International Journal of Engineering Innovation & Research
Volume 6, Issue 2, ISSN: 2277 – 5668

Improved Computational Time for Circular / Linear Convolution using FFT by Matrix Multiplication

Priyanka Bhalawi[1*] and Ashish Raghuwanshi[2]
[2]Asst. Prof. IES College of Technology, Bhopal, M.P., India
*Corresponding author email id: priyankabhalawi41@gmail.com

Date of publication (dd/mm/yyyy): 08/03/2017

Abstract **– Matrix multiplication is a primary computation for several scientific computing, graphics processing units and engineering applications. Matrix multiplication is also use in convolution operation of two discrete signals in DFT (Discrete Fourier Transform) and FFT (Fast Fourier Transform) application. In this work, we considered two different examples (circular convolution and linear convolution) of matrix multiplier architecture where speed is the main constraint.**

The simulation shows the accuracy of architecture with low computational time and limited number of gate count. The computational time require for byte multiplication is 10ns i.e. 0.01µs. Thus for 7X7 matrix it is 0.07us. In 0.07us total 7 words are process. This will process 100 words in 1usec and 100 MW/sec is the throughput of this matrix size. For 64 point i.e. for 63X63 matrix computational time is 0.63 µs. For 128 point i.e. for 127X127 matrix computational time is 1.27us µs. For 256 point i.e. for 255X255 matrix computational time is 2.55 µs. For 512 point i.e. for 511X511 matrix computational time is 5.11µs.

In this work, we considered two different examples (circular convolution and linear convolution) of matrix multiplier architecture where speed is the main constraint.

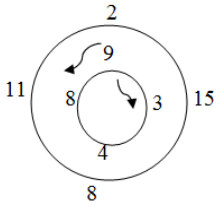
Keywords **– DFT, FFT, VHDL, MAC, PE.**

I. INTRODUCTION

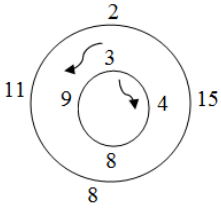
The computation time and accuracy of multiplication in convolution operation is crucial for the performance digital signal processing operations. Its computational complexity is large due to the time requires for multiplication, shifting, delay and addition operations. The performance of convolution multiplication process can be improved by the design of parallel architecture. The internal architecture of processing elements (PE) includes addresable register, a mutiplier unit by byte multiplication, an adder for mutilplier output, FIFO block to add and shift the multiplier output, and a controlling unit for controlling the operations within the specific PE. The counters use for the matrix formation to set the signal samples in row wise, column wise and in matrix wise. PEs also design using a pipeline registers, simple logic gates, and other miscellaneous blocks. The addressable data which is to be processing components of the transformed sequences for the multiplier and multiplicand matrices are send to the processing element.

II. CIRCULAR CONVOLUTION

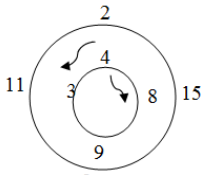
In circular convolution two signal which is to be convolve is represented in anticlockwise direction on a circle. N samples of these signals are place on the circumference of the two concentric outer circle. The signal which is to be folded is place its sample in clockwise direction on a inner circle. Multiply corresponding samples on inner and outer circle and add them to get output sample. To shift the shifted signal, rotate the inner circle in clockwise direction and multiply corresponding samples on inner and outer circle and add them to get output sample.



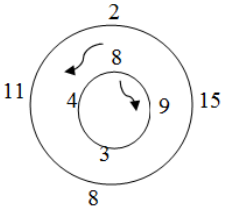
Y(m) = 2X9+11X8+8X4+15X4 = 183



Y(m) = 2X3+11X9+8X8+15X4 = 229



Y(m) = 2X4+11X3+8X9+15X8 = 235



Y(m) = 2X8+11X4+8X3+15X9 = 219

III. LINEAR CONVOLUTION

In linear convolution can be performed graphically by reflecting and shifting one of the signal which is to be convolve. In this case when first signal is of length 4, and second signal is of length 2, the linear convolution results samples is of length 4 + 2 - 1 = 5.

	2	11	8	15
9	18	99	72	135
3	6	33	24	45
4	8	44	32	60
8	16	88	64	120

Y(m) = 18, 6+99, 8+33+72, 16+44+24+135, 88+32+45, 64+60, 120
i.e. Y(m) = 18, 105, 113, 219, 165, 124, 120

Matrix Method for Convolution

Due to the importance of Discrete Fourier Transform (DFT) in signal processing application, it is critical to have an efficient method to compute this algorithm. DFT operates on a N -point sequence of numbers, referred to as x(n) . The value x(n) is presented in time domain data and usually can be taught as a uniformly sampled version of a finite period of a continuous function f (x) . The DFT of x(n) sequence is transformed to X(k) in frequency domain representation employing by using Discrete Fourier Transform. The functions x(n) and X(k) is generally represented in complex signal form, given by

$$\sum_{n=0}^{N-1} x(n)\, e - \frac{j2\pi kn}{N} = X(K)$$

where x(n) is the input time domain representation and N is the number of input to the DFT. The value n represents the discrete time-domain index and k is the normalized frequency domain index. The description of efficient computation is discussed on DFT methods since the IDFT and DFT consumes the same type of computational algorithm. From the computation of each value of k , it is observed that direct computation of X(k) involves N complex multiplications (4N real multiplications) and N −1 complex additions (4N − 2 real additions). Eventually, to compute all N values of the DFT requires N^2 complex multiplications and N^2 − N complex additions.

The multiplication of two discrete time signal in discrete fourier transform is equivalent to the circular convolution of there sequences in time domain. For x(n) and h(n) signal convolution is express as:

$$\sum_{n=0}^{N-1} x(n)h((m-n))N = y(m)$$

Here the term $h(m-n)_N$ indicates the circular convolution.

The convolution in time domain of two signal x and h is perform by multiplying its discrete fourier transform and the converting it in time domain by inverse discrete fourier transform. The equation of DFT is the summation of discrete signal multiplied by twiddle factor given as:
$X(K)= \sum_{n=0}^{N-1} x(n)\, e^{-j2\pi kn/N}$

Where, $e^{-j2\pi kn/N}$ is called as twiddle factor.

For long convolution the FFT is faster method as compare to DFT.

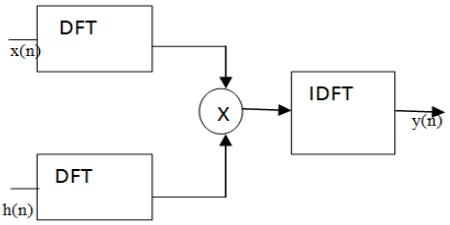


Fig. 1. Convolution by DFT-IDFT Method.

The same convolution process is done by matrix method as:

$$y(m) = \begin{bmatrix} 1 & 1 & 0 & 2 \\ 2 & 1 & 1 & 0 \\ 0 & 2 & 1 & 1 \\ 1 & 0 & 2 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 2 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 6 \\ 7 \\ 6 \\ 5 \end{bmatrix}$$

The NxM matrix multiplication equation is written as:
y(0) = h(0)x(0) + h(N-1)x(1) + h(N-2)x(2) + ---------- h(2)x(N-2) + h(1)x(N-1)
y(1) = h(1)x(0) + h(0)x(1) + h(N-1)x(2) + ---------- h(2)x(N-2) + h(1)x(N-1)
y(2) = h(2)x(0) + h(1)x(1) + h(0)x(2) + ----------h(4)x(N-2) + h(3)x(N-1)

y(N-2) = h(N-2)x(0) + h(N-3)x(1) + h(N-4)x(2) + ---------- --h(0)x(N-2) + h(N-1)x(N-1)
y(N-1) = h(N-1)x(0) + h(N-2)x(1) + h(N-3)x(2) + ---------- --h(1)x(N-2) + h(0)x(N-1)

Thus circular convolution is obtaining quickly using matrix multiplication approach.

Design Technique

The matrix multiplication process through distributed memory approach and shared memory approach is use in many related work. This work is based on the architecture which consists of identical processing elements (PEs). The number of PE requires in design is depends on the size of the matrices. Each PE performs the necessary multiply accumulate (MAC) operation. Each PE operates independently with connection only to the input and output ports. This greatly helps in reducing the interconnection between the PEs and as a result the hardware resource utilization is minimized. The distributed memory processing techniques are used to improve the performance of the matrix multiplier. The proposed design is implemented using the VHDL hardware description language. The implementation supports a range of parameters to facilitate the experimental evaluation of design choices. A simulation test bench is used to verify the correctness of the implementation, by checking the produced results.

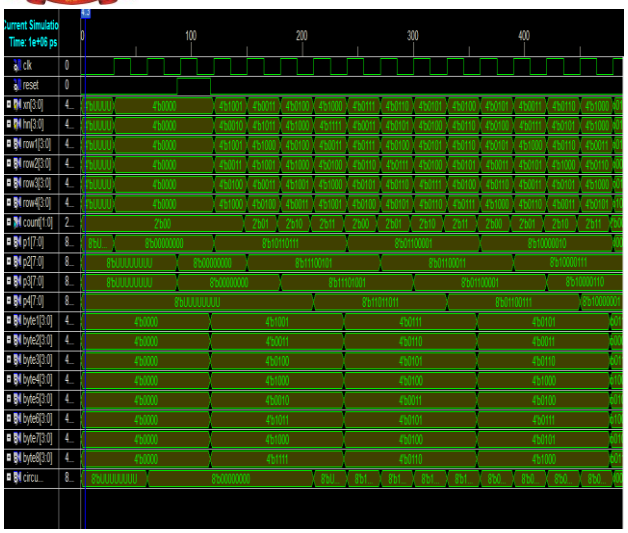


Fig. Circular Convolution of x(n) ={0111, 0110, 0101, 1101} and h(n)={0011, 1100, 1100, 0110}

$$\begin{bmatrix} 1001 & 1000 & 0100 & 0011 \\ 0011 & 1001 & 1000 & 0100 \\ 0100 & 0011 & 1001 & 1000 \\ 1000 & 0100 & 0011 & 1001 \end{bmatrix} \begin{bmatrix} 0010 \\ 1011 \\ 1000 \\ 1111 \end{bmatrix} = \begin{bmatrix} 10110111 \\ 11100101 \\ 11101001 \\ 11011011 \end{bmatrix}$$

Fig. shows the timing simulation of circular convolution operation on timing scale of 0ns to 500 ns. F or example the two signal x(n) = {9,3,4,8} is convolve with the signal h(n)={2,11,8,15} is shown in matrix method format. The matrix of circular convolution is arrange in row1, row2, row3, row4shown in timing simulation. This matrix is process at every four clock triggered. The result of this matrix will generate the output Y(n)= {183, 229, 233, 219}.

Synthesis Report

# Multipliers	: 16
4x4-bit multiplier	: 16
# Adders/Subtractors	: 16
8-bit adder	: 12
9-bit adder	: 3
9-bit addsub	: 1
# Counters	: 1
9-bit up counter	: 1
# Registers	: 66
Flip-Flops	: 66
# Multiplexers	: 3
4-bit 4-to-1 multiplexer	: 2
8-bit 4-to-1 multiplexer	: 1

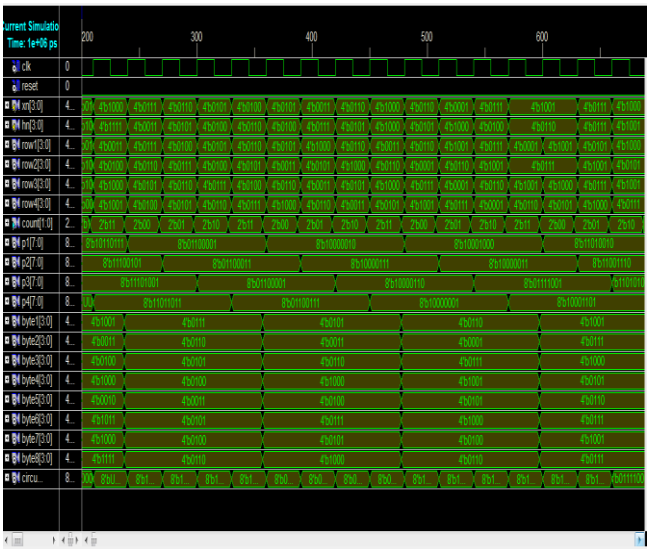


Fig 6.8. Circular Convolution of x(n) ={0101, 0011, 0110, 1011} and h(n)={0100, 1101, 1011, 1100}

Fig. shows the timing simulation of circular convolution operation on timing scale of 0ns to 500 ns. For example the two signal x(n) = {5,3,6,11} is convolve with the signal h(n)={4,13,11,12} is shown in matrix method format. The matrix of circular convolution is arrange in row1, row2, row3, row4shown in timing simulation. This matrix is process at every four clock triggered.

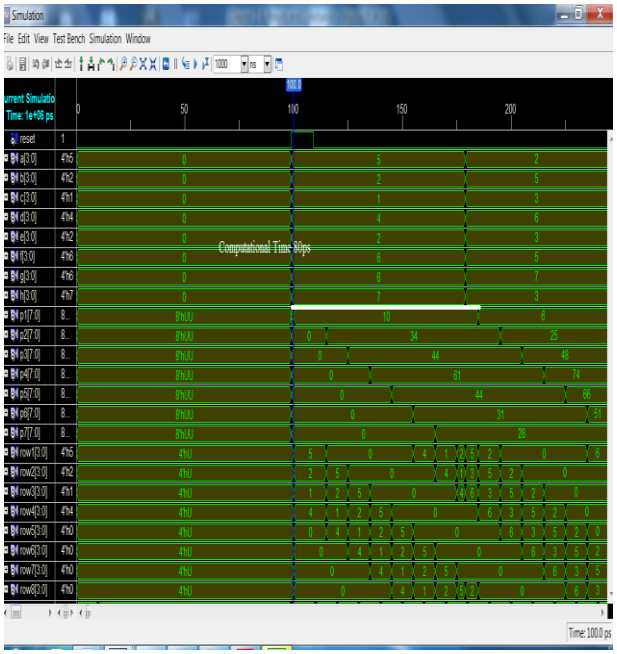


Fig. 6.9. Linear Convolution of x(n) ={5,2,1,4} and h(n)={2,6,6,7}= {10,34,44,61,44,31,28}

Fig. shows the timing simulation of linear convolution. The number of sample in x(n) is denoted by L and number of samples in h(n) are denoted by N then the number of convolve samples is M=N+L-1. But the number of samples generated in circular convolution is N=L=M. To generate the M number of samples in linear convolution using matrix method the zeros are padded in lower rows of matrix and

the same process of circular convolution generates the result which is same as the result of linear convolution.

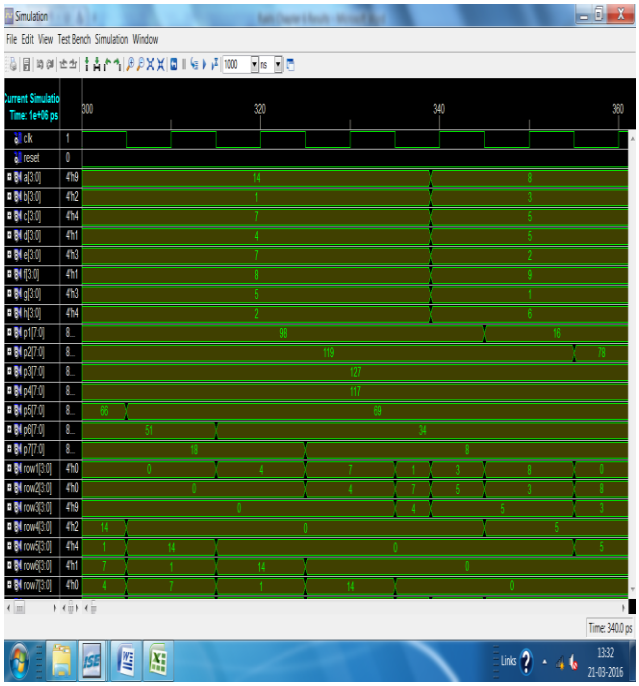


Fig. 6.10. Linear Convolution of x(n) ={14,1,7,4} and h(n)={7,8,5,2}

$$\begin{bmatrix} 14 & 0 & 0 & 0 & 4 & 7 & 1 \\ 1 & 14 & 0 & 0 & 0 & 4 & 7 \\ 7 & 1 & 14 & 0 & 0 & 0 & 4 \\ 4 & 7 & 1 & 14 & 0 & 0 & 0 \\ 0 & 4 & 7 & 1 & 14 & & 0 \\ 0 & 0 & 4 & 7 & 1 & 14 & 0 \\ 0 & 0 & 0 & 4 & 7 & 1 & 14 \end{bmatrix} \begin{bmatrix} 7 \\ 8 \\ 5 \\ 2 \\ 0 \\ 0 \\ 0 \end{bmatrix} .= \begin{bmatrix} 98 \\ 119 \\ 127 \\ 117 \\ 69 \\ 34 \\ 8 \end{bmatrix}$$

Fig. shows the linear convolution. For example the convolution of signal x(n)= {14,1,7,4} with signal h(n)={7,8,5,2} generates the linear convolution result of {98,119,127,117,69,34,8}.The zeros are padded in the last three rows of first column and the bytes are shifted and rooted in next columns of matrix. The timing simulation are shown on the timing scale of 300ns to 360ns scale.

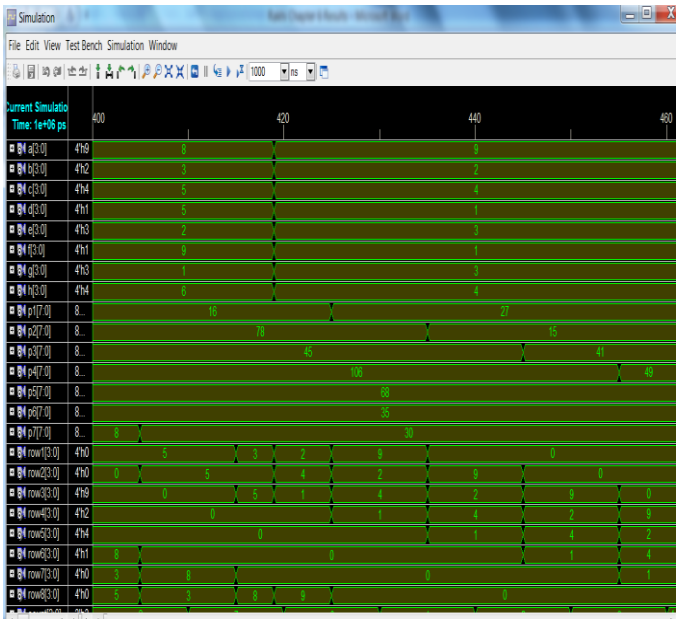


Fig. 6.11. Linear Convolution of x(n) ={9,2,4,1} and h(n)={3,1,3,4}

$$\begin{bmatrix} 9 & 0 & 0 & 0 & 1 & 4 & 2 \\ 2 & 9 & 0 & 0 & 0 & 1 & 4 \\ 4 & 2 & 9 & 0 & 0 & 0 & 1 \\ 1 & 4 & 2 & 9 & 0 & 0 & 0 \\ 0 & 1 & 4 & 2 & 9 & & 0 \\ 0 & 0 & 1 & 4 & 2 & 9 & 0 \\ 0 & 0 & 0 & 1 & 4 & 2 & 9 \end{bmatrix} \begin{bmatrix} 3 \\ 1 \\ 3 \\ 4 \\ 0 \\ 0 \\ 0 \end{bmatrix} .= \begin{bmatrix} 27 \\ 15 \\ 41 \\ 49 \\ 21 \\ 19 \\ 4 \end{bmatrix}$$

Fig. shows the linear convolution. For example the convolution of signal x(n)= {9,2,4,1} with signal h(n)={3,1,3,4} generates the linear convolution result of {27,15,41,49,21,19,4}.The zeros are padded in the last three rows of first column and the bytes are shifted and rooted in next columns of matrix. The timing simulation are shown on the timing scale of 400ns to 460ns scale.

Table 1: Analysis for matrix to matrix multiplication.

N Point	Computational Time (µs)	Throughput (MW/s)
8	0.008	454.545
64	0.064	492.61
128	0.128	497.512
256	0.256	498.007
512	0.512	499.001

CONCLUSION

In this work, we considered two different examples (circular convolution and linear convolution) of matrix multiplier architecture where speed is the main constraint. The architecture of matrix multiplication operate concurrently, and then the additions performed simultaneously. This parallelism can improved the computational time. Our designs minimize the number gate count require for multiplier, adder, FIFO, counter and control logic modules. It improvements in latency, computational-time, throughput for performing matrix multiplication. The simulation shows the accuracy of architecture with low computational time and limited

number of gate count. The computational time require for byte multiplication is 10ns i.e. 0.01µs. For the process of linear convolution the computational time is computed to 0.008usec. In this time the system process 7 bytes and in 0.016us system process 7 words. In 0.016usec it process 7 words and in 0.0022 sec it process1Mwords. Thus the throughput obtain in one second is 454.545MW/s. Notice that changes in N have a little effect on the throughput, which is due to the highly parallely distributed memory approach in proposed architecture.

REFERENCES

[1] Soydan Redif, and Server Kasap "Novel Reconfigurable Hardware Architecture for Polynomial Matrix Multiplications" IEEE Transactions On Very Large Scale Integration (Vlsi) Systems" March 10, 2014.

[2] Tai-Chi Lee, Mark White, and Michael Gubody "Matrix Multiplication on FPGA-Based Platform" Proceedings of the World Congress on Engineering and Computer Science 2013 Vol I.

[3] Bahram Hamraz, Nicholas HM Caldwell, and P. John Clarkson "A Matrix-Calculation-Based Algorithm for Numerical Change Propagation Analysis" IEEE Transactions On Engineering Management, Vol. 60, No. 1, February 2013 pp. no. 186.

[4] Nan Zhang "A Novel Parallel Scan for Multicore Processors and Its Application in Sparse Matrix-Vector Multiplication" IEEE Transactions On Parallel And Distributed Systems, Vol. 23, No. 3, March 2012 pp. no. 397.

[5] Mr. Rounak R. Gupta, 2Prof. Atul S. Joshi "Matrix Manipulation Using High Computing Field Programmable Gate Arrays" International Journal of Enterprise Computing and Business System Vol 2 issue 2 Jully 2012.

[6] Syed M. Qasim, Ahmed A. Telba and Abdulhameed Y. AlMazroo "Syed M. Qasim, Ahmed A. Telba and Abdulhameed Y. AlMazroo" IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.2, February 2010 pp no. 168.

[7] Syed M. Qasim, Ahmed A. Telba and Abdulhameed Y. AlMazroo "FPGA Design and Implementation of Matrix Multiplier Architectures for Image and Signal Processing Applications" IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.2, February 2010 pp. no. 168.