EE 714 Digital IC Design Exercise Report

2020, Fall

Supervisor: Prof. Kim, Jinsang
ID: 2015104027 Name: Park, Jung Jin

Design: A Single Path Delay 32 Point Radix-2 FFT

Abstract:

In DSP, *Digital Signal Processing*, the FFT, *Fast Fourier Transform*, is one of the most fundamental and essential systems. The FFT reduces on a high computational effort which the DFT, *Discrete Fourier Transform*, required. However, whereas the software implementation of the FFT can be achieved quickly, the FFT in hardware takes some issues: taking large hardware resources, and less straightforward in its implementation. Even though we take account of these issues, the hardware FFT reaps many benefits in terms of high performance. The software FFT is severely constrained to execute instructions serially, and therefore the throughput of the processor (generally CPU) is not efficient. However, the hardware FFT performs its tasks in parallel, so high throughput can be achieved, and of course, it is useful for high-speed real-time DSP. In this report, I focused on the Cooley-Tukey Radix-2 FFT algorithm, which is also called Butterfly. Also, in various FFT architectures, I selected R2SDF, Radix-2 Single Delay Feedback. I review the simple mathematical basis of the algorithm, the operation of R2SDF, and the synthesis of the architecture.

I. Introduction:

The DFT is a very crucial mathematical algorithm in the digital domain. The DFT can be defined as:

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

Also, from the equation, we can define W_N, which is known as Twiddle Factor or Nth root-of-unity. The twiddle factor expresses as the complex value phase factor. This twiddle factor can be defined as:

$$W_N = e^{-j\left(\frac{2\pi}{N}\right)}$$

Hence, $X[k]$ can be re-defined as:

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}, k = 0,1,2,\cdots,N-1$$

However, observing the equation of the DFT, N^2 complex multiplications and $N(N-1)$ complex addition is required for the computational result. Hence, its Big O is $O(N^2)$, which indicates that the DFT requires high effort. For the sake of reducing the computation complexity of the DFT algorithm, some mathematical tricks have been used to the DFT from the perspective of convenience and efficiency. Consequently, due to the tricks, the FFT can be the numerically efficient algorithms to compute the DFT.

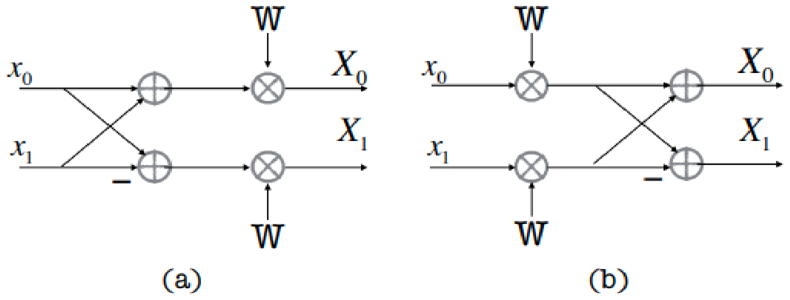
The FFT algorithm is based on the concept of Divide and Conquer. Cooley and Tukey have first introduced the idea of FFT by making effective use of symmetry and periodicity properties of the twiddle factors.

$$W_N^{k+\frac{N}{2}} = -W_N^k \ (Symmetry)$$

$$W_N^{k+N} = W_N^k \ (Periodicity)$$

With these properties, $x[n]$ is decomposed into smaller sequences until making 2-point DFT. At the end of the decomposition, each sequence computes its DFT and combine its results. Due to this approach, the FFT's big O can be $O(Nlog_2N)$, which indicates that the computational cost is significantly decreased compared to the DFT.

Before knowing how the sequences are decomposed, we should classify DIT, *Decimation In Tim*e, and DIF, *Decimation in Frequency*. The DIT is that the sequences in time domain are divided into smaller sequences. On the contrary, the DIF is that the sequences in frequency domain are divided into smaller sequences. To be specific, the DIT separates the sequences in time domain into even and odd, whereas the DIF separates the sequences, which are the results of the Fourier Transform, into even and odd. Also, the difference between these two algorithms is the location of the twiddle factors. The figure below illustrates the different positions of the twiddle factors, respectively. The position of twiddle factors in (a) is radix-2 DIF butterfly and in (b) is radix-2 DIT butterfly.



Both the algorithms are having the same computational effort, but they are different in input and output computational arrangement. Hence, the position of a Sort module, which is explicated later, is different, like the different position of twiddle factors. In this report, I take radix-2 DIF butterfly. Also, I prove the algorithm simply to understand how the algorithm is worked.

$$X[k] = \sum_{n=0}^{N-1} x[n]W_N^{kn}, k = 0,1,2,\cdots, N-1$$

$$X[k] = \sum_{n=0}^{N/2-1} x[n]W_N^{kn} + \sum_{n=N/2}^{N-1} x[n]W_N^{kn} \ (\because \ separate\ 0\sim N-1\ to\ 0\sim N/2-1\ and\ N/2\ \sim to\ N-1)$$

$$X[k] = \sum_{n=0}^{N/2-1} x[n]W_N^{kn} + W_N^{k(N/2)} \sum_{n=0}^{N/2-1} x\left[n+\frac{N}{2}\right]W_N^{kn} \ (\because \ n\text{--}> \ n+N/2)$$

$$X[k] = \sum_{n=0}^{N/2-1} (x[n] + (-1)^k x\left[n+\frac{N}{2}\right])W_N^{kn} \ \left(\because \ W_N^{k(N/2)} \ = \ (-1)^k\right)$$

Consequently, we divide the equation into even and odd,

$$X[2r] = \sum_{n=0}^{N/2-1} (x[n] + x\left[n+\frac{N}{2}\right])W_N^{2rn}$$

$$X[2r+1] = \sum_{n=0}^{N/2-1} (x[n] - x\left[n+\frac{N}{2}\right])W_N^{(2r+1)n} = \sum_{n=0}^{N/2-1} (x[n] - x\left[n+\frac{N}{2}\right])W_N^{2rn}W_N^n$$

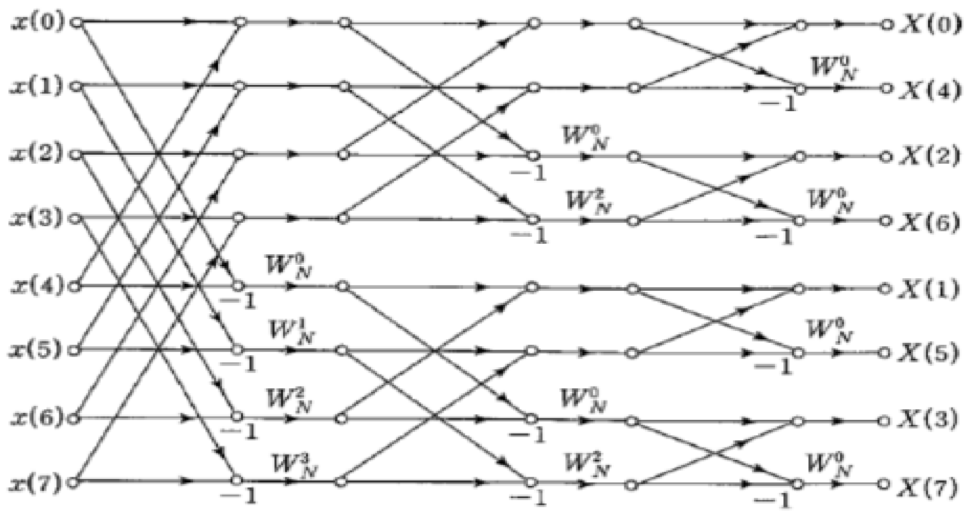
By the property of the twiddle factors,

$$W_N^{2rn} = W_{N/2}^n$$

$$X[2r] = \sum_{n=0}^{N/2-1} (x[n] + x\left[n+\frac{N}{2}\right])W_{N/2}^n$$

$$X[2r+1] = \sum_{n=0}^{N/2-1} [(x[n] - x\left[n+\frac{N}{2}\right])W_N^n]W_{N/2}^n$$

As can be seen, the N-point DFT is divided into N/2-point DFT. Based on this prove, the FFT can be represented by the below figure. The figure is 8-Point DIF - FFT.

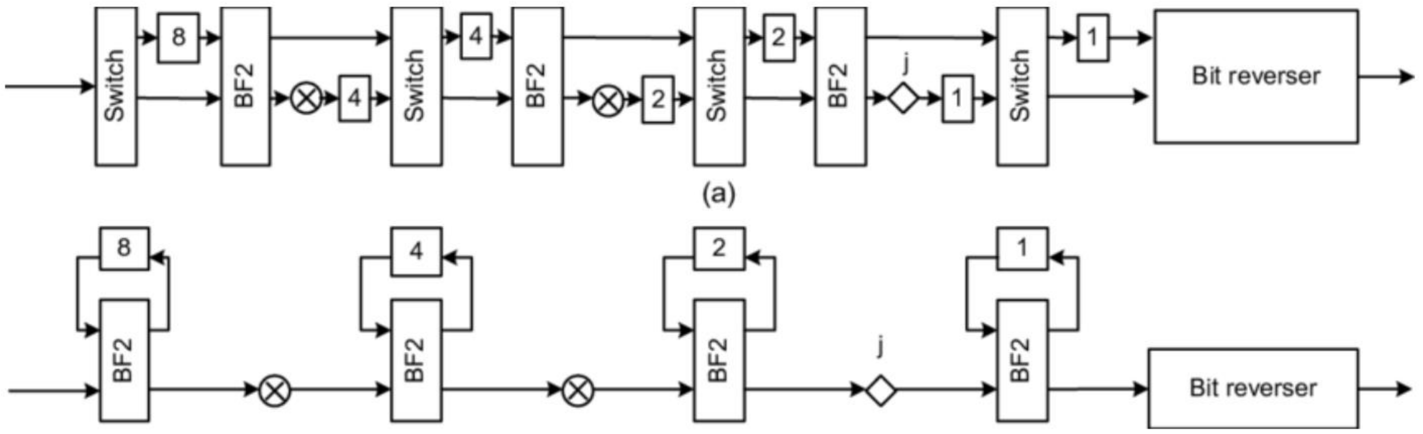


From this proof, we also can find the meaning of the radix-2. The name of the radix-2 is called since the number of inputs of the radix-2 FFT must be 2^M. For example, 32 can be represented 2^5, so the 32-point FFT is divided into five stages.

II. Hardware Implementation

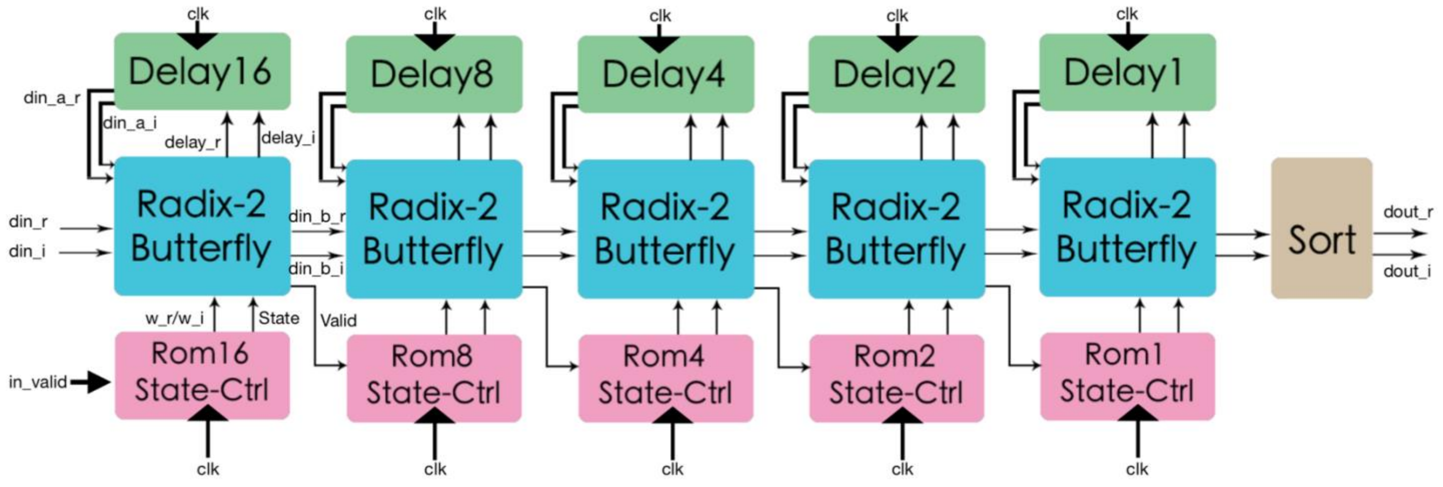
As following Section I, we know that a 32-point DIF FFT requires five stages. In this section, we review how the FFT can be implemented in terms of hardware.

Basically, for the FFT, the hardware requires adders, multiplications, and a memory that stores twiddle factors. The below figure shows the conventional pipelined structures for the FFT. The upper one is called R2MDC, *Radix-2 Multiple Delay Commutator*, and the lower one is called R2SDC, *Radix-2 Single Delay Feedback*. The Bit reverser is the "Sort module" that I mentioned above.



The R2MDC breaks the input sequence into two parallel data streams flowing forward with correct distance between data elements and the butterfly scheduled proper delays. The R2SDF makes a single data stream that goes through the butterfly at every stage. In terms of performance, the R2MDC is better than the R2SDF. However, in terms of utilization, hardware resources, the R2SDF uses fewer registers than the R2MDC. Also, the R2SDC requires fewer input ports since the input data is stored in shift registers(delay function) for waiting next half sequences. For example, in 32-point FFT, the R2SDC input data is 16-point, and it means that the structure accepts the sequences twice.

Below figure is a specific block diagram of the R2SDC FFT.



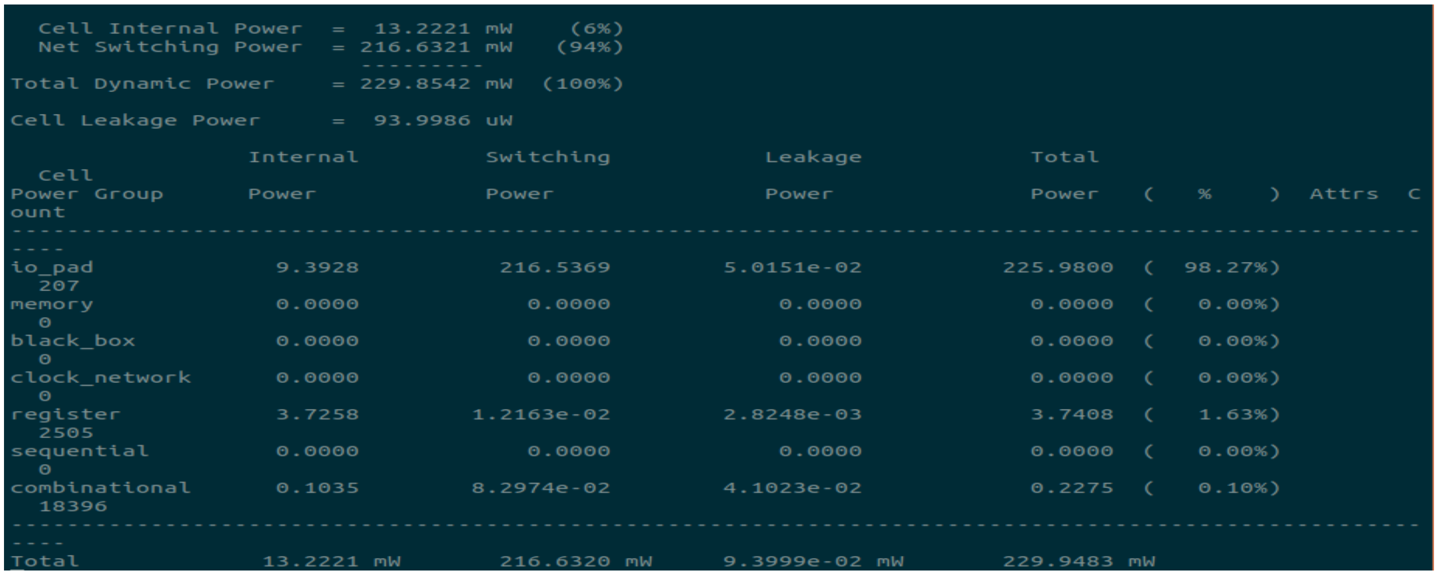
To illustrate the operation of the module, the Radix-2 Butterfly module plays a crucial role. The Radix-2 Butterfly module has three states: Waiting, First Half, and Second Half. The Waiting state is the state that waits next half data sequences, x[16]~x[31]. In this state, the first half of the data sequences send to the Delay module to calculate the DFT with the next half data sequences (e.g., x[0]+x[16]). The First Half state is the state that calculates the summation of two indexes, e.g., x[0](from the delay module) + x[16](from the input data stream). The Second Half state is the last state that multiplies the summation in the First half state and twiddle factors from the ROM. The Sort module rearranges the index of the outputs since we know the output signal order. It plays a role in placing the value in the right index. For example, the 2nd index of the last Radix-2 Butterfly module is X[15], so we place it to the 15th index of the real output.

III. Logic Synthesis

The referred code is synthesized by the UMC 180nm technology. However, in this report, I used the Samsung 65nm technology. The comparison of design specification was defined as below table.

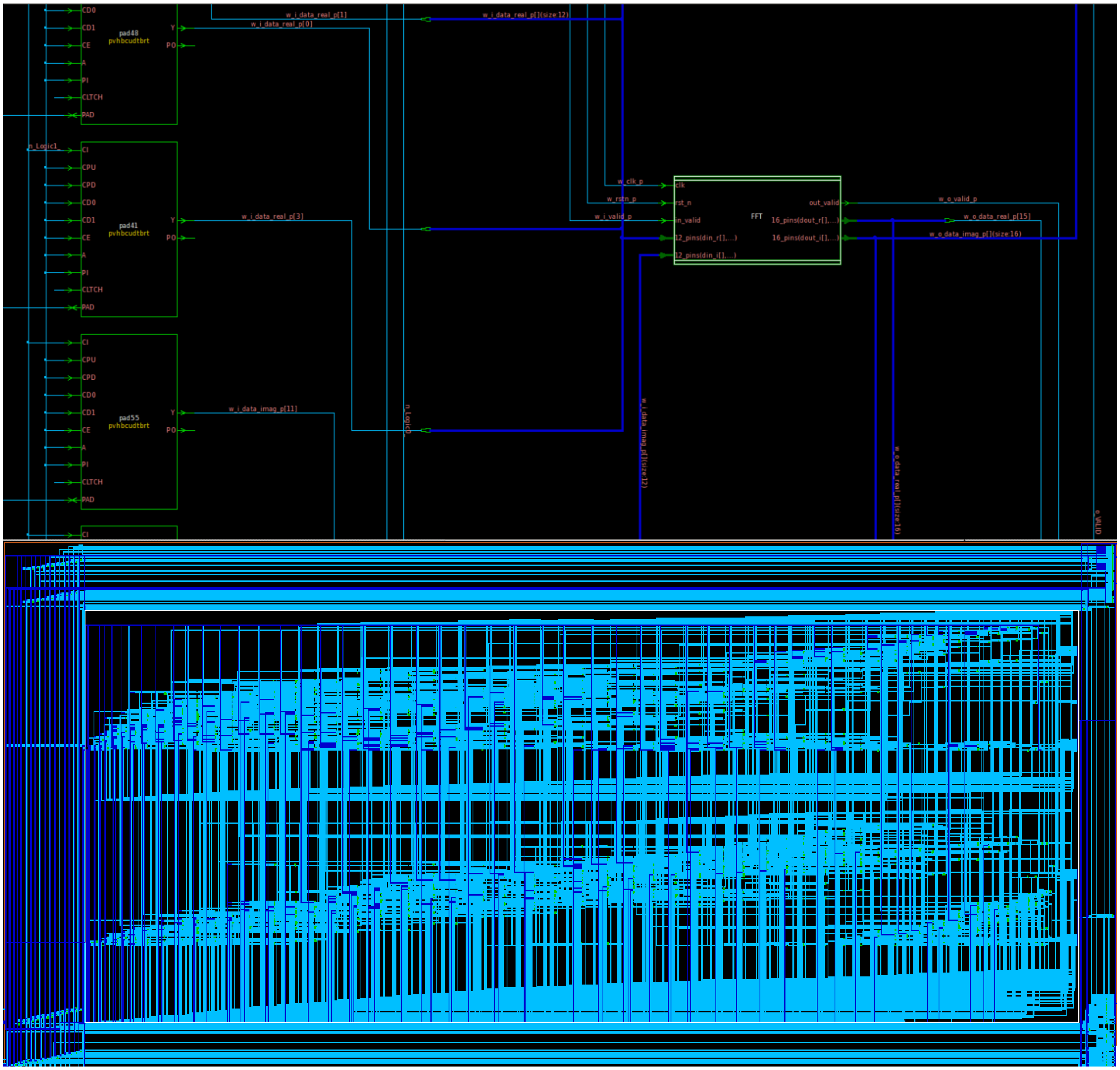
UMC 180nm		Samsung 65nm	
Spec	Value	Spec	Value
Main clock	100 MHz	Main clock	185 MHz
Area	486696.98 μm^2	Area	73540.80 μm^2
Power	10.77 mW	Power	229.95 mW

As can be seen in the comparison between the two tables, as the feature size is smaller, the speed is faster, and the area is smaller. However, the power is significantly increased. The below figure is the report of the power.


```
  Cell Internal Power  =  13.2221 mW    (6%)
  Net Switching Power  = 216.6321 mW   (94%)
                         ---------
Total Dynamic Power    = 229.8542 mW  (100%)

Cell Leakage Power     =  93.9986 uW

                 Internal         Switching           Leakage            Total
  Cell
Power Group      Power            Power               Power              Power   (   %    )  Attrs  C
ount
----------------------------------------------------------------------------------------------------
----
io_pad             9.3928          216.5369          5.0151e-02          225.9800  (  98.27%)
  207
memory             0.0000            0.0000              0.0000            0.0000  (   0.00%)
  0
black_box          0.0000            0.0000              0.0000            0.0000  (   0.00%)
  0
clock_network      0.0000            0.0000              0.0000            0.0000  (   0.00%)
  0
register           3.7258          1.2163e-02          2.8248e-03            3.7408  (   1.63%)
  2505
sequential         0.0000            0.0000              0.0000            0.0000  (   0.00%)
  0
combinational      0.1035          8.2974e-02          4.1023e-02            0.2275  (   0.10%)
  18396
----------------------------------------------------------------------------------------------------
----
Total             13.2221 mW       216.6320 mW       9.3999e-02 mW       229.9483 mW
```

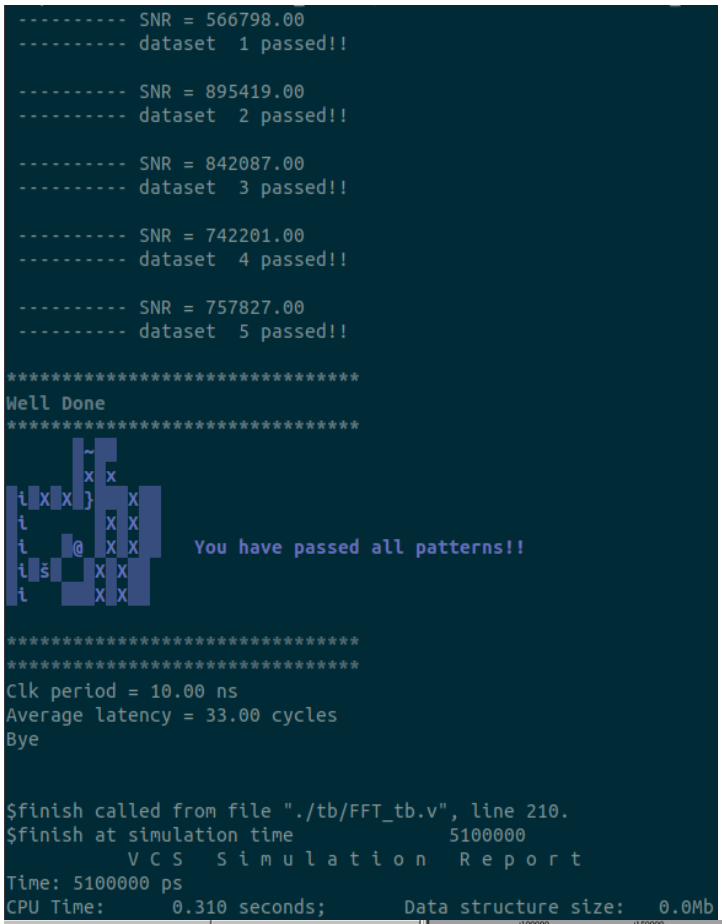
The report indicates that the (net) switching power takes up a great proportion of total power. The dynamic power is defined as the equation: $P = \alpha C_L V_{DD}^2 f_{sw}$. In the components of the dynamic power, as the scaling-down, the load capacitance C_L is getting lower, and the supply voltage V_{DD} remains the same or be a little smaller. However, the switching frequency f_{sw} is going to higher significantly. Hence, as scaling down the channel length, the dynamic power usually increases. To reduce the dynamic power, the design should reduce the activity factor α by clock gating. The clock has a high activity factor, $\alpha = 1$, so it considerably affects the dynamic power of the design. However, if the clock gating is adopted, the complexity of the timing analysis of the design can be an issue. Hence, in this report, the clock gating is not adopted.

Below figure is the schematic of the FFT module after logic synthesis.

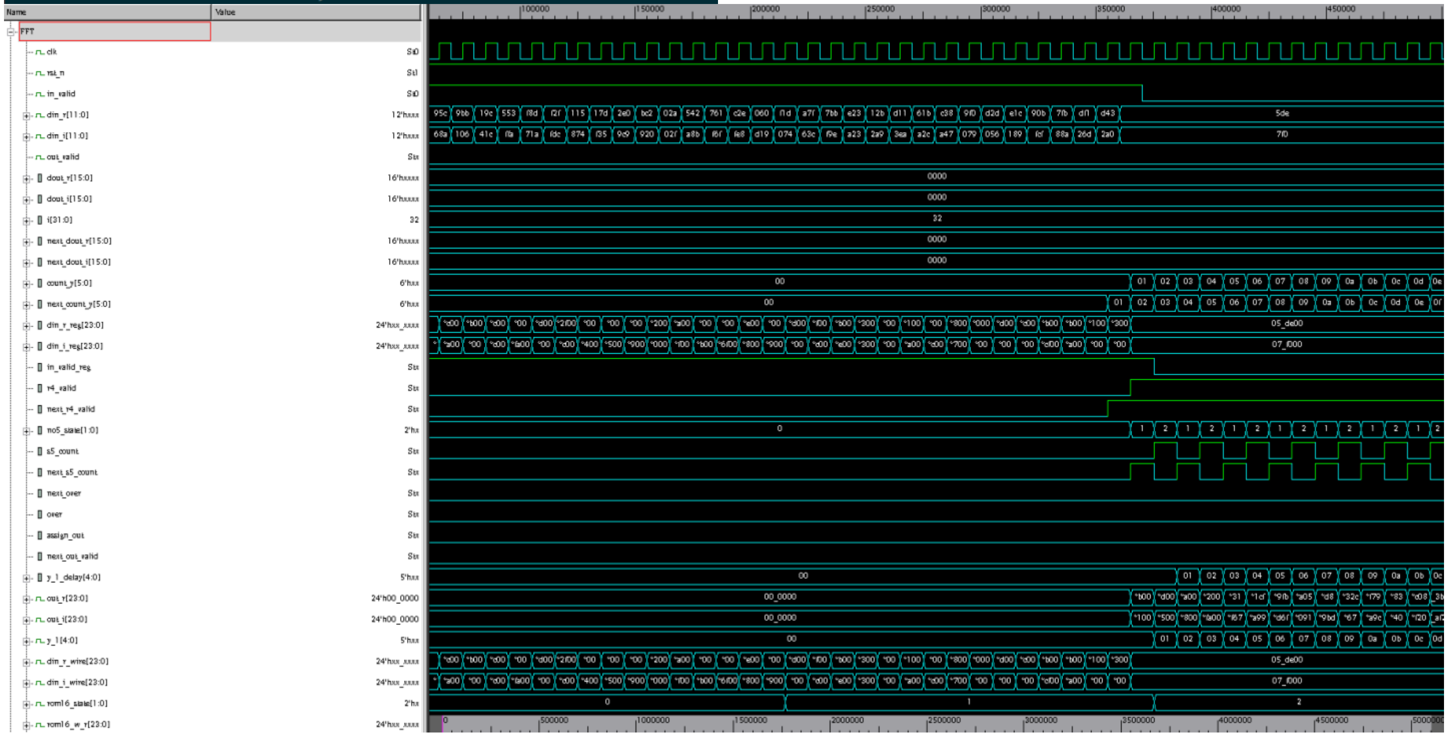


IV. Verification

After synthesis, the verification is essential. To verify, the tools, Formality and VCS, are used. Formality is a tool that checks whether the gate-level netlist is logically same function as the designed RTL model. VCS simulates the coded test bench and creates a waveform with a file extension called vcd. The test bench was coded to check the output in the specified test pattern. The output from the netlist is compared to the result from the FFT computation by software. The waveform of the gate-level netlist to which delay was applied by using sdf, *synopsys delay format*, was also checked. Below figures shows the result of the Formality and the VCS.



```
 ---------- SNR = 566798.00
 ---------- dataset  1 passed!!

 ---------- SNR = 895419.00
 ---------- dataset  2 passed!!

 ---------- SNR = 842087.00
 ---------- dataset  3 passed!!

 ---------- SNR = 742201.00
 ---------- dataset  4 passed!!

 ---------- SNR = 757827.00
 ---------- dataset  5 passed!!

*********************************
Well Done
*********************************
       ~
       x x
i x x }   x
i       x x
i    @  x x     You have passed all patterns!!
i š    x x
i      x x

*********************************
*********************************
Clk period = 10.00 ns
Average latency = 33.00 cycles
Bye


$finish called from file "./tb/FFT_tb.v", line 210.
$finish at simulation time              5100000
           V C S   S i m u l a t i o n   R e p o r t
Time: 5100000 ps
CPU Time:      0.310 seconds;       Data structure size:   0.0Mb
```



Reference:

[1] S. M. Khare, M. Z. Alam, "FPGA Based Design and Simulation of 32-Point FFT Through Radix-2 DIT Algorithm", IJERT, vol. 5, issue. 2, Feb. 2016.
[2] Gao, Y. (2015). *Hardware Implementation of a 32-point Radix-2 FFT Architecture* [Master's thesis, Lund University, Sweden]
[3] Slade, G. (2013). The Fast Fourier Transform in Hardware: A Tutorial Based on an FPGA Implementation.
[4] jasonlin316 2019, A-Single-Path-Delay-32-Point-FFT-Processor, GitHub, viewed 20 Nov 2020, <https://github.com/jasonlin316/A-Single-Path-Delay-32-Point-FFT-Processor>