INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH & DEVELOPMENT

***ISSN: 2278 – 0211 (Online)***

# High Speed Signed Multiplier For Digital Signal Processing Applications

**Mr. Jagadish.K.N**
PG Student, VLSI Design and Embedded System,
B.G.S. Institute of Technology, B.G.Nagar, Karnataka, India
**Mr. Nagaraj.C**
Assistant Professor, Dept. of Electronics and Communication Engineering,
B.G.S. Institute of Technology, B.G.Nagar, Karnataka, India

***Abstract:***

*The speed of a multiplier is of utmost importance to any Digital Signal Processor (DSPs). Along with the speed its precision also plays a major role. Although Floating point multipliers provide required precision they tend to consume more silicon area and are relatively slower compared to fixed point (Q-format) multipliers. In this paper we propose a method for fast fixed point signed multiplication based on Urdhava Tiryakbhyam method of Vedic mathematics. The coding is done for 16 bit (Q15) and 32 bit (Q31) fractional fixed point multiplications using Verilog and synthesized using Xilinx ISE version 12.2. Further the speed comparison of this multiplier with normal booth multiplier and Xilinx LogiCore parallel multiplier Intellectual Property (IP) is presented. The results clearly indicate that Urdhava Tiryakbhyam can have a great impact on improving the speed of Digital Signal Processors.*

***Key words:*** *Q-format; Urdhava Tiryakbhyam; Vedic Mathematics; Fractional fixed point;Intellectual Property.*

**1.Introduction**

Vedic Mathematics hails from the ancient Indian scriptures called "*Vedas*" or the source of knowledge. This system of computation covers all forms of mathematics, be it geometry, trigonometry or algebra. The striking feature of Vedic Mathematics is the coherence in its algorithms which are designed the way our mind naturally works. This makes it the easiest and fastest way to perform any mathematical calculation mentally. Vedic Mathematics is believed to be created around 1500 BC and was rediscovered between 1911 to 1918 by Sri Bharti Krishna Tirthaji (1884-1960) who was a Sanskrit scholar, mathematician and a philosopher [1]. He organized and classified the whole of Vedic Mathematics into 16 formulae or also called as *sutras*. These formulae form the backbone of Vedic mathematics. Great amount of research has been done all these years to implement algorithms of Vedic mathematics on digital processors. It has been observed that due to coherence and symmetry in these algorithms it can have a regular silicon layout and consume less area [2,3 ] along with lower power consumption.

Normally signal processing algorithms are developed using high level languages like C or Matlab using floating point number representations. The algorithm to architecture mapping using floating point number representation consumes more hardware which tends to be expensive. Fixed point number representation is a good option to implement at silicon level. Hence our focus in this work is to develop optimized hardware modules for multiplication operation which is one of the most frequently used operation in signal processing applications like Fourier transforms, FIR and IIR filters, image processing systems, seismic signal processing, optical signal processing etc. Any attempt to come out with an optimized architecture for this basic block is advantageous during the product development stages.

Considering fixed point representation, 16 bit Q15 format and 32 bit Q31 format provide required precision for most of the digital signal processing applications and it is best suited for implementation on processors. The advantage it provides over floating point multipliers is in the fact that Qformat fraction multiplications can be carried out using integer multipliers which are faster and consume less die area. DSP Processors like TMS320 series from Texas Instruments work on 16 bit Q15 format. In this paper we propose the implementation of fixed point Q-format [6] high speed multiplier using Urdhava Tiryakbhyam method of Vedic mathematics. Further we have also implemented multipliers using normal booth algorithm [8] and Xilinx parallel multiplier Intellectual

Property and presented a comparative study on maximum frequency or speed of these multipliers.

The paper is organized into VI sections. Section II explains fixed point or Q-format representation of a number; III spreads light over Urdhava Tiryakbhyam method of Vedic mathematics; IV explains the architecture of proposed Qformat Urdhava multipliers; V presents the results and comparison and lastly VI provides conclusion of the work.

**2.Fixed Point Arithmetic**

An N-bit fixed point number [6] can be interpreted as either an integer or a fractional number. Integer fixed point is difficult to use in processors due to possible overflow. For e.g. In a 16-bit processor for signed integers the dynamic range is from $-2^{15}$ to $2^{15}-1$ i.e. 32768 to 32767. If 500 is multiplied by 800 the result is 40000 which is an overflow. In order to overcome this situation fractional fixed point representation also known as Q-format is used.

**3.Q-format Representation**

In general any Q-format representation is denoted by Qm.n, where m is the number of bits to represent integer, n denotes number of bits to represent fractional part and the total number of bits is given by N = m+n+1 for signed numbers. For e.g. Q4.11 format signifies that a total of 16 bits are required to represent a fractional number in which 4 bits are reserved for the integer part and 11 bits for the fractional part and 1 bit indicates sign. Special cases of Q-format consist of zero bits to represent the integer part. Q0.15 (Q15) and Q0.31 (Q31) are two such formats for 16 bit and 32 bit representations respectively. As there is no integer part the fractional number has a range between 1 and -1. Therefore the products of such numbers also lie between 1 and -1. This property is best suited for implementing multipliers as the bit length of the product is same as the input bit length and thus Q-format finds its application in digital signal processing hardware.

An N-bit number in Q*m.n* format is represented as follows.[6]

$$a_{n+m}a_{n+m-1} \ldots \ldots a_n \,.\, a_{n-1} \ldots \ldots a_1\, a_0 \qquad (1)$$

Here the '.' between $a_n$. $a_{n-1}$ represents the fixed point and value of (1) is given by,

$$(a_{n+m}2^{N-1} + a_{n+m-1}2^{N-2} \ldots \ldots + a_2 2^2 + a_1 2 + a_0)2^{-n}.$$

When we want to convert a fractional number in the range of the desired Q*m.n* format, we multiply it with $2^n$. The resultant value is truncated or rounded off to the nearest integer. Therefore a small amount of precision loss is involved which reduces as the number of bits representing the fractional part increases. We prefer rounding technique since its error bias in both positive and negative direction is same [6]. Therefore the rounded value will be more precise.

For e.g. Conversion of 0.2625 to Q15 format is done by multiplying it with $2^{15}$ which equals to 8601.6 which when rounded gives 8602. This is stored as 0010000110011010 in a 16 bit memory location. The most significant bit indicates sign of the number. If it is negative then 2's complement method is followed to store the number. Thus a fraction is converted to an integer in a Q-format and the choice of the decimal point lies entirely in the hands of the programmer. In general a Q*m.n* format has a resolution of $2^{-n}$ and its dynamic range lies between $-2^m$ to $2^m - 2^{-n}$. Therefore as the number of bits for fractional representation increases the resolution increases and as the number of bits for integer part increases the dynamic range increases. The resolution of Q15 format is $2^{-15}$, and for Q31format it is $2^{-31}$. Therefore a number represented in Q31 format has higher resolution and is more precise than the one in Q15 format. In this paper we mainly concentrate on Q15 and Q31 formats since they are best suited for implementing multipliers for DSP applications.

**4.Q-format Multiplication**

When two Q15 numbers are multiplied their product is 32 bits long as illustrated in Fig. 1. The product has a redundant or extended sign bit. Since the product stored in memory should also be a Q15 number we left shift the product by one bit and the most significant 16 bits (including sign bit) is

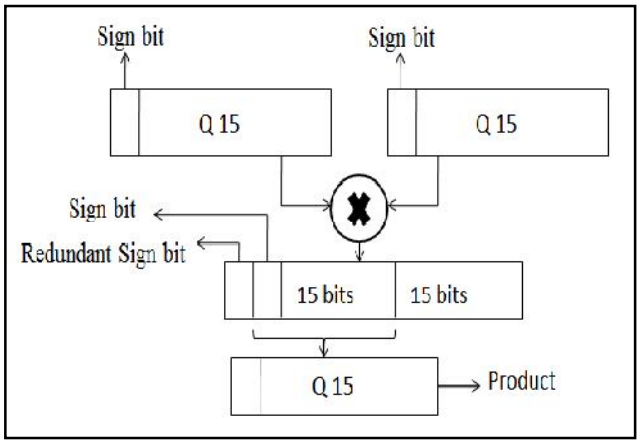


*Figure 1: Multiplication of two Q15 format numbers yielding the*

product in Q15 format itself. stored in the memory. Fig. 1 demonstrates multiplication of two Q15 format numbers. The process remains same for Q31 format wherein after left shifting the product by one bit, the most significant 32 bits are stored in the memory. Therefore with Q-format, multiplications of two fractional numbers can be carried out by using integer multiplications. Integer multiplications consume less area and are faster compared to floating point multipliers which is the major advantage of Qformat representation.

**5.Urdhava Tiryakbhyam Method**

Urdhava Tiryakbhyam [2] is a Sanskrit word which means vertically and crosswire in English. The method is a general multiplication formula applicable to all cases of multiplication. It is based on a novel concept through which all partial products are generated concurrently. Fig. 2 demonstrates a 4 x 4 binary multiplication using this method. The method can be generalized for any N x N bit multiplication. This type of multiplier is independent of the clock frequency of the processor because the partial products and their sums are calculated in parallel. The net advantage is that it reduces the need of microprocessors to operate at increasingly higher clock frequencies. As the operating frequency of a processor increases the number of switching instances also increases. This results in more power consumption and also dissipation in the form of heat which results in higher device operating temperatures. Another advantage of Urdhva Tiryakbhyam multiplier is its scalability. The processing power can easily be increased by increasing the input and output data bus widths since it has a regular structure [3]. Due to its regular structure, it can be easily layout in a silicon chip and also consumes

optimum area [2]. As the number of input bits increase, gate delay and area increase very slowly as compared to other multipliers. Therefore Urdhava Tiryakbhyam multiplier is time, space and power efficient.

The line diagram in fig. 2 illustrates the algorithm for multiplying two 4-bit binary numbers $a_3a_2a_1a_0$ and $b_3b_2b_1b_0$. The procedure is divided into 7 steps and each step generates partial products. Initially as shown in step 1 of fig. 2, method.[7]

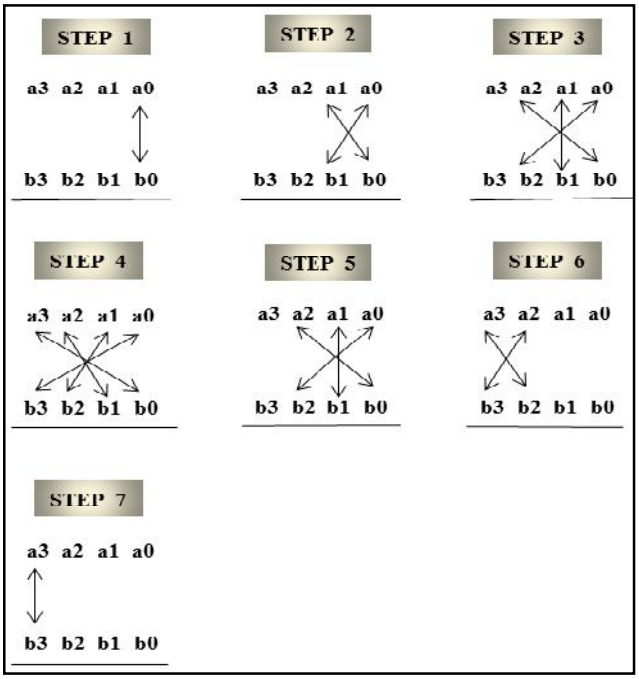


*Figure 2: Multiplication of two 4 bit numbers using Urdhava Tiryakbhyam*

the least significant bit (LSB) of the multiplier is multiplied with least significant bit of the multiplicand (vertical multiplication). This result forms the LSB of the product. In step 2 next higher bit of the multiplier is multiplied with the LSB of the multiplicand and the LSB of the multiplier is multiplied with the next higher bit of the multiplicand (crosswire multiplication). These two partial products are added and the LSB of the sum is the next higher bit of the final product and the remaining bits are carried to the next step. For example, if in some intermediate step, we get the result as 1101, then 1 will act as the result bit(referred as rn) and 110 as the carry (referred as cn). Therefore cn may be a multi-bit number. Similarly other steps are carried out as indicated by the line diagram.

The important feature is that all the partial products and their sums for every step can be calculated in parallel.

Thus every step in fig. 2 has a corresponding expression as follows:

r0=a0b0. (1)

c1r1=a1b0+a0b1. (2)

c2r2=c1+a2b0+a1b1 + a0b2. (3)

c3r3=c2+a3b0+a2b1 + a1b2 + a0b3. (4)

c4r4=c3+a3b1+a2b2 + a1b3. (5)

c5r5=c4+a3b2+a2b3. (6)

c6r6=c5+a3b3 (7)

With c6r6r5r4r3r2r1r0 being the final product [5]. Hence this is the general mathematical formula applicable to all cases of multiplication and its hardware architecture is shown in fig. 3. In order to multiply two 8-bit numbers using 4-bit multiplier we proceed as follows. Consider two 8 bit numbers denoted as AHAL and BHBL where AH and BH corresponds to the most significant 4 bits, AL and BL are the least significant 4 bits of an 8-bit number. When the numbers are

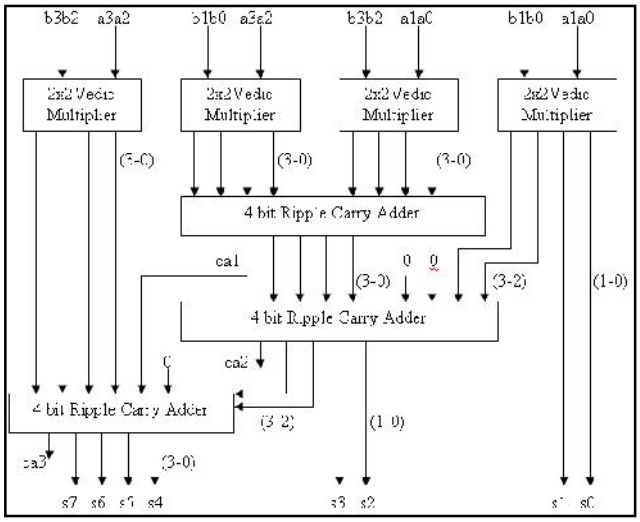


*Figure 3: Hardware architecture of 4 X 4 Urdhva Tiryakbhyam multiplier. [5]*

multiplied according to Urdhava Tiryakbhyam (vertically and crosswire) method, we get,

AH AL

BH BL

(AH x BH) + (AH x BL + BH x AL) + (AL x BL).

Thus we need four 4-bit multipliers and two adders to add the partial products and 4-bit intermediate carry generated. Since product of a 4 x 4 multiplier is 8 bits long, in every step the least significant 4 bits correspond to the product and the remaining 4 bits are carried to the next step. This process continues for 3 steps in this case. Similarly, 16 bit multiplier has four 8 x 8 multiplier and two 16 bit adders with 8 bit carry. Therefore we see that the multiplier is highly modular in nature. Hence it leads to regularity and scalability of the multiplier layout.

**6.Architecture**

Our design of Q-format signed multiplier includes Urdhava Tiryakbhyam integer multiplier [4] with certain modifications as follows. This multiplier is faster since all the partial products are computed concurrently. Considering a 16 bit Q15 multiplier, the product is also a Q15 number which is 16 bits long. Firstly, if the MSB of input is 1 then it is a negative number. Therefore 2's complement of the number is taken before proceeding with multiplication. Since the MSB denotes sign it is excluded and a '0' is placed in this position while multiplying. A Q15 format multiplier consists of four 8 x 8 Urdhava multipliers and the resulting product is 32 bits long as shown in fig. 4. But the product of a Q15 number is also a

Q15 number which should be 16 bits long. Therefore the 32 bit product is left shifted by 1 bit to remove the redundant sign bit and only the most significant 16 bits of this product are considered which constitute the final product. An xor operation is performed on the input sign bits to determine the sign of the result. If the output is '1' it enables the conversion of the 16 bit final result to its 2's complement format indicating a negative

product.

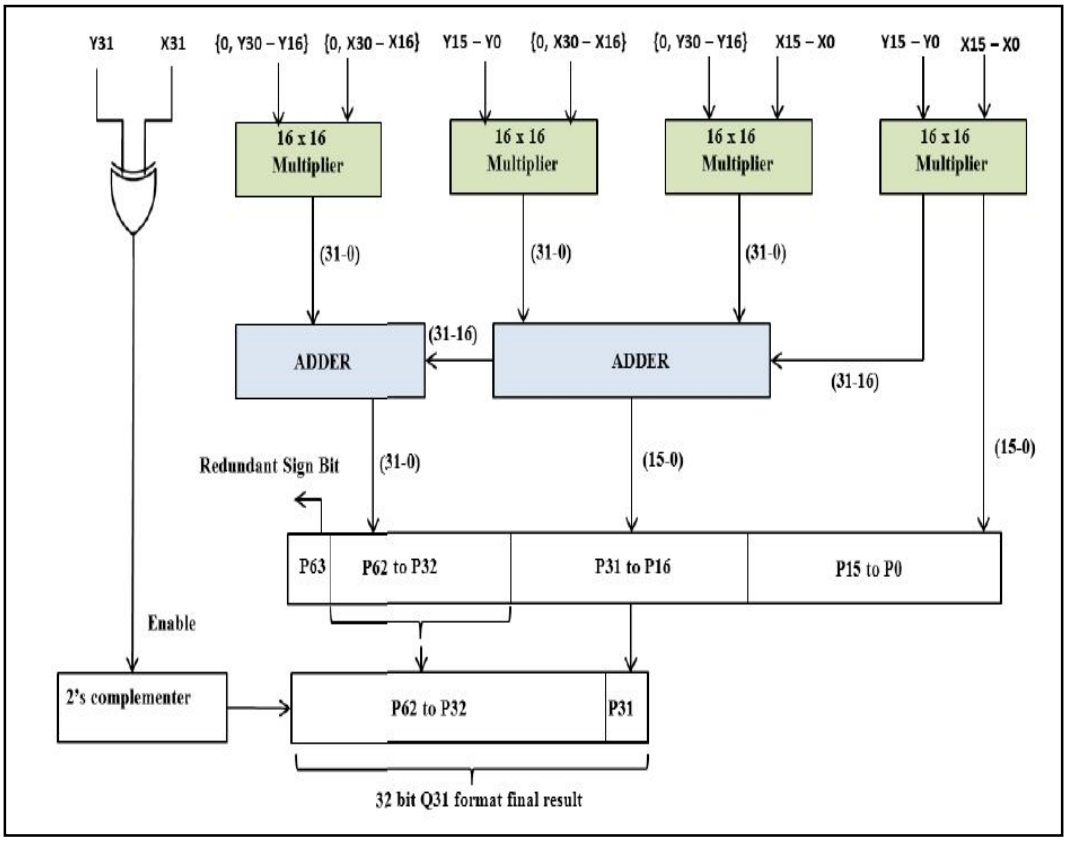
Y31
X31
{0, Y30 – Y16}
{0, X30 – X16}
Y15 – Y0
{0, X30 – X16}
{0, Y30 – Y16}
X15 – X0
Y15 – Y0
X15 – X0
16 x 16 Multiplier
16 x 16 Multiplier
16 x 16 Multiplier
16 x 16 Multiplier
(31-0)
(31-0)
(31-0)
ADDER
(31-16)
ADDER
(31-16)
Redundant Sign Bit
(31-0)
(15-0)
(15-0)
P63
P62 to P32
P31 to P16
P15 to P0
Enable
2's complementer
P62 to P32
P31
32 bit Q31 format final result

*Figure 4*

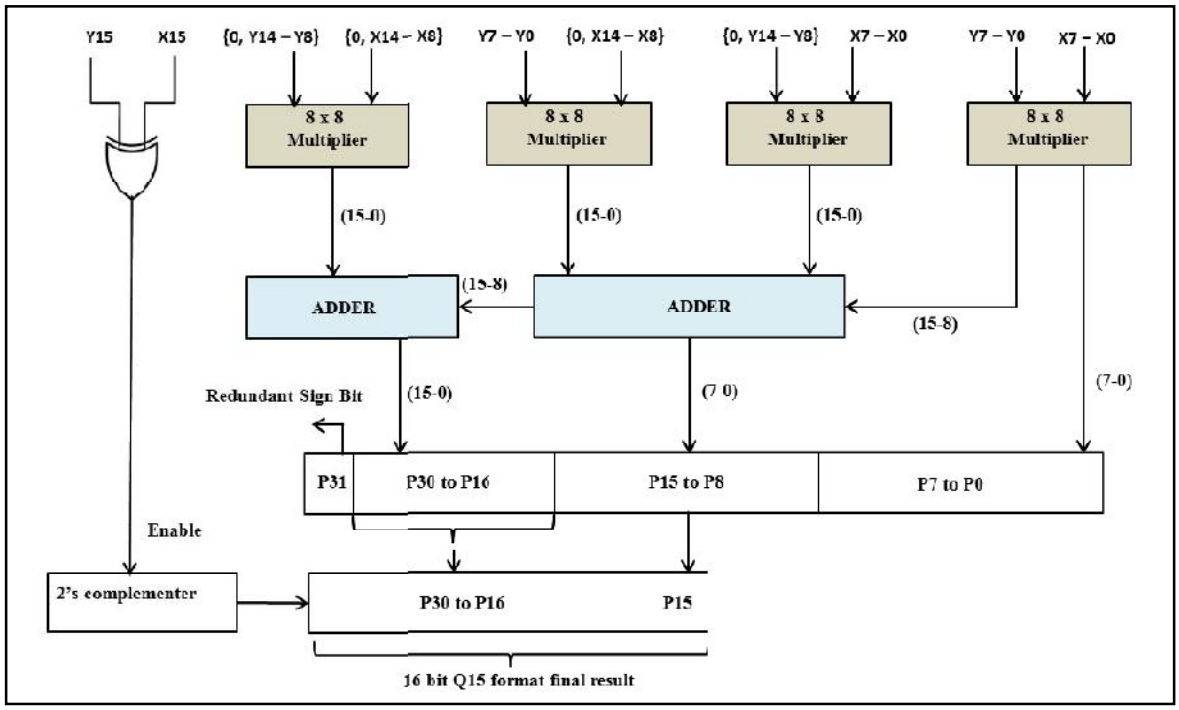


*Figure 5: Architecture of Q31format multiplier. Multiplication of two Q15 numbers X and Y results in a Q15 product denoted by P in the figure.*

Similarly, for a 32 bit Q31 format multiplier as shown in fig. 5, four 16 X 16 Urdhava multipliers are used and only the most significant 32 bits after left shifting by one bit are considered which constitute the final 32 bit Q31 format product. An xor operation similar to Q15 multiplier is used to change the result to 2's complement format if it is negative.

**5.Implementation And Results**

The proposed Urdhava Tiryakbhyam Q-format multiplier is designed using Verilog hardware description language and structural form of coding. The basic block of both Q15 and Q31 multiplier is a 4 x 4 Urdhava Tiryakbhyam integer multiplier which in turn is made up of two 2 x 2 multiplier blocks. The design is completely synchronized by the clock. Further, the Q-format multipliers were also implemented using normal booth's algorithm and Xilinx parallel multiplier Intellectual Property (IP) generated by Xilinx Core Generator which is optimized for speed with no pipelining stage. The code is completely synthesized using Xilinx XST and implemented on device family Virtex-5, device XC5VL50, package FF324 with speed grade -2.

**6.Simulation Results**

The design was simulated using Isim on Xilinx ISE 12.2 version.

For Q15 format multiplication as shown in fig. 6,

Input1 =-0.75 = 1010 0000 0000 0000

Input2 =- 0.25 = 1100 0000 0000 0000

Output = 0.1875 = 0001 1000 0000 0000

For Q31 format multiplication as shown in fig. 7,

Input1= -0.666666= 10101010101010101011000001000010

Input2= 0.333333= 00101010101010101010011111011111

Output= 1110 0011 1000 1110 0011 1100 1001 1110 whose value is -0.222221777774393558502197265625.But the actual value of the product is -0.222221777778. Therefore precision loss is involved in this multiplication and is found to be 3.60644E-12 which is less than the resolution of Q31 representation i.e. $2^{-31}$. Thus it provides 32 bit accurate product which is acceptable for most of the DSP applications. As shown in table 1, the comparison report suggests that a Q31 format Urdhava multiplier is faster than Xilinx parallel multiplier intellectual property (IP) by 1.25 times although the slice LUT usage is more for Urdhava multiplier.

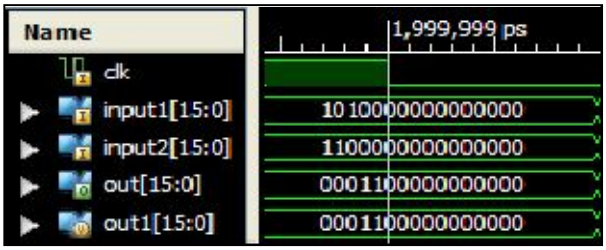


*Figure 6: Q15 multiplication*

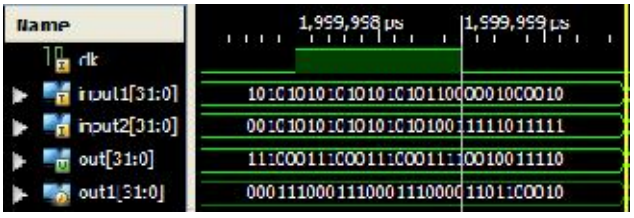


*Figure 7: Q31 multiplication*

Also compared to booth multiplier the speed of Urdhava Qformat multiplier is faster by 2.61 times. For a Q-15 format multiplier, as seen in table 2 the speed factor improvement

is 1.40 and 1.84 times compared to Xilinx parallel multiplier IP and booth multiplier respectively. When Virtex-5 DSP48E slices were used with Xilinx parallel multiplier IP, Urdhava multiplier still proved to be faster indicating that it is the best choice for implementing faster multipliers on FPGA..

| | **Maximum Frequency (in MHz)** | **6-input Slice LUT Usage** | **Factor by which Urdhava Multiplier is faster** |
|---|---|---|---|
| Urdhava Multiplier | 158.90 | 2710/19200 | - |
| Xilinx parallel multiplier IP | 126.24 | 1896/19200 | 1.25 times |
| Normal Booth Multiplier | 60.88 | 3047/19200 | 2.61 times |

*Table 1: Comparison Of 32 Bit Q31-Format Multipliers*

| | **Maximum Frequency (in MHz)** | **6-input Slice LUT Usage** | **Factor by which Urdhava Multiplier is faster** |
|---|---|---|---|
| Urdhava Multiplier | 236.18 | 662/19200 | - |
| Xilinx parallel multiplier IP | 168.77 | 434/19200 | 1.40 times |
| Normal Booth Multiplier | 128.35 | 718/19200 | 1.84 times |

*Table 2: Comparison Of 16 Bit Q15-Format Multipliers*

| | **Q15 format Maximum Frequency** | **Q31 format Maximum Frequency** |
|---|---|---|
| Xilinx IP using on board DSP48E blocks | 205.04 MHz | 113.21MHz |
| | 1/ 32 blocks used | 4/32 blocks used |
| Urdhava multiplier using LUT's only. | 236.18 MHz | 158.90 MHz |
| Speed factor Improvement | 1.15 times | 1.40 times |

*Table 3: Comparison Of Q-Format Multipliers Using Virtex 5 Dsp48e Blocks*

**7.Conclusion**

This paper proposed a fast multiplier architecture for signed Q-format multiplications using Urdhava Tiryakbhyam method of Vedic mathematics. Since Q-format representation is widely used in Digital Signal Processors the proposed multiplier can substantially speed up the multiplication operation which is the basic hardware block. They occupy less area and are faster than the booth multipliers. It is also shown that the Urdhava Tiryakbhyam Q-format multiplier is faster than Xilinx parallel multiplier Intellectual Property (IP) using slice LUT's and also using DSP48E blocks which are meant specifically for digital signal processing operations like multiply-accumulate, multiply-add etc. Therefore the Urdhava Tiryakbhyam Q-format multiplier is best suited for signal processing applications requiring faster multiplications. Future work lies in the direction of introducing pipeline stages in the multiplier architecture for maximizing throughput.

**8.Reference**


1. Jagadguru Swami Sri Bharati Krisna Tirthaji Maharaja, "Vedic Mathematics: Sixteen Simple Mathematical Formulae from the Veda," Motilal Banarasidas Publishers, Delhi, 2009, pp. 5-45.
2. H. Thapliyal and M. B. Shrinivas and H. Arbania, "Design and Analysis of a VLSI Based High Performance Low Power Parallel Square Architecture," Int. Conf. Algo.Math.Comp. Sc., Las Vegas, June 2005, pp. 72-76.
3. Himanshu Thapliyal and M. B. Srinivas,"An efficient method of elliptic curve encryption using Ancient Indian Vedic Mathematics,"48th IEEE International Midwest Symposium on Circuits and Systems, 2005, vol.1, pp. 826-828.
4. M. Pradhan and R. Panda, "Design and Implementation of Vedic Multiplier," A.M.S.E Journal, Computer Science and Statistics, France vol. 15, July 2010, pp. 1-19.
5. Harpreet Singh Dhillon , Abhijit Mitra, "A Reduced-Bit Multiplication Algorithm for Digital Arithmetics," International Journal of Computational and Mathematical Sciences, Spring 2008, pp.64-69.
6. Sen-Maw Kuo and Woon-Seng Gan, "Digital Signal Processor, architectures ,implementations and applications," Pearson Prentice Hall, 2005, pp. 253-323.
7. S. S. Kerur, Prakash Narchi, Jayashree C .N., Harish M.Kittur and GirishV.A."Implementation of Vedic Multiplier for Digital Signal Processing," International Journal of Computer Applications, 2011, vol. 16, pp. 1-5.
8. A.D. Booth, "A Signed Binary Multiplication Technique", Qrt. J. Mech. App. Math., vol. 4, 1951, pp. 236–240.