Digital Communication Circuit Lab.

Lab. 6

Fast Fourier Transform (FFT)

(First Part: 12/13, Second Part: 12/23)

I. Purpose

In this lab., we will learn the principle of FFT operation and to realize how to use hardware to implement discrete Fourier transform.

II. Principle

Discrete Fourier transform is widely used in signal processing to transform periodic discrete signal between time domain and frequency domain. Given time-domain signal x_n and frequency-domain signal X_k, its equations are provided as follows.

$$X_k = \sum_{n=0}^{N-1} x_n e^{-j2\pi nk/N},\ k = 0,1,\dots,N-1 \quad (1)$$

$$x_n = \frac{1}{N}\sum_{k=0}^{N-1} X_k e^{j2\pi nk/N},\ n = 0,1,\dots,N-1 \quad (2)$$

where x_n and X_k are time-domain and frequency-domain signals, N is the period. The computation complexity of discrete Fourier transform is quite large. Its multiplication complexity is $O(N^2)$, which is proportional to the signal period. When N is power of 2, we have a fast algorithm to implement the discrete Fourier transform so that the multiplication complexity can be reduced to $O(Nlog_2N)$. In this lab, we will implement 32-point fast Fourier transform.

A. Fast Fourier transform (FFT) algorithm

In this lab., we will use the radix-2 FFT algorithm. The discrete Fourier transform can be written as $X_k = \sum_{n=0}^{N-1} x_n W_N^{nk}$, where $W_N = e^{-j2\pi/N}$. Separating the frequency-domain signal into the even-indexed part and the odd-indexed part, we can obtain $X_{2r} = \sum_{n=0}^{N-1} x_n W_N^{2rn}$ and $X_{2r+1} = \sum_{n=0}^{N-1} x_n W_N^{(2r+1)n}$. The time-domain signal is then further classified into the first half and the second half. Thus,

$$\begin{aligned} X_{2r} &= \sum_{n=0}^{\frac{N}{2}-1} x_n W_N^{2rn} + \sum_{n=\frac{N}{2}}^{N-1} x_n W_N^{2rn} \\ &= \sum_{n=0}^{N/2-1} x_n (W_N^2)^{rn} + \sum_{n=0}^{N/2-1} x_{(n+\frac{N}{2})} (W_N^2)^{r(n+\frac{N}{2})}. \end{aligned} \quad (3)$$

Because $(W_N^2)^{r(n+\frac{N}{2})} = W_N^{2rn} W_N^{rN} = W_N^{2rn}$, Eq. (3) can be further simplified as

$$X_{2r} = \sum_{n=0}^{N/2-1} x_n (W_N^2)^{rn} + \sum_{n=0}^{N/2-1} x_{(n+\frac{N}{2})} (W_N^2)^{rn}$$

$$= \sum_{n=0}^{N/2-1} \left(x_n + x_{(n+\frac{N}{2})} \right) W_{N/2}^{rn} \quad (4)$$

It can be regarded as $N/2$-point discrete Fourier transform of the new samples that are generated by adding the first half to the second half. Similarly, the frequency-domain odd-indexed part can be written as

$$X_{2r+1} = \sum_{n=0}^{N/2-1} x_n W_N^{(2r+1)n} + \sum_{n=N/2}^{N-1} x_n W_N^{(2r+1)n}$$

$$= \sum_{n=0}^{N/2-1} x_n W_N^{(2r+1)n} + \sum_{n=0}^{N/2-1} x_{(n+\frac{N}{2})} W_N^{(2r+1)(n+\frac{N}{2})}. \quad (5)$$

Because $W_N^{n+N/2} = -W_N^n$, Eq. (5) becomes

$$X_{2r+1} = \sum_{n=0}^{N/2-1} x_n W_N^{(2r+1)n} - \sum_{n=0}^{N/2-1} x_{(n+\frac{N}{2})} W_N^{(2r+1)n}$$

$$= \sum_{n=0}^{\frac{N}{2}-1} \left(x_n - x_{\left(n+\frac{N}{2}\right)} \right) W_N^n W_{N/2}^{rn}. \quad (6)$$

It can be interpreted as the $N/2$-point discrete Fourier transform of the new samples that are generated by multiplying W_N^n to the addition output of the first half and the second half. W_N^n is called twiddle factor. Fig. 1 shows the data flow of decomposition of one 8-point discrete Fourier transform into two 4-point discrete Fourier transforms.

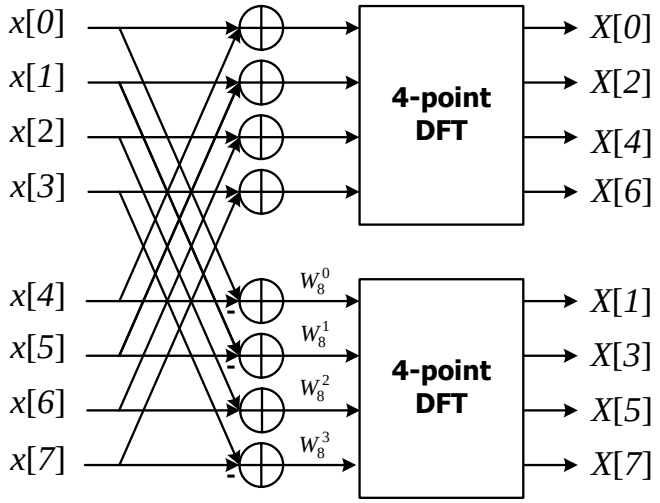

Fig. 1 Decomposition of 8-point discrete Fourier transform

If we proceed the decomposition to compute every $N/2$-point discrete Fourier transform by two $N/4$-point discrete Fourier transform, then we can obtain the fast algorithm to compute N-point discrete Fourier transform.

$$X_{4m} = G_{2m} = \sum_{n=0}^{\frac{N}{2}-1} g_n W_{N/2}^{2mn} = \sum_{n=0}^{\frac{N}{2}-1} (g_n + g_{n+N/4}) W_{N/4}^{mn} \quad (7)$$

$$X_{4m+2} = G_{2m+1} = \sum_{n=0}^{\frac{N}{2}-1} g_n W_{N/2}^{2mn+n} = \sum_{n=0}^{\frac{N}{2}-1} (g_n - g_{n+N/4}) W_{N/2}^{n} W_{N/4}^{mn}. \quad (8)$$

Fig. 2 shows the signal flow graph (SFG) of an 8-point fast Fourier transform. It is called decimation-in-frequency FFT because the frequency domain samples do not appear in the normal order.

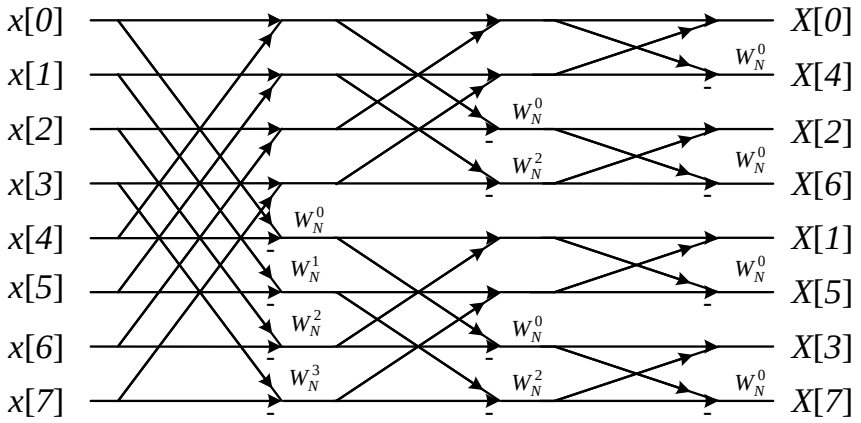

Fig. 2 Signal flow graph (SFG) of an 8-point fast Fourier transform.

To make the frequency-domain samples output in order, we need to re-arrange the decimated frequency-domain samples. Observing the sequence indexes, we can realize that the outputs are in an order that uses the bit-reversed binary representation, as shown in Fig. 3. Consequently, if we want to get output in order, we only need to reverse the binary representation of the index. This operation is called bit-reversal or bit-reverse re-ordering.

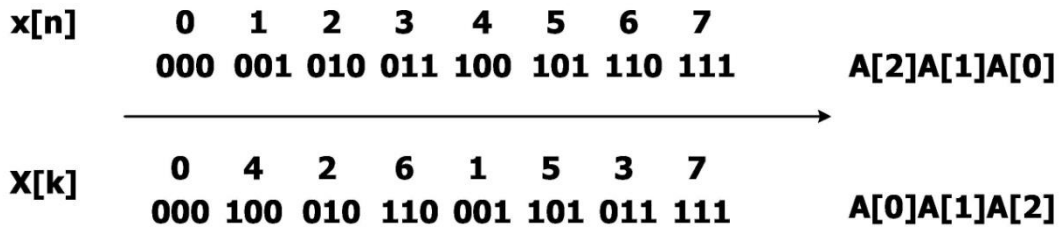

Fig. 3 Frequency-domain samples in bit-reversed order.

B. Pipelined architecture for FFT

In this lab., we will implement the fast Fourier transform by pipelined architecture, named single-path delay feedback (SDF) architecture, which processes inputs in serial and generates serial outputs. The major advantage of the SDF architecture is that it has the least delay elements [1]. Fig. 4 is the block diagram of an 8-point SDF FFT by cascading three processing-element stages. Each processing elements contains a delay buffer and a butterfly unit. The butterfly unit computes addition and subtraction of two

input signals for 50% of the execution time. For the remaining time, it simply bypasses the input signal to the delay buffer and sends the content of the delay buffer to the output for twiddle-factor multiplication. We also need to control the multiplicand to the multipliers to arrange the correct twiddle factor for multiplication. For radix-2 algorithm, the multiplier has 50% utilization and is idle for the remaining 50% execution time.

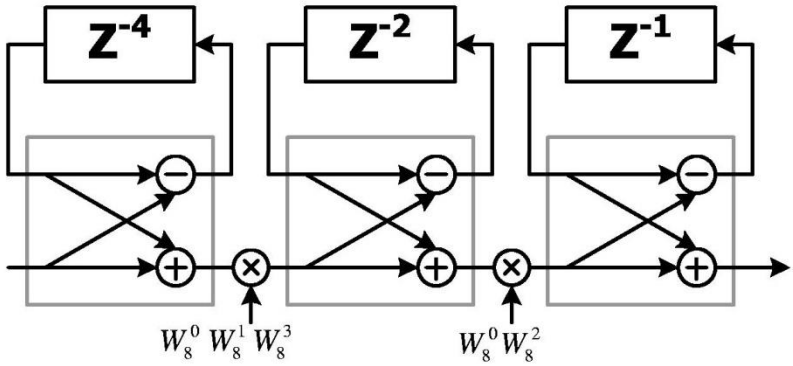

Fig. 4 pipelined architecture for 8-point FFT.

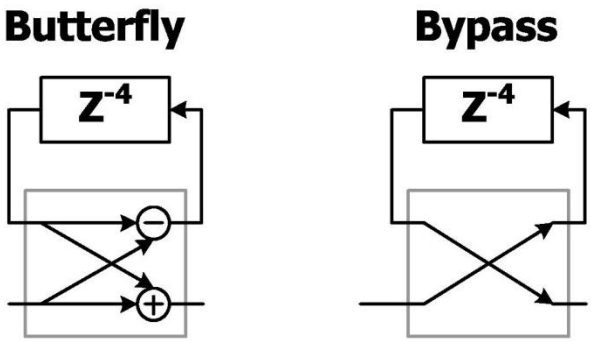

Fig. 5 Two operation modes for the butterfly unit.

Using 8-point FFT as an example, we can realize the operation of the radix-2 single-path delay feedback architecture with the signal-flow graph. As shown in Table I, two inputs of the butterfly unit are labeled as the upper input and the lower input. Similarly, it has the upper output and the lower output. The upper output is sent to the delay unit while the lower output is sent to the multiplier. Assume that input signals are denoted as $x_0,\ x_1, \ldots,\ x_7$. The sum of x_k and $x_{k+N/2}$ is denoted as g_k. The difference of x_k and $x_{k+N/2}$ is denoted as h_k. p_k is the sum of g_k and $g_{k+N/4}$. q_k is the difference of g_k and $g_{k+N/4}$. m_k is the sum of h'_k and $h'_{k+N/4}$. n_k is the difference of h'_k and $h'_{k+N/4}$. h'_k is the output of h_k multiplied by the twiddle factor.

C. Lookup table for twiddle factor

Twiddle factor can be saved in a table with the possible sin and cosine values. Use the counter to control the table output as the multiplicand to the multiplier.

D. Inverse Fourier transform

From the duality of DFT and IDFT in Eqs. (1) and (2), we can see that the inverse FFT can be interpreted as the following operation.

$$x_n = \frac{1}{N}\sum_{k=0}^{N-1} X_k e^{j2\pi nk/N} = \frac{1}{N}\left(\sum_{k=0}^{N-1} (X_k)^* e^{-j2\pi nk/N}\right)^*$$

It can be interpreted as using the complex conjugate frequency domain signal $(X_k)^*$ as the input and then sending it to the FFT module. The time-domain signal x_n can be obtained by the complex conjugate of the FFT output divided by N. However, since N is power of 2, it is not necessary to implement a divider. We only need to change the scaling when we translate the fixed-point integer to the true values. Hence, the IFFT can be implemented by the FFT module with change of the sign values of the imaginary part of the FFT inputs and outputs.

Table I Time scheduling of an 8-point FFT operation.

	Time	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Stage 1	Upper Input	0	0	0	0	x_0	x_1	x_2	x_3	h_0	h_1	h_2	h_3			
	Lower Input	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7							
	Upper Output					h_0	h_1	h_2	h_3							
	Lower Output					g_0	g_1	g_2	g_3							
	Multiplier	W_8^0	W_8^1	W_8^0	W_8^3	1	1	1	1	W_8^0	W_8^1	W_8^0	W_8^3	1	1	1
Stage 2	Upper Input							g_0	g_1	q_0	q_1	h'_0	h'_1			
	Lower Input					g_0	g_1	g_2	g_3	h'_0	h'_1	h'_2	h'_3			
	Upper Output							q_0	q_1			n_0	n_1			
	Lower Output							p_0	p_1			m_0	m_1			

	Multiplier	W_8^0	W_8^2	1	1	W_8^0	W_8^2	1	1	W_8^0	W_8^2	1	1	W_8^0	W_8^2	1
Stage 3	Upper Input								p_0	A_4	q'_0	A_6	m_0	A_5	n'_0	A_7
	Lower Input							p_0	p_1	q'_0	q'_1	m_0	m_1	n'_0	n'_1	
	Upper Output								A_4		A_6		A_5		A_7	
	Lower Output								X_0	X_4	X_2	X_6	X_1	X_5	X_3	X_7
	Multiplier	W_8^0	1	W_8^0	1	W_8^0	1	W_8^0	1	W_8^0	1	W_8^0	1	W_8^0	1	W_8^0

We can extend the concept of the 8-point FFT architecture to the 32-point FFT operation. The hardware architecture of 32-point FFT is shown in Fig. 6.

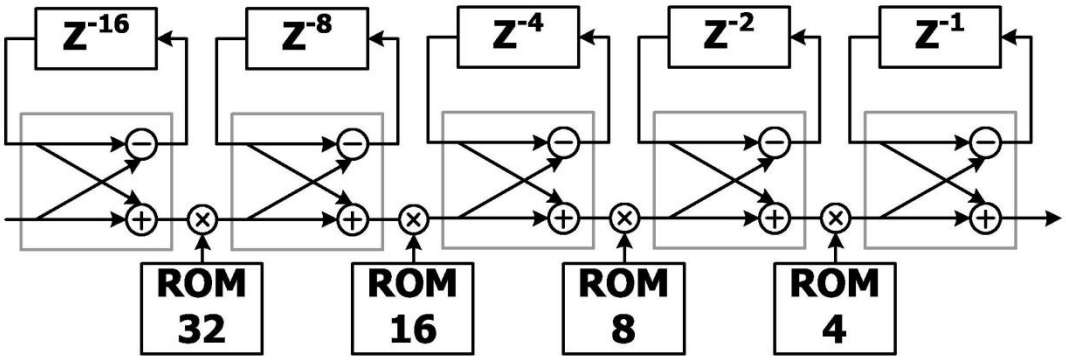

Fig. 6 Architecture of the 32-point FFT operation.

III. Procedures

1. Given a set S=$\frac{1}{\sqrt{2}}${1+j, 1-j, -1+j, -1-j}, please randomly generate 8 samples from the set S, denoted as $Y_0 \sim Y_7$. Use Matlab command "ifft" to transform $Y_0 \sim Y_7$ into the time-domain signal $x_0 \sim x_7$.
2. From Table I, given that there is a counter counting from 0 to $N-1$ when the first valid input enters into the N-point FFT module. Show your control signal that sends to the butterfly module at each stage to control two operation modes. (1: Butterfly operation, 0: bypass operation)
3. Derive your control signal to the complex multiplier block to control the multiplication and bypass operation. (1: Multiplication, 0: bypass operation)

4. Write Matlab program to implement SDF FFT. Use $x_0 \sim x_7$ as the SDF FFT input. Denote the SDF FFT output as $X_0 \sim X_7$.
5. Insert the bit-reverse reordering buffer to transform the frequency-domain outputs into normal order $X_0 \sim X_7$. Compare $X_0 \sim X_7$ with $Y_0 \sim Y_7$.
6. Implement the 8-point SDF FFT by Verilog. Use $x_0 \sim x_7$ as the FFT input. Denote the FFT output as $X_0 \sim X_7$. Check its difference with $Y_0 \sim Y_7$. Assume that the input signal is represented by 13 bits including the sign bit. Increase the word-length of each stage by one bit to avoid overflow after the butterfly operation, namely the output word-lengths of 3 stages are 14, 15, and 16.

==========

7. Cascade 5 stages of your butterfly module and multiplication module to implement 32-point radix-2 FFT operation. Write Matlab program to implement 32-point SDF FFT. Use $x_0 \sim x_{31}$ as the SDF FFT input. Insert the bit-reverse reordering buffer to transform the frequency-domain outputs into normal order $X_0 \sim X_{31}$. Compare $X_0 \sim X_{31}$ with $Y_0 \sim Y_{31}$.
8. Write the Verilog codes for 32-point FFT with 5-stage PEs. Assume that the input signal is represented by 12 bits including the sign bit (S0.11). Increase the word-length of each stage by one bit to avoid overflow after the butterfly operation, namely the word-lengths of 5 stages are 12, 13, 14, 15, and 16 bits, respectively. Use 13 bits to represent twiddle factors (S1.11) at all stages. Do the truncation for the multiplier output so that the wordlength of multiplier output is the same as the multiplier input along the datapath. Verify the hardware simulation results and the Matlab simulation results.
9. Design the bit-reversal module so that the decimated frequency-domain samples can be sent out in order. You can allocate two memory arrays, each having 32 elements. The serial FFT output is first saved in one memory array and the normal-order output is fetched from the other memory array. This is called ping-pong access.
10. Synthesis and program your codes into FPGA and show your measurement results.

IV. Results

1. Draw the real-part and imaginary-part of $Y_0 \sim Y_7$ and $x_0 \sim x_7$ (5%).
2. Draw the timing diagram of the control signals to the butterfly module at all the stages. (10%)
3. Draw the timing diagram of the control signals to the complex multiplier blocks at all the stages. (10%)
4. Use Matlab program to implement 8-point SDF FFT architecture. Draw the real-part and imaginary-part of $X_0 \sim X_7$, which is in bit-reversed order. (15%)
5. Use Matlab program to implement the bit-reversal module. Compare them with the

real-part and imaginary-part of $Y_0 \sim Y_7$. Draw the difference between $X_0 \sim X_7$ and $Y_0 \sim Y_7$.(10%)

6. Show the timing diagram of your Verilog behavior and post-route simulation results of 8-point MDC FFT. Compare with the Matlab results to check your implementation error. Depict the error of the real part and imaginary part of each point using figures. (30%)

==========

7. Use Matlab program to implement 32-point SDF IFFT architecture and the bit-reversal module. Draw the real-part and imaginary-part of $y_0 \sim y_{31}$ and $X_0 \sim X_{31}$. Compare them with the real-part and imaginary-part of $Y_0 \sim Y_{31}$. Depict the error. (20%)
8. Show the timing diagram of your Verilog behavior and post-route simulation results of outputs after 32-point SDF FFT and after the bit-reverse reordering (40%). Compare the outputs after bit-reverse reordering with the Matlab results to check your implementation error. Depict the real-part and imaginary-part error. (20%)
9. Show your synthesis timing report (10%) and print out the measurement results after bit-reverse reordering (timing diagram of measurement). (30%) Demo the measurement results to TAs before 12/27.