RV College of Engineering®

Autonomous institution affiliated to Visvesvaraya Technological University, Belagavi) | Approved by AICTE, New Delhi

Go, change the world

Design of Vedic Multiplier using High-speed Area Efficient 3 Operands Adder

Major Project Report

(18MVE41)

Submitted by,

Mohammed Ikramuddin **1RV20LVS18**

Under the guidance of

Dr. Govinda Raju M
Assistant Professor
Dept. of ECE
RV College of Engineering

Dr. Kariyappa B.S
Professor
Dept. of ECE
RV College of Engineering

In partial fulfillment of the requirements for the degree of

Master of Technology in

VLSI Design & Embedded Systems

2021-2022

RV College of Engineering®, Bengaluru

(Autonomous institution affiliated to VTU, Belagavi)

Department of Electronics and Communication Engineering

.

<u>CERTIFICATE</u>

Certified that the technical seminar project (18MVE42)work titled ***Design of Vedic Multiplier using High-speed Area Efficient 3 Operands Adder*** is carried out by **Mohammed Ikramuddin** (**1RV20LVS18**) who is bonafide student of RV College of Engineering, Bengaluru, in partial fulfillment of the requirements for the degree of **Master of Technology** in **VLSI Design & Embedded Systems** of the Visvesvaraya Technological University, Belagavi during the year 2021-2022. It is certified that all corrections/suggestions indicated for the Internal Assessment have been incorporated in the technical seminar project report deposited in the departmental library. The technical seminar project report has been approved as it satisfies the academic requirements in respect of technical seminar project work prescribed by the institution for the said degree.

Signature of Guide	Signature of Head of the Department	Signature of Principal
Dr. Govinda Raju M	Dr. K S Geetha	Dr. K. N. Subramanya

External Viva

Name of Examiners	Signature with Date
1.	
2.	

DECLARATION

I , **Mohammed Ikramuddin** students of fourth semester M.Tech in VLSI Design & Embedded Systems, Department of Electronics and Communication Engineering, RV College of Engineering, Bengaluru, hereby declare that the minor project titled '**Design of Vedic Multiplier using High-speed Area Efficient 3 Operands Adder** ' has been carried out by me and submitted in partial fulfilment for the award of degree of **Master of Technology** in **VLSI Design & Embedded Systems** during the year 2021-2022.

Further I declare that the content of the dissertation has not been submitted previously by anybody for the award of any degree or diploma to any other university.

I also declare that any Intellectual Property Rights generated out of this project carried out at RVCE will be the property of RV College of Engineering, Bengaluru and we will be one of the authors of the same.

Place: Bengaluru

Date:

	Name	Signature
1.	Mohammed Ikramuddin(1RV20LVS18)	

ACKNOWLEDGEMENT

I am indebted to my guide, **Dr. Govinda Raju M**, Assistant Professor, RV College of Engineering . and **Dr. Kariyappa B.S**, Professor, RV College of Engineering . for the wholehearted support, suggestions and invaluable advice throughout my project work and also helped in the preparation of this thesis.

I also express our gratitude to my panel members **Dr. Kariyappa B.S**, Professor Department of Electronics and Communication Engineering for his valuable comments and suggestions during the phase evaluations.

My sincere thanks to the project coordinators **Dr Sujata D Badiger**, for her timely instructions and support in coordinating the project.

My gratitude to **Prof. Narashimaraja P** for the organized latex template which made report writing easy and interesting.

My sincere thanks to **Dr. K S Geetha**, Professor and Head, Department of Electronics and Communication Engineering, RVCE for the support and encouragement.

I express sincere gratitude to our beloved Principal, **Dr. K. N. Subramanya** for the appreciation towards this project work.

I thank all the teaching staff and technical staff of Electronics and Communication Engineering department, RVCE for their help.

Lastly, I take this opportunity to thank my family members and friends who provided all the backup support throughout the project work.

ABSTRACT

Three-operand binary adder is the basic functional unit to perform the modular arithmetic in various cryptography and pseudorandom bit generator (PRBG) algorithms. Carry save adder (CS3A) is the widely used technique to perform the three-operand addition. However, the ripple-carry stage in the CS3A leads to a high propagation delay of O(n). Moreover, a parallel prefix two-operand adder such as Han-Carlson (HCA) can also be used for three-operand addition that significantly reduces the critical path delay at the cost of additional hardware. Hence, a new high-speed and area-efficient adder architecture is proposed using pre-compute bitwise addition followed by carry prefix computation logic to perform the three-operand binary addition that consumes substantially less area, low power and drastically reduces the adder delay to O($log_2 n$).

A 16x16 Vedic Multiplier is designed using Carry Save Adder and High-Speed Area Efficient Adder and both the multipliers are implemented and simulated using Verilog code in Xilinx Vivado ISE. The delay and power consumption are measured for both the multipliers.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1
Introduction

CHAPTER 1

INTRODUCTION

1.1 Overview

As multipliers play a critical role in any digital design. It is necessary to implement faster multipliers occupying lesser area and consuming less power. Multiplication operation can be done using Array multiplier, redundant binary structures, and tree structures but they have problem of larger delay. Vedic mathematics is the name given to the ancient Indian system of mathematics that was rediscovered in early 20th century. Vedic multiplication uses Urdhva Triyakbhyam (Vertically and Cross wise) Sutras.

1.2 Motivation

- Vedic mathematics helps in simplifying calculations.
- High-speed Area Efficient 3 operands Adder provides better speed and less area than carry save adder.

1.3 Problem statement

To achieve better speed and area efficient 16x16 Vedic multiplier using High-speed Area Efficient three operands Adder and compare the same using 16x16 Vedic multiplier using Carry Save Adder.

1.4 Objectives

The objectives of the project are

1. To design a Verilog code and implement 16x16 Vedic multiplier using Carry Save Adder.
2. To design a Verilog code and implement 16x16 Vedic multiplier using High Speed Area Efficient 3 operands Adder.
3. Comparing delay and power consumption for both the adders.

1.5 Literature Review

To achieve optimal system performance while maintaining physical security, it is necessary to implement the cryptography algorithms on hardware [1] – [3]. Modular arithmetic such as modular exponentiation, modular multiplication and modular addition is frequently used for the arithmetic operations in various cryptography algorithms [4]. Therefore, the performance of the cryptography algorithm depends on the efficient implementation of the congruential modular arithmetic operation. The most efficient approach to implement the modular multiplication and exponentiation is the Montgomery algorithm [5] – [7] whose critical operation is based on three-operand binary addition [6] – [8]. The three-operand binary addition is also a primary arithmetic operation in the linear congruential generator (LCG) based pseudo-random bit generators (PRBG) such as coupled LCG (CLCG) [9], modified dual-CLCG (MDCLCG) [10] and coupled variable input LCG (CVLCG) [11]. Modified dual-CLCG (MDCLCG) is the most secure and highly random PRBG method among all the LCG-based and other existing PRBG methods. It is polynomial-time unpredictable and secure if n _ 32-bits. Therefore, the security of the MDCLCG enhances with the increase of operand size. However, the area and critical path delay increases linearly since its hardware architecture consists of four three-operand modulo-2n adders, two comparators, four multiplexers area [10]. Hence, the performance of the MDCLCG can be improved by the efficient implementation of the three-operand adder. The three-operand binary addition can be carried out either by using two two-operand adders or one three-operand adder. Carry-save adder (CS3A) is the area-efficient and widely adopted technique to perform the three-operand binary addition in the modular arithmetic used in cryptography algorithms [5] – [8], [12] – [14] and PRBG methods [9] – [11]. However, the longer carry propagation delay in the ripple-carry stage of CS3A seriously influences the performance of the MDCLCG and other cryptography architectures on IoT based hardware devices. To shorten the critical path delay, a parallel prefixed two-operand adder such as Han-Carlson (HCA) can also be used for three-operand binary addition. It reduces the critical path delay in the order of O(log2 n) but increases the area in the order of O(n log2 n) [15]. Therefore, it is necessary to develop an efficient VLSI architecture to carry out the fast three-operand binary addition with minimum hardware resources. Hence, a new high-speed area-efficient adder technique is proposed using pre-compute bitwise addition followed by carry-prefix computation logic to perform the three-operand addition in this paper that consumes considerably less gate area while minimizing the propagation delay in comparison to the HCA-based three-operand adder (HC3A).

1.6 Brief Methodology of the project

A 2x2 Vedic Multiplier is used to design 16x16 Vedic multiplier using carry save adder and high-speed area efficient (HSAE) adder. The fig. 3.1 shows the design of 2x2 Vedic Multiplier as shown below. The design of 4x4 Vedic Multiplier using 2x2 Vedic Multiplier

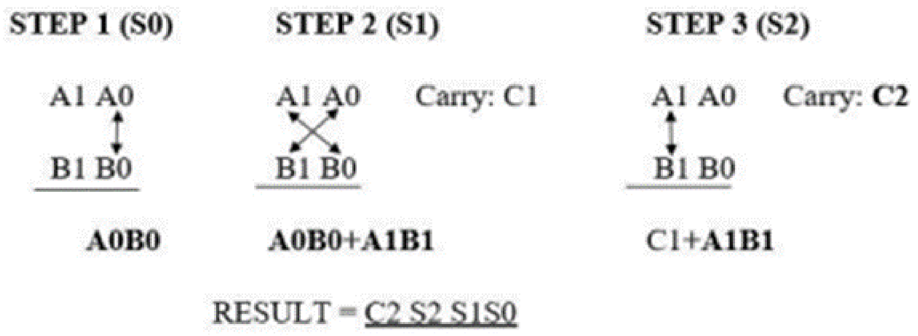

Figure 1.1: 2x2 Vedic Multiplier

and CSA is shown below in fig. 1.2 Similar to 4x4 Vedic Multiplier, the 8x8 and 16x16

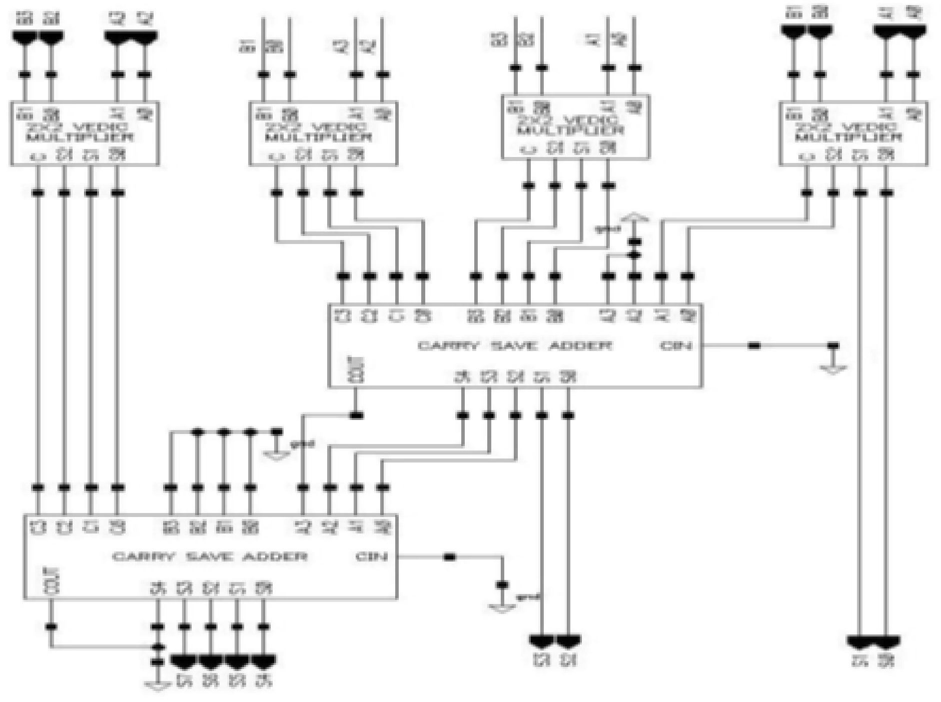

Figure 1.2: 4x4 Vedic Multiplier

Vedic Multipliers are designed using CSA and HSAE adder. $S'\_i = a\_i \oplus b\_i \oplus c\_i$

1.7 Organization of the report

This report is organized as follows. Write the discussions in each chapter. A sample is as follows.

- Chapter 2 discusses the fundamentals of three-operand binary adder techniques.
- Chapter 3 discusses the design of 16 x 16 Vedic multiplier.
- Chapter 4 discusses the results and discussions.
- Chapter 5 discusses the conclusion and future scope.

.

Chapter 2

Three-Operand Binary Adder Techniques

CHAPTER 2

THREE-OPERAND BINARY ADDER TECHNIQUES

The three-operands binary addition is one of the critical arithmetic operation in the congruential modular arithmetic architectures [5] – [8] and LCG-based PRBG methods such as CLCG [9], MDCLCG [10] and CVLCG [11]. It can be implemented either by using two stages of two-operands adders or one stage of three-operand adder.

2.1 Three operands Carry Save Adder

Carry-save adder (CSA) is the commonly used technique to perform the three-operand binary addition [9]–[14]. It computes the addition of three operands in two stages. The first stage is the array of full adders. Each full adder computes “carry” bit and “sum” bit concurrently from three binary input a_i , b_i and c_i . The second stage is the ripple-carry adder that computes the final n-bit size “sum” and one-bit size “carry-out” signals at the output of three-operand addition. The “carry-out” signal is propagated through the n number of full adders in the ripple-carry stage. Therefore, the delay increases linearly with the increase of bit length. The architecture of the three-operand carry-save adder is shown in Fig. 2.1 and the critical path delay is highlighted with a dashed line. It shows that the critical path delay depends on the carry propagation delay of ripple carry stage and is evaluated as follows,

$$T_{CS3A} = (n+1)T_{FA} = 3T_X + 2_n T_G \tag{2.1}$$

Similarly, the total area is evaluated as follows,

$$A_{CS3A} = 2nA_{FA} = 4nA_X + 6nA_G \tag{2.2}$$

Here, A_G and T_G indicate the area and propagation delay of basic 2-input gate (AND/OR/-NAND/NOR) respectively. A_X and T_X indicate the area and propagation delay of 2-input XOR gate respectively. The major drawback of the CS3A is the larger critical path delay which increases with an increase of bit length. This critical propagation path delay influences the overall latency of the congruential modular arithmetic based cryptography and PRBG architectures, where three-operand adder is the primary component.

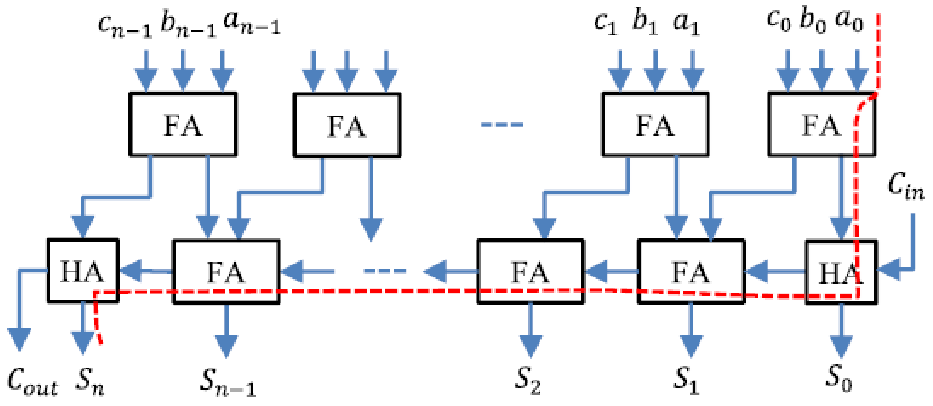

Figure 2.1: Three-operands carry-save adder (CS3A)

2.2 High-Speed Area Efficient Three Operands Adder

This section presents a new adder technique and its VLSI architecture to perform the three-operand addition in modular arithmetic. The adder technique is a parallel prefix adder.

2.2.1 Overview of HSAE Adder

The HSAE adder has four-stage structures instead three-stage structures in prefix adder to compute the addition of three binary input operands such as bit-addition logic, base logic, PG (propagate and generate) logic and sum logic. The logical expression of all these four stages is defined as follows, The proposed VLSI architecture of the three-operand binary adder and its internal structure is shown in Fig. 2.2. The new adder technique performs the addition of three n-bit binary inputs in four different stages. In the first stage (bit-addition logic), the bitwise addition of three n-bit binary input operands is performed with the array of full adders, and each full adder computes "sum (S'_i)" and "carry (cy_i)" signals as highlighted in Fig. 2.2 (a). The logical expressions for computing sum (S'_i) and carry (cy_i) signals are defined in Stage-1, and the logical diagram of the bit-addition logic is shown in Fig. 2.2 (b).

In the first stage, the output signal "sum (S'_i)" bit of current full adder and the output signal "carry" bit of its right-adjacent full adder are used together to compute the generate (G_i) and propagate (P_i) signals in the second stage (base logic). The computation of G_i and P_i signals are represented by the "squared saltire-cell" as shown in Fig. 2.2 (a) and

Stage-1: Bit Addition Logic:

$$S'_i = a_i \oplus b_i \oplus c_i,$$
$$cy_i = a_i \cdot b_i + b_i \cdot c_i + c_i \cdot a_i$$

Stage-2: Base Logic:

$$G_{i:i} = G_i = S'_i \cdot cy_{i-1}, \quad G_{0:0} = G_0 = S'_0 \cdot C_{in}$$
$$P_{i:i} = P_i = S'_i \oplus cy_{i-1}, \quad P_{0:0} = P_0 = S'_0 \oplus C_{in}$$

Stage-3: PG (Generate and Propagate) Logic:

$$G_{i:j} = G_{i:k} + P_{i:k} \cdot G_{k-1:j},$$
$$P_{i:j} = P_{i:k} \cdot P_{k-1:j}$$

Stage-4: Sum Logic:

$$S_i = (P_i \oplus G_{i-1:0}), \quad S_0 = P_0, \; C_{out} = G_{n:0}$$

there is n+1 number of saltire-cells in the base logic stage. The logic diagram of the saltire-cell is shown in Fig. 2.2 (b), and it is realized by the following logical expression,

$$G_{i:i} = G_i = S'_i \cdot cy_{i-1} \tag{2.3}$$

$$P_{i:i} = P_i = S'_i \cdot cy_{i-1} \tag{2.4}$$

The external carry-input signal (C_{in}) is also taken into consideration for three-operand addition in the proposed adder technique. This additional carry-input signal (C_{in}) is taken as input to base logic while computing the $G_0(S'_0 \cdot C_{in})$ in the first saltire-cell of the base logic. The third stage is the carry computation stage called "generate and propagate logic" (PG) to pre-compute the carry bit and is the combination of black and grey cell logics. The logical diagram of black and grey cell is shown in Fig. 2.2 (b) that computes the carry generate $G_{i:j}$ and propagate $P_{i:j}$ signals with the following logical expression,

$$G_{i:j} = G_{i:k} + P_{i:k} \cdot G_{k-1:j} \tag{2.5}$$

$$P_{i:j} = P_{i:k} \cdot P_{k-1:j} \tag{2.6}$$

The number of prefix computation stages for the proposed adder is $(\log_2 n + 1)$, and therefore, the critical path delay of the proposed adder is mainly influenced by this carry

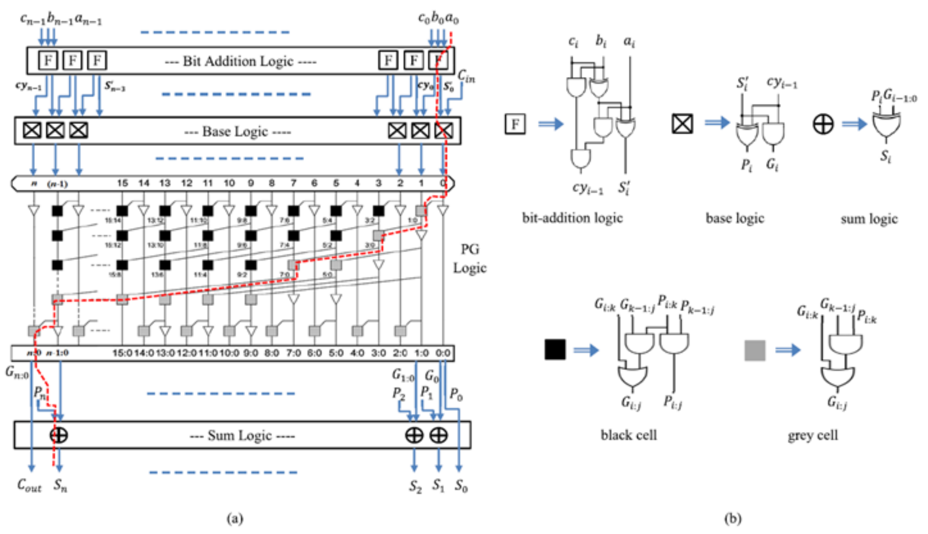

Figure 2.2: High Speed Area Efficient Adder

propagate chain.

The final stage is represented as sum logic in which the "sum (S_i)" bits are computed from the carry generate $G_{i:j}$ and carry propagate P_i bits using the logical expression, $S_i = (P_i \oplus G_{i-1:0})$. The carryout signal ($C_{out}$) is directly obtained from the carry generate bit $G_{n:0}$.

Chapter 3

Design Of 16x16 Vedic Multiplier

CHAPTER 3

DESIGN OF 16X16 VEDIC MULTIPLIER

The Verilog code and the testbench are written for a 16x16 Vedic multiplier using a carry-save adder and high-speed area efficient adder.

3.1 Verilog Code and Testbench for 16x16 Vedic Multiplier

Figure 3.1 illustrates the steps to multiply two 2-bit numbers. Converting the above figure to a hardware equivalent we have 3 and gates that will act as 2-bit multipliers and two half adders to add the products to get the final product. Here is the hardware detail of the multiplier. Where "a" and "b" are two numbers to be multiplied and "q" is the

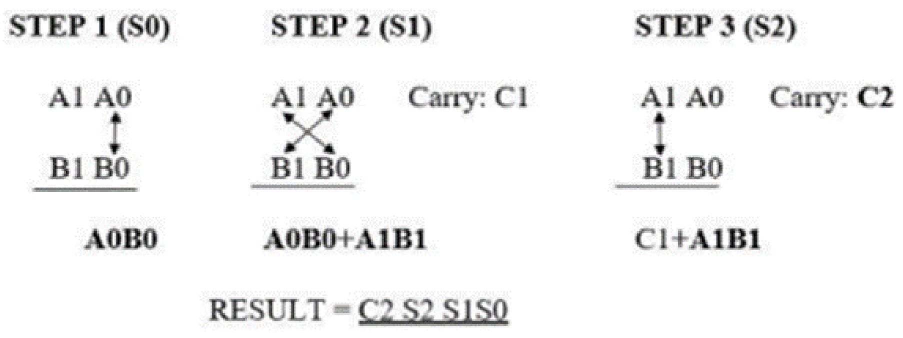

Figure 3.1: Steps to Multiply 2x2 Vedic Multiplier

product. With this design we are now ready to code this in verilog easily using and gates and HA (half adders). To make the design more modular we try to write code for HA first and then instantiate it to have the final product.

The Verilog code for 2x2 Vedic Multiplier is

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 12:34:18 08/01/2013
// Design Name:
// Module Name: vedic_2_x_2
```

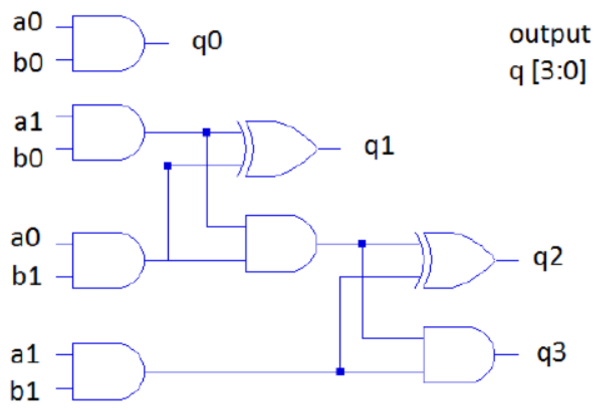

Figure 3.2: Design of 2x2 Vedic Multiplier

```
// Project Name:
// Target Devices:
// Tool versions:
// Description:
// // Dependencies:
// // Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module vedic_2_x_2(
a,
b,
c
);
input [1:0]a;
input [1:0]b;
output [3:0]c;
wire [3:0]c;
wire [3:0]temp;
//stage 1
// four multiplication operation of bits according to vedic logic done using and gates
assign c[0]=a[0]&b[0];
assign temp[0]=a[1]&b[0];
assign temp[1]=a[0]&b[1];
assign temp[2]=a[1]&b[1];
//stage two
// using two half adders
ha z1(temp[0],temp[1],c[1],temp[3]);
ha z2(temp[2],temp[3],c[2],c[3]);
endmodule
```

The Verilog code for 4x4 Vedic Multiplier is

```
'timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 14:30:08 08/01/2013
// Design Name:
// Module Name: vedic_4_x_4
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0·01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module vedic_4_x_4(
a,b,c
);
input [3:0]a;
input [3:0]b;
output [7:0]c;

wire [3:0]q0;
wire [3:0]q1;
wire [3:0]q2;
wire [3:0]q3;
wire [7:0]c;
wire [3:0]temp1;
wire [5:0]temp2;
wire [5:0]temp3;
wire [5:0]temp4;
wire [3:0]q4;
wire [5:0]q5;
wire [5:0]q6;
// using 4 2x2 multipliers
vedic_2_x_2 z1(a[1:0],b[1:0],q0[3:0]);
vedic_2_x_2 z2(a[3:2],b[1:0],q1[3:0]);
vedic_2_x_2 z3(a[1:0],b[3:2],q2[3:0]);
vedic_2_x_2 z4(a[3:2],b[3:2],q3[3:0]);
// stage 1 adders
assign temp1 =2'b0,q0[3:2];
add_4_bit z5(q1[3:0],temp1,q4);
assign temp2 =2'b0,q2[3:0];
assign temp3 =q3[3:0],2'b0;
add_6_bit z6(temp2,temp3,q5);
assign temp4=2'b0,q4[3:0];
// stage 2 adder
add_6_bit z7(temp4,q5,q6);
// fnal output assignment
```

```
assign c[1:0]=q0[1:0];
assign c[7:2]=q6[5:0];

endmodule
```

The Verilog code for 8x8 Vedic Multiplier is

```
‘timescale 1ns / 1ps
////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 15:22:39 08/01/2013
// Design Name:
// Module Name: vedic_8X8
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////
module vedic_8X8(a,b,c
);

input [7:0]a;
input [7:0]b;
```

```
output [15:0]c;

wire [15:0]q0;
wire [15:0]q1;
wire [15:0]q2;
wire [15:0]q3;
wire [15:0]c;
wire [7:0]temp1;
wire [11:0]temp2;
wire [11:0]temp3;
wire [11:0]temp4;
wire [7:0]q4;
wire [11:0]q5;
wire [11:0]q6;
// using 4 4x4 multipliers
vedic_4_x_4 z1(a[3:0],b[3:0],q0[15:0]);
vedic_4_x_4 z2(a[7:4],b[3:0],q1[15:0]);
vedic_4_x_4 z3(a[3:0],b[7:4],q2[15:0]);
vedic_4_x_4 z4(a[7:4],b[7:4],q3[15:0]);

// stage 1 adders
assign temp1 =4'b0,q0[7:4];
add_8_bit z5(q1[7:0],temp1,q4);
assign temp2 =4'b0,q2[7:0];
assign temp3 =q3[7:0],4'b0;
add_12_bit z6(temp2,temp3,q5);
assign temp4=4'b0,q4[7:0];
// stage 2 adder
add_12_bit z7(temp4,q5,q6);
// fnal output assignment
assign c[3:0]=q0[3:0];
assign c[15:4]=q6[11:0];
```

```
endmodule
```

The Verilog code for 16x16 Vedic Multiplier is

```
‘timescale 1ns / 1ps ////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 15:45:52 08/01/2013
// Design Name:
// Module Name: vedic_16x16
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module vedic_16x16(a,b,c
);
input [15:0]a;
input [15:0]b;
output [31:0]c;

wire [15:0]q0;
wire [15:0]q1;
wire [15:0]q2;
wire [15:0]q3;
wire [31:0]c;
wire [15:0]temp1;
wire [23:0]temp2;
wire [23:0]temp3;
wire [23:0]temp4;
wire [15:0]q4;
wire [23:0]q5;
wire [23:0]q6;
// using 4 8x8 multipliers
vedic_8X8 z1(a[7:0],b[7:0],q0[15:0]);
vedic_8X8 z2(a[15:8],b[7:0],q1[15:0]);
vedic_8X8 z3(a[7:0],b[15:8],q2[15:0]);
vedic_8X8 z4(a[15:8],b[15:8],q3[15:0]);

// stage 1 adders
assign temp1 =8'b0,q0[15:8];
add_16_bit z5(q1[15:0],temp1,q4);
assign temp2 =8'b0,q2[15:0];
assign temp3 =q3[15:0],8'b0;
add_24_bit z6(temp2,temp3,q5);
assign temp4=8'b0,q4[15:0];

//stage 2 adder
add_24_bit z7(temp4,q5,q6);
// final output assignment
assign c[7:0]=q0[7:0];
assign c[31:8]=q6[23:0];

endmodule
```

The Verilog Tesbench code for 16x16 Vedic Multiplier is

```
‘timescale 1ns / 1ps

////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 16:15:22 08/01/2013
// Design Name: vedic_16x16
// Module Name: D:/lfsr/test_vedic_16 · v
// Project Name: lfsr
// Target Device:
// Tool versions:
// Description:
//
// Verilog Test Fixture created by ISE for module:
vedic_16x16
//
// Dependencies:
//
// Revision:
// Revision 0·01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////

module test_vedic_16;

// Inputs
reg [15:0] a;
reg [15:0] b;
```

```
// Outputs
wire [31:0] c;

// Instantiate the Unit Under Test (UUT)
vedic_16x16 uut (
·a(a),
·b(b),
·c(c)
);

initial begin
// Initialize Inputs
a = 0;
b = 0;
# 100;

a = 16'd12;
b = 16'd12;
# 100;

a = 16'd15;
b = 16'd13;
# 100;

a = 16'd24;
b = 16'd2;
# 100;

a = 16'd200;
b = 16'd21;
# 100;
```

```
a = 16'd36;
b = 16'd48;
# 100;



end

endmodule
```

Chapter 4

Results & Discussions

CHAPTER 4

RESULTS & DISCUSSIONS

This chapter provide details related to results of the design implementation. The results are divided as simulation result of 16 x 16 Vedic Multiplier using carry save adder and result of 16 x 16 Vedic Multiplier using high-speed area efficient adder.

4.1 Simulation Result

The simulation analysis of the 16 x 16 Vedic Multiplier using carry save adder and of 16 x 16 Vedic Multiplier using high-speed area efficient adder is carried out using Xilinx ISE Software. The simulation result is shown in Fig. 4.1.

Here A & B are the input 16-bit data and C is the 32 bit multiplier output. One can

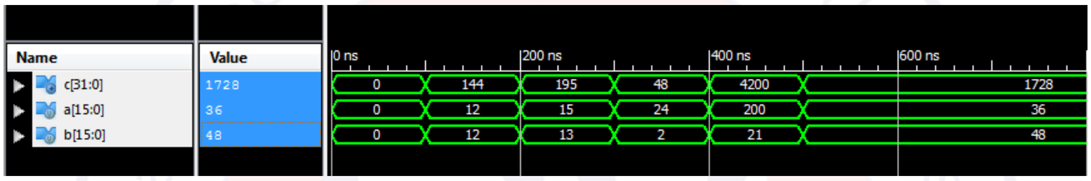

Figure 4.1: Simulation Result of 16 x 16 Vedic Multiplier

easily observe that the output for 16 x 16 Vedic Multiplier using carry save adder and 16 x 16 Vedic Multiplier using high-speed area efficient adder has been obtained successfully. The multiplication of 21 & 200 yields 4200 and multiplication of 13 & 15 yields 195, simlarly the other multiplication input combinations can be observed and the results are true with no errors. From the table 4.1 we can observe the results that the 16x16 Vedic

Table 4.1: Comparision of Critical Path Delay and Total Power

SL NO	16x16 Vedic Multiplier using	Critical Path Delay(ns)	Total Power (µW)
1	**CSA**	0.882	70
2	**HSAE Adder**	0.824	55

Multiplier using High-speed-area-efficient adder is having the total power and critical path delay of 55µW and 824ps respectively which is much better as compared to 16x16 Vedic Multiplier using CSA.

Chapter 5

Conclusion and Future Scope

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

5.1 Conclusion

Multiplication is a resource hungry operation and is a major computational need in complex circuits. Through literature survey, it was found that As multipliers play a critical role in any digital design. It is necessary to implement faster multipliers occupying lesser area and consuming less power. Multiplication operation can be done using Array multiplier, redundant binary structures and tree structures but they have problem of larger delay. Vedic Multipliers are faster compared to traditional Array multiplier, Wallace tree multiplier etc.

The 16x16 Vedic Multiplier using High-speed-area-efficient adder is having the power consumption and total delay of 55µW and 824ps respectively which is much better as compared to 16x16 Vedic Multiplier using CSA.

5.2 Future Scope

Further work can be taken up to increase bit-length and improve accuracy. Also, this modified Vedic Multiplier can be tested on different dataset or can be explored for different application.

BIBLIOGRAPHY

[1] A. K. Panda, R. Palisetty, and K. C. Ray, “High-speed area-efficient vlsi architecture of three-operand binary adder,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 11, pp. 3944–3953, 2020.

[2] M. B. Murugesh, S Nagaraj, J Jayasree, and G. V. K. Reddy, “Modified high speed 32-bit vedic multiplier design and implementation,” in *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*, IEEE, 2020, pp. 929–932.

[3] M Pushpa and S Nagaraj, “Design and analysis of 8-bit array carry save array braun wallace tree and vedic multipliers,” in *IEEE Sponsored International Conference On New Trends In Engineering & Technology (ICNTET 2018).*, 2018.

[4] V Anbumani, S Soviya, S Sneha, and L Saran, “Speed and power efficient vedic multiplier using adders with mux,” in *2021 Innovations in Power and Advanced Computing Technologies (i-PACT)*, IEEE, 2021, pp. 1–5.

[5] A. K. Itawadiya, R. Mahle, V. Patel, and D. Kumar, “Design a dsp operations using vedic mathematics,” in *2013 International Conference on Communication and Signal Processing*, IEEE, 2013, pp. 897–902.

[6] D. Jaina, K. Sethi, and R. Panda, “Vedic mathematics based multiply accumulate unit,” in *2011 International Conference on Computational Intelligence and Communication Networks*, IEEE, 2011, pp. 754–757.

[7] S. Thakur and P. Kumar, “Area-efficient & high speed ripple carry based vedic multiplier,” *SSRG International Journal of Electronics and Communication Engineering (SSRG-IJECE)–EFES April 2015*, pp. 6–10, 2015.

[8] S. S. Meti, C. Bharath, Y. P. Kumar, and B. Kariyappa, “Design and implementation of 8-bit vedic multiplier using mgdi technique,” in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, IEEE, 2017, pp. 1923–1927.

[9] M. M. Islam, M. S. Hossain, M. K. Hasan, M. Shahjalal, and Y. M. Jang, “Fpga implementation of high-speed area-efficient processor for elliptic curve point multiplication over prime field,” *IEEE Access*, vol. 7, pp. 178 811–178 826, 2019.

[10] Z. Liu, J. Großschädl, Z. Hu, K. Järvinen, H. Wang, and I. Verbauwhede, "Elliptic curve cryptography with efficiently computable endomorphisms and its hardware implementations for the internet of things," *IEEE Transactions on Computers*, vol. 66, no. 5, pp. 773–785, 2016.

[11] Z. Liu, D. Liu, and X. Zou, "An efficient and flexible hardware implementation of the dual-field elliptic curve cryptographic processor," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 3, pp. 2353–2362, 2016.

[12] P. L. Montgomery, "Modular multiplication without trial division," *Mathematics of computation*, vol. 44, no. 170, pp. 519–521, 1985.

[13] S.-R. Kuang, K.-Y. Wu, and R.-Y. Lu, "Low-cost high-performance vlsi architecture for montgomery modular multiplication," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 2, pp. 434–443, 2015.

[14] S. S. Erdem, T. Yanık, and A. Çelebi, "A general digit-serial architecture for montgomery modular multiplication," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 5, pp. 1658–1668, 2017.

[15] R. S. Katti and S. K. Srinivasan, "Efficient hardware implementation of a new pseudo-random bit sequence generator," in *2009 IEEE International Symposium on Circuits and Systems*, IEEE, 2009, pp. 1393–1396.

[16] A. K. Panda and K. C. Ray, "Modified dual-clcg method and its vlsi architecture for pseudorandom bit generation," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 3, pp. 989–1002, 2018.

[17] A. K. Panda and K. C. Ray, "A coupled variable input lcg method and its vlsi architecture for pseudorandom bit generation," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 4, pp. 1011–1019, 2019.

[18] T. Kim, W. Jao, and S. Tjiang, "Circuit optimization using carry-save-adder cells," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, no. 10, pp. 974–984, 1998.

[19] A. Rezai and P. Keshavarzi, "High-throughput modular multiplication and exponentiation algorithms using multibit-scan–multibit-shift technique," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 9, pp. 1710–1719, 2014.

[20] A. K. Panda and K. C. Ray, “Design and fpga prototype of 1024-bit blum-blum-shub prbg architecture,” in *2018 IEEE International Conference on Information Communication and Signal Processing (ICICSP)*, IEEE, 2018, pp. 38–43.

[21] T. Han and D. A. Carlson, “Fast area-efficient vlsi adders,” in *1987 IEEE 8th symposium on computer arithmetic (ARITH)*, IEEE, 1987, pp. 49–56.

[22] D. L. Harris, *Parallel prefix networks that make tradeoffs between logic levels, fanout and wiring racks*, US Patent 7,152,089, 2006.

[23] H. Ling, “High-speed binary adder,” *IBM Journal of Research and Development*, vol. 25, no. 3, pp. 156–166, 1981.

[24] R. Jackson and S. Talwar, “High speed binary addition,” in *Conference Record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers, 2004.*, IEEE, vol. 2, 2004, pp. 1350–1353.

[25] K. S. Pandey, D. Kumar, N. Goel, and H. Shrimali, “An ultra-fast parallel prefix adder,” in *2019 IEEE 26th Symposium on Computer Arithmetic (ARITH)*, IEEE, 2019, pp. 125–134.

[26] F. Jafarzadehpour, A. S. Molahosseini, A. A. E. Zarandi, and L. Sousa, “New energy-efficient hybrid wide-operand adder architecture,” *IET Circuits, Devices & Systems*, vol. 13, no. 8, pp. 1221–1231, 2019.

[27] S. M. Sudhakar, K. P. Chidambaram, and E. E. Swartzlander, “Hybrid han-carlson adder,” in *2012 IEEE 55th International Midwest Symposium on Circuits and Systems (MWSCAS)*, IEEE, 2012, pp. 818–821.