KLE Technological University | Creating Value, Leveraging Knowledge

Dr. M. S. Sheshgiri Campus, Belagavi

Department of Electronics & Communication Engineering

Architecture Design of Integrated Circuit

23EECE302

Course project

"8 Point FFT using Verilog HDL"

Submitted by:

Sl. No.	Name	SRN	Signature
1	Basavesh Patil	02FE21BEC018	
2	Pratik Patil	02FE21BEC063	

Submitted for the partial fulfillment of the course

Mentored by:

Dr. Kunjan D. Shinde

Assistant Professor, Dept. of E&CE,

KLE Technological University, Dr. M S Sheshgiri Campus, Belagavi.

Academic Year 2023-24

Contents

1. Problem Statement:

An 8-point FFT (Fast Fourier Transform) is a computational algorithm used to quickly calculate the discrete Fourier transform of a sequence of 8 complex numbers. It efficiently breaks down a time-domain signal into its frequency components, allowing for analysis of its frequency spectrum

2. Introduction:

The main purpose in digital signal processing is to reduce the requirements of hardware and increased the computation. The Fast Fourier transform (FFT) is an algorithm which samples the signal over some space and splits them into frequency points. This method is used to get the discrete Fourier transform of the signal and it converts the signal into its frequency domain. It has advantage over DFT because it reduces the complexity of DFT from O(N^2) to O(NlogN) where N represents the data points. The fast method to compute the DFT (Discrete Fourier transform) is the FFT (fast Fourier transform). It has many applications like spectral analysis, matched filtering, image processing and disease extractions. When FFT was developed the real time digital signal become realized in many fields such as radar and communications. There are many situations where FFT computation is required to face the real-time data like medical diagnosis and earthquake monitoring and so on . The detailed information about spectrum of signal in frequencies and amplitude would be difficult to analyze in time domain. Due to this reason the signal would be converted into frequency domain to get the complete information. In IoMT (Internet of Medical Things) FFT plays an important role. As EEG and ECG are highly noisy and difficult for processing. That’s why extraction information form this signal is the main issue. To extract the detailed and meaningful information from noisy signal in order to get the best analytics FFT is used. It converts the noisy data into desired information and reduces the noise level. It provides the detailed information in frequency domain.

3. Objectives & Methodology:

1. Implement an 8-point DIT FFT processor in Verilog.
2. Verify the functionality and accuracy of the processor through simulation and test bench development.
3. Analyze waveform outputs to ensure correct signal processing and timing.
4. Verifying the values in the console window

Methodology:

1. There are two types of division known division in time and division in frequency
2. The division in time requires bit reversed output and produces normal output order while division is frequency takes normal input order and generates bit reversed output
3. Here we using the division in time method:
   Decimation In Time(DIT) FFT algorithm rearranges the DFT formula into 2 parts, as a sum of odd and even parts.

$$\mathbf{X(K)} = \sum_{n=0}^{N-1} \mathrm{x(n)} e^{-(\frac{i2nk}{n})}$$

$$= \sum_{n=0}^{\frac{\mathrm{N}}{2}-1} \mathrm{x(2n)} e^{-(\frac{i(2r)X(2n)k}{n})} + \sum_{n=0}^{N/2-1} \mathrm{x(2n+1)} e^{-(\frac{i2r(2n+1)k}{n})}$$

$$= \sum_{n=0}^{\frac{\mathrm{N}}{2}-1} \mathrm{x(2n)} e^{-(2rnk/(\frac{N}{2}))} + \sum_{n=0}^{N/2-1} \mathrm{x(2n+1)} e^{-(2rnk/(\frac{N}{2}))}$$

4. Design/ Architectures :

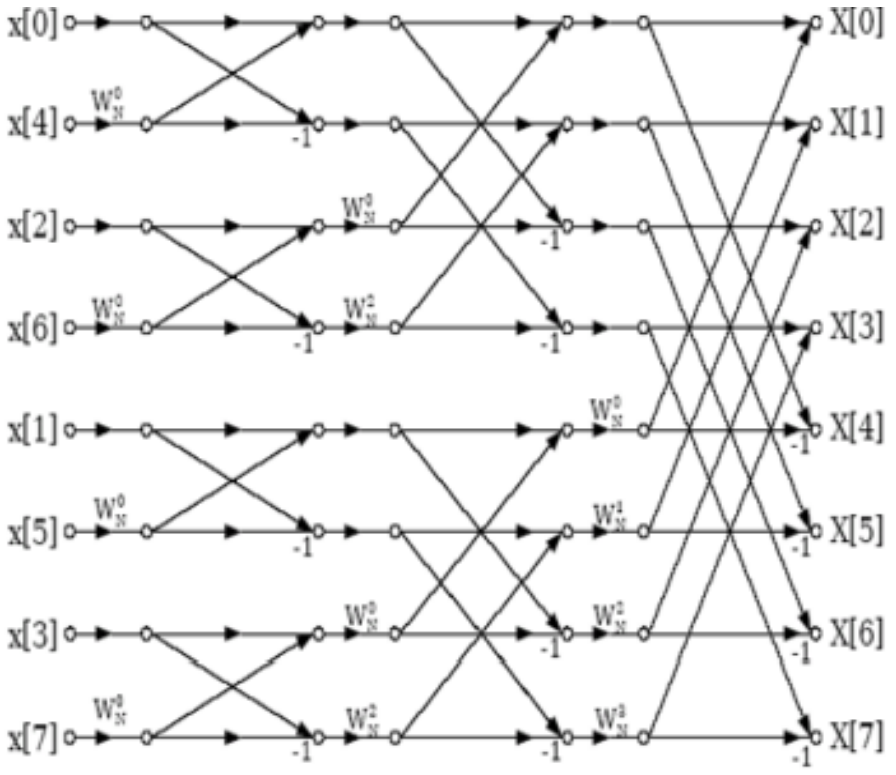


Fig (a) Flow Graph of 8-Point DIT FFT Algorithm

3 stages to construct an 8-point DFT using Radix-2 FFT algorithm:

- STAGE 1: Consists of 4 butterflies. Each butterfly has 2 inputs and two outputs. The inputs are given after the bit reversal of the input sequence.

- STAGE 2: The input samples to each butterfly are separated by N/4 samples i.e., 2 samples and there are two sets of butterflies. In each set of butterflies, the twiddle factor exponents are the same and separated by two.

- STAGE 3: This stage includes decomposing of N/4 points transforms into N/8 points transforms.

Bit Reversal:

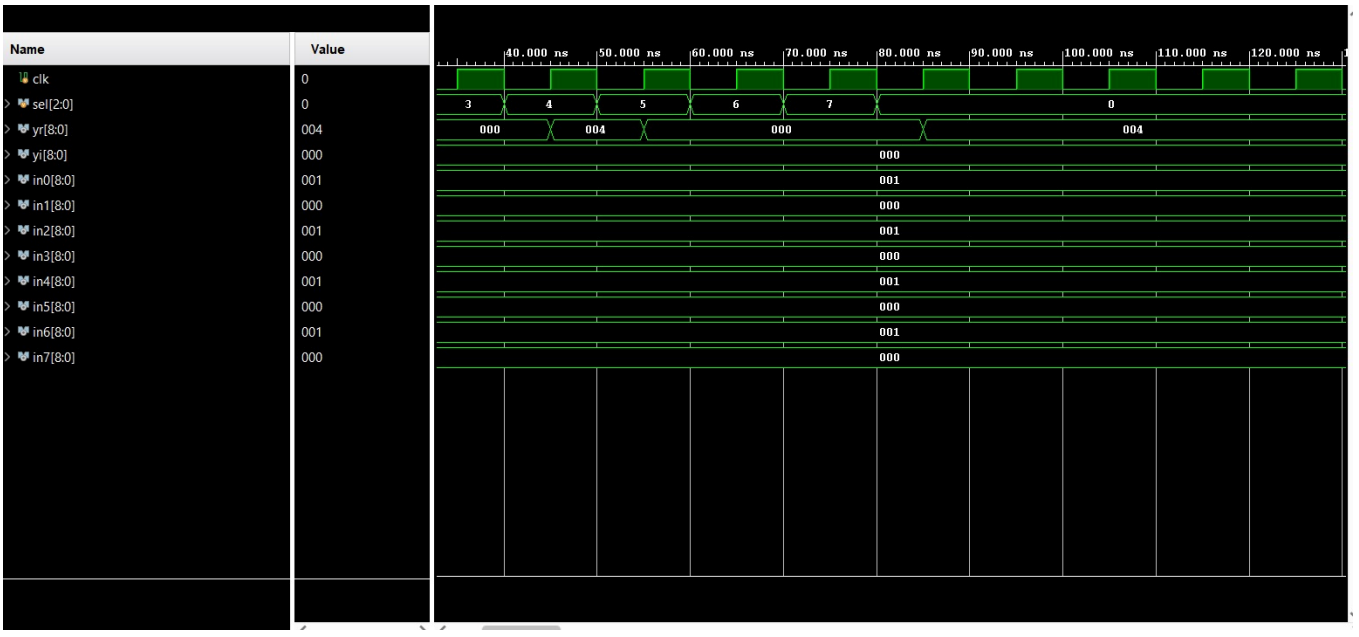
BIT REVERSED BINARY(INPUT)	BINARY(OUTPUT)
000 x(0)	000 X(0)
100 x(4)	001 X(1)
010 x(2)	010 X(2)
110 x(6)	011 X(3)
001 x(1)	100 X(4)
101 x(5)	101 X(5)
111 x(7)	111 X(7)

Design summary:

1. The number of input samples N = 2M , where, M is an integer.
2. The input sequence is shuffled through bit-reversal.
3. The number of stages in the flowgraph is given by M=log2N.
4. Each stage consists of N/2 butterflies.
5. Inputs/outputs of each butterfly are separated by 2^(m-1) samples,where m represents the stage index, i.e., for first stage m=1 and for the second stage m=2 so on.
6. The number of complex multiplications is given by (N/2)**g2** ***N***.
7. The number of complex additions is given by N*log2N.
8. The number of sets or sections of butterflies in each stage is given by the formula 2^(M-m). 10)The exponent repeat factor (ERF), which is the number of times the exponent sequence associated with m is repeated is given by 2^(M-n).

5. Results and Discussions

a. Simulation Results



Input(1,0,1,0,1,0,1,0,1) = output(4,0,0,0,4,0,0,0

```
At time 10ns:
x(0) = 1
y(0) = 4 + 0i
At time 20ns:
x(1) = 0
y(1) = 0 + 0i
At time 30ns:
x(2) = 1
y(2) = 0 + 0i
At time 40ns:
x(3) = 0
y(3) = 0 + 0i
At time 50ns:
x(4) = 1
y(4) = 4 + 0i
At time 60ns:
x(5) = 0
y(5) = 0 + 0i
At time 70ns:
x(6) = 1
y(6) = 0 + 0i
At time 80ns:
x(7) = 0
y(7) = 0 + 0i
INFO: [USF-XSim-96] XSim completed. Design snapshot 'testbench_be
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:02 ; elapsed = 00:00:14
```

b. Comparative Analysis

Parameters	Non pipelined	Pipelined
Time	4.467ns	3.491ns
Power	0.391W	0.390W

Non pipelined :

```
Timing Report

Slack:                    inf
  Source:                 yr_reg[2]/C
                            (rising edge-triggered cell FDRE)
  Destination:            yr[2]
                            (output port)
  Path Group:             (none)
  Path Type:              Max at Slow Process Corner
  Data Path Delay:        4.467ns  (logic 2.765ns (61.903%)  route 1.702ns (38.097%))
  Logic Levels:           2  (FDRE=1 OBUF=1)

    Location             Delay type                Incr(ns)  Path(ns)    Netlist Resource(s)
  -------------------------------------------------------------------    -------------------
    SLICE_X0Y101         FDRE                         0.000     0.000 r  yr_reg[2]/C
    SLICE_X0Y101         FDRE (Prop_fdre_C_Q)         0.379     0.379 r  yr_reg[2]/Q
                         net (fo=1, routed)           1.702     2.081    yr_OBUF[2]
    U18                  OBUF (Prop_obuf_I_O)         2.386     4.467 r  yr_OBUF[2]_inst/O
                         net (fo=0)                   0.000     4.467    yr[2]
    U18                                                         r  yr[2] (OUT)
  -------------------------------------------------------------------    -------------------
```

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.391 W**
Design Power Budget:	**Not Specified**
Process:	typical
Power Budget Margin:	**N/A**
Junction Temperature:	**26.3°C**
Thermal Margin:	73.7°C (21.6 W)
Ambient Temperature:	25.0 °C
Effective ϑJA:	3.3°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

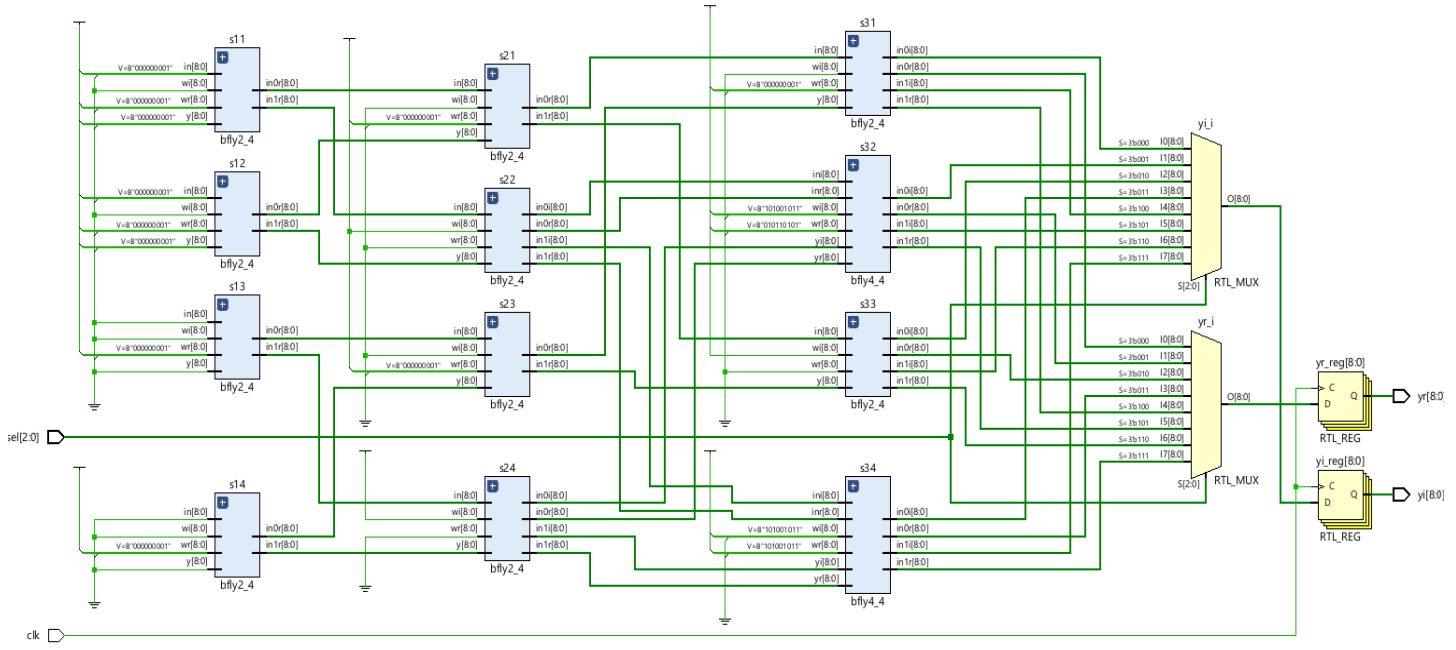
Pipelined:

```
Timing Report

Slack:                    inf
  Source:                 yr_reg[2]/C
                            (rising edge-triggered cell FDRE)
  Destination:            yr[2]
                            (output port)
  Path Group:             (none)
  Path Type:              Max at Slow Process Corner
  Data Path Delay:        3.491ns  (logic 2.846ns (81.509%)  route 0.646ns (18.491%))
  Logic Levels:           2  (FDRE=1 OBUF=1)

    Location             Delay type                Incr(ns)  Path(ns)    Netlist Resource(s)
  -------------------------------------------------------------------    -------------------
                         FDRE                         0.000     0.000 r  yr_reg[2]/C
                         FDRE (Prop_fdre_C_Q)         0.379     0.379 r  yr_reg[2]/Q
                         net (fo=1, unplaced)         0.646     1.025    yr_OBUF[2]
                         OBUF (Prop_obuf_I_O)         2.467     3.491 r  yr_OBUF[2]_inst/O
                         net (fo=0)                   0.000     3.491    yr[2]
                                                                      r  yr[2] (OUT)
  -------------------------------------------------------------------    -------------------
```

c. Implementation



6. Advantages and applications

- **Digital Signal Processing (DSP):** FFT is extensively used in DSP applications for spectrum analysis, filtering, convolution, correlation, and noise reduction. For example, in telecommunications, FFT is used for channel equalization
- **Audio and Speech Processing:** FFT is used in audio spectrum analysis, audio compression (e.g., MP3 encoding), and various audio effects processing (e.g., equalization, echo cancellation).
- **Biomedical Signal Analysis:** FFT is used for analyzing EEG (electroencephalogram) signals, ECG (electrocardiogram) signals, and other biomedical signals to extract frequency domain information for diagnostic purposes.

Advantages:

- **Speed**: FFT algorithms, including DIT FFT, provide faster computation of discrete Fourier transforms compared to the direct computation methods, especially for large datasets. In the case of an 8-point FFT, the advantage in speed may not be as pronounced as with larger FFT sizes, but it still provides computational efficiency.
- **Resource Efficiency**: Hardware implementations of FFT algorithms can be optimized to utilize FPGA resources efficiently, balancing speed and resource consumption. Vivado provides tools to help optimize these implementations for specific FPGA architectures.

7. Conclusion:

Designing an 8-point DIT FFT using Verilog HDL in Vivado offers significant advantages in terms of computational speed, resource efficiency, and accuracy compared to traditional methods. This hardware implementation enables real-time signal processing crucial for applications in telecommunications, audio processing, radar systems, biomedical signal analysis, and more. The FFT algorithm's ability to efficiently compute frequency domain representations makes it indispensable in digital signal processing tasks such as spectrum analysis, filtering, and modulation/demodulation. Moreover, optimizing FPGA resources through Vivado ensures that FFT implementations are tailored to specific application requirements, enhancing overall performance and reliability in diverse technological fields

8. References :

- https://github.com/Devashrutha/FFT-using-Verilog-RADIX-2?tab=readme-ov-file
- Magsi, Hina, and Ali Hassan Sodhro. "FAST FOURIER TRANSFORM BY VERILOG."
- Kangralkar, Sonali, and Rajashri Khanai. "Design and Implementation of 8-point FFT using Verilog HDL." International Journal of Computer Applications 177.11 (2019): 4-6.