Neusha Javidnia

810197480

Computer Assignment 6

1398.5.7

A)

A) $\tanh x = x - \frac{x^3}{3} + \frac{2x^5}{15} - \frac{17x^7}{315} + \frac{62x^9}{2835} - \frac{1382x^{11}}{155925} + \dots$

Rom:

$i=0$ $\quad \frac{1}{3} = (0.33333)_{10} = 0.0101010101010101$

$i=1$ $\quad \frac{\frac{2}{15}}{\frac{1}{3}} = \frac{2}{5} = (0.40000)_{10} = 0.0110011001100110011$

$i=2$ $\quad \frac{\frac{17}{315}}{\frac{2}{15}} = \frac{17}{42} = (0.40476)_{10} = 0.01100111100111111$

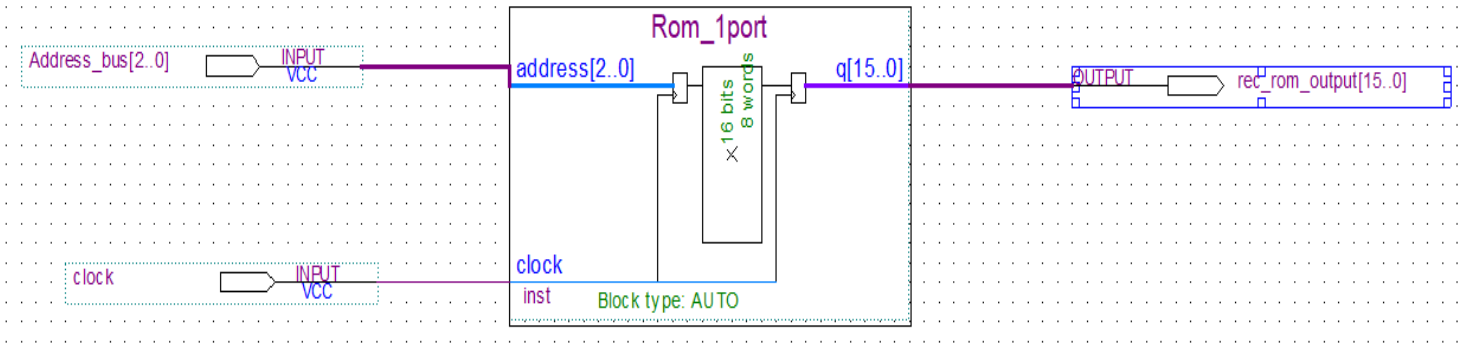
$i=3$ $\quad \frac{\frac{62}{2835}}{\frac{17}{315}} = \frac{62}{153} = (0.40523)_{10} = 0.01100111101111 0$

$i=4$ $\quad \frac{\frac{1382}{155925}}{\frac{62}{153}} = (0.40528)_{10} = 0.0110011111 0000$

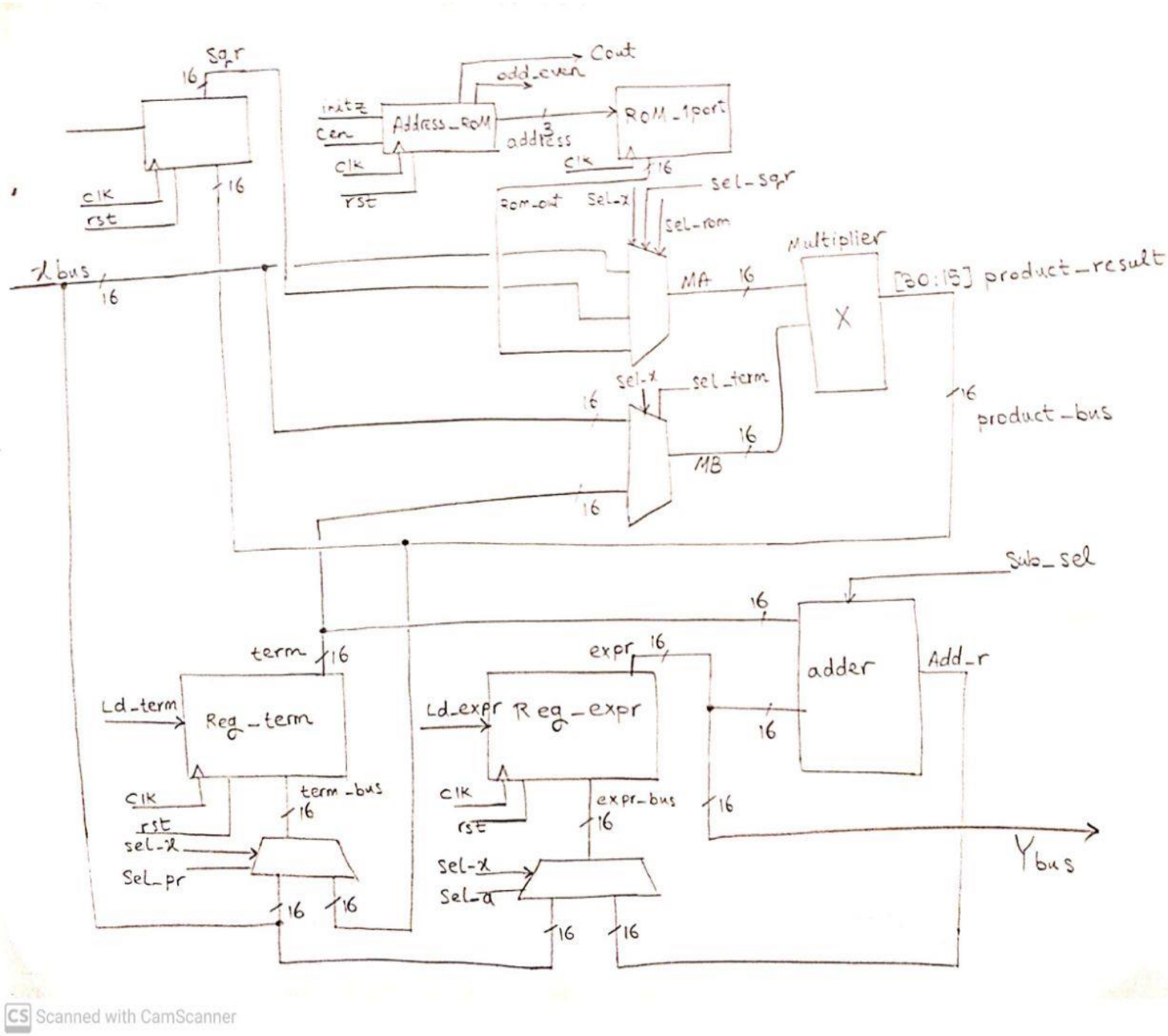
$i=5$ $\quad \frac{929569}{2293620} = (0.40528)_{10} = 0.0110011111 0000$

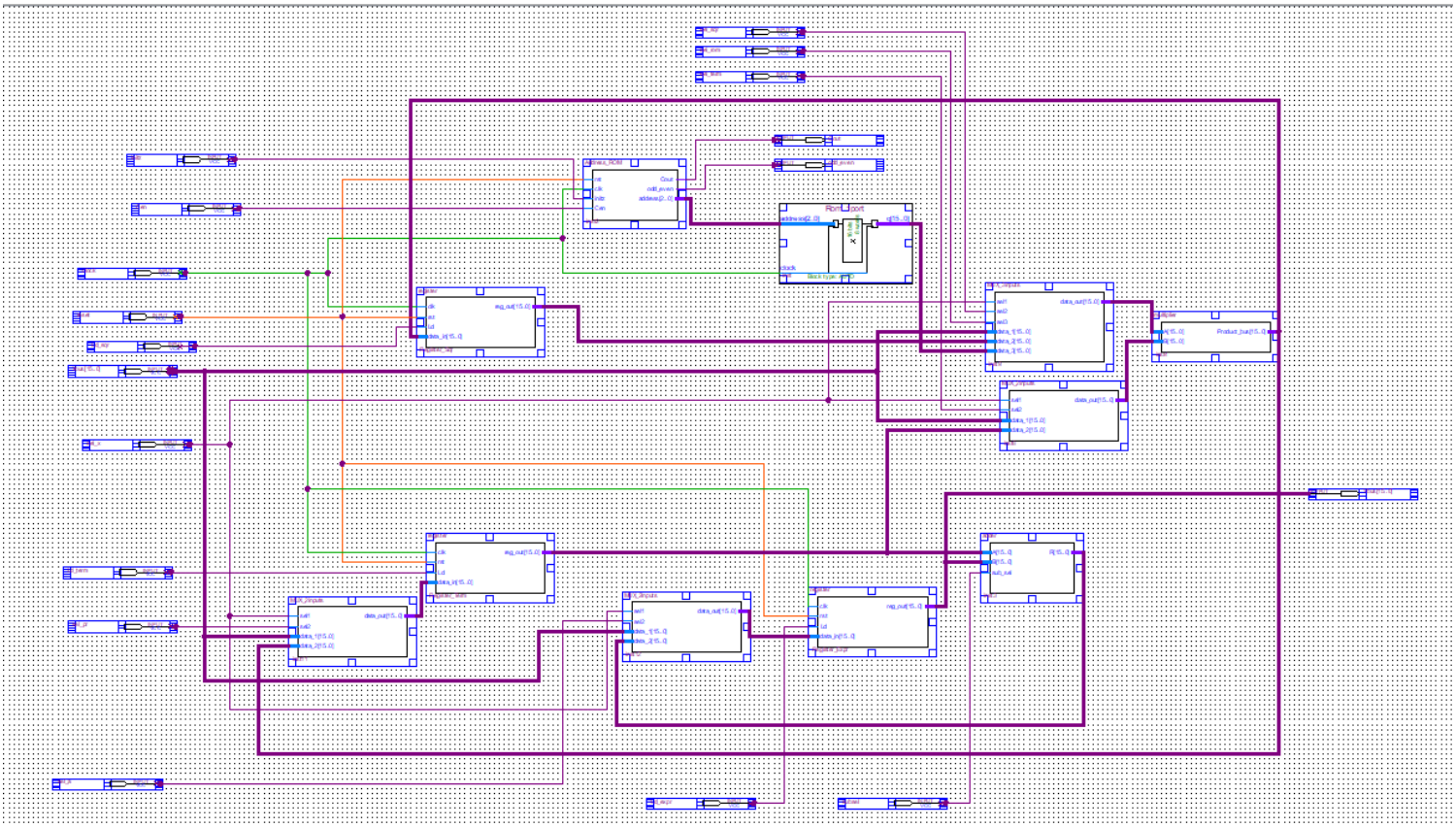
$\frac{6404582}{15802673} = (0.40528)_{10} = 0.0\ 110011111 0000$

Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	0010101010101010	0011001100110011	0011001111001111	0011001111011110	0011001111100000	0011001111100000	0011001111100000	0011001111100000	.....



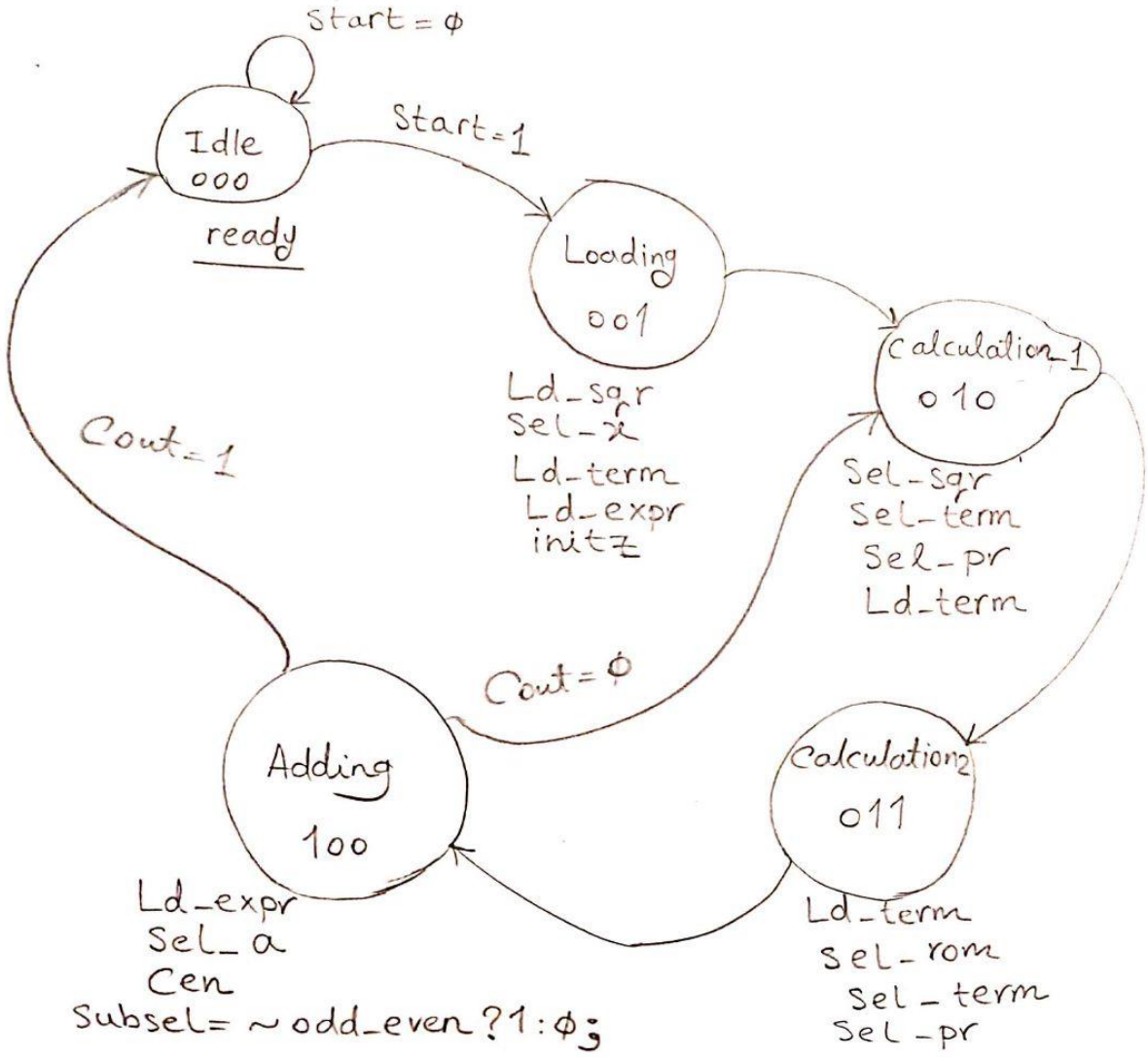
B)





```
module Address_ROM(input rst,clk,initz,Cen,output Cout,odd_even,output reg [2:0]address);
assign Cout=&{address};
assign odd_even=address[0];
always@(posedge clk, posedge rst)
 if(rst) address<=3'b0;
 else if(initz) address<=3'b0;
 else if(Cen) address<=address+1;
 endmodule

 module adder(input [15:0]A,B,input sub_sel,output [15:0] R);
assign R = sub_sel ? A-B : A+B;
endmodule

module multiplier(input [15:0]A,B,output [15:0] Product_bus);
wire [31:0] Product_result;
assign Product_result= A * B;
assign Product_bus=Product_result[30:15];
endmodule

module register(input clk,rst,Ld,input [15:0] data_in,output reg [15:0] reg_out);
always@(posedge clk, posedge rst)
 if(rst) reg_out<=16'b0;
 else if(Ld) reg_out<=data_in;
endmodule
 module MUX_2inputs(input sel1,sel2,input[15:0] data_1,data_2,output [15:0] data_out);
   assign data_out= (sel1==1)? data_1:(sel2==1)? data_2:16'b 0;
 endmodule

 module MUX_3inputs(input sel1,sel2,sel3,input[15:0] data_1,data_2,data_3,output [15:0] data_out);
   assign data_out= (sel1==1)? data_1:(sel2==1)? data_2:(sel3==1)?data_3:16'b 0;
 endmodule
```

C)


C)
Start = ϕ
Idle
000
ready
Start=1
Loading
001
Ld_sqr
Sel_x
Ld_term
Ld_expr
initz
Calculation_1
010
Sel_sqr
Sel_term
Sel_pr
Ld_term
Cout=1
Cout=ϕ
Adding
100
Ld_expr
Sel_a
Cen
Subsel= ~odd_even?1:ϕ;
Calculation2
011
Ld_term
sel_rom
Sel_term
Sel_pr

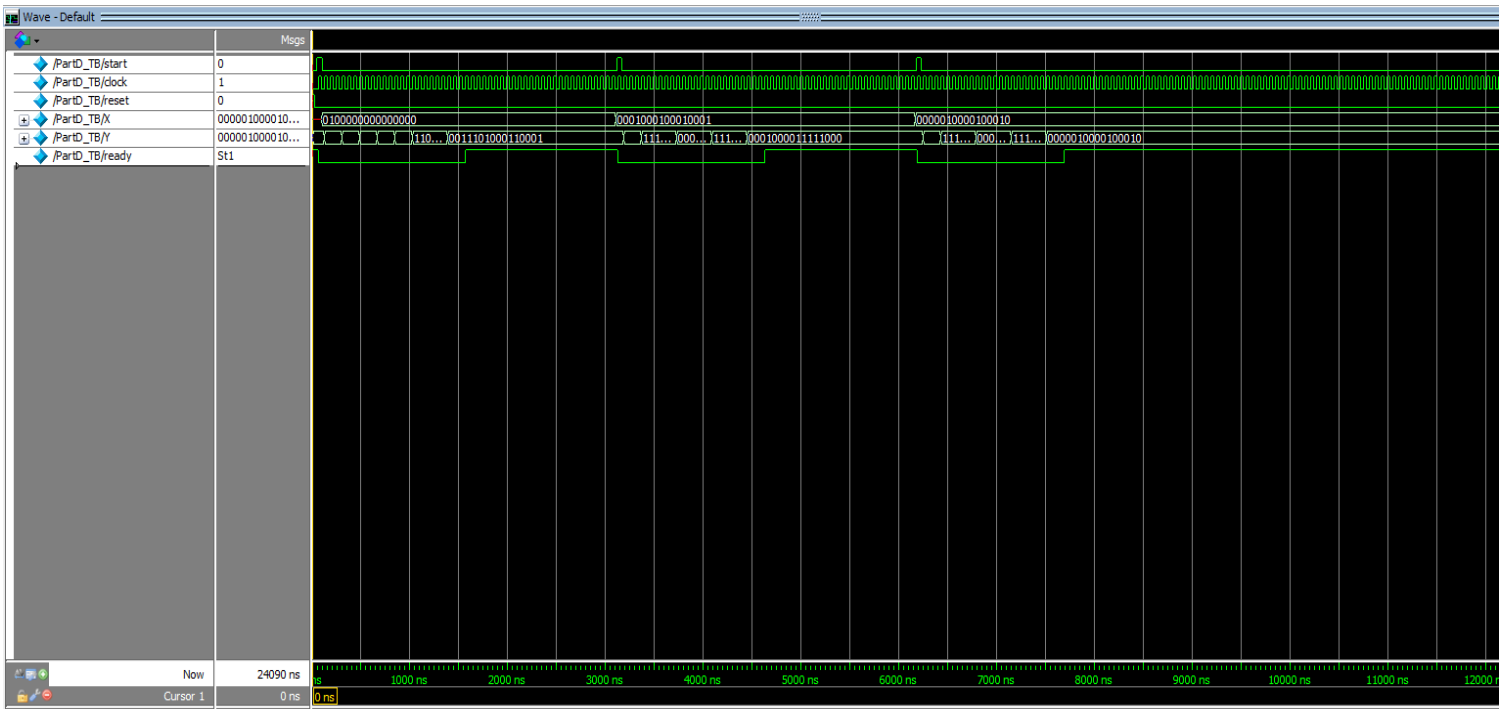
```
Ln#
 2  module controlunit_tanh(input start,clk, rst,odd_even, Cout,output reg initz,
 3  Cen, Ld_term, Ld_expr, Ld_sqr,sel_x, sel_rom,
 4   sel_sqr, sel_term, sel_a, sel_pr,subsel,ready);
 5   reg [2:0] ps,ns;
 6  always@(posedge clk, posedge rst)
 7   if(rst) ps<= 3'b0;
 8   else ps<=ns;
 9  always@(ps,start,Cout)begin
10  ns=3'b0;
11  case(ps)
12    0:ns=(start==1)?3'b001:3'b0;
13    1:ns=3'b010;
14    2:ns=3'b011;
15    3:ns=3'b100;
16    4:ns=(Cout==1)?3'b0:3'b010;
17    default:ns=3'b0;
18  endcase
19  end
20  always@(ps,odd_even)begin
21  {initz,Cen,Ld_term, Ld_expr, Ld_sqr,sel_x, sel_rom,
22   sel_sqr, sel_term, sel_a, sel_pr,subsel,ready}=13'b0;
23   case(ps)
24     0:ready=1;
25     1:{Ld_sqr,sel_x,Ld_term,Ld_expr,initz}=5'b11111;
26     2:{sel_sqr,sel_term,sel_pr,Ld_term}=4'b1111;
27     3:{Ld_term,sel_pr,sel_rom,sel_term}=4'b1111;
28     4:begin {Ld_expr,sel_a,Cen}=3'b111; subsel=(odd_even==1)?1'b0:1'b1;end
29     default:{initz,Cen,Ld_term, Ld_expr, Ld_sqr,sel_x, sel_rom,
30   sel_sqr, sel_term, sel_a, sel_pr,subsel,ready}=13'b0;
31   endcase
32   end
33  endmodule
```

D)

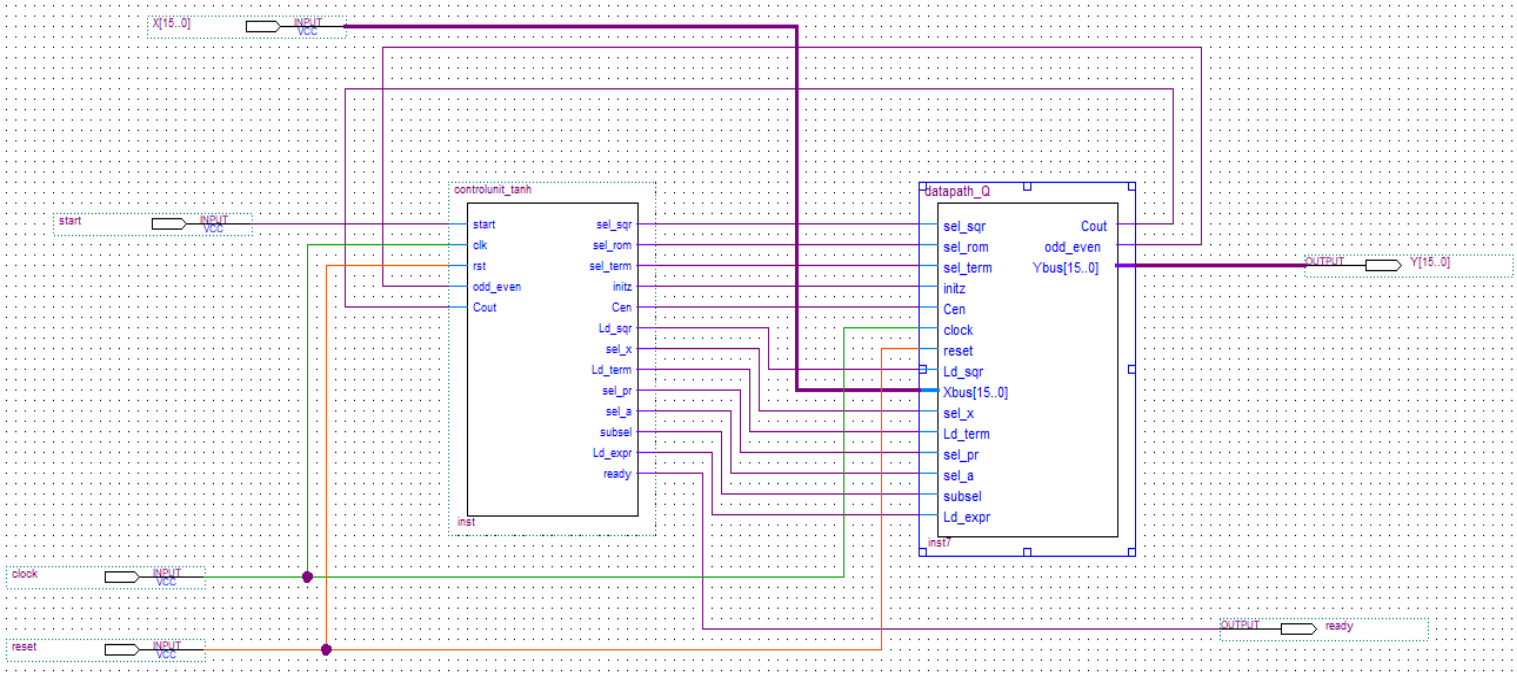
```
`timescale 1ns/1ns
module datapath_tanh(input [15:0] xBus,input clk, rst, initz,
Cen, Ld_term, Ld_expr, Ld_sqr,sel_x, sel_rom, sel_sqr, sel_term, sel_a, sel_pr,
 subsel,output [15:0] yBus,output odd_even, Cout);
reg [2:0] address;
reg [15:0] term,expr,sqr;
wire [15:0] MA,MB,Add_r,Product_bus,expr_bus,term_bus,Rom_out;
wire [31:0] Product_r;
assign Add_r = subsel ? term-expr : term+expr;
assign Product_r= MA * MB;
assign Product_bus=Product_r[30:15];
assign yBus=expr;
assign MA = sel_x ? xBus : sel_rom ? Rom_out:sel_sqr? sqr:16'b 0;
assign MB = sel_x ? xBus : sel_term ? term : 16'b 0;
assign expr_bus = sel_x ? xBus : sel_a ? Add_r : 16'b 0;
assign term_bus = sel_x ? xBus : sel_pr ? Product_bus : 16'b 0;
assign Rom_out= (address==0)? 16'b 0010101010101010:
(address==1)? 16'b 0011001100110011:
(address==2)? 16'b 0011001111001111:
(address==3)? 16'b 0011001111011110:
(address==4)? 16'b 0011001111100000:
(address==5)? 16'b 0011001111100000:
(address==6)? 16'b 0011001111100000:
(address==7)? 16'b 0011001111100000: 16'b 0;
assign Cout=&{address};
assign odd_even=address[0];
always@(posedge clk, posedge rst)
 if(rst) address<=3'b0;
 else if(initz) address<=3'b0;
 else if(Cen) address<=address+1;
always@(posedge clk, posedge rst)
 if(rst) term<=16'b0;
 else if(Ld_term) term<=term_bus;
always@(posedge clk, posedge rst)
 if(rst) expr<=16'b0;
 else if(Ld_expr) expr<=expr_bus;
always@(posedge clk, posedge rst)
 if(rst) sqr<=16'b0;
 else if(Ld_sqr) sqr<=Product_bus;
endmodule
```

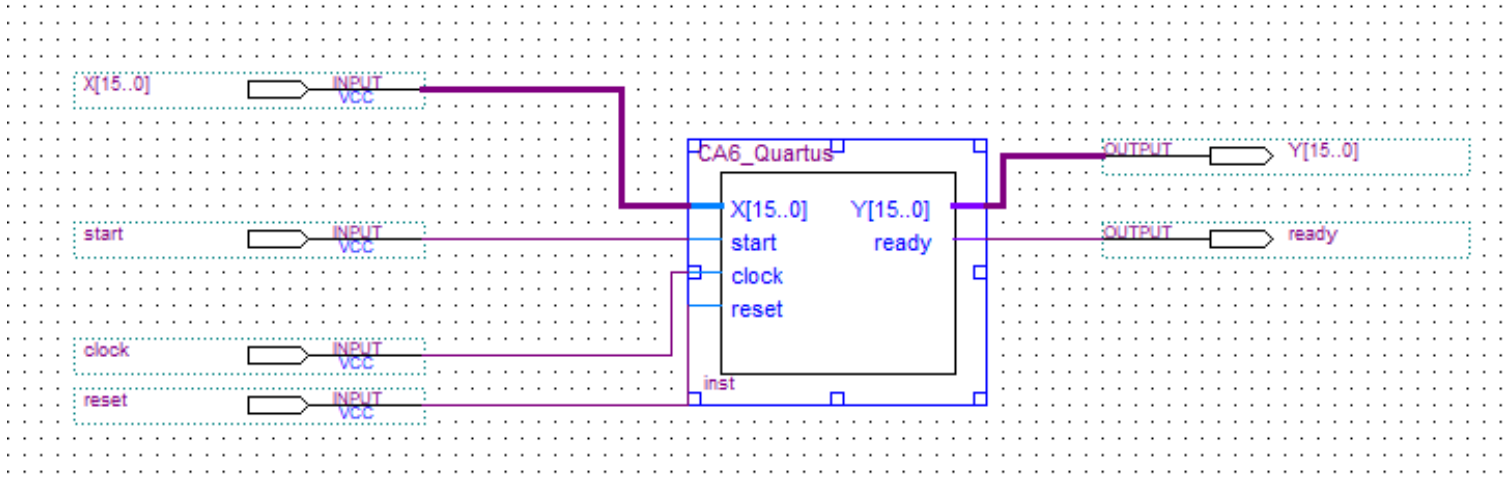
```
`timescale 1ns/1ns
module tanh_modelsim(input [15:0] X,input start,clk,rst,output ready , output [15:0] Y);
wire initz,Cen, Ld_term, Ld_expr, Ld_sqr,sel_x,
sel_rom, sel_sqr, sel_term, sel_a, sel_pr,subsel;
wire odd_even, Cout;
datapath_tanh I1(.xBus(X),.clk(clk),.rst(rst),.initz(initz),
.Cen(Cen),.Ld_term(Ld_term), .Ld_expr(Ld_expr), .Ld_sqr(Ld_sqr),.sel_x(sel_x),
.sel_rom(sel_rom),.sel_sqr(sel_sqr),
.sel_term(sel_term),.sel_a(sel_a),.sel_pr(sel_pr),
.subsel(subsel),.yBus(Y),.odd_even(odd_even),.Cout(Cout));
controlunit_tanh I2(.start(start),.clk(clk),.rst(rst),.odd_even(odd_even),.Cout(Cout),
.initz(initz),.Cen(Cen),.Ld_term(Ld_term), .Ld_expr(Ld_expr),.Ld_sqr(Ld_sqr),.sel_x(sel_x),
.sel_rom(sel_rom),.sel_sqr(sel_sqr),.sel_term(sel_term),.sel_a(sel_a),.sel_pr(sel_pr),
.subsel(subsel),.ready(ready));
endmodule
```

```
Ln#
 1  `timescale 1ns/1ns
 2  module PartD_TB();
 3  reg start=0,clock=0,reset=0;
 4  reg [15:0]X;
 5  wire [15:0]Y;
 6  wire ready;
 7  tanh_modelsim MUT(X,start,clock,reset,ready,Y);
 8  initial begin
 9    #10 reset=1;
10    #20 reset=0;
11    #20 start=1;
12    #50 X=16'b 0100000000000000;
13    #10 start=0;
14    #3000 X=16'b 0001000100010001;
15    #10 start=1;
16    #50 start=0;
17    #3000 X=16'b 0000010000100010;
18    #10 start=1;
19    #50 start=0;
20    #20000 $stop;
21  end
22  initial begin
23    #70 clock=1;
24    repeat(800) #30 clock=~clock;
25    #20 $stop;
26  end
27  endmodule
28
```



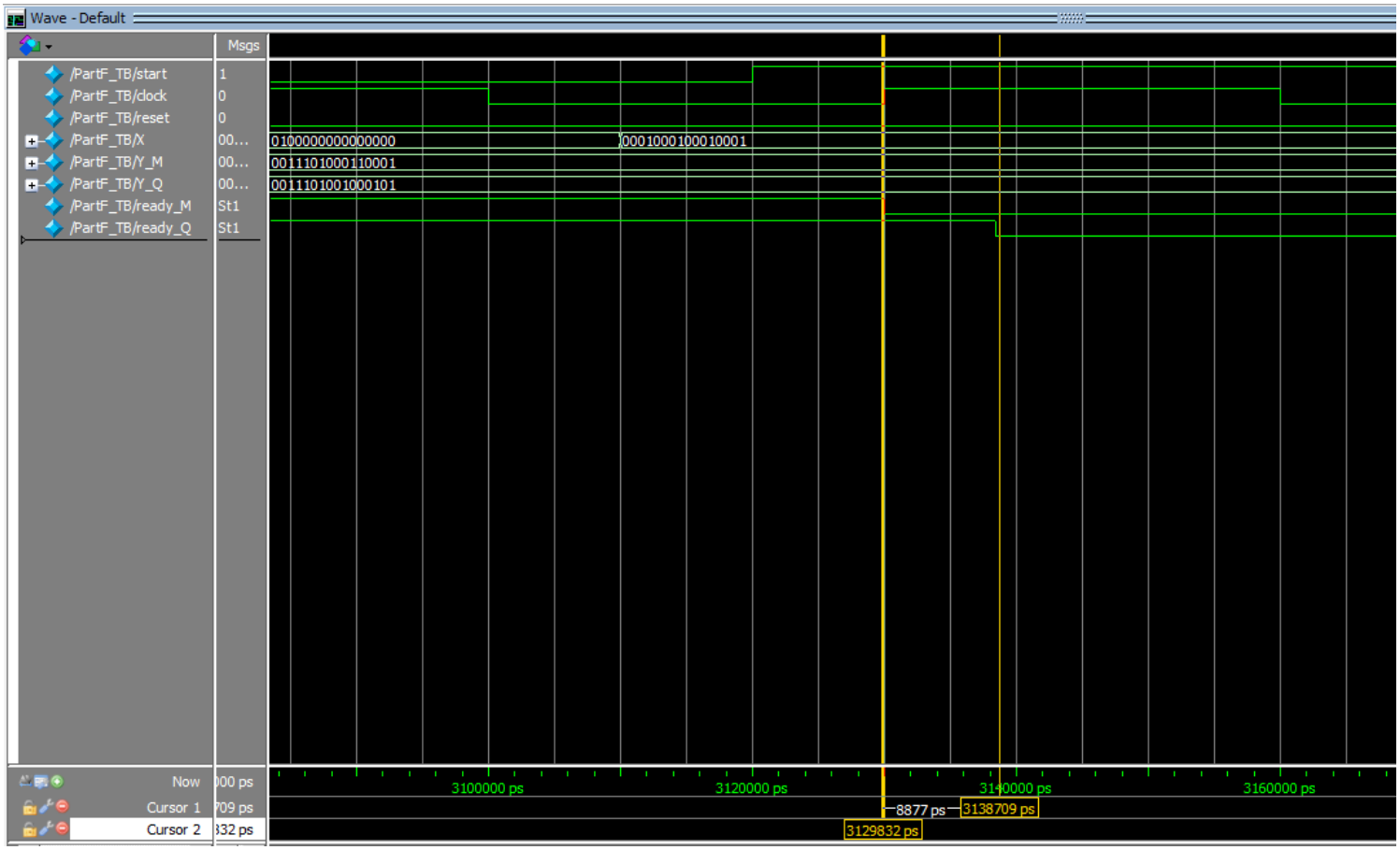
E)

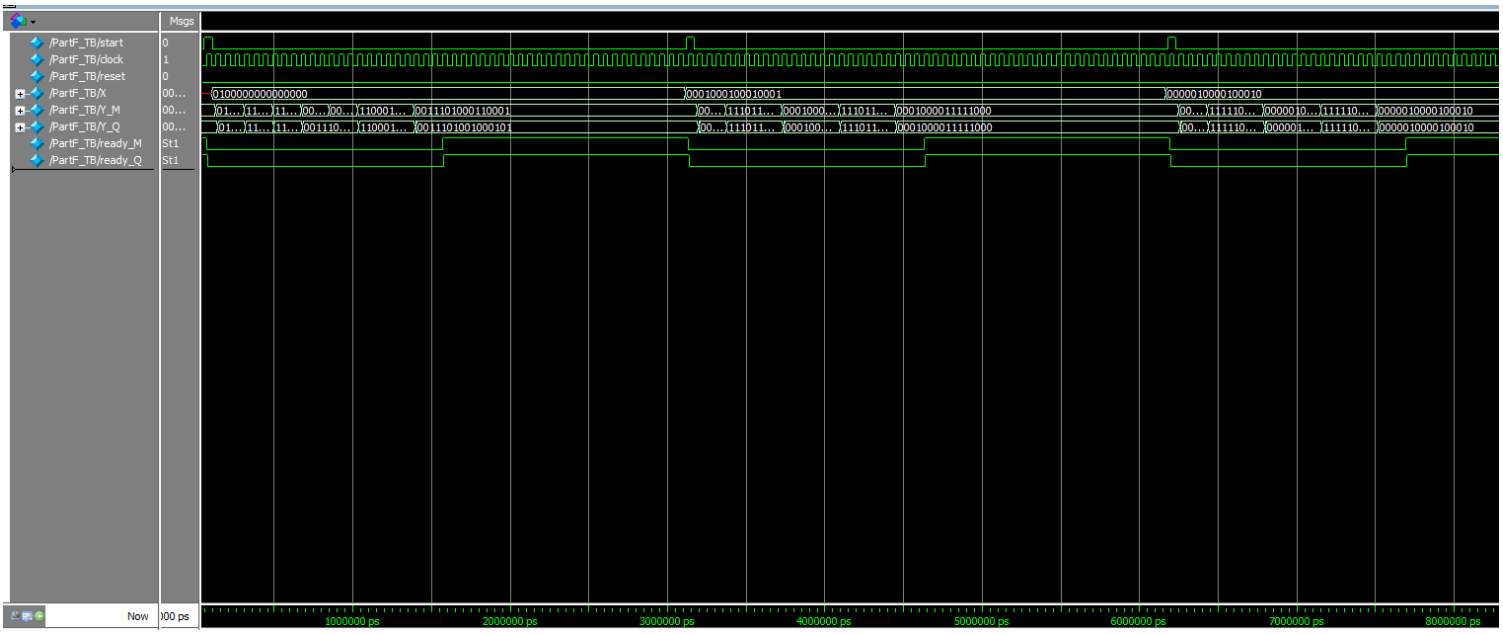




F)

F) As we can see in the waveforms post-synthesis description's outputs (Ybus & ready) takes more time than pre-synthesis to be represented, Because post-synthesis is closer to reality. Quartus add a few delay to post-synthesis circuit. (We have used cyclon IV GX, so it has its own delay)


Wave - Default
Msgs
/PartF_TB/start
/PartF_TB/clock
/PartF_TB/reset
/PartF_TB/X
/PartF_TB/Y_M
/PartF_TB/Y_Q
/PartF_TB/ready_M
/PartF_TB/ready_Q
1
0
0
00...
00...
00...
St1
St1
0100000000000000
0001000100010001
0011101000110001
0011101001000101
Now
Cursor 1
Cursor 2
3100000 ps
3120000 ps
3140000 ps
3160000 ps
8877 ps
3138709 ps
3129832 ps


Msgs
/PartF_TB/start
/PartF_TB/clock
/PartF_TB/reset
/PartF_TB/X
/PartF_TB/Y_M
/PartF_TB/Y_Q
/PartF_TB/ready_M
/PartF_TB/ready_Q
Now
1000000 ps
2000000 ps
3000000 ps
4000000 ps
5000000 ps
6000000 ps
7000000 ps
8000000 ps