ISSN NO. 2320-5407

Journal homepage: http://www.journalijar.com

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH

RESEARCH ARTICLE

Time Efficient Implementations of Matrix Multiplication.

Rakhi Sharma[1], Soheb Munir[2].

1. Student, Department of ECE, Lakshmi Narain College Of Technology, Bhopal, India.
2. Assistant professor, Department of ECE, Lakshmi Narain College Of Tech, Bhopal, India.

Manuscript Info

Manuscript History:

Received: 18 January 2016
Final Accepted: 25 February 2016
Published Online: March 2016

Key words:
VHDL, (Processing Element) PF, (First in First out) FIFO, RAM

***Corresponding Author**

Rakhi Sharma.

Abstract

Matrix multiplication is use in various areas of engineering field such as digital signal and image processing, microprocessor and microcontroller base design etc. In digital signal processing circular convolution of two signals in discrete fourier transform is done through matrix method. The Filter design in DSP requires efficient multiplication and accumulation operations are perform using matrix method. In this work a processing element is design which includes the small sub modules of adder, counters, multipliers, matrix arrangement block, SRAM and FIFO etc. The work on paper is base on VHDL (Very High Speed Integrated Circuit Hardware Description Language) implementation of well-organized design of parallel matrix multiplication with reduce gate counts. The computational time, latency and throughput is improved. The results are simulated to demonstrate the accuracy and matrix size capacity of the architecture. The design is done on the basis of blocking and parallelization. Blocked matrix multiplication enables processing randomly large matrices using partial memory capability, and decrease the bandwidth requires across the device boundaries by recycle the existing elements. The propose algorithm verify low execution times while limiting the gate count.

Introduction:-

MATRIX multiplication has received interest. Because in multiplication process computational complexity, latency and throughput is reduces. In discrete Fourier transform the convolution operation is perform on discrete time signal using matrix multiplication method. Matrix multiplication is moreover utilize in convolution operation of two discrete signals in DFT (Discrete Fourier Transform) and FFT (Fast Fourier Transform) application. The speed and accuracy of matrix multiplication is essential for the performance of such applications. Its computational complexity is large than communication complexity. Because it takes large amount of multiplication and addition operations, so its performance can be improved significantly through parallel architecture design.
Convolution operation takes large amount of multiplication and addition operations, so its performance can be improved significantly through parallel architecture design.

Matrix Method for Convolution:-

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 $h(m\text{-}n)_N$ indicates the circular convolution.

For the convolution of two discrete signals matrix method is very convenient. In this method one discrete time signal sequence is repeated via circular shifting of signals. Matrix method convolution contain same number of sample as that of in x(n) and h(n) signals. It is represented as :

$$\begin{bmatrix} y(0) \\ y(1) \\ y(2) \\ \vdots \\ \vdots \\ \vdots \\ y(N-2) \\ y(N-1) \end{bmatrix} = \begin{bmatrix} h(0) & h(N-1) & h(N-2) & h(N-3) \text{----} h(2) & h(1) \\ h(1) & h(0)) & h(N-1) & h(N-2) \text{----} h(3) & h(2) \\ h(2) & h(1) & h(0) & h(N-1) \text{----} h(4) & h(3) \\ \vdots & \vdots & \vdots & & \\ \vdots & \vdots & \vdots & & \\ \vdots & \vdots & \vdots & & \\ h(N-2) & h(N-3) & h(N-4) & h(N-5) \text{----} h(0) & h(N-1) \\ h(N-1) & h(N-2) & h(N-3) & h(N-4) \text{----} h(1) & h(0) \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ \\ \\ \\ x(N-2) \\ x(N-1) \end{bmatrix}$$

The 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)

The convolution of two signals using matrix multiplication for x(n)= {2,1,2,1} and h(n)={1,2,3,4} is shown below

$$\begin{bmatrix} y(0) \\ Y(1) \\ y(2) \\ y(3) \end{bmatrix} \begin{bmatrix} 1 & 4 & 3 & 2 \\ 2 & 1 & 4 & 3 \\ 3 & 2 & 1 & 4 \\ 4 & 3 & 2 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 14 \\ 16 \\ 14 \\ 16 \end{bmatrix}$$

Thus circular convolution is obtain quickly using matrix multiplication approach.

Multiplication methodology:-

The processing steps implement the VHDL design constituting the proposed architecture which are sequentially executed in a parallel manner for sets of input matrices to achieve the maximum throughput. In this section the algorithm for matrix multiplication is discuss.

1. Initially data bytes of both matrix A and matrix B is obtain form RAM memory. This is done in row by placing one sequence by repeating via circular shifting of samples in row and column basis. The resulting sequence are stored in memory depends on addressing.
2. Each memory block contains dual input ports. The computation is done parallely. In next step these two signals are applied on processing element (PE) which includes a multiplier, adder, FIFO, row and column counter in such a way that the result accordingly stored into a second set of similar memory blocks location.
3. In processing element the row and column counters are use for NxN matrix formation and the FIFO is use for byte element to byte element multiplication of row of one matrix to the column of second matrix. The result of each byte multiplication is shifted toward a parallel adder through firs in first out register. The complete time require to generate the relutant byte by multiplication of all byte in a row of first matrix with all bytes in a column of second matrix is called as computational time.
4. All the require sub modules are design and interconnect to for the processing element of matrix multiplication.
5. Calculate the computational time, latency and throughput

Architecture of Matrix Multiplication Processing Element:-

For the matrix multiplication architecture consist of sub modules to design processing element (PE) are SRAM Memory blocks, Matrix arrangement of multiplier units, FIFO (First In First out), Adders and counters.

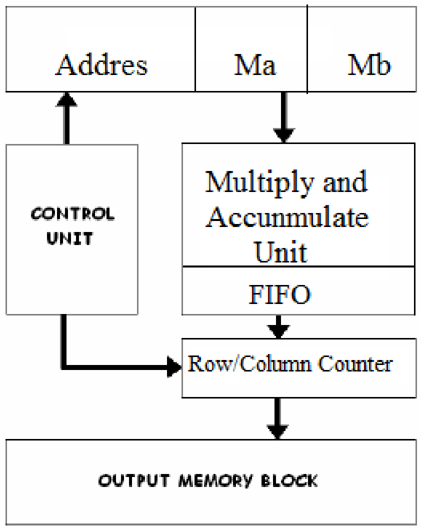

Fig1. Processing Element

Multiplier module:-

Matrix multiplication requires a number of multiply and adds units. Processing structure take several clock cycles to do all the essential multiply add operations. By analyzing the described technique for parallel matrix multiplication, we can improve the performance of the matrix multiplication. The overall performance is limited by the number of multiplications and additions that could be arranged in parallel. Multiplier Module consists of multiplication of two eight bit numbers which create the results of sixteen bit numbers, then adding the result to the earlier accumulated value, which must then be restored in the registers for upcoming accumulations. This circuit is also check for overflow, due to the huge number of multiplication operation. Matrix multiplication structural design is based on serial-parallel sign multiplier and adder module. This modification will eliminate the intercommunication between parallel processing elements (PEs), and allows each PE to operate in isolation.

Computational accuracy:-

Matrix multiplication reveals significant data reuse. Each element of matrix A is multiplied by N elements of matrix B (a row), and each element of matrix B is multiplied by M elements of matrix A (a column). Also, each element of matrix C is updated L times. Multiply-add units perform all of the arithmetic computations. Each multiply add unit consist of a pipelined multiplier and a pipelined adder. The multiplier and the adder are also considered as the arithmetic operators. Each multiply add unit consists of a pipelined multiplier and a pipelined adder. The multiplier and the adder are also considered as the arithmetic operators. Matrix A element and Matrix B element is applied during the clock time tclk, while c should be provided ('Delay multiplier") tclk + 1 cycles later, to account for the multiplier pipeline and the register at the output of the multiplier. As a result, r will be available during cycle t + Delay multiplier + Delay adder + 2, which corresponds to the latencies of the multiplier and adder pipelines, and the two registers at their outputs. Another version of Equation that reflects the timing relationships is: r(t + Delay multiplier + Delay adder + 2) = A(tclk) X B(tclk) + c(tclk)

The matrix method convolutions for two signals simulation are shown in fig…………….. At the time scale of 200ns to 250ns the discrete signal x(n)= {2,1,2,1} and h(n)={1,2,3,4} shows the result of y(m) as

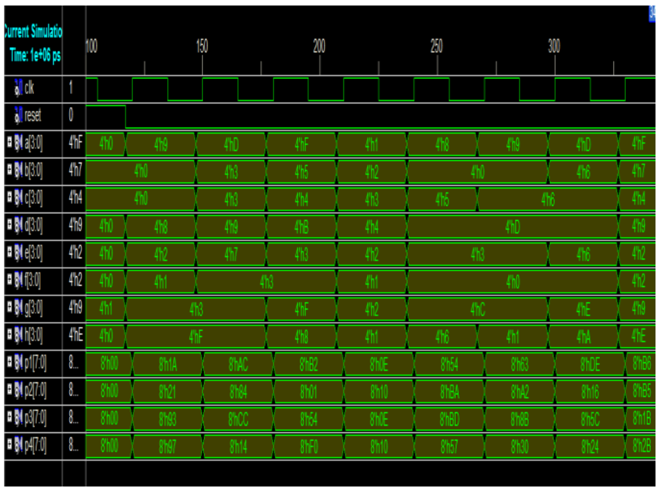
Fig2. Matrix Method for Convolution of Two signals

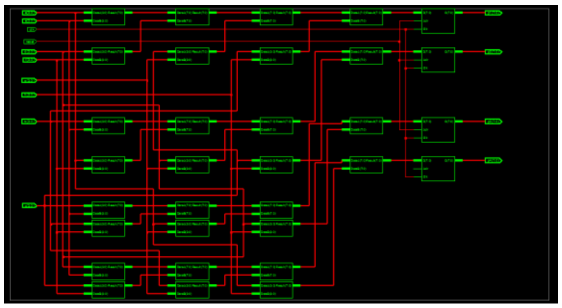
Fig3.RTL View of Covolution using Matrix Method**.**

Synthesis report shows the gate count of:

Multipliers	**:**	**16 Numbers**
4x4-bit multiplier	**:**	**16 Numbers**
8-bit adder	**:**	**12 Numbers**
Flip-Flops	**:**	**32 Numbers**

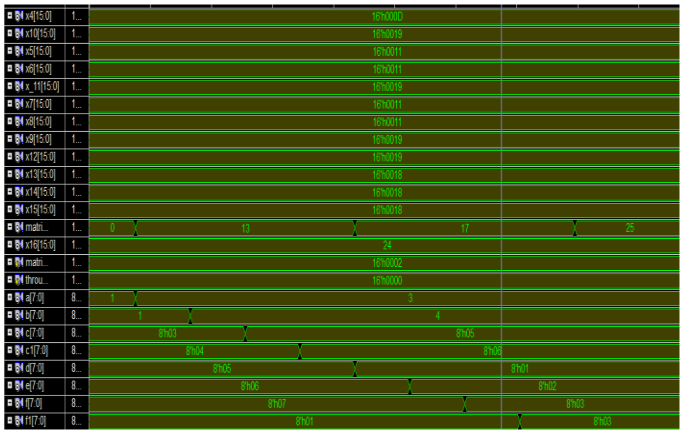
Fig4. FIFO Register Shifting

Number of Bits	Throughput Reference paper	Throughput This work
8	-	195.5
64	68.20	162.5
128	69.63	95
256	70.21	85
512	70.62	80

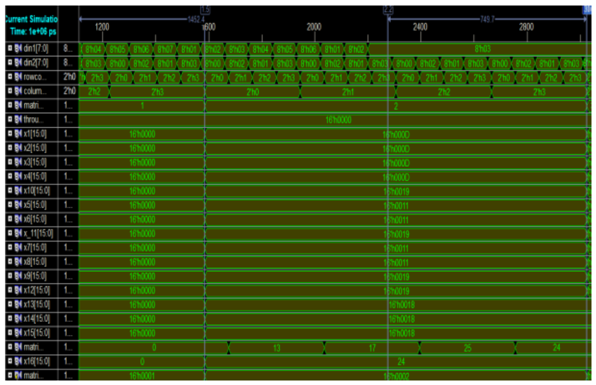
Fig5.Timing Simulation for Matrix Multiplication

Conclusion:-

The simulation shows the accuracy of architecture with low computational time and limited number of gate count. In this paper, we considered two different examples of matrix multiplier architecture where speed is the main constraint. The performance is evaluated by computing its execution time on simulator. 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, RAM, counter and control logic modules. It improvements in latency, computational-time, throughput for performing matrix multiplication. The expected outcomes are:

- Hardware resource utilization will minimized.
- It will reduces the system latency.
- Minimize the computational time.
- Increase in processing throughput.
- Increase the speed of matrix multiplication.

Algorithm proposed removes the intercommunication between parallel processing elements (PEs), and allows each PE to operate in isolation. Also, this algorithm allows the implementation using matrices of random dimension.

References:-

1. Soydan Redif, and Server Kasap "Novel Reconfigurable Hardware Architecture for Polynomial Matrix Multiplications" IEEE Transactions On Very Large Scale Integration (VLSI) Systems vol 12 issue 6 Feb 2015.
2. Syed M. Qasim, Shuja A. Abbasi,and Bandar A. Almashary "Throughput Latency Implementation ofMatrix Multiplication using Field ProgrammableGateArray" IEEE Transactions On Very Large Scale Integration (Vlsi) Systems, Vol. 26, No. 6, Nov2012.
3. Davide Anastasia and Yiannis Andreopoulos "Throughput-Distortion Computation of Generic Matrix Multiplication: Toward a Computation Channel for Digital Signal Processing Systems" IEEE Transactions On Signal Processing, Vol. 60, No. 4, April 2012 pp no 2024.
4. 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.
5. Nan Zhang "A Novel Parallel Scan for Multicore Processors and Its Application in Sparse
6. Matrix-Vector Multiplication" IEEE Transactions On Parallel And Distributed Systems, Vol. 23, No. 3, March 2012 pp no 397.