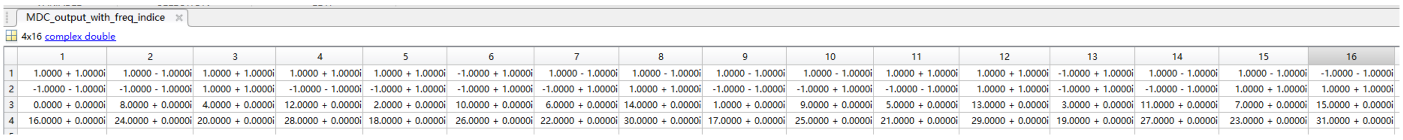
DSP in VLSI

HW5

電子所 ICS 組, R13943015, 張根齊

1. (Step 1) Please use FFTInput32.mat as the inputs. Use your MDC FFT Matlab/Python program to generate the 32 FFT outputs. The program outputs should be in a 2 × 16 array. Because they are in bit-reversed order, write a program to generate their associated frequency domain indices. Now, you have an 4 × 16 array containing the frequency domain indices and the MDC outputs. List the array. (Maybe you can copy the results in Work Space and paste onto your report.) (10%)

row 1 and row 2 are 32 FFT outputs (complexed number). And row3, row4 are corresponding frequency domain indices. (integer number)



MDC_output_with_freq_indice

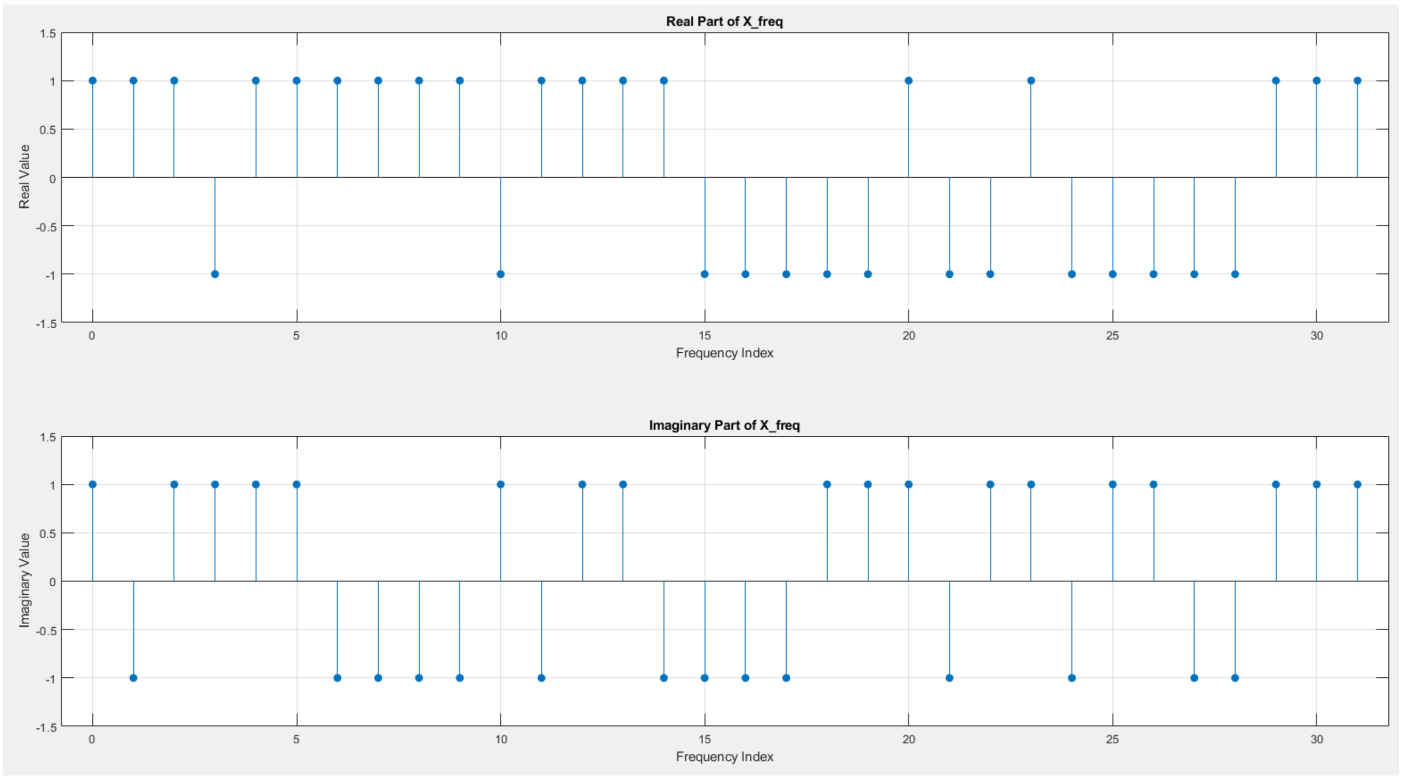
4x16 complex double

	1	2	3	4	5	6	7	8
1	1.0000 + 1.0000i	1.0000 - 1.0000i	1.0000 + 1.0000i	1.0000 + 1.0000i	1.0000 + 1.0000i	-1.0000 + 1.0000i	1.0000 - 1.0000i	1.0000 - 1.0000i
2	-1.0000 - 1.0000i	-1.0000 - 1.0000i	1.0000 + 1.0000i	-1.0000 - 1.0000i	-1.0000 + 1.0000i	-1.0000 + 1.0000i	-1.0000 + 1.0000i	1.0000 + 1.0000i
3	0.0000 + 0.0000i	8.0000 + 0.0000i	4.0000 + 0.0000i	12.0000 + 0.0000i	2.0000 + 0.0000i	10.0000 + 0.0000i	6.0000 + 0.0000i	14.0000 + 0.0000i
4	16.0000 + 0.0000i	24.0000 + 0.0000i	20.0000 + 0.0000i	28.0000 + 0.0000i	18.0000 + 0.0000i	26.0000 + 0.0000i	22.0000 + 0.0000i	30.0000 + 0.0000i

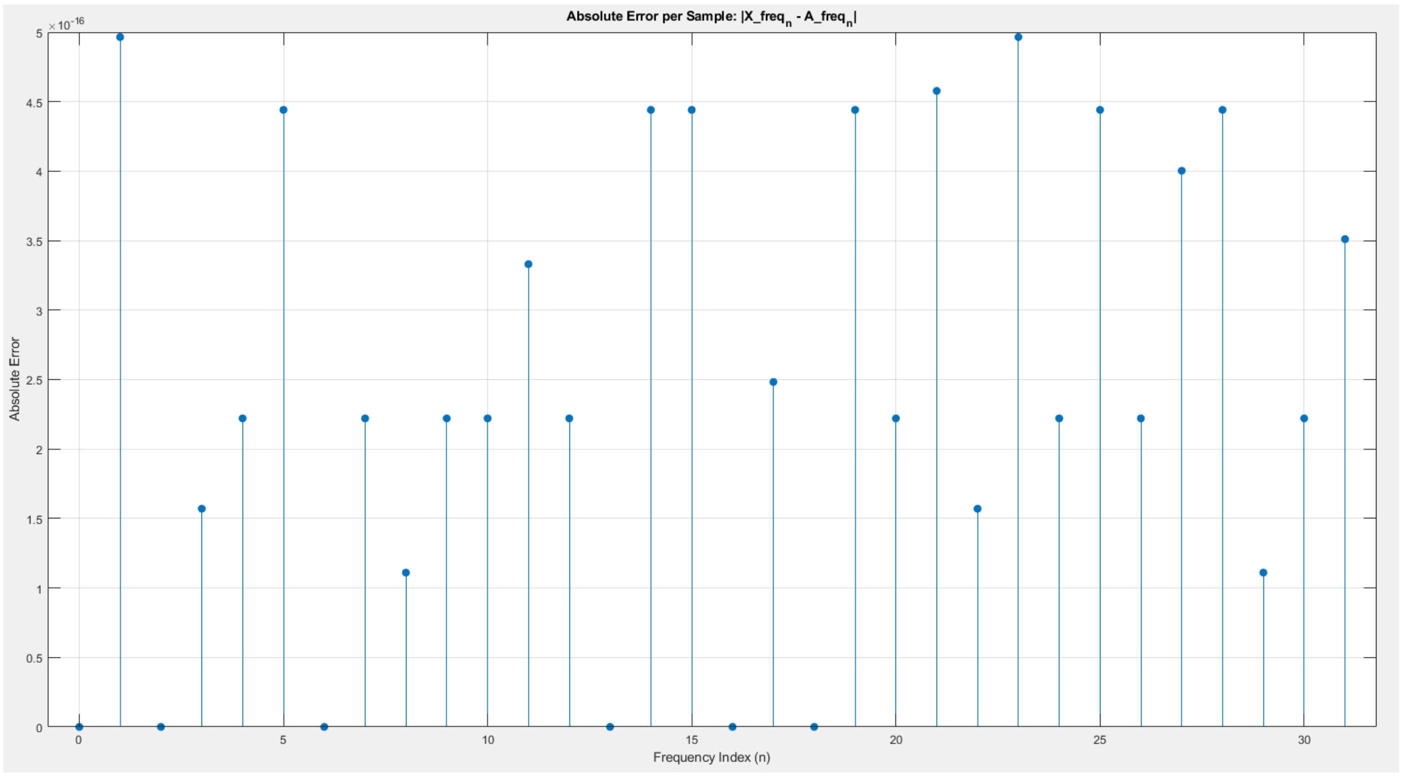
	9	10	11	12	13	14	15	16
1	1.0000 - 1.0000i	1.0000 - 1.0000i	1.0000 + 1.0000i	1.0000 + 1.0000i	-1.0000 + 1.0000i	1.0000 - 1.0000i	1.0000 - 1.0000i	-1.0000 - 1.0000i
2	-1.0000 - 1.0000i	-1.0000 + 1.0000i	-1.0000 - 1.0000i	1.0000 + 1.0000i	-1.0000 + 1.0000i	-1.0000 - 1.0000i	1.0000 + 1.0000i	1.0000 + 1.0000i
3	1.0000 + 0.0000i	9.0000 + 0.0000i	5.0000 + 0.0000i	13.0000 + 0.0000i	3.0000 + 0.0000i	11.0000 + 0.0000i	7.0000 + 0.0000i	15.0000 + 0.0000i
4	17.0000 + 0.0000i	25.0000 + 0.0000i	21.0000 + 0.0000i	29.0000 + 0.0000i	19.0000 + 0.0000i	27.0000 + 0.0000i	23.0000 + 0.0000i	31.0000 + 0.0000i

2. (Step 2) Please use FFTInput32.mat as the inputs. Show that your program for output re-ordering is correct. Draw the real part and imaginary part of $X0$~$X31$. (10%) Compare the results of $X0$~$X31$ to A0~A31 . Draw the absolute error of each sample. The error should be small than 10^-10 because of double-precision floating-point operations. If the error is not acceptable, the first 20 points will be deducted, too. (10%)

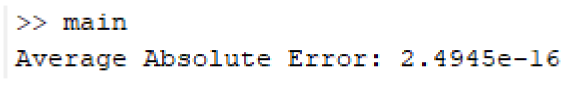
2.1 Draw the real part and imaginary part of $X0$~$X31$



2.2 Draw the absolute error of each sample



Average error (< 10^-10):

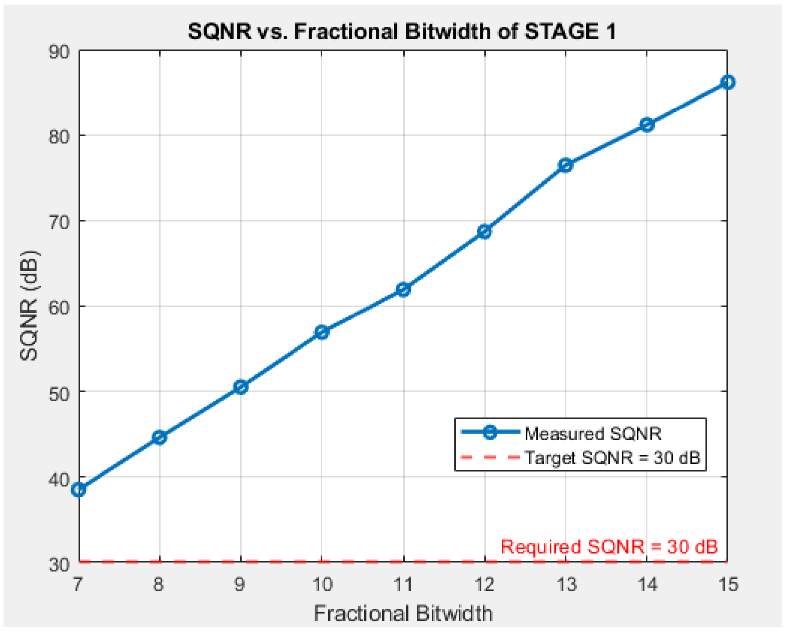


```
>> main
Average Absolute Error: 2.4945e-16
```

3. (Step 3) Generate your own inputs of 96 samples. Draw the SQNR versus fractional part word-length N stage by stage. (30%)

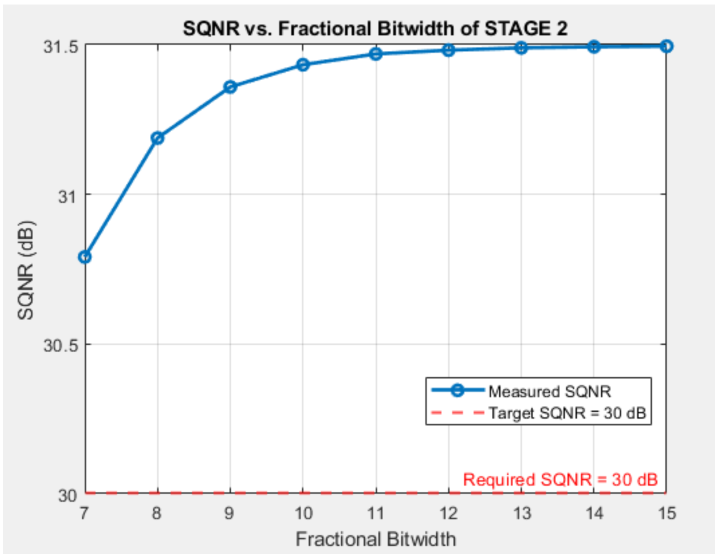
(a) Quantize the first stage. Evaluate the FFT output SQNR versus fractional part word-length N for N=7, 8, 9, 10, …, 13, 14,15 for stage 1. Draw the figure.

Choose stage 1 fractional bit-width = 10 (reserved margin).



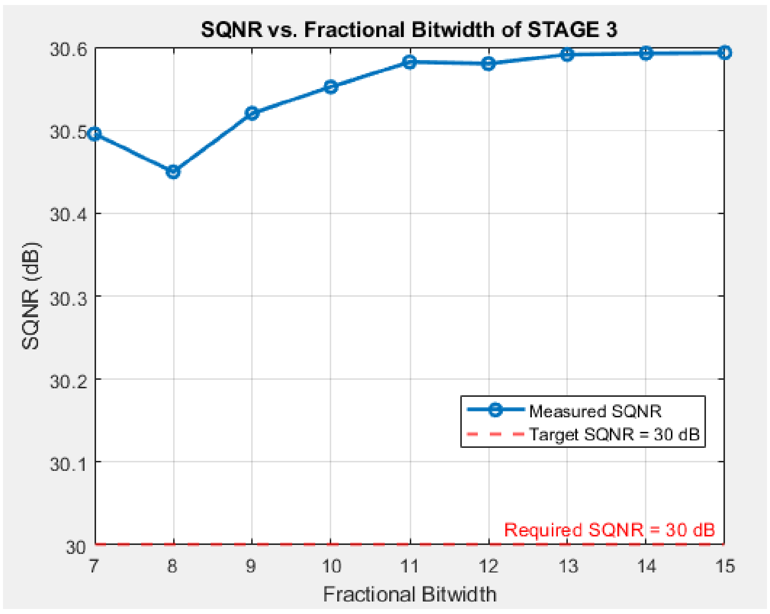
(b) Fix the setting for stage 1. Quantize the second stage. Evaluate the FFT output SQNR versus fractional part word-length N for N=7, 8, 9, 10,…, 13, 14,15 for stage 2. Draw the figure.

Choose stage 2 fractional bit-width = 10 (reserved margin):



(c) Fix the setting for stage 1 and 2. Quantize the third stage. Evaluate the FFT output SQNR versus fractional part word-length N for N=7, 8, 9, 10,…, 13,14,15 for stage 3 and so on. Draw the figure

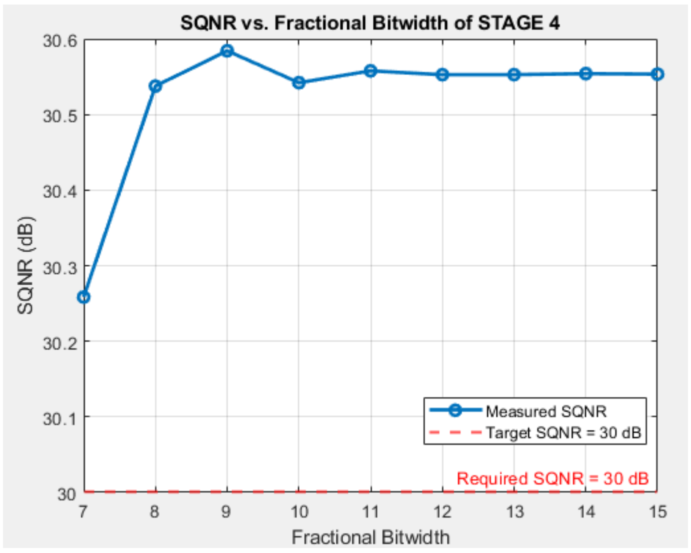
Choose stage 3 fractional bit-width = 10 (reserved margin):



(d) Repeat until stage 5. So, you have five figures.

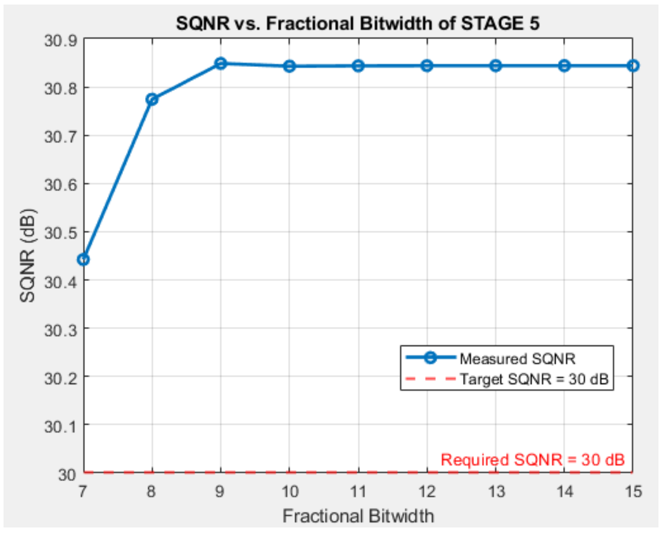
d-1: stage 4:

Choose stage 4 fractional bit-width = 10 (reserved margin):



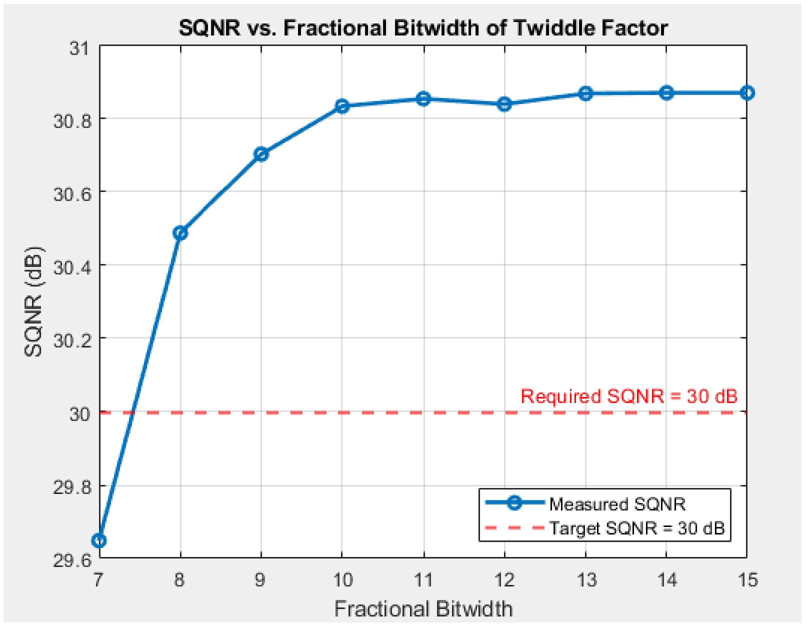
d-2: stage 5:

Choose stage 5 fractional bit-width = 10 (reserved margin):



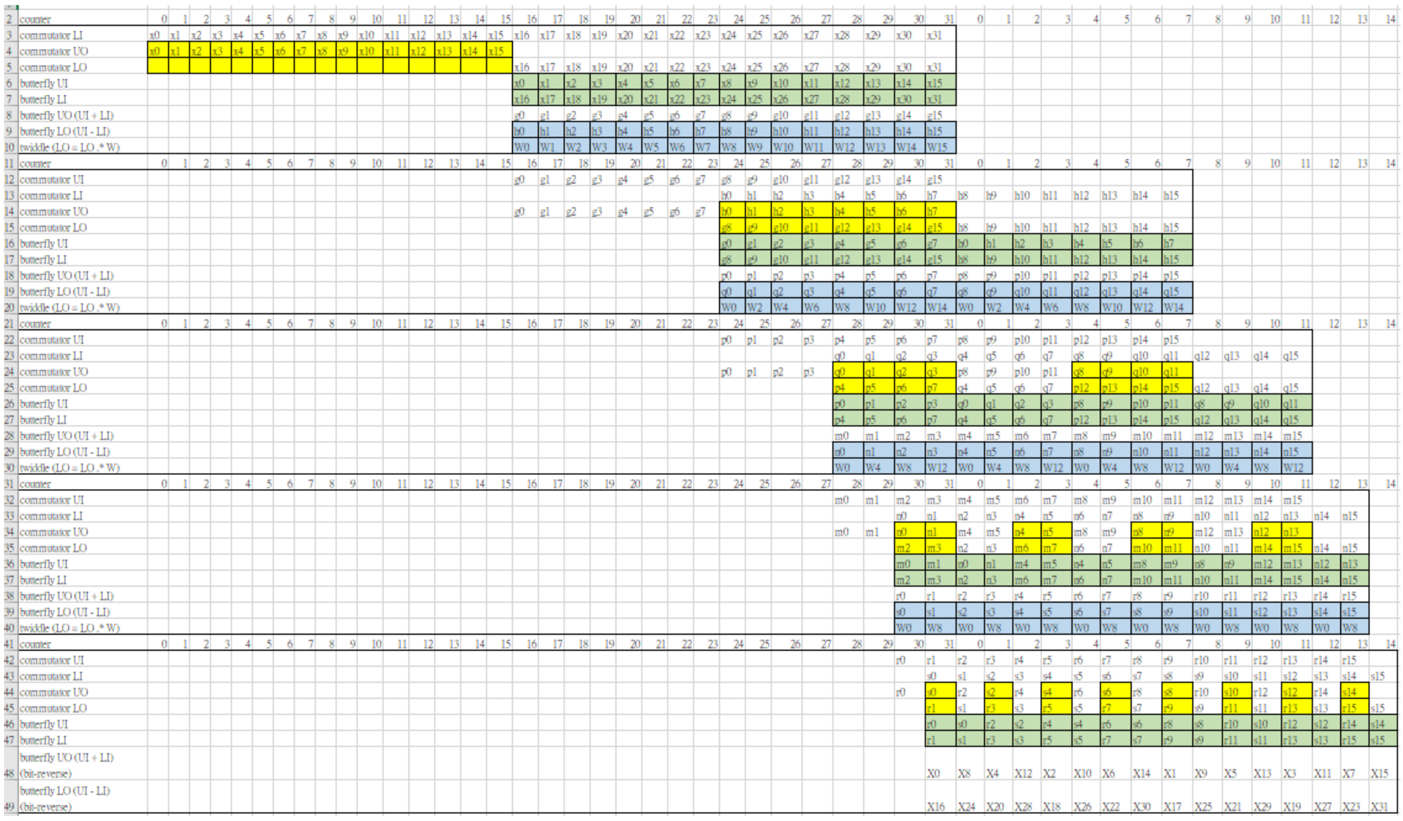
(e) Then, decide the fractional part word-length for twiddle factors of all the stages. Draw the FFT output SQNR versus fractional part word-length of twiddle factors.

Choose twiddle factors fractional bit-width = 10:



4. (Step 4) Now, use a 5-bit counter counting from 0 to 31, which is synchronized to the input index. Generate the control signals of commutator and butterfly units for each stage. Show how you generate them and why? (20%)

The overall schedule is shown below (The file is included in the submitted assignment package). The yellow sections indicate that the control signal for the commutator should be set to switch mode; otherwise, it should be in bypass mode. The green sections indicate that the control signal for the butterfly should be set to computation mode; otherwise, it should be in bypass mode.



2	counter	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
3	commutator LI	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15	x16	x17	x18	x19	x20	x21	x22	x23	x24	x25	x26	x27	x28	x29	x30	x31															
4	commutator UO	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15																															
5	commutator LO																	x16	x17	x18	x19	x20	x21	x22	x23	x24	x25	x26	x27	x28	x29	x30	x31															
6	butterfly UI																	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15															
7	butterfly LI																	x16	x17	x18	x19	x20	x21	x22	x23	x24	x25	x26	x27	x28	x29	x30	x31															
8	butterfly UO (UI + LI)																	g0	g1	g2	g3	g4	g5	g6	g7	g8	g9	g10	g11	g12	g13	g14	g15															
9	butterfly LO (UI - LI)																	h0	h1	h2	h3	h4	h5	h6	h7	h8	h9	h10	h11	h12	h13	h14	h15															
10	twiddle (LO = LO .* W)																	W0	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15															
11	counter	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
12	commutator UI																	g0	g1	g2	g3	g4	g5	g6	g7	g8	g9	g10	g11	g12	g13	g14	g15															
13	commutator LI																									h0	h1	h2	h3	h4	h5	h6	h7	h8	h9	h10	h11	h12	h13	h14	h15							
14	commutator UO																	g0	g1	g2	g3	g4	g5	g6	g7	h0	h1	h2	h3	h4	h5	h6	h7															
15	commutator LO																									g8	g9	g10	g11	g12	g13	g14	g15	h8	h9	h10	h11	h12	h13	h14	h15							
16	butterfly UI																									g0	g1	g2	g3	g4	g5	g6	g7	h0	h1	h2	h3	h4	h5	h6	h7							
17	butterfly LI																									g8	g9	g10	g11	g12	g13	g14	g15	h8	h9	h10	h11	h12	h13	h14	h15							
18	butterfly UO (UI + LI)																									p0	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13	p14	p15							
19	butterfly LO (UI - LI)																									q0	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	q11	q12	q13	q14	q15							
20	twiddle (LO = LO .* W)																									W0	W2	W4	W6	W8	W10	W12	W14	W0	W2	W4	W6	W8	W10	W12	W14							
21	counter	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
22	commutator UI																									p0	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13	p14	p15							
23	commutator LI																													q0	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	q11	q12	q13	q14	q15			
24	commutator UO																									p0	p1	p2	p3	q0	q1	q2	q3	p8	p9	p10	p11	q8	q9	q10	q11							
25	commutator LO																													p4	p5	p6	p7	q4	q5	q6	q7	p12	p13	p14	p15	q12	q13	q14	q15			
26	butterfly UI																													p0	p1	p2	p3	q0	q1	q2	q3	p8	p9	p10	p11	q8	q9	q10	q11			
27	butterfly LI																													p4	p5	p6	p7	q4	q5	q6	q7	p12	p13	p14	p15	q12	q13	q14	q15			
28	butterfly UO (UI + LI)																													m0	m1	m2	m3	m4	m5	m6	m7	m8	m9	m10	m11	m12	m13	m14	m15			
29	butterfly LO (UI - LI)																													n0	n1	n2	n3	n4	n5	n6	n7	n8	n9	n10	n11	n12	n13	n14	n15			
30	twiddle (LO = LO .* W)																													W0	W4	W8	W12	W0	W4	W8	W12	W0	W4	W8	W12	W0	W4	W8	W12			
31	counter	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
32	commutator UI																													m0	m1	m2	m3	m4	m5	m6	m7	m8	m9	m10	m11	m12	m13	m14	m15			
33	commutator LI																															n0	n1	n2	n3	n4	n5	n6	n7	n8	n9	n10	n11	n12	n13	n14	n15	
34	commutator UO																													m0	m1	n0	n1	m4	m5	n4	n5	m8	m9	n8	n9	m12	m13	n12	n13			
35	commutator LO																															m2	m3	n2	n3	m6	m7	n6	n7	m10	m11	n10	n11	m14	m15	n14	n15	
36	butterfly UI																															m0	m1	n0	n1	m4	m5	n4	n5	m8	m9	n8	n9	m12	m13	n12	n13	
37	butterfly LI																															m2	m3	n2	n3	m6	m7	n6	n7	m10	m11	n10	n11	m14	m15	n14	n15	
38	butterfly UO (UI + LI)																															r0	r1	r2	r3	r4	r5	r6	r7	r8	r9	r10	r11	r12	r13	r14	r15	
39	butterfly LO (UI - LI)																															s0	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15	
40	twiddle (LO = LO .* W)																															W0	W8	W0	W8	W0	W8	W0	W8	W0	W8	W0	W8	W0	W8	W0	W8	
41	counter	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
42	commutator UI																															r0	r1	r2	r3	r4	r5	r6	r7	r8	r9	r10	r11	r12	r13	r14	r15	
43	commutator LI																																s0	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
44	commutator UO																															r0	s0	r2	s2	r4	s4	r6	s6	r8	s8	r10	s10	r12	s12	r14	s14	
45	commutator LO																																r1	s1	r3	s3	r5	s5	r7	s7	r9	s9	r11	s11	r13	s13	r15	s15
46	butterfly UI																																r0	s0	r2	s2	r4	s4	r6	s6	r8	s8	r10	s10	r12	s12	r14	s14
47	butterfly LI																																r1	s1	r3	s3	r5	s5	r7	s7	r9	s9	r11	s11	r13	s13	r15	s15
48	butterfly UO (UI + LI) (bit-reverse)																																X0	X8	X4	X12	X2	X10	X6	X14	X1	X9	X5	X13	X3	X11	X7	X15
49	butterfly LO (UI - LI) (bit-reverse)																																X16	X24	X20	X28	X18	X26	X22	X30	X17	X25	X21	X29	X19	X27	X23	X31

4.1. Stage 1

4.1.1 Commutator

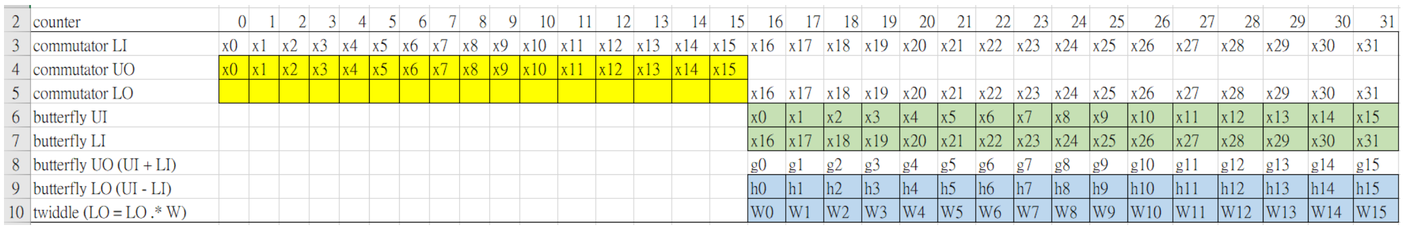
Counter = 0 ~ 15: switch mode

else: bypass mode

4.1.2 Butterfly

Counter = 16 ~ 31: computation mode

else: bypass mode



2	counter	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
3	commutator LI	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15	x16	x17	x18	x19	x20	x21	x22	x23	x24	x25	x26	x27	x28	x29	x30	x31
4	commutator UO	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15																
5	commutator LO																	x16	x17	x18	x19	x20	x21	x22	x23	x24	x25	x26	x27	x28	x29	x30	x31
6	butterfly UI																	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15
7	butterfly LI																	x16	x17	x18	x19	x20	x21	x22	x23	x24	x25	x26	x27	x28	x29	x30	x31
8	butterfly UO (UI + LI)																	g0	g1	g2	g3	g4	g5	g6	g7	g8	g9	g10	g11	g12	g13	g14	g15
9	butterfly LO (UI - LI)																	h0	h1	h2	h3	h4	h5	h6	h7	h8	h9	h10	h11	h12	h13	h14	h15
10	twiddle (LO = LO .* W)																	W0	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15

4.2. Stage 2

4.2.1 Commutator

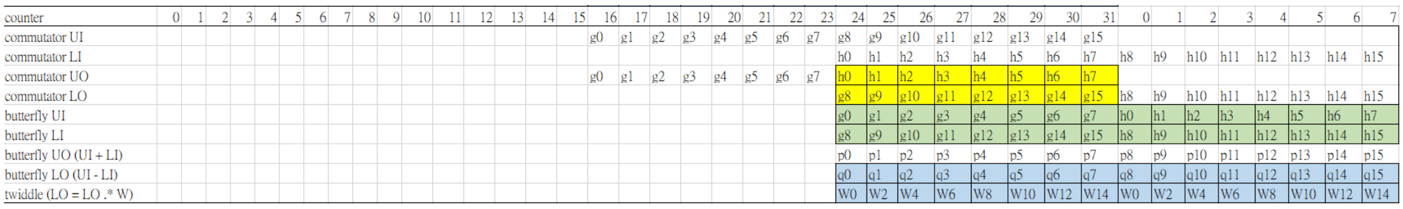
Counter = 24 ~ 31: switch mode

else: bypass mode

4.2.2 Butterfly

Counter = 24 ~ 31, 0 ~ 7: computation mode

else: bypass mode



counter	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	0	1	2	3	4	5	6	7
commutator UI																	g0	g1	g2	g3	g4	g5	g6	g7	g8	g9	g10	g11	g12	g13	g14	g15								
commutator LI																									h0	h1	h2	h3	h4	h5	h6	h7	h8	h9	h10	h11	h12	h13	h14	h15
commutator UO																	g0	g1	g2	g3	g4	g5	g6	g7	h0	h1	h2	h3	h4	h5	h6	h7								
commutator LO																									g8	g9	g10	g11	g12	g13	g14	g15	h8	h9	h10	h11	h12	h13	h14	h15
butterfly UI																									g0	g1	g2	g3	g4	g5	g6	g7	h0	h1	h2	h3	h4	h5	h6	h7
butterfly LI																									g8	g9	g10	g11	g12	g13	g14	g15	h8	h9	h10	h11	h12	h13	h14	h15
butterfly UO (UI + LI)																									p0	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13	p14	p15
butterfly LO (UI - LI)																									q0	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	q11	q12	q13	q14	q15
twiddle (LO = LO .* W)																									W0	W2	W4	W6	W8	W10	W12	W14	W0	W2	W4	W6	W8	W10	W12	W14

4.3. Stage 3

4.3.1 Commutator

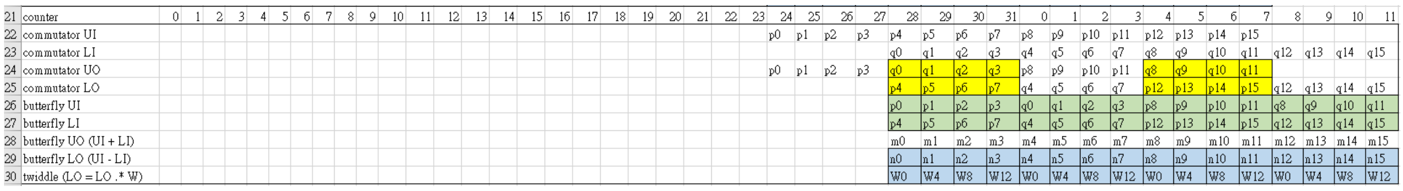
Counter = 28 ~ 31, 4 ~ 7: switch mode

else: bypass mode

4.3.2 Butterfly

Counter = 28 ~ 31, 0 ~ 11: computation mode

else: bypass mode



21	counter	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	0	1	2	3	4	5	6	7	8	9	10	11
22	commutator UI																									p0	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13	p14	p15				
23	commutator LI																													q0	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	q11	q12	q13	q14	q15
24	commutator UO																									p0	p1	p2	p3	q0	q1	q2	q3	p8	p9	p10	p11	q8	q9	q10	q11				
25	commutator LO																													p4	p5	p6	p7	q4	q5	q6	q7	p12	p13	p14	p15	q12	q13	q14	q15
26	butterfly UI																													p0	p1	p2	p3	q0	q1	q2	q3	p8	p9	p10	p11	q8	q9	q10	q11
27	butterfly LI																													p4	p5	p6	p7	q4	q5	q6	q7	p12	p13	p14	p15	q12	q13	q14	q15
28	butterfly UO (UI + LI)																													m0	m1	m2	m3	m4	m5	m6	m7	m8	m9	m10	m11	m12	m13	m14	m15
29	butterfly LO (UI - LI)																													n0	n1	n2	n3	n4	n5	n6	n7	n8	n9	n10	n11	n12	n13	n14	n15
30	twiddle (LO = LO .* W)																													W0	W4	W8	W12	W0	W4	W8	W12	W0	W4	W8	W12	W0	W4	W8	W12

4.4. Stage 4

4.4.1 Commutator

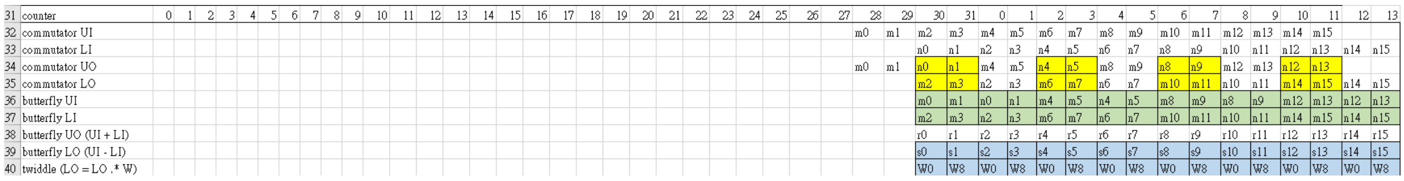
Counter = 30 ~ 31, 2 ~ 3, 6 ~ 7, 10 ~ 11: switch mode

else: bypass mode

4.4.2 Butterfly

Counter = 30 ~ 31, 0 ~ 13: computation mode

else: bypass mode



31	counter	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	0	1	2	3	4	5	6	7	8	9	10	11	12	13
32	commutator UI																													m0	m1	m2	m3	m4	m5	m6	m7	m8	m9	m10	m11	m12	m13	m14	m15		
33	commutator LI																															n0	n1	n2	n3	n4	n5	n6	n7	n8	n9	n10	n11	n12	n13	n14	n15
34	commutator UO																													m0	m1	n0	n1	m4	m5	n4	n5	m8	m9	n8	n9	m12	m13	n12	n13		
35	commutator LO																															m2	m3	n2	n3	m6	m7	n6	n7	m10	m11	n10	n11	m14	m15	n14	n15
36	butterfly UI																															m0	m1	n0	n1	m4	m5	n4	n5	m8	m9	n8	n9	m12	m13	n12	n13
37	butterfly LI																															m2	m3	n2	n3	m6	m7	n6	n7	m10	m11	n10	n11	m14	m15	n14	n15
38	butterfly UO (UI + LI)																															r0	r1	r2	r3	r4	r5	r6	r7	r8	r9	r10	r11	r12	r13	r14	r15
39	butterfly LO (UI - LI)																															s0	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
40	twiddle (LO = LO .* W)																															W0	W8	W0	W8	W0	W8	W0	W8	W0	W8	W0	W8	W0	W8	W0	W8

4.5. Stage 5

4.5.1 Commutator

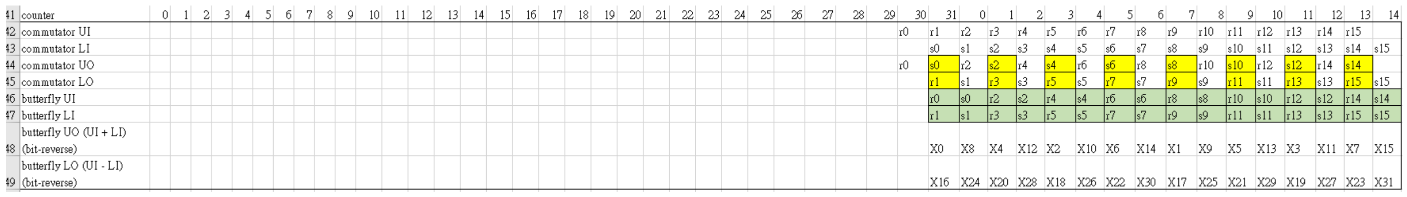
Counter = 31, 1, 3, 5, 7, 9, 11, 13: switch mode

else: bypass mode

4.5.2 Butterfly

Counter = 31, 0 ~ 14: computation mode

else: bypass mode



counter	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
commutator UI																															r0	r1	r2	r3	r4	r5	r6	r7	r8	r9	r10	r11	r12	r13	r14	r15	
commutator LI																																s0	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
commutator UO																															r0	s0	r2	s2	r4	s4	r6	s6	r8	s8	r10	s10	r12	s12	r14	s14	
commutator LO																																r1	s1	r3	s3	r5	s5	r7	s7	r9	s9	r11	s11	r13	s13	r15	s15
butterfly UI																																r0	s0	r2	s2	r4	s4	r6	s6	r8	s8	r10	s10	r12	s12	r14	s14
butterfly LI																																r1	s1	r3	s3	r5	s5	r7	s7	r9	s9	r11	s11	r13	s13	r15	s15
butterfly UO (UI + LI) (bit-reverse)																																X0	X8	X4	X12	X2	X10	X6	X14	X1	X9	X5	X13	X3	X11	X7	X15
butterfly LO (UI - LI) (bit-reverse)																																X16	X24	X20	X28	X18	X26	X22	X30	X17	X25	X21	X29	X19	X27	X23	X31

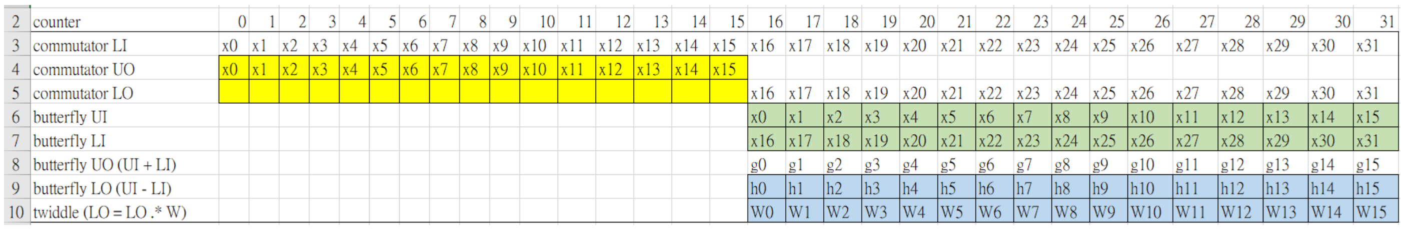
5. (Step 5) Generate the control signals of complex multipliers for each stage. Show how you generate them and why? (10%)

5.1. Stage 1

5.1.1 Multipliers

Counter = 16 ~ 31: Multiplication mode

else: bypass mode



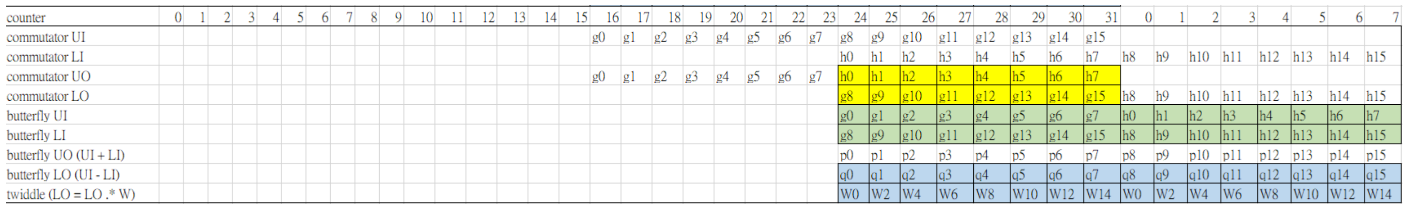
counter	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
commutator LI	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15	x16	x17	x18	x19	x20	x21	x22	x23	x24	x25	x26	x27	x28	x29	x30	x31
commutator UO	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15																
commutator LO																	x16	x17	x18	x19	x20	x21	x22	x23	x24	x25	x26	x27	x28	x29	x30	x31
butterfly UI																	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15
butterfly LI																	x16	x17	x18	x19	x20	x21	x22	x23	x24	x25	x26	x27	x28	x29	x30	x31
butterfly UO (UI + LI)																	g0	g1	g2	g3	g4	g5	g6	g7	g8	g9	g10	g11	g12	g13	g14	g15
butterfly LO (UI - LI)																	h0	h1	h2	h3	h4	h5	h6	h7	h8	h9	h10	h11	h12	h13	h14	h15
twiddle (LO = LO .* W)																	W0	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15

5.2. Stage 2

5.2.1 Multipliers

Counter = 24 ~ 31, 0 ~ 7: Multiplication mode

else: bypass mode



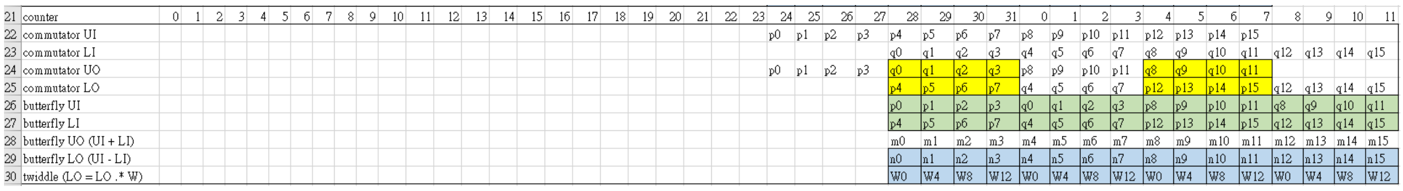
counter	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	0	1	2	3	4	5	6	7
commutator UI																	g0	g1	g2	g3	g4	g5	g6	g7	g8	g9	g10	g11	g12	g13	g14	g15								
commutator LI																									h0	h1	h2	h3	h4	h5	h6	h7	h8	h9	h10	h11	h12	h13	h14	h15
commutator UO																	g0	g1	g2	g3	g4	g5	g6	g7	h0	h1	h2	h3	h4	h5	h6	h7								
commutator LO																									g8	g9	g10	g11	g12	g13	g14	g15	h8	h9	h10	h11	h12	h13	h14	h15
butterfly UI																									g0	g1	g2	g3	g4	g5	g6	g7	h0	h1	h2	h3	h4	h5	h6	h7
butterfly LI																									g8	g9	g10	g11	g12	g13	g14	g15	h8	h9	h10	h11	h12	h13	h14	h15
butterfly UO (UI + LI)																									p0	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13	p14	p15
butterfly LO (UI - LI)																									q0	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	q11	q12	q13	q14	q15
twiddle (LO = LO .* W)																									W0	W2	W4	W6	W8	W10	W12	W14	W0	W2	W4	W6	W8	W10	W12	W14

5.3. Stage 3

5.3.1 Multipliers

Counter = 28 ~ 31, 0 ~ 11: Multiplication mode

else: bypass mode



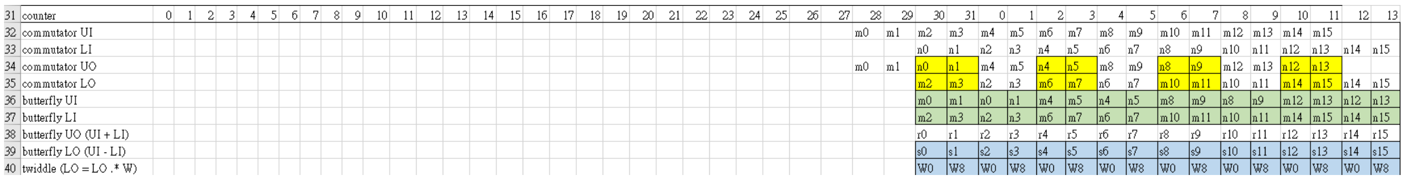
21	counter	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	0	1	2	3	4	5	6	7	8	9	10	11
22	commutator UI																									p0	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13	p14	p15				
23	commutator LI																													q0	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	q11	q12	q13	q14	q15
24	commutator UO																									p0	p1	p2	p3	q0	q1	q2	q3	p8	p9	p10	p11	q8	q9	q10	q11				
25	commutator LO																													p4	p5	p6	p7	q4	q5	q6	q7	p12	p13	p14	p15	q12	q13	q14	q15
26	butterfly UI																													p0	p1	p2	p3	q0	q1	q2	q3	p8	p9	p10	p11	q8	q9	q10	q11
27	butterfly LI																													p4	p5	p6	p7	q4	q5	q6	q7	p12	p13	p14	p15	q12	q13	q14	q15
28	butterfly UO (UI + LI)																													m0	m1	m2	m3	m4	m5	m6	m7	m8	m9	m10	m11	m12	m13	m14	m15
29	butterfly LO (UI - LI)																													n0	n1	n2	n3	n4	n5	n6	n7	n8	n9	n10	n11	n12	n13	n14	n15
30	twiddle (LO = LO .* W)																													W0	W4	W8	W12	W0	W4	W8	W12	W0	W4	W8	W12	W0	W4	W8	W12

5.4. Stage 4

5.4.1 Multipliers

Counter = 30 ~ 31, 0 ~ 13: Multiplication mode

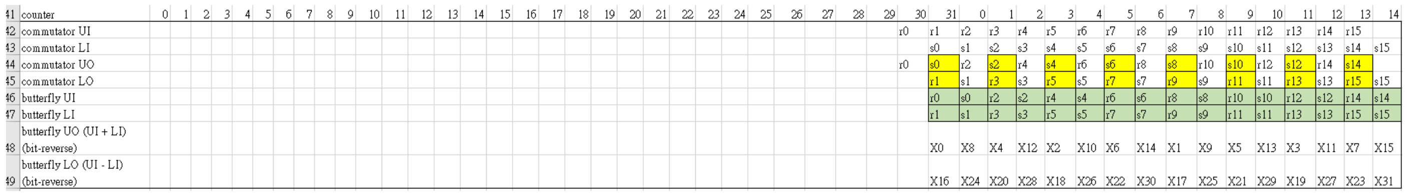
else: bypass mode



31	counter	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	0	1	2	3	4	5	6	7	8	9	10	11	12	13
32	commutator UI																													m0	m1	m2	m3	m4	m5	m6	m7	m8	m9	m10	m11	m12	m13	m14	m15		
33	commutator LI																															n0	n1	n2	n3	n4	n5	n6	n7	n8	n9	n10	n11	n12	n13	n14	n15
34	commutator UO																													m0	m1	n0	n1	m4	m5	n4	n5	m8	m9	n8	n9	m12	m13	n12	n13		
35	commutator LO																															m2	m3	n2	n3	m6	m7	n6	n7	m10	m11	n10	n11	m14	m15	n14	n15
36	butterfly UI																															m0	m1	n0	n1	m4	m5	n4	n5	m8	m9	n8	n9	m12	m13	n12	n13
37	butterfly LI																															m2	m3	n2	n3	m6	m7	n6	n7	m10	m11	n10	n11	m14	m15	n14	n15
38	butterfly UO (UI + LI)																															r0	r1	r2	r3	r4	r5	r6	r7	r8	r9	r10	r11	r12	r13	r14	r15
39	butterfly LO (UI - LI)																															s0	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
40	twiddle (LO = LO .* W)																															W0	W8	W0	W8	W0	W8	W0	W8	W0	W8	W0	W8	W0	W8	W0	W8

4.5. Stage 5

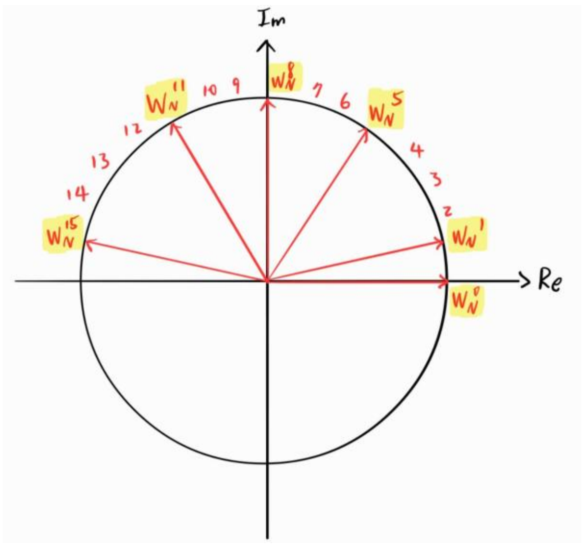
4.5.1 Multipliers: bypass mode (no need to use a multiplier)

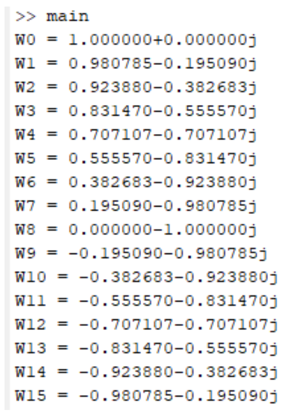


41	counter	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
42	commutator UI																															r0	r1	r2	r3	r4	r5	r6	r7	r8	r9	r10	r11	r12	r13	r14	r15	
43	commutator LI																																s0	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
44	commutator UO																															r0	s0	r2	s2	r4	s4	r6	s6	r8	s8	r10	s10	r12	s12	r14	s14	
45	commutator LO																																r1	s1	r3	s3	r5	s5	r7	s7	r9	s9	r11	s11	r13	s13	r15	s15
46	butterfly UI																																r0	s0	r2	s2	r4	s4	r6	s6	r8	s8	r10	s10	r12	s12	r14	s14
47	butterfly LI																																r1	s1	r3	s3	r5	s5	r7	s7	r9	s9	r11	s11	r13	s13	r15	s15
48	butterfly UO (UI + LI) (bit-reverse)																																X0	X8	X4	X12	X2	X10	X6	X14	X1	X9	X5	X13	X3	X11	X7	X15
49	butterfly LO (UI - LI) (bit-reverse)																																X16	X24	X20	X28	X18	X26	X22	X30	X17	X25	X21	X29	X19	X27	X23	X31

6. (Step 6) Explain the way that you use the sine/cosine values in the first quadrant to generate the required 32 phases for ROM 32. Draw the block diagram (10%)

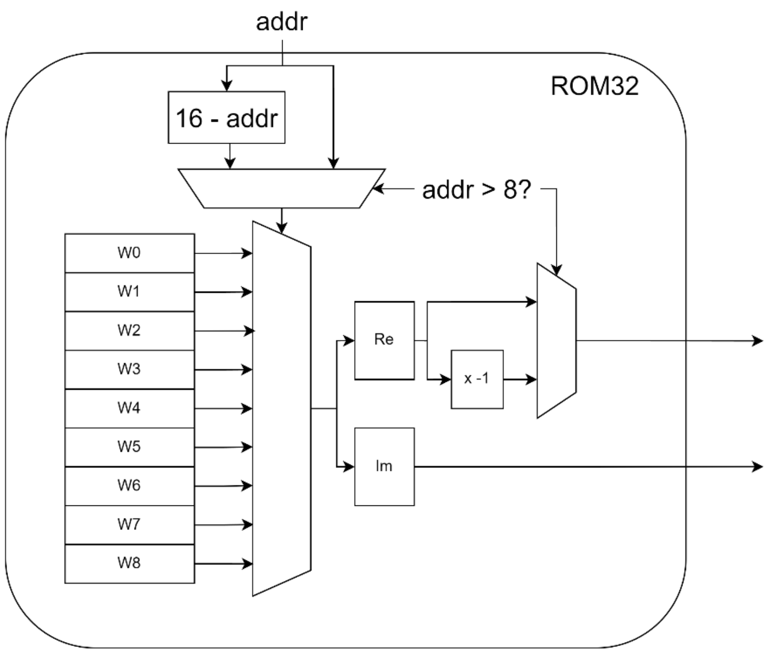
As shown in the figure below, we can observe that the values in the first quadrant (W1 to W7) are the same as those in the second quadrant (W15 to W9), except for the sign of the real part. Ex: $Im\{W_N^5\} = Im\{W_N^{11}\},\ Re\{W_N^5\} = -Re\{W_N^{11}\}$. So, we only need to save the values in the first quadrant.





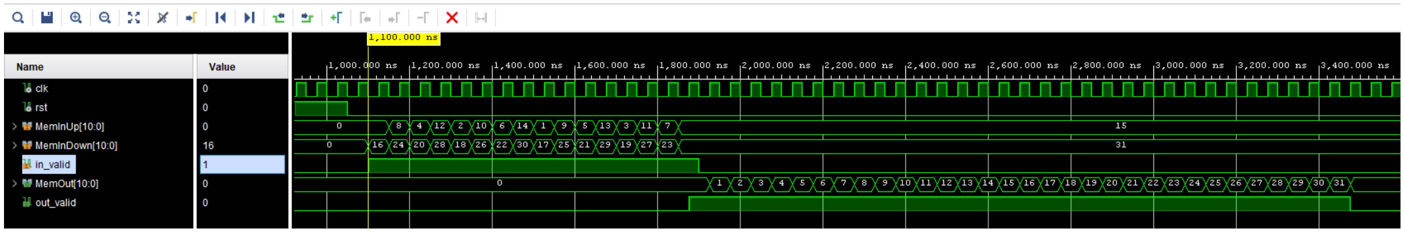
```
>> main
W0 = 1.000000+0.000000j
W1 = 0.980785-0.195090j
W2 = 0.923880-0.382683j
W3 = 0.831470-0.555570j
W4 = 0.707107-0.707107j
W5 = 0.555570-0.831470j
W6 = 0.382683-0.923880j
W7 = 0.195090-0.980785j
W8 = 0.000000-1.000000j
W9 = -0.195090-0.980785j
W10 = -0.382683-0.923880j
W11 = -0.555570-0.831470j
W12 = -0.707107-0.707107j
W13 = -0.831470-0.555570j
W14 = -0.923880-0.382683j
W15 = -0.980785-0.195090j
```

6.1. Block diagram

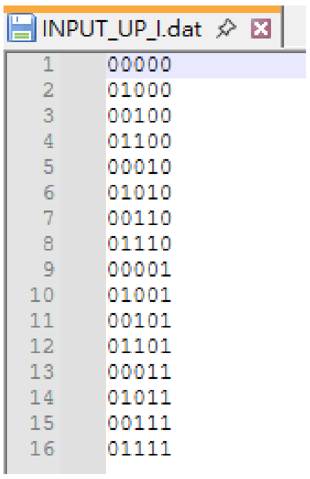


7. (Step 7) Show the timing diagram of behavior simulation for bit-reverse re-ordering with ping-pong accessed bit-reversal buffer as in Fig. 9. (15%)

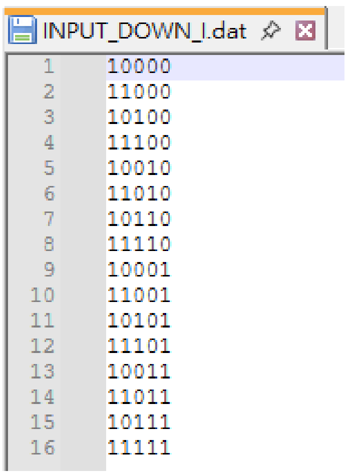
7.1 timing diagram:



7.2 test pattern: MemInUp & MemInDown


INPUT_UP_I.dat

```
00000
01000
00100
01100
00010
01010
00110
01110
00001
01001
00101
01101
00011
01011
00111
01111
```

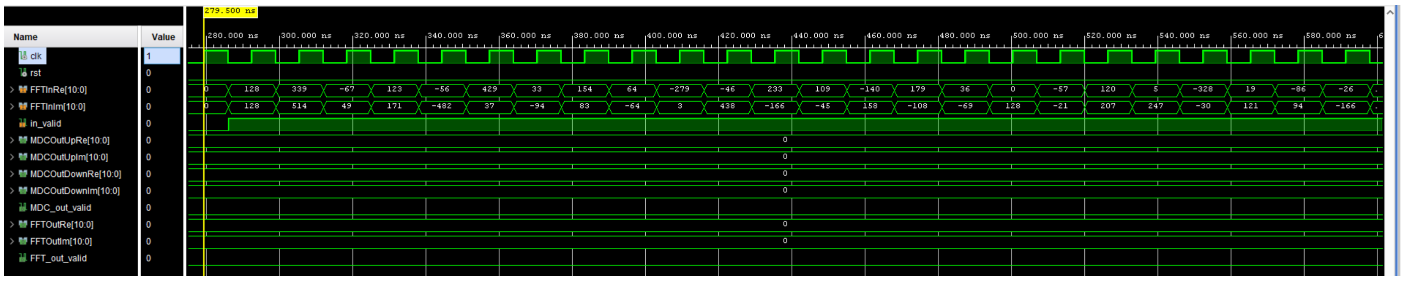

INPUT_DOWN_I.dat

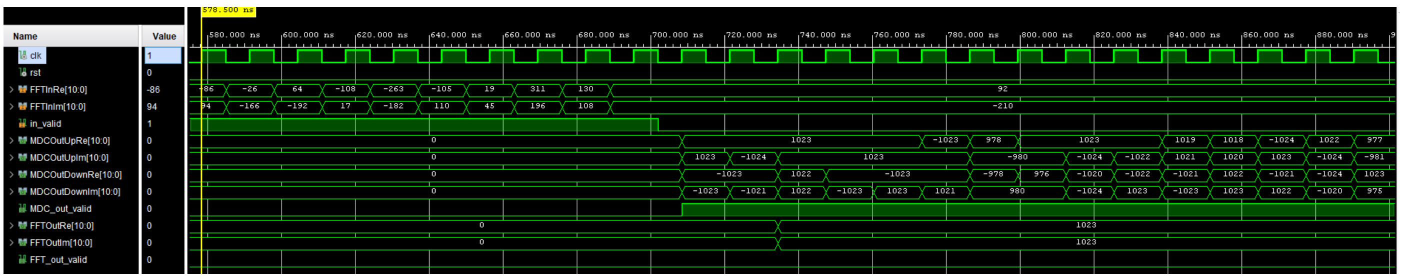
```
10000
11000
10100
11100
10010
11010
10110
11110
10001
11001
10101
11101
10011
11011
10111
11111
```

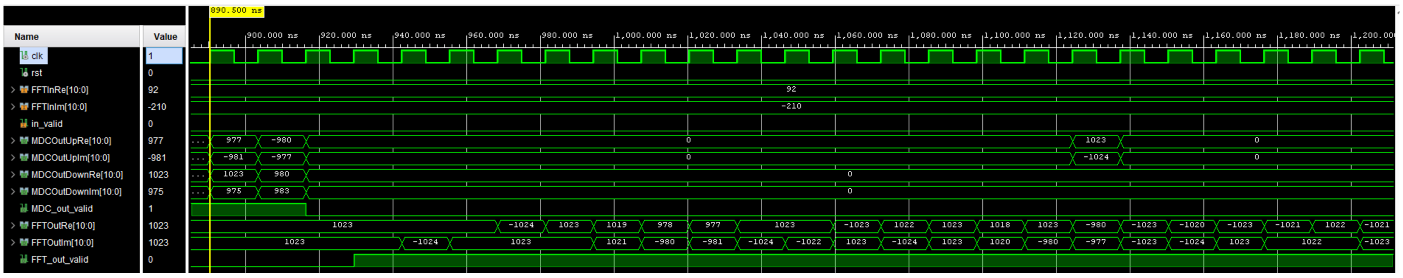
8. (Step 8) Please use FFTInput32.mat as the inputs. Quantize them using the word-length selected by yourself. Show the timing diagram of behavior simulation for MDC FFT. Compared the results to your answer in Q1(Step 1). Draw the error for 32 samples of real part and imaginary part. (25%)

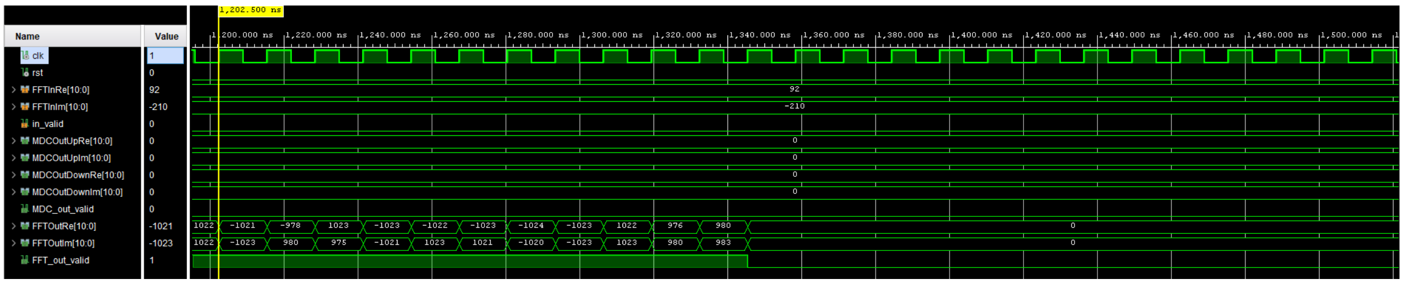
8.1. timing diagram of behavior simulation

The values are shown in signed decimal with 10 fractional bit-width, so divided it by 1024 is the real value.

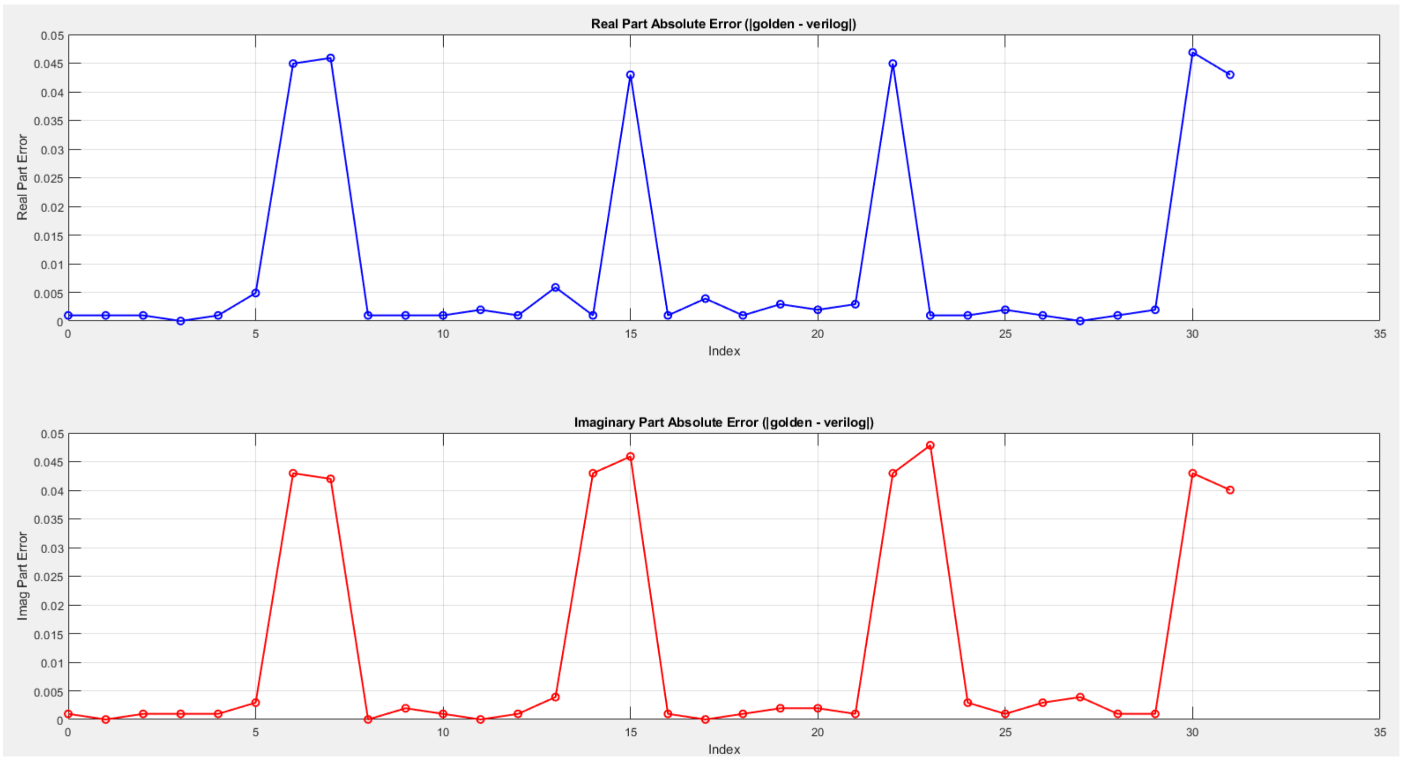








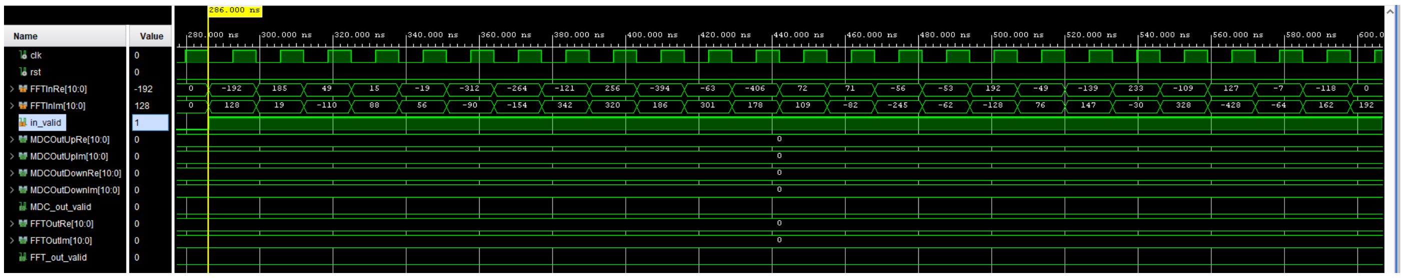
8.2. The error for 32 samples of real part and imaginary part

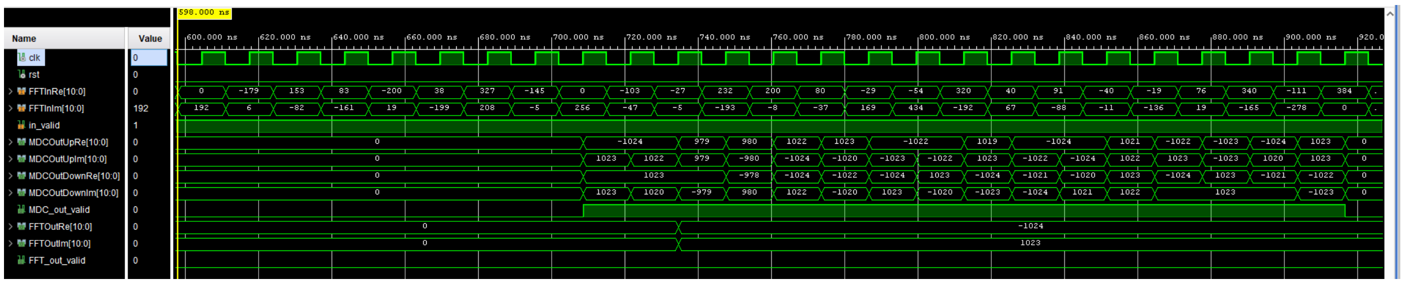


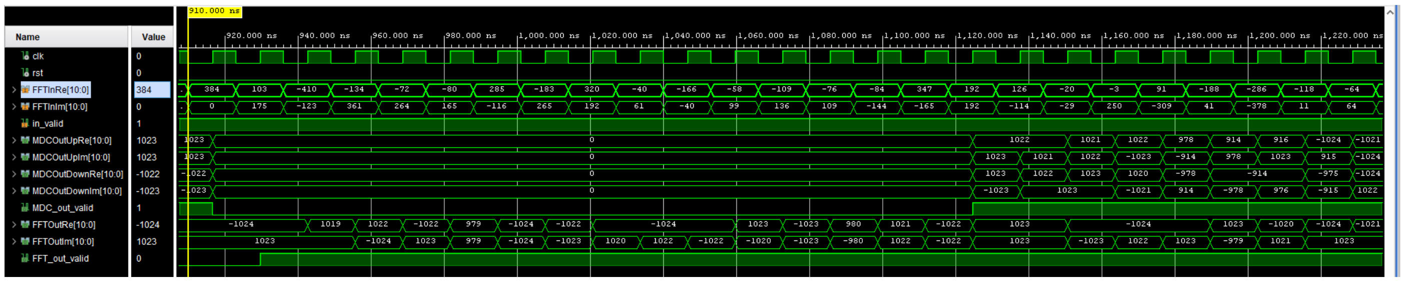
9. (Step 9) Use your own inputs of 96 samples in Q3 (Step3). Show the timing diagram of behavior simulation with streaming-input and streaming-output. Draw the error for 96 samples of real part and imaginary part. (25%) Calculate the SQNR of 96 samples.

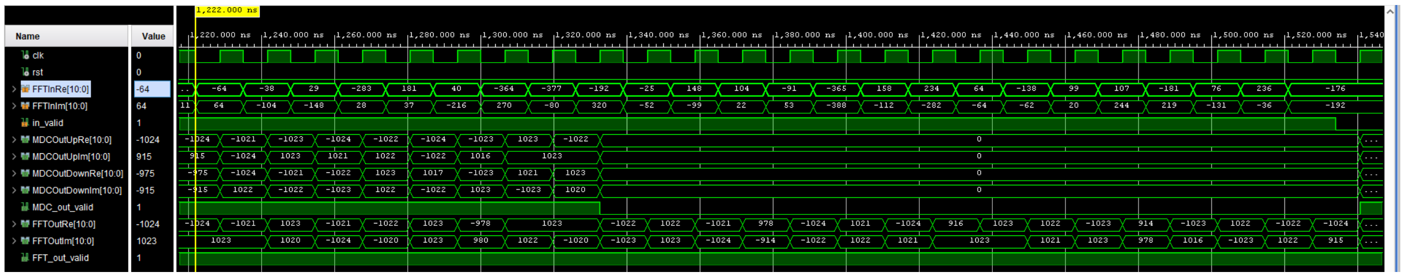
9.1 timing diagram of behavior simulation

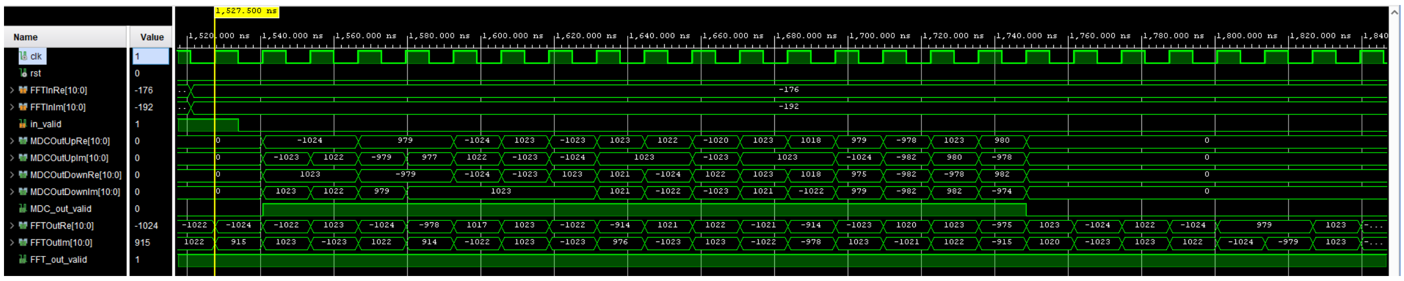
The values are shown in signed decimal with 10 fractional bit-width, so divided it by 1024 is the real value.



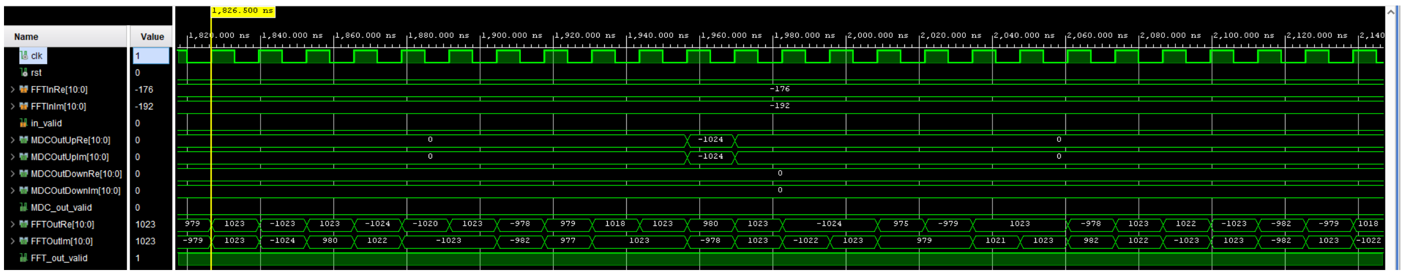




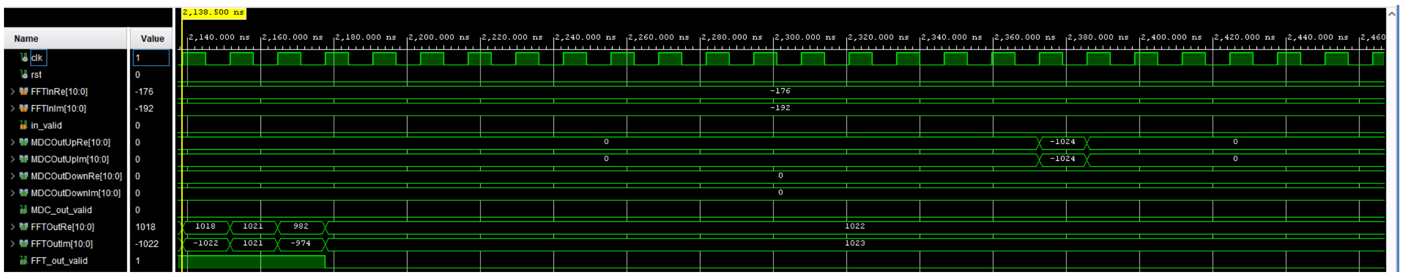




Name	Value
clk	1
rst	0
FFTInRe[10:0]	-176
FFTInIm[10:0]	-192
in_valid	1
MDCOutUpRe[10:0]	0
MDCOutUpIm[10:0]	0
MDCOutDownRe[10:0]	0
MDCOutDownIm[10:0]	0
MDC_out_valid	0
FFTOutRe[10:0]	-1024
FFTOutIm[10:0]	915
FFT_out_valid	1

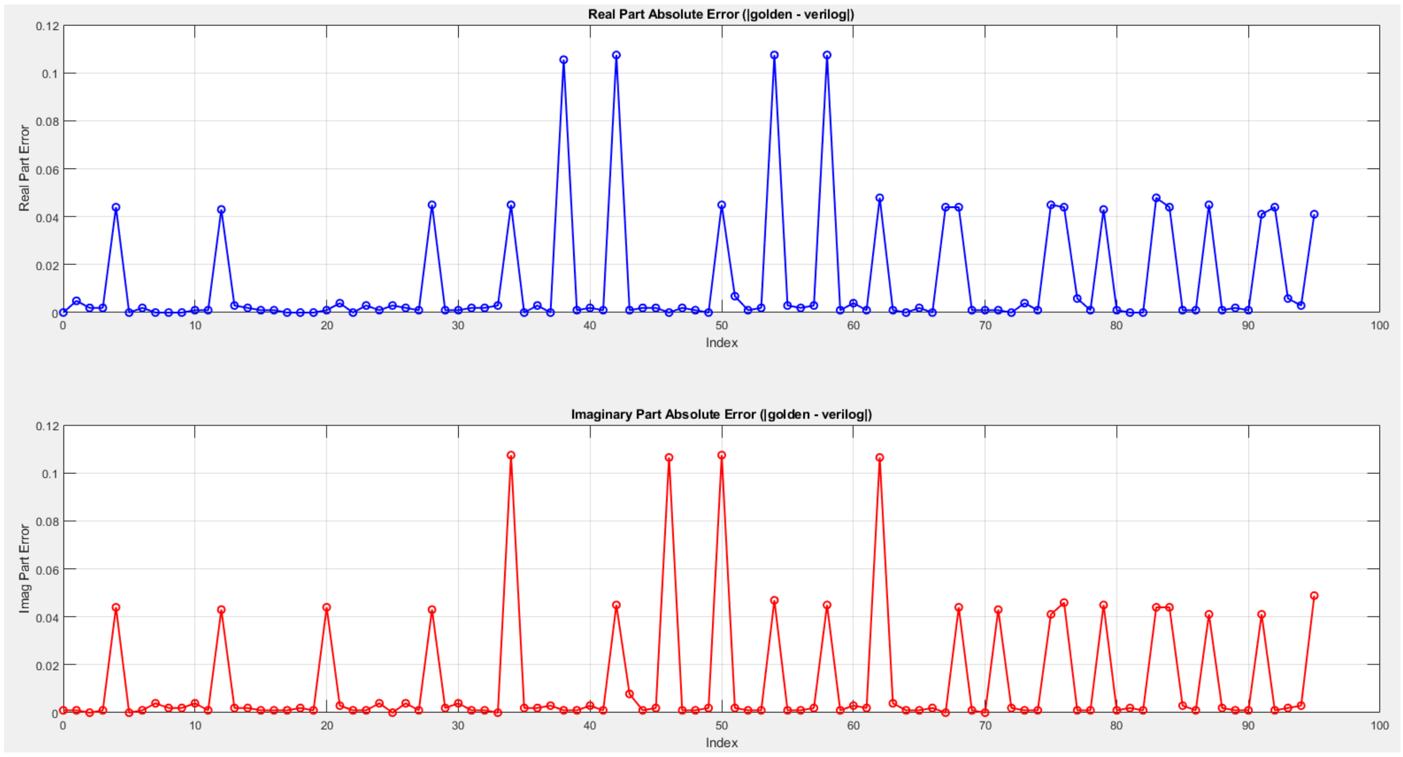


Name	Value
clk	1
rst	0
FFTInRe[10:0]	-176
FFTInIm[10:0]	-192
in_valid	0
MDCOutUpRe[10:0]	0
MDCOutUpIm[10:0]	0
MDCOutDownRe[10:0]	0
MDCOutDownIm[10:0]	0
MDC_out_valid	0
FFTOutRe[10:0]	1023
FFTOutIm[10:0]	1023
FFT_out_valid	1



Name	Value
clk	1
rst	0
FFTInRe[10:0]	-176
FFTInIm[10:0]	-192
in_valid	0
MDCOutUpRe[10:0]	0
MDCOutUpIm[10:0]	0
MDCOutDownRe[10:0]	0
MDCOutDownIm[10:0]	0
MDC_out_valid	0
FFTOutRe[10:0]	1018
FFTOutIm[10:0]	-1022
FFT_out_valid	1

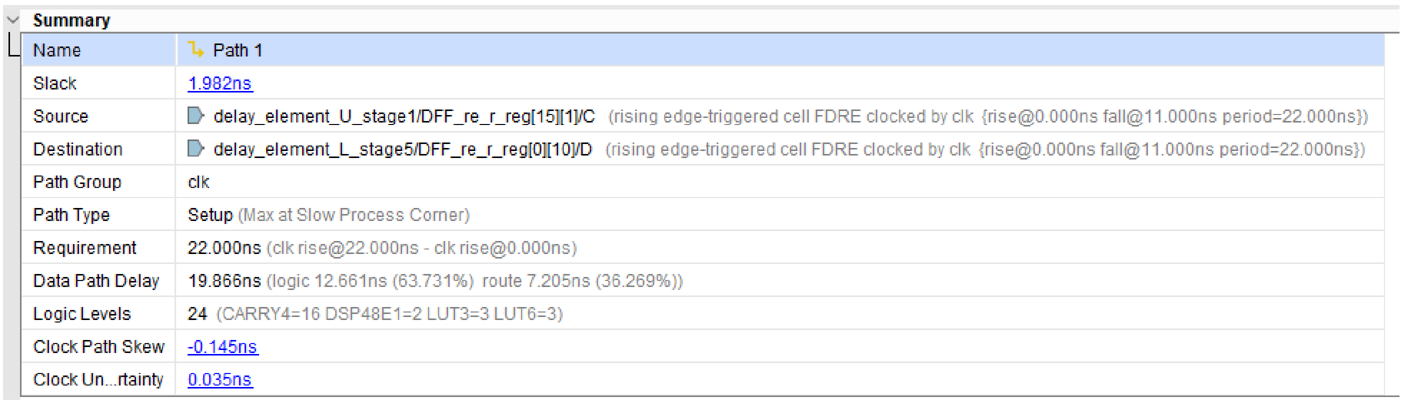
9.2 the error for 96 samples of real part and imaginary part and SQNR



```
FFT96 SQNR = 30.83 dB
```

10. (Step 10) Insert D flip-flop at the input and output. Synthesize your design. Provide the report of max delay. Note that if you did not insert internal pipeline registers, the critical path is long and the operating frequency is not high. (10%)

max delay = 19.866ns (operating frequency = 50.3MHz)

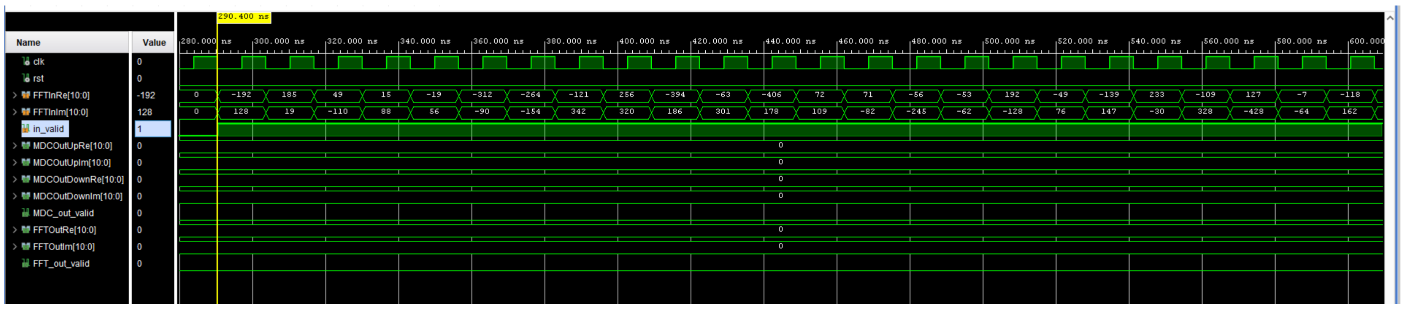


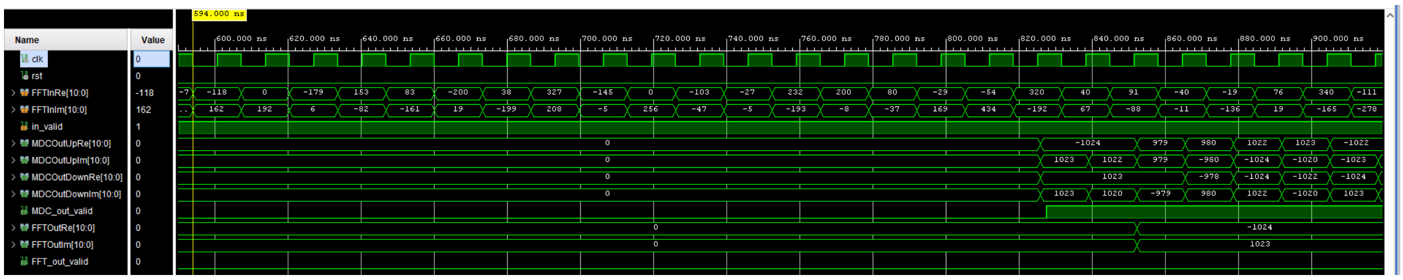
Summary

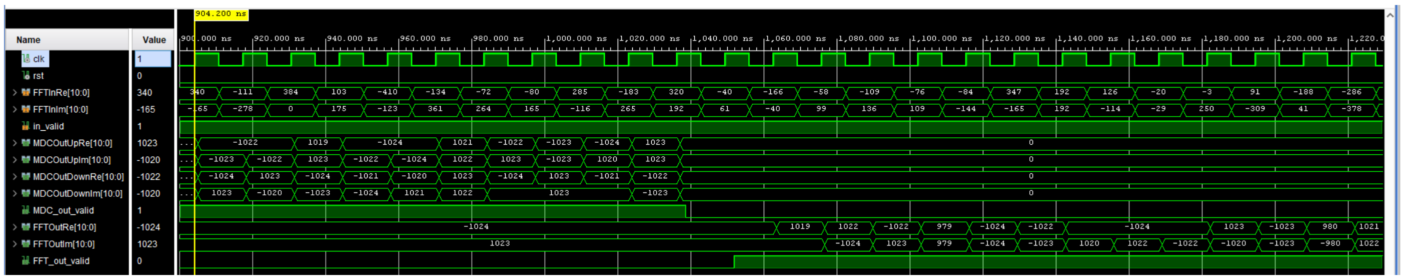
Name	Path 1
Slack	1.982ns
Source	delay_element_U_stage1/DFF_re_r_reg[15][1]/C (rising edge-triggered cell FDRE clocked by clk {rise@0.000ns fall@11.000ns period=22.000ns})
Destination	delay_element_L_stage5/DFF_re_r_reg[0][10]/D (rising edge-triggered cell FDRE clocked by clk {rise@0.000ns fall@11.000ns period=22.000ns})
Path Group	clk
Path Type	Setup (Max at Slow Process Corner)
Requirement	22.000ns (clk rise@22.000ns - clk rise@0.000ns)
Data Path Delay	19.866ns (logic 12.661ns (63.731%) route 7.205ns (36.269%))
Logic Levels	24 (CARRY4=16 DSP48E1=2 LUT3=3 LUT6=3)
Clock Path Skew	-0.145ns
Clock Un...rtainty	0.035ns

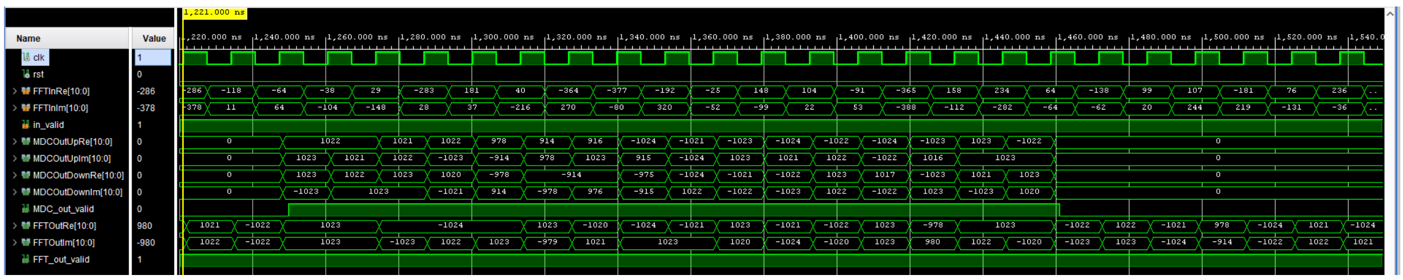
11. (Step 11) Insert pipeline registers to accelerate your design. For FPGA flow, the target operating clock frequency (fs) is 75MHz. For cell-based design flow, the target operating frequency (fs) is 130MHz. Show the timing diagram of post-synthesis simulation results with proper clock period settings ($1/fs$) . Draw the error for 96 samples of real part and imaginary part. (25%)

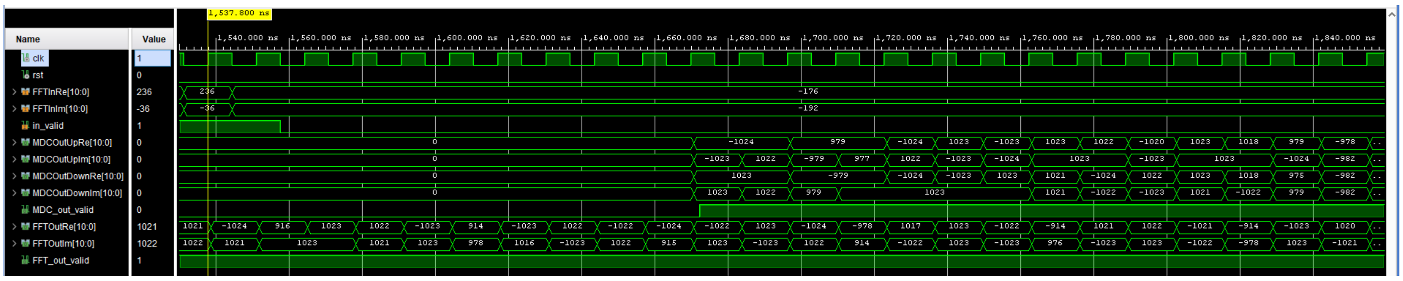
11.1 Clk period = 13.2 ns (operating frequency = 75.75MHz):

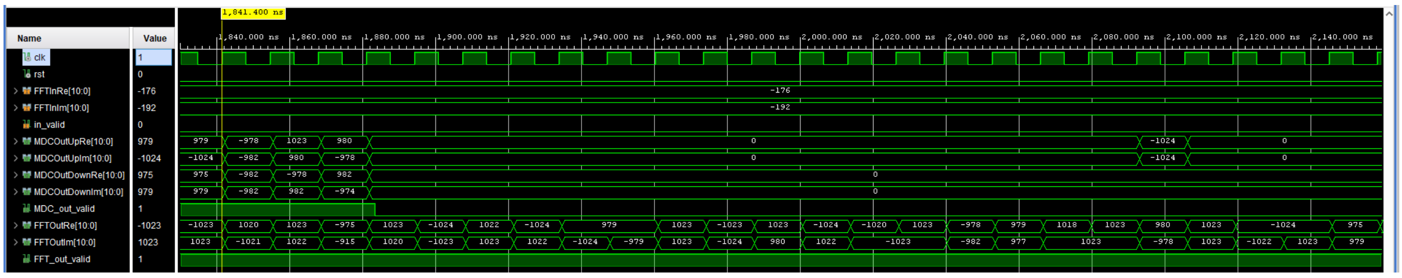


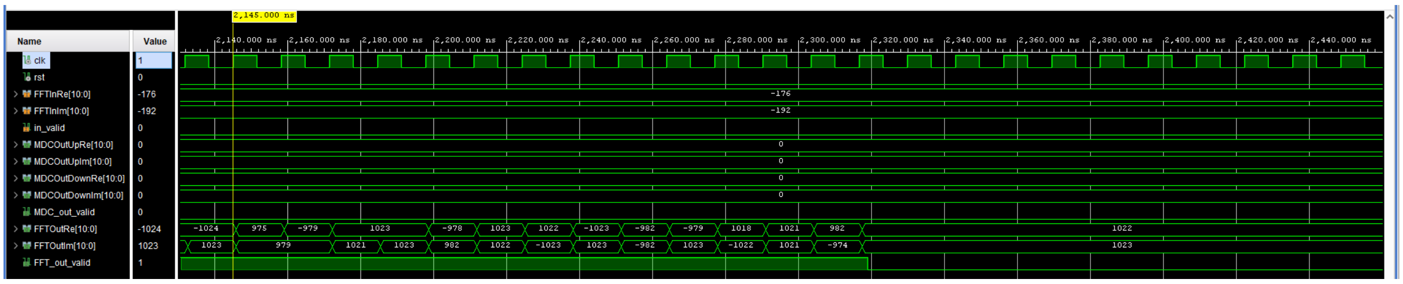




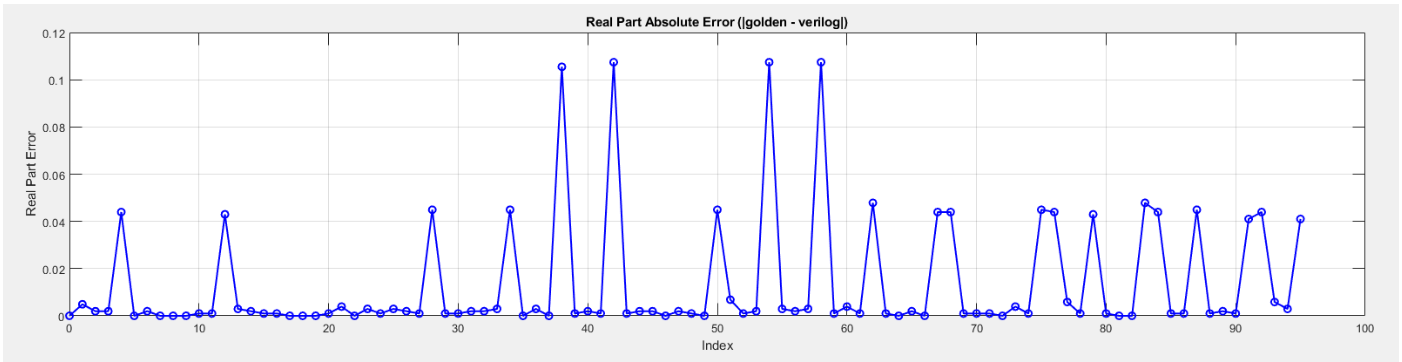


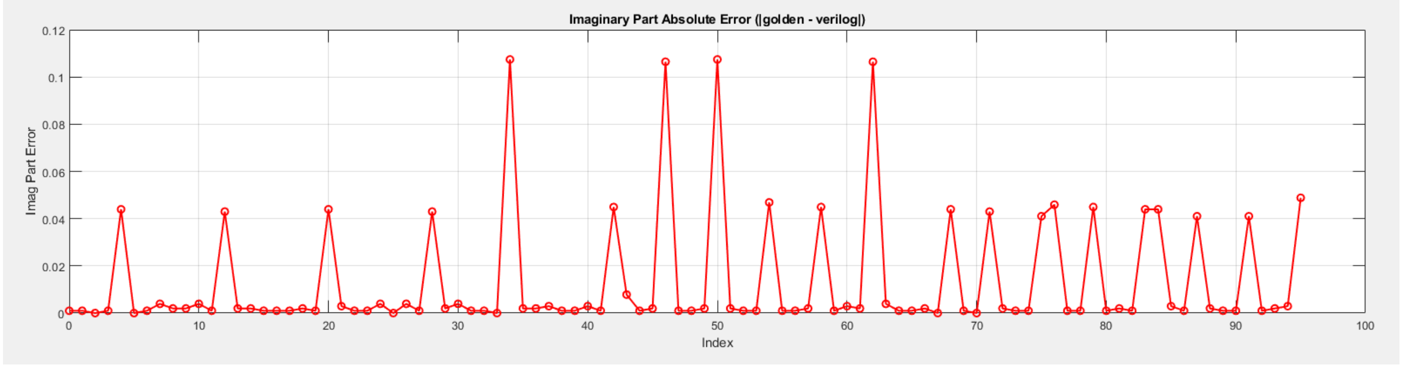






11.2 the error for 96 samples of real part and imaginary part


Real Part Absolute Error (|golden - verilog|)
Real Part Error
Index


Imaginary Part Absolute Error (|golden - verilog|)
Imag Part Error
Index