Design and simulation of a low power and high speed Fast Fourier Transform for medical image compression

Dr. R. S. Ernest Ravindran
Dr. Ngangbam Phalguni Singh
Mr. Sudhakiran Gunda

OBJECTIVE:

➢To design and concentrate on the development of high-performance FFT algorithm (based on Decimation-In- Time (DIT) domain) to apply in the real time medical diagnosis.

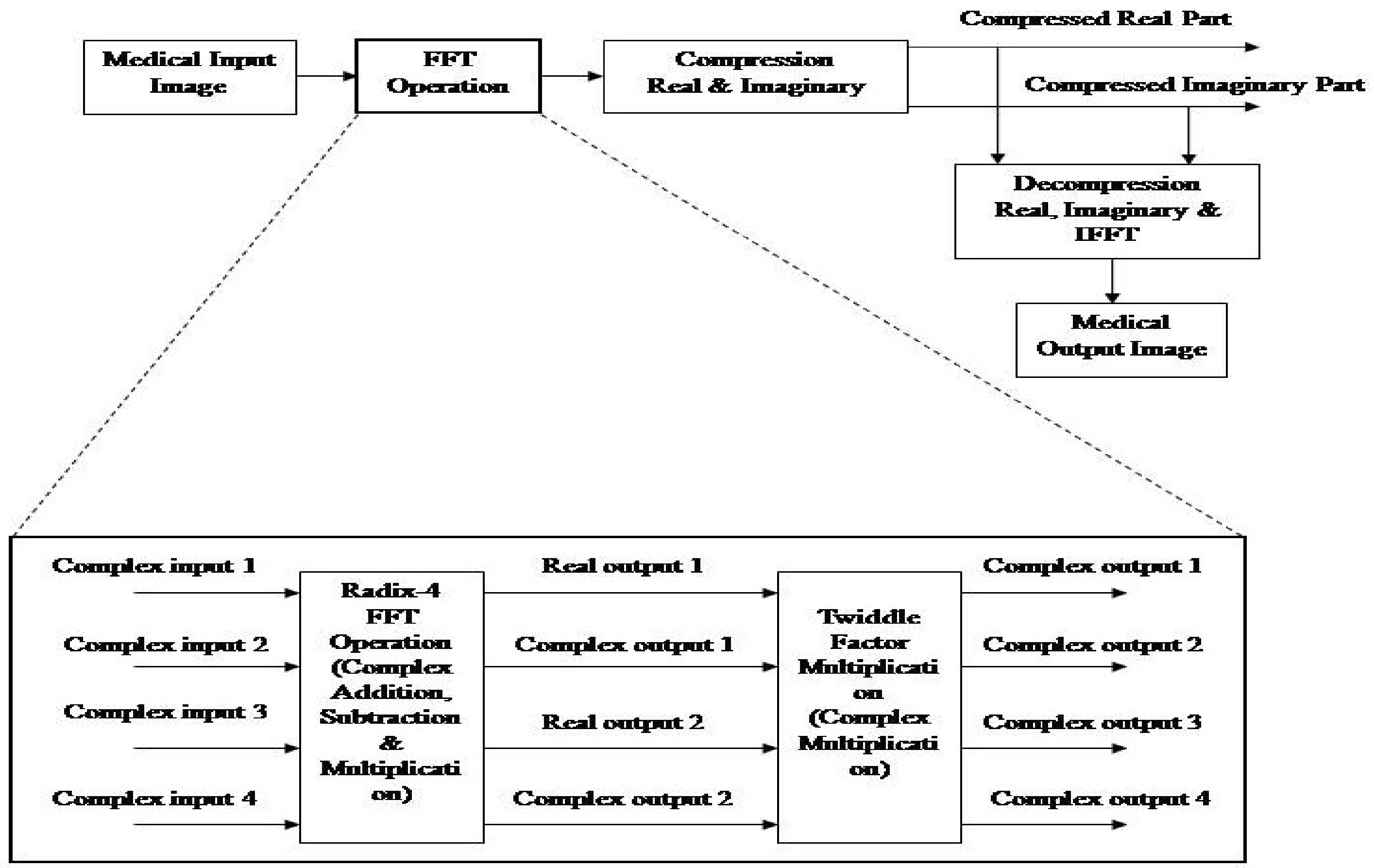


Fig: Block representation of medical image compression using proposed FFT

INTRO…..

- ➢Medical applications use FFT for image reconstruction and frequency domain analysis.
- ➢Medical imaging method/technology - CT, MRI, Ultrasound and Optical Imaging provides images of the human body for clinical diagnosis.
- ➢Images consists of patient's most vital information in the form of pixels
- ➢As hospitals are progressing into digitization, filmless imaging and telemedicine, the medical imagery becomes significantly important in the health sector.
- ➢Led to the major difficulty of developing compression algorithms that prevent diagnostic errors and have a high compression ratio for lower bandwidth and storages.

EXISTING SYSTEM:

- Deemed from paper titled "Design and Simulation of 32-Point FFT Using Radix-2 Algorithm for FPGA Implementation"
- Implementation can be done depending on 2^M
- Follows DOA algorithm

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\left(\frac{2\pi}{N}\right)kn},\ 0 \le k \le N-1$$

$$x(n) = 1/N \sum_{k=0}^{N-1} X(K) e^{j\left(\frac{2\pi}{N}\right)kn};0 \le n \le N-1$$

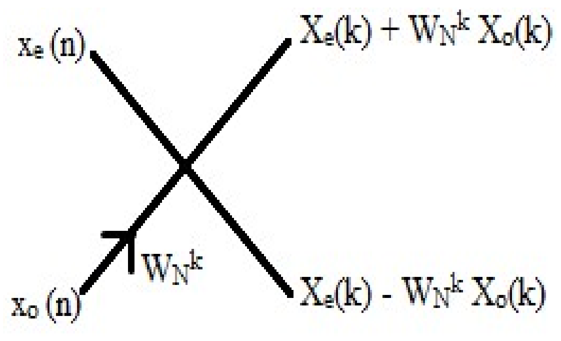


Fig: Radix-2 butterfly

- $W_N^k = e^{-j2\pi k/N}$ - represents twiddle factor
- Decimation in time FFT:
  - Number of stages = $\log_2 N$
  - Number of blocks/stage = $N/2^{stage}$
  - Number of butterflies/block = $2^{stage-1}$
  - Number of multiplications = $N/2 \log_2 N$
  - Number of addition = $N\log_2 N$

FLOW CHART:

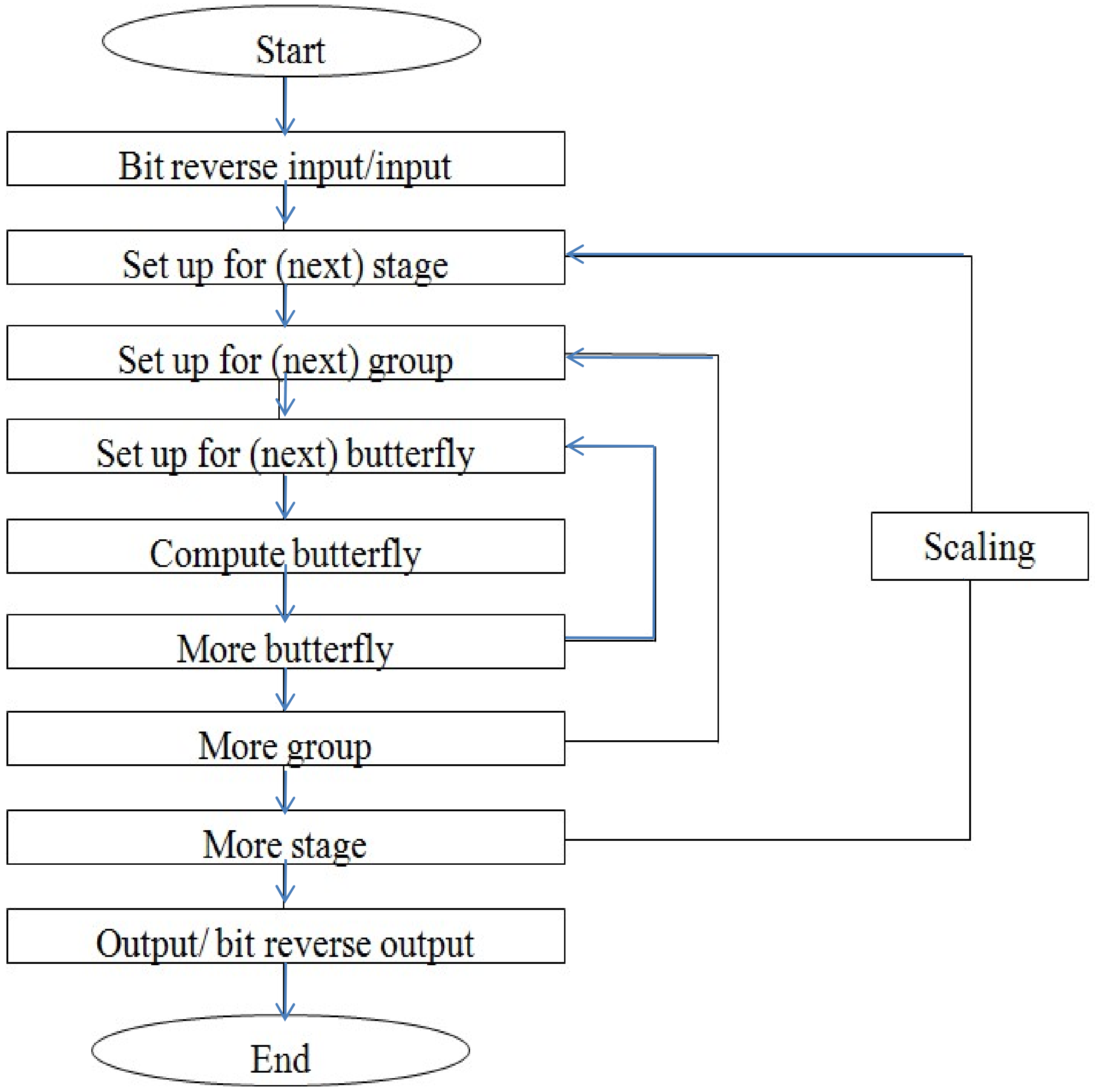


Fig: Flow chart for DIT- FFT Algorithm

STRUCTURE OF 32-POINT RADIX-2 FFT:

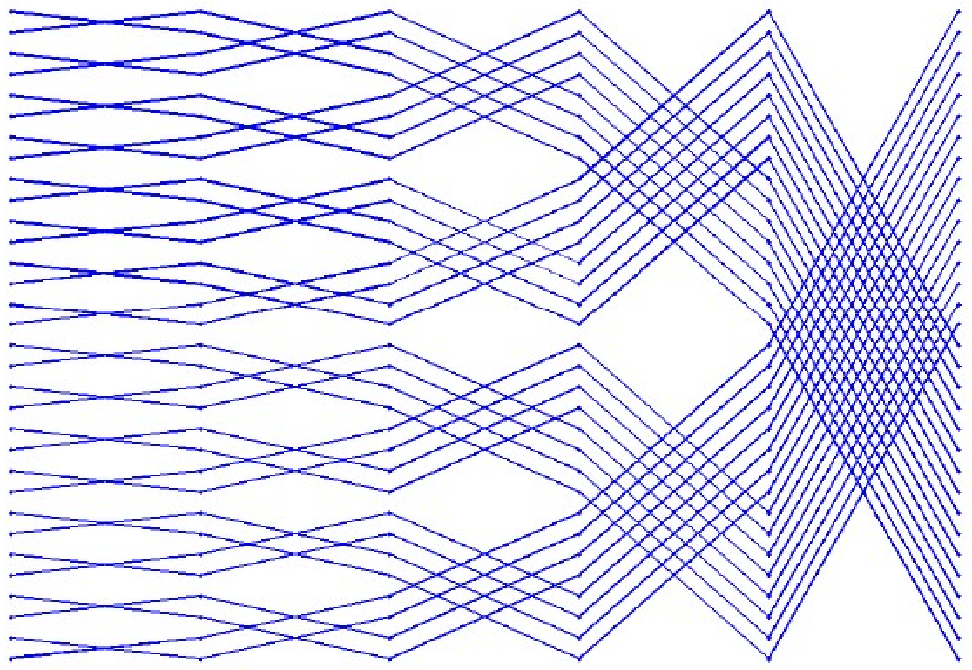


Fig: 32-point radix-2 fft algorithm

LIMITATIONS OF EXISTING SYSTEM:

- ➢Large computational requirements implies less speed
- ➢Maximum amount of resources
- ➢ Number of input and output pins also very low

PROPOSED SYSTEM:

- *Radix-4 algorithm*
- Butterfly of a radix 4 algorithm consists of four inputs and four outputs.
- FFT length is 4^M .
- Samples lies in subsequent locations

BASIC R4 BUTTERFLY:

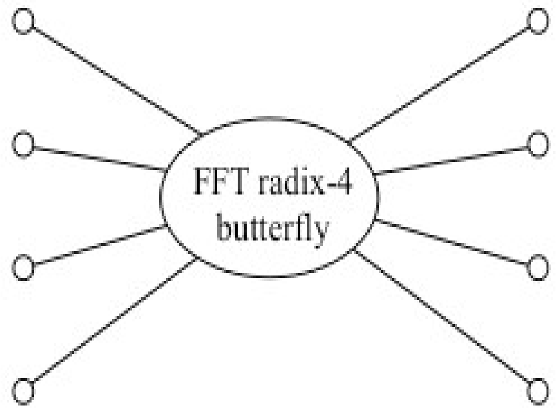


Fig: Basic Radix 4 butterfly diagram

- R-4 DIT-FFT gains its speed by reusing the results of smaller, intermediate stages
- It rearranges the DFT equation into 4 parts as

  n = [0, 4, 8, ... N - 4], n = [1, 5, 9, ... N - 3],

  n = [2, 6, 10, ... N - 2] and

  n = [3, 7, 11, ... N - 1]
- Twiddle factors

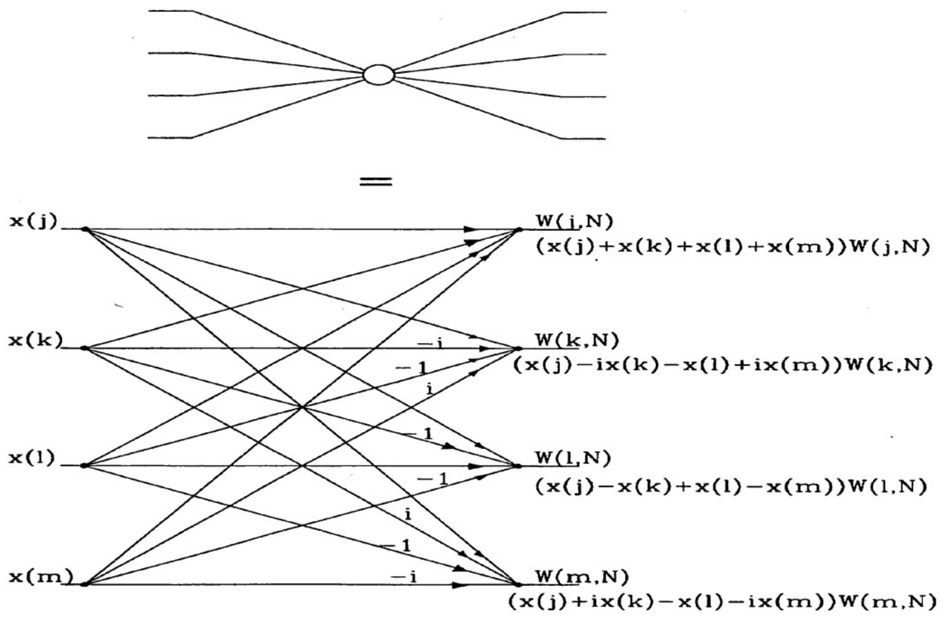

=
x(j)
x(k)
x(l)
x(m)
W(j,N)
(x(j)+x(k)+x(l)+x(m))W(j,N)
W(k,N)
(x(j)−ix(k)−x(l)+ix(m))W(k,N)
W(l,N)
(x(j)−x(k)+x(l)−x(m))W(l,N)
W(m,N)
(x(j)+ix(k)−x(l)−ix(m))W(m,N)
−i
−1
i
−1
−1
i
−1
−i

Fig: Operation of R4 DIT FFT

- If the input sequence is [1,0,1,1] then the output sequence is [3,j,1,-j]

64 POINT R4 BUTTERFLY SCHEMATIC DIAGRAM:

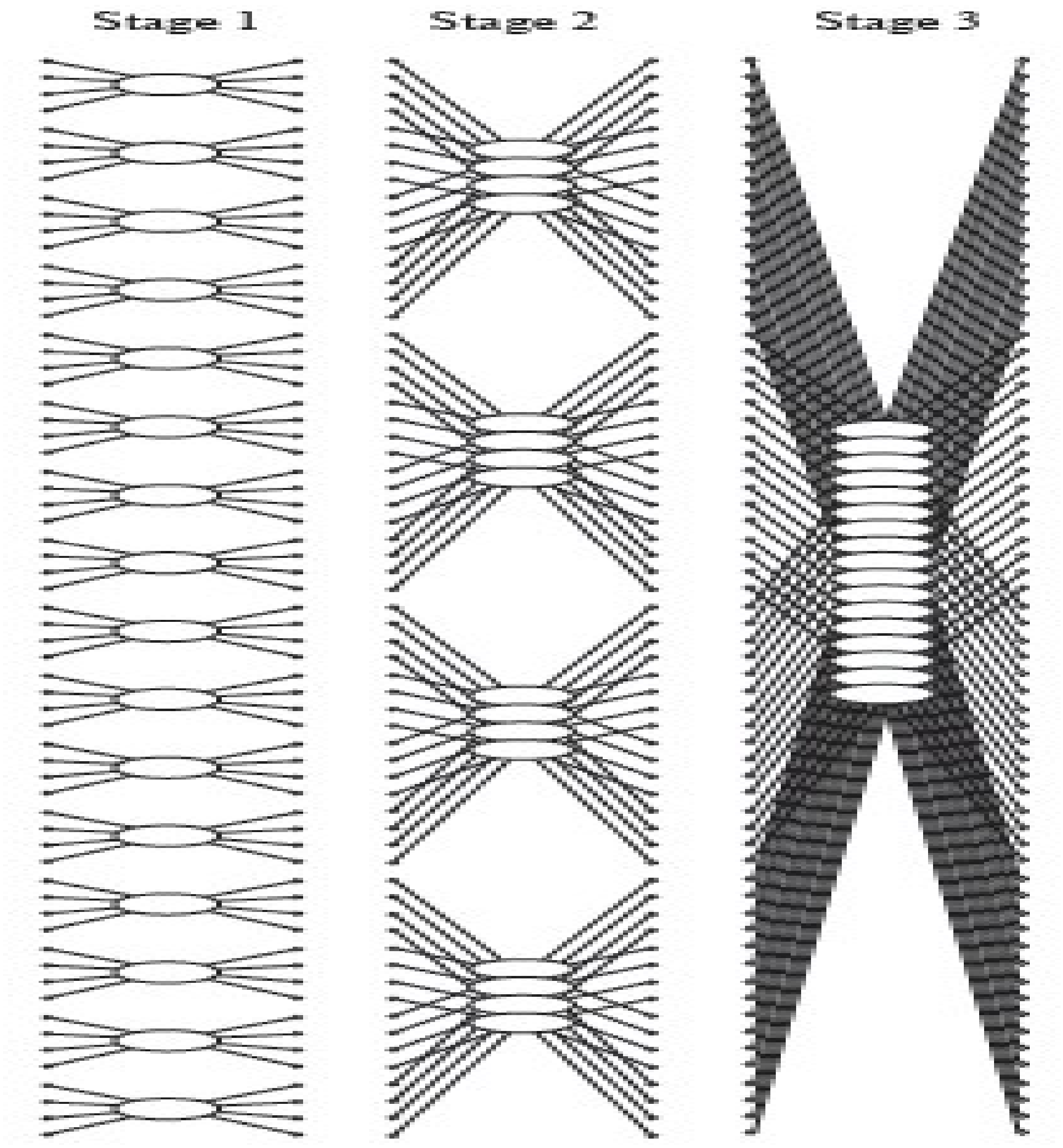


Fig: Radix-4 butterfly diagram

256-bit 64-point R-4 butterfly contains the following:

- Data split of 256 bits and 64 bits
- Even and odd
- DFT four
- Butterfly 4, 8, 16
- Comutator
- Top butter

ALGORITHM:

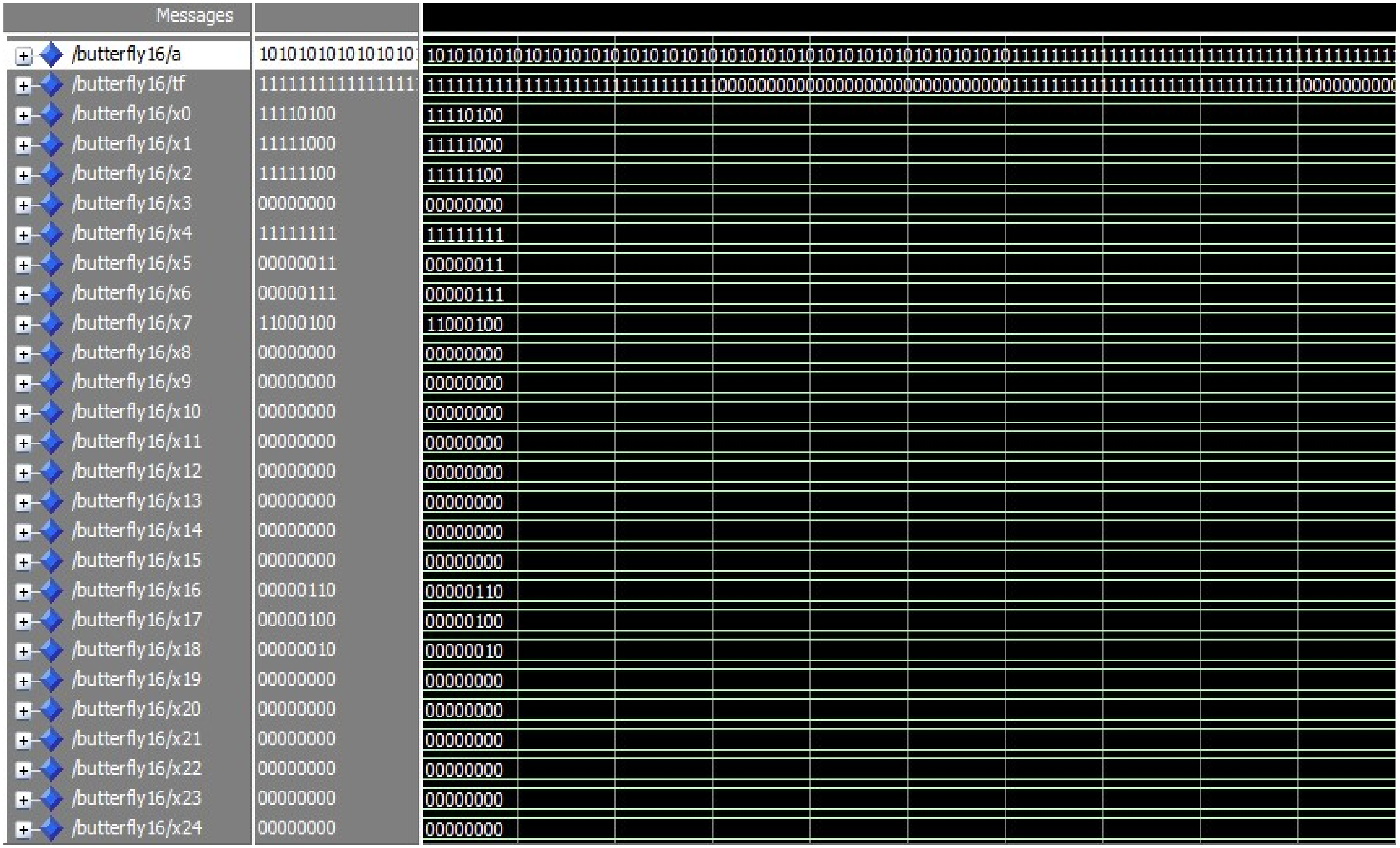
- Declare the input (256 bits), output (64 points each of 8 bits) and other signals (twiddle factors of 256 bits).
- Call the data split and top butter and declare the signals as

```
COMPONENT datasplit256 is
    Port(A: in signed (0 to 255);
        B,C,D,E:OUT SIGNED (0 TO 63));
    End COMPONENT;
    COMPONENT Topbutter is
    port(A: in signed (0 to 63);
        Tf1 ......... Tf16:in signed(0 to 3);
        od11 ......... od88:out signed(0 to 7);
        ev11 ......... ev88:out signed(0 to 7));
    End COMPONENT;
    SIGNAL B,C,D,E:SIGNED (0 TO 63);
```

```
BEGIN
        M1:datasplit
        port
map(A=>A,B=>B,C=>C,D=>D,E=>E);
        M2:Topbutter
        PORT MAP(A=>B,Tf1=>Tf (0 TO
3)......... Tf16=>Tf (60 TO 63),
        od11=>x0,......... od88=>x7,
        ev11=>x8,......... ev88=>x15);
        M3:Topbutter A=>C
        M4:Topbutter A=>D
        M5:Topbutter A=>E
```

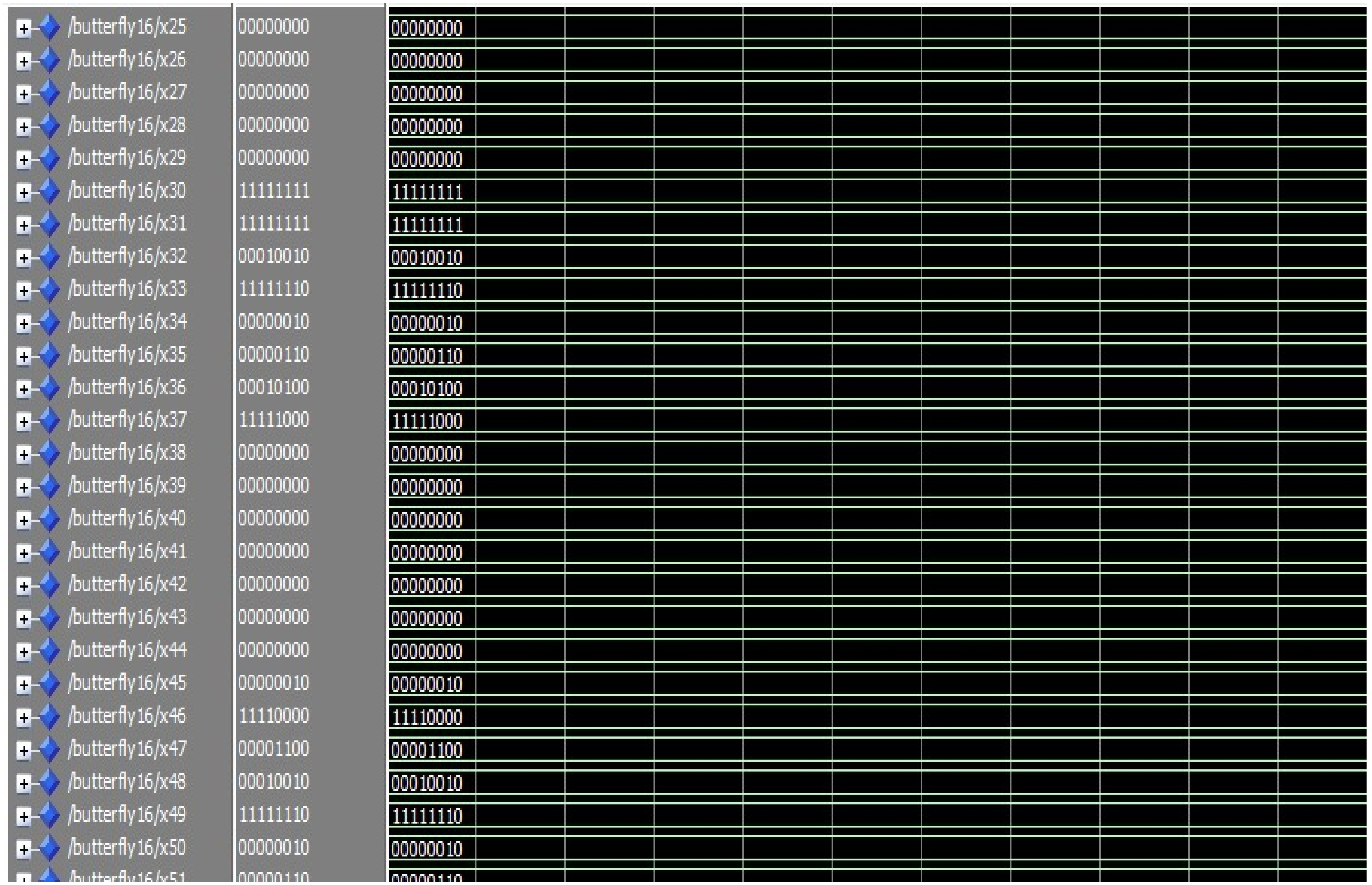
SIMULATION RESULTS:

Simulation results are:



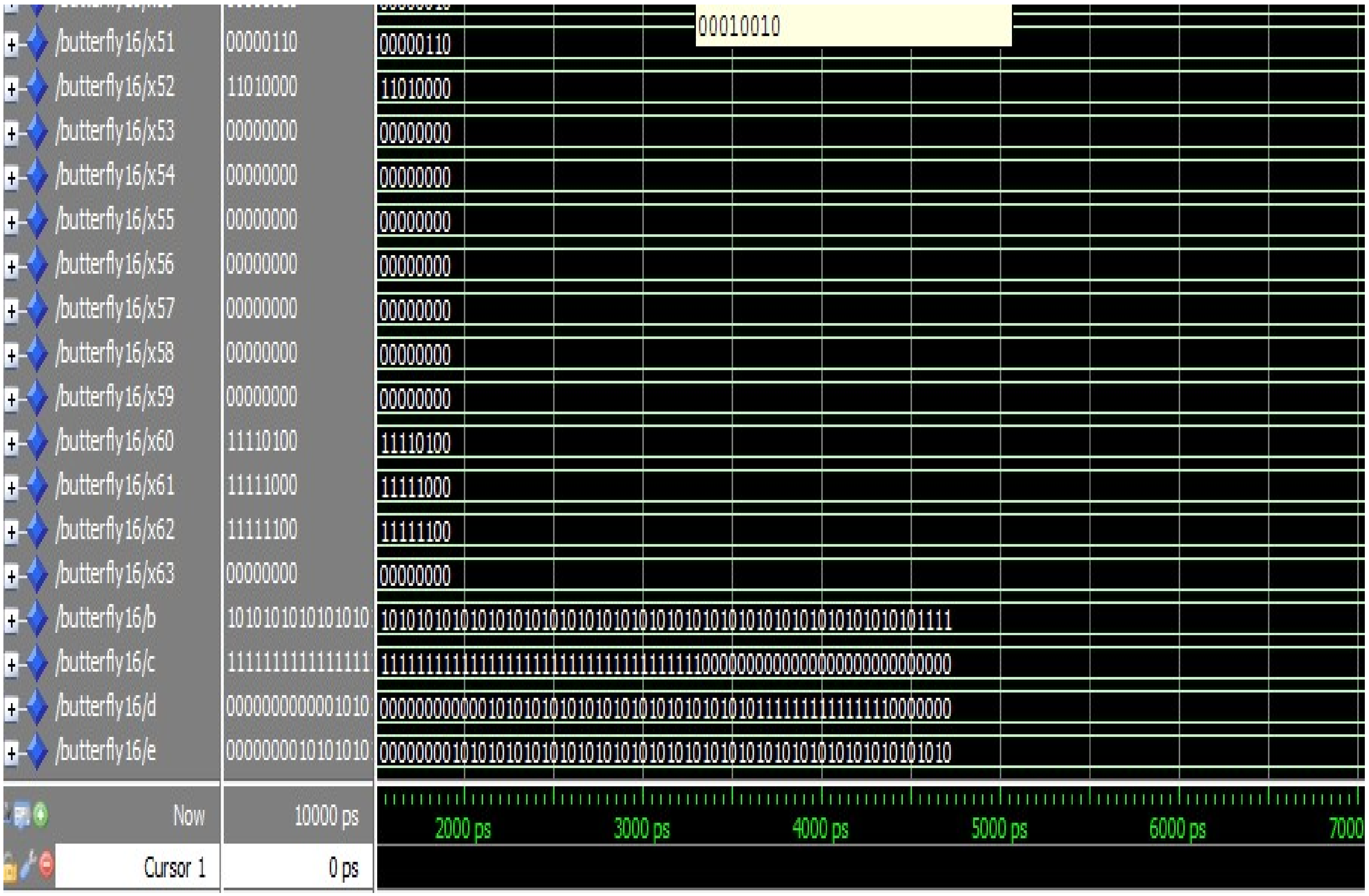
```
Messages
/butterfly16/a     1010101010101010   10101010101010101010101010101010101010101010101010101010101011111111111111111111111111111111111111
/butterfly16/tf    1111111111111111   11111111111111111111111111111000000000000000000000000000000111111111111111111111111111111000000000
/butterfly16/x0    11110100           11110100
/butterfly16/x1    11111000           11111000
/butterfly16/x2    11111100           11111100
/butterfly16/x3    00000000           00000000
/butterfly16/x4    11111111           11111111
/butterfly16/x5    00000011           00000011
/butterfly16/x6    00000111           00000111
/butterfly16/x7    11000100           11000100
/butterfly16/x8    00000000           00000000
/butterfly16/x9    00000000           00000000
/butterfly16/x10   00000000           00000000
/butterfly16/x11   00000000           00000000
/butterfly16/x12   00000000           00000000
/butterfly16/x13   00000000           00000000
/butterfly16/x14   00000000           00000000
/butterfly16/x15   00000000           00000000
/butterfly16/x16   00000110           00000110
/butterfly16/x17   00000100           00000100
/butterfly16/x18   00000010           00000010
/butterfly16/x19   00000000           00000000
/butterfly16/x20   00000000           00000000
/butterfly16/x21   00000000           00000000
/butterfly16/x22   00000000           00000000
/butterfly16/x23   00000000           00000000
/butterfly16/x24   00000000           00000000
```

Cont …..



```
/butterfly16/x25   00000000   00000000
/butterfly16/x26   00000000   00000000
/butterfly16/x27   00000000   00000000
/butterfly16/x28   00000000   00000000
/butterfly16/x29   00000000   00000000
/butterfly16/x30   11111111   11111111
/butterfly16/x31   11111111   11111111
/butterfly16/x32   00010010   00010010
/butterfly16/x33   11111110   11111110
/butterfly16/x34   00000010   00000010
/butterfly16/x35   00000110   00000110
/butterfly16/x36   00010100   00010100
/butterfly16/x37   11111000   11111000
/butterfly16/x38   00000000   00000000
/butterfly16/x39   00000000   00000000
/butterfly16/x40   00000000   00000000
/butterfly16/x41   00000000   00000000
/butterfly16/x42   00000000   00000000
/butterfly16/x43   00000000   00000000
/butterfly16/x44   00000000   00000000
/butterfly16/x45   00000010   00000010
/butterfly16/x46   11110000   11110000
/butterfly16/x47   00001100   00001100
/butterfly16/x48   00010010   00010010
/butterfly16/x49   11111110   11111110
/butterfly16/x50   00000010   00000010
/butterfly16/x51   00000110   00000110
```

Cont …..



```
/butterfly16/x51   00000110            00000110
/butterfly16/x52   11010000            11010000
/butterfly16/x53   00000000            00000000
/butterfly16/x54   00000000            00000000
/butterfly16/x55   00000000            00000000
/butterfly16/x56   00000000            00000000
/butterfly16/x57   00000000            00000000
/butterfly16/x58   00000000            00000000
/butterfly16/x59   00000000            00000000
/butterfly16/x60   11110100            11110100
/butterfly16/x61   11111000            11111000
/butterfly16/x62   11111100            11111100
/butterfly16/x63   00000000            00000000
/butterfly16/b     1010101010101010    1010101010101010101010101010101010101010101010101010101010101111
/butterfly16/c     1111111111111111    1111111111111111111111111111111111111000000000000000000000000000
/butterfly16/d     0000000000001010    0000000000001010101010101010101010101010101111111111111110000000
/butterfly16/e     0000000010101010    0000000010101010101010101010101010101010101010101010101010101010
Now                10000 ps
Cursor 1           0 ps
```

00010010

CHRONALOGICAL VIEW OF PROPOSED ALGORITHM:

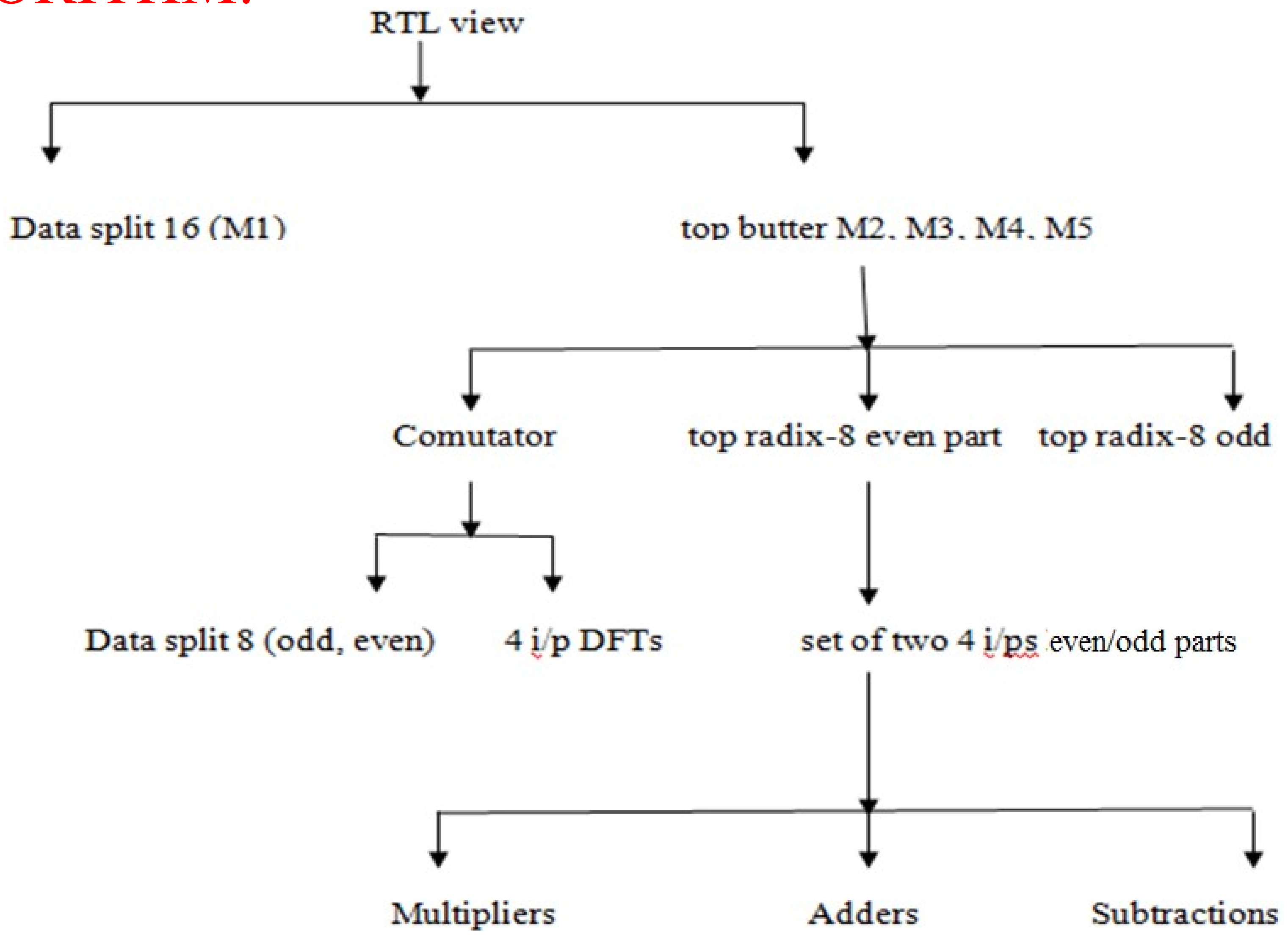


Fig: Tree diagram of functioning of proposed system

RTL VIEW OF R4 ALGORITHM:

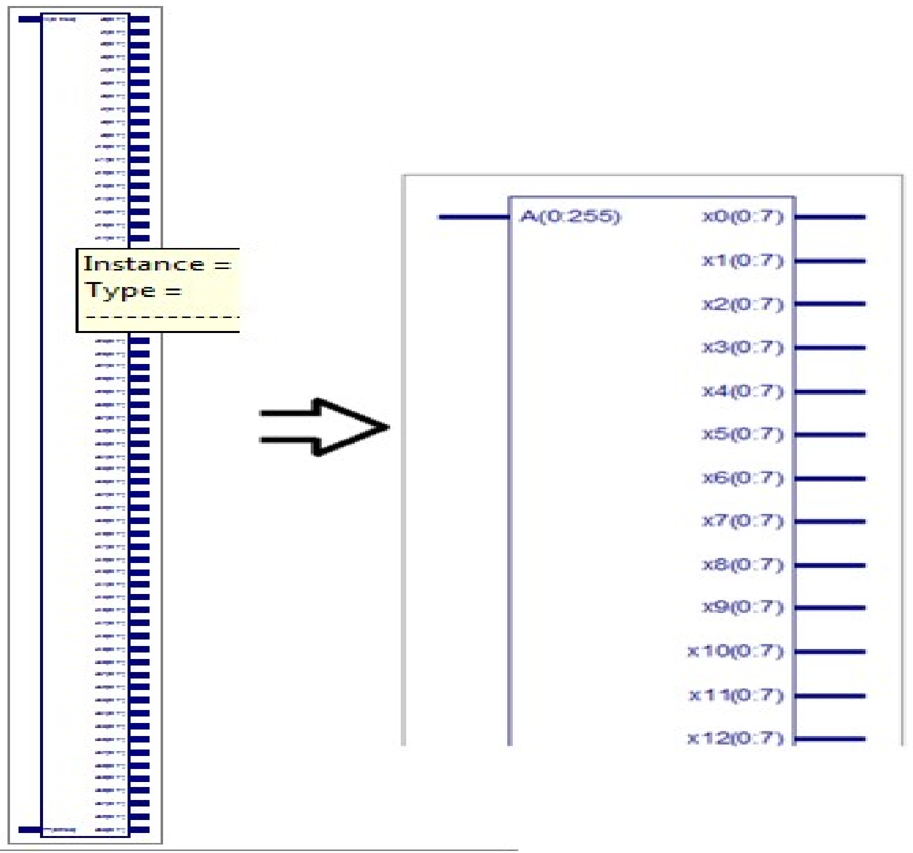


Fig: RTL view of radix-4 algorithm

DATA SPLIT OF 64-POINT:

➢A is an input sequence of 256-bits (0 to 255), divide into 4 equal parts of each 16-point, 64-bit (0 to 63, 64 to 127, 128 to 191, 192 t0 255).

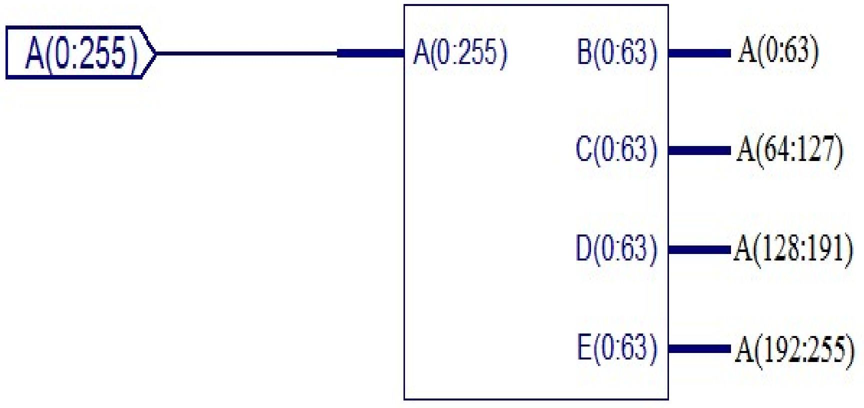


Fig. : RTL view of data split of 64-points

RTL INTERNAL VIEW:

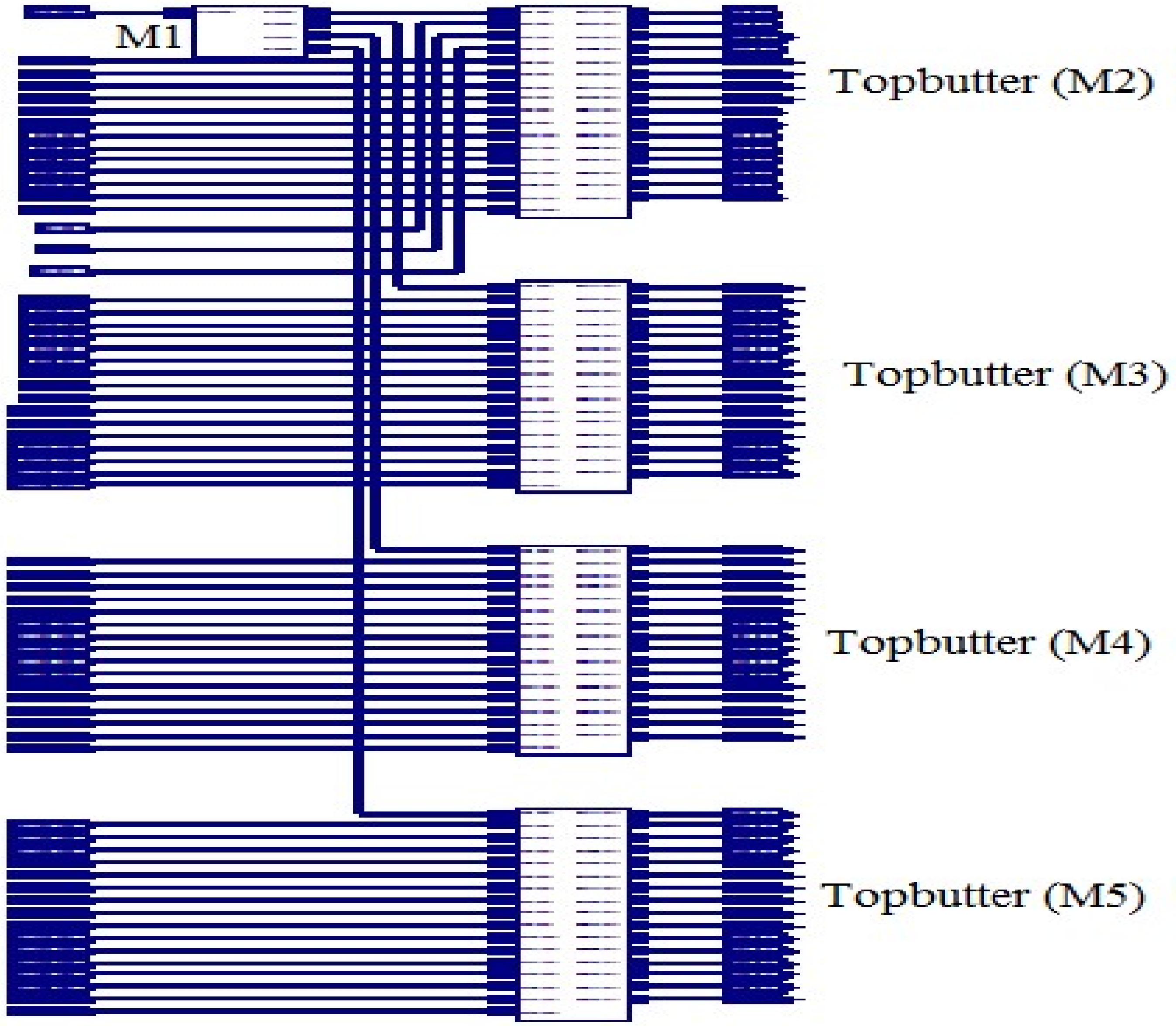


Fig: RTL internal view

OVERVIEW OF TOP BUTTER (M2):

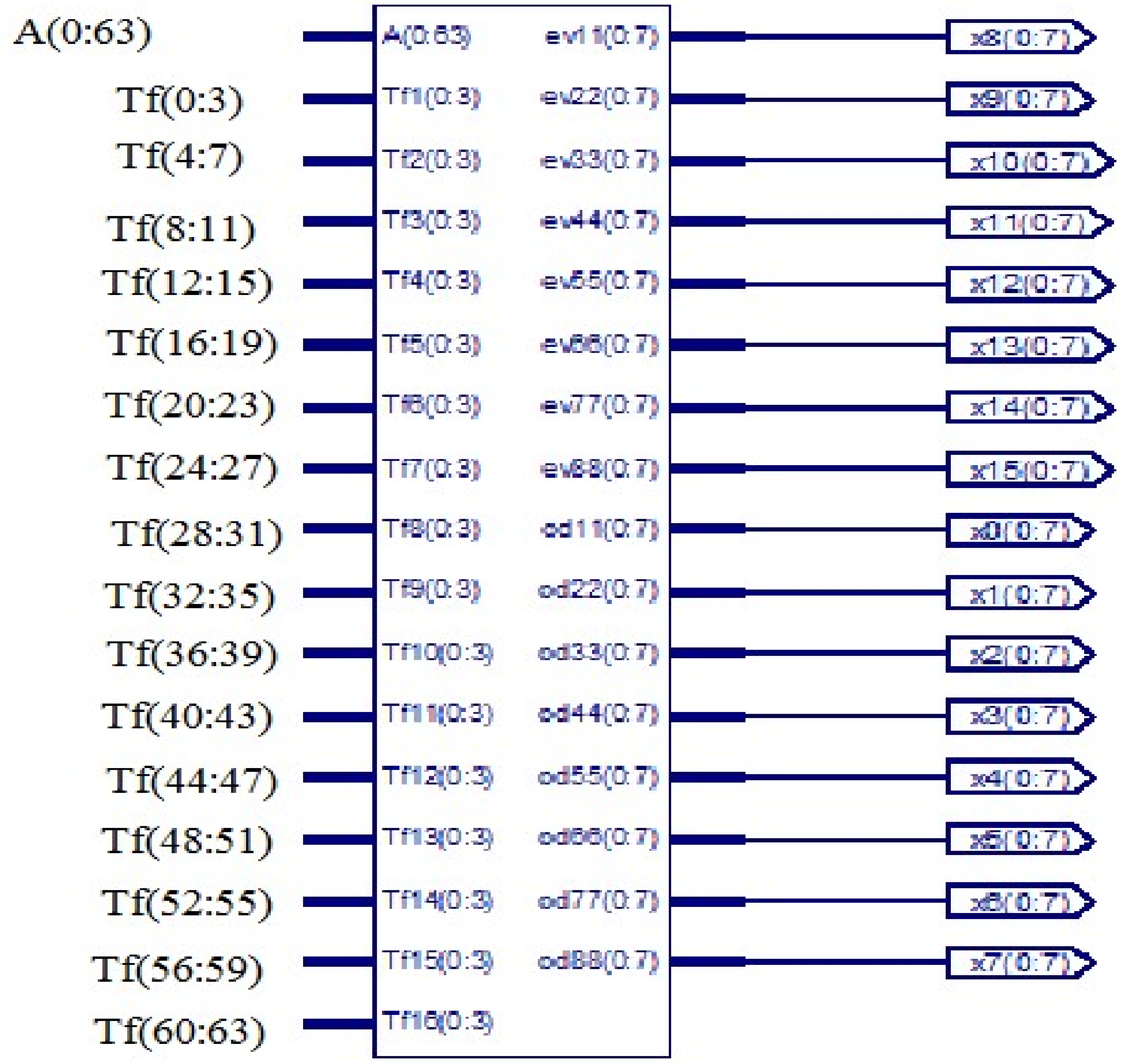


Fig: Overview of Top butter (M2).

TOP BUTTER:

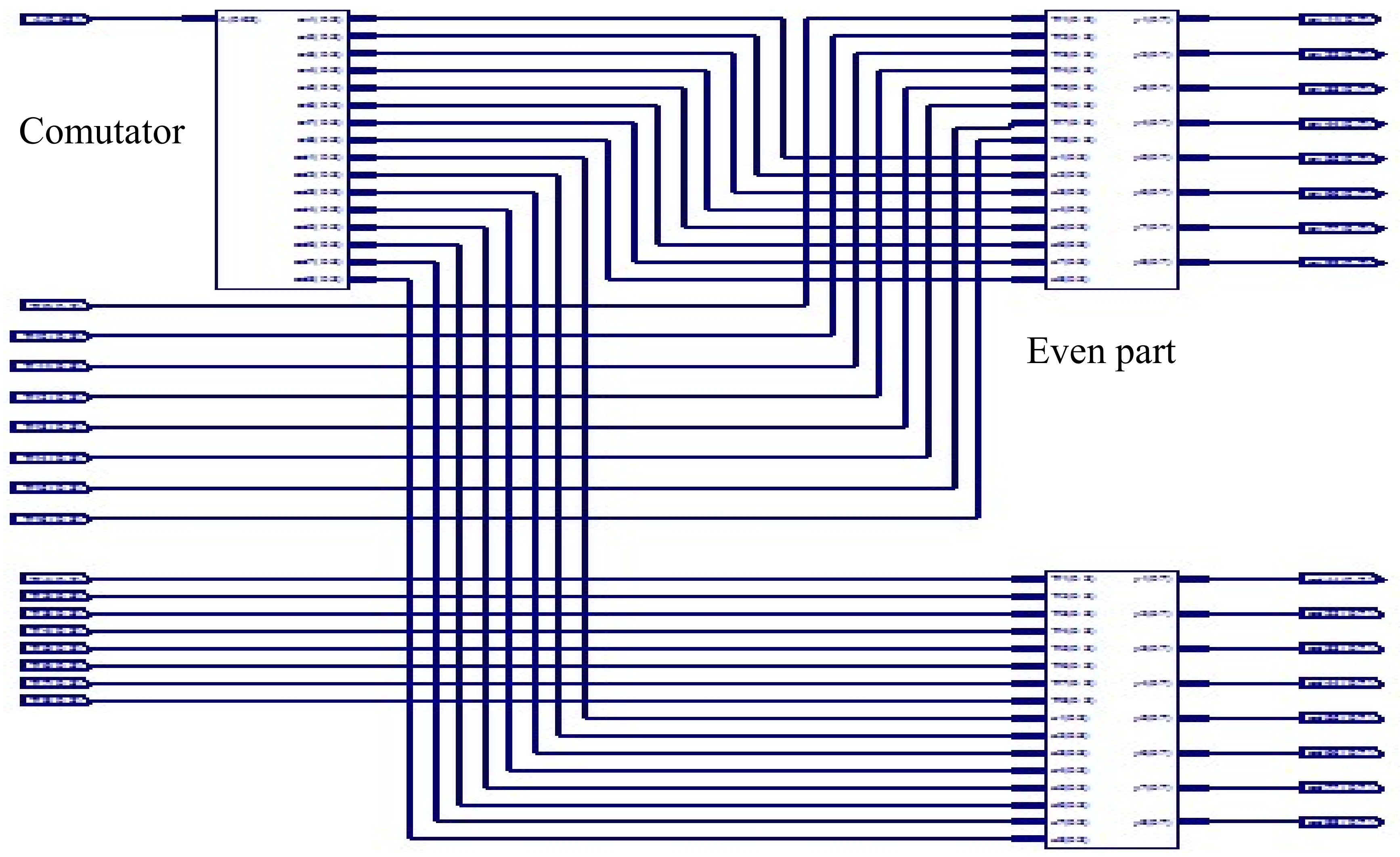


Fig: Internal view of Top butter (M2)

COMUTATOR:

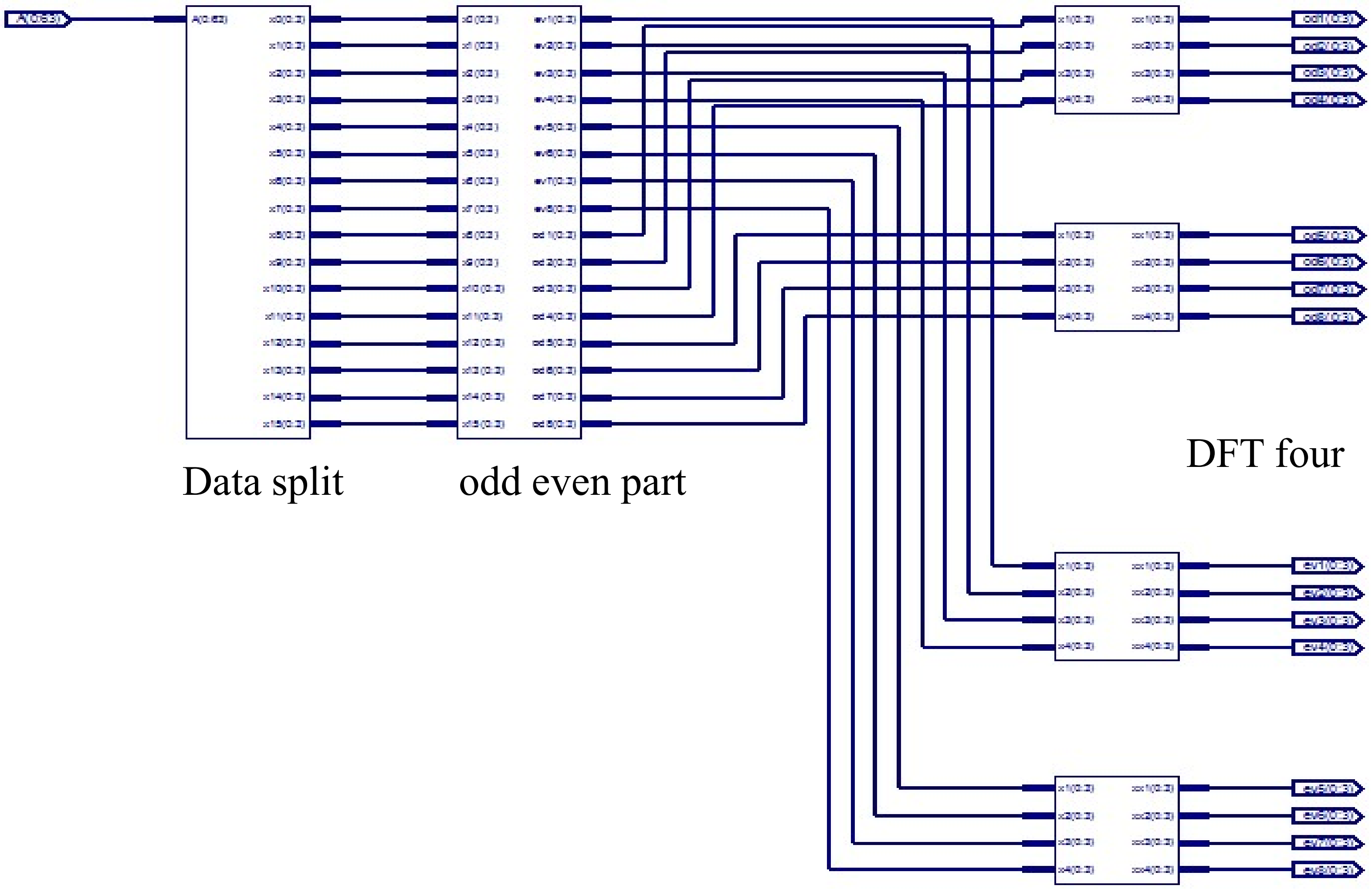


Fig: Comutator internal view

DFT FOUR:

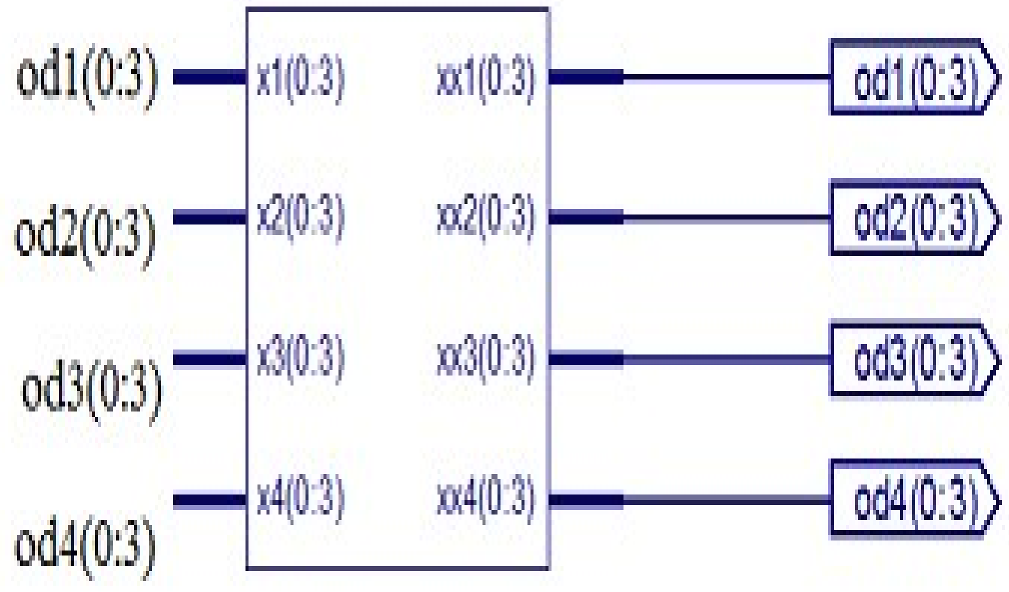


Fig: The arrangement of even and odd parts

- ➢DFT four is the basic unit in radix-4 structure.
- ➢Four inputs and four outputs

BUTTER R8:

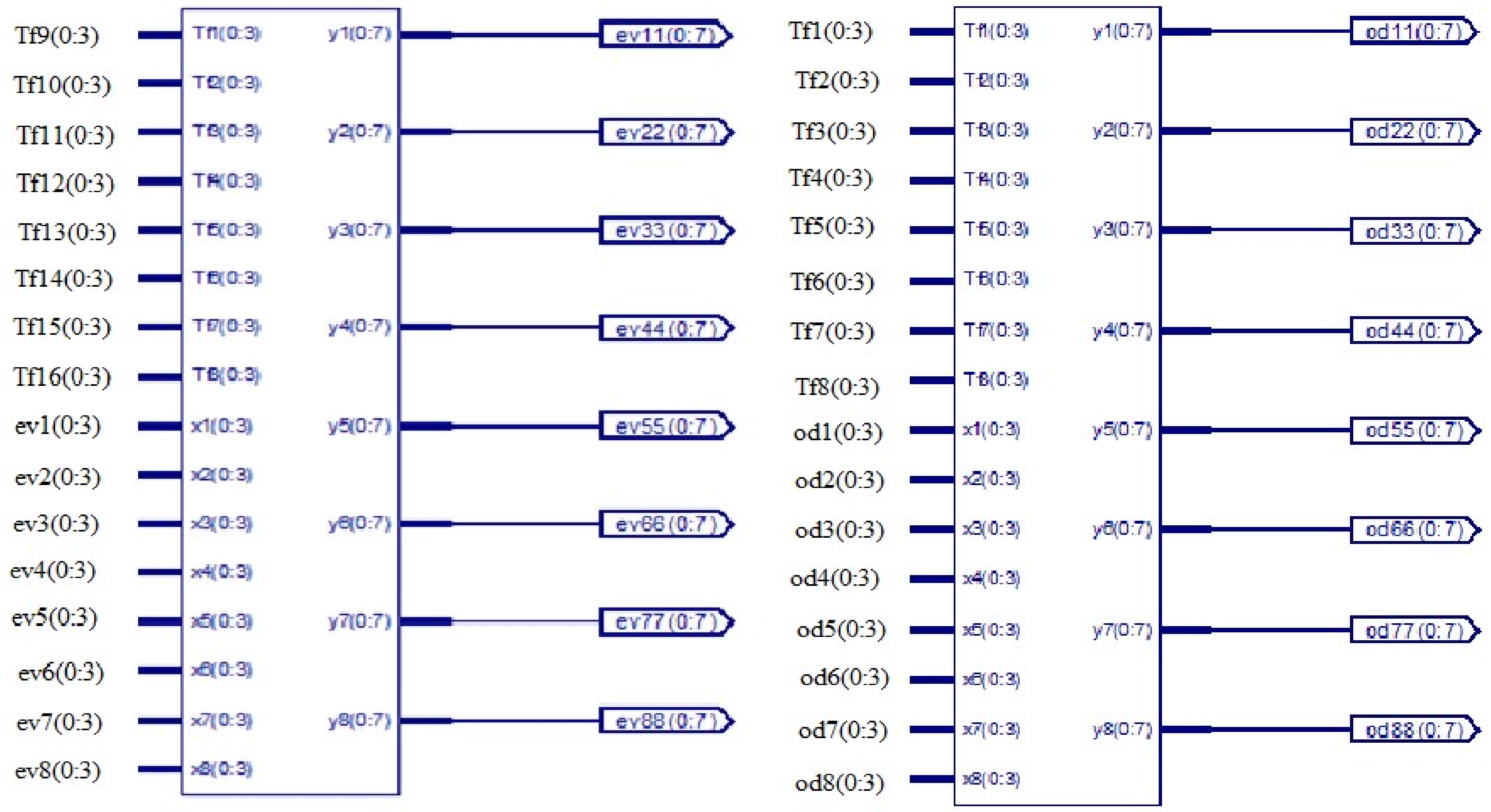


Fig: Over view of butter R8

BUTTER R4:

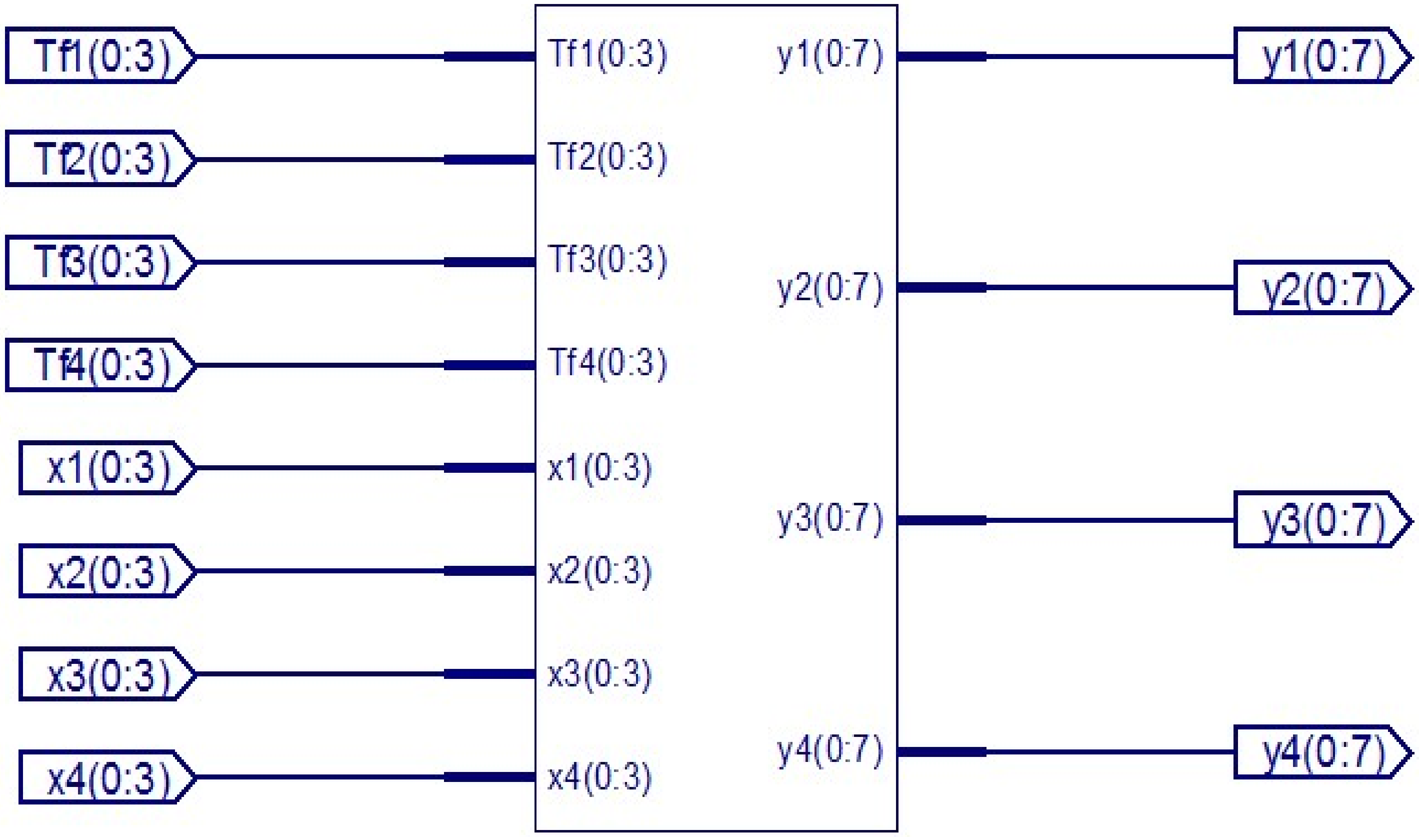


Fig: Over view of butter R4

EVEN AND ODD PARTS:

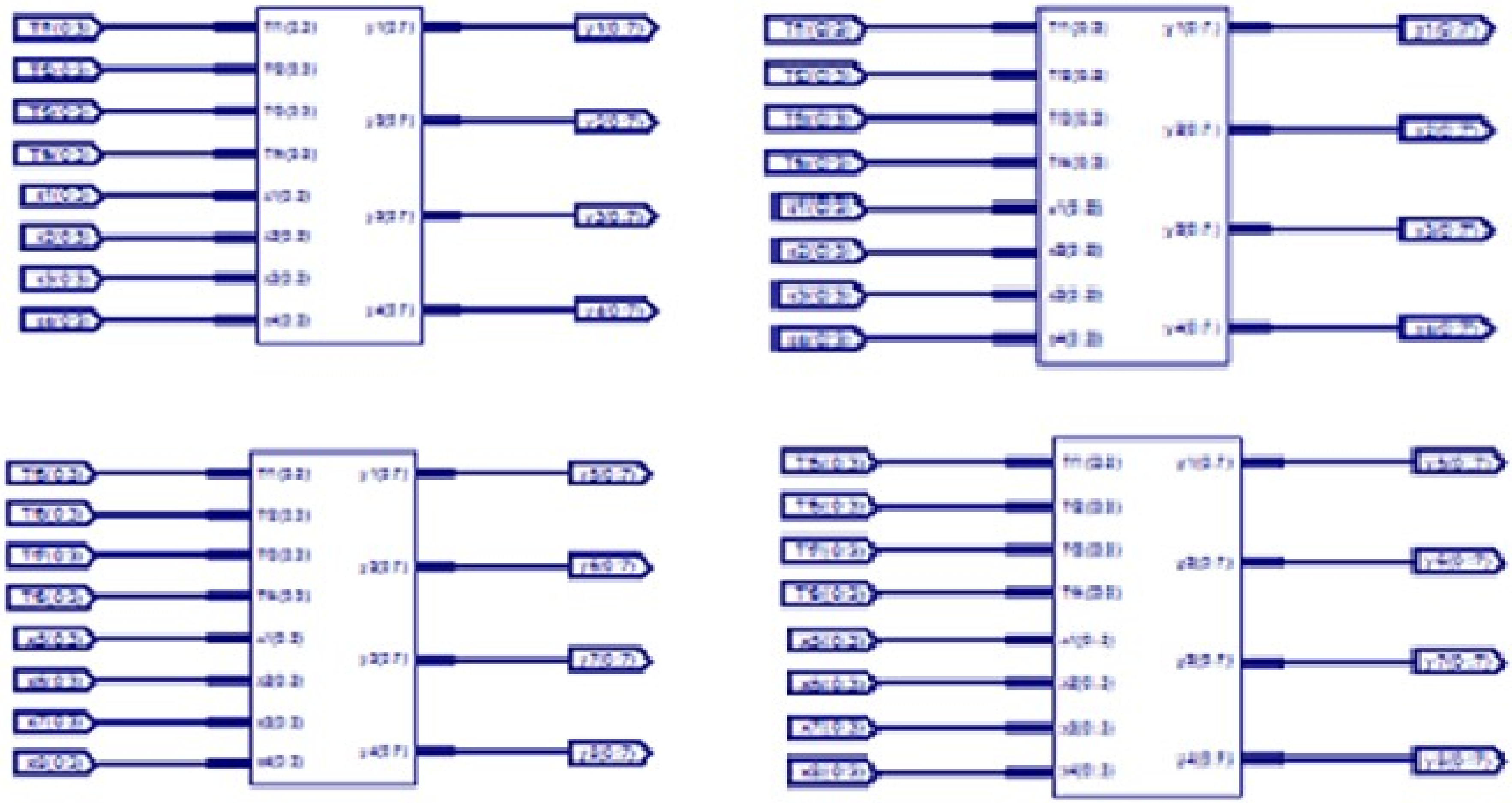
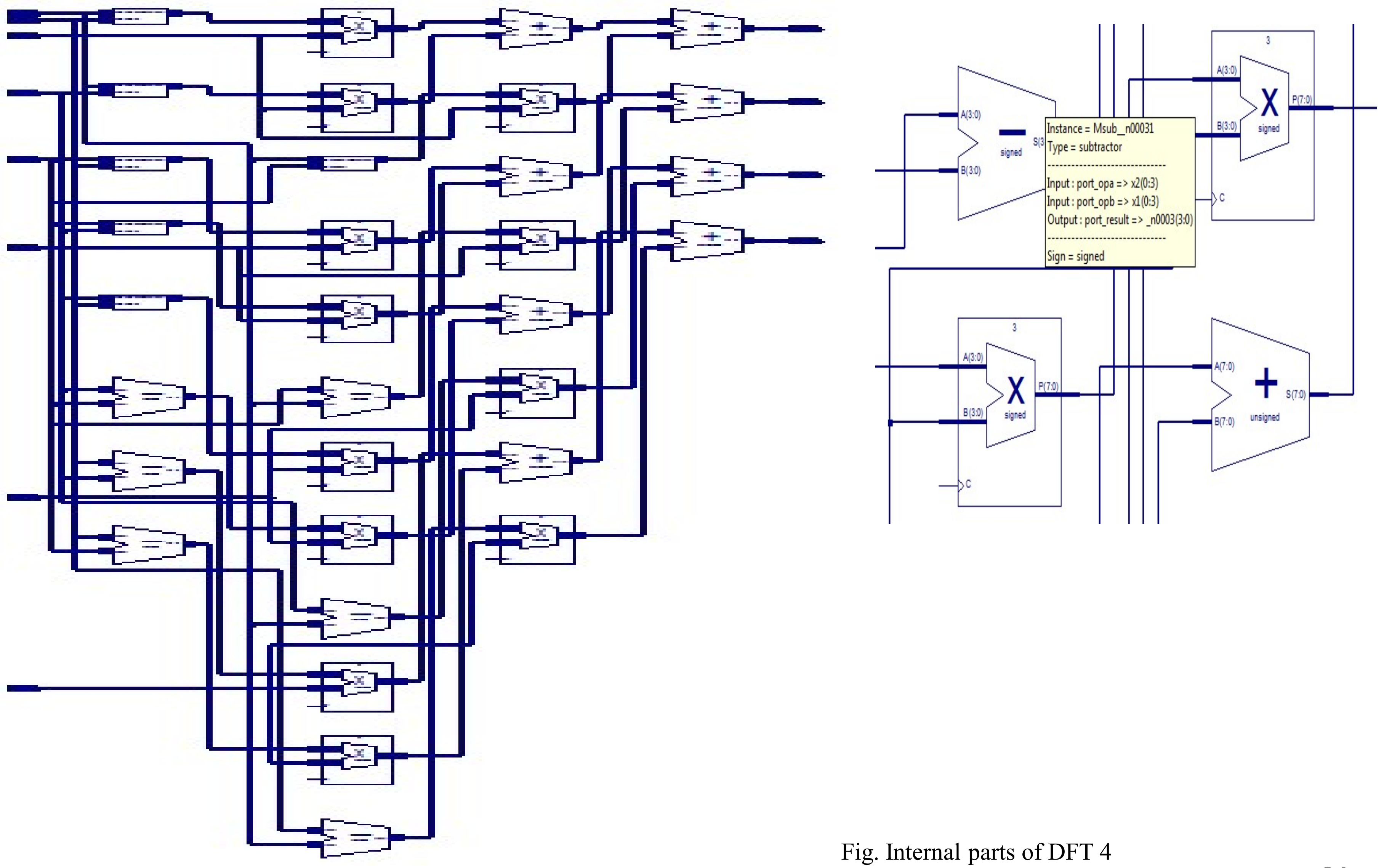


Fig: The even and odd parts

Cont ……



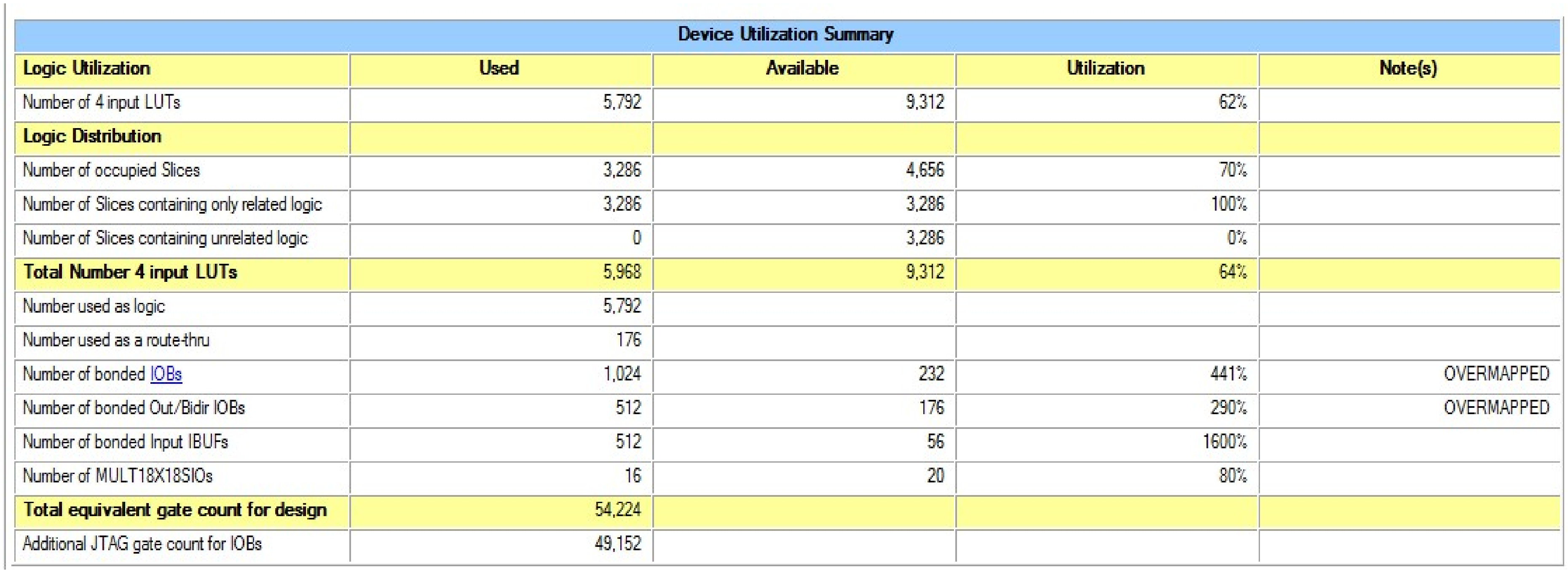
Instance = Msub_n00031
Type = subtractor

Input : port_opa => x2(0:3)
Input : port_opb => x1(0:3)
Output : port_result => _n0003(3:0)

Sign = signed

Fig. Internal parts of DFT 4

DESIGN SUMMARY:



Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of 4 input LUTs	5,792	9,312	62%	
Logic Distribution				
Number of occupied Slices	3,286	4,656	70%	
Number of Slices containing only related logic	3,286	3,286	100%	
Number of Slices containing unrelated logic	0	3,286	0%	
Total Number 4 input LUTs	5,968	9,312	64%	
Number used as logic	5,792			
Number used as a route-thru	176			
Number of bonded IOBs	1,024	232	441%	OVERMAPPED
Number of bonded Out/Bidir IOBs	512	176	290%	OVERMAPPED
Number of bonded Input IBUFs	512	56	1600%	
Number of MULT18X18SIOs	16	20	80%	
Total equivalent gate count for design	54,224			
Additional JTAG gate count for IOBs	49,152			

COMPARISON TABLE:

Table: comparison of radix-4 and radix-2 algorithm

DEVICE UTILIZATION	RADIX-2				RADIX-4			
	USED	AVAILABLE	UTILIZATION	RESULTS	USED	AVAILABLE	UTILIZATION	RESULTS
Number of Slices	2388	4656	51%		3332	4656	71%	
Number of 4 input LUTs	4282	9312	45%		5936	9312	63%	
Number of bonded IOBs	135	232	58%		1024	232	441%	Overused
Number of MULT18X18SIOs	----	----------	-------------	----------	16	20	80%	
Number of slices FFs	235	9312	2%		---	---------	------------	--------
Number of GCLKs	3	24	12%		-----	----------	-----------	--------
Minimum delay	**75.050ns** **(44.212ns logic, 30.838ns route)(58.9% logic, 41.1% route)**				**18.963ns** **(11.549ns logic, 7.414ns route) (60.9% logic, 39.1% route)**			
Total memory usage	**206720 kilobytes**				**228928 kilobytes**			

ADVANTAGES OF RADIX-4 ALGORITHM:

- In radix-2 implementation, $64=2^6$, FFT index is changed as, there are *6 Σ computations*. This corresponds to *six stages* architecture.
- If in case of radix-4 implementation, $64=4^3$, FFT index is changed as, there are *3 Σ computations*. This corresponds to *three stages* architecture.
- Complex multiplications are 75% of R2 and additions are same
- High speed and throughput

Performance Comparison

Parameter	This work	[2]	[8]	[10]	[14]
FFT Size	64-4	32-8	64-4	16-4	16-4
Delay (ns)	18.96	419	8.10	2.2	2.67
Power(mW)	12.68	739.5	33.5	3.5	4

APPLICATIONS:

- Communications
- Convolution
- Signal filtering
- Image/Data compression
- The proposed algorithm for image compression is simulated using target device xc3s500e-5fg320.
- The considered medical image has compressed using different tolerance values like 0.0007625, 0.003246, 0.013075 and 0.03924 and it also returns the drop values calculated using the formula **Drop ratio = (Total number of nonzero Fourier coefficients dropped/Total number of initially nonzero Fourier coefficients)** given as 0.10, 0.31, 0.61 and 0.83 respectively

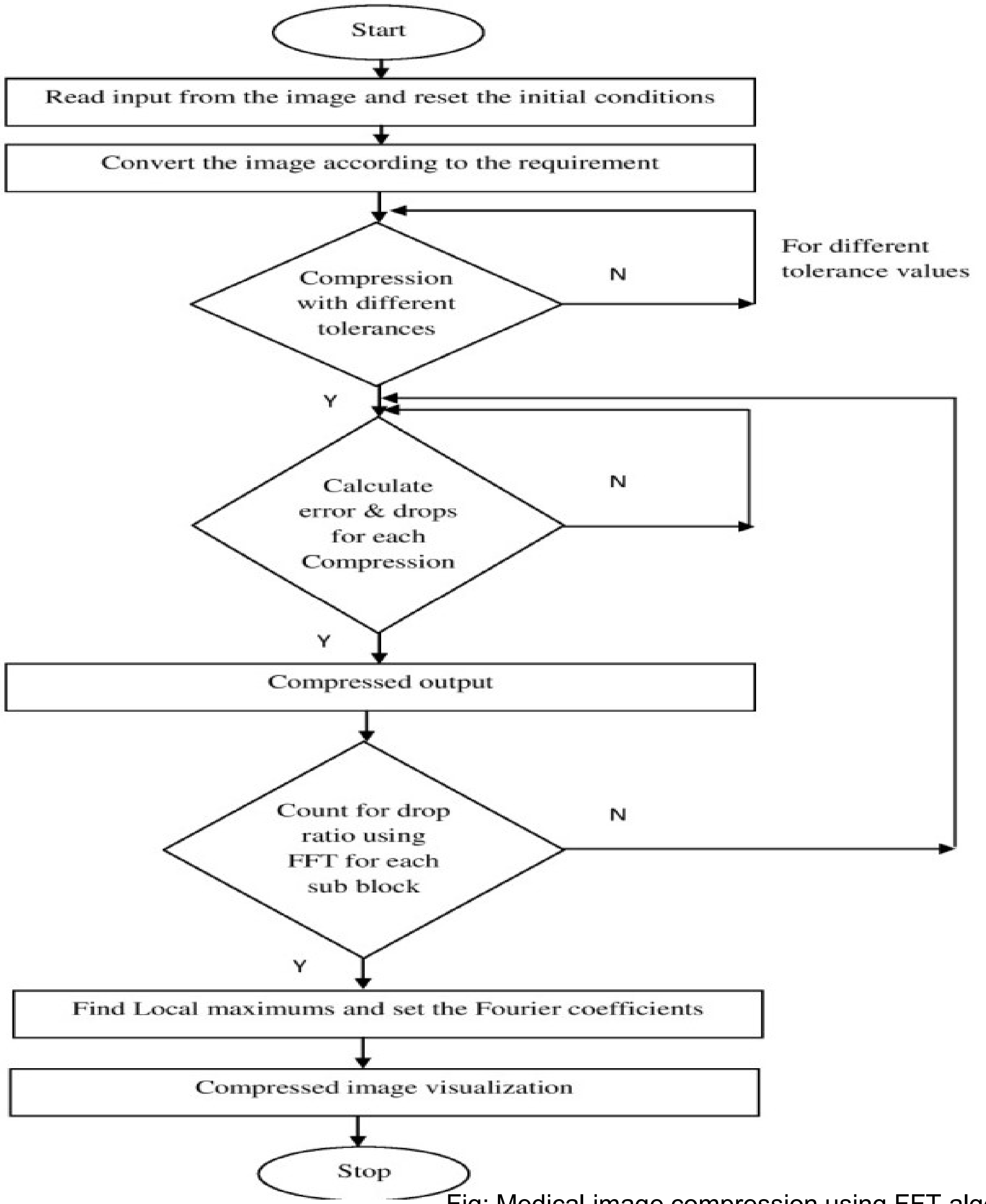

Start
Read input from the image and reset the initial conditions
Convert the image according to the requirement
Compression with different tolerances
N
For different tolerance values
Y
Calculate error & drops for each Compression
N
Y
Compressed output
Count for drop ratio using FFT for each sub block
N
Y
Find Local maximums and set the Fourier coefficients
Compressed image visualization
Stop

Fig: Medical image compression using FFT algorithm flowchart

COMPRESSION IMAGES

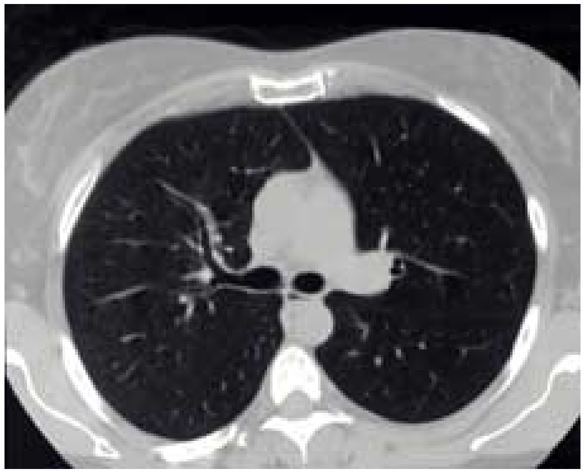


Fig: Original image

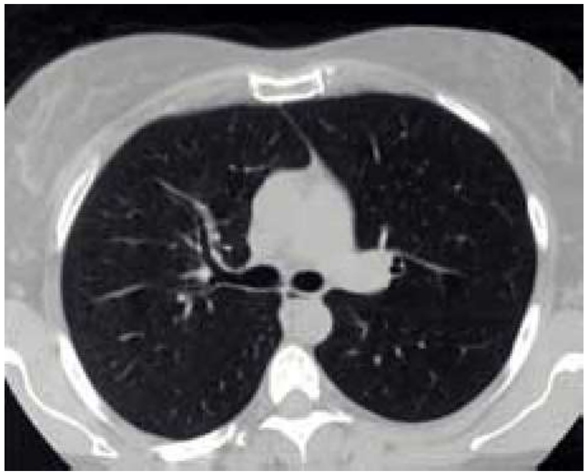


Fig: Compressed image with tolerenece=0.0007625 resulting drop ratio of 0.10

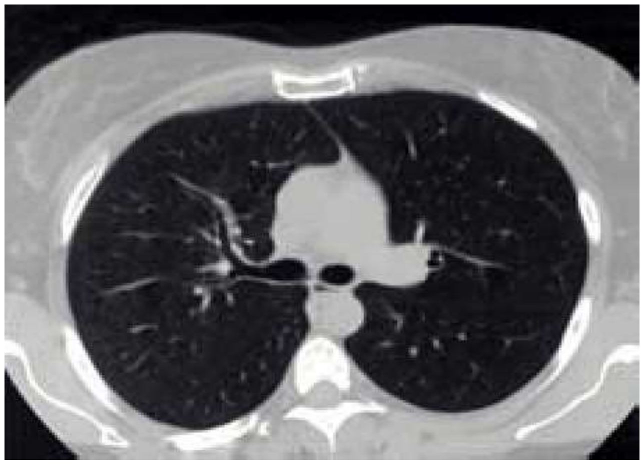


Fig: Compressed image with tolerenece=0.003246 resulting drop ratio of 0.31

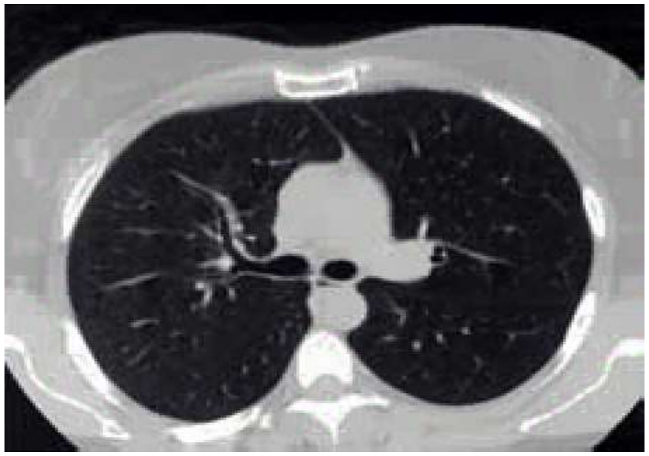
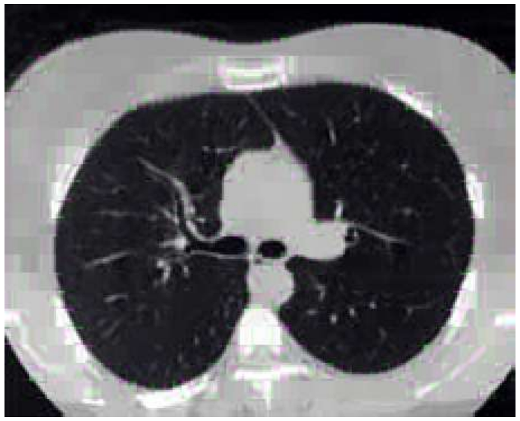


Fig: Compressed image with tolerenece=0.013075 resulting drop ratio of 0.61

Compressed image with tol = 0.03924 resulting a drop ratio of 0.83



CONCLUSION:

- Radix-4 having less delay in processing input when compared with radix-2. Regarding time delay, approximately 75% of time is saved in radix-4 algorithm. Delay time is reduced, then the fastness of the system is increased.
- The level of compression and tolerance values for medical images can be chosen based on the drop ratio and application.

LIMITATIONS AND FUTURE SCOPE

- An improvement is possible further, by minimizing the twiddle factors which in turn reduces the area, size and cost of the implementation.
- The proposed work will suitable only to the values of N in terms of power 4, it makes the limitation of the proposed work which further can be solved by using composite radix methods.

BIBLIOGRAPHY:

1. B. Shashikala, B. Sudha and S. Sarkar, Efficient Implementation of Radix-2 FFT Architecture using CORDIC for Signal Processing Applications. International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT) 2020 (pp. 137-142). IEEE.
2. Y. Jyotsna, N. Nithiyameenatchi, E. Konguvel and M. Kannan, Performance analysis of radix-2/3/5 decompositions in fixed point DIT FFT algorithms. International Conference on Computer Communication and Informatics (ICCCI), 2020, (pp. 1-7). IEEE.
3. A. Ganguly, A. Chakraborty & A. Banerjee A novel VLSI design of radix-4 DFT in current mode, International Journal of Electronics, 2019; 106(12):1845-63.
4. Sonali D. Patil, Manish Sharma. A 2048-point Split-Radix Fast Fourier Transform Computed using Radix-4 Butterfly Units, International Journal of Recent Technology and Engineering (IJRTE) 2019; 8(3):2043-46.
5. G. Ramprabu, V. Rajmohan, V. R. Prakash and N. Shankar. Analysis of Feed forward Radix-2^2 FFT 4-Parallel Architecture. International Conference on Smart Systems and Inventive Technology (ICSSIT) 2019, (pp. 168-172). IEEE.
6. S. M. Noor, E. John and M. Panday. Design and Implementation of an Ultralow-Energy FFT ASIC for Processing ECG in Cardiac Pacemakers, in IEEE Transactions on Very Large Scale Integration (VLSI) Systems 2019; 27(4):983-7,
7. Z. A. Abbas, N. B. Sulaiman, N. A. M. Yunus, W. Z. Wan Hasan and M. K. Ahmed, An FPGA implementation and performance analysis between Radix-2 and Radix-4 of 4096 point FFT. IEEE 5th International Conference on Smart Instrumentation, Measurement and Application (ICSIMA) 2018, (pp. 1-4). IEEE.
8. Anitha T G, K Vijayalakshmi, FFT Based Compression Approach For Medical Images. International Journal of Applied Engineering Research 2018; 13(6):3550-67.
9. Mario Garrido Gálvez, Miguel Angel Sanchez, Maria Luisa Lopez-Vallejo and Jesus Grajal. A 4096-Point Radix-4 Memory-Based FFT Using DSP Slices. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 2017; 25(1):375-9.
10. B. N. Mohapatra and R. K. Mohapatra. FFT and sparse FFT techniques and applications. Fourteenth International Conference on Wireless and Optical Communications Networks (WOCN), 2017, (pp. 1-5). IEEE.
11. R. H. Neuenfeld, M. B. Fonseca, E. A. C. da Costa and J. P. Oses. Exploiting addition schemes for the improvement of optimized radix-2 and radix-4 FFT butterflies. IEEE 8th Latin American Symposium on Circuits & Systems (LASCAS), 2017 (pp. 1-4). IEEE.
12. Anitha T.G, K Vijayalakshmi. Design of Novel FFT Based Image Compression Algorithms and Architectures. International Journal of Progressive Science and Technology (IJPSAT) 2017; 5(1):24-42.
13. SumeetWalia, Sachin Majithia, Adaptive Gaussian Filter Based Image Recovery Using Local Segmentation, International Journal Of Technology And Computing (IJTC) 2016; 2(1).
14. R. Neuenfeld, M. Fonseca and E. Costa, Design of optimized radix-2 and radix-4 butterflies from FFT with decimation in time, IEEE 7th Latin American Symposium on Circuits & Systems (LASCAS), 2016, (pp. 171-174). IEEE.
15. Z. Qian and M. Margala, "Low-Power Split-Radix FFT Processors Using Radix-2 Butterfly Units," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2016; 24(9):3008-12.
16. Z.-G. Ma, X.-B. Yin, and F. Yu, A novel memory-based FFT architecture for real-valued signals based on a radix-2 decimation-in-frequency algorithm, *IEEE Trans. Circuits Syst. II, Exp. Briefs*, 2015; 62(9):876–80.
17. K. Jayaram and C. Arun. Survey report for Radix-2, Radix-4 and Radix-8 FFT Algorithms, in International Journal of Innovative Research in Science, Engineering and Technology 2015; 4(7):5149-54.
18. Brundavani P. FPGA Implementation of 256-Bit, 64-Point DIT-FFT Using Radix-4 Algorithm, in International Journal of Advanced Research in Computer Science and Software Engineering, 2015; 3(9):126- 33.
19. Zhuo Qian, Nasibeh Nasiri, Oren Segal, and Martin Margala. FPGA implementation of low-power split-radix fast fourier transform processors, in Proc. 24th International Conference. Field Program. Logic Applications. Munich, Germany, 2014, (pp. 1–2). IEEE.
20. Amarnath Reddy and Venkata Suman, Design and Simulation of FFT Processor Using Radix-4 Algorithm Using FPGA, International Journal of Advanced Science and Technology, 2013; 61:53-62.