Design And Hardware Implementation Of Vedic Multiplier

N. G. Nirmal, Dr. D. T. Ingole
Department of Electronics & Communication Engineering S.S.G.B.C.O.E&T, Bhusawal,
Prof Ram Meghe Institute of Technology & Research, Badnera

Abstract

In this paper multiplier architecture is proposed based on algorithm of ancient Indian Vedic Mathematics, high speed applications. It is based on the Vertical and Crosswise algorithm of ancient Indian Vedic Mathematics, Urdhva Tiryakbhyam Sutra generating all partial products and their sums in one step. The design basic block which are adders are designed in a generic way so N-bit multiplier design can be done using the designed architecture. The design implementation is done using VHDL (Hardware Description Language). The design code is tested using Modelsim 6.0 Simulator. The code is synthesized in Xilinx ISE 9.2i using: Xilinx, Family: Spartan XC2S200PQ208 device, Speed Grade: -5.The combinational delay of the 16×16 Vedic multiplier is found to be 80.155ns.

1. Introduction

Multiplication is an important fundamental function in arithmetic operations. Multiplication operations are among some of the frequently used functions currently implemented in many Digital Signal Processing applications such as convolution, Fast Fourier Transform, filtering and in microprocessors in its arithmetic and logic unit. Since multiplication dominates the execution time of most DSP algorithms, so there is a need of high speed multiplier. One of the key arithmetic operations in DSP applications is multiplication and the development of fast multiplier circuit has been a subject of interest over decades. Reducing the time delay and power consumption are very essential requirements for many applications .In this paper Urdhva Tiryakbhyam Sutra is first applied to the binary number system and is used to develop digital multiplier architecture. This is shown to be very similar to the popular array multiplier architecture. This paper presents a systematic design methodology for fast and area efficient digit multiplier based on Vedic mathematics. The Multiplier Architecture is based on the Vertical and Crosswise algorithm of ancient Indian Vedic Mathematics. Urdhva Tiryakbhyam Sutra is the general multiplication formula applicable to all cases of multiplication. The tools Modelsim6.0E has been used for simulation. XILINX ISE9.2i has been used for synthesis and verification. The Vedic multiplication technique is based on one of the 16 Vedic sutras or aphorisms, which are actually word formulae describing natural ways of solving a whole range of mathematical problems [1].

2. Vedic Mathematics

2.1. Ancient Vedic Mathematics

Ancient Indian mathematics is called as Vedic Mathematics [1]. Vedic mathematics from Vedas was first proposed by Sri Bharati Krishna Tirtha, after his survey on Vedas. Vedic mathematics reduces the complexity in calculations that exist in conventional mathematics. From the sixteen sutras available in Vedic mathematics. Among them only two sutras are applicable for multiplication operation. They are Urdhva Triyakbhyam Sutra (literally means vertically and cross wise) and Nikhilam Sutra (literally means all from 9 and last from 10).Vedic Mathematics provides some effective algorithms which can be applied to various application fields of engineering.

2.2. Urdhva Tiryakbhyam Sutra

The given Vedic multiplier based on the Vedic multiplication formulae (Sutra). This Sutra has been traditionally used for the multiplication of two numbers. In this design, we have applied the same ideas to make the idea implemented in digital hardware. Urdhva Tiryakbhyam Sutra is a general multiplication formula applicable to all cases of multiplication. It means “Vertically and crosswise”. The digits on the two ends of the line are multiplied and the result is added with the previous carry. When there are more lines in one step, all the results are added to the previous carry. The least significant digit of the number thus obtained acts as one of the result digits and the rest act as the carry for the next step. Initially the carry is taken to be as zero. The algorithmic diagram for multiplication of two 4-bit numbers is as shown in Figure 1.

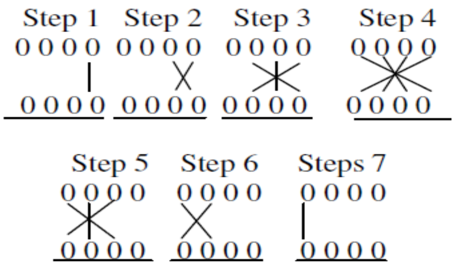


Figure 1. Algorithmic diagram for multiplication of two 4-bit numbers.

3. Vedic multiplier architecture

3.1. Architecture of Vedic Multipliers

The hardware architecture of 2×2, 4×4, 16×16 bit Vedic multiplier (VM) modules are displayed in the below sections. In 2×2 bit multiplier, the multiplicand has 2 bits each and the result of multiplication is of 4 bits. So in input the range of inputs goes from (00) to (11) and output lies in the set of (0000, 0001, 0010, 0011, 0100, 0110, 1001). By using Urdhva Tiryakbhyam, the multiplication takes place as illustrated in Figure. 2. Here multiplicands are a0,a1 and b0,b1. The output can be of four digits, say Q3Q2Q1Q0. As per basic method of multiplication, result is obtained after getting partial product and doing addition. The first step in the multiplication is vertical multiplication of LSB of both multiplicands, then in the second step, that is crosswise multiplication and addition of the partial products. Then Step 3 involves vertical multiplication of MSB of the multiplicands and addition with the carry propagated from Step 2.

Multiplier	=	a1	a0	
Multiplicand	=	b1	b0	
			a1b0	a0b0
		a1b1	a0b1	
Q3		Q2	Q1	Q0

Figure 2. Algorithm for 2×2 multiplier

Product: Q0: a0b0
Q1: (a1b0) xor (a0b1)
Q2: (a1b1) xor (a1b0 and a0b1)
Q3: (a1b1and a1b0 and a0b1)

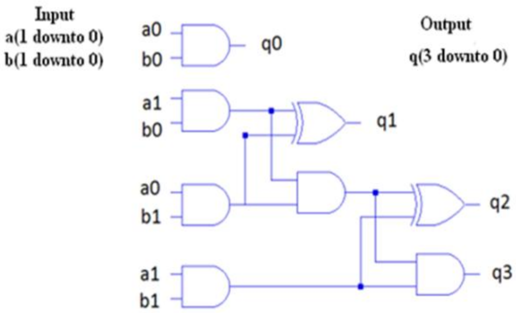


Figure 3. Hardware realization of 2×2 multiplier

3.2. Generic adder block

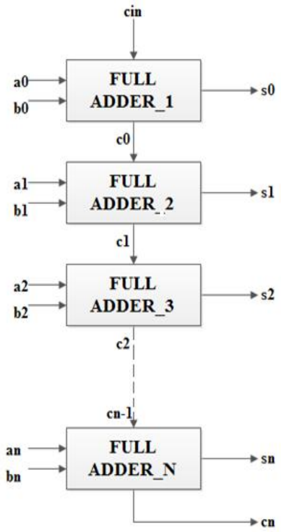


Figure 4. Hardware realization of generic adder

The above shown Figure. 4 is an N bit generic adder. Here we are using ripple carry adder where initially the carry to the first block is zero by default. The carry generated to the first block is applied to the second block and carry generated to second block is applied to the next one and this process is repeated up to the N times. The Boolean function of a generic N bit ripple-carry adder can be written as:

For addition

Sum(i) = x(i) XOR y(i) XOR c(i-1) , $0 \le i \le N-1$;

Where i represent the current ith value and the range of i is between zero (0) to N-1.The result of this calculation is saved in ith value of sum(i).Each 1-bit adder in above Figure. 4 is described by above mentioned formulas.

For carry generation:

C(i) = x(i)y(i) + x(i)c(i-1) + y(i)c(i-1) , $0 \le i \le N-1$;

Cout=c(N-1) and c(-1)=cin

Where i represent the current ith value and the range of i is between zero (0) to N-1.The result of this calculation is saved in ith value of sum(i). Each 1-bit carry generated in above Figure.4. is described by above mentioned formulae.

3.3. 4×4 Vedic Multiplier

The 4x4 Multiplier is made by using 4, 2x2 multiplier sub blocks. Here, the multiplicands are having the bit size of (n=4) whereas, the result is of 8 bit in size. The input is broken in to smaller groups of size of n/2 = 2, for both inputs, that is a and b. These newly formed groups of 2 bits are given as input to 2×2 multiplier block and the result produced 4 bits, which are the output produced from 2×2 multiplier block are sent for addition to an addition tree.

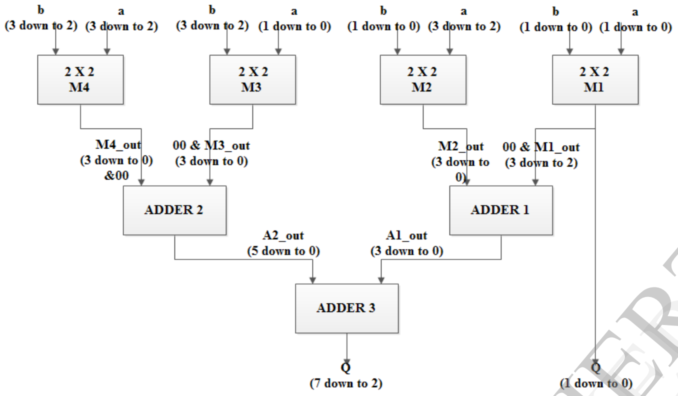


Figure 5. Hardware realization of 4×4 multiplier

As the generic adder is designed the designing of high bit multipliers in not an issue using the structural modeling it becomes easy for just call the predefined components and design the multiplier. Figure. 6. Shows the hardware realization diagram of an 16×16 multiplier.

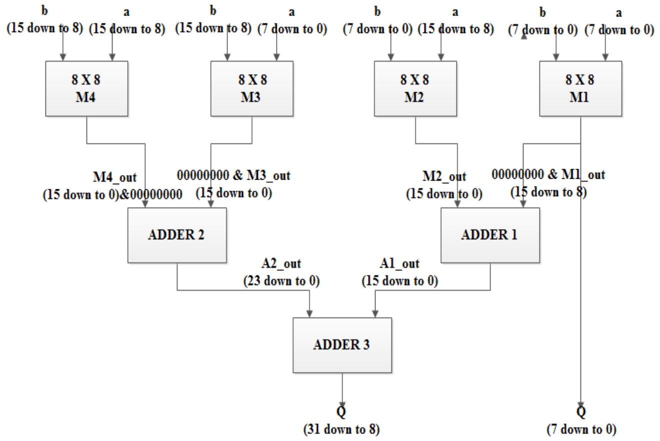


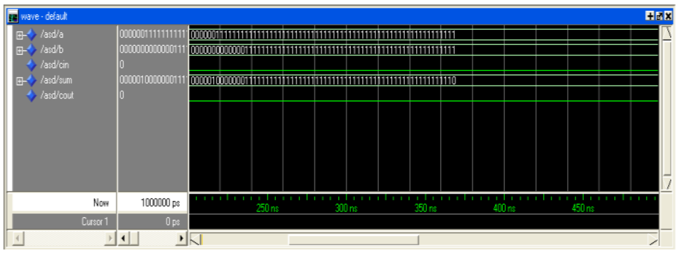
Figure. 6. Hardware realization of 16×16 multiplier

4. Simulation of Vedic multiplier and results

4.1.VHDL Test Bench for Simulation of 16 bit Multiplier.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;
ENTITY tb_vhd IS
END tb_vhd;
Architecture tb_vhd1 of tb_vhd is
 component ved16 is
 Port ( a,b : in  STD_LOGIC_VECTOR (15 downto 0);
    qout : out  STD_LOGIC_VECTOR (31 downto 0));
 End component;
 Signal a_s : std_logic_vector ( 15 downto 0 );
 Signal b_s : std_logic_vector ( 15 downto 0 );
 Signal qout_s : std_logic_vector ( 31 downto 0 );
  begin
  U0 : ved16
      Port map ( a => a_s, b => b_s,qout => qout_s );
       Test : Process
        begin
         L1 : for i in 0 to 65535 loop
         L2 : for j in 0 to  65535 loop
            a_s <= conv_std_logic_vector( i,16);
            b_s   <= Conv_std_logic_vector( j,16);
                          wait for 10 ns;
          end Loop L2;
         end Loop L1;
         wait;
        end process;
   end tb_vhd1;
```

4.2. Simulation result using Modelsim6.0



```
/asd/a     0000001111111111  0000001111111111111111111111111111111111111111111111111
/asd/b     0000000000000111  0000000000000111111111111111111111111111111111111111111
/asd/cin   0
/asd/sum   0000010000000111  0000010000000111111111111111111111111111111111111111110
/asd/cout  0
Now        1000000 ps
Cursor 1   0 ps
```

Figure. 7. Simulation result of generic adder

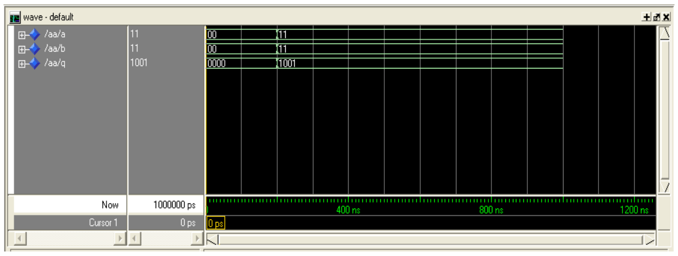


Figure. 8. Simulation result of 2×2 Multiplier

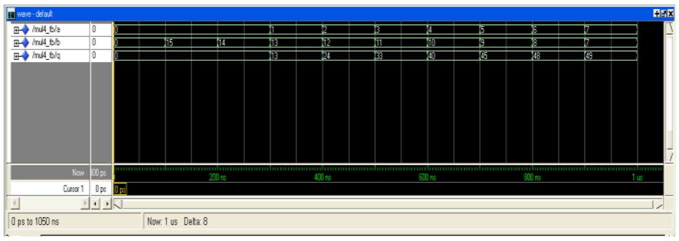


Figure. 9. Simulation result of 4×4 Multiplier

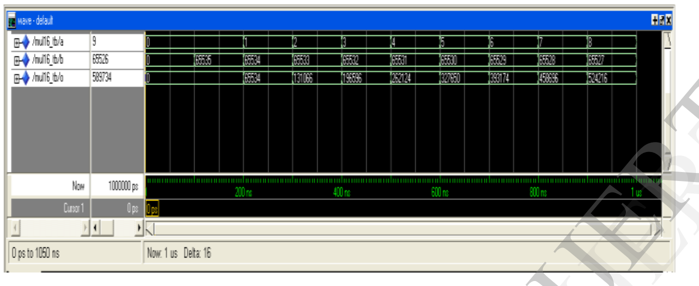


Figure.10. Simulation result of 16×16 Multiplier

4.3. Synthesis results for 16×16 multiplier using Xilinx

ISE9.2i Selected Device: XC2s200pq208-5

The table below shows the synthesis report data that is the device utilization summary by the design 16×16 multiplier.

Table 1 : Device Utilization Summary

SR. No	Parameter	Utilization	Percentage
1	Number of Slices:	475 out of 2352	20%
2	Number of 4 input LUTs:	823 out of 4704	17%
3	Number of IOs:	64	0%
4	Number of bonded IOBs:	164 out of 140	45%

Timing Summary:-Speed Grade: -5

Total Time: 80.155ns (29.385ns logic, 50.770ns route)

Memory summary:-Total memory usage is 151340 kilobytes.

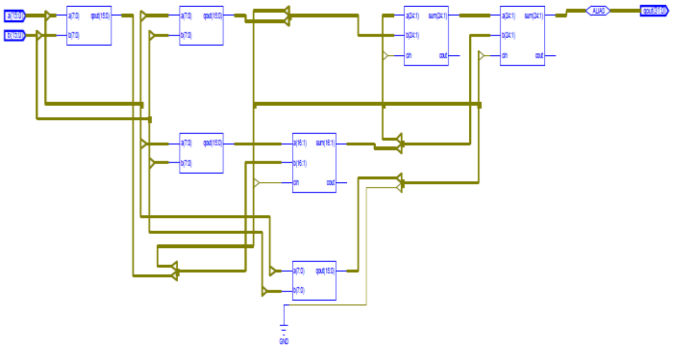


Figure.11.RTL Schematic of 16×16 multiplier

10. References

[1] Jagadguru Swami, Sri Bharati Krishsna Tirthji Maharaja, "*Vedic Mathematics*", Motilal Banarsidas, Varanasi, India,1986, pp. 40-63.

[2] Harpreet Singh Dhilon and Abhijit Mitra, "*A Reduced-Bit Multiplication Algorithm for Digital Arithmetic*", International Journal of Computational and Mathematical Sciences, Waset.

[3] John P. Uyemura, "*Introduction to VLSI Circuits and Systems*", John Wiley & Sons, Inc.

[4] Alex Panato, Sandro Silva, Flavio Wagner, Marcelo Johann, Ricardo Reis, Sergio Bampi, "*Design of Very Deep Pipelined Multipliers for FPGAs*", Proceedings of the Design, Automation and Test in Europe Conference and Exhibition Designers' Forum IEEE 2004.

[5] Ramesh Pushpangdan, Vineeth Sukumaran, Rono Innocent, Dinesh Sasikumar, Vaisak Sundar "*High Speed Vedic Multiplier for Digital Signal Processor*".IETE Journal of Research ,Vol- 55,Issue-6,Nov-Dec 2009.

[6] Prabir Saha, Arindam Banerjee, Partha Bhattacharyya, Anup Dandapat, " *High Speed ASIC Design of Complex Multiplier Using Vedic Mathematics*" ,Proceeding of the 2011 IEEE Students' Technology Symposium 14-16 January, 2011.

[7] Purushottam D. Chidgupkar, Mangesh T. Karad, "*The Implementation of Vedic Algorithms in Digital Signal Processing*", Global Journal of Engineering. Education, Vol.8, No.2, 2004, pp. 153-157.

[8] Vedamu.org, http://www.vedamu.org ,2012.

[9] Himanshu Thapliyal and M.B. Srinivas,"*High Speed Efficient N×N Bit Parallel Hierarchical Overlay Multiplier Architecture Based On Ancient Indian Vedic Mathematics*", Transactions on Engineering, Computing and Technology v2 December 2004.