Design and FPGA Implementation of High Speed Vedic Multiplier

Sudeep.M.C
UG Scholar, Dept. of ECE
Christ University, Bangalore

Sharath Bimba.M
Asst. Professor, Dept. of ECE
Christ University, Bangalore

Mahendra Vucha
Asst. Professor, Dept. of ECE
Christ University, Bangalore

ABSTRACT

Multiplication is an operation much needed in Digital Signal Processing for various applications. This paper puts 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.

General Terms

Urdhva Tiryagbhyam Algorithm, Kogge Stone Algorithm, Vedic Multiplier.

Keywords

VM-Vedic Multiplier, KSA-Kogge Stone Adder, RCA-Ripple Carry adder, CLA-Carry look-ahead Adder, CSA-Carry Save Adder.

1. INTRODUCTION

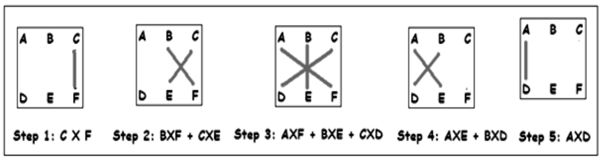
Vedic Mathematics is an ancient system of math practiced during Vedic age which was reconstructed by Jagadguru Swami Sri Bharati Krishna Tirthaji Maharaja[1] between 1911 and 1918 from certain Sanskrit manuscripts. It is perhaps the most refined and efficient mathematical system possible. One of such efficient technique has been employed to enhance the design of a multiplier. Multipliers are the key blocks of a Digital Signal processor. Multiplication is the key aspect, whereby improvement in computational speed of multiplication decreases the processing time of Digital Signal Processors. Convolution, Fast Fourier transforms and various other transforms make use of multiplier blocks.

A faster method for multiplication based on ancient Indian Vedic mathematics is studied in this paper. Among various methods of multiplications in Vedic mathematics, Urdhva Tiryagbhyam is efficient [2]. Urdhva Tiryagbhyam is a general multiplication formula applicable to all cases of multiplication. For addition of partial products in the multiplier Kogge Stone algorithm is used and realized. The code is written in Verilog HDL[3] and synthesized using Xilinx ISE 9.1i and implemented on Spartan 3 and 6 FPGA devices.

2. LITERATURE REVIEW

The algorithms and multiplier architecture was studied from [4, 5, 6, 7, 8, 9, 10, 11] and are represented below.

2.1 Urdhva Tiryagbhyam



Consider ABC as multiplicand and DEF as the multiplier. The steps of multiplication are descriptive in the figure above and the examples are solved below for better understanding. The intermediate carry generated is appended to the very next bit.

2.1.1 Illustration:

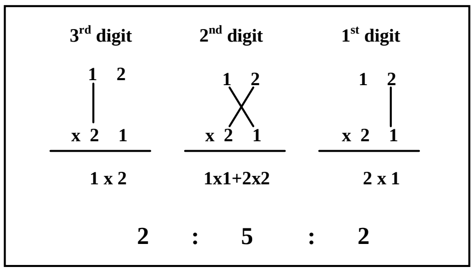


Figure 1: Illustration of decimal multiplication using Vedic technique

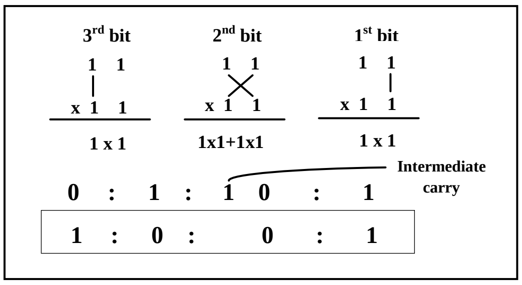


Figure 2: Illustration of binary multiplication using Vedic technique

2.1.2 How is it better than the conventional method?

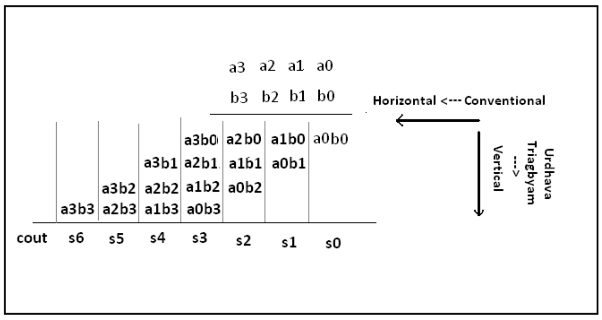


Figure 3: Difference between Conventional Multiplication and Vedic technique

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.

2.2 Kogge Stone Algorithm

Kogge Stone algorithm was developed by Peter M. Kogge and Harold S. Stone and published in an IEEE seminar in 1973[6]. It generates carry in O (log n) time and is used in the industry for high performance arithmetic circuits considering it to be the fastest adder. Carries are computed faster using KSA [9, 10, 11] at the cost of increased area.

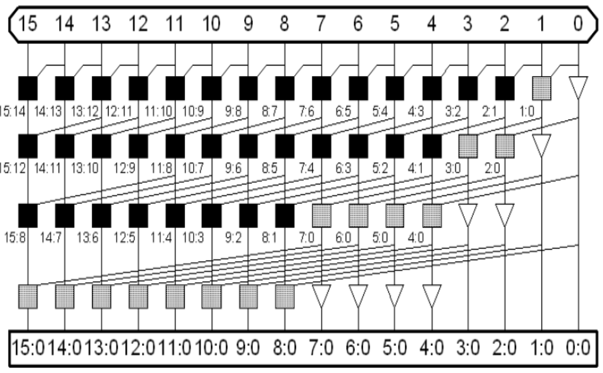


Figure 4: 16-bit Kogge stone adder network [8]

This is an attempt to apprehend the functioning of KSA in three distinct steps:

1. **Pre processing**

This step involves computation of generate and propagate signals corresponding to each pair of bits in A and B. These signals are given by the logic equations below:

$$P(i) = A(i) \oplus B(i)$$

$$G(i) = A(i) \cdot B(i)$$

2. **Carry look ahead network**

This is the block responsible for advancement in speed. This step involves computation of carries corresponding to each bit. It uses group propagate and generate as intermediate signals which are given by the logic equations below:

$$Pi{:}j = P(i{:}k+1) \cdot P(k{:}j)$$

$$Gi{:}j = G(i{:}k+1) + (P(i{:}k+1) \cdot G(k{:}j))$$

In the Figure 4, Black box represents the computation of both *Pi:j* and *Gi:j* whereas grey box represents the computation of Gi:j alone and White triangular objects are buffers.

3. **Post processing**

It involves computation of sum bits. Sum bits are computed by the logic given below:

$$S(i) = P(i) \oplus C(i-1)$$

3. PROPOSED MULTIPLIER

Proposed multiplier architecture of 2x2, 4x4, and 8x8 bit VM module are displayed below. The basic architecture was comprehended from the base paper [4] and modified to obtain the right output as well as gain speed. The major change adopted here in the architecture is that we have used Kogge stone algorithm to add partial products rather than RCA, CLA and CSA.

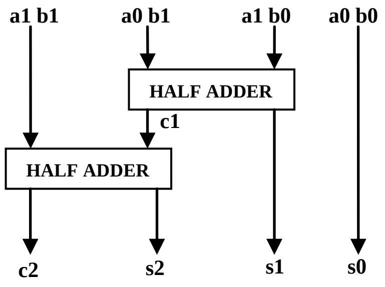


Figure 5: 2x2 Vedic Multiplier

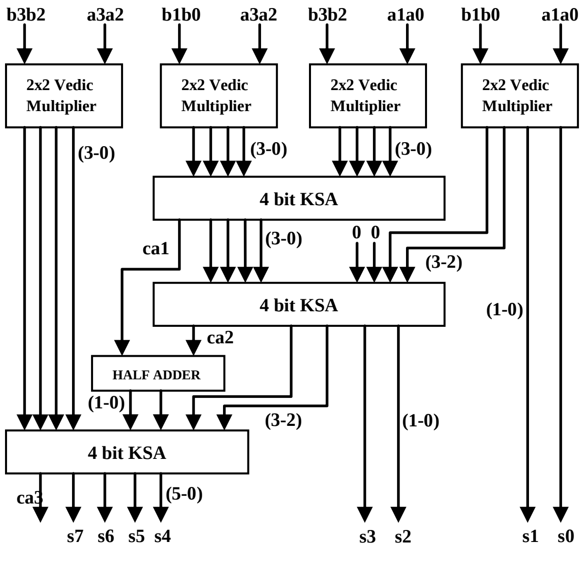


Figure 6: 4x4 Vedic Multiplier using KSA

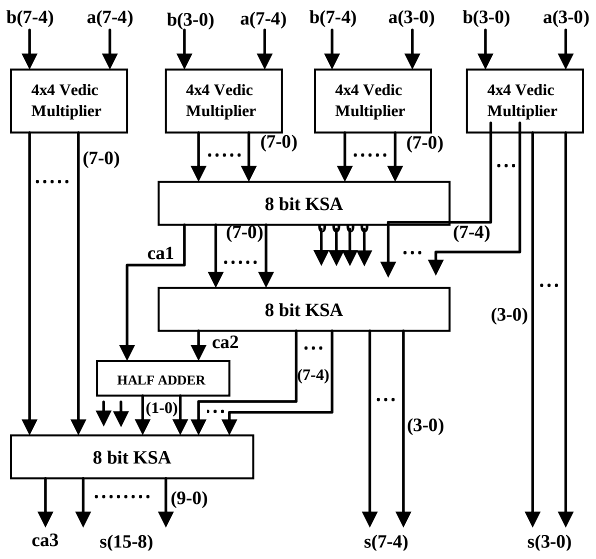
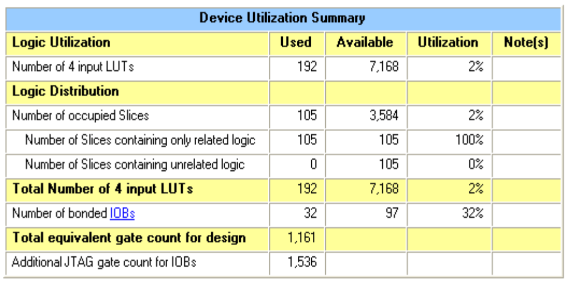


Figure 7: 8x8 Vedic Multiplier using KSA

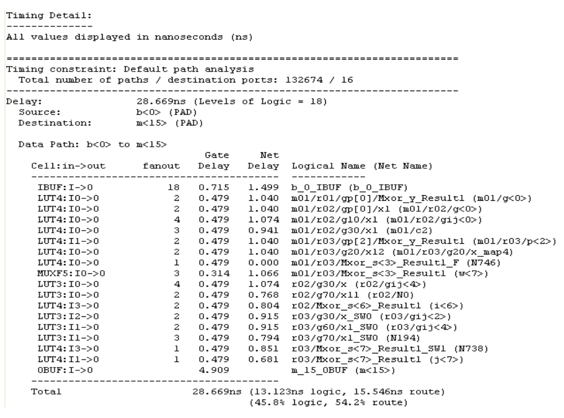
4. RESULTS AND SIMULATIONS

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 below. 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 [4] are shown in Table.1.



Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of 4 input LUTs	192	7,168	2%	
Logic Distribution				
Number of occupied Slices	105	3,584	2%	
Number of Slices containing only related logic	105	105	100%	
Number of Slices containing unrelated logic	0	105	0%	
Total Number of 4 input LUTs	192	7,168	2%	
Number of bonded IOBs	32	97	32%	
Total equivalent gate count for design	1,161			
Additional JTAG gate count for IOBs	1,536			

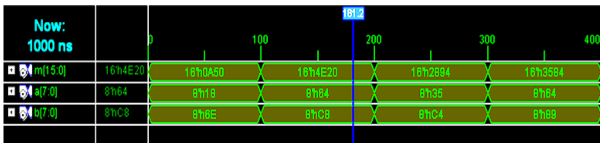
Figure 8: Device Utilization Summary

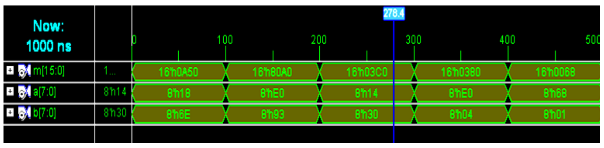


```
Timing Detail:
--------------
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default path analysis
  Total number of paths / destination ports: 132674 / 16
-------------------------------------------------------------------------
Delay:                 28.669ns (Levels of Logic = 18)
  Source:              b<0> (PAD)
  Destination:         m<15> (PAD)

  Data Path: b<0> to m<15>
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O            18   0.715   1.499  b_0_IBUF (b_0_IBUF)
     LUT4:I0->O            2   0.479   1.040  m01/r01/gp[0]/Mxor_y_Result1 (m01/g<0>)
     LUT4:I0->O            2   0.479   1.040  m01/r02/gp[0]/x1 (m01/r02/g<0>)
     LUT4:I0->O            4   0.479   1.074  m01/r02/g10/x1 (m01/r02/gij<0>)
     LUT4:I0->O            3   0.479   0.941  m01/r02/g30/x1 (m01/c2)
     LUT4:I1->O            2   0.479   1.040  m01/r03/gp[2]/Mxor_y_Result1 (m01/r03/p<2>)
     LUT4:I0->O            2   0.479   1.040  m01/r03/g20/x12 (m01/r03/g20/x_map4)
     LUT4:I0->O            1   0.479   0.000  m01/r03/Mxor_s<3>_Result1_F (N746)
     MUXF5:I0->O           3   0.314   1.066  m01/r03/Mxor_s<3>_Result1 (w<7>)
     LUT3:I0->O            4   0.479   1.074  r02/g30/x (r02/gij<4>)
     LUT3:I0->O            2   0.479   0.768  r02/g70/x11 (r02/N0)
     LUT4:I3->O            2   0.479   0.804  r02/Mxor_s<6>_Result1 (i<6>)
     LUT3:I2->O            2   0.479   0.915  r03/g30/x_SW0 (r03/gij<2>)
     LUT3:I1->O            2   0.479   0.915  r03/g60/x1_SW0 (r03/gij<4>)
     LUT3:I1->O            3   0.479   0.794  r03/g70/x1_SW0 (N194)
     LUT4:I3->O            1   0.479   0.851  r03/Mxor_s<7>_Result1_SW1 (N738)
     LUT4:I1->O            1   0.479   0.681  r03/Mxor_s<7>_Result1 (j<7>)
     OBUF:I->O                 4.909          m_15_OBUF (m<15>)
    ----------------------------------------
    Total                     28.669ns (13.123ns logic, 15.546ns route)
                                       (45.8% logic, 54.2% route)
```

Figure 9: Timing Details

4.1 Outputs of Simulation:

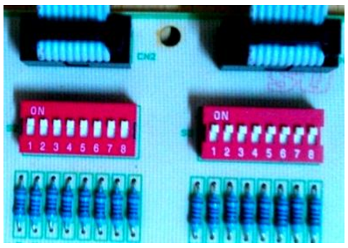




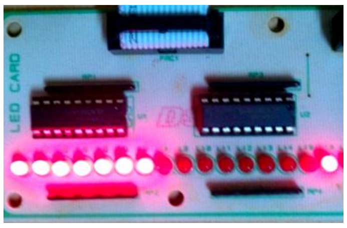
4.2 Outputs of Implementation:

1. $\frac{(11111111)2*(11111111)2}{=(1111111000000001)2}$

Inputs:

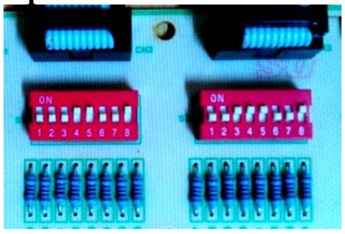


Output:

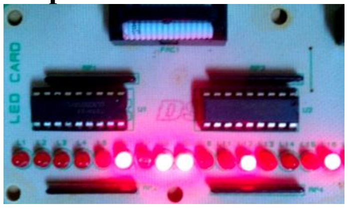


2. $\frac{(00111001)2*(00011001)2}{=(0000010110010001)2}$

Inputs:



Output:



4.3 Comparison

Table 1. Comparison of delay produced by 8x8 VM using RCA and KSA adders.

DEVICE:	**Xc3s700-4fg484**
PROGRAM	**DELAY(nS)**
VM8x8RCA[4]	[a]. 28.27
VM8x8RCA	[b]. 27.586
VM8x8KSA	[c]. 26.178
[a]-[b]	0.684
[a]-[c]	2.092
[b]-[c]	1.408

Table 2. Comparison of delay produced by 8x8 VM on SPARTAN 3 and SPARTAN 6 device.

DEVICE	**PROGRAM**	**DELAY(nS)**
xc3s400-5tq144	8X8 Vedic Multiplier	28.699
xc6slx75t-3fgg676	8x8 Vedic Multiplier	15.752

Table 3. Comparison of delay produced by various 8x8 multipliers.

DEVICE	**xc3s50a-5tq144**
PROGRAM	**DELAY(nS)**
Array Multiplier[5]	32.01
Booth Multiplier [5]	29.549
VM8x8KSA	23.644

5. CONCLUSION

By comparing tables 1, 2 and 3 we conclude that the proposed technique of multiplication using UrdhvaTiragbyam 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[9]. 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.

6. AUTHORS

Sudeep.M.C pursuing his B.Tech degree at Christ Faculty of Engineering, Christ University, Bangalore, Karnataka, India. His areas of interest are VLSI Technologies and Embedded systems.

Sharath Bimba.M received her B.Tech degree in Electrical and Electronics Engineering from Sri Venkateshwara University, Tirupati, Andrhapradesh, India in 2008 and M.Tech degree in VLSI Systems in 2010 from National Institute of Technology, Trichy, Tamilnadu, India.

She is currently working as Asst. Professor in the dept. of ECE, Christ University, Bangalore, India. Her areas of interest are Microelectronics, VLSI systems and Nanotechnology.

Mahendra Vucha received his B.Tech degree in Electronics and Communication engineering from JNTU, Hyderabad in 2007 and M.Tech degree in VLSI and Embedded System Design from MANIT, Bhopal in 2009. He is currently working for his PhD degree at MANIT and also working as Asst.Professor in the Dept. of ECE, Christ University, Bangalore, India. His areas of interest are Hardware Software Co-Design, Analog Circuit design, Digital System Design and Embedded System Design.

7. REFERENCES

[1] Jagadguru Swami, Sri Bharati Krishna, Tirthaji Maharaja, “Vedic Mathematics or Sixteen Simple Mathematical Formulae from the Veda, Delhi (1965)”, Motilal Banarsidass, Varanasi, India.

[2] Ramchandran.S and Kruthi.S.Pande. May-June 2012. “Design, Implementation and Performance Analysis of an Integrated Vedic Multiplier Architecture.” Amrita college of engineering, Bangalore.

[3] Samir Palnitkar. “Verilog HDL, A guide to Digital Design and Synthesis.” 2nd edition. Dorling Kindersley and Pearson Education, Inc. 2008.

[4] Poornima M., Shivaraj Kumar Patil, Shivukumar, Shridhar K P and Sanjay H. May, 2013. “Implementation of Multiplier using Vedic Algorithm”. MVJCE, Bangalore.

[5] PushpalataVerma and K.K.Mehta. June 2012. “Implementation of an efficient Multiplier based on Vedic Mathematics using EDA tool.”

[6] Peter M. Kogge and Harold S. Stone. August 1973. “A Parallel algorithm for the efficient solution of a general class of recurrence equations.”

[7] Pavan Kumar U.C.S, Saiprasad Goud A and A.Radhika. March 2013. “FPGA Implementation of High Speed 8-bit Vedic Multiplier Using Barrel Shifter.”

[8] Neil H.E. Weste and Kamran Eshraghian. “Principles of CMOS VLSI Design.” 3rd edition. Addision-Wesley Reading, MA.

[9] Pakkiraiah Chakali, Madhu Kumar Patnala. February 2013. “Design of High Speed Kogge-Stone Based Carry Select Adder.”

[10] P.Annapurna Bai, M.Vijaya Laxmi. August 2013. “Design of 128-bit Kogge-Stone Low Power Parallel Prefix VLSI Adder for High Speed Arithmetic Circuits.”

[11] Abhijith Kini G. September 2011. “Asynchronous Hybrid Kogge-Stone Structure Carry Select Adder Based IEEE-754 Double-Precision Floating-Point Adder.” Department of Electronics and Communication Engineering, National Institute of Technology Karnataka, NITK-Surathkal. Surathkal, Karnataka 575025, India.