International Journal of Electrical and Electronics Research ISSN 2348-6988 (online)
Vol. 3, Issue 4, pp: (63-69), Month: October - December 2015, Available at: www.researchpublish.com

Analysis of Radix-2^2 SDF Pipeline FFT Architecture in VLSI Using Chip Scope

G. Mohana Durga[1], D.V.R. Mohan[2]

[1]M.Tech Student, [2]Professor, Department of ECE, SRKR Engineering College, Bhimavaram, Andhra Pradesh, India

***Abstract*: Pipelined Fast Fourier Transform (FFT) architectures are efficient for long instances in digital communication. For long instances, Single-Path Delay-Feedback (SDF) FFT architectures minimize required memory, which can dominate circuit area and power dissipation. This paper discusses about the architecture of Radix-2^2 Single Delay Feedback for N = 1024 samples and its implementation. Then the code is dumped into Xilinx Spartan 3E kit to analysis Radix-2^2 SDF architecture by using chip scope. Radix-2^2 SDF algorithm has the same multiplicative complexity as radix-4 algorithm, but retains the butterfly structure of radix-2 algorithm. Simulation and Synthesis are carried on Modelsim 6.3 and Xilinx ISE 12.2. The design has been coded in Verilog [6] and targeted into Xilinx Spartan3E FPGA kit.**

***Keywords*: Fast Fourier Transform (FFT), Single-Path Delay-Feedback (SDF), Radix-2^2, Twiddle factor, ROM, Butterfly unit.**

1. INTRODUCTION

A Fast Fourier Transform (FFT) [3] is an algorithm to compute the Discrete Fourier Transform (DFT) and its inverse. Fourier analysis converts time (or space) to frequency and vice versa. The DFT [1] is obtained by decomposing a sequence of values into components of different frequencies. A FFT is a way to compute the same result more quickly, computing the DFT of N samples, it takes O(N^2) arithmetical operations, while a FFT can compute the same DFT in only O(N log N) operations. The difference in speed can be enormous, especially for long data sets where N may be in the thousands or millions. Evaluating the DFT's sums directly involves N^2 complex multiplications and N (N−1) complex additions.

Pipeline FFT processor is a special class of processors for DFT computation utilizing fast algorithms. It is characterized with real-time, non-stopping processing as the data sequence passing the processor. It is an AT^2 non-optimal approach with $AT^2 = O(N^3)$, since the area lower bound is O (N). It is a real-time processing with a new metric for non-optimal is given the time complexity of O (N). For FFT computation it requires O (long) of Arithmetic Elements (AE) (adders and multipliers). For hardware implementation, various FFT processors have been proposed. These implementations can be mainly classified into memory-based and pipeline architecture based. Memory-based architecture is widely adapted to design FFT processor, also known as single processing element (PE) approach. This design style is usually composed of a main PE and several memory units, thus the hardware cost and power consumption both is lower than the other architecture style. However, this kind of architecture style has long latency, low throughput and cannot be parallelized. On the other hand, the pipeline architecture style can get rid of the disadvantages of the foregoing style, at the cost of an acceptable hardware overhead. Generally, the pipeline FFT processors have two popular design types. One uses single-path delay feedback (SDF) pipeline architecture and the other uses multiple-path delay commutator (MDC) pipeline architecture. Such implementations are advantageous to low-power design, especially for the applications in portable DSP devices. Based on these reasons, the SDF pipeline FFT is adopted in this. The proposed architecture includes a distributed arithmetic based complex multiplier instead of using multiplier to store twiddle factors. A hardware oriented Radix-2^2 algorithm is then developed by integrating a twiddle factor decomposition technique in divide and conquer approach to

form a spatially regular signal flow graph (SFG). Mapping the algorithm to the cascading delay feedback structure leads to the proposed architecture.

2. RADIX-2^2 SDF FFT ARCHITECTURE

Radix-2^2 Single-path Delay Feedback (R2^2SDF) is a new VLSI architecture. R$\mathbf{2^2}$SDF architecture for N=1024 is shown in the fig I. This architecture data-path is similar to R2SDF and it can reduce required number of multipliers. The implementation uses two types of butterflies; one identical to that in R2SDF, the other is similar to R2SDF but it also contains the logic to implement the trivial twiddle factor multiplication, as shown in fig II and fig III respectively. x(n) which provides input samples to R2^2SDF architecture W1(n),W2(n),W3(n) are twiddle factors and X(k) represents output samples which are in bit reversal order.

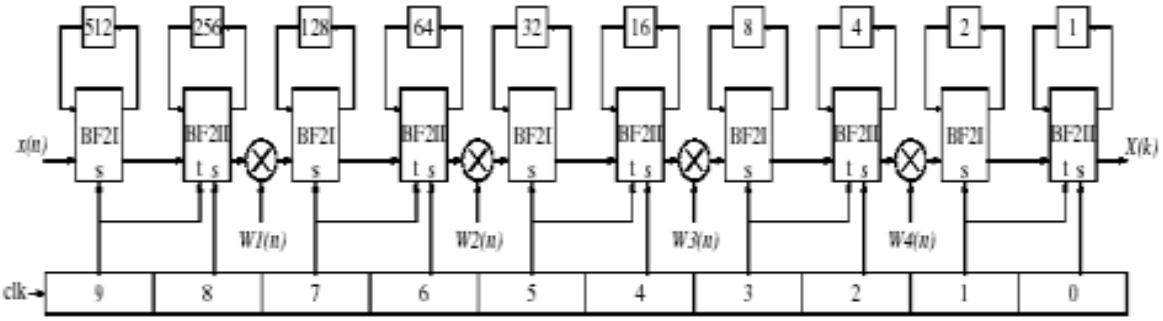

FIG 1: R2^2SDF PIPELINE FFT ARCHITECTURE FOR N= 1024

Due to the spatial regularity of Radix-2^2 algorithm, the synchronization control of the processor is very simple. A ((log2N) bit binary counter serves two purposes: synchronization controller and address counter for twiddle factor reading in each stage.

Butterfly circuitry at each stage combines data which are n/2 samples apart where n is equal to 2^m for DIT and 2^log2 (N-1-m) for DIF architectures, and m is the incrementing butterfly stage number. In the Radix-2^2 SDF architecture, two unique types of butterfly structures are used (BF1 and BF2). The BF1 butterfly, which is identical to those used in Radix-2 SDF pipelines, computes a 2-point DFT. The depth of the delay-line (d) is a function of the number of points in the transform (N) and the stage number (m) shows the BF1 structure from the proposed design.

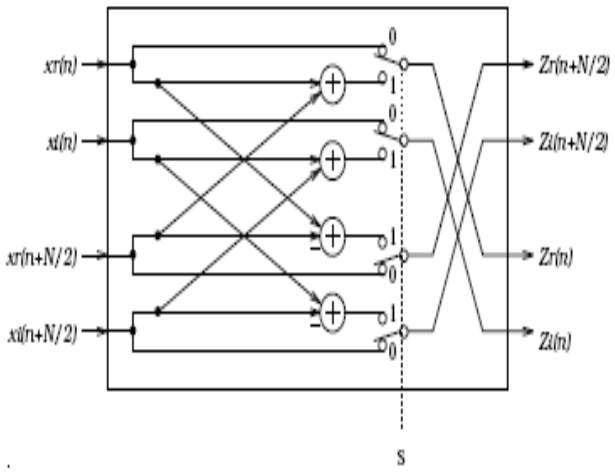

FIG 2: BF2I ARCHITECTURE

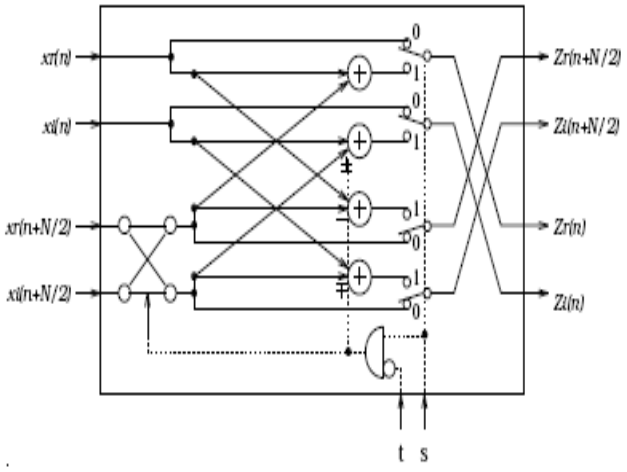

FIG 3: BF2II ARCHITECTURE

During the first N/2 clock samples when‘s’ is low, multiplexors direct the input data to the feedback registers. On the next N/2 clock samples ‘s’ is asserted high, the multiplexors are switched and the butterfly addition is performed between the input data and feedback output. This periodic process is continued until samples have been processed.

$$Z1(n) = x(n) + x(n + N/2) \quad (1)$$

$$Z1(n + N/2) = x(n) - x(n + N/2) \quad (2)$$

$$0 \le n < N/2$$

The butterfly output Z1(n) is sent to apply the twiddle factor, and Z1(n + N/2) is sent back to the shift registers to be "multiplied" in still next N/2 cycles when the first half of the next frame of time sequence is loaded in. The operation of the second butterfly is similar to that of the first one, except the "distance" of butterfly input sequence are just N/4 and the trivial twiddle factor multiplication has been implemented by real-imaginary swapping with a commutator and controlled Add/subtract operations, which requires two bit control signal from the synchronizing counter. After N-1 clock cycles, the complete DFT transform result streams out to the right, in bit-reversed order. The next frame of transform can be computed without pausing due to the pipelined processing of each stage. In practical implementation, pipeline register should be inserted between each multiplier and butterfly stage to improve performance.

3. RADIX-2^2SDF FFT ALGORITHM

In this hardware oriented algorithm will be that it has the same number of non-trivial multiplications at the same positions in the SFG as of radix-4 algorithms, but has the same butterfly structure as that of radix-2 algorithms. These algorithms with this feature are not completely new. Another algorithm combining radix-4 and radix-"4+2" in DIT form has been used to decrease the scaling error in R2MDC architecture, without altering the multiplier requirement. The clear derivation of the algorithm in DIF form with perception of reducing the hardware requirement in the context pipeline FFT processor is, however, yet to be developed. To avoid confusing with the well known radix-2/4 split-radix algorithm and the mixed radix-"4 + 2" algorithms, the notion of radix-2^2 algorithm is used to clearly reflect the structural relation with radix-2 algorithm and the identical computational requirement with radix-4 algorithm.

The Discrete Fourier Transforms (DFT) X (k) of an N-point discrete-time signal x (n) is defined by

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \quad (3)$$

Where W_N denotes the N^{th} primitive root of unity, with its exponent evaluated modulo N. To make the derivation of the new algorithm clearer, consider the first 2 steps of decomposition in the radix-2 DIF FFT together. Applying a 3-dimensional linear index map,

$$n = < N_1 n_1 + N_2 n_2 + \ldots + N_m n_m >_N$$

$$k = < K_1 k_1 + K_2 k_2 + \ldots + K_m k_m >_N \quad (4)$$

When the above substitutions are applied to DFT definition,

$$N = \frac{N}{4}.2.2, \quad n = < N/2\, n_1 + N/4\, n_2 + n_3 >_N, \; k = < k_1 + 2k_2 + 4k_3 >_N, \quad (5)$$

$$X(k_1 + 2k_2 + 4k_3) = \sum_{n3=0}^{\frac{N}{4}-1} \sum_{n2=0}^{1} \sum_{n1=0}^{1} x\left(\frac{N}{2} n_1 + \frac{N}{4} n_2 + n_3\right) W_N^{\left(\frac{N}{2} n_1 + \frac{N}{4} n_2 + n_3\right)(k_1 + 2k_2 + 4k_3)}$$

$$= \sum_{n3=0}^{\frac{N}{4}-1} \sum_{n2=0}^{1} \left\{ B_{\frac{N}{2}}^{k1}\left(\frac{N}{4} n_2 + n_3\right) W_N^{\left(\frac{N}{4} n_2 + n_3\right)k_1} \right\} W_N^{\left(\frac{N}{4} n_2 + n_3\right)(2k_2 + 4k_3)} \quad (6)$$

Where the butterfly structure has the form of

$$B_{\frac{N}{2}}^{k1}\left(\frac{N}{4} n_2 + n_3\right) = x\left(\frac{N}{4} n_2 + n_3\right) + (-1)^{k_1} x\left(\frac{N}{4} n_2 + n_3 + \frac{N}{2}\right) \quad (7)$$

If the expression within the braces of eqn. (6) is to be computed before further decomposition, an ordinary radix-2 DIF FFT results. The key idea of the new algorithm is to proceed the second step decomposition to the remaining DFT coefficients, including the "twiddle factor" $W_N^{\left(\frac{N}{4} n_2 + n_3\right)k_1}$, to exploit the exceptional values in multiplication before the next butterfly is constructed. Decomposing the composite twiddle factor and observe that

$$W_N^{\left(\frac{N}{4} n_2 + n_3\right)(k_1 + 2k_2 + 4k_3)} = W_N^{N n_2 k_3} W_N^{\frac{N}{4} n_2 (k_1 + 2k_2)} W_N^{n_3 (k_1 + 2k_2)} W_N^{4 n_3 k_3}$$

$$= (-j)^{n_2 (k_1 + 2k_2)} W_N^{n_3 (k_1 + 2k_2)} W_N^{4 n_3 k_3} \quad (8)$$

$$X(k_1 + 2k_2 + 4k_3) = \sum_{n_3=0}^{\frac{N}{4}-1} \left[H(k_1, k_2, n_3) W_N^{n_3(k_1+2k_2)}\right] W_{\frac{N}{4}}^{n_3 k_3} \quad (9)$$

Where $H(k_1, k_2, n_3)$is expressed in eqn. (10).

$$H(k_1, k_2, n_3) = \left[x(n_3) + (-1)^{k_1} x\left(n_3 + \frac{N}{2}\right)\right] + (-j)^{(k_1+2k_2)} \left[x\left(n_3 + \frac{N}{4}\right) + (-1)^{k_1} x\left(n_3 + \frac{3}{4}N\right)\right] \quad (10)$$

$H(k_1, k_2, n_3)$ represents the first two stages of butterflies with only trivial multiplications in the SFG, as BF I and BF II. After these two stages, full multipliers are required to compute the product of the decomposed twiddle factor $\boldsymbol{W_N^{n_3(k_1+2k_2)}}$ in eqn. (8). Note the order of the twiddle factors is different from that of radix-4 algorithm. Applying this CFA procedure recursively to the remaining DFTs of length N=1024 in eqn. (5), the complete Radix-2^2 DIF FFT algorithm is obtained. For example Radix-2^2 DIF SDF FFT flow graph for N = 16 as shown in the figure 4.

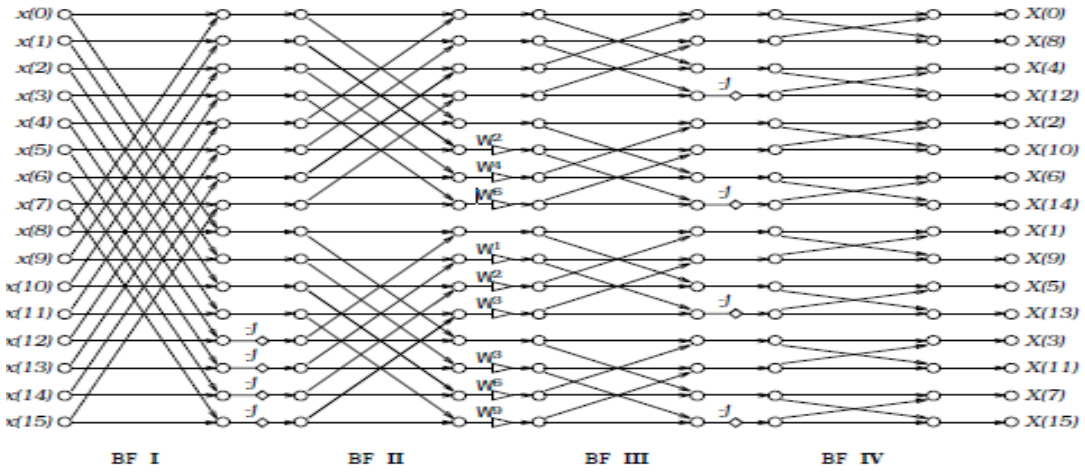

Fig 4: Radix-2^2 DIF SDF FFT flow graph for N = 16

4. SIMULATION RESULTS

Radix-2^2 SDF FFT programming consists of number of modules. For each module we should provide input data (real and imaginary), output data (real and imaginary), synchronous reset, clock and then start simulation.

4.1 Internal Architecture:

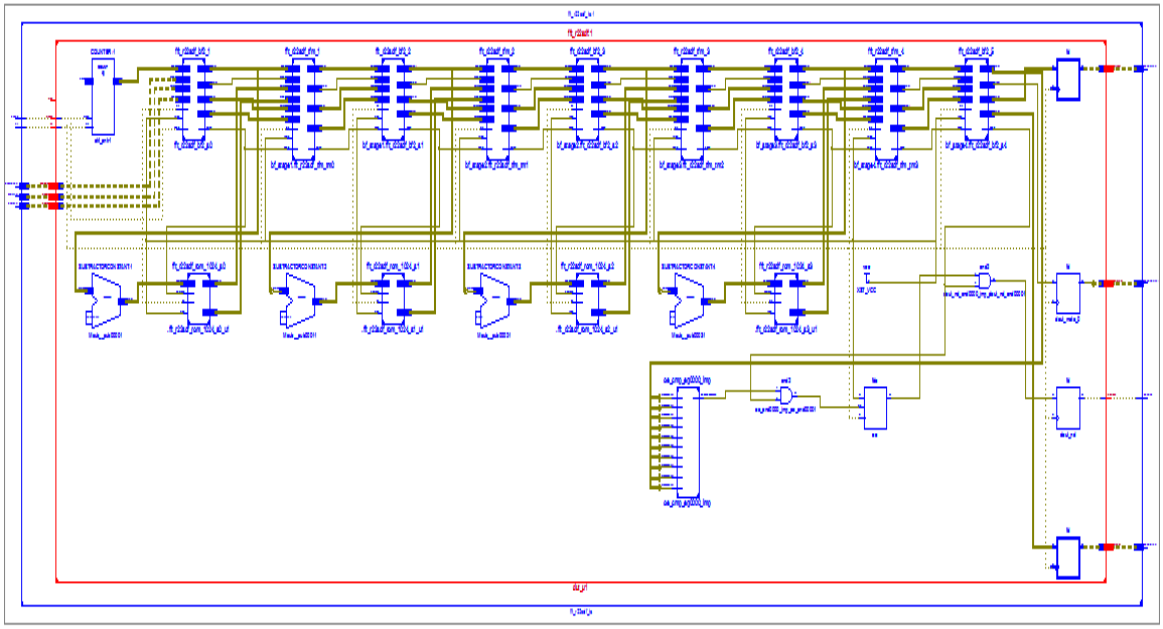

Fig 4: Internal Architecture of Radix2^2 SDF FFT N=1024

4.2 BF2 Simulation:

BF2 module is a combination of BF2I and BF2II modules.

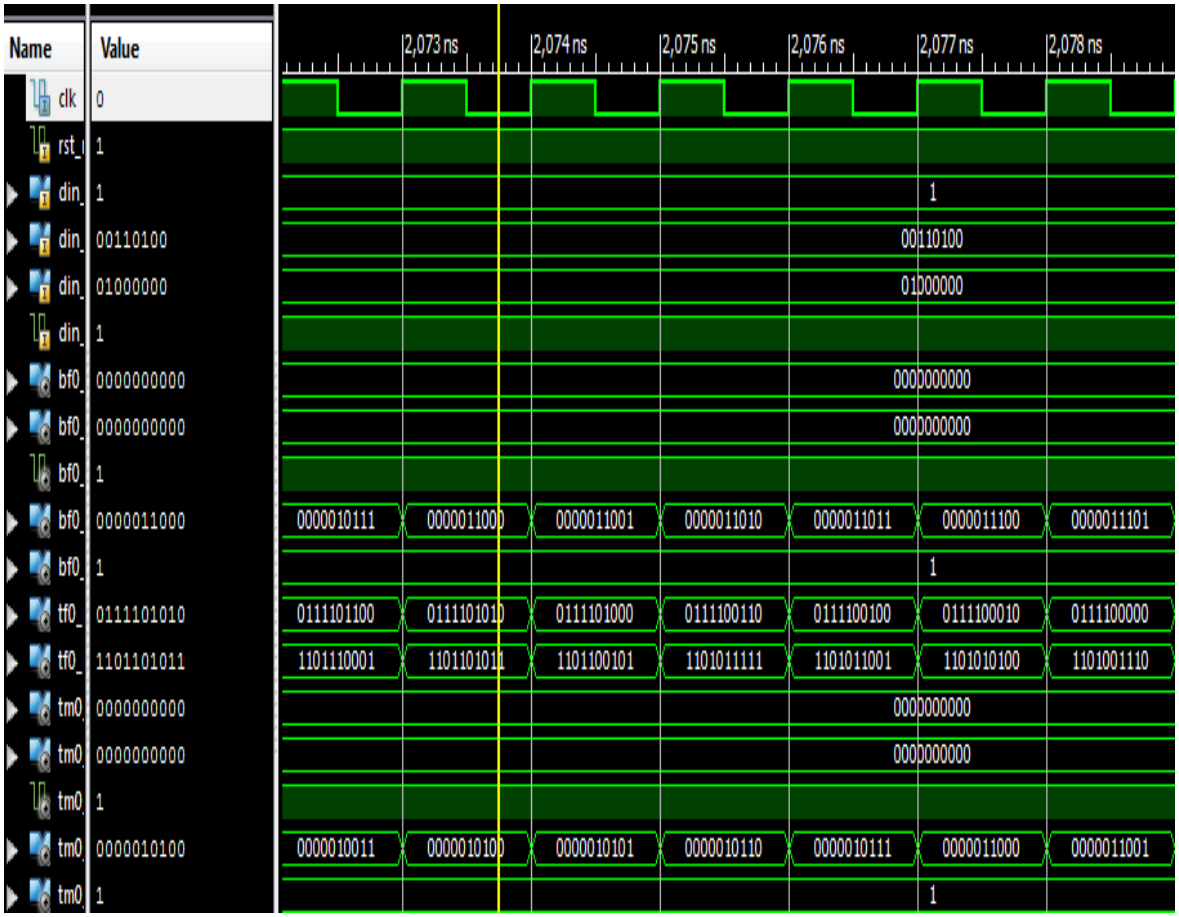

FIG 5: BF2 SIMULATION

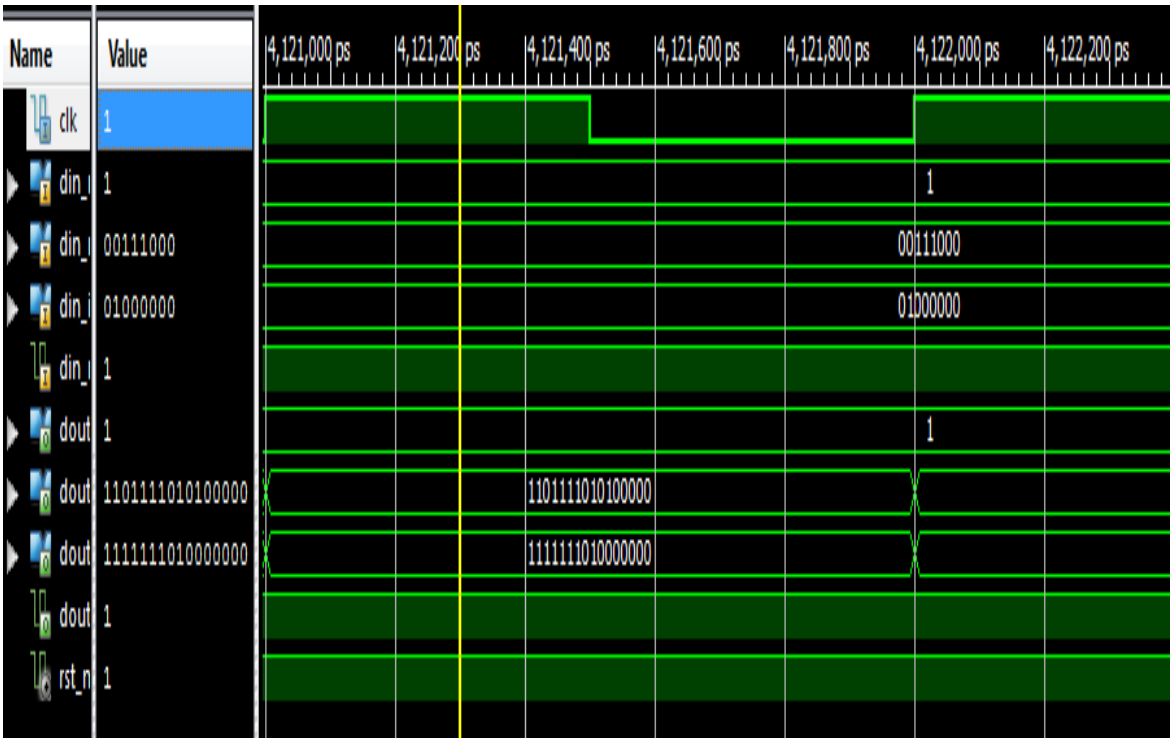

FIG 6: RADIX 2^2 SDF FFT N=1024 SIMULATION

4.3 IMPLEMENTATION RESULTS:

TABLE 1: IMPLEMENTATION RESULTS FOR RADIX 2^2 SDF FOR N=1024

Logic Utilization	Used	Available	Utilization
Total Number Slice Registers	1,466	9,312	15%
Number used as Flip Flops	1,465		
Number used as Latches	1		
Number of 4 input LUTs	3,064	9,312	32%
Number of occupied Slices	1,877	4,656	40%
Number of Slices containing only related logic	1,877	1,877	100%
Number of Slices containing unrelated logic	0	1,877	0%
Total Number of 4 input LUTs	3,183	9,312	34%
Number used as logic	1,511		
Number used as a route-thru	119		
Number used as Shift registers	1,553		
Number of bonded IOBs	53	232	22%
Number of RAMB16s	6	20	30%
Number of BUFGMUXs	2	24	8%
Number of BSCANs	1	1	100%
Number of MULT18X18SIOs	16	20	80%

4.4 CHIP SCOPE PRO ANALYZER RESULT:

ChipScope is a set of tools made by Xilinx [2] that allows to easily probing the internal signals of the design inside an FPGA, by using a logic analyzer. For example, while the design is running on the FPGA, it can trigger when certain events take place and view internal signals of the design.

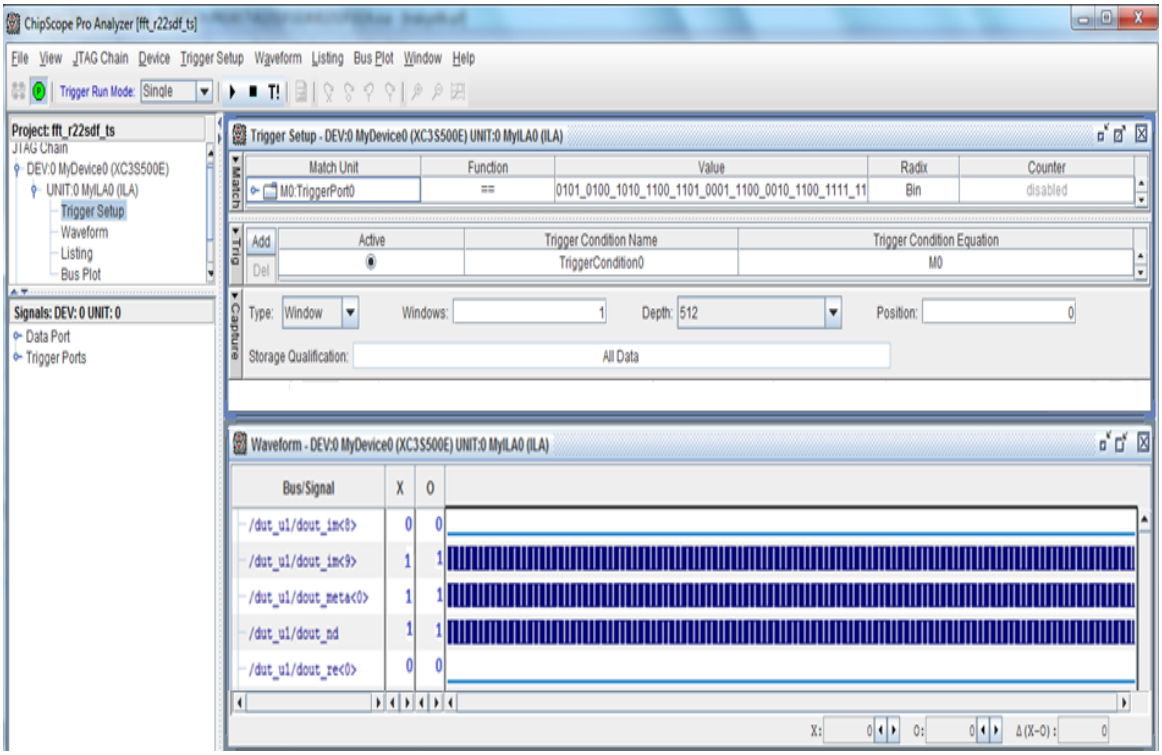

Fig 7: Chip Scope for Radix 2^2 SDF for N=1024

5. CONCLUSION

The simulation of Radix-2^2 single-path delay feedback pipelined FFT/IFFT processor has been designed. Based on this algorithm, efficient pipeline FFT architecture, the Radix-2^2 SDF architecture, is put forward and simulated using Model Sim and Xilinx. The simulation results of BF2I, BF2II and BF2 having input data (real and imaginary), output data (real and imaginary), synchronous reset: rst, clock. Then the code is dumped into Xilinx Spartan 3E kit for analyzing Radix-2^2 SDF architecture by using chip scope.

6. FUTURE SCOPE

Radix-2^2 Single-path Delay Feedback pipeline FFT for N=1024 input samples design is completed. This design can be extended for N=4096 input samples and programming is coded in Verilog [6] and targeted into Xilinx Spartan6 FPGA kit. Pipeline N= 4096 point Radix-4^2SDF and Radix-4^3SDF based FFT VLSI architecture will also be designed to reduce number of butterfly stages and complex multipliers.

REFERENCES

[1] S. He and M. Torkelson. A new expandable 2D systolic array for DFT computation based on symbiosis of 1D array. In Proc. ICA3 PP‟95, pages 12–19, Brisbane, Australia, Apr. 1995.

[2] Xilinx Inc., Logic Core, Product Specification “High-Performance 32-Point Complex FFT/IFFT V3.0”.

[3] C. D. Thompson. Fourier transform in VLSI.IEEE *Trans* .*Compute*r, C-32(11):1047–1057, Nov. 1983.

[4] L.R. Rabiner and B. Gold. Theory and Application of Digital Signal Processing. Prentice-Hall, Inc., 1996.

[5] http://www.hindawi.com

[6] Digital VLSI Systems Design, (A Design Manual for Implementation of Projects on FPGAs and ASICs using Verilog), By Seetharaman Ramachandran.

[7] www.ijesat.org.