VOL. 10, NO. 5, MARCH 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences


www.arpnjournals.com

IMPLEMENTATION OF SIGNED VEDIC MULTIPLIER TARGETED AT FPGA ARCHITECTURES

Paldurai K. and K. Hariharan
Department of Electronics and Communication Engineering, TCE, Madurai, Tamilnadu, India
E-Mail: paultce15@gmail.com

ABSTRACT

Signed Multiplications are very expensive and used in many of the Digital Signal Processing (DSP) applications such as Multiple-Accumulate unit and Fast Fourier Transforms (FFT). The performance of DSP computational blocks are often dominated by the speed at which a multiplication operation can be executed. Therefore a high speed signed multiplier is highly desirable to achieve high speed DSps. This paper proposes design and implementation of a novel high speed signed multiplier based on Vedic mathematics. The proposed architecture has the advantages of reduced delay and less area over conventional Booth radix-2 multiplier. It uses unsigned multiplier based on Urdhva Tiryakbhyam and 2s complement circuit. The proposed signed multiplier and conventional booth multiplier are coded in Verilog, synthesized and simulated using ISE simulator. It is implemented on the iwavesystems Unified Learning Kit Spartan6 family xc6slx25t-2fgg484 FPGA. The Area and maximum combinational path delay of proposed Signed Multiplier and conventional Booth Multiplier are compared.

Keywords: urdhva tiryakbhyam, radix-2, 2s complement, signed binary numbers.

1. INTRODUCTION

In order to represent signed binary numbers we have five methods: Signed Magnitude Representation, One's complement, Two's complement, Excess-k and Base-2. Among these Two's complement is the best way to represent signed numbers. Because, Signed magnitude has multiple representations for zero (00000000 (+0) and 10000000 (−0)) and one's complement has multiple representation for zero (00000000 (+0) and 11111111 (−0)) and it is necessary to do add an end-around carry back into the resulting sum. This is illustrated in the following example. Consider the addition of -5(11111010) and +7(00000111).

	Binary	Decimal	
	11111010	-5	
+	00000111	+7	
1	00000001	+1	
		1	+1
	10	+2	

Figure-1. Addition of +5 and-7 in 1s complement.

In the above case, the binary addition alone gives 00000001, which is incorrect. Only when the carry is added in back then it gives the correct output.

A. Two's complement

Two's Complement is a method used to represent negative binary numbers in a signed binary number system. In two's complement form, a negative number is the 2's complement of its positive number with the subtraction of two numbers being A – B = A + (2's complement of B) using the same process as before as two's complement is one's complement + 1.

The main advantage of two's complement over the one's complement is that there is no double-zero problem and it is an easy to generate the two's complement of a signed binary number. Hence, arithmetic operations are relatively easier to perform when the numbers are represented in the two's complement format.

Method to find the negation of a number in two's complement:

1. Invert all the bits through the number.
2. Add one.

Example: For 37 which is 00100101 in binary:

1. Negation of (00100101) → 11011010
2. Add 1 → 11011011 (-37 in two's complement)

Similarly we can find the negation of any given number. The range of two's complement number is – 2^N/2 to + (2^N-1 - 1). I.e. by an 8 bit number (N=8), we can represent -128 to +127.

Manish Chaudhary [2] proposed a Booth's multiplier using Carry Save Adder (CSA) and full adders & half adders; here Ripple carry Adder (RCA) has been used. It shows improvement when compared to other adders. Anju [10] compared the performance of signed booth multiplier and unsigned Vedic Multiplier, here a signed Vedic Multiplier has been proposed and it shows the improvement when compared to signed booth multiplier. Sudeep. M.C [1] proposed an unsigned multiplier based on Urdhva Tiryakbhyam. K. Harika [8] compared the performance of different multipliers; the proposed multiplier shows the improvement of all the multipliers.

2. BOOTH'S MULTILPLICATION ALGORITHM

Two's complement method is the best way to represent signed numbers. So Booth's multiplication algorithm multiplies two signed bit numbers in two's complement notation. It calculates the signed output by repeatedly adding two values named A and S to a product P, then perform Arithmetic Right Shift (ARS) on P.

A. Steps involved in booth multiplication

Take two numbers M and R, where M is the multiplier and R is the multiplicand. Let x and y be the total number of bits in M and R respectively.

1. Determine the values of A and S, and the initial value of P. All of these numbers should have a length equal to (x + y + 1).
   - **A**: To find A Fill the most significant x bits with the value of M and fill the remaining (y + 1) bits with zeros.
   - **S**: Take 2's complement of M, which gives -M. To Calculate S fill the most significant x bits with the value of -M then fill the remaining (y + 1) bits with zeros.
   - **P**: To find P fill the most significant x bits with zeros. To the right of this, append the value of R then fill the least significant bit with a zero.
2. Then determine the two least significant bits of P and perform the following steps.
   - If they are 01, calculate the value of P + A. Ignore Most Significant Bit (MSB) in case of overflow.
   - If they are 10, calculate the value of P + S. Ignore Most Significant Bit (MSB) in case of overflow.
   - If they are 00, no operations required. Use P directly in the next step.
   - If they are 11, no operations required. Use P directly in the next step.
3. Perform Arithmetic Right Shift to value obtained in the 2nd step. Consider this is the new P value.
4. Take the new P value and repeat steps 2 and 3 until they have been done y times.
5. Drop the least significant (rightmost) bit from P. This is the product of M and R.

Find 9 × (−13), with **M** = 9 and **R** = −13, and *x* = 5 and *y* = 5:

- M = 01001, -M = 10111, R = 10011
- A = 01001 00000 0
- S = 10111 00000 0
- P = 00000 10011 0

Perform the loop five times:

a) P = 0000 10011 0. The last two bits are 10.
   i. 10111 10011 0. P = P + S.
   ii. P = 11011 11001 1. Arithmetic right shift.

b) P = 11011 11001 1. The last two bits are 11.
   i. P = 11101 11100 1. Arithmetic right shift.

c) P = 11101 11100 1. The last two bits are 01.
   i. P = 00110 11100 1. P = P + A.
   ii. P = 00011 01110 0. Arithmetic right shift.

d) P = 00011 01110 0. The last two bits are 00.
   i. P = 00001 10111 0 Arithmetic right shift.

e) P = 00001 10111 0. The last two bits are 10.
   i. P = 11000 10111 0. P = P + S.
   ii. P = 11100 01011 1. Arithmetic right shift.

The product is 1110001011 which is -117

3. VEDIC MULTIPLIER

Vedic mathematics is mainly based on 16 sutras.VM is based on the 14th sutra which is Urdhva Tiryakbhyam (Vertically and Crosswise). 2x2 Multiplier is the basic building block of Vedic Multiplier. It multiply two 2-bit binary number and produces 4-bit number as a result. It is shown in Figure-2. It uses 4 AND gates and two Half Adders (HA).

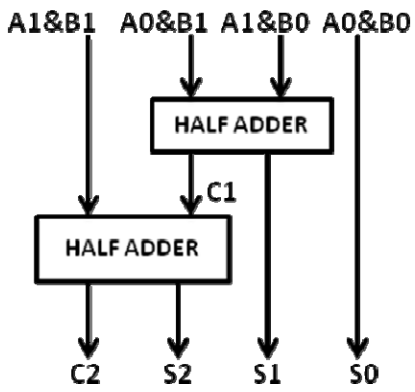

Figure-2. 2x2 multiply block.

a) N*N Vedic multiplier

The architecture for NxN unsigned VM is shown in Figure-3.For N-bit Multiplier design, first the basic block 2x2 bit multiplier is designed (Figure-2), then a 4x4 block is designed using this 2x2 block, then a 8x8 block is designed using this 4x4 block, then a 16x16 bit block is designed using this 8x8 block, 16x16 bit block have been made and using this 8x8 block, then a 32x32 bit block is designed using this 16x16 block, finally a N bit Multiplier is designed. From these multipliers we have concluded that to construct a NxN bit Vedic Multiplier, four N/2xN/2 VMs and one N-bit Ripple Carry Adders (RCAs), Adder 2 and Adder 3 are required.

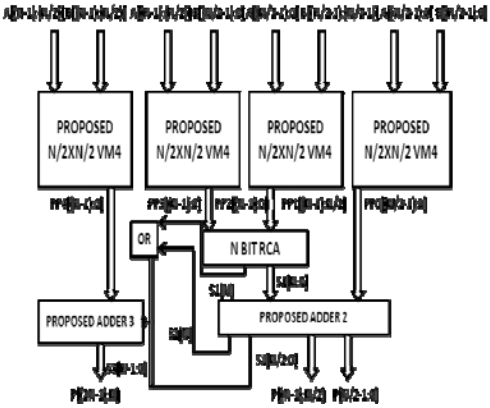

Figure-3. NxN unsigned VM.

b) N-Bit Ripple Carry Adder (RCA)

The architecture for N-bit RCA is shown in Figure-4. It adds two N-bit binary number and produces N+1 bit binary number as a result. So it consists of 1 Half Adder (HA) and N-1 Full Adders (FA).

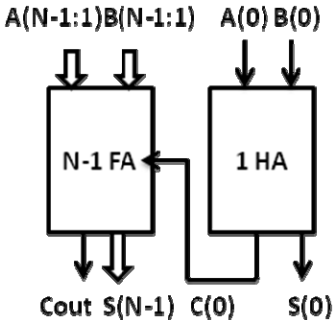

Figure-4. N bit Ripple Carry Adder (RCA).

c) Adder 2 for N*N VM

The generalized architecture for proposed Adder 2 is shown in Figure-5. Adder 2 uses N/2-1 FAs and N/2+1 HAs.

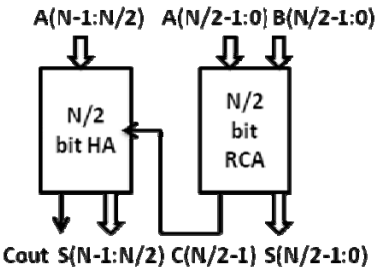

Figure-5. Generalized Architecture for Adder 2.

d) Adder 3 for N*N VM

The generalized architecture of proposed Adder is shown in Figure-6. Adder 3 has N/2 FAs and N/2 HAs.

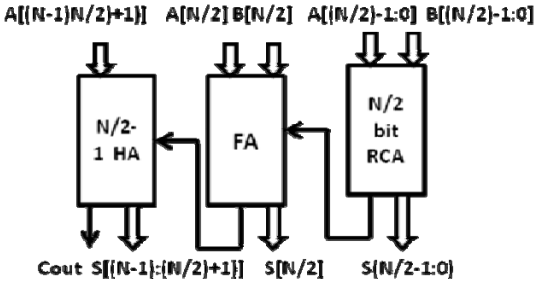

Figure-6. Generalized Architecture for Adder 3.

4. FLOWGRAPH FOR PROPOSED SIGNED MULTIPLIER

Find 7 × (−6):

- a=0111;b=1010;
- MSB of a=0, so consider the four bits as it was.
- MSB of b=1, so take 2s compliment of 4 bits, it is 0110.
- Give the above 4 bits to a unsigned VM and get the output as 00101010.
- Calculate the sign bit by EXOR the MSB of the multiplier and multiplicand. Here 0 EXOR 1 is 1.
- Sign bit is 1, so take the 2s compliment of output of the unsigned multiplier. Otherwise consider the unsigned multiplier output as the final output.
- In this case sign bit is 1. So take 2scomplement or unsigned multiplier, which gives the final result as 11010110.

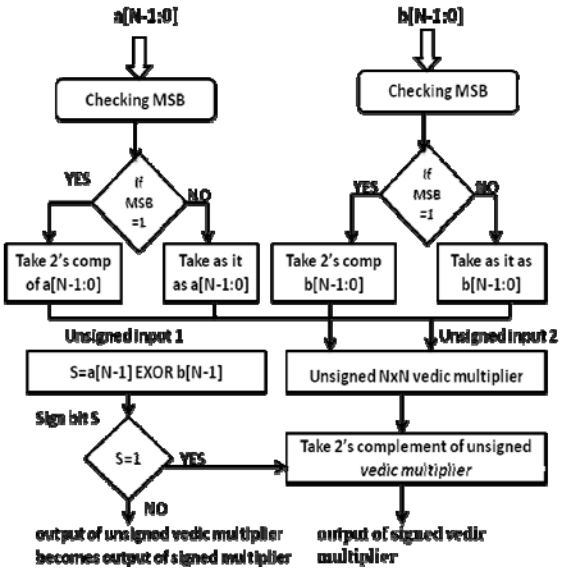

Figure-7. Flowchart for proposed signed VM.

5. RESULTS AND DISCUSSIONS

a) Area and path delay comparison

The proposed Signed VM and conventional Booth Multiplier are coded in Verilog, synthesized and simulated using ISE simulator. It is implemented on the iWavesystems Unified Learning Kit Spartan6 family xc6slx25t-2fgg484 FPGA. Table-1 shows the comparison of no. of slices used by both the multipliers.

Table-1. Area comparison.

Multiplier	Proposed signed VM	Conventional Booth
4x4	39	47
8x8	164	248
16x16	652	956
32x32	2584	3583

Table-2 shows the comparison of maximum combinational path delay of both the multipliers.

Table-2. Path delay comparison.

Multiplier	Proposed signed VM (ns)	Conventional Booth (ns)
4x4	16.932	17.86
8x8	25.170	39.28
16x16	41.846	75.23
32x32	72.243	154.66

From the above two table, one can infer that the proposed signed multiplier has low path delay and it utilizes minimum no. of slices required on FPGA.

b) Implementation on ULK

Figure-8 shows the output waveform for the positive and negative numbers.

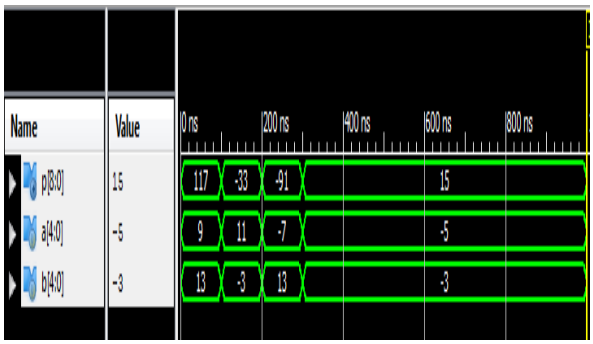

Figure-8. Output waveform.

c) Implementation on ULK

Suppose if we want to multiply -21846*21845. The output is -477225870, which is 11100011100011100001110001110010.

In ULK we have only 8 output LEDs. In order to display 32 bit output here two switches and four modes have been selected. If the switch positions are 00 then it will display the lower 8 bits of the output. If the is 01 then next 8 bits will be displayed. If it is 10 then next 8 bits will be displayed. If it is 11 the upper 8 bits will be displayed as shown in the below figures.

Mode 1: Switch position 00, output 01110010

Figure-9. ULK output for Mode 1.

Mode 2: Switch position 01, output 00011100

Figure-10. ULK output for Mode 2

Mode 3: Switch position 10, output 10001110

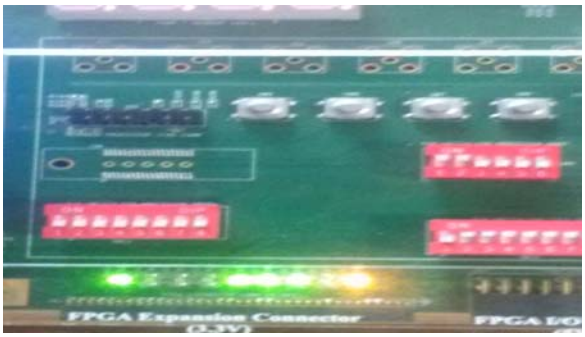

Figure-11. ULK output for Mode 3.

Mode 4: Switch position 11, output 11100011:

Figure-12. ULK output for Mode 4.

6. CONCLUSIONS

In our design, efforts have been made to reduce the propagation delay and area utilization and achieved an improvement in the reduction of delay with 53.29% and area with 27.88% for 32x32 Proposed Signed VM when compared to conventional Booth Multiplier. The proposed architecture is applicable to all cases of multiplication with increase speed of devices and minimum delay. The high speed implementation of such a multiplier has wide range of applications in image processing, arithmetic logic unit in DSP and VLSI signal processing. Further the proposed multiplier work can be extended for low power VLSI applications.

REFERENCES

[1] Sudeep M.C., Sharath Bimba M. and Mahendra Vucha. 2014. Design and FPGA Implementation of High Speed Vedic Multiplier. International Journal of Computer Applications (IJCA), ISSN: 0975-8887, Volume-90, Issue -60, March, pp.6-9.

[2] Manish Chaudhary. and Mandeep Singh Narula. 2013. FPGA Implementation of Booth's and Baugh-wooley Multiplier using Verilog. International Journal of Innovative Technology and Exploring Engineering (IJITEE), ISSN: 2278-3075. Volume-3, Issue -1, June.

[3] S. Nagaraj. and R. Mallikarjuna Reddy. 2013. FPGA Implementation of Modified Booth Multiplier. International Journal of Engineering Research and Applications (IJERA), ISSN: 2248-9622. Volume-3, Issue -2, March-April. pp. 295-299.

[4] Ramalatha M. Dayalan. K. D. Dharani, P. Priya. and S. Deoborah. 2009. High Speed Energy Efficient ALU Design using Vedic Multiplication Techniques", International Conference on Advances In Computational Tools for Engineering Applications (ACTEA) IEEE, pp. 600-603, July15-17.

[5] Laxman S., Darshan Prabhu R., Mahesh S. Shetty., Mrs. Manjula BM. and Dr. Chirag Sharma. 2012. FPGA Implementation of Different Multiplier Architectures. International Journal of Emerging Technology and Advanced Engineering (IJETAE), ISSN:2250-2459. Volume-2,Issue -6, June.

[6] Sandesh S. Saokar, R. M. Banakar. and Saroja Siddamal. 2012. High Speed Signed Multiplier for Digital Signal. Processing Applications. IEEE.

[7] V. R. Raut. and P. R. Loya. 2014. FPGA Implementation of Low Power Booth Multiplier Using Radix-4 Algorithm. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering (IJAREEIE), ISSN: 2320-3765. Volume-3, Issue -8, August.

[8] K. Harika, B.V. Swetha, B. Renuka, D. Lakshman Rao, and S. Sridhar. 2014. Analysis of Different Multiplication Algorithms & FPGA Implementation. IOSR Journal of VLSI and Signal Processing (IOSR-JVSP) ,ISSN:2319-4200, Volume-4, Issue -2, March-April. pp. 29-35.

[9] Pushpalata Verma. and K. K. Mehta. 2012. Implementation of an Efficient Multiplier based on Vedic Mathematics Using EDA Tool. International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958. Volume-1, Issue-5, June.

[10] Anju. 2013. Performance Comparison of Vedic Multiplier and Booth Multiplier. International Journal of Engineering and Advanced Technology (IJEAT), ISSN: 2249-8958. Volume-2, Issue -5, June 2013.

[11] Sneha Manohar, Ramteke, Yogeshwar, Khandagre and Alok Dubey. 2014. Implementation of Low Power Booth's Multiplier by Utilizing Ripple Carry Adder. International Journal of Scientific and Engineering Research (IJSER), ISSN: 2229 –5518, Volume-5, Issue-5, May.

[12] Garima Tiwari. 2013. Analysis, Verification and FPGA Implementation of Low Power Multiplier. International Journal of Scientific and Engineering Research (IJSER), ISSN: 2319 –1163, Volume-2, Issue-3, May.

[13] Sukhmeet Kaur, Suman. and Manpreet Signh Manna. 2013. Implementation of Modified Booth Algorithm (Radix 4) and its Comparison with Booth Algorithm (Radix 2). Advance in Electronic and Electric Engineering, ISSN: 2231 –1297, Volume-3, Issue-6, pp.683-690.