2021 Digital IC Design

Homework 5: Frequency Analysis System

1 Introduction

In this homework, you are requested to design a system constructed with a Finite ***Impulse Response filter*** (***FIR filter***), a ***Fast Fourier Transform*** (***FFT***) circuit and an ***Analysis*** circuit. This system can filter out the noise with ***FIR Filter*** and then transform the signals from time domain into frequency domain with ***FFT*** circuit. Finally, the main frequency band of the signal can be found out with ***Analysis*** circuit, which can be applied as a sensing system. The functionality of the system will be described in detail in the following parts.

2 Design Specifications

2.1 Block overview

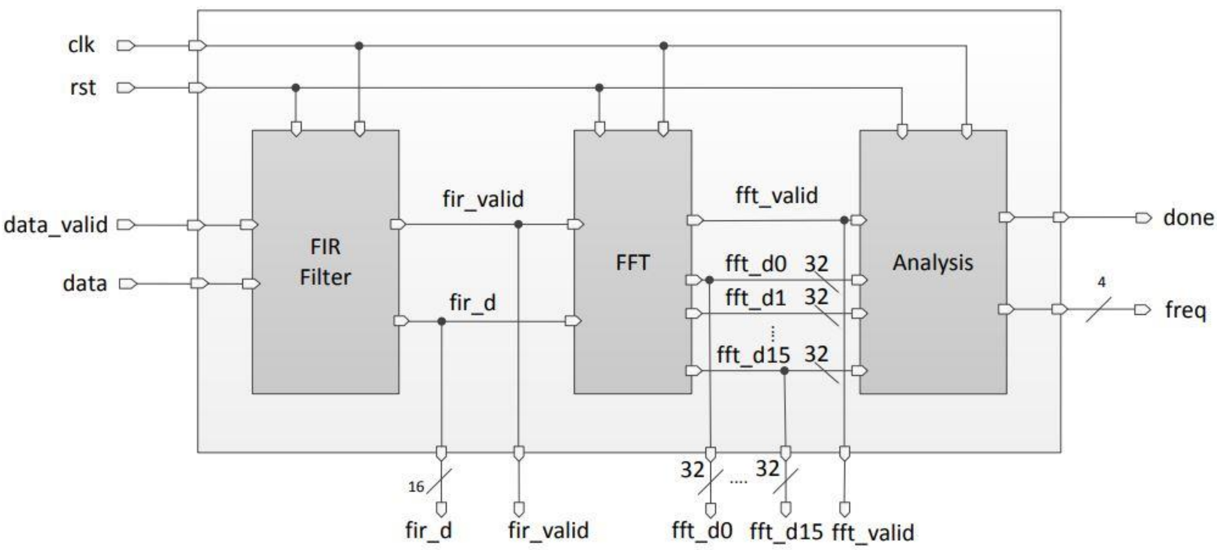

Fig. 1 – System block overview.

2.2 I/O Interface

Name	I/O	Width	Description
clk	I	1	System clock signal. This system is synchronized with the positive edge of the clock.
rst	I	1	Active-high asynchronous reset signal.

data_valid	I	1	When the host is ready to send data, this signal will be set as high.
data	I	16	Time domain signal from the host.
done	O	1	When the system complete calculation, this signal should be set as high.
fir_d	O	16	Output data signal of ***FIR filter***.
fir_valid	O	1	Data valid signal of ***FIR filter***.
fft_d0 ~ fft_d15	O	32	Output data signal of ***FFT***.
fft_valid	O	1	Data valid signal of ***FFT***.
freq	O	4	Output signal for the main frequency.

2.3 Function Description

The system gets input time domain signal from the host, the example is shown in figure 2. Then the noise in the input signal is filter out with ***FIR filter***, the filtered result of signal in figure 2 is shown in figure 3.

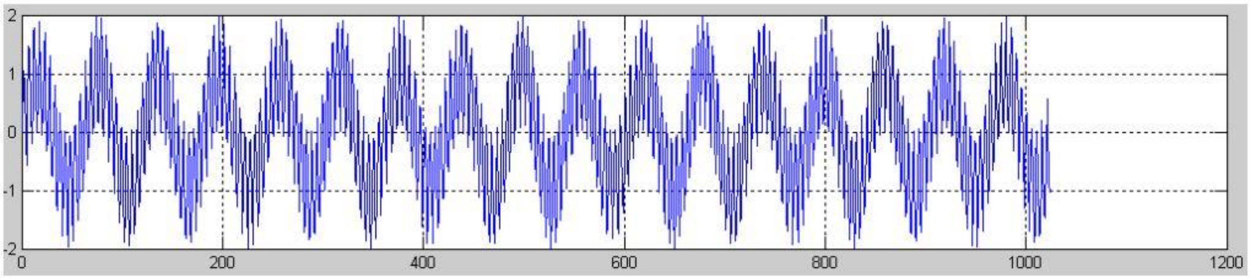

Fig. 2 – Time domain signals from the host.

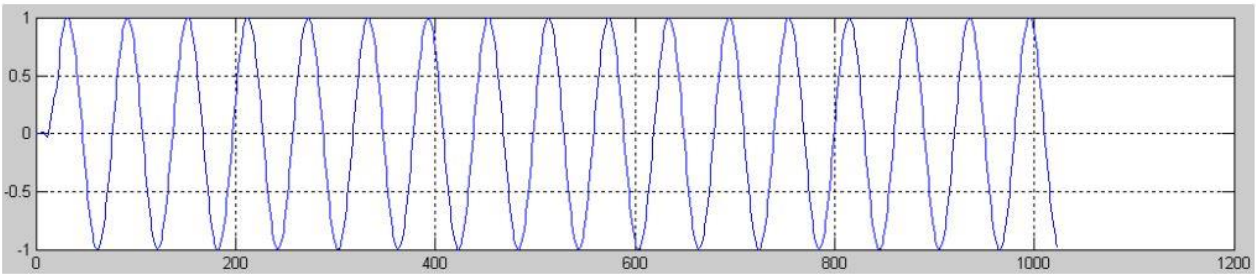

Fig. 3 – Signals which passed the ***FIR filter***.

After the input signal is filtered and is ready to output, pull up the *fir_valid* signal. And use *fir_d* signal to transmit one data in each cycle. The filtered signal will then be processed by ***FFT***, and the frequency domain signal can be obtained. (Take figure 4 as example.) Pull up *fft_valid* signal and use *fft_d* signal to transmit one set of data (*fft_d0 ~ fft_d15*) to the Analysis circuit. When the Analysis circuit complete calculation, pull up done signal and output the main frequency band by *freq* signal. Among the signals *fft_d0 ~ fft_d15*, *fft_d0* represents the frequency band 0, *fft_d1* represents the frequency band 1, and so on.

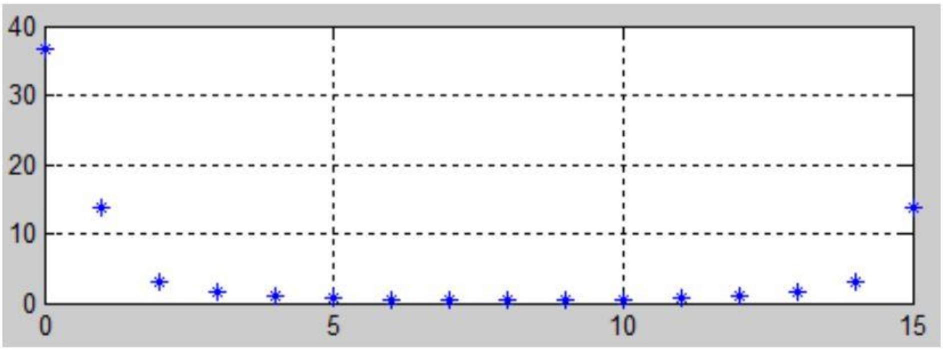

Fig. 4 – Spectrum diagram of signals which are processed by ***FFT***.

2.4 Timing Diagram

The timing specification of this design contains four parts: The system timing specification, the input signal timing specification, the ***FIR*** output timing specification, and the ***FFT*** output timing specification.

2.4.1 System timing specification

The timing diagram in figure 5 shows the system timing specification. After the system is reset, the serial input signals pass ***FIR filter*** and output by *fir_d* serially. Every sixteen output signals from ***FIR filter*** will be parallel input to the ***FFT*** circuit. The ***FFT*** circuit will output the processed signal parallel with signals *fft_d0 ~ fft_d15*. The Analysis circuit takes these signals and processes them to find the main frequency band. Finally, the main frequency band is output with *freq* signal. The done signal has to be pulled up at the same time to inform the host that the set of sixteen signals have been processed. The *fir_valid*, *fft_valid*, and *done* signals all maintain as high for 1 cycle for each valid output data. Besides, there is no overlap between any two sets of valid output of ***FIR filter***. For example, the first set of ***FFT*** parallel input is constructed with *fir_d(0) ~ fir_d(15)*, and the second set of ***FFT*** parallel input will be constructed with *fir_d(16) ~ fir_d(31)*, and so on. In this homework, the host will input 1024 data to the system, so there will be 64 calculation results output by the system.

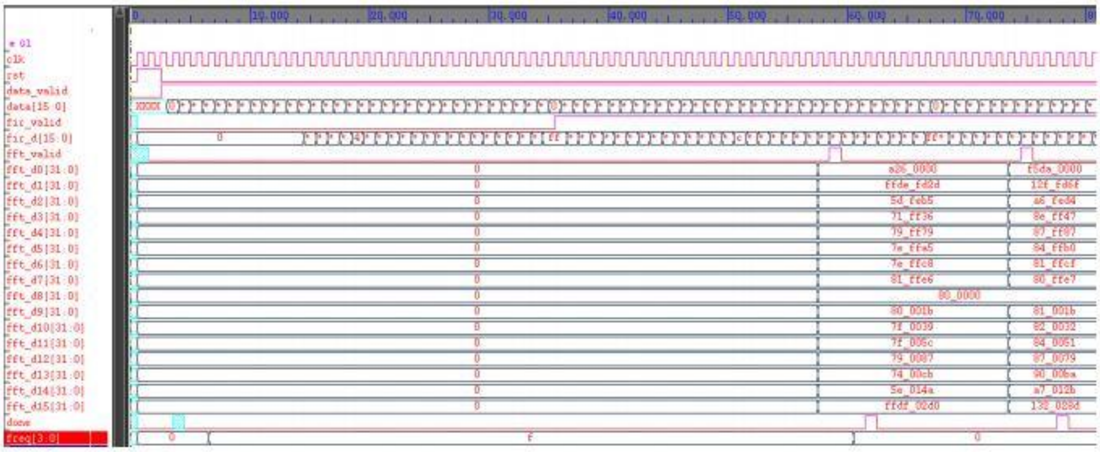

Fig. 5 – Timing diagram of the system.

2.4.2 Input signal timing specification

When the *data_valid* signal is set as high by the host, the *data* port will send one data in each cycle. Its timing specification is shown in figure 6. t_{CYCLE} represents the clock width of the system. The width of *data* signal is 16 bits, which is constructed with a sign bit, 7 bits of integer data, and 8 bits of floating number data. The construction of data is shown in figure 10.

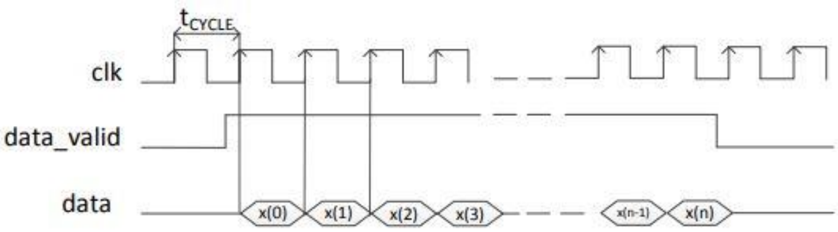

Fig. 6 – Timing diagram of the host data transmission.

2.4.3 ***FIR*** output timing specification

After the signals are filtered with ***FIR filter***, they will be output to the ***FFT*** circuit. When the data is transmitting, the *fir_valid* signal should be set as high. And the testbench will verify the ***FIR*** output synchronously. The output timing of ***FIR filter*** is shown in figure 7.

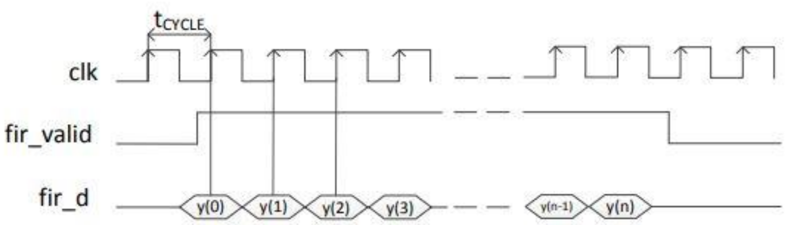

Fig. 7 – Timing diagram of ***FIR*** output data.

2.4.4 ***FFT*** output timing specification

After the FIR output data are processed with ***FFT*** circuit, they will be output to the ***Analysis*** circuit. When the data is transmitting, the *fft_valid* signal should be set as high. And the testbench will verify the ***FFT*** output synchronously. The output timing of ***FFT*** circuit is shown in figure 8.

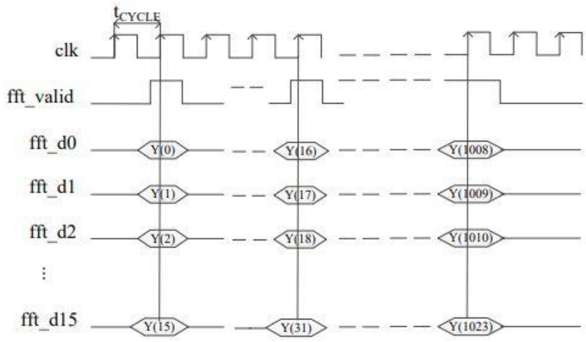

Fig. 8 – Timing diagram of ***FFT*** output data.

2.5 Functionality of ***FIR filter***

The ***FIR filter*** in the system is a low pass filter with 32 coefficients, and it is responsible for filtering out the high frequency noise. The coefficients of the filter are fixed, and they are shown in table 2. (They are also stored in the file "FIR_coefficient.dat".) The first valid output will be calculated after the thirty-second data is input to the ***FIR filter***. Equation 1 shows the calculation process of ***FIR filter***. Figure 9 shows its hardware architecture, and figure 10 shows the format of input *data* and output *fir_d*.

$$y(n-(N-1)) = \sum_{k=0}^{N-1} h(k)\ x(n-k) \qquad \text{(equation 1)}$$

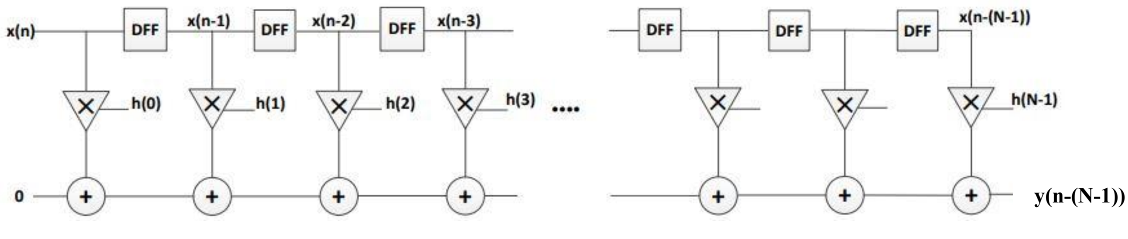

Fig. 9 – Hardware architecture of ***FIR filter***.

Low-pass Filter Coefficient (h)			
h(0)	-0.001505748051548	h(16)	0.229154203266836
h(1)	-0.001868548463782	h(17)	0.186019113110601
h(2)	-0.001366448872269	h(18)	0.115911664195512
h(3)	9.086560980884849e-04	h(19)	0.043365841112764
h(4)	0.005060978234550	h(20)	-0.009960448073993
h(5)	0.008948776436908	h(21)	-0.033966171510252
h(6)	0.008342764987706	h(22)	-0.032084609166509
h(7)	-4.333516292017973e-04	h(23)	-0.016526671675409
h(8)	-0.016526671675409	h(24)	-4.333516292017973e-04
h(9)	-0.032084609166509	h(25)	0.008342764987706
h(10)	-0.033966171510252	h(26)	0.008948776436908
h(11)	-0.009960448073993	h(27)	0.005060978234550
h(12)	0.043365841112764	h(28)	9.086560980884849e-04
h(13)	0.115911664195512	h(29)	-0.001366448872269
h(14)	0.186019113110601	h(30)	-0.001868548463782
h(15)	0.229154203266836	h(31)	-0.001505748051548

TABLE. 2 – Coefficients of low pass filter.

sign bit	integer	floating number
1 bit	7 bits	8 bits

Fig. 10 – Data format (*data, fir_d*)

2.6 Functionality of ***FFT*** circuit

In this system, you are requested to complete a sixteen-point fast Fourier transform. Its hardware architecture is shown in figure 11. The ***FFT*** circuit is used to transform the time domain signals into frequency domain signals for following analysis.

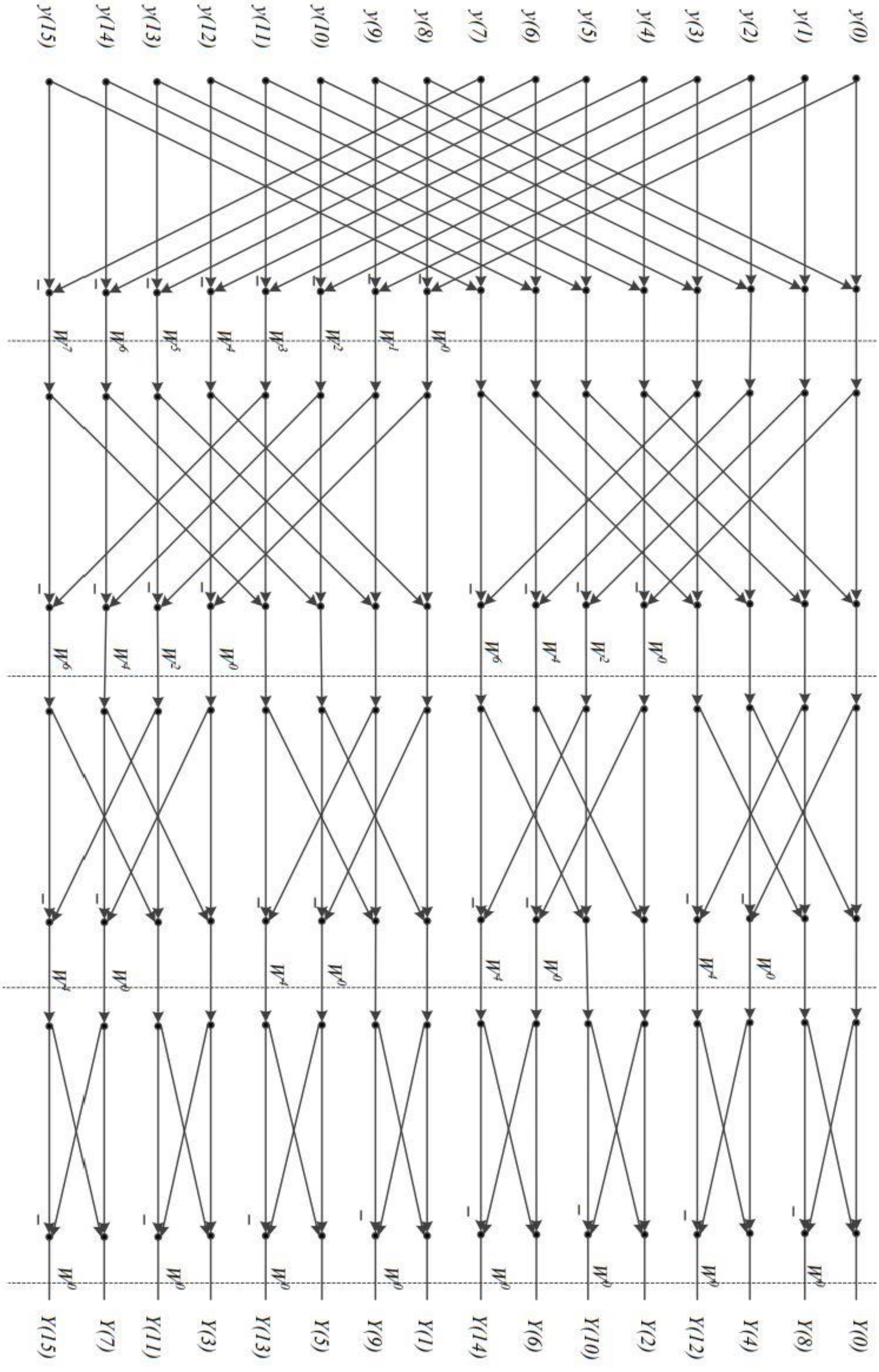

Fig. 11 – Hardware architecture of 16-point ***FFT*** circuit.

Figure 12 is an example of the ***FFT*** calculation process. The minus sign in the path of *fft_b* represents the calculation of subtract *Y* from *X*, and W^n is the ***FFT*** coefficient. The ***FFT*** coefficient contains real part (W^n*_real*) and imaginary part (W^n*_imag*). To obtain the results, the complex number operations have to be calculated. Figure 13 shows the result of *fft_b* after the multiplication.

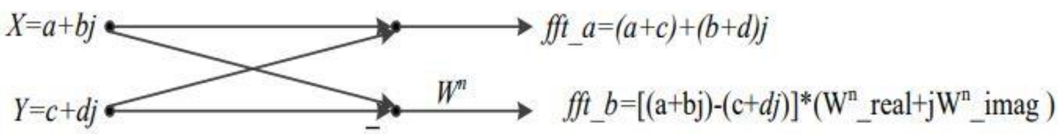

Fig. 12 – Example of ***FFT*** calculation.

Real part of *fft_b*	(a - c)*W^n*_real* + (d - b)*W^n*_imag*
Imaginary part of *fft_b*	(a - c)*W^n*_imag* + (b - d)*W^n*_real*

Fig. 13 – Multiplication result of *fft_b*.

The values of ***FFT*** coefficients are shown in table 3. The real part coefficients are stored in the file “Real_Value_Ref.dat”, and the imaginary part coefficients are stored in the file “Imag_Value_Ref.dat”. Figure 14 shows the data format of output of ***FFT*** circuit (*fft_d0* ~ *fft_d15*).

W^n			
W^0	1.000 + 0.000j	W^4	0.000 - 1.0000j
W^1	0.923879532511287 - 0.382683432365090j	W^5	-0.382683432365090 - 0.923879532511287j
W^2	0.707106781186548 - 0.707106781186548j	W^6	-0.707106781186548 - 0.707106781186547j
W^3	0.382683432365090 - 0.923879532511287j	W^7	-0.923879532511287 - 0.382683432365089j

TABLE. 3 – W^n coefficients for ***FFT*** process.

Signed bit	Integer of real part	Floating number of real part	Sign bit	Integer of imaginary part	Floating number of imaginary part
1 bit	7 bits	8 bits	1 bit	7 bits	8 bits

Fig. 14 – Data format (*fft_d0* ~ *fft_d15*).

Because the output from ***FIR filter*** is serial, a ***serial to parallel*** circuit has to be designed so that the ***FFT*** circuit can meet the timing specification. Figure 15 shows an example of the ***FFT*** circuit with the ***serial to parallel*** circuit.

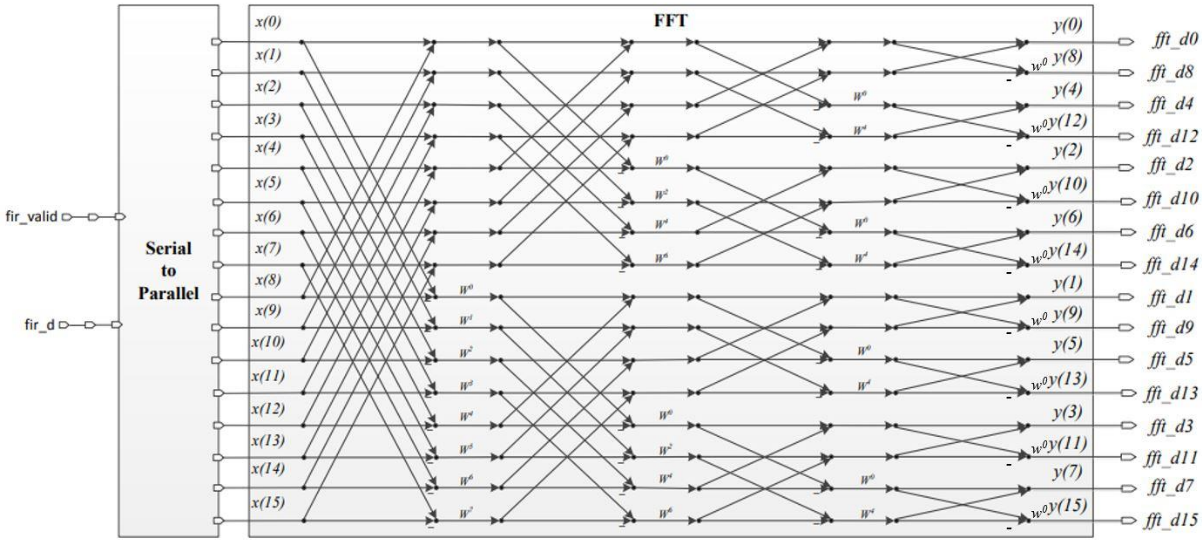

Fig. 15 – 16-point ***FFT*** circuit with the ***serial to parallel*** circuit.

2.7 Functionality of ***Analysis*** circuit

After the output of ***FFT*** circuit is input, the ***Analysis*** circuit has to find out the main frequency band. The definition of main frequency band is: the frequency whose sum of square of the real part data and imaginary part data is maximum. For example, Y(n) = a + bj (n = 0 ~ 15), then $a^2 + b^2$ has to be calculated for comparison. Finally, the main frequency band should be output with *freq* signal. If Y(2) has maximum sum of square, the *freq* should output 4’b0010.

3 Scoring

This homework has two testbench. To get full score, your design must be able to pass the two testbench both. If you just output the golden data without fulfilling the functionality of the system, you will get 0 points.

3.1 Functional Simulation **[60%]**

The simulation results of ***FIR*** output, ***FFT*** output, and ***Analysis*** results each account for 20% of score. If all of the results are generated correctly, you will get the following message in ModelSim simulation.

```
# -----------------------------------------------------
# FFT dataout on pattern        880 ~        895, PASS!!
# FFT dataout on pattern        896 ~        911, PASS!!
# FFT dataout on pattern        912 ~        927, PASS!!
# FFT dataout on pattern        928 ~        943, PASS!!
# FFT dataout on pattern        944 ~        959, PASS!!
# FFT dataout on pattern        960 ~        975, PASS!!
# FFT dataout on pattern        976 ~        991, PASS!!
# FFT dataout on pattern        992 ~       1007, PASS!!
# FFT dataout on pattern       1008 ~       1023, PASS!!
# -----------------------------------------------------
#
# Congratulations! All data have been generated successfully!
#
# -------------------------PASS------------------------
#
# ** Note: $finish    : C:/Users/user/Desktop/test_hw5/testfixture1.v(240)
#    Time: 53600 ns  Iteration: 0  Instance: /testfixture1
```

3.2 Gate Level Simulation **[30%]**

3.2.1 Synthesis

Your code should be synthesizable. After it is synthesized in Quartus, a file named *FAS.vo* will be obtained.

3.2.2 Simulation

The simulation results of ***FIR*** output, ***FFT*** output, and ***Analysis*** results each account for 10% of score. If all of the results are generated correctly using *FAS.vo*, you will get the following message in ModelSim simulation.

```
# -----------------------------------------------------
# FFT dataout on pattern        880 ~        895, PASS!!
# FFT dataout on pattern        896 ~        911, PASS!!
# FFT dataout on pattern        912 ~        927, PASS!!
# FFT dataout on pattern        928 ~        943, PASS!!
# FFT dataout on pattern        944 ~        959, PASS!!
# FFT dataout on pattern        960 ~        975, PASS!!
# FFT dataout on pattern        976 ~        991, PASS!!
# FFT dataout on pattern        992 ~       1007, PASS!!
# FFT dataout on pattern       1008 ~       1023, PASS!!
# -----------------------------------------------------
#
# Congratulations! All data have been generated successfully!
#
# -------------------------PASS------------------------
#
# ** Note: $finish    : C:/Users/user/Desktop/test_hw5/testfixture1.v(240)
#    Time: 53600 ns  Iteration: 0  Instance: /testfixture1
```

Device：**Cyclone II EP2C70F896C8**

3.3 Performance **[10%]**

The performance is scored by the total logic elements, total memory bit, and embedded multiplier 9-bit element your design used in gate-level simulation and the simulation time your design takes. The score will be decided by your ranking in all received homework. (The smaller, the better)

*Scoring = (Total logic elements + total memory bit + 9*embedded multiplier 9-bit element) × (longest gate-level simulation time in ns)*

4 Submission

4.1 Submitted files

You should classify your files into four directories and compress them to .zip format. The naming rule is HW5_studentID_name.zip. If your file is not named according to the naming rule, you will lose five points.

	RTL category
*.v	All of your Verilog RTL code
	Gate-Level category
*.vo	Gate-Level netlist generated by Quartus
*.sdo	SDF timing information generated by Quartus
	Documentary category
*.pdf	The report file of your design (in pdf).

4.2 Report file

Please follow the spec of report. If your report does not meet the spec, you may lose part of score. You are asked to describe how the circuit is designed as detailed as possible, and the flow summary result is necessary in the report. Please fill the fields of total logic elements, total memory bits, and embedded multiplier 9-bit elements according to the flow summary of your synthesized design. And fill the field of gate-level simulation time according to the gate-level simulation result that Modelsim shows.

Flow Summary	
Flow Status	Successful - Tue May 11 16:15:18 2021
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	MFE
Top-level Entity Name	MFE
Family	Cyclone II
Device	EP2C70F896C8
Timing Models	Final
Total logic elements	319 / 68,416 (< 1 %)
Total combinational functions	311 / 68,416 (< 1 %)
Dedicated logic registers	92 / 68,416 (< 1 %)
Total registers	92
Total pins	57 / 622 (9 %)
Total virtual pins	0
Total memory bits	0 / 1,152,000 (0 %)
Embedded Multiplier 9-bit elements	0 / 300 (0 %)
Total PLLs	0 / 4 (0 %)

4.3 Note

In this homework, you are allowed to modify the defined CYCLE in testbench file. End_CYCLE, which decides the maximum cycles your circuit took to complete simulation, can also be modified according to your design. Please do not modify any other content of the testbench.

Please submit your .zip file to folder HW5 in moodle.

Deadline: 2021/7/2 23:55

If you have any problem, please contact TA by email.

ht920055@gmail.com

Appendix

In the appendix, the processing results of the first sixteen data of testbench 1 using FFT is provided for debugging usage.

Stage1		Stage2		Stage3	
實數	虛數	實數	虛數	實數	虛數
32'hFFFFDA00	32'h00000000	32'hFFFFFE00	32'h00000000	32'h00002D00	32'h00000000
32'hFFFFEA00	32'h00000000	32'h00001900	32'h00000000	32'h00005800	32'h00000000
32'hFFFFFE00	32'h00000000	32'h00002F00	32'h00000000	32'hFFFFCF00	32'h00000000
32'h00001300	32'h00000000	32'h00003F00	32'h00000000	32'h00000000	32'h00002600
32'h00002400	32'h00000000	32'hFFFFB600	32'h00000000	32'hFFFFB600	32'h00003300
32'h00002F00	32'h00000000	32'hFFFFCF36	32'h000030CA	32'hFFFFE0E3	32'h00004277
32'h00003100	32'h00000000	32'h00000000	32'h00003300	32'hFFFFB600	32'hFFFFCD00
32'h00002C00	32'h00000000	32'h000011AD	32'h000011AD	32'h00001F1D	32'h00004277
32'h00014400	32'h00000000	32'h00014400	32'hFFFEA000	32'h0002BE4A	32'hFFFD518C
32'h0001A370	32'hFFFF5244	32'h00016079	32'hFFFEB097	32'h0002EA48	32'hFFFD5813
32'h0001645F	32'hFFFE9BA1	32'h00017A4A	32'hFFFEB18C	32'hFFFFC9B6	32'hFFFFEE74
32'h0000B379	32'hFFFE4EB5	32'h000189CF	32'hFFFEA77C	32'h0000091B	32'h00002956
32'h00000000	32'hFFFEA000	32'h00014400	32'h00016000	32'hFFFFC9B6	32'h0000118C
32'hFFFFBD09	32'hFFFF5E53	32'h00014F67	32'hFFFE9F8B	32'hFFFFF6E7	32'h00002958
32'h000015EB	32'h000015EB	32'hFFFE85B6	32'hFFFEB18C	32'h0002BE4A	32'h0002AE74
32'h0000D656	32'h000058C7	32'hFFFEA780	32'h000189CD	32'hFFFD15BE	32'hFFFD5819

Stage4	
實數	虛數
32'h00008500	32'h00000000
32'hFFFFD500	32'h00000000
32'hFFFFCF00	32'h00002600
32'hFFFFCF00	32'hFFFFDA00
32'hFFFF96E3	32'h00007577
32'hFFFFD51D	32'hFFFFF089
32'hFFFFD51D	32'h00000F77
32'hFFFF96E3	32'hFFFF8A89
32'h0005A892	32'hFFFAA99F
32'hFFFFD402	32'hFFFFF979
32'hFFFFD2D1	32'h000017CA
32'hFFFFC09B	32'hFFFFC51E
32'hFFFFC09D	32'h00003AE4
32'hFFFFD2CF	32'hFFFFE834
32'hFFFFD408	32'h0000068D