Current Sensing Completion Detection for High Speed and Area Efficient Arithmetic

by

Balapradeep Gadamsetti

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science.

Auburn, Alabama
May 09, 2011

Keywords: current sensing completion detection, adders, multipliers

Approved by

Adit D. Singh, Chair, James B. Davis Professor of Electrical and Computer Engineering
Vishwani D. Agrawal, James J. Danaher Professor of Electrical and Computer Engineering
Victor P. Nelson, Professor of Electrical and Computer Engineering

Abstract

Adders and multipliers are the most widely used computational units in integrated circuits. In many compact low power and high speed designs, more complex arithmetic operations such as multiplication are performed through repeated additions. In such applications, providing carry completion signaling in low cost ripple carry adders can allow the control logic to schedule the next addition as soon as an earlier one is complete, thereby achieving the average case, rather than worst case addition delay over a set of computations. Earlier attempts at using current sensing for such carry completion signaling suffered from serious limitations. This thesis presents a new approach for the design of a ripple carry adder with a current sensing capability which observes late settling carry signal nodes in the circuit and indicates when they reach a quiescent state. By this functional circuitry we avoid key problems of more hardware and power. Simulations for new design of ripple carry adder with current sensing completion detection technique show better than 50% speedup, on average, with less than 10% area overhead. The performance improvement with respect to carry look ahead adder which takes 40% more area than our ripple carry adder design is found to be 20%. This could give significant area and power savings in smaller circuit designs.

To demonstrate a potential application of ripple carry adder with completion detection capability, we incorporate our carry ripple adder into a Booth multiplier design and study the performance gain over a traditional ripple carry adder based design. Simulation results show that a 32-bit Booth Multiplier using the new completion signaling circuits can outperform a 32-bit Booth Multiplier with ripple carry adder (RCA) by 20-50%, while requiring less than 2%

additional silicon area. This is comparable to the gains from the best carry look ahead adder designs at a fraction of the area overhead costs. The new approach eliminates the limitations suffered from the previous current sensing techniques. The overall performance of is improved with a minimal area overhead.

Acknowledgements

At the outset I would like to thank my major advisor Dr. Adit Singh for his immense support and guidance in completing this thesis. He has been a constant source of inspiration and encouragement. I am highly grateful to him for his suggestions and cooperation right from the start of this thesis work. His guidance helped me in all the time of research and my studies at Auburn University.

I express my gratitude to my thesis committee members, Dr. Vishwani Agrawal and Dr. Victor Nelson, for being part of my graduate committee and co-advising my thesis. Their lectures as part of my graduate coursework helped me in learning the basic knowledge required for my thesis work.

I want to express my sincere thanks to my senior Nitin Yogi and my friend Rakshit Venkatesh for helping me academically and giving me valuable suggestions throughout my graduate studies at Auburn University.

Finally I owe my deepest gratitude to my parents for their moral support during all the times. They stood by me unconditionally motivating me in every aspect of my life and education.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

CSCD	Current Sensing Completion Detection
RCA	Ripple Carry Adder
CLA	Carry Look Ahead Adder
BM	Booth Multiplier
ALU	Arithmetic Logic Unit
PFA	Partial Full Adder
CDG	Clock Delay Generator
PDA	Personal Digital Assistants
DSP	Digital Signal Processors
IC	Integrated Circuit

Chapter 1

Introduction

In recent years there has been a rise in use of hand held devices such as Media Players, Smart Phones, Digital Cameras and Personal Digital Assistants (PDAs). The features on these mobile devices will require physically smaller designs with efficient area and power management. The future of digital design holds ever larger and more complex digital circuits than have been achieved so far. Many functions on these devices are expected to be specific to a target application. A generic Integrated Circuit (IC) with versatile functionality may turn out to be too expensive for a customized application which can do with fewer functions. In practice, many available functions may not be required in the application; these functions consume a significant portion of the component chip area and cost. In such situations the generic IC can be replaced with a smaller circuit with fewer functions matched to the requirements of the customized application.

Mobile systems incorporate Digital Signal Processors (DSPs). A DSP is a specialized processor with an optimized architecture for the fast computational needs of digital signal processing. Some of the most common functions performed in the signal processing are signal filtering, convolution and Fourier transforms. In mathematical terms, these functions perform a series of dot products, multiply and accumulate (MAC), additions and multiplications. General-purpose microprocessors can also execute DSP algorithms successfully, but are generally not suitable for use in mobile devices such as mobile phones and PDAs because of power and space

constraints. A customized digital signal processor, however, can provide a lower-cost solution, with better performance and lower power consumption.

1.1 Background

As performance and battery life requirements are becoming more crucial in small and lightweight systems like laptops and handheld devices, power consumption is becoming a critical problem. For example, processors and microcontrollers for these small portable systems should provide higher system performance while keeping power consumption within specified limits. The performance of these chips is mainly dependent on the Arithmetic Logic Units (ALUs) used in them to perform complex mathematical operations such as addition, subtraction, division and multiplication. To save silicon area and power, some chips do not include dedicated arithmetic circuits for multiplication and other complex computations which are considered to be very area intensive and power hungry; multiplication is performed through repeated addition. In such scenarios, there is a critical need for fast and power efficient adder circuits which are almost constantly active during computation, and can have a major impact on overall system performance.

Current trends in the CMOS VLSI design methodologies show steady scaling of feature sizes to meet the speed and performance requirements for complex applications in the areas of communication, aerospace, defense and consumer electronics. Extensive scaling of CMOS process has given rise to many problems such as leakage current, short circuit current, large process variations, etc. It is proving to be difficult to achieve higher performance with transistor channel length scaled to few nanometers. These problems have to be addressed while designing circuits for low power and high speed applications. Also, with CMOS technology approaching

its limit, new approaches for processes and design architectures are needed. Apart from scaling, there has been much focus on very low power consumption and area efficient design methodologies in developing consumer applications.

1.2 Motivation

Digital circuits are classified into synchronous and asynchronous circuits. Synchronous circuits (and systems) include clocked elements like flip flops, latches and registers. The input of these elements should be stable before the next active edge of the clock arrives. Therefore, the behavior of these circuits can be predicted exactly. One major problem with these systems is the distribution of clock signals which must be in phase without significant skew every place in the chip. This makes designing a complex synchronous circuit operating at high frequency difficult due to timing skew and delays.

Asynchronous circuits (and systems) are not governed by a clock circuit driving a global clock signal, but instead they need only to wait for signals that indicate completion of operations from a previous stage using a handshaking approach. In the handshaking approach, a functional block needs to supply a "done" signal indicating that the results of the functional block are available and valid.

In synchronous digital systems to achieve higher performances the clock should be operated at higher speeds. But clock skew is consuming a significant portion of the clock cycle and is limiting the maximum clock speed of the system. The clock frequency in these circuits also depends on the worst case delay between any two clocked elements. With these problems alternative ways of speeding up the operating frequency are required. One method is to optimize the combinational block between the clocked elements to reduce the worst case delay. Another

method is to use techniques for detecting the operation of the combinational block and generate a signal that indicates when the combinational block or its part has completed computation. This type of completion signaling can also be used in self-timed asynchronous designs, where the data between logic blocks are transferred asynchronously utilizing handshake circuits for signaling completion rather than clocks.

Some methods of signaling computation completion proposed in the literature are Activity-Monitoring Completion Detection (AMCD) [23, 24], Current-Sensing Completion Detection (CSCD) [2, 22], and Transition-Monitoring Completion Detection (TMCD) [23, 24]. In Chapter 2 of this thesis AMCD, CSCD, and TMCD methods are discussed from the design point of view. Some benefits and drawbacks of activity monitoring methods are presented. The methods reviewed in Chapter 2 suffer from serious limitations in low power and low voltage applications so in Chapter 3 we propose a new method for completion detection which addresses the limitations suffered by earlier methods and also present a faster current sensor design for synchronous designs.

As discussed earlier, the performance of the digital circuits depend on the computation capabilities of ALUs which they incorporate. The major blocks in these ALUs are adder and multiplier units; multipliers also incorporate adders. Various designs of adder circuits have been proposed earlier which are efficient in area or delay. These adders are discussed in section 1.2.1, which gives a detailed analysis with respect to area and delay. Our idea here is to select the adder design which will suit the need for smaller circuit designs for mobile applications. Also we will try to incorporate additional capabilities of completion detection techniques to enhance the performance of the overall design.

1.2.1 Background on Adder Circuits

Adder circuits are the considered important system blocks for arithmetic logic units used in microprocessors and microcontrollers which perform complex computations in applications. To achieve better performance different types of adder circuits have been designed. Their usages in different applications depend on the following three factors:

1. Delay
2. Area
3. Power consumption

It is hard to design adder circuits which take all these factors into consideration and deliver higher system performance. This made researchers look for alternative ways to design circuits in an optimized and customized way to achieve performance by speeding up operations in complex applications. It is well known that faster computation capability will result in a higher system performance. Often computation capability is mainly dependent on adder circuits used in the arithmetic logic units of the processor.

Some basic adder circuits in use are listed below. Each of these adders has his own advantages and disadvantages. These are:

a) Ripple Carry Adder
b) Carry Look Ahead Adder
c) Carry Skip Adder
d) Carry Select Adder

This section presents a detailed discussion of each adder circuit with respect to delay and area and also discusses their limitations in practical use.

a) Ripple Carry Adder

The Ripple Carry Adder, being the simplest one, uses the least hardware circuitry but suffers from the worst case carry propagation delay. Figure 1 shows a 4-bit ripple carry adder with worst case delay path from C_0 to C_4. A Higher impact on performance is seen for 32-bit or 64-bit adder circuits where the worst case carry propagates from the LSB to MSB block. The delay of the ripple carry adder increases linearly with the number of bits with a worst case delay of O(n) [13]. The RCA has a very compact design area given by O(n) [13]. This worst case delay makes it slow when large bit sizes are used. The advantage for ripple carry adder is that it uses much less hardware circuitry when compared to all other traditional adder circuits in use.

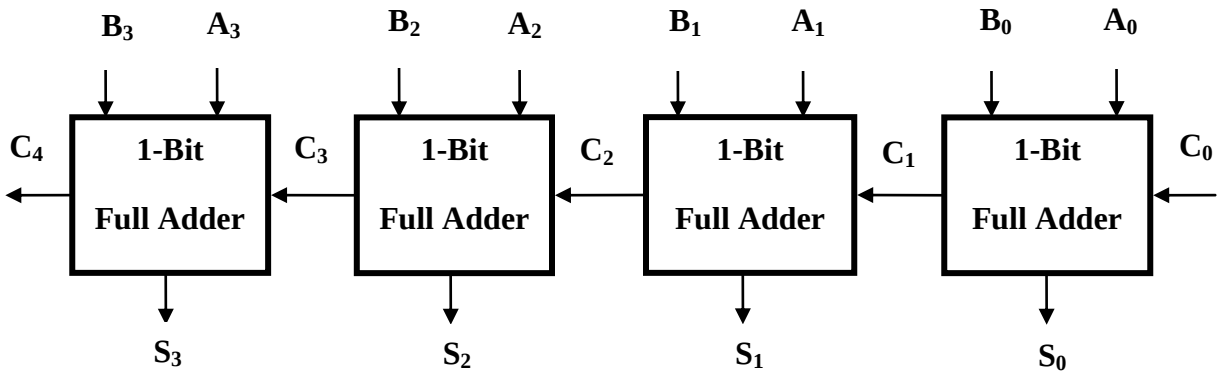


Figure 1: Ripple Carry Adder

b) Carry Look Ahead Adder

The Carry Look Ahead Adder has lower delay but requires much more complex circuitry in achieving its performance. The complex circuitry is added to find in advance the generation and propagation of carry bits. The carry at i^{th} bit position is given by

$$C_i = G_i + G_{i-2}\, P_{i-1} + G_{i-3}\, P_{i-2}\, P_{i-1} + \ldots\ldots + G_0\, P_1\, P_2 \ldots\ldots P_{i-1} + C_0\, P_1\, P_2 \ldots\ldots P_{i-1}$$

The size and complexity for a large adder with the carry signals generated using this equation is not practical. So the carry generation and propagation bits are computed by making groups of carry blocks (usually four bits) as shown in Figure 2. Such a block generates a group carry which give the carry bit to the next block, giving time to the sum units to perform their calculation. This block also calculates generation and propagation information and provides it to next higher level block in the hierarchy.

The added circuitry contributes to more power consumption. Also large fan out within each block circuitry contributes to additional delay in the adder circuit. The delay of a carry look ahead is given by O(log(n)) [1] but will require much larger area given by O(nlog(n) [1].

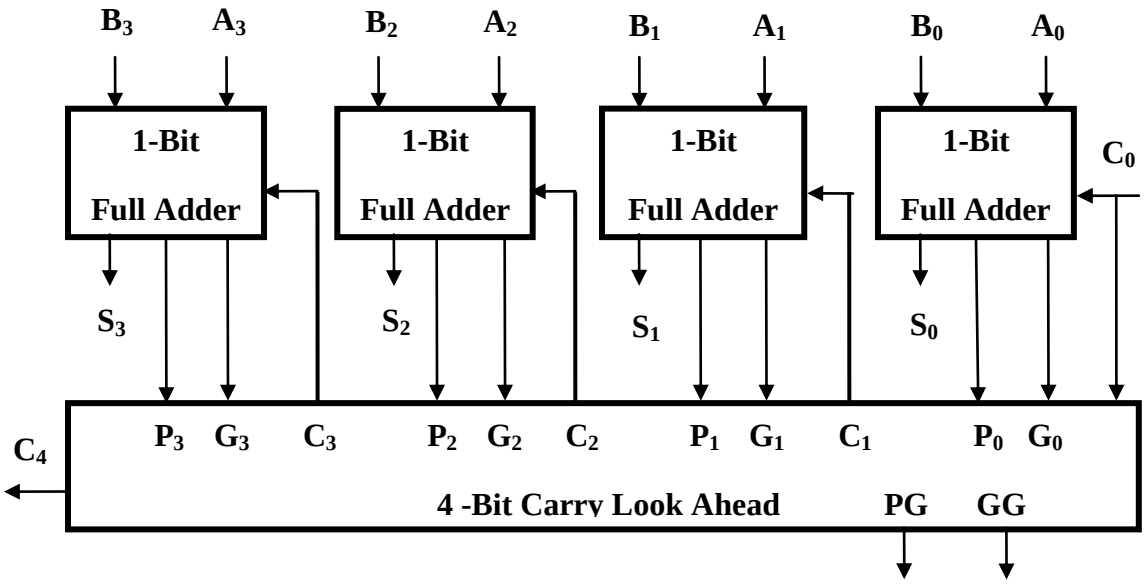


Figure 2: Carry Look Ahead Adder

c) Carry Skip Adder

The Carry Skip Adder reduces the carry-propagation time by skipping over groups of consecutive adder stages. The carry skip adder is usually comparable in speed to the carry look-

ahead adders, but it requires less chip area and consumes less power compared to carry look ahead adders. To accelerate the carry propagation it adds additional group carry bypass paths to its ripple path and the carries can bypass the ripple path when the group propagate signal is high.

Figure 3 shows the basic structure of an N-bits carry-skip adder with k-bits per group. Each block is composed of ripple carry adder blocks of fixed size and carry skip chain. Each block in the carry skip adder generates a propagation bit, P_i, when $P_i = 1$, the carry in of the i^{th} block is passed to the $(i+1)^{th}$ block without waiting for the input carry to ripple. The group propagate function is given by

$$P_i = (A_{k-1} \wedge B_{k-1}) \cdot (A_{k-2} \wedge B_{k-2}) \ \ldots\ldots\ldots \cdot (A_1 \wedge B_1) \cdot (A_0 \wedge B_0)$$

This scheme is faster than a Ripple Carry adder with a small overhead in adding bypass circuitry to decrease the carry propagation delay. However, the delay is linearly dependent on the number of bits N [7].

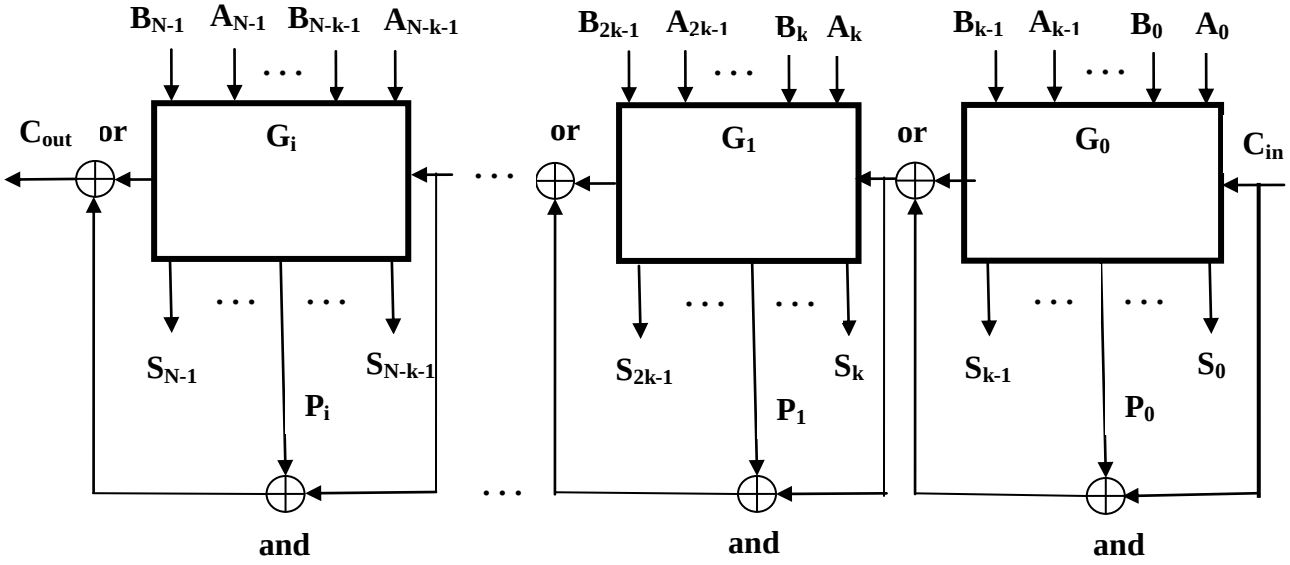


Figure 3: Carry Skip Adder

d) Carry Select Adder

The Carry Select Adder consists of two ripple carry adders and one multiplexer. The two adders are used to calculate the addition twice; one addition is computed assuming carry input '1' and the other as '0'. This adder implementation does not wait for propagated carry signal; rather it first generates sum and carry-out pairs by using both possibilities of carry-in signal at each bit position. The correct output is then selected upon the arrival of carry-in signal. The typical delay of this design is given by O(√n) [7] but at the cost of much larger hardware circuitry which is twice that of ripple carry adder as shown in Figure 4 for a 16-bit carry select adder. This design also contributes to much larger power consumption due to its complex circuitry.

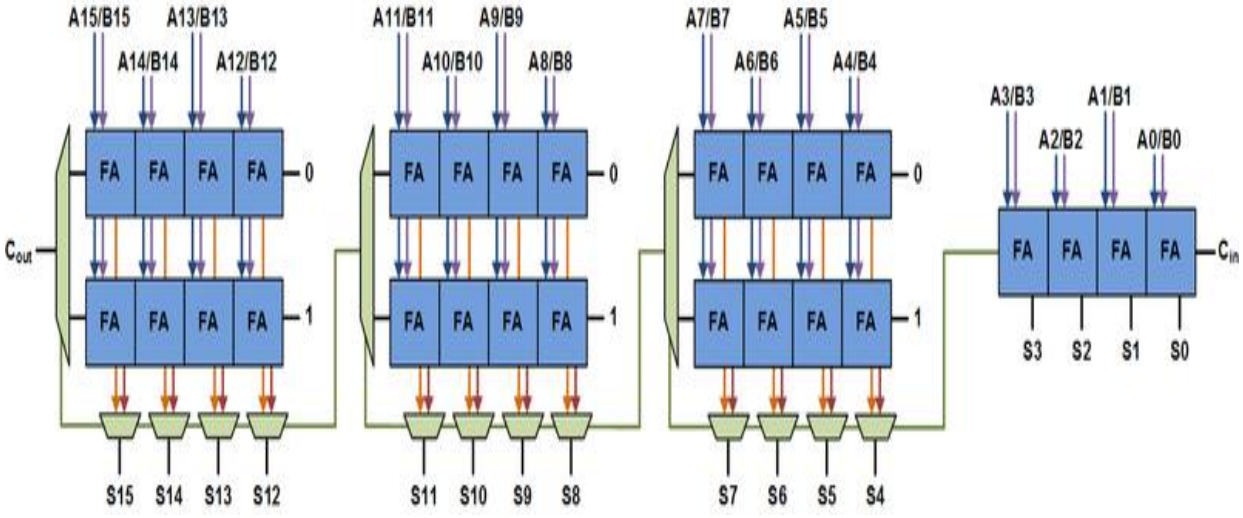


Figure 4: Carry Select Adder

1.2.2 Ripple Carry Adder and Carry Look Ahead Adder Discussion

In section 1.2.1 we have analyzed the different available adder circuits but the most suitable adder circuits for small customized applications is the ripple carry adder. The idea in this thesis here is to modify the ripple carry adder by adding additional capabilities to suit the performance and area needs. This modification should result in performance comparable to the other fast adder circuits without suffering from significant additional overhead issues. Although the classical Ripple Carry Adder has the most compact design (O(n) area) [13] of all adders, it is slow due to its long worst case carry-chain length. But this worst case of carry propagation is seen only in very few cases in practice. So designing a circuit that meets this worst case propagation delay will have impact on the system's performance in most of the other cases where the result is available earlier. The idea here is to design a self timed circuit which performs operations on a much faster rate taking into consideration the worst case performance of the adder circuit.

Our proposed idea is to monitor the circuit activity only on the specific selected nodes of the circuit which are considered to be critical for the overall circuit delay. This monitoring is based on current sensing completion detection which relies on the basics of CMOS operation which states that during any active phase of a CMOS circuit the current is drawn from the power supply. This current is also known as switching current. Based on this fact, we can deduce that a circuit has completed all its operations when the switching current ceases or falls below a certain reference threshold value. One point to be considered here is that the current sensing circuitry adds some additional overhead to the circuit design with respect to the area and delay. The challenge is to keep this as small as possible in designing the sensing circuitry so as to have a minimal impact on the overall circuit performance.

Carry Look-ahead Adders are significantly faster than their counterpart Ripple Carry Adders but require much larger area (O(nlog(n)) [1]. CLAs speed up the computation of the addition operation by using complex circuitry for calculating the carry generation and propagation information at each bit position in advance. This helps in speeding the addition operation but at the cost of much larger circuitry. This larger circuitry also results in substantially higher power dissipation. In contrast, for ripple carry adder to determine the carry for a specific bit position the carry input at the previous bit position must be calculated, this adds up to much larger delay in the calculation. This implementation of Ripple Carry Adder uses less circuitry when compared to traditional Carry Look Ahead Adder.

While timing in designs using conventional RCA designs must allow for the worst case carry ripple delay for every addition, for many input cases the correct result is in fact available a lot earlier. The incorporation of a completion signaling mechanism into a RCA offers a way for improving the "worst case" RCA performance, which is given by O(n) [13] to "average case" RCA performance O(log(n)) [13] over a set of repeated additions. This can provide a mechanism for the RCA to signal to the higher level circuitry controlling it that it has completed the current operation and to be ready to schedule the next operation. Thus, for example, if 32 repeated additions are to be performed to multiply two 32 bit numbers, using completion signaling to initialize the next addition (before waiting out the worst case delay) can cut down the total multiplication time from 32 worst case addition delays, to 32 "average" case delays. This can achieve performance comparable to that attainable from much more expensive carry look ahead adders. These benefits also hold for more typically used Booth's algorithm based multiplication implementations that reduce the number of additions needed by skipping additions for strings of 0's and strings of 1's in the multiplier input.

1.3 Problem Statement

While the earlier work has established that sensing the current from the power supply of a CMOS block offers a valid means for generating a completion signal, introducing current sensors in the functional power supply has not proven practical. These current sensors suffered from significant issues. In this thesis we try to address the problems encountered in earlier designs. The problems addressed here are

1) Development of new approach for current sensing completion detection.
2) Design of current sensors for Synchronous and Asynchronous designs.
3) Design of adder and multiplier circuits with completion signaling capability.

1.4 Contribution of This Thesis

In this thesis we present a novel approach where key individual (late switching) circuit nodes are probed (through sensor lines) and monitored to observe the absence of further transitions. These monitors again detect a quiescent stable state in the probed signals by sensing the current drawn from the power supply, but in this case only the supply current drawn by the sensor circuits need be observed. The overall switching current in the functional block does not need to be monitored. This avoids the key problems associated with the classical Current-Sensing Completion Detection (CSCD) approach.

This thesis presents the analysis between a traditional ripple carry adder with carry completion detection and carry look ahead adder. A classical RCA design uses a string of full-adders, consisting of 2 gate delays each, that are interconnected with the (ripple) carry signals. In an experiment running 100000 random input vectors through such a 32 bit ripple carry adder, the

average delay is observed to be 12-14 gate delays associated with an average 6-7 bit longest carry chain length. (Other equal size or shorter carry chains can also be simultaneously active for the same computation.) Compared to the worst case 32-bit carry chain length, which is 64 gate delays, the average case delay reflects a dramatic 5-6X performance improvement. This suggests that a RCA with computation completion signaling can potentially incorporate the low power and area of RCAs with average performance comparable to that of CLAs. However, there is an additional overhead associated with the circuitry producing the completion signal; the challenge is to keep this as low as possible.

1.5 Organization of Thesis

The thesis is organized as follows. In Chapter 2, we discuss a general background of the earlier techniques for current sensing based completion detection, their pros and cons, and their use in practical applications. Also we discuss their limitations with respect to current technology. In Chapter 3, a novel approach for current sensing completion detection is presented and its operation is discussed. Chapter 4 presents a detailed overview of the performance of Ripple Carry Adder and Carry Look Ahead adders. New design of ripple carry adder with current sensing completion detection is proposed and discussed in Chapter 5. In Chapter 6, a Booth multiplier with new CSCD circuitry is presented and its performance improvement with traditional Booth multiplier is evaluated. Conclusions and future work are discussed in Chapter 7.

Chapter 2

Prior Work on CSCD Design

Researchers in the past have developed many methods for completion detection. This chapter discusses some of these techniques and past work on current sensing methods, their limitations and how these design methods would fail in the current scenarios where technology is being scaled to nanometers. This part also discusses several possible designs (in standard low-power CMOS process) that can be used as the current sensing elements in CSCD circuitry.

2.1 Completion Detection Methods

Transitions in a CMOS digital circuit can be monitored by two methods. The first method is based on the current sensing. A CMOS circuit draws current from the power supply when both N and P transistor are ON. When the current ceases indicates that all the transitions in the circuit are completed. Monitoring the current drawn from the power supply provides a means for signaling the completion of the CMOS block. The second method is voltage monitoring. During transitions the internal node capacitances are charged or discharged. Completion signaling is done by monitoring the voltage transitions on internal node capacitances. In literature various methods are proposed for current and voltage monitoring. The methods used for voltage monitoring are Activity Monitoring Completion Detection (AMCD) and Transition Monitoring Completion Detection (TMCD) [23, 24]. In AMCD, Activity-Monitoring (AM) circuitry is used to observe the transitions on the internal nodes of the CMOS block. The TMCD method is similar to AMCD but offers additional functionality to observe activity from glitches. The issues

with these methods are that they add additional capacitances on the monitoring nodes which introduce additional delays in the circuit. Also extra routing lines are needed for the activity monitoring circuitry to observe transitions on the internal node. With respect to performance and area, AMCD method gives better performance than TMCD method while consuming less area and power. The approach used for current monitoring is Current Sensing Completion Detection (CSCD) method where the current drawn from the power supply by the CMOS block in transition is monitored. The current is collected in a current sensor and compared with a pre-selected threshold. When the monitored current is above the threshold value indicates the CMOS block is in operation and when it drops below the threshold completion is signaled.

2.2 Classical CSCD Design

In theory, CMOS designs offer a simple way to determine when all switching activity has ceased and the circuit has reached steady state: there should be only minimal leakage current drawn from the power supply. A number of previous researchers have investigated ways to implement Current-Sensing Completion Detection (CSCD) [1-4] using circuits that monitor the power supply of circuit blocks as shown in Figure 5 and indicate when the outputs stabilize. Unfortunately, monitoring the power supply current necessarily involves introducing additional circuitry between the logic gates and the power supply. This can seriously degrade the supply voltage available to the logic, particularly in low-power, low-voltage designs, and thereby significantly impact circuit performance. Furthermore, it remains very challenging to design current monitors with a sufficient dynamic range to provide for the peak switching current demands of a functional block of a hundred or more gates, while still being sensitive enough to reliably indicate when the last gate completes its switching transition. These difficulties have

prevented the practical implementation of current sensing based completion detection. Classical CMOS circuits have the inherent property of only consuming power during transitions when both pull up and pull down transistors are on and circuit capacitances have to be charged and discharged. Only a small leakage current is drawn from the power supply in steady state. It is therefore straight forward to see how monitoring current consumption provides real-time information on circuit switching.

The implementation of a CSCD design has a few requirements.

1. First, the circuit that is being monitored needs to be capable of being an asynchronous, stand alone, functional block that can take advantage of the completion signaling.
2. Second, the circuit being monitored needs to be combinational logic that contains a subset of signals in which at least one will be in the process of transitioning at any given moment if the operation is still in progress.

Assuming these requirements are satisfied, any combinational block can, in principal, be modified to use CSCD circuitry. Previous CSCD designs [1-4], including those implemented for ripple carry adders have all employed circuit architecture conceptually similar to the one shown in Figure 5. Unfortunately, such designs have some significant drawbacks. Perhaps most important is the voltage degradation across the current sensor circuits, which can severely impact the switching delay of the CMOS logic. Furthermore, since typically a single pair of current sensors is used to monitor supply current for the complete functional block (e. g. a 32-bit ripple carry adder), the current sensors must be designed to allow a large peak current to pass through, while being able to accurately discriminate about a threshold current due to a single active gate. Managing such a large dynamic range of supply currents is the classical challenge faced by designers of on chip current sensors [1-4]. The only real solution is to minimize the number of

gates in each block being monitored, but that greatly increases the number of current sensors needed, and results in unacceptable area overhead.

Earlier CSCD designs which directly monitor the circuit power supply current to create a completion signal are shown in Figure 5. Recall that CMOS circuits only consume power during switching and therefore when the switching stops it can be assumed the calculation is completed. In Figure 5, the power supply current sensors are to detect when the current level has been reduced below a threshold. At this point a "done" signal is asserted to signify to the higher level control circuitry that the operation has completed.

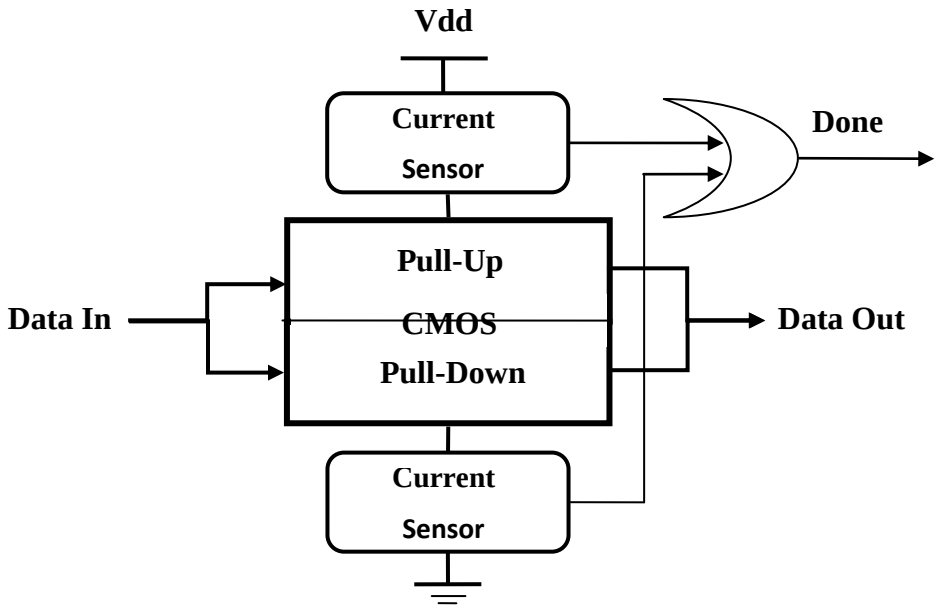


Figure 5: Current-Sensing Completion Detection (CSCD) circuitry for a CMOS Block

2.3 Current Sensor Design Approaches

The two main designs of current sensors for completion detection are resistance based and current mirror based.

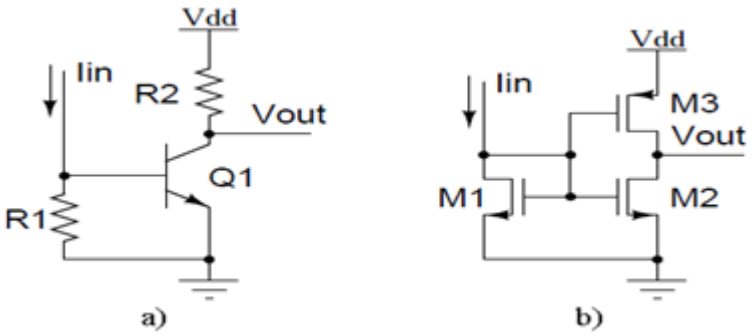


Figure 6: Basic Current Sensors

The first method is resistance based shown in Figure 6a the detected current flows through resistance R1. The value of the resistance R1 should be chosen such that the voltage between base-emitter junction of the BJT is above 0.7 – 0.8 V. Here a large value resistance is required as the sensed current has low values. This resistance based method has limitations when used in low-voltage applications where supply voltage is as low as 1V. The first being the unacceptable voltage drop across the base emitter junction and second the large value of the resistance R1 which limits the current in the functional path of the logic block. Also future trends concentrating on CMOS designs, inclusion of bipolar transistors is not suggested.

The second method is the current mirror based shown in Figure 6b. Transistors M1 and M2 are connected in a push-pull inverter configuration. These transistors operate in saturation and act as current mirror. The sensed current Iin is mirrored and converted into voltage Vout. The disadvantage with this design is large values for W/L ratios are needed since current values are

low. Also, overall delay of the current sensor increases due to high gate capacitances, thus making faster transitions invisible to the sensor.

2.4 Applications Using Current Sensor Completion Detection

The basic configuration for incorporating CSCD in CMOS block is shown in Figure 7 which incorporates two current sensors. The first one between the logic block and VCC, the second between the GND and the logic block. The current sensors are used to detect a current above a predefined threshold and generate a signal as long as the transient current flow is above this threshold.

Ideally two current sensors are needed in both the GND and VCC supply paths. When the circuit drives large loads in those situations the current mostly flows either through the NMOS or PMOS network discharging or charging the output load capacitances. This configuration is expected to work independent of the load variations on the output of the logic function.

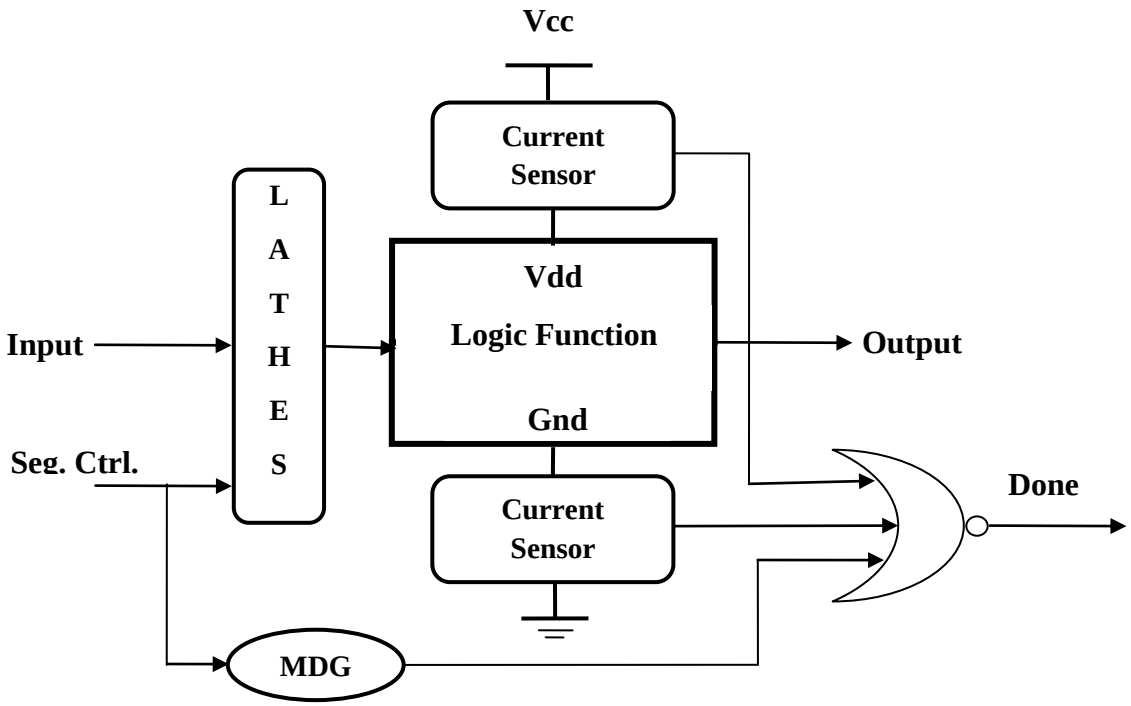


Figure 7: Basic CSCD configuration

The CSCD design also incorporates few other components. These are:

1. Latches: The input latches are used to provide inputs to the logic function. These latches also remove any the differences in the arrival time of the input signals at the logic block.
2. MDG: The Minimum Delay Generator (MDG) is incorporated to guarantee that minimum time has elapsed for the logic function to compute correct outputs. The input to the MDG is from the sequential control which is the moment when the inputs are fed to the logic function. This time is considered as start time for the computation.
3. NOR Gate: The NOR gate is provided inputs from for the current sensors and minimum-delay generator. When the all the inputs from them are active low the completion signal ‘done’ is signaled.

The two important factors which could affect the operation or performance of the logic function being monitored are:

1. Voltage drop across the current sensor which limits the supply voltage to the logic.
2. Resistance offered by the current sensor which limits the current flow to the logic function which slows down the switching speed of the logic function.

CSCD as presented here suffer from serious limitations of static power consumption and voltage drop in the power supply voltage provide to the logic function thus making them unsuitable for low-power and low-voltage applications. The next section proposes a novel idea which addresses the issues presented here.

Chapter 3

Proposed CSCD Design

Previous methods using the technique of current sensing for completion detection introduced additional overhead in delays, degrading the performance of the overall circuit. This thesis presents a novel approach for completion detection design that does not monitor the current in the power supply of the functional block. Instead, only a selected set of individual late settling signal nodes in the circuit are observed by connecting simple inverter sensors to them, and monitoring the current drawn by these sensors from the power rail.

3.1 CMOS Inverter

A CMOS inverter is constructed by two complimentary transistors, PMOS and NMOS. It gives an output voltage which is of the opposite logic-level to its input. A CMOS inverter draws current from the power supply when both the transistors are 'ON'. This current is seen when there is transition at the input of the inverter from Logic '1' to Logic '0' and vice versa. Figure 8 represents a simple CMOS inverter and Figure 9 shows the plots of V-I characteristics of the inverter. Observe from the V-I graph that the drain current I_d is generated only when the signal at the input of the inverter makes a transition from High to Low and vice versa. This simple concept can be used to detect the transitions on the late settling nodes in a combinational circuit and determine when they have reached a steady state.

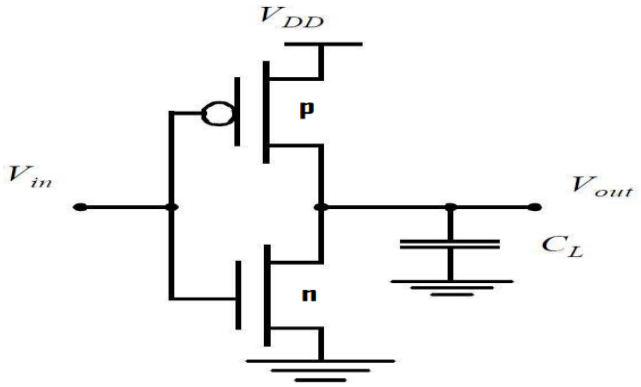


Figure 8 : A Simple CMOS Inverter

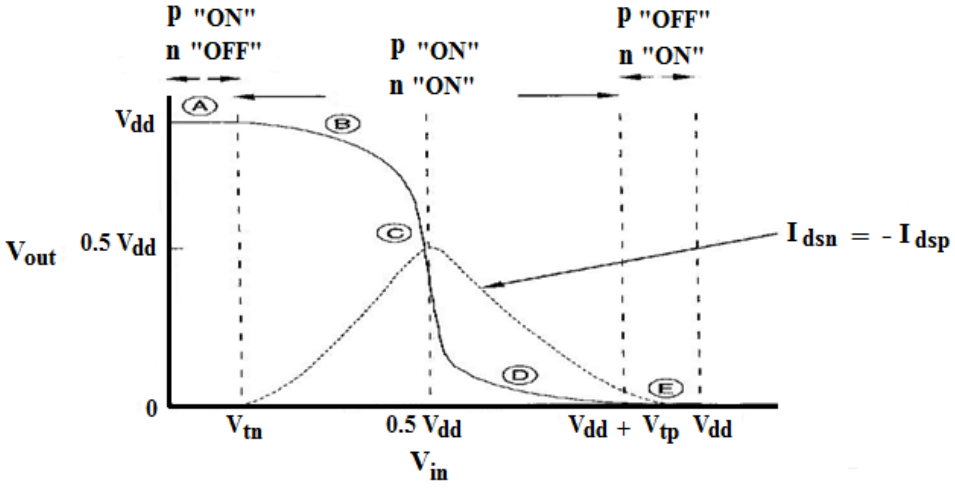


Figure 9: V-I Characteristics of CMOS inverter

3.2 Sense Inverter Design

The proposed method for implementing the current sensor involves using a sense-inverter such as the one shown in Figure 10. CMOS inverters draw current from the power supply as long as their input is midrange between a high and a low voltage, shown in Figure 9, such that both the P and N transistors see a gate source voltage above their respective threshold voltages. The supply current does not flow once the input voltage levels go below the threshold of either transistor types. Thus, monitoring the supply current of these inverters provides a means of determining if the input has stabilized close (within a threshold voltage) to a high or low.

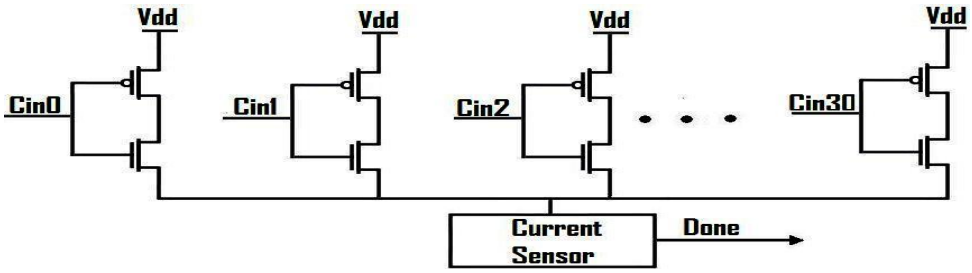


Figure 10: Sense Inverter Implementation

Incorporating a sense-inverter current-sensing circuit implementation into a standard 32-bit RCA involves adding a minimally-sized inverter to each of the 31 carry signals as shown in Figure 10. The carry output of the final stage is neglected in this case. Recall that standard cells in typical cell libraries generally use transistors that are sized up in width by a 5-10X factor for N transistors and 10-20X (or more) for P transistors for performance. Using a minimally-sized inverter in this application ensures that the sense-inverter does not significantly load the carry signals in comparison to the normal load on the lines, minimizing any performance impact.

Observe in Figure 10 that only the current in the sense-inverters is monitored. Furthermore the sense inverters have no load at the output, other than parasitic capacitances associated with the small minimum sized transistors. Thus a single current monitor connected on the ground rail is sufficient to provide a reliable switching completion signal. In the case of large capacitive load on outputs, it is possible that the N transistor turns off while the P transistor is still on and charging the output capacitance for significantly longer. This is the reason for the double sensors, one for each supply rail, in classical CSCD designs. Importantly, the entire current-sensing circuitry is only associated with the sense inverters, which are not in the functional path. The power supply of the functional logic is not monitored. Thus impact on functional performance is minimal. Selecting only a limited number of circuit nodes to be monitored also reduces the peak and dynamic range in the current observed at the current sensor. Even so, observe in Figure 10 that the peak current, when all the inputs simultaneously make transitions, can be 31 times (in case of 32 bit Ripple Carry Adder) the largest current seen in an inverter.

By adding sense invertors the overall circuit switching current is not monitored. In this way, functional performance in only minimally impacted (by the small additional fan out loading due to the sensors), while the completion of switching at the observed nodes can be reliably detected by monitoring the current drawn by the sensors. The proposed new design approach thus overcomes key limitations of earlier approaches discussed in Chapter 2.

3.3 Current Sensor Design

In this section we present two designs of current sensors for synchronous and asynchronous circuits.

3.3.1 Current Sensor for asynchronous designs

Asynchronous designs do not implement a clock. These circuits are self timed; when the activity in the circuits stops the output 'Done' signal is flagged indicating the completion of the operation. Figure 11 shows the design of a current sensor for asynchronous logic circuits. The size of transistor P1 is chosen to be L\W=5\1 to get sufficient bias current to keep both the transistors N1 and N2 in saturation. So when a small current from the sensor flows, the current is mirrored into Iout. The input voltage to the inverter combination of transistors N4, N3 and P3 gets a voltage equal to Vdd minus the voltage drop across drain and source of transistor P2. Transistor N4 is used to make sure that a small voltage swing at the input of the inverter causes a full swing at the output. Previous CSCD designs have also used a Minimal Delay Generator (MDG) that is also adapted in our design. The purpose of the MDG is to ensure that enough time has elapsed for the current-sensing circuit to begin functioning properly. As shown in Figure 12, the output of the MDG should be asserted and the comparator must be de-asserted in order for the "done" signal to be asserted. Figure 12 shows the waveforms for the simulation of a 32 stage carry propagation.

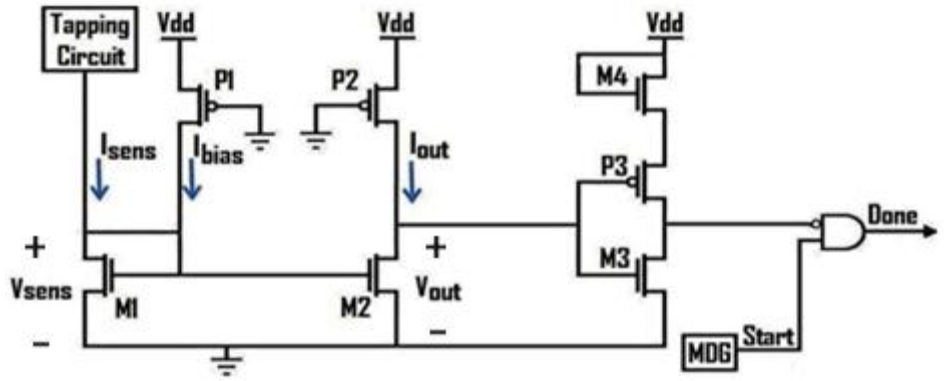


Figure 11: Current Sensor for Asynchronous Designs

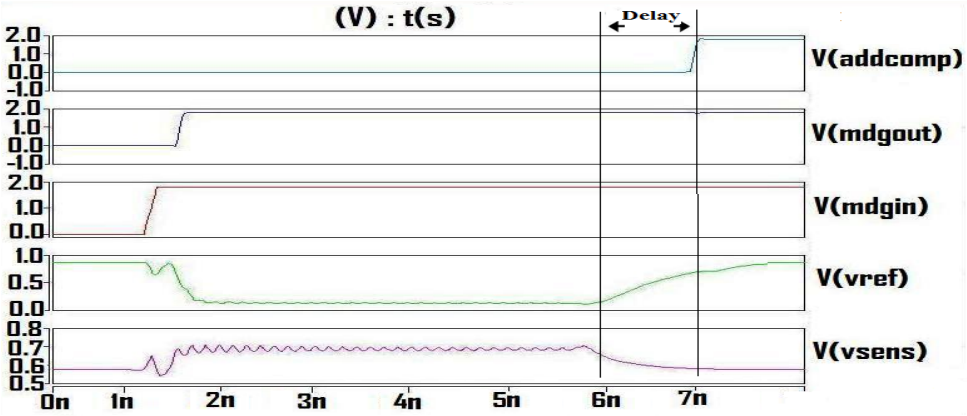


Figure 12: Waveforms for Asynchronous Sensor

It should be noted that the discharge time of Vsens depends on the time constant approximated by the product of the capacitance on Isens line and the channel resistance of transistor M1 in parallel with channel resistance of transistor P1. Also, the charge time of Vout depends on the time constant calculated by the product of the capacitance on Iout line and the channel resistance of transistor M2 in parallel with channel resistance of transistor P2. For correct operation of the current sensor we need a voltage swing of 0-1V on the Vout node. To achieve this, the resistance seen on Vout line should be high. This high value of resistance increases the charging time constant on Vout node increasing the response time of the current

sensor, thus affecting the performance of the circuit. The additional delay added in the current sensor response is found to be 1.2ns but the actual completion is over a lot earlier. This overhead is always added after completion of each operation degrading the overall performance of the circuit. With this degraded performance the asynchronous current sensor cannot be used in practice. The synchronous current sensor presented in Figure 13 has to sense only a differential voltage change when compared to large voltage swing required for asynchronous sensor. After sensing the differential voltage the full voltage swing is provided by the latch operation of inverters connected in regenerative feedback configuration, thus we can see better response by the sensor. The next section presents the design of synchronous current sensor.

3.3.2 Current Sensor for synchronous designs

Synchronous designs have a clock so all the operations within the circuits are synchronized with the master clock. The ‘Done’ signal must be asserted before the active edge of the clock so as to synchronize the next operation. Our current sensor also uses a master clock from which signals to control the current sensor are generated. Referring to Figure 12, the current sensor consists of two inverters, Inv1 and Inv2, connected in a regenerative feedback configuration. The sources of the two PMOS transistors, T5 and T6, of these two inverters are connected to power through another P transistor, T3, which serves as a switch to trigger these inverters. The measured voltages are connected to the sensor through two pass transistors, namely T1 and T2. The gates of T1, T2, T3 and T4 are connected to ‘accum’, ‘precharge’ and ‘eval’ control signals generated from control logic.

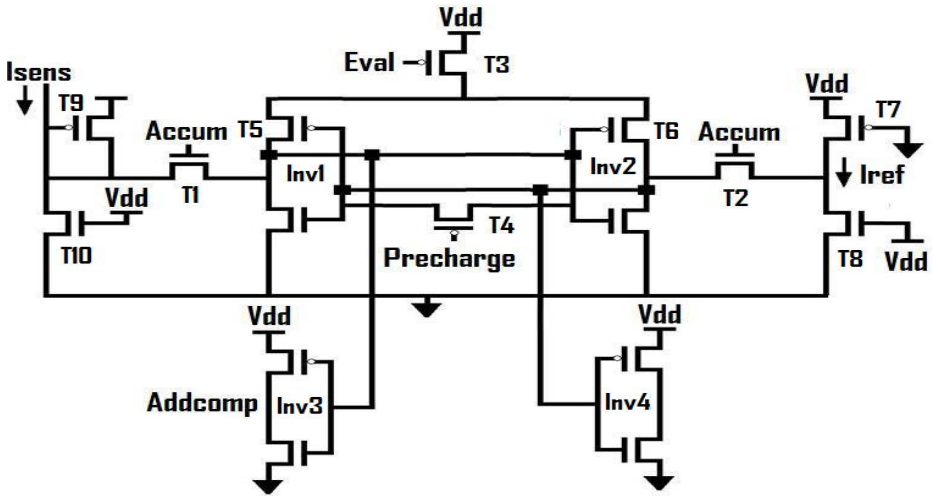


Figure 13: Current Sensor for Synchronous Designs

During the precharge phase, T4 is on which maintains the inverter inputs at same voltage. In the accumulation phase, T1 and T2 are on. This allows the charge from the measured voltages to build up on the inverter inputs, but the output the inverters are not affected as T3 is turned off, which latches the voltages into the inverter inputs. In the evaluation phase, the power to the inverters is turned on. When the inverters are powered up, T3 turned on, the inverter with the higher input voltage dominates, and its output is pulled low. The outputs from both inverters are connected to another set of inverters Inv3 and Inv4, which serve to buffer the sensor output from the rest of the circuit. If the current from the current generator is greater than the reference current, then the node that was connected to the current generator (output of Inv1) is pulled high. This node is connected to one of the output inverters (Inv3) and pulls its output low. This creates the active-low signal that indicates carry propagation. It should be noted that the current sensor is triggered before the evaluation period and the output of the RCA is valid only when the 'addcomp' signal is high, if not, the next cycle is tested for the addition completion.

In practice, the supply current cannot be more than about 20X the detection threshold current. The reduced voltage across the inverters automatically limits the current drawn from the power supply. Note that while this serious degradation in available voltage across the sense-inverters is acceptable in our design, because these sense-gates are not in the functional path, the performance impact of such supply degradation would be unacceptable if the current sensors were in the functional power supply, as in traditional CSCD designs. This is a major reason why past CSCD designs have not been very practical. In fact, design of the current sensor circuits to meet this demanding requirement of handling a very wide range of supply currents without excessively degrading functional performance was the key challenge in earlier designs.

4.3.2.1. Control Logic Block

As shown in Figure 14 the control signals are generated using the clock and a delayed clock signal as inputs. The Clock Delay Generator (CDG) is used to generate the delayed clock. The delay of the CDG is set to the evaluation time of the current sensor. Typically we see this delay as an additional overhead on the overall circuit delay over one addition operation. This overhead can be minimized by designing the current sensor with faster invertors in the regenerative configuration.

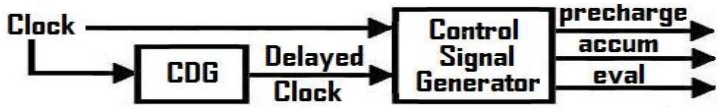


Figure 14: Control Logic Block

4.3.2.2. Control Signals

The Control logic generates three signals, corresponding to the three phases Precharge, Accumulation and Evaluation, of the current sensor operation. The states of transistors T1, T2, T3 and T4 of Figure 13 are shown in Figure 15.

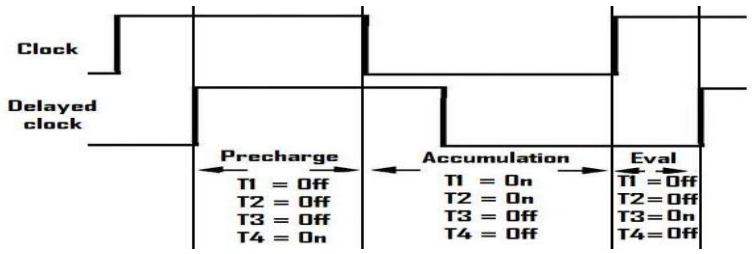


Figure 15: Control Signals

4.3.2.3. Waveforms

Figure 16 shows the voltage waveforms at different nodes of the current sensor circuit for a 32 bit addition, where 32 stage carry propagation is generated. These waveforms show analysis for three evaluation phases.

1. Start: Start indicates the time when the inputs are fed to the circuitry. The inputs for transistors T1, T2, T3 and T4 of Figure 13 are shown in Figure 15.
2. Evaluation Phase-1: Here the sensing voltage, V(sens), is greater than the reference voltage, V(ref). This indicates the operation is not completed yet so V(addcomp) is Active LOW.
3. Evaluation Phase-2: The sensing voltage, V(sens), is still greater than the reference voltage, V(ref). This indicates the carry is still in propagation and V(addcomp) is Active LOW.

4. Evaluation Phase-3: Now the sensing voltage, V(sens), is less than the reference voltage, V(ref), indicating the operation is completed and V(addcomp) is set to Active HIGH.

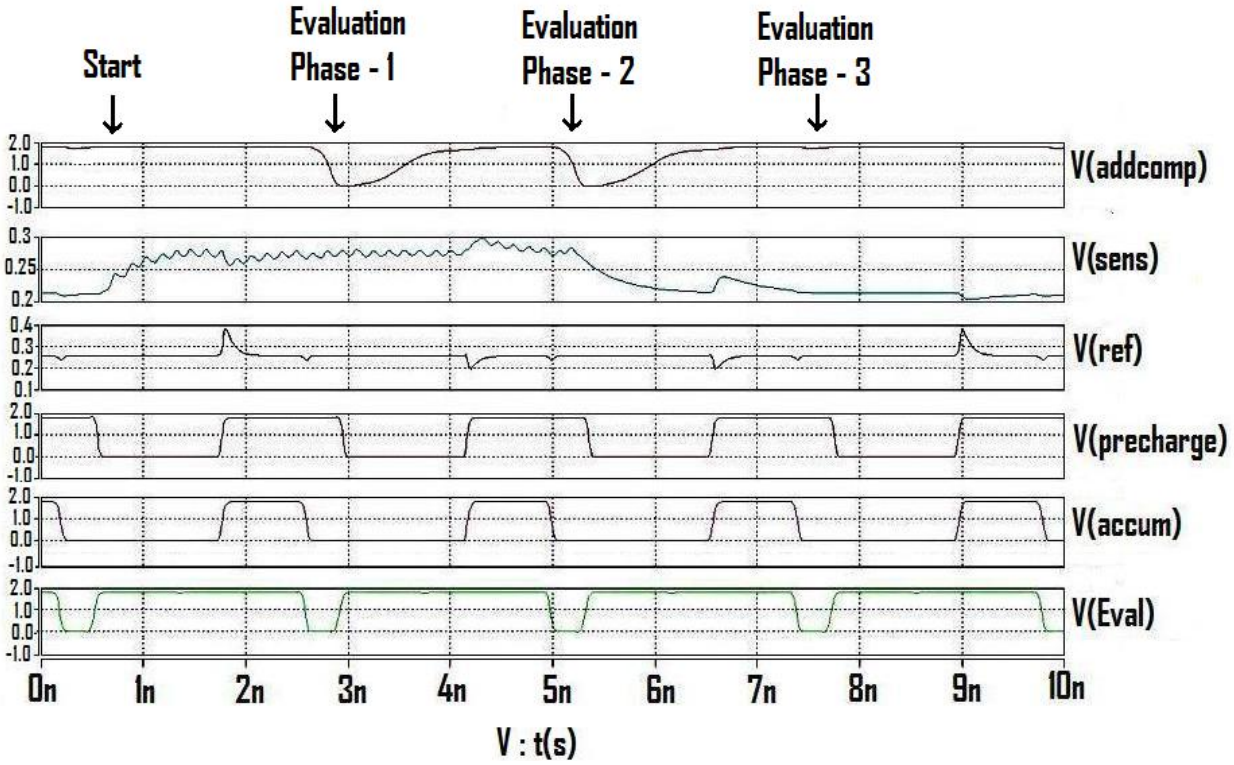


Figure 16: Waveforms for Synchronous sensor

Chapter 4

Ripple Carry Adder and Carry Look Ahead Adder Analysis

In this chapter the performance of a ripple carry adder for delay in the best and worst case scenarios is discussed. Also ripple carry adder performance is compared to a carry look ahead adder with carry propagate and carry generate block designed using grouping of 4 bits.

4.1 Ripple Carry Adder

A Ripple carry adder consists of blocks of 1-bit full adders connected in series with the carry out of one block serving as carry in to the next block. Figure 17 shows the interconnection of a 4-bit ripple carry adder. It can be observed that the critical path for this design is the path from C_0 to C_4.

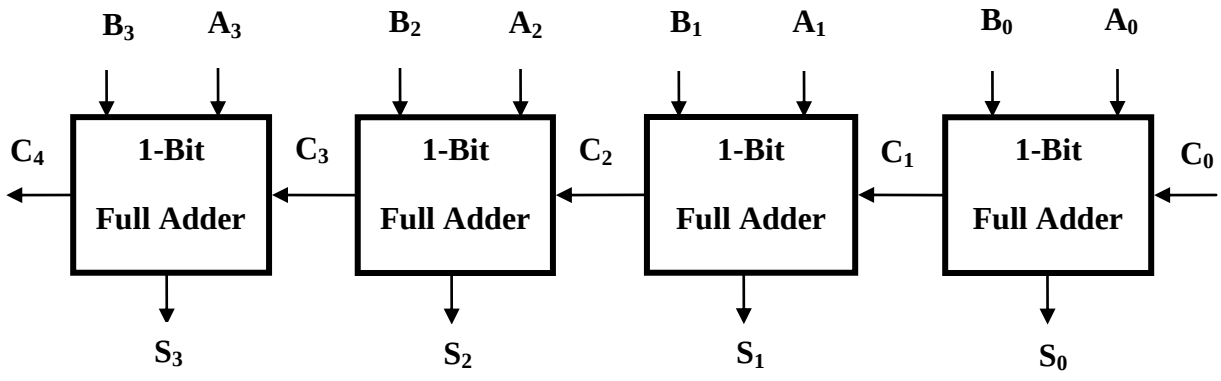


Figure 17: 4-Bit Full Adder Circuit

4.2 Full Adder

A full adder performs additions of three 1-bit numbers, A, B, C_{in}, and gives outputs Sum, and Carry out, C_{out}. The expressions for sum and carry are given by

S = $A \oplus B \oplus Cin$

Cout = $(A \bullet B) + (Cin \bullet (A \oplus B))$

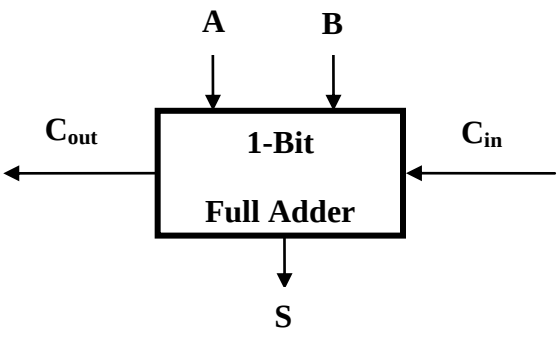


Figure 18 : Full Adder Block

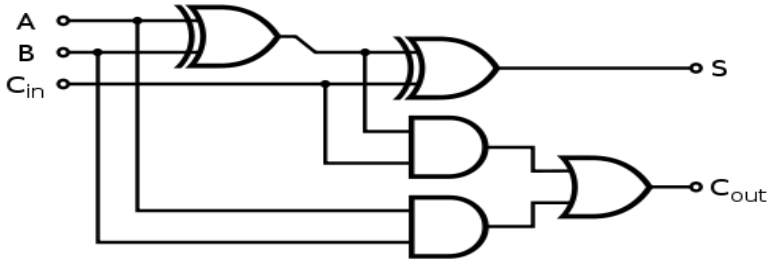


Figure 19: Gate Level Representation of Full Adder Circuit

Figure 19 shows the gate level representation of a full adder circuit with sum and carry outputs. The sum output takes two XOR gate delays and the carry out has delay of two gates; one AND gate and one OR gate.

4.3 Performance Analysis of RCA

Figure 20 shows the gate level representation of 4-bit full adder circuit with sum and carry outputs. The addition of two bits at any stage depends on the carry generated by the addition of the two bits in the previous stage. Thus, the sum of the most significant bit is only available after the carry signal has rippled through the adder from the least significant stage to the most significant stage. The critical path for this design is the carry propagation path from C_0 to C_4 which is 8 gate delays. Extending the design concept for 32- bit ripple carry adder gives us the critical path delay of 64 gates.

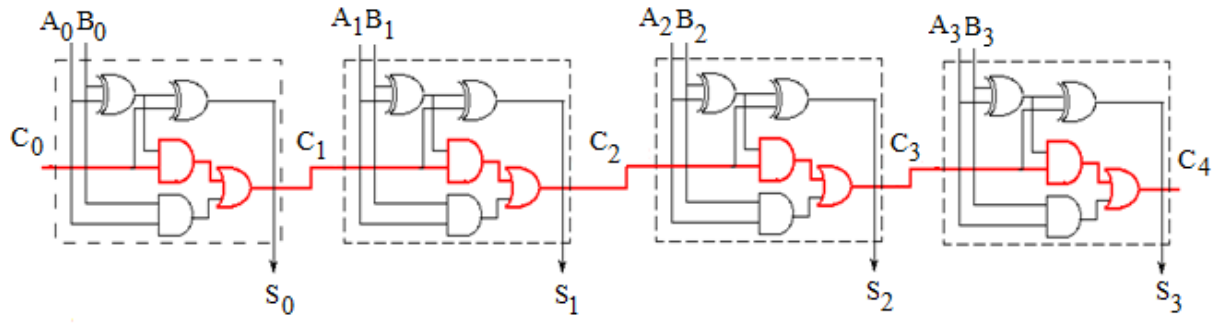
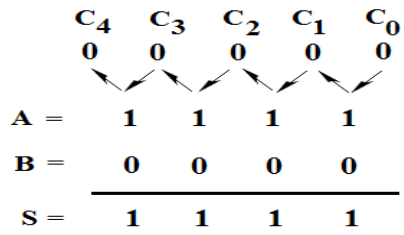


Figure 20: Gate level representation of 4-bit ripple carry adder

4.3.1 Best Case Performance

The best case performance for a ripple carry adder is seen when there is no carry generated at each bit position. In the case of the 4-bit adder in Figure 20 the sum outputs are available just after two xor gate delays. This can be observed with input values A = 1111, B = 0000 and C_0=0, which gives C_1=C_2=C_3=C_4=0 at each bit position. For this case there are no carry bits generated and since no propagation takes place the sum outputs are valid after two gate delays.



Delays:

S_0 = 2 gate delays

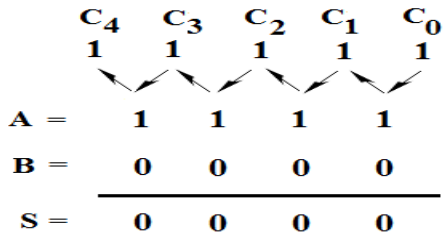
S_1 = 2 gate delays

S_3 = 2 gate delays

S_4 = 2 gate delays

4.3.2 Worst Case Performance

The worst case performance for a ripple carry adder is seen when the input carry, C_{in} is propagated to next stage by each bit position. Because of carry propagation from the least significant position to most significant position the sum bits generation takes variable time delays. In the case of the 4-bit adder in Figure 20, this can be observed with A = 1111, B = 0000 and C_0=1, which gives C_1=C_2=C_3=C_4=1 at each bit position. Since the carry is propagated the delays for the sum outputs is observed to be two to eight gate delays.



Delay:

S_0 = 2 gate delays

S_1 = 4 gate delays

S_3 = 6 gate delays

S_4 = 8 gate delays

4.4 Carry Look Ahead Adder

The carry look ahead adder calculates carry bits simultaneously using complex logic circuitry. This minimizes the worst case delay to calculate the sum bits. Due to large fan out on the gates, the carry generation and propagation bits are calculated by making groups of carry blocks (usually four bits) as shown in Figure 22.

The primary block of the carry look ahead adder is Partial Full Adder (PFA). Each block of PFA takes three bits A, B, C; as inputs and generates three outputs G, P and S. These are

Generate function, G = A · B

Propagate function, P = $A \oplus B$

Sum function, S = $A \oplus B \oplus C$

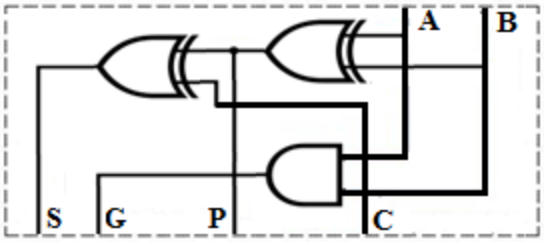


Figure 21: Partial Full Adder

4.5 Performance Analysis for 4-bit CLA

Figure 22 shows the gate level architecture of a 4-bit carry look ahead adder. In order to construct a 4-bit CLA four PFAs are needed to generate the signals. When $P_i = 1$, an incoming carry is propagated to the next bit position from C_i to C_{i+1}. For $P_i = 0$, carry propagation to the bit position is blocked. Regardless of the value of P_i, when $G_i = 1$, the carry output from the current position is '1'.

The carry output has the following logic:

$C_1 = G_0 + P_0C_0$

$C_2 = G_1 + P_1C_1 = G_1 + P_1G_0 + P_1P_0C_0$

$C_3 = G_2 + P_2C_2 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$

$C_4 = G_3 + P_3C_3 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0$

The carry look ahead block takes only two gate delays to generate carry bits from C_1 through C_3. The implementation of C_4 is becomes more complicated when this 4-bit carry look ahead adder is extended to multiples of 4 bits, such as 16 bits and 32 bits. The carry look ahead adder for 4 bits computes carry bits at all the bit positions simultaneously. The longest delay in the 4-bit carry look ahead adder is 4 gate delays, compared with 8 gate delays in the ripple carry adder. The improvement is very modest but at the cost of much additional hardware. From Figure 22 it can be observed that the fan in for generating the C_4 is 4, if we extend the same concept to generate more carry bits simultaneously then the high fan-in for generating the carry bits could contribute to additional adder delays. So in such cases the carry bits are limited to groups of 4 and are extend to next higher level of blocks to generate the carry bits. For a 16 bit carry look ahead adder the higher numbered bits 4 - 7, 8 - 11, and 12 - 15 are grouped together. For this in positions 4, 8, and 12 we would like the carry to be produced as fast as possible

without using excessive fan-in. The estimated worst case delay for a 32-bit CLA is 8 gate delays but high fan-in in the carry generation block could add additional delays in the adder circuit.

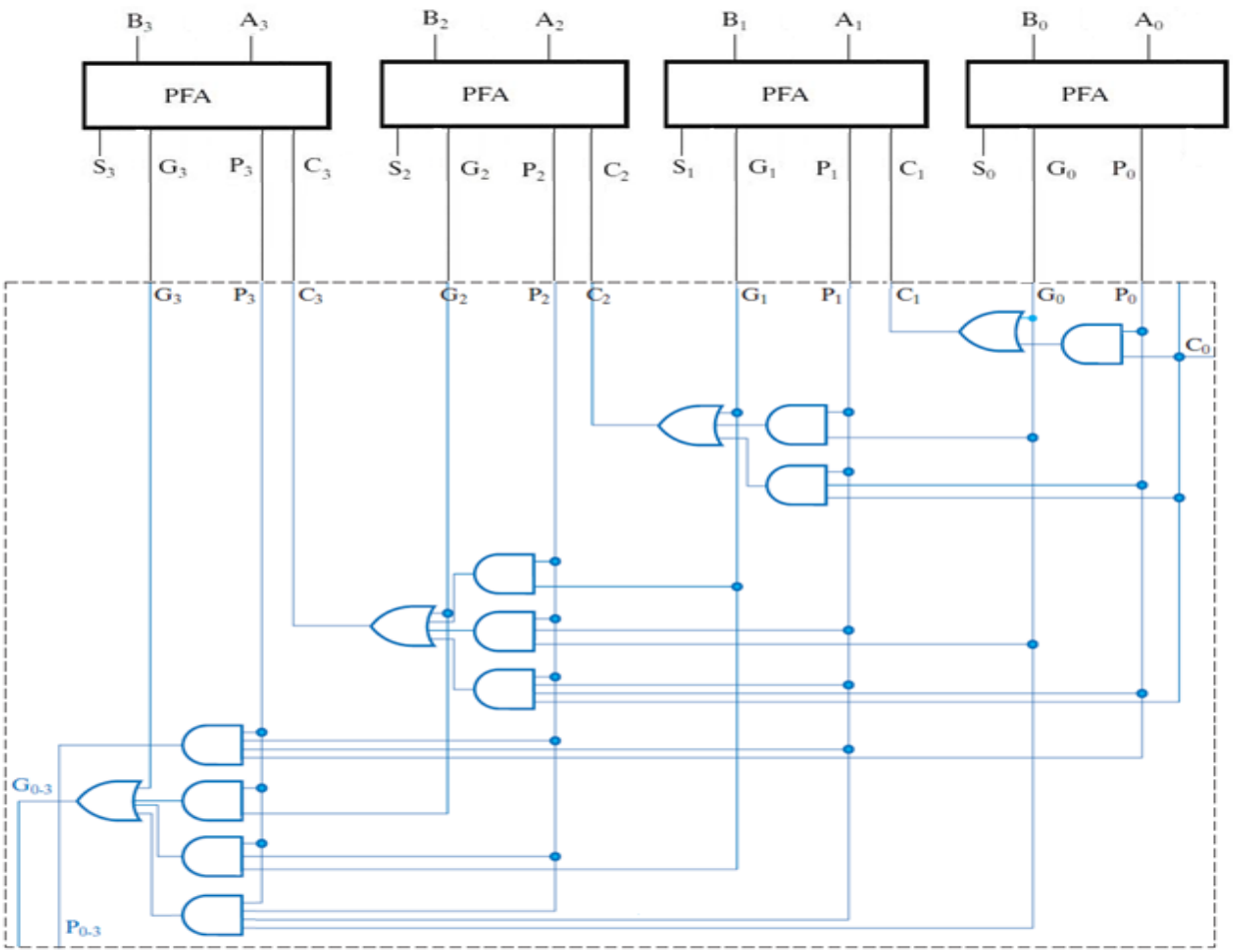


Figure 22: Carry Look Ahead Adder

Chapter 5

Proposed Ripple Carry Adder Design with Completion Signaling

5.1 Background

A classical ripple carry adder design uses a string of full-adders, consisting of 2 gate delays each, that are interconnected with the (ripple) carry signals. In 1946 Burks, Goldstine, and von Neumann published a proof that the expected maximum length of carry propagation in the addition of two binary n digits each does not exceed $\log_2 n$ [26]. In an experiment running 100000 random input vectors through such a 32 bit ripple carry adder, the average delay is observed to be 12-14 gate delays, associated with an average 6-7 bit longest carry chain length as shown in Figure 23. (Other equal size or shorter carry chains can also be simultaneously active for the same computation.) Compared to the worst case 32-bit carry chain length, which is 64 gate delays, the average case delay reflects a dramatic 5-6X performance improvement. This suggests that a ripple carry adder with computation completion signaling can potentially incorporate the low power and area of ripple carry adders with average performance comparable to that of carry look ahead adders. However, there is an additional overhead associated with the circuitry producing the completion signal; the challenge is to keep this as low as possible.

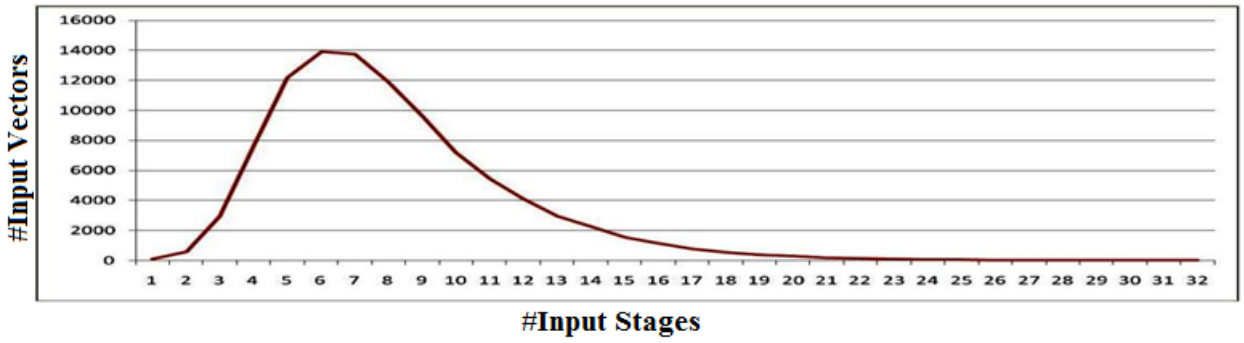


Figure 23: Carry Propagation graph for two 32 bit additions for 100000 random vectors.

5.2 RCA Design

The new proposed design for ripple carry adder with completion signaling is shown in Figure 24. The new carry completion signaling design discussed in chapter 3 is incorporated in a 32-bit ripple carry adder to produce a carry completion signal. This modified ripple carry adder has a strong delay dependency on carry chain length, which is the critical path for the design. So monitoring the activity on the carry out lines only gives us information about the delays seen in calculating the sum bits for different input vectors. A clock is used for the operation of the current sensor circuitry. As discussed earlier this clock is used to generate control signals.

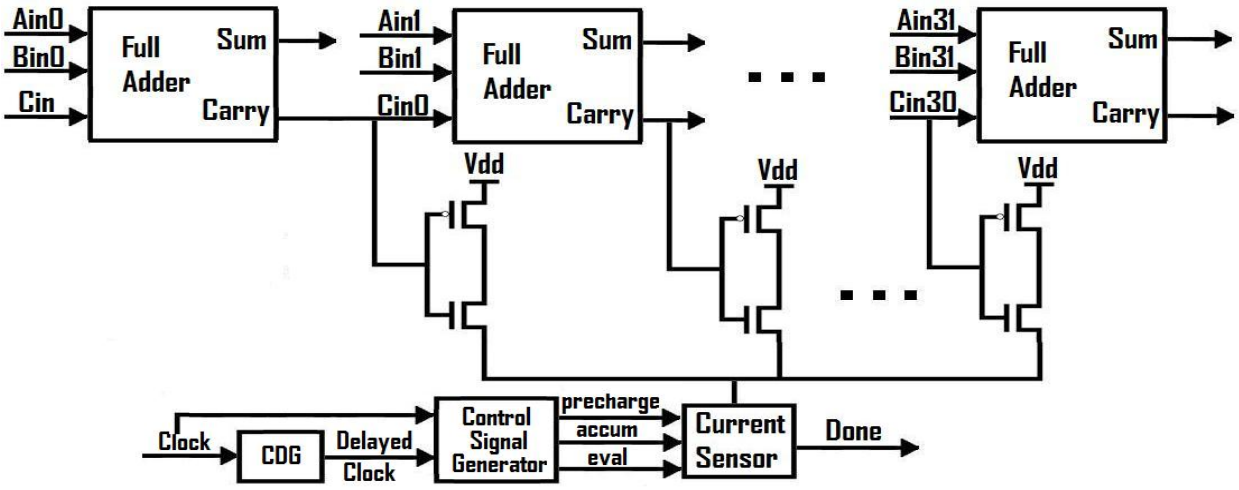


Figure 24: Ripple Carry Adder with sensor circuitry.

5.3 RCA Operation

The sense inverters are added to carry out signals from all the output blocks except the final stage. The carry out of the final stage is neglected, as it does not play any role in calculation of sum bits. For 32-bit additions if there is any carry propagation then the carry signals have transitions on their lines. Due to these transitions, current is drawn from the power supply by the sense invertors. This current is captured into the current sensor and is compared with a pre-chosen threshold. Whenever the captured current is above the threshold, the output done line will be active low, indicating carry propagation by the full adder blocks. But when the current falls below the threshold, there are no transitions on the carry out lines and the circuit has reached stable state values. At this moment the done signal goes high and the adder is ready is compute the next addition.

In case of asynchronous current sensor designs the done signal is flagged as soon as all the carry lines settle to a stable state. But the design here is synchronous, which includes a clock signal. So we compare current once in each clock cycle and provide a done signal. This clock

signal can be used as a master reference clock and another clock with high frequency can be used to create multiple evaluation points which fit into the clock period of the master clock.

The general operation for the adder is to load the values to the registers after waiting for the worst case delay; for the RCA this is equivalent to the 64 gate delays associated with the carry propagation path. For loading the values we use a master clock (Clk_1) which is operated at a frequency equal to the worst case delay (assuming the set up and hold time constraints are met). The new RCA with sensor circuitry uses a different clock faster, (Clk_2/n) by a factor of 'n' with respect to the main clock. So the values are now loaded for the next addition on the rising edge of the clock depending on the value of the 'Done' signal as shown in Figure 25. If, at the first clock edge the 'Done' signal is active high, indicating addition is complete, the new values are loaded to the registers; if not so, then the next clock edge is used for loading the values.

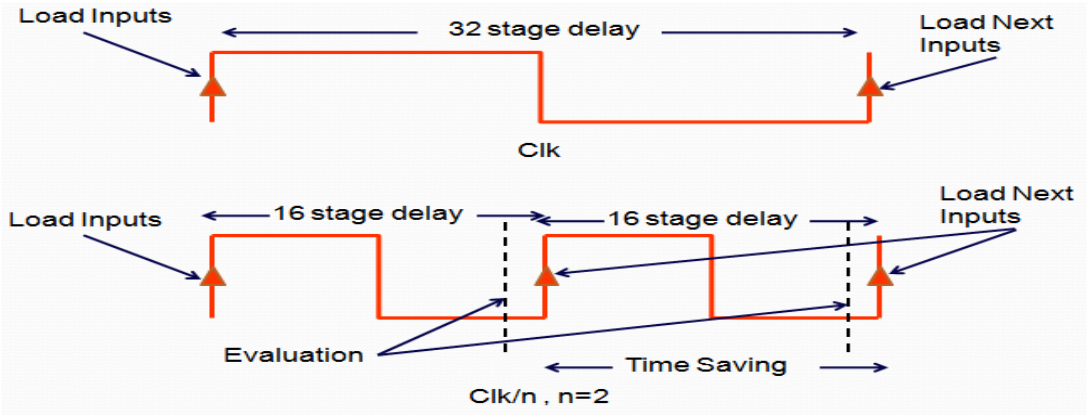


Figure 25 : Clock Division

5.4 Calculations

$Clk_1 = T_{wcd}$ = 32 stage worst case delay without sense invertors

$Clk_2 = T_{wcd}$ + 0.2ns* +0.4ns**

* Delay introduced due to sense invertors on carry signals

** Delay added for evaluation over one complete addition

Timed saved = $(T_1-T_2) / T_1$

Where, T_1 = Total time with Clock set to Clk_1

T_2 = Total time with Clock set to Clk_2/n, n = 2, 3, 4.

Here to have multiple comparison slots we divide the Clk_2 by a factor 'n'. Here 'n' indicates the number of comparisons.

5.5 Optimizations

In the analysis different types of optimizations are used in automatic synthesis tool Leonardo spectrum for generating the netlist. Figures 26 and 27 show the gate level representation of a 1-bit full adder block after the optimizations are performed by the synthesis tool.

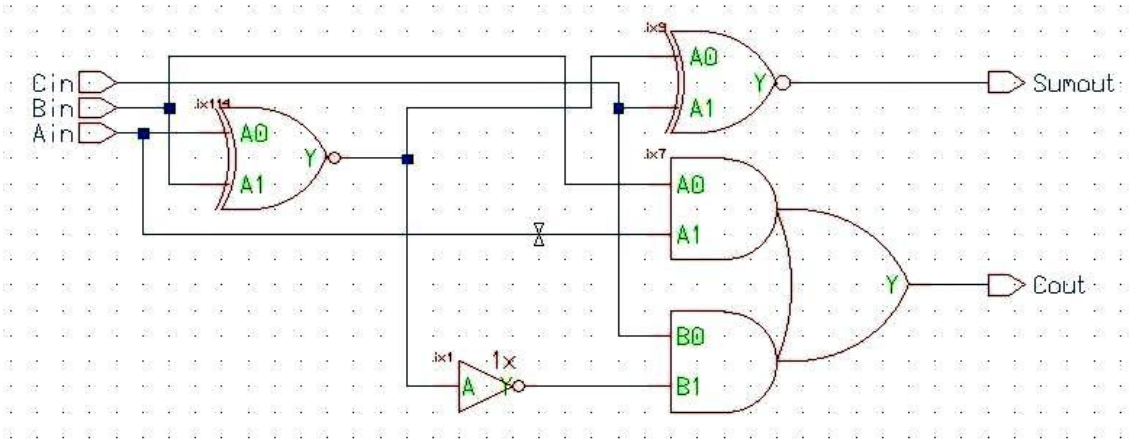


Figure 26: Optimization w.r.t Area

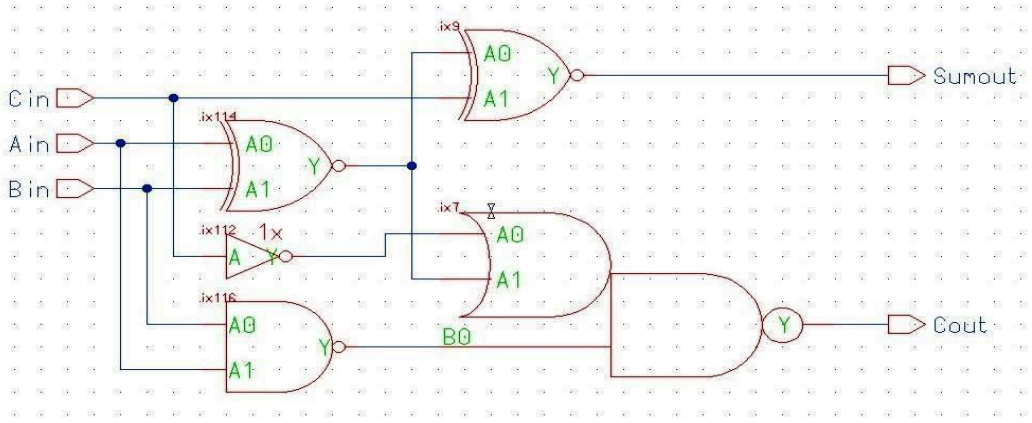


Figure 27: Optimization w.r.t Delay

5.6 Simulations

5.6.1 Ripple Carry Adder with Sensor Circuitry

The SPICE simulations were performed with 100000 random vectors to explore the correlation between delay for the RCA with and without the sensor circuitry. The Clock Delay Generator (CDG), for the design with the sensor, is set to 0.4ns which is the evaluation time of the current sensor. The netlist parameters were obtained by using three different types of synthesis optimization in automatic synthesis tool Leonardo Spectrum using 180nm technology.

1) Area + Delay

$Clk_1 = T_{wcd} = 5.0ns$

$Clk_2 = T_{wcd} + 0.2ns + 0.4ns = 5.6ns$

Clock(ns)	$Clk_2/2$	$Clk_2/3$	$Clk_2/4$
Time Saved	41.09%	52.18%	53.02%

Table 1: Simulation results with area and delay optimization for RCA with and without sensor circuitry.

2) Area

$Clk_1 = T_{wcd} = 5.1ns$

$Clk_2 = T_{wcd} + 0.2ns + 0.4ns = 5.7ns$

Clock(ns)	$Clk_2/2$	$Clk_2/3$	$Clk_2/4$
Time Saved	41.21%	52.28%	53.01%

Table 2: Simulation results with area optimization for RCA with and without sensor circuitry.

3) Delay

$Clk_1 = T_{wcd} = 4.8ns$

$Clk_2 = T_{wcd} + 0.2ns + 0.4ns = 5.4ns$

Clock(ns)	$Clk_2/2$	$Clk_2/3$	$Clk_2/4$
Time Saved	40.83%	51.96%	52.89%

Table 3: Simulation results with delay optimization for RCA with and without sensor circuitry.

5.6.2 RCA Vs CLA

The SPICE simulations performed here are to explore the correlation between delay for the RCA with the sensor circuitry and Carry Look ahead adder. These netlist parameters were obtained by using three different types of synthesis optimization in automatic synthesis tool Leonardo Spectrum using 180nm technology.

1) Area + Delay

$Clk_1 = 3.1ns$

$Clk_2 = 5.0ns + 0.2ns + 0.4ns = 5.6ns$

Clock(ns)	$Clk_2/2$	$Clk_2/3$	$Clk_2/4$
Time Saved	5.25%	21.91%	23.68%

Table 4: Simulation results with Area + Delay optimization for RCA and CLA.

Design	RCA + Sensor	CLA	Overhead
#Transistors	1182	1450	22.67%

Table 5: Area Overhead calculations with Area + Delay optimization.

2) Area

$Clk_1 = 3.2ns$

$Clk_2 = 5.1ns + 0.2ns + 0.4ns = 5.7ns$

Clock(ns)	$Clk_2/2$	$Clk_2/3$	$Clk_2/4$
Time Saved	6.57%	22.99%	24.74%

Table 6: Simulation results with Area optimization for RCA and CLA.

Design	RCA + Sensor	CLA	Overhead
#Transistors	1182	1450	22.67%

Table 7: Area Overhead calculations with Area + Delay optimization.

3) Delay

$Clk_1 = 2.9ns$

$Clk_2 = 4.8ns + 0.2ns + 0.4ns = 5.4ns$

Clock(ns)	$Clk_2/2$	$Clk_2/3$	$Clk_2/4$
Time Saved	2.33%	19.49%	21.32%

Table 8: Simulation results with Delay optimization for RCA and CLA.

Design	RCA + Sensor	CLA	Overhead
#Transistors	1451	2008	38.39%

Table 9: Area Overhead calculations with Delay optimization.

5.7 Power Analysis

5.7.1 Average Power Calculations

Table 10 gives the average power (in mW) calculations for the three circuit designs, Ripple carry adder, Ripple carry adder with sensor circuitry and the Carry look ahead adder.

Longest Carry Chain Length / Inputs	RCA	RCA with CSCD			CLA
		Clk/2	Clk/3	Clk/4	
0 (0x 22222222 + 0x 11111111)	0.555	0.647	0.681	0.788	0.867
2 (0x 33333333 + 0x 11111111)	0.811	0.883	1.061	1.170	1.441
4 (0x 0F0F0F0F + 0x 01010101)	1.014	1.093	1.270	1.378	1.682
6 (0x 3F3F3F3F + 0x 01010101)	1.668	1.776	1.883	1.990	2.512
8 (0x 00FF00FF + 0x 00010001)	1.143	1.184	1.360	1.446	1.771
10 (0x 03FF03FF + 0x 00010001)	1.446	1.567	1.675	1.780	2.204
12 (0x 0FFF0FFF + 0x 00010001)	1.757	1.943	2.052	2.156	2.719
16 (0x 0000FFFF + 0x 00000001)	1.049	1.161	1.250	1.351	1.835
20 (0x 000FFFFF + 0x 00000001)	1.191	1.264	1.338	1.476	2.290
24 (0x 00FFFFFF + 0x 00000001)	1.322	1.465	1.576	1.605	2.767
28 (0x 0FFFFFFF + 0x 00000001)	1.457	1.545	1.636	1.737	3.234
32 (0x FFFFFFFF + 0x 00000001)	1.583	1.636	1.727	1.856	3.658

Table 10: Average Power Calculations

5.7.2 Energy Calculations

Table 11 gives the energy per addition (in mJ) calculations for the three circuit designs, Ripple carry adder, Ripple carry adder with sensor circuitry and the Carry look ahead adder.

Longest Carry Chain Length / Inputs	RCA	RCA with CSCD			CLA
		Clk/2	Clk/3	Clk/4	
0 (0x 22222222 + 0x 11111111)	0.160	0.186	0.196	0.227	0.260
2 (0x 33333333 + 0x 11111111)	0.467	0.509	0.612	0.674	1.730
4 (0x 0F0F0F0F + 0x 01010101)	0.877	0.945	1.098	1.192	2.018
6 (0x 3F3F3F3F + 0x 01010101)	1.923	2.048	2.171	2.294	5.275
8 (0x 00FF00FF + 0x 00010001)	1.647	1.707	1.960	2.083	3.720
10 (0x 03FF03FF + 0x 00010001)	2.500	2.71	2.896	3.078	4.628
12 (0x 0FFF0FFF + 0x 00010001)	3.545	3.921	4.140	4.349	5.709
16 (0x 0000FFFF + 0x 00000001)	2.721	3.011	3.242	3.504	3.853
20 (0x 000FFFFF + 0x 00000001)	3.776	4.009	4.244	4.680	6.871
24 (0x 00FFFFFF + 0x 00000001)	4.953	5.49	5.872	6.014	8.301
28 (0x 0FFFFFFF + 0x 00000001)	6.301	6.68	7.073	7.508	9.701
32 (0x FFFFFFFF + 0x 00000001)	7.758	8.016	8.460	9.095	10.973

Table 11: Energy per addition calculations

5.8 Results Analysis

The design of the 32-bit modified RCA used 1182 transistors in its design, which includes the CSCD circuitry. In contrast, the 32-bit RCA without sensor circuitry used 1079 transistors. This suggests a 9-10% area overhead. The average speedup achieved was between 40-55% in all the three cases of optimizations.

The design the 32-bit CLA used 1450 transistors which is 23% more compared to 32-bit RCA with CSCD circuitry. The performance improvement achieved was between 5-25% in all the three cases of optimizations.

The modified Ripple carry adder with sensor circuitry consumes 10-20% more power than the standard Ripple carry adder. But the consumption is less when compared to Carry Look Ahead adder which takes 10-50% more power.

Chapter 6

Booth Multiplier

6.1 Background

The Booth multiplier performs the multiplication of two signed numbers using the signed binary notations. The Booth multiplier tries to decrease the number of partial products generated for an n-bit multiplication. In general, for an n-bit multiplication n partial products are generated and added to form the final result of the product.

The basic idea for multiplying two binary numbers is “shift and add”. That is, for each column in the multiplier, shift the multiplicand the appropriate number of columns and multiply it by the value of the digit in that column of the multiplier, to obtain a partial product. The partial products are then added to obtain the final result. As shown in example below, we get four partial products to add for 4-bit multiplication.

```
          0  0  1  1      (3)
      x   0  1  1  0      (6)
-------------------------------
          0  0  0  0
       0  0  1  1
    0  0  1  1
 0  0  0  0
-------------------------------
 0  0  1  0  0  1  0      (18)
```

Radix-4 Booth encoding reduces the number of partial products by half. The idea here is that, instead of shifting and adding for every column of the multiplier term and multiplying by 1 or 0, we only take every second column, and multiply by ±1, ±2, or 0, to obtain the same results. In the radix-4 booth algorithm the multiplier is divided into substring of 3 bits and based on the bit pattern, operation is performed as per Booth encoding shown in Table 12.

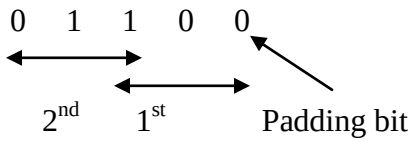
Bit Pattern	Operation
000	0
001	1 * Multiplicand
010	1 * Multiplicand
011	2 * Multiplicand
100	-2 * Multiplicand
101	-1 * Multiplicand
110	-1 * Multiplicand
111	0

Table 12: Booth Encoding

Booth encoding has an advantage of reducing the number of partial products. For radix-4 encoding the number of partial products is halved. This is important in circuit design as it relates to the propagation delay in the operation of the circuit, and the complexity and power consumption of its implementation.

The above example can be multiplied using the Booth encoding as below, we consider the bits in blocks of three, such that each block overlaps the previous block by one bit. The overlap is necessary so that we know what happened in the last block, as the MSB of the block acts like a sign bit. Grouping starts from the LSB, and the first block only uses two bits of the multiplier.

Since there is no previous block to overlap a padding of 1-bit is used, which is added to the right towards the least significant bit.



For 1st substring, Addition term as per booth encoding = (-1) * Multiplicand

For 2nd substring, Addition term as per booth encoding = (2) * Multiplicand

The final product is obtained by adding the two partial products with a 2nd partial product shifted left by two bits.

6.2 Modified Booth Multiplier

The Booth multiplier performs the addition of the partial products. To achieve time saving, Ripple Carry Adder (RCA) with completion signaling is used in our Radix-4 Booth Multiplier by replacing the standard RCA in the design as shown in Figure 28.

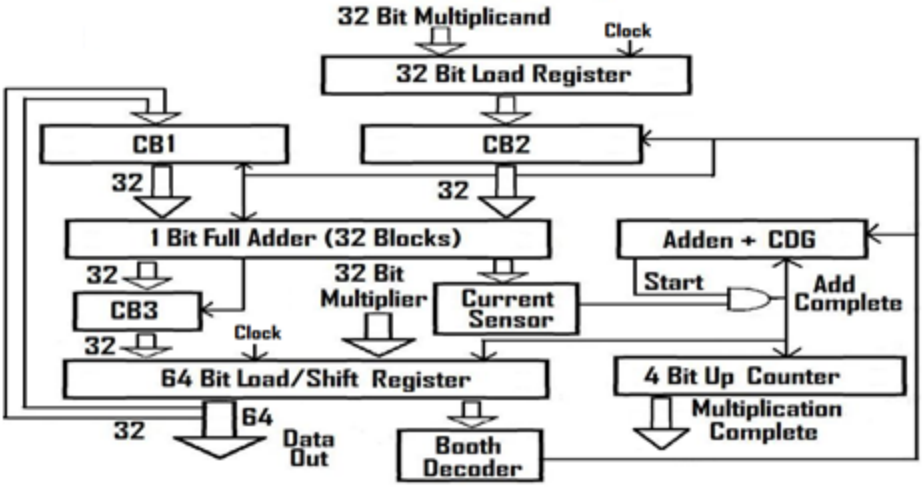


Figure 28: Booth Multiplier with Ripple Carry Adder and sensor circuitry.

6.3 Calculations

The Booth multiplier uses a master clock to load values to the registers. We run the master clock at a frequency equal to the worst case delay required for adders.

1. When RCA is used the clock is set to Clk_1

   Clk_1 = 32 stage delay without sense invertors

2. When RCA with completion detection circuitry is used the clock is set to Clk_2

   $Clk_2 = Clk_1 + 0.2ns^* + 0.4ns^{**}$

   * Delay added due to sense invertors on carry signals

   ** Delay added for evaluation over one complete addition

Here we divide the Clk_2 by a factor ‘n’, where ‘n’ indicates the number of comparisons. So we now operate the Booth multiplier at a faster clock rate given by Clk_2/n.

3. When CLA is uses the clock is set to Clk_3

   Clk_3 = delay seen for calculating carry for last block.

Timed saved = $(T_1-T_2)/T_1$

Where, T_1 = Total time with Clock set to Clk_1

T_2 = Total time with Clock set to Clk_2/n or Clk_3/n, n = 2, 3, 4.

It is important to note that when '0' is added , it's a simple shift operation without an addition, so a faster clock can be used and when there is addition the worst case delay wait is added. This can improve the overall performance of the Booth multiplier.

6.4 Simulations

6.4.1 BM with RCA vs BM with RCA and Current Sensor

For comparison, simulations for 1000 random input vectors with two designs of booth multiplier, one with RCA and other with RCA and current sensor circuitry are performed. In each simulation run two clocks, Clk_1 and Clk_2, are used with time period scaled by a factor 'n', where 'n' is number of evaluations per addition. A performance improvement between 30-50% was observed with overall area overhead less than 2% of the silicon area (Table 13).

For BM with RCA, the shift operation is done in a Clk_1/n (n=2, 3, 4) time period and for additions the worst case delay is applied and the new values are loaded in Clk_1 time period. For BM with sensor circuitry, the Clk_2/n time period is used and values are loaded only the when add completion signal is high at an active clock edge. The Clock Delay Generator in RCA with Current sensor circuitry is set to a delay of 0.4ns, which is the response time of the Current sensor.

Clock(ns)	$Clk_2/2$	$Clk_2/3$	$Clk_2/4$
Time Saved	33.69%	46.89%	49.25%

Table 13: Simulation results with BM with RCA with and without sensor circuitry.

6.4.2 BM with CLA vs BM with RCA and Current Sensor

For comparison, simulation for 1000 random input vectors with two designs of booth multiplier, one with CLA and other with RCA and current sensor circuitry are performed. In each simulation run two clocks, Clk_1 and Clk_2, are used with time period scaled by a factor 'n'(n=2, 3, 4). A performance improvement between 15-30% was observed while keeping the overall area overhead less than 2% of the silicon area (Table 14).

For BM with CLA, the shift operation is done Clk_1/n time period and for additions the worst case delay is applied and the new values are loaded in Clk_1 time period. For BM with sensor circuitry uses Clk_2/n time period, values are loaded only when add completion signal is high at active clock edge. The Clock Delay Generator in RCA with Current sensor circuitry is set to a delay of 0.4ns i.e. response time of the Current sensor.

Clock(ns)	$Clk_2/2$	$Clk_2/3$	$Clk_2/4$
Time Saved	16.45%	24.45%	29.98%

Table 14: Simulation results with BM with RCA with sensor circuitry and CLA.

6.5 Results Analysis

The Booth multiplier when replaced with the RCA with Current sensor circuitry can be operated at a higher clock rate to get improved performance. This average time saving achieved is found to be between 30-50%. When the same design is compared with a booth multiplier which incorporates CLA for addition the performance improvement is between 20-30%.

Chapter 7

Conclusion

Providing carry completion signaling in low cost ripple carry adders can allow the control logic to schedule the next addition as soon as an earlier one is complete, thereby achieving the average case, rather than worst case addition delay over a set of computations. Earlier attempts at using current sensing for such carry completion signaling suffered from serious limitations. In this thesis we presented a new approach for the design of a ripple carry adder with a current sensing capability which observes late settling carry signal nodes in the circuit and indicates when they reach a quiescent state. Simulations show better than 50% speedup, on average, with less than 10% area overhead. To demonstrate a potential application of such an approach, we incorporated our carry completion adder into a Booth multiplier design and studied the performance gain over a traditional ripple carry adder based design. Simulation results show that a 32-bit Booth Multiplier using the new completion signaling circuits can outperform a 32-bit Booth Multiplier with ripple carry adder (RCA) by 30-50%, while requiring less than 2% additional silicon area, This is comparable to the gains from the best carry look ahead adder designs at a fraction of the area overhead costs.

Also, the observed performance improvement over a carry look ahead adder is 20-30%, with an area overhead less than 40% of that required by a carry look ahead adder. This area overhead will save power and area in smaller circuit designs. When the clock is set to average case RCA delay, the average case delay is found to be 8-9 times the longest carry chain length. At first this number appears bigger than the 6-7 long average case carry chain length for 32-bit

adders found by our preliminary simulations presented in Figure 23, recall however that our design incorporates the current sensor that limits the earliest completion signal by adding an additional overhead of 0.6ns, 0.2ns for additional load on carry lines and 0.4ns for the current sensor response over one complete addition. Careful designs along with reduction of the response time of the sensor can potentially further improve the average case delay of our design by 10-20%.

Future work is focused on investigating other applications of the completion signaling approach presented here.

References

1. F. Cheng, S. Unger, and M. Theobald, “Self-Timed Carry Lookahead Adders”. IEEE Transactions on Computers, vol. 49, no. 7, July 2000.
2. M. E. Dean, D. I. Dill, and M. Horowitz, “Self-timed logic using current-sensing completion detection (CSCD)”. Journal of CLSI Signal Processing, vol. 7, no. 1-2, pp. 7-16, February 1994.
3. E. Grass and S. Jones, “Asynchronous circuits based on multiple localized current-sensing completion detection”. Proceedings: Second Working Conference on Asynchronous Design Methodologies. pp. viii+223, 170-177, London, UK, May 1995.
4. O. A. Izosimov, I. I. Shagurin, and V. V. Tsylyov, “Physical approach to CMOS Self-Timing”. Electronic Letters, vol. 26, pp. 1835-1836, 1990.
5. C. G. Knight, A. D. Singh, and V. P. Nelson, “An IDDQ Sensor for Concurrent Timing Error Detection”. IEEE Journal of Solid-State Circuits, vol. 33, no. 10, October 1998.
6. H. Lampinen, O. Vain, “Circuit design for current-sensing completion detection”. ISCAS ’98: Proceedings of the 1998 IEEE International Symposium on Circuits and Systems. Monterey, CA, vol. 2, pp. 185-188,June1998
7. Weste, N., and Eshraghian, K.: “Principles of CMOS VLSI design” (Addison Wesley, 1993)
8. Beerel, P.A.: ‘Asynchronous circuits: an increasingly practical design solution’. Proc. Int. Symp. on Quality Electronic Design, 18–21 March 2002, pp. 367–372
9. Perri, S., Corsonello, P., and Cocorullo, G.: ‘VLSI circuits for lowpower high-speed asynchronous addition’, IEEE Trans. VLSI, 2002, 10, (5), pp. 608–613

10. De Gloria, A., and Olivieri, M.: ‘Completion-detecting carry select addition’, IEE Proc., Comput. Digit. Tech., 2000, 23, pp. 93–100
11. Ruiz, G.A., and Manzano, M.A.: ‘Compact 32-bit CMOS adder in multiple-output DCVS logic for self-timed circuits’, IEE Proc., Circuits Devices Syst., 2000, pp. 183–188
12. De Gloria, A., and Olivieri, M.: ‘ Statistical carry lookahead adders’, IEEE Trans. Comput., 1996, pp. 340–347 7 Escrib!a, J., and Carrasco, J.A.: ‘Self-timed Manchester chain carry propagate adder’, Electron. Lett., 1996, 32, (8), pp. 708–710
13. Reitwiesner, G.W.: ‘The determination of carry propagation length for binary addition’, IRE Trans. Electron. Comput., 1960, pp. 35–38.
14. Nowick, S.M.: ‘Design of a low-latency asynchronous adder using speculative completion’, IEE Proc., Comput. Digit. Tech., 1996, 143, (5), pp. 301–307
15. Garside, J.D.: ‘A CMOS VLSI implementation of an asynchronous ALU’. Proc. IFIP Conf. on Asynchronous Design Methodologies, Manchester, UK, 1993, pp. 181–192
16. Ruiz, G.A.: ‘Evaluation of three 32-bit CMOS adders in DCVS for self-timed circuits’, IEEE J. Solid-State Circuits, 1998, 33, (4), pp. 604–613
17. Martin, A.J.: ‘Asynchronous datapaths and the design of an asynchronous adder’, Form. Methods Syst. Des., 1992, 1, (1), pp. 119–137
18. Cheng, F.-C., Unger, S.H., Theobald, M., and Cho, W.-C.: ‘Delay insensitive carry-lookahead adders’. Proc. 10th Int. Conf. VLSI Design, Jan. 1997, pp. 322–328
19. Cheng, F.-C., Unger, S.H., and Theobald, M.: ‘Self-timed carrylookahead adders’, IEEE Trans. Comput., 2000, 49, (7), pp. 659–672
20. Hauck, S.: ‘Asynchronous design methodologies: an overview’, Proc. IEEE, 1995, 83, (1), pp. 69–93

21. Parhami, B.: ‘Computer arithmetic: algorithms and hardware design’ (Oxford University Press, 2000)
22. H. Lampinen and 0. Vainio, “Circuit Design for Current- Sensing Completion Detection,” IEEE Intemational Symposium on Circuits and Systems, Monterey, California, USA, Vol. 2, pp. 185-188, May 31 -June 3 1998.
23. E. Grass, V. Bartlett, and I. Kale, “Completion-Detection Techniques for Asynchronous Circuits,” IEICE Tramactions on Zilfomzation and Systems, Vol. E80-D, No. 3, pp.
24. E. Grass and S. Jones, “Activity-Monitoring Completion- Detection (AMCD): A New Approach to Achieve Self- Timing,” Electronics Letters, Vol. 32, No. 2, pp. 86-88, Jan. 1996.
25. H. Lampinen, Olli Vainio, “Current-Sensing Completion Detection Method For Standard Cell Based Digital System Design”, 2002.
26. W. Burks, Herman H. Goldstine, John Von Neumann, “Preliminary discussion of the logical design of an electronic computing instrument”.
27. Harri Lampinen, Olli Vainio, “Dynamically Biased Current Sensor for Completion Detection”.