IMPLEMENTATION OF VEDIC MULTIPLIER ON FPGA BOARD

A
Report Submitted
In Partial Fulfillment of the Requirements
For the Degree of

BECHALOR OF TECHNOLOGY

in
Electronics and Communications Dept

by
Nabeel Nomani (115043408)
Sorabh Tiwari(1150434016)
Vibha Dubey(1150434022)
Manish Singh(1160434006)
Sameer Kr Panday(1160434018)

Under the supervision of
Mr Amit Kumar
Sir
BBD University, Lucknow

School of Engineering
BABU BANARASI DAS UNIVERSITY
LUCKNOW

May, 2020

CERTIFICATE

It is certified that the work contained in this Project entitled **"implementation of vedic multiplier on fpga board using vhdl"** by Nabeel Nomani, SaurabhTiwari, Vibha Dubey & ManishSingh

(Roll No.1150434008,1150434016,1150434022,1160434006), for the award of **Bachelor of Technology** from Babu Banarasi Das University has been carried out under my supervision.

Mr Amit Kumar
Electronics and Comm.
School of Engineering
BBD University
Lucknow (U.P)

Dr Nitin Jain
Head, Electronics and Comm.
School of Engineering
BBD University
Lucknow (U.P)

Date:16/04/20

ABSTRACT

The need for high speed multiplier is increasing as the need of high speed processors are increasing in the market. A multiplier is one of the key hardware blocks in most fast processing system which is not only a high delay block but also a major source of power dissipation.

Today's technology has raised demand for fast and real time signal processing operation. Multiplication is one of the most important arithmetic operations. In this paper, we have proposed design of vedic multiplier using "Urdhva Tiryagbhyam" sutra in Xilinx ISE. This design takes lesser time for operation than currently available multipliers .It encompasses wide era of image processing and digital signal processing in much efficient way with increase in speed and thus leading to higher performance rating

KEYWORDS: Vedic Multiplier, Urdhva Tiryagbhyam, Digital Signal Processing, Image processing

A conventional processor requires substantially more hardware resources and processing time in the multiplication operation rather than addition and substraction. Multiplication is one of the most important arithmetic operation in signal processing. All signal and data processing requires multiplication. First method URDHAVA TRIYAKBHYAM sutra which is similar to array multiplication .When number of bits increases, gate delay and area increases slowlycompared to other multiplier. So the advanced technique called NIKHILAM sutra is employed. These sutras are meantfor faster mental calculation. Though faster when implemented in hardware, it consumes more power than theconventional ones. In this project both the techniques are compared and found that nikhilam is best. This projectpresents a technique to modify the architecture of the Vedic multiplier by using some existing methods in orderincrease the processor speed.

ACKNOWLEDGEMENT

I am grateful to my thesis guide **Amit Kumar Sin** IIIT Allahabad, for showing faith in me, supporting me throughout my thesis

I am thankful to my faculty and my friends for helping me in all manners of life and for their kind cooperation and moral support

Nabeel Nomani

Saurabh Tiwari

Vibha Dubey

Manish Singh

Sameer Panday

TABLE OF CONTENTS

List of Figures

List of Tables

INTRODUCTION

Multiplication is an important fundamental function in arithmetic operations. Multiplication based operations such as Multiply and Accumulate(MAC) and inner product are among some of the frequently used Computation- Intensive Arithmetic Functions(CIAF) currently implemented in many Digital Signal Processing (DSP) applications such as convolution, Fast Fourier Transform(FFT), filtering and in microprocessors in its arithmetic and logic unit .The performance of multiplication is crucial for multimedia applications such as 3D graphics and signal processing systems,which depend on the execution of large numbers of multiplications Since multiplication dominates the execution time of most DSP algorithms, so there is a need of high speed multiplier. Currently, multiplication time is still the dominant factor in determining the instruction cycle time of DSP chip.

In microprocessors multiplication operation is performed in a variety of forms in hardware and software depending on the cost and transistor budget allocated for this particular operation. In the beginning stages of computer development of any complex operation was usually programmed in software or coded in the micro-code of the machine. Design developed for a multiplier which generates the product of two numbers using purely combinational logic.Today it is more likely to find full hardware implementation of the multiplication in order to satisfy growing demand for speed and due to the decreasing cost of hardware. The execution time of most DSP algorithms is dependent on its multipliers, and hence need for high speed multiplier arises.

Vedic mathematics is part of four Vedas (books of wisdom). It is part of Sthapatya- Veda (book on civil engineering and architecture), which is an upa-veda (supplement) of Atharva Veda. It gives explanation of several mathematical terms including arithmetic, geometry (plane, co-ordinate), trigonometry, quadratic equations, factorization and even calculus. His Holiness Jagadguru Shankaracharya Bharati Krishna Teerthaji Maharaja (1884- 1960) comprised all this work together and gave its mathematical explanation while discussing it for various applications. Swamiji constructed 16 sutras (formulae) and 16 Upa sutras (sub formulae) after extensive research in Atharva Veda. Obviously these formulae are not to be found in present text of Atharva Veda because these formulae were constructed by Swamiji himself. Vedic mathematics is not only a mathematical wonder but also it is logical. That's why it has such a degree of eminence which cannot be disapproved. Due these phenomenal characteristics, Vedic maths has already crossed the boundaries of India and has become an interesting topic of research abroad. Vedic maths deals with several basic as well as complex mathematical operations. Especially, methods of basic arithmetic are extremely simple and powerful. The word "Vedic" is derived from the word "Veda" which means the store-house of all knowledge. Vedic mathematics is mainly based on 16 Sutras (or aphorisms) dealing with various branches of mathematics like arithmetic, algebra, geometry etc. These Sutras along with their brief meanings are enlisted below alphabetically. (Anurupye) Shunyamanyat – If one is in ratio, the other is zero.

In microprocessors multiplication operation is performed in a variety of forms in hardware and

software depending on the cost and transistor budget allocated for this particular operation. In the beginning stages of computer development of any complex operation was usually programmed in software or coded in the micro-code of the machine. Design developed for a multiplier which generates the product of two numbers using purely combinational logic Today it is more likely to find full hardware implementation of the multiplication in order to satisfy growing demand for speed and due to the decreasing cost of hardware. The execution time of most DSP algorithms is dependent on its multipliers, and hence need for high speed multiplier arises.In many DSP algorithms, the multiplier lies in the critical delay path and ultimately determines the performance of algorithm.

The speed of multiplication operation is of great importance in DSP as well as in general processor. In the past multiplication was implemented generally with a sequence of addition, subtraction and shift operations. There have been many algorithms proposals in literature to perform multiplication, each offering different advantages and having trade-off in terms of speed, circuit complexity, and area and power consumption.

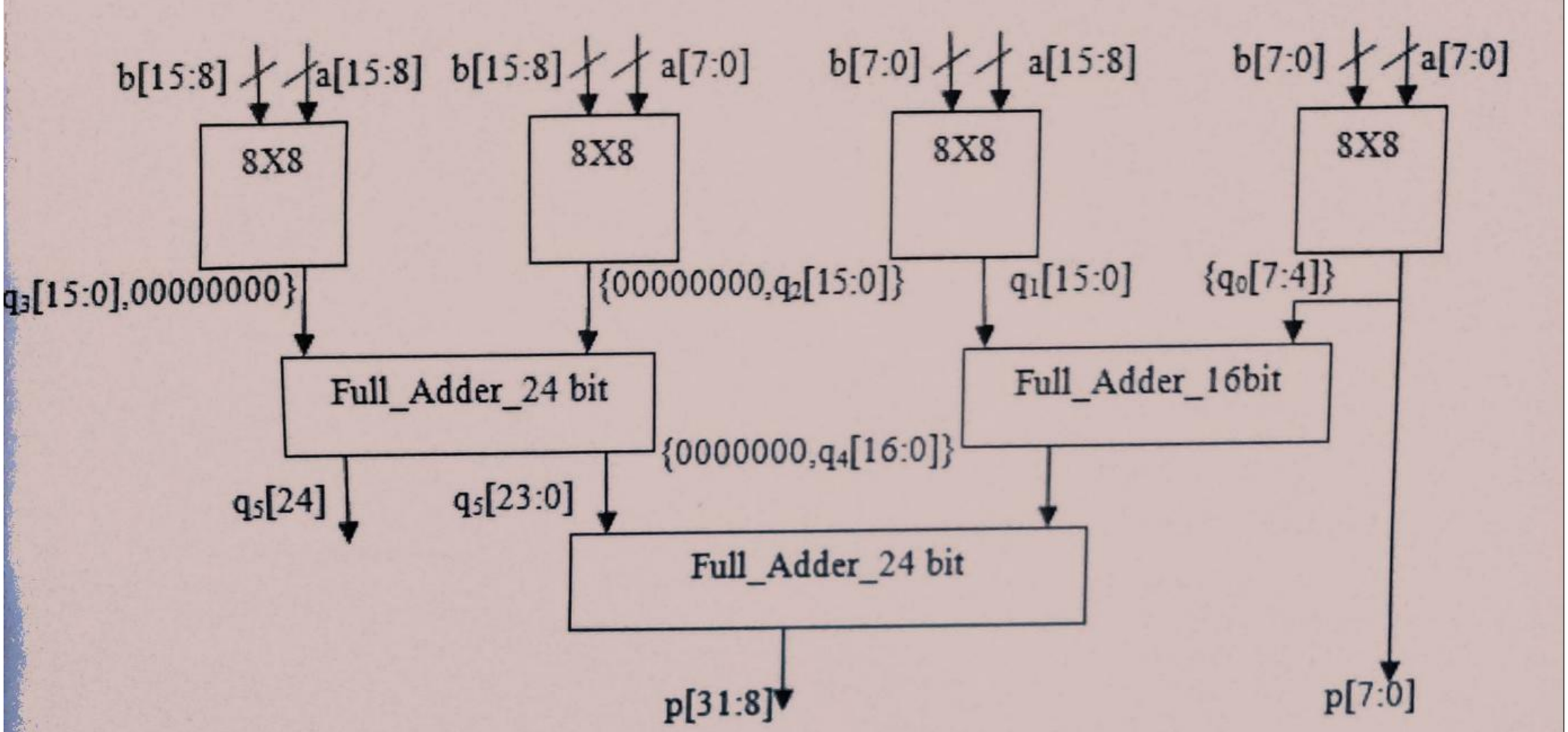

Figure 1

WHAT IS A MULTIPLIER?

A binary multiplier is an electronic circuit used in digital electronics, such as a computer, to multiply two binary numbers. It is built using binary adders.

A variety of computer arithmetic techniques can be used to implement a digital multiplier. Most techniques involve computing a set of partial products, and then summing the partial products together. This process is similar to the method taught to primary schoolchildren for conducting long multiplication on base-10 integers, but has been modified here for application to a base-2 (binary) numeral system.

In microprocessors multiplication operation is performed in a variety of forms in hardware and software depending on the cost and transistor budget allocated for this particular operation. In the beginning stages of computer development of any complex operation was usually programmed in software or coded in the micro-code of the machine. Design developed for a multiplier which generates the product of two numbers using purely combinational logic Today it is more likely to find full hardware implementation of the multiplication in order to satisfy growing demand for speed and due to the decreasing cost of hardware. The execution time of most DSP algorithms is dependent on its multipliers, and hence need for high speed multiplier arise.

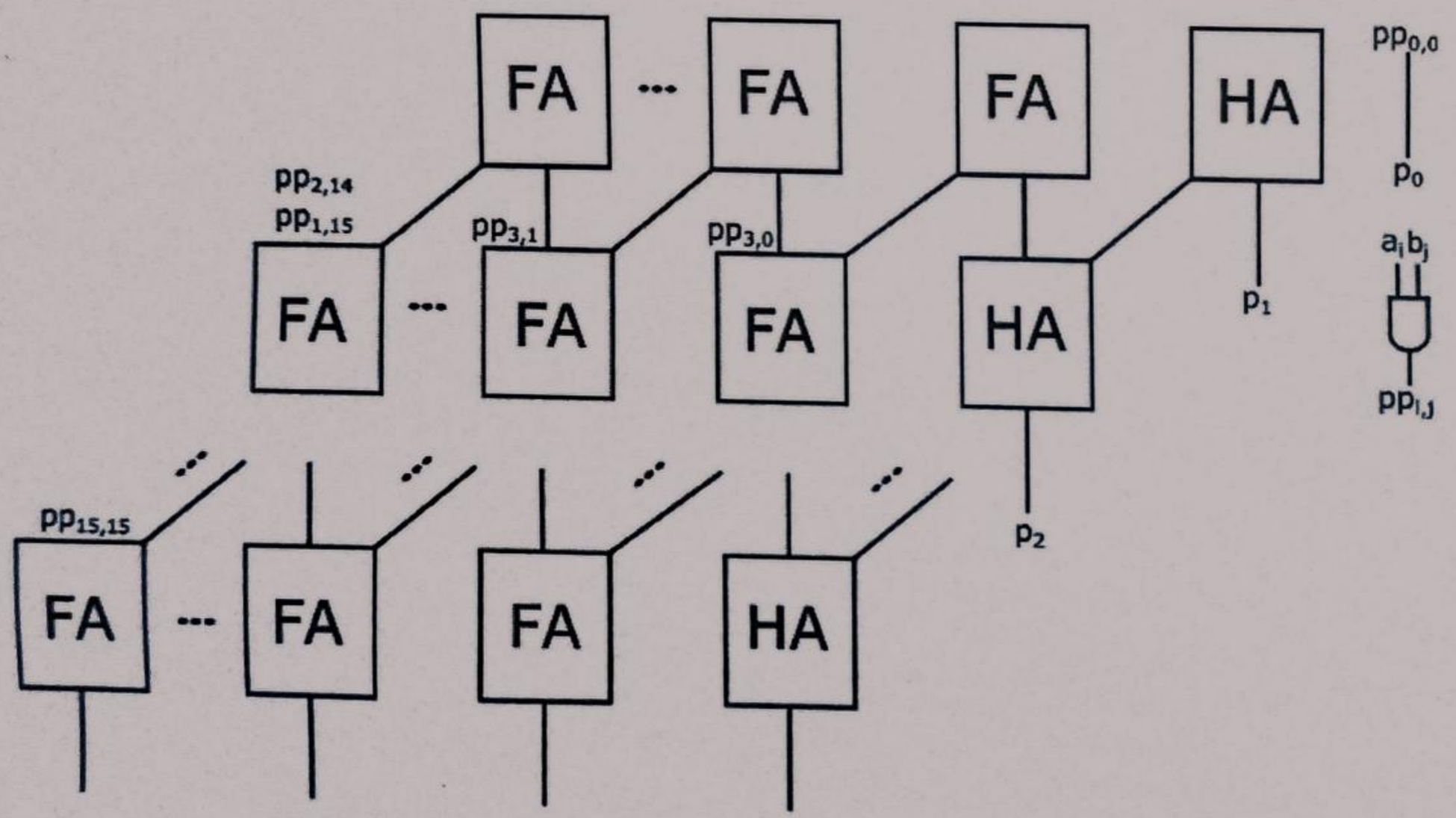

Figure 2

ARRAY MULTIPLIER

An array multiplier is a digital combinational circuit used for multiplying two binary numbers by employing an array of full adders and half adders. This array is used for the nearly simultaneous addition of the various product terms involved. To form the various product terms, an array of AND gates is used before the Adder array.

Checking the bits of the multiplier one at a time and forming partial products is a sequential operation that requires a sequence of add and shift micro-operations. The multiplication of two binary numbers can be done with one micro-operation by means of a combinational circuit that forms the product bits all at once. This is a fast way of multiplying two numbers since all it takes is the time for the signals to propagate through the gates that form the multiplication array. However, an array multiplier requires a large number of gates, and for this reason it was not economical until the development of integrated circuits.

IMPLEMENTATION:

The two's complement multiplication is converted to an equivalent parallel array addition problem in which each partial product bit is the AND of a multiplier bit and a multiplicand bit, and the signs of all the partial product bits are positive[2].In array multiplier, consider two binary numbers A and B, of m and n bits. There are mn summands that are produced in parallel by a set of mn AND gates. n x n multiplier requires n (n-2) full adders, n half-adders and n2 AND gates. Also, in array multiplier worst case delay would be (2n+1) td.Array Multiplier gives more power consumption as well as optimum number of components required, but delay for this multiplier is larger. It also requires larger number of gates because of which area is also increased; due to this array multiplier is less economical Thus, it is a fast multiplier but hardware complexity is high.

To clarify more on the concept, let us take the example of a 2X2 bit multiplication with A and B being the multiplicand and the multiplier respectively. Assuming A = a(1)a(0) and B= b(1)b(0), the various bits of the final product term P can be written as:-

$P(0)= a(0)b(0)$

$P(1)=a(1)b(0) + b(1)a(0)$

$P(2) = a(1)b(1) + C1$ where C1 is the carry generated during the addition for the P(1) term.

P(3)=C2 where C2 is the carry generated during the addition for the P(2) term.

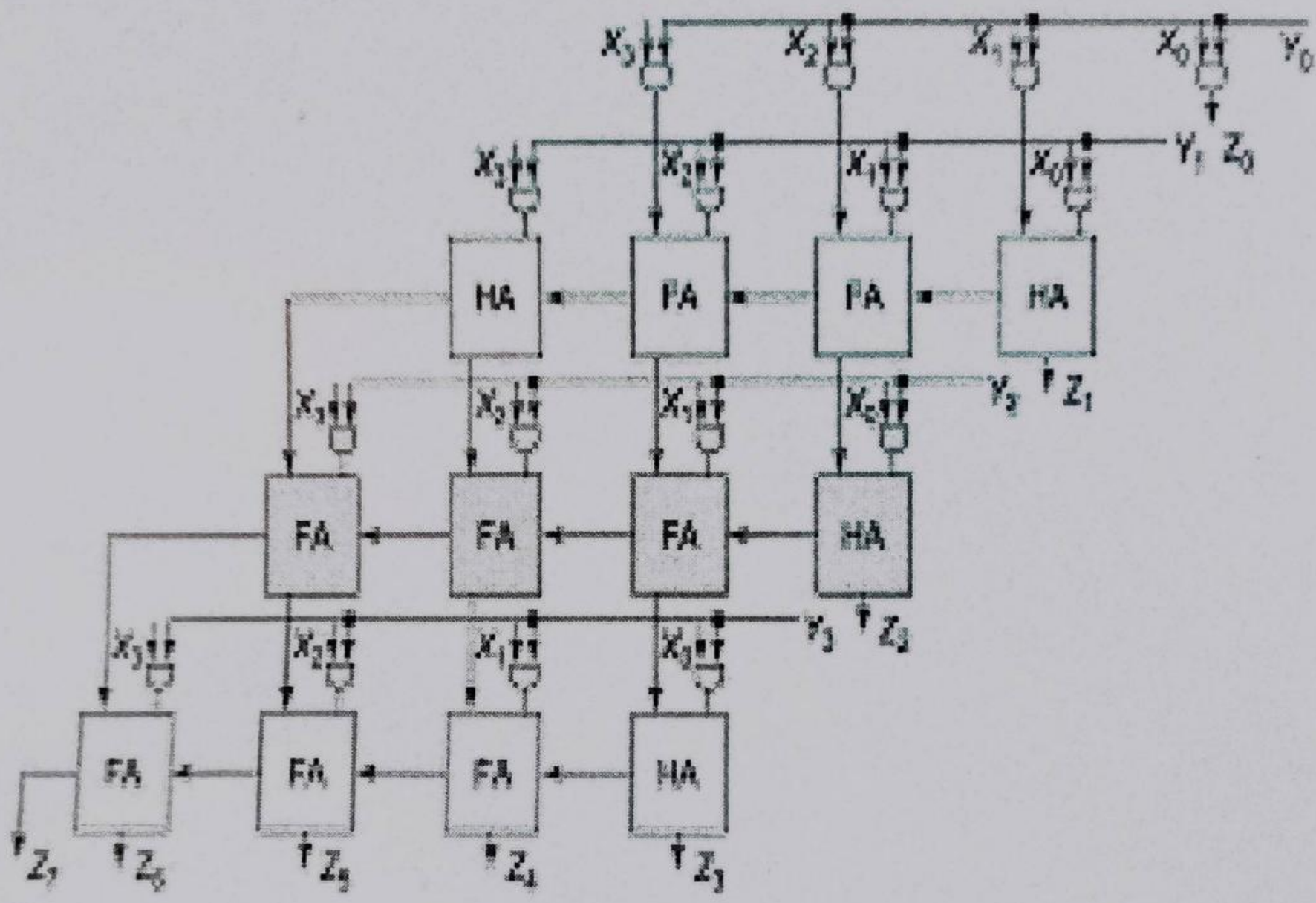

Figure 3

VEDIC MATHEMATICS

Vedic mathematics is part of four Vedas (books of wisdom). It is part of Sthapatya- Veda (book on civil engineering and architecture), which is an upa-veda (supplement) of Atharva Veda. It gives explanation of several mathematical terms including arithmetic, geometry (plane, co-ordinate), trigonometry, quadratic equations, factorization and even calculus.

His Holiness Jagadguru Shankaracharya Bharati Krishna Teerthaji Maharaja (1884- 1960) comprised all this work together and gave its mathematical explanation while discussing it for various applications. Swamiji constructed 16 sutras (formulae) and 16 Upa sutras (sub formulae) after extensive research in Atharva Veda. Obviously these formulae are not to be found in present text of Atharva Veda because these formulae were constructed by Swamiji himself. Vedic mathematics is not only a mathematical wonder but also it is logical. That's why it has such a degree of eminence which cannot be disapproved.

Due these phenomenal characteristics, Vedic maths has already crossed the boundaries of India and has become an interesting topic of research abroad. Vedic maths deals with several basic as well as complex mathematical operations. Especially, methods of basic arithmetic are extremely simple and powerful. The word "Vedic" is derived from the word "Veda" which means the store-house of all knowledge. Vedic mathematics is mainly based on 16 Sutras (or aphorisms) dealing with various branches of mathematics like arithmetic, algebra, geometry etc.

According to Krishna Tirtha, the sutras and other accessory content were found after years of solitary study of the Vedas—a set of sacred ancient Hindu scriptures—in a forest. They were supposedly contained in the pariśiṣṭa—a supplementary text/appendix—of the Atharvaveda. He does not provide any more bibliographic clarification on the sourcing. The book's editor, Professor V. S. Agrawala argues that since the Vedas are defined as the traditional repositories of all knowledge, any knowledge can be de-facto assumed to be in the Vedas, irrespective of whether it may be physically located in them; he even went to the extent of deeming Krishna Tirtha's work as a pariśiṣṭa in itself.

However, numerous mathematicians and STS scholars (Dani, Kim Plofker, K.S. Shukla, Jan Hogendijk et al) note that the Vedas do not contain any of those sutras and sub-sutras.[2][5][6][3] When challenged by Shukla, a mathematician and a historiographer of ancient Indian mathematics, to locate the sutras in the Parishishta of a standard edition of the Atharvaveda, Krishna Tirtha claimed that they were not included in the standard editions but only in a hitherto-undiscovered version, chanced upon by him; the foreword and introduction of the book also takes a similar stand. Sanskrit scholars have also confirmed that the linguistic style did not correspond to the claimed time-spans but rather reflected contemporary Sanskrit.

Dani points out that the contents of the book have "practically nothing in common" with the mathematics of the Vedic period or even with subsequent developments in Indian mathematics. Shukla reiterates the observations, on a per-chapter basis. For example, multiple techniques in

the book involve the use of high-precision decimals. These were unknown during the Vedic times and were introduced in India only in the sixteenth century; works of numerous ancient mathematicians such as Aryabhata, Brahmagupta and Bhaskara were entirely based on fractions. Some of the sutras even claimed to run parallel to the General Leibniz rule and Taylor's theorem (which, per Krishna Tirtha, were to be yet studied by the western world during the time of his writing) but did ultimately boil down to the sub-elementary operations of basic differentiation on polynomials. From a historiographic perspective, India had no minimal knowledge about the conceptual notions of differentiation and integration.Sutras have been further leveraged to claim that analytic geometry of conics occupied an important tier in Vedic mathematics, which runs contrary to all available evidence.

These claims have been since rejected in their entirety.Krishna Tirtha failed to produce the claimed sources, and scholars unanimously note it to be a mere compendium of tricks for increasing the speed of elementary mathematical calculations with no overlap with historical mathematical developments during the Vedic period. However, there has been a proliferation of publications in this area and multiple attempts to integrate the subject into mainstream education by right-wing Hindu nationalist governments.

The book contains metaphorical aphorisms in the form of sixteen sutras and thirteen sub-sutras, which Krishna Tirtha claimed to allude to significant mathematical tools.The range of their asserted applications spans from topic as diverse as statics and pneumatics to astronomy and financial domains.Tirtha claimed that no part of advanced mathematics lay beyond the realms of his book and propounded that studying it for a couple of hours every day for a year equated to spending about two decades in any standardized education system to become professionally trained in the discipline of mathematics.

Contra the hyperbolic claims of the author and publisher, the book is primarily a compendium of tricks that can be applied in elementary, middle and high school arithmetic and algebra, to gain faster results.[2] The sutras and sub-sutras are abstract literary expressions ("as much less", "one less than previous one" et al.) prone to creative interpretations; Krishna Tirtha exploited this to the extent of manipulating the same shloka to generate widely different mathematical equivalencies across a multitude of contexts.

Different types of Vedic Sutras

These Sutras along with their brief meanings are enlisted below alphabetically. (Anurupye) Shunyamanyat – If one is in ratio, the other is zero.

There are basically 15 vedic sutras used for various mathematical calculations, these sutras make calculations quick and easy.

1. Chalana-Kalanabyham – Differences and Similarities.

2. Ekadhikina Purvena – By one more than the previous One.

3. Ekanyunena Purvena – By one less than the previous one.

4. Gunakasamuchyah – The factors of the sum is equal to the sum of the factors.

5. Gunitasamuchyah – The product of the sum is equal to the sum of the product.

6. Nikhilam Navatashcaramam Dashatah – All from 9 and last from 10.

7. Paraavartya Yojayet – Transpose and adjust.

8. Puranapuranabyham – By the completion or noncompletion.

9. Sankalana- vyavakalanabhyam – By addition and by subtraction.

10. Shesanyankena Charamena – The remainders by the last digit.

11. Shunyam Saamyasamuccaye – When the sum is the same that sum is zero.

12. Sopaantyadvayamantyam – The ultimate and twice the penultimate.

13. Urdhva-tiryagbhyam – Vertically and crosswise.

14. Vyashtisamanstih – Part and Whole.

15. Yaavadunam – Whatever the extent of its Deficiency

16.(Anurupye) Shunyamanyat - If one is in ratio, the other is zero.

Sixteen Sutras and their corollaries

Sl. No	Sutras	Sub sutras or Corollaries
1.	Ekādhikena Pūrvena (also a corollary)	Ānurūpyena
2.	Nikhilam Navataścaramam Daśatah	Śisyate Śesamjnah
3.	Ūrdhva - tiryagbhyām	Ādyamādyenantyamantyena
4.	Parāvartya Yojayet	Kevalaih Saptakam Gunỹat
5.	Sūnyam Samyasamuccaye	Vestanam
6.	(Ānurūpye) Śūnyamanyat	Yāvadūnam Tāvadūnam
7.	Sankalana - vyavakalanābhyām	Yāvadūnam Tāvadūnīkrtya Vargañca Yojayet
8.	Puranāpuranābhyām	Antyayordasake' pi
9.	Calanā kalanābhyām	Antyayoreva
10.	Yāvadūnam	Samuccayagunitah
11.	Vyastisamastih	Lopanasthāpanabhyām
12.	Śesānyankena Caramena	Vilokanam
13.	Sopantyadvayamantyam	Gunitasamuccayah Samuccayagunitah
14.	Ekanyūnena Pūrvena	
15.	Gunitasamuccayah	
16.	Gunakasamuccayah	

Table 1

VEDIC MULTIPLIER

The use of Vedic mathematics lies in the fact that it reduces the typical calculations in conventional mathematics to very simple ones. This is so because the Vedic formulae are claimed to be based on the natural principles on which the human mind works. Vedic Mathematics is a methodology of arithmetic rules that allow more efficient speed implementation. This is a very interesting field and presents some effective algorithms which can be applied to various branches of engineering such as computing.

Vedic mathematics is an old mathematics which is more effective than other mathematic procedures. Vedic maths is utilized as a part of numerous applications, for example, hypothesis of numbers, compound duplications, squaring, cubing, square root and solid shape root and so on. Absolutely there are 16 sutras and 14 sub-sutras in Vedic maths. Among those sutras, just 3 sutras and 2 sub-sutras are utilized for augmentation. Multiplier is a very important part of a microprocessor as multiplication is performed continuously in all calculative procedures. This paper is in importance of a 8-bit multiplier designed in 90 nm technology. Urdhva-Tiryakbyham is the sutra that is used for multiplication in Vedic mathematics. Actualizing the different scientific operations utilizing Vedic Mathematics causes us accomplish better speed, bring down unpredictability and higher execution.The technique used is Gate Diffusion Input (GDI) which is a more refined way to design a circuit which less complex than circuits designed by other techniques

Vedic mathematics is based on 16 Sutras (or aphorisms) dealing with various branches oF mathematics like arithmetic, algebra, geometry etc.These sutras are meant for faster mental calculation .These Sutras along with their brief meanings are enlisted below alphabetically.These methods and ideas can be directly applied to trigonometry, plain and spherical geometry, conics, calculus (both differential and integral) and applied mathematics of various kinds.

Sri BharatiKrisnaTirtha.The implementation of vedic sutras ensure substantial reduction of propagation delay in comparison with Wallace Tree (WTM), modified Booth Algorithm (MBA), Baugh Wooley (BWM) and Row Bypassing and Parallel Architecture (RBPA) based implementation which are most commonly used architectures.

Vedic mathematics is the ancient Indian system of mathematics which mainly deals with Vedic mathematical formulae and their application to various branches of mathematics. The word „Vedic" is derived from the word „Veda" which means the store-house of all knowledge. Vedic mathematics was reconstructed from the ancient Indian scriptures (Vedas) by Sri Bharati Krishna Tirtha (1884-1960) after his eight years of research on Vedas. According to his research, Vedic mathematics is mainly based on sixteen principles or word-formulae which are termed as Sutras. The beauty of Vedic mathematics lies in the fact that it reduces otherwise cumbersome looking calculations in conventional mathematics to a very simple one. This is so because the Vedic formulae are claimed to be based on the natural principles on which the human mind works. This is a very interesting field and presents some effective algorithms which can be applied to various branches of engineering such as computing and digital signal processing.

Figure

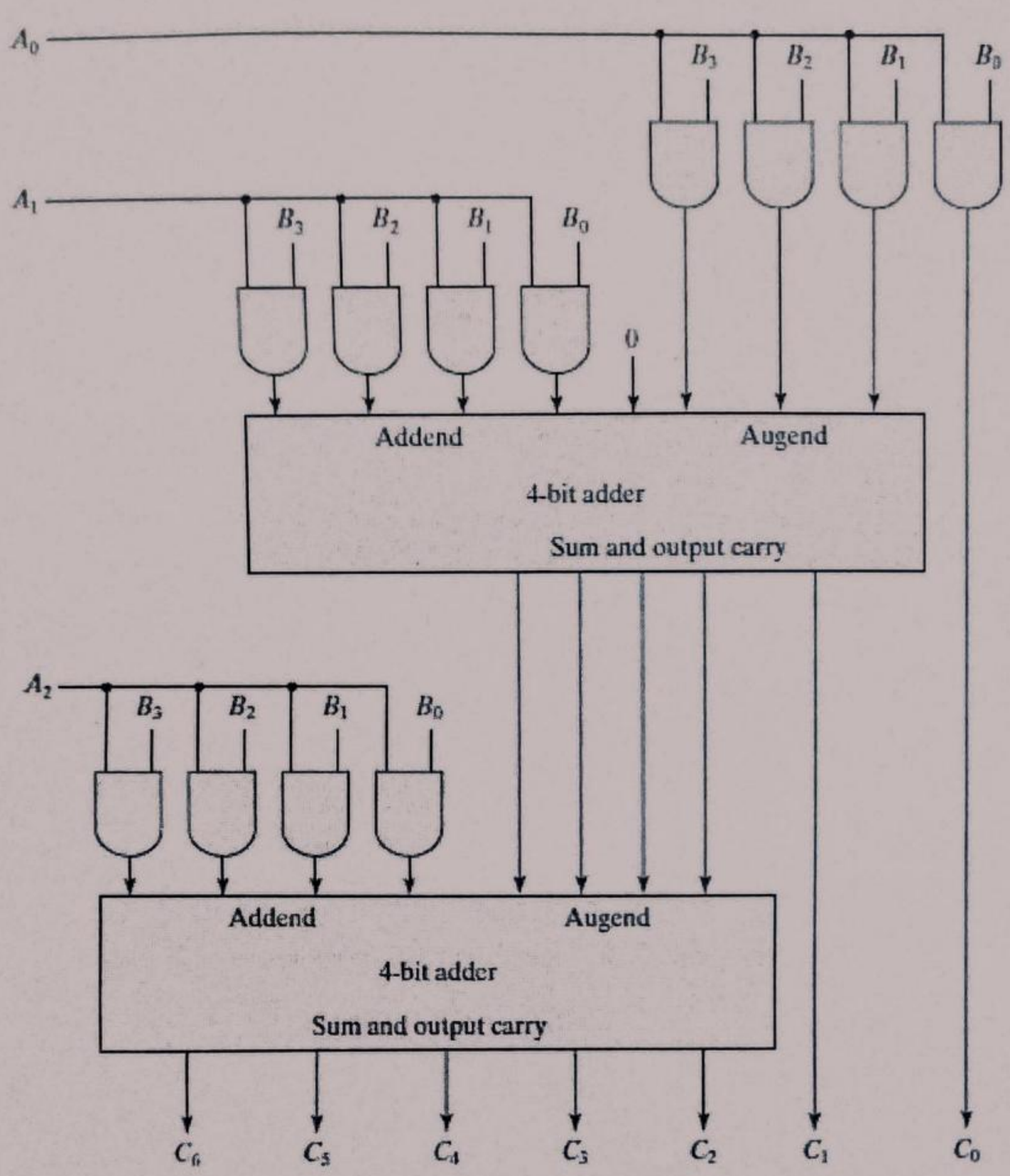

Figure 6

Proposed 4x4 Multiplier

The 4-bit vedic multiplier architecture is implemented by using logic gates, steps involved in implementation of vedic multiplier is first the design of 2-bit Vedic multiplier and second the design of 4-bit vedic multiplier for this four 2- bit vedic multiplier is used and they are interconnected by ripple carry adders. The 8-bit vedic multiplier architecture is implemented by using multiplexers, all logic gates are replaced by multiplexers.

Divide the no. of bits in the inputs equally in two parts . Let's analyze 4x4 bit multiplication, say

multiplicand A=A3A2A1A0 and multiplier B= B3B2B1B0.Following are the output line for the multiplication result, S7S6S5S4S3S2S1S0. Let's divide A and B into two parts, say "A3 A2" & "A1 A0" for A and "B3 B2" & "B1B0" for B.

Using the fundamental of Vedic multiplication, taking two bit at a time and using 2 bit multiplier block, we can have the following structure for 4x4 bit multiplication ..

The 4-bit vedic multiplier using multiplexers designed and compared with 4-bit vedic multiplier using basic gates in terms of total delay, logic delay ,route delay, total memory usage and number of logic levels .The results are tabulated in table and it shows that there is a reduction in total delay, logic levels and memory usage. Thus, the proposed design is more efficient than the Vedic multiplier using basic gates.

Consider two 4-bit binary numbers a3a2a1a0 and b3b2b1b0.

The partial products (P7P6P5P4P3P2P1P0) generated are given by the following equations:

i. P0= a0b0

ii. P1= a0b1 + a1b0

iii. P2 = a0b2 + a1b1 + a2b0+ P1

iv. P3= a0b3 + a1b2 + a2b1 + a3b0+ P2

v. P4 = a1b3 + a2b2 + a3b1 + P3

vi. P5 = a1b2 + a2b1 + P4

vii. P6 = a3b3 + P5

viii. P7 = carry of P6

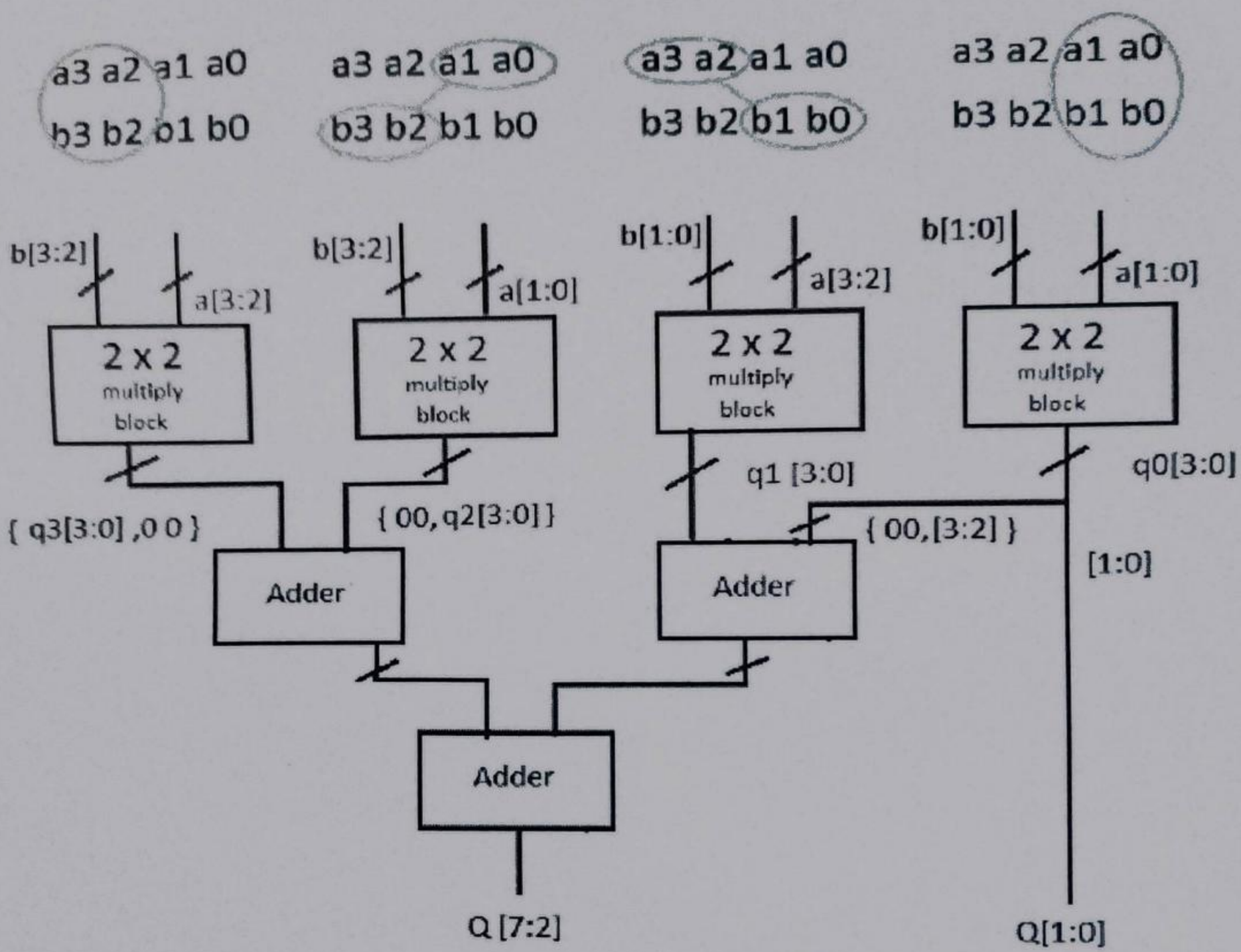

Figure 8

Proposed 8x8 Multiplier

Proposed 8-bit adder performs the function of 8-bit addition that gives two bits of sum and one carry as output. Its block diagram contains one full adder (FA) and two half adders (HF) is given in Figure 4.The existing 8-bit vedic multiplier consists of four 4-bit vedic multipliers and three 8-bit ripple carry adder. Let us take A and B are of 8-bit input which gives an output S of sixteen bit and results are obtained after getting partial product and doing addition.

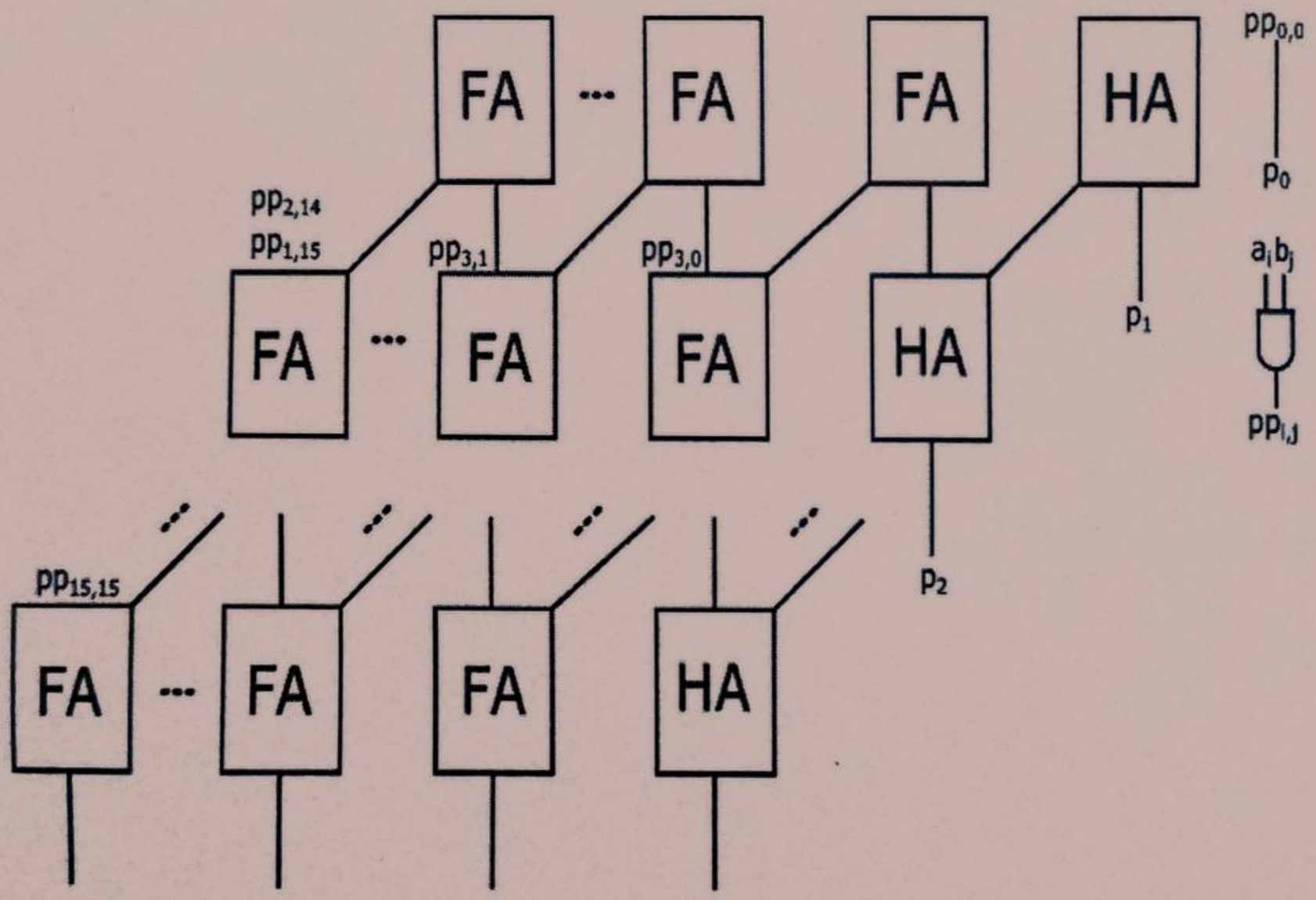

Figure 9

The 8-bit vedic multiplier architecture is implemented by using logic gates, steps involved in implementation of vedic multiplier is first the design of 2-bit Vedic multiplier and second the design of 4-bit vedic multiplier for this four 2- bit vedic multiplier is used and they are interconnected by ripple carry adders. The 8-bit vedic multiplier architecture is implemented by using multiplexers, all logic gates are replaced by multiplexers.

Take a and b are of 2-bit input which gives an output q of 4-bits here all the logic gates are replaced by multiplexers and summing of partial products is performed by half adders these adders also replaced by logic gates same procedure is used in designing of 4-bit Vedic multiplier in place of half adders ripple carry adders are used and they also replaced by multiplexers. By using two, four bit Vedic multiplier and 8-bit ripple carry adder using multiplexers an 8-bit Vedic multiplier using multiplexers. The proposed 8-bit Vedic multiplier is coded in VHDL, simulated using Xilinx ISim simulator, synthesized using Xilinx XST and verified for the inputs a=00000011 and b=00000001which gives output of s=0000000000000011.

The8-bit vedic multiplier using multiplexers designed and compared with 8-bit vedic multiplier using basic gatesin terms of total delay, logic delay ,route delay, total memory usage and number of logic levels .The results are tabulated in table 1 and it shows that there is a reduction in total delay, logic levels and memory usage. Thus, the proposed design is more efficient than the Vedic multiplier using basic gates.

Here, A, B, C, D are four inputs. S0 and S1 are LSB and MSB of Sum outputs respectively and Sum is the sum of four inputs. C0 is the carry bit. To reduce the delay, a 8x8 multiplier is implemented using half adder, full adder and the proposed 4-bit adder as shown in Figure below.

APPLICATIONS

1. The speed of multiplication operation is of great importance in DSP. Digital Signal processing is a technology that is present in almost every engineering discipline. It is also the fastest growing technology of the century and hence it posses tremendous challenges to the engineering community.

2. Faster addition and multiplication are of extreme importance in DSP for Convolution, DFT and Digital filters .The core computing process is always a multiplication routine and hence DSP engineers are constantly looking for new algorithms and hardware to implement them. The methods in Vedic Multipliers are complementary directly and easy.

3. Frequency domain filtering (FIR and IIR), frequency-time transformations (FFT), Correlation, Digital Image processing.

4. Because of high speed of Vedic multiplication ALU utilizes this algorithm to give reliable output.

5. Low power VLSI system design.

Future Work Enhancements:

Vedic Mathematics, developed about 2500 years ago, gives us a clue of symmetric computation. Vedic mathematics deals with various topics of mathematics such as basic arithmetic, geometry, trigonometry, calculus etc. All these methods are very efficient as far as manual calculations are concerned. If all those methods effectively implement hardware, it will reduce the computational speed drastically. Therefore, it could be possible to implement a complete ALU using all these methods using Vedic mathematics methods. Vedic mathematics is long been known but has not been implemented in the DSP and ADSP processors employing large number of multiplications in calculating the various transforms like FFTs and control applications such as P, PI, PID Controller implementing in FPGA etc.

CONCLUSION

This paper presents a novel way of realizing a high speed multiplier using Urdhva Tiryagbhyam sutra. A 8-bit modified multiplier is designed by using the proposed 8 bit adder. The proposed 8x8 multiplier gives a total delay of 12.825 ns which is less when compared to the total delay of existing multiplier architecture. Results also indicate a 13.19% increase in the speed when compared to normal Vedic multiplier. Our design more preferable over all other designs.

It can be concluded that Vedic Multiplier is superior in all respect like speed, delay, complexity.

However Array Multiplier requires more powerconsumption and gives optimum number of components required. Ancient Indian Vedic Mathematics gives efficient algorithms orformulae for multiplication which increase the speed of devices. Urdhva Tiryakbhyam, is general mathematical formula and equally works the best. applicable to all cases of multiplication. Also, the architecture based on this sutra is seen to be similar to the popular array multiplier where an array of adders is required to arrive at the final product.

References

1. Swami Bharati Krishna Tirthaji Maharaja, "Vedic Mathematics", MotilalBanarsidass Publishers, 1965.

2. Implementation of Multiplier using Vedic Algorithm, (IJITEE) ISSN: 2278-3075, Volume-2.

3. Rakshith T R and RakshithSaligram, "Design of High Speed Low Power Multiplier using Reversible logic: a Vedic Mathematical Approach", International Conference on Circuits, Power and Computing Technologies.

4. M.E. Paramasivam and Dr. R.S. Sabeenian, "An Efficient Bit Reduction Binary Multiplication Algorithm using Vedic Methods", IEEE 2nd International Advance Computing Conference, 2010

APPENDIX

----------VEDIC MULTIPLIER----------

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
entity twobitvedicmultiplier is
port(
a: in std_logic_vector(1 downto 0);
b: in std_logic_vector(1 downto 0);
c: out std_logic_vector(3 downto 0));
end twobitvedicmultiplier;

architecture Behavioral of twobitvedicmultiplier is
signal s1,s2,s3,s4:std_logic;
begin
c(0)<=a(0) and b(0);
s1<=a(1) and b(0);
```

----------half adder----------

```
library ieee;
use ieee.std_logic_1164.all;

entity half_adder is
 port (a, b: in std_logic;
  sum,carry_out: out std_logic);
end half_adder;

architecture dataflow of half_adder is
 begin
 sum <= a xor b;
 carry_out <= a and b;
end dataflow;
```

----------full adder----------

```
library ieee;
use ieee.std_logic_1164.all;
entity Full_Adder is
   port( X, Y, Cin : in std_logic;
         sum, Cout : out std_logic);
end Full_Adder;


architecture bhv of Full_Adder is
begin
  sum <= (X xor Y) xor Cin;
  Cout <= (X and (Y or Cin)) or (Cin and Y);
end bhv;
```

```
----------4 bit adder----------
--------------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
entity fourbitAdder is
  port( X, Y: in std_logic_vector(3 downto 0);
        sum : out std_logic_vector(3 downto 0);
                    carry : out std_logic);
end fourbitAdder;
architecture bhv of fourbitAdder is
component half_adder
  port (
       a, b: in std_logic;
       sum, carry_out: out std_logic);
end component;
component Full_Adder
  port(
            X, Y, Cin : in std_logic;
      sum, Cout : out std_logic);
end component;
signal c: std_logic_vector(2 downto 0);
begin
a1:half_adder port map(X(0),Y(0),sum(0),c(0));
a2:Full_adder port map(X(1),Y(1),c(0),sum(1),c(1));
a3:Full_adder port map(X(2),Y(2),c(1),sum(2),c(2));
a4:Full_adder port map(X(3),Y(3),c(2),sum(3),carry);
end bhv;
```

```
----------6 bit adder----------
--------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
entity sixbitAdder is
  port( X, Y: in std_logic_vector(5 downto 0);
        sum : out std_logic_vector(5 downto 0);
                    carry : out std_logic);
end sixbitAdder;
architecture bhv of sixbitAdder is
component half_adder
  port (
      a, b: in std_logic;
      sum, carry_out: out std_logic);
end component;
component Full_Adder
  port(
            X, Y, Cin : in std_logic;
       sum, Cout : out std_logic);
end component;
signal c: std_logic_vector(4 downto 0);
begin
a1:half_adder port map(X(0),Y(0),sum(0),c(0));
a2:Full_adder port map(X(1),Y(1),c(0),sum(1),c(1));
a3:Full_adder port map(X(2),Y(2),c(1),sum(2),c(2));
a4:Full_adder port map(X(3),Y(3),c(2),sum(3),c(3));
a5:Full_adder port map(X(4),Y(4),c(3),sum(4),c(4));
```

```
----------8 bit adder----------
------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
entity eightbitAdder is
  port( X, Y: in std_logic_vector(7 downto 0);
        sum : out std_logic_vector(7 downto 0);
                    carry : out std_logic);
end eightbitAdder;
architecture bhv of eightbitAdder is
component half_adder
 port (
     a, b: in std_logic;
     sum, carry_out: out std_logic);
end component;
component Full_Adder
  port(
          X, Y, Cin : in std_logic;
      sum, Cout : out std_logic);
end component;
signal c: std_logic_vector(6 downto 0);
begin

a1:half_adder port map(X(0),Y(0),sum(0),c(0));
a2:Full_adder port map(X(1),Y(1),c(0),sum(1),c(1));
a3:Full_adder port map(X(2),Y(2),c(1),sum(2),c(2));
a4:Full_adder port map(X(3),Y(3),c(2),sum(3),c(3));
```

```
a5:Full_adder port map(X(4),Y(4),c(3),sum(4),c(4));
a6:Full_adder port map(X(5),Y(5),c(4),sum(5),c(5));
a7:Full_adder port map(X(6),Y(6),c(5),sum(6),c(6));
a8:Full_adder port map(X(7),Y(7),c(6),sum(7),carry);
end bhv;
```

```
----------12 bit adder----------
------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
entity twelvebitAdder is
  port( X, Y: in std_logic_vector(11 downto 0);
       sum : out std_logic_vector(11 downto 0);
                        carry : out std_logic);
end twelvebitAdder;
architecture bhv of twelvebitAdder is
component half_adder
 port (
      a, b: in std_logic;
      sum, carry_out: out std_logic);
end component;
component Full_Adder
  port(
            X, Y, Cin : in std_logic;
       sum, Cout : out std_logic);
 end component;
signal c: std_logic_vector(10 downto 0);
 begin

a1:half_adder port map(X(0),Y(0),sum(0),c(0));
a2:Full_adder port map(X(1),Y(1),c(0),sum(1),c(1));
a3:Full_adder port map(X(2),Y(2),c(1),sum(2),c(2));
a4:Full_adder port map(X(3),Y(3),c(2),sum(3),c(3));
```

```
a5:Full_adder port map(X(4),Y(4),c(3),sum(4),c(4));
a6:Full_adder port map(X(5),Y(5),c(4),sum(5),c(5));
a7:Full_adder port map(X(6),Y(6),c(5),sum(6),c(6));
a8:Full_adder port map(X(7),Y(7),c(6),sum(7),c(7));
a9:Full_adder port map(X(8),Y(8),c(7),sum(8),c(8));
a10:Full_adder port map(X(9),Y(9),c(8),sum(9),c(9));
a11:Full_adder port map(X(10),Y(10),c(9),sum(10),c(10));
a12:Full_adder port map(X(11),Y(11),c(10),sum(11),carry);
end bhv;
```

```
----------8 bit multiplier----------

library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity Vedic_Multiplier_8bit is
    Port ( a : in  STD_LOGIC_VECTOR (7 downto 0);
           b : in  STD_LOGIC_VECTOR (7 downto 0);
           c : out  STD_LOGIC_VECTOR (15 downto 0));
end Vedic_Multiplier_8bit;

architecture Behavioral of Vedic_Multiplier_8bit is

signal s: STD_LOGIC_VECTOR(7 downto 0);signal s1: STD_LOGIC_VECTOR(7 downto 0);
signal s2: STD_LOGIC_VECTOR(7 downto 0);signal s3: STD_LOGIC_VECTOR(7 downto 0);
signal s4: STD_LOGIC_VECTOR(7 downto 0);signal s5: STD_LOGIC_VECTOR(8 downto 0);
signal s6: STD_LOGIC_VECTOR(11 downto 0);signal s7: STD_LOGIC_VECTOR(11 downto
0);
```

```
signal s8: STD_LOGIC_VECTOR(11 downto 0);signal s9: STD_LOGIC_VECTOR(11 downto
0);

component fourbitvedicmultiplier

port(

a: in std_logic_vector(3 downto 0);

b: in std_logic_vector(3 downto 0);

c: out std_logic_vector(7 downto 0));

end component;

component eightbitAdder

  port( X, Y: in std_logic_vector(7 downto 0);

       sum : out std_logic_vector(7 downto 0);

                          carry : out std_logic);

end component;

component twelvebitAdder

  port( X, Y: in std_logic_vector(11 downto 0);

       sum : out std_logic_vector(11 downto 0);

                          carry : out std_logic);

end component;

begin


u1:fourbitvedicmultiplier port map( a(3 downto 0),b(3 downto 0),s);

u2:fourbitvedicmultiplier port map( a(7 downto 4),b(3 downto 0),s1);

u3:fourbitvedicmultiplier port map( a(3 downto 0),b(7 downto 4),s2);

u4:fourbitvedicmultiplier port map( a(7 downto 4),b(7 downto 4),s3);

c(3 downto 0)<= s(3 downto 0);

s4<=("0000"&s(7 downto 4));

u5:eightbitAdder port map(s1,s4,s5(7 downto 0),s5(8));

s6<=("0000"&s2(7 downto 0));
```

```
s7<=(s3(7 downto 0)&"0000");
u6:twelvebitAdder port map(s6(11 downto 0),s7,s8(11 downto 0));
s9<=("0000"&s5(7 downto 0));
u7:twelvebitAdder port map(s9,s8,c(15 downto 4));
end Behavioral;
```