3.5350E-01 LD=7.8860E-08 KP=3.8312E-05
+ UO=105.4 THETA=3.3670E-02 RSH=1.6950E+01 GAMMA=0.7396
+ NSUB=2.1770E+17 NFS=5.9080E+11 VMAX=1.5650E+05 ETA=1.7260E-02
+ KAPP=8.8780E+00 CGDO=9.0000E-11 CGSO=9.0000E-11
+ CGBO=3.6237E-10 CJ=9.19E-04 MJ=0.321 CJSW=4.60E-10
+ MJSW=0.100 PB=0.42
* Weff = Wdrawn - Delta_W
* The suggested Delta_W is 3.3680E-07
```
* Netlist for Brent-Kung 32-bit adder *
* For transistor size variations r *
* Parag Shah
* March 17, 1998

.param L=0.5u
.param Wmin=1u
.param r=0.5
.param k=0
.param Cw='Wmin*k'

.param Winv=Wmin
.param Wnand=Wmin
.param Wnor=Wmin
.param Wxor=Wmin
.param Wxnor=Wmin

.param WGPa=Wmin
.param WGPbara=Wmin
.param WGPb=(9*Wmin+9*Cw)*r'
.param WGPbarb=(4*Wmin+4*Cw+WGPb)*r'
.param WGPc=(2*Wmin+3*Cw+WGPbarb)*r'

.param WCa=Wmin
.param WCBara=Wmin
.param WCb=(3*Wmin+2*Cw)*r'
.param WCbarb=(17*Wmin+17*Cw+WCb)*r'
.param WCc=(9*Wmin+9*Cw+WCbarb)*r'
.param WCbarc=(5*Wmin+5*Cw+WCc)*r'
.param WCd=(3*Wmin+3*Cw+WCbarc)*r'

.SUBCKT inverter
+ IN OUT width=1u length=Wmin deltaVt=dVt
Vbn Nin in deltaVt
Vbp in Pin deltaVt
MXI_2 OUT Nin 0 pwell N w=width l=length AS=16.0P AD=16.0P PS=12.0U + PD=12.0U
MXI_1 OUT Pin VDD nwell P w='2*width' l=length AS=16.0P AD=16.0P PS=12.0U + PD=12.0U
.ENDS * inverter *
.SUBCKT nor
  + PLBAR POUT PRBAR width=Wnor length=L deltaVt=dVt
  VbAn PLBARn PLBAR deltaVt
  VbAp PLBAR PLBARp deltaVt
  VbBn PRBARn PRBAR deltaVt
  VbBp PRBAR PRBARp deltaVt
  MXI_4 POUT PRBARp N_6 nwell P w='2*width' l=length m=1 AS=16.0P AD=16.0P
      + PS=12.0U PD=12.0U
  MXI_3 N_6 PLBARp VDD nwell P w='2*width' l=length m=1 AS=16.0P AD=16.0P
      + PS=12.0U PD=12.0U
  MXI_2 POUT PRBARn GND pwell N w=width l=length m=1 AS=16.0P AD=16.0P
      + PS=12.0U PD=12.0U
  MXI_1 POUT PLBARn GND pwell N w=width l=length m=1 AS=16.0P AD=16.0P
      + PS=12.0U PD=12.0U
.ENDS * nor *

.SUBCKT nand
  + PL POUTBAR PR width=Wnand length=L deltaVt=dVt
  VbAn PLn PL deltaVt
  VbAp PL PLp deltaVt
  VbBn PRn PR deltaVt
  VbBp PR PRp deltaVt
  MXI_4 POUTBAR PLp VDD nwell P w='2*width' l=length m=1 AS=16.0P AD=16.0P
      + PS=12.0U PD=12.0U
  MXI_3 POUTBAR PRp VDD nwell P w='2*width' l=length m=1 AS=16.0P AD=16.0P
      + PS=12.0U PD=12.0U
  MXI_2 POUTBAR PLn N_10 pwell N w=width l=length m=1 AS=16.0P AD=16.0P
      + PS=12.0U PD=12.0U
  MXI_1 N_10 PRn GND pwell N w=width l=length m=1 AS=16.0P AD=16.0P
      + PS=12.0U PD=12.0U
.ENDS * nand *

.SUBCKT Ga
  + GL GOUTBAR GR PL width=Wmin length=L deltaVt=dVt
  VbGLn GLn GL deltaVt
  VbGLp GL GLp deltaVt
  VbGRn GRn GR deltaVt
  VbGRp GR GRp deltaVt
  VbPLn PLn PL deltaVt
  VbPLp PL PLp deltaVt
  MXI_18 GOUTBAR GLn GND pwell N w=width l=length m=1 AS=16.0P AD=16.0P
      + PS=12.0U PD=12.0U
  MXI_17 GOUTBAR GLp N_6 nwell P w='2*width' l=length m=1 AS=16.0P AD=16.0P
      + PS=12.0U PD=12.0U
  MXI_4 N_6 GRp VDD nwell P w='2*width' l=length m=1 AS=16.0P AD=16.0P
+ PS=12.0U PD=12.0U
MXI_3 N_6 PLp VDD nwell P w='2*width' l=length m=1 AS=16.0P AD=16.0P
+ PS=12.0U PD=12.0U
MXI_2 GOUTBAR PLn N_5 pwell N w=width l=length m=1 AS=16.0P AD=16.0P
+ PS=12.0U PD=12.0U
MXI_1 N_5 GRn GND pwell N w=width l=length m=1 AS=16.0P AD=16.0P
+ PS=12.0U PD=12.0U
.ENDS * Ga *

.SUBCKT Gbara
+ GLBAR GOUT GRBAR PLBAR width=Wmin length=L deltaVt=dVt
VbGLn GLBARn GLBAR deltaVt
VbGLp GLBAR GRBARp deltaVt
VbGRn GRBARn GRBAR deltaVt
VbGRp GRBAR GRBARp deltaVt
VbPLn PLBARn PLBAR deltaVt
VbPLp PLBAR PLBARp deltaVt
MXI_6 N_10 GRBARn GND pwell N w=width l=length m=1 AS=16.0P AD=16.0P
+ PS=12.0U PD=12.0U
MXI_5 GOUT GRBARp N_14 nwell P w='2*width' l=length m=1 AS=16.0P AD=16.0P
+ PS=12.0U PD=12.0U
MXI_4 N_14 PLBARp VDD nwell P w='2*width' l=length m=1 AS=16.0P AD=16.0P
+ PS=12.0U PD=12.0U
MXI_3 GOUT GLBARp VDD nwell P w='2*width' l=length m=1 AS=16.0P
+ AD=16.0P PS=12.0U PD=12.0U
MXI_2 GOUT GLBARn N_10 pwell N w=width l=length m=1 AS=16.0P AD=16.0P
+ PS=12.0U PD=12.0U
MXI_1 N_10 PLBARn GND pwell N w=width l=length m=1 AS=16.0P AD=16.0P
+ PS=12.0U PD=12.0U
.ENDS * Gbara *

.SUBCKT gpgen
XI_621 N_620 GP[0] inverter
XI_620 AB[1] N_620 AB[0] nor
XI_215 N_213 GP[1] inverter
XI_4 AB[1] N_213 AB[0] nand
.ENDS * gpgen *

.SUBCKT xor
+ A B OUT width=Wxor length=L deltaVt=dVt
VbAn An A deltaVt
VbAp A Ap deltaVt
VbBn Bn B deltaVt
VbBp B Bp deltaVt
MXI_8 N_11 N_203 GND pwell N w=width l=length m=1 AS=16.0P AD=16.0P + PS=12.0U PD=12.0U
MXI_7 N_9 Bn GND pwell N w=width l=length m=1 AS=16.0P AD=16.0P + PS=12.0U PD=12.0U
MXI_6 OUT N_18 N_11 pwell N w=width l=length m=1 AS=16.0P AD=16.0P + PS=12.0U PD=12.0U
MXI_5 OUT An N_9 pwell N w=width l=length m=1 AS=16.0P AD=16.0P + PS=12.0U PD=12.0U
MXI_4 OUT N_203 N_4 nwell P w=2*width l=length m=1 AS=16.0P AD=16.0P + PS=12.0U PD=12.0U
MXI_3 OUT N_18 N_4 nwell P w=2*width l=length m=1 AS=16.0P AD=16.0P + PS=12.0U PD=12.0U
MXI_2 N_4 Bp VDD nwell P w=2*width l=length m=1 AS=16.0P AD=16.0P + PS=12.0U PD=12.0U
MXI_1 N_4 Ap VDD nwell P w=2*width l=length m=1 AS=16.0P AD=16.0P + PS=12.0U PD=12.0U
XI_211 B N_203 inverter width=Wxor
XI_210 A N_18 inverter width=Wxor
.ENDS * xor *

.SUBCKT inverter2
+ IN[0] IN[1] OUT[0] OUT[1]
 XI_1 IN[0] OUT[0] inverter
.ENDS * inverter2 *

.SUBCKT Ca
+ COUTBAR GPLEFT[0] GPLEFT[1] GR
 XI_465 GPLEFT[1] COUTBAR GR GPLEFT[0] Ga width=WCa
.ENDS * Ca *

.SUBCKT GPbara
 + width=WGPbara
 XI_5 GPLEFTBAR[0] GOUT[0] GPRIGHTBAR[0] nor width=WGPbara
.ENDS * GPbara *

.SUBCKT GPa
 XI_207 GLEFT[0] GOUTBAR[0] GPRIGHT[0] nand width=WGPa
.ENDS * GPa *
.SUBCKT GPb
  XI_207 GPLEFT[0] GPOUTBAR[0] GPRIGHT[0] nand width=WGPb
  .ENDS * GPb *

.SUBCKT Cbara
  + COUT GPLEFTBAR[0] GPLEFTBAR[1] GR
  XI_213 GPLEFTBAR[1] COUT GR GPLEFTBAR[0] Gbara width=WCbara
  .ENDS * Cbara *

.SUBCKT Cbarb
  + COUT GPLEFTBAR[0] GPLEFTBAR[1] GR
  XI_213 GPLEFTBAR[1] COUT GR GPLEFTBAR[0] Gbara width=WCbarb
  .ENDS * Cbarb *

.SUBCKT GPc
  XI_207 GPLEFT[0] GPOUTBAR[0] GPRIGHT[0] nand width=WGPc
  .ENDS * GPc *

.SUBCKT Cd
  + COUTBAR GPLEFT[0] GPLEFT[1] GR
  XI_465 GPLEFT[1] COUTBAR GR GPLEFT[0] Ga width=WCd
  .ENDS * Cd *

.SUBCKT xnor
  + A B OUT width=Wxnor length=L deltaVt=dVt
  VbAn An A deltaVt
  VbAp A Ap deltaVt
  VbBn Bn B deltaVt
  VbBp B Bp deltaVt
  XI_26 B N_19 inverter width=Wxnor
  MXI_20 N_8 N_19 GND pwell N w=width l=length m=1 AS=16.0P AD=16.0P
  + PS=12.0U PD=12.0U
  MXI_18 OUT N_204 N_10 pwell N w=width l=length m=1 AS=16.0P AD=16.0P
  + PS=12.0U PD=12.0U
  MXI_17 OUT An N_8 pwell N w=width l=length m=1 AS=16.0P AD=16.0P
  + PS=12.0U PD=12.0U
  MXI_21 N_10 Bn GND pwell N w=width l=length m=1 AS=16.0P AD=16.0P
  + PS=12.0U PD=12.0U
  MXI_15 OUT Bp N_2 nwell P w=2*width l=length m=1 AS=16.0P AD=16.0P
  + PS=12.0U PD=12.0U
  MXI_14 OUT N_204 N_2 nwell P w=2*width l=length m=1 AS=16.0P AD=16.0P
XI_3896 N_4148 N_3661[0] N_3661[1] N_4170 Cc
XI_4249 N_4160 N_4066 SUM[1] xnor
XI_5103 CIN N_4069 SUM[0] xor
XI_4246 N_4154 N_4057 SUM[4] xnor
XI_4245 N_4152 N_4054 SUM[5] xnor
XI_4244 N_4150 N_4051 SUM[6] xnor
XI_4243 N_4148 N_4048 SUM[7] xnor
XI_4231 N_4130 N_4021 SUM[16] xnor
XI_3881 N_4170 N_3607[0] N_3607[1] N_4160 Cbarc
.ENDS * adder1a *
APPENDIX D Random Input Generator

#include <stdio.h>

/* genclk.c generates clock waves for simulations
   Usage: genclk freq(Mhz) edgerate(ns) maxsimtime(ns)
   outputfile_seed_name
   VCC(volts) deadtime(ns) start(1/0) maxwave(#) */

main(argc, argv)
int argc;
char *argv[];
{
    float t1,tmax,v1,v2,freq,edge,vcc,deadtime,startv,maxwav;
    char *outfileseed;
    int i,j,k;
    FILE *fp;
    char filename[100];

    if (argc != 9)
    {
        printf(" number of arguments incorrect\n");
        printf(" Usage: genclk frequency(Mhz) edgerate(ns)
               maxsimtime(ns)\n");
        printf(" outputfile_seed_name vcc(volts) deadtime(ns)
               start(1/0)\n");
        printf(" NumberOfWaves(n)\n");
        exit(-1);
    }

    sscanf(argv[1], "f", &freq);
    sscanf(argv[2], "%f", &edge);
    sscanf(argv[3], "%f", &tmax);
    sscanf(argv[5], "%f", &vcc);
    sscanf(argv[6], "%f", &deadtime);
    sscanf(argv[7], "%f", &startv);
    sscanf(argv[8], "%f", &maxwav);

    outfileseed = argv[4];

    sprintf(filename,"%s.wav",outfileseed);
    fp = fopen(filename,"w");

    for (i=0; i<=maxwav; i++)
    {
        if (startv == 1)
            v2 = vcc;
else
    v2 = 0;

srand(time(0));

if (i < maxwav/2)
    fprintf(fp,"Va%d a%d 0 pwl 0nS %gV ",i,i,v2);
else if (i < maxwav)
{
    j=i-maxwav/2;
    fprintf(fp,"Vb%d b%d 0 pwl 0nS %gV ",j,j,v2);
}
else
    fprintf(fp,"Vcin cin 0 pwl 0nS %gV ",v2);

for(t1=deadtime;t1<tmax;t1=t1+(1/freq)*1000)
{
    v1 = (rand()%2) * vcc;
    if ((v2 != v1) && (t1 >= deadtime))
    {
        k++;
        if ( k==4 )
        {
            fprintf(fp, "\n + ");
            k=0;
        }
        fprintf(fp,"%gS %gV ",t1, v2);
        fprintf(fp,"%gS %gV ",t1+edge, v1);
        v2 = v1;
    }

    fprintf(fp, "\n");
    sleep(1);
}
fclose(fp);
return 0;