IMPLEMENTATION AND ANALYSIS OF 16 x 16 VEDIC MULTIPLIER

For

Mini Project – 2B: FPGA Design Project (ECM501)
(REV- 2019 'C' Scheme) of Third Year (Semester-VI)
Bachelors in Engineering

By

Shrutika Patel D14B 51

Supervisor

Dr. (Mrs.) Saylee Gharge

Department of Electronics and Telecommunication Engineering
Vivekanand Education Society's Institute of Technology
Mumbai University
2021-2022

Mini Project-2B Approval

Project entitled **Implementation and Analysis of 16 x 16 Vedic Multiplier** by Shrutika Saket Patel (51) is approved for the degree of Bachelor of Engineering.

Examiners

1. _____
2. _____

Supervisors

1. Dr. (Mrs.) Saylee Gharge

Date: 18th April, 2022

Place: Mumbai

Certificate

This is to certify that Team Tech Quartet have completed the project report on the topic **Implementation and Analysis of 16 x 16 Vedic Multiplier** satisfactorily in partial fulfilment of the requirements for the award of Mini Project 2B (REV-2019 'C' Scheme) of Third Year, (Semester-VI) in Electronics and Telecommunication under the guidance of Dr. (Mrs.) Saylee Gharge during the year 2021-2022 as prescribed by University of Mumbai.

_____ _____

Supervisor
Dr. (Mrs.) Saylee Gharge

Head of Department
Dr. Chandansingh Rawat

_____

Principal
Mrs. Jayalekshmi Nair

_____ _____

Examiner 1

Examiner 2

Declaration

We declare that this written submission represents our ideas in our words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misinterpreted or fabricated or falsified any idea/data/fact/source in my submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Shrutika Patel

Date: 18th April, 2022

Table of Contents

Abstract

Multipliers are utilized in a wide range of DSP applications nowadays, including vector product, filtering, convolution operations, matrix multiplication, etc. The parameters which are important to consider with precision are speed of operation, chip space occupied, ease of design, power consumption, high noise immunity, and so on. With advances in technology, many researchers have been trying to design multipliers which offer either of the following design targets: high speed, low power consumption, regularity of layout and hence less area or even combination of them in one multiplier thus making them suitable for various high speed, low power and compact VLSI implementation. In this project, the design of 16 x 16 Vedic Multiplier Using Urdhva Tiryagbhyam Sutra is implemented. Also, the comparison of maximum combinational path latency, chip area consumption, and total on-chip power of an 8-bit Vedic multiplier using Urdhva Tiryagbhyam sutra, an 8-bit Wallace tree multiplier, and an 8-bit Array Multiplier written in Verilog is done. For proper comparison, all multipliers are made with full adders, half adders, n-bit adders, and basic gates. Creation and simulation of the stated multipliers using Xilinx ISE 14.7 on device 6slx9tqg144-2 and implementation of 8-bit Vedic multiplier on EDGE Spartan 7 FPGA Board is done to validate the same. The goal is to make a fair comparison of all of the multipliers presented.

List of Figures

List of Tables

Abbreviations

ADC	Analog to Digital Converter
ALU	Arithmetic Logic Unit
DAC	Digital to Analog Converter
DSP	Digital Signal Processing
FPGA	Field Programmable Gate Array
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
KSA	Kogge Stone Adder
RISC	Reduced Instruction Set Computer
RTL	Register Transfer Level
UTS	Urdhva Tiryagbhyam
VM	Vedic Multiplier

Chapter 1
Introduction

Introduction

1.1 Need of Vedic Multipliers

1.2 Proposed Solution

Introduction

Bharati Krishna Tirthaji was the catalyst for the development of Vedic mathematics in science and engineering. Identified as an Indian monk who wrote Vedic Mathematics, which was originally published in 1965. Vedic Mathematics is a list of control algorithms that the author claims were retrieved from the Vedas and allegedly encompassed all mathematical knowledge. Vedic Mathematics is a fast-calculation method based on Vedic Sutras that saves time in arithmetic while also providing benefits and features to increase mental math and math speed. The Yoga Sutras contain 16 main sutras and 13 sub-sutras. They can help in measurement, arithmetic, algebra, geometry, calculus, and data commercial math. Shri Bharati Krishna Tirthaji presented the concept of Vedic mathematics after eight years of research on the Atharva Vedas [9]. This branch of mathematics is based on the sixteen sutras. Vedic mathematics is an intriguing subject with several helpful algorithms. The application of Vedic Mathematics concepts to multiplication would result in a reduction in computational time.

As the number of digital devices grows, the most popular method of processing digital data is to use a Digital Signal Processing (DSP) unit. Text, audio, image and video are all examples of digital data. Multiplication is a significant and important arithmetic operation for processing digital data.

1.1 Need of Vedic Multipliers

In digital signal processing, multiplication is an essential hardware component for several activities. As a result, the multiplier's performance is critical in determining the whole system's performance. This is due to the multiplier being the system's slowest and most time-consuming element of any system. As the time required to execute a multiplication operation is longer, the goal should be to reduce delay in the critical path. To do this, designers should pay special attention to the adders that will be used in the multiplier architecture, because the propagation delay of the data inputs is heavily influenced by the path delay of the carry signal of the adders that will be utilized. As a result, the performance and cost of digital signal processors are affected. As a result, it is essential to select the type of multipliers and adders to be implemented in a system.

1.2 Proposed Solution

The 16 x 16 Vedic Multiplier is designed using:

1. 2 x 2 Vedic Multiplier
2. 4 x 4 Vedic Multiplier using 2 x 2 Vedic Multiplier
3. 8 x 8 Vedic Multiplier using 4 x 4 Vedic Multiplier and finally
4. 16 x 16 Vedic Multiplier using 8 x 8 Vedic Multiplier.

Integrating Vedic mathematics for the multiplier design will enhance the speed of multiplication operation. Urdhva - Tiryagbhyam is the most efficient algorithm that gives minimum delay for multiplication for all types of numbers irrespective of their size. To enhance speed numerous modifications over the standard booth algorithm, Wallace tree methods for multiplier design have been made and several new techniques are being worked upon. Amongst these Vedic multipliers based on Vedic mathematics are currently under focus [2]. In this report, the analysis for gate delay, computation time, look up tables, slices, total delay, no. of bounded IOB's and many more were performed for the 16 x 16 Vedic multiplier.

Chapter 2

Review of Literature

2.1 Referred Papers

2.1.1. FPGA Implementation of high speed 8-bit Vedic multiplier using barrel shifter

2.1.2. Design and FPGA Implementation of High-Speed Vedic Multiplier

2.1.3. Design and Implementation of 32-bit Complex Multiplier using Vedic Algorithm

2.1.4. Design and Implementation of 64-bit Multiplier using Vedic Algorithm

2.1.5. Performance comparison of Conventional and Vedic Multiplier using Simulator

2.1.6. Modified High Speed 32-bit Vedic Multiplier Design and Implementation

2.2 Summary of literature review

Literature Review

In the previous chapter we saw about the necessity of having of Vedic Multipliers in DSP applications. We also introduced how we can implement a 16-bit Vedic Multiplier. In this chapter we will have a brief look on how can we implement Vedic Multipliers in various ways. Following are the papers which were referred during the implementation of the project – Implementation and Analysis of 16 x 16 Vedic Multiplier.

2.1 Referred Papers

2.1.1. FPGA Implementation of high speed 8-bit Vedic multiplier using barrel shifter [1]

The paper describes the implementation of an 8-bit Vedic multiplier enhanced in terms of propagation delay when compared with conventional multiplier like array multiplier, Braun multiplier, modified booth multiplier and Wallace tree multiplier. They utilized 8-bit barrel shifter which requires only one clock cycle for 'n' number of shifts. The design was implemented and verified using FPGA and ISE Simulator. The core was implemented on Xilinx Spartan-6 family xc6s1x75T-3-fgg676 FPGA. The propagation delay comparison was extracted from the synthesis report and static timing report as well. They claimed that the design could achieve propagation delay of 6.781ns using barrel shifter in base selection module and multiplier.

The base selection module and the power index determinant form integral part of multiplier architecture. The architecture computes the mathematical expression in equation1.Barrel shifter used in this architecture. The two input numbers are fed to the base selection module from which the base is obtained. The outputs of base selection module (BSM) and the input numbers 'X' and 'Y' are fed to the subtractors. The subtractor blocks are required to extract the residual parts z1 and z2. The inputs to the power index determinant are from base selection module of respective input numbers. The sub-section of power index determinant (PID) is used to extract the power of the base and followed by subtractor to calculate the value. The outputs of subtractor are fed to the multiplier that feeds the input to the second adder or subtractor. Likewise the outputs of power index determinant are fed to the third subtractor that feeds the input to the barrel shifter. The input number 'X' and the output of barrel shifter are rendered to first adder/subtractor and the output of it is applied to the second barrel shifter which will provide the intermediate value. The last sub-section of this multiplier architecture is the second adder/subtractor which will provide the required result.

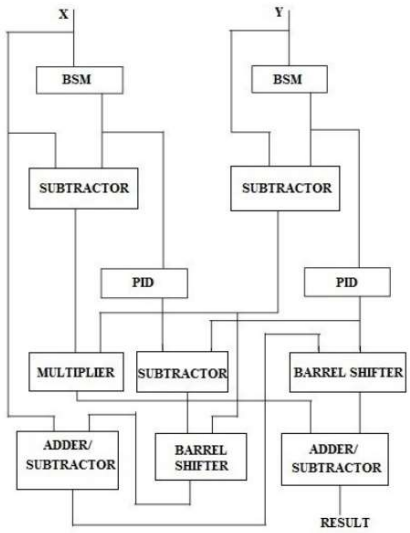

Fig. 2.1. Architecture of high speed 8-bit Vedic multiplier using barrel shifter [1]

Comparison between conventional multipliers and proposed design was concluded by the authors. Around 75% of reduction in delay can be observed from the proposed design with respect to array multiplier, whereas the conventional Vedic multiplier contributes to 43% of reduction in delay with respect to array multiplier. The analysis provides much in-depth coverage between conventional multipliers and modified Vedic multiplier architecture. In the author's design, efforts have been made to reduce the propagation delay and achieved an improvement in the reduction of delay with 45% when compared to array multiplier, booth multiplier and conventional Vedic multiplier implementation on FPGA. The high-speed implementation of such a multiplier has wide range of applications in image processing, arithmetic logic unit and VLSI signal processing. The future scope of this particular work can be extended in design of ALU's in RISC processor.

2.1.2. Design and FPGA Implementation of High-Speed Vedic Multiplier [2]

The authors of this paper put forward a high-speed Vedic multiplier which is efficient in terms of speed, making use of Urdhva Tiryagbhyam, a sutra from Vedic Math for multiplication and Kogge Stone algorithm for performing addition of partial products and also compares it with the characteristics of existing algorithms. The below two algorithms' aids to parallel generation of partial products and faster carry generation respectively, leading to better performance. The code is written in Verilog HDL and implemented on Xilinx Spartan 3 and Spartan 6 FPGA kit using Xilinx ISE 9.1i. The propagation delay of the implemented architecture is obtained to be 28.699ns and 15.752ns respectively. The authors justified that in conventional method, partial products are summated only after every partial product is obtained. Whereas, in Vedic technique, partial products are obtained vertically as shown in the figure

above and simultaneously once all the elements of a column are obtained, respective partial products are added. Hence, leads to advancement in speed over the conventional method.

Proposed multiplier architecture of 2x2, 4x4, and 8x8 bit VM module were displayed. They used of kogge stone adder as the adders required in the multiplier.

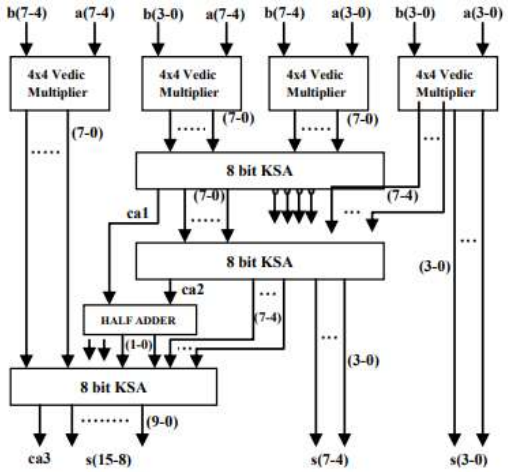

Fig. 2.2. 8x8 Vedic Multiplier using KSA [2]

The Verilog code of 8x8 Vedic multiplier was synthesized using Xilinx ISE 9.1i and was implemented on FPGA device xc3s400-5tq144 of SPARTAN 3 Family. The results are shown. DIP switches are used as input devices and LEDs are used as output devices. Comparison of delays between 8x8 modified Vedic multipliers using RCA and KSA executed on xc3s700-afg484 and VM using RCA represented in the paper.

Table 2.1: Comparison of propagation delay

DEVICE - xc3s50a-5tq144	
Multipliers	**Delay (nS)**
Array Multiplier	32.01
Booth Multiplier	29.549
VM8x8KSA	23.644

By the table 2.1, the authors conclude that the proposed technique of multiplication using Urdhva Tiryagbhyam algorithm and Kogge Stone algorithm causes less latency when compared to available techniques in literature. The proposed technique when implemented for 8x8 bit multiplication, the delay is found to be 28.699ns on SPARTAN 3 and 15.752 ns on SPARTAN 6. The adoption of KSA algorithm for higher bit size multipliers will further show improvement in speed. Further, higher speeds can be achieved by making use of pipelining and

parallel processing techniques. This work will increase awareness of Vedic mathematics techniques in the field of engineering and delivers high performance in DSP Processors.

2.1.3. Design and FPGA Implementation of High-Speed Vedic Multiplier [3]

This paper discusses the design of 32-bit Complex Multiplier using the techniques of Ancient Indian Vedic Mathematics that have been modified to improve performance. Vedic Mathematics is the ancient system of mathematics which has a unique technique of calculations based on 16 Sutras. The Vedic method used here for complex multiplication is Urdhva Tiryagbhyam (vertically and Cross wise). Urdhva Tiryagbhyam Sutra is the most efficient Sutra (Algorithm) that gives minimum delay for multiplication of small or large types of numbers. This paper also presents comparison of 8-bit, 16-bit, 32-bit complex multiplier on various performance parameters like power and delay. The proposed system is designed using VHDL and Verilog and is implemented through Xilinx ISE 14.2 navigator and modelsim v6.3 software's.

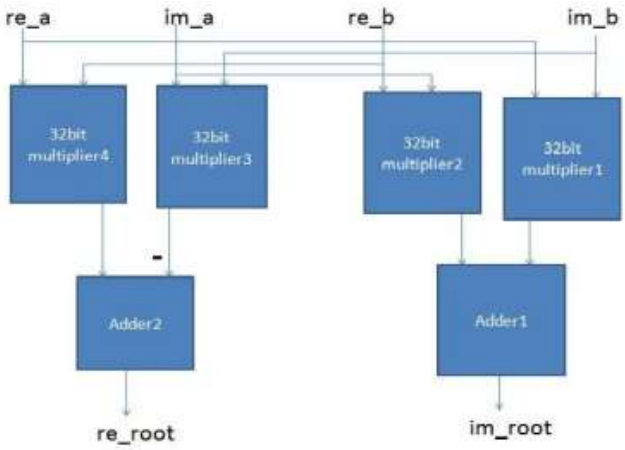

Fig. 2.3. Block Diagram of 32-bit complex multiplier [3]

Fig. 2.3. shows the block diagram of 32x32 complex number multiplier. It requires four 32x32bit Vedic multiplier modules and adders/subtractors. Let (re_a+j im_a) and (re_b+j im_b) be the two 32-bit complex numbers. re_root+jim_root = (re_a+j im_a) (re_b+j im_b). Gauss's algorithm for complex number multiplication gives two separate final results to calculate real and imaginary part. The real part of the output can be computed using (re_a.re_b - im_a.im_b), and the imaginary part of the result can be computed using (re_a.im_b + re_b.im_a). Thus four separate multiplications and addition/subtraction are required to produce the real as well as imaginary part numbers.

In this paper the authors have proposed architecture, capable of multiplying two 32-bit complex numbers using Vedic Multiplier Algorithm. The multiplier algorithm is basically based on fundamentals of Ancient Indian Vedic Mathematics (Urdhva Tiryagbhyam sutra) and is implemented using Verilog and VHDL. The comparison of 8-bit, 16-bit, 32-bit complex multiplier on various performance parameters like power and delay are discussed from which we infer that there is 10% and 19% increase in power as we go from 8-bits to 16-bits and 16-bits to 32-bits respectively. Delay constrained varies nonlinearly as we increase the number of bits. The above results shows that Urdhva Tiryagbhyam sutra is used to implement complex multiplier efficiently in DSP algorithms by decreasing propagation delay (ns).

2.1.4. Design and Implementation of 64-bit Multiplier using Vedic Algorithm [4]

This paper presents the highly efficient 64-bit multiplier for the mantissa calculation using rule or sutra of Vedic mathematics called Urdhva Tiryagbhyam Sutra which deals with vertically and crosswise multiplication. Using this sutra in the computation algorithm of DSP processors, can enhance the efficiency and at the same time can reduce the complexity, area, power consumption and delay. Starting from the design of 2-bit Vedic multiplier the authors went up to design 64-bit Vedic multiplier as presented in this paper. Vedic multiplier is coded in Verilog HDL and targeted to three different families of FPGA Spartan6, Virtex5 and Virtex6 in Xilinx 13.1 ISE software. Result is compared with the Karatsuba, Vedic-Karatsuba and Optimized Vedic multiplier and found 33% reduction in delay.

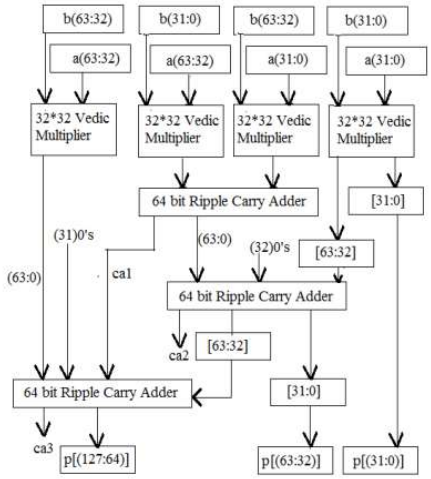

Fig. 2.4. Schematic of 64 x 64 Vedic multiplier [4]

From the available multiplier of 4 x 4 number the authors developed multiplier of 8*8 bit, then multiplier of 16 x 16-bit, multiplier of 32 x 32 bit and finally they have developed 64*64-bit multiplier. The simulation results of all the multipliers were presented.

Table 2.2: Delay Comparison of different Algorithms

Bit Size = 64 Bit				
Algorithm	*Karatsuba [5]*	*Vedic-Karatsuba [5]*	*Optimized Vedic [5]*	*Propo-sed Vedic*
Delay (ns)	150.922	92.448	44.87	15.770

A very effective method, i.e. Urdhva-Tiryagbhyam Sutra based on Vedic mathematics is used in this paper for the multiplication of two 64-bit numbers. This method allowed the author to design a multiplier of any bit number. From the comparative analysis it can be seen that the proposed multiplier has a better delay than previously designed multiplier of a same bit number. The authors' aim was to reduce the delay for 64-bit multiplier and in this case, it is found to be 15.770 ns, Hence the authors we have achieved their aim.

2.1.5. Performance comparison of Conventional and Vedic Multiplier using Simulator [5]

In this paper the implementation of an ancient Vedic Multiplier (VM) for 8-bit x 8-bit is proposed using Urdhva Tiryagbhyam Sutra (UTS). The proposed Vedic multiplier is compared with existing conventional multipliers. Multipliers are coded in Verilog and simulation is done in XILINX software 14.3. Further the performance metrics of multipliers such as area and delay are determined and compared.

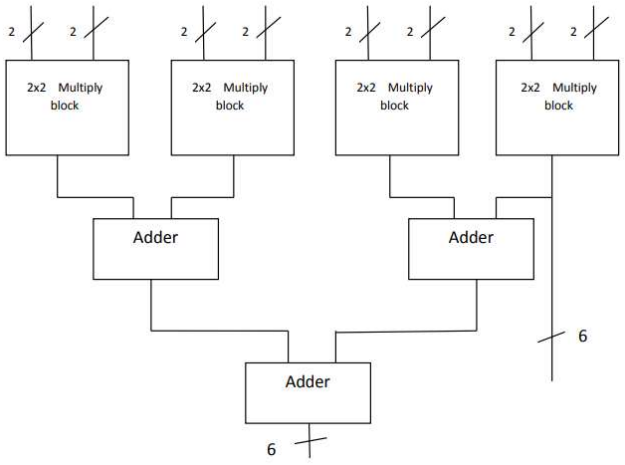

Fig. 2.5. Schematic of 8 x 8 Vedic multiplier

The proposed multiplier architecture is based on Urdhva Tiryagbhyam (vertical and cross-wise algorithm) sutra and for partial product addition Wallace tree method is used. The 4-bit x 4-bit multiplication has been done in a single line in Urdhva Tiryagbhyam sutra whereas in shift and add (conventional) method, four partial products have to be added to get the result. Thus, by using Urdhva Tiryagbhyam Sutra in binary multiplication, the number of steps required

calculating the final product will be reduced and hence there is a reduction in computational time and increase in speed of the multiplier. The steps for 4-bit x 4-bit Vedic multiplier using Urdhva Tiryagbhyam Sutra are shown in Fig.2.5 and the block diagram for 8-bit x 8-bit Vedic Multiplier is shown in Fig.8. The design starts with the implementation of 2-bit x 2-bit Vedic multiplier. Vedic Multiplier block is then instantiated for 4-bit x 4-bit, 8-bit x 8 bit.

Table 2.3: Delay Comparison of different Multipliers

SL.NO	SIZE	DESIGN	LUT'S	SLICES	DELAY(NS)
1	8 bit x8 bit	Booth	139	272	47.23
2	8 bit x8 bit	Baugh Wooley	141	81	29.19
3	8 bit x 8 bit	Vedic	167	96	27.65

Multipliers are coded in Verilog; simulation is done in XILINX software 14.3. From Table 3 it is concluded that Vedic multiplier using Urdhva Tiryagbhyam sutra shows improved performance in terms of delay by 20% when compared to Booth Multiplier and by 2% for Baugh Wooley Multiplier. The area occupied by Baugh Wooley Multiplier is 3% lesser than Vedic multiplier and 16% lesser than Booth Multiplier. The authors concluded that the power of Vedic Mathematics can be explored to implement high performance Multiplier in VLSI applications.

2.1.6. Modified High Speed 32-bit Vedic Multiplier Design and Implementation [6]

The proposed research work specifies the modified version of binary Vedic multiplier using vedic sutras of ancient Vedic mathematics. It provides modification in preliminarily implemented Vedic multiplier. The modified binary Vedic multiplier is preferable has shown improvement in the terms of the time delay and also device utilization. The proposed technique was designed and implemented in Verilog HDL. For simulation, modelsim tool is used and for circuit synthesis, Xilinx is used. The simulation has been done for 4-bit, 8-bit, 16-bit,32-bit multiplication operation. Only for 32-bit binary Vedic multiplier technique the simulation results are shown. This modified multiplication technique is extended for larger sizes. The outcomes of this multiplication technique are compared with existing Vedic multiplier techniques.

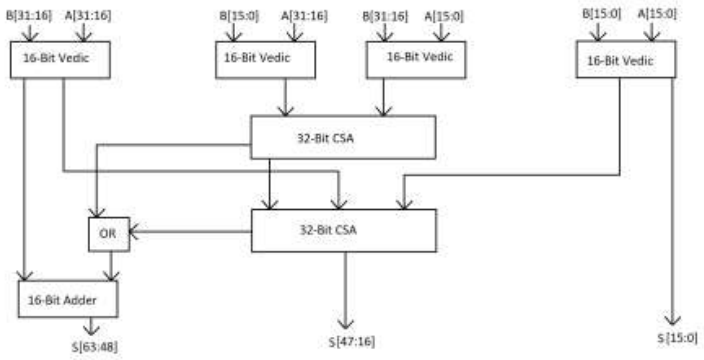

Fig. 2.6. Modified 32 Bit Vedic Multiplier [6]

First, a 4-bit Vedic multiplier is designed and in the same manner the size of Vedic multiplier is increased up to 32- bit i.e., 8-bit, 16-bit, and then 32-bit using RCA and then by using CSA, the modified 4-bit Vedic multiplier is implemented and in the same way the size of the modified vedic multiplier is increased up to 32-bit i.e., 8, 16, and 32- bit. Functionality verification was done using MODELSIM and the final synthesis was done by using XILINX ISE DESIGN SUITE.

Table 2.4 : Modified Vedic multiplier's delay and area

SNO	Modified Vedic Multiplier	Delay(in ns)	LUT's
1	4-bit	14.215	38
2	8-bit	24.186	201
3	16-bit	43.714	903
4	32-bit	68.859	3802

This paper has presented a systematic method for binary multiplier circuits which is based on Vedic mathematics. When it comes to the terms of time delay then the proposed system is more efficient than existing methods. Elongation for a higher bit size can be done with help of proposed technique. Moreover, adders of different architectures can be used in the CSA Carry Save Adder design used in the proposed modified Vedic multiplier. Among many techniques modified architecture is used to increase and speed up the multiplication. In this technique hike in area occurred it is a drawback.

2.2 Summary of Literature Review

This section gives the summary of the research papers described in detail in 2.1, which are depicted in form of a table below.

Table 2.5: Literature Review Summary

Sr. no	Title of Technical paper	Name of Author	Year of pub.	Methodology	Results	Drawbacks
1.	FPGA Implementation of high speed 8-bit Vedic multiplier using barrel shifter.[1]	Pavan Kumar U.C.S, Saiprasad Goud A, A.Radhika	2013	8-bit Vedic multiplier compared with conventional multiplier like conventional vedic multiplier, array multiplier, Braun multiplier, modified booth multiplier and Wallace tree multiplier.	Achieved an improvement in the reduction of delay with 45% when compared to array multiplier, booth multiplier and conventional Vedic multiplier implementation on FPGA.	Conventional Vedic multiplier has more propagation delay than Vedic multiplier using barrel shifter.
2.	Design and FPGA Implementation of High-Speed Vedic Multiplier.[2]	Sudeep.M. C, Sharath Bimba.M, Mahendra Vucha.	2014	8-bit multiplier using 2- and 4-bit multiplier. The major change adopted here in the architecture is that they have used Kogge stone algorithm to add partial product.	the proposed technique of multiplication using Urdhva Tiryagbhyam algorithm and Kogge Stone algorithm causes less latency when compared to available techniques.	Higher speeds can be achieved by making use of pipelining and parallel processing techniques.
3.	Design and Implementation of 32bit Complex Multiplier using Vedic Algorithm.[3]	Ankush Nikam, Swati Salunke, Sweta Bhurse	2015	Design of 32-bit Multiplier using the Indian Vedic Mathematics. Also presents comparison of 8bit, 16bit, 32-bit complex multiplier on various performance parameters like power and delay.	The 32 x 32-bit multiplier using Vedic algorithm is implemented using VHDL and Verilog and functionally verified using Xilinx ISE 14.2	As the number of bits increased, the hardware for the multiplier design increases as well.
4.	Design and Implementation of 64-Bit Multiplier using Vedic Algorithm.[4]	Amish Jais, Prasanna Palsodkar	2016	Highly efficient 64-bit multiplier for the mantissa calculation using rule or sutra of Vedic mathematics called Urdhva Tiryagbhyam Sutra.	The proposed multiplier has a better delay than previously designed multiplier of a same bit number.	64 x 64 Vedic multiplier using 2,4,8,16,32-bit multiplier has larger delay than 64 x 64-bit multiplier using 8,18,32-bit multiplier.
5.	Performance comparison of conventional multiplier and Vedic Multiplier	Dr. G.S. Sunitha, Rakesh H.M	2018	The proposed Vedic multiplier is compared with existing conventional multipliers like Booth and Baugh woolly which are	The performance metrics of multipliers such as area and delay are determined and compared.	Vedic multiplier shows improved performance but the area occupied by Baugh Wooley

	Using Simulator.[5]			coded in Verilog and simulation is done in Xilinx.		Multiplier is 3% lesser than Vedic multiplier and 16% lesser than Booth Multiplier.
6.	Modified High Speed 32-bit Vedic Multiplier Design and Implementation.[6]	M.Bala Murugesh, S.Nagaraj, J.Jayasree, G.Vijay Kumar Reddy	2020	First, a 4-bit Vedic multiplier is designed and in the same manner the size of Vedic multiplier is increased up to 32-bit i.e., 8-bit, 16-bit, and then 32-bit using RCA and then by using CSA.	The paper proposed that when it comes to the terms of time delay then the proposed system is more efficient than existing multipliers.	Drawbacks (propagation delay) due to RSA can be eradicated by using adders of different architectures can be used like Carry Save Adder to make the design faster.

Chapter 3

Project Description

3.1. Problem Statement

3.2. Steps Involved

3.3. Block Diagram of 16 x 16 Vedic Multiplier

3.4. Component Description

3.4.1 Hardware

3.4.2 Software

3.5. Working of Iot based air pollution monitoring system

3.5.1 Working

3.5.2 Calibration Method for measuring different gases

Project Description

3.1. Problem Statement

To design, implement and analyze 16 x 16 Vedic Multiplier using Urdhva Tiryagbhyam Veda. Urdhva -Tiryagbhyam Veda means “Vertical and crosswise technique”. The analysis for gate delay, computation time, look up tables, slices, total delay, no. of bounded IOB’s, etc. will be performed for the 16 x 16 Vedic Multiplier.

3.2. Steps involved

Before starting the physical implementation of this project, literature survey was carried out. A thorough research was done upon this project. Relevant papers and information on the websites(internet) were analysed and noted down carefully.

The 16 x 16 Vedic Multiplier will require the following modules:

1) 8 x 8 Vedic multiplier.
2) 4 x 4 Vedic multiplier.
3) 2 x 2 Vedic multiplier.
4) n-bit Adders.

Analysis of 16-bit and implementation of 8-bit Vedic Multiplier on EDGE Spartan 7 FPGA board was done.

3.3. Block Diagram of the 16 x 16 Vedic Multiplier

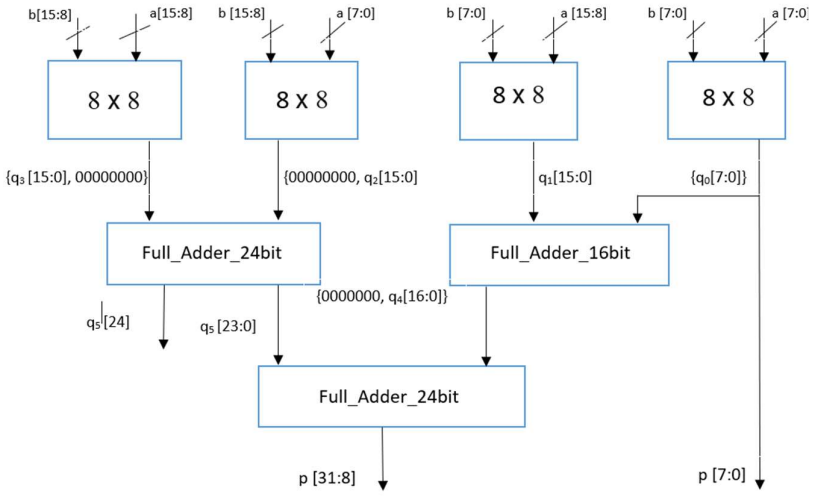

Fig. 3.1. Block Diagram of 16 x 16 Vedic Multiplier

As seen from the Figure no. 3.1, to construct a 16-bit Vedic Multiplier we will need four 8 x 8 Vedic Multipliers, two 24-bit Full adder and a 16-bit Full adder. a and b are the 16- bits input ranging from a0 to a15 and b0 to b15 respectively. p is the output of 32-bits, as 16*16 bits would result into 32-bits output. The first 8 bits are obtained directly from the first 8-bit multiplier and the remaining 24-bits come from the lowest placed 24-bit full adder as shown in the above figure.

3.4. Component Description

3.4.1. Hardware

1) EDGE Spartan 7 FPGA Development board [7]

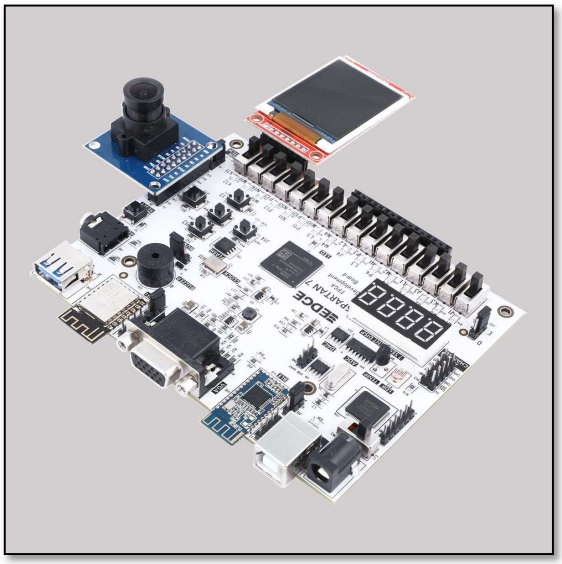

Fig. 3.2. EDGE Spartan 7 FPGA Development board [7]

EDGE Spartan 7 FPGA Development board is the low cost and feature rich development board which is upgraded from EDGE Spartan 6 kit. Both EDGE Spartan 6 and Spartan 7 FPGA kits share the same interfaces except FPGA IC. An advantage of EDGE Spartan 7 over Spartan 6 is fully compatible with Vivado design suite. Its features include FPGA, SPI FLASH, Wi-Fi, Bluetooth, ADC, DAC, LCD, 7 segment, VGA, PS2, Stereo Jack, buzzer, Push Button, Slide Switch, LED, Temperature Sensor, LDR and UART. The Board also provides additional interface like CMOS Camera and TFT Display at the expansion connectors. EDGE FPGA kit is ready to use Laboratory kit for ECE Curriculum. It can be useful for developing basic to advanced level digital circuits. Advantage of EDGE FPGA kit is easy to implement plenty of applications ranging from Wireless control, Image/video Processing, Internet of Things without additional interfaces. Xilinx offers free Webpack™ versions of Vivado design suite, so designs can be implemented at no additional cost

2) LEDs (Inbuilt on the FPGA board)
   - Used 16 LEDs to show the output of 8-bit Vedic Multiplier in binary format (0 and 1).

3) Switches (Inbuilt on the FPGA board)
   - Used 16 switches to give the input. Switches [0-7] for first input that is 'a' and switches [8-15] for the second input that is 'b'.

3.4.1. Software

1) Vivado 2019.1 HLx Edition [8]

Vivado Design Suite is a software suite produced by Xilinx for synthesis and analysis of hardware description language (HDL) designs, superseding Xilinx ISE with additional features for system on a chip development and high-level synthesis. Vivado represents a ground-up rewrite and re-thinking of the entire design flow (compared to ISE). Like the later versions of ISE, Vivado includes the in-built logic simulator. Vivado also introduces high-level synthesis, with a toolchain that converts C code into programmable logic. The Vivado High-Level Synthesis compiler enables C, C++ and System C programs to be directly targeted into Xilinx devices without the need to manually create RTL. Vivado HLx is widely reviewed to increase developer productivity, and is confirmed to support C++ classes, templates, functions and operator overloading. Vivado 2014.1 introduced support for automatically converting OpenCL kernels to IP for Xilinx devices. OpenCL kernels are programs that execute across various CPU, GPU and FPGA platforms. The Vivado Simulator is a component of the Vivado Design Suite. It is a compiled-language simulator that supports mixed-language, Tcl scripts, encrypted IP and enhanced verification. The Vivado IP Integrator allows engineers to quickly integrate and configure IP from the large Xilinx IP library. The Integrator is also tuned for MathWorks Simulink designs built with Xilinx's System Generator and Vivado High-Level Synthesis. The Vivado Tcl Store is a scripting system for developing add-ons to Vivado, and can be used to add and modify Vivado's capabilities. Tcl is the scripting language on which Vivado itself is based. All of Vivado's underlying functions can be invoked and controlled via Tcl scripts.

3.5. Working of proposed project

As discussed in 3.2, The 16 x 16 Vedic Multiplier will require the following modules: 8 x 8 Vedic multiplier, 4 x 4 Vedic multiplier, 2 x 2 Vedic multiplier and n-bit Adders. So, now

we will be looking at the working of 2-bit and 4-bit VM in detail followed by 8-bit and finally 16-bit.

3.5.1. Design of 2-bit and 4-bit Vedic Multiplier

From Fig.3.3., we can understand the working of Vedic Multiplication using Urdhva Tiryagbhyam Sutra. To elaborate the same, first, the LSB of both the 2-bit binary numbers are multiplied. Here, 0 multiplied with 0 which can be also seen as 'and operation' between the zeros. The result of the first step is stored, say R1 with no carry. Now, we perform the crosswise multiplication. i.e. 1 multiplied with 0 plus 0 multiplied with 1 which gives us some result, say R2 with no carry. The final step is vertical multiplication of MSB bits which is 1 multiplied by 1 to give the result say R3. Thus we have obtained R1 = 0, R2 = 0, and R3 = 1. Which verifies:

$$(10)_2 * (10)_2 = (100)_2$$

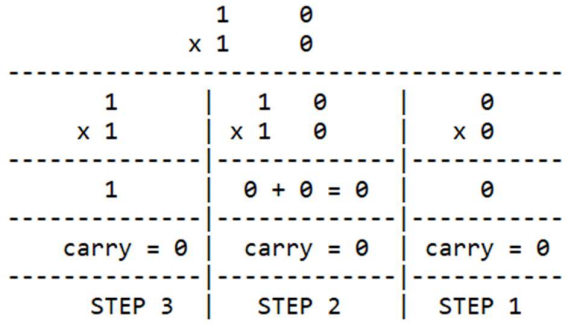

Fig. 3.3. Method for multiplying 2-bits

i.e. 2 * 2 = 4. We can simply derive the circuit schematic for a 2-bit multiplier using the method described above, as shown in Fig. 3.4. where the LSB bit is multiplied by the AND gate, and the result c0 is the final answer's LSB. The diagonal bits are multiplied with AND gates and added with the help of a half adder for cross-sectional multiplication. The second bit of the output is the sum bit c1, and the carry is transferred to the next half adder, which has the other input being the AND operation of the multiplier's and multiplicand's MSB bits.

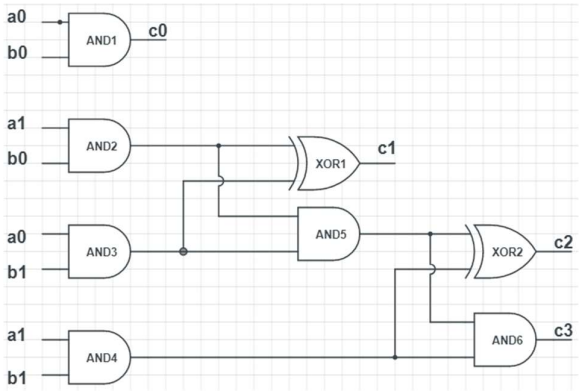

Fig. 3.4. 2-bit Vedic Multiplier Circuit

From Fig.3.5. using four 2x2 multipliers and 3 adders we can build a 4-bit multiplier. Here we multiply 2 bits at a time because we are using the 2x2 Vedic multiplier that has been already designed in the previous step.

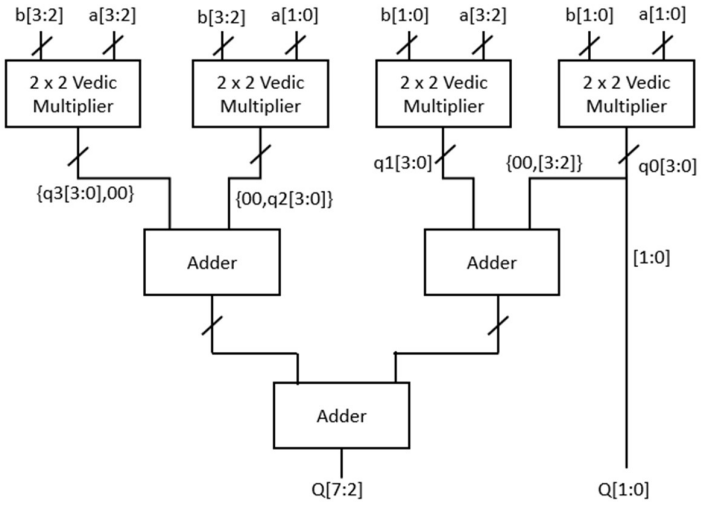

Fig. 3.5. Block Diagram of 4-bit Vedic Multiplier

We begin with the first step of multiplying the first 2 bits of A and first 2 bits of B which would give a result consisting of 4 bits. We consider the result obtained as C0. Now the last 2 bits from C0 would be the last 2 bits of the final answer as well. Then we cross multiply the next 2 bits of A and the first 2 bits of B the result obtained is called C1. The diagram shows the yellow colour bits of C0 and blue-coloured bits of C1 in the addition tree. Now the zero padding is done for C0 represented by white-coloured bubbles. The result of both these multipliers is then sent to the adder from where the 4-bit result is received. Next, the cross multiplication of the first 2 bits of A and the last 2 bits of B is carried out and the 4-bit result obtained is block C2 highlighted using green colour in the tree and the similar is sent to the 2x2 Vedic multiplier and then lastly, we multiply the last 2 bits of both A B using 2x2 Vedic multiplier for which the C3 block is highlighted using the red colour. Then again, the results from both these multipliers are sent to the adder, but this time 6 bits from C2 and another 6 bits from C3. which are inclusive of zeros padded are sent. So, finally, both the adders give us the 6 bit and 4 bits result respectively which is finally sent to the last adder from where the 6-bit output is received, and another 2 bits come from the place which was present initially (q[1:0]). Thus the block diagram explains the algorithm depicted in Fig. 3.6.

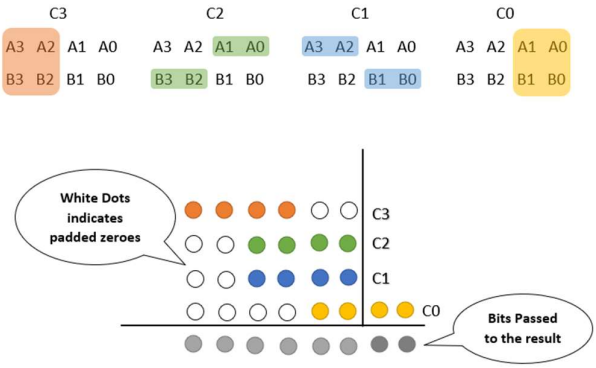

Fig. 3.6. 4-bit Vedic Multiplier Addition tree.

3.5.2. Design of 8-bit and 16-bit Vedic Multiplier

Similar procedure that was used for 4-bit multiplier using 2-bit multiplier is used for 8-bit Vedic multiplier using 4-bit. The difference is that the input will be 8-bit and the output will be of 16-bit. From the Fig. 3.7. we can see that, four 4-bit Vedic multiplier, 8-bit adder and two 12-bit adder is required to build an 8-bit Vedic multiplier.

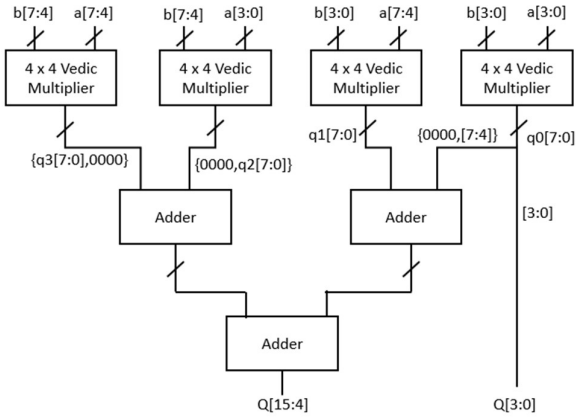

Fig. 3.7. Block Diagram of 8 x 8 Vedic Multiplier

Finally, we have our 16 x 16 vedic multiplier made using 8x8 vedic multiplier, procedure remaining the same and output we obtain is 32 bits, which is depicted in Fig. 3.8.

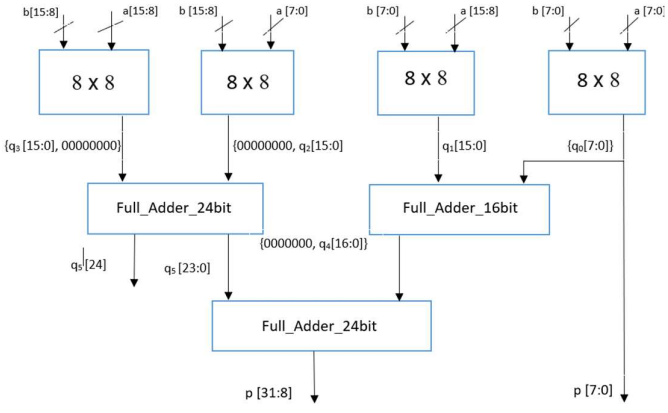

Fig. 3.8. Block Diagram of 16 x 16 Vedic Multiplier

Chapter 4

Results

4.1. RTL and Technology Schematics

4.2. Simulation Results of Vedic Multiplier

4.3. Hardware Implementation

4.4. Device Utilization Summary

4.5. Comparison of 8-bit VM, 8-bit Array Multiplier and 8-bit Wallace Multiplier.

Results

4.1. RTL and Technology Schematics

Fig.4.1 and 4.2 are the RTL and Technology schematics obtained from Vivado 2019.1 Hlx Edition. As explained in the block diagram (Fig. 3.8), to make a 16-bit Vedic Multiplier, we have used four 8-bit Vedic Multipliers (z1, z2, z3 and z4) and one 16-bit adder (z5) along with two 24-bit adders (z6, z7). These blocks are connected according to the block diagram as mentioned above.

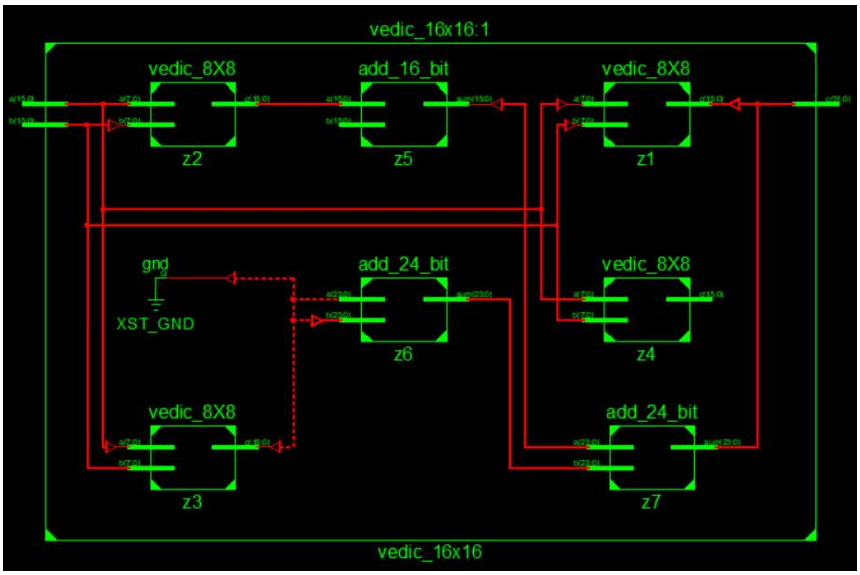

Fig.4.1. RTL Schematic of 16-bit Vedic Multiplier

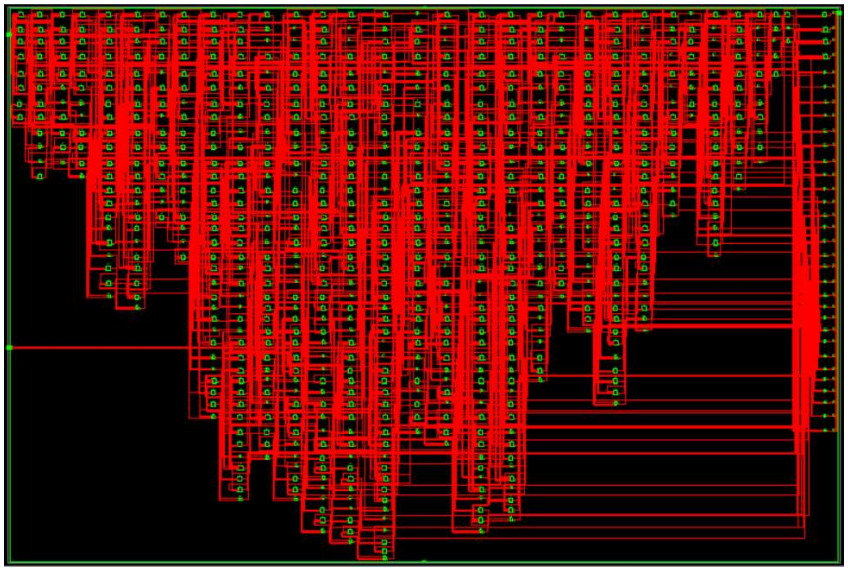

Fig.4.2. Technology Schematic of 16-bit Vedic Multiplier

Fig. 4.2 represents the look-up tables (flip flops and registers) used and connected inside the FPGA board.

4.2. Simulation Results of Vedic Multiplier

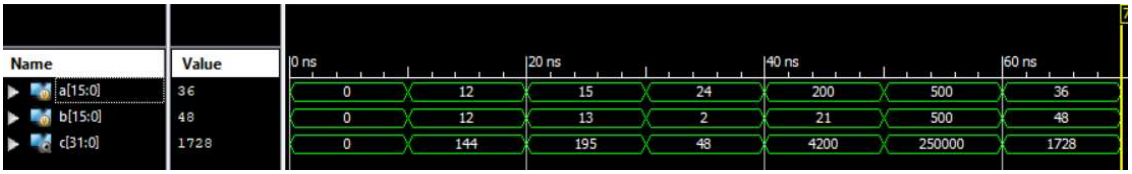

Fig.4.3. Simulation results of 16-bit Vedic Multiplier

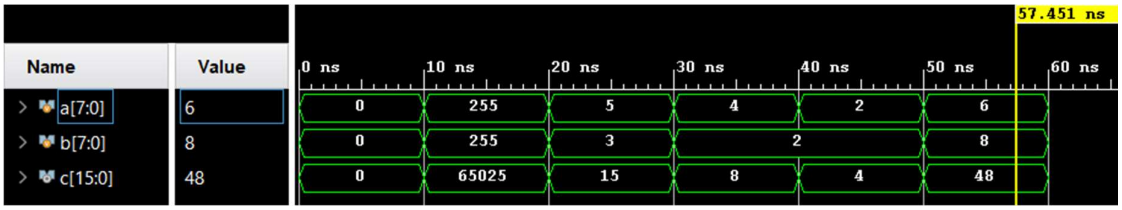

Fig.4.4. Simulation results of 8-bit Vedic Multiplier

The above figures, Fig. 4.3. and Fig. 4.4. are the simulation of 16-bit and 8-bit VM simulated in Vivado 2019.1 Hlx edition. Fig. 4.3. depicts 16-bit VM where a and b are the 16-bits input and c are the 32-bit output, whereas in fig.4.4. a and b are 8-bit inputs and c is the 16-bit output. For better explanation, let's see fig.4.4. observe the numbers between 10ns to 20ns, it is 255*255 = 65025. Now see the numbers between 30ns to 40ns, it is 4*2 = 8.

4.3. Hardware Implementation.

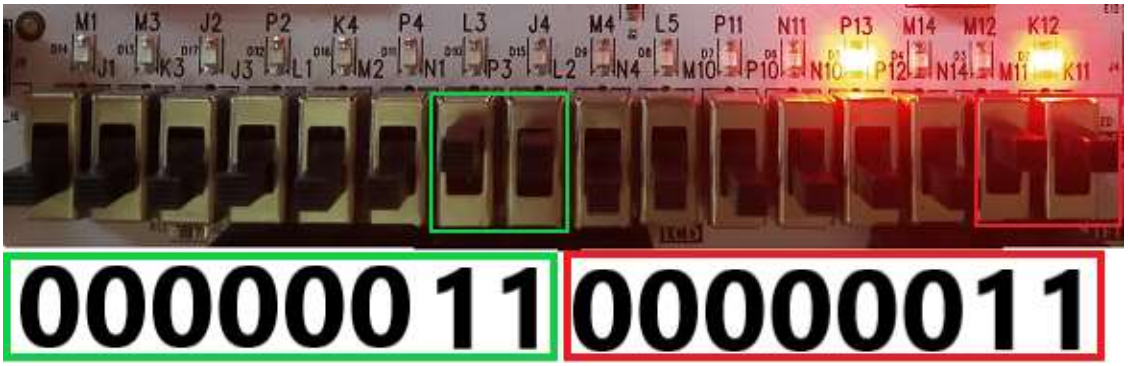

Fig.4.5. Results of 8-bit VM for $(3)_{10}$ * $(3)_{10}$ = $(9)_{10}$ = $(0000000000001001)_2$

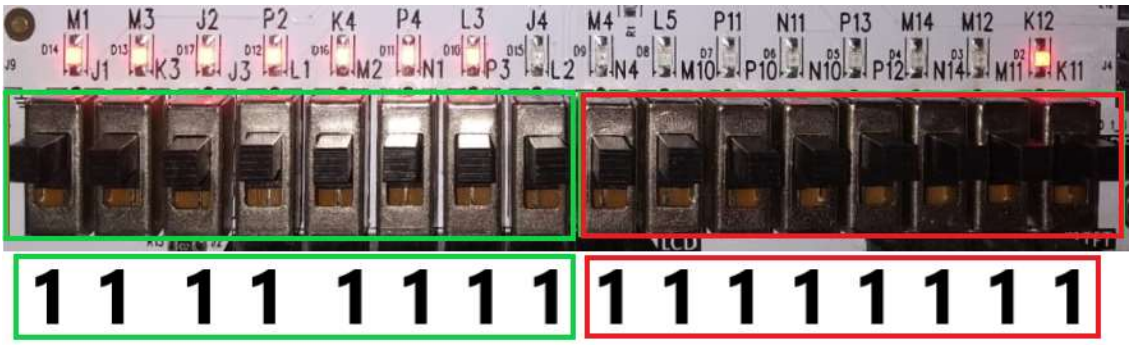

Fig.4.6. Results of 8-bit VM for $(255)_{10}$ * $(255)_{10}$ = $(65025)_{10}$ = $(1111111111111111)_2$

Fig. 4.5. and 4.6. are the images of the implementation of 8-bit Vedic Multiplier on Spartan EDGE 7 FPGA Board. In Fig. 4.5. two numbers; 1st number and 2nd number is decimal $3_{(10)}$ which is $00000011_{(2)}$ in binary are multiplied together. The result of which is $3*3 = 9_{(10)}$ in decimal which when converted to binary gives $0000000000001001_{(2)}$. Similarly, for Fig. 4.6. the two numbers are $255_{(10)}$ which $1111111111111111_{(2)}$ in binary that results into $65025_{(10)}$ in decimal which is $1111111000000001_{(2)}$.

4.4. Device Utilization Summary

Fig. 4.7. shows the utilization summary i.e. the number of slices used (hardware used inside the board) as well as the bounded I/O used for 6slx9tqg144-2 whereas Fig. 4.8. shows the total time taken (propagation delay) for 16-bit VM to generate an output.

```
Device utilization summary:
---------------------------

Selected Device : 6slx9tqg144-2


Slice Logic Utilization:
 Number of Slice LUTs:                  502  out of   5720     8%
    Number used as Logic:               502  out of   5720     8%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:    502
   Number with an unused Flip Flop:     502  out of    502   100%
   Number with an unused LUT:             0  out of    502     0%
   Number of fully used LUT-FF pairs:     0  out of    502     0%
   Number of unique control sets:         0

IO Utilization:
 Number of IOs:                          64
 Number of bonded IOBs:                  64  out of    102    62%
```

Fig.4.7. Device Utilization Summary for 16-bit VM

```
Timing Details:
---------------
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default path analysis
  Total number of paths / destination ports: 63663750 / 32
-------------------------------------------------------------------------
Delay:               21.662ns (Levels of Logic = 36)
  Source:            a<6> (PAD)
  Destination:       c<31> (PAD)

  Data Path: a<6> to c<31>
```

Fig.4.8. Propagation Delay of 16-bit VM

4.5. Comparison of 8-bit VM, 8-bit Array Multiplier and 8-bit Wallace Multiplier.

Table 4.1: Comparison of different types of 8-bit multiplier

Device Used - 6slx9tqg144-2			
Multipliers	**Path delay(ns)**	**Slice LUTs**	**On-chip Power (mW)**
Vedic Multiplier	17.154	91	50.84
Wallace Multiplier	23.545	93	54.55
Array Multiplier	28.870	84	65.41

Table no.4.1 depicts the characteristics of the three proposed multipliers in this project. According to the comparative table, the 8-bit Wallace Multiplier has a maximum combinational path latency of 37.25 percent more than the 8-bit Vedic Multiplier, while the 8-bit Array Multiplier has a maximum combinational path delay of 68.29 percent more. In addition, the Vedic Multiplier uses less overall on-chip power than the other two multipliers. The Wallace tree multiplier takes up the greatest area on the chip, while the Array multiplier takes up the least. The Wallace tree Multiplier and the Vedic Multiplier employed almost the same amount of Slice LUTs out of a total of 5720 Slice LUTs.

Chapter 5

Conclusion and Future Scope

Conclusion and Future Scope of 16 x 16 Vedic Multiplier

5.1 Conclusion

5.2 Future Scope

Conclusion and Future Scope

5.1 Conclusion of 16-bit Vedic Multiplier

The proposed project was successfully designed, simulated and implemented in Verilog Language. The simulation was done on Vivado Hlx and Xilinx ISE. The design for 8-bit Vedic Multiplier was verified using testbenches and tested using Spartan EDGE 7 FPGA Board. Also, the other two multipliers that is Wallace tree multiplier and Array multiplier were designed using Xilinx ISE.

From the above results, we can conclude that Vedic Multiplier has less propagation delay which is directly proportional to greater speed than the other two multiplier. It's noteworthy that, unlike adders, lower bit multipliers cannot be cascaded to construct a multiplier capable of multiplying more bits, which is a disadvantage of developing a higher bit multiplier. The Vedic multiplier can be used to address this problem. A Vedic multiplier of 8 bits, for example, can be used to make a 16-bit, 32-bit, and so on. Wallace Tree Multiplier is significantly faster than a regular array multiplier since its height is logarithmic in word size rather than linear. As a result, when design complexity is an issue, designers frequently avoid Wallace trees. As a result, the Vedic Multipliers can be used in cases where speed and power consumption are both important factors. Hence, we can conclude that Vedic Multipliers are superior to other multiplier where speed and power consumption is an important factor.

5.2 Future Scope

In future, we can build a better and a faster ALU, Vedic Multipliers can be used. These multipliers can be used in DSP and ADSP Processors which are used to calculate various transforms like FFTs and IFFTs. Furthermore, we can compare other multipliers like DADDA Multiplier, Boough-wooley Multiplier, etc. and give a detailed analysis on area, power consumption and propagation delay.

References

[1] U. C. S. Pavan Kumar, A. Saiprasad Goud and A. Radhika, "FPGA implementation of high speed 8-bit Vedic multiplier using barrel shifter," 2013 International Conference on Energy Efficient Technologies for Sustainability, 2013, pp. 14-17, doi: 10.1109/ICEETS.2013.6533349.

[2] M.C, Sudeep & Bimba, Sharath & Vucha, Mahendra. (2014). Article: Design and FPGA Implementation of High-Speed Vedic Multiplier. International Journal of Computer Applications. 90. 6-9. 10.5120/15802-4641.

[3] Ankush Nikam, Swati Salunke, Sweta Bhurse, 2015, Design and Implementation of 32bit Complex Multiplier using Vedic Algorithm, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) Volume 04, Issue 03 (March 2015).

[4] A. Jais and P. Palsodkar, "Design and implementation of 64 bit multiplier using vedic algorithm," 2016 International Conference on Communication and Signal Processing (ICCSP), 2016, pp. 0775-0779, doi: 10.1109/ICCSP.2016.7754250.

[5] Dr. G.S. Sunitha, Rakesh H.M, "Performance comparison of conventional multiplier and Vedic Multiplier Using Simulator.", International Journal of Engineering and Manufacturing Science, © Research India Publications, ISSN 2249-3115 Volume 8, Number 1 (2018) pp. 95-103.

[6] M. B. Murugesh, S. Nagaraj, J. Jayasree and G. V. K. Reddy, "Modified High Speed 32-bit Vedic Multiplier Design and Implementation," 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), 2020, pp. 929-932, doi: 10.1109/ICESC48915.2020.9155882.

[7] https://allaboutfpga.com/edge-spartan-7-fpga-development-board-user-manual/#Programming_Quad_SPI_FLASH_Memory_using_Vivado_Design_Suite , Last Accessed on: 17th April 2022.

[8] https://en.wikipedia.org/wiki/Xilinx_Vivado , Last Accessed on: 17th April 2022.

[9] https://en.wikipedia.org/wiki/Vedic_Mathematics , Last Accessed on: 17th April 2022.