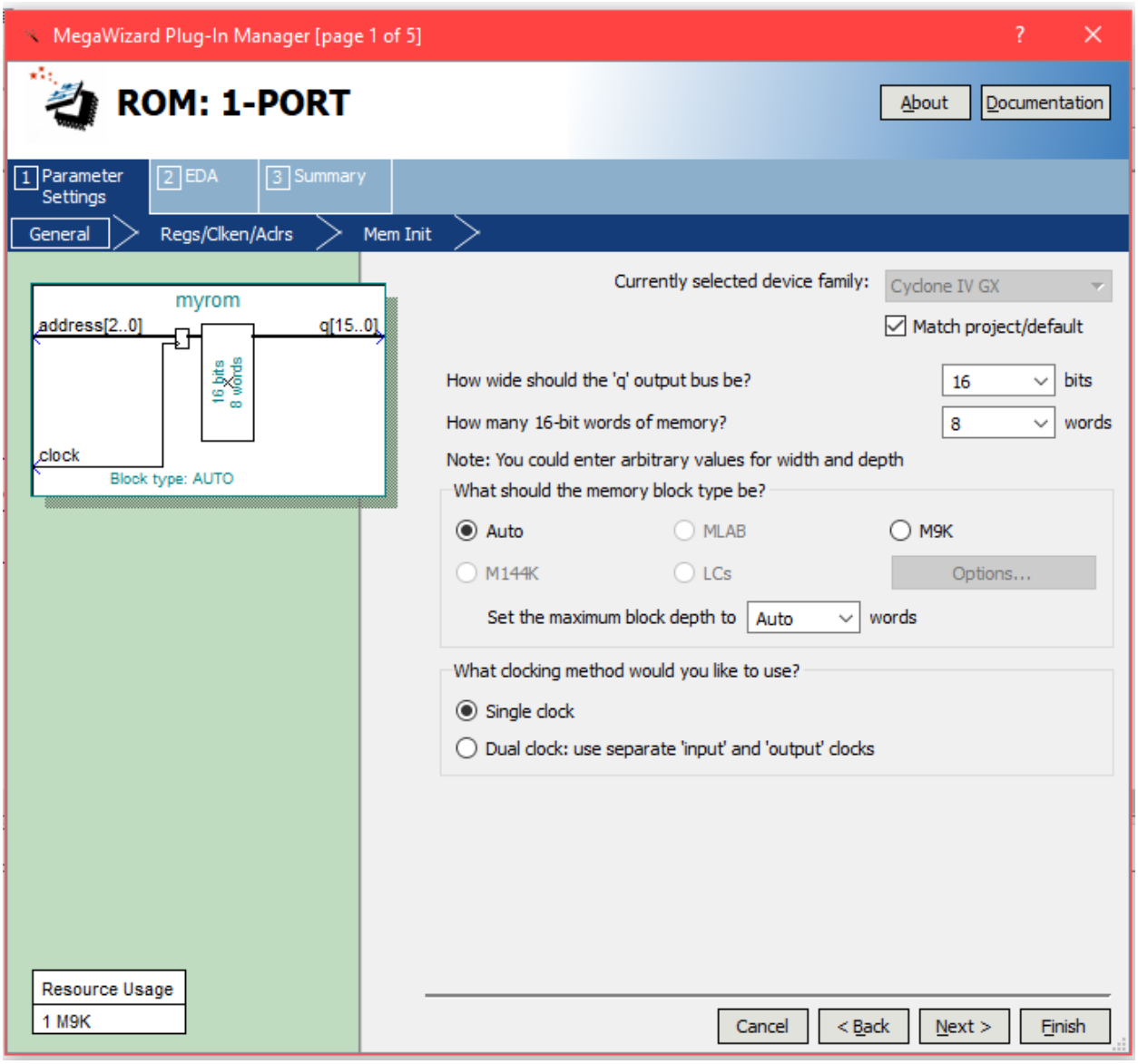
CA2 report

Erfan iravani

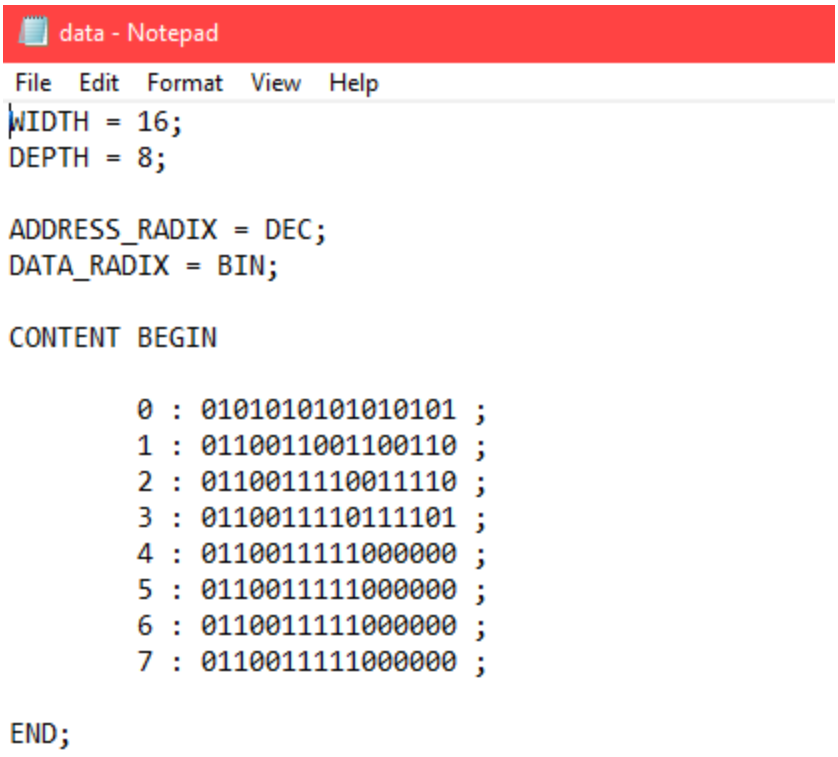
810197462

A

رام را با استفاده از بلاک های موجود در کوارتوس میسازیم و فایل اعداد ذخیره شده در رام را در آن قرار میدهیم

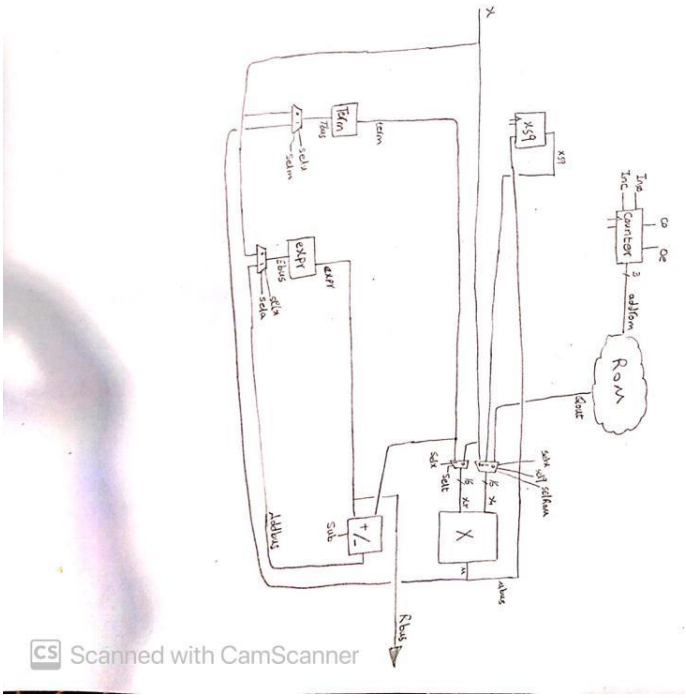


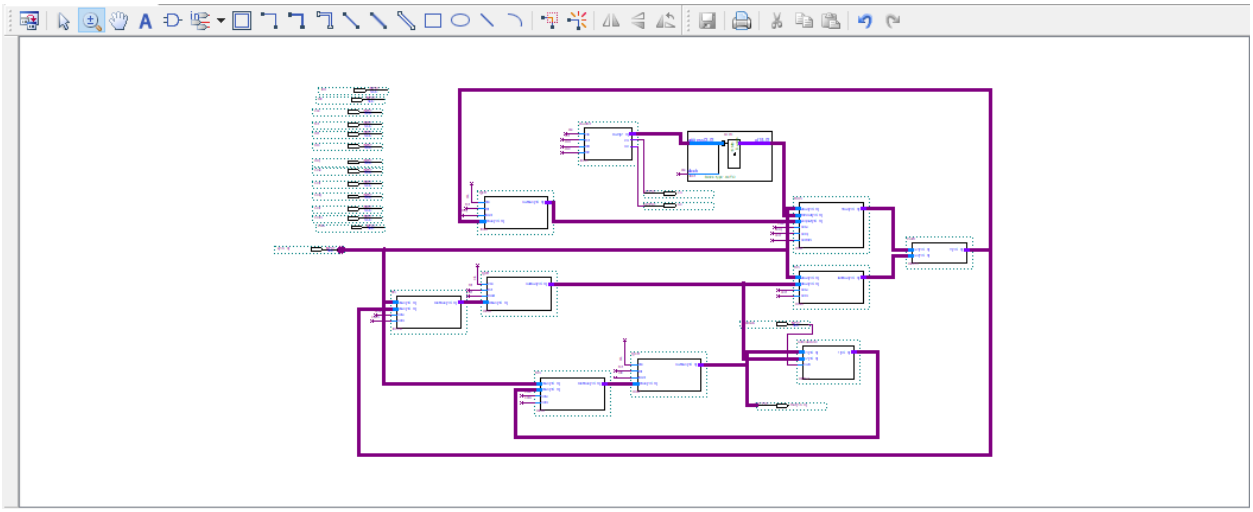
برای به دست آوردن مقادیر مورد نیاز در رام کافی است ضرایب را در هر مرحله به دست آورده و با احتساب 16 بیت برای اعشار مانند زیر بنویسیم


data - Notepad
File Edit Format View Help
WIDTH = 16;
DEPTH = 8;
ADDRESS_RADIX = DEC;
DATA_RADIX = BIN;
CONTENT BEGIN
0 : 0101010101010101 ;
1 : 0110011001100110 ;
2 : 0110011110011110 ;
3 : 0110011110111101 ;
4 : 0110011111000000 ;
5 : 0110011111000000 ;
6 : 0110011111000000 ;
7 : 0110011111000000 ;
END;

B

Erfan Ifavani #CA6 8101977462





```
1  `timescale lns/lns
2  module tim(input [15:0]xbus,obus,input selx,selo,output [15:0]lastbus);
3    assign lastbus=(selx)? xbus:(selo)? obus:lastbus ;
4  endmodule
```

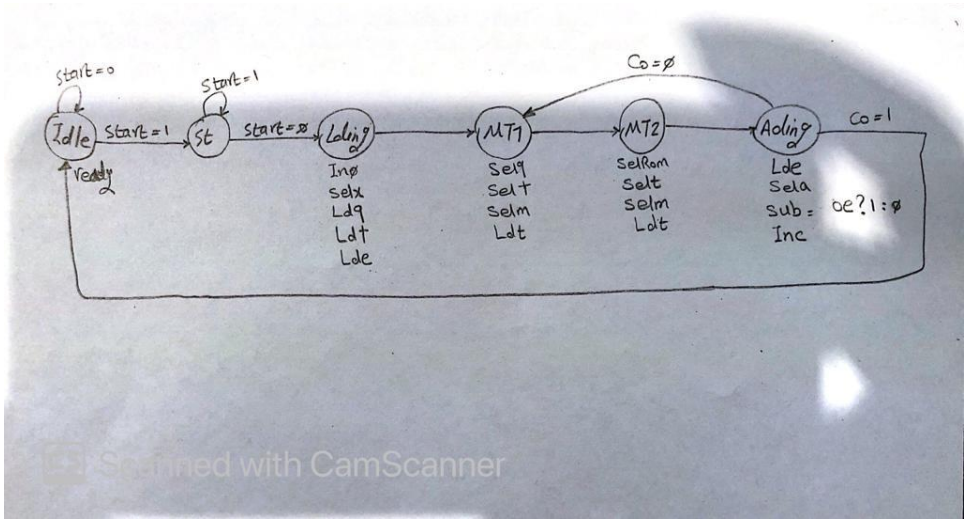
```
1  `timescale lns/lns
2  module thim(input [15:0]xbus,romout,xsqout,input selx,selq,selrom,output [15:0]fbus);
3    assign fbus=(selx)? xbus:(selq)? xsqout: (selrom) ? romout : fbus;
4  endmodule
```

```
1  `timescale lns/lns
2  module addsubcode(input [15:0]x1,x2,input sub,output[15:0]f);
3  assign f = sub ? x1-x2:x1+x2;
4  endmodule
```

```
1  `timescale lns/lns
2  module multi(input [15:0]xx1,xx2,output [15:0]ff);
3  wire [31:0]m;
4  assign m=xx1*xx2;
5  assign ff=m[31:16];
6  endmodule
```

```
1  `timescale 1ns/1ns
2  module rgstr(input clk,rst,load,input [15:0]inbus,output reg [15:0]outbus);
3  always@(posedge clk,posedge rst) begin
4      if(rst) outbus <= 16'b0;
5      else if(load) outbus <= inbus;
6      else outbus <= outbus;
7    end
8   endmodule
```

C

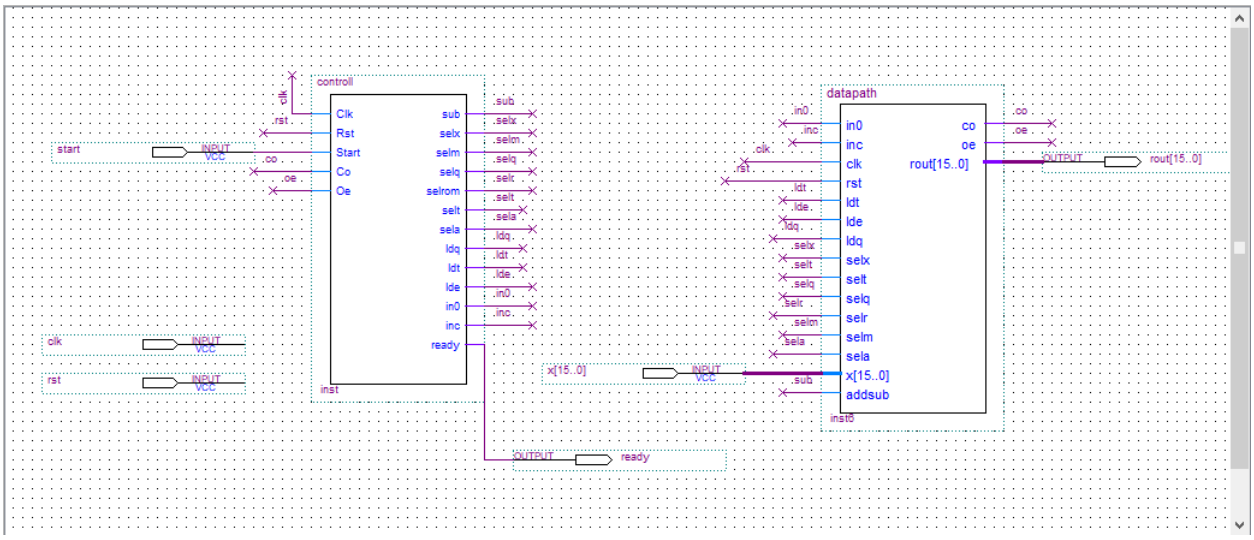



```
1  `timescale 1ns/1ns
2  module controll(input Clk,Rst,Start,Co,Oe,output reg sub,selx,selm,selq,selrom,selt,sela,ldq,ldt,lde,in0,inc,ready);
3    reg [2:0]ps,ns;
4    always @(posedge Clk,posedge Rst)begin
5      if(Rst) ps<=3'b0;
6      else ps<=ns;
7    end
8    always @(Start,Co,ps,Oe)begin
9      ns=3'b0;
10     {inc,in0,ldt,ldq,lde,selm,selx,selq,selrom,selt,sela,ready,sub}=13'b0;
11     case(ps)
12       3'b000:begin
13         ns = Start ? 3'b001:3'b000;
14         ready = 1'b1;
15       end
16       3'b001: begin
17         ns = Start ? 3'b001:3'b010;
18       end
19       3'b010:begin
20         ns = 3'b011;
21         in0 = 1'b1;
22         selx = 1'b1;
23         ldq = 1'b1;
24         lde = 1'b1;
```

```
14          ready = 1'b1;
15        end
16        3'b001: begin
17          ns = Start ? 3'b001:3'b010;
18        end
19        3'b010:begin
20          ns = 3'b011;
21          in0 = 1'b1;
22          selx = 1'b1;
23          ldq = 1'b1;
24          lde = 1'b1;
25          ldt = 1'b1;
26        end
27        3'b011:begin
28          ns = 3'b100;
29          selq = 1'b1;
30          selt = 1'b1;
31          selm = 1'b1;
32          ldt = 1'b1;
33        end
34        3'b100:begin
35          ns = 3'b101;
36          selrom = 1'b1;
37          selt = 1'b1;
38          selm = 1'b1;
39          ldt = 1'b1;
40        end
```

```
26        end
27        3'b011:begin
28          ns = 3'b100;
29          selq = 1'b1;
30          selt = 1'b1;
31          selm = 1'b1;
32          ldt = 1'b1;
33        end
34        3'b100:begin
35          ns = 3'b101;
36          selrom = 1'b1;
37          selt = 1'b1;
38          selm = 1'b1;
39          ldt = 1'b1;
40        end
41        3'b101:begin
42          ns = Co ? 3'b000:3'b011;
43          lde = 1'b1;
44          sela = 1'b1;
45          inc = 1'b1;
46          sub = ~(Oe)?1'b1:1'b0;
47        end
48        default:{inc,in0,ldt,ldq,lde,selm,selx,selq,selrom,selt,sela,ready,sub}=13'b0;
49      endcase
50    end
51  endmodule
52
```

D

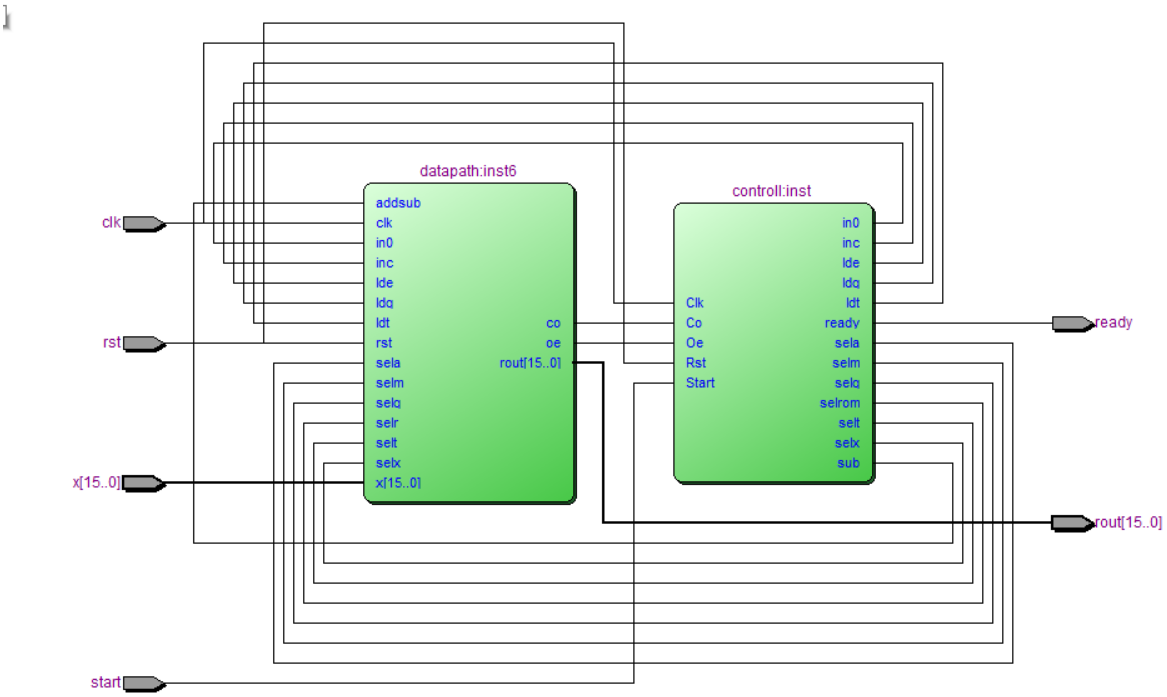


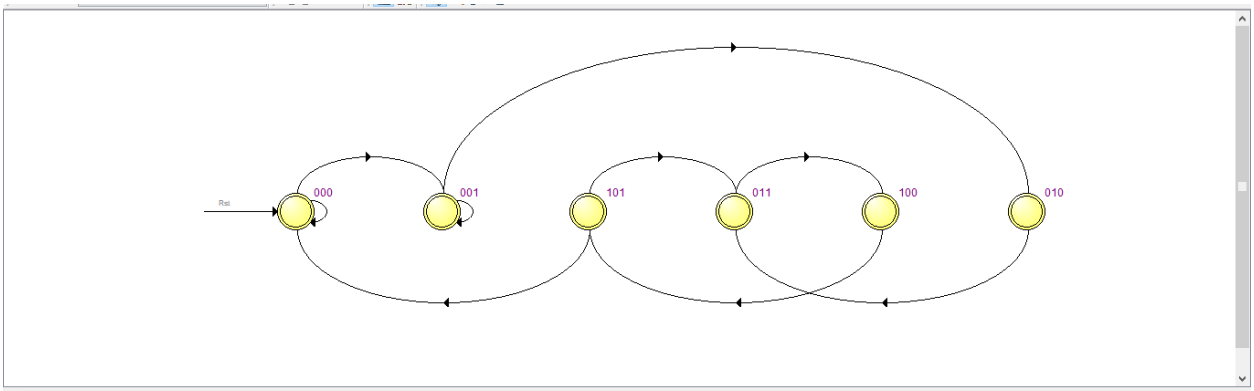
Ln#

```
1  `timescale 1ns/1ns
2  module tnhxtst();
3    reg clk=0;
4    reg rst=1;
5    reg start=0;
6    reg [15:0]Xbus=16'b0101010101010101;
7    //wire [15:0]Rbus;
8    wire [15:0]Rbus,Rbuscode;
9    wire ready;
10   tanhcal ime(ready,clk,rst,start,Xbus,Rbus);
11   tnhx s(clk,rst,start,Xbus,ready,Rbuscode);
12   initial begin
13     #1000;
14     rst=0;
15     #1000;
16     start=1;
17     #2000;
18     start=0;
19   end
20   initial begin
21     #1000;
22     repeat(100)begin
23     clk=~clk;
24     #1000;
25     end
26   end
27   initial begin
28   #200000;
```

```
C:/altera/13.0sp1/tnhxtst.v (/tnhxtst) - Default
Ln#
 5     reg start=0;
 6     reg [15:0]Xbus=16'b1011000010100011;
 7     //wire [15:0]Rbus;
 8     wire [15:0]Rbus,Rbuscode;
 9     wire ready;
10     tanhcal ime(ready,clk,rst,start,Xbus,Rbus);
11     tnhx shayan(clk,rst,start,Xbus,ready,Rbuscode);
12     initial begin
13       #1000;
14       rst=0;
15       #1000;
16       start=1;
17       #2000;
18       start=0;
19     end
20     initial begin
21       #1000;
22       repeat(100)begin
23       clk=~clk;
24       #1000;
25       end
26     end
27     initial begin
28     #200000;
29     $stop;
30     end
31   endmodule
```

E





	Source State	Destination State	Condition
1	000	001	(Start)
2	000	000	(!Start)
3	001	001	(Start)
4	001	010	(!Start)
5	010	011	
6	011	100	
7	100	101	
8	101	011	(!Co)

State Table

Transitions / Encoding

F

C:/altera/13.0sp1/dtpth.v (/tnhxtst/shayan/DP) - Default

Ln#

```
1   `timescale 1ns/1ns
2   module dtpth(input Clk,Rst,sub,selx,selm,selq,selrom,selt,sela,ldq,ldt,lde,inc,in0,input [15:0]X,outpu
3     reg [15:0]X1,X2,Mbus,Addbus,Ebus,Tbus,Expr,Xsq,Term,Qout;
4     reg [2:0]Addrom;
5     reg [31:0]M;
6     //adding or subtracting
7     always @(sub,Expr,Term)begin
8       Addbus<=sub?(Expr-Term):(Expr+Term);
9     end
10    //multiply
11    always @(X1,X2)begin
12      M=X1*X2;
13    end
14    //multiplier output
15    always @(M)begin
16      Mbus<=M[31:16];
17    end
18    //first mux for multiplier
19    always @(selx,selq,selrom,Qout)begin
20      X1<=selx?X:
21      selq?Xsq:
22      selrom?Qout:16'b0;
23    end
24    //second mux for multiplier
25    always @(selx,selt,Term)begin
26      X2<=selx?X:
27      selt?Term:16'b0;
28    end
```

C:/altera/13.0sp1/dtpth.v (/tnhxtst/shayan/DP) - Default

```
24    //second mux for multiplier
25    always @(selx,selt,Term)begin
26      X2<=selx?X:
27      selt?Term:16'b0;
28    end
29    //output
30    always @(Rbus,Expr)begin
31      Rbus<=Expr;
32    end
33    //mux for term register
34    always @(selx,selm,Mbus)begin
35      Tbus<=selx?X:
36      selm?Mbus:16'b0;
37    end
38    //mux for expr register
39    always @(selx,sela,Addbus)begin
40      Ebus<=selx?X:
41      sela?Addbus:16'b0;
42    end
43    //xsq register
44    always @(posedge Clk,posedge Rst)begin
45      if(Rst) Xsq<=16'b0;
46      else if(ldq)Xsq<=Mbus;
47    end
48    //term register
49    always @(posedge Clk,posedge Rst)begin
50      if(Rst) Term<=16'b0;
51      else if(ldt) Term<=Tbus;
```

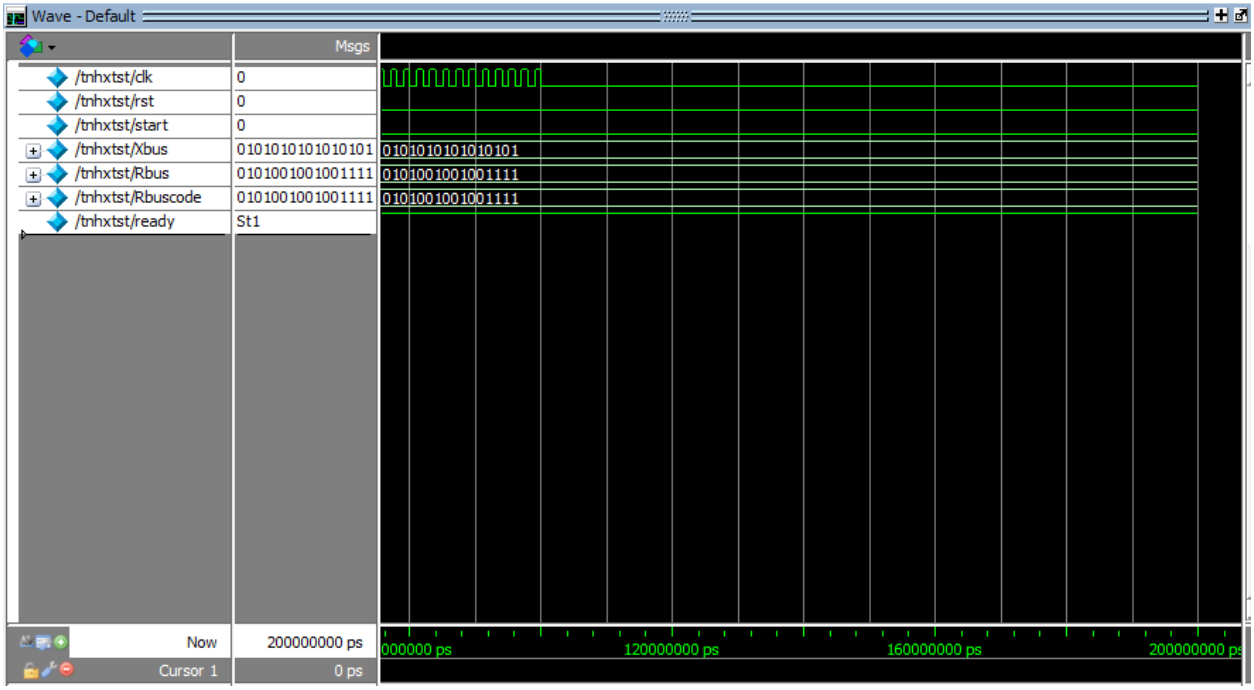
C:/altera/13.0sp1/dtpth.v (/tnhxtst/shayan/DP) - Default

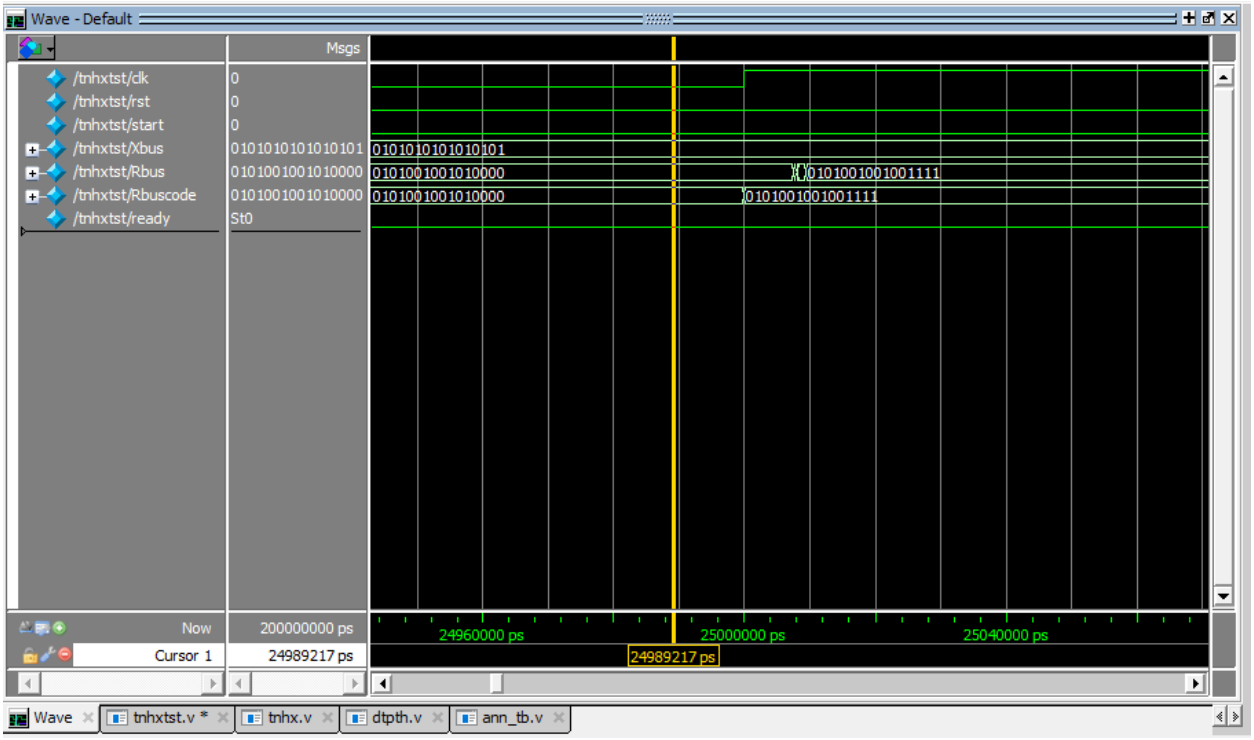
```
48    //term register
49    always @(posedge Clk,posedge Rst)begin
50      if(Rst) Term<=16'b0;
51      else if(ldt) Term<=Tbus;
52    end
53    //expr register
54    always @(posedge Clk,posedge Rst)begin
55      if(Rst) Expr<=16'b0;
56      else if(lde) Expr<=Ebus;
57    end
58    //counter for ROM
59    always @(posedge Clk,posedge Rst)begin
60      if(Rst) Addrom<=3'b0;
61      else if(in0) Addrom<=3'b0;
62      else if(inc) Addrom<=Addrom+1;
63    end
64    //carryout of counter
65    always @(Addrom)begin
66      Co<=Addrom[0]&Addrom[1]&Addrom[2];
67    end
68    //Even or Odd flag
69    always @(Addrom)begin
70      Oe<=Addrom[0];
71    end
72    //ROM
73    always @(Addrom)begin
74      Qout<=(Addrom==3'b000)?16'b0101010101010101:
75            (Addrom==3'b001)?16'b0110011001100110:
```

```
Ln#
62        else if(inc) Addrom<=Addrom+1;
63      end
64      //carryout of counter
65      always @(Addrom)begin
66        Co<=Addrom[0]&Addrom[1]&Addrom[2];
67      end
68      //Even or Odd flag
69      always @(Addrom)begin
70        Oe<=Addrom[0];
71      end
72      //ROM
73      always @(Addrom)begin
74        Qout<=(Addrom==3'b000)?16'b0101010101010101:
75               (Addrom==3'b001)?16'b0110011001100110:
76               (Addrom==3'b010)?16'b0110011110011110:
77               (Addrom==3'b011)?16'b0110011110111101:
78               (Addrom==3'b100)?16'b0110011111000000:
79               (Addrom==3'b101)?16'b0110011111000000:
80               (Addrom==3'b110)?16'b0110011111000000:
81               (Addrom==3'b111)?16'b0110011111000000:16'b0;
82      end
83
84
85    endmodule
86
87
88
```

C:/altera/13.0sp1/tnhx.v (/tnhxtst/shayan) - Default *

```
Ln#                                                                    Now
1   `timescale 1 ns/ 1 ns
2   module tnhx(input clk,rst,start,input[15:0]Xbus,output ready,output[15:0]Rbus);
3     wire In0c,Upc,selq,selr,selt,sela,selx,selm,Ldq,LdT,Lde,addsub,Oe,Co;
4
5     dtpth DP(clk,rst,addsub,selx,selm,selq,selr,selt,sela,Ldq,LdT,Lde,Upc,In0c,Xbus,Rbus,Co,Oe);
6
7     cntrlunt shcu(clk,rst,start,Co,Oe,addsub,selx,selm,selq,selr,selt,sela,Ldq,LdT,Lde,In0c,Upc,ready);
8
9   endmodule
10
```





با توجه به نتایج میبینیم که در هر دو حالت سنتز و بدون سنتز نتایج یکسان و به درستی به دست می آیند و تفاوت تنها در مقدار دیلی های نمونه سنتز شده است که به این دلیل است که درواقع نمونه سنتز شده از دیلی های واقعی استفاده میکند و تفاوت دیگر این است که در نمونه سنتز شده هر بار یک بیت تغییر میکند