KOTAKONDA ABITHA RAO
194238-B

Aim:-

To build the verilog model for a 4 bit ALU having 16 operations.

Tools Used:-

Vivado Xilinx 2019.2

Theory:-

The ALU performs simple operations as follows:-

Select pin(s)	Operation
0000	A+B
0001	A-B
0010	A+1
0011	A-1
0100	B+1
0101	B-1
0110	compare A & B
0111	A and B
1000	A OR B
1001	A XOR B
1010	A XNOR B
1011	Display A
1100	Display A'
1101	Display B
1110	Display B'
1111	------

KOTAKONDA ABITHA RAO
194238-B

Code + Testbench:-

```
module ALU (A, B, S, Cin, Cout, O);
input [3:0] A,B;
input Cin;
input [3:0] S;
output reg Cout;
output reg [3:0] O;

always @ (-S[0] or -S[1] or ~S[2] or ~S[3])
begin
case (S)
4'b0000 : {Cout, O} = A + B + Cin;
4'b0001 : {Cout, O} = A - B - Cin;
4'b0010 : {Cout, O} = A+1;
4'b0011 : {Cout, O} = A-1;
4'b0100 : {Cout, O} = B+1;
4'b0101 : {Cout, O} = B-1;
4'b0110 : begin
if (A>B)
O = 4'b0000;
else if (A<B)
O = 4'b0001;
else
O = 4'b0010;
Cout = 0;
end
4'b0111 : {Cout, O} = A & B;
4'b1000 : {Cout, O} = A | B;
4'b1001 : {Cout, O} = A ^ B;
4'b1010 : {Cout, O} = ~(A ^ B);
4'b1011 : {Cout, O} = A;
```

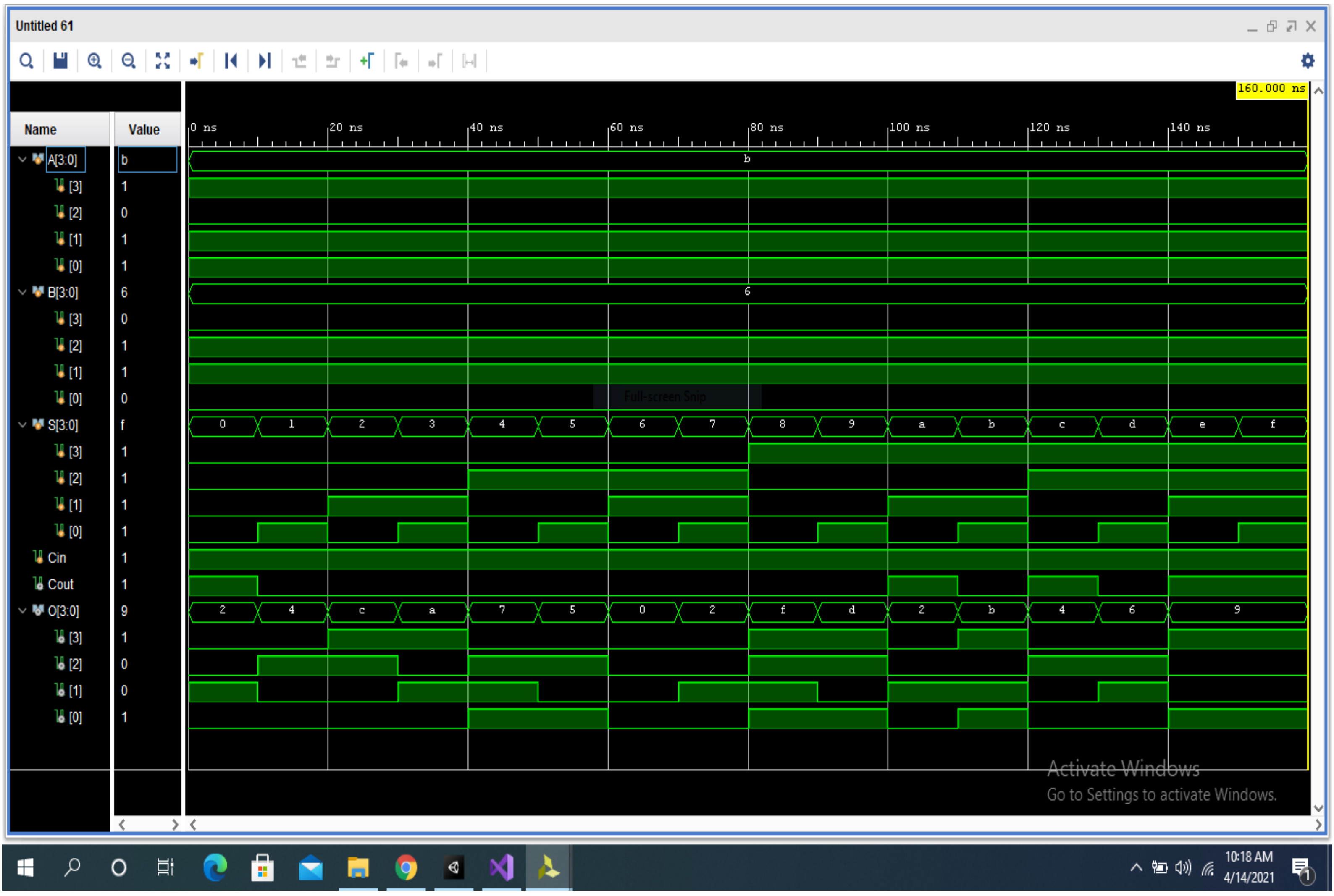
KOTAKONDA ABITHA RAO
194238-B

```
4'b1100 : {Cout, O} = ~A;
4'b1101 : {Cout, O} = B;
4'b1110 : {Cout, O} = ~B;
endcase
end

module ALU_tb();
reg [3:0] A, B, S;
reg Cin;
wire Cout;
wire [3:0] O;

ALU uut (A, B, S, Cin, Cout, O);

initial begin
A = 4'b1011; B = 4'b0110; Cin = 1'b1;
S = 4'b0001;
#10
S = 4'b0010;
#10
S = 4'b0011;
#10
S = 4'b0100;
#10
S = 4'b0101;
#10
S = 4'b0110;
#10
S = 4'b0110;
#10
S = 4'b0111;
```

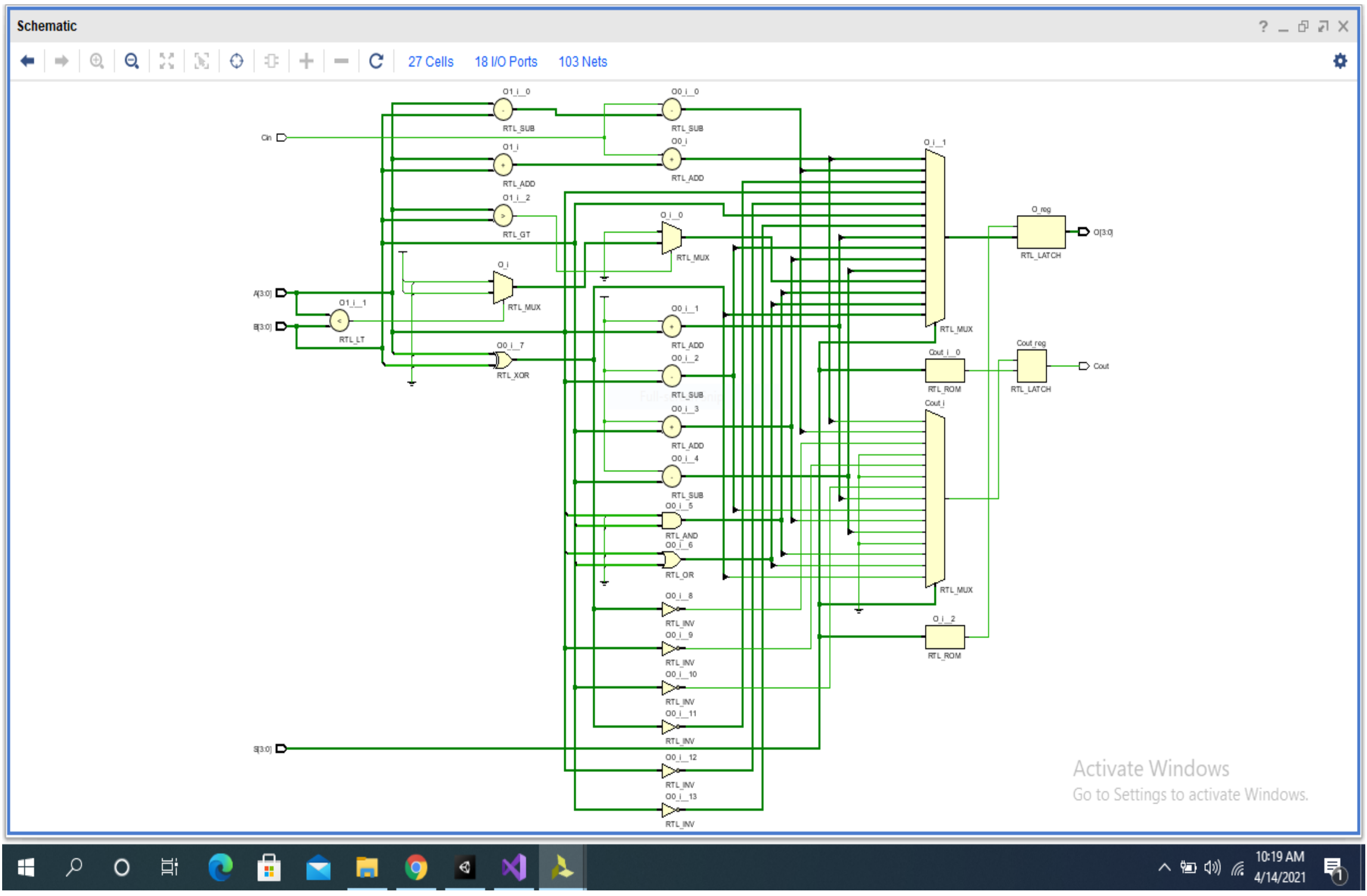
```
#10
S=4'b1000;
#10
S=4'b1001;
#10
S=4'b1010;
#10
S=4'b1011;
#10
S=4'b1100;
#10
S=4'b1101;
#10
S=4'b1110;
#10
S=4'b1111;
#10
$stop;
end
endmodule
```

Result:-

We have implemented the logic of ALU with 16 operations and obtained the correct waveform for the given testbench. The RTL schematic can also be seen in the next page.



Untitled 61
160.000 ns
Name
Value
0 ns
20 ns
40 ns
60 ns
80 ns
100 ns
120 ns
140 ns
A[3:0] b
[3] 1
[2] 0
[1] 1
[0] 1
B[3:0] 6
[3] 0
[2] 1
[1] 1
[0] 0
S[3:0] f
0 1 2 3 4 5 6 7 8 9 a b c d e f
[3] 1
[2] 1
[1] 1
[0] 1
Cin 1
Cout 1
O[3:0] 9
2 4 c a 7 5 0 2 f d 2 b 4 6 9
[3] 1
[2] 0
[1] 0
[0] 1
Activate Windows
Go to Settings to activate Windows.
10:18 AM
4/14/2021



Schematic
27 Cells
18 I/O Ports
103 Nets
Cin
A[3:0]
B[3:0]
S[3:0]
O1_i__0
RTL_SUB
O1_i
RTL_ADD
O1_i__2
RTL_GT
O_i
RTL_MUX
O1_i__1
RTL_LT
O0_i__7
RTL_XOR
O0_i__0
RTL_SUB
O0_i
RTL_ADD
O_i__0
RTL_MUX
O0_i__1
RTL_ADD
O0_i__2
RTL_SUB
O0_i__3
RTL_ADD
O0_i__4
RTL_SUB
O0_i__5
RTL_AND
O0_i__6
RTL_OR
O0_i__8
RTL_INV
O0_i__9
RTL_INV
O0_i__10
RTL_INV
O0_i__11
RTL_INV
O0_i__12
RTL_INV
O0_i__13
RTL_INV
O_i__1
RTL_MUX
Cout_i
RTL_MUX
Cout_i__0
RTL_ROM
O_i__2
RTL_ROM
O_reg
RTL_LATCH
O[3:0]
Cout_reg
RTL_LATCH
Cout
Activate Windows
Go to Settings to activate Windows.
10:19 AM
4/14/2021

KOTAKONDA ABITHA RAO
194238-B

Aim:-

To build the verilog model for the sequence detector (1011) which is both overlapping and non-overlapping in different modes.

Tools Used:-

Xilinx Vivado 2019.2

Theory:-

Sequence detector is used a detect a sequence piece in the give binary number. Here we are going to detect the sequence '1011'.
In the overlapping mode, it checks every possible continuous bits for the sequence.
In the non-overlapping modes, it checks first 'n' bits, then the next 'n' bits and so on where n is the size of the sequence.
When m=0, it is overlapping mode
when m=1, it is in non-overlapping mode.

KOTAKONDA ABITHA RAO
194238 - B

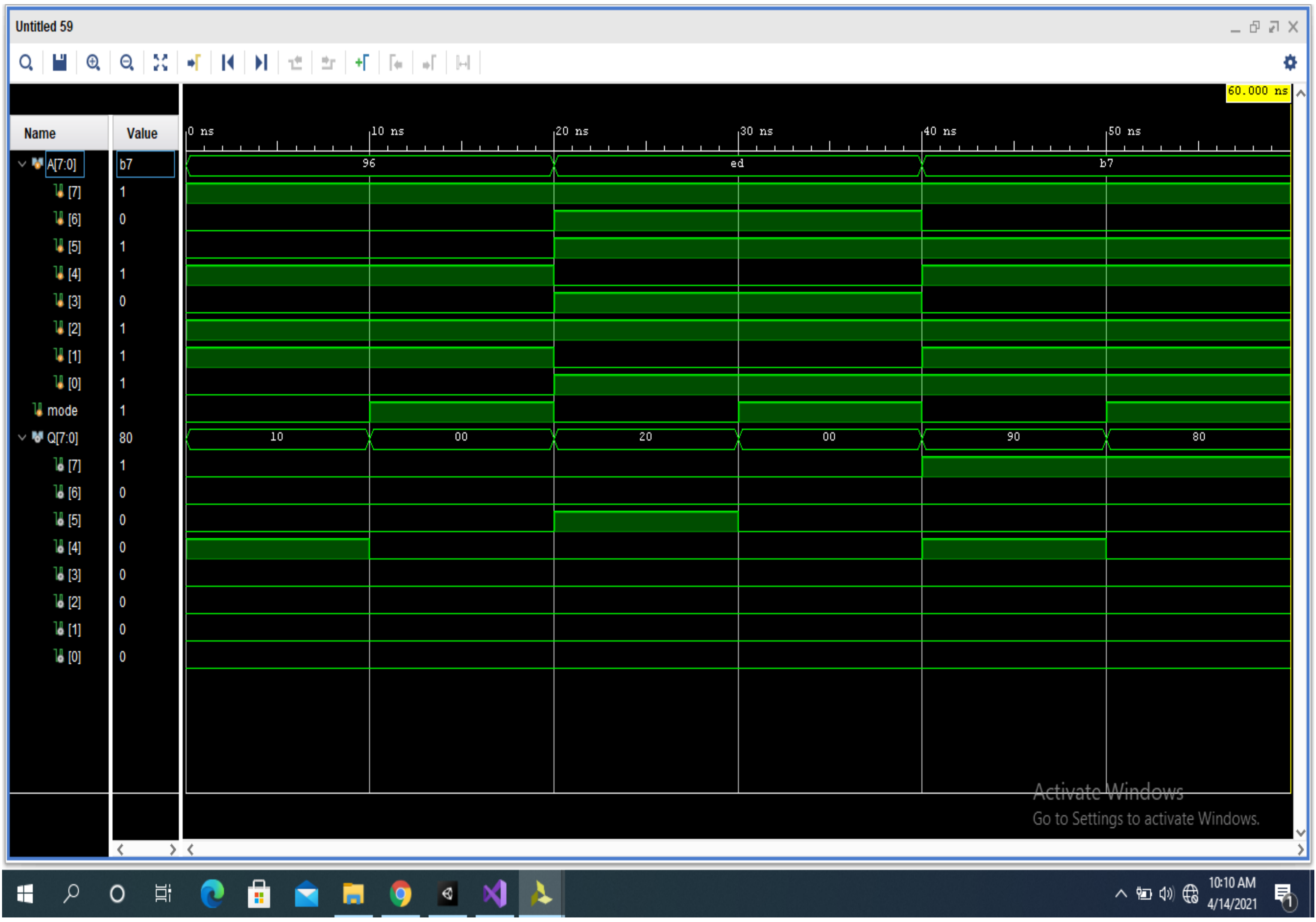
Code + Testbench

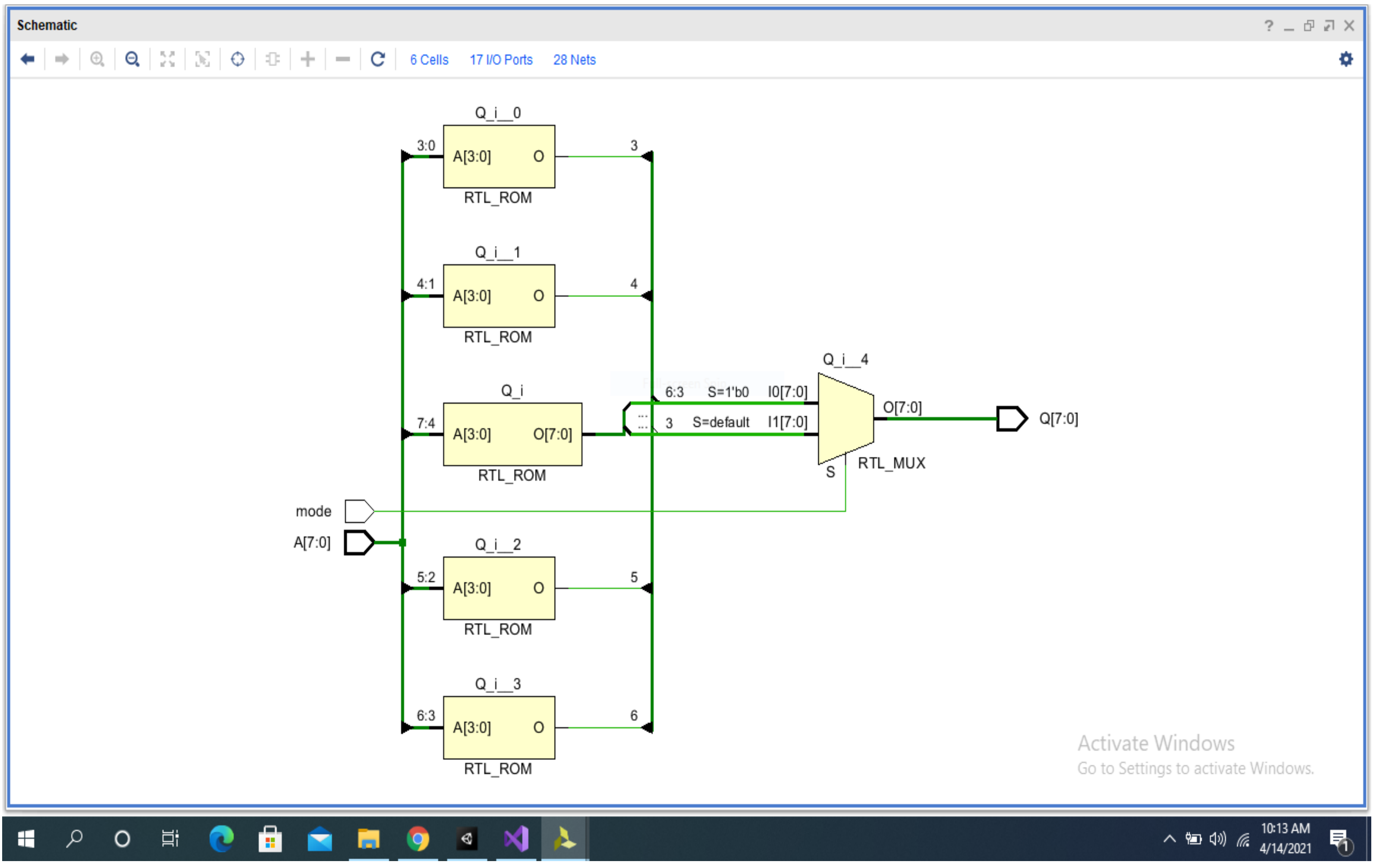
```
module seq_dec (A, Q, mode);
input [7:0] A;
input mode;
output reg [7:0] Q;

integer i;
always@ (A or ~A or mode or ~mode)
begin
Q = 8'b0000_0000;
if (mode ==0)
    for (i=7; i>=3; i=i-1)
    begin
        if (A[i-:4] == 4'b1011)
            Q[i]= 1'b1;
        else
            Q[i]= 1'b0;
    end
else
    for (i=7; i>=3; i=i-4)
    begin
        if [A[i-:4] == 4'b1011)
            Q[i]= 1'b1;
        else
            Q[i] = 1'b0;
    end
end
endmodule
```

KOTAKONDA ABITHA RAO
194238 - B

```
module seq-dec ();
reg [7:0] A;
reg mode;
wire [7:0] Q;

seq_dec uut (A, Q, mode);

initial
begin
A = 8'b1001_0110;
mode = 1'b0;
#10
mode = 1'b1;
#10
A = 8'b1110_1101;
mode = 1'b0;
#10
mode = 1'b1;
#10
A = 8'b1011_0111;
mode = 1'b0;
#10
mode = 1'b1;
#10
$stop;
end
endmodule.
```

Result:-
We have implemented the code for detecting the sequence '1011' and have also obtained the corresponding RTL. The modes cover both overlapping and non-overlapping.


Untitled 59
60.000 ns
Name
Value
A[7:0]
b7
mode
Q[7:0]
80
0 ns
10 ns
20 ns
30 ns
40 ns
50 ns
96
ed
b7
10
00
20
00
90
80
Activate Windows
Go to Settings to activate Windows.
10:10 AM
4/14/2021


Schematic
6 Cells
17 I/O Ports
28 Nets
Q_i__0
A[3:0]
O
RTL_ROM
3:0
3
Q_i__1
4:1
4
Q_i
7:4
O[7:0]
Q_i__4
6:3
S=1'b0
I0[7:0]
3
S=default
I1[7:0]
O[7:0]
S
RTL_MUX
Q[7:0]
mode
A[7:0]
Q_i__2
5:2
5
Q_i__3
6:3
6
Activate Windows
Go to Settings to activate Windows.
10:13 AM
4/14/2021