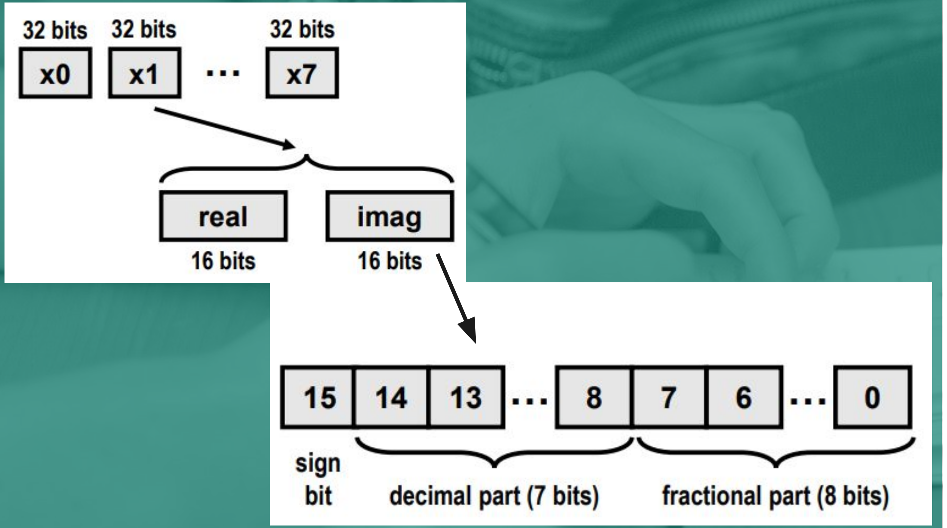
Design, Implementation and Simulation of an 8-point FFT Circuit in 45nm CMOS

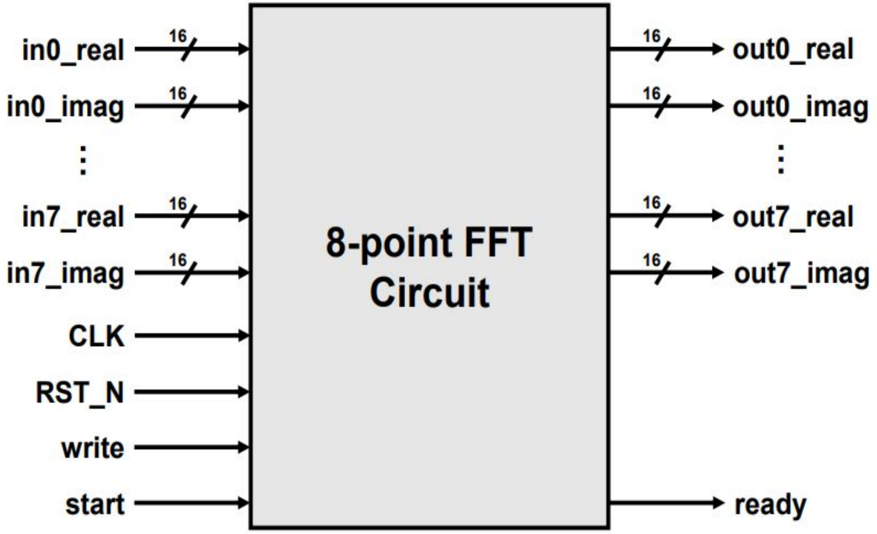
Outline

Project Specifications

Each input element x_m (for $m = 0,1, \dots , 7$) is a complex number with real and imaginary parts each represented as a signed 16-bit fixed point quantity with 1 sign bit, 7 bits for decimal part and 8 bits for fractional part, as shown below:



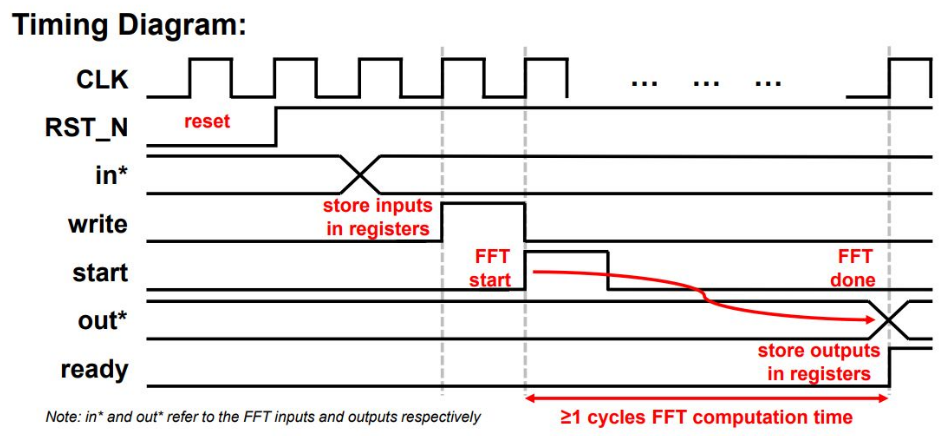
Block Diagram



Design Specifications

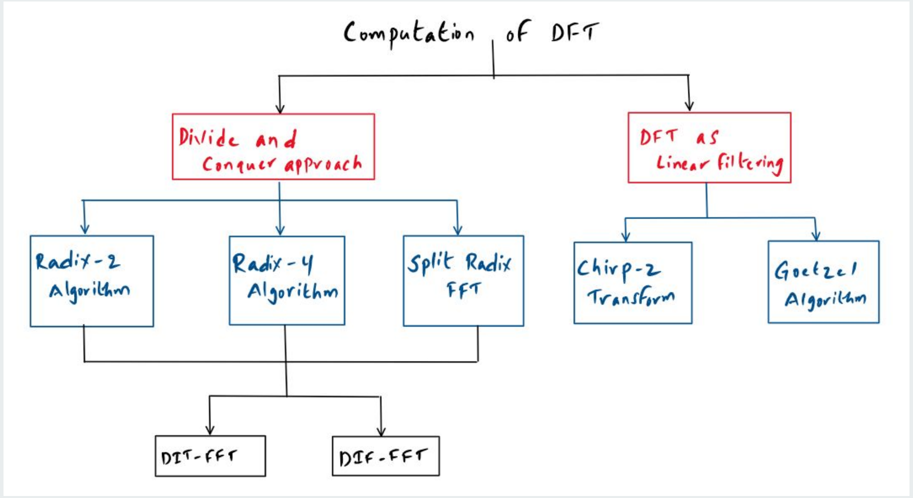
Signal	Type	Description
CLK	Input	Clock Signal
RST_N	Input	Active-Low Reset Signal
WRITE	Input	Input Write Signal
START	Input	FFT Computation Start Signal
READY	Output	FFT Computation Done Signal

Timing Diagram


Timing Diagram:
CLK
RST_N
in*
write
start
out*
ready
reset
store inputs in registers
FFT start
FFT done
store outputs in registers
≥1 cycles FFT computation time
Note: in* and out* refer to the FFT inputs and outputs respectively

Radix-2 DIT-FFT

01



$$x[n] \xleftrightarrow[\text{DFT}]{N\text{-Point}} X(k)$$

$$X(k) = \sum_{n=0}^{N-1} x[n]\, e^{-\frac{j2\pi kn}{N}} \quad ; \; k = 0, 1, 2, \ldots, (N-1)$$

$$= \sum_{n=0}^{N-1} x[n]\, W_N^{kn} \quad ; \; W_N = e^{-\frac{j2\pi}{N}} \text{ (Twiddle Factor)}$$

$$X(k) = x[0]\underset{\substack{\uparrow \\ \text{1st multiplication}}}{W_N^{0k}} \underset{\substack{\uparrow \\ \text{1st addition}}}{+} x[1]W_N^{1k} \underset{\substack{\uparrow \\ \text{2nd addition}}}{+} x[2]\underset{\substack{\uparrow \\ \text{3rd multiplication}}}{W_N^{2k}} + \cdots\cdots \underset{\substack{\uparrow \\ (N-1)^{th} \text{ addition}}}{+} x[N-1]\underset{\substack{\uparrow \\ N^{th} \text{ multiplication}}}{W_N^{(N-1)k}}$$

$$\text{Total No. of Complex additions} = N(N-1) = N^2 - N$$

$$\text{Total No. of Complex multiplications} = N.N = N^2$$

If we compute the N-Point DFT using the DFT equation it increases the complexity, as there are complex multiplications and complex additions to be performed. So, there are various techniques to efficiently calculate the N-point DFT. One such technique is Radix-2 Decimation in Time (DIT) Fast Fourier Transform Algorithm.

DIT FFT Algorithm Explanation

We have; $X(K) = \sum_{n=0}^{N-1} x[n] W_N^{kn}$; $K = 0, 1, 2, \ldots, (N-1)$

Splitting $x[n]$ into two parts;

$f_1[n]$ Containing even numbered samples of $x[n]$.

$f_2[n]$ Containing odd numbered samples of $x[n]$.

$$f_1[n] = x[2n] \qquad n = 0, 1, 2, \ldots, (\tfrac{N}{2} - 1)$$
$$f_2[n] = x[2n+1]$$

We can write;

$$X(K) = \sum_{\text{Even}} x[n] W_N^{kn} + \sum_{\text{odd}} x[n] W_N^{kn}$$
$$= \sum_{n=0}^{\frac{N}{2}-1} x[2m] W_N^{2mk} + \sum_{n=0}^{\frac{N}{2}-1} x[2m+1] W_N^{(2m+1)k}$$
$$= \sum_{m=0}^{\frac{N}{2}-1} f_1[m] W_N^{2mk} + \sum_{m=0}^{\frac{N}{2}-1} f_2[m] W_N^{2km} \cdot W_N^{k} \quad \ldots (i)$$
$$= \underbrace{\sum_{m=0}^{\frac{N}{2}-1} f_1[m] W_{N/2}^{k}}_{\substack{\frac{N}{2}\text{ point DFT of} \\ \frac{N}{2}\text{ point sequence} \\ f_1[m]}} + W_N^{k} \underbrace{\sum_{m=0}^{\frac{N}{2}-1} f_2[m] W_{N/2}^{km}}_{\substack{\frac{N}{2}\text{ point DFT of} \\ \frac{N}{2}\text{ point sequence } f_2[m]}}$$

$$\underset{N\text{-Point}}{\underline{X(K)}} = \underset{\frac{N}{2}\text{-Point}}{\underline{F_1(K)}} + W_N^{K} \underset{\frac{N}{2}\text{-Point}}{\underline{F_2(K)}} \quad ; k = 0, 1, 2, \ldots, (N-1) \quad \ldots (ii)$$

$$\underset{N\text{-Point}}{\underline{X(K)}} = \underset{\frac{N}{2}\text{-Point}}{\underline{F_1(K)}} + W_N^{K} \underset{\frac{N}{2}\text{-Point}}{\underline{F_2(K)}} \quad ; k = 0, 1, 2, \ldots, (N-1) \quad \ldots (ii)$$

From periodicity property;

$$X(K+N) = X(K)$$
$$F_1(K + N/2) = F_1(K)$$
$$F_2(K + N/2) = F_2(K)$$

Replacing K by $K + N/2$ in equ (ii);

$$X(K + N/2) = F_1(K + N/2) + W_N^{(K+N/2)} F_2(K + N/2)$$
$$= F_1(K) - W_N^{K} F_2(K)$$

Hence;

$$X(K) = F_1(K) + W_N^{K} F_2(K)$$
$$X(K + N/2) = F_1(K) - W_N^{K} F_2(K) \quad ; K = 0, 1, 2, \ldots, (\tfrac{N}{2} - 1)$$

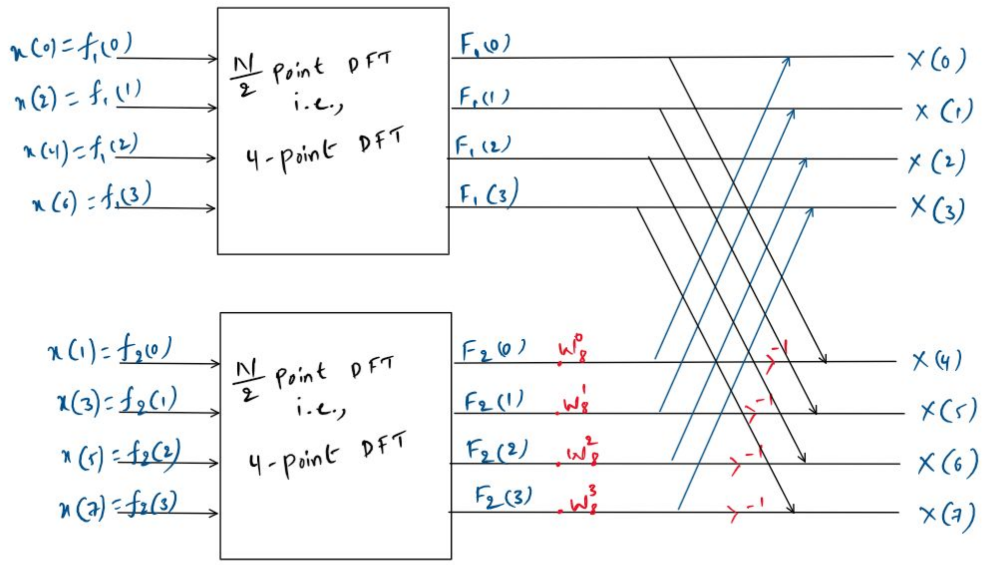
DIT FFT Algorithm Explanation

Consider an example of $N = 8$;

$$x[n] = \{x(0), x(1), x(2), \ldots\ldots, x(7)\}$$

$$x[2n] = f_1[n] = \{x(0), x(2), x(4), x(6)\}$$

$$x[2n+1] = f_2[n] = \{x(1), x(3), x(5), x(7)\}$$



DIT FFT Algorithm Explanation

Second stage of decomposition:-

Splitting sequence ($\frac{N}{2}$ point); $f_1[n]$ and $f_2[n]$ in two parts of their Even and odd numbered samples i.e.,

$$\begin{tikzcd} & f_1[n] \arrow[dl] \arrow[dr] & \\ v_{11}[n]=f_1[2n] & & v_{12}[n]=f_1[2n+1] \end{tikzcd}$$

$\{f_1(0), f_1(2)\}$ $\{f_1(1), f_1(3)\}$

$$\begin{tikzcd} & f_1[n] \arrow[dl] \arrow[dr] & \\ v_{21}[n]=f_2[2n] & & v_{22}[n]=f_2[2n+1] \end{tikzcd}$$

$\{f_2(0), f_2(2)\}$ $\{f_2(1), f_2(3)\}$

We have;

$$X(k) = F_1(k) + W_N^k F_2(k)$$
$$X(k+\tfrac{N}{2}) = F_1(k) - W_N^k F_2(k)$$

We can get;

$$F_1(k) = V_{11}(k) + W_{N/2}^k V_{12}(k) \quad ; k = 0, 1, 2, \ldots, (\tfrac{N}{4} - 1)$$
$$F_1(k+N/4) = V_{11}(k) - W_{N/2}^k V_{12}(k)$$
$$F_2(k) = V_{21}(k) + W_{N/2}^k V_{22}(k) \quad ; k = 0, 1, 2, \ldots, (\tfrac{N}{4} - 1)$$
$$F_2(k+N/4) = V_{21}(k) - W_{N/2}^k V_{22}(k)$$

DIT FFT Algorithm Explanation

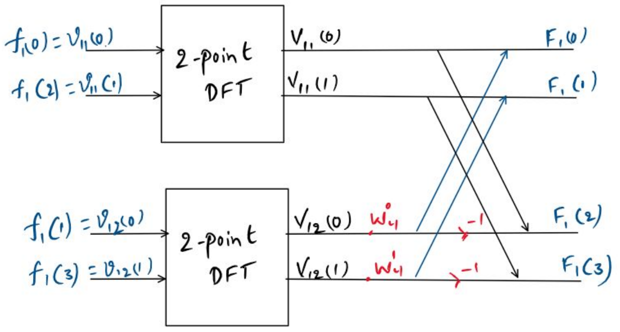
Consider an example of $N=8$;

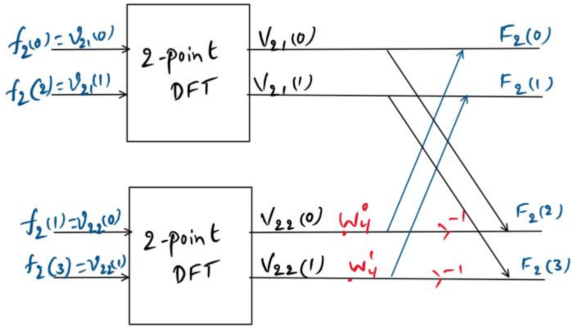
$F_1(k) = V_{11}(k) + W_4^k \, V_{12}(k) \quad ; k = 0,1$

$F_1(k+2) = V_{11}(k) - W_4^k \, V_{12}(k)$

$F_2(k) = V_{21}(k) + W_4^k \, V_{22}(k) \quad ; k = 0,1$

$F_2(k+2) = V_{21}(k) - W_4^k \, V_{22}(k)$





Calculation of 2-point DFT;

$X(k) = \sum_{n=0}^{N-1} x[n] W_N^{kn} \quad ; k = 0,1,2,\dots,N-1$

$$v_{11}[n] \xleftrightarrow{\text{2-Point DFT}} V_{11}(k)$$

$V_{11}(k) = \sum_{n=0}^{1} v_{11}[n] W_2^{kn} \quad ; k = 0,1$

We get;

$V_{11}(0) = v_{11}(0) W_2^0 + v_{11}(1) W_2^0$

$= v_{11}(0) + v_{11}(1)$

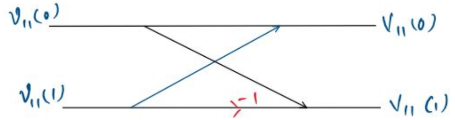
$V_{11}(1) = v_{11}(0) W_2^0 + v_{11}(1) W_2^1$

$= v_{11}(0) - v_{11}(1)$

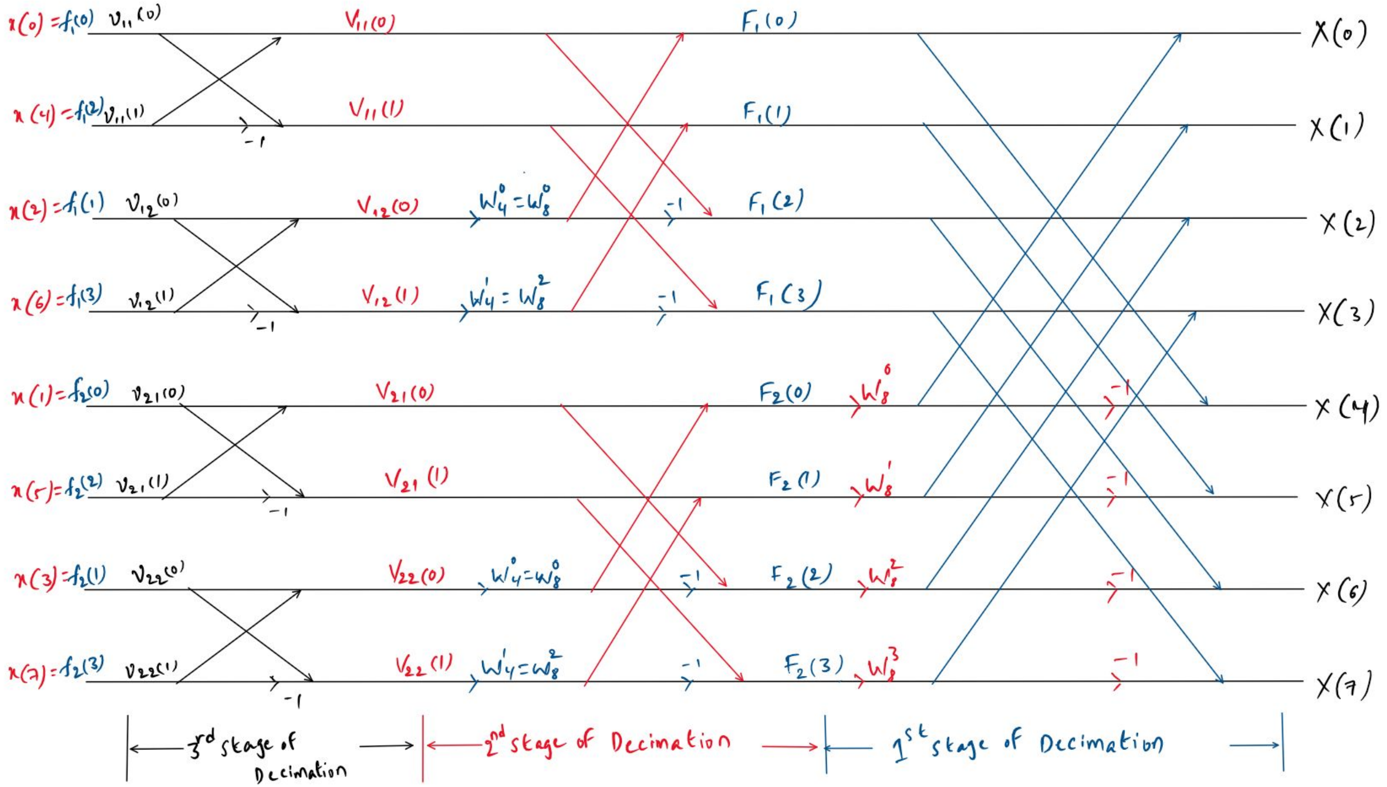
$W_2^1 = e^{-\frac{j2\pi}{2}\times 1}$

$= e^{-j\pi}$

$= -1$



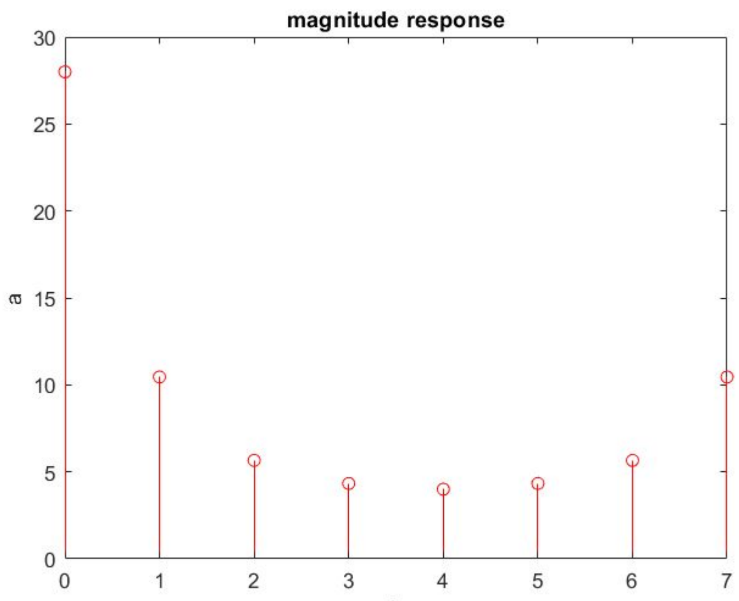
8-Point Radix 2 DIT FFT Butterfly Diagram

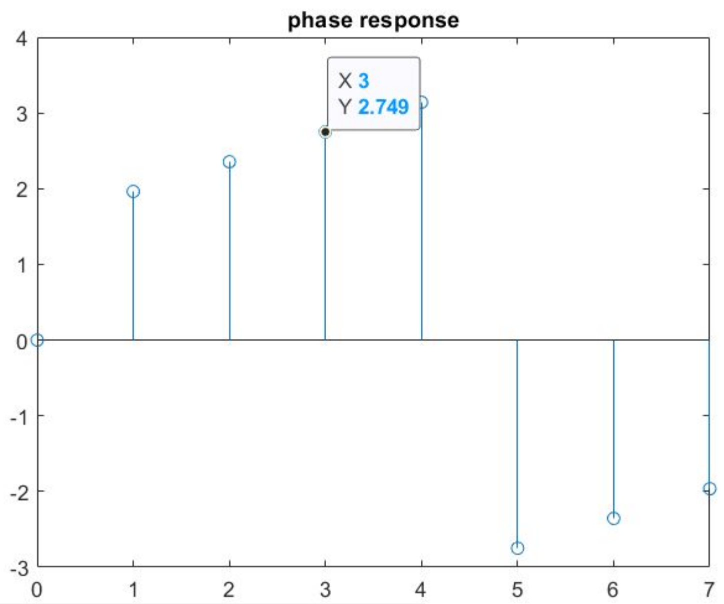


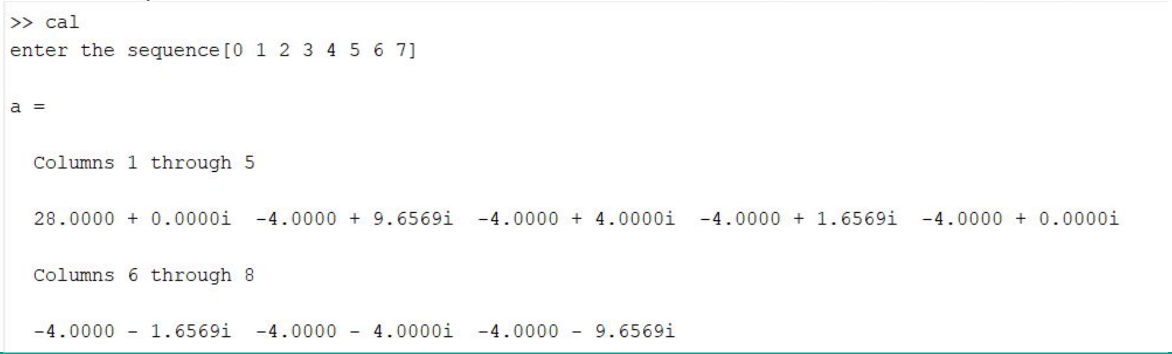
Simulation in MATLAB

02

```
x=input('enter the sequence');
N=length(x);
a=fft(x)
k=0:1:N-1;
figure(1);
stem(k,abs(a),'r');
xlabel('k'),ylabel('a');
title('magnitude response');
figure(2);
stem(k,angle(a));
title('phase response');
```





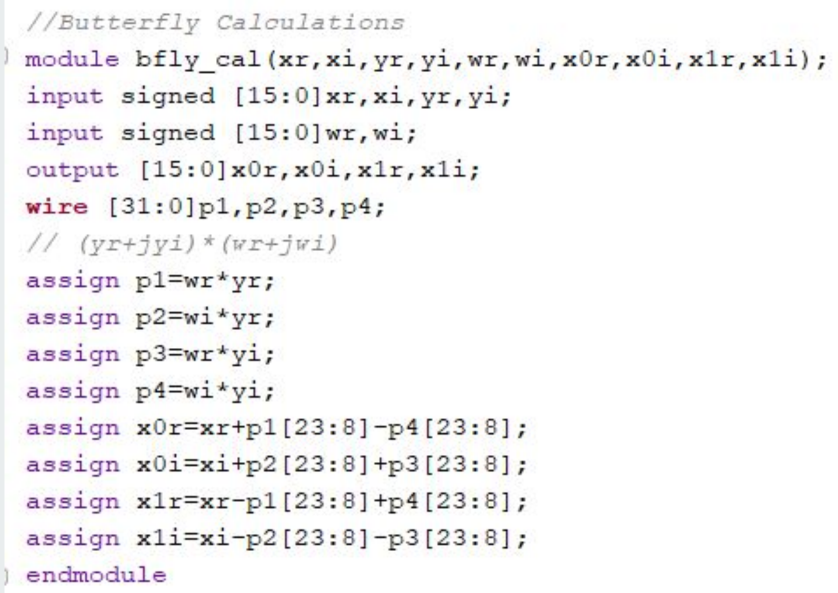


```
>> cal
enter the sequence[0 1 2 3 4 5 6 7]

a =

  Columns 1 through 5

  28.0000 + 0.0000i  -4.0000 + 9.6569i  -4.0000 + 4.0000i  -4.0000 + 1.6569i  -4.0000 + 0.0000i

  Columns 6 through 8

  -4.0000 - 1.6569i  -4.0000 - 4.0000i  -4.0000 - 9.6569i
```

MATLAB Simulation Results

Verilog Code

03

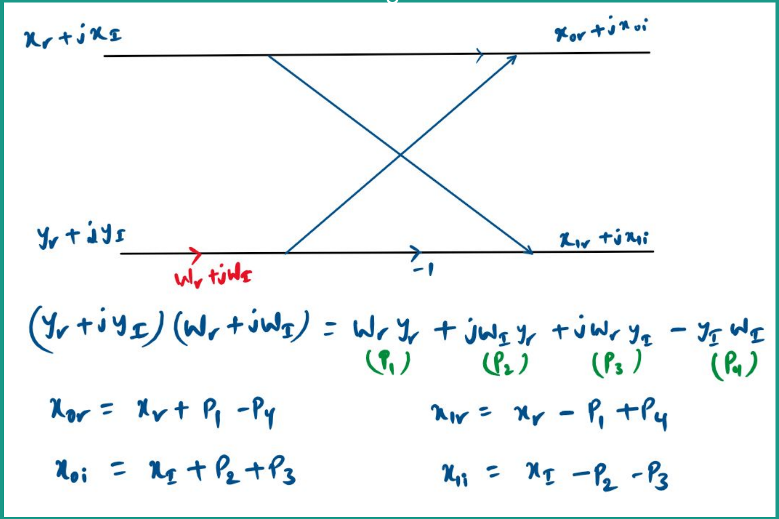


```
//Butterfly Calculations
module bfly_cal(xr,xi,yr,yi,wr,wi,x0r,x0i,x1r,x1i);
input signed [15:0]xr,xi,yr,yi;
input signed [15:0]wr,wi;
output [15:0]x0r,x0i,x1r,x1i;
wire [31:0]p1,p2,p3,p4;
//  (yr+jyi)*(wr+jwi)
assign p1=wr*yr;
assign p2=wi*yr;
assign p3=wr*yi;
assign p4=wi*yi;
assign x0r=xr+p1[23:8]-p4[23:8];
assign x0i=xi+p2[23:8]+p3[23:8];
assign x1r=xr-p1[23:8]+p4[23:8];
assign x1i=xi-p2[23:8]-p3[23:8];
endmodule
```

Verilog Code Explanation

1. Define Twiddle Factors
   As we are using the 8 Point DIT FFT Algorithm the Twiddle Factors will be W_8^k (k=0,1,2,3).
   So make the Twiddle Factors as Constant Values.

2. Butterfly Calculations
   Now we need to implement Butterfly Unit as shown.
   Here, we are performing Multiplication and Addition on 16 bit Signed Integers which has Least 8 bits as fractional part value.
   So, if we multiply two 16 bit Numbers we get 32 bit Result. Hence, we are defining product terms as a 32 bit array.
   Then while adding the result we are considering only [23:8] bit positions and neglecting remaining bits as these are the significant bit positions. (We can understand this by considering wr=0.707 which is represented as 16’b00000000_10110101 and we can observe that product of any number with this results in [32:24] bits as 0’s).


$x_r + jx_I$
$x_{0r} + jx_{0i}$
$y_r + jy_I$
$x_{1r} + jx_{1i}$
$W_r + jW_I$
-1
$(y_r + jy_I)(W_r + jW_I) = \underbrace{W_r y_r}_{(P_1)} + \underbrace{jW_I y_r}_{(P_2)} + \underbrace{jW_r y_I}_{(P_3)} - \underbrace{y_I W_I}_{(P_4)}$
$x_{0r} = x_r + P_1 - P_4$
$x_{1r} = x_r - P_1 + P_4$
$x_{0i} = x_I + P_2 + P_3$
$x_{1i} = x_I - P_2 - P_3$

Verilog Code Explanation

3. Butterfly Stages Calculations
   Now we need to Perform Stage wise Butterfly Calculations as we have seen in Radix-2 DIT FFT Algorithm.
   To perform this Calculations we have defined temporary variables to save the intermediate values.

4. Meeting Design Specifications
   As mentioned in the Design Specifications we need to read the Input Values when write Signal is HIGH, So we are checking for the write signal to be HIGH and then storing the input values into temporary input values.
   Then, we need to start FFT computation when start signal is HIGH, So we are checking for Start signal to be HIGH and then assigning the temporary input values into the Butterfly Calculations functions.
   After completing the FFT Computation we set the Ready Signal to HIGH informing that FFT Computation is finished.

```
`timescale 1ns / 1ps
module fft8 (clk, rst, write, start, ready, state, x0r, x0i, x1r, x1i, x2r, x2i, x3r, x3i, x4r, x4i, x5r, x5i, x6r, x6i, x7r, x7i, y0r, y0i, y1r, y1i, y2r,
y2i, y3r, y3i, y4r, y4i, y5r, y5i, y6r, y6i, y7r, y7i);
input clk,rst,write,start;
input [15:0]x0r,x0i,x1r,x1i,x2r,x2i,x3r,x3i,x4r,x4i,x5r,x5i,x6r,x6i,x7r,x7i;
output reg [15:0] y0r,y0i,y1r,y1i,y2r,y2i,y3r,y3i,y4r,y4i,y5r,y5i,y6r,y6i,y7r,y7i;
output reg ready;
output reg [1:0]state;

wire [15:0] y0r_t,y0i_t,y1r_t,y1i_t,y2r_t,y2i_t,y3r_t,y3i_t,y4r_t,y4i_t,y5r_t,y5i_t,y6r_t,y6i_t,y7r_t,y7i_t;
wire [15:0] x20r,x20i,x21r,x21i,x22r,x22i,x23r,x23i,x24r,x24i,x25r,x25i,x26r,x26i,x27r,x27i;
wire [15:0] x10r,x10i,x11r,x11i,x12r,x12i,x13r,x13i,x14r,x14i,x15r,x15i,x16r,x16i,x17r,x17i;
reg[15:0] x0r_temp, x0i_temp, x1r_temp, x1i_temp, x2r_temp, x2i_temp, x3r_temp, x3i_temp, x4r_temp, x4i_temp, x5r_temp,
x5i_temp,x6r_temp,x6i_temp,x7r_temp,x7i_temp;
reg[15:0] x0r_t,x0i_t,x1r_t,x1i_t,x2r_t,x2i_t,x3r_t,x3i_t,x4r_t,x4i_t,x5r_t,x5i_t,x6r_t,x6i_t,x7r_t,x7i_t;

//Define Twiddle Factors
parameter w0r=16'b1_00000000;
parameter w0i=16'b0_00000000;
parameter w1r=16'b00000000_10110101;//0.707=0.10110101
parameter w1i=16'b11111111_01001011;//-0.707=1.01001011
parameter w2r=16'b00000000_00000000;
parameter w2i=16'b11111111_00000000;//-1
parameter w3r=16'b11111111_01001011;//-0.707=1.01001011
parameter w3i=16'b11111111_01001011;//-0.707=1.01001011
```

Verilog Code

```
always @(posedge clk) begin
                if (~rst) begin
                  //If Reset Signal is Low then output=0
                  state<=2'b00; ready <= 1'b0;
                end
                else begin
                 case(state)
                   2'b00 : if(write) begin
                                  state<=2'b01; ready<=1'b0;
                             end
                             else begin
                                  state<=2'b00; ready<=1'b0;
                            end
                   2'b01: begin
                            //Storing input values into temporary variables
                             x0r_temp<=x0r; x0i_temp<=x0i;
                            x1r_temp<=x1r; x1i_temp<=x1i;
                            x2r_temp<=x2r; x2i_temp<=x2i;
                            x3r_temp<=x3r; x3i_temp<=x3i;
                            x4r_temp<=x4r; x4i_temp<=x4i;
                            x5r_temp<=x5r; x5i_temp<=x5i;
                            x6r_temp<=x6r; x6i_temp<=x6i;
                            x7r_temp<=x7r; x7i_temp<=x7i;
                            state<=2'b10;
                            end
```

Verilog Code

```
                2'b10: begin
                        if(start) begin
                               x0r_t<=x0r_temp; x0i_t<=x0i_temp;
                                x1r_t<=x1r_temp; x1i_t<=x1i_temp;
                                x2r_t<=x2r_temp; x2i_t<=x2i_temp;
                                x3r_t<=x3r_temp; x3i_t<=x3i_temp;
                                x4r_t<=x4r_temp; x4i_t<=x4i_temp;
                                x5r_t<=x5r_temp; x5i_t<=x5i_temp;
                                x6r_t<=x6r_temp; x6i_t<=x6i_temp;
                                x7r_t<=x7r_temp; x7i_t<=x7i_temp;
                                state<=2'b11;
                                 end
                         end
                2'b11: begin  //assigning outputs to registers
                         y0r<=y0r_t; y0i<=y0i_t;
                         y1r<=y1r_t; y1i<=y1i_t;
                         y2r<=y2r_t; y2i<=y2i_t;
                         y3r<=y3r_t; y3i<=y3i_t;
                         y4r<=y4r_t; y4i<=y4i_t;
                         y5r<=y5r_t; y5i<=y5i_t;
                         y6r<=y6r_t; y6i<=y6i_t;
                         y7r<=y7r_t; y7i<=y7i_t;
                         ready<=1'b1;
                       end
                default : state<=2'b00;
               endcase
           end
       end
```

Verilog Code

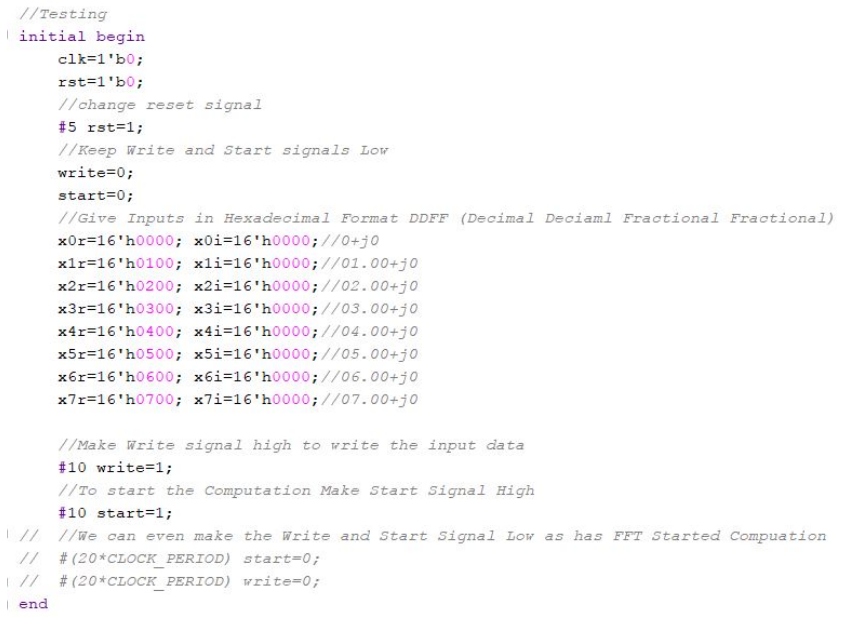
```
//Butterfly Stage 1
bfly_cal s11(x0r_t,x0i_t,x4r_t,x4i_t,w0r,w0i,x10r,x10i,x11r,x11i);
bfly_cal s12(x2r_t,x2i_t,x6r_t,x6i_t,w0r,w0i,x12r,x12i,x13r,x13i);
bfly_cal s13(x1r_t,x1i_t,x5r_t,x5i_t,w0r,w0i,x14r,x14i,x15r,x15i);
bfly_cal s14(x3r_t,x3i_t,x7r_t,x7i_t,w0r,w0i,x16r,x16i,x17r,x17i);

//Butterfly Stage 2
bfly_cal s21(x10r,x10i,x12r,x12i,w0r,w0i,x20r,x20i,x22r,x22i);
bfly_cal s22(x11r,x11i,x13r,x13i,w2r,w2i,x21r,x21i,x23r,x23i);
bfly_cal s23(x14r,x14i,x16r,x16i,w0r,w0i,x24r,x24i,x26r,x26i);
bfly_cal s24(x15r,x15i,x17r,x17i,w2r,w2i,x25r,x25i,x27r,x27i);

//Butterfly Stage 3
bfly_cal s31(x20r,x20i,x24r,x24i,w0r,w0i,y0r_t,y0i_t,y4r_t,y4i_t);
bfly_cal s32(x21r,x21i,x25r,x25i,w1r,w1i,y1r_t,y1i_t,y5r_t,y5i_t);
bfly_cal s33(x22r,x22i,x26r,x26i,w2r,w2i,y2r_t,y2i_t,y6r_t,y6i_t);
bfly_cal s34(x23r,x23i,x27r,x27i,w3r,w3i,y3r_t,y3i_t,y7r_t,y7i_t);

endmodule
```

```
//Butterfly Calculations
module bfly_cal(xr, xi, yr, yi, wr, wi, x0r, x0i, x1r, x1i);
input signed [15:0]xr, xi, yr, yi;
input signed [15:0]wr, wi;
output [15:0]x0r, x0i, x1r, x1i;
wire [31:0]p1, p2, p3, p4;
// (yr+jyi)*(wr+jwi)
assign p1=wr*yr;
assign p2=wi*yr;
assign p3=wr*yi;
assign p4=wi*yi;
assign x0r=xr+p1[23:8]-p4[23:8];
assign x0i=xi+p2[23:8]+p3[23:8];
assign x1r=xr-p1[23:8]+p4[23:8];
assign x1i=xi-p2[23:8]-p3[23:8];
endmodule
```

Verilog Code

Verilog Test Bench

04



```
//Testing
initial begin
    clk=1'b0;
    rst=1'b0;
    //change reset signal
    #5 rst=1;
    //Keep Write and Start signals Low
    write=0;
    start=0;
    //Give Inputs in Hexadecimal Format DDFF (Decimal Deciaml Fractional Fractional)
    x0r=16'h0000; x0i=16'h0000;//0+j0
    x1r=16'h0100; x1i=16'h0000;//01.00+j0
    x2r=16'h0200; x2i=16'h0000;//02.00+j0
    x3r=16'h0300; x3i=16'h0000;//03.00+j0
    x4r=16'h0400; x4i=16'h0000;//04.00+j0
    x5r=16'h0500; x5i=16'h0000;//05.00+j0
    x6r=16'h0600; x6i=16'h0000;//06.00+j0
    x7r=16'h0700; x7i=16'h0000;//07.00+j0

    //Make Write signal high to write the input data
    #10 write=1;
    //To start the Computation Make Start Signal High
    #10 start=1;
//  //We can even make the Write and Start Signal Low as has FFT Started Compuation
//  #(20*CLOCK_PERIOD) start=0;
//  #(20*CLOCK_PERIOD) write=0;
end
```

Test Bench Explanation

1. Generate Clock Signal
   We can generate a clock using some delay (Time Period). Invert clk signal after some delay and keep this in an always block so that we can generate a clk signal with required Time Period.
   # (Time Period/2) clk = ~clk

2. Testing the Code
   Check conditions for Reset, Write and Start.
   Give Inputs in Hexadecimal Format Decimal Decimal Fractional Fractional (DDFF) for both Real and Imaginary parts of Input Sequence.
   Now make write signal HIGH, so that Inputs will be read into temporary variables. Then, to start the FFT Computation make start signal HIGH.

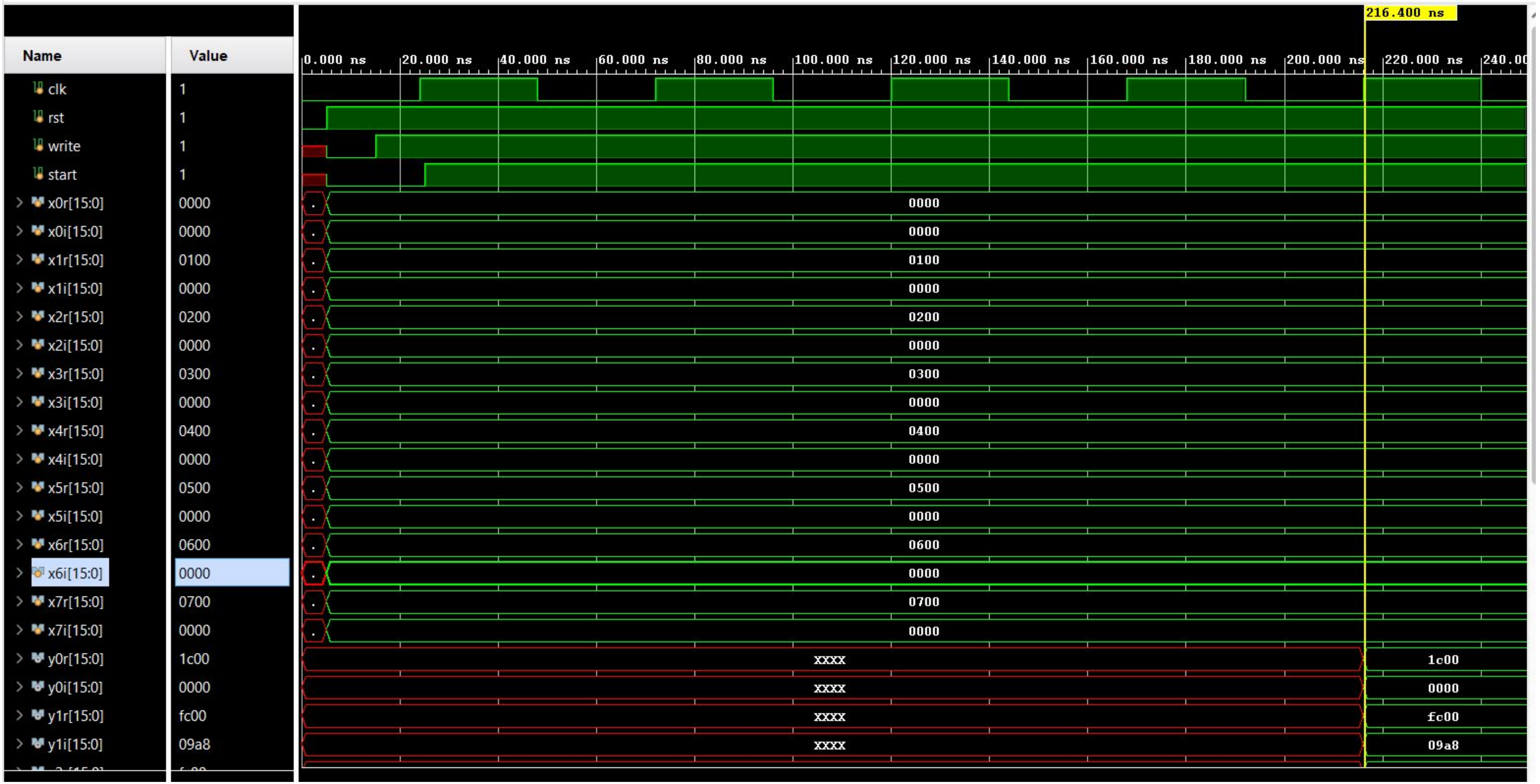
```
`timescale 1ns/1ps
module ffttestbench ();
parameter CLOCK_PERIOD = 49; // 49ns Time Period
reg clk, rst, write, start;
reg signed [15:0] x0r, x0i, x1r, x1i, x2r, x2i ,x3r, x3i, x4r, x4i, x5r, x5i, x6r, x6i, x7r, x7i;
wire signed [15:0] y0r, y0i, y1r, y1i, y2r, y2i, y3r, y3i, y4r, y4i, y5r, y5i, y6r, y6i, y7r, y7i;
wire ready;
wire [1:0]state;

//Module used for testing
fft8 fft8_test(.clk(clk), .rst(rst), .write(write), .start(start), .ready(ready), .state(state), .x0r(x0r), .x0i(x0i), .x1r(x1r), .x1i(x1i), .x2r(x2r), .x2i(x2i),
.x3r(x3r), .x3i(x3i), .x4r(x4r), .x4i(x4i), .x5r(x5r), .x5i(x5i), .x6r(x6r), .x6i(x6i), .x7r(x7r), .x7i(x7i), .y0r(y0r), .y0i(y0i), .y1r(y1r), .y1i(y1i), .y2r(y2r),
.y2i(y2i), .y3r(y3r), .y3i(y3i), .y4r(y4r), .y4i(y4i), .y5r(y5r), .y5i(y5i), .y6r(y6r), .y6i(y6i), .y7r(y7r), .y7i(y7i));

//Testing
initial begin
        clk=1'b0;
        rst=1'b0;
        //change reset signal
        #5 rst=1;
        //Keep Write and Start signals Low
        write=0;
        start=0;
```

Verilog Test Bench

```
          //Give Inputs in Hexadecimal Format DDFF (Decimal Decimal Fractional Fractional)
          x0r=16'h0000; x0i=16'h0000;//0+j0
          x1r=16'h0100; x1i=16'h0000;//01.00+j0
          x2r=16'h0200; x2i=16'h0000;//02.00+j0
          x3r=16'h0300; x3i=16'h0000;//03.00+j0
          x4r=16'h0400; x4i=16'h0000;//04.00+j0
          x5r=16'h0500; x5i=16'h0000;//05.00+j0
          x6r=16'h0600; x6i=16'h0000;//06.00+j0
          x7r=16'h0700; x7i=16'h0000;//07.00+j0

          //Make Write signal high to write the input data
          #10 write=1;
          //To start the Computation Make Start Signal High
          #10 start=1;
//        //We can even make the Write and Start Signal Low as has FFT Started Computation
//        #(20*CLOCK_PERIOD) start=0;
//        #(20*CLOCK_PERIOD) write=0;
end

// System clock generator
always begin
          //after delay of clockperiod/2 we are inverting the clk value
          #(CLOCK_PERIOD/2) clk = ~clk;
end
```

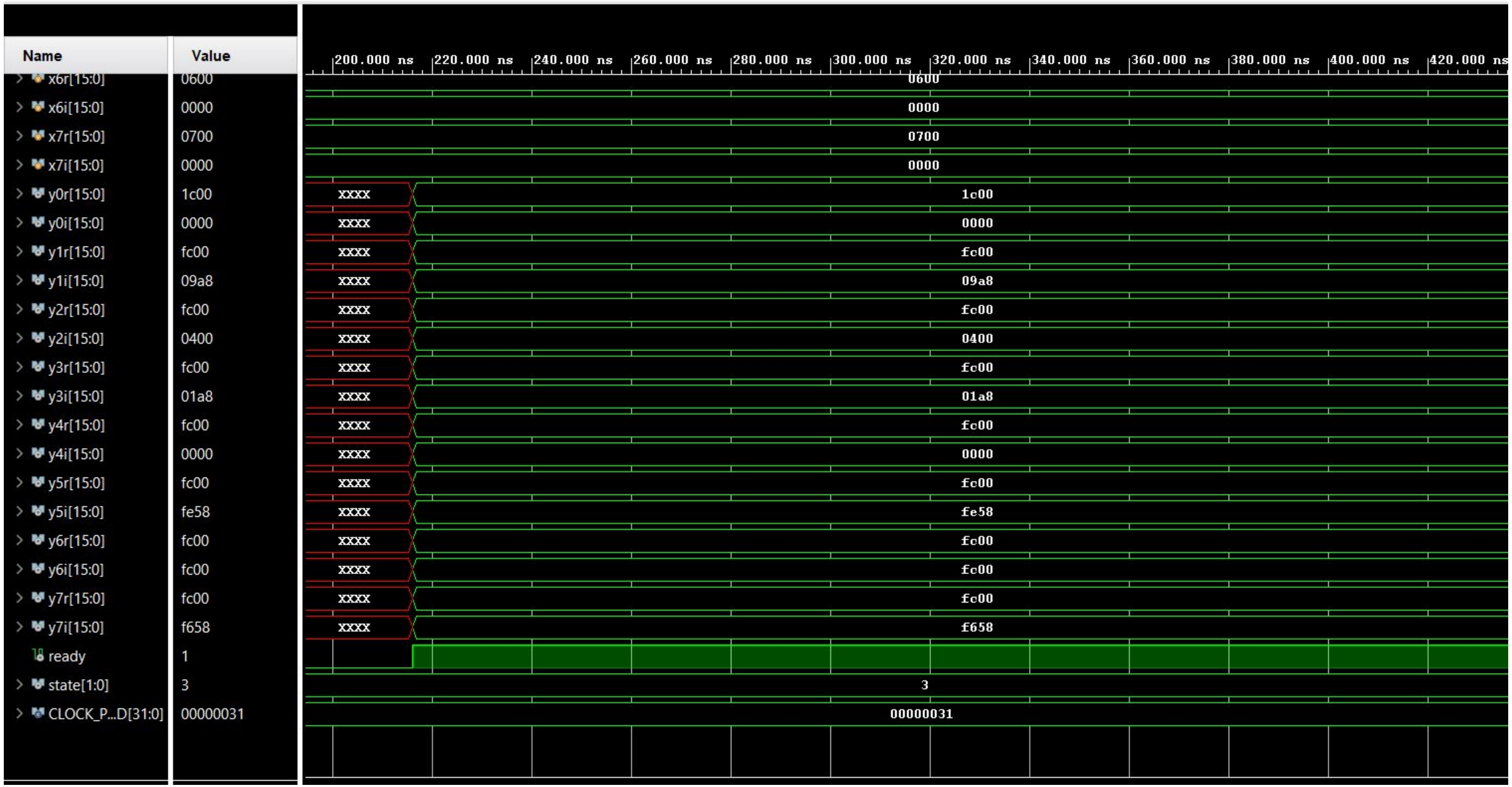
```
// VCD dump
initial begin
          $dumpfile("fft.vcd");
          $dumpvars(0, fft8_test);
          #500 $finish;
end
endmodule
```

Verilog Test Bench



Name	Value
clk	1
rst	1
write	1
start	1
x0r[15:0]	0000
x0i[15:0]	0000
x1r[15:0]	0100
x1i[15:0]	0000
x2r[15:0]	0200
x2i[15:0]	0000
x3r[15:0]	0300
x3i[15:0]	0000
x4r[15:0]	0400
x4i[15:0]	0000
x5r[15:0]	0500
x5i[15:0]	0000
x6r[15:0]	0600
x6i[15:0]	0000
x7r[15:0]	0700
x7i[15:0]	0000
y0r[15:0]	1c00
y0i[15:0]	0000
y1r[15:0]	fc00
y1i[15:0]	09a8

Test Bench Simulations


```
Name            Value     200.000 ns  220.000 ns  240.000 ns  260.000 ns  280.000 ns  300.000 ns  320.000 ns  340.000 ns  360.000 ns  380.000 ns  400.000 ns  420.000 ns
x6r[15:0]       0600      0600
x6i[15:0]       0000      0000
x7r[15:0]       0700      0700
x7i[15:0]       0000      0000
y0r[15:0]       1c00      XXXX  1c00
y0i[15:0]       0000      XXXX  0000
y1r[15:0]       fc00      XXXX  fc00
y1i[15:0]       09a8      XXXX  09a8
y2r[15:0]       fc00      XXXX  fc00
y2i[15:0]       0400      XXXX  0400
y3r[15:0]       fc00      XXXX  fc00
y3i[15:0]       01a8      XXXX  01a8
y4r[15:0]       fc00      XXXX  fc00
y4i[15:0]       0000      XXXX  0000
y5r[15:0]       fc00      XXXX  fc00
y5i[15:0]       fe58      XXXX  fe58
y6r[15:0]       fc00      XXXX  fc00
y6i[15:0]       fc00      XXXX  fc00
y7r[15:0]       fc00      XXXX  fc00
y7i[15:0]       f658      XXXX  f658
ready           1
state[1:0]      3         3
CLOCK_P...D[31:0] 00000031  00000031
```

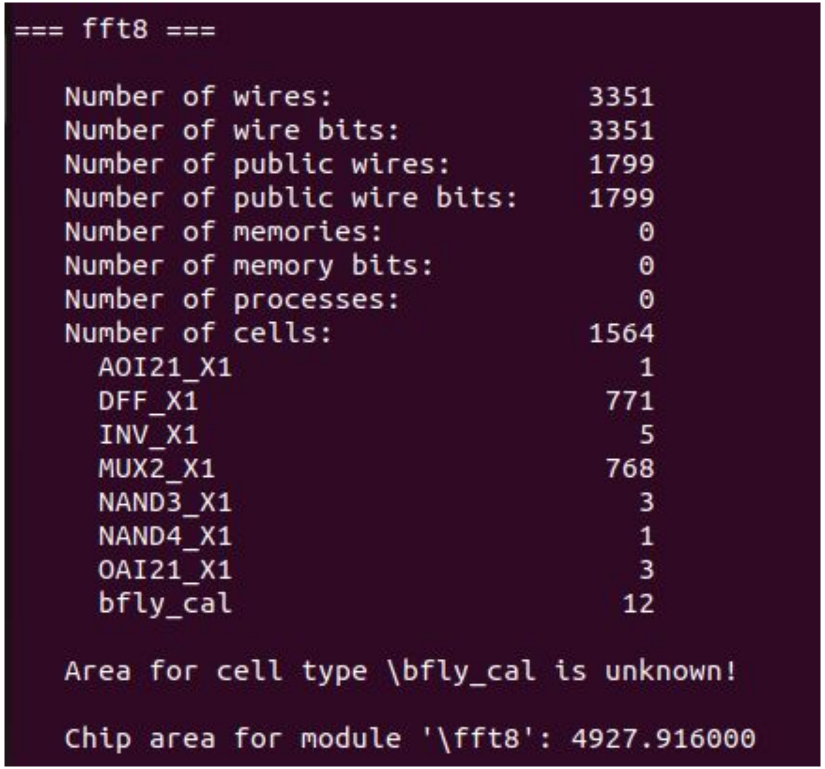
Test Bench Simulations

Performance Metrics

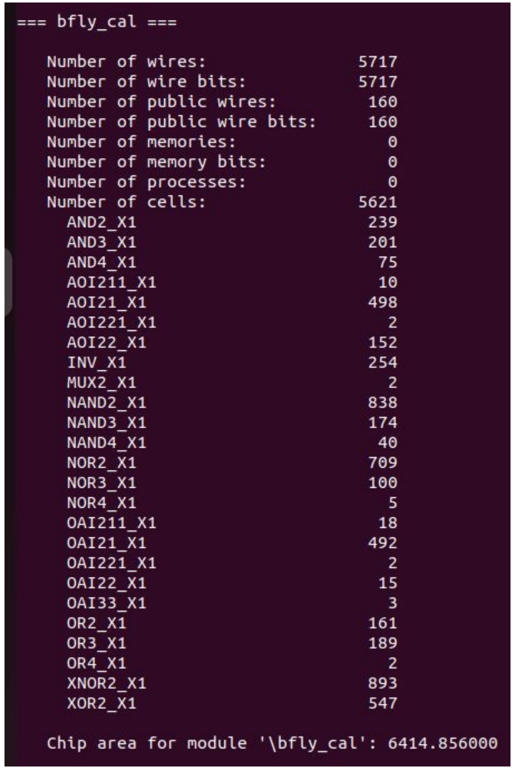
05

1. Create a file named fftdesign using command iverilog -o fftdesign testbench.v fft8.v
2. Run the Simulation using the command vvp fftdesign. To view the simulation waveform, run the command fft.vcd
3. To Synthesize the Digital Circuit use the command yosys synthfft.ys. We can observe the Chip Area required to design the modules.
4. Run the sta command to launch the OpenSTA shell and type the following commands:-
   ```
   read_liberty stdcells.lib
   read_verilog synthfft.v
   link_design fft
   create clock -name clk -period 5.5 {clk]
   set_power_activity -input -activity 0.5
   set_power_activity -global -activity 0.5
   ```
5. Type the report_checks command to display timing analysis.
6. Run the report_power command to display power estimation results.

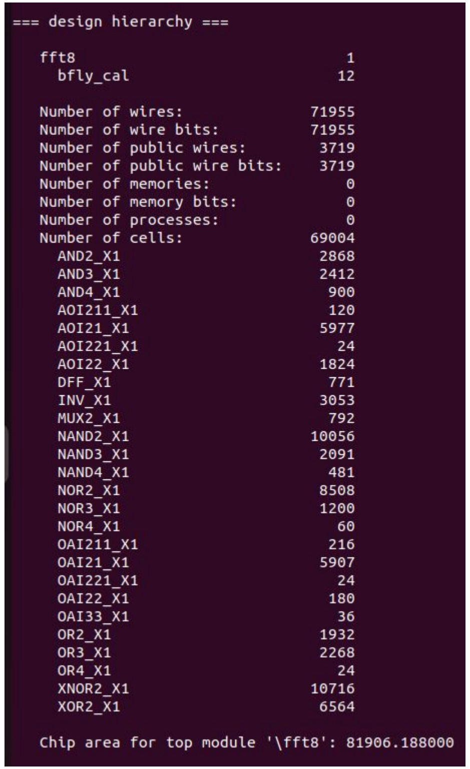
Area of Synthesized Design



```
=== fft8 ===

   Number of wires:                 3351
   Number of wire bits:             3351
   Number of public wires:          1799
   Number of public wire bits:      1799
   Number of memories:                 0
   Number of memory bits:              0
   Number of processes:                0
   Number of cells:                 1564
     AOI21_X1                          1
     DFF_X1                          771
     INV_X1                            5
     MUX2_X1                         768
     NAND3_X1                          3
     NAND4_X1                          1
     OAI21_X1                          3
     bfly_cal                         12

   Area for cell type \bfly_cal is unknown!

   Chip area for module '\fft8': 4927.916000
```

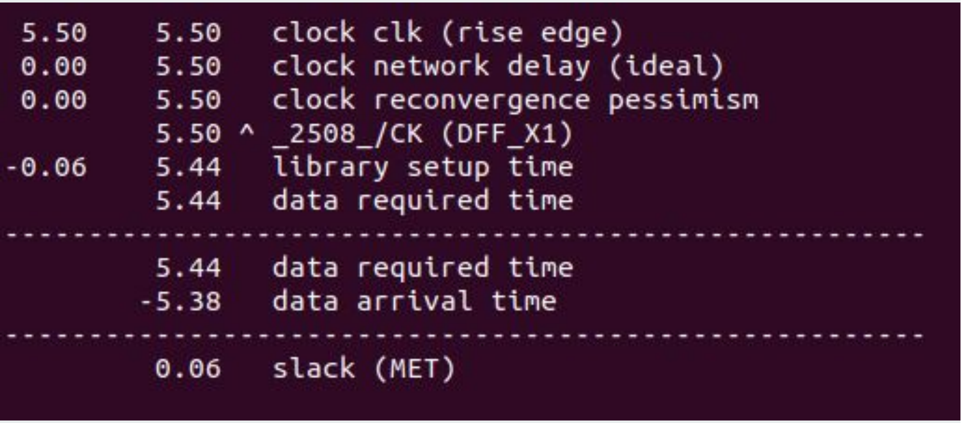


```
=== bfly_cal ===

   Number of wires:                 5717
   Number of wire bits:             5717
   Number of public wires:           160
   Number of public wire bits:       160
   Number of memories:                 0
   Number of memory bits:              0
   Number of processes:                0
   Number of cells:                 5621
     AND2_X1                         239
     AND3_X1                         201
     AND4_X1                          75
     AOI211_X1                        10
     AOI21_X1                        498
     AOI221_X1                         2
     AOI22_X1                        152
     INV_X1                          254
     MUX2_X1                           2
     NAND2_X1                        838
     NAND3_X1                        174
     NAND4_X1                         40
     NOR2_X1                         709
     NOR3_X1                         100
     NOR4_X1                           5
     OAI211_X1                        18
     OAI21_X1                        492
     OAI221_X1                         2
     OAI22_X1                         15
     OAI33_X1                          3
     OR2_X1                          161
     OR3_X1                          189
     OR4_X1                            2
     XNOR2_X1                        893
     XOR2_X1                         547

   Chip area for module '\bfly_cal': 6414.856000
```



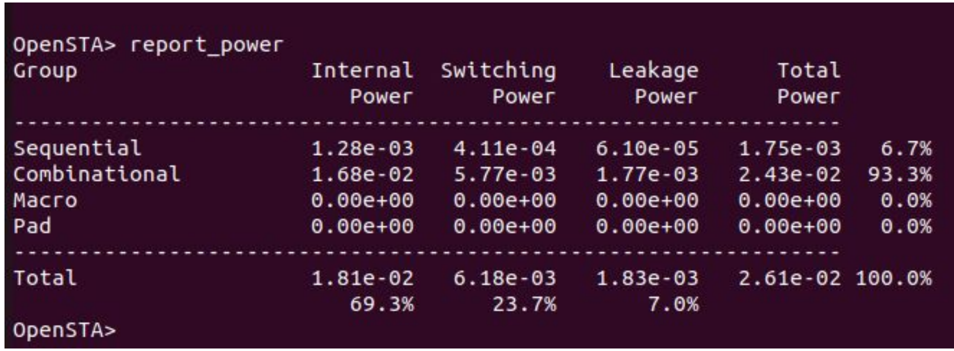
```
=== design hierarchy ===

   fft8                              1
     bfly_cal                       12

   Number of wires:              71955
   Number of wire bits:          71955
   Number of public wires:        3719
   Number of public wire bits:    3719
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:              69004
     AND2_X1                      2868
     AND3_X1                      2412
     AND4_X1                       900
     AOI211_X1                     120
     AOI21_X1                     5977
     AOI221_X1                      24
     AOI22_X1                     1824
     DFF_X1                        771
     INV_X1                       3053
     MUX2_X1                       792
     NAND2_X1                    10056
     NAND3_X1                     2091
     NAND4_X1                      481
     NOR2_X1                      8508
     NOR3_X1                      1200
     NOR4_X1                        60
     OAI211_X1                     216
     OAI21_X1                     5907
     OAI221_X1                      24
     OAI22_X1                      180
     OAI33_X1                       36
     OR2_X1                       1932
     OR3_X1                       2268
     OR4_X1                         24
     XNOR2_X1                    10716
     XOR2_X1                      6564

   Chip area for top module '\fft8': 81906.188000
```

Max Clock Frequency



```
 5.50    5.50   clock clk (rise edge)
 0.00    5.50   clock network delay (ideal)
 0.00    5.50   clock reconvergence pessimism
         5.50 ^ _2508_/CK (DFF_X1)
-0.06    5.44   library setup time
         5.44   data required time
-------------------------------------------------------
         5.44   data required time
        -5.38   data arrival time
-------------------------------------------------------
         0.06   slack (MET)
```

Total Power Consumption



```
OpenSTA> report_power
Group                  Internal  Switching    Leakage      Total
                          Power      Power      Power      Power
----------------------------------------------------------------
Sequential             1.28e-03   4.11e-04   6.10e-05   1.75e-03   6.7%
Combinational          1.68e-02   5.77e-03   1.77e-03   2.43e-02  93.3%
Macro                  0.00e+00   0.00e+00   0.00e+00   0.00e+00   0.0%
Pad                    0.00e+00   0.00e+00   0.00e+00   0.00e+00   0.0%
----------------------------------------------------------------
Total                  1.81e-02   6.18e-03   1.83e-03   2.61e-02 100.0%
                          69.3%      23.7%       7.0%
OpenSTA>
```

Timeline

STEP-1	STEP-2	STEP-3	STEP-4	STEP-5	STEP-6	STEP-7
Understanding DFT	DIT-FFT Algorithm	Functional Verification (MATLAB Simulation)	Verilog Code	Test Bench in Verilog	Implementation Metrics	Design Trade-offs

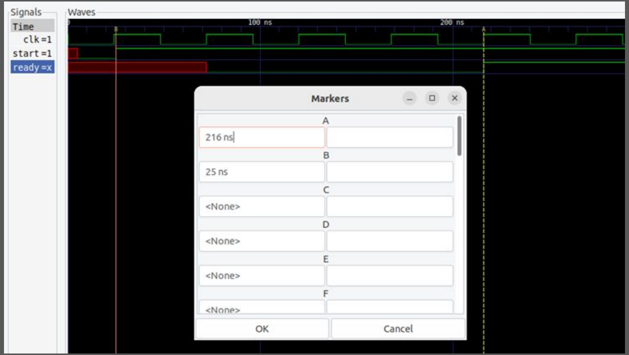
Analysis

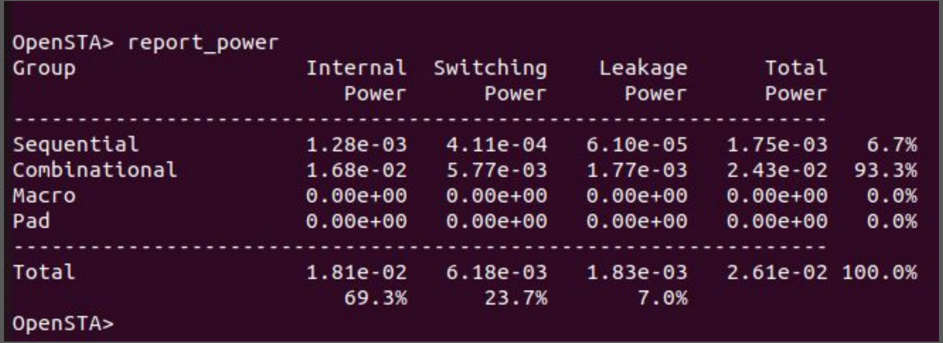
1. What is the Number of Clock Cycles required per FFT Computation?
   4 Clock Cycles [191 nsec]

2. What is the Area of the Synthesized Design?
   81906.188 um^2 for fft8 module

3. What is the Maximum Clock Frequency supported by the synthesized design?
   T_{min}=5.5nsec, F_{max} = 181MHz

4. What is the Total Power Consumption of the synthesized design when operating at its maximum clock frequency?
   Internal Power:- 18.1mW (69.3%)
   Switching Power:-6.18mW (23.7%)
   Leakage Power:-1.83mW (7%)
   Total Power:-26.11mW





```
OpenSTA> report_power
Group                  Internal  Switching    Leakage      Total
                          Power      Power      Power      Power
----------------------------------------------------------------
Sequential             1.28e-03   4.11e-04   6.10e-05   1.75e-03   6.7%
Combinational          1.68e-02   5.77e-03   1.77e-03   2.43e-02  93.3%
Macro                  0.00e+00   0.00e+00   0.00e+00   0.00e+00   0.0%
Pad                    0.00e+00   0.00e+00   0.00e+00   0.00e+00   0.0%
----------------------------------------------------------------
Total                  1.81e-02   6.18e-03   1.83e-03   2.61e-02 100.0%
                          69.3%      23.7%       7.0%
OpenSTA>
```

Analysis

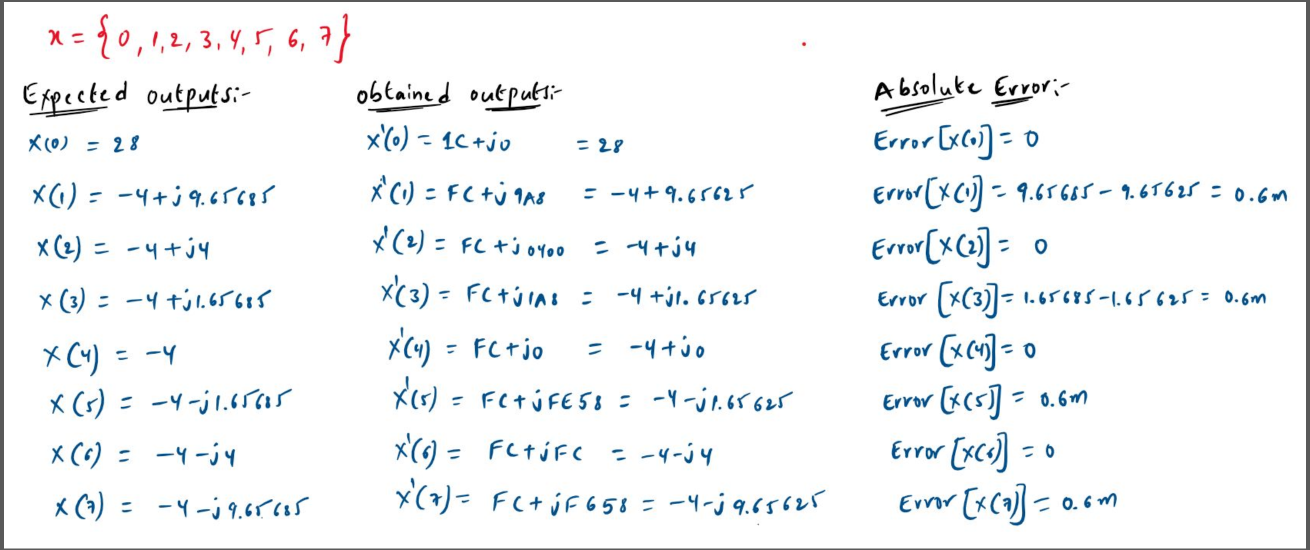
5. What is the Total Energy Consumption of the Synthesized Design per FFT Computation?
   Energy = Power * Time
   Energy= 26.1mW * 191ns
   = 4.98nJ

6. How does the Total Energy Consumption of the Synthesized design depend on the Clock Frequency?
   Energy = Power * Time
   = Power/ Frequency
   So, as Clock Frequency Increases, Energy decreases.

7. What is the Process Corner, Supply Voltage and Operating Temperature at which the above metrics have been simulated?
   We have used Nangate Typical Process Corner as standard cell library.
   For Typical Process:-
   Supply Voltage=1.1V
   Temperature=25 $^{\circ}$C

Analysis

8. What is the maximum absolute error when computing FFT of {0,1,2,3,4,5,6,7}?
Maximum Absolute Error= 0.0006

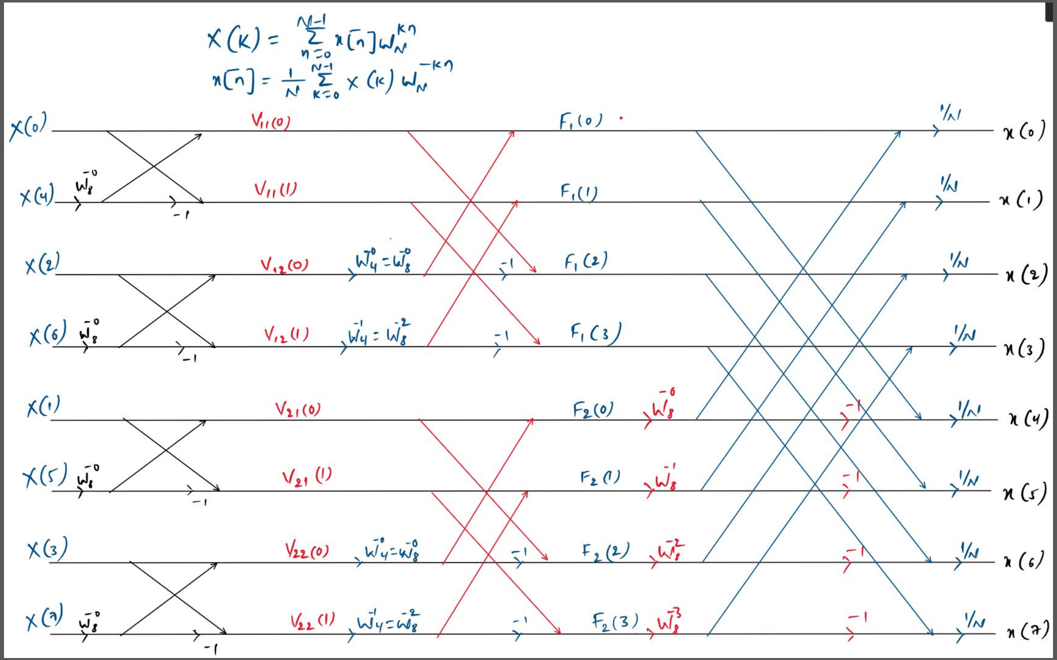


$x = \{0, 1, 2, 3, 4, 5, 6, 7\}$

Expected outputs:-	obtained outputs:-	Absolute Error:-
$X(0) = 28$	$X'(0) = 1C + j0 = 28$	$\text{Error}[X(0)] = 0$
$X(1) = -4 + j9.65685$	$X'(1) = FC + j9A8 = -4 + 9.65625$	$\text{Error}[X(1)] = 9.65685 - 9.65625 = 0.6m$
$X(2) = -4 + j4$	$X'(2) = FC + j0400 = -4 + j4$	$\text{Error}[X(2)] = 0$
$X(3) = -4 + j1.65685$	$X'(3) = FC + j1A8 = -4 + j1.65625$	$\text{Error}[X(3)] = 1.65685 - 1.65625 = 0.6m$
$X(4) = -4$	$X'(4) = FC + j0 = -4 + j0$	$\text{Error}[X(4)] = 0$
$X(5) = -4 - j1.65685$	$X'(5) = FC + jFE58 = -4 - j1.65625$	$\text{Error}[X(5)] = 0.6m$
$X(6) = -4 - j4$	$X'(6) = FC + jFC = -4 - j4$	$\text{Error}[X(6)] = 0$
$X(7) = -4 - j9.65685$	$X'(7) = FC + jF658 = -4 - j9.65625$	$\text{Error}[X(7)] = 0.6m$

Analysis

9. How can the FFT Circuit be modified to compute Inverse FFT
   From the DFT Formula we can observe that by changing the Twiddle Factor Values and at the last step Divide the Value by N, we can obtain the Inverse FFT.



References

The intuition behind Fourier and Laplace transforms

But what is the Fourier Transform?

The Fast Fourier Transform (FFT): Most Ingenious Algorithm Ever?

CMLab, NTU

Radix-4 DIF FFT Algorithm- Texas Instruments

Detailed Concept of FFT

Hardware Modelling Using Verilog - Prof. Indranil Sengupta

Verilog Tutorials

Team Members

Sri Sai Nomula

SR NO:- 22986
srisainomula@iisc.ac.in

Contributions:-

- FFT Theory
- Verilog Code (Algorithm for FFT, Butterfly Calculations)
- Performance Calculations
- Document Preparation

Sai Charan Katakam

SR NO:-22845
saicharank@iisc.ac.in

Contributions:-

- FFT Theory
- Verilog Code (Integrating Sequential Logic)
- Verilog TestBench