S2 6201011631188 โสภณ สุขสมบูรณ์

17 กุมภาพันธ์ 2565

ปฏิบัติการที่ 4Convolution sum of fixed-point in FPGA

การทดลองที่ 1 แสดงผลการ Convolution sum โดยใช้ Matlab

```
%ประกาศตัวแปร
f1=11e3;
f2=88e3;
fs=f2*4;
n=0:200;
t=n*(1/fs);
l=length(t);
N=2.^nextpow2(l);
f=fs*(0:(N/2))/N;

x1=1*sin(2*pi*f1*t);
x2=2*cos(2*pi*f2*t);

y=x1+x2;
y_fix=num2fixpt(y,sfix(8),2^-5); %Binary

%C of FIR Highpass-filter
Coe_filter=[0.004960022027719, 0.005577211783262, -0.01899219729718,  -0.1066292338452,...
    -0.2272326527617,   0.7195743880616,  -0.2272326527617,  -0.1066292338452,...
   -0.01899219729718, 0.005577211783262, 0.004960022027719];
hpf2fix=num2fixpt(Coe_filter,sfix(8),2^-7);

%Z in binary
z=conv(y_fix,hpf2fix);

```

```
26 -    intY = y_fix*(2^5);
27 -    intH = hpf2fix*(2^7);
28 -    binY=dec2bin(intY);
29 -    binH=dec2bin(intH);
30 -    Z=conv(intY,intH);
31 -    binZ=dec2bin(Z)/(2^12);
32
33
```

คำอธิบาย

-ตัวแปร y_fix คือ ตัวแปร y ที่เป็นค่า fixed-point

-ตัวแปร hpf2fix คือ ตัวแปรที่รับค่าสัมประสิทธิ์ของhigh-pass filter ที่เป็น fixed-point

-ตัวแปร intY และ intH เป็นการนำค่า y และ H (Coefficient of high-pass filter) ที่ทำ fixed-point แล้วมา ยกกำลังกลับด้วยค่า2^-n โดยที่ n ตำแหน่งของLSB ที่เรากำหนดในคำสั่ง num2fixpt เพื่อนำค่าที่ได้ ไปใช้ในคำสั่งถัดไป

-ตัวแปร binY ,binHและ binZ เป็นตัวแปรที่จะดำเนินคำสั่ง dec2bin ซึ่งเป็นคำสั่งที่แปลงค่าของเลขฐานสิบ เป็น เลขฐานสอง ด้วยวิธีนี้ ค่าในตัวแปร binY ,binH และ binZ จะเป็นbinary ทั้งหมด

การแสดงผล

Input : y_fix

1x201 double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	2	0.1875	-1.6250	0.5313	2.6875	0.8125	-1.0938	0.9688	3	0.9688	-1.0938	0.8125	2.6875	0.5313	-1.6250	0.1875
2																
3																
4																
5																

intY (สำหรับQuartus II)

1x201 double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	64	6	-52	17	86	26	-35	31	96	31	-35	26	86	17	-52	6	64	-7	-77	-18	41
2																					
3																					
4																					

Impulse response : Hpf2fix

hpf2fix | intH | z | Z | binY | binH | binZ | y_fix | intY

1x11 double

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	-0.0234	-0.1094	-0.2344	0.7188	-0.2344	-0.1094	-0.0234	0	0	
2												

intH (สำหรับQuartus II)

1x11 double

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	-3	-14	-30	92	-30	-14	-3	0	0	
2												
3												

Output : z

1x211 double

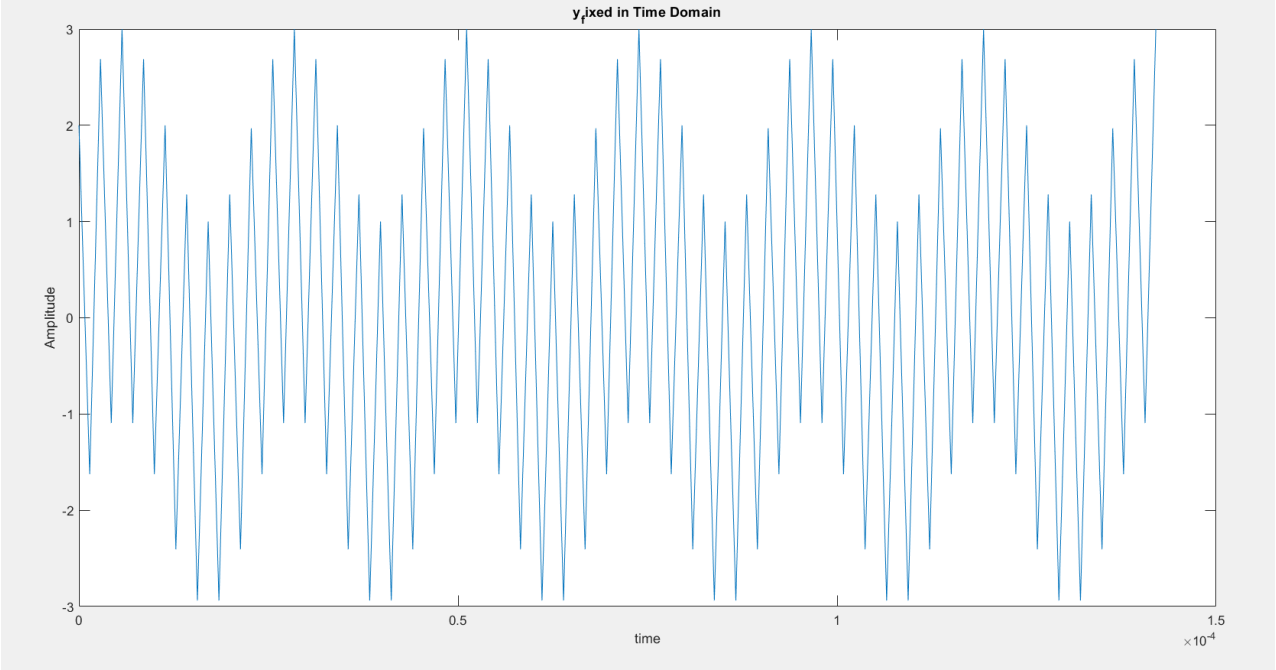
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	0	0	-0.0469	-0.2231	-0.4512	1.5588	-0.0742	-1.8682	0.0022	1.8870	0.0142	-1.8608	0.0173	1.9033	0.0173	-1.8608	0.0142	1.8870	0.0022	-1.8630	0.0061
2																					
3																					
4																					

Z (สำหรับเช็คผลการทดลองMatlabและQuartus II)

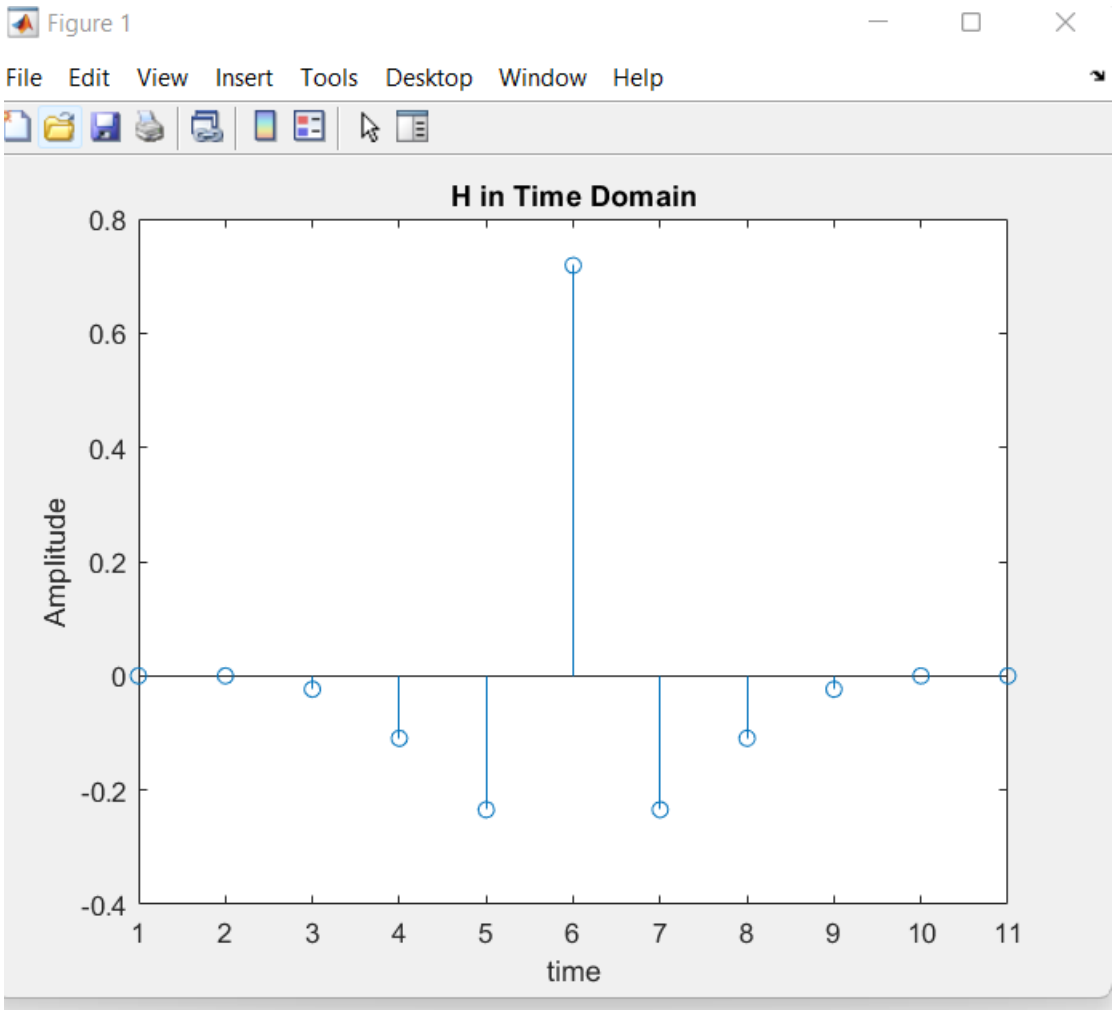
1x211 double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	0	0	-192	-914	-1848	6385	-304	-7652	9	7729	58	-7622	71	7796	71	-7622	58	7729	9	-7631	25
2																					
3																					
4																					

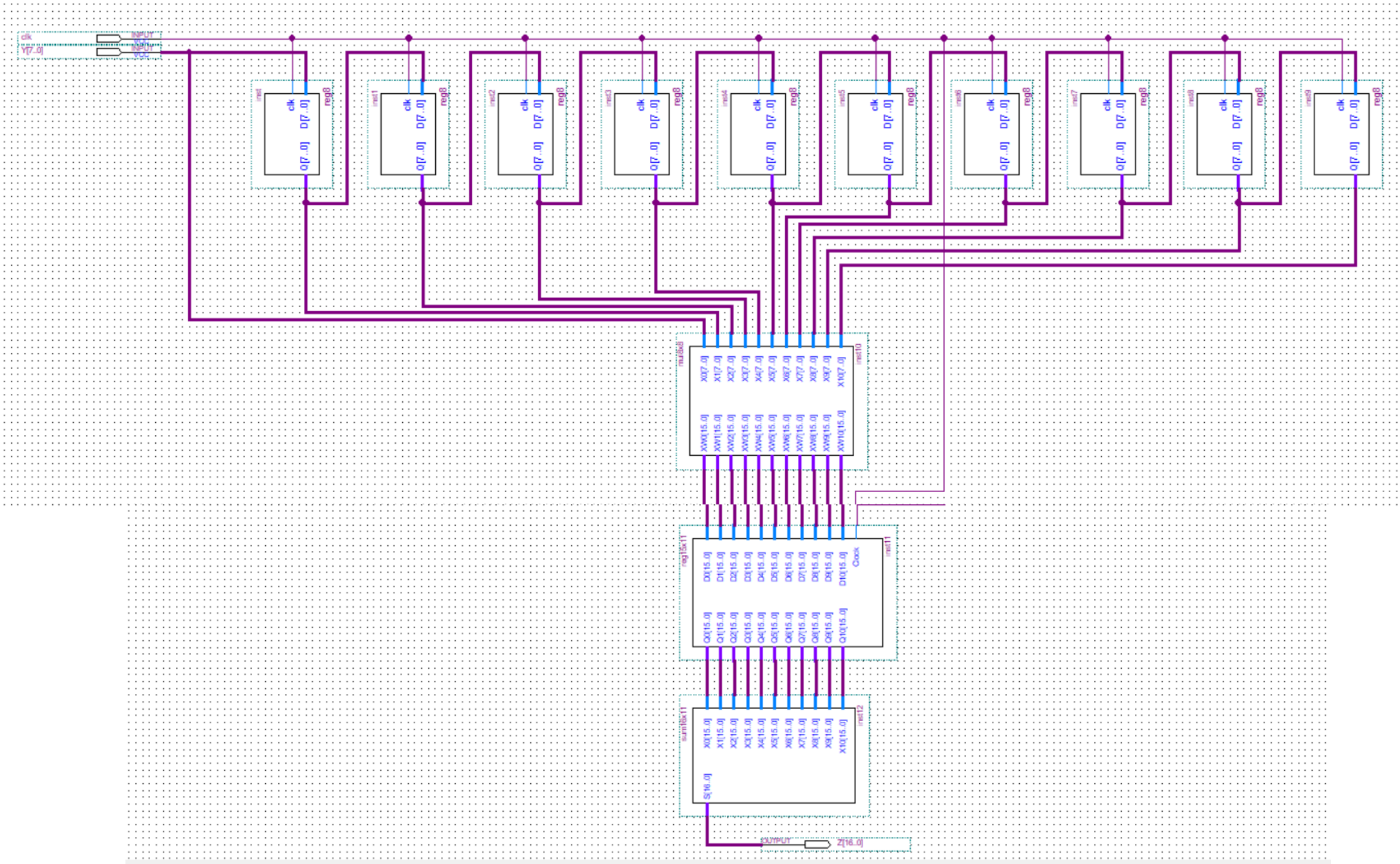
Plot(t,y_fixed)



Stem(hpf2fix)

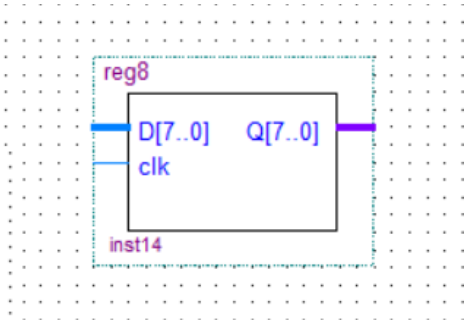


การทดลองที่2 แสดงผลการ Convolution sum in Quartus II (FPGA)

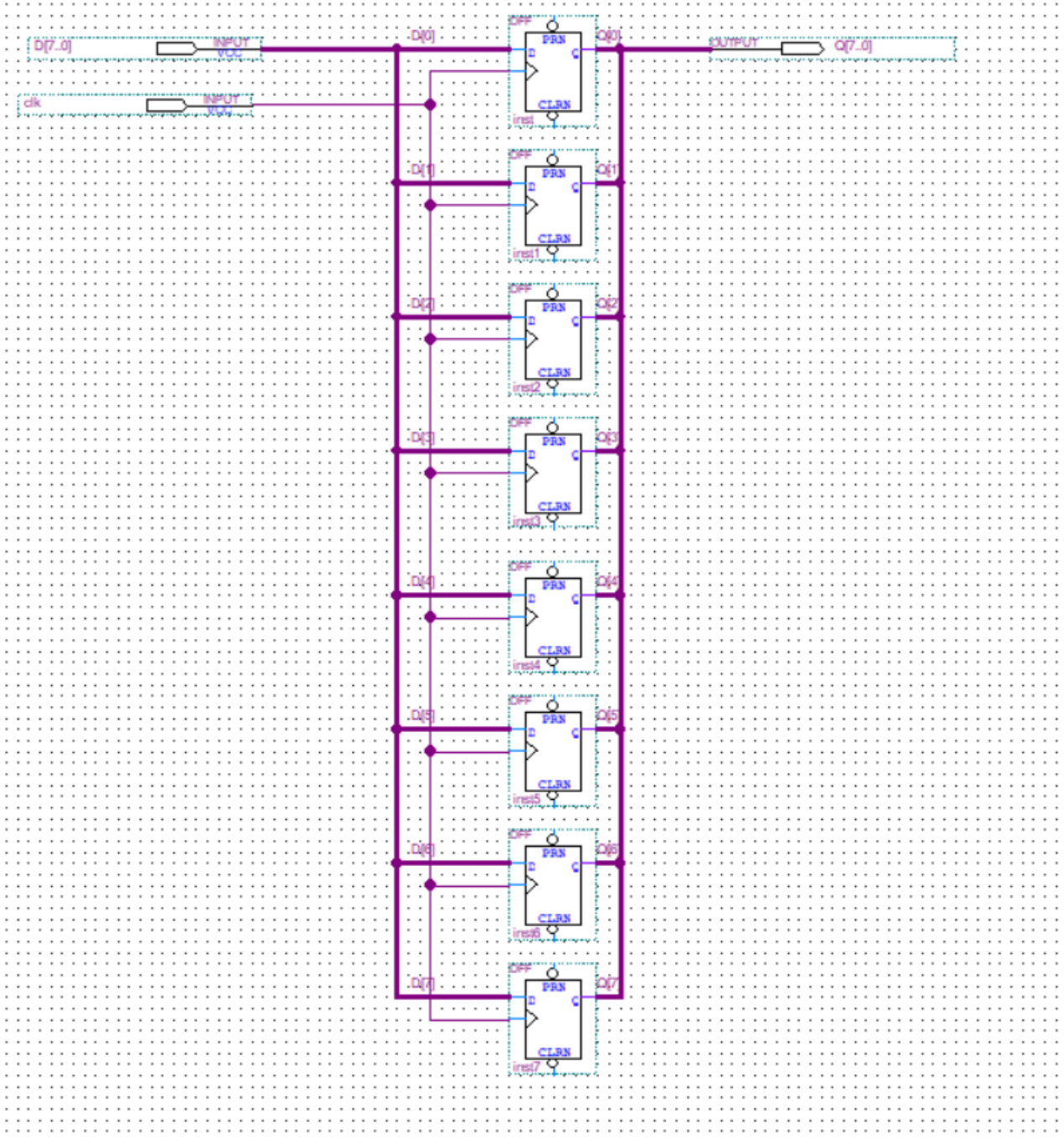


คำอธิบาย Symbol

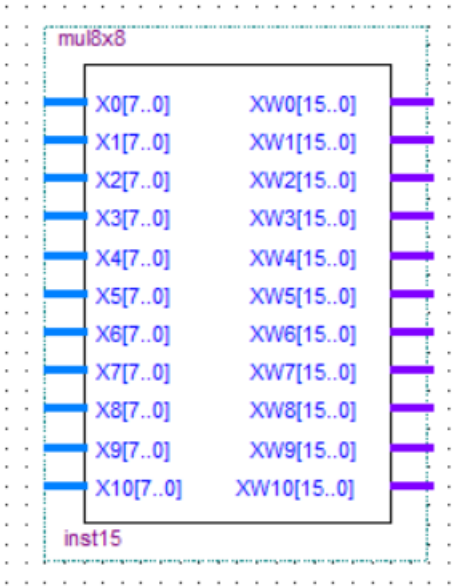
1. Rigister เก็บค่า 8 bits



ภายในSymbol



2.mul8x8 (วงจรคูณ)



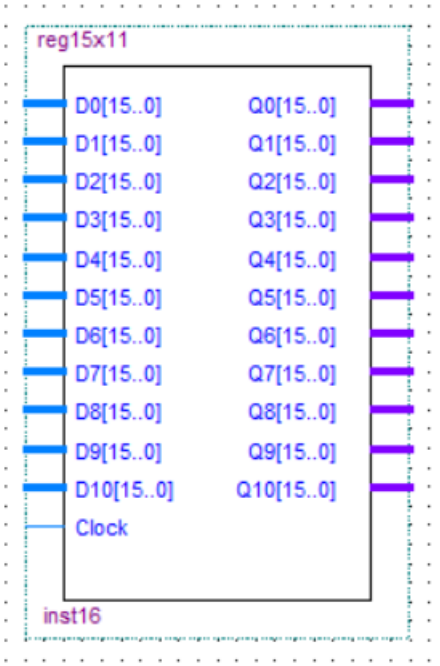
ภายในSymbol

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3   use IEEE.NUMERIC_STD.ALL;
4
5   ENTITY mul8x8 IS
6   PORT    ( X0,X1,X2,X3,X4,X5,X6,X7,X8,X9,X10 :IN SIGNED (7 DOWNTO 0);
7             XW0,XW1,XW2,XW3,XW4,XW5,XW6,XW7,XW8,XW9,XW10 :OUT SIGNED (15 DOWNTO 0)
8           );
9   END mul8x8;
10
11  ARCHITECTURE Behavior OF mul8x8 IS
12  SIGNAL W0,W1,W2,W3,W4,W5,W6,W7,W8,W9,W10 : SIGNED (7 DOWNTO 0) ;
13  begin
14      W0 <= "00000000";
15      W1 <= "00000000";
16      W2 <= "11111101";
17      W3 <= "11110010";
18      W4 <= "11100010";
19      W5 <= "01011100";
20      W6 <= "11100010";
21      W7 <= "11110010";
22      W8 <= "11111101";
23      W9 <= "00000000";
24      W10 <= "00000000";
25
26      XW0 <= X0 * W0 ;
27      XW1 <= X1 * W1 ;
28      XW2 <= X2 * W2 ;
29      XW3 <= X3 * W3 ;
30      XW4 <= X4 * W4 ;
31      XW5 <= X5 * W5 ;
32      XW6 <= X6 * W6 ;
33      XW7 <= X7 * W7 ;
34      XW8 <= X8 * W8 ;
35      XW9 <= X9 * W9 ;
36      XW10 <= X10 * W10 ;
37
38  end Behavior;
39
```

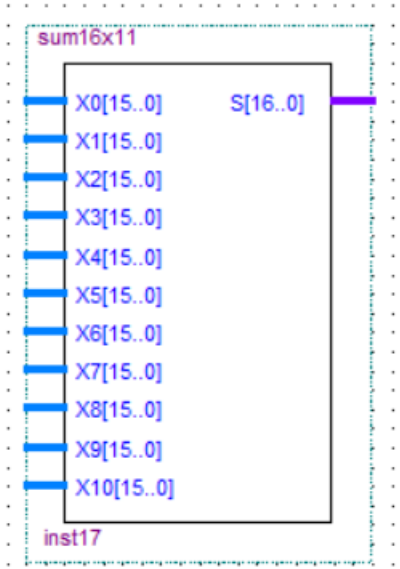
3.reg15x11 (register 16 bits)

ภายในsymbol

```
1   library IEEE;
2   USE IEEE.STD_LOGIC_1164.ALL;
3
4   ENTITY reg15x11 IS
5   PORT ( D0,D1,D2,D3,D4,D5,D6,D7,D8,D9,D10 : IN STD_LOGIC_VECTOR(15 DOWNTO 0) ;
6           Clock : IN STD_LOGIC ;
7           Q0,Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8,Q9,Q10 : OUT STD_LOGIC_VECTOR(15 DOWNTO 0)
8           );
9   END reg15x11 ;
10
11  ARCHITECTURE Behavior OF reg15x11 IS
12  BEGIN
13      PROCESS (Clock)
14      BEGIN
15          IF Clock'EVENT AND Clock= '1' THEN
16          Q0 <= D0 ;
17          Q1 <= D1 ;
18          Q2 <= D2 ;
19          Q3 <= D3 ;
20          Q4 <= D4 ;
21          Q5 <= D5 ;
22          Q6 <= D6 ;
23          Q7 <= D7 ;
24          Q8 <= D8 ;
25          Q9 <= D9 ;
26          Q10 <= D10;
27          END IF ;
28      END PROCESS;
29  END Behavior ;
30
```



4.sum16x11 (วงจรบวก)

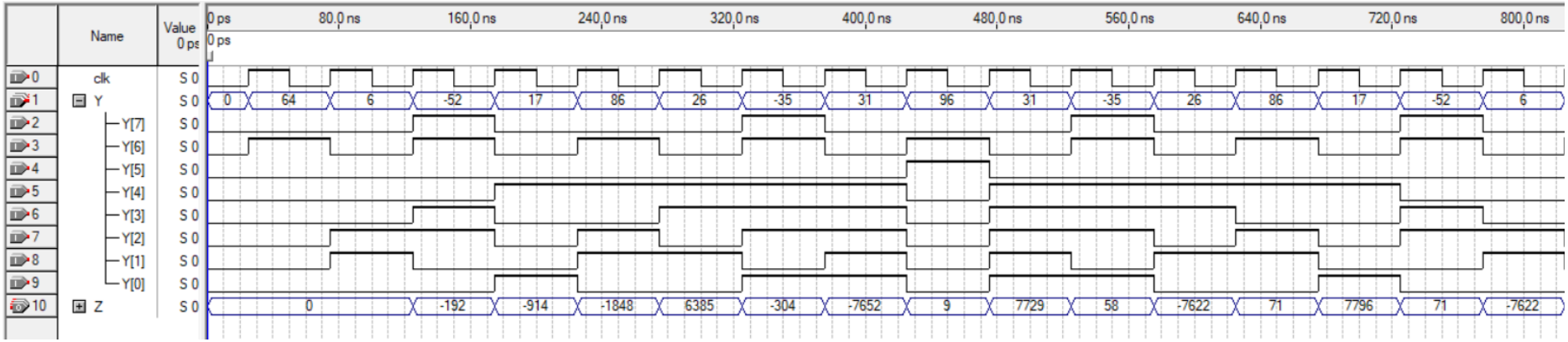


ภายใน symbol

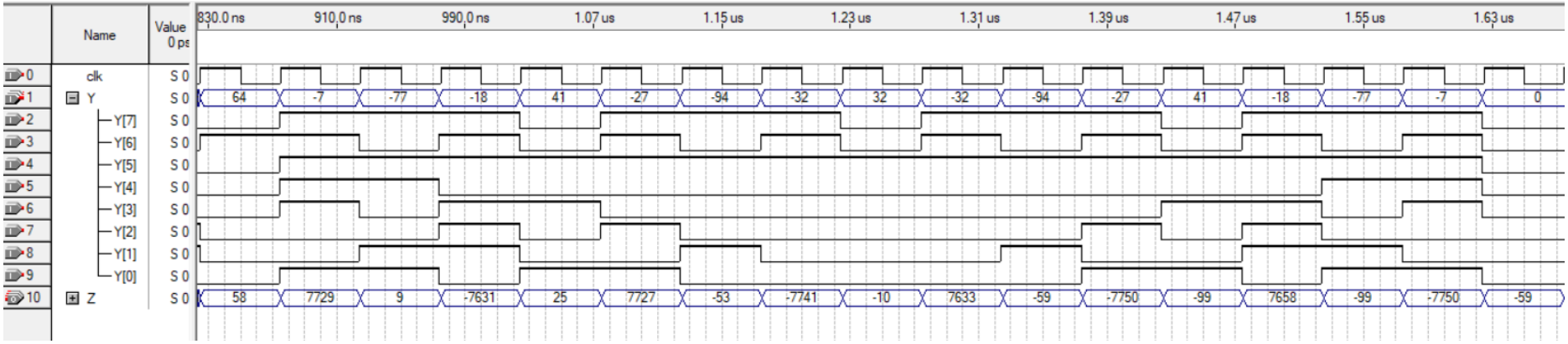
```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3   use IEEE.NUMERIC_STD.ALL;
4
5   ENTITY sum16x11 IS
6   PORT    ( X0,X1,X2,X3,X4,X5,X6,X7,X8,X9,X10 :IN SIGNED (15 DOWNTO 0);
7                                                 S :OUT SIGNED (16 DOWNTO 0)
8           );
9   END sum16x11;
10
11  ARCHITECTURE Behavior OF sum16x11 IS
12  SIGNAL Y0,Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8,Y9,Y10 : SIGNED (16 DOWNTO 0 );
13  begin
14  process (X0,X1,X2,X3,X4,X5,X6,X7,X8,X9)
15  begin
16  if X0(15)= '1' then Y0 <= '1'&X0; else Y0 <= '0'&X0;end if;
17  if X1(15)= '1' then Y1 <= '1'&X1; else Y1 <= '0'&X1;end if;
18  if X2(15)= '1' then Y2 <= '1'&X2; else Y2 <= '0'&X2;end if;
19  if X3(15)= '1' then Y3 <= '1'&X3; else Y3 <= '0'&X3;end if;
20  if X4(15)= '1' then Y4 <= '1'&X4; else Y4 <= '0'&X4;end if;
21  if X5(15)= '1' then Y5 <= '1'&X5; else Y5 <= '0'&X5;end if;
22  if X6(15)= '1' then Y6 <= '1'&X6; else Y6 <= '0'&X6;end if;
23  if X7(15)= '1' then Y7 <= '1'&X7; else Y7 <= '0'&X7;end if;
24  if X8(15)= '1' then Y8 <= '1'&X8; else Y8 <= '0'&X8;end if;
25  if X9(15)= '1' then Y9 <= '1'&X9; else Y9 <= '0'&X9;end if;
26  if X10(15)= '1' then Y10 <= '1'&X9; else Y10 <= '0'&X10;end if;
27  end process;
28  S <= Y0+Y1+Y2+Y3+Y4+Y5+Y6+Y7+Y8+Y9+Y10;
29  end Behavior;
30
```

การแสดงผล Simulation ด้วย Quartus II

Timing Diagram (0-825 ns ; Functional Mode)



Timing Diagram (825-1675 ns ; Functional Mode)

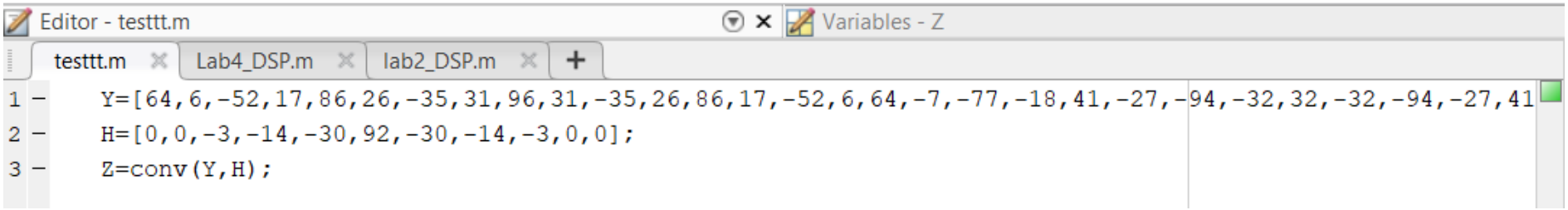


สรุปผลการทดลอง

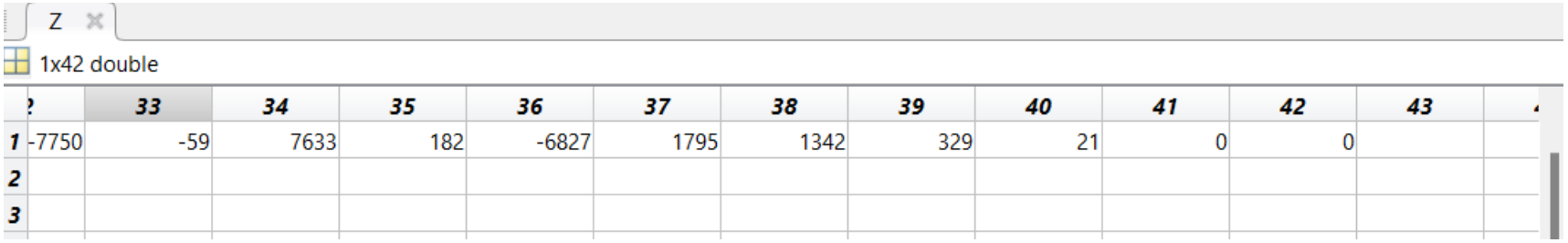
จากการทดลองด้วยโปรแกรมทั้ง2วิธี (Matlab & Quartus II) พบว่า การแสดงผล Output หรือค่า Z ให้ผลเท่ากัน ซึ่งหากเราต้องการค่าจริง เราต้องนำที่ได้จากการแสดงผล ไปหารออกด้วยค่า 2^(-(n+m)) โดยที่ n และ m แทนลำดับของ LSB ที่เรากำหนดตอนทำ Fixed-point เมื่อกระทำการเช่นนั้นแล้วจะได้ค่า Output ที่แท้จริงออกมา

ข้อผิดพลาดของการทดลอง

จากการทดลองพบว่า ค่าOutputที่มีผลลัพธ์ตรงกับการคำนวณด้วยโปรแกรม matlab นั้น มีเพียง 34 ค่า และอีก6ค่าให้ผลลัพธ์ไม่ตรง ทั้งนี้ ขอสันนิษฐานว่า เกิดจาก Absolute Error เนื่องจากmatlab ทำการ Convolution sum ค่า Y และ H จำนวน 201 x 11 ค่า แต่ใน Quartus II เราทำการเลือกค่าเพียงแค่ 32 ค่าตัวอย่างมาทำการทดลอง ทั้งนี้เราได้ทำการตรวจสอบด้วยการนำค่า Y จากผลที่ได้จาก intY มา 32 ค่า และค่า H จาก intH 11ค่า มาทำการ Convolution sum โดยเฉพาะ ดังนั้น

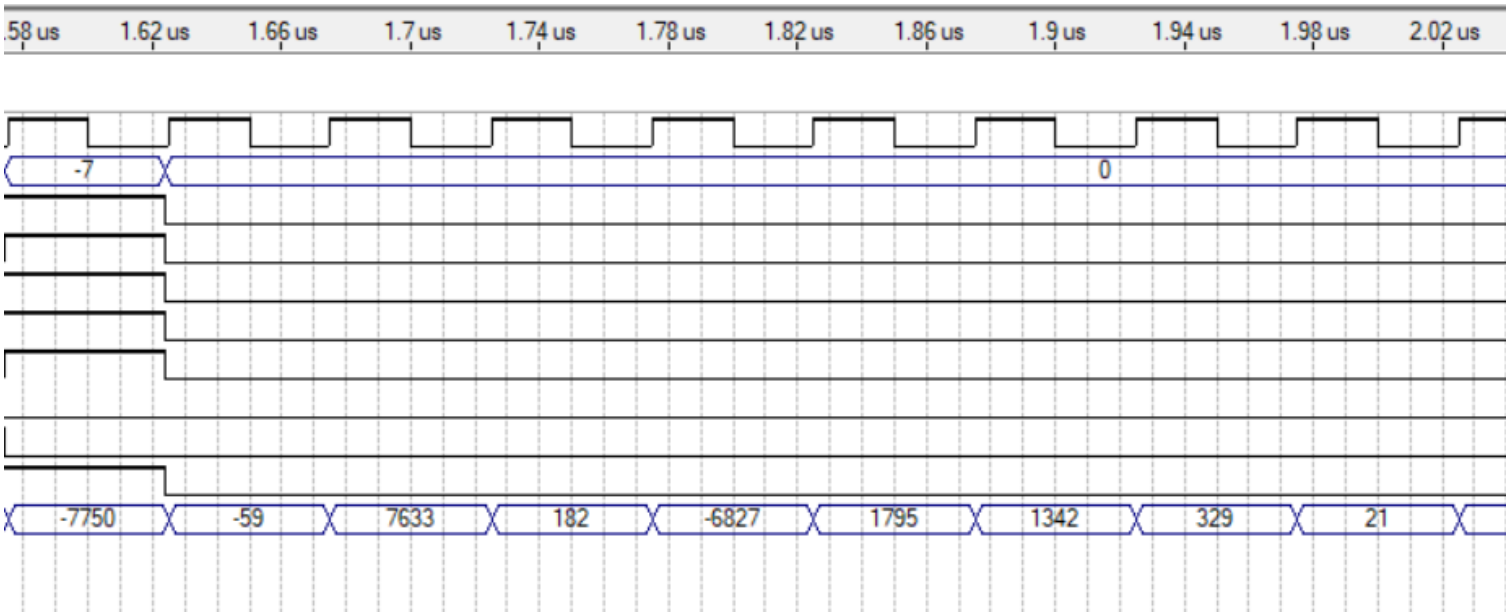


ผลที่ได้



	33	34	35	36	37	38	39	40	41	42	43
1	-7750	-59	7633	182	-6827	1795	1342	329	21	0	0
2											
3											

ซึ่งเมื่อนำมาเทียบกับ Timing Diagram ของ Quartus II



เราจะสังเกตเห็นว่า ตั้งแต่ตำแหน่งที่35เป็นต้นไป ผลลัพธ์ที่ได้นั้น ได้ค่าเหมือนกัน ทั้งนี้เราสรุปได้ว่า การยกตัวอย่างสุ่มมาเพื่อทำการconvolution sum ในจำนวนที่จำกัด จะเกิดค่า Error ของOutput บางส่วน ซึ่งเกิดจาก Absolute Error นั่นเอง