ELD Lab 7
Exploring AXI-FFT IP

Objective

- Understand the basics of Fourier Transform and AXI-FFT IP
- Design floating point FFT for input of size 8 elements
- **Lab Homework 1:** Design the floating point FFT for input of size 32 elements
- **Lab Homework 2:** Design the floating point arithmetic circuit to implement following equation

$$z = \frac{x}{y} + \sqrt{\frac{2 * \ln t}{y}}$$

- Here x, y and t are positive integers and t > y > x.

Theory

Fourier Transform (FT)

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-\frac{2\pi kn}{N} i}$$

- It is basically matrix vector multiplication

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} e^{-\frac{2\pi 0.0}{4} i} & e^{-\frac{2\pi 0.1}{4} i} & e^{-\frac{2\pi 0.2}{4} i} & e^{-\frac{2\pi 0.3}{4} i} \\ e^{-\frac{2\pi 1.0}{4} i} & e^{-\frac{2\pi 1.1}{4} i} & e^{-\frac{2\pi 1.2}{4} i} & e^{-\frac{2\pi 1.3}{4} i} \\ e^{-\frac{2\pi 2.0}{4} i} & e^{-\frac{2\pi 2.1}{4} i} & e^{-\frac{2\pi 2.2}{4} i} & e^{-\frac{2\pi 2.3}{4} i} \\ e^{-\frac{2\pi 3.0}{4} i} & e^{-\frac{2\pi 3.1}{4} i} & e^{-\frac{2\pi 3.2}{4} i} & e^{-\frac{2\pi 3.3}{4} i} \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix}$$

Fourier Transform (FT)

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-\frac{2\pi k n}{N} i}$$

- It is basically matrix vector multiplication

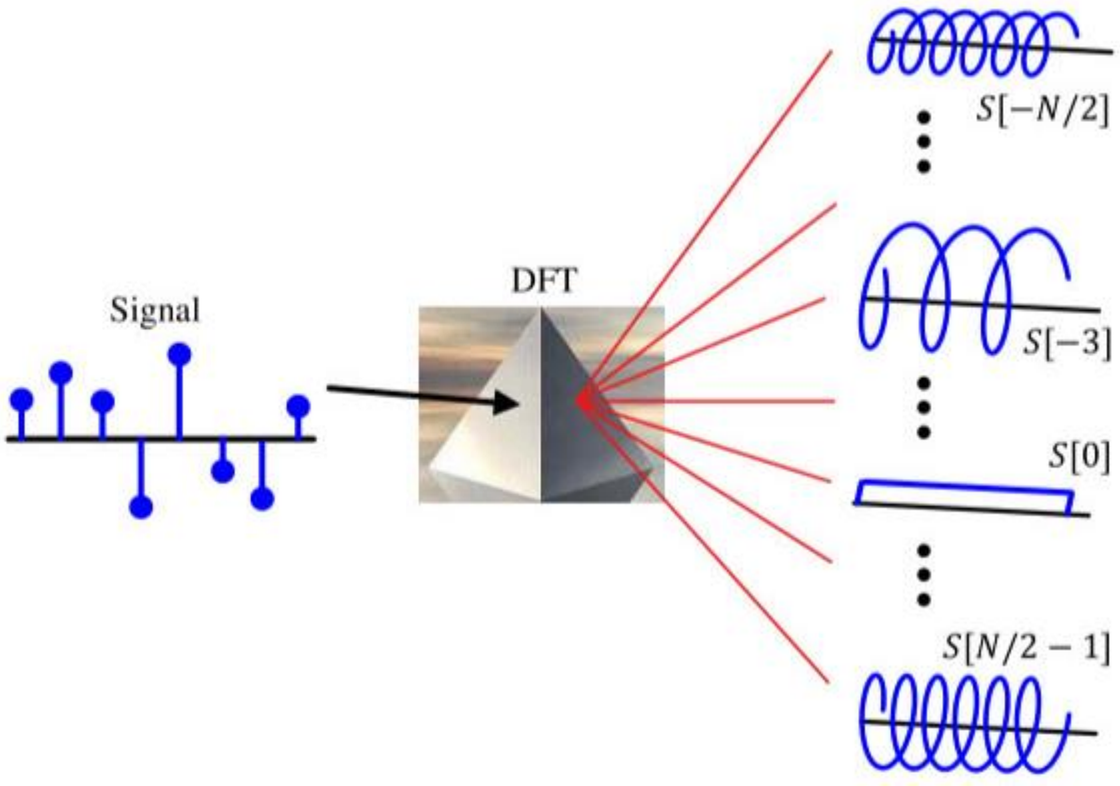
$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & i \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix}$$

8-point FT

$$
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & \frac{1-j}{\sqrt{2}} & -j & \frac{-(1+j)}{\sqrt{2}} & -1 & \frac{-(1-j)}{\sqrt{2}} & j & \frac{1+j}{\sqrt{2}} \\
1 & -j & -1 & j & 1 & -j & -1 & j \\
1 & \frac{-(1+j)}{\sqrt{2}} & j & \frac{1-j}{\sqrt{2}} & -1 & \frac{1+j}{\sqrt{2}} & -j & \frac{-(1-j)}{\sqrt{2}} \\
1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\
1 & \frac{-(1-j)}{\sqrt{2}} & -j & \frac{1+j}{\sqrt{2}} & -1 & \frac{1-j}{\sqrt{2}} & j & \frac{-(1+j)}{\sqrt{2}} \\
1 & j & -1 & -j & 1 & j & -1 & -j \\
1 & \frac{1+j}{\sqrt{2}} & j & \frac{-(1-j)}{\sqrt{2}} & -1 & \frac{-(1+j)}{\sqrt{2}} & -j & \frac{1-j}{\sqrt{2}}
\end{bmatrix}
$$

Fourier Transform (FT)

- Fourier Transform (FT): Analyze the frequency content of the given signal



Fourier Transform (FT)

- Fourier Transform (FT): Analyze the frequency content of the given signal
- Assumes that signal comprising of multiple combinations of sine waves of different frequencies. Using correlation and orthogonal properties, FT finds out the contribution of each frequency in the given signal.
- For instance, for N-point FT, we consider N sine waves with frequency $\frac{kF_s}{N}$ where k varies from 0 to (N-1) i.e. correlation of each of these sine waves with input signal.

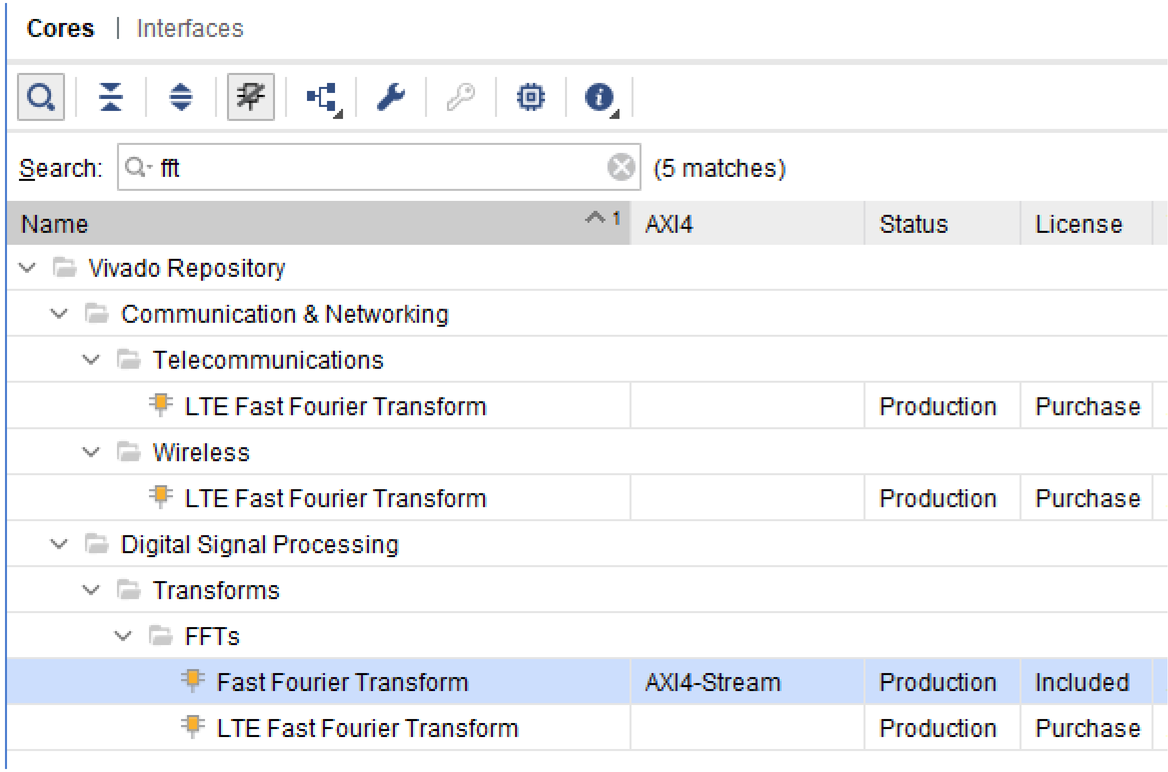
$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-\frac{2\pi kn}{N}i}$$

Lab

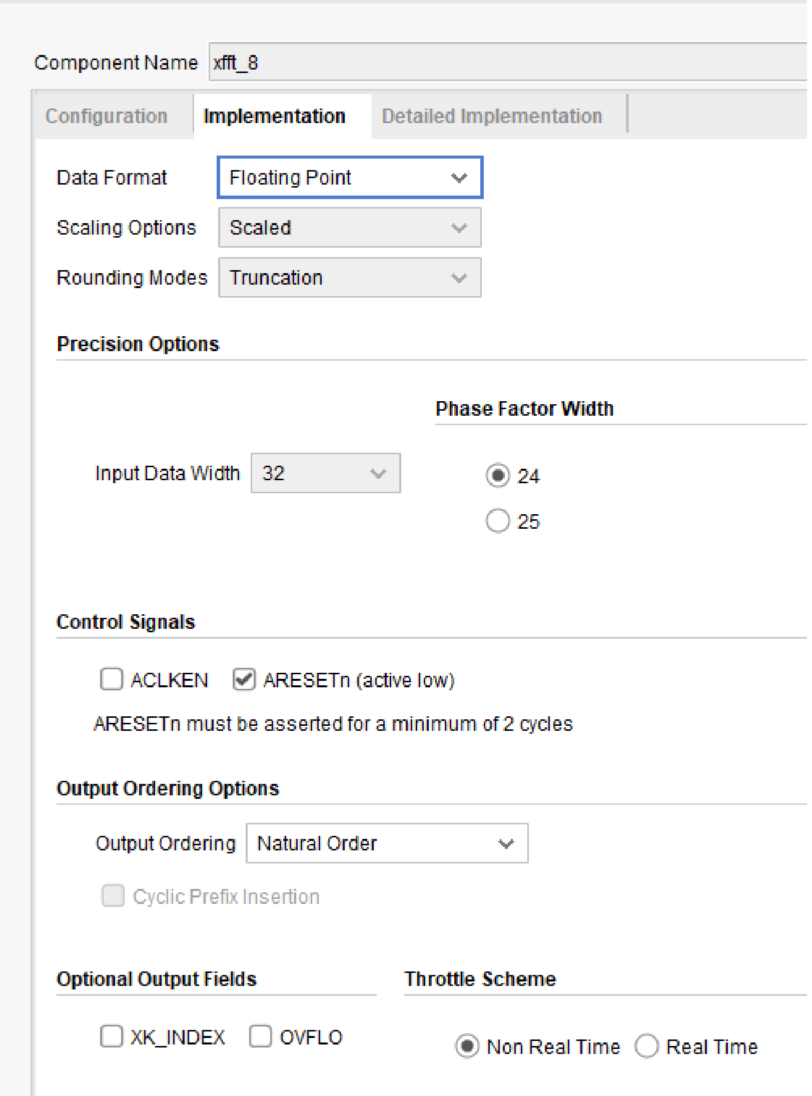
Proposed Approach

- Same as Lab 6

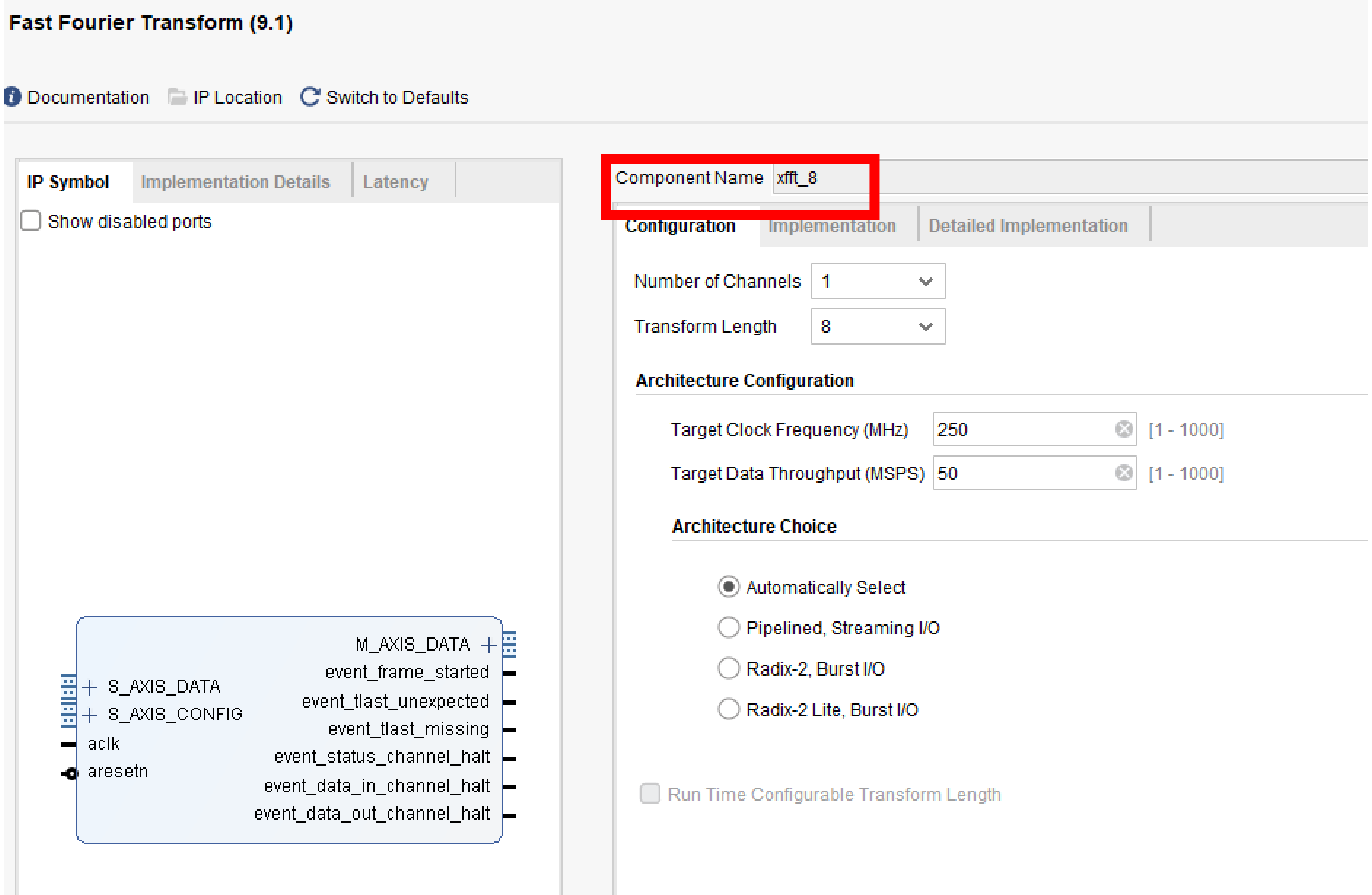
Locate the IP in Vivado



FFT

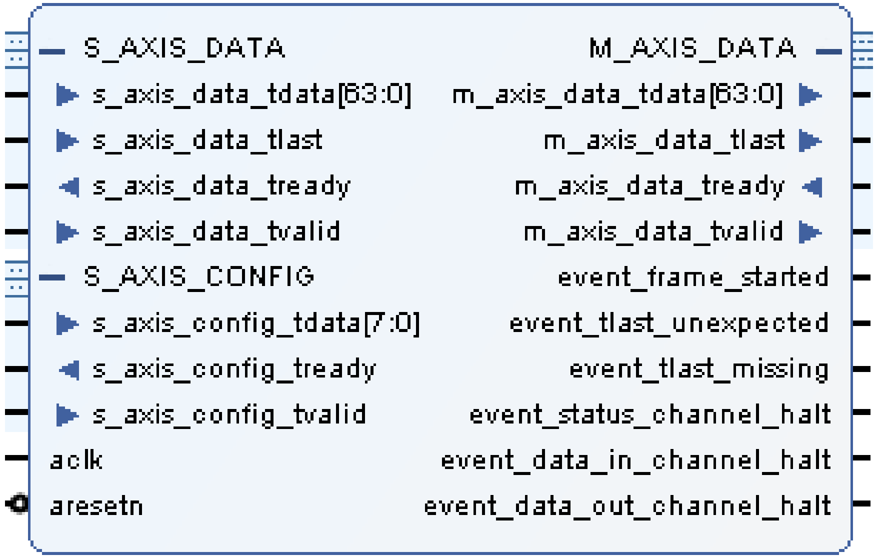


Component Name xfft_8
Configuration
Implementation
Detailed Implementation
Data Format Floating Point
Scaling Options Scaled
Rounding Modes Truncation
Precision Options
Phase Factor Width
Input Data Width 32
24
25
Control Signals
ACLKEN
ARESETn (active low)
ARESETn must be asserted for a minimum of 2 cycles
Output Ordering Options
Output Ordering Natural Order
Cyclic Prefix Insertion
Optional Output Fields
Throttle Scheme
XK_INDEX
OVFLO
Non Real Time
Real Time



Fast Fourier Transform (9.1)
Documentation
IP Location
Switch to Defaults
IP Symbol
Implementation Details
Latency
Show disabled ports
M_AXIS_DATA
event_frame_started
event_tlast_unexpected
event_tlast_missing
event_status_channel_halt
event_data_in_channel_halt
event_data_out_channel_halt
S_AXIS_DATA
S_AXIS_CONFIG
aclk
aresetn
Component Name xfft_8
Configuration
Implementation
Detailed Implementation
Number of Channels 1
Transform Length 8
Architecture Configuration
Target Clock Frequency (MHz) 250 [1 - 1000]
Target Data Throughput (MSPS) 50 [1 - 1000]
Architecture Choice
Automatically Select
Pipelined, Streaming I/O
Radix-2, Burst I/O
Radix-2 Lite, Burst I/O
Run Time Configurable Transform Length

FFT Interface



- Three AXI Stream Interface
- Slave Config Interface: 8-bit data
- Slave/Master Data Interface: 64-bit data (32 bit real and 32 bit imag.)
- For N-point FFT, send N samples sequentially and receive N samples sequentially

Testbench

- Floating point converter: https://www.h-schmidt.net/FloatConverter/IEEE754.html

```
module FFT_TB(

    );
    reg clock, resetn,c_valid,ffti_valid,ffti_last,ffto_ready;
    reg [63:0] ffti_data;
    reg [7:0] c_data;
    wire c_ready,ffti_ready,ffto_valid,ffto_last;
    wire [63:0] ffto_data;

    initial begin // Input initialization
        clock = 0;
        resetn = 0;
        c_valid = 0;
        c_data = 0;
        ffti_valid = 0;
        ffti_data = 0;
        ffti_last = 0;
        ffto_ready = 1;
    end

    always
     #5 clock = ~clock; // Clock Generation
```

Testbench

- Prepare input data

```
reg [31:0] input_data_r [7:0];
reg [31:0] input_data_im [7:0];
reg [31:0] output_data_r [7:0];
reg [31:0] output_data_im [7:0];
initial begin
    // Complex input data to be passed to FFT
    input_data_r[0] = 32'h3f800000;
    input_data_im[0] = 32'h0;
    input_data_r[1] = 32'h0;
    input_data_im[1] = 32'h0;
    input_data_r[2] = 32'h0;
    input_data_im[2] = 32'h0;
    input_data_r[3] = 32'h0;
    input_data_im[3] = 32'h0;
    input_data_r[4] = 32'h0;
    input_data_im[4] = 32'h0;
    input_data_r[5] = 32'h0;
    input_data_im[5] = 32'h0;
    input_data_r[6] = 32'h0;
    input_data_im[6] = 32'h0;
    input_data_r[7] = 32'h0;
    input_data_im[7] = 32'h0;
end
```

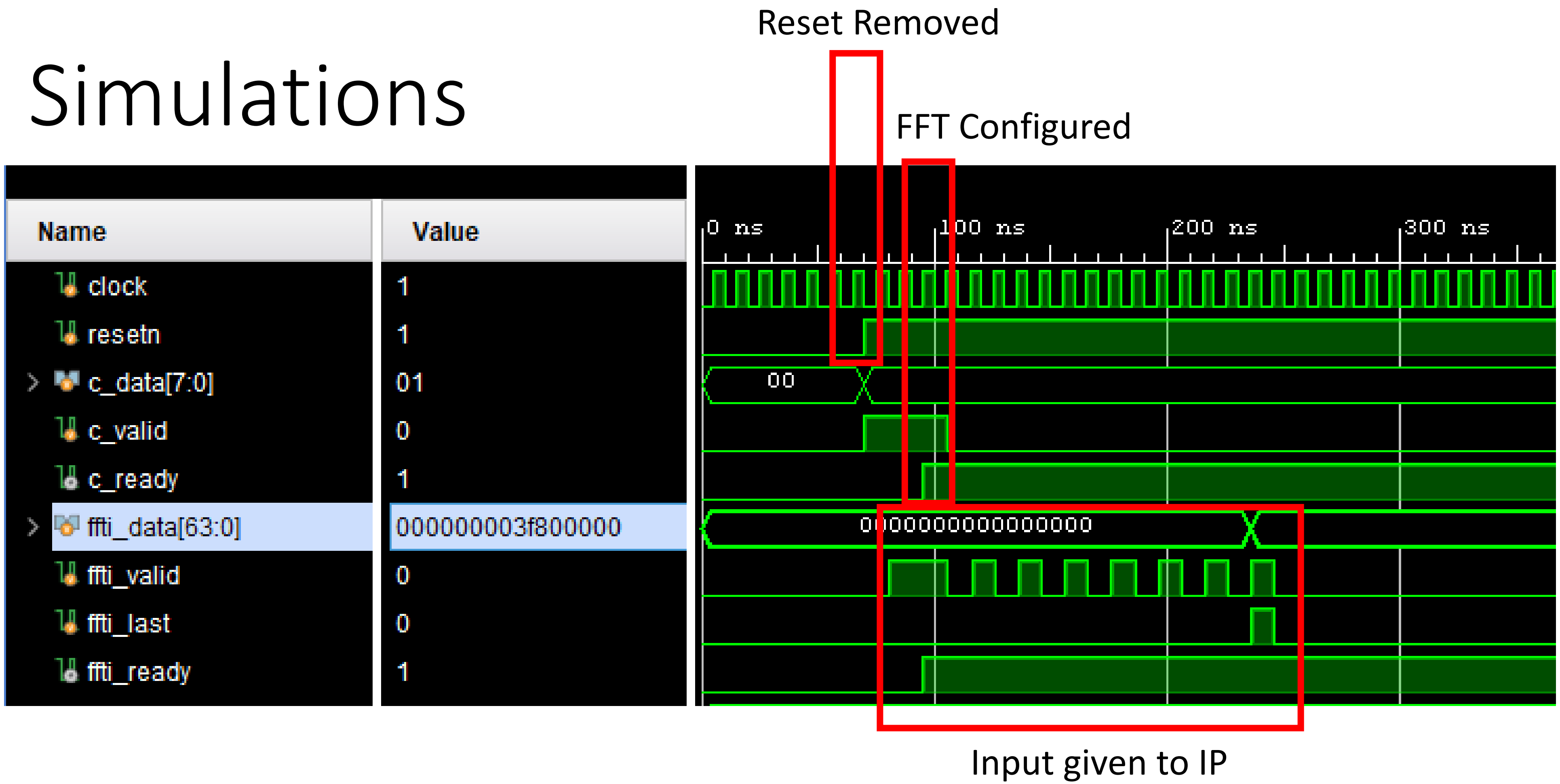
Testbench

```
// Configure the IP for FFT operation
initial begin
    #70 resetn = 1;
    c_data = 1;    // 1 refers to forward FFT
    c_valid = 1;
    while (c_ready == 0)
      #2  c_valid = 1;
    #10 c_valid = 0;
end
```

- Copy output 64 bit data into respective real and imaginary arrays
- Do not forget to instantiate the FFT IP

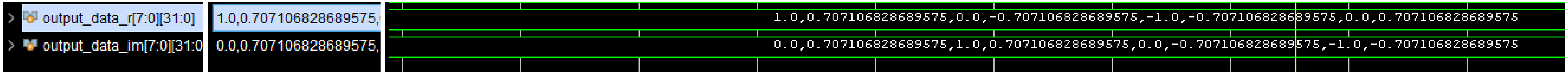
```
integer i,j;
initial begin
    #70 for(i=7; i>=0; i=i-1) begin
        #10 if(i == 0)
            ffti_last = 1;
        ffti_data = {input_data_im[i],input_data_r[i]};
        ffti_valid = 1;
        while (ffti_ready == 0)
          #2  ffti_valid = 1;
        #10 ffti_valid = 0;
        ffti_last = 0;
    end
end
initial begin
    for(j=7; j>=0; j=j-1) begin
         #5 ffto_ready = 1;
        wait(ffto_valid == 1);
        {output_data_im[j],output_data_r[j]} = ffto_data;
        #10  ffto_ready = 0;
    end
    #20 $stop;
end
```

Simulations



Simulations

Output from IP



Output from Matlab

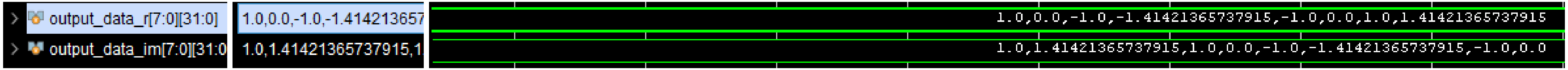
```
>> x= [0 0 0 0 0 0 0 1];
>> fft(x)

ans =

   1.0000 + 0.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i  -0.7071 + 0.7071i  -1.0000 + 0.0000i  -0.7071 - 0.7071i   0.0000 - 1.0000i   0.7071 - 0.7071i
```

Simulations

Output from IP



Output from Matlab

```
>> x= [0 0 0 0 0 0 0 1+1j];
>> fft(x)

ans =

   1.0000 + 1.0000i   0.0000 + 1.4142i  -1.0000 + 1.0000i  -1.4142 + 0.0000i  -1.0000 - 1.0000i   0.0000 - 1.4142i   1.0000 - 1.0000i   1.4142 + 0.0000i
```