JAAST
ONLINE ISSN 2277- 1565
PRINT ISSN 0976 - 4828

International Archive of Applied Sciences and Technology
IAAST; Vol 4 [2] June 2013: 62-69

[ISO9001: 2008 Certified Organization]
www.soeagra.com/iaast/iaast.htm

CODEN: IAASCA ORIGINAL ARTICLE

An Efficient Method for Implementation of Convolution

[1]G.Ramanjaneya Reddy, [2]A. Srinivasulu
[1]Srinivasa Institute of Tech & Sci, Kadapa,
[2]GITAM University, Banglore campus , India.
E-mail: [1]rama29raman@gmail.com, [2]sreesri.avvaru@gmail.com

ABSTRACT
This paper presents an on the spot methodology of reducing convolution processing time using hardware computing and implementations of discrete linear convolution of two finite length sequences (NXN). This implementation method is realized by simplifying the convolution building blocks. The purpose of this analysis is to prove the feasibility of an FPGA that performs a convolution on an acquired image in real time. The proposed implementation uses a changed hierarchical design approach, which efficiently and accurately quickens computation; reduces power, hardware resources, and area considerably. The efficiency of the proposed convolution circuit is tested by embedding it during a prime level FPGA. In addition, the presented circuit uses less power consumption and delay from input to output. It additionally provides the required modularity, expandability, and regularity to form different convolutions for any variety of bits.-+
KEY WORDS: *convolution, FPGA, power consumption, time delay*

Received 29.04.2013 Accepted 07.06.2013

INTRODUCTION

Convolution provides the mathematical framework for Digital Signal Processor(DSP). It is the single most important technique in Digital Signal Processing. Convolution is a mathematical way of combining two signals to form a third signal[1].

Many approaches have been attempted to reduce the convolution processing time using hardware and software algorithms. But they are restricted to specific applications. Some approaches are listed below:

1. Presented a design for fast convolve for CDMA signals. This is based on avoiding complex operations such as FFT based convolves. They used substitution of the FFT for a Walsh which reduces the operations three times[2] because it uses only real additions but it requires more hardware like counters and RAM blocks which increases activity factor[10].

2. Using image processing functions such as convolution filtering, high performance can be achieved by exploiting parallelism and minimizing hardware cost[11], but different filter widths and thus potentially different hardware structures are needed for different applications. It is therefore difficult to make a fixed parallel structure efficient[3].

3. The main problem in implementing and computing convolution is speed, area and power which affect any DSP system[6]. Speeding up convolution using a Hardware Description Language for design entry not only increases (improves) the level of abstraction, but also opens new possibilities for using programmable devices[3][7].

4. In an application involving spatial scaling of images, for example, a larger filter kernel would be required for large scale factors, a small one for modest scaling[8]. It would be expensive to implement the entire largest desired filter kernel[4], and wasteful for small scale factors convolution can check all the phase shifts in one step. This is usually done by using the known FFT-based convolution. Each FFT (or IFFT) requires N *log* N complex multiplications and N *log* N complex additions. Therefore, some algorithm require approximately 3N (*log* N) +N complex multiplications and 3N(*log*N)+N additions[10]. Implementing the algorithm in parallel hardware will speed up the process but the implementation itself is very complex and requires a huge silicon area[7].

5. Today, most DSPs suffer from limitations in available address space, or the ability to interface with surrounding systems. The use of high speed FPGAs[5][11], together with DSPs, can often increase the system bandwidth, by providing additional functionality to the general purpose DSP[8][9].

PROPOSED ARCHITECTURE

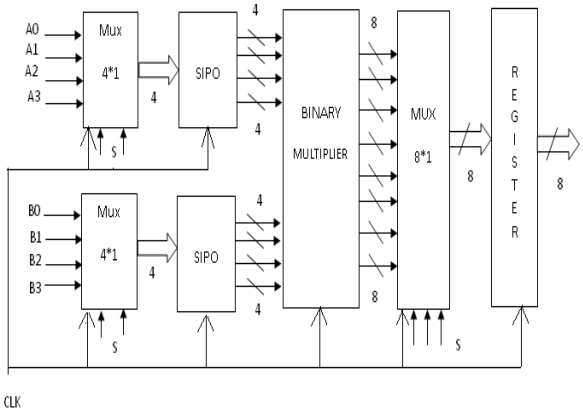

Fig 1. Block diagram of overall convolution Process

The input is applied to the multiplexers. Based on the selection line the data will be selected and it will produce the output in each clock cycle. The output data from the multiplexer is applied to the serial input and parallel output block, the data will be convert serial to parallel. The output of the serial input parallel output block is connected to the binary multiplier so the binary multipliers do the multiplication operation and the output is converted into parallel to serial. The data will be stored in the register.

MULTIPLEXERS 4*1 AND 8*1

A multiplexer, sometimes referred to as a "multiplexor" or simply "mux", is a device that selects between a numbers of input signals. In its simplest form, a multiplexer will have two signal inputs, one control input, and one output. A multiplexer is a device which selects any one of the inputs from 2^n inputs and directed to output depending on n-select lines.

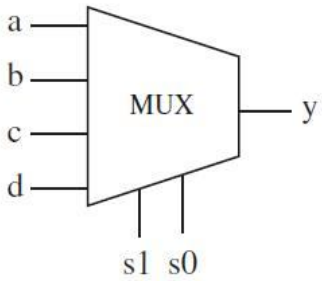

Fig 2: **4*1 multiplexer**

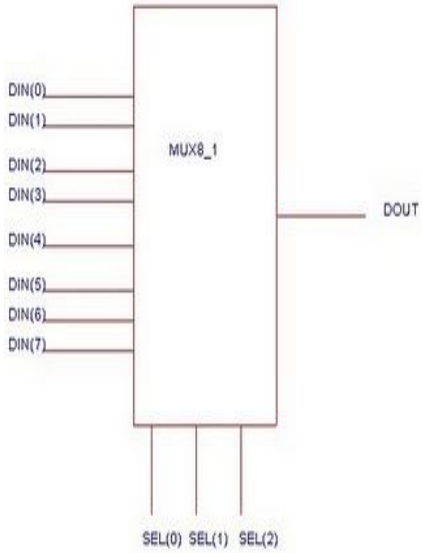

Fig 3: **8*1 multiplexer**

The higher order multiplexers can be implemented using the lower order multiplexers. The 4*1 multiplexer can be implemented using two 2*1 multiplexers and so on. Similarly an 8*1 multiplexer can be implemented using two 4*1 multiplexers.Working of multiplexer:
1. Selects any one of the input from 2^n inputs
2. Directs to the output depending on n-selection lines.

SERIAL IN PARALLEL OUT:
A serial-in/parallel-out shift register is similar to the serial-in/ serial-out shift register in that it shifts data into internal storage elements and shifts data out at the serial-out, data-out, pin. It is different in that it makes all the internal stages available as outputs.

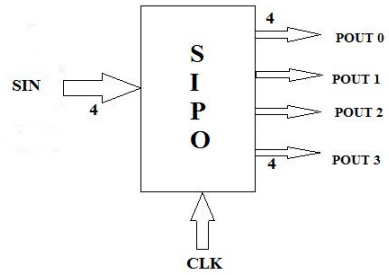

Fig 4: SIPO Block Diagram

Therefore, a serial in/parallel-out shift register converts data from serial format to parallel format. If four data bits are shifted in by four clock pulses via a single wire at data-in, below, the data becomes available simultaneously on the four Outputs Q_A to Q_D after the fourth clock pulse.

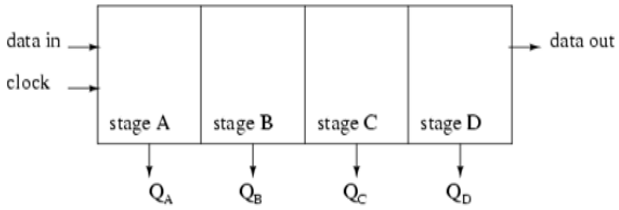

Fig 5. serial in parallel out Shift Register with 4- stages

The practical application of the serial-in/parallel-out shift register is to convert data from serial format on a single wire to parallel format on multiple wires. Perhaps, we will illuminate four LEDs (Light Emitting Diodes) with the four outputs (**Q_A Q_B Q_C Q_D**).

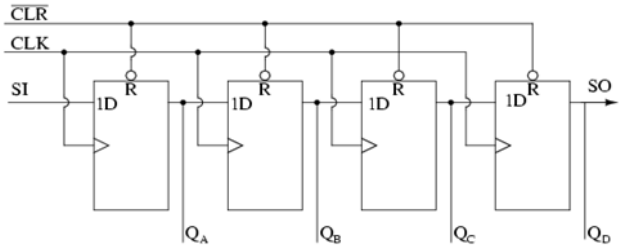

Fig6. Serial in parallel out shift Register Details

The above details of the serial-in/parallel-out shift register are fairly simple. It looks like a serial-in/ serial-out shift register with taps added to each stage output. Serial data shifts in at SI (Serial Input). After a number of clocks equal to the number of stages, the first data bit in appears at SO (Q_D) in the above figure. In general, there is no SO pin. The last stage (Q_D above) serves as SO and is cascaded to the next package if it exists.

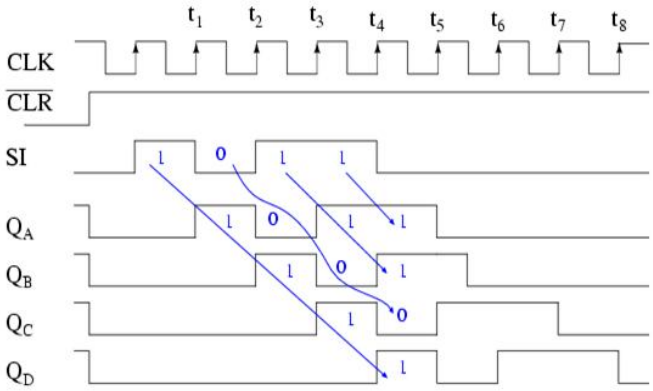

Fig 7. SIPO Example waveform

The shift register has been cleared prior to any data by CLR', an active low signal, which clears all type D Flip-Flops within the shift register. Note the serial data 1011 pattern presented at the SI input. This data is synchronized with the clock CLK. This would be the case if it is being shifted in from something like another shift register, for example, a parallel-in/ serial-out shift register.

On the first clock at t1, the data 1 at SI is shifted from D to Q of the first shift register stage. After t2 this first data bit is at Q_B. After t3 it is at Q_C. After t4 it is at Q_D. Four clock pulses have shifted the first data bit all the way to the last stage Q_D. The second data bit a 0 is at Q_C after the 4th clock. The third data bit a 1 is at Q_B. The fourth data bit another 1 is at Q_A. Thus, the serial data input pattern 1011 is contained in (Q_D Q_C Q_B Q_A). It is now available on the four outputs. It will available on the four outputs from just after clock t_4 to just before t_5. This parallel data must be used or stored between these two times, or it will be lost due to shifting out the Q_D stage on following clocks t_5 to t_8 as shown.

BINARY MULTIPLIER:

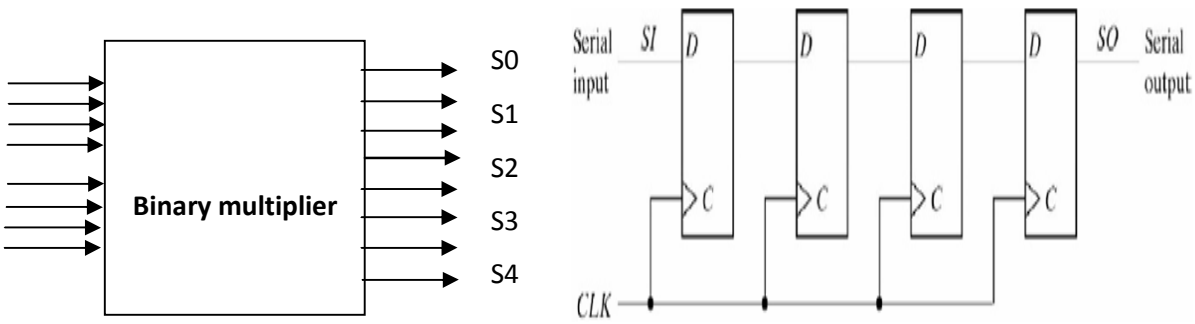

Fig 8: Binary multiplier

The binary multiplier used here is a 4-bit multiplier which takes two four bit inputs and gives an 8-bit output.

The binary multiplier which is employed in convolution here in the present project has a special characteristic that the internal carry will not be forwarded to next stage. So the number of outputs obtained here is seven only because in binary multiplier the MSB part is nothing but the carry obtained from the second MSB so as carry is not forwarded only seven bits will be obtained as output.

RIGISTER:

A circuit with flip-flops is considered a sequential circuit even in the absence of Combinational logic. Circuits that include flip-flops are usually classified by the function they perform. Two such circuits are registers and counters.

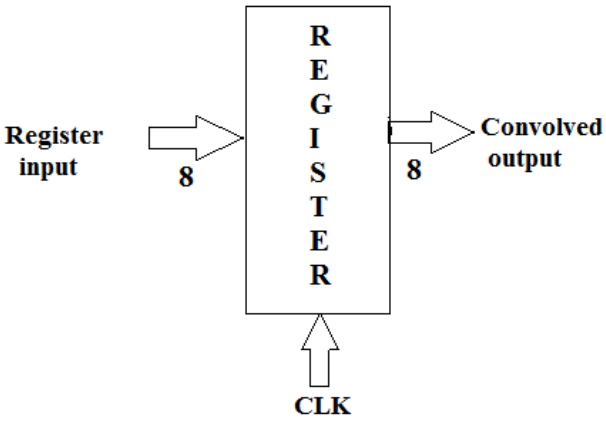

Fig 9. Register

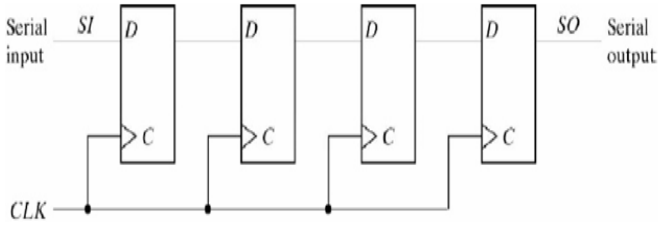

Fig 10: A 4-Bit Register

A Register is a group of flip-flops. Its basic function is to hold information within a digital system so as to make it available to the logic units during the computing process. However, a register may also have additional capabilities associated with it. It may have combinational gates that perform certain data-processing tasks.

Various types of registers are available on the market. A simple 4-bit register is shown in fig 10. The common clock input triggers all flip-flops and the binary data available at the four inputs are transferred into the register. The clear input is useful for clearing the register to all 0's output. Registers capable of shifting their binary contents in one or both directions. A unidirectional 4-bit shift register that uses only flip-flops is in fig 11:

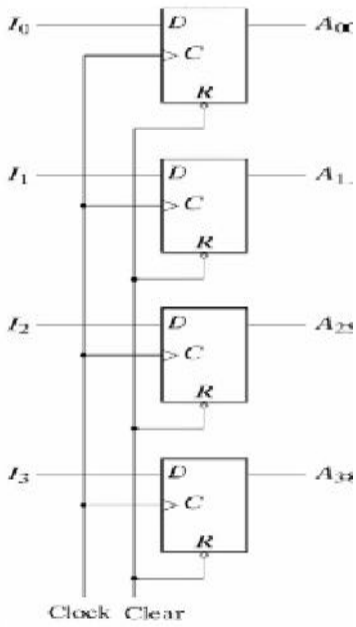

Fig 11: 4 bit Shift register

SIMULATION RESULTS

The Convolution process and the developed architecture for the required functionality were discussed in the previous chapters. Now this chapter deals with the simulation and synthesis results of the Convolution process. Here Modelsim tool is used in order to simulate the design and checks the functionality of the design. Once the functional verification is done, the design will be taken to the Xilinx tool for Synthesis process.

The Appropriate test cases have been identified in order to test this modeled Convolution process architecture. Based on the identified values, the simulation results which describes the operation of the process has been achieved. This proves that the modeled design works properly as per its functionality.

TOP MODULE:

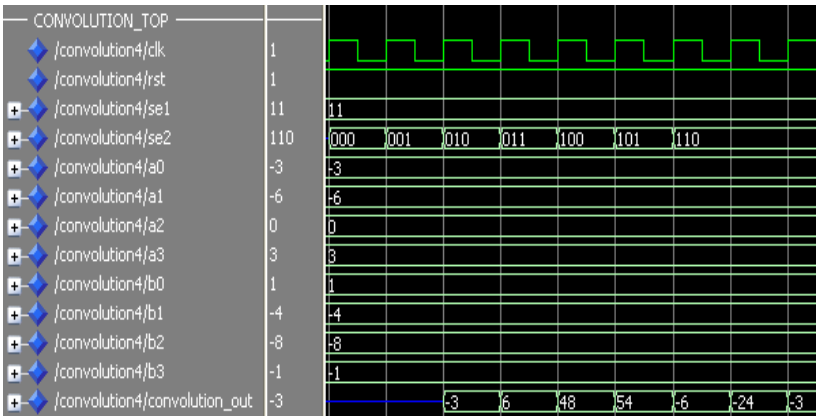

Fig 12. Simulation Result of Convolution top module

The top module shows the processes of convolution. The input is applied to the multiplexers. Based on the selection line the data will be selected and it will produce the output in each clock cycle. The output data from the multiplexer is applied to the serial input and parallel output block, the data will be convert serial to parallel. The output of the serial input parallel output block is connected to the binary multiplier so the binary multipliers do the multiplication operation and the output is converted into parallel to serial. The data will be stored in the register.

4*1 MULTIPLEXER:

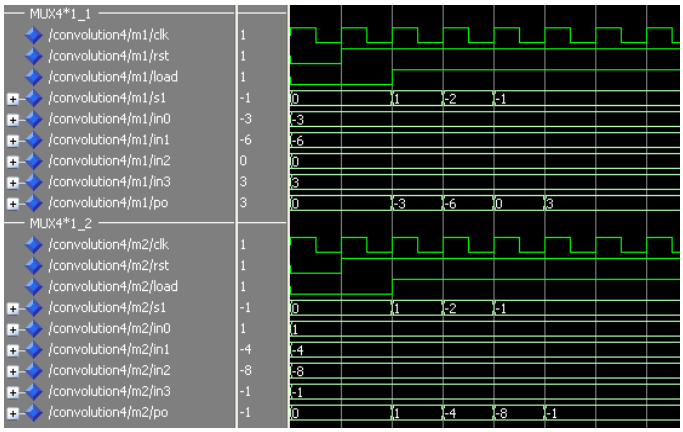

Fig 13. Simulation Result of 4*1 Multiplexer

In general the multiplexer will have '2^n' number of inputs and n selection lines and one output. Here we are using 4:1 multiplexer, so it will have 4 inputs and 2 selection lines and one output. Based on selection line the input will be selected and we will get the output. Here for doing convolution we have the blocks multiplexer 2:1 of two blocks. The above figure shows the simulation results of 4:1 multiplexer.

Serial Input Parallel Output:

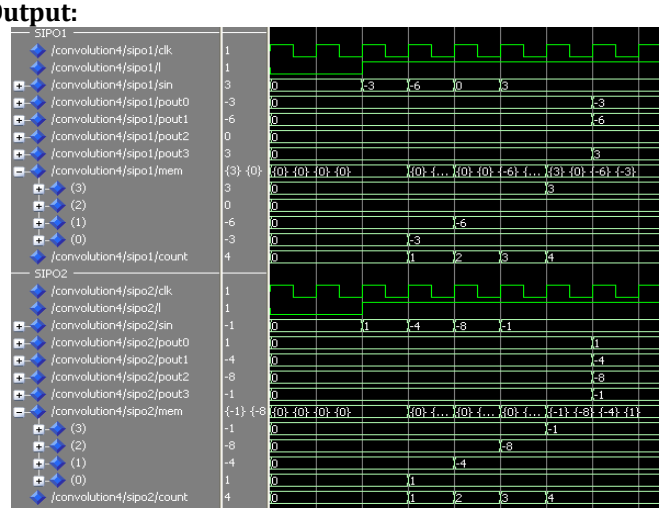

Fig 14. Simulation Result of serial input and parallel output

In this block the input is the output of the multiplexer. The serial input and parallel output block will do, the data from the multiplexer it will take as the input and it will hold the value up to four clock cycles and it will convert the data serial into parallel. The above figure shows the simulation results of the Serial input of data into parallel output.

BINARY MULTIPLIER

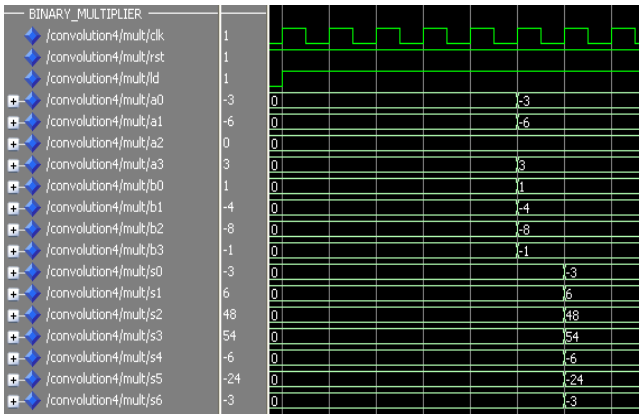

Fig 15. Simulation Result of Binary multiplier

The binary multiplier will do the multiplication operation. For the binary multiplier the input is the data which we are getting from the serial input parallel output block. Binary multiplier do the multiplication from the serial input and parallel output blocks.

8*1 MULTIPLEXER:

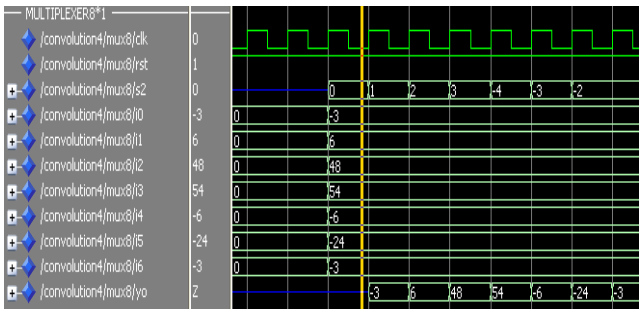

Fig 16. Simulation Result of 8*1 Multiplexer

The data from the binary multiplier is applied to the multiplexer. The multiplexer convert the parallel data into the serial data and it will be stored into the register.

REGISTER:

The outputs of multiplexer are stored in the register which contains flip-flops.

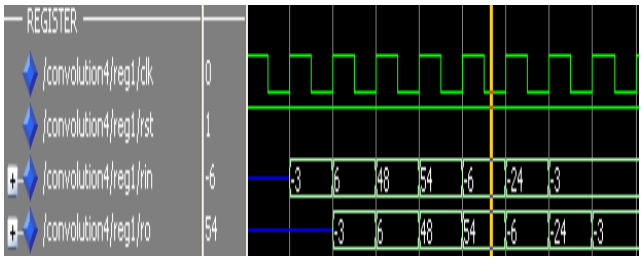

Fig 17. Simulation Result of Register

CONCLUSION

We presented an optimized implementation of discrete linear convolution. This particular model has the advantage of being fine tuned for signal processing; in this case it uses the mean squared error measurement and objective measures of enhancement to achieve a more effective signal processing model. This implementation has the advantage of being optimized based on operation, power and area. To accurately analyze our proposed system, we have coded our design using the Verilog hardware description language and have synthesized it for FPGA products using ISE, Modelsim and DC compiler for other processor usage.

The proposed circuit uses only **5mw** and saves almost **35%** area and it takes **20ns** to complete. This shows improvement of more than 50% less power. As FPGA technology matures and much larger arrays become practical, techniques that allow the automatic generation of highly-parallel architectures will become central to high performance computing.

We have also described some simple techniques for generation of convolution pipelines for image processing and other applications. Higher level techniques and approaches are also needed. FPGAs permit restructurable processing, and restructurable interconnects are also becoming available.

REFERENCES

1. John W. Pierre, (1996).“A Novel Method for Calculating the Convolution Sum of Two Finite Length Sequences”, IEEE transaction on education, VOL. 39, NO. 1.
2. W. W. Smith, J. M. Smith,(1995). “Handbook f Real-Time Fast Fourier Transforms”, IEEE Press, p. 28.
3. R. G. Shoup,(1994).“Parameterized convolution filtering in a field programmable gate array,” in selected papers from the Oxford 1993 international workshop on field programmable logic and applications on More FPGAs. Oxford, United Kingdom: Abingdon EE&CS Books, pp. 274–280.
4. Iván Rodríguez, (2008). “Parallel Cyclic Convolution Based on Recursive Formulations of Block Pseudocirculant MatricesMarvi Teixeira”, IEEE, transaction on signal processing,
5. Thomas Oelsner ,“Implementation of Data Convolution Algorithms in FPGAs” , QuickLogic Europe http://www.quicklogic.com/images/appnote18.pdf
6. Chao Cheng , Keshab K. Parhi ,(2007). “Low-Cost Fast VLSI Algorithm for Discrete Fourier Transform”, IEEE,. IEEE transaction on circuits and systems, VOL. 54.
7. J. I. Guo, C. M. Liu, and C. W. Jen, (1992).“The efficient memory-based VLSI array designs for DFT and DCT,” IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process., vol. 37, no. 10, pp. 723–733.
8. T. S. Chang, J. I. Guo, and C. W. Jen,(2000). “Hardware-efficient DFT designs with cyclic convolution and subexpression sharing”,IEEE Trans. Circuits Syst. II, Analog Digital Signal Process., vol. 47, no. 9, pp. 886– 892.
9. C. Cheng and K. K. Parhi,(2007).“Hardware efficient fast DCT based on novel cyclic convolution structures”, IEEE Trans. Signal Process., vol. 54, no.11, pp. 4419–4434.
10. Chao Cheng , Keshab K. Parhi(2004). ”Hardware Efficient Fast Parallel FIR Filter Structures Based on Iterated Short Convolution” IEEE, and, IEEE transaction on circuits and systems, VOL. 51, NO. 8.http://www.tc.umn.edu/~chen0867/ParallelFIR2004_TCASI.pdf.
11. Abdulqadir Alaqeeli, Janusz Starzyk, “Hardware Implementation for Fast Convolution with a PN Code Using Field Programmable Gate”, Ohio University, http://www.ent. ohiou.edu/ ~starzyk/network / Research /Papers /Recent%20conferences/Conv_FPGA_PN_code_SSST2001.pdf.

Citation of Article: G.Ramanjaneya Reddy and A. Srinivasulu. An Efficient Method for Implementation of Convolution. Int. Arch. App. Sci. Technol., Vol 4 [2] June 2013: 62-69