UTM
UNIVERSITI TEKNOLOGI MALAYSIA

Faculty of Electrical Engineering

CAD with HDL

(SKEL 4273)

Section 1

"7-Tap FIR Filter"

Lecturer's Name:

Mdm Norlina Binti Paraman

Name	Matric No.	IC No.
Hong Sing Min	A12KE0149	920707-13-5390
Low Chea Seng	A12KE0162	921114-01-5099

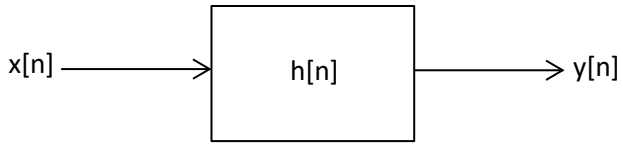
Table of Content

OBJECTIVE

1. To design a 7-tap FIR filter using RTL Design Methodology.

INTRODUCTION

Finite impulse response (FIR) filters are digital filters with finite impulse response. They are also known as non-recursive digital filters as they do not have feedback.



FIR Filter

For a causal discrete-time FIR filter of order *N*, each value of the output sequence is a weighted sum of the most recent input values:

$$h[n] = \sum_{i=0}^{N} b_i \cdot \delta[n-i]$$

$$= \begin{cases} b_n & 0 \le n \le N \\ 0 & otherwise \end{cases}$$

$$\begin{aligned} y[n] &= h[n] * x[n] \\ &= b_0 x[n] + b_1 x[n-1] + \cdots + b_N x[n-N] \\ &= \sum_{i=0}^{N} b_i \cdot x[n-i] \end{aligned}$$

where:

- $x[n]$ is the input signal,
- $h[n]$ is the impulse response,
- $y[n]$ is the output signal,
- N is the filter order; an Nth-order filter has $(N+1)$ terms on the right-hand side
- b_i is the value of the impulse response at the i'th instant for $0 \le i \le N$ of an Nth-order FIR filter. If the filter is a direct form FIR filter then b_i is also a coefficient of the filter.

This computation is also known as discrete convolution.

The $x[n-i]$ in these terms are commonly referred to as *taps*, based on the structure of a tapped delay line that in many implementations or block diagrams provides the delayed inputs to the multiplication operations.

For our project, we are required to design a 7-tap FIR filter, which is of 6th order.
The impulse response of the filter as defined is nonzero over a finite duration.

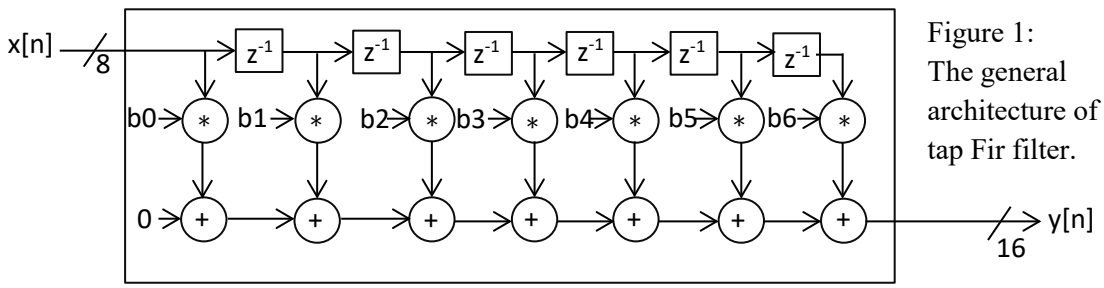
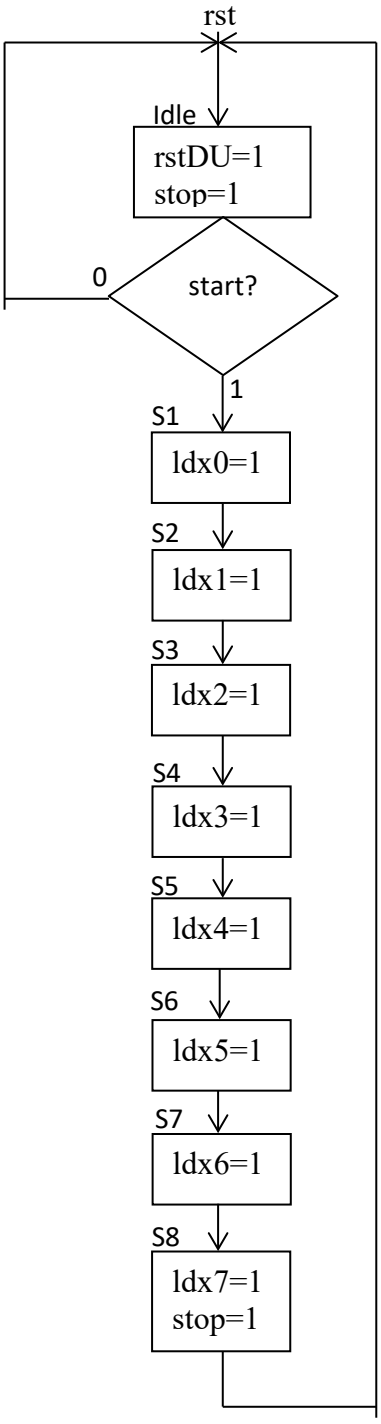


Figure 1: The general architecture of 7-tap Fir filter.

I. ASM Chart

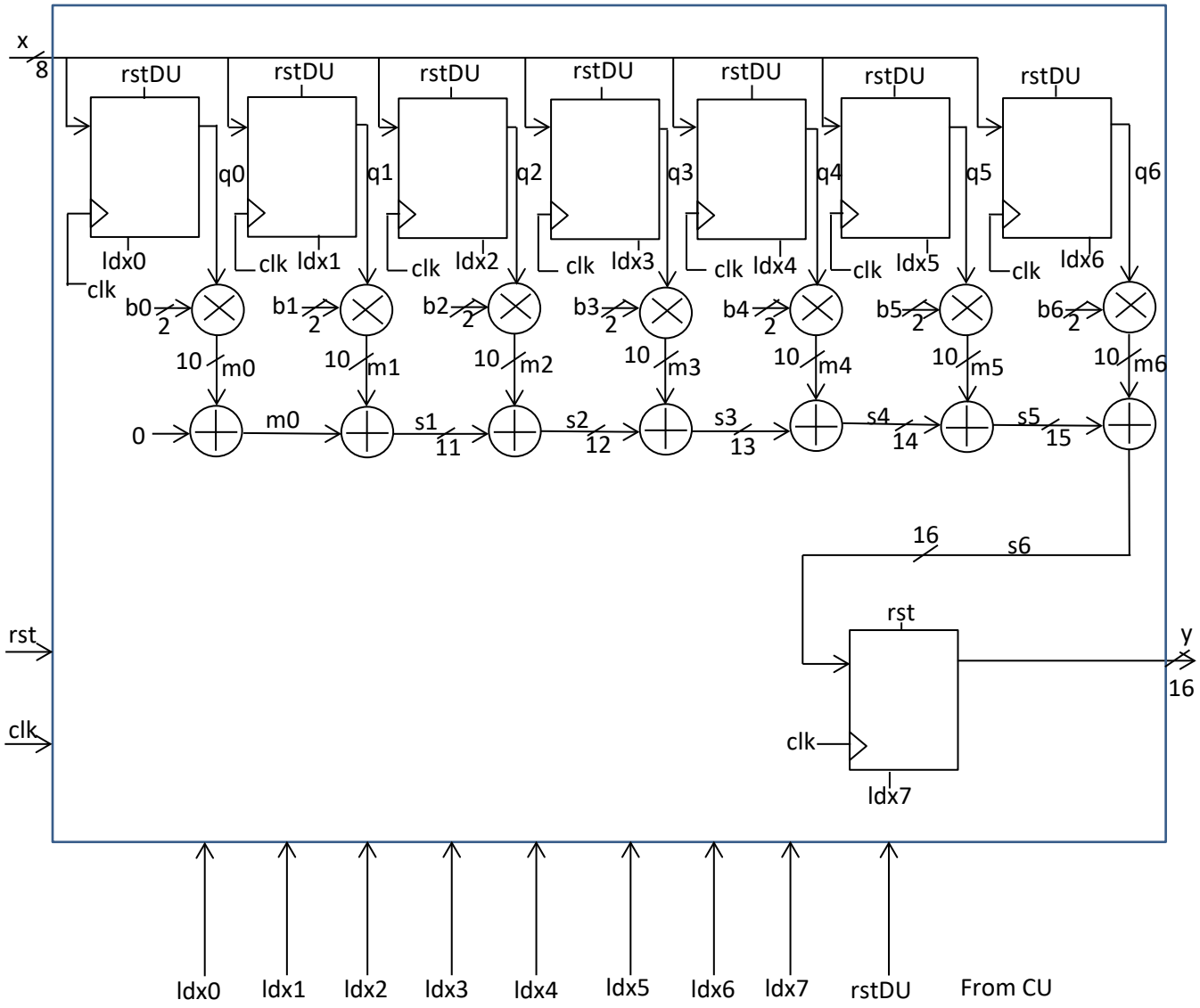

rst
Idle
rstDU=1
stop=1
start?
0
1
S1
ldx0=1
S2
ldx1=1
S3
ldx2=1
S4
ldx3=1
S5
ldx4=1
S6
ldx5=1
S7
ldx6=1
S8
ldx7=1
stop=1

II. RTL Code

Idle:	q ← 0; stop = 0; (start)/ go to S1; (!start)/ go to Idle;
S1:	q0←x0;
S2:	q1 ←x1;
S3:	q2 ←x2;
S4:	q3 ←x3;
S5:	q4 ←x4;
S6:	q5 ←x5;
S7:	q6 ←x6;
S8:	q7 ←x7; stop = 1;

III. Datapath Unit (DU) Design

a. Functional Block Diagram



b. Verilog HDL Coding

- DU Coding

```
module DU (clk,rst,rstDU,x,y,ldx0,ldx1,ldx2,ldx3,ldx4,ldx5,ldx6,ldx7);
        input clk,rst,rstDU,ldx0,ldx1,ldx2,ldx3,ldx4,ldx5,ldx6,ldx7;
        input [7:0] x;
        output [15:0] y;
        parameter [1:0] b0=2,b1=1,b2=3,b3=1,b4=2,b5=3,b6=1;  //Comment1
        wire [9:0] m0,m1,m2,m3,m4,m5,m6;
        wire [10:0] s1; wire [11:0] s2; wire [12:0] s3; wire [13:0] s4; wire [14:0] s5;
        wire [15:0] s6;
        wire [7:0] q0,q1,q2,q3,q4,q5,q6;

        reg8 regx0(x,ldx0,~clk,0,rstDU,q0);  //Comment2
        assign m0=b0*q0;

        reg8 regx1(x,ldx1,~clk,0,rstDU,q1);
        assign m1=b1*q1;
        RCA16 RCAx1(0,m0,m1,s1[10],s1[9:0]);

        reg8 regx2(x,ldx2,~clk,0,rstDU,q2);
        assign m2=b2*q2;
        RCA16 RCAx2(s1[10],s1[9:0],m2,s2[11],s2[10:0]);

        reg8 regx3(x,ldx3,~clk,0,rstDU,q3);
        assign m3=b3*q3;
        RCA16 RCAx3(s2[11],s2[10:0],m3,s3[12],s3[11:0]);

        reg8 regx4(x,ldx4,~clk,0,rstDU,q4);
        assign m4=b4*q4;
        RCA16 RCAx4(s3[12],s3[11:0],m4,s4[13],s4[12:0]);

        reg8 regx5(x,ldx5,~clk,0,rstDU,q5);
        assign m5=b5*q5;
        RCA16 RCAx5(s4[13],s4[12:0],m5,s5[14],s5[13:0]);

        reg8 regx6(x,ldx6,~clk,0,rstDU,q6);
        assign m6=b6*q6;
        RCA16 RCAx6(s5[14],s5[13:0],m6,s6[15],s6[14:0]);

        reg16 regx7(s6,ldx7,~clk,0,rst,y);
endmodule
```

Discussion

Comment1:
The coefficient b is set to have 2-bit value which can take decimals of 0 to 3 in order to fulfill the requirement to get 16 bits of y value. In this case, the coefficients $b[n] = \{2\ 1\ 3\ 1\ 2\ 3\ 1\}$.

Comment2:
The clock for Datapath unit is set to be negative-edged to satisfy the setup time and the hold time.

- 8-bit Register Coding

```
module reg8 (d,en,clk,pst,rst,q);
    input [7:0] d;
    input en,clk,pst,rst;
    output [7:0] q;
    reg [7:0] q;

    always@(posedge clk or posedge pst or posedge rst)
            if (rst) q<=0;
            else if (pst) q<=1;
            else if (en) q<=d;
            else q<=q;
endmodule
```

- 16-bit Ripple Carry Adder Coding

```
    module RCA16(Cin,x,y,Cout,Sum);
            input Cin;
            input [15:0] x,y;
            output [15:0] Sum;
            output Cout;
            wire [16:0] c;

            assign c[0]=Cin;
            FA u0(c[0],x[0],y[0],Sum[0],c[1]);
            FA u1(c[1],x[1],y[1],Sum[1],c[2]);
            FA u2(c[2],x[2],y[2],Sum[2],c[3]);
            FA u3(c[3],x[3],y[3],Sum[3],c[4]);
            FA u4(c[4],x[4],y[4],Sum[4],c[5]);
            FA u5(c[5],x[5],y[5],Sum[5],c[6]);
            FA u6(c[6],x[6],y[6],Sum[6],c[7]);
            FA u7(c[7],x[7],y[7],Sum[7],c[8]);
            FA u8(c[8],x[8],y[8],Sum[8],c[9]);
            FA u9(c[9],x[9],y[9],Sum[9],c[10]);
            FA u10(c[10],x[10],y[10],Sum[10],c[11]);
            FA u11(c[11],x[11],y[11],Sum[11],c[12]);
            FA u12(c[12],x[12],y[12],Sum[12],c[13]);
            FA u13(c[13],x[13],y[13],Sum[13],c[14]);
            FA u14(c[14],x[14],y[14],Sum[14],c[15]);
            FA u15(c[15],x[15],y[15],Sum[15],c[16]);
            assign Cout=c[16];

    endmodule
```

- 16-bit Register Coding

```
module reg16 (d,en,clk,pst,rst,q);
    input [15:0] d;
    input en,clk,pst,rst;
    output [15:0] q;
```

```
    reg [15:0] q;

    always@(posedge clk or posedge pst or posedge rst)
        if (rst) q<=0;
        else if (pst) q<=1;
        else if (en) q<=d;
        else q<=q;
endmodule
```

IV. Control Unit (CU) Design

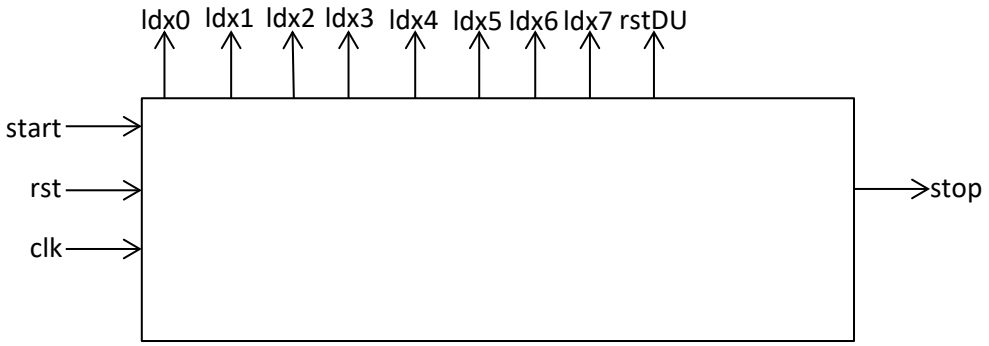
a. RTL-CS Table

State	Control Signal									
	ldx0	ldx1	ldx2	ldx3	ldx4	ldx5	ldx6	ldx7	rstDU	stop

Idle
if (start), goto S1
if (!start), goto Idle
	0	0	0	0	0	0	0	0	1	0

S1 goto S2	1	0	0	0	0	0	0	0	0	0
S2 goto S3	0	1	0	0	0	0	0	0	0	0
S3 goto S4	0	0	1	0	0	0	0	0	0	0
S4 goto S5	0	0	0	1	0	0	0	0	0	0
S5 goto S6	0	0	0	0	1	0	0	0	0	0
S6 goto S7	0	0	0	0	0	1	0	0	0	0
S7 goto S8	0	0	0	0	0	0	1	0	0	0
S8 goto Idle	0	0	0	0	0	0	0	1	0	1

b. Functional Block Diagram

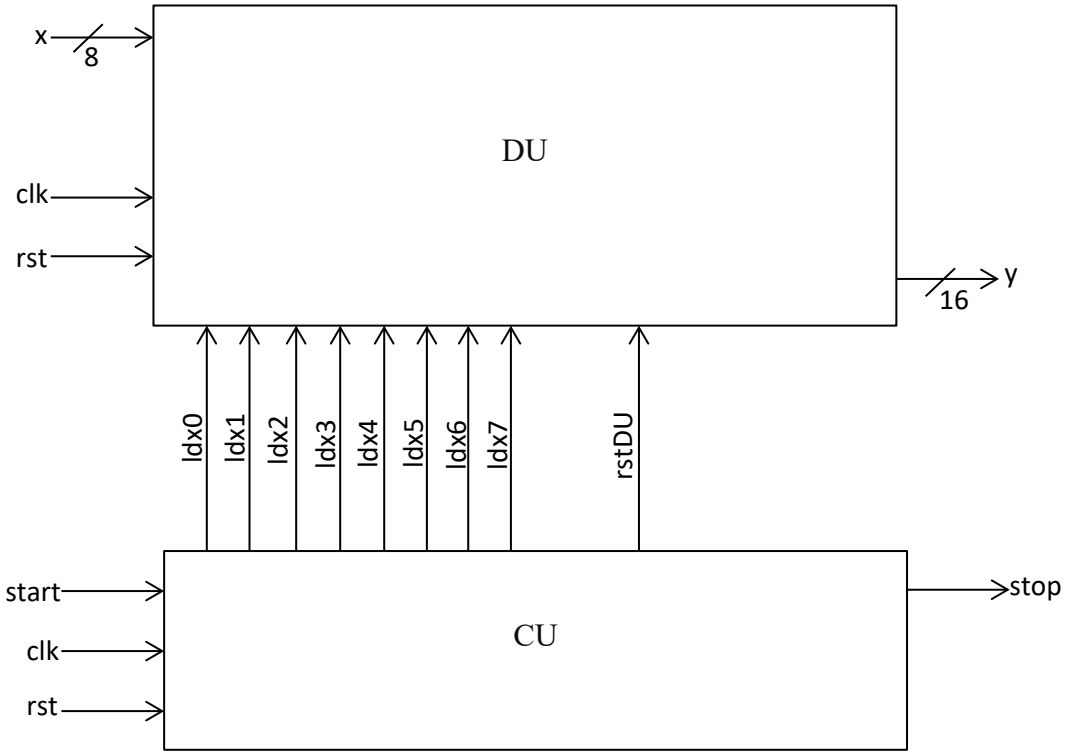


c. Verilog HDL Coding

```
module CU (start,rst,clk,ldx0,ldx1,ldx2,ldx3,ldx4,ldx5,ldx6,ldx7,rstDU,stop);
	input start,rst,clk;
	output ldx0,ldx1,ldx2,ldx3,ldx4,ldx5,ldx6,ldx7,rstDU,stop;
	parameter [3:0] s0=0,s1=1,s2=2,s3=3,s4=4,s5=5,s6=6,s7=7,s8=8;
	reg [3:0] PS,NS;
	reg [9:0] Control;

	//state reg
	always@(posedge clk or posedge rst) begin
		if (rst) PS<=0;
		else PS<=NS; end

	//NS and output logic
	always@(PS or start) begin
		Control=0;
		case (PS)
			s0: begin NS=start? s1:s0; Control=10'b0000000010; end
			s1: begin NS=s2; Control=10'b1000000000; end
			s2: begin NS=s3; Control=10'b0100000000; end
			s3: begin NS=s4; Control=10'b0010000000; end
			s4: begin NS=s5; Control=10'b0001000000; end
			s5: begin NS=s6; Control=10'b0000100000; end
			s6: begin NS=s7; Control=10'b0000010000; end
			s7: begin NS=s8; Control=10'b0000001000; end
			s8: begin NS=s0; Control=10'b0000000101; end
		endcase
	end

	assign {ldx0,ldx1,ldx2,ldx3,ldx4,ldx5,ldx6,ldx7,rstDU,stop}=Control;
endmodule
```

V. CU-DU Integration

a. Top-Level Functional Block Diagram



b. Top-Level Verilog HDL Coding

```
module SevenTapFIRFilter(start,rst,clk,stop,x,y);
	input start,rst,clk;
	input [7:0] x;
	output stop;
	output [15:0] y;
	wire w0,w1,w2,w3,w4,w5,w6,w7;


	CU u0(start,rst,clk,w0,w1,w2,w3,w4,w5,w6,w7,rstDU,stop);
	DU u1(clk,rst,rstDU,x,y,w0,w1,w2,w3,w4,w5,w6,w7);

endmodule
```

i. Design Simulation

a. Datapath Unit (DU)

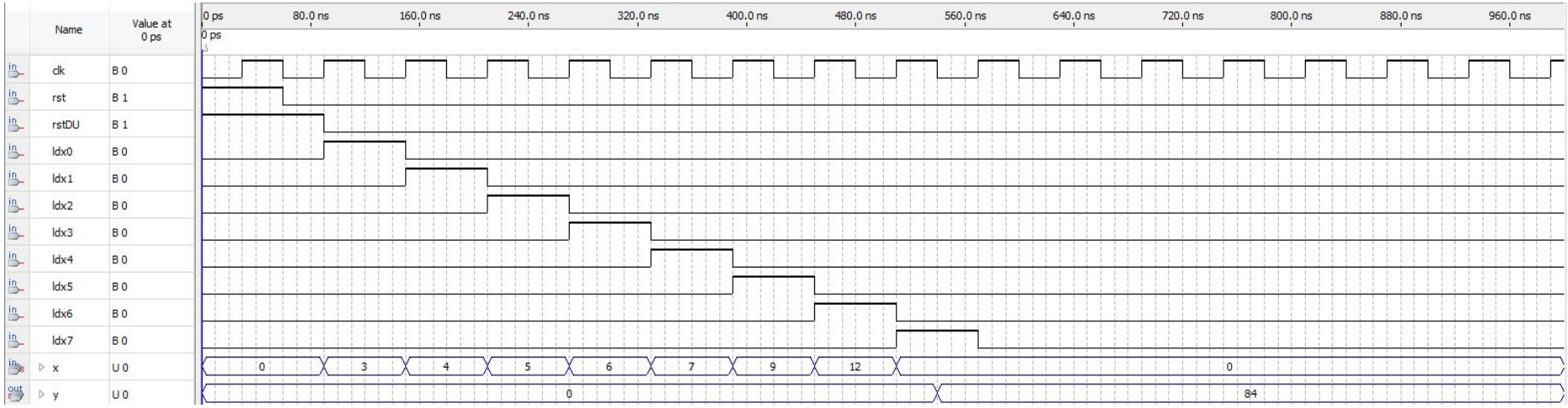


Figure 2: DU Waveform.

Discussion:

The input is set to be $x[n] = \{3\ 4\ 5\ 6\ 7\ 9\ 12\}$. The expected output is $y[n] = 84$. Thus, the waveform matches the output.

b. Control Unit (CU)

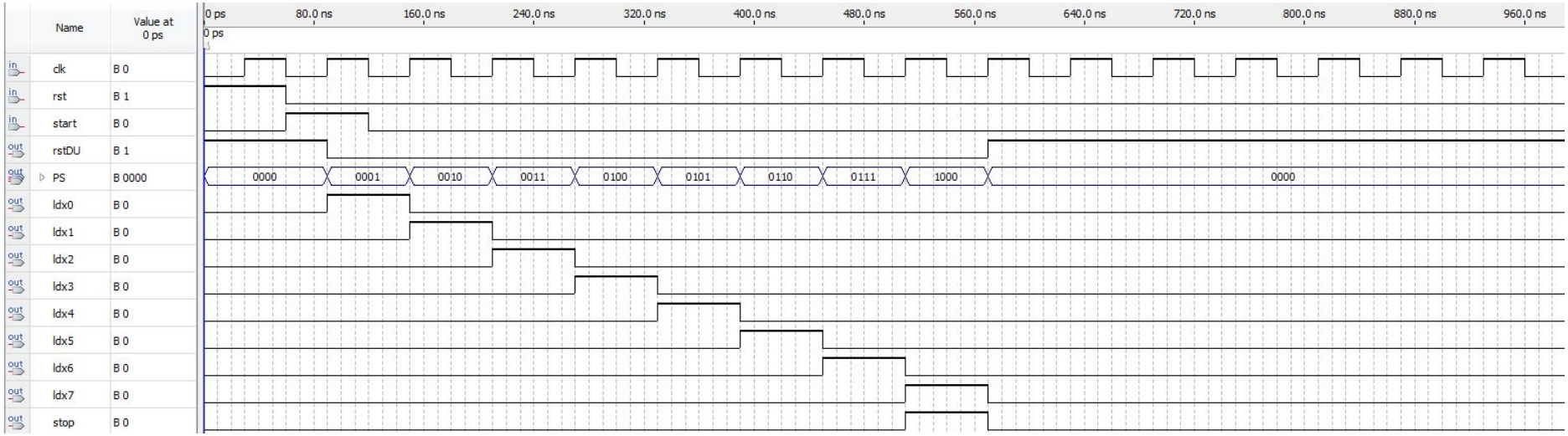


Figure 3: CU Waveform with Present State (PS) shown.

c. CU-DU Integration

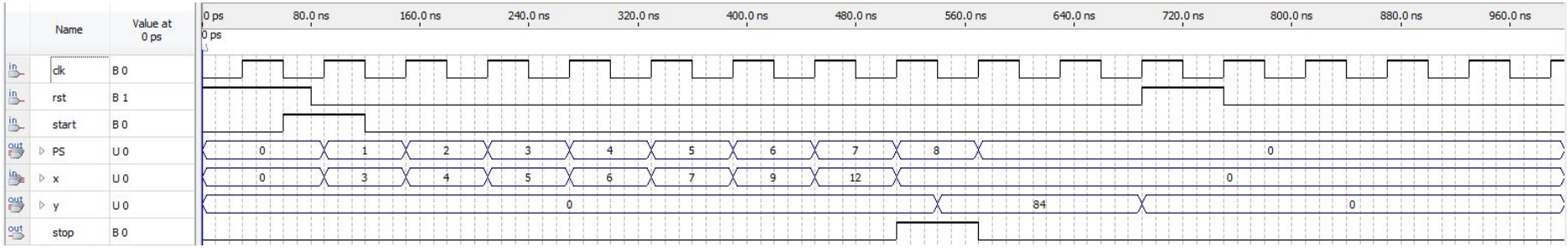


Figure 4: The Top-level Waveform for 7-tap FIR filter.

DISCUSSION

The coefficient we set is $b[n] = \{2\ 1\ 3\ 1\ 2\ 3\ 1\}$. The coefficient b is set to have 2-bit value which can take decimals of 0 to 3 in order to fulfill the requirement to get 16 bits of y value. The input is set to be $x[n] = \{3\ 4\ 5\ 6\ 7\ 9\ 12\}$. The expected output is $y[n] = 84$. Thus, the waveform matches the output.

The clock for Datapath unit is set to be negative-edged to satisfy the setup time and the hold time.

CONCLUSION

The output matches the calculation where the $b[n] = \{2\ 1\ 3\ 1\ 2\ 3\ 1\}$ and $x[n] = \{3\ 4\ 5\ 6\ 7\ 9\ 12\}$ would get $y[n] = 84$. A 7-tap FIR filter is successfully designed.