High speed and low power arithmetic design using limited switching dynamic logic

Cong Nguyen
Lehigh University

Follow this and additional works at: http://preserve.lehigh.edu/etd

Recommended Citation
Nguyen, Cong, "High speed and low power arithmetic design using limited switching dynamic logic" (2004). Theses and Dissertations. Paper 836.
Nguyen, Cong
High Speed and Low Power Arithmetic Design using Limited Switching Dynamic Logic

May 2004
High speed and low power arithmetic
design using Limited Switching Dynamic Logic

by

CONG NGUYEN

A Thesis

Presented to the Graduate and Research Committee

of

Lehigh University

In Candidacy for the degree of Master of Science

in

The department of Electrical and Computer Engineering

at

Lehigh University

May 2004
This Thesis is accepted and approved in partial fulfillment of the requirement for the Master of Science

May 2004

Dr. Marvin White

Thesis Advisor

Donald M. Bolle

Chair of Department
I would like to express my sincere gratitude to my manager, Mr. Marc Lang, for his understanding and supports in terms of encouragement and getting the necessary paper work in order to get the financial grant from IBM for the course of my study and, doing this research on this subject and also making the resources available for me to work on this project. At the same time, I am also sincerely grateful for Dr. Marvin White who advised and gave me the inspiration to get started on this thesis and also gave invaluable suggestions, ideas for this project. Without their supports, this project would not have been possible.
Table of Contents

Acknowledgement iii
List of Figures xii
Abstract 1

1. Introduction

1.1 Motivation ................................................................. 2
1.2 Thesis overview........................................................... 3

2. Background and Conventional Dynamic Circuit

2.1 Dynamic and Domino circuits....................................... 5
2.2 Bulk CMOS vs. SOI technology: Partially and Fully Depleted SOI..... 10
2.3 Source of power consumption in static and dynamic circuits ........... 12

3. Computer Arithmetic Review

3.1 CLA Addition Theory.................................................. 14
3.2 Adder Architectural descriptions.................................. 17

CLU: Second level of Abstraction ........................................ 18

4. Limited Switching Dynamic style (LSD)

4.1 LSD circuit description and analysis ................................ 22
4.2 LSD Clocking and feedback Keeper functionality................... 30
4.3 Transistor sizing for Dynamic and Static circuits.................... 31
5. Adder Design and Implementation

5.1 64-bit and Static CLA adder designs .........................................40
5.2 Physical design tools and CLA adder layouts .................................41
5.3 IBM’s CMOS8S0 SOI technology ..................................................45
5.4 Design checking in physical design ...............................................47

6. Adder Evaluations

6.1 Static adder performance evaluations .............................................49
6.2 LSD adder performance evaluations .............................................50

7. Conclusions and future works ......................................................55

Appendixes

A. Static schematics and layouts ......................................................55
B. LSD schematics and layouts .......................................................57
C. SOI Partially Depleted Simplified Equations ...................................61
D. Power Calculation Panels ............................................................62

References 63

Vitae 66
### List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Dynamic circuit configuration</td>
<td>7</td>
</tr>
<tr>
<td>2.2</td>
<td>Domino circuit configuration</td>
<td>8</td>
</tr>
<tr>
<td>3.1</td>
<td>4-bit carry block</td>
<td>19</td>
</tr>
<tr>
<td>3.2</td>
<td>16-bit CLA</td>
<td>20</td>
</tr>
<tr>
<td>3.3</td>
<td>64-bit CLA</td>
<td>21</td>
</tr>
<tr>
<td>4.1</td>
<td>LSD clock generation scheme</td>
<td>22</td>
</tr>
<tr>
<td>4.2</td>
<td>LSD AOI circuit configuration</td>
<td>23</td>
</tr>
<tr>
<td>4.3</td>
<td>LSD inverter configuration</td>
<td>25</td>
</tr>
<tr>
<td>4.4</td>
<td>LSD inverter with clk=0 and regardless of input A, Z=1</td>
<td>26</td>
</tr>
<tr>
<td>4.5</td>
<td>LSD inverter configuration, clk=1 and A=1, Z=0</td>
<td>27</td>
</tr>
<tr>
<td>4.6</td>
<td>LSD inverter with clk=1, A=1 and Z=0</td>
<td>28</td>
</tr>
<tr>
<td>4.7</td>
<td>Dyn/Dom and Lsd test circuits</td>
<td>28</td>
</tr>
<tr>
<td>4.8</td>
<td>LSD’s inverter wave-form outputs</td>
<td>29</td>
</tr>
<tr>
<td>4.9</td>
<td>Sizing Static inverter</td>
<td>37</td>
</tr>
<tr>
<td>4.10</td>
<td>Sizing LSD circuit (AOI)</td>
<td>38</td>
</tr>
<tr>
<td>5.1</td>
<td>flowchart for Virtuoso XL</td>
<td>42</td>
</tr>
<tr>
<td>5.2</td>
<td>CMOS physical design methodology flow</td>
<td>44</td>
</tr>
</tbody>
</table>
## List of Tables

Table 2.1 IBM's activity coefficients  
Table 4.1 IBM CMOS design parameters  
Table 4.2 IBM CMOS design guideline for stacked devices  
Table 5.1 The IBM CMOS8S0 features  
Table 6.1 Comparison between Static and LSD adders
Abstract

This thesis presents a 64-bit carry look ahead adder (CLA) using a level-sensitive dynamic logic topology design known at IBM as a Limited Switching Dynamic (LSD) circuit. This style of circuit is typically used in data paths oriented in a highly critical path application due to its compactness and relatively low power consumption compared to other circuit dynamic logic styles. The advantages offered by the LSD circuit style are high density coupled with a single rail power supply design. This 64-bit Carry Look Ahead Adder is a compact design using various techniques and tools currently available at IBM to design and implement circuits from schematics to layouts. Some aspects of the SOI technology is presented and briefly discussed. The LSD adder is more compact and switches at the same clock rate or higher, requires more tuning, and consumes approximately 10 to 15 percent less power than its static counterpart. A full description of the LSD circuit operation is given and as well as the description of the architecture for the implemented LSD Carry Look Ahead Adder.
Chapter I

Introduction

In a modern Digital Signal Processing IC and high speed microprocessors, the data path is the core where all the arithmetic and highly intensive operation are carried out and performed. A system's performance is largely determined by the design and implementation of its architecture, data paths and finally the logic circuit styles. Over the year may types of dynamic logic design styles were proposed and implemented.

1.1 Motivation

Computer arithmetic plays an important part in designing a high performance microprocessor. Since very often arithmetic operations are being used to compute various types of memory addresses and also performing addition, multiplication and division therefore designing fast arithmetic circuits and low power consumption are very important because arithmetic circuits are architecturally complex that require large number of transistors to implement as a result more and more transistors being packed into smaller silicon in order to accommodate all the features. Apparently, high transistors count does not only lead to high power consumption but also reduce system liabilities due to running at high temperature and not to mention the cost of cooling these systems. To remedy this problem, dynamic circuits are often utilized more and more in the design where data path oriented designs become bottleneck in terms of performance since dynamic circuits are compact and potentially have low power consumption even though dynamic circuit topology has some well known drawbacks such as charge leakage, charge sharing, clock-skewed intolerant, and non inversion outputs.

Despite the fact that the current available dynamic circuit topology offers significant advantage over other logic circuit styles in terms of circuit density, relatively low power consumption and higher clock speed. However the VLSI design community is still somewhat reluctant to adapt dynamic or domino circuit style over the well known
other CMOS circuit topology such as static due to reliability issues, charge leakage, power consumption and especially low noise margin.

This proposed LSD (Limited Switching Dynamic) circuit is an attempt to address some of the problems encountered by the conventional dynamic as well as domino logic circuit families. Static CMOS has been well known for its reliability and noise tolerance compare to other circuit topologies. As a case in point, IBM CMOS static circuits still play an essential and dominant part in microprocessor design and implementation, but dynamic logic circuits have slowly gained more and more acceptance and have been employed in Gigaherz processor design methodology. This is due in part to their high performance and compactness as microprocessor switching in static CMOS circuits becomes more difficult to achieve with increased clock frequency which results in high power consumption. Several types of dynamic and domino circuit styles are currently being implemented in microprocessor design methodology; however, LSD circuit topology is often chosen over other dynamic logic style since LSD offers significant advantages in comparison with conventional dynamic logic due to its latching and single rail capability.

1.2 Thesis overview

The thesis is organized as follows: Chapter 1 is an introduction and Chapter 2 briefly goes over some background information, including a discussion of previous work, which includes domino and dynamic logic design styles with single phase clock. These logic circuit styles are constructed either with latches or without latches. Chapter 2 also reviews and describes some of the basic dynamic and domino circuits and their functionality with a discussion of some of the shortfalls of these circuits. Also, chapter 2 describes the need for high speed circuit design styles for data-path oriented design and explores various techniques and design issues using Silicon on Insulator (SOI) technology.
Chapter 3 reviews the computer arithmetic of addition by providing a summary of the Carry Look Ahead (CLA) Adder Boolean derivations which are used to implement the CLA Adders. Chapter 4 provides a detail description of the Limited Switching Dynamic (LSD) circuit in terms of circuit analysis, functionality, and the pros and cons of the LSD implementation are discussed. Chapter 5 discusses the sizing of static and dynamic transistors including the static design methodology currently being implemented at IBM in order to quickly perform transistor sizing. For sizing transistor in dynamic circuits, an iteration method is used to determine the transistor sizes using IBM’s methodologies and tools.

Both static and dynamic CLA Adder implementations are also discussed in Chapter 5 with IBM’s standard library cells along with the custom design of dynamic circuits. The designs are verified using Cadence tools, such as Design Rule Check (DRC), Layout Versus Schematic (LVS) along with other IBM’s internally developed tools to further check the design for beta, clock-lines and other yield issues. Also, Chapter 5 presents the designs of both LSD and static CLA Adders using CMOS8S0 IBM’s SOI technology to implement the physical design of the Adders. The features of IBM’s CMOS8S0 Partially-Depleted Silicon on Insulator (PDSOI) technology will be briefly discussed in chapter 5.

Finally, the evaluations of both CLA Adders, static and dynamic, are presented in Chapter 6 in terms of chip area, performance and power consumption. Chapter 7 is a summary which includes some thoughts of looking forward to future works and applications for LSD circuit utilization at IBM.
Chapter II

Background and Conventional Dynamic Circuit

This section briefly describes a few of the conventional dynamic and domino logic design styles and includes a general discussion of circuits with these topologies. The dynamic and domino circuits are slowly gaining acceptance due to their high-speed and lower power consumption, especially in the Silicon on Insulator (SOI) CMOS process. In general, some authors concentrate on simplicity of design, low cost and reduced area with improved performance [1] and [2], while other authors focus on the issues of more speed and elaborated architectures. At the same, there are also papers that describe dynamic design techniques using pipeline latches in attempt to reduce power consumption with increasing clock speed. For example, high speed CMOS design has been discussed with different techniques to implement dynamic circuits using single and multiple phase clocking [1]. However, the subject of dynamic and domino logic is broad and there are many different topologies and styles currently available therefore this thesis will focus on the conventional and basic configurations of dynamic and domino logic styles.

2.1 Dynamic and Domino Circuits

Over the years many papers have proposed and studies performed on dynamic and domino circuits with techniques ranging from simple to complex architecture implementations. These papers focused on improvement in circuit performance, while reducing problems associated with the use of dynamic topologies, such as charge sharing, leakage, clock skew sensitivity, and noise intolerance. Inherently, dynamic CMOS circuit structures are sensitive to noise and have lower noise margins compare to static CMOS structures. Since there are a vast number of papers on dynamic circuits, this thesis will confine the discussion to the area of circuit structures and the subject of low power and performance.
Dynamic logic is a circuit style well suited for high-performance microprocessor design. It offers significant performance over the competing static logic style, with the same or better performance, reduced chip area, and lower power consumption. However, dynamic circuits are more susceptible to noise, charge leakage, crosstalk, and sensitivity to clock-skewing. Limited Switching Dynamic (LSD) is similar to the dynamic logic circuit family except LSD circuits have the ability to latch and store a value, which is helpful to reduce power consumption.

In static logic, the output nodes are continually driven - either high or low depending on the input logic and the logic circuit spends very little time during a transition from low to high and vice versa. Thus, there is virtually no DC path between logic states. There is a small amount of current called ‘feed-through’, which is dynamic with a time depending on the sizing of the circuits. Unlike the case of a static circuit, however, the output node of a dynamic circuit is pulled high, low, or floating with the logic value stored on the output node capacitance. This value is changed when the inputs to the logic circuit make a transition from low to high and the clock is high or in an evaluation mode. A typical dynamic circuit has a precharge, where the evaluation transistors are pulled down when the clock is high, and the logic is evaluated at a high value for the evaluated or ‘footed’ device.

Figure 2.1 is a typical example of dynamic circuit, which implements an AOI function. The circuit has two distinctive operations - precharge and evaluation phases. In the precharge phase, the precharge transistor is ON while the clock is low and the output is forced to a high state for a period of time. As soon as the clock makes a transition to high, the evaluated or ‘footed’ device is ON. Next, when the data arrives on time, depending on the value of the input logic state, a decision is made whether or not a complete path will be formed so current can flow and discharge to ground. If the input logic states are all high, then the result is evaluated as a high and a complete discharge path to ground for the current, thereby, causing the output to stay low. Otherwise, the inputs must remain stable in the evaluation phase or transition monotonically from high.
to low only during the evaluation phase where the output can only make a transition from high to low.

Figure 2.1 Dynamic circuit configuration

The dynamic circuit has an inverted output and extremely sensitive to clock skew and data, charge leakage and crosstalk. Thus, the dynamic circuit can not be cascaded unless the clock and data have been properly delayed in order to synchronize with the arrival of the data inputs, otherwise, the circuit will be subject to output errors, and the data is unrecoverable. To remedy this problem, as shown in Figure 2.2, an inverter is added to the output of the dynamic circuit and, in addition, a feedback device known as a “keeper”, is included to help prevent charge sharing and leakage, while synchronizing the clock and data to the input of the circuit. This is a commonly used technique in dynamic/domino structures. The domino gate is composed of an NMOS pulled-down device and a pull-up PMOS device followed by an inverter. The addition of an inverter permits domino circuits to be cascaded where the output of each stage is driven by the output of the preceding domino stage. The presence of an inverter guarantees the inputs of each stage are all de-asserted prior to the start of an evaluation phase. Similar to
Dynamic circuits, domino circuits can only have non-inverting outputs and the input can only be a slowly varying value or non-inverting function.

![Dynamic Circuit Diagram](attachment:dynamic_circuit.png)

**Figure 2.2 Domino circuit configuration**

There are many types of dynamic and domino logic present in different logic families, but in this thesis we will limit our discussion and concentrate on a single rail dynamic/domino circuit with a half-latch including a keeper since the LSD logic style is closely resembles the previous logic families.

The circuits mentioned above are susceptible to noise and charge leakage. The conventional dynamic/domino circuits continuously switch the outputs during a pre-charge and evaluation modes when the input logic is high; however, when the input logic is low the output is latched since the dynamic node is floating. Consequently, the dynamic node is being held high by the output node capacitance during the evaluation and precharge phases. In this condition, the output stays low until the input making a transition from high to low. This is because the feedback node is tied directly to the dynamic node and the 'keeper' pulls up this node when the output going low. In addition, there is no isolation between the dynamic node and the output inverter as this will cause
the output to continue to toggle resulting in high power consumption. This is because the power consumption depends highly on input values, operational frequency, switching activity, and the nodes and load capacitance of the dynamic circuit.

In addition to the issue of high power consumption, there are concerns for single rail power supply dynamic logic since there are no non-inverting outputs available, consequently dynamic circuit is hard to use in conjunction with static or other logic circuit styles due to this incompatibility. In order to maintain compatibility among other logic circuit families, inverted outputs must also be available so the dynamic/dyno interface circuitry will not be required. To generate inverted outputs for dynamic or domino logics, dual rail power supply methodology is needed, which requires extra complementary logics to generate the inverted outputs. As a result, the dual rail power supply methodology does not only require more time, effort and extra circuitry but also make the design more complex and higher power consumption. High device count does not only take more area but also complicates and requires more wiring tracks and also makes the cell becomes more irregular since the wiring track, in metal 1, usually determine how large the cell become based on the metal 1 track wiring availability of a cell. Typically, the output of a dynamic circuit drives the input of other directly of static gate and dynamic, without the inverter the input dynamic gate reduces the noise margin when noise occurs it's difficult for the circuit to recover the errors, the cycle needs to be stretched giving more time for the circuit to recover however there is a time that the circuit can't from the erroneous states.

Dynamic circuits are also sensitive to clock delay and subject to noise margins being reduced because the output is not completely latched in the use of the dynamic circuit without the inverter. For domino circuit, the output is latched the presence of the half-latch when the output is low when the clock delay differential is great the output will prematurely switch leading to an erroneous state which it can’t recover, also the presence of the inverter could also help the noise margin but very often the input is sized, beta, to favor the rising/falling edges which reduces the over all noise margin of the circuit since there is no isolation between the output inverter and the dynamic node.
2.2 Bulk CMOS vs. SOI technology: Partially and Fully Depleted SOI

For the past few decades Bulk CMOS technology has been advancing along in an exponential path and of shrinking device dimensions, increasing density, increasing speed and decreasing cost. The scaling of CMOS technology has progressed rapidly for the last few decades, however this may soon come to an end because the power dissipation constraints, leakage current and various other issues. The primary problem is static power dissipation, which is caused by leakage current from channel tunneling and others, also the level of integration on VLSI chip is not only bigger but also having more complex functions in a single chip that leads to extremely high number of transistors count. The motivation for moving to SOI clearly is to extent scalability of CMOS and to continue to enjoy performance as well as density benefits of migrating design and architecture to the next generation lithography.

Silicon-on-insulator (SOI) offers a 15% to 35% performance gain over bulk CMOS. High-performance microprocessors using CMOS SOI have been designed and available for production. As the technology moves to the 0.13um generation, the SOI CMOS is being used in more and more application and spreading to lower-end microprocessors and memory products. As we move to the 0.1um generation and beyond, SOI CMOS is expected to be the technology of choice for system-on-a-chip (SOC) application which requires high performance CMOS device, high density and lower power consumption.

Currently, there are two ways to implement the SOI device, one is fully depleted (FD) and the other partially depleted (PD) techniques. Although, both are designed to reduce the body and other junction capacitances but the partially depleted process gains more acceptance due to being more compatible to the current bulk CMOS with little change in the process and the financial and practicality. The fully depleted, FD, achieves more significant reduction in S/D junction and body capacitances compare to the PD however to achieve this significant gain, a really thin film, si, is required and this makes it more expenses and highly incompatible to the current bulk CMOS process without a
major retooling and financial investment is too great for a company, and company is reluctant to take on, as a result the PD process currently gains more acceptance in the main stream with some performance tradeoffs.

PD-SOI can be thought of as an evolution rather than revolutionary device structure, in that with the exception of wafer SIMOX (Separation by Implantation of Oxygen) formation, the CMOS wafer fabrication is performed on the same bulk CMOS tool set, targeting parameters which are very similar to or identical to bulk CMOS. Scalability: A device definition which, when scaled through successive lithography, continuing to produce a viable product is very important. Because PD-SOI does not require the ultra thin active silicon thickness typically used in full-depleted SOI (FD), investments in its developed can be returned over more generations. Using identical materials, tools, process, and parameters target as in is bulk predecessor, the spatial variability in SOI electrical parameters is well established and accommodated in software tools commonly used. That leaves the use dependent variation, which can be shown to be well modeled. With FD-SOI, the entire body of the transistor is inverted. The threshold voltage, therefore, is a function of the charge contained in the body, and can vary substantially across the chip.

It is becoming more evident that continued CMOS scaling into deep submicron region is causing the limits of bulk substrate CMOS to emerge. With diminishing returns and performance and higher leakage currents in store for the coming lithography generation, Partially Depleted SOI becomes more and more attractive option for high speed and low power design alternative technology. Advances in processing and fabrication of SIMOX wafers, as well as a better understanding of SOI circuit behavior make this technology a candidate for volume application in the present generation. SOI also known for design liabilities which, because the more simple bulk CMOS option provided acceptable power delay, discouraged its past use. With the substantial advantages SOI is now noted for, it has become worthwhile to examine the more closely its electrical behavior so that robust, complex SOI product may be initiated.
2.3 Source of power consumption in static and dynamic circuits

Static circuit topology, the most significant part of the power consumption occurs only during transition at the outputs is charging or discharging. The output capacitance is consists of three components $C_{\text{int}}$, $C_{\text{wire}}$ and $C_{\text{load}}$. $C_{\text{int}}$ represents the internal capacitance of the gate. This $C_{\text{int}}$ mostly consists of the diffusion capacitance of the drain. $C_{\text{wire}}$ represents the capacitance due to the physical interconnections and finally $C_{\text{load}}$ represents the sum of the input gates and capacitances of the transistor being fed by the outputs. The $C_{\text{int}}$ and $C_{\text{load}}$ mostly depends on specific generation of technology and can be obtained and pre-calculated from the technology library, however $C_{\text{wire}}$ is highly unpredictable due to routing dependent and usually depends on the length of the wires and what layer of metal the and how optimal the connection is made.

There are three major sources of the power dissipation in the static circuits as described in equation (2.1)

\[
P_{\text{DYN-S}} = \alpha (f_c C_L V_{DD}^2) + (I_{SC} \cdot V_{DD} + I_L \cdot V_{DD})
\]  

(2.1)

The 1st term represents the switching component of the power, where $C_L$ is the loading capacitance and $f_c$ is the operating frequency of the system, and $\alpha$ is the probability that a power consumption transition occurs, also known as the activity factor. The 2nd term of the equation (2.1) is due to the short-circuit when both the NFET and PFET are active and conducting current for a short period during a transition high to low or low to high. Finally, the leakage current, $I_L$, which can arise from substrate injection and sub-threshold effects, is primarily determined by the technology fabrication under consideration. In equation (2.1) the dominant terms are $\alpha$, $C_L$, $f_c$ and $V_{DD}$, short-circuit power consumption.

\[
P_{\text{SC}} = \frac{\beta \tau}{12T} (V_{DD} - V_T - V_T)^3
\]  

(2.2)

and the static power due to leakage current.

\[
P_S = I_L V_{DD}
\]  

(2.3)
Dynamic power consumption is somewhat different from static CMOS since a dynamic gate is clocked and operated in two phases: precharge and evaluation modes. Because of the precharging and discharging operations, dynamic circuits consume power in both precharge and discharge. Since the dynamic circuit can also dissipate energy even the input value staying constant thus the power consumption expression for dynamic logic circuit can be summarized below,

\[ P_{\text{dyn}} = \frac{1}{2} f_c C_i V_{DD}^2 \]  \hspace{1cm} (2.4)

Similar to the e static power consumption, \( P_{\text{dyn}} \) also depends on the operating frequency \( f_c \) and the capacitive load \( C_i \) and \( V_{dd}^2 \) except the power consumption is half that of static circuit. Finally, the switching \( \alpha \) probability is defined as,

\[ \alpha_{0\rightarrow1} = \lim_{N \to \infty} \frac{n(N)}{N} \]  \hspace{1cm} (2.5)

where \( N \) is the number of cycles and \( n(N) \) of is the number of \( 0 \rightarrow 1 \) transitions in \( N \) clock cycles. The above formulas and coefficients below are the information that IBM’s Power Calculator uses to estimate the power consumption for the design.

<table>
<thead>
<tr>
<th>Data types</th>
<th>Activity ( (\alpha_{\text{activity}}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock</td>
<td>0.5</td>
</tr>
<tr>
<td>Random data signal</td>
<td>0.5</td>
</tr>
<tr>
<td>Simple logic driven by random data</td>
<td>0.4 – 0.5</td>
</tr>
<tr>
<td>Finite state machines</td>
<td>0.08 – 0.18</td>
</tr>
<tr>
<td>Conclusion</td>
<td>0.05 – 0.5</td>
</tr>
</tbody>
</table>

Table 2.1: IBM’s activity coefficients
Chapter III  
Computer Arithmetic Review

This section is intended to provide some basic background in arithmetic addition review and explain the basic theory behind the Carry Look Ahead addition algorithm and its derivation. Two implementations of the CLA, in terms of the level abstraction, are discussed with a comparison of two CLA architectures.

3.1 CLA addition theory

Since the most critical path of an adder is its carry therefore there have been many proposals that have been developed over the years. The Carry Look Ahead algorithm CLA is one of the fastest carry generators for an adder. The CLA computes the carry bit by using the inputs in parallel. In binary arithmetic addition it is possible to pre-compute the carry in parallel. In binary addition the sum of two bit binary numbers is describes as

\[ s_i = x'y'c_i + x'y c'_i + x_i y c'_i + x_i y_i c_i \quad \rightarrow \quad s_i = (x_i \oplus y_i) \oplus c_i \]  \hspace{1cm} (3.1)

and the carry,

\[ c_{i+1} = x_i y c'_i + x_i y_i c_i + x'_i y c_i + x'_i y_i c_i \quad \rightarrow \quad c_{i+1} = x_i y_i + (x_i \oplus y_i)c_i \]  \hspace{1cm} (3.2)

However if the carry equation is further refined by expanding equation 3.2 a pattern can be seen, we can redefine an other set of equation to represent the carry generate and propagate as below,

\[ g_i = a_i b_i \quad \text{and} \quad p_i = a_i + b_i \quad \text{or} \quad p_i = a_i \oplus b_i \]  \hspace{1cm} (3.3)

where \( g_i = a_i b_i \) is used to represents the generate signal a carry and if a carry is present and the \( p_i = a_i + b_i \) is to propagate a carry if a carry position is generated and these are the basic principal of the CLA algorithm. Using the new notations for the generate \( g_i \) and propagate \( p_i \), a carry enters in position \( i+1 \) and if a carry is generated \( i \) or a carry enters position \( i \) and is propagated through the carry chain is found to be.
\[ c_{i+1} = g_i + p_i c_i \]  
(3.4)

Solving this equation recursively,

\[ c_{i+2} = g_{i+1} + p_{i+1} c_{i+1} \]  
(3.5)

\[ = g_{i+1} + p_{i+1} (g_i + p_i c_i) \]

\[ = g_{i+1} + p_{i+1} g_i + p_{i+1} p_i c_i \]

if we further extend the carry to one more position for a 4-bit adder,

\[ c_{i+3} = g_{i+2} + p_{i+2} c_{i+2} \]  
(3.6)

\[ = g_{i+2} + p_{i+2} (g_{i+1} + p_{i+1} g_i + p_{i+1} p_i c_i) \]

\[ = g_{i+2} + p_{i+2} g_{i+1} + p_{i+2} p_{i+1} g_i + p_{i+2} p_{i+1} p_i c_i \]

In general, the above equations summarize the CLA pre-computing carry algorithm. The equation for the carry into position \( i + r \) is given as,

\[ c_{i+r} = \sum_{i+k}^{i+r-1} g_i \left( \prod_{j=i+1}^{k+r-1} p_j \right) + c_k \prod_{j=k}^{k+r-1} p_j \]  
(3.7)

To implement the above equation requires \( i + 1 \) gates, which have a maximum fan-in of \( i + 1 \), and to produce all carries from \( c_{i+1} \) to \( c_{i+r} \) requires a total gate count of,

\[ \sum_{i=1}^{n} (i+1) = \frac{(n+3)n}{2} \]  
(3.8)

The set of equation 3.4 - 3.6 represent the 1st level of abstraction of the adder because the ‘generate’ and ‘propagate’ are individually obtained from equation 3.3 . Theoretically, the CLA algorithm can extend the carry computation to any number of \( N \) bits; however, due to fan-in limitations the practical implementation in hardware for the CLA algorithm is usually limited to a fan-in of 4 and in 4-bits block modules. The first level of abstraction is only useful for an adder with \( N \) less than 16-bits because as the operand gets larger it becomes less efficient and incurs more delay. To implement the
CLA with a larger operand, greater than 32-bits, the further expansion of the CLA is required. If we consider a 2nd level of abstraction, then we can define 4-bit block CLA ‘propagate’ and ‘generate’ supper signals as,

\[ P_{30} = p_3 p_2 p_1 p_0 \]  
\[ g_{30} = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 \]

where \( p_{30} \) and \( g_{30} \) are the ‘propagate’ and ‘generate’ signals when a ‘propagate’ is true to propagate the carry if present and a carry is also generated, if a condition exists for a carry. The supper \( p_{30} \) and \( g_{30} \) are similar to the 1st CLA’s abstraction; however, the supper \( p_{30} \) and \( g_{30} \) are the product of individual ‘propagate’ bits and the sum of product of individual ‘propagate’ and ‘generate’ signals. From expressions 3.9 and 3.10, these are not embedded in carry precomputing, equation 3.4 to 3.6, as a result the next carry of a 4-bit CLA is simply reduced to,

\[ c_4 = g_{30} + p_{30} c_0 \]  

As seen in equation 3.11, to compute the next carry only requires 3 inputs AOI (And Or Invert) gate. This is one of the reason that makes larger operands more efficient and incurring smaller delay. Interchangeably, the next carry often is expressed as,

\[ c_{i+1} = G_i + P_i c_0 \]

where \( P_i \) and \( G_i \) are supper ‘propagate’ and ‘generate’ signals, respectively. Finally, the sum is expressed in terms of the previous carry and the propagate signal. In this case, it is an individual propagate signal which is computed from the input bits. \( a \) and \( b \).

\[ s_i = c_{i-1} \oplus \pi \quad \text{where} \quad \pi = a_i \oplus b_i \]
3.2 CLA Architectural description

Theoretically, the CLA algorithm is one of the fastest ways to pre-calculate and predict the carry by using the inputs; however, there is a hidden penalty in input fan-out and area as the number of bits begins to increase. When the number of input bits is greater than 4, equation 3.6 cannot be easily extended because of fan-in and fan-out limitations. As we observed in equations 3.4 to 3.6, the number of literals of each product term increases by one with each successive equation, which in turn requires a gate with fan-in of N+1. In practice, technology usually limits the fan-in of an input gate and usually the number of inputs, which is available for a typical AND/OR gate, is limited to 3 or 4 due to the stacking NFET devices for an AND gate and PFET devices for an OR gate implementation. For the reason described above, the CLA adder grows exponentially when N is greater 4 and as a result, the CLA algorithm is best optimal at 4-bits due to fan-in technology limitations.

The Carry Look Ahead Adder is selected for this implementation due to its parallel carry computing and prediction and its relatively more regular structure as compared to other implementations, such as the conditional sum adder, carry and save, carry-skip adder, and various other architectures available for implementing the addition algorithms. Also, it is important to keep the adder scalable to easily migrate to a new technology. This can be done with relative ease with the CLA architecture. In a straightforward implementation, directly from the above equations and close-form expressions, we can derive the number of gates needed and predict the delay and the most critical path of the adder. The adder is more regular, which can be implemented in a modular 4-bit architecture and configured, either in parallel or serial, with several high-level abstractions depending on the size of the operand. However, the optimized size is a fan-in of 4 with three levels of abstraction and very little additional hardware required.
CLA: Second level of Abstraction

The 4-bit CLA adder can be implemented with a carry block in place of the 4-bit generator that is separating the carry function which is previously embedded in the carry Look ahead block. This CLA configuration reduces the critical carry path to a smaller number of gate delays and the structure is much more regular due to the additional 4-bit carry block that is used to generate the carry function. The equation (3.12), in chapter 3, describes the dedicated carry block to compute the carry externally and the feed to the next stage of CLA.

\[ c_{i+1} = G_i + P_i c_0 \]  
(3.12)

The above equation can be easily implemented with a simple OR function, the carry is simultaneously outside the carry generate block as shown in figure 3.4. This external carry computation block is implemented directly using the equation and the block implement the propagate signals using the below,

\[ p_{3_0} = p_3 p_2 p_1 p_0, \quad g_{3_0} = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 \]  
(3.9) and (3.10)

where \( p \) is the propagate signal when a carry is generated and \( g \) is the carry generate if there is a present of a carry. The carry of a 4-bit CAL can be expressed as,

\[ c_4 = g_{3_0} + p_{3_0} c_0 \]  
(3.11)

\[ c_{i+1} = G_i + P_i c_0 \]  
(3.12)

and the sum is,

\[ s_i = c_{i+1} \oplus p \quad \text{where} \quad p = a_i \oplus b_i \]  
(3.13)

*Figure 3.1* shows a 4-bit CLA. The top part shows a block diagram of the 4-bit CLA and the bottom shows a detail diagram of the adder.
The 4-bit adder consists of a 4-bit reduced adder, which includes the propagate and generate functions, and an external dedicated carry function. The 4-bit propagate, generate and sum is composed of equations (3.9) and (3.10) above these are used to compute in parallel potential the propagate and generate outputs from the inputs \( x \) and \( y \). The sum is computed by the reduced half adder. The significant of *Figure 3.1* is the carry block since this block is a very simple function to implement and it only need \( p_4 \) and \( g_4 \) in order to compute the output carry for the next stage. This reduces the complexity and speed up the carry output significantly.
Figure 3.2 shows a 16-bit CLA diagrams of the 16-bit block and a detailed diagram of the CLA. This 16-bit adder consists of four 4-bit CLA module as previously described, Figure 3.1. Each of the 4-bit adder takes in two operands x<0:3> and y<0:3> and produces a sum<0:3> and two bits propagate and generate p<4> and g<4>. The propagate and generate outputs from each of the 4-bit adder are fed to the 4-bit propagate and generate in order to produce 2 bit outputs supper P and G. These P and G bits will be fed to the next level of the CLA in order to make 64-bit CLA. The output carry of each 4-bit adder stage is serially fed to the next stage 4-bit adder to complete the 16-bit CLA module.
Finally, the 64-bit adder is composed of four 16-bit modules as shown in Figure 3.3. The \((p_{16}, g_{16}), (p_{32}, g_{32}), (p_{48}, g_{48})\) and \((p_{64}, g_{64})\) are generated by each of the 16-bit module. These four set of propagate and generate are served as the inputs for the carry block which computes the carry output and serially feeds to the next stage of the CLA. The 64-bit CLA is configured similar to a ripple carry adder. It consists of 4 carry function blocks that are used to simultaneously computes the carry outputs and feeding back the carry in order to generate the sum outputs. With a dedicated carry adder in the architecture, it takes 476 gates with 2 levels of abstractions and has 13 gate delays.
Chapter IV

Limited Switching Dynamic Style (LSD)

This chapter presents a detail description of the LSD (Limited Switching Dynamic) logic style in terms of its functionality, circuit analysis. The 1st part is to describe the LSD circuit and detail analysis and its advantages and disadvantages over the conventional circuit style and its application. Delay has always been an important parameter in circuit design therefore is discussed in this section and also transistor sizing in Static and LSD CMOS circuits.

4.1 LSD descriptions and analysis

A two-phase clock design is used for the LSD circuit and the CLA (Carry Look Ahead Adder) however LSD logic style can also be used to speedup the data path by interleaving between static, with no logic level translation needed. as a result the LSD circuit can be used twice in a clock cycle provided the precharge and evaluate cycle are relatively short and relative to the system clock which is typically generated using LCBs (Local Clock Buffer generation). As shown in Figure 4.1 above. This interleaving technique is often used at IBM to reduce and speed up the critical paths at the cost of time

![LSD Clock Pulse Generation](image)

Figure 4.1: LSD clock generation scheme
and efforts since LSD circuits requires more elaborate, design iteration and simulation with more corners than the typical CMOS static circuits logic styles. Also, to interleave LSD circuits between static also requires more hardware such as LBCs and clock pulse generation in order to generate short pulse widths relative to the system clock, namely $\text{ph1}$ and $\text{ph2}$ clock pulses. In chapter 2, a brief discussion

![LSD AOI circuit configuration](image)

between advantages and disadvantages of the conventional dynamic and domino circuits due to clock sensitive and inability to latch in the value as a result the output will constantly change as the dynamic nodes toggle regardless of the input value swings from a high to low transition or staying constant for a long period of time this results to high power consumption in the precharge and evaluation modes.

In Figure 4.2 is a typical LSD AOI circuit topology which consists of Nfet devices to implement logic inputs of the AOI function. A Pfet and footed devices are used for evaluation and precharge. In addition, LSD circuit also includes a latching mechanism that consists of an inverter and a half latch similar to the conventional domino
circuits however the LSD has 4 extra devices, \( N1 \) \( N2 \) \( N3 \) and \( P1 \), that are inserted between the output inverter and input logic tree, these 4 devices act as a tri-state inverter, which is used to latch in the value or to isolate the input and output stage, depending on the feedback from the output during the clock is high or in evaluation phase. This so-called tri-state inverter provides an isolation between the input dynamic node and the output state, together with the output half-latch they act like a transparent latch which is driven by the feedback value of the output. When the output makes a transition for low to high, the feedback Nfet device will be turned on and the tri stated inverter operates like a normal inverter thus latching the output value regardless of the precharge cycle. This output value will not change until the next set of input logics being evaluated to high thus the output will be pulled low and the keeper Pfet device will be activated to latch in this value.

Since the output value is stored on the latch node which is based on the drain junction capacitance value and gate input capacitance depending on the size of the tri-state inverter. The output value is stored and get latched at this node therefore the output value does not switch as often as the input value stays high or low for a over long period of time, over more than one cycle, compared to the conventional dynamic or domino logic families. The output only makes transition when clock is high and when value is high the LSD circuits behave like the typical dynamic circuit, that is the output always makes a monotonic transition similar to typical dynamic circuit, due to tri-state inverter becomes active and the precharge value will propagate through and the feedback and the "keeper", Pfet device \( P2 \), is turned off when the clock is high. In an evaluation mode, the precharge Pfet is turned off depending on the input value being evaluated whether to high or low. The two footed devices, connected to the tri-state inverter, are combined as one if the input evaluation to a low then output stat does not change whenever the result is high the circuit is evaluated and the dynamic node is pulled down by the footed device and the dynamic node will discharge this has to transition at the output node from a high to a low this in terms turn on the "keeper" together this half-latch will latch in the value regardless the precharge value and the precharge cycle until there is a
change at the input logic and the input is evaluated to a logic high once again in order for the output node to switch or change state. When the output stays low, in subsequent precharge cycle the output will remain low due to Pfet keeper being turned on and the tri-state inverter goes to high impedance state as a result the output stays until the logic inputs evaluated to a high value thus forcing the output making a transition from high to low.

*Figure 4.3* shows a LSD inverter, unlike the conventional dynamic or domino circuit, the output is completely latched by the tri-state and the half-latch structure regardless of the input. The conventional dynamic circuit only latches the output when the input stays low or in a precharge cycle since the dynamic left floating and being pulled high by the precharge Pfet P0 causing the output to go high then it gets latched however the input stays high for more than one cycle the dynamic will toggle as a precharge and evaluate cycle proceeds as a result.
the output will toggle continuously as a result the output follows the dynamic node so long as the input stays high or evaluated to a high value. On top of the latching capability of the LSD circuit, the LSD topology also provides a better noise immunity because the tri-state inverter and the output inverter effectively isolate the static or latched node thus it takes much more efforts to cause a false transition at the latched or static node. The tri-state inverter and the output inverter can also be sized to provide a much better noise margin compare to that of conventional dynamic or domino logic family as it can be seen in Figure 4.4.

The output is totally latched unless the inputs are making a transition. When the input stays either high or low for a long period of time and the output still stays latched this helps the power consumption and also when a transition is made, the static node does not make a full 1 to 1 transition. LSD logic styles also have the ability to locally invert the output at the expense of delay, area and some skews between the complementary outputs, without resorting to dual
rail design, with an addition of a buffer unlike other dynamic/domino a full complementary logic tree and additional inverter. The complementary LSD’s outputs inverters will need to be sized carefully in order to minimize the impact, these inverters can also be sized to favor the rising or falling edges accordingly.

Figures 4.4, 4.5 and 4.6 demonstrate how the LSD inverter behaves with different value of input A. When the clk is low, the LSD inverter is in precharge mode, the output Z always stays high due to Nfet N0 is turned off and all other devices are active. Since the output Z goes high, it turns on the Nfet N2 and N3 which complete a path to activate the tri-state inverter as seen in Figure 4.4. The circuit will act like a half-latch as long as the output value remained high. In evaluation mode and the clk is high, N0 and N1 Nfets device become active. If the input value A is high or evaluated to a high, the LSD inverter will make a transition from Figure 4.5 to Figure 4.6 and the feedback Pfet device, the keeper, is turned on because the output Z making a transition from high to low this in turn will latch in the output value. The tri-state inverter stays active since Nfets device N1 provides a complete path to ground for the tri-state inverter due to the circuit is in evaluate mode. clk is high.
In Figure 4.7 describes a test schematic consists of three different types of circuits, dynamic, domino and LSD circuits.

Figure 4.6: LSD inverter with clk=1, A=1 and Z=0

Figure 4.7: Dyn/Dom and Lsd test circuits
In Figure 4.7 describes a test schematic consists of three different types of circuits, dynamic, domino and LSD circuits.
As it can be seen in this schematic, a simple inverter of Dynamic, Domino and LSD circuit is given and a ASX, similar to SPICE transistor simulator, simulation is performed on these circuits with an input test sequence for the input \( A \). With the Dynamic and Domino inverter have a single output compared to the LSD circuit which can locally generates the non-inverted output without affecting the internal loading of the circuit since the internal is completely latched and isolated by the tri-state inverter. \textbf{Figure 4.8} shows the output’s wave forms of the test circuit presented in Figure 4.7, the output of the LSD’s \( Z \_ \text{LSD} \) is completely latched despite the input values being toggle during the clock cycle. This output only makes a transition when the input value \( A \) stays low for at least one cycle.
Unlike the LSD inverter, the outputs of the Dynamic and Domino circuits are not latched and keep toggling so long as the input value continues to stay high and the clock continue to switch. The internal node of the LSD’s circuit shows that it toggles just like the Dynamic and Domino outputs however the stat_fbk_lat node is a static node and latches the value as soon as the LSD’s output makes a transition from high to low respectively.

### 4.2 LSD Clocking and feedback Keeper functionality

Clocking LSD circuits is different from clocking static latches or typical dynamic domino circuits. There is still a synchronizing system clock, but the LSD stages themselves utilize locally generated pulses which have been triggered from a transition of the system clock. Figure 4.1 is a diagram that shows how LSD clock pulses relate to the system clock. Note that the term system clock is analogous here to the global clock referring to Figure 4.1 but in this LSD adder design and implementation the above clock description is not used because IBM’s proprietary methodology and confidential materials at IBM.

The keeper prevents inadvertent switching due to charge sharing in non-output-switching evaluation phases. Because all evaluation tree inputs must be stable, by design, by the time the clock goes high (thus evaluating the stage), if the tree does not evaluate, there is no charge transfer occurring in the tree (all internal nodes would have already been charged or discharged during the precharge phase). If the stage does evaluate, desired charge transfer occurs. This implies that if the inputs are not supposed to affect a positive evaluation, any internal nodes that would have otherwise drained charge from the dynamic node (in a domino circuit) are already charged by the time the clock goes high, thus, little or no charge transfer takes place in the LSD circuit during a non-evaluating evaluation phase. The keeper aids in dynamic node charge recovery from noise events.
It should also be noted that as the evaluation phase for each LSD stage is much shorter than its associated precharge phase, there exists a lessened probability (compared to "regular" dynamic circuits) that a noise event will be synchronous with the evaluation phase to possibly cause a noise related failure. The keeper aids in charge maintenance during long evaluation phase durations, as might exist during test or other diagnostic or low frequency operating conditions, whereby current leakage through the evaluation tree would otherwise drain charge from the dynamic node, resulting in inadvertent output switching.

4.3 Transistor sizing for Static and Dynamic circuits

At IBM, there is a strict constraint in the microprocessor design processes and methodology, in general the class of circuits allowed are very few, mostly Static however with some exception in using other circuit styles including low threshold devices. The SOI technology available at IBM only allows to stack 3 height for Nfet and 4 for Pfet devices. In Static CMOS design, the beta ratio of the Nfet and Pfet devices is typically 3 and the Gain for a logic stage is 4 or beta $\beta$ for an inverter stage is 3, beta being fined as a Gain for a digital circuit such as inverter, Nand, Nor etc. The restrictions are primarily due to and related to yield issues in SOI technology and to limit unnecessary power consumption and most importantly of all is to maintain a consistency throughout the microprocessor design community. The limitations and guidelines are strictly enforced via CAD (Computer Aided Design) tools and tool checking available at IBM for any potential violations in the design.

There are many different ways or methods to start the design from scratch, in this case starting with transistor level schematic and finally translating the schematic to layout or physical design after meeting all design specifications. However, it has not been always easy to convert the transistor level to physical design and meeting timing requirements because there are many factors that come in to play and the design is often complicated and typically involves hundreds and hundreds of gates and complex wiring scheme using different level of metal layers as a result it is most of the time it is not
practical to calculate transistor’s sizes and estimate wire loading by hand and not to mention it may several trial and errors to get all timing requirement to converge.

For this reason, The method for static transistor sizing starts with the circuit driver’s load and being scaled backward depending on the complication of the circuits. The driver is often a set of inverter that is designed to drive heavy load, high fanouts or long wire. Unusually, the beta ratio between stages is limited between 2.5 to 3 in order to maintain a consistency and not to exceed the power consumption being allocated for the design. Since transistor circuits is dependent on the availability of technology in terms of transistor stacking, physical designs and wiring tracks. *Table 4.1* provides a preliminary way to estimate the capacitances of the Nfet and Pfet devices and metal layers to derive transistors’ sizes in order to meet timing in early stages of the designs.

<table>
<thead>
<tr>
<th>Design Parameters</th>
<th>Values</th>
<th>Descriptions/comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gate capacitance</td>
<td>$0.35 \frac{fF}{\mu m}$</td>
<td>Minimum geometry, effective average</td>
</tr>
<tr>
<td>Source/Drain capacitance</td>
<td>$0.19 \frac{fF}{\mu m}$</td>
<td>SOI has little dependence on diffusion area</td>
</tr>
<tr>
<td>Driving into with gate tied off</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Metal 1 resistance</td>
<td>$0.121 \Omega \cdot L$ (0.8 $\mu m$ wide)</td>
<td></td>
</tr>
<tr>
<td>Metal 2 resistance</td>
<td>$0.106 \Omega \cdot L$ (0.8 $\mu m$ wide)</td>
<td></td>
</tr>
<tr>
<td>Metal 3, 4 and 5 resistance</td>
<td>$0.082 \Omega \cdot L$ (0.9 $\mu m$ wide)</td>
<td>$L= \text{Length}(\mu m, \text{drawn})$, $\text{Temp} = 85^\circ C$</td>
</tr>
<tr>
<td>Metal 1 capacitance</td>
<td>$0.051 fF \cdot L$</td>
<td></td>
</tr>
<tr>
<td>Metal 2 capacitance</td>
<td>$0.048 fF \cdot L$</td>
<td></td>
</tr>
<tr>
<td>Metal 3 capacitance</td>
<td>$0.047 fF \cdot L$</td>
<td>$L= \text{Length}(\mu m, \text{drawn})$, Assumes minimum width/min space and fully populated above and below</td>
</tr>
</tbody>
</table>

*Table 4.1: IBM CMOS design parameters*
Even though, the table parameters are somewhat simplified but these parameters provide a highly effective method of estimated timing and reduce the number iteration that normally take to fine tune the designs thus cutting down design cycle. To deal with device stacking for Nfet and Pfet, *Table 4.2* summarizes the limits of SOI technology and constraint of transistor level design. This table, *Table 4.2*, sets the limit for N and Pets stacking and scaling factors for different stacking scenarios. The reason for limiting stacking in SOI design is due to the charge leakage and other issues related yield. Unlike bulk CMOS, SOI is highly vulnerable when it come to stacking the device N and as well as P fetes devices more 4 devices high.

<table>
<thead>
<tr>
<th>Stack height</th>
<th>Effective Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 high nFET</td>
<td>$W$</td>
</tr>
<tr>
<td>2 high nFET</td>
<td>$\frac{W}{1.72}$</td>
</tr>
<tr>
<td>3 high nFET</td>
<td>$\frac{W}{2.44}$</td>
</tr>
<tr>
<td>4 high nFET</td>
<td>$\frac{W}{3.16}$</td>
</tr>
<tr>
<td>1 high pFET</td>
<td>$W$</td>
</tr>
<tr>
<td>2 high pFET</td>
<td>$\frac{W}{1.94}$</td>
</tr>
<tr>
<td>3 high pFET</td>
<td>$\frac{W}{2.95}$</td>
</tr>
</tbody>
</table>

*Table 4.2: IBM CMOS design guideline for stacked devices*

*Table 4.2* provides a guideline for stacking N and Pfet device in different situations other than just to size an inverter. According to *Table 4.2*, the actual width of the device, $W$, is divided by a scale factor depending on the type of the device and how high they’re being stacked. This scaling factor is used to reflect the actual performance of the device because the higher the devices are stacked the large the scale factor become thus degrading the over all performance of the circuit as a result Pfet can on be placed in series of 3 high and Nfet device is 4 high. For stacking design such as Nand or Nor table
4.2 provides a scaling factors for sizing Nfet and Pfet devices since stacking device require more special attention due threshold voltage variation and the floating body for SOI technology in particular. For example, 3 Pfet device stacking high the width W is divided by 2.95 and 3 stack high Nfet is scaled by 2.44. A stack 4 Pfet devices is forbidden at IBM due to the limit of SOI technology at IBM and charge leakage and hysteretic of the Pfet device. because of the effective resistance of both N and Pfet devices.

To illustrate a simple and relatively effective technique of sizing transistor, a set of simple equations are given below. These equations are very simple and using the design parameter provided in Table 4.1 and 4.2, together this technique works rather well for initial sizing the transistor level design. It may take few iterations to finally have the design specifications met. At IBM, after sizing the transistors in the design, an EinsTLT is used to verify to whether all the timing specifications of the design are met. EinsTLT is a static timing analysis, it analyses the design at transistor level to ensure there are no violations in terms of timing, slews and power are met. Equation (4.1) shows a simplified expression for finding the size of Nfet transistor given the beta, gain of logic stage and the output capacitances are known.

\[
\text{fetsize}_N = \frac{C_{\text{load}}}{H(\beta + 1)}
\]

where \( \beta \) is the ratio of Nfet and Pfet devices, and \( H = C_{\text{load}} / C_{\text{in}} \) is the Gain of a logic stage. \( C_{\text{load}} \) is the total capacitance seen by the driver which includes wires, \( C_{\text{wire}} \) and the input gate’s capacitances of the receiving stage, \( C_{\text{in}} \), in terms of its width. The input stage can be any type of static logic gate. The amount of capacitances that associate with the routing layer is called \( C_{\text{wire}} \) which depends on the estimated length and the metal layer. The output capacitance for the driver is defined as.

\[
C_{\text{load}} = C_{\text{wire}} + \sum_{j=0}^{N} C_{i,j}
\]

\( C_{i} \) is the sum of total input gate’s capacitances in gate/\( \mu m \) equivalence of a receiving logic state in terms of the transistor’s widths or \( C_{i} = (p + n) \). \( N \) indicates the number of
gates. From equation (4.1), the Gain can also be expressed as a function of the width of the Nfet transistor as seen below

$$H = \frac{C_{\text{load}}}{fetsize_N (\beta + 1)}$$

(4.3)

The delay can be estimated using the below equation from the Logical Effort (LE) technique [15]. $h$ is the ratio of $C_{\text{out}}/C_{\text{in}}$, $p$ is the parasitic or an intrinsic capacitance expressed in $\text{gate/um}$. Therefore, the delay for a logic stage can be estimated as

$$d = (hg + p)\tau$$

(4.4)

As it can be seen in equation (4.4), the $h = C_{\text{load}}/C_{\text{in}}$ is the same as $H$ in equation (4.3) thus we can substitute $H$ for $h$ in equation (4.4). $g$ is defined as the electrical gate effort for certain type of logic gate which indicates the strength of the output current of the gate being used in the design. $\tau$, is the delay that associates with certain technology process for a fan out of 1. $FOI$ is defined as a minimum size inverter driving an identical copy, and the delay for a $FOI$ for the technology implemented here is approximately 2.5ps. Substituting equation (4.3) into (4.4) leads to an expression for a delay stage,

$$d = [g \frac{C_{\text{load}}}{fetsize_N (\beta + 1)} + p] \tau$$

(4.5)

For an example, an inverter driver of unknown size needs to be appropriately sized in order to drive 6 inverters of size 6/3 $\mu m$ from a distance of 300$\mu m$ using minimum width metal1 then the size of the inverter that drives the 6 inverters can estimated as follows,

$$C_{\text{in}} = (p + n)$$

$$C_{\text{in}} = 6 * (6 + 3)\mu m = 54\mu m \text{ (Total of 6 inverters' input gates)}$$

Since the length of the wire is 300$\mu m$ with minimum spacing and the metal layer being used is metal1 from the table 4.1 the capacitance per 1$\mu m$ is 0.054fF. The conversion factor to $\text{gate/um}$ for metal1 is 0.35$fF/\mu m$. Using the parameters in table 4.1, the size of the driver can be calculated assuming $\beta$ is 2 and the gain is 3. The $C_{\text{eff}}$ is estimated as.
\[ C_{\text{wire}} = 300 \mu m * (0.051 \frac{fF}{\mu m}) = 15.0 \frac{fF}{\mu m} \]

Converting to gate micron,
\[ C_{\text{wire}} = 15.0 \frac{fF}{\mu m} * \frac{\mu m}{0.35 fF} = 43.7 \mu m \]
\[ C_{\text{load}} = C_{\text{wire}} + C_{\text{in}} = 43.7 \mu m + 54 \mu m = 97 \mu m \]

Using equation (4.1), the width of the Nfet can be calculated as,
\[ fetsize_N = \frac{C_{\text{load}}}{H(\beta + 1)}, \quad fetsize_N = \frac{97 \mu m}{3(2 + 1)} \approx 10.8 \mu m \]

since the \( \beta \) is 2 thus the Pfet size is,
\[ Pfet = \beta(Nfet) = 2(10.8 \mu m) = 22.6 \mu m \]

The electrical effort for the inverter is 1 and ignoring \( p \), the delay is

\[ d = [g \frac{C_{\text{load}}}{fetsize_N (\beta + 1)} + p]r \quad \rightarrow \quad [(1) \frac{97 \mu m}{10.8(2 + 1) \mu m} + 0](2.5 ps) = 7.5 ps \]

Using the process described above, one can estimate the sizes of the static gates and the amount of delay that associate with certain logic stage. Calculating parasitic \( p \) capacitance for Nfet and Pfet devices is relatively involved and does not significantly improve the accuracy therefore it can be ignored. To account for the parasitic capacitance \( p \), we can add 5% to 15% margin into the design.
The set of equations above can be used to estimate the size for any transistor sizes, it may takes few iterations to have timing and slews to converge and the most importantly it provides a quick way to estimate the sizes of the transistor in the design without involving complex equations. With the simplicity these equations, these can be programmed to quickly do the conversion and calculate all the sizes for more complicated transistor structures.

Dynamic logic has become increasingly important in high density VLSI design because of compatibility with CMOS fabrication and relatively easy to integrate into the CMOS design process because of its area efficient requirement and delay and power performance characteristics however in order for the dynamic logic circuit to optimally perform the circuit has to be fine tune for delay, power and performance. The procedure to be used in this is to iterate and scale the according starting with footed device and tapering the Nfet devices as shown in Figure 4.10. Dynamic logic has become increasingly important in high density VLSI design because of compatibility with CMOS
The set of equations above can be used to estimate the size for any transistor sizes, it may take few iterations to have timing and slews to converge and the most importantly it provides a quick way to estimate the sizes of the transistor in the design without involving complex equations. With the simplicity these equations, these can be programmed to quickly do the conversion and calculate all the sizes for more complicated transistor structures.

Dynamic logic has become increasingly important in high density VLSI design because of compatibility with CMOS fabrication and relatively easy to integrate into the CMOS design process because of its area efficient requirement and delay and power performance characteristics however in order for the dynamic logic circuit to optimally perform the circuit has to be fine tune for delay, power and performance. The procedure to be used in this is to iterate and scale the according starting with footed device and tapering the Nef devices as shown in Figure 4.10. Dynamic logic has become increasingly important in high density VLSI design because of compatibility with CMOS.
fabrication and relatively easy to integrate into the CMOS design process because of its area efficient requirement and delay and power performance characteristics however in order for the dynamic logic circuit to optimally performance the circuit has to be fine tune for delay, power and performance. The procedure to be used in this is to iterate and scale the according starting with footed device and tapering the Nfet devices as shown in Figure 4.10.

![Figure 4.10: LSD AOI Nfet device sizing](image)

Dynamic circuits sizing is very much different to static circuit in terms of transistor level because dynamic circuits involve Nfet stacking device and a Pfet load device as a load and not to mention the Nfet devices have to be charged and discharged periodically. In a way dynamic circuit is similar to Pseudo Nmos circuit however the Pfet device has to sized adequately in order to hold enough charges during precharge cycle. If the Pfet load device is sized too large then it would take much longer to discharge and this would not be able to operate at the desired frequency. And also if the Nfet devices are too large in
Fabrication and relatively easy to integrate into the CMOS design process because of its area efficient requirement and delay and power performance characteristics however in order for the dynamic logic circuit to optimally perform the circuit has to be fine tune for delay, power and performance. The procedure to be used in this is to iterate and scale the according starting with footed device and tapering the Nfet devices as shown in Figure 4.10.

![DIAGRAM](image)

**Figure 4.10: LSD AOI Nfet device sizing**

Dynamic circuits sizing is very much different to static circuit in terms of transistor level because dynamic circuits involve Nfet stacking device and a Pfet load device as a load and not to mention the Nfet devices have to be charged and discharged periodically. In a way dynamic circuit is similar to Pseudo Nmos circuit however the Pfet device has to sized adequately in order to hold enough charges during precharge cycle. If the Pfet load device is sized too large then it would take much longer to discharge and this would not be able to operate at the desired frequency. And also if the Nfet devices are too large in
terms of width and the Pfet load device is too small the circuit would not be able to hold enough charges for a whole clock cycle.

In general, according to [13], the Nfet chain device need to be scaled in a tapering fashion to minimize the delay. For an AND or NOR gates, the series Nfet devices are scaled starts with the bottom or footed device with a fixed size \( W \) and progressively scaled down smaller and smaller size until the last Nfet is reached as shown in Figure 4.10. The device \( M1 \) has width of 5 and the last Nfet device is 2.4. Scaling CMOS AOI dynamic circuit is similar to static CMOS transistor sizing procedure. Sizing Dynamic circuit also starts with the capacitance load that circuit is designed to drive and to meet the slews and timing criteria or constrains the same way as static CMOS. The circuit starts with a minimum sizes device and being scaled according to the load that it is driving. Because of the nature of Domino logic circuit thus the transistors are progressively sized smaller as closer to the dynamic node to reduce this node capacitance and also to make the layout more area efficient. The set of schematic starts with some minimum values as described above and the schematic is fed into a tool called Einstuner which is used to tune Dynamic circuits. This procedure may make few trials and errors before the design converge to meet design specifications. Due the complexity of the deriving for scaling the dynamic circuits therefore the equations are not included in this papers except some basic MOSFET's equations.
Chapter V

Adder Designs and Implementations

This chapter describes the procedures that implement the adders in terms of using IBM's and Cadence's VLSI design tools. Section 1 presents the designs of the two adder, static and LSD, in terms of implementing the architecture basing on the Boolean expression that were derived in the previous, the essential equations are included for reference purposes and taking into an account the number of gate to implement the static designs. Part 2 describes the physical designs aspects of the adders and the simplified physical design flow at IBM using internal and vendor VLSI design tools. Also included is the design parameters in the provided table, device's parameter in terms of gate, metal layers, metal thickness and the availability of the metal for routability also pitch and metal width. These parameters are extremely useful in the early stage of the design using these model the delay and timing of the circuit, macros, can be estimated. The designs will be iterated until macros meet all design criteria.

5.1 LSD 64-bit and Static CLA adder designs

The implementation of the adder is based on second level abstraction architecture described in chapter 3 which using the below expressions.

\[ P_{30} = p_3 p_2 p_1 p_0 \]  
\[ g_{30} = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 \]  
\[ c_{i+1} = G_i + P_i c_0 \]  
\[ s_i = c_{i-1} \oplus (a_i \oplus b_i) \]

Expressions (5.1 – 5.2) form a block which generates the super propagate and generate signals called \( p_{30} \) and \( g_{30} \) or \( P_i \) and \( G_i \). The equations (5.3) is implemented as a dedicated carry block which inputs are the generate and propagate, \( p_i \) and \( g_i \) signals from the individual bit \( x_i \) and \( y_i \).
The 64-bit static and LSD CMOS adders are designed using the Boolean expressions described above. To implement the static CLA is relative straight forward, the static adder is directly implemented using the equations described in chapter 3 and 5. These equations are implemented using CMOS static standard library gates which primarily consist of NAND, NOR, XNOR and INVERTER gates therefore the final designs have to be optimized in order to reduce delay and power consumption.

For the LSD adder is however more difficult since the equations expressed in terms of gate level and the LSD is a dynamic circuit logic style in nature and only monotonically evaluate the inputs therefore if an expression that involves an XOR which requires dual polarity from inputs. The LSD requires a non-inversion output in order to accommodate this polarity different. This inverted output can be easily implemented in the LSD logic style.

5.2 Physical design tools and CLA adder layouts

The physical design is an important part of the IC design process, there are many iteration which have to be done in order to finalize the design since the schematic captures most often do not necessarily include all the details and all the intrinsic and parasitic capacitances of the devices and on top of that the lump RC delays model is relative accurate a short distance however these are notoriously bad at a long interconnect wire. Usually, the lump and PI-RC model are only good a for certain optimal length of wire, besides accurately model and reflect all the modeling detail and delay in the schematic capture quite often take or at least require much more time, a lot of time to see the least, as a result it is often requires several iteration to get the design working properly.
Figure 5.1 shows a simplified chart for a schematic transistor design process at IBM which typically requires many iterations depending on the initially estimated sizes of the transistors. The initial sizes of the transistors are usually estimated and calculated using the techniques described in chapter 4. The calculations take into account all capacitance loading including wiring tracks and layers that the transistors are intended to drive.

The wirings could be between metal 1 to 3 and track is the width of the wire, which can be minimum, double or triple wide. After the transistors are approximately sized then the schematic netlist is extracted, simulated and finally taken through the static timing analysis tool called EinsTLT. Since every design has specifications and the EinsTLT is used to check to see whether the design is to meet the desired specifications, this is a prelayout schematic design phase. If the schematic design meets all the desired criteria then the design is translated to a physical layout of all the transistors using Cadence’s Virtuoso XL physical design tool. This Virtuoso is used to do place and route the design and all the necessary checks such as DRC, LVS, and METH are performed.
using this tool. The finally step is to extract the layout and the design once again to be simulated and static timing check is performed to ensure the design meeting all the specifications however the whole process is repeated again if the design does not meet the desired specifications.

*Figure 5.2* depicts a flow chart that describes physical design process at IBM using Cadence computer aided design tools, using Virtuoso. The Virtuoso XL is a custom place and route computer aided design tool available at IBM physical design environment, this helps to speed up the physical design process. Virtuoso XL is proven to be extremely helpful in translating schematic to physical designs even though it may take several iteration to get the design converge in terms of timing. Conventionally, converting schematic circuit to a physical design (layouts) is a manual laboring process however with the design becomes more and more complex and with thousand and thousand of transistor being put in small and smaller area this manual design is no longer feasible it is not only the design itself but also when the design need to be modified to include new logic or incorporate new feature in the designs. Even though, this process seems similar to synthesis but it is not a synthesis process because the design who still full control over the design unlike the automatic synthesis. Using Virtuoso XL, the designer has the ability to optimize in terms of placing and routing the design.
The end product is not as good and optimal as full custom design but the trade off is time saving and this saving is proven very critical, in generally the design cycle is getting shorter and shorter in the VLSI business. Custom design, in practical terms, means control over the circuit styles, topology, device sizes and the physical designs of both transistors and interconnects. By manually design the macro, the circuit can be optimized and minimizing the parasitic capacitances and the number stages. logic gate level in order to minimize delay vs. the ASICs or cell based design, synthesis.
5.3 IBM’s CMOS8S0 SOI technology

At IBM the new technology which is currently available at IBM is the SOI (Silicon On Insulation) is the partially depleted as opposed to the fully-depleted since the partially depleted is much more stable and more closely comparable to the conventional CMOS process with channel length of 0.08 and gate oxide thickness 10 Angstrom (Å). The CMOS8S0 FET is Partially Depleted (PD) SOI devices. In a PD-SOI FET, the body of the device is not fully depleted of charge carriers at a gate bias equal to the threshold voltage.

<table>
<thead>
<tr>
<th>Gate $L_{\text{off}}$</th>
<th>0.09 $\mu$m</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gate oxide</td>
<td>2.3 nm</td>
</tr>
<tr>
<td>Metal layers</td>
<td>width thickness</td>
</tr>
<tr>
<td>$M1$</td>
<td>0.5 $\mu$m 0.3 $\mu$m</td>
</tr>
<tr>
<td>$M2$</td>
<td>0.63 $\mu$m 0.3 $\mu$m</td>
</tr>
<tr>
<td>$M3-M5$</td>
<td>0.63 $\mu$m 0.42 $\mu$m</td>
</tr>
<tr>
<td>$M6(MQ)$</td>
<td>1.26 $\mu$m 0.92 $\mu$m</td>
</tr>
<tr>
<td>$M7(ML)$</td>
<td>1.26 $\mu$m 0.92 $\mu$m</td>
</tr>
<tr>
<td>Dielectric $\varepsilon_r$</td>
<td>$\sim 4.2$</td>
</tr>
<tr>
<td>$V_{dd}$</td>
<td>1.6V</td>
</tr>
</tbody>
</table>

Table 5.1 The IBM CMOS8S0 features

This type of design is considered to have manufacturability advantages since the low bias drain bias $V_t$ does not depend on the device silicon layer thickness or the amount of oxide charge in the buried oxide (BOX). Since the quasi region exists when the device is on.
charge can accumulation can occur in the electrically isolated body, this leads to some unique effects. The Nfet and Pfet extension, halo and source or drain implant have been optimized to reduce the floating body effects inherent in SOI FETs. The magnitude of the floating body effect has been reduced by increasing the junction leakage significantly. The major difference between Silicon On Insulation (SOI) and Bulk Silicon is that the SOI devices are fully oxide isolated. Individual device region define by the RX mask level are isolated laterally by the shallow trench oxide and from the substrate below by the buried oxide (BOX). This isolation improvement results in a number of features and effects which are unique to SOI technology. N+/P+ and well spacing constraints due to latch up are relaxed due to the improved isolation. The improved circuits performance provide by SOI technology is partially due to a reduction in junction capacitance.

The technology being used in this project to implement the adders is 0.18μm CMOS CMOS8S0 SOI technology at IBM, as shown in Table 5.1. This is the fifth generation of SOI technology being used to fabricate state of the art microprocessor at IBM, it features seven level of copper wiring layers, with a special level of slightly doped which is used for standard cell connection on besides the metal one layer for interconnection within the standard cell. This special layer provides an extra layer and proven extremely useful and area saving compare to that of other process since standard and memory cells are being used in many places and repeated so many times in the chip/microprocessor therefore it is imperative to make these cell as small as possible in order to reduce power consumption and make the design as compact and optimal as possible thus eliminating the need for using metal 2 to connect signals for a complex cell such as XOR/XNOR. Table 5.1 summarizes some basic features of the CMOS8S0 SOI technology parameters such as the effective gate length and gate the gate oxide thickness and several metal width and thickness of the metal routing level.
5.4 Design checking in physical design

Design checking at the transistor level provides the designers with a degree of freedom not allowed in a cell-based methodology. However, this comes with a price, expanding the design space to include additional circuit topologies and families demands additional rigor checking so that improper design circuits do not find their way into manufacturing. The definition of “best design practices” is subjective and depends on many factors, class of circuits. In general, it is a set of guidelines to limit the range of possible legal and illegal circuits. Einscheck provides a mean for discriminating legal and illegal circuits constructs and device sizes and perform a basic checks on electrical relationship such as noise margin, capacitive coupling and interconnects current carrying capacity. The checking strategy was instrumental in ensuring that the final chip could be assembled with a minimum of problems. Because of the size of the chip, a large number of problems at the end of the design cycle would be too difficult to detect and fix.

Therefore, checking methodology was developed to treat each unit and the core as a “chiplet”. In addition, a robust set of methodology checks were developed to ensure that all macros, units, and the core could be correctly integrated at the next level of hierarchy.

In order to check an entity as a chiplet, it is necessary to understand the environment in which the chiplet resides in the chip. To model this environment, the cover (routing contract with the parent) and the parent cover (fixed chip level infrastructure) were added to the unit for DRC and LVS verification. Since the cover contains blockage layers, as opposed to manufacturable shapes to the blockage of the same layers, a separate set of checks were included. No minimum area checks were done on blockage shapes, since they are not required to comply with the area rules. A separate methodology for DRC deck was created to check additional design constraints above and beyond design rules necessary for manufacturing specified in the design rules. These checks concentrated on ensuring the quality of the blocks as well as their ability to be integrated at the next level of hierarchy. The types of design constraints checked included ensuring all manufacturable shapes are “one-half ground rule” for the boundary floor-plan block, power buses are one the correct periodicity, and clock pins are in the correct
tracks. Addition checks maintained the quality of the design for routing, for example pins are checked to ensure that they are on grid and accessible from the same layer or the layer above.

The final step in physical checking is to extract the design to get a representative netlist that includes resistance and capacitive coupling to a net, in some cases the nets also coupled with noise depending the layer of the metal that the nets are connected to. With this extracted netlist, all the checks are performed again including static timing analysis and simulation to finally verify the design to whether all the specifications are met.
Chapter VI
Adder Evaluations

This chapter is the overall analysis of the performance of the two adders which are constructed with two different circuit styles and topologies, one is conventional static and the other is LSD logic style. The adders will be evaluated on the power consumption, performance and area and ease of technology mapping point of views. For the power consumption, each adder block is simulated using Power Spice which is similar to the regular Spice circuit simulator to extract the power consumption and then all of the block are put together and the values of the power consumption estimated since using equations described in the previous sections also the model being used for power simulation. As for the performance, the adders will be evaluated also using PowerSpice for its clock rate and capacitance load which is designed to drive and other various aspects of the designs.

In a conventional dynamic or domino circuit styles, which consists of precharge and evaluate cycles. The clocking scheme typically employed single or multiple phases depending on application. In a dynamic topology, the circuit can only implement inverting function that is the signal can only transition from high to low versus domino circuit is applicable for implementing non-inverting function which can only changes from low to high. Since either of these circuits styles, dynamic or domino, consist any latch or storage element as a result the charge and precharge cycle continuously charge and discharging the circuit therefore consuming extra power to refresh the value even the output value doesn’t change.

6.1 Static adder performance evaluations

Static Circuit Power Consumption: Power consumption appears to be primarily related to clock switching at the latches. The total energy consumed in the first seven cycles (0.394ns to 5.997ns) is about 6.683mWns, which includes all of the functional switching for the simulation. The total energy consumed during the next seven cycles (5.998ns-11.600ns), during which no functional switching takes place (but the clock is still
switching), is 4.389mWns. The total energy consumption for a cycle in which clocking is the only switching going on was 0.627mWns. The total energy consumption for the first half of this simulation (0 12.5ns) which includes all of the simulation's functional switching and some time of no functional switching was 11.972mWns. The total energy consumption for this simulation (25ns) is 21.731mWns.

### 6.2 LSD adder performance evaluations

SOI technology has been touted over the years as a great process for reduced power consumption due to the reduction of total capacitance on the device as well the chip. As shown in equation 6.1, it shows the relation between power (P), capacitance (C), voltage (V), frequency (f) and the switching activity. The power consumption savings come primarily via

\[ p_{\text{dyn}} = \frac{1}{2} f_c C_i V_d^2 \]  

(6.1)

two opportunities. The first is the total capacitance has been diminished due to the reduction of the diffusion capacitance. Since the junction capacitance is not only capacitance on the chip, this results in approximation of 10% to 15% reduction power savings compared to bulk process. Second, With the performance advantages possible with SOI technology it is possible to operate the circuits at the same frequency however with a lower voltage since the power consumption depends non-linearly with \( V_d^2 \) this resulting to a much larger in power saving. Makes it possible to have a larger saving in power consumption comparing to bulk CMOS. Comparing power consumption between static and LSD circuits is also somewhat difficult because common patterns don't exercise the two circuit families' worst case conditions. and there exist different means by which power is consumed in the two types of circuits.

In a convention dynamic or domino circuit styles which consists of precharge and evaluate cycles. The clocking scheme typically employed single or multiple phases depending on application. In a dynamic topology, the circuit can only implement inverting function that is the signal can only transition from high to low versus domino.
circuit is applicable for implementing non-inverting function which can only changes from low to high. Since either of these circuits styles, dynamic or domino, consist any latch or storage element as a result the charge and precharge cycle continuously charge and discharging the circuit therefore consuming extra power to refresh the value even the output value doesn’t change.

**LSD Circuit Power Consumption:** There was a comparable amount of functional switching going on during this simulation as compared to that for the static, although it was carried out over a longer period of time. The total energy consumption for the first half of this simulation (0 to 12.5ns) which includes all of the simulation's functional switching and some time of no functional switching was 6.002mW. The total energy consumption for this simulation (25ns) is 17.113mW. The fact that the LSD circuits do consume dynamic power under certain data input conditions seems to be offset by the fact that there is less clock-controlled device width required than in the static equivalent.

As far as noise is concerned, several experiments were performed to study the effects of unwanted signal coupling onto LSD input signals and the overall impression is that a very significant and unusual condition must exist for noise coupling to cause a functional problem. All the noise simulations were performed using the "Noise" simulation corner provided in the ASX environment, which is believed forces the worst case noise condition. Other conditions may exist where coupling onto the dynamic node or some other node local in the LSD stage may occur, but typically vigilant physical design methods should eliminate these extreme cases. If the dynamic node connection is not made longer than 500um long then this would significantly reduce the LSD circuit to falsely switch.

Generally speaking, power consumption in static circuits is dominated by transistor gate switching, internal gate and diffusion nodes switching during functional evaluation, and all the interconnecting wire capacitance switching. (For this discussion, power consumption due to leakage will not be considered.) In the LSD circuit family, in addition to all of these intrinsic means of power consumption listed for static circuits,
there also exists power consumption related to the dynamic clocking of the circuits. Specifically, depending on the inputs to the evaluation tree, the "dynamic node" will recharge (from gnd to vdd) after each cycle that the evaluation tree fully dissipates the dynamic node charge to gnd. Also, nodes internal to the evaluation tree that were part of a complete discharge path from the dynamic node to gnd can charge and discharge with or without the stage's output switching. These two types of power consumption have been referred to as secondary or implied clocking power. They are solely data dependent, and under worst-case conditions, can account for a significant portion of the total power consumed in a design. If the inputs are held low cycle after cycle, there will exist a constant path of conduction from the dynamic node to the node just above the evaluation tree's foot device. Even though that stage's output will not switch from cycle to cycle, the dynamic node and all the internal nodes along the path of the input nfets will charge during each precharge phase and discharge during each subsequent evaluation phase, cycle after cycle (again, with no functional output switching even happening).

Depending on the worst-case secondary clocking power consumed in the logic, it seems unlikely that the overall power consumption of an LSD design would surpass that of a static equivalent. However, as the LSD design will likely be smaller, power density problems could arise. This could be mitigated by introducing decoupling capacitors in the design, but this would offset the area benefit to some degree.

Table 6.1 summarizes the results of the two adders. The physical layout area of the LSD's adder is approximately one fourth the size of the Static one, this is due to the size of the LSD circuits and the reduction of using Pnet devices since Dynamic circuit is mostly consists of Nfet devices and they are relatively smaller size compare to Static CMOS circuit. The power consumption of the LSD adder is roughly 20% less than the Static but this power reduction of the LSD adder does not reflect the fact that it is one-fourth the size of the Static adder, this is because the constant switching nature of Dynamic circuit. Both of these adders are operated at ~200ps cycle and the adders' outputs are measured using EinsTLT which is a transistor level static timing analysis internal tool at IBM.
<table>
<thead>
<tr>
<th>Type of adder</th>
<th>Number of gates</th>
<th>Dimension (μm)</th>
<th>Area (μm$^2$)</th>
<th>Power (mW)</th>
<th>Output valid (ps)</th>
<th>Cycle (ps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static CMOS</td>
<td>476</td>
<td>330x675</td>
<td>222000</td>
<td>626</td>
<td>123ps</td>
<td>~400</td>
</tr>
<tr>
<td>LSD CMOS</td>
<td>n/a</td>
<td>90x190</td>
<td>51000</td>
<td>492</td>
<td>90ps</td>
<td>~400</td>
</tr>
</tbody>
</table>

Table 6.1: Comparison between Static and LSD adders

Included are the layouts of both adders with routing metals being turned off for easy viewing and the heretical of both adders are maintained between schematic and physical layouts.

64-bit Static CMOS CLA Area 210000, 400x500
<table>
<thead>
<tr>
<th>Type of adder</th>
<th>Number of gates</th>
<th>Dimension ($\mu m$)</th>
<th>Area ($\mu m^2$)</th>
<th>Power (mW)</th>
<th>Output valid (ps)</th>
<th>Cycle (ps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static CMOS</td>
<td>476</td>
<td>330x675</td>
<td>222000</td>
<td>626</td>
<td>123ps</td>
<td>~400</td>
</tr>
<tr>
<td>LSD CMOS</td>
<td>n/a</td>
<td>90x190</td>
<td>51000</td>
<td>492</td>
<td>90ps</td>
<td>~400</td>
</tr>
</tbody>
</table>

Table 6.1: Comparison between Static and LSD adders

Included are the layouts of both adders with routing metals being turned off for easy viewing and the heretical of both adders are maintained between schematic and physical layouts.

![64-bit Static CMOS CLA Area 210000, 400x500](image)
64-bit LSD CMOS CLA Area 51000, 90x190
64-bit LSD CMOS CLA Area 51000, 90x190
Chapter VII
Conclusions and future works

In an attempt to solve some of problems of the conventional dynamic and domino logic styles, this thesis presented a new logic dynamic circuit style which is called LSD (Limited Switching Dynamic) due to its latching capability and also included the design and implementation of the CLA (Carry Look Ahead) adder. In the process of design and implement the two adder, various internal tools are IBM were use to, at the very least, preprocessing of design data for the Verity, EinsTLT, and Echk tools via SPAM (IBM's Subcircuit Pattern Matching tool) would have to be taught a variety of things (e.g., electrical behavior) with regard to LSD. Initial attempts at running Verity on LSD circuits failed, apparently due to Verity's inadequate comprehension of the net list topography.

The two adder designs in this experiment performed relatively well however the LSD circuits are much more sensitive to wire loading, clock skews and data setup time thus in design all the dynamic stages have to be careful in terms of device sizing, and the driver of the output stages. Also, according to the IBM's CMOS8s0 technology manual states that in the interest of accurate model to hardware correlation, FETs devices' widths should not be smaller than $2 \mu m$ and avoid long wire as a direct gate's input. As far as the physical design for the adder is concerned, it's much more challenging to layout and wire the LSD adder compare to that of Static adder because differential inputs and outputs that require for some the LSD stages and also LSD circuit is more sensitive to capacitive loading, data and clock skewed than Static circuit. For these reasons, designing with LSD circuit takes a lot more efforts, time, careful planning and the most important of all the design has to rigorously check and test if it is to be fabricated into real Silicon.

Since the growth of VLSI's ending is not in sight for a long time and the microprocessor will only get more and more complex with forever packing more feature, transistor count, power consumption and increasing speed therefore it only make sense to include more and more Dynamic and other logic circuit styles in order to alleviate some
of the problems and speed up the critical paths, especially data oriented paths in highly number crunching applications. Designing VLSI using CMOS SOI technology is some more different than Bulk CMOS because SOI is relatively new comer in VLSI arena and the process is not completely well understood. On top of that, Dynamic circuits being fabricated on CMOS SOI is even more difficult to get good yields and performance due to Dynamic circuit model in SOI does correlate well between the model and the hardware however as the SOI technology becomes more mature and the modeling well understood, there is no doubt Dynamic circuits will be an important part in the VLSI. Despite some of the problems that Dynamic circuit posses, it still is very much inclusive in the IBM’s microprocessor design community where dynamic circuit serves as the last resort solution to remedy some of the area, power consumption and speed. Presently, Dynamic circuit has gained more acceptance and that does not expect to change in anytime near in the future.
Appendix A

Static schematics and layouts

This appendix includes the basic schematics of the static since the implementation of the static is directly based on the Boolean expression thus it relatively straightforward to understand the functionality of all the blocks of the CLA. A top level schematics of the 64-bit CLA and the Reduced full adder are included for referencing purposes. An equivalent physical layout of the adder is provided, with all the metal levels are turned off.

Schematic of Static 64-bit CLA
Appendix A
Static schematics and layouts

This appendix includes the basic schematics of the static since the implementation of the static is directly based on the Boolean expression thus it relatively straightforward to understand the functionality of all the blocks of the CLA. A top level schematics of the 64-bit CLA and the Reduced full adder are included for referencing purposes. An equivalent physical layout of the adder is provided, with all the metal levels are turned off.

Schematic of Static 64-bit CLA
Schematic of Static 4-bit RA (Reduced Adder)

Layout of Static 64-bit CLA
Schematic of Static 4-bit RA (Reduced Adder)

Layout of Static 64-bit CLA
Appendix B

LSD schematics and layouts

Schematic of LSD 64-bit CLA

Physical layout of LSD 64-bit CLA
Appendix B

LSD schematics and layouts

Schematic of LSD 64-bit CLA

Physical layout of LSD 64-bit CLA
Schematic of LSD 16-bit CLA

Schematic of LSD 4-bit gen&grop and sum
Schematic of LSD 16-bit CLA

Schematic of LSD 4-bit gen&grop and sum
Schematic of LSD 4-bit Gen&Prop, Carry Look Ahead

Schematic of LSD 2-bit Supper Gen&Prop
Schematic of LSD 4-bit Gen&Prop, Carry Look Ahead

Schematic of LSD 2-bit Supper Gen&Prop
Schematic of LSD Supper Carry Gen Blk
Schematic of LSD Supper Carry Gen Blk
Appendix C

SOI Partially-Depleted Simplified Drain Equations

The following MOSFET’s equations are extracted from the BSIMP4D models, it shows a simplified drain equation and its parameters. Unlike Bulk CMOS, the drain is much more complex and depends on many factors due the nature of SOI technology.

\[
I_{d_{s,MOSFET}} = \frac{I_{d_{0}}}{1 + \frac{R_{ds}I_{d_{0}}}{V_{d_{seff}}}} \left(1 + \frac{V_{d_{s}} - V_{d_{seff}}}{V_{A}}\right)
\]

\[
I_{d_{0}} = \frac{\beta V_{g_{s_{eff}}} V_{d_{seff}}}{1 + \frac{V_{d_{seff}}}{E_{sat}L_{eff}}} \left(1 - A_{bulk} \frac{V_{d_{seff}}}{2(V_{g_{s_{eff}}} + 2V_{A})}\right)
\]

where \( \beta = \mu_{eff} C_{ox} \frac{W_{eff}}{L_{eff}} \)

\( V_{th} \) → the Threshold voltage
\( A_{bulk} \) → Bulk charge factor
\( V_{d_{seff}} \) → the effective drain voltage
\( V_{g_{s_{eff}}} \) → the effective gate over-drive voltage
\( R_{ds} \) → Source and Drain series Resistance
\( \mu_{eff} \) → the effective electron mobility
\( E_{sat} \) → the critical Electrical field where carrier velocity become saturated
\( V_{A} \) → the Early voltage which accounts for channel length modulation
# Appendix D

## Static and Dynamic Power Calculations

Included below is the PowerCal panel which is used at IBM to estimate the power consumption for either Schematic or Physical layout of a macro or block. The power estimation algorithm is based on the formulas that were described in chapter II section 2.3. In reality, the algorithms that are used in power estimation tool is much more complex than the expressions that were given in section chapter II.

<table>
<thead>
<tr>
<th>Interactive or Batch Mode</th>
<th>Interactive</th>
<th>Batch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Library Name</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### INTERACTIVE:

#### Power Info Handling Mode

- **Calculate**
- **Query**
- **Add**
- **Update**

If in Query mode, descend to leaf cells?  **NO**  **YES**

<table>
<thead>
<tr>
<th>Cell Name</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell View</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Static area (%)</th>
<th>0</th>
<th>Switching Factor (%)</th>
<th>10</th>
<th>Power Density (W/mm²)</th>
<th>0.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic area (%)</td>
<td>0</td>
<td>Switching Factor (%)</td>
<td>100</td>
<td>Power Density (W/mm²)</td>
<td>1</td>
</tr>
<tr>
<td>Large array area (%)</td>
<td>0</td>
<td>Switching Factor (%)</td>
<td>100</td>
<td>Power Density (W/mm²)</td>
<td>0.2</td>
</tr>
<tr>
<td>Small array area (%)</td>
<td>0</td>
<td>Switching Factor (%)</td>
<td>100</td>
<td>Power Density (W/mm²)</td>
<td>0.9</td>
</tr>
<tr>
<td>CPL area (%)</td>
<td>0</td>
<td>Switching Factor (%)</td>
<td>20</td>
<td>Power Density (W/mm²)</td>
<td>0.3</td>
</tr>
</tbody>
</table>

| Clock Frequency (x1GHz) | 1 |

<table>
<thead>
<tr>
<th>Power Density (W/mm²)</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power (W)</td>
<td>0</td>
</tr>
</tbody>
</table>

### BATCH:

Input File | `/tmp/PowerIn.txt`

Output File | `/tmp/PowerOut.txt`
References


**Vitae**

Cong Nguyen was born in April 20\textsuperscript{th}, 1962 in Viet Nam. He migrated to the US in 1986 and finished undergraduate study in Columbus, Ohio. After graduation, he worked for Lucent Technology Inc, which was spun off from AT&T, in the area of VLSI, developing and modeling custom high speed Library cells for Read-channel computer product development. He moved, after few years, over to work for AA.net.com which is a network product division of PMC-SIERRA inc located in Allentown, PA. He worked on the SERDES (Serializer and Deserializer) chip and Analog circuit designs.

Recently, he is working at IBM in Poughkeepise and East Fishkill New York. The projects that he has been involved in are the PowerPc for eSever, main frame server and Intel low end series servers products. Currently, he is moving to IBM’s EDA (Electronic Design Automation) and Technology division to work and develop hardware models, and Electronic Design Automation software tools for the Server and ASICs design community.
END OF TITLE