International Journal of Engineering & Technology, 7 (2.7) (2018) 626-630

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research Paper

Implementation of Fast Convolution using Robust Vedic Multiplier of Radix-2

Damarla Paradhasaradhi, Bharinala Haridhar, A.V. Sreekanth Reddy, Dudipalli Sri Charan, Atyam Lekhaz

Department of Electronics and Communication Engineering,
Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur, Andhra Pradesh, India – 522502.
**Email: dparadhasaradhi@kluniversity.in*

Abstract

Convolution is an algorithm which is mainly used in video, audio and image processing. Convolution calculation is simple in steps however it consumes a lot of memory as well as power in the computational process. It is a mathematical algorithm which is also used in the applications like filtering, edge detection, de-noising, compression etc., as it can be exploit computational power. In this paper, we implemented the speed of discrete linear convolution using robust Vedic multiplier which is one of the fastest multipliers with two finite-length sequences. By implementing convolution with Vedic multiplier power, area and delay are reduced. This implementation process can be realized by simplifying the convolution building block.

Keywords: *Convolution, De-Noising, Edge Detection, Full Adder, Robust Vedic Multiplier.*

1. Introduction

The importance of convolution implementation on FPGA (Field Programmable Gate Array) is to reduce the design area of an FPGA [6] that includes reduction of processing time and computational power. Mathematically, combining two signals to generate a third signal is called 'Convolution' [5]. For the processing of images, audio and video signals of complex mathematical computations are required, that consumes a lot of memory and increase the design complexity [1]. This limitation can be overcome with the help of linear convolution where it can solve the complex computations in relatively simple mathematical steps [7]. It is used in numerical analysis and linear algebra to build the filters with the finite impulse response in Digital signal processing. The linear convolution can be realized using the sampling technique for the continuous signals with respect to time and a unit step for a discrete time signal[5], where it is decomposed into an infinite impulse thereby applying mathematical steps. And it is also applied as a filter to signals, to implement functions to reduce the noise. To increase the speed of convolution we convert the linear convolution to digital signals [4], it uses the Boolean algebra properties to implement the computations. The kernel operation convolution is the multiplication of multi-digit number, which can be efficiently performed with transfer functions.

2. Literature survey of convolution

Convolution process is more important because it contains three signals of concern, output, input and impulse response. As compared to different decomposition methods, the two methods namely, the impulse decomposition and The Fourier decomposition were the backbones of the signal processing [4]. When impulse response is decomposed, then the procedure can be mathematically defined by the operation called convolution [5]. The impulse response carries synonyms in different applications. If impulse response is used in the design of filter then the filter is called as the filter kernel, the convolution kernel, or simply the kernel. If impulse response used in image processing then it is known as the point spread function**.** Convolution is a mathematical operation, just as multiplication, integration and addition [3]. It is used in many fields of mathematics, such as probability and stochastics process etc.

3. Linear convolution sum method

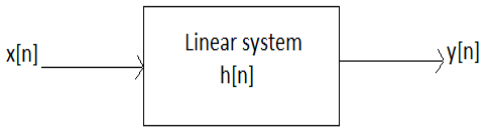


Fig. 1: Linear System

i. The study of LSI (linear shift invert Systems) method is very useful because to obtain the sum of elementary signal we decompose input signal.
ii. After the elementary input signals are taken into consideration then it is given to the system individually. Now apply linearity property to the decomposed input signal and obtain the output response and it is simply added and this should afford us the total response of the system to any other given input signal [1].
iii. Convolution involves operations like shifting, folding, summation and multiplication.

Copyright © 2018 Authors. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

iv. If there are N number of samples in h(n) and M number of samples in x(n) then the maximum number of output samples in y(n) is equal to M+N-1[3].

3.1. Simple multiplication form

To perform Simple multiplication between input signal X(n) and impulse response H(n) therefore the convoluted output is stored in Y(n).

$$X(n) = \{x1, x2, x3, x4\}$$
$$H(n) = \{h1, h2, h3, h4\}$$
$$Y(n) = X(n)*H(n)$$

4. Vedic Multiplier

Based on multiplier performance the speed of a processor depends, so there is a great demand for high-speed multipliers. While seeing for high-speed multipliers we have to keep in mind that it must consume less area and low power [2]. There are few popular multipliers in modern VLSI design like Booth's and Array multiplier algorithms, but they are having their scope of boundaries[8][9]. In these multipliers, multiplication process is having several intermediate operations like additions, subtractions and comparisons. Which reduces speed of the multiplier. Hence for implementation of robust and fast multipliers, booths based multipliers are not the proper choice [8].

Vedic mathematics is an earliest mathematical method which can solve several mathematical challenges in present days. This Vedic mathematics already exists and used effectively in ancient India and it is again discovered by a well-known mathematician named Sri Bharati Krishna Tirthaji [2]. In Vedic mathematics, the partial products calculation required for multiplication can be obtained in advance compared to the formal method of multiplication. Thus these partial products obtained after multiplication can be added by using Vedic mathematics algorithm to get the final result.

The term 'Urdhwa Tiryakbhyam' is a Sanskrit word that means "Vertically and crosswise" respectively. The main reason for using this algorithm other than existing multiplication technique is that Urdhwa Tiryakbhyam sutra[7] utilizes Full adders, Half adders and logical AND only to do complete multiplication operation. Thus, it saves a lot of processing time [10].

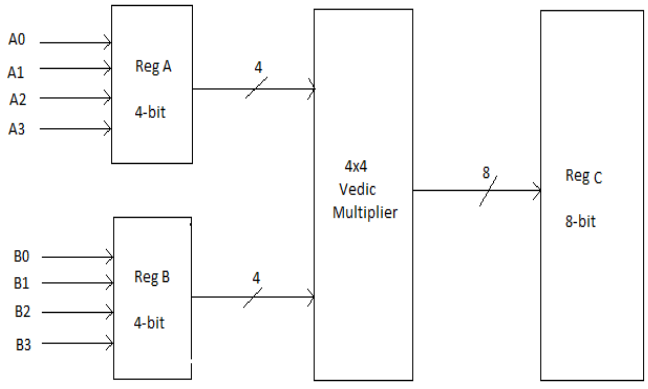


Fig.2: Block Diagram

Convolution algorithm is performed between the two 4-bit binary data that are stored in RegA and RegB respectively, thereby performing 4x4 Vedic Multiplication.The resultant was an 8-bit convoluted output in RegC.

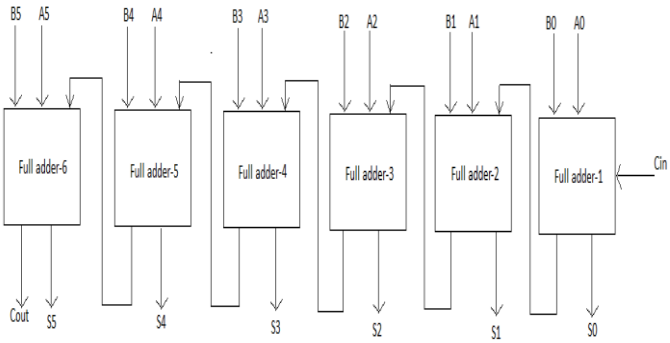


Fig. 3: 6-bit Ripple Carry Adder

The 6-bit full adder is designed by the principle of ripple carry adder logic. It consists 6-full adder which were arranged in serially and the 6-bit data given to the adders, initially first full adder is two input data, the third one is carry-in it is set to zero. In a similar way, two-bit data is given to second full adder but carry in is a carry out of first full adder. In remaining full adders to do the same process, but carrier signal is propagated to one another. Finally, the carry out occurs it can be ignored.

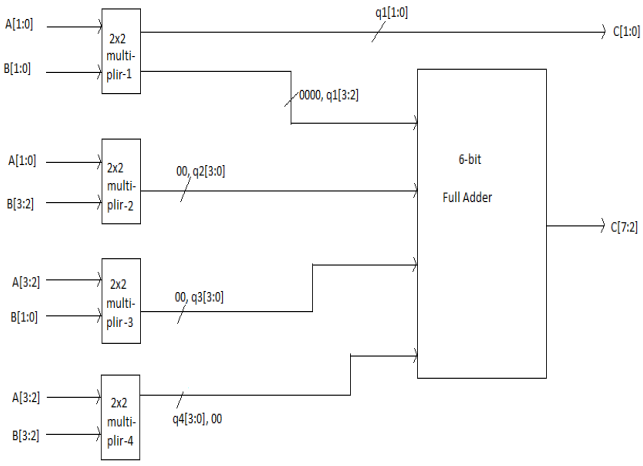


Fig. 4: 4X4 Vedic Multiplier

5. Implementation methodology of Urdhwa Tiryakbhyam

i. RegA and RegB are two 4-bit data register 1001 and 1101 respectively.
ii. Upon multiplying LSB bit of RegA with LSB and MSB bits of RegB gives 0001 and 0011 respectively.
iii. On repetition of step-ii i.e. multiplying MSB bit of RegA with RegB gives 0010 and 0110 respectively.
iv. After getting result of all four multipliers, from the result of the first multiplier, the lower order two bits are directly given to the lower order of the output register RegC and remaining 2-bits are append with four zeros is given to the 6-bit full adder.
v. The remaining three multiplier results are append with two zeros and is directly given to 6-bit full adder.
vi. By adding four 6-bit data we get a 6-bit result and that is stored as MSB of output in register RegC.
vii. Finally, we are having 8-bit convoluted output as 01110101.

6. Results and Discussion

Regarding performance comparison we considered three mostly used multipliers in VLSI they are Vedic, Booth and Array multipliers. The number of 4 input LUTs utilized by Vedic multiplier is less when compared to Booth and Array multipliers. A number of occupied Slices of the Vedic multiplier are 15 whereas Booth and Array multiplier are 39 and 145 respectively. Delay performance of Vedic multiplier is 13ns while coming to Booth multiplier it is 18ns and Array multiplier is 45ns which shows that Vedic multiplier will give the result faster than remaining two multipliers. Finally, the power consumption of Vedic multiplier is very less compared to Booth and Array Multiplier. By considering all these specifications number of LUTs, a number of slices, delay and power we can say that Vedic multiplier is using less power, LUTs, slices and delay than remaining two Booth and Array multipliers. Thus, the Vedic multiplier is faster than Booth and Array Multiplier.

Table 1: Comparison of Different Multipliers

Parameters	Vedic multiplier	Booth multiplier	Array multiplier
No.of 4 input LUTs	27	70	226
No.of Occupied Slices	15	39	145
Delay (Nano sec)	13	18	45
Power (uW)	0.144	0.57	1.035

Experimentally the implementation of the fast convolution is done by the robust Vedic multiplier, because of its less area, processing delay is less and power consumption is also less, by considering all these we choose this Vedic multiplier. To observe the convolution results as shown in the below Table 2. For example, consider the data RegA is 0011 and RegB is 0111 so finally, the convoluted output of RegA and RegB is 00010101 this result is stored in output register RegC.Similarly, RegA is 1111 and RegB is 1111 so finally, the convoluted output of RegA and RegB is 11100001 this result is stored in output register RegC. The maximum length of output data is 8-bits.

Table 2: Simulation Results

RegA	RegB	RegC
1111	1111	11100001
1000	0111	00111000
0011	0111	00010101
1001	101	01110101

The simulated output waveforms can be observed in figures Fig. 5 and Fig. 6. Also, we saw the number of 4-bit LUTs are 27, the number of occupied slices are 15 and the overall delay of the multiplier is 13.54nsec.

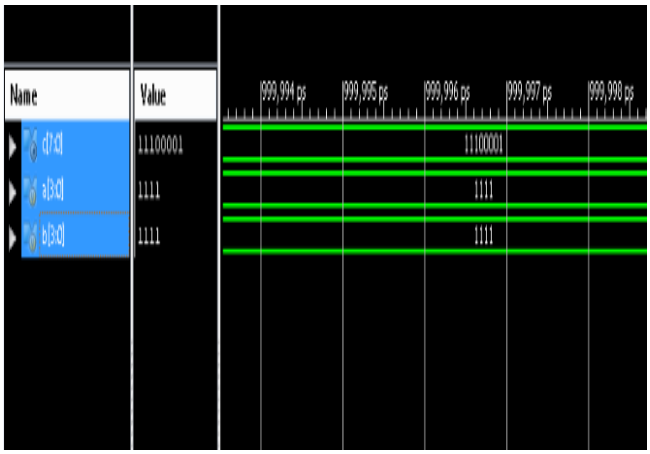


Fig. 5: Simulation 1

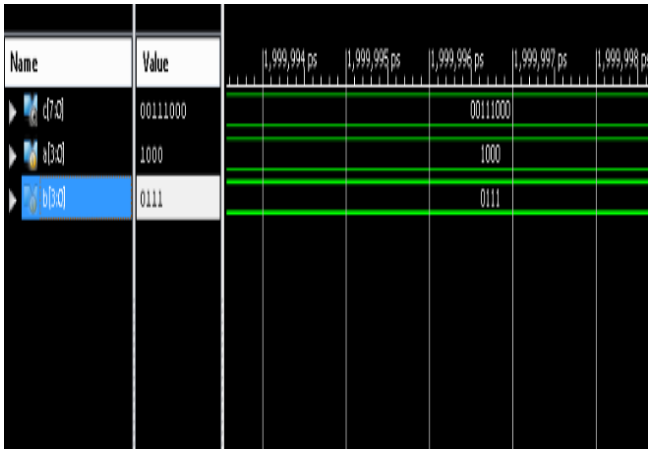


Fig. 6: Simulation 2

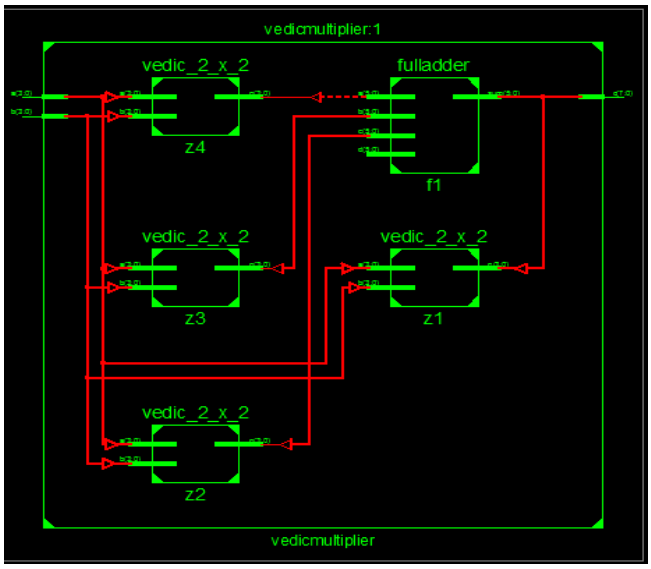


Fig. 7: RTL Schematic of Vedic multiplier

The RTL schematic of block diagram is shown in the above Fig. 7. The number of inputs, outputs and building blocks were verified. The number of inputs of each block and the internal building blocks are shown in above Fig.7. For building of 4x4 vedic multiplier, it utilizes two input lut's are 3, three input lut's is 1 and four input luts are 23, the number of io buffers are 16.

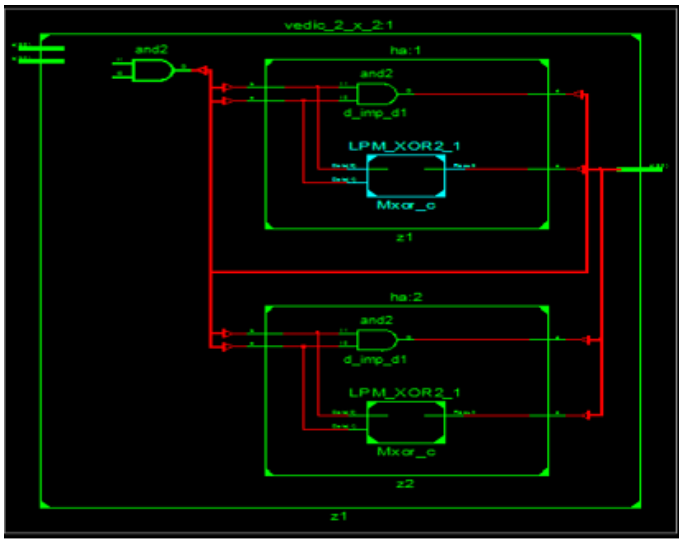


Fig. 8: RTL Schematic of 2x2 multiplier

The RTL schematic of 4X4 Vedic multiplier contains two major blocks one is a 2X2 multiplier and the second one is 6-bit full adder. The design of 2X2 multiplier is shown in above Fig.8, it has two half adders which consist of two input XOR gates and two input AND gates. Thus, the complexity of 2X2 multiplier is reduced by using these logic gates. The overall delay of 2X2 multiplier is 0.482nsec.

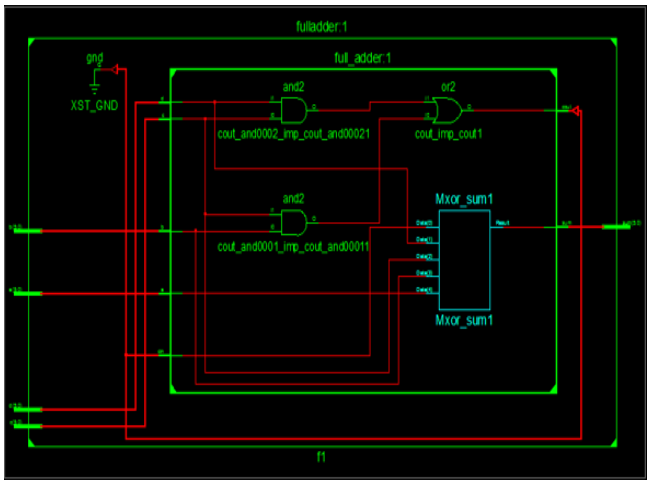


Fig. 9: RTL Schematic of 6-bit Full adder

The second building block of Vedic multiplier is 6-bit full adder as shown in above Fig.9. To reduce the design complexity of the circuit, ripple carry adder is used, which is simple among the remaining adders. This is designed with the help of XOR, OR and AND gates. These are basic logic gates which reduces the area and delay. The overall delay of 6-bit full adder is 0.451nsec.

Upon completion of all building blocks, to observe the overall performance of the design. We consider total area, slices and number of bonded IOBs. These parameters are shown in below table.

Table 3: Synthesis Report

Logic Utilization	Used	Available	Utilization
No.of 4 input LUTs	27	9312	1%
No.of Occupied slices	15	4656	1%
No.of slices co-ntaining only relat-ed logic	15	15	100%
No.of slices co ntaining unrelated logic	0	15	0%
Total No.of 4 input LUTs	27	9312	1%
No.of bonded IOBs	16	232	6%
Average fan-out of non-clock nets	3.36		

Finally, the delay of each and every individual blocks i.e. gate and net delays is shown in the below Fig.10. Finally, the total delay is 13.547ns, from this 69.8% is utilized for logic and remaining 30.2% is utilized for route. These results can be observed in the above Table 3.

```
Timing constraint: Default path analysis
  Total number of paths / destination ports: 222 / 8
--------------------------------------------------------------------------------
Delay:               13.547ns (Levels of Logic = 9)
  Source:            a<0> (PAD)
  Destination:       c<6> (PAD)

  Data Path: a<0> to c<6>
                                Gate     Net
    Cell:in->out      fanout   Delay    Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O             11   1.218   0.968  a_0_IBUF (a_0_IBUF)
     LUT4:I2->O             2   0.704   0.482  z1/z2/Mxor_c_Result1 (q0<2>)
     LUT3:I2->O             2   0.704   0.451  f1/u0/cout11 (N111)
     LUT4:I3->O             1   0.704   0.424  f1/u1/cout_SW0 (N25)
     LUT4:I3->O             2   0.704   0.451  f1/u1/cout (f1/cin<1>)
     LUT4:I3->O             2   0.704   0.451  f1/u2/cout1 (f1/cin<2>)
     LUT4:I3->O             2   0.704   0.447  f1/u4/Mxor_sum_xo<0>1 (N01)
     MUXF5:S->O             1   0.739   0.420  f1/u4/Mxor_sum_xo<0>_f5 (c_6_OBUF)
     OBUF:I->O                  3.272          c_6_OBUF (c<6>)
    ----------------------------------------
    Total                      13.547ns (9.453ns logic, 4.094ns route)
```

Fig. 10: Timing Analysis

7. Conclusion

In this paper, the linear convolution between two signals was implemented using 'Robust Vedic Multiplier' where the computational delay drastically reduced. This work can also be extended further using fastest hybrid adders in place of 'Ripple Carry Adder'.

References

[1] Pankaj Katkar, Sridhar T N, Sharath G M, Sivanantham S* and Sivasankaran K "VLSI Implementation of Fast Convolution" 2015 Online International Conference on Green Engineering and Technologies (IC-GET 2015).

[2] Sushma R. Huddar and Sudhir Rao, Kalpana M, Surabhi Mohan"Novel High speed Vedic Mathematics Multiplier using Compressors", 978-1-4673-5090-7/13/2013 IEEE.
[3] Yuke Wang, Keshab Parhi "Explicit Cook-Toom Algorithm for Linear' Convolution" 0-7803-6293-4/00/ 2000 IEEE.
[4] Deyun Wei, Student Member, IEEE, Qiwen Ran, Yuanmin Li, Jing Ma, and Liying Tan "A Convolution and Product Theorem for the Linear Canonical Transform" IEEE SIGNAL PROCESSING LETTERS, VOL. 16, NO. 10, OCTOBER 2009.
[5] Zdenka Babic', Danilo P. Mandic2" A Fast Algorithm for Linear Convolution of Discrete Time Signals" 19-21 September 2001, Ni5, Yugoslavia.
[6] Khader Mohammad, Sos Agaian" Efficient FPGA implementation of convolution" Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics San Antonio, TX, USA - October 2009.
[7] Karishma P Dighorikar, S. L. Haridas" Area Efficient Architecture for Convolution Using Vedic Mathematics" International Journal of Science and Research (IJSR), ISSN (Online): 2319-7064.
[8] Jyoti Prakash Mohanty, Ritisnigdha Das, Siba Kumar Panda "Design and Simulation of Convolution using Booth encoder Wallace Tree Multiplier" IOSR Journal of Electronics and Communication Engineering (IOSR-JECE) e-ISSN: 2278-2834,p- ISSN: 2278-8735. PP 42-46.
[9] Kaustubh Manikrao Gaikwad, Mahesh Shrikanth Chavan "Analysis of Array Multiplier and Vedic Multiplier using Xilinx" Communications on Applied Electronics (CAE) – ISSN : 2394-4714 Foundation of Computer Science FCS, New York, USA Volume 5 – No.1, May 2016.
[10] M Siva Kumar, Sanath Kumar Tulasi, N Srinivasulu, Vijaya Lakshmi Bandi, K Hari Kishore "Bit wise and Delay of vedic multiplier" International Journal of Engineering & Technology,pp 26-30.
[11] Lee Shing Jie1, a) and Siti Hawa Ruslan1, b)" A 2x2 Bit Vedic Multiplier with Different Adders in 90nm CMOS Technology" Published by AIP Publishing. 978-0-7354-1563-8.
[12] Damarla Paradhasaradhi, M. Prashanthi and N Vivek, "A modified Wallace tree multiplier using efficient square root carry select adder", IEEE International conference in green computing, communication and electrical engineering, March 2014.
[13] Khushbu Maheshwari, Prof. Mukesh Tiwari" Design and Implementation of 4-bit Array Multiplier for Low Power in 45nm CMOS Technology" International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181 Vol. 3 Issue 4, April – 2014.