VLSI Implementation of Matrix-Diagonal Method of Binary Multiplication

Prashant Nair, Darshan Paranji and S. S. Rathod

***Abstract*—In this paper, an algorithm of ancient Indian Vedic mathematics, has been implemented on a new multiplier for low power, high speed applications which is a space and time space efficient method. The algorithm 'Urdhava Tiryakbhyam' is modified to generate concurrent carry for the next stage. It is based on the generation and addition of concurrent partial sums produced within the proposed Matrix representation architecture. The algorithm is useful for math coprocessors in the field of computers. Algorithm is implemented on SPARTAN-II FPGA (Field Programmable Gate Array). The speed improvements gained due to the proposed algorithm and the concurrency characteristic of the proposed Matrix representation architecture enable large saving of resources in FPGA applications.**

***Index Terms*—Booth Algorithm, Array Multiplier, VHDL**

I. Introduction

BINARY arithmetic multiplication has been a very important issue for almost all the microprocessor and microcontroller manufacturers. Multiplication is most often an operation which consumes the most of the processor time. Although multiplication techniques such as 'Booth Algorithm Multiplier' have been effective over conventional 'Array Multiplication' technique, their disadvantage of time consumption has not been completely removed. The Matrix-Diagonal method of binary multiplication overcomes these by reducing processing time with effective use of concurrency in computation. The space utilization is also optimized by this method. Matrix-Diagonal method of binary multiplication uses matrix arrangement of representation of two binary numbers. The two numbers are then concurrently viewed bitwise in an order suggested in Vedic mathematics. Logical AND operation output at every node is then summed up diagonally and its carry transferred to the next stage. Half adders are constantly used instead of full adders in an attempt to optimize space constraint without any effect on performance.

Prashant Nair is with Department of Electronics Engineering at Sardar Patel Institute of Technology, Mumbai 400058 INDIA. (e-mail:parthos17@gmail.com).

Darshan Paranji is with Department of Electronics and Telecomm. Engineering at Sardar Patel Institute of Technology, Mumbai 400058 INDIA. (e-mail:daaru1947@yahoo.com).

S. S. Rathod is with Department of Electronics Engineering at Sardar Patel Institute of Technology, Mumbai 400058 INDIA. (e-mail: rathod_spce@yahoo.com).

II. Various Multiplication Algorithms

There are various algorithms suggested for efficient multiplication of binary numbers [1] [2] [3] [4]. The most popular and efficient algorithms available are the Array multiplication and Booth Multiplication Algorithm.

A. Array Multiplication Algorithm

This multiplication algorithm [1] is analogous to the one we do in day-to-day life. This algorithm for multiplication of two four bit numbers is explained as follows.

For example

```
        1 0 1 0    {10}
      x 1 1 0 1    {13}
      ---------
        1 0 1 0
      0 0 0 0
    1 0 1 0
  1 0 1 0
  -----------
= 1 0 0 0 0 0 1 0    {130}
```

An array multiplier algorithm:

1. Start
2. A: A3 A2 A1 A0
   B: B3 B2 B1 B0
   X: X3 X2 X1 X0 (each 4-bit registers)
   Y: Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0
3. for i: 0 to 3
   if Bi = 1
   Xi = A
   Shift left Xi 'i' times
4. for i: 0 to 3
   Y = Y + Xi
5. Result is stored in Y
6. Stop

This algorithm uses the Adder numerous times and has the disadvantage of using a 2n bit adder for a n bit multiplication. Thus time taken for the execution of algorithm will be more.

B. Booth Multiplication Algorithm

Booth's multiplication algorithm [5] is extensively used in many computing machines. This algorithm was invented by Andrew Booth in 1951. This algorithm is particularly useful for machines that can shift bits faster than adding them.

If *x* is the count of bits of the multiplicand, and *y* is the count of bits of the multiplier:

- Draw a grid of three rows, each with columns for *x* + *y* + 1 bits. Label the lines respectively A (add), S (subtract), and P (product).
- In two's complement notation, fill the first *x* bits of each line with :
  - A: the multiplicand
  - S: the negative of the multiplicand (in 2's complement format)
  - P: zeroes
- Fill the next *y* bits of each line with :
  - A: zeroes
  - S: zeroes
  - P: the multiplier
- Fill the last bit of each line with a zero.
- Do both of these steps *y* times :
  1. If the last two bits in the product are...
     - 00 or 11: do nothing.
     - 01: P = P + A. Ignore any overflow.
     - 10: P = P + S. Ignore any overflow.
  2. Arithmetically shift the product right one position.
- Drop the first (we count from right to left when dealing with bits) bit from the product for the final result.

Booth Multiplier has been used popularly in microprocessor Arithmetic and Logical Units for binary multiplication. This algorithm is also noted as one of the highly efficient algorithms.

C. Proposed Matrix Diagonal Algorithm

The proposed matrix diagonal multiplication uses the algorithm of vedic mathematic multiplication using Urdhava Tiryakbhyam [2] [3] [4]. The algorithm is based on the creation of partial products is done on the concurrent addition of these partial products. The parallel concurrent action is due to the use of 'Urdhava Tiryakbhyam'. The action for a 4 x 4 bit multiplication is explain in Fig. 1.

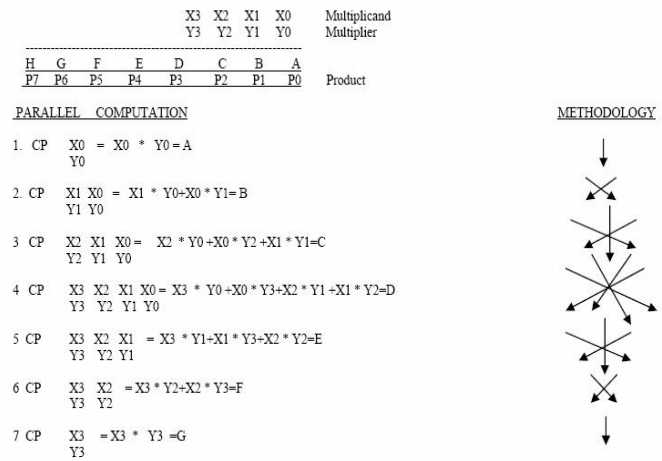

Fig. 1. Multiplication using Urdhava Tiryakbhyam technique

This algorithm can be generalized for an n x n bit multiplication. The multiplier will work independent of clock frequency. Although the advantage of higher clock frequency is increased processing power, it has several disadvantages like higher power dissipation.

The proposed Matrix-Diagonal algorithm is an extension of this Vedic technique. It uses a novel form of data representation to optimize results. This form of representation is for a 4 x 4 multiplier is shown in the Fig. 2.

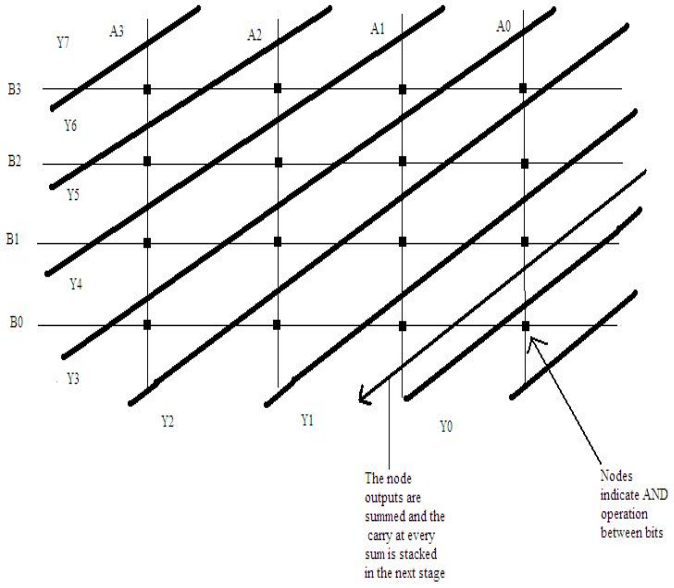

Fig. 2. Multiplication using Matrix-Diagonal Algorithm

The carry generated at every summing node is stacked in the next stage, which will be added along with the other nodes in the next stage. This process also saves ALU space and is thus a truly efficient method. Thus, we can see that this structure when implemented for n x n multiplier will prove to be a time efficient and a space efficient solution. The multiplier has the advantage as the number of bits increases the gate delay and area increases very slowly compared to other multipliers. Thus this architecture is efficient in terms of silicon area/speed.

III. VLSI IMPLEMENTATION AND RESULTS

The algorithms discussed in section II are implemented using Active-HDL [6] VHDL [8] simulator. The synthesis and implementation is done by using Xilinx ISE 8.0 [7]. The algorithms are implemented on Xilinx FPGA SPARTAN-2 device 2s30pq208-5.

The array multiplier was implemented with regular shift and adds partial product logic. The VHDL simulation waveforms generated are as shown in the Fig. 3 and found functioning normally. The technology schematic generated using synthesizer is shown in Fig. 4 and the placement-routing results generated are shown in Fig. 5. The Fig. 5 indicates the amount of resources utilized from the selected FPGA when array multiplier is implemented.

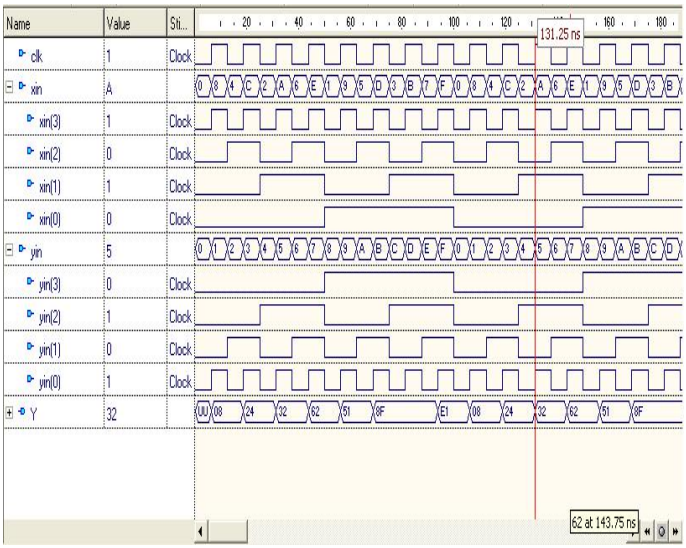

Fig. 3. VHDL Simulation Results of Array Multiplier

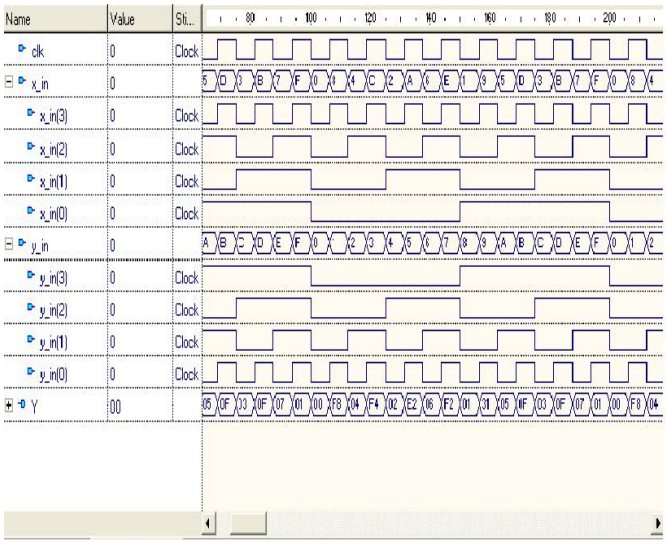

Fig. 6. VHDL Simulation Results of Booth Multiplier

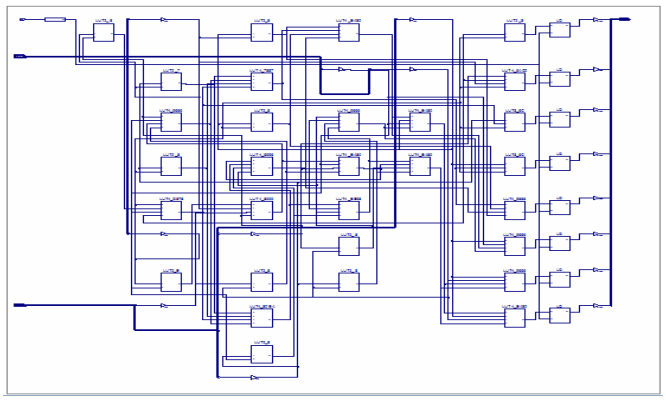

Fig. 4. Xilinx Synthesis Results of Array Multiplier

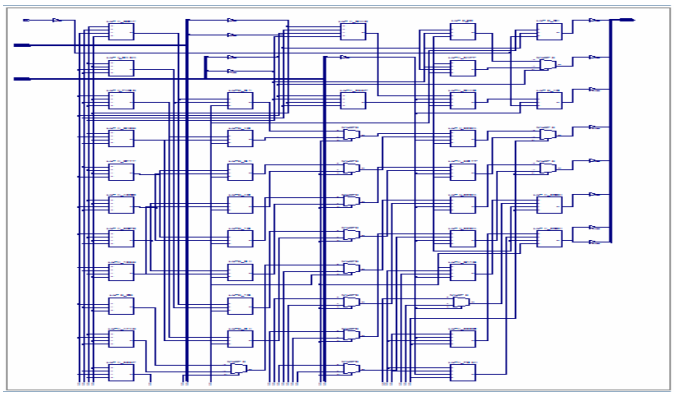

Fig. 7. Xilinx Synthesis Results of Booth Multiplier

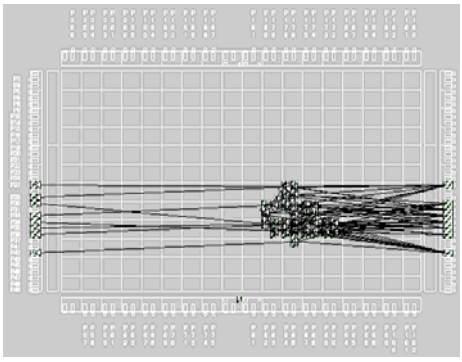

Fig. 5. Placement and Routing Results of Array Multiplier

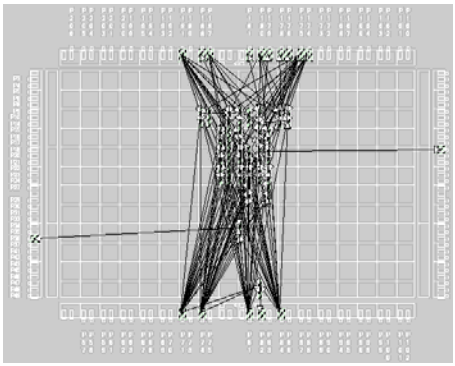

Fig. 8. Placement and Routing Results of Booth Multiplier

The more efficient Booth algorithm multiplier VHDL simulation waveforms are as shown in the Fig. 6 and found to be correct. The multiplier is designed for two 4 bit numbers. The results can be extended for n bit numbers as well. The disadvantage of this multiplier is that it requires at 4 states of the internal clock perform multiplication. Fig.7 shows technological schematic and Fig. 8 is the generated result of placement and routing for the selected FPGA when Booth Multiplier is implemented. One can easily see from the placement and routing result in the figure that, in spite of its efficiency, this multiplier consumes space and hardware.

The proposed Matrix-Diagonal multiplier was implemented with concurrency in VHDL and its simulation waveforms are as shown in the Fig. 9. From simulation results we can easily observe that concurrency is one of the prime features for the increased efficiency of the matrix multiplier. Fig.10 shows technological schematic and Fig. 11 is the generated result of placement and routing for the selected FPGA when Proposed Matrix-Diagonal Multiplier is implemented. It is clear from Fig. 5, Fig. 8 and Fig. 11 that the less resources are utilized from FPGA as compared to Array multiplier and Booth multiplier.

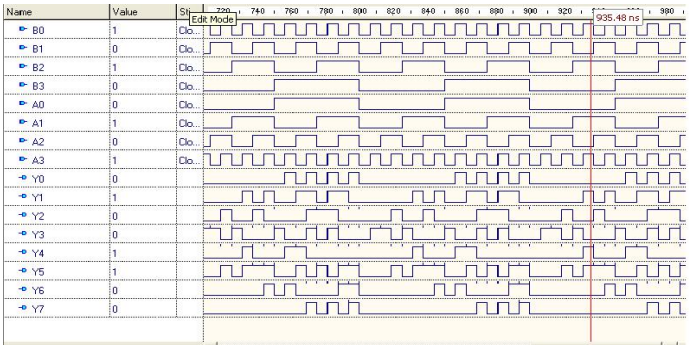

Fig. 9. VHDL Simulation Results of Matrix-Diagonal Multiplier

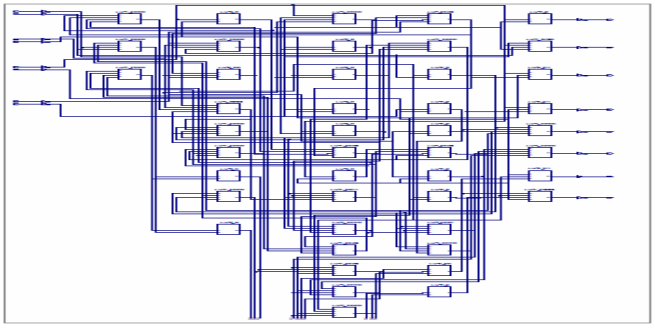

Fig. 10. Xilinx Synthesis Results of Matrix-Diagonal Multiplier

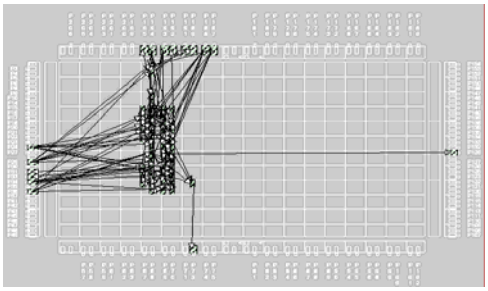

Fig. 11. Placement and Routing Results of Matrix-Diagonal Multiplier

TABLE I
COMPARISION OF SYNTHESIS RESULTS

Algorithm	Array	Booth	Matrix-Diagonal
Delay	72.564ns	72.452ns	34.950ns
Device Utilization			
Slices:	17 out of 432 (3%)	31 out of 432(7%)	27 out of 432 (6%)
4 input LUTs	30 out of 864 (3%)	57 out of 864 (6%)	47 out of 864 (5%)
Bonded IOBs	17 out of 136 (12%)	17 out of 136 (12%)	16 out of 136 (11%)

IV. Conclusions

Table I indicates that the delay in multiplication is almost halved in the proposed Matrix-Diagonal multiplication implementation; this is primarily due to concurrency. Further, the proposed Matrix-Diagonal multiplication is space efficient too as compared to Booth multiplier implementation. Thus Matrix-Diagonal Multiplier is faster than Array and Booth Multiplier. Due to factors such as timing efficiency, speed and lesser area, Matrix-Diagonal Multiplier can be implemented in Arithmetic and Logical Units replacing traditional multipliers. The speed improvements are gained due to concurrency and such a design must enable large saving of resources when used in FPGA for digital signal processing and image processing.

V. Acknowledgment

The authors gratefully acknowledge the facilities provided in VLSI Laboratory of Electronics Engineering Department, Sardar Patel Institute of Technology, Mumbai.

VI. References

[1] Kevin Biswas, "Multiplexer Based Array Multipliers," A Ph.D. Dissertation, University of Windsor, Electrical and Computer Engineering, Apr. 2005.
[2] Himanshu Thapliyal and Hamid R. Arabnia, "A time area power efficient multiplier and square architecture based on ancient Indian Vedic mathematics," www.vedicmathsindia.org.
[3] Vishal Verma and Himanshu Thapliyal , “High Speed Efficient N X N Bit Multiplier Based On Ancient Indian Vedic Mathematics”, Proceedings International Conference On VLSI, Las Vegas, June 2003.
[4] A.P. Nicholas, K.R Williams, J. Pickles ,“Application of Urdhava Sutra”, Spiritual Study Group, Roorkee (India),1984
[5] Stephen Brown and Zvonko Vranesic, "Fundamentals of digital logic with VHDL design," 1st Ed. New York: McGraw-Hill, 2007
[6] “Active-HDL 7.2User Manual”, Active HDL Inc, USA, 2007.
[7] “Xilinx ISE User Manual”, Xilinx Inc, USA, 2007
[8] “VHDL Reference Manual”, IEEE standard, 1993

VII. Biographies

Prashant Nair (M’2007) was born in Mumbai in India on 1987. He is graduate student of Electronics Engineering Department of Sardar Patel Institute of Technology, Mumbai. He won many awards in various competitions. He is also a member of IEEE and Head of Activities of Electronics Students Association. His area of interest includes microelectronics, robotics and VLSI design and embedded systems.

Darshan Paranji S (M’2007) was born in Chennai in India on 1988. He is graduate student of Electronics and Telecommunication Engineering Department of Sardar Patel Institute of Technology, Mumbai. He won many awards in various competitions. He is also a member of IEEE. His area of interest includes VLSI, wireless communication and networking.

S. S. Rathod (M’2007) was born in Amravati in India on Feb 28, 1975. He graduated from the College of Engineering Badnera, and completed his post graduation in the field of Electronics Engineering from V.J.T.I., Mumbai-India. Currently he is pursuing his doctoral research at Indian Institute of Technology, Roorkee-India.

His employment experience includes eight years as an educationalist. He has published more that 20 papers in various national and international conferences. His special fields of interest include VLSI Design, process and device modeling. His name is listed in the science category of Marques Who’s Who USA. He received outstanding achieving ward by the Energy Society, India. He is member of IEEE, ISTE and ISNT.