Reduced Swing Domino Techniques for Low Power and High Performance Arithmetic Circuits

by

Shahrzad Naraghi

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada 2004

I hereby declare that I am the sole author of this thesis.
I authorize the University of Waterloo to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Shahrzad Naraghi

I authorize the University of Waterloo to reproduce this thesis by photocopying or other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Shahrzad Naraghi

The University of Waterloo requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Acknowledgements

First, I would like to thank my supervisor Professor Manoj Sachdev for his great guidance, support and patience. His advice and support were always greatly appreciated. I also want to thank Dr. Opal and Dr. Anis, my thesis readers.

I'd like to thank Bhaskar Chatterjee for his great help on my research; Phil Regier for his great help on computer problems; and my good friends for bringing me joy and laughter during these years that I was away from my family.

Most importantly, I'd like to thank my family for their supporting and encouraging comments, their love and faith in me.

Abstract

The increasing frequency of operation and the larger number of transistors on the chip, along with slower decrease in supply voltage have led to more power dissipation and high chip power density which cause problems in chip thermal management and heat removal process. Therefore, low power circuit techniques have gained more attention in VLSI design. In this thesis, a high performance (2.9GHz) 32-bit adder is designed and is used as a benchmark for applying a low power technique without performance degradation. A dual supply technique for compound domino logic is proposed and is implemented in the adder. With this technique we have achieved an average power saving of 22% and 3.5% performance improvement over the conventional design. In addition to that, a novel carry select adder is introduced for power and area saving.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The arithmetic logic unit (ALU) is the heart of every microprocessor and determines its throughput. The core of arithmetic logic unit is adder. Therefore a high performance adder is essential to maximize microprocessor's speed. However, the high data activity associated with this unit results in high power and thermal density leading to increased cooling costs. Thus, there is a critical need for breakthrough ideas in VLSI design methodology to reduce the adder power consumption while maintaining the high performance target.

Recently different design techniques have been proposed in order to decrease the power dissipation in different types of circuit design and the most efficient one is the power supply voltage reduction because of the quadratic relationship between power consumption and voltage supply. But the supply voltage decrease causes an increase in the delay time and reduces the circuit speed. Therefore, we need to investigate new techniques for reducing the power dissipation without any performance penalty.

The aim of this research is to explore different circuit techniques and propose some new

methods that can be used in the adder structure to reduce its power consumption without performance degradation.

1.2 Thesis Organization

The thesis is organized as follows. Chapter 2 provides background information on existing adder architectures and introduces a hybrid architecture for designing a high performance 32-bit adder. Chapter 3 provides the back ground information on different logic styles used in today's digital circuits. Different styles are compared in terms of speed, power dissipation, robustness and silicon area and the selected logic styles for implementing different parts of the hybrid 32-bit adder are addressed. Chapter 4 introduces the Logical effort as a back of the envelope calculation method for transistor sizing. This method is then applied to the designed 32-bit adder to obtain the maximum speed. In Chapter 5, some high performance and low power circuit techniques are investigated and a dual supply technique for compound domino logic is proposed for power reduction without performance penalty in digital circuits. In chapter 6, the proposed dual supply technique is applied to the design of high performance 32-bit adder and its power and delay are compared with the conventional designs'. Chapter 7 concludes the thesis with a summary and possible future work.

Chapter 2

Adder Architectures

2.1 Introduction

The design of faster, smaller and more efficient adder architecture has been the focus of many research efforts and has resulted in a large number of adder architectures. Each architecture provides different insight and thus suggests different implementations. The aim of this chapter is to provide the essential background information to some of the well known adder architectures.

2.2 Full Adder Operation

Operation of a full adder is defined by the Boolean equations for the Sum and Carry signals:

$$S = A \oplus B \oplus C_{in} \tag{2.1}$$

$$C_{out} = AB + BC_{in} + AC_{in} \tag{2.2}$$

There is another way to calculate Sum and Carry based on two variables, Propagate (p) and Generate (g) which are defined for each pair of inputs as follows:

$$P_i = A_i + B_i \tag{2.3}$$

$$G_i = A_i B_i \tag{2.4}$$

Then, Sum and Carry outputs for each input pair are computed with these parameters in the following way:

$$C_{out} = G_i + P_i C_{in} \tag{2.5}$$

$$S_i = P_i \oplus C_{in} \tag{2.6}$$

According to the above equations, whenever 'G' for a pair of bits is one, the output carry will be set to one and we do not need to know about the input carry and whenever 'P' is one and 'G' is zero, the input carry will be directed to the output without any change. These properties of 'P' and 'G' signals are useful to make high performance adders as will be seen in the following sections.

2.3 Ripple Carry Adder

A ripple carry adder for N-bit number is implemented by concatenating N full adders as shown in Figure 2.1 [1]. At the i_{th} bit position, operands A_i and B_i and the carry output signal from the preceding stage are used to generate the i_{th} bit of the sum (S_i) and the carry output (C_i) to the next adder block. This is called a ripple carry adder, since the

carry signal ripples from the least significant bit position to the most significant one. The delay through the circuit depends on the number of logic stages that must be traversed and is a function of the applied input signals [1]. It is important to know that, for some input signals no rippling effect occurs while for the others, the carry has to ripple all the way from the least significant bit (lsb) to the most significant bit (msb) which is the worst case delay over all possible input patterns. Therefore, the N-bit ripple carry adder worst case delay is:

$$t_{adder} = (N-1)t_{carry} + t_{sum} \tag{2.7}$$

Where, t_{carry} is the propagation delay from the input carry to the output carry signal and t_{sum} is the propagation delay for 'S' (sum) at the output of each full adder block. This delay shows the longest delay in the circuit or the critical path delay. As we can see in the above equation, the longest delay increases linearly with the number of input bits; therefore this kind of adder architecture can not be used in high performance processors which are designed for more than 64-bit data path.

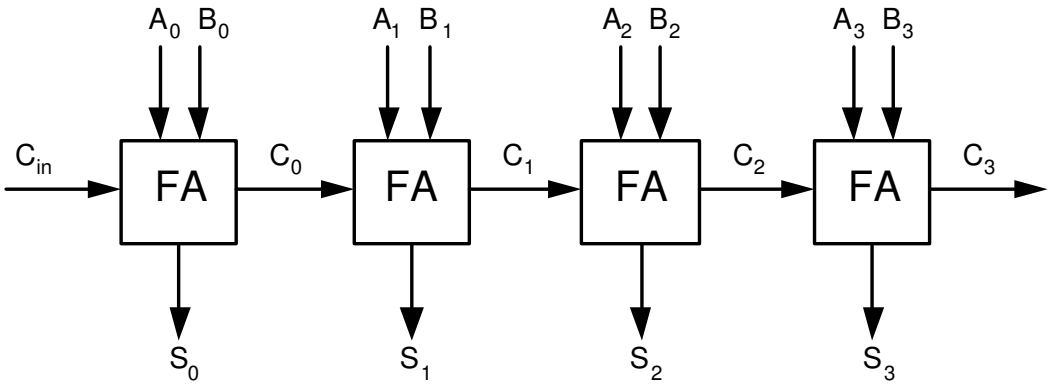


Figure 2.1: 4-bit ripple carry adder

2.4 Carry Skip Adder

Since the path from C_{in} to C_{out} represents the longest path in the ripple carry adder, an obvious attempt is to accelerate carry propagation through the adder. This is accomplished by using carry propagate signals within a group of bits. If all the P_i signals within the group are one, the conditions exist for the carry to bypass the entire group. The structure of a carry bypass adder is shown in Figure 2.2. We divide the N bit inputs to $\frac{N}{M}$ groups of M bits and for each group we do a carry skip addition. The worst case delay for this adder can be derived by the following formula [1]:

$$t_p = t_p g + M t_{carry} + (\frac{N}{M} - 1) t_{bypass} + M t_{carry} + t_{sum} \tag{2.8}$$

Where t_{pg} is the time required to create 'P' and 'G' signals, t_{carry} is the propagation delay of the carry signal through one block, t_{bypass} is the delay through the bypass multiplexer of a single stage and t_{sum} is the time to generate the sum of the final stage. As we can see in the above formula, the worst case delay is still proportional to the number of input bits but it is less than ripple carry adder delay. This architecture increases circuit complexity as well.

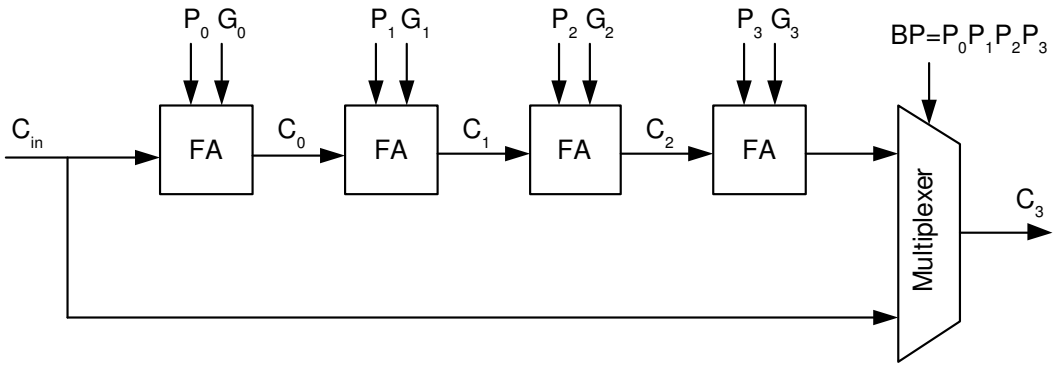


Figure 2.2: Carry skip adder structure

2.5 Carry Select Adder

The basic idea in carry select adder (CSLA) architectures is to save the time, waiting for the carry inputs to arrive by doing two sets of operations [1]. The CSLA divides the input words into a number of blocks and generates two sums for each block in parallel, one assuming a carry input of zero and the other assuming a carry input of one; therefore it is not required for every full adder to wait for the incoming carry before doing the addition. As it is shown as an example of a 16-bit carry select adder in Figure 2.3, the carry output from the previous blocks controls a multiplexer that selects the appropriate sum. If the number of input bits is N and we divide all the inputs to $\frac{N}{M}$ groups with M bits in each, then the worst case propagation delay is computed as follows:

$$t_p = t_p g + M t_{carry} + (\frac{N}{M}) t_{bypass} + t_{sum} \tag{2.9}$$

where t_{pg} is the delay to create propagate and generate signals, t_{bypass} is the multiplexer delay and t_{sum} is the time it takes to compute the sum for each block respectively.

2.6 Carry-lookahead Adder

A significant improvement in the implementation of a parallel adder was introduced by a carry-lookahead (CLA) adder developed by Weinberger and Smith in 1958. The CLA adder is theoretically one of the fastest schemes used for the addition of two numbers [2], because the delay to add two N-bit numbers no longer depends on N, but it depends on the logarithm of N which is smaller. Here, the main idea is to calculate all the required carry outputs in parallel based on propagate (P) and generate (G) parameters. In general, we can calculate 'P' and 'G' parameters for each pair of input bits and compute the output

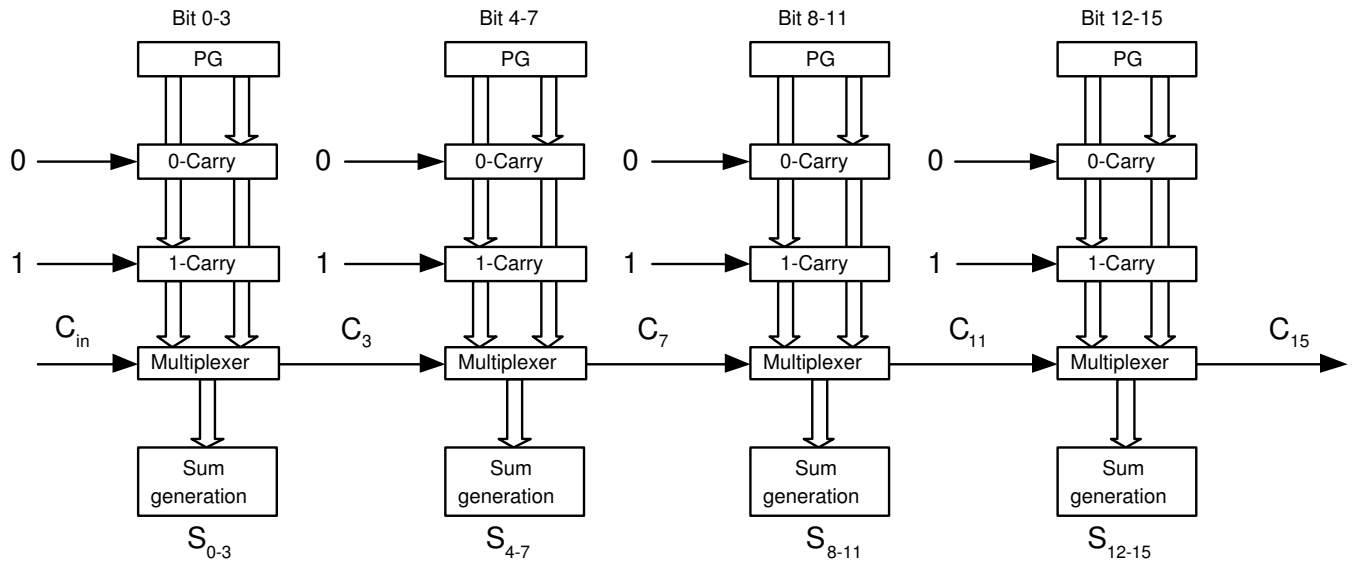


Figure 2.3: 16-bit carry select adder

carries based on the following equation:

$$C_{i+1} = G_i + P_i C_i \tag{2.10}$$

Then we can recursively expand carry formula as follows:

$$C_{i+1} = G_i + P_i(G_{i-1} + P_{i-1}C_{i-1}) \tag{2.11}$$

$$C_{i+1} = G_i + P_i G_{i-1} + P_i P_{i-1}(G_{i-2} + P_{i-1}C_{i-2}) \tag{2.12}$$

As we can see in the above equations, the expanded formula does not depend on intermediate carries and this feature allows us to compute carry for each bit, independently and in parallel. The diagram of a carry lookahead adder is shown in Figure 2.4.

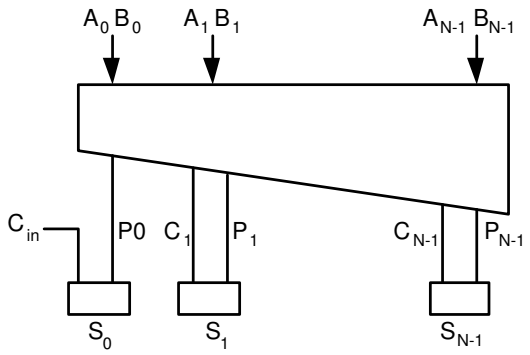


Figure 2.4: Carry-lookahead adder diagram

The main drawback of this type of architecture is that, going to deeper carries requires gates with large fan-in in implementation and results in a slow operation. That is the reason why carry-lookahead adder is the fastest adder just in theory.

2.7 Logarithmic Adder

The class of adders based on solving recurrence equations were first introduced by Bilgory and Gajski and Brent and Kung and were based on the previous work by Kogg and Stone [2, 3]. In order to formulate the operation of these adders, we define an operator like "." which takes two sets of inputs and produces a set of two outputs. Its definition is brought in the following equation:

$$(g, p).(g', p') = (g + pg', pp') \tag{2.13}$$

The "." function is sometimes called "prefix operator". It has two essential properties which allows greater parallelism and hence faster circuits. The first one is the *associative*

property which is shown in the following equation:

$$[(g,p).(g',p')].(g",p") = (g,p).[(g',p').(g",p")] \tag{2.14}$$

The second one is *idempotency* which allows the sub-terms to have overlap:

$$(g,p)_{h...j}.(g,p)_{i...k} = (g,p)_{h...k} \tag{2.15}$$

By taking advantage of these properties, a carry output can be found at a depth proportional to $\log_2(N)$ [4]. As an example, in Figure 2.5, two schemes are shown for computing C_7 (carry output from the seventh input bits). In the first block (a), seven stages should be passed to reach to the output based on the recursive operation, while in the second one (b), only three stages are required to compute C_7 [1]. By combining associative property and recursive formulas, this binary tree (b) is able to give us the least delay.

Because of the associative and idempotency properties of "." operator, the intermediate outputs can be grouped in any order and carry outputs can be computed in a different number of levels. Therefore, we can create different topologies of prefix adders which are found in the literature and are attractive to VLSI designers because of their minimum depth and delay [5, 6, 7]. The main features of these adders are as follows:

1. A good layout
2. The fan-out can be controlled and limited to no more than two
3. Trade-offs between fan-in, fan-out and hierarchical layout topologies can be achieved.

Among different adder architectures based on prefix operations, we will address three of them which are more popular: Brent-Kung, Kogg-Stone and Han-Carlson adders. They are basically implemented with the same idea, the only difference is in the way the intermediate outputs are grouped (fan-in,fan-out) and the way output carries are computed.

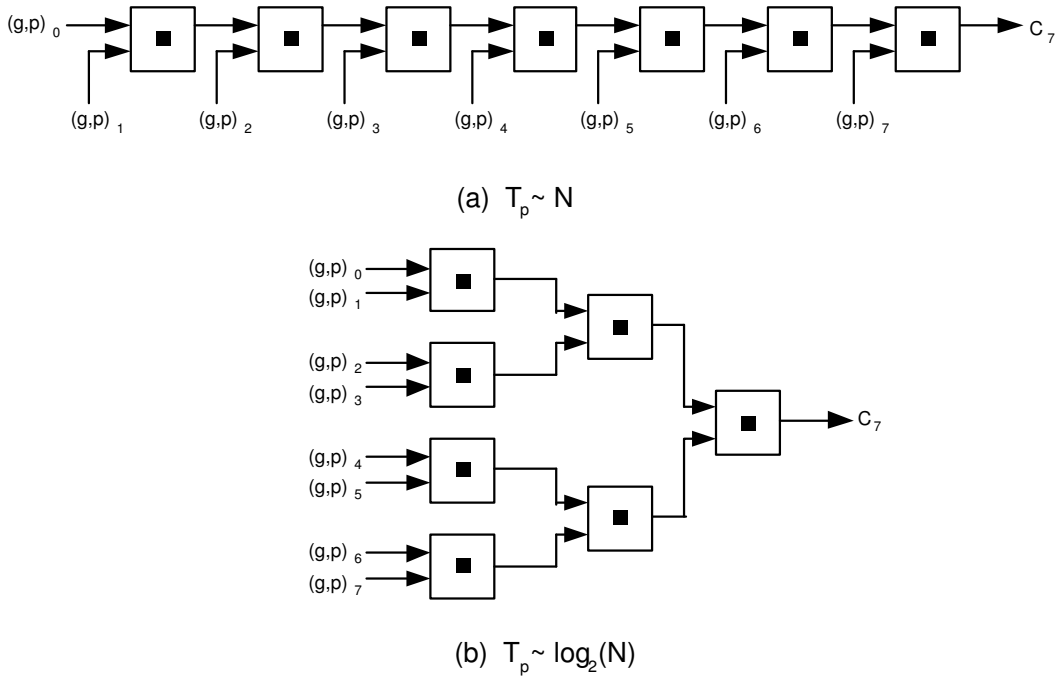


Figure 2.5: (a) Linear and (b) Logarithmic adder

2.7.1 Brent-Kung Adder

The idea of this algorithm [3, 8] is to combine 'P's and 'G's into groups of two, by only using the associative property. The main point is that, Brent-Kung algorithm keeps a fan-out of two, but does not give the minimal logic depth. At the first level, intermediate nodes are combined again based on cell sizes to form the secondary level intermediate nodes and the upper level nodes are not reused for the next levels. The structure of a 8-bit Brent-Kung adder is shown in Figure 2.6 for different cell sizes of 2, 3 and 4. As we can see in the figure, the depth or the number of levels depends on the cell sizes and the number of input bits. The total delay then depends on the depth and each building block. If K is the number of

inputs, then the cost or the number of cells in this adder is $2K$-2-log_2K and the number of levels is $2log_2K$-2 [8].

2.7.2 Kogge-Stone Adder

Kogge-stone adder is similar to Brent-Kung adder in principle [9]. The only difference is that it uses the idempotent property as well. In this architecture, the adjacent bits are grouped based on the cell sizes and they are reused by adjacent nodes. Therefore fan-out equals to the cell size and it has the least number of levels comparing to other structures. If the number of inputs is K, then the total cost or the number of used cells is $Klog_2K - (K - 1)$ and number of levels is log_2K. The structure of a 8-bit Kogge-Stone adder is shown in Figure 2.7 for cell sizes of 2 and 4.

2.7.3 Han-Carlson Adder

Han-Carlson adder is another architecture of prefix adders which is similar to Kogge-Stone, but it has area-time trade-off [10, 8]. In other words, it increases the logic depth for a reduction in fan-out. In this architecture, at the first level, bits are grouped based on the cell sizes and at the second level, the nodes have N number of inputs from previous level results based on the cell sizes (N). If K is the number of inputs, then the cost of this adder is O($\frac{K}{2}log_2K$) and number of level is O (log_2K+1). The configuration of a 16-bit Han-Carlson adder with cell sizes of 2 is shown in Figure 2.8.

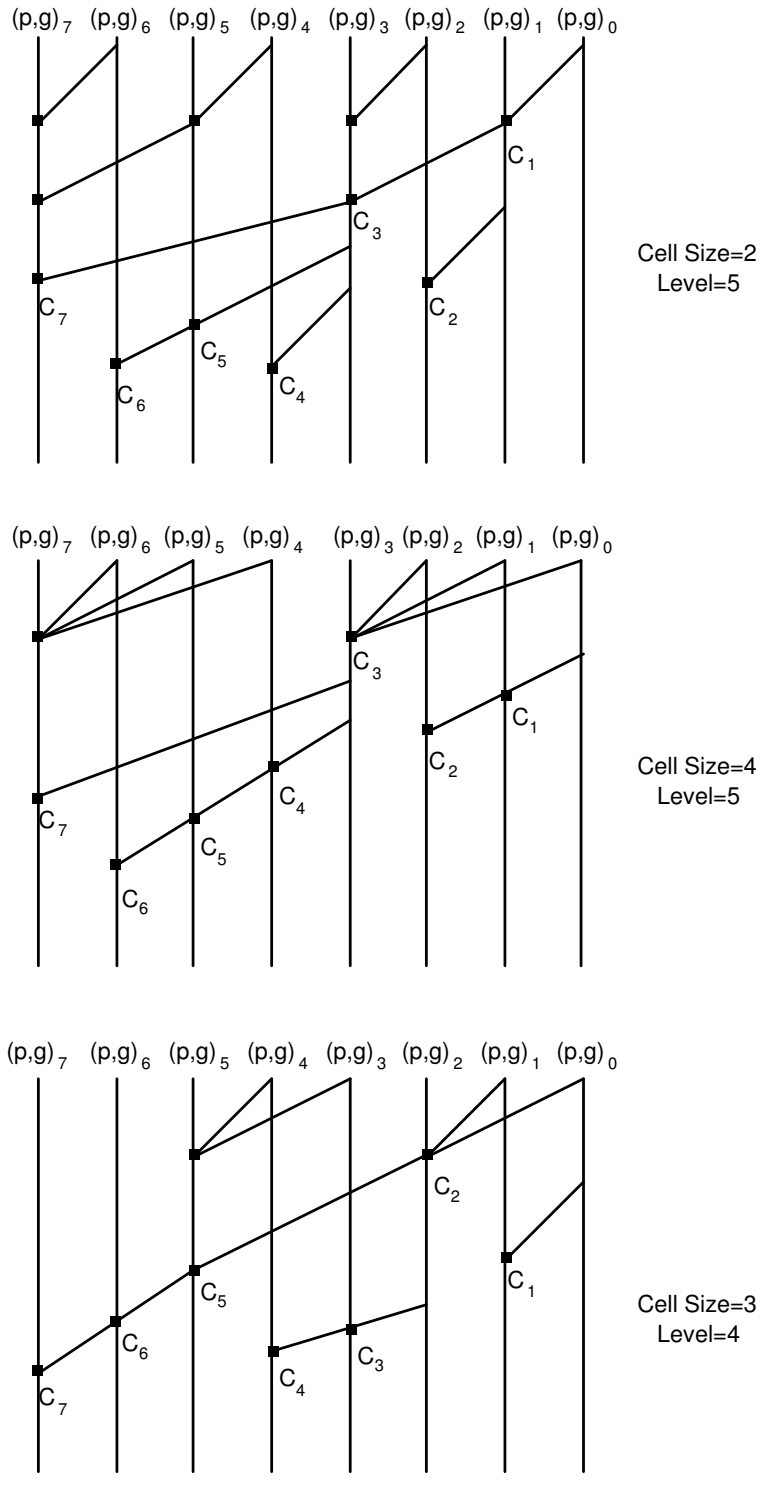


Figure 2.6: 8-bit Brent-Kung adder

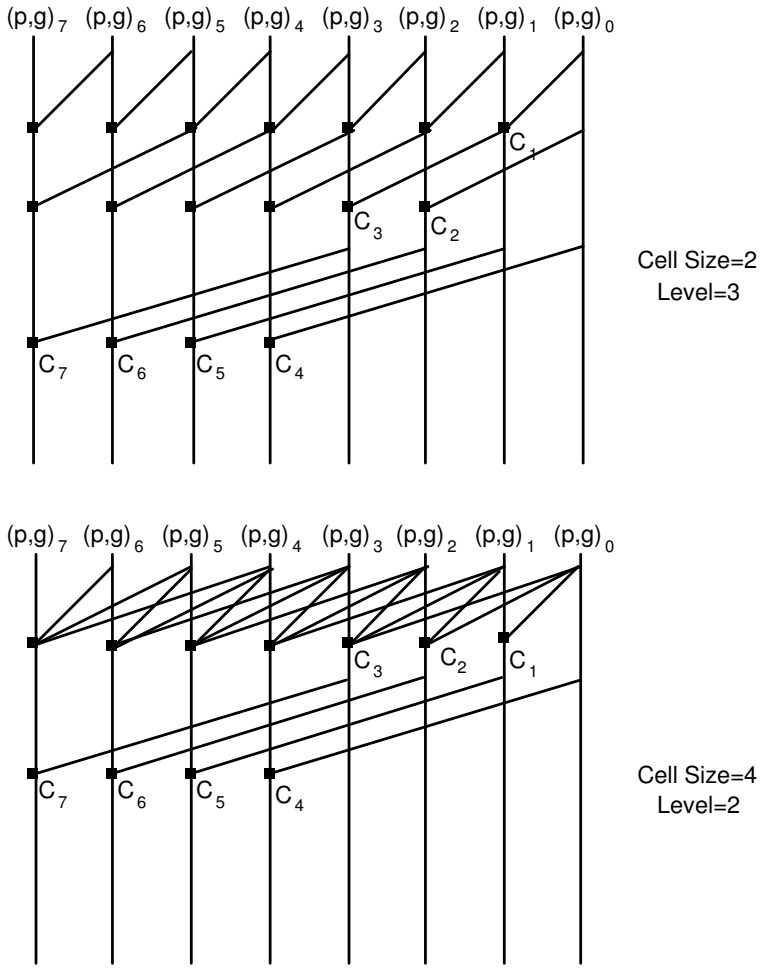


Figure 2.7: 8-bit Kogge-Stone adder

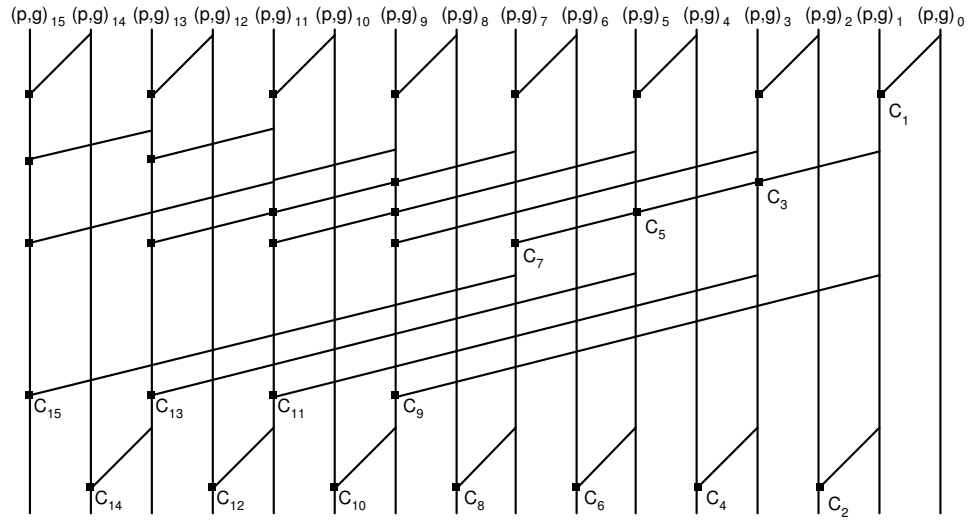


Figure 2.8: 16-bit Han-Carlson adder

2.8 Hybrid Adders

Hybrid adders are obtained by combining elements of different adder architectures like ripple carry adder, carry-lookahead adder, prefix adders, carry skip adders or carry select adders [11, 12]. Different adder topologies can be created by combining different structures and in general, we can achieve high performance, cost effectiveness and low power consumption by using hybrid adders. Figure 2.9 shows an example of hybrid adder in which, ripple carry adder and carry-lookahead structure are combined.

2.9 High Performance 32-bit Adder Design

In this thesis, we have implemented a 32-bit hybrid adder by combination of carry select adder, carry ripple adder and parallel prefix adder structures. In this design, 32 bits are divided into eight groups, consisting of 4-bits. Each of 4-bits are added in a ripple carry

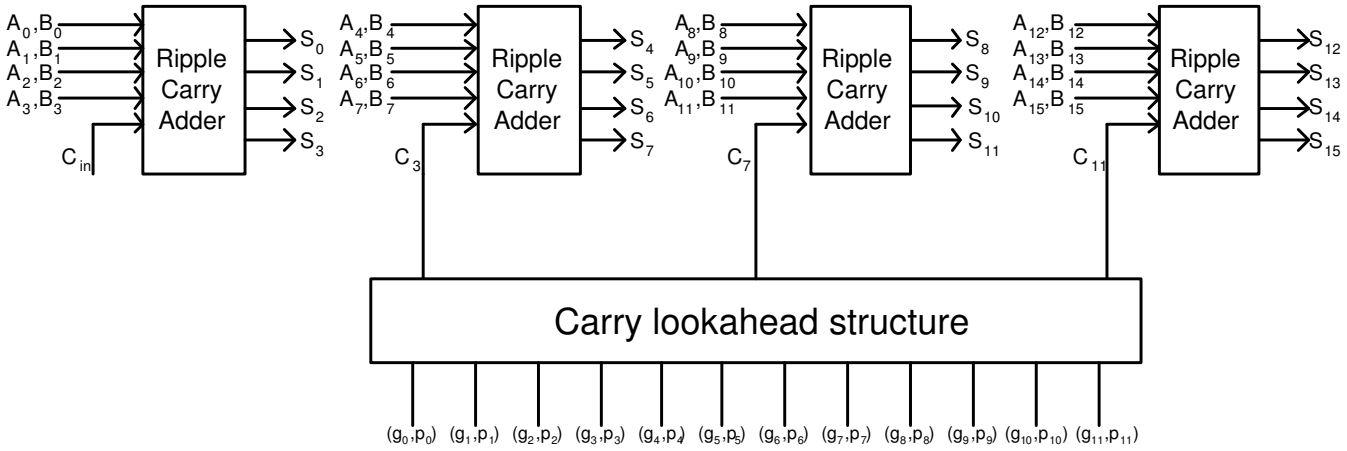


Figure 2.9: An example of hybrid adder

adder block, in which two additions are performed, one is done assuming that carry input is zero and the other is done assuming that carry input is one. In another block, we have a carry merge tree for generating carry outputs: C_3, C_7,C_{11}, C_{15}, C_{19}, C_{23}, C_{27} and C_{31}. This tree is a radix-2 Han-carlson architecture, in which some specific, not all the carry outputs are generated. By radix-2 we mean that the number of stages is equal to $\log_2 N$. At the output of ripple carry adders, there are eight multiplexers which select between the two sets of additions performed in the ripple carry adders based on carry merge tree outputs. This architecture is used to build high performance adders in today's microprocessors [13, 14]. Its structure is shown in Figure 2.10.

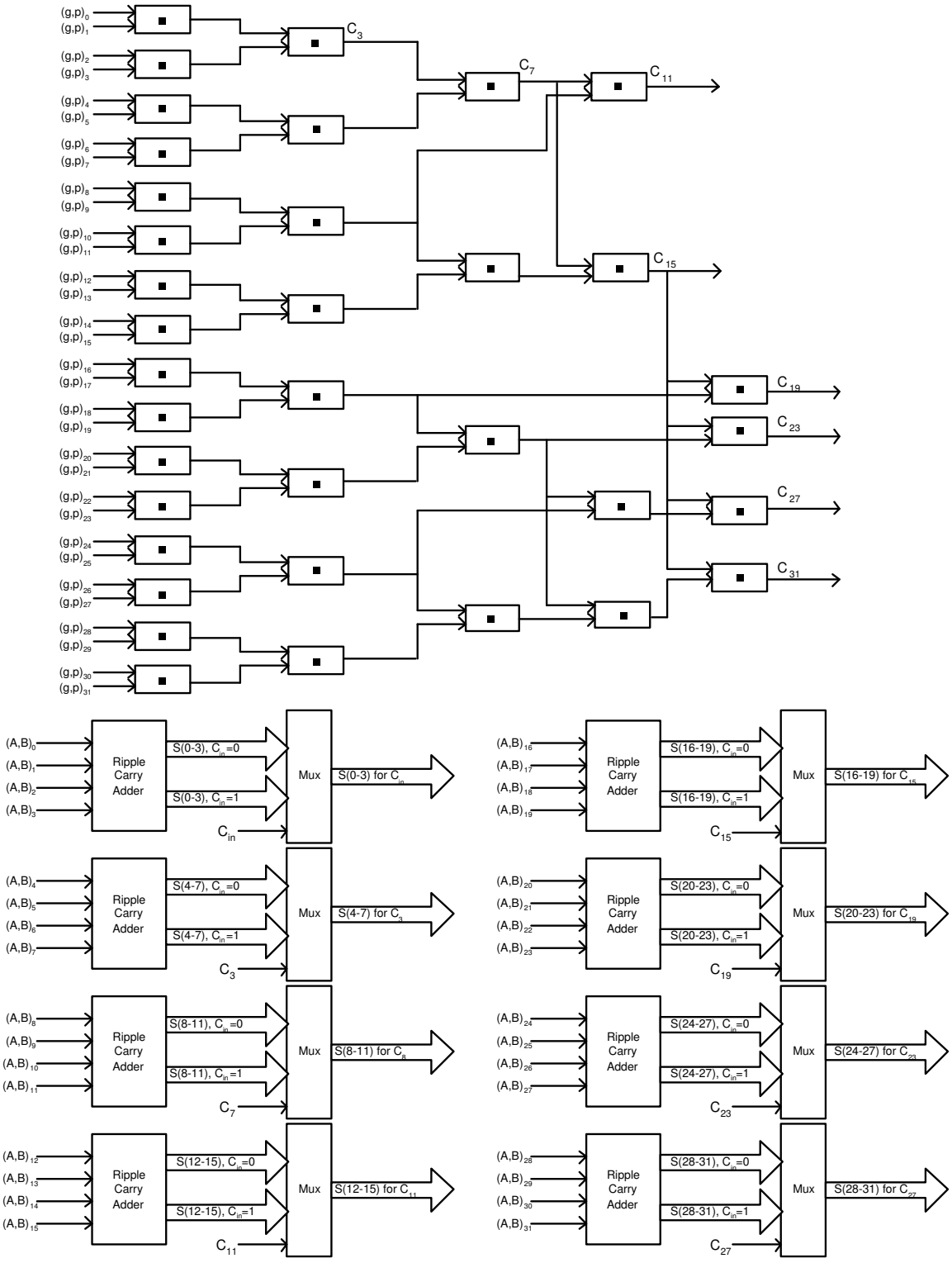


Figure 2.10: High performance 32-bit adder design

Chapter 3

Digital Circuit Styles

3.1 Introduction

CMOS Technology scaling allowed the reduction of MOSFET (Metal Oxide Semiconductor Field Effect Transistor) dimensions from 10μm in the 1970's to a present day of 0.10μm. High speed, low power and high density has made CMOS technology the dominant one in today's integrated circuits and the further scaling down of MOSFET devices shows that it is a promising technology for future IC generations as well. There are different circuit styles implemented with MOSFET devices for constructing logic gates used in today's microprocessor units like adders. However, there is a tradeoff between speed, power , area and robustness between all of them. So, for each design, depending on its' application, the best balance for these three items has to be found. In this section, an overview of the CMOS logic styles and their characteristics is given and different styles are compared in terms of speed, power dissipation, robustness and silicon area. Finally, the selected logic styles for implementing different parts of the 32-bit adder are addressed. Since the objective is to investigate the trade-offs that are possible at the circuit level in order to

reduce power dissipation while maintaining the overall system throughput, we must first study the parameters that affect the power dissipation and the speed of a circuit.

3.2 Power and Delay in CMOS Circuits

Ideally, CMOS circuits dissipate no static (DC) power, since in the steady state there is no direct path from V_{DD} to ground. However, this situation never happens in practice because MOS transistor is not a perfect switch. There is always leakage currents and substrate injection currents which leads to static power dissipation in CMOS circuits [15, 16]. One of the dynamic components of power dissipation arises from the transient switching behavior of the CMOS devices. At some point during the switching transient, both the NMOS and PMOS devices are on and a short circuit current exists between V_{DD} and ground. Another component of dynamic power dissipation is charging and discharging of parasitic capacitances which consume most of the power used in CMOS circuits. This leads to the conclusion that CMOS power consumption depends on the switching activity of the signals involved. If we show the switching activity by a parameter α, then we can compute the whole power dissipation through the following equation:

$$P = \alpha C_L V_{DD}^2 f + (I_{sc} + I_{Leakage}) V_{DD} \tag{3.1}$$

Where, f is the frequency of logic operation, C_L is the total capacitance charged and discharged every cycle and V_{DD} is the power supply voltage. I_{sc} and $I_{leakage}$ are the short circuit current and leakage current respectively. As we see in the above formula, power supply voltage has a quadratic relationship with the power; therefore voltage reduction offers the most dramatic means of minimizing energy consumption. This issue will be widely discussed in the following chapters.

Even though the exact analysis of circuit delay is quite complex, a simple first order derivation can be used [17, 18] in order to show its dependency on the circuit parameters:

$$T_d \propto \frac{C_L V_{DD}}{K(V_{DD} - V_{TH})^{\alpha}} \tag{3.2}$$

Where K depends on the transistors' aspect ratio $\frac{W}{L}$ and the other device parameters, V_{TH} is the transistor threshold voltage and α is the velocity saturation index which varies between 1 and 2. Since a quadratic improvement in power dissipation may be obtained by lowering the supply voltage, many people have investigated the effects of lowering the supply voltage in VLSI gates. However, as we see in the above formula, reducing the supply voltage would increase the delay and this effect would be more dramatic when the voltage is close to the threshold voltage.

3.3 Clocked and Nonclocked Logic Styles

In general we can divide the logic circuit styles to clocked and nonclocked CMOS circuit styles [19, 2]. As it is apparent from their names, in clocked style we need a clock signal to synchronize the whole system while in the nonclocked structure, there is no need for such a signal. In the following section, we will introduce the most popular circuits in both families.

3.3.1 Nonclocked Logic Styles

Nonclocked logic circuits are basically the ones that are static and their outputs are always valid independent on any extra signal like clock. The most popular nonclocked circuit styles are static combinational CMOS, pseudo-NMOS logic and pass transistor logic which will be introduced here [20, 21, 19].

Static Combinational CMOS:

Complementary static CMOS is basically one pull up path to V_{DD} which is built by PMOS transistors and one pull down path to ground through NMOS transistors which is the complement of the pull up path. At all times, either the pull up or pull down switches are on, so the output is always driven by power supply or ground. That is the reason why they have a large amount of noise immunity and they are forgiving to defects and process variations. The structure of a NAND gate implemented in static CMOS is shown in Figure 3.1. In this structure, PMOS/NMOS device ratio defines the switching point of the logic. Since the PMOS device mobility is lower than NMOS device, in order to have equal driving capability for both pathes, PMOS sizes should be bigger than NMOS sizes. Therefore static gates are large in area and add a large load to the preceding gates due to their large capacitance at the inputs. In addition to that, while the inputs to a static gate transits from high to low or vise versa, both pull up and pull down pathes are on for a short interval and we have short circuit current which consumes almost 15% of the total chip power implemented with these gates.

Pseudo-NMOS Logic:

Pseudo-NMOS logic style has a similar NMOS pull-down network as combinational CMOS style. The configuration of a NAND gate implemented in this style is shown if Figure 3.2. The only difference is that the PMOS pull-up network is replaced by a PMOS transistor whose gate is connected to ground and causes static power dissipation when the output is low. This logic is ratioed; meaning that the output transition and the delay depends on the ratio of NMOS and PMOS transistors and this feature makes it vulnerable to process variation. Its advantage is the fact that the transistor count is less and it can be used in

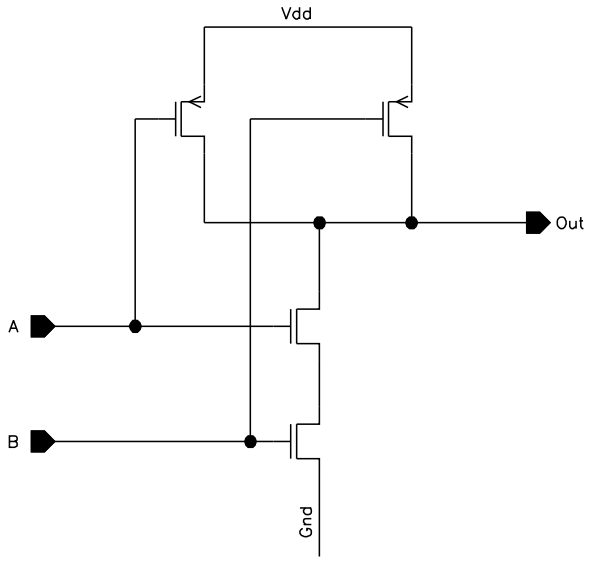


Figure 3.1: Static NAND gate

wide fan-in gates.

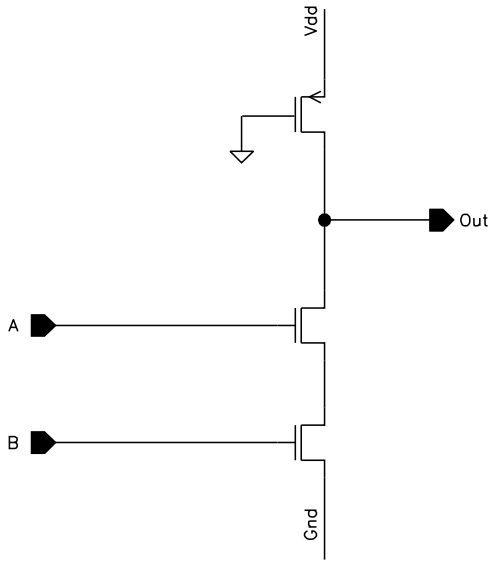


Figure 3.2: Pseudo-NMOS NAND gate

Pass Transistor and Transmission Gate Logic:

This logic style uses MOS transistor as a simple switch. Basically the boolean function is implemented with NMOS transistors only or transmission gates. In CPL (complementary pass-transistor logic) gates, the function is just implemented with NMOS transistors, like the XOR gate shown in Figure 3.3.

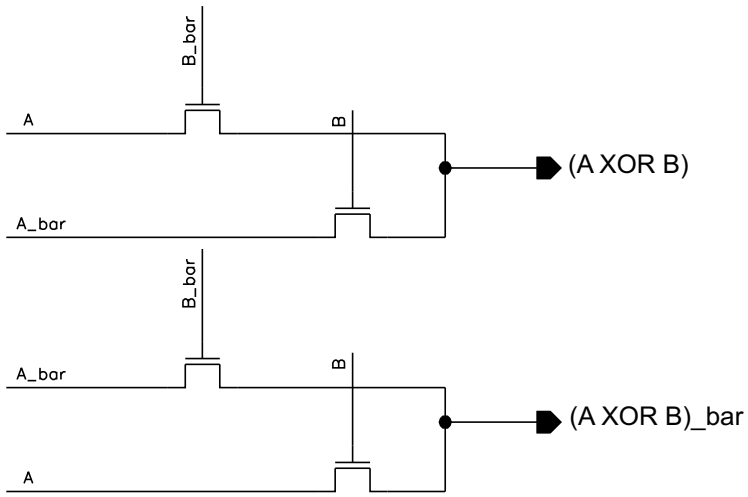


Figure 3.3: CPL XOR gate

CPL has the advantage of efficient implementation of complex functions. In general, pass transistor logic network has fewer transistors than combinational CMOS logic; therefore they have less parasitic capacitances and consume less power and are sometimes even faster than complementary static gates. Pass transistor gates are specially attractive in making XOR gates in adder structures, since they utilize almost half of the transistor numbers used in combinational gates. This logic style is easy to synthesize starting from the boolean expressions using binary decision diagram (BDD). However, they have some disadvantages; they are more sensitive to voltage scaling compared to combinational static logic and their delay increase dramatically when the supply voltage is reduced. In addition

to that, since only NMOS transistors are used in CPL gates, the voltage swing at the end of a pass transistor network has a swing from 0 to $V_{DD} - V_{TH}$; therefore PMOS transistors at the following static gates are not completely off resulting in static power dissipation. This problem is usually solved by a PMOS level restorer transistor to pull up the node to V_{DD}. The level restorer adds hysteresis to the gate and degrade the performance. Also, the delay of pass transistor networks increases quadratically with the number of stages and as a result, some intermediate buffers should be used to make strong V_{DD} and ground. All these problems arise from the fact that NMOS transistors can not pass V_{DD} faithfully to the other side. The solution is using a complementary PMOS transistor in parallel with NMOS to generate a strong V_{DD} at the output. This structure is a transmission gate which behaves almost as a fixed resistance during switching time and has more noise immunity because of its stronger driving capability. The structure of a XOR gate implemented with transmission gates is shown in Figure 3.4.

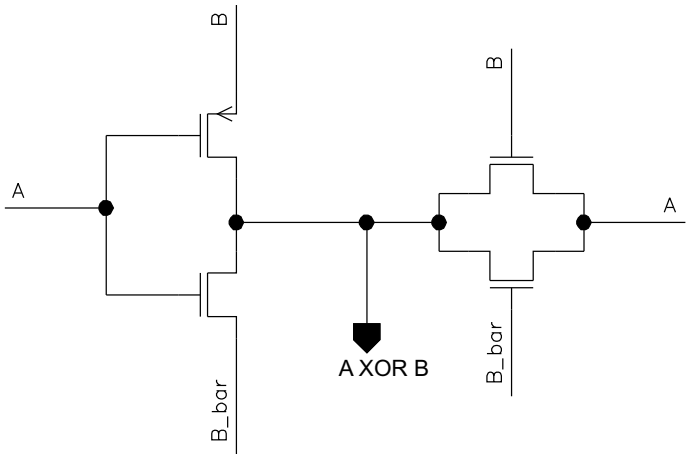


Figure 3.4: XOR implementation with CMOS transmission gate

3.3.2 Clocked Logic Styles

Clocked logic families are those in which the circuit does the logical function during one phase of the clock period. In the previous section, some of the popular static (nonclocked logic) styles were explained. As it was mentioned there, in static CMOS, circuit operation must be realized using both NMOS and PMOS transistors which reduces the performance by adding more parasitic capacitances at the output and more capacitive load at the input. In addition to that, because PMOS devices' mobility is almost half of NMOS, they should be generally two times of NMOS devices to have a balanced transition and that makes those effects even worse. On the other hand, in the clocked logic styles, the logic evaluation happens in one direction; therefore only a single device polarity is used in the evaluation path and as a result, the parasitic capacitances are dramatically smaller. The most popular clocked logic styles are Domino CMOS, Multiple output Domino (MODL), Compound domino logic, Noise tolerant precharge logic (NTP), clock-delayed domino. Each will be explained briefly [19].

Domino CMOS:

The single-ended domino structure is common in high speed logic design due to its simplicity and fast operation. The structure of a NAND logic is shown in Figure 3.5. As it is common between all clocked gates, there are two phases of operation: precharge and evaluation. At precharge time (CLK=0), the output of dynamic gates are pulled high to V_{DD} and during evaluation time (CLK=1) the output may be discharged according to the inputs or it may stay at high voltage. Therefore, the evaluation phase is always in one direction at which dynamic outputs go from high to low. This feature will help us to design high performance gates as will be seen later. Since the inputs to a domino circuit from a prior stage and cycle may corrupt the next cycle's precharge before new valid inputs can

be delivered, one inversion should be done after each dynamic gate output and before the next dynamic stage. Dynamic gates can contain a substantial amount of logical width in the NOR direction and depth in the NAND direction. This allows significant logic gain along a path of dominos. There are some main advantages and disadvantages for this logic style and in general for clocked gates.

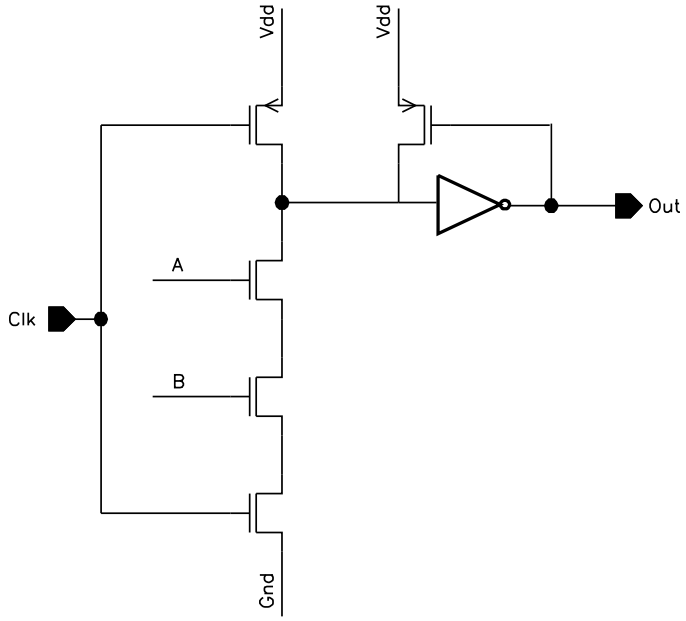


Figure 3.5: NAND implementation with domino gate

Advantages:

1. The current of logic evaluation device is devoted to discharge the output capacitors rather than sinking the short circuit current through PMOS transistor.

2. The switch point of the circuit is no longer $\frac{V_{DD}}{2}$ like static gates, but it is approximately the threshold voltage of NMOS transistors in the evaluation path that causes a faster switching time.

3. Since PMOS transistors are eliminated, the output capacitance is smaller leading to higher speed during switching time.

4. Dynamic logic typically takes smaller area than the equivalent static circuits.

Disadvantages:

1. Logic clocking can substantially increase power consumption and may consume 20% of the total chip power.

2. Dynamic nodes are kept floating at the evaluation time and this makes them vulnerable to noise, leakage and failure mechanisms. This problem is almost alleviated by using PMOS keeper transistors. The keepers along with the output inverters form a half latch and cause hysteresis adding delay to the circuit.

3. Switching activity is higher than the equivalent static logic, because from cycle to cycle, even if the output state of a given domino circuit does not go to high, it will be precharged in the next precharge cycle and will be discharged during evaluation cycle again. In contrast, static circuits will not switch if the inputs do not change.

Multiple Output Domino Logic:

Multiple output domino logic (MODL) uses the intermediate nodes at the evaluate tree of the domino gates as the outputs, if the logic is implemented in such a way that the intermediate results are subset of the greater function accomplished by the entire circuit. The structure of a MODL circuit is shown in Figure 3.6. By using this logic style, we can save area and reduce delay by doing more than one logic operation in the dynamic evaluate tree. However, there is a performance trade-off with precharging the intermediate nodes and increasing the device sizes at the bottom of the tree to sink more current.

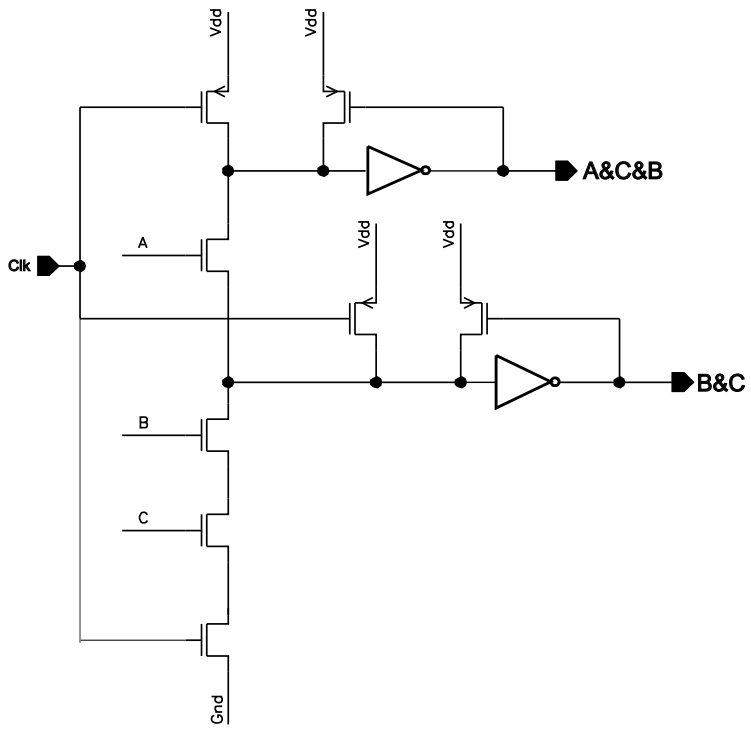


Figure 3.6: An example of MODL gate

Compound Domino Logic:

The structure of a compound domino logic (CDL) is shown in Figure 3.7. The idea of compound domino is using complex static gates like NAND and NOR gates at the output of domino gates rather than inserting a simple inverter. In this way, we can have inversion at the output to guarantee the correct functionality and at the same time, do some logical operation along with inversion to avoid having extra gates. In addition to that, dynamic logic can be formed by parallel evaluate trees resulting in lower fan-in gates. Attention must be paid to the delay of static output gates so as not to compromise the gain realized by domino action. This logic style can enhance performance in some critical path structures of arithmetic units like adders by combining dynamic and static gates and taking advantage of their benefits together.

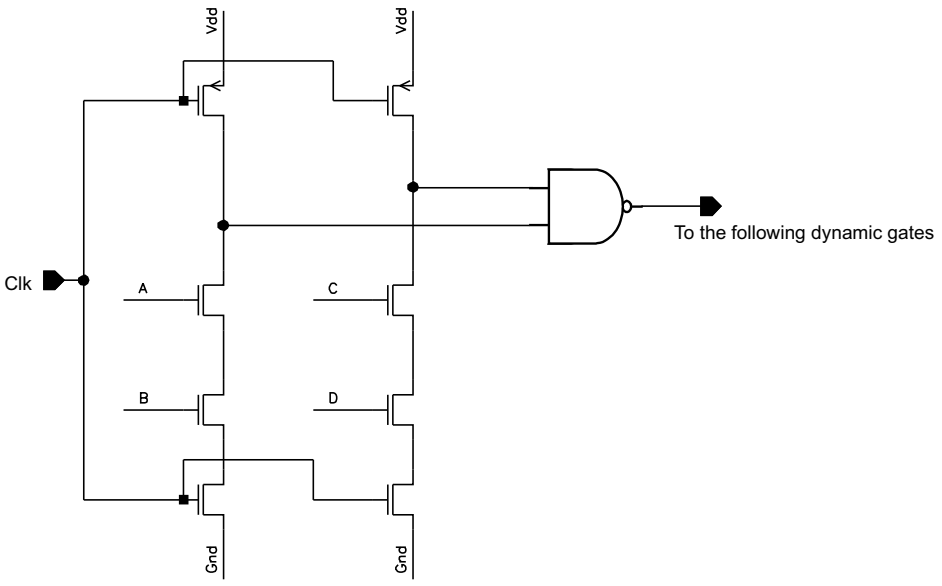


Figure 3.7: An example of CDL gate

Noise Tolerant Precharge Logic:

In this logic style, the complement of evaluation path implemented with PMOS transistors is added to the domino gates. This PMOS structure never allows a precharged node to float, resembling the static equivalent of that function and is on, providing a path to V_{DD} when the evaluate path is off. The PMOS transistor network is not large enough to precharge the dynamic output but can provide a reasonable noise immunity to the logic. The structure of a NTP gate is shown in Figure 3.8. By using this architecture, we can avoid having keepers and thus hysteresis at the output which is compromised by the performance, but on the other hand, the trade off is that PMOS structure adds parasitic capacitors at the output, increases area and adds more fan-out load to the previous stages.

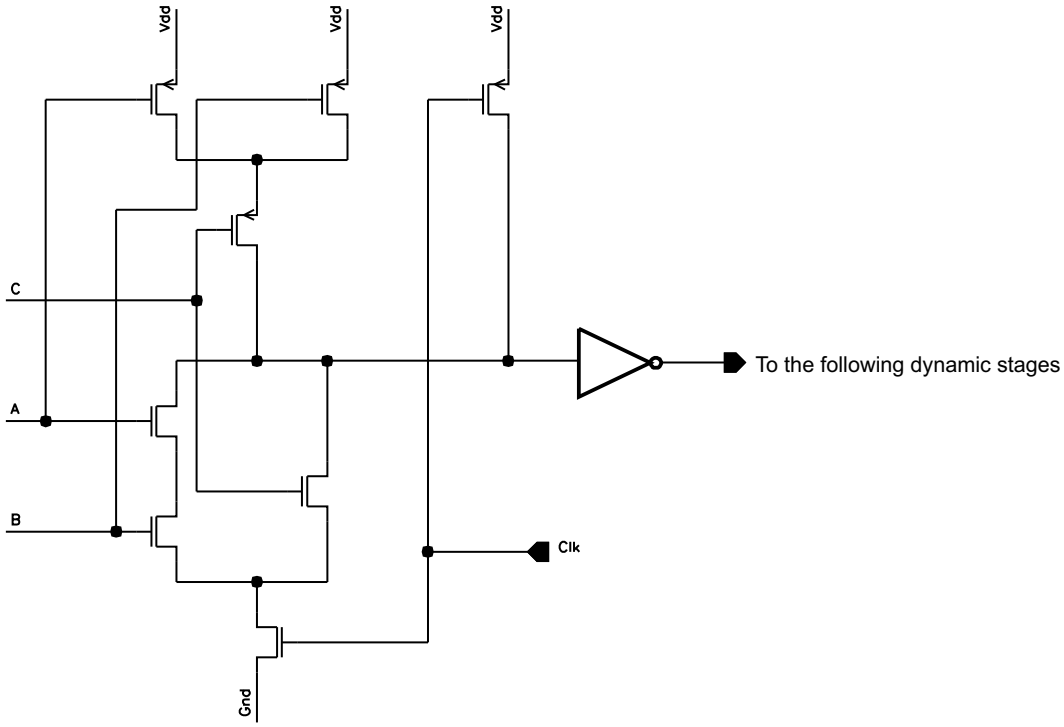


Figure 3.8: An example of NTP logic

Clocked-delayed domino:

As it was explained before, we must ensure that dynamic inputs never make a one to zero transition while in evaluation. One solution to this problem was to use inverter at the output of dynamic gates. Another solution could be using a delayed version of clock for the following gates. By using this technique, clock-delayed (CD) domino was developed to provide gates with either inverting or non-inverting outputs and with the high speed and layout compactness of dynamic logic. The delay of clock signal should be tuned such that the clock always arrives after the interval in which evaluation of the gates have been done and some design margins must be added to the delay element to guarantee the correct functionality across process, circuit and application variations. The main problem of this technique is the sensitivity to skew, process variation and noise.

3.4 Circuit Styles used in the Designed High Performance Adder

The upper level architecture of the 32-bit Han-Carlson adder designed in this thesis was introduced in the previous chapter. As it was explained there, it consists of a carry-merge tree for producing the carry outputs, a ripple-carry adder structure for 4-bit addition and some multiplexers at the output. In this section, we will describe the circuit styles that are used for each part and the reason for choosing them.

3.4.1 Carry-Merge Tree

Carry-merge tree is composed of five stages of cells which do the "." operation defined in chapter 2. Outputs of the 4-bit ripple carry adders are ready well before the carry

outputs of this tree are computed; therefor, the adder performance totally depends on the speed of the carry-merge tree structure. As a result, in this part of the adder architecture, the main focus is to achieve high performance. Among all the logic styles introduced in the last sections, clocked ones offer higher speed and between them, single domino is the simplest one which consumes less power and is more tolerant to noise and process sensitivity (compared to CD logic). It is also found in the literature that domino still remains the circuit style of choice for high-performance ALU design, since it provides a favorable trade-off between performance and noise margin where significant performance gain is achieved at a reduced but adequate noise immunity in comparison with static logic [22]. However, we need some inversions at the outputs of dynamic gates to guarantee its correct functionality, so it is more reasonable to use compound domino logic (CDL) in this design and use conventional CMOS logic gates at the output of domino gates instead of explicit inverters to combine inversion and functionality together at these static gates and get fewer number of stages. Therefore, odd stages are implemented in dynamic style and even stages in static, resulting in the formation of a chain of alternate dynamic and static gates. This arrangement enhances circuit performance without compromising robustness and is used mainly in today's high performance ALUs [23, 13].

The keeper transistor along with the required inverter are positioned at the dynamic gate outputs and off the critical path. Since the dynamic evaluation path consists of one or two branches and transistors are in the stack, the amount of leakage is not so large to have strong keepers that limit the performance. Therefore, keeper contention is not a big issue in this design. By looking at the high level architecture of the carry merge tree, we realize that the critical path in this design consists of the first stage (PG block), the second stage cell, the third stage cell which generates C_3 , the fourth stage which produces C_7 and the fifth stage that generates C_{15}, because of its highest fan-out, and finally each of the sixth

stages that produce C_{19}, C_{23}, C_{27} and C_{31}. This path is the longest one in carry merge tree and defines the worst case delay in the adder.

If we implement the first stage, PG block, with domino CMOS, then the outputs of the second stage will be low during the precharge time and are domino compatible signals. Therefore, we can remove the clock footer transistors in the following stages. This task reduces the number of transistors in the stack which results in performance enhancement and also reduces clock network power [19]. In Figure 3.9, the implementations of "." operation in static logic (used in the second stage) and dynamic logic (used in the third stage) without footer transistors are sketched. The second and third stage cells are repeated through the whole carry merge tree alternatively, but with different transistor sizes. Transistor sizing is explained in the next chapter.

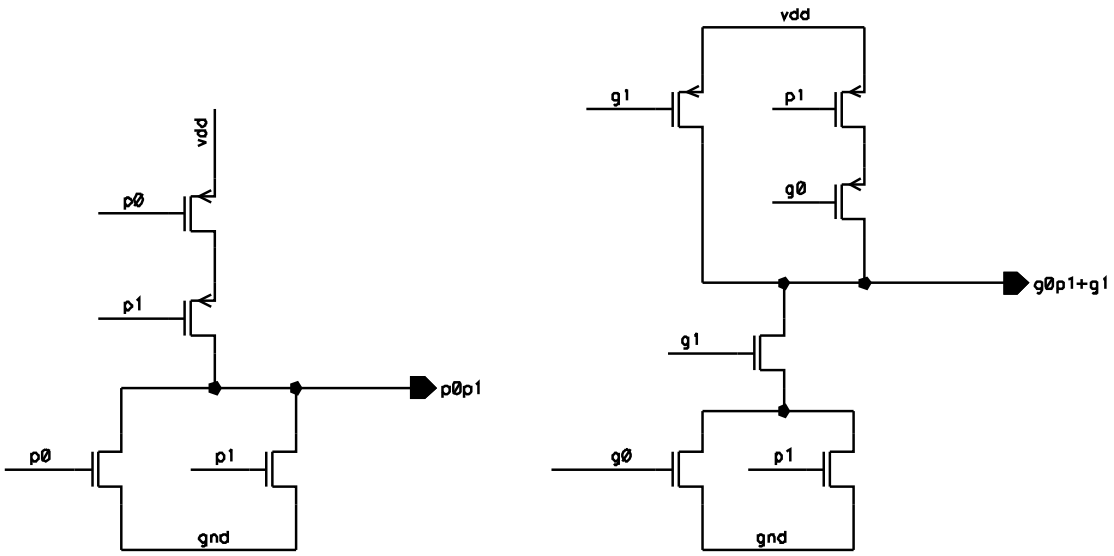
3.4.2 Ripple Carry Adder

In ripple carry adder block, we need to add four bit slices two times, once assuming the input carry is zero and the other time assuming the input carry is one. The number of stages in this block is fewer than the carry-merge tree and therefore, it is not on the critical path and can be implemented with static logic to save power. As it was seen before, one of the most important building blocks in adder operation is XOR gate and it can be efficiently implemented with 'transmission gate logic'. The sum outputs are generated out of XOR gates and the carry outputs are computed in conventional static gates according to the following formula:

$$Sum = A \oplus B \oplus C_{in} \tag{3.3}$$

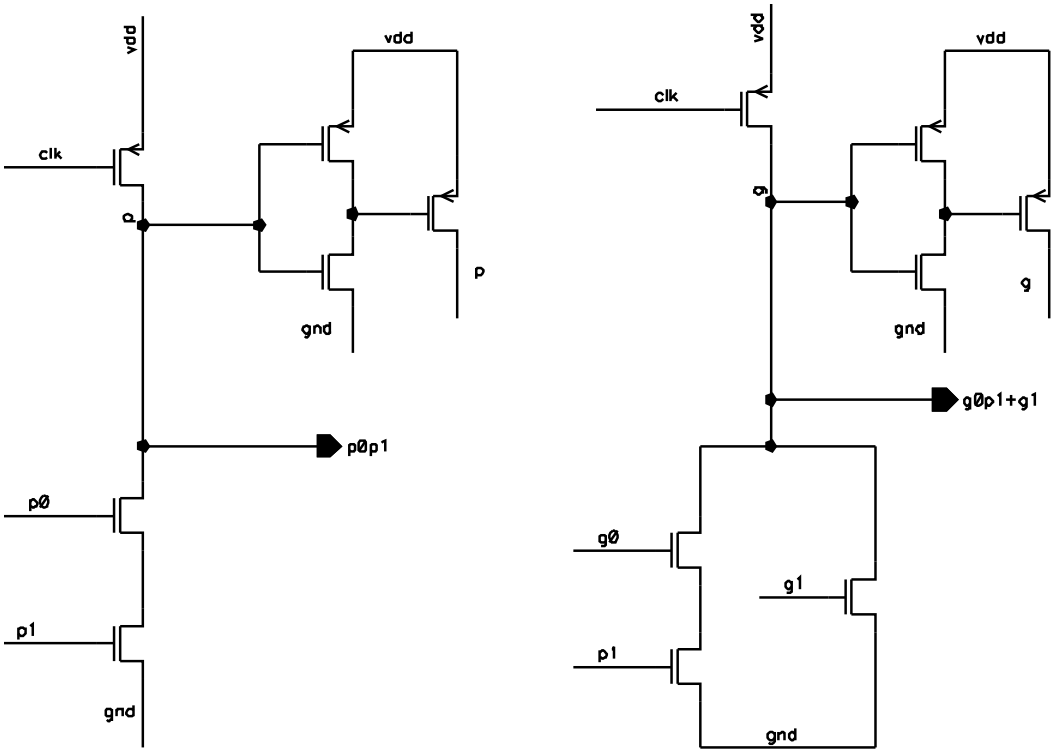
$$C_{out} = AB + (A + B)C_{in} \tag{3.4}$$

The circuit architecture of 1-bit block of the ripple carry adder is sketched in Figure 3.10.



(A) P block in static gate

(B) G block in static gate



(C) P block in dynamic gate

(D) G block in dynamic gate

Figure 3.9: Carry merge tree basic cells

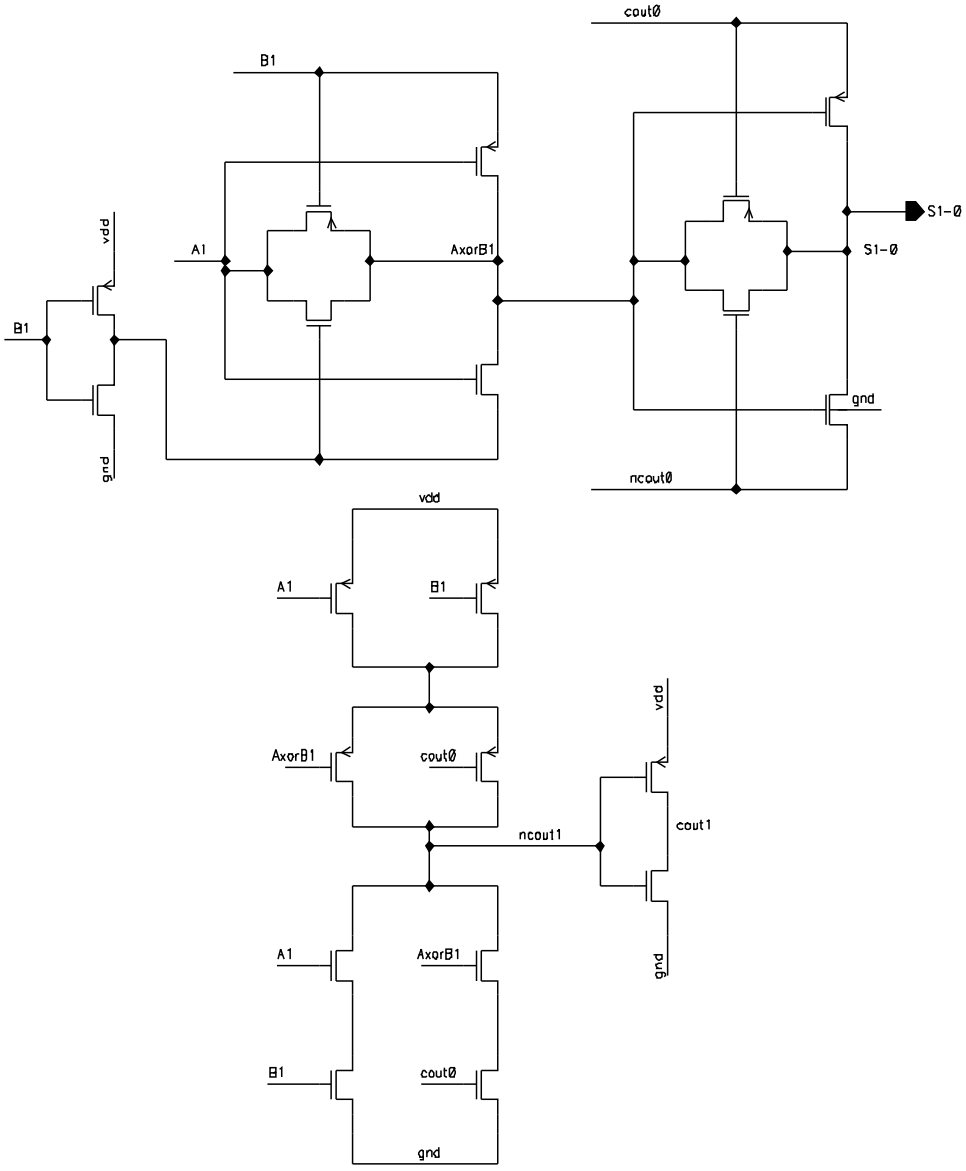


Figure 3.10: One block of the 4-bit ripple carry adder circuit

One conventional approach in carry-select adder architectures is to have two similar blocks for ripple carry adders, in one of them the input carry is zero and the outputs are: $S_0 - 0$, $S_1 - 0$, $S_2 - 0$, $S_3 - 0$, and for the other one the input carry is one and outputs

are :$S_0 - 1$, $S_1 - 1$, $S_2 - 1$, $S_3 - 1$. By taking a quick look at sum and carry equations, we will find that the sum output for two bits are complement if the carry inputs are different as we can see in table 3.1.

	C_{in}=0	C_{in}=1
A B	Sum C_{out}	Sum C_{out}
0 0	0 0	1 0
0 1	1 0	0 1
1 0	1 0	0 1
1 1	0 1	1 1

Table 3.1: Ripple carry outputs

Therefore, we can conclude that the output bits computed by ripple carry adder with C_{in}=1 are complement of outputs computed with C_{in}=0 if the carry inputs propagated to those bits are different, otherwise they are the same. For the first bit, S_0, we can say that $S_0 - 1 = \overline{S_0 - 0}$, because the input carry to the first bits are different, so in this case we do not need to compute $S_0 - 1$ as it is simply obtained by inverting $S_0 - 0$. We can expand this relation for other bits as well and omit the other parallel ripple carry adder to save area and power. This kind of carry select adders have been proposed in the literature before [24, 25], but we have implemented it in another fashion based on the following observations: according to the above table, carry outputs for C_{in}=0 and C_{in}=1 for each bit pair (A,B) are the same when A XOR B=0 and are complement when A XOR B=1. We already have ($A_{0,1,2}$ XOR $B_{0,1,2}$) results in the 4-bit adder with $C_{in} = 0$ and these signals can decide if $S_{1,2,3} - 1$ are equal to or complement of $S_{1,2,3} - 0$ and this decision can simply be implemented by XOR functions. The structure of this reduced area ripple carry adder is shown in the Figure 3.11. By using this structure, instead of using 62 transistors

to implement another ripple carry adder in parallel with the first one, we can implement three XOR gates and one inverter with 26 transistors.

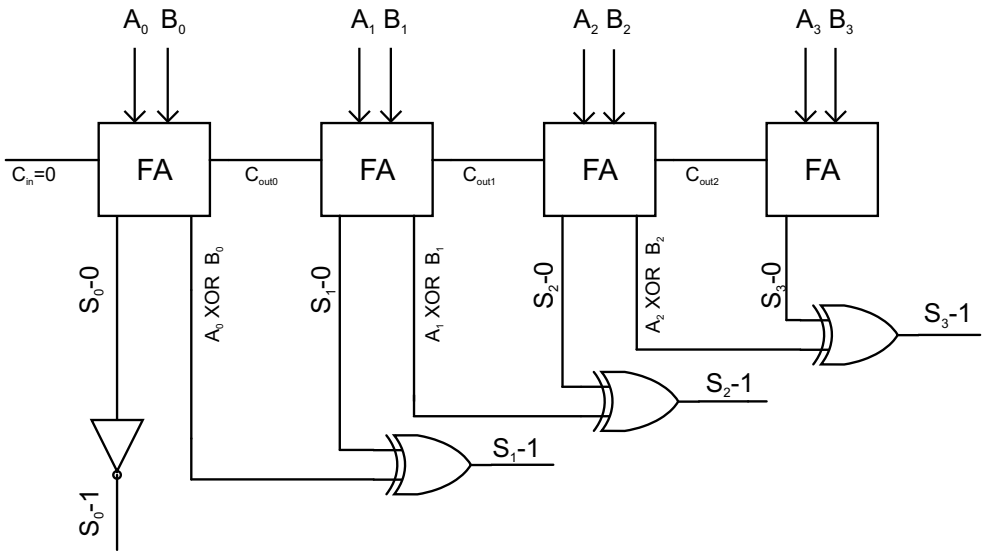


Figure 3.11: Carry select adder implemented with one ripple carry adder and XOR gates

3.4.3 Output Multiplexer and Latches

As it was sketched in the adder architecture, the outputs of the ripple carry adders go to the multiplexer inputs and carry outputs of the carry merge tree go to the multiplexer select inputs. Since the multiplexers are in the critical path, they are implemented with domino gates to be as fast as possible [13, 14]. The output latch has a simple static structure to save power and is also compatible with dual supply design (proposed in the next chapters), since putting the clock signal on a lower voltage than the main power supply does not cause static power dissipation. The structure of multiplexer and output latches are shown in the Figures 3.12 and 3.13 respectively.

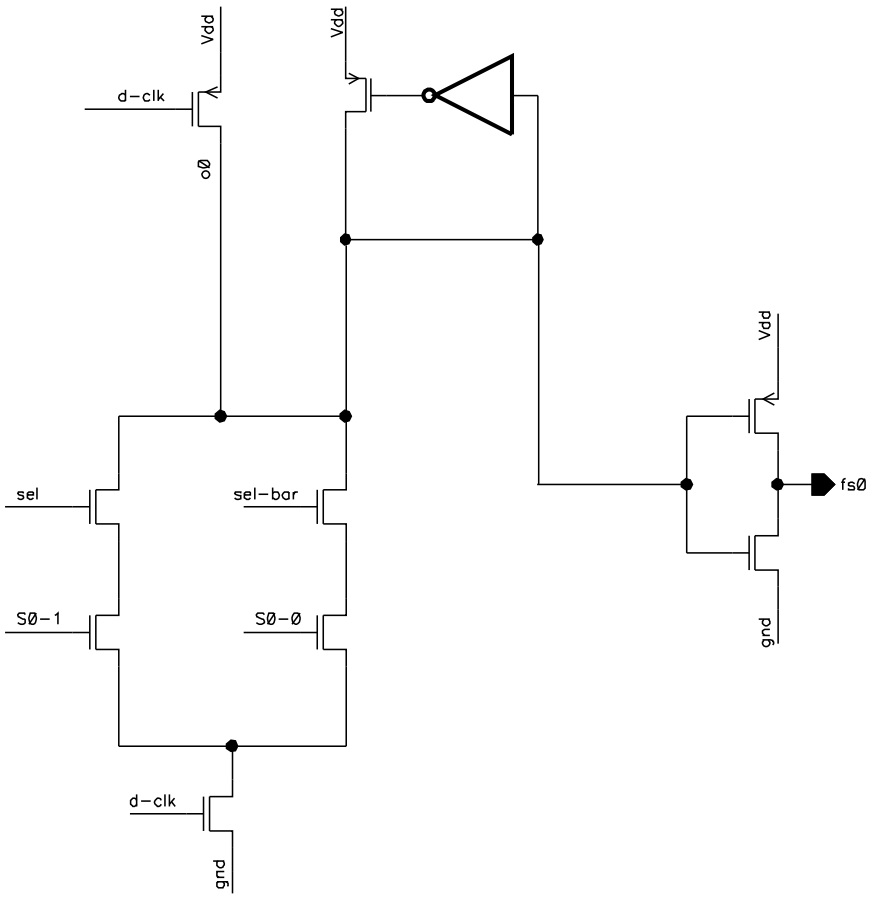


Figure 3.12: Output multiplexer circuit

According to the ALU system design, the outputs of the adder should be finally ready after passing positive and negative phases of the clock. In this design, the negative edge boundary starts from the multiplexers. Therefore, the multiplexer and its following latches are evaluating at the negative edge of the clock. The $\overline{clk}$ signal is asserted one inverter delay sooner than clk in order to speed up the critical path operation. The output latches should be transparent in a short period of time to save the adder results and after that it should become opaque, otherwise the latch data will be distorted when the multiplexer

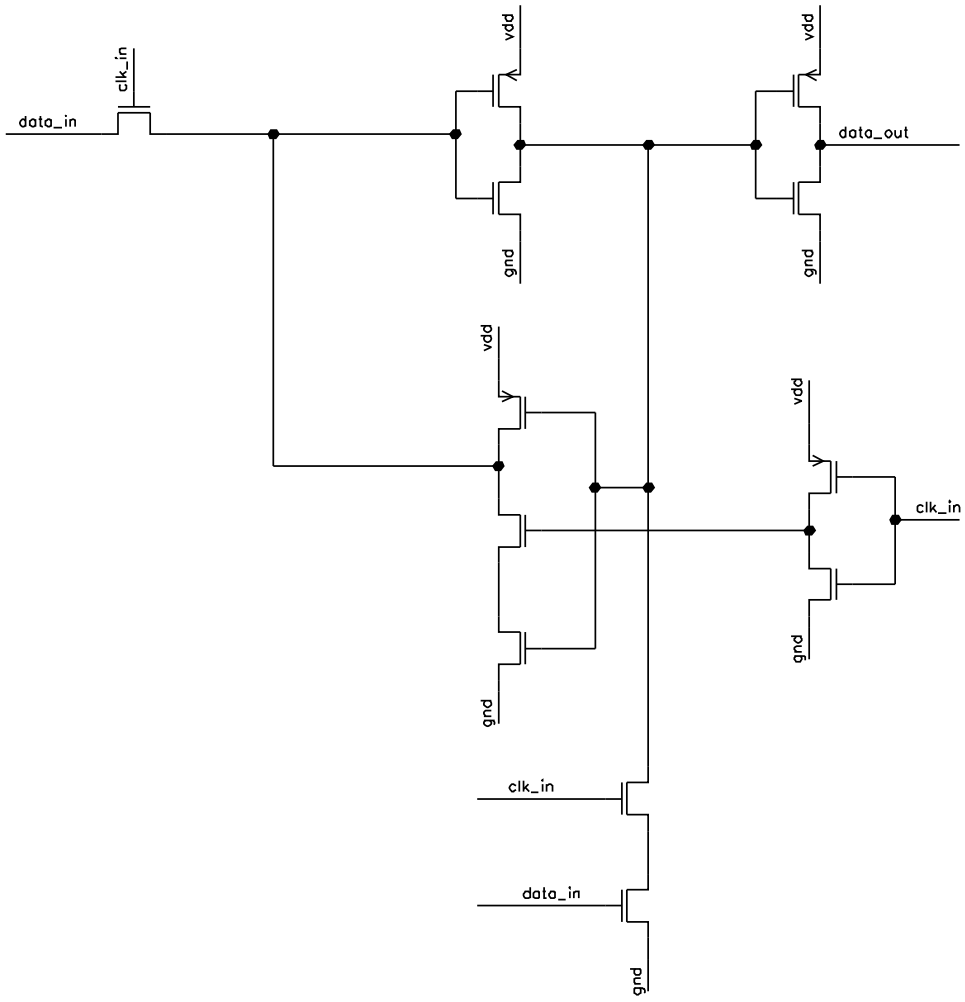


Figure 3.13: Output latch circuit

output is precharged.

Chapter 4

Adder Transistor Sizing

4.1 Introduction

When a circuit is implemented, transistor sizes determine if the circuit works and wether it meets the requirements or not. In this chapter, we introduce a simple model for calculating transistor sizes based on the theory of logical effort [26, 27, 28, 29]. In this simple model, we assume a linear relationship between the gate driving strength and the delay, by assigning a simple RC circuit to each gate. The gate driving strength determines the values of transistor widths. Finally, we apply this method to the designed adder to find appropriate transistor sizes for driving the load and meeting the power and performance requirements.

4.2 Calibrating RC Models

A simple way to compute the gate delay is to model the gate with an equivalent resistor (R) and capacitor (C). Considering a simple inverter, during pull up or pull down, the transistors can be replaced by a resistor (R). The inverter output capacitor is composed of

two capacitive effects; one is the internal gate capacitors like diffusion caps (C_{diff}) and the other is the load capacitor (C_{load}) which is imposed by the following gates. According to the first momentum approximation, the delay can be estimated with the following formula:

$$Delay = R * (C_{diff} + C_{load}) \tag{4.1}$$

With this simple equation, we can obtain a rough estimation for the gate delay. The precise calculation of gate delay is very complicated and depends on too many factors; like the input rise and fall times, the transistor behavior in each region of operation (linear, saturation,..), short channel effects and threshold voltage variation due to DIBL (drain induced barrier lowering) effect and the load capacitor of the following gates. Even the load capacitor which is the input capacitance of the next stage varies during different operational region of transistors. For example, when the transistors are in saturation, their input capacitance is $\frac{2}{3}C_{gs}$ and when they are in linear region their input capacitance is just C_{gs}. Therefore, computing the precise gate delay and optimizing the transistor widths based on them needs a very complete CAD tool. However, we need to do some hand calculation at the first step of the design to estimate the transistor widths. The method of logical effort is based on the first momentum approximation (Equation 4.1) and offers a good idea for designing a chain of gates, like the one we have in the designed adder.

MOSFET devices are not linear resistors and MOSFET capacitances are not linear either, since they both vary over the different operational regions of transistors. In order to have a good approximations for R and C to calculate the gate delay, we use calibration simulation methods [26]. In calibration method, we do a number of simulations for a specific technology, like 0.18μm and 0.13μm , to derive the equivalent R and C for our models. For example, to derive the equivalent capacitor for each gate, we measure the delay for the two circuits shown in Figure 4.1 and we try to adjust C_{load} to match their

delay in order to find an approximation for C_{gate} of the inverter B. The same approach can be taken for computing the diffusion caps.

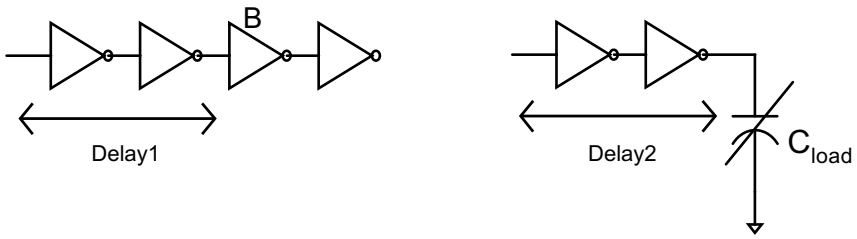


Figure 4.1: Capacitor model calibration

In order to derive the gate resistance, we do simulation on the two circuits shown in Figure 4.2 to match their delay and find the resistor value.

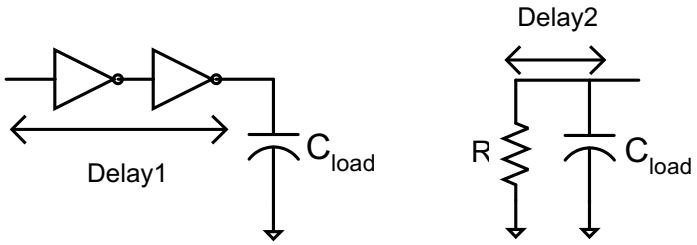


Figure 4.2: Resistor model calibration

After finding the capacitor and resistor for a simple inverter, we do normalize them to the width of transistors used in the inverter, so that we can use them as units for calculating the resistance and capacitance of other gates as well. In order to have equal rise and fall times in static gates, PMOS size is $(\frac{\mu_n}{\mu_p})$ larger than NMOS size, where, μ_n and μ_p represents carrier mobility in NMOS and PMOS devices respectively. If the normalized gate capacitance and resistance for an inverter are C_g and R_{sq}, then we can estimate the following parameters for an inverter with NMOS size of W_n and $(\frac{\mu_n}{\mu_p})$=2.

1. Input capacitance is : C_{in}=$3W_nC_g$

2. Parasitic or diffusion capacitances are assumed to be a fraction of input capacitance, because they are proportional to the transistor width, this prpoprtionality is denoted by parameter α: $C_{diff} = \alpha C_{in}$

3. The gate driving resistance is: $R_{drive} = R_{sq}\frac{W}{L}$

4. Assuming the inverter is driving C_{load}, delay is calculated through this equation:

$$Delay = 3R_{sq}LC_g(\frac{C_{load}}{C_{in}} + \alpha) \tag{4.2}$$

The first parameter in the above equation ($3R_{sq}LC_g$) is constant for each technology and as of now we simply show it by τ_{inv}. The above calibration could be done for other gates like NAND and NOR as well. By knowing the normalized value of τ for each gate and the approximate value of α for each technology, we can estimate the gate delay to some extent without dealing with complex formulas.

4.3 Logical Effort Methodology

The method of Logical effort introduced by Ivan Sutherland [27] is an easy way to estimate delay in CMOS circuits. It provides insight into how the transistor sizes, gate topology and number of stages affect the circuit delay. It is easy to use at the early stages of the design and good as a starting point for more intricate optimizations like the ones done by CAD tools. This model separates the delay caused by the capacitive load that the logic drives and by the topology of the logic gates and quantifies them.

The delay of a logic gate is composed of two components: a fixed parameter which is caused by the parasitic capacitances (p) and another part which is proportional to the

load that the gate drives and we call it effort delay (f). The effort delay itself depends on the load and the gate topology which are expressed with two terms: logical effort and electrical effort. Logical effort (g) is related to the properties of a gate and electrical effort (h) characterizes the load. Therefore, the total gate delay is :

$$delay = gh + p \tag{4.3}$$

Electrical effort (h) combines the effects of external load C_{load} and the transistor sizes in the gates, in the following manner:

$$h = \frac{C_{load}}{C_{in}} \tag{4.4}$$

It basically determine the load driving capability of the gate. Logical effort (g) describes the ability of the gate to drive currents in comparison with a simple inverter and is described by the following formula:

$$g = \frac{\tau_{gate}}{\tau_{inv}} = \frac{(R_{drive}.C_{in})_{gate}}{(R_{drive}.C_{in})_{inv}} \tag{4.5}$$

Logical effort shows how much more input capacitance a gate needs in order to deliver the same output current as an inverter. In order to calculate the logical effort for different gates, we usually assign the appropriate sizes to the transistors in the gate to have the same resistance as an inverter and then we take the ratio of the input capacitance that each input sees to the input capacitance of a simple inverter. Therefore, the logical effort of an inverter is one and as the gates become more complicated, their logical effort become larger. The logical efforts of some simple gates are shown in Figure 4.3. As we can see in the figure, the sizes of NMOS transistors in the stack are twice of the inverter to have the same resistance as an inverter.

The normalized delay for an inverter and a 2-input NAND gate and a 2-input NOR gate are plotted in terms of Electrical effort in Figure 4.4. When the fan out is zero, gate delay

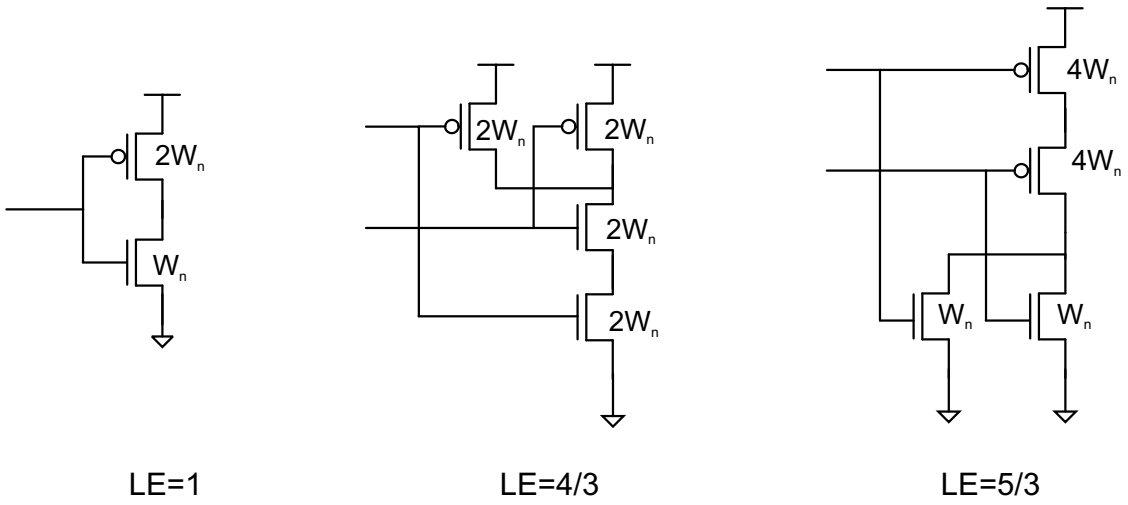


Figure 4.3: Inverter, NAND and NOR Logical efforts

is equal to its parasitic delay (P) and the slope of the curve is the gate logical effort. As it is observed in the plot, more complex gates have larger logical effort and larger parasitics.

Now we want to see how we can take advantages of the introduced parameters for optimizing the transistor sizes in a chain. Considering a simple chain of gates shown in Figure 4.5, the total delay is computed through the Equation 4.6.

$$\frac{Delay}{\tau_{inv}} = \frac{\tau_{nand}}{\tau_{inv}}(\frac{C_{j+1}}{C_j} + \alpha_{nand}) + \frac{\tau_{inv}}{\tau_{inv}}(\frac{C_{j+2}}{C_{j+1}} + \alpha_{inv}) + \frac{\tau_{nor}}{\tau_{inv}}(\frac{C_{j+3}}{C_{j+2}} + \alpha_{nor}) \tag{4.6}$$

This equation can be written in another fashion to include the logical effort and electrical effort parameters of each gate.

$$Delay = (g_{nand}h_{nand} + p_{nand}) + (g_{inv}h_{inv} + p_{inv}) + (g_{nor}h_{nor} + p_{nor}) \tag{4.7}$$

To minimize the delay, we take the partial derivative with respect to 'h' parameter for each

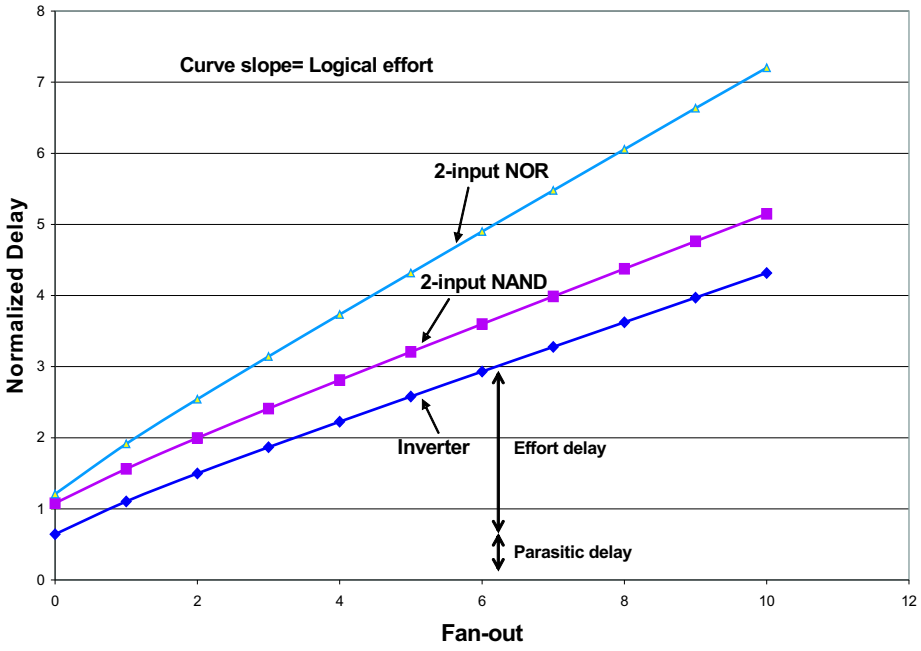


Figure 4.4: Delay versus Fan-out (Electrical effort)

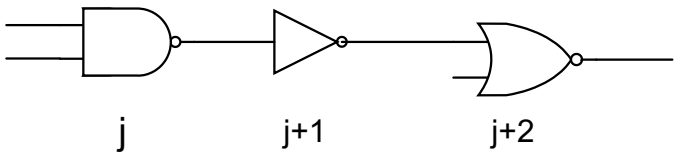


Figure 4.5: Simple chain of gate

gate which results in the fact that the delay is minimized when each stage bears the same effort:

$$g_{nand}h_{nand} = g_{inv}h_{inv} = g_{nor}h_{nor} = f \tag{4.8}$$

In general, when we have a chain of gates, we define a term called path effort as follows:

$$F = (g_1h_1)(g_2h_2)(g_3h_3)...(g_nh_n) \tag{4.9}$$

The minimum delay is achieved when all the gate efforts are equal:

$$F = f^n \tag{4.10}$$

and the total delay is:

$$Delay = \sum(g_ih_i + p_i) = nf^{\frac{1}{n}} + P \tag{4.11}$$

It should be mentioned here that, if a gate is connected to more than one gate in a chain, then a factor called "branching effort" appears at the above equation which is the product of all number of branches along the chain. Since the parasitic delays are constant, they do not affect the procedure of finding the least delay. Now, we can find the electrical effort of each gate by the following equation:

$$h_i = \frac{f}{g_i} \tag{4.12}$$

By using the above equation, we can find transistor sizes in the chain, starting from the end of the path and going backward. Since the capacitive load at the output of a chain is defined, by knowing the stage effort and logical effort for the last gate, we can determine the required input capacitance of that stage. Input capacitance then can be distributed

appropriately among the transistors connected to the input. After that, we can find the input capacitance of the next stage because we know its output load, and so on. It is good to point that the $\frac{C_{out}}{C_{in}}$ ratio or electrical effort is simply the transistor width ratio if the transistors have the same length which is usually the case in digital design.

Usually there are two cases encountered during gate sizing in a path. In one of them, the input and output capacitances are known and the number of stages are fixed; therefore the whole path effort is determined and based on that we can find the stage effort and size the transistors for each gate by using the above equations. In the other case, the input and output capacitances are known, but the number of stages are flexible. In this case, even if the logic has already been implemented by a specific number of gates, we can add extra inverters to optimize the path. If we assume that the logic is implemented by n_1 stages and now we have added some inverters to make them N stages, then the delay would be:

$$D = NF^{\frac{1}{N}} + (\sum P_i) + (N - n_1)p_{inv} \tag{4.13}$$

Taking the derivative of the above equation with respect to N, gives the following equation:

$$P_{inv} + \rho(1 - \ln \rho) = 0 \tag{4.14}$$

Where ρ is the stage effort or $F^{(\frac{1}{N})}$. As we can see in the above equation, the optimum value of ρ depends on the parasitic delay of inverters. Finding the value of ρ leads to finding the optimum number of stages which is estimated by $(\frac{\ln F}{\ln \rho})$. Assuming $P_{inv} = 0$, ρ is computed to be e=2.718. There is not a fixed solution for this parameter, but if we sketch the delay in terms of it, we will see that it has a smooth change from 2 to 5; therefore, we can choose ρ in this range and find the optimal stage effort and also number of stages based on trial and error. If we need to add extra inverter stages, we should be careful to add even number of them and this limitation may deviate us from the optimum

number, but it is not that much critical, since the deviation is so small that can not be sensed in the total delay. Sometimes the number of stages in a design is more than the optimum number gained by logical effort formulas, so we have to stick to the stage effort number (ρ) and size our transistors based on that.

The concept of logical effort can help us to decide on the type of logic families as well. For instance, a pseudo-NMOS circuit does not have a large PMOS transistor network in the pull-up path; therefore it has less input capacitance than its conventional static gate counterpart and hence lower logical effort. Domino circuits are even better than pseudo-NMOS gates because they avoid static power consumption and the contention of PMOS current during evaluation. In dynamic gates, the pull down path transistors are sized for current equal to that of a unit inverter and the input capacitance is not wasted on slow PMOS transistors like static gates; therefore their logical effort is lower than static circuits and they are almost two times faster. As a rule of thumb, it has been indicated by professional designers that stage effort of 2 usually gives the optimal answer in domino logic design [19, 27]. There are two special cases in logical effort approach which are explained here:

Logical effort for asymmetric logic gates:

Unlike the NAND and NOR gates, not all logic gates have the same logical effort for each input. For instance, the static gates used in our designed adder are asymmetric gates and each input has a different logical effort. In these kind of gates, the path effort calculation is done for each input separately. As an example, one asymmetric gate is shown in Figure 4.6. In this gate, it is assumed that $\frac{\mu_n}{\mu_p}$ is two and the gate is sized to have the same rise and fall time. Due to the asymmetry, input C has a different capacitance from input A and B and therefore, its logical effort is $\frac{5}{3}$ while logical effort for input A and B is $\frac{6}{3}$.

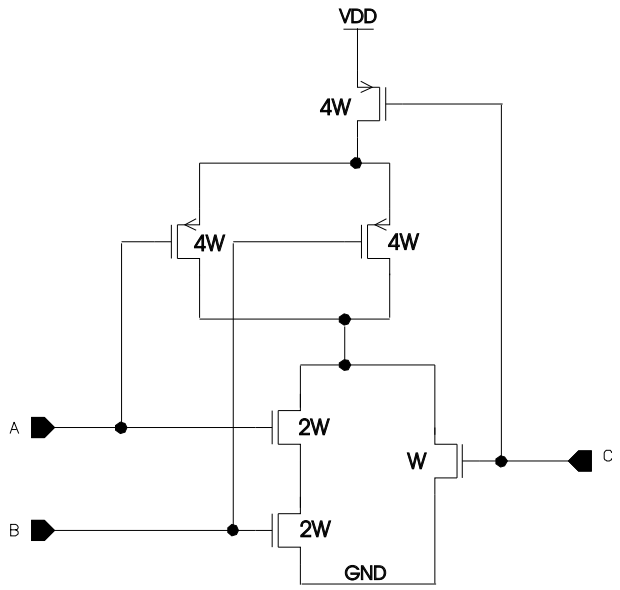


Figure 4.6: Logical effort for asymmetric logic

Logical effort for skewed logic gates:

In a specific circuit, when the transition in one direction is more critical than transition in the other direction, it is possible to design the gates in favor of that critical transition [30]. These kind of gates are called *skewed gates* and they use more fraction of their input capacitance for the critical transition. Skewed gates are sized such that they produce current equal to the inverter in critical transitions and less current in noncritical transition. Therefore, they have lower logical effort for the critical transition. It should be pointed that since the gates are skewed in one direction, rise and fall times are not equal and path effort should be calculated for each edge separately.

In domino design, since the evaluation is done through one direction, dynamic outputs going down and static outputs going up, the static gates between the domino stages can be highly skewed in favor of this transition, meaning that PMOS pull-up transistors can be stronger than NMOS pull-down transistors. But, there is a limit for how far they can

be skewed, in other words, how much we can make the noncritical transistors smaller. Skew improves the logical effort and hence the delay in one direction (evaluation time) but it retards the noncritical transition drastically. In domino design, noncritical transition happens during the precharge time which is not important for us in terms of data and information, but the problem is that we should be able to precharge all the nodes inside the circuit in half of the clock period and consume as small power as we can. Therefore the amount of skew should be such that the precharge time does not limit the performance and does not consume too much power.

Series stacks revisited:

One of the prime assumptions in our models for logical effort was that connecting two devices in series doubles R_{drive} and so the two circuits in the Figure 4.7 have identical current to discharge the output capacitor and hence the same delay ignoring the stack effect and threshold variation of the transistors in the stack.

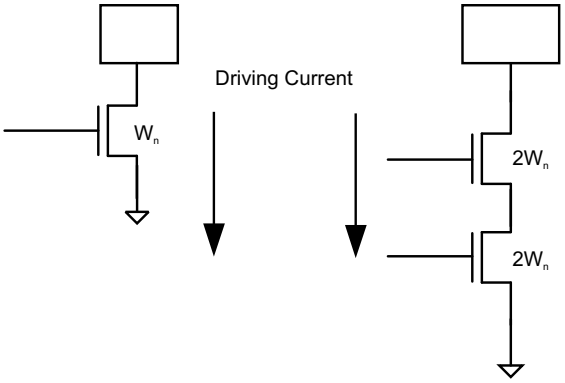


Figure 4.7: Series stack comparison

It can be proved that two transistors in series are equivalent to one transistor of twice the channel length [31]. Since the transistor length is twice, it is less velocity saturated or it

is less affected by short channel effects than the transistor with half of its length; therefore it has more than half of the current of the original transistor. It can be analyzed through the velocity saturation model as well [32]. In short channel regime, NMOS transistor current is computed through the following formula:

$$I_D = W v_{sat} C_{ox} \frac{(V_{GS} - V_{TH})^2}{(V_{GS} - V_{TH}) + E_c L} \tag{4.15}$$

As we can see in this formula, increasing L by a factor of two does not give half of the original current. For instance in 0.35 μm technology , E_cL=2.4 for V_{GS}=3.3v, so doubling the transistor length drops the current to 68% rather than 50% and we need to make transistors in the stack almost 1.5 times wider to have the same driving capability as one NMOS transistor [26]. As a result, the logical effort of the gates becomes smaller than before. Simulation results in 0.18μm technology have shown that NMOS transistor in a two series stack should be 1.6 times wider than one single NMOS transistor and in a three series stack, almost 2.3 times of one single transistor to have the same driving current in the pull down path. Since PMOS transistors are less affected by velocity saturation effects, they should still be about 1.9-2 times wider than a single PMOS transistor to have the same current, as it is proved through the simulation.

4.4 Application of Logical Effort to CDL Chain of Gates

Here, the implementation of logical effort technique on a chain of compound domino (CDL) gates is demonstrated. This chain of gate, shown in Figure 4.8 is considered as a simple vehicle for applying different techniques discussed in the next chapter and is similar to the

longest path in the carry merge tree structure. The load size at the output is 20um and the stage effort is 2. The static gates are skewed by a factor of 3 and we can not go beyond that, because we want to keep the precharge time almost the same as the evaluation time. The logical effort of dynamic gates with two transistors in the stack is 1.6/3 instead of 2 and 2.3/3 for dynamic gates with three transistors in series according to the previous discussion. At the first step, the PMOS precharge transistors are sized such that they provide half of the current provided by NMOS pull down path, but after the sizing is done we should upsize them so that the precharge time does not exceed the evaluation time.

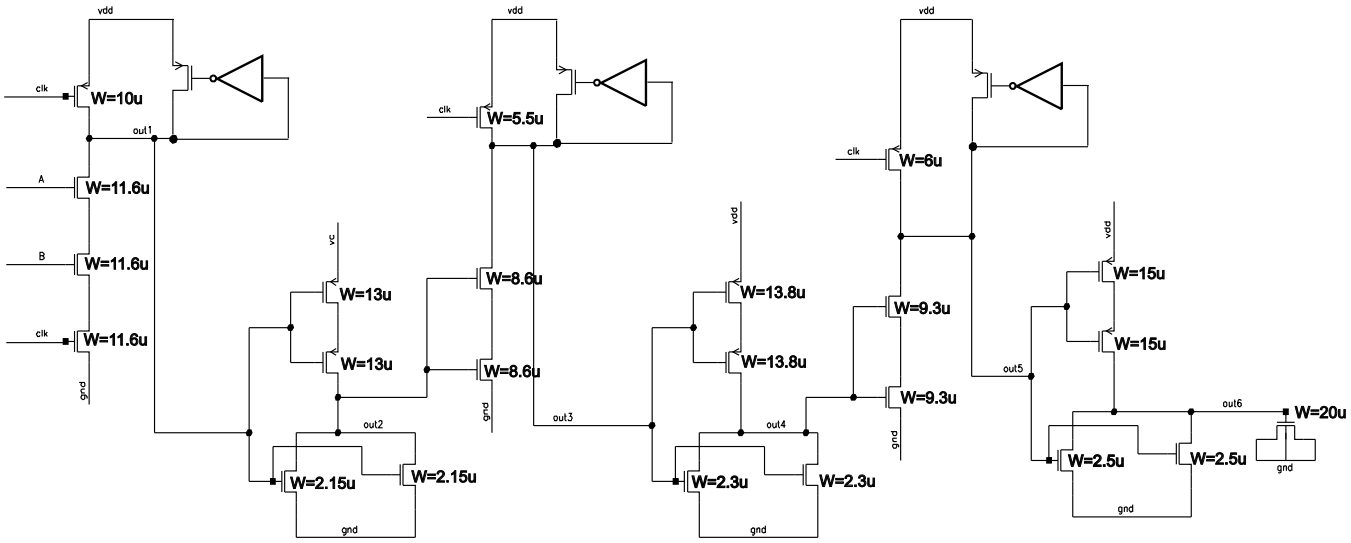


Figure 4.8: CDL chain of gate sizing

4.5 32-bit Adder Transistor Sizing

As explained before, carry merge tree and ripple carry adder outputs are connected to the multiplexer inputs and multiplexer outputs are driving latches. In order to size the transistors in the adder blocks, we should know about the load sizes driven at the outputs.

In the ALU system, the outputs of the latches are seeing a load of approximately 20um size; therefore the transistors in the latch are sized according to these loads and based on the logical effort. In this design, we have assumed stage ratio of two for sizing the gates in a chain and after the sizing is done, we fine tune transistor sizes in the critical path to improve the delay. This task can be completely done by an advanced CAD tool which is found in the industry and is mostly based on Lagrange multipler . We size the transistors starting from the latch and going backward to the first stage of the carry merge tree and ripple carry adder with a stage effort of two.

By doing some calibration simulation in 0.13μm technology, we have observed that in order to have the same rise and fall delay in a simple inverter, the PMOS size should be approximately two times of the NMOS transistor. This rule should be applied to transistor sizing in the ripple carry adder, since delay should remain the same for both transitions. Worst case delay in the ripple carry adder is measured at the time that the worst case input vectors are exerted to that. These vectors are the ones that cause the carry to ripple through all four bits and are either '1111', '0001' , Cin=0 or '1111', '0000', Cin=1. On the other hand, in order to have the same rise and fall time in the wave shape, the PMOS transistor should be approximately three times of NMOS, and this rule is applied to the input and clock buffer sizing, since there, we need a symmetric edge shape for both transitions.

In the carry merge tree structure, the evaluation is done in one direction as it was mentioned before; therefore the static gates are skewed intentionally in favor of evaluation phase. We have used PMOS/NMOS ratio of 4 and 5 in static gates across the carry merge tree. The PMOS precharge transistors in the dynamic gates are sized to be almost 1/3 of the equivalent NMOS transistor of the pull down path, but this rule can not be followed through the whole design, because the precharge time limits us in another way. Since

the NMOS footer transistors are eliminated from all dynamic gates except the first stage, the precharge time gets very slow due to the contention that exists in dynamic gates. In dynamic gates, the PMOS precharge transistor wants to pull up the dynamic node while the static gate outputs have not been pulled down yet and thus keep the NMOS transistor in the dynamic gates on during precharge, causing short circuit current which leads to extra power consumption as well. As of now we call this effect, *"precharge contention"* to be referred easily in the next chapters. Due to the mentioned effect, the PMOS precharge transistors should be sized large enough to be able to pull up the dynamic nodes at half of the clock period and the static gates can not be skewed too much in favor of evaluation time. The worst case input vector for the carry merge tree is 'FFFF', '0001', because it causes the carry to ripple through all stages of the tree. The designed adder was simulated under the worst condition temperature which is 110 °[C] and worst condition input vectors and the delay was measured to be 342 ps in 0.13 μm technology.

Chapter 5

High performance and Low power Circuit Techniques

5.1 Introduction

In order to meet the demand for high performance applications, the cycle time in microprocessors has been dramatically decreased. The frequency of operation has increased faster than that provided by technology scaling due to the high level of pipelining in the architecture level and using new high performance circuit techniques at the circuit level. The frequency increase and the larger number of transistors on the chip along with slower decrease in supply voltage have led to more power dissipation and high chip power density which cause problems in chip thermal management and heat removal process. Recently different design techniques were proposed in order to decrease the power dissipation in different types of circuit design and the most efficient is the power supply voltage reduction. But the supply voltage decrease causes an increase in the delay time and retards the circuit speed. Therefore, we need to investigate the new techniques that decrease the

power dissipation without any performance penalty.

In this chapter, we explain some techniques which have been investigated to be used in the designed adder to reduce its power consumption. In order to introduce these techniques, we use the simple chain of compound domino gates explained in the previous chapter as a benchmark for its simplicity.

5.2 Dynamic gate power consumption

Domino circuits are a popular circuit style to meet the performance requirement of today's microprocessor circuits. However they are less power efficient compared to static logic due to significant clock loading and high switching activity. The total power in dynamic logic block is composed of the clock tree power and the power burned in the gates [33]. The total power consumption of a dynamic logic blocks is represented by:

$$P_{total} = P_{clk} + P_{logic} \tag{5.1}$$

$$P_{clk} = \alpha_{clk}.C_{clk}.V_{DD}.V_{clk-swing}.f_{clk} \tag{5.2}$$

$$P_{logic} = \sum_{i} \alpha_{logic(i)}.C_{logic(i)}.V_{DD}.V_{logic-swing}.f_{clk} \tag{5.3}$$

Where, α_{clk} and $\alpha_{logic(i)}$ are the the switching probability of clock and each domino gate respectively. C_{clk} is the total transistor capacitance in the clock network and $C_{logic(i)}$ is the output node capacitance of each domino gate (i). $V_{clk-swing}$ and $V_{logic-swing}$ are the voltage swing level of clock and dynamic logic gates which are equal to the used power supply in conventional design and f_{clk} is the operating clock frequency. It should be pointed out that the static gates (the inverter or complex static gates after dynamic stages) power consumption is not included in the above equations. Since the power consumption is proportional to the clock and logic voltage swing, voltage swing reduction can be an efficient

way to reduce the total power. In the compound domino logic structure, the evaluation is done in one direction in which precharged dynamic outputs go low and pre-discharged static outputs go high. Therefore, not only reducing the voltage swing at the output of dynamic gates do not degrade the circuit speed in this direction but it can improve it, because dynamic outputs reach to the static gate trip point sooner. On the other hand, we can not reduce the static gates voltage swing because their delay is directly proportional to that in evaluation direction. As it can be observed in Figure 5.1, static gate delay degradation due to voltage swing reduction is more than dynamic gate speed enhancement due to lower swing. Therefore, in order to achieve power saving without any performance trade-off, we have to use a low swing design for dynamic gates and keep the static gates in their previous voltage swing.

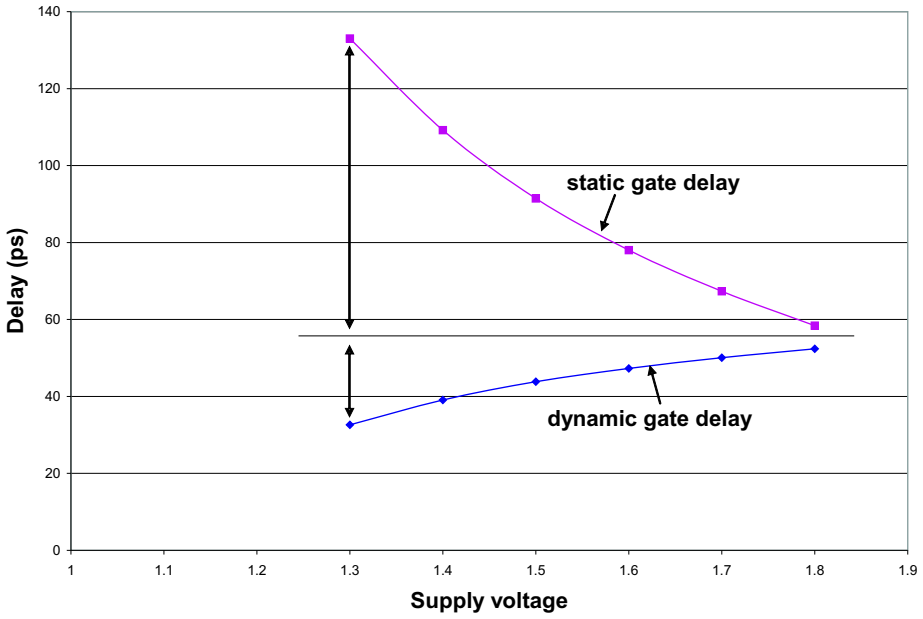


Figure 5.1: Static and dynamic gate delay versus voltage swing

5.3 Low Swing Domino Techniques

There are some low swing domino circuits [34, 35, 36, 37] proposed in the literature, in which, one NMOS transistor is used as a level shifter to shift the voltage by an amount of V_{TH}. They are all shown in Figure 5.2.

In circuit 'a', NMOS transistor is used as the precharge transistor to obtain a swing of $V_{DD} - V_{TH}$ at the dynamic node. Since the precharge is done through NMOS transistors, we need two inverters at the output to drive the NMOS keeper. These two inversions at the output increase the contention time between the keeper and the pull down path during evaluation and affects the performance and power. In circuit 'b' and 'c' , one NMOS transistor is positioned after the precharge transistor to shift the voltage. Circuit 'd' is using NMOS transistor to make a virtual power supply at the source node. According to our simulation results on adder critical path (CDL chain of gates) shown in Figure 5.3 in 0.13μm technology, in all these techniques, we are able to achieve almost 4% performance improvement but we do not have power saving.

It is important to mention that this speed enhancement is smaller than what is expected in low swing gates. If we model the dynamic branch transistors with ideal current sources during output discharge, according to the Equation 5.4, we expect 23% speed improvement when we move from 1.3V to 1V, because the static gate's switching threshold voltage, for both is the same.

$$Delay \propto \frac{C}{I}V \tag{5.4}$$

However, while the delay gets better, the transition fall time at the output node is degraded by almost 10%, because the amount of current in the above equation changes with transistors' V_{DS} voltage and more V_{DS} causes stronger current sources and thus a sharper

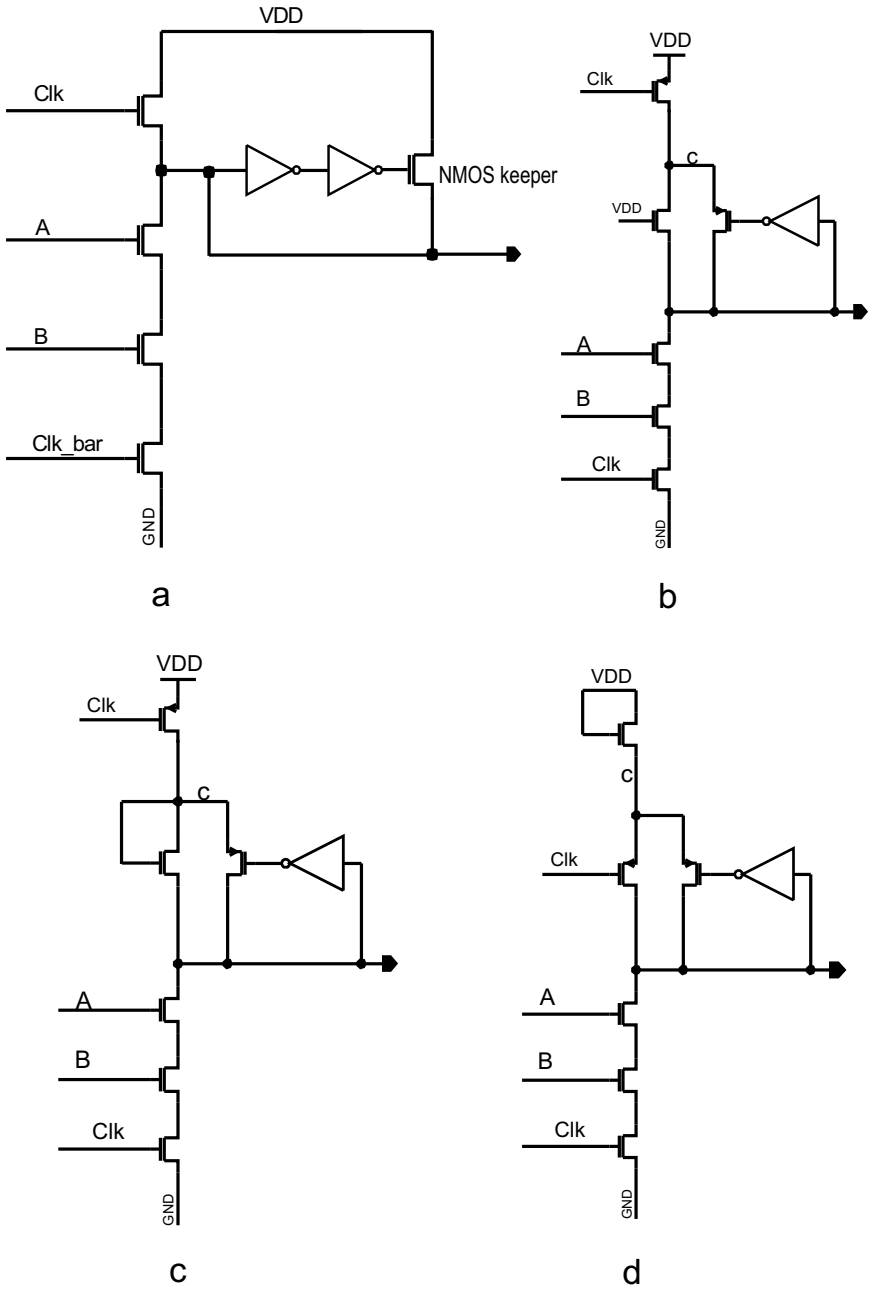


Figure 5.2: Low swing domino gates

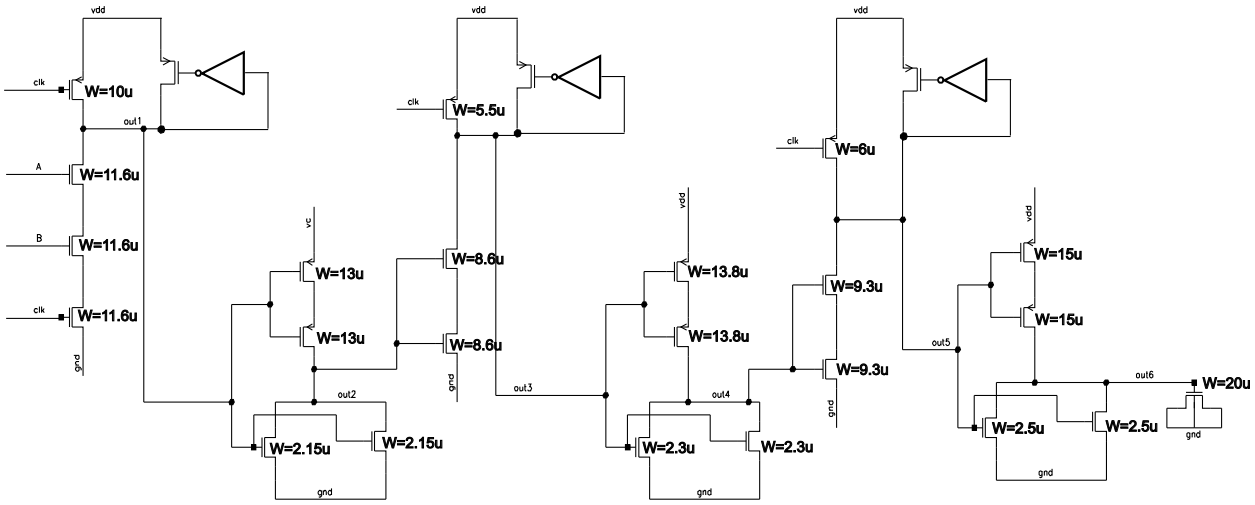


Figure 5.3: CDL chain of gate

transition at the output. Slower fall time in the static gates inputs adds to their delays and thus cancel out the delay reduction obtained through lower swing on dynamic gates and remain almost 4% performance enhancement. In addition to that, slower fall time at the static gates' inputs causes more short circuit power consumption and cancel the power saving achieved on low swing domino gates. There are some effects like keeper contention in circuit 'a' and capacitor charging and discharging in NMOS level shifter's drain (node c) in circuits 'b' and 'c' which cause more power consumption in these configurations. However, there is a major problem in these circuit styles that we name it as *"dynamic offset voltage"*. Our investigation has shown that after some cycles of inactivity in the dynamic gates, the output voltage will increase by an order of 50mv. This effect is shown in the CDL chain output waveforms in Figure 5.4.

This offset becomes larger for longer time of inactivity. Therefore, the circuit does not show the same delay and power all the time and its behavior changes with the input and previous state condition which makes it an unreliable design style. Here, we explain the

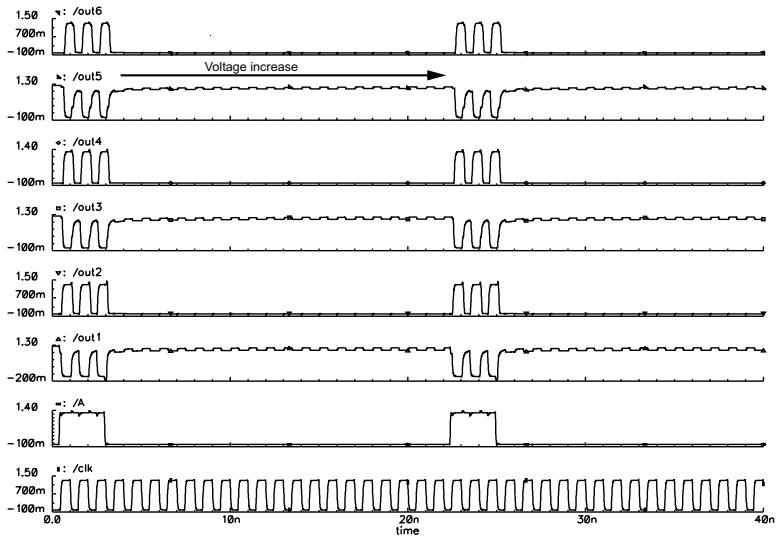


Figure 5.4: Output of CDL chain implemented with low swing dynamic gates

reasons behind this observation.

Leakage current:

In digital gates, when we pass the power supply through NMOS transistor, we assume that the source node is charged up to $V_{DD} - V_{TH}$ and after that the transistor is turned off, while in real life it is not true and we have leakage current through that transistor which can steadily increase the voltage over source node [31]. Therefore, if the NMOS transistor stays at precharge mode for a long time, the source voltage will go up beyond the amount of $V_{DD} - V_{TH}$ that we expected and we no longer get the benefit of the low swing effects. In other words, the low swing voltage is dependent on data activity of the circuit.

Charge injection:

When NMOS transistor is placed on the supply path in a circuit, it acts as a diode, because it does not convey current from the source to drain as long as the source node is lower than

the drain node voltage. This diode causes some bootstrap effects in these circuits. When the transistor turns off, its channel charge discharges from both ends and the charge which finds the path through the source, stays there and accumulates at the source node. This phenomena happens in normal designs with PMOS precharge transistors as well, but the extra charge will go back to the supply in the next cycle that the PMOS transistor turns on, while when we have NMOS transistors, the charge does not go away because of its diode behavior. This effect is similar to charge injection effect in sample and hold circuits. Therefore, some voltage offset at the dynamic gate output happens when we have some cycles of inactivity.

Since the charge injection problem is similar to what which happens in sample and hold circuits, we have investigated some solutions which are suggested in those circuits [38] to avoid this offset problem to some extent. Circuit D in the Figure 5.5 uses a PMOS dummy switch at the drain node of precharge transistor (it can be placed at the source node or both, as well) which switches in the opposite way of precharge transistor and therefore is able to sink the channel charge inside and cancel the charge injection effect but its sizing should be carefully done to behave in this way. The problem of having dummy transistors is that they consume power and it is not desirable at all. In transmission gate precharge form, circuit E in Figure 5.5, we add one NMOS transistor in parallel to the PMOS precharge to be switched in the opposite way and cancel the charge injection effect. It can help PMOS transistor in the precharge mode and therefore it does not waste power.

In both techniques (D, E), the sizing is very important because it should have the same amount of channel charge as the PMOS transistor, so they are very sensitive to process variation. The CDL chain output waveforms which uses technique E are shown in Figure 5.6. As we can see through the figure, the wave forms have become flat, but still we see the voltage increase after some cycles of inactivity due to the leakage current and having

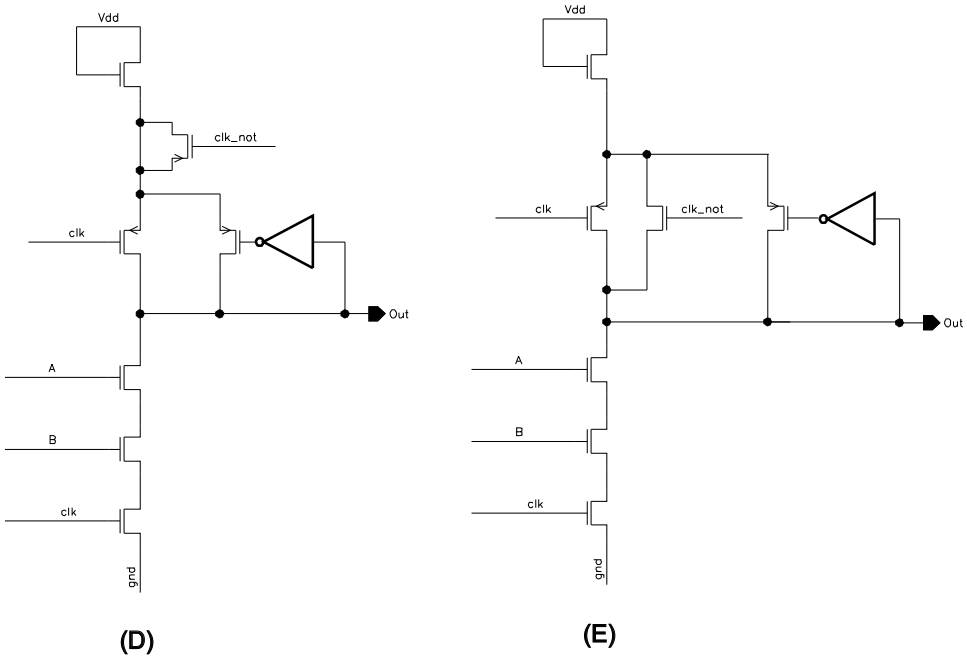


Figure 5.5: Solutions for charge injection problem

unmatched transistors.

Another solution is to add a large capacitor at the NMOS source (node c in Figure 5.2), so that the charge injected at that node can not change the voltage significantly for the worst case data activity condition. If we use a fairly large capacitor like 8 pF at that node, we see some fluctuations by an order of 50mv during 10-20 cycles of inactivity as is shown by waveform 'O1' in Figure 5.7.

All the circuit styles which were introduced as low swing domino gates are not robust and do not show a constant behavior for random data activities. Therefore, they should be used in designs that has a certain amount of data activity. In some digital blocks, we want to reduce the delay in some parts of the critical path without changing the configuration too much, these circuit styles can be helpful in those, if we can make sure they show a

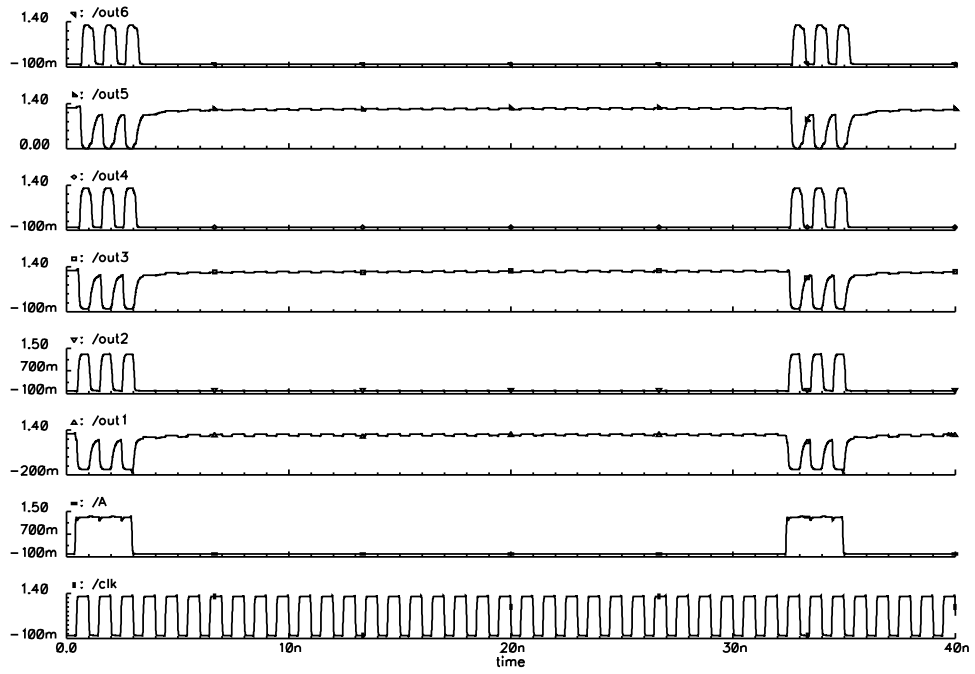


Figure 5.6: Outputs of CDL chain implemented with circuit E

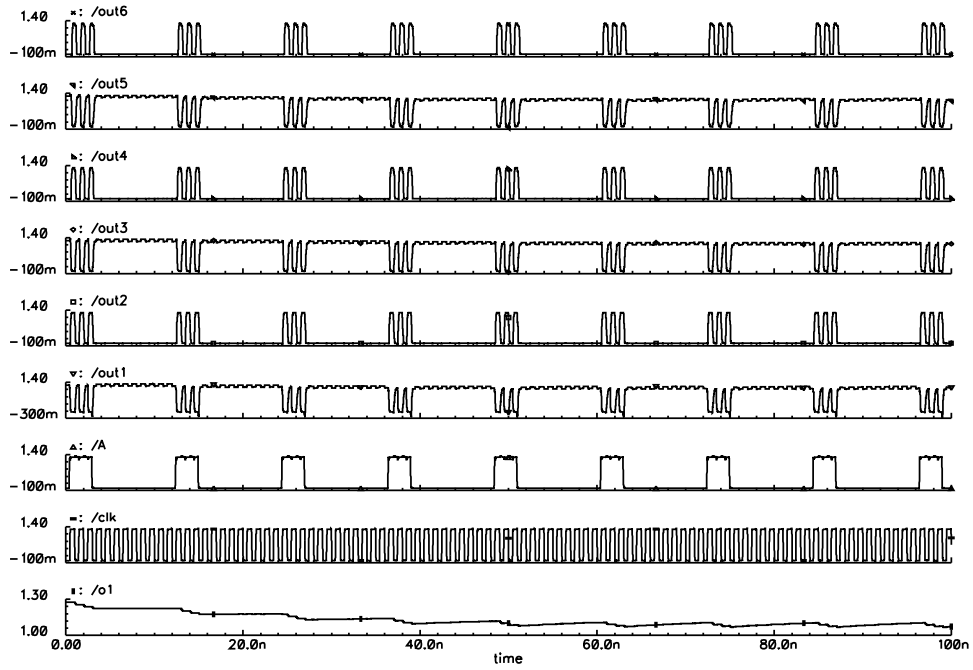


Figure 5.7: Outputs of CDL chain implemented with circuit C and large capacitors

continuous behavior for the circuit function.

5.3.1 Dual Supply Scheme for CDL gates

As it was seen before, low swing domino circuits do not show a robust behavior for all circuit conditions. The other option is using a lower supply for dynamic gates and clock network and keeping the static gates on the conventional supply as illustrated in Figure 5.8. In this configuration, both swing and supply voltage is reduced and therefore we expect a quadratic power reduction in clock network and dynamic gates. However, there are some issues in this style which will be explained here.

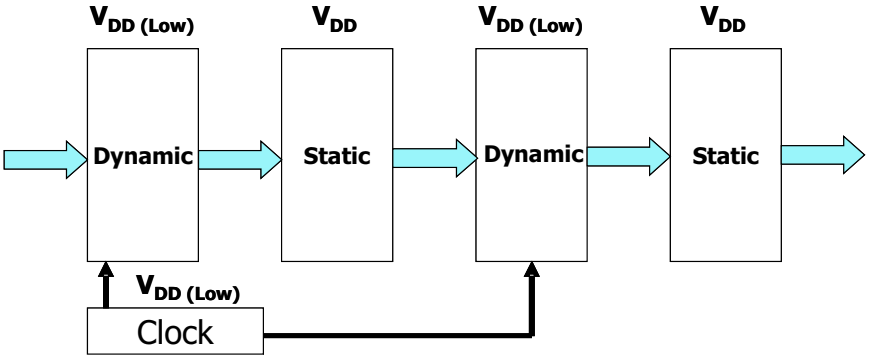


Figure 5.8: Dual supply scheme for CDL chain of gates

5.3.2 Static Leakage Current

Keeping the static gates at a higher supply than dynamic gates', causes static leakage current in the circuit, because the PMOS transistors in the static gates pull up path are not completely off and as a result, the subthreshold leakage current increases. This leakage current puts a constraint on how far we can reduce the dynamic gate supply voltage. In the Figure 5.9, the average leakage current of the designed CDL chain is shown for two

technologies: 0.18μm and 0.13μm and for different supply voltages on domino gates. The measurement is done at 110 °[C] temperature for the worst case analysis, since the leakage current increases with temperature. As we can see in the figure for 0.18μm, the curve has a breakpoint around 1.5V and we have a sharp increase in leakage current below this voltage which is not tolerable in today's microprocessors' chips. The same effect can be seen for 0.13μm technology around 1.1V. By comparing these two technologies (0.18 and 0.13μm) we find that the voltage range in which the leakage current is tolerable is wider for 0.18μm technology. The reason behind this observation is that PMOS threshold voltage is higher in 0.18μm technology and we have more room for voltage reduction until PMOS transistors turn on. We can conclude that for shorter channel lengths, this range will become too small that we can not get any benefit out of dual supply technique.

5.3.3 Increased Noise Vulnerability

Low voltage swing at the domino outputs make them more vulnerable to the noise, because the amount of voltage drop on the domino nodes to reach the static gates' trip point is lower and the dynamic gates inputs are driven by the higher supply voltage. In order to measure the quality of dynamic gates' noise robustness, we use the concept of UNGM (unity noise gain margin) [39] which is defined as the amount of dc-noise level on the inputs of the precharged gate generating an equal level of noise at the output of the static inverter immediately after the precharged gate. In order to measure UNGM in simulation, we apply a slow ramp at the input of dynamic gates and measure the voltage intersection of the ramp input with the static gate output. The ratio of this voltage to the supply voltage is taken as UNGM factor. Simulation results for 0.18μm technology on CDL chain show 7.5% reduction in UNGM when the dynamic gates are on 1.5V in comparison with the conventional design in which all the gates are on 1.8V. In order to pull up the noise

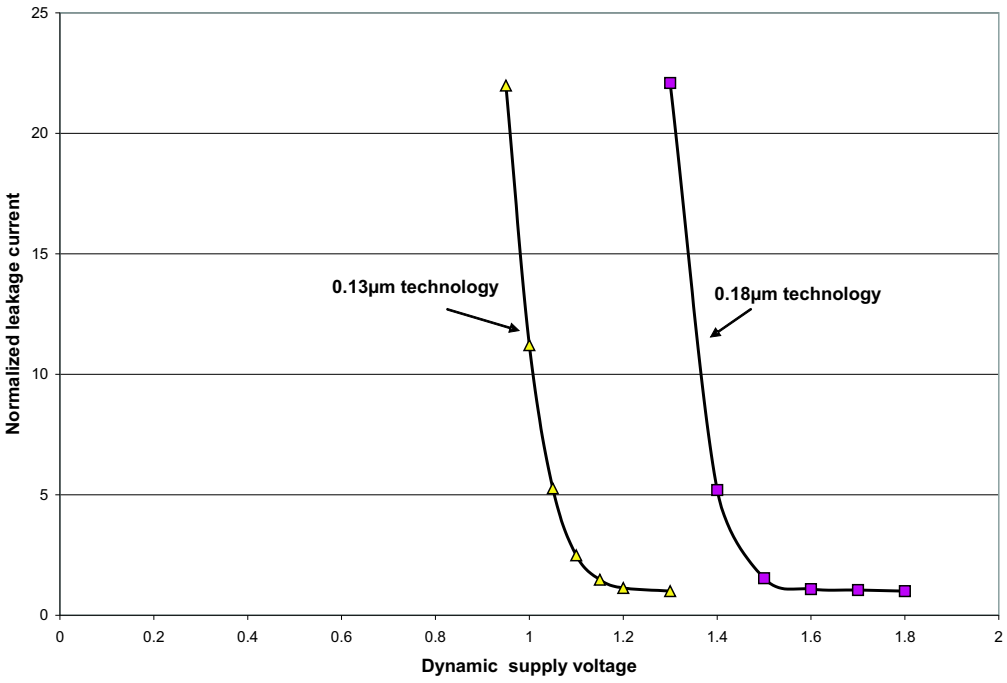


Figure 5.9: Normalized leakage current in two technologies

robustness of the dual supply design, we can increase the PMOS keeper transistor sizes. Simulation results have shown that the keeper sizes in dual supply technique for 0.18μm design should be boosted almost 1.6 times to reach the same robustness as that of the single supply design as it is shown in Figure 5.10.

5.3.4 Precharge Time Increase

Since the NMOS transistors in the static gate pull down path are driven by lower supply, they are not strong enough to pre-discharge the static gates and forward the precharge wave along the chain, therefore; the precharge delay in dual supply style is more than the single supply design. Large delay in precharge cycle can limit the performance and should

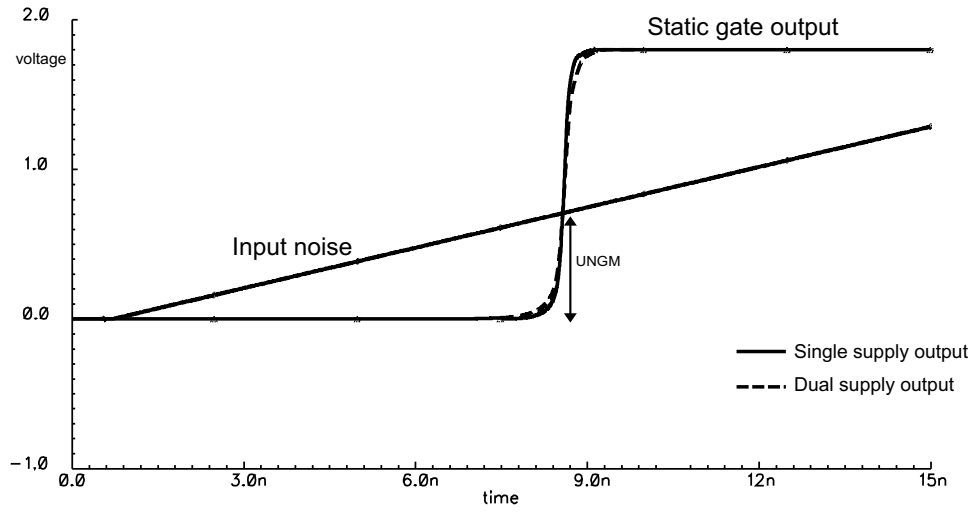


Figure 5.10: UNGM measurement

be avoided by using extra reset devices. In this design, we can simply use some NMOS pre-discharge transistors driven by $\overline{clk}$ at the output of static gates, shown in Figure 5.11. These NMOS transistors should be carefully sized to enhance the precharge delay while they do not burn too much power. The first stage NMOS transistors are the most effective ones and should be strong enough to pre-discharge the first stage static gates and moves the preacharge wave faster to forward to impede the contention caused by *"precharge contention"* effect as much as possible. In other words, by looking at the transitions in precharge cycle, we find that the first stage NMOS transistors are more efficient in power saving and reducing the precahrge time delay. Simulation results in 0.18μm technology has shown that the pre-discharge NMOS transistor at the first stage can improve the precharge time by 22% and reduce the total energy by 4% in comparison with the design without the pre-discharge transistor. It should be mentioned that the amount of energy which is wasted due to charge and discharge of these NMOS parasitic capacitances have been included in the above numbers.

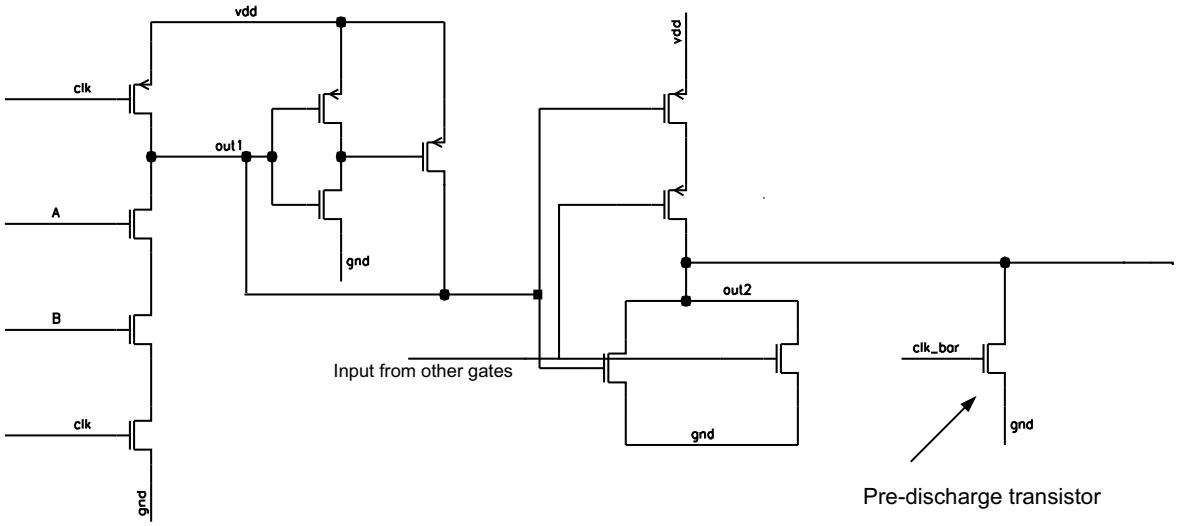


Figure 5.11: Insertion of NMOS pre-discharge transistor

5.3.5 Dual Supply Simulation Results for 0.18μm Technology

Here, we want to compare the delay and power between the conventional or single supply design and dual supply style. According to the leakage current measurement done in 0.18μm technology, we have chosen a supply voltage of 1.5V for dynamic gates and clock network and 1.8V for the static gates. The transistor sizes are kept the same as conventional design and NMOS pre-discharge transistors are added at the static gate outputs to have the same precharge time as single supply design. The PMOS keeper sizes at the output of dynamic gates have also been upsized by a factor of 1.6 to have the same noise robustness as single supply design. Simulation results have shown 3.5% performance improvement over the single supply design. It is important to mention that this speed enhancement is smaller than what is expected in low swing gates due to the fact that was mentioned before in low swing domino circuits. Slower fall time in the static gates inputs adds to their

delays and that, along with lower driving voltage on the first stage clock footer transistor, larger keeper sizes and NMOS pre-discharge parasitic capacitors approximately cancel out the performance improvement we achieved through low swing dynamic gates.

Domino outputs precharge and static outputs pre-discharged conditions are considered as the worst case situation in terms of leakage current, since at that time the PMOS transistors in the static gates are leaking and there is a path for this current to ground in the NMOS pull down branch. The longer the clock cycle is, the more energy is wasted in static gates, so the leakage energy is proportional to clock period. On the other hand, during evaluation cycle, if the dynamic gate does not discharge the output, it will remain in precharge mode and worst case leakage condition, so, the leakage power is closely proportional to data activity or dynamic output switching probability. As a result of the above mentioned facts, in order to have a fair energy comparison between the two designs, we have measured the total energy over equal clock cycles for a data activity range of 0.05-0.6 and for different clock frequencies. The results are shown in Figure 5.12. 'D' and 'S' signs beside each curve show if it is related to dual supply or single supply design, respectively and 'a' is the data activity factor.

As we can see in Figure 5.12, the energy benefit out of dual supply technique diminishes as the frequency of operation decreases, on the other hand as the clock period decreases and gets closer to the total path evaluation time which is the usual case in microprocessor design, we see more energy saving in dual supply technique. As we expected from the above discussion, energy saving in dual supply technique is more when we have high data activity, but we see the benefit even at 0.05, which is considered as the lowest data activity in digital blocks. On the average, simulation results has shown about 23% energy saving in dual supply technique in comparison with single supply at 1GHz clock frequency. Since energy has quadratic relationship with supply voltage, theoretically the amount of energy

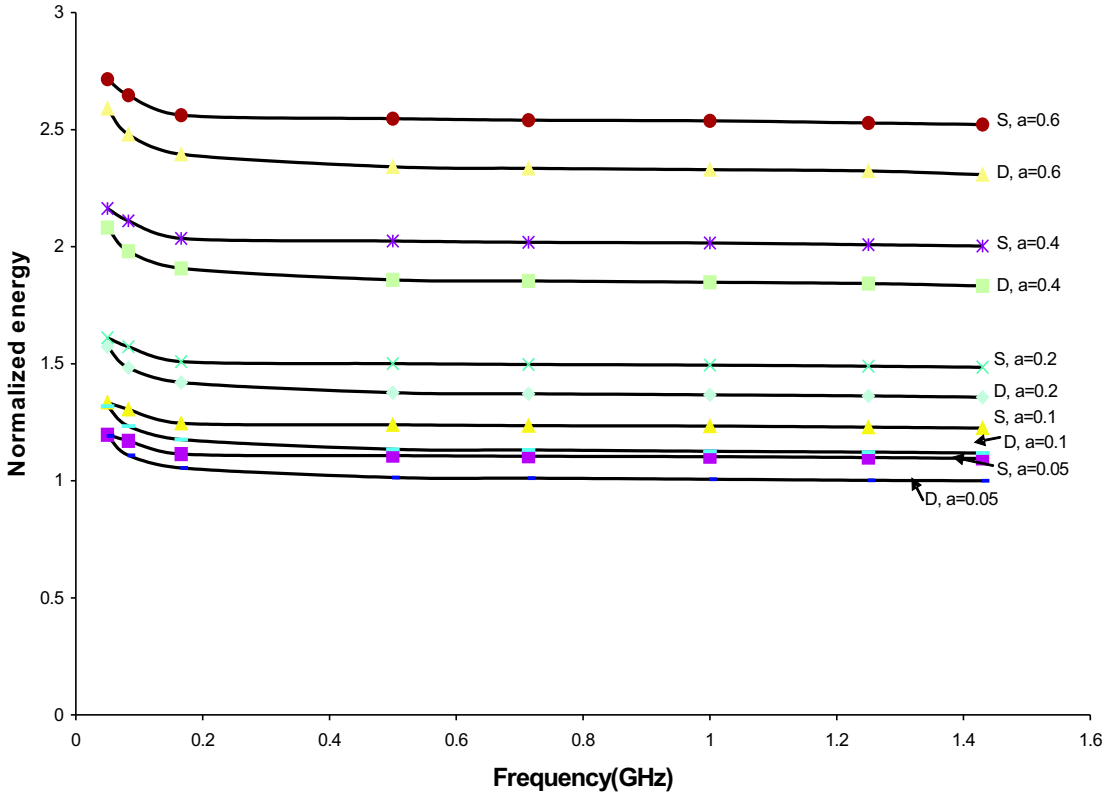


Figure 5.12: Energy comparison in dual supply technique

saving is 30% by changing the supply voltage from 1.8V to 1.5V; therefore the number that we got through simulation is reasonable, because static gates are still on higher supply and in addition we have more leakage current and short circuit current due to the slow transitions at the static gates'inputs.

5.3.6 Dual Supply Simulation Results for 0.13μm Technology

Dual supply technique has been applied to the CDL chain of gates in 0.13μm technology as well, in which the dynamic gates and clock network are on 1.1V voltage supply and static

gates are on 1.3V supply. The NMOS transistors have been used at the static gates output to reduce the precharge time to reach the single supply precharge time and PMOS keepers have been upsized by a factor of 1.5 to have the same noise robustness as single supply design. The performance enhancement in this technology is around 3% and the energy comparison for different data activities is sketched in the Figure 5.13. According to the

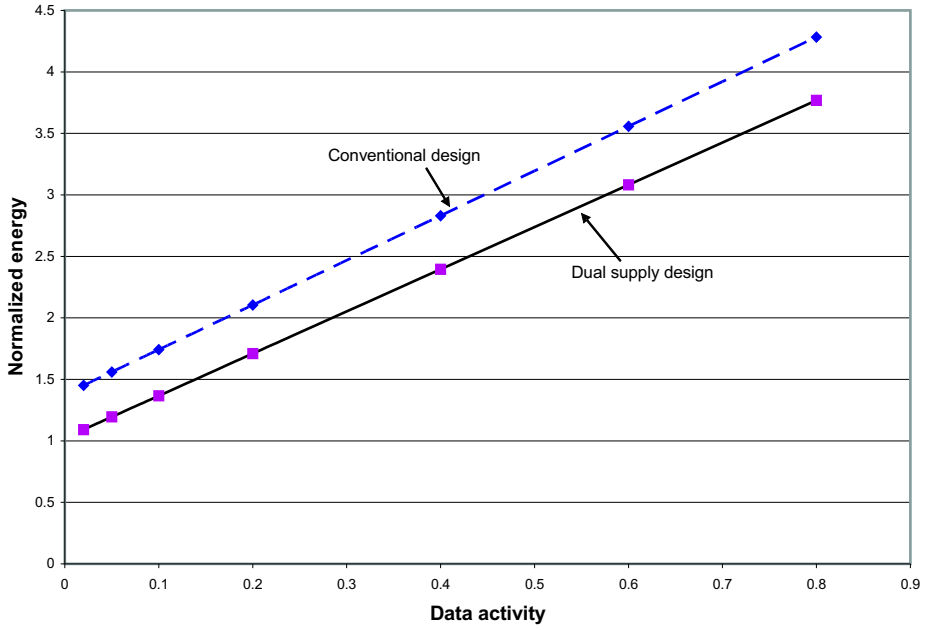


Figure 5.13: Energy comparison for different data activities

simulation results, we have almost 21% energy improvement in this style, in comparison with the conventional design.

Chapter 6

Low Power 32-bit Adder

6.1 Dual Supply Technique implementation on the High Performance Adder

In the previous chapter, some circuit ideas were introduced to increase performance and reduce energy in digital circuits. In addition, their advantages and disadvantages were investigated. Among all of them, dual supply technique shows better energy-delay product enhancement and is the most robust and reliable technique. In this section, we apply dual supply technique to the designed 32-bit adder, introduced in chapter 3, and compare its energy and delay with the conventional design which is implemented with single voltage supply.

Carry merge tree is composed of compound domino logic and therefore it can be designed with two voltage supplies. At 0.13μm technology, dynamic gates and clock network are on 1.1V and static gates are on 1.3V. The keeper sizes have been upsized for all dynamic gates to have the same noise robustness as single supply design. Since the dynamic gates in the

adder block have different transistor sizes in their pull down path and have different static gates sizes in front, we have done the measurement and boosted the keeper sizes to have the same UNGM as of single supply design for each dynamic cell separately. In this way, we make sure that all the dynamic cells have the same noise robustness as the conventional design and we can do a fair comparison between them. In addition to that, NMOS pre-discharge transistors have been added at the first and final stage static gates' outputs to reduce the precharge time which is degraded due to the lower voltage swing on dynamic gates. According to the simulation results, we have about 3.5% performance improvement over single supply design. In addition to that, simulation results show that the amount of switching energy during evaluation cycles is reduced by 16.7% . This energy saving comes from lower swing on both dynamic gates and clock network. During the cycles that we do not have switching in dynamic gates, we have 26.4% energy saving, meaning that, the amount of power reduction on clock switching is more than the leakage power increase. These comparison is shown for normalized values of delay and energy in Figure 6.1. Now, the amount of total power saving depends on the number of cycles that we have activity in the circuits. For example for data activity of 0.1, we have 24% energy saving and for data activity of 0.5, we have 19.4% energy saving.

6.2 Clock Network Power Reduction

Since the clock network switching is one of the big sources of energy wasting in dynamic digital blocks, reducing the clock load can significantly help to save the energy. In the designed adder, only the first stage dynamic gates have clock footer transistors. Since there are two parallel pathes in the first stage, for generating 'p' and 'g' signals, we can share the footer transistor between them and size it according to the larger transistor

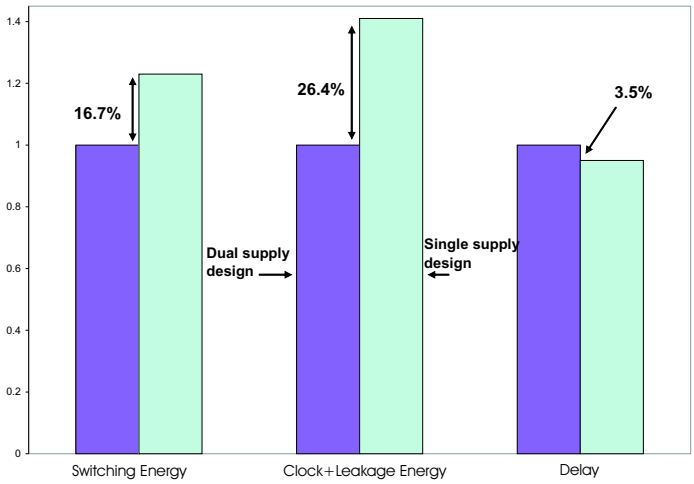


Figure 6.1: Energy and delay comparison for dual and single supply 32-bit adder designs

between the two branches as it is shown in Figure 6.2.

Clock footer transistor sharing does not degrade the performance, because any extra delay is absorbed in the performance gain that we have achieved before. However, it can save clock network switching energy by an amount of 15% which is proved through the simulation results.

Delay and power measurement results for single supply design, dual supply design and dual supply with shared clock footer transistors are summarized in table 6.1.

	Delay	Switching energy	Leakage+clock energy
Single supply	342 ps	122.5mW	46.2mW
Dual supply	330 ps	102mW	34mW
Dual supply with shared clock transistors	332 ps	100mW	29mW

Table 6.1: 32-bit adder simulation results

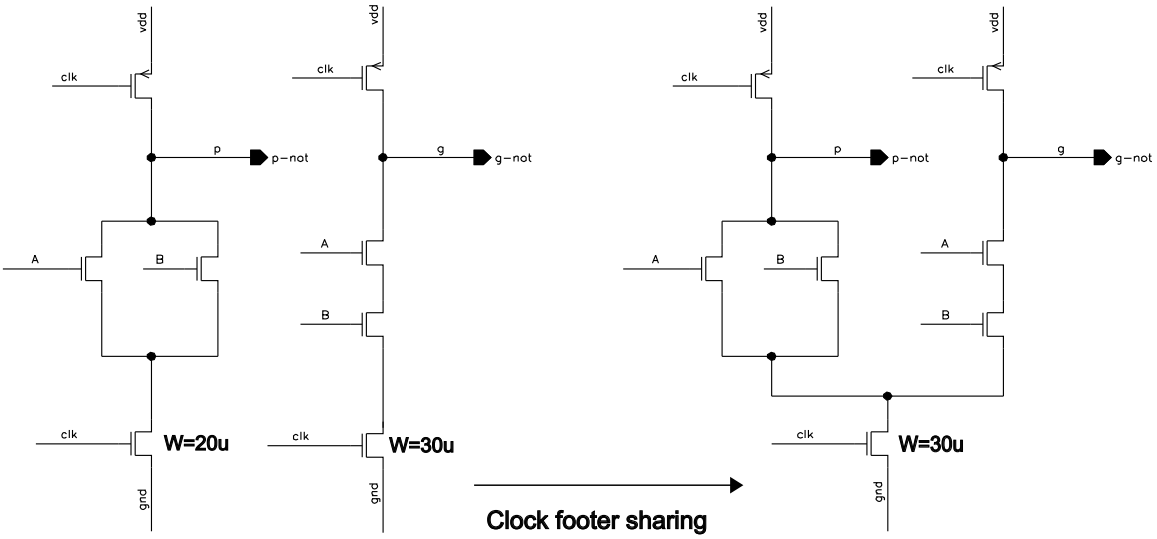


Figure 6.2: Clock footer sharing at the first stage

6.3 Post-layout Simulation Results

The layout of the CDL chain of gates (shown in Figure 6.3) which simulates the critical path in the adder structure has been done in 0.18um technology for both single supply and dual supply techniques. The long wires in the carry merge tree has been imitated in this chain as well. The chip has also been submitted. Since the gates with different supply are placed in a row, routing two power supply is easier than the case in which they are in different columns. In addition to that, routing overhead can be mitigated if we use the shared Nwell process [40]. The amount of delay in layout in only 2% higher than schematic, because the delay imposed by the long wires has been canceled by smaller transistor parasitic capacitors gained by finger techniques.

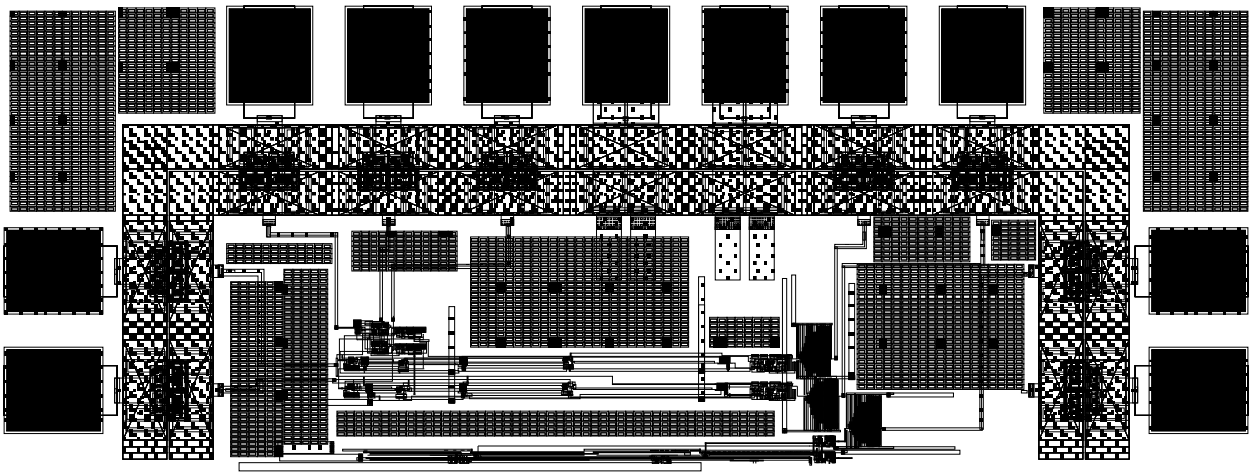


Figure 6.3: CDL chain of gate layout

6.4 DC-DC converters

In dual supply technique, there is an overhead due to the requirement for having two different power supplies. We need to have a DC-DC converter which is usually built on board. DC-DC converters subject is a vast area in electronic design and is beyond the scope of this thesis. The main point here is the converter efficiency. For every DC-DC converter, there is an efficiency number associated with it. One of the common types of DC-DC converters are switched-capacitor ones [16] which can have 75%-98% efficiency factor. In order to have a fair energy comparison between dual and single supply technique, we should include converter efficiency factor. For instance if the efficiency is 90%, then the total power saving is 12% rather than 22%. Therefore, in order to have significant power saving, we need to have a DC-DC converter with high efficiency.

Chapter 7

Conclusions

In this thesis, different state of the art adder architectures were explained and a hybrid architecture which is combined of carry select adder and radix-2 Han-Carlson carry merge tree was introduced for designing a high performance 32-bit adder. After that, different logic styles which are used in today's digital circuits were addressed and among them, transmission gate logic and compound domino logic (CDL) were selected to be used in the designed adder in order to achieve high speed and to dissipate as little power as we can. Logical effort methodology for transistor sizing was discussed and applied to the designed adder to obtain a high speed of 2.9GHz. Then, some low power and high performance circuit techniques were analyzed and their energy and delay were compared with the conventional circuit methods. Finally, a dual supply technique for compound domino logic was proposed and implemented in the designed adder. With this technique, we achieved an average power saving of 22% and 3.5% performance improvement over the conventional design. However, there is an area overhead due to dual supply routing on the chip and there is a requirement for having a high efficiency DC-DC converter.

7.1 Future Works

A. Adaptive Scaling of the Supply Voltage

In today's digital system design, power factor goes side-by-side with performance when designing a computer system. For some systems, careful circuit design may be sufficient to meet the system's power goals. Often, however, the power and performance goals are aggressive enough that a range of circuit, architecture, and software techniques must be used together. There are some power aware softwares and compatible architectures which are able to manage the total system power according to the applications and performance requirement for each function. Different techniques have been suggested in the literature to be used at the architectural level to make it adaptable in terms of performance and energy and one of the most efficient one is adaptive power supply scaling [41, 42, 43]. In this approach, the power supply of some circuit blocks is dynamically changed according to the work load and system speed requirement. The introduced dual supply adder can be used easily in these compatible architectures. By changing the supply voltage on static gates, we can change the performance and energy in both directions. In other words, while the whole system is on the lower supply like 1.1V, changing the static gates' supply toward 1.1V reduces the power and pushing it toward 1.3V enhances the circuit speed.

The power management can be applied inside the adder itself, according to the input data vectors [43]. As it was mentioned before, the adder performance is defined for the worst case input vector which is 'FFFF' and '0001' and causes the carry input to be propagated in the whole carry merge tree. In other words, the worst case delay condition happens when the group P ($P_{0...31}$) signal is one. Now, if we divide the number of inputs to N sections and generate the group P signal for each N parts, we can predict if the carry outputs can be defined sooner or we should wait for the whole carry merge tree to generate the carry

outputs, and according to that, we can switch between different supplies on static gates to meet the worst case delay time and save energy. This technique can be applied if we have enough setup time at the input to generate group P signals and switch the proper power supplies on static gates. The amount of power saving is based on the statistics of the input data. Further exploration of this topic is beyond the scope of this project and is encouraged to be done for future work.

B. Leakage Power Reduction

Another task which is mostly done by power aware softwares is to position the circuit blocks in sleep mode during their inactivity periods to avoid leakage power dissipation. Using sleep mode transistors, adaptive body biasing and applying specific sleep vectors are some common techniques for leakage power reduction during idle times. The dual supply adder dissipates more leakage energy than the conventional one during the precharge condition, because the PMOS transistors in the static gates are not completely off. Therefore, one method for leakage energy reduction during idle times could be applying body bias to PMOS transistors in the static gates to increase their threshold voltage. An other method is using sleep vectors like 'FFFF' that causes all the dynamic gates to be discharged and all the static gates to be charged. This condition is the low-leakage inactive state, in which the voltage difference over the gates does not affect the leakage power.

Bibliography

[1] Jan M. Rabaey. *"Digital Integrated Circuits: A design Perspective"*. Prentice-Hall, 1996.

[2] Anantha Chandarksan, William J. Bowhill, and Frank Fox. *"Design of High-performance Microprocessor Circuits"*. Wiley-IEEE Press, 2000.

[3] R.P. Brent and H.T. Kung. "A Regular Layout for Parallel Adders". *IEEE Transactions on Computers*, 31(3):260–264, March 1982.

[4] Sung-Mo Kang and Yusuf Leblebici. *"CMOS Digital Integrated Circuits Analysis and Design"*. Mcgraw-Hill, 2002.

[5] F.K. Gurkayna and et al. "Higher Radix Kogge-Stone Parallel Prefix Adder Architectures". In *Proceedings of the 2000 IEEE International Symposium on Circuits and Systems*, volume 5, pages 609 – 612, May 2000.

[6] A. Beaumont-Smith and C.-C. Lim. "Parallel Prefix Adder Design". In *Proceedings of 15th IEEE Symposium on Computer Arithmetic*, pages 218–225, June 2001.

[7] S. Knowles. "A Family of Adders". In *Proceedings of 15th IEEE Symposium on Computer Arithmetic*, pages 277–281, June 2001.

[8] Wei-Li Chen, Steve Fang, and Ho-Yan Wong. "Design and Evaluation of Prefix Adders". *University of California, Los Angeles.*

[9] P.M. Kogge and H.S. Stone. "A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations". *IEEE Transactions on Computers*, 22(8), August 1973.

[10] Michael J. Flynn and et al. *"Advanced Computer Arithmetic Design"*. John Wiley, 2001.

[11] Y. Wang, C. Pai, and X. Song. "Higher Radix Kogge-Stone Parallel Prefix Adder Architectures". *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 49(1):16 – 24, Jan 2002.

[12] Yuke Wang and K.K. Parhi. "A Unified Adder Design". In *2001. Conference Record of the Thirty-Fifth Asilomar Conference on Signals, Systems and Computers*, volume 1, pages 177 – 182, Nov 2001.

[13] S.K. Mathew and et al. "Sub-500-ps 64-b ALUs in 0.18-μm SOI/bulk CMOS: design and scaling trends". *IEEE Journal of Solid-State Circuits*, 36(11):1636–1646, Nov 2001.

[14] S. Vangal and et al. " A 25 GHz 32 b Integer-execution Core in 130 nm Dual-VT CMOS". *IEEE Journal of Solid-State Circuits*, 37(11):1421–1432, Nov 2002.

[15] Jan M. Rabaey and Massoud Pedram. *"Low Power Design Methodologies"*. Kluwer Academic Publisher, 1996.

[16] Anantha P. Chandrakasan and Robert W. Brodersen. *"Low Power Digital CMOS Design"*. Kluwer Academic Publisher, 1995.

[17] Mohamed I. Elmasry and Abdellatif Bellaouar. *"Low-power Digital VLSI Design : Circuits and Systems"*. Kluwer Academic Publisher, 1995.

[18] T. Sakurai and A.R. Newton. "Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas". *IEEE Journal of Solid-State Circuits*, 25(1):584 – 594, April 1990.

[19] Kerry Bernstein, Keith M. Carrig, Christopher M. Durham, Patrick R. Hansen, David Hogenmiller, Edward J. Nowak, and Norman J. Rohrer. *"High Speed CMOS Design Styles"*. Kluwer Academic Publisher, 1998.

[20] M. Kontiala, M. Kuulusa, and J. Nurmi. "Comparison of Static Logic Styles for Low-voltage Digital Design". In *The 8th IEEE International Conference on Electronics, Circuits and Systems*, volume 3, pages 1421 – 1424, Sept 2001.

[21] Ko. Uming, T. Balsara, and Lee Wai. "Low-power Design Techniques for High-performance CMOS Adders". *IEEE Transactions on Very Large Scale Integration Systems*, 3(2):327 – 333, June 1995.

[22] Ye Yibin, J. Tschanz, S. Narendra, S. Borkar, M. Stan, and De. Vivek. "Comparative Delay, Noise and Energy of High-performance Domino Adders with Stack Node Preconditioning (SNP)". In *2000 Symposium on VLSI Circuits*, pages 188 – 191, June 2000.

[23] M. Sachdev, B. Chatterjee, and A. Keshavarzi. "DFT for Delay Fault Testing of High-Performance Digital Circuits". *IEEE Design and Test of Computers*, 21(3):248 – 258, May 2004.

[24] K. Rawat, T. Darwish, and M. Bayoumi. "A Low Power and Reduced Area Carry Select Adder". In *The 2002 45th Midwest Symposium on Circuits and Systems*, volume 1, pages 467–470, Aug 2002.

[25] A. Tyagi. "A Reduced Area Scheme for Carry-select Adders". In *Proceedings of 1990 IEEE International Conference on Computer Design*, pages 255 – 258, Sept 1990.

[26] Boris Murmann. Digital mos integrated circuits lectures. Available at: http://www.stanford.edu/class/ee313/handouts.html.

[27] Ivan Sutherland, Bob Sproull, and David Harris. *"Logical Effort: Designing Fast CMOS Circuits "*. Morgan Kaufmann Publisher, San Francisco, California, 1999.

[28] David Harris. Logical effort: Designing for speed on the back of an envelope. availabe at: research.sun.com/async/Publications/ KPDisclosed/LogEfftSlides/letalk.pdf.

[29] V.G. Oklobdzija, B.R. Zeydel, H. Dao, S. Mathew, and Ram Krishnamurthy. "Energy-delay Estimation Technique for High-performance Microprocessor VLSI Adders". In *Proceedings of 16th IEEE Symposium on Computer Arithmetic*, pages 272 – 279, June 2003.

[30] Cao Aiqun, N. Sirisantana, Koh Cheng-Kok, and K. Roy. "Synthesis of Selectively Clocked Skewed Logic Circuits". In *Proceedings of International Symposium on Quality Electronic Design*, pages 229 – 234, March 2002.

[31] Behzad Razavi. *"Design Of Analog Cmos Integrated Circuits"*. McGraw-Hill, 2000.

[32] Yuan Taur and Tak H. Ning. *"Fundamentals of Modern VLSI Devices"*. Cambridge University Press, Cambrige, UK, 1998.

[33] Seong-Ook Jung, Ki-Wook Kim, and Sung-Mo Kang. "Low-swing Clock Domino Logic Incorporating Dual Supply and Dual Threshold Voltages". In *Proceedings of 39th Design Automation Conference*, pages 467 – 472, June 2002.

[34] Roy Mader and Ivan Kourtev. "Reduced Dynamic Swing Domino Logic". In *GLSVLSI Symposium*, pages 33–36, April 2003.

[35] A. Rjoub and O. Koufopavlou. "Efficient Drivers, Receivers and Repeaters for Low power CMOS Bus Architectures". In *Proceedings of The 6th IEEE International Conference on Electronics, Circuits and Systems*, volume 2, pages 789 – 794, Sept 1999.

[36] A. Rjoub, O. Koufopavlou, and S. Nikolaidis. "Low-power/Low-swing Domino CMOS Logic". In *Proceedings of the 1998 IEEE International Symposium on Circuits and Systems*, volume 2, pages 13 – 16, May 1998.

[37] H. Kawaguchi and T. Sakurai. "A Reduced Clock-swing Flip-flop (RCSFF) for 63Reduction". *IEEE Journal of Solid-State Circuits*, 33(5):807 – 811, May 1998.

[38] Behzad Razavi. *"Principles of Data Conversion System Design"*. John Wiley, 1994.

[39] A. Alvandpour, R.K. Krishnamurthy, K. Soumyanath, and S.Y. Borkar. "A sub-130-nm Conditional Keeper Technique". *IEEE Journal of Solid-State Circuits*, 37(5):633 – 638, May 2002.

[40] Y. Shimazaki, R. Zlatanovici, and B. Nikolic. "A Shared-well Dual-supply-voltage 64-bit ALU". *IEEE Journal of Solid-State Circuits*, 39(3):494 – 500, March 2004.

[41] L.S Nielsen, C. Niessen, J. Sparso, and K. van Berkel. "Low-power Operation Using Self-timed Circuits and Adaptive Scaling of the Supply Voltage". *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2(4):391 – 397, Dec 1994.

[42] Von Kaenel, P. Macken, and M.G.R. Degrauwe. "A Voltage Reduction Technique for Battery-operated Systems". *IEEE Journal of Solid-State Circuits*, 25(5):1136 – 1140, Oct 1990.

[43] Christopher Batten. Adaptive gate-level voltage scaling using dual voltage domino logic. available at: www.mit.edu/ cbatten/work/aglvs-vlsi00.pdf.