2019

A Low-Area, Energy-Efficient 64-Bit Reconfigurable Carry Select Modified Tree-Based Adder for Media Signal Processing

Priscilla Sharon Allwin
Wright State University

A LOW-AREA, ENERGY-EFFICIENT 64-BIT RECONFIGURABLE CARRY SELECT MODIFIED TREE BASED ADDER FOR MEDIA SIGNAL PROCESSING

A Thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Electrical Engineering

by

PRISCILLA SHARON ALLWIN
B.E., Anna University, 2016

2019
Wright State University

Wright State University
GRADUATE SCHOOL

July 30, 2019

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Priscilla Sharon Allwin ENTITLED A Low-Area, Energy-Efficient 64-Bit Reconfigurable Carry Select Modified Tree Based Adder for Media Signal Processing BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science in Electrical Engineering.

Henry Chen, Ph.D.
Thesis Director

Fred D. Garber, Ph.D.
Chair, Department of Electrical Engineering

Committee on
Final Examination

Henry Chen, Ph.D.

Saiyu Ren, Ph.D.

Raymond E. Siferd, Ph.D.

Barry Milligan, Ph.D.
Interim Dean of the Graduate School

ABSTRACT

Allwin, Priscilla Sharon. M.S.E.E., Department of Electrical Engineering, Wright State University, 2019. *A Low-Area, Energy-Efficient 64-Bit Reconfigurable Carry Select Modified Tree Based Adder for Media Signal Processing.*

Multimedia systems play an essential part in our daily lives and have drastically improved the quality of life over time. Multimedia devices like cellphones, radios, televisions, and computers require low-area and low-power reconfigurable adders to process greedy computation algorithms for the real-time audio/video signal and image processing such as discrete cosine transform, inverse discrete cosine transform, and fast Fourier transform, etc. In this thesis, a novel 64-bit reconfigurable adder is proposed and implemented to reduce the area and power consumption. This adder can be run-time reconfigured to different reconfigurable word lengths, i.e., one 64- bit, two 32-bits, four 16-bits or eight 8-bits addition, depending on the partition signal command. A Carry Select Modified Tree (CSMT) based adder is used in the reconfigurable adder to reduce the area by 22 % and the power consumption by 47 % when compared to the conventional design. The proposed adder, implemented in 180 nm CMOS technology at 1.8-volt supply, has a worst-case Delay of 20.67 nanoseconds with an overall area of $36.417\,\mu\mathrm{m}^2$ and power consumption of $447.93\,\mu\mathrm{W}$

Contents

List of Figures

List of Tables

Acknowledgment

First and foremost I would like to thank God Almighty for giving me the strength and encouragement to complete this thesis.

I take this opportunity to express my sincere thanks to my Advisor Dr. Henry Chen, without whom this research would have not been possible. I would also like to extend my thanks to Dr. Saiyu Ren and Dr. Ray Siferd for serving as my committee members and offering their valuable suggestions. I would also like to thank the Electrical Engineering Department for giving me the opportunity and the required resources to complete this thesis sucessfully.

I would also like to thank my best friend Mano for always being there for me and constantly encouraging and pushing me to do my best.

Finally, I would like to take this moment to thank my parents for giving me the freedom to chase my dreams. Without their constant support and unconditional love, this would have not been possible.

Dedicated to

Amma and Appa

Introduction

1.1 High Speed Adders

An adder is a digital circuit that manipulates a combination of electronic signals where a high signal is represented as 1 and a low signal is represented as 0. Various patterns of signals are added to produce different results. Adders are the basic circuits for constructing various complex designs such as subtractors, dividers and multipliers which perform more advanced arithmetic operations in many media signal processing applications. Ultimately an adder can be called as the fundamental unit of modern computing due to its extensive usage in real-time signal processes. Modern-day circuits emphasize on the faster operation, smaller area, and minimal power consumption, which leads to the need for better high speed, low area, and energy-efficient adder designs.

1.1.1 Description of a basic adder

A simple adder circuit can be constructed using an XOR & AND gate. Such a circuit is called the Half Adder (HA) which performs simple binary additions. The circuit below describes the construction of a half adder along with its truth table.

Table 1.1: The truth table of a Half Adder

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

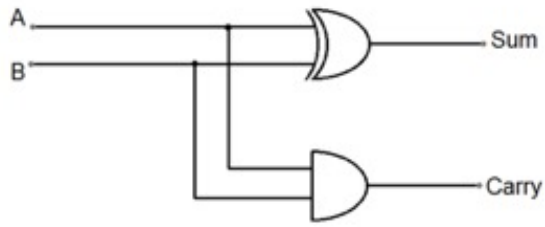


Figure 1.1: Half Adder circuit [28]

Another type of adder called the Full Adder (FA) takes a carry bit into account in addition to adding two binary numbers. The carry bit is from the Least Significant Bit (LSB) position. The logic expressions (Equations 1.1 and 1.2), truth table, and circuit construction are as follows.

$$S = A \oplus B \oplus C \tag{1.1}$$

$$C_{OUT} = AB + BC + CA \tag{1.2}$$

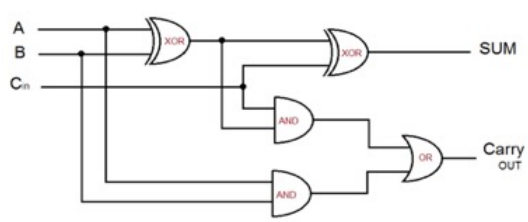


Figure 1.2: Full Adder circuit [29]

Table 1.2: The truth table of a Full Adder

A	B	Ci	S	Co
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

1.1.2 Critical path in adders

A critical path is described as the longest path in the entire circuit, which has the maximum delay. It is an important parameter which has more considerable significance when designing high-speed circuits. As technology evolves, there is a need for circuits to compute large amounts of data in a short period. Therefore, the need for reducing the length of the critical path becomes an essential factor for high-speed designs. Generally, for a Full Adder circuit, the critical path begins from the carry-in bit and ends at the carry-out bit. It goes through one XOR gate and two other gates (AND & OR) to reach the output; therefore, the delay can be represented as given in Eq.(1.3).

$$T_{CRITICAL} = T_{XOR} + T_{AND} + T_{AND} \tag{1.3}$$

1.1.3 Evolution of various adder designs

Ripple Carry Adder (RCA): A basic 1-bit adder circuit can be modified to perform N- bit additions by cascading N 1-bit adders with the carry bit rippling from the first one to the last one to reach the output. The design is simple but slow since each bit has to wait for the carry bit rippling from the previous bit.

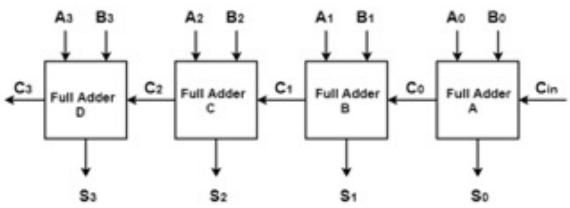


Figure 1.3: Ripple Carry Adder Design [30]

Engineers started developing better designs to try and minimize the computational delay [1]-[4],[6] which gave rise to different architectures such as Carry Look Ahead Adder, Carry Select Adder, Carry Skip Adder and so on.

Carry Look Ahead Adder (CLA): This type of adder introduces two signals for each bit position called the Propagate (P) and Generate (G) signal. Both the signals are derived from Eq. (1.4) and (1.5).

$$P = A + B \tag{1.4}$$

$$G = A.B \tag{1.5}$$

A carry is Propagated (either A =1 or B = 1), Generated (both A and B = 1) or Killed (both A and B = 0), depending on the input values. Various modifications were made based on this architecture such as Manchester Carry Chain, Brent Kung Adder and Kogge-Stone Adder.

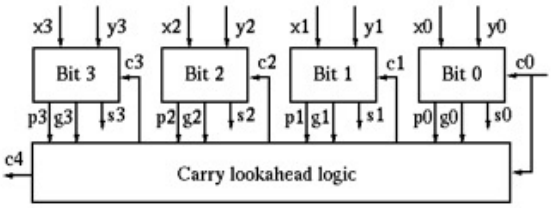


Figure 1.4: Carry Look Ahead Adder Design [31]

Carry Select Adder (CSA): The construction is simple; it consists of a set of identical N-bit Ripple Carry Adders along with a Multiplexer. Each of the Ripple Carry Adder performs normal addition, assuming that the carry-in is 0 (RCA 1) and 1 (RCA 2).

After both the results have been pre-calculated, the actual sum and carry out is selected using the multiplexers depending on the correct carry-in (Cin) value. The number of bits in each block can be uniform or variable depending on the application. This adder structure can also be combined with other structures to improve adder performance based on the circuits necessity.

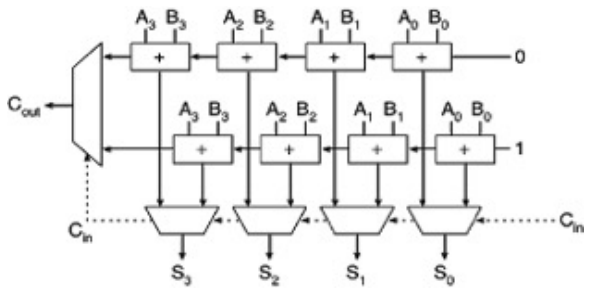


Figure 1.5: Carry Select Adder Design [32]

Carry Skip Adder (CSK): This type of adder is also called the carry by-pass adder, and it helps to improve the delay performance of a ripple carry adder in a much simpler manner. It utilizes the propagate signal of each block which is combined to form a select line for the multiplexer. Depending on the select signal the carry-out signal is obtained either by rippling through the entire circuit or the original carry in by-passes and reaches the output.

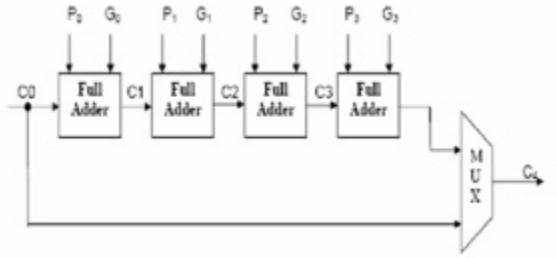


Figure 1.6: Carry Skip Adder Design [33]

1.2 Applications of adders in Media Signal Processing

Modern communication technology is an integral part of our routine lives and has drastically evolved changing our lifestyle. There are four major places in which technical developments are happening all the time [7].

i. The rate at which the data is being transferred

ii. The presence of Packed Switch Networks in every realm

iii. The evolution of wireless communications

iv. The ever-increasing demand for broadband access

All the four points mentioned above require the need of very high speed data path systems [1]- [6] that perform some of the most critical functions that are needed for the proper functioning and coordination of various sub-blocks that make up a multimedia system such as cellphones, radios, Personal Computers, workstations, etc. Adders find their importance in various aspects of media signal processing and are continually evolving to meet the growing needs of all consumers.

1.2.1 Goals of Multimedia Signal Processing

Multimedia signal processing is more than just putting texts, audio, video, and images together. It is the integration and interaction among different media that gives rise to newer systems with unforeseen challenges and opportunities. Media Signal Processing is defined as [9] ”the representation, interpretation, encoding, and decoding of multimedia data by utilizing signal processing tools.” The ultimate goal is the effective and efficient access, manipulation, fast exchange, and storage of multimedia content for various applications.

1.2.2 Usage of adders in MSP/DSP algorithms

Digital Signal Processors / Media Signal Processors [8] [9] utilize various algorithms to ensure the proper functioning of each block. The basic building blocks of these algorithms are a combination of adders and multipliers, along with other components. Therefore, there is a constant need for adders to perform computations faster to achieve faster processing speeds. Some of the algorithms involving the use of adders are described below,

i. Digital Filters

Digital filters are used to remove unwanted components from a signal when it is being processed. Depending on the type of application, it can be classified into two types, Finite Impulse Response filters [3] [4] [8] and Infinite Impulse Response filters. These filters are of enormous importance in multimedia applications. It is used for the noise suppression in bio-imaging devices, bio-signal devices [5] and for retrieving signals stored in analog media. Filters are also used for the enhancement of specific selective ranges of frequency for audio systems (Equalizers) and some enhancement techniques of images [10] [11] [12] [14].

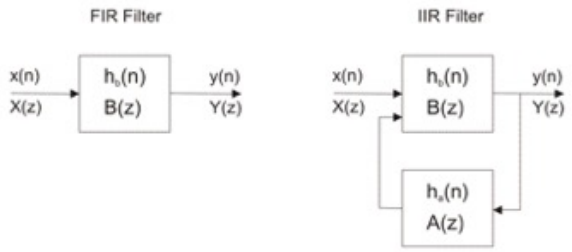


Figure 1.7: Transfer Function Representation of FIR and IIR filter Design [34]

They also contribute to the process of attenuation/ removal of specific frequencies that causes interference.Filters are also used in the bandwidth limitation for sampling through the use of anti-aliasing filters to make sure that a transmitted signal is occupying only its allotted frequency band.

ii. Discrete Fourier Transforms

Fast Fourier Transform (FFT) [8] is the algorithm, which is used for computing the Discrete Fourier Transform, which is one among many computation greedy functions. These algorithms are mainly used in signal and image processing [5]. FFT has made it computationally possible to work with the frequency domain, making it similar to working in either time or space domain. These FFT algorithms are utilized to perform large integer and polynomial multiplications, faster matrix multiplications for various structured matrices, filtering algorithms such as over-lap add, over-lap save and also for generating algorithms for Discrete Cosine or Sine Transforms [3] [4] in data encoding and decoding techniques.

iii. Discrete Cosine Transforms

Discrete Cosine Transform [3] [4] [8] is another computation hungry function which is majorly used in the lossy/irreversible compression of images (e.g., JPEG) and audio (e.g., MP3, WMA). The usage of the Cosine function is significant in compression since the number of cosine functions required to estimate a typical signal is fewer than the sine function. DCT is also the best choice when it comes to image and signal processing due to its high energy compression property.

1.2.3 Adders in processor units

Processors are the core of every multimedia device such as personal digital assistants, mobile phones, digital cameras, gaming consoles, personal computers, and workstations. They help in coordinating several processes to happen simultaneously. When we observe the block diagram of any processor, we can keep the presence of adder circuits in different configurations, each serving a different purpose. The Central Processing Unit (CPU) contains the Arithmetic and Logic Unit (ALU) [6] which is a fundamental block that performs various arithmetic operations such as addition, subtraction, multiplication and logical operations, etc.

The fundamental unit of the arithmetic block is again the adder [21] which is modified to create subtractors and multipliers which efficiently perform more complex operations. Adders are also used in CPUs to fetch instructions, calculate addresses, table indices,and perform increment/decrement operations. Graphical Processor Units (GPU) are similar to CPUs, but the highly parallel structure makes them highly efficient when compared to general-purpose CPUs. These processors are mainly used in applications that require high-level manipulation of computer graphics and image processes [5] and are often found in embedded systems, personal computers, game consoles, and workstations.

Digital Signal Processor (DSP) is similar to a microprocessor with instruction sets which performs the following simple basic operations,

i. Arithmetic functions such as ADD, SUB, MUL, etc.

ii. Logic functions such as AND, OR, NOT, XOR, etc.

iii. Multiply and Accumulate function (MAC)

iv. Storage of immediate results using registers

v. Storage of signal samples and filter coefficients using on-chip memories

The adders are utilized in the Multiplier blocks, MACs, and Core ALU blocks in a Digital Signal Processors [1]-[3]. The basic block of a multiplier is an Adder. Parallel Multipliers (also called Array Multipliers) have replaced the traditional shift and add multipliers and takes only a single processor cycle to fetch, execute and store information. Another place where the adder is used is in the Multiply and Accumulate Unit. Most signal processing applications require a sum of products from a chain of continuous multiplications, for which the MAC unit is utilized. It comprises of a multiplier and a special register. It implements the function described in Eq (1.6).

$$A + BC \tag{1.6}$$

1.3 Reconfigurable Architecture

1.3.1 Run time Reconfiguration

Configuring an architecture at run-time or on-the-go is termed as run-time reconfiguration (or) dynamic reconfiguration (or) in-circuit reconfiguration. Reconfiguration allows a system to be modified during its normal operation, which means there is no need to reset the remaining circuitry or remove any reconfigurable elements for programming. In principle, any RAM or FLASH memory-based FPGAs [13] can be dynamically reconfigured to different configurations, even when the rest of the blocks are fully operational. The primary purpose of reconfiguration approach is to reduce component count and power consumption.

Run-time reconfiguration is primarily used in communication technologies such as multimedia signal processing, networking, and cryptography. Some examples being the creation of signal processing algorithms such as adaptive image filtering [10], retrieval of arbitrarily shaped objects within images or video frames [11], usage of FIR filters in re-

configurable FPGAs to store multiple lines of pixels [12] [13] and for implementing fractal image compression techniques [14]. Other applications include military-based Software radio, airborne applications, and remote sensors and for consumer applications such as cellphones, televisions, computers and gaming consoles.

1.3.2 Types of Implementation

There are two main approaches in run-time reconfigurations: total reconfiguration and partial reconfiguration. The FPGA resources are reconfigured or deleted between different configurations. Partial reconfiguration is the one in which only the variations between the configurations are modified.

1.3.3 Benefits of Run time Reconfiguration

A run-time reconfigurable system [13] offers the best performance with maximum hardware utilization. They offer the fastest way possible to change an active FPGA circuit since only those parts that require configuration are interrupted. This leads to quick system operation. The scope for employing dynamic reconfiguration is improved since the gate count of individual FPGA's continues to increase.

Current FPGA capabilities for dynamic reconfiguration is mainly applied in military and wireless communication technologies. When the capacity of programmable logic devices increases, the demand for the flexible re-use of FPGA is also increased. This leads to the advancements in various areas such as device configuration and speed of reconfiguration[15] [20]. The techniques also improve recovery and built-in error detection skills and even in the design simplicity of reconfigurable modes[18].

1.4 Research Motivation and Objective

1.4.1 Research Objective

The objective of this thesis is to design a High speed, energy-efficient reconfigurable adder architecture for Multimedia Signal Processing applications. A hybrid Carry Select Modified Tree (CSMT) [26] [27] based adder is utilized to implement this minimum area, low power architecture. Data paths are generally built using efficient reconfigurable components [15]-[18] [20]-[23] such as adders and multipliers which are used in media signal processors to compute the real-time audio/video signals through signal processing algorithms. Therefore, the need for high-speed adders is a necessity for the proper functioning of these multimedia signal processes.

1.4.2 Research Motivation

Semiconductor companies are working towards incorporating billions of transistors on a single chip due to a large number of functionalities it performs. Low-power design is a high demand field since modern-day consumers require electronic devices that have long-lasting battery life. The amount of power consumed for each task is reduced by compromising on other vital functionalities such as the speed and area. Therefore, the critical goal is the ability to meet the operational speeds while trying to reduce the rate of power dissipation.

The dream for every chip designer is to come up with designs that take up less power, runs at high speed, and has a lower cost. But it is nearly impossible to achieve all parameters at the same time since there is always a trade-off [23] [25] between each component. Consider the example of a Ripple Carry Adder, which is easier to design, but has a longer critical path which makes the design slower. Similarly, individual parallel prefix adders such as Kogge - Stone adder, Brent Kung style adders [15], Carry Propagate based Alti

Vec PowerPC architecture [35] reach higher speeds but has a substantial increase in silicon area and power consumption [18] [19].

Specific reconfigurable architectures [16] can also be designed to reduce the chip area by optimizing hardware utilization through sharing resources. These performance specifications can be attained by appropriate scaling of transistor sizes. Presently, technology scaling is moving to atomic dimensions. Proper scaling, along with innovative and competent techniques for implementing digital circuits, can assist in the reduction of silicon area and power consumption while maintaining an optimal operational speed.

There are many advancements undertaken for the implementation of high speed, low power adders. Multimedia devices [16] [17] such as cellphones, radios, tablets, computers, and game consoles utilize specific computation hungry signal processing algorithms such as discrete cosine transforms; inverse discrete cosine transforms, fast Fourier transforms, motion compensation, etc. to achieve real-time processing of media signals efficiently. The CMOS technologies are advancing rapidly; thus, circuit designers have to come up with practical reconfigurable computational elements such as adders and multipliers [36]-[38] which aid in the efficient functioning of these signal processing algorithms. Hence, to come up with high-speed, low-power media signal processors (MSP), adder circuits are required to be reconfigurable, consuming low power at high-speeds.

1.5 Thesis Organization

The rest of this thesis is organized as follows, Chapter 2 discusses the design features of the proposed 64-bit adder and its sub-components. The implementation of the 64-bit architecture and its subcomponents in the TSMC 180 nm CMOS technology and its corresponding results are presented in Chapter 3, followed by the conclusion and future work in chapter 4.

Proposed 64 - bit Reconfigurable Adder Architecture

2.1 Top-level Architectural Description

The Top-level description of the proposed 64-bit adder is similar to the original design [17]. It is made up of a series of non-uniform linearly increasing blocks of the following order: 1-bit, 3-bit, 4-bit, 5-bit, 6-bit, 7-bit, 8-bit, 9-bit, 10-bit and 11-bit with 1-bit being the Least Significant Bit (LSB) block and 11-bit being the Most Significant Bit (MSB) block. This adder is run-time reconfigurable and can be configured to perform one 64-bit addition, two 32-bit additions, four 16-bit additions or eight 8-bit additions,depending on the need.

This reconfiguration happens on demand and is made possible with the help of 2 partition signals (P0 and P1) that control the partitioning. Since the lower precision of operation is 8 bits, the least significant blocks (1-bit, 3-bit, and 4-bit) do not require the partition signals. When no partitioning is required it works like a conventional 64-bit carry skip adder. When partitioning is needed, the internal blocks are reconfigured to perform the necessary operation. The most significant blocks are reconfigured by assigning a particular value for each configuration, which is described in table 2.1.

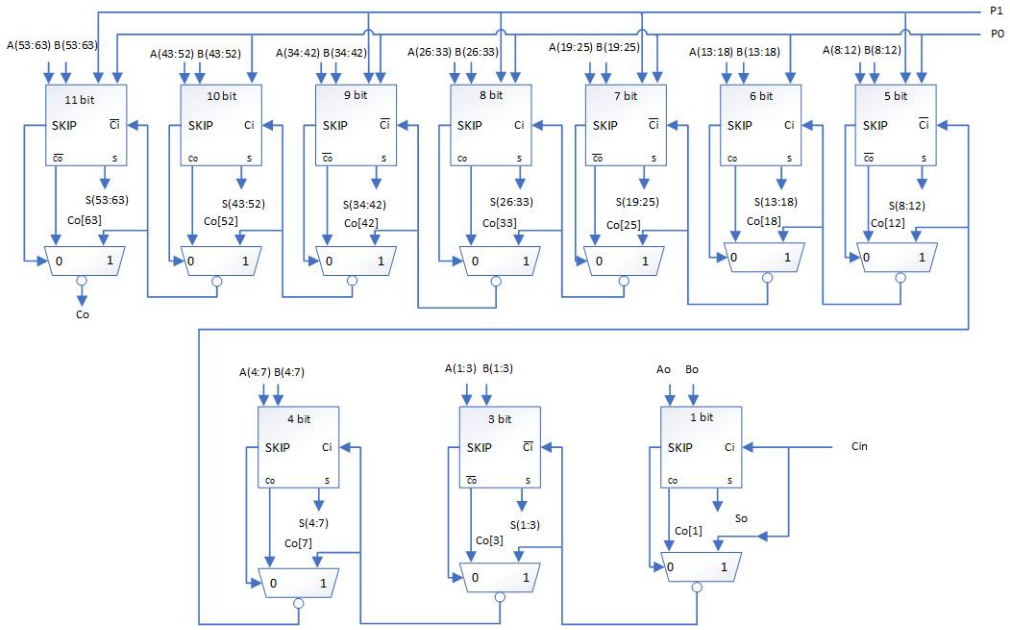


Figure 2.1: Block diagram of the proposed 64-bit CSMT based Adder Design

Table 2.1 lists each individual (x + y) block as two sub-blocks of bits (x bit, y-bit) with x-bit being the most significant and y-bit being the least significant. When the partition is required, the control signals (P0 and P1) ensures that no carry propagation occurs between these two separated sub-blocks. Take the example of an 8-bit addition partitioning, the 6-bit block is split in the third-bit position, and the 5-bit block is split in the first-bit position so that the carry from the previous blocks don't enter into this block.

Table 2.1: Description of the partition signal commands

Sub-blocks	P0 = 0 P1 = 0	P0 = 0 P1 = 1	P0 = 1 P1 = 0	P0 = 1 P1 = 1
	64-bit	32-bit	16-bit	8-bit
5-bit	Previous block carry	Previous block carry	Previous block carry	Original Cin
6-bit	-	-	(3-b, 3-b)	(3-b, 3-b)
7-bit	-	-	-	(2-b, 5-b)
8-bit	-	(2-b, 6-b)	(2-b, 6-b)	(2-b, 6-b)
9-bit	-	-	-	(3-b, 6-b)
10-bit	-	-	(5-b, 5-b)	(5-b, 5-b)
11-bit	-	-	-	(8-b, 3-b)

When observing the overall architecture Fig(2.1), it is noted that the odd number bit blocks except the 1-bit have an inverted carry coming in and an inverted carry going out, while the other blocks receive the normal carry in. This is due to the usage of inverted multiplexers (IMUX) for the carry skip process. The IMUX is a suitable choice over normal multiplexer & AND-OR structures since it helps to reduce the power and delay more efficiently.

2.1.1 Proposed Design Modification

Although the proposed design is functionally the same as the original one, a minor modification is made in the sub-component level of the new design. The original design uses XOR and XNOR gates to generate the sum and uses inverted 2:1 multiplexer (IMUX2) to create the carry. The proposed design uses a Carry Select Modified Tree (CSMT) Adder which uses multiplexers for both sum and carry generation, instead of using the XOR/XNOR and inverted multiplexers. This simplifies the circuit, eliminating the need for auxiliary signals, thereby reducing the cell count leading to a reduction in area. The architectural description of the CSMT adder used in this design is described in the following section.

2.1.2 Description of a Carry Select Modified Tree Adder

Figure 2.2 describes a bit slice of a multiplexer-based adder, which is constructed using the CSMT adder principle [27]. Consider an addition function, say

$$Y = A + B$$

Where, $Y = Y_{w-1}....Y_0$ and $A = A_{w-1}....A_0$ and $B = B_{w-1}....B_0$ represent the W-bit binary numbers. A carry-save adder can be implemented as a multiplexer-based adder. The

carry update recursion in this CSMT principle does not use the propagate (P) and Generate (G) signals. Instead, a new recording is defined in Eqs. (2.1) and (2.2) where,

$$a_{ir} = a_i.b_i \tag{2.1}$$

$$b_{ir} = a_i + b_i \tag{2.2}$$

This new equation rewrites (a_i, b_i) = (1,0) as (0,1) creating a dont care condition which reduces the complexity of the circuit and can be mapped to a multiplexer. Consider the recursive sum and carry generate Eqs. (2.3) and (2.4),

$$S = a_i \oplus b_i \oplus c_i \tag{2.3}$$

$$C_{i+1} = a_i b_i + a_i c_i + b_i c_i \tag{2.4}$$

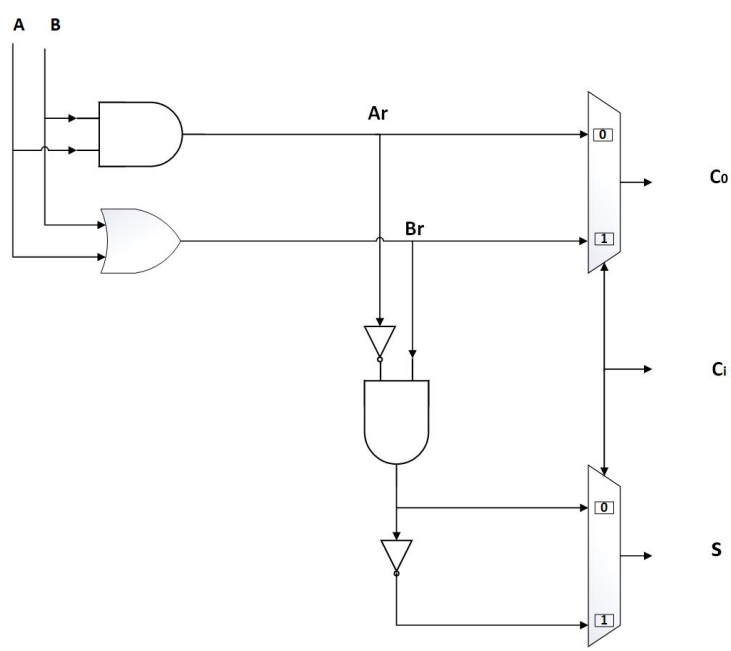


Figure 2.2: 1-bit CSMT based Adder Design

Table 2.2 provides the truth table description for the sum, carry, and recoding. Using this, we can come up with multiplexer-based equations for both the carry update and the sum operations given in Equations (2.5) and (2.6).

$$C_{i+1} = c_i.b_{ir} + \overline{c_i}.a_{ir} \tag{2.5}$$

$$S_{i+1} = ci(\overline{b_{ir}} + a_{ir}) + \overline{c_i}(\overline{a_{ir}} + b_{ir}) \tag{2.6}$$

Table 2.2: The truth table for Sum, Carry and Recoding of a 1-bit Adder

A_i	B_i	Recoding		C_{i+1}	S_i
		$A_{i,r}$	$B_{i,r}$		
0	0	0	0	0	c_i
0	1	0	1	ci	$\overline{c_i}$
1	0	0	1	ci	$\overline{c_i}$
1	1	1	1	1	c_i

2.2 Description of the Sub-components

The sub-blocks fall into two categories one that requires partitioning and the other that doesn't require partitioning. The partitioning is done based on the partition signals decoding table mentioned in table 3. The reconfigurable sub-blocks are designed in such a way as to avoid the insertion of additional circuitry in the carry propagation path when compared to the non-reconfigurable blocks.

2.2.1 Design of the Least Significant Blocks

The 1-bit, 3-bit, and 4-bit blocks do not require reconfiguration since the lowest bit operation is 8-bit, that is (1 + 3 + 4 = 8). The design of these blocks is simple and straight

forward. Figures 2.3 and 2.4 show the schematics of the 1- and 4-bit blocks, which are coded using VHSIC Hardware Description Language (VHDL).

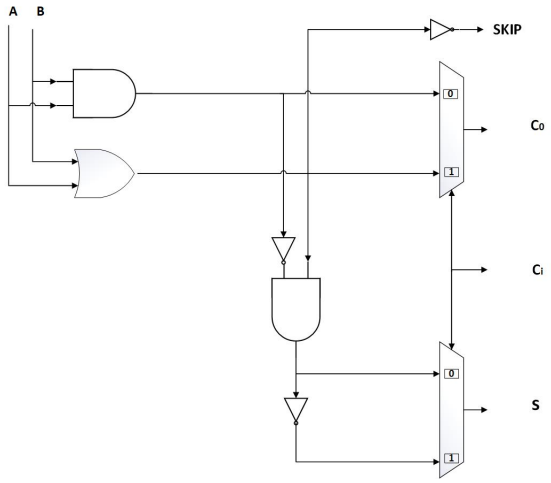


Figure 2.3: 1-bit Sub-block Design

The working principle of the 1-bit block is described as follows. The inputs, a, b and cin, are given to the block, according to the updated multiplexer- based equations, the carry-in bit acts as the select line for both the sum and carry out bit. The re-coded inputs are utilized to produce the sum and carry. When extended to an N-bit block say 3-bit or 4-bit, the carry-out bit of the previous block becomes the select line input for the next block as seen in Figure 2.4. The skip signal is used as a select line for the inverted multiplexer in the carry skip circuit to skip using the carry-out signal generated from the present block to the next block.

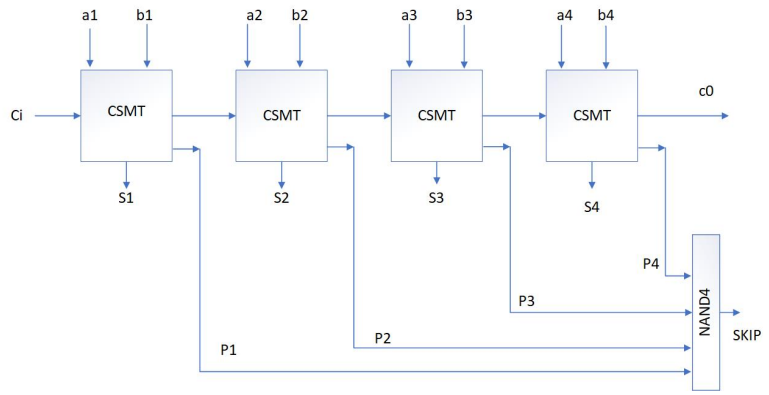


Figure 2.4: 4-bit Sub-block Design

2.2.2 Design of the Most Significant Blocks

The design of the most significant blocks is more important since it involves the partition process. This most significant blocks can be further divided into three categories depending on the partition command equations.

i. 6-bit and 10-bit designs partitioning is needed only for 8- or 16-bit addition

ii. 5-bit,7-bit,9-bit, and 11-bit designs partition required only for 8-bit addition

iii. 8-bit design partitioning necessary for 8-, 16- and 32-bit additions

i. 6-bit and 10-bit designs

The 6-bit and 10-bit designs use a simple inverter and a multiplexer to perform the partitioning. The 6-bit design is divided into (3-b,3-b) sub-block and the 10-bit design is divided into (5-b,5-b) sub-block after decoding the partition signal $PAR = \overline{P0}$. Depending on the arrival of the partitioned signal, the carry-in from the previous block is either passed to the current block, or by-pass to the next block. Both the designs are described as a structural VHDL code. Figure 2.5 shows the 6-bit design. Both 6 and 10-bit designs are constructed similarly.

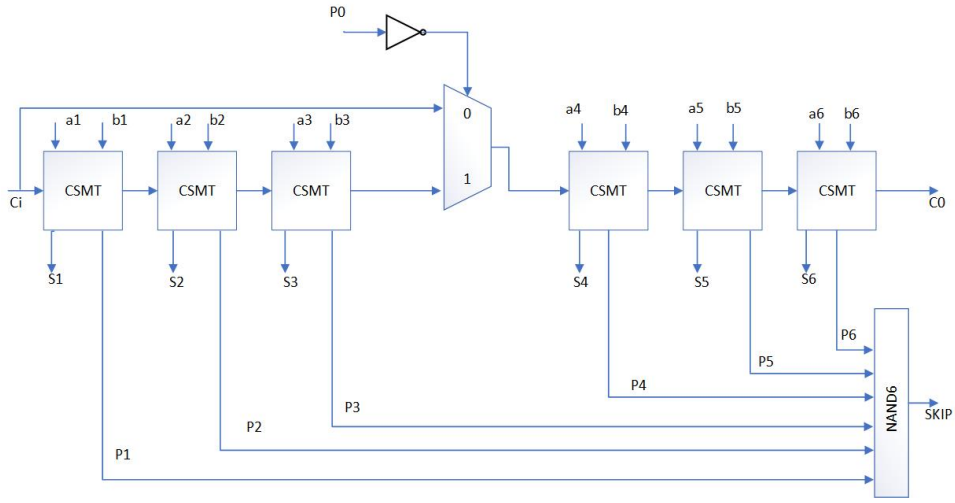


Figure 2.5: 6-bit Sub-block Design

ii. 5-bit, 7-bit, 9-bit, and 11-bit designs

The 5-bit, 7-bit, 9-bit, and 11-bit blocks require partitioning only when it performs an 8-bit addition operation. Therefore, a NAND gate along with a multiplexer is used to decode the partition command $PAR = \overline{P0.P1}$. The 5-bit block is split at the 1st-bit position since it should not receive the carry from the previous 4-bit block during an 8-bit addition partition. The 7-bit block is split into (2-b, 5-b) sub-block, the 9-bit block is split into (3-b, 6-b) sub-block and the 11-bit block is split into (8-b,3-b) sub-blocks. The schematics of the 5-bit and 9-bit blocks are shown in Figures 2.6 and 2.7.

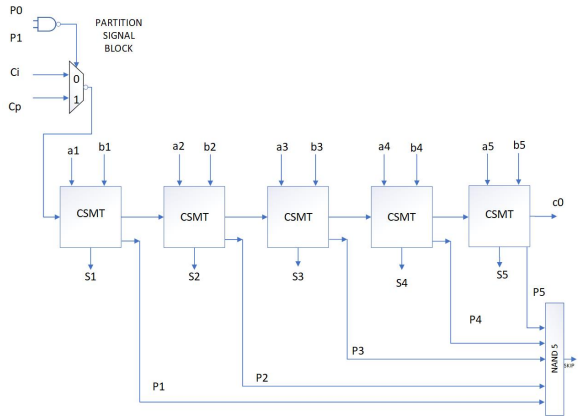


Figure 2.6: 5-bit Sub-block Design

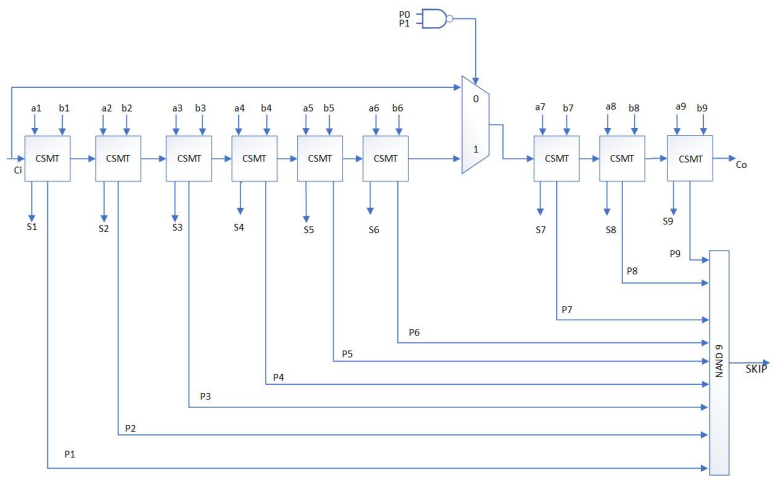


Figure 2.7: 9-bit Sub-block Design

iii. 8-bit design

The 8-bit design requires constant partitioning unless it performs the conventional 64-bit addition. A NOR gate is used along with a multiplexer to decode the partitioning command $PAR = \overline{P0 + P1}$. The block is split into (2-b,6-b) sub-blocks, as shown in Figure 2.8.

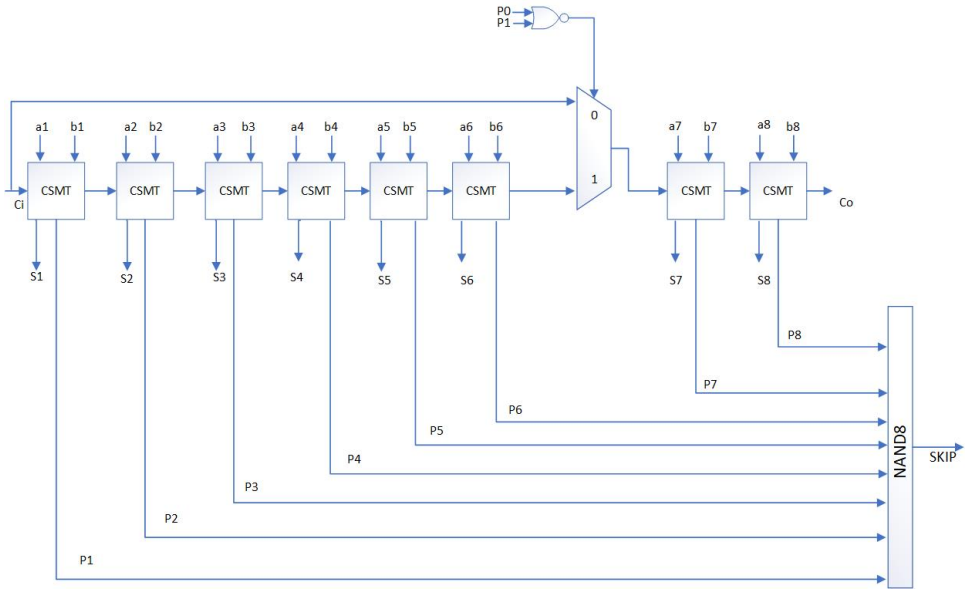


Figure 2.8: 8-bit Sub-block Design

Design Implementation in 180 nm CMOS Technology

Chapter 2 discussed the design process of the 64-bit architecture and its sub-components in detail. The basic schematics were translated into a structural VHDL code (RTL). The VHDL codes were inputted into the Cadence simulation tool (NC-Sim) for checking and functional verification, and after which the fault analysis is completed using the Synopsys TetraMAX software.

The next step is to synthesize and optimize the design using Synopsys Design Compiler. Synthesis is the process of translating an RTL design into a generic design (GTECH library), and after which it is optimized and mapped to its target technology (here 180 nm CMOS technology) through the process of compiling and the reports for the area, power and timing are generated. Detailed analysis of all the results will be explained in section 3.3. The implementation process of the sub-blocks and the entire architecture will be described in sections 3.1 and 3.2.

3.1 Implementation of the Sub-components

3.1.1 Implementation of the Least Significant Blocks

The implementation process begins with verifying the functionality of the VHDL codes to make sure that the sub-blocks are functioning the way test benches with various test cases are designed. The next step involves checking the circuits for fault analysis using Synopsys TetraMAX, where it provides a list of fault models that simulates logical problems that might occur in the circuit namely stuck-at faults, bridging faults, transistor faults, open circuit faults and so on.

The most common fault model taken for consideration is the stuck-at fault model, which means that a wire in the circuit can always be stuck at signal 0' or 1'. Once the reports for each of the circuit is generated, the TetraMAX ATPG software provides several test patterns which can be used to detect most of the listed faults. Once this is complete, the sub-components are loaded into the Synopsys Design Compiler to synthesize the design and map them to the target technology for further analysis. Figures 3.1, 3.2, and 3.3 show the functional verification results for the 1-bit, 3-bit, and 4-bit blocks.

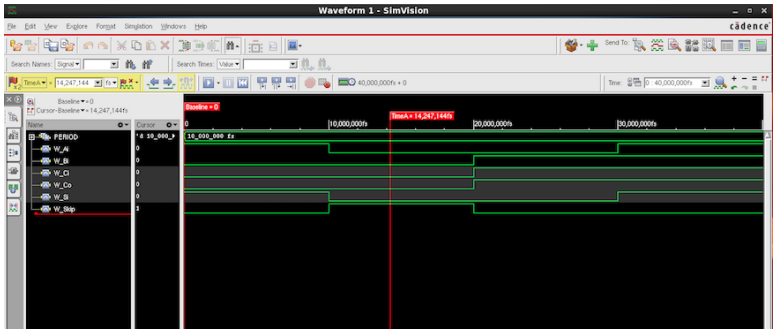


Figure 3.1: Functional Verification of 1-bit block

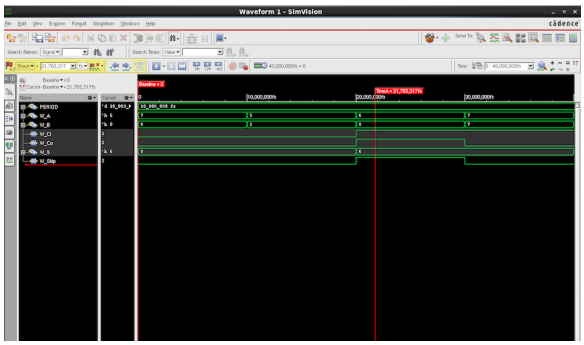


Figure 3.2: Functional Verification of 3-bit block

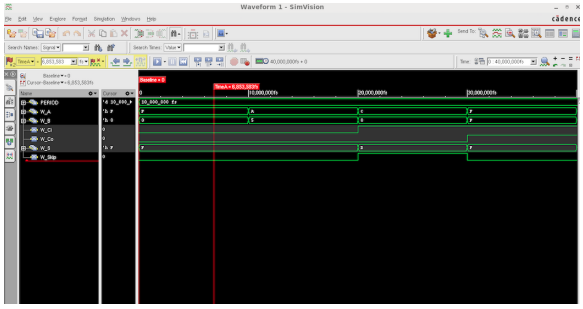


Figure 3.3: Functional Verification of 4-bit block

3.1.2 Implementation of the Most Significant Blocks

The functional verification of the most significant sub-blocks is highly essential since these components involve the partitioning of bits within each sub-block. The test benches are written accordingly, to verify a few test cases. Once the functionality is verified, the next few steps are the same as described in sub-section 3.1.1. The functional verification results for the most significant blocks are shown below in figures 3.4 to 3.10.

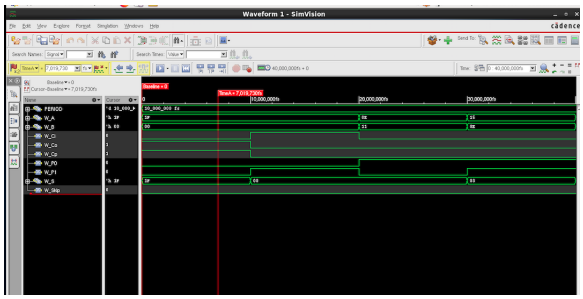


Figure 3.4: Functional Verification of 5-bit block

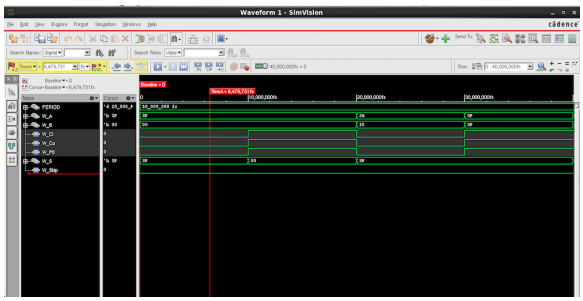


Figure 3.5: Functional Verification of 6-bit block

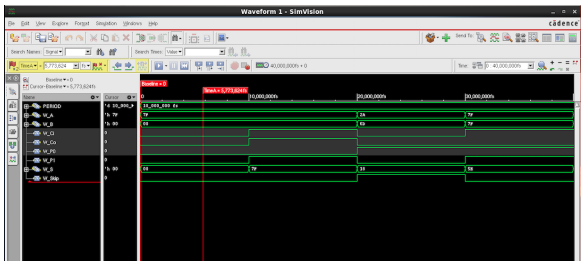


Figure 3.6: Functional Verification of 7-bit block

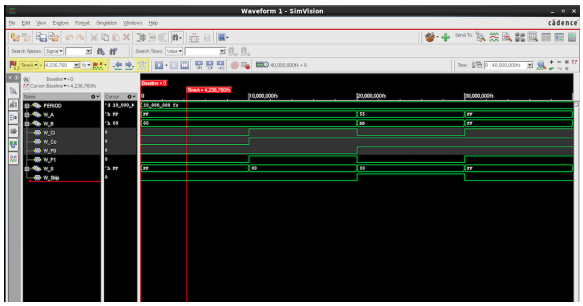


Figure 3.7: Functional Verification of 8-bit block

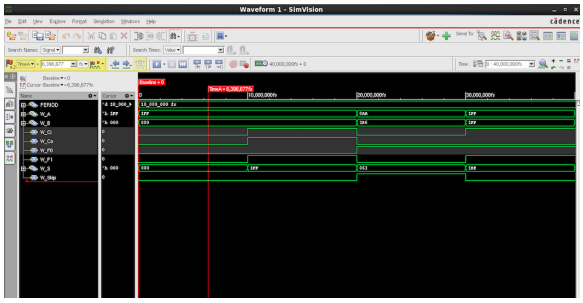


Figure 3.8: Functional Verification of 9-bit block

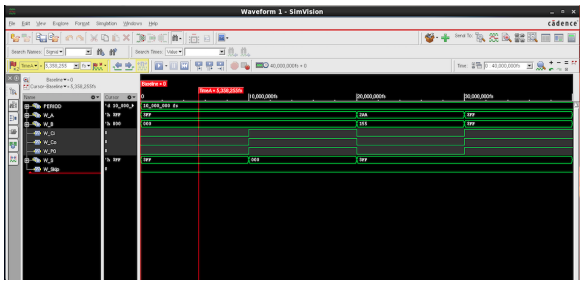


Figure 3.9: Functional Verification of 10-bit block

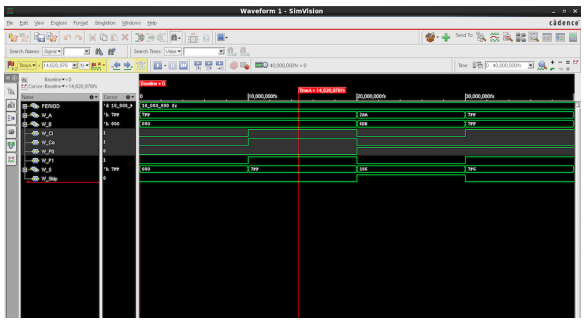


Figure 3.10: Functional Verification of 11-bit block

3.2 Implementation of the 64-Bit Architecture

Once the sub-blocks are verified, the entire 64- bit adder is tested to make sure it runs properly for all the partition signal commands (P0 and P1), for each reconfiguration 8-, 16-, 32- and 64-bit additions. The test cases and their functional verification waveform for each of the partition commands are generated.The design is then loaded onto the Synopsys TetraMAX ATPG tool, which generates a report on the number of faults, location of the faults and a test pattern report for detecting these faults so that it can be fixed.

The next step involves the Synopsys Design Compiler tool which translates the RTL code into the gate-level netlist (GTECH library) and then mapped and optimized to the target 180 nm CMOS technology library. These library files have a list of logic gate descriptions, timing, and power specifications in accordance with that particular technology.

Once the data are analyzed and elaborated, the hierarchy of the circuit is checked to make sure that it is the same as all the other schematics generated in each of the EDA software that is used. Once that has been done, the design is compiled during which the mapping and optimization take place. The final step in this process is to generate the area, power, and timing reports for this design. A detailed analysis of all the results is provided in the upcoming section.

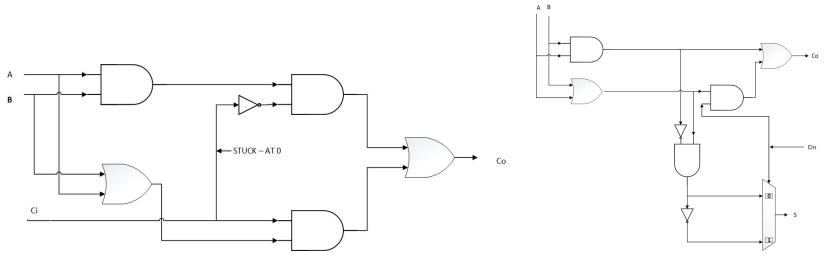
3.3 Results and Discussion

3.3.1 Result Analysis of the Sub-components

i. Fault Analysis

The stuck-at fault analysis is completed for each of the sub-components using TetraMAX ATPG software to generate a corresponding test pattern file listing test patterns to detect the faults found by the software. The undetected redundant (UR) faults can be removed by removing the redundant connections or gates to make the circuits 100% fault-free. The primary fault location of the CSMT based adder [27] used in this design is in the input Ci going to the multiplexer of the Co (through the inverter and input B of the AND gate).

The fault location is shown in Figure 3.11(a). This fault can be removed by removing both the logic gates in the circuit, which leads rise to a fault-free design shown in Figure 3.11(b). Since each sub-block has the same fault, which is rippling through each bit configuration, it can be removed by fixing the fault in the primary 1-bit slice.



(a) Fault location

(b) Modified, Fault-free Circuit

Figure 3.11: Location of the fault and the modified circuit

ii. Area

The table 3.1 below provides a detailed analysis of the area comparison between the CSMT based sub-block designs and the original sub-block designs. When we observe the values in the table, the 1-bit and 3-bit original design sub-blocks take up lesser area than the proposed design, but as we go down the table, the trend changes and the area occupied by the original design sub-blocks increases sharply when compared to the proposed sub-block designs. This trend can be observed in the following graphical representation of the table in Figure 3.12.

Table 3.1: Area of each sub-block for both original and proposed designs

Sub-blocks	CSMT based 64-bit adder (μm^2)	Original 64-bit adder (μm^2)
1-bit	582.83	368.93
3-bit	1794.43	1541.69
4-bit	2386.45	2317.84
5-bit	3051.95	3371.05
6-bit	3810.77	4235.01
7-bit	4439.50	5020.15
8-bit	5031.33	5768.38
9-bit	5660.32	6581.45
10-bit	6252.34	7413.12
11-bit	6881.10	8193.34

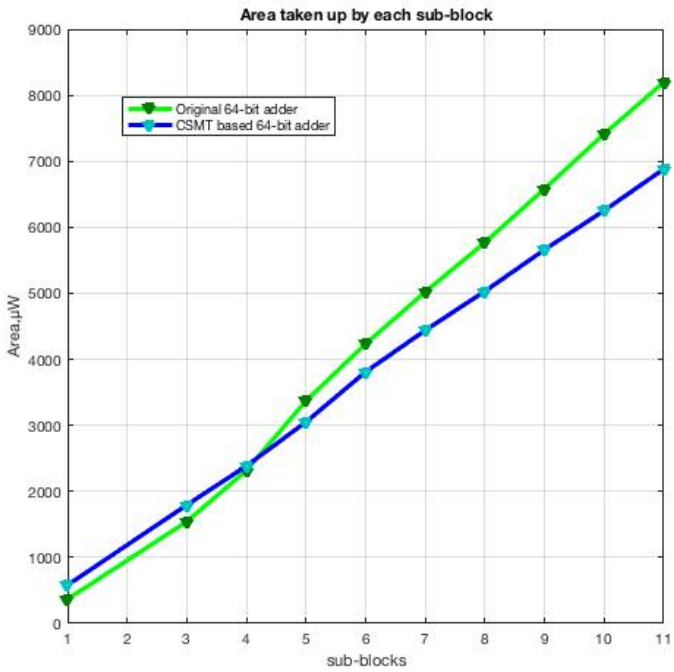


Figure 3.12: Area comparison of Original and proposed design sub-blocks

iii. Power

The power consumed by each sub-block is presented in table (3.2), and it is observed that in the initial stages, the original sub-blocks seem to be consuming lesser power than the proposed design, but in the later stages there is a shift in the observed pattern and the power consumption of the original design increases when compared to the proposed sub-block designs. This can be seen clearly in the following graph in Figure 3.13.

Table 3.2: Power consumption of each sub-block for both original and proposed designs

Sub-blocks	CSMT based 64-bit adder (μW)	Original 64-bit adder (μW)
1-bit	5.82	8.45
3-bit	17.38	27.31
4-bit	24.13	42.37
5-bit	30.27	57.28
6-bit	39.27	96.75
7-bit	42.52	122.02
8-bit	49.59	142.16
9-bit	57.20	153.22
10-bit	62.02	159.02
11-bit	65.12	192.36

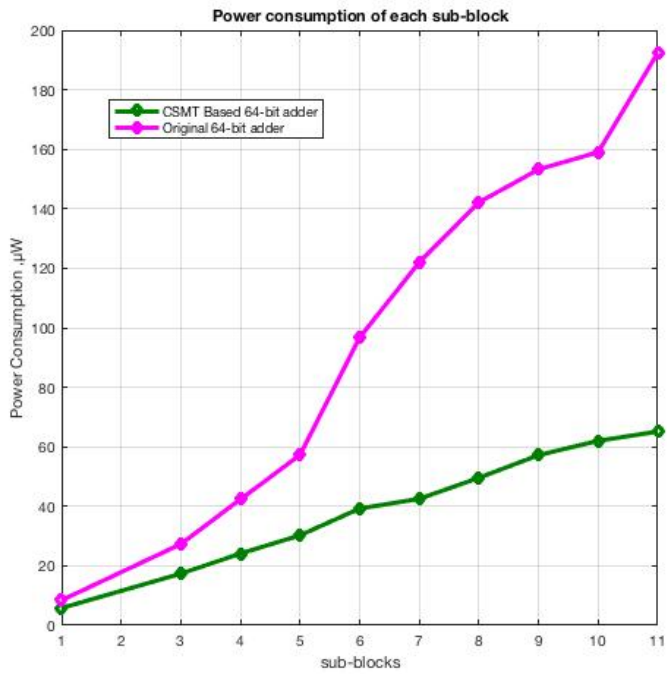


Figure 3.13: Power comparison of Original and proposed design sub-blocks

iv. Time

Tables (3.3) and (3.4) show the data arrival times from the carry-in bit to the corresponding sub-block carry output for both the original design and the proposed design. It is shown that the proposed design achieves a timing delay more or less close to the original

design. The graph shown in Fig. (3.14) gives a clear comparison of speeds between both the designs.

Table 3.3: 64 Bit Reconfigurable Original design Data Arrival Time VS Bit Configuration

Bit Configuration	Source	Destination	Data Arrival time (nS)
1 Bit	Ci	Net C1	0.209
4 bits	Ci	Net C3	1.328
8 bits	Ci	Net C7	2.628
12 bits	Ci	Net G6/U20/Y	4.152
16 bits	Ci	Net G8/U11/Y	5.513
24 bits	Ci	Net G10/U25/Y	7.812
32 bits	Ci	Net G12/U32/Y	10.280
48 bits	Ci	Net G16/U25/Y	14.956
56 bits	Ci	Net G18/U11/Y	17.254
64 bits	Ci	Co	19.532
64 bits (Worst case)	B [0]	Co	19.819

Table 3.4: 64 Bit Reconfigurable CSMT design Data Arrival Time VS Bit Configuration

Bit Configuration	Source	Destination	Data Arrival time (nS)
1 Bit	Ci	Net C1	0.237
4 bits	Ci	Net C3	1.439
8 bits	Ci	Net C7	2.879
12 bits	Ci	Net G6/G5/Co	4.684
16 bits	Ci	Net G8/G2/Co	6.132
24 bits	Ci	Net G10/G4/Co	8.988
32 bits	Ci	Net G12/G5/Co	11.839
48 bits	Ci	Net G16/G4/Co	14.732
56 bits	Ci	Net G18/G2/Co	17.608
64 bits	Ci	Co	20.465
64 bits (Worst case)	B [0]	Co	20.675

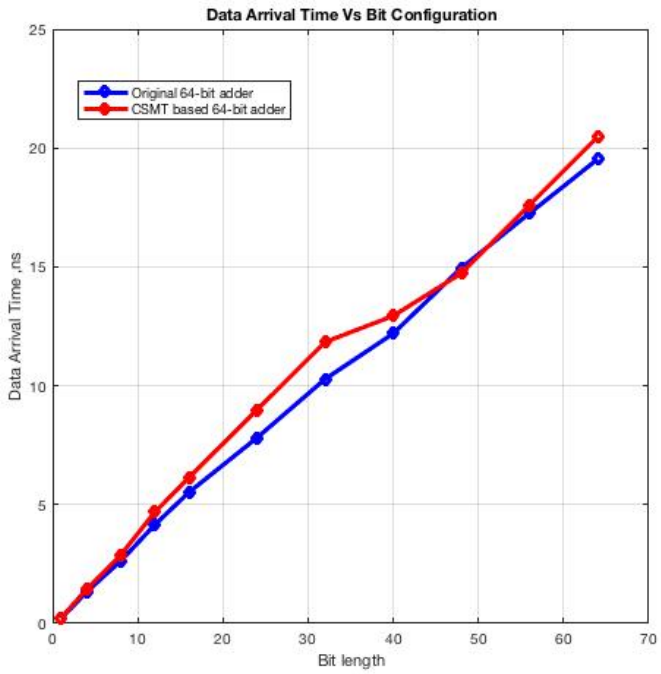


Figure 3.14: Timing Comparison of Original and Proposed 64-Bit Designs

3.3.2 Result Analysis of the 64-Bit Architecture

i. Fault Analysis

The fault analysis of 64-bit adder circuit is performed similarly as with the individual sub-blocks. The Fault analysis report for the entire 64-bit architecture is shown below. It provides a detailed description of the number of faults that are Detected(DT), Undetected(UD), Not Detected(ND), Possibly Detected(PT) or ATPG Untestable(AU). The fault removal process is done using the same steps, which are described in the sub-component fault analysis.

```
 Uncollapsed Transition Fault Summary Report
-----------------------------------------------
fault class                     code   #faults
------------------------------  ----  ---------
Detected                         DT       5786
  detected_by_simulation         DS      (5786)
Possibly detected                PT          0
Undetectable                     UD        210
  undetectable-tied              UT        (4)
  undetectable-redundant         UR      (206)
ATPG untestable                  AU          0
Not detected                     ND          0
-----------------------------------------------
total faults                              5996
test coverage                           100.00%
fault coverage                           96.50%
ATPG effectiveness                      100.00%
-----------------------------------------------
           Pattern Summary Report
-----------------------------------------------
#internal patterns                          50
    #basic_scan patterns                    50
-----------------------------------------------
```

Figure 3.15: Fault Report for Proposed 64-Bit Design

ii. Area

The area report of the 64-bit design which was generated by the Synopsys Design Compiler is shown below in Fig 3.16. It gives a detailed analysis of the number of cells used to construct the design, the number of buffers/inverters used in the design, the number of sequential components (registers, flip flops), and combinational components used and the total area of the design.

```
****************************************
Report : area
Design : CSMT64
Version: O-2018.06-SP5-3
Date   : Thu Jul 11 12:47:40 2019
****************************************

Information: Updating design information... (UID-85)
Library(s) Used:

    vtvt_tsmc180 (File: /home/w093pxa/libs/vtvt_tsmc180.db)

Number of ports:                          4191
Number of nets:                           4977
Number of cells:                          2127
Number of combinational cells:             851
Number of sequential cells:                  0
Number of macros/black boxes:                0
Number of buf/inv:                         198
Number of references:                       20

Combinational area:               36417.485886
Buf/Inv area:                      5499.430286
Noncombinational area:                0.000000
Macro/Black Box area:                 0.000000
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:                  36417.485886
Total area:                 undefined
```

Figure 3.16: Area Report for Proposed 64-Bit Design

iii. Power

The power analysis can be done by generating a report from the Synopsys Design Compiler software. This report provides detailed information about the percentage of dynamic power, internal power, switching cell power, and leakage power of the given 64-bit adder circuit shown in Fig. 3.17. It also provides a spilt up of the power used by various components in the circuit (clock network, registers, sequential and combinational elements, etc.)

```
****************************************
Report : power
        -analysis_effort high
        -verbose
Design : CSMT64
Version: P-2019.03-SP3
Date   : Thu Aug  1 12:38:40 2019
****************************************


Library(s) Used:

    vtvt_tsmc180 (File: /home/w093pxa/libs/vtvt_tsmc180.db)


Operating Conditions: nom_pvt   Library: vtvt_tsmc180
Wire Load Model Mode: top


Global Operating Voltage = 1.8
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000ff
    Time Units = 1ps
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1mW


  Cell Internal Power  = 294.2890 uW   (66%)
  Net Switching Power  = 153.6491 uW   (34%)
                         ---------
Total Dynamic Power    = 447.9381 uW  (100%)

Cell Leakage Power     =   5.9723 nW

Information: report_power power group summary does not include estimated clock tree power. (PWR-789)
                 Internal         Switching           Leakage            Total                          Cell
Power Group      Power            Power               Power              Power   (   %    )  Attrs   Count
--------------------------------------------------------------------------------------------------------
io_pad             0.0000           0.0000            0.0000            0.0000  (   0.00%)            0
memory             0.0000           0.0000            0.0000            0.0000  (   0.00%)            0
black_box          0.0000           0.0000            0.0000            0.0000  (   0.00%)            0
clock_network      0.0000           0.0000            0.0000            0.0000  (   0.00%)            0
register           0.0000           0.0000            0.0000            0.0000  (   0.00%)            0
sequential         0.0000           0.0000            0.0000            0.0000  (   0.00%)            0
combinational      0.2943           0.1536        5.9723e-06            0.4479  ( 100.00%)          281
--------------------------------------------------------------------------------------------------------
Total              0.2943 mW        0.1536 mW     5.9723e-06 mW         0.4479 mW
```

Figure 3.17: Power Report for Proposed 64-Bit Design

3.3.3 Result Comparison with the Original Design

The proposed 64-bit reconfigurable adder circuit and the original 64-bit reconfigurable adder circuit were both synthesized and optimized using Synopsys Design Compiler with the 180 nm CMOS technology for uniform comparison. The results of the comparison are provided in table (3.5).

Table 3.5: Area, Power, and Timing results of both the original and proposed design

Parameters	CSMT based 64-bit adder	Original 64-bit adder
Area (μm^2)	36417	46851
Power (μW)	447.93	845.67
Timing (ns)	20.67	19.84

From the above table, it is observed that the proposed 64-bit reconfigurable adder design has a 22% reduction in area, 47% reduction in power consumption and a slight 4% increase in delay when compared with the original 64-bit reconfigurable adder architecture.

Conclusion and Future Work

4.1 Conclusion

This thesis presented a low area, energy-efficient, high-speed 64-bit reconfigurable adder architecture, which can be run-time reconfigured to perform either eight 8-bits addition or four 16-bits addition or two 32-bits addition. It is used as a data path for multimedia signal processing applications. The proposed architecture utilized the carry select modified tree (CSMT) - based adder to design the sub-blocks. All the designs were coded using VHSIC Hardware Description Language (VHDL) and verified for proper functionality, tested for faults and optimized and synthesized with various design constraints for better performance. This design achieved a 22% decrease in area and a 47% reduction in power consumption with a slight 4% increase in delay when compared to the existing architecture. This design was implemented in 180 nm CMOS technology at 1.8-volt supply at a temperature of 25C to obtain these results.

4.2 Future Work

i. Improved high-speed media signal processing algorithms such as DCT, IDCT, FFT and

ii. high-speed, low-power, low-area multipliers can be implemented based on the proposed adder.

iii. Carry propagation reduction approaches can be further analyzed and applied for further delay reduction in the design.

iv. Process-voltage-temperature variations can be analyzed to test the robustness of the design by conducting a Monte Carlo analysis to determine the best-case scenario.

REFERENCES

[1] Gowthaman, Naveen Balaji & Deni Johnson, S & Giridharan, S & Aswanth, R. (2017). Approximate Adders for digital signal processing (DSP) applications A relative study.

[2] N. Zhu, W. L. Goh, W. Zhang, K. S. Yeo, and Z. H. Kong, ”Design of Low-Power High-Speed Truncation-Error-Tolerant Adder and Its Application in Digital Signal Processing,” in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 18, no. 8, pp. 1225-1229, Aug. 2010. doi:10.1109/TVLSI.2009.2020591

[3] V. Gupta, D. Mohapatra, A. Raghunathan and K. Roy, ”Low-Power Digital Signal Processing Using Approximate Adders,” in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 32, no. 1, pp. 124-137, Jan. 2013.doi: 10.1109/T-CAD.2012.2217962

[4] M. Pashaeifar, M. Kamal, A. Afzali-Kusha and M. Pedram, ”Approximate Reverse Carry Propagate Adder for Energy-Efficient DSP Applications,” in IEEE Transactions on Very Large-Scale Integration (VLSI) Systems, vol. 26, no. 11, pp. 2530-2541, Nov. 2018. doi: 10.1109/TVLSI.2018.2859939

[5] A. Madanayake et al., ”Low-Power VLSI Architectures for DCTDWT: Precision vs. Approximation for HD Video, Biomedical, and Smart Antenna Applications,” in IEEE Circuits and Systems Magazine, vol. 15, no. 1, pp. 25-47, First quarter 2015. doi: 10.1109/MCAS.2014.2385553

[6] P. Nautiyal, P. Madduri and S. Negi, ”Implementation of an ALU using modified carry select adder for low power and area-efficient application,” in International Conference on Computer and Computational Sciences (ICCCS), Noida, India, 2015.

[7] Audio Signal Processing for Next-Generation Multimedia Communication Systems, Yiteng (Arden) Huang, Jacob Benesty, Springer Science & Business Media, 2007, ISBN: 1402077696, 9781402077692, 374 pages

[8] https://elearningatria.files.wordpress.com/2013/10/ece-vii-dsp-algorithms-architecture-10ec751-notes.pdf

[9] http://ptgmedia.pearsoncmg.com/images/chap3_013031398X/elementLinks /3139H.pdf

[10] N. Srivastava, J. L. Trahan, R. Vaidyanathan, and S. Rai, ”Adaptive image filtering using run-time reconfiguration,” Proceedings International Parallel and Distributed Processing Symposium, Nice, France, 2003, pp. 7 pp.-. doi: 10.1109/IPDPS.2003.1213332

[11] J. Gause, P. Y. K. Cheung and W. Luk, ”Reconfigurable shape-adaptive template matching architectures,” Proceedings. 10th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, Napa, CA, USA, 2002, pp. 98-107. doi: 10.1109/FPGA.2002.1106665

[12] R. Kreuger, ”Virtex-EM FIR Filter for Video Applications,” Xilinx application note XAPP241 Xilinx Inc., 2000.

[13] J. S. N. Jean, K. Tomko, V. Yavagal, J. Shah and R. Cook, ”Dynamic reconfiguration to support concurrent applications,” in IEEE Transactions on Computers, vol. 48, no. 6, pp. 591-602, June 1999. doi: 10.1109/12.773796

[14] H. Nagano, A. Matsura, A. Nagoya, ”An Efficient Implementation Method of Fractal Image Compression on Dynamically Reconfigurable Architecture,” Proc. 6th Reconfig. Arch. Workshop (Parallel and Distributed Processing; Lect. Notes Comp. Sci. #1586), pp. 670-678, 1999.

[15] C. K. V., S. P. P., S. E. Ahmed, S. Veeramachaneni, N. M. Muthukrishnan, and M. B. Srinivas, ”A Prefix Based Reconfigurable Adder,” 2011 IEEE Computer Society Annual Symposium on VLSI, Chennai, 2011, pp. 349-350.doi: 10.1109/ISVLSI.2011.69

[16] S. Perri, P. Corsonello and G. Cocorullo, ”64-bit reconfigurable adder for low power media processing,” in Electronics Letters, vol. 38, no. 9, pp. 397-399, 25 April 2002.doi: 10.1049/el:20020295

[17] Perri Stefania, Corsonello Pasquale, Cocorullo Giuseppe, ”A high-speed energy-efficient 64-bit reconfigurable binary adder”, IEEE Trans. VLSI Systems, vol. 11, no. 5, pp. 939-943, Oct. 2003.

[18] A.A. Farooqui, V. Oklobdzija, E Chechrazi, ”64-bit media adder”, IEEE Int. Symp. on Circuits and Systems, May 1999.

[19] Jin-Fu Li, Jiunn-Der Yu, Yu-Jen Huang, ”A Design Methodology for Hybrid Carry-lookahead/Carry-Select Adders with Reconfigurability,” IEEE International Symposium on Circuits and Systems, vol. 1, pp. 77-80, May 2005.

[20] Kumar, C. N. Vijay, and Sagara Pandu. ” Design and Implementation of CVNS Based Low Power 64-Bit Adder”(2013).

[21] B. K. Mohanty and S. K. Patel, ”AreaDelayPower Efficient Carry-Select Adder,” in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 61, no. 6, pp. 418-422, June 2014. doi:10.1109/TCSII.2014.2319695

[22] Yajuan He, Chip-Hong Chang and Jiangmin Gu, ”An area efficient 64-bit square root carry-select adder for low power applications,” 2005 IEEE International Symposium on Circuits and Systems, Kobe, 2005, pp. 4082-4085 Vol. 4. doi:10.1109/ISCAS.2005.1465528

[23] V. Benara and S. Purini, ”Accurus: A Fast Convergence Technique for Accuracy Configurable Approximate Adder Circuits,” 2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Pittsburgh, PA, 2016, pp. 577-582. doi: 10.1109/ISVLSI.2016.58

[24] B. Ramkumar and H. M. kittur, ”Low-power and Area-Efficient Carry Select Adder,” in IEEE Transaction on Very Large Scale Integration (VLSI) Systems, 2012.

[25] R. Abhilash, S. Dubey and M. Chinnaiah, ”ASIC design of signed and unsigned multipliers using compressors,” in International Conference on Microelectronics, Computing and Communications (MicroCom), Durgapur, 2016.

[26] K. K. Parhi, ”Low-energy CSMT carry generators and binary adders,” in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 7, no. 4, pp. 450-462, Dec. 1999. doi: 10.1109/92.805752

[27] K. K. Parhi, ”Comments on ”Low-energy CSMT carry generators and binary adders,” in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 21, no. 4, pp. 791-791, April 2013. doi: 10.1109/TVLSI.2012.219077

[28] http://www.theorycircuit.com/half-adder-circuit-diagram/

[29] http://www.theorycircuit.com/full-adder-circuit-diagram/

[30] https://www.gatevidyalay.com/ripple-carry-adder/

[31] http://verilogcodes.blogspot.com/2017/11/verilog-code-for-carry-select-adder.html

[32] http://fourier.eng.hmc.edu/e85_old/lectures/arithmetic_html/node7.html

[33] https://www.researchgate.net/figure/4-bit-carry-bypass-adder_fig2_259590209

[34] https://www.mikroe.com/ebooks/digital-filter-design/introduction-fir-filter

[35] M. S. Schmookler, M. Putrino, C. Roth, M. Sharma, A. Mather, J. Tyler, H. Van Nguyen, M. N. Pham, and J. Lent, A low-power, high-speed implementation of a PowerPC microprocessor vector extension, in Proc. 14th Arithmetic Conf., Adelaide, Australia, Apr. 1999.

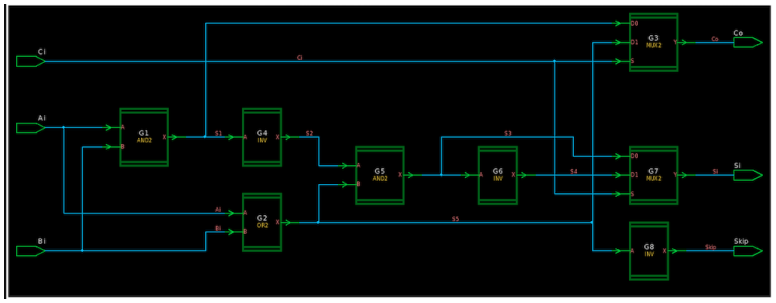
[36] A. Peleg and U. Weiser, MMX technology, IEEE Micro, vol. 16, pp. 4250, Aug. 1996.

[37] R. B. Lee, Subword parallelism with MAX-2, IEEE Micro, vol. 16, pp. 5159, Aug. 1996.

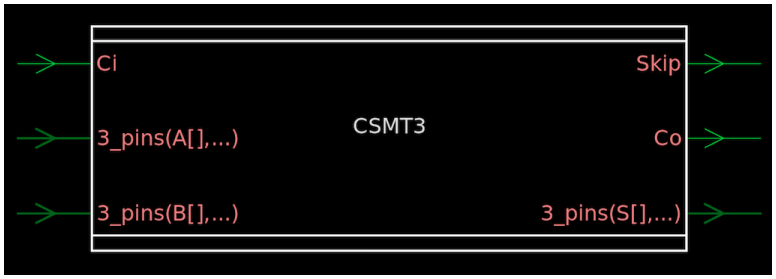
[38] M. Tremblay, J. M. OConnor, V. Narayanan, and H. Liang, VIS speeds new media processing, IEEE Micro, vol. 16, pp. 1020, Aug. 1996.

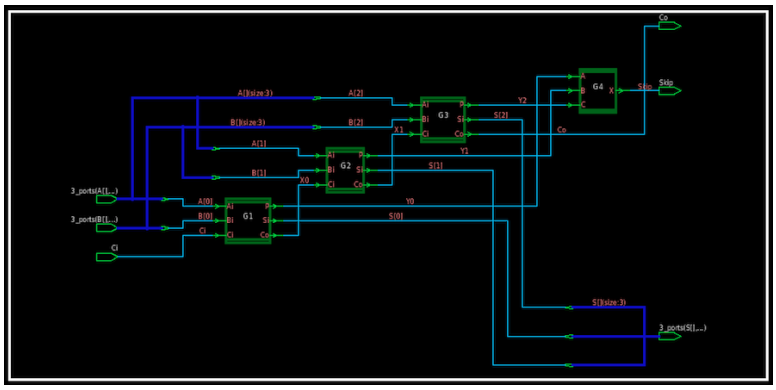
BLOCK DIAGRAMS GENERATED BY SYNOPSYS DESIGN COMPILER

Schematic of a 1-bit block and its Hierarchy

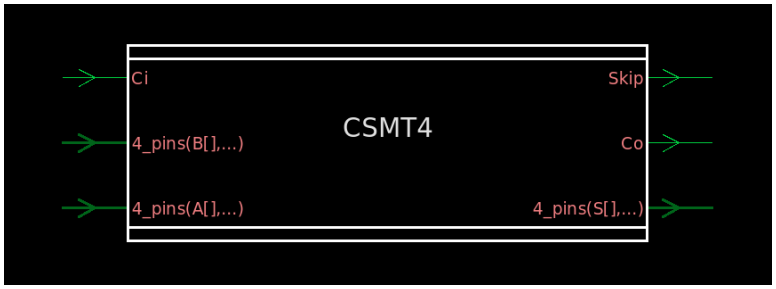


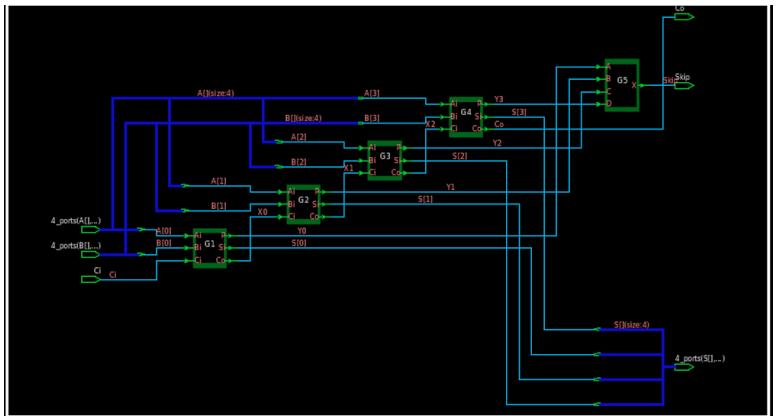
Schematic of a 3-bit block and its Hierarchy



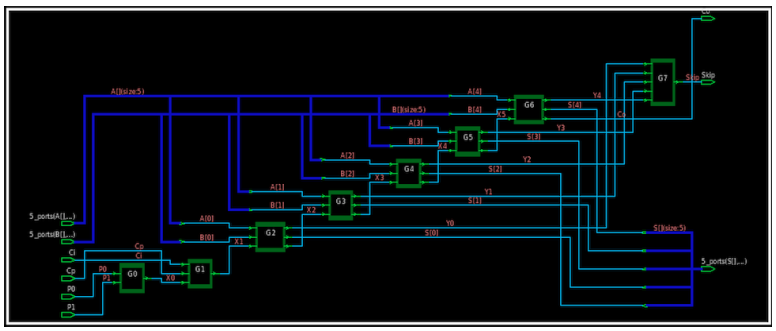


Schematic of a 4-bit block and its Hierarchy

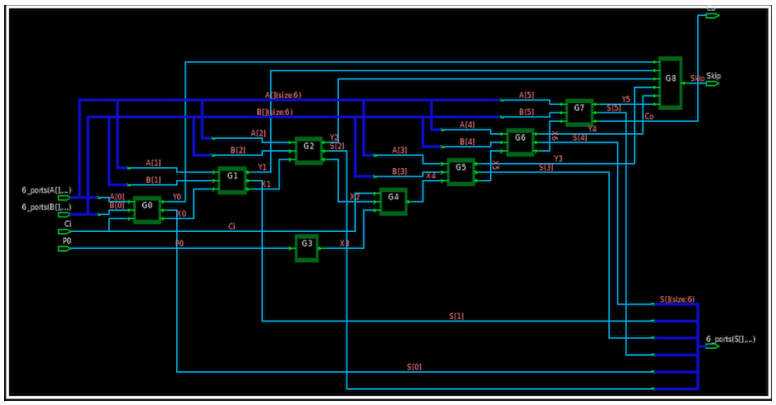




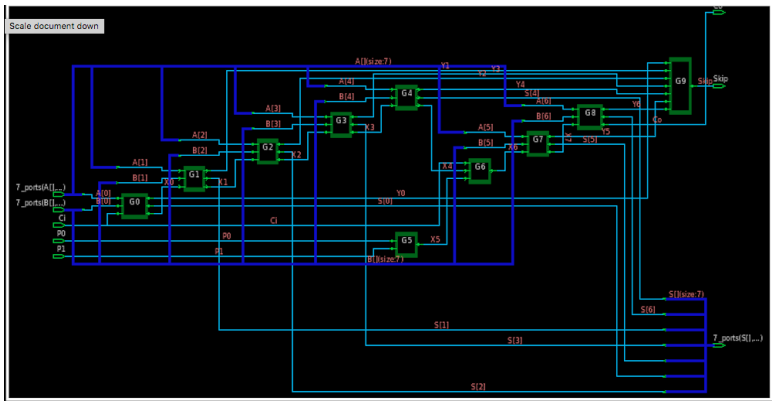
Schematic of a 5-bit block and its Hierarchy



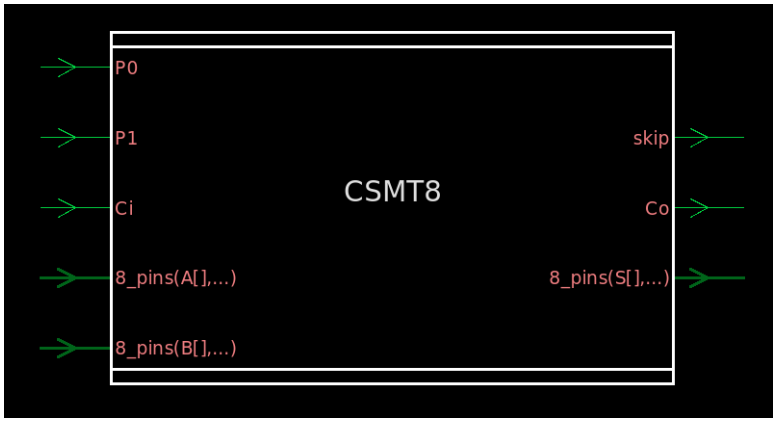
Schematic of a 6-bit block and its Hierarchy

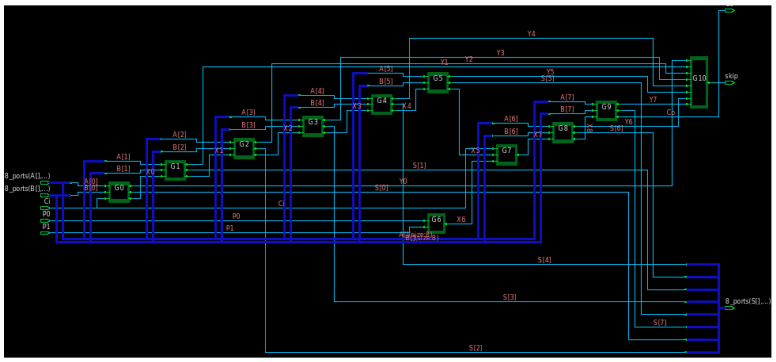


Schematic of a 7-bit block and its Hierarchy

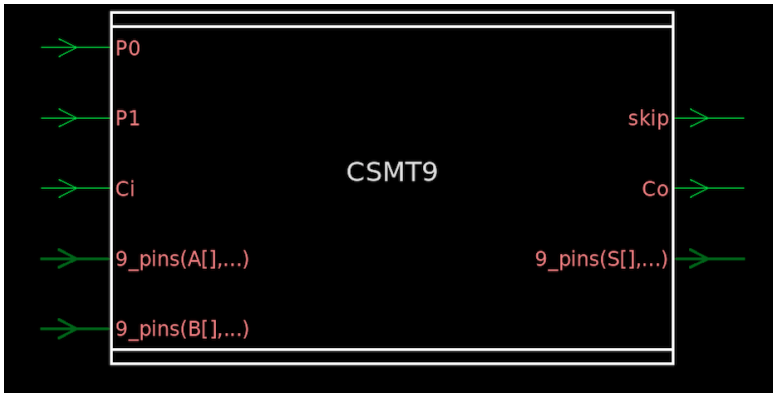


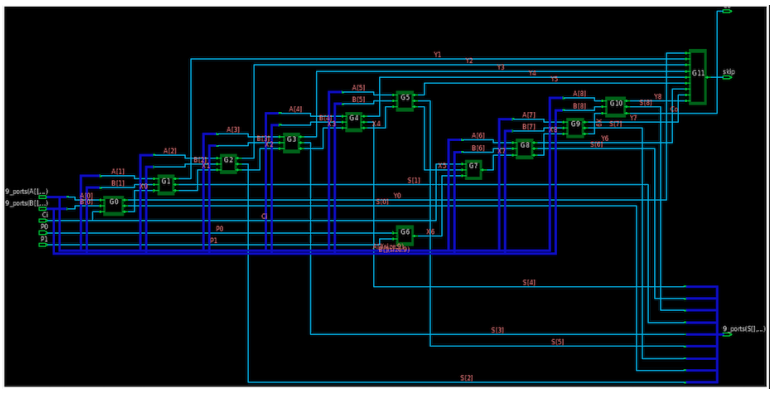
Schematic of a 8-bit block and its Hierarchy



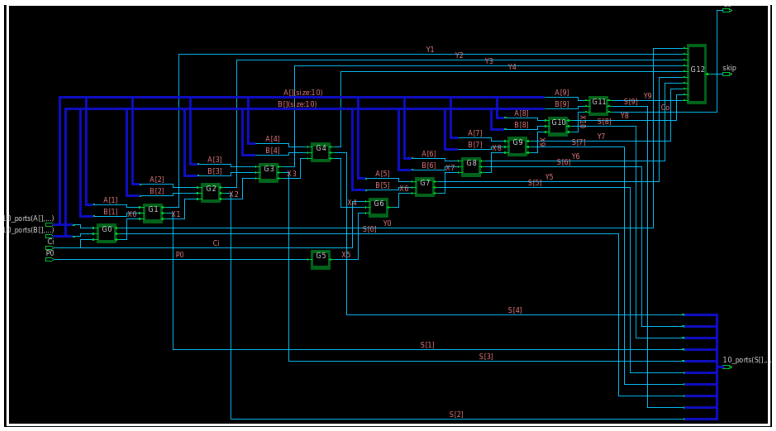


Schematic of a 9-bit block and its Hierarchy





Schematic of a 10-bit block and its Hierarchy



Schematic of a 11-bit block and its Hierarchy