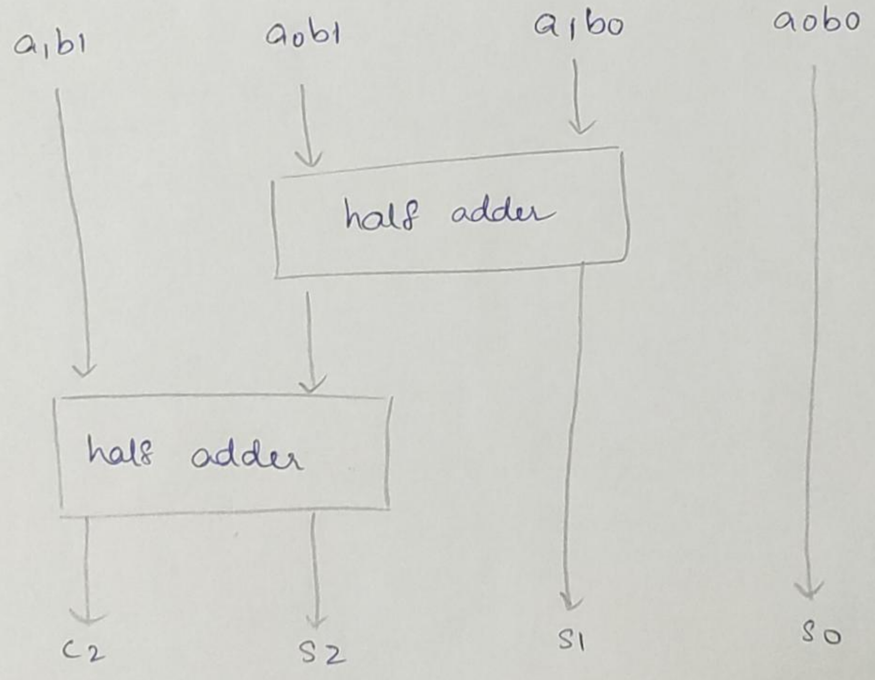
KOTAKONDA ABITHA RAO
194238-B

Aim:-
Build the verilog model for a 16-bit Vedic multiplier

Tools Used:-
Xilinx Virado 2019.2

Theory:-
Vedic multiplier is used to multiply 2 binary numbers and hence this multiplier can cause less power consumption and less delay.

2×2 bit Vedic Multipier is as follows:-



a1a0 and b1b0 are the numbers to be multiplied.

KOTAKONDA ABITHA RAO
194238-B

→ A 4x4 bit Vedic multiplier can be built from 4 2x2 Vedic multipliers and 3 RCA's.

→ Similarly, 8x8 bit vedic multiplier can be built from 4 4x4 Vedic multipliers and 3 adders.

→ And again 16x16 bit Vedic multiplier can be built from 4 8x8 Vedic multipliers and 3 adders.

Code + Testbench:-

```
module Vedic (A,B,Q);
input [15:0] A,B;
output [31:0] Q;

wire [15:0] W1,W2, W3_W4, W5, W6;
wire [23:0] W7,W8,W9;

Ved_8bit V1(A[7:0], B[7:0], W1);
Ved_8bit V2(A[7:0], B[15:8], W2);
Ved _8bit V3(A[15:8], B[7:0], W3);
Ved_8bit V4(A[15:8], B[15:8], W4);

assign W5 = {8'b0000_0000, W1[15:8]};
assign W6= W5+W2;
assign W7= {8'b000_0000, W3};
assign W8= {W4, 8'b000_0000};
assign W9 = W7+W8;
assign Q[31:8] = W9 + W6;
assign Q[7:0] = W1;
```

KOTAKONDA ABITHA RAO
194288-B

```
endmodule.

module  Ved_8bit (A,B,Q);
input  [7:0]  A,B;
output  [15:0]  Q;

wire  [7:0]  w1,w2,w3, w4,w5,w6;
wire  (11:0]  w7,w8, w9;

Ved_4bit  V1 [A[3:0], B[3:0] , w1);
Ved_ 4bit  V2( A[3:0] ,B[7:4], w2);
Ved - 4bit  V3(A [7:4], B[3:0], w3);
Ved _4 bit  V4 (A[7:4] , B[7:4],w4);

assign  w5 = {4'b0000, w1[7:4]};
assign  w6 = w5+ w2;
assign  w7= {4'b0000; w3};
assign  w8= {w4, 4'b0000};
assign  w9= w7+w8;
assign  Q[15:4] = w9+w6;
assign  Q [3:0] =w1;

endmodule


module Ved_4bit (A,B,Q);
input  [3:0] A,B;
output [7:0] Q;

wire   [3:0] w1, w2,w3, w4, w5, w6, w7, w8, w9, CA1,
       CA2 , CA3;

Ved _2bit  V1 (A [1:0], B[1:0] , w1);
Ved -2bit  V2 (A [1:0] , B [3:2] ,w2);
```

KOTAKONDA ABITHA RAO
194238-B

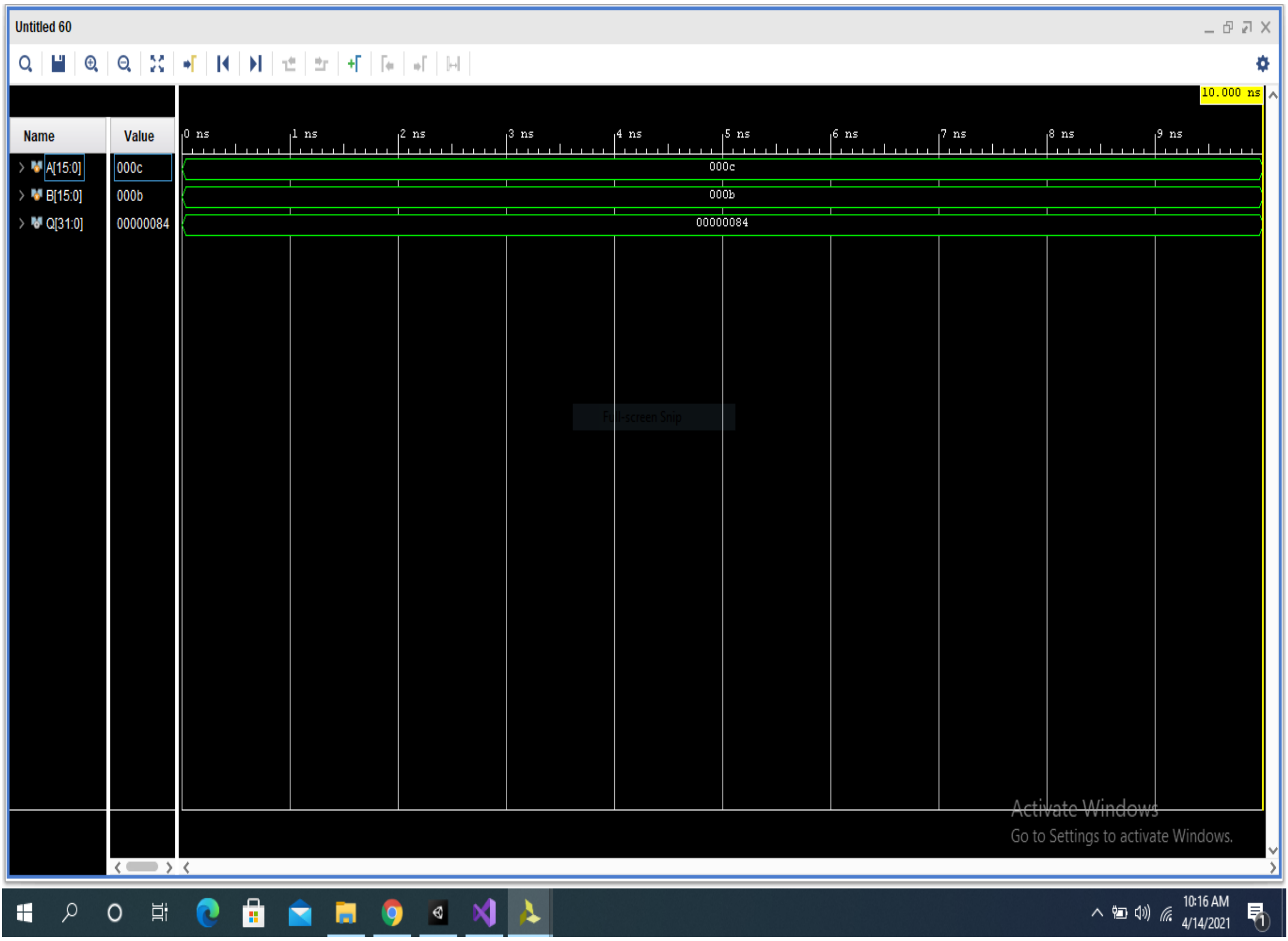
```
ved_2bit   V3 (A [3:2] , B [1:0] , w3);
ved_2bit   V4(A [3:2] , B[3:2] , w4);

assign  W6 = { 1'b0, 1'b0, w1[3], w1[2]};
RCA_4bit  R1(w2, w3, 0, CA1, w5);
RCA_4bit  R2(w5, w6, 0, CA2, w7);
assign  w8 = { CA1, 1'b0, w7[3], w7[2]};
RCA_4bit  R3 (w4,w8,0, CA3, w9);

assign {Q[1],Q[0]} = w[1:0];
assign {Q[3],Q[2]} = w7[1:0];
assign {Q[7],Q[6], Q[5], Q[4]} = w9;

endmodule

module  RCA_4bit (a, b, cin, cout, s);
input [3:0] a,b;
input cin;
output cout;
output [3:0] s;

wire [3:0] c;

for (genvar i=0; i<=3; i=i+1) begin
    if (i==0) begin
        assign {c[i],s[i]} = a[i] +b[i]+ cin;
    end
    else begin
        assign {c[i],s[i]}= a[i]+ b[i]+ c[i-1];
    end
assign cout = c[3];
endmodule;
```

KOTAKONDA ABITHA RAO
194238-B

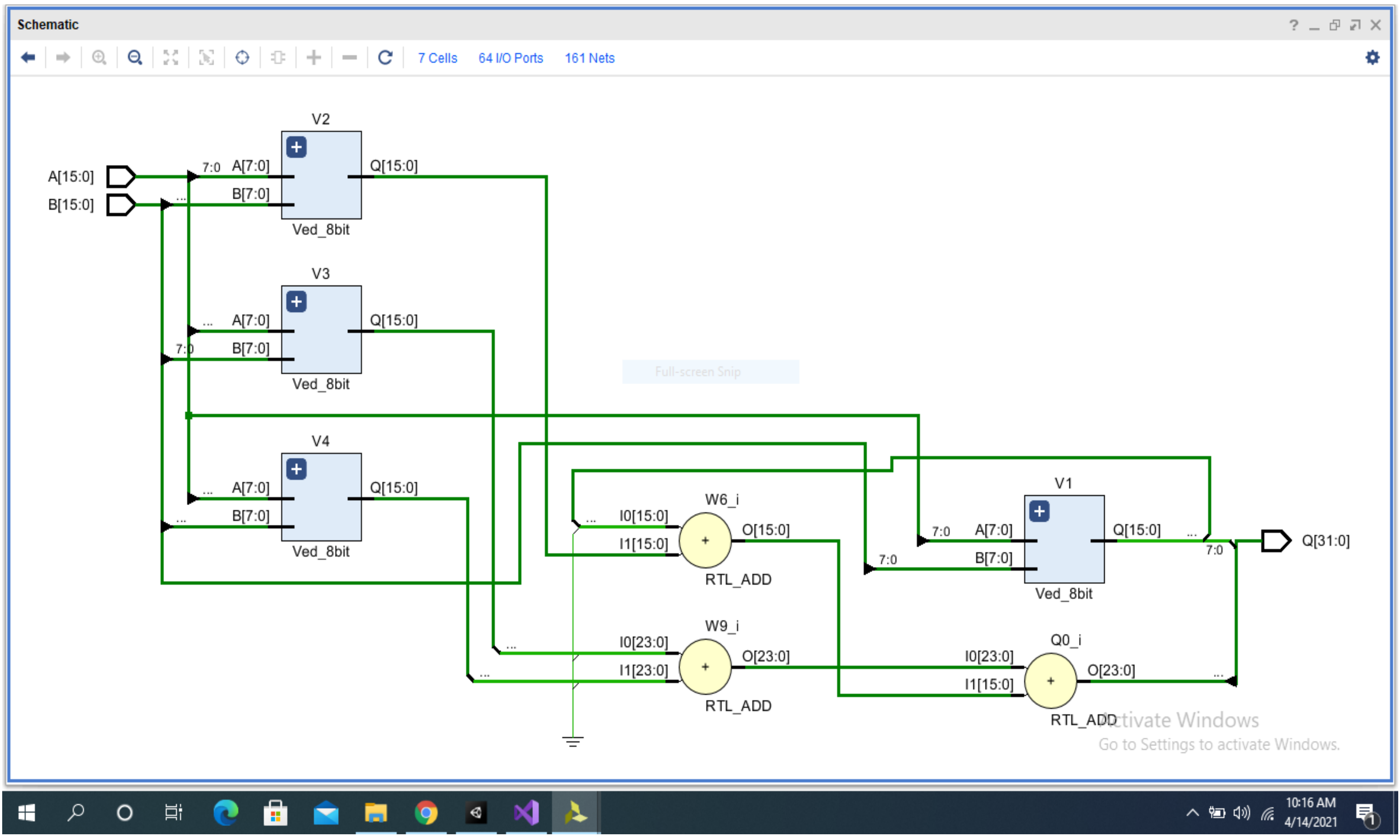
```
module ved_2bit (A,B,P);
input [1:0] A, B;
output [3:0] P;

wire t1, t2, t3, t4;

and ( P[0], A[0], B[0]);
and (t1, A[1], B[0]);
and (t2, A[0], B[1]);
and (t3, A[1], B[1]);
assign {t4, P[1]} = t1 + t2;
assign {P[3], P[2]}= t4 + t3 ;

endmodule.
```

```
module vedic_tb();
reg [15:0] A,B;
wire [31:0] Q;

vedic uut (A,B,Q);

initial
begin
A = 16'b0000_0000_0000_1100;
B= 16'b0000_0000_0000_1011;

#10
$ stop;
end
end module.
```

Result:-
we have implemented the logic for a Vedic multiplier and obtained its RTL schematic.



Name	Value
A[15:0]	000c
B[15:0]	000b
Q[31:0]	00000084



KOTAKONDA ABITHA RAO
194238-B

AIM:-

Build the verilog model for a 7bit Modified Booth's Multiplier

Tools Used:-

Xilinx Virado 2019.2;

Theory:-

Booth's multiplication algorithm is a multiplication algorithm that multiplies two signed
In Booth's modified multiplies, instead of Multiplier bit pair, we take 3 bits for Multiplier bit pair. It is then used to calculate the recoded multiplier following the values:-

0	0	→	0
1	1	→	0
0	1	→	+1
1	0	→	-1

First 0 is appended at the LSB and then it is extended with value equal to the signed bit (multiplier)
Then the bit pair recoded at the ith position is used to calculate the sup multiplication value and they are added to give the end result.

KOTAKONDA ABITHA RAO
194238-B

Code + Testbench:-

```
module  booths_mul(A, B, Q);
input  [6:0]  A,B;
output reg  [13:0] Q;

reg  [8:0]  B_comp;
reg  [13:0]  w, w1;
reg  [6:0]  w2;
integer i;
integer R;

always@(A  or  ~A  or  B  or  ~B)
begin
Q = 14'b0000_0000_0000_0000;
w2 = ~B;
B_comp = w2 + 1'b1;
B_comp = B_comp<<1;
B_comp[8]= B_comp[7];
for (i=1; i<=7; i=i+2)
begin
     R = 2* (B_comp[i] -B_comp[i+1]) +
              (B_comp[i-1] - B_comp [i]);
     if (R==1)
     begin
         assign  w1 = A;
         assign  w = w1 << i-1;
         Q = Q +w;
     end
    if (R== -1)
    begin
        assign  w1 = A;
        assign  w= ~w1+1;
        assign  w= w <<i-1;
```

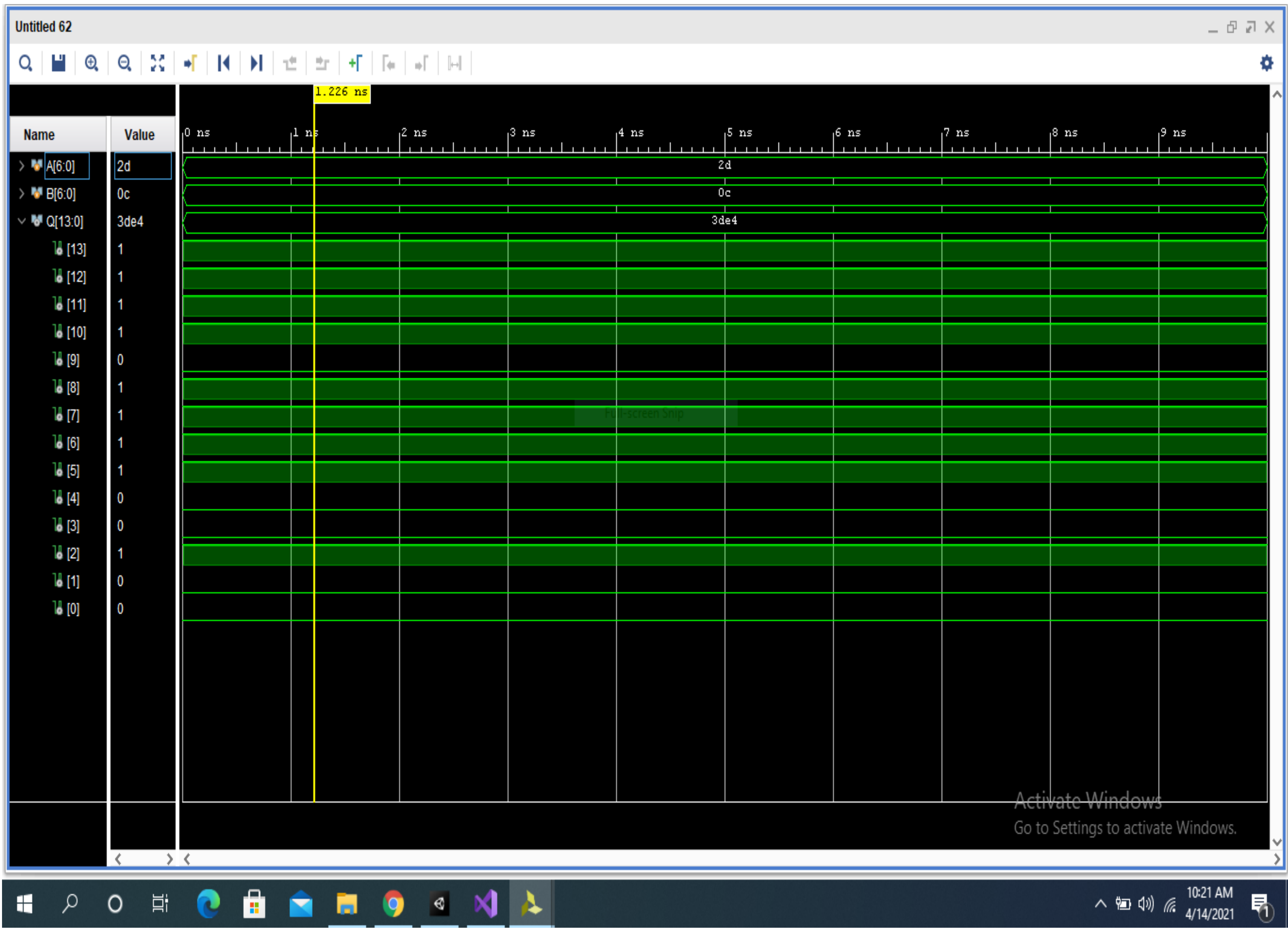
KOTAKONDA ABITHA RAO
194238-B

```
        Q = Q+w;
    end
    else if (R== -2)
    begin
         w1 = A;
         w = ~w1;
         w = w<< i-1;
         Q = Q+w;
     end
     else if (R==2)
     begin
         w1= A;
         w= w1<<i;
         Q = Q+w;
     end
     else
         Q = Q;
end
end
endmodule


module booths_mul_tb ();
reg [6:0] A,B;
wire [13:0] Q;

booths_mul  uut (A,B,Q);

initial begin
 B= 7'b000_1100;
 A = 7'b010_1101;
 #10
 $stop;
```

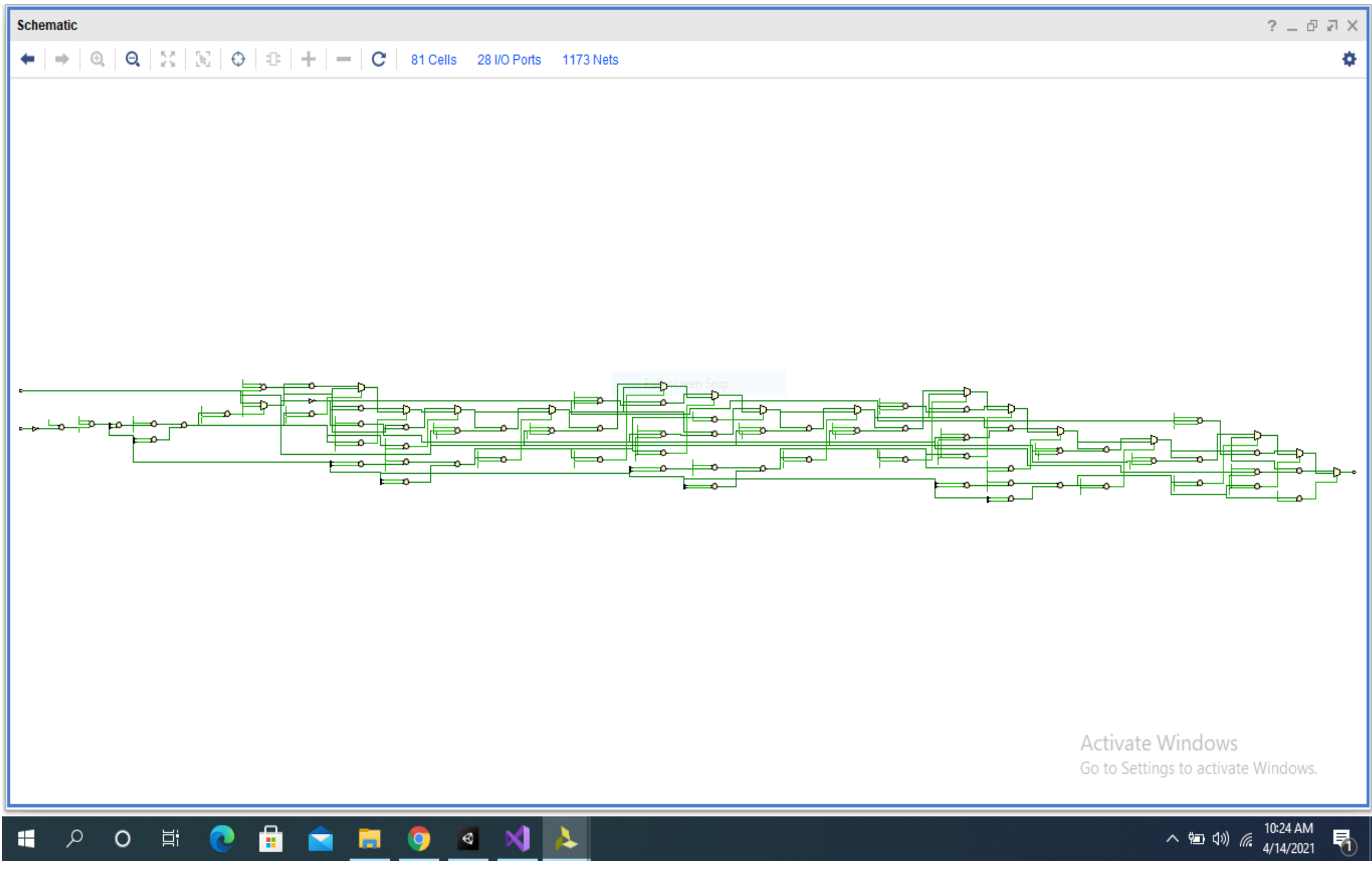
```
end
endmodule
```

Result :-

We have implemented the code for the modified Booth's multiplier of 7 bits. the waveform confirms that the code is correct. We have also obtained the RTL schematic for the modified Booth's multiplier.



Name	Value
A[6:0]	2d
B[6:0]	0c
Q[13:0]	3de4
[13]	1
[12]	1
[11]	1
[10]	1
[9]	0
[8]	1
[7]	1
[6]	1
[5]	1
[4]	0
[3]	0
[2]	1
[1]	0
[0]	0


Schematic
81 Cells
28 I/O Ports
1173 Nets
Activate Windows
Go to Settings to activate Windows.
10:24 AM
4/14/2021