22bec121

Experiment – 2 Date:-25/1/2024

Lab Work

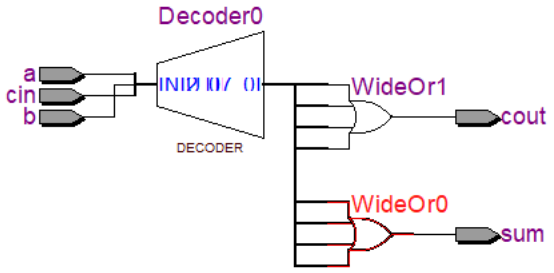
Q1 . Write down the code for 1-bit full adder using behavioural style of modelling .

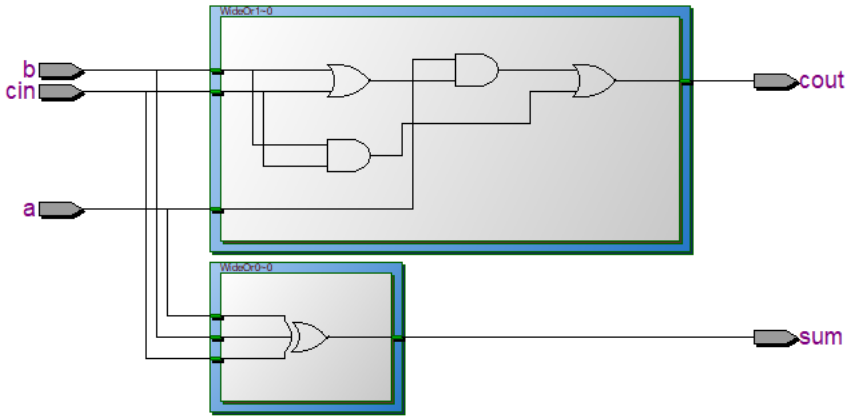
Code

```
module sneh20(input a,b,cin,output reg sum,cout);

always@(*)
begin
        case({a,b,cin})
                3'b000: begin sum=0;cout=0; end
                3'b001: begin sum=1;cout=0; end
                3'b010: begin sum=1;cout=0 ;end
                3'b011: begin sum=0;cout=1 ;end
                3'b100: begin sum=1;cout=0 ;end
                3'b101: begin sum=0;cout=1 ;end
                3'b110: begin sum=0;cout=1; end
                3'b111: begin sum=1;cout=1; end
        endcase
end

endmodule
```

```
1  //Case Statements
2  module sneh20(input a,b,cin,output reg sum,cout);
3
4  always@(*)
5  begin
6     case({a,b,cin})
7        3'b000: begin sum=0;cout=0; end
8        3'b001: begin sum=1;cout=0; end
9        3'b010: begin sum=1;cout=0 ;end
10       3'b011: begin sum=0;cout=1 ;end
11       3'b100: begin sum=1;cout=0 ;end
12       3'b101: begin sum=0;cout=1 ;end
13       3'b110: begin sum=0;cout=1; end
14       3'b111: begin sum=1;cout=1; end
15    endcase
16 end
17
18 endmodule
19
```

Output 1)Cyclone II (RTL View), 2) Cyclone II (TTL View)





Q2 . Modify the code that can perform Addition and Subtraction operation using a 'Mode Control Switch'.

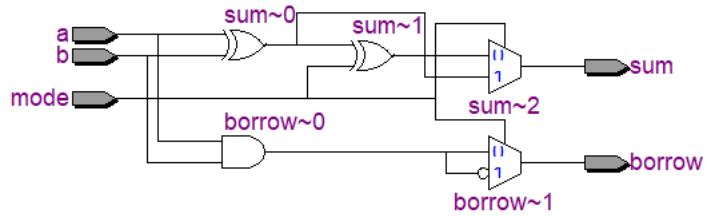
Code

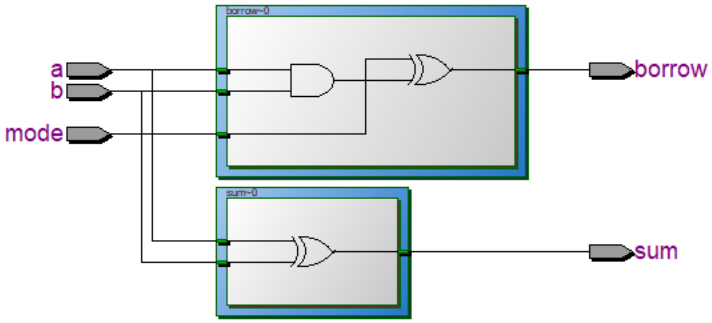
```
module sneh20(
   input wire a,
   input wire b,
   input wire mode,
   output wire sum,
   output wire borrow
);

assign sum = mode ? a ^ b : a ^ b ^ mode;
assign borrow = mode ? ~(a & b) : a & b;

endmodule
```

```
2  module sneh20(
3      input wire a,
4      input wire b,
5      input wire mode,
6      output wire sum,
7      output wire borrow
8  );
9
0  assign sum = mode ? a ^ b : a ^ b ^ mode;
1  assign borrow = mode ? ~(a & b) : a & b;
2
3  endmodule
4
```

Output

1)Cyclone II (RTL View), 2) Cyclone II (TTL View)





Code

```
module sneh20(
   input  a,
   input  b,
   input  mode,
   output reg sum,
   output reg borrow
);


always @(*) begin
   if (mode == 0) begin
```

sum = a + b;

borrow = 0;

end else begin

sum = a - b;

borrow = (a < b) ? 1 : 0;

end

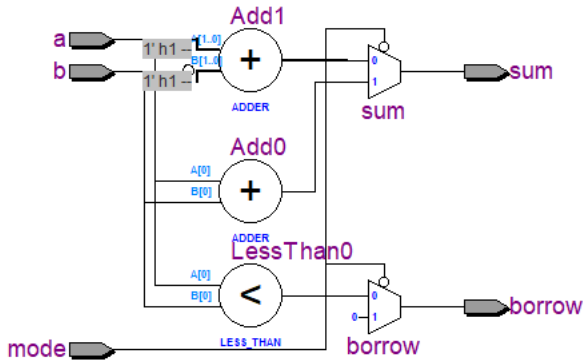
end

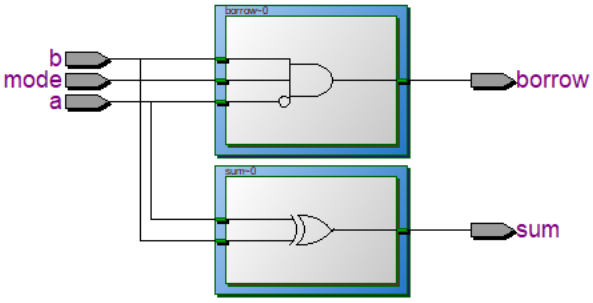
endmodule

```
35  module sneh20(
36      input  a,
37      input  b,
38      input  mode,
39      output reg sum,
40      output reg borrow
41  );
42
43  always @(*) begin
44      if (mode == 0) begin
45          sum = a + b;
46          borrow = 0;
47      end else begin
48          sum = a - b;
49          borrow = (a < b) ? 1 : 0;
50      end
51  end
52
53  endmodule
54
```

Output

1)Cyclone II (RTL View), 2) Cyclone II (TTL View)





Q3 . Carry out the modelling using If-ElseIf-Else , Nested If .

Code

```
module sneh20(input a, b, cin, output reg sum, cout);



always @(*) begin

   if ({a, b, cin} == 3'b000) begin

      sum = 0;

      cout = 0;

   end else if ({a, b, cin} == 3'b001) begin
```

```
    sum = 1;
    cout = 0;
end else if ({a, b, cin} == 3'b010) begin
    sum = 1;
    cout = 0;
end else if ({a, b, cin} == 3'b011) begin
    sum = 0;
    cout = 1;
end else if ({a, b, cin} == 3'b100) begin
    sum = 1;
    cout = 0;
end else if ({a, b, cin} == 3'b101) begin
    sum = 0;
    cout = 1;
end else if ({a, b, cin} == 3'b110) begin
    sum = 0;
    cout = 1;
end else begin
    sum = 1;
```

cout = 1;

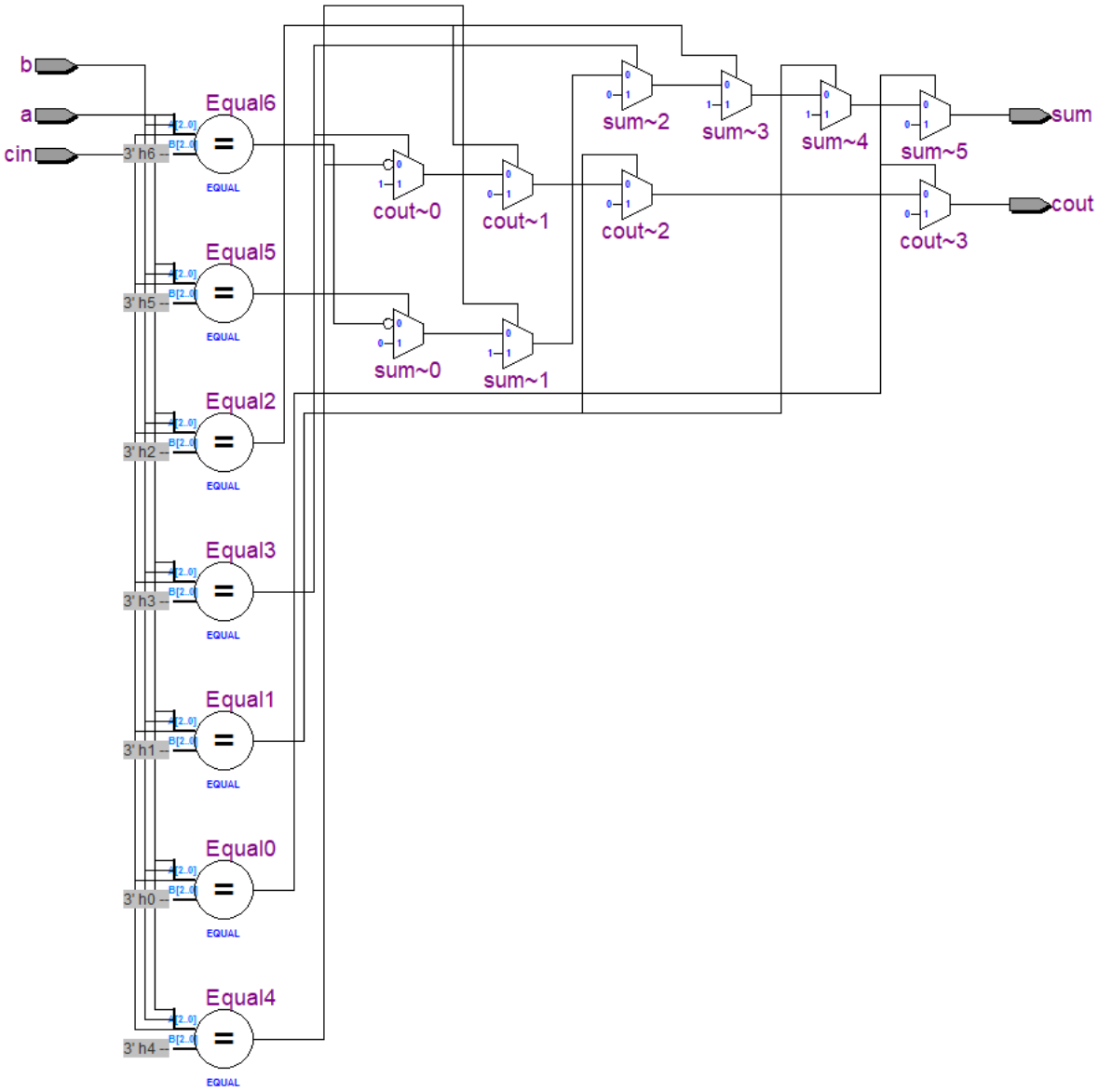
end

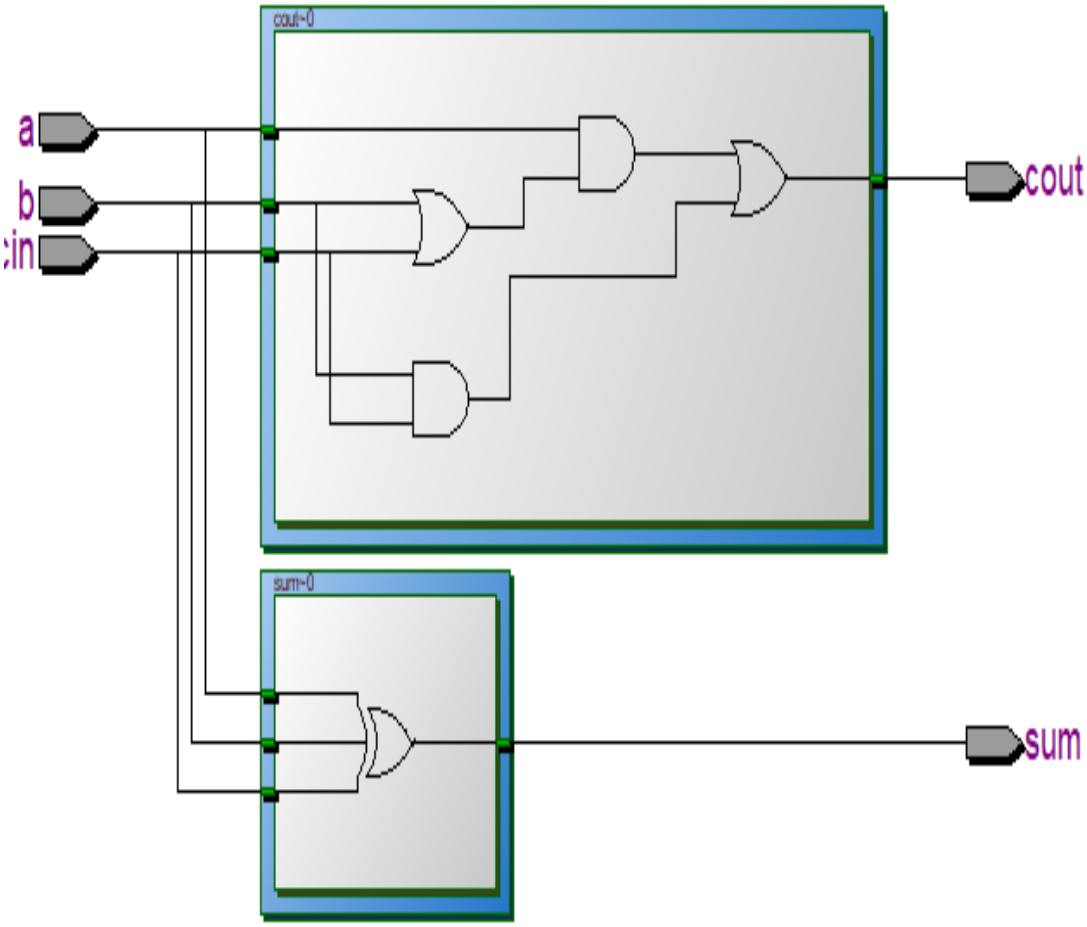
end

endmodule

```
56  module sneh20(input a, b, cin, output reg sum, cout);
57
58  always @(*) begin
59      if ({a, b, cin} == 3'b000) begin
60          sum = 0;
61          cout = 0;
62      end else if ({a, b, cin} == 3'b001) begin
63          sum = 1;
64          cout = 0;
65      end else if ({a, b, cin} == 3'b010) begin
66          sum = 1;
67          cout = 0;
68      end else if ({a, b, cin} == 3'b011) begin
69          sum = 0;
70          cout = 1;
71      end else if ({a, b, cin} == 3'b100) begin
72          sum = 1;
73          cout = 0;
74      end else if ({a, b, cin} == 3'b101) begin
75          sum = 0;
76          cout = 1;
77      end else if ({a, b, cin} == 3'b110) begin
78          sum = 0;
79          cout = 1;
80      end else begin
81          sum = 1;
82          cout = 1;
83      end
84  end
85
86  endmodule
```

Output 1)Cyclone II (RTL View), 2) Cyclone II (TTL View)


b
a
cin
Equal6
3' h6
EQUAL
Equal5
3' h5
EQUAL
Equal2
3' h2
EQUAL
Equal3
3' h3
EQUAL
Equal1
3' h1
EQUAL
Equal0
3' h0
EQUAL
Equal4
3' h4
EQUAL
cout~0
cout~1
cout~2
cout~3
sum~0
sum~1
sum~2
sum~3
sum~4
sum~5
sum
cout



Code

```
module sneh20(input a, b, cin, output reg sum, cout);


always @(*) begin
   if (a == 0) begin
      if (b == 0) begin
         if (cin == 0) begin
            sum = 0;
```

```
            cout = 0;
        end else begin
            sum = 1;
            cout = 0;
        end
    end else begin
        if (cin == 0) begin
            sum = 1;
            cout = 0;
        end else begin
            sum = 0;
            cout = 1;
        end
    end
end else begin
    if (b == 0) begin
        if (cin == 0) begin
            sum = 1;
            cout = 0;
        end else begin
            sum = 0;
            cout = 1;
        end
    end else begin
        if (cin == 0) begin
            sum = 0;
            cout = 1;
        end else begin
```

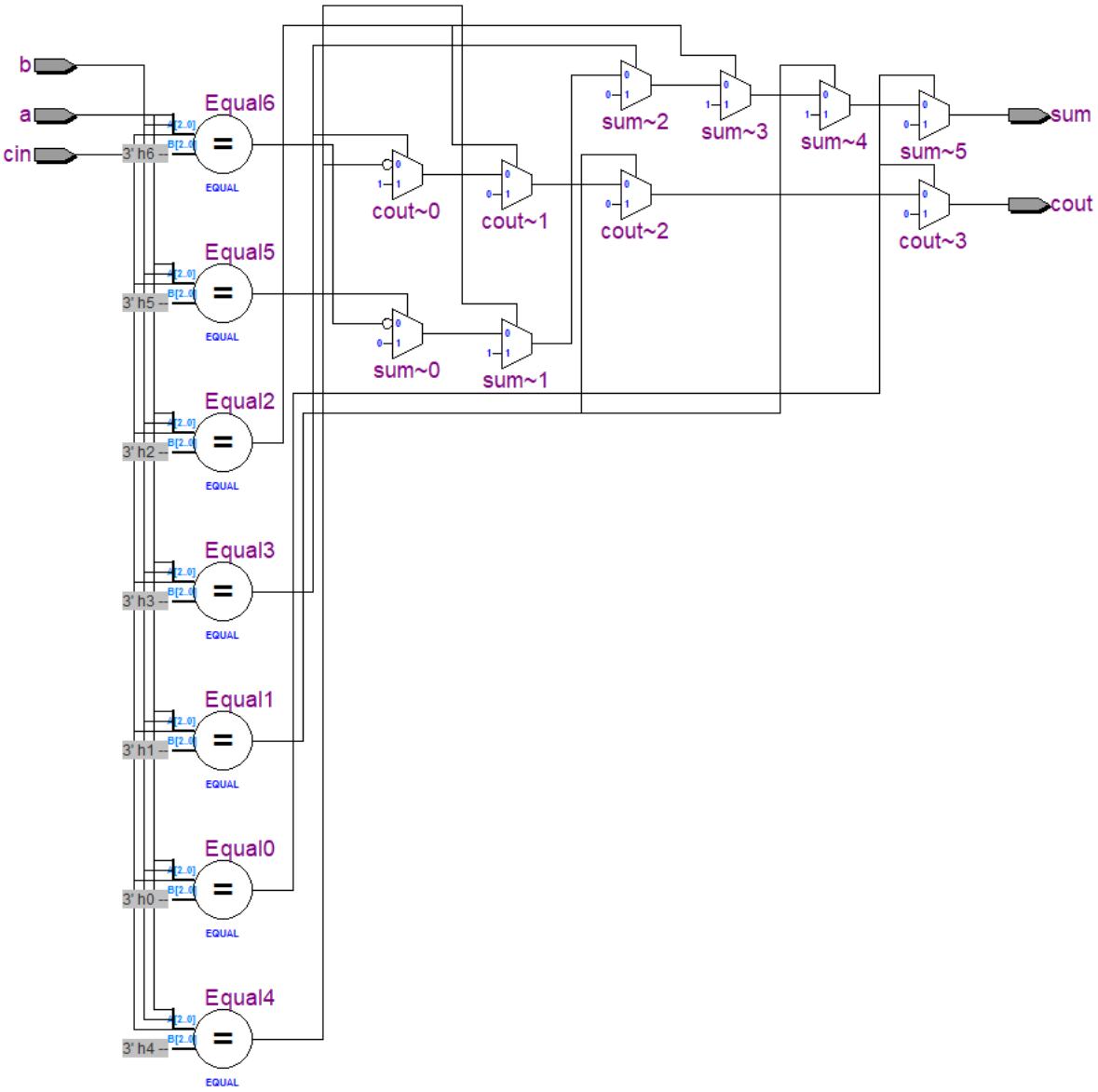
```
        sum = 1;
        cout = 1;
      end
    end
  end
end
```

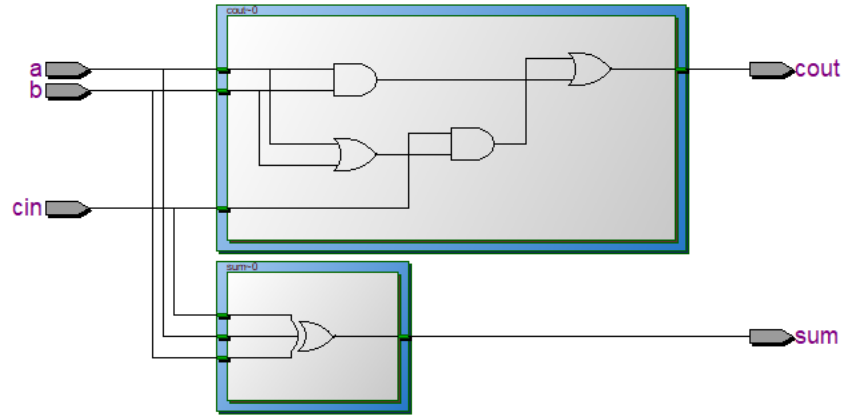
endmodule

```
89   module sneh20(input a, b, cin, output reg sum, cout);
90
91  always @(*) begin
92      if (a == 0) begin
93          if (b == 0) begin
94              if (cin == 0) begin
95                  sum = 0;
96                  cout = 0;
97              end else begin
98                  sum = 1;
99                  cout = 0;
00              end
01          end else begin
02              if (cin == 0) begin
03                  sum = 1;
04                  cout = 0;
05              end else begin
06                  sum = 0;
07                  cout = 1;
08              end
09          end
10      end else begin
11          if (b == 0) begin
12              if (cin == 0) begin
13                  sum = 1;
14                  cout = 0;
15              end else begin
16                  sum = 0;
17                  cout = 1;
```

```
116                 sum = 0;
117                 cout = 1;
118             end
119         end else begin
120             if (cin == 0) begin
121                 sum = 0;
122                 cout = 1;
123             end else begin
124                 sum = 1;
125                 cout = 1;
126             end
127         end
128     end
129 end
130
131 endmodule
132
```

Output 1)Cyclone II (RTL View), 2) Cyclone II (TTL View)


b
a
cin
Equal6
3' h6
EQUAL
Equal5
3' h5
Equal2
3' h2
Equal3
3' h3
Equal1
3' h1
Equal0
3' h0
Equal4
3' h4
cout~0
cout~1
cout~2
sum~0
sum~1
sum~2
sum~3
sum~4
sum~5
sum
cout
cout~3


a
b
cin
cout
sum

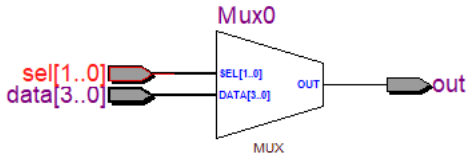
Q4 . Repeat the experiment execution flow for 4x1 multiplexer.

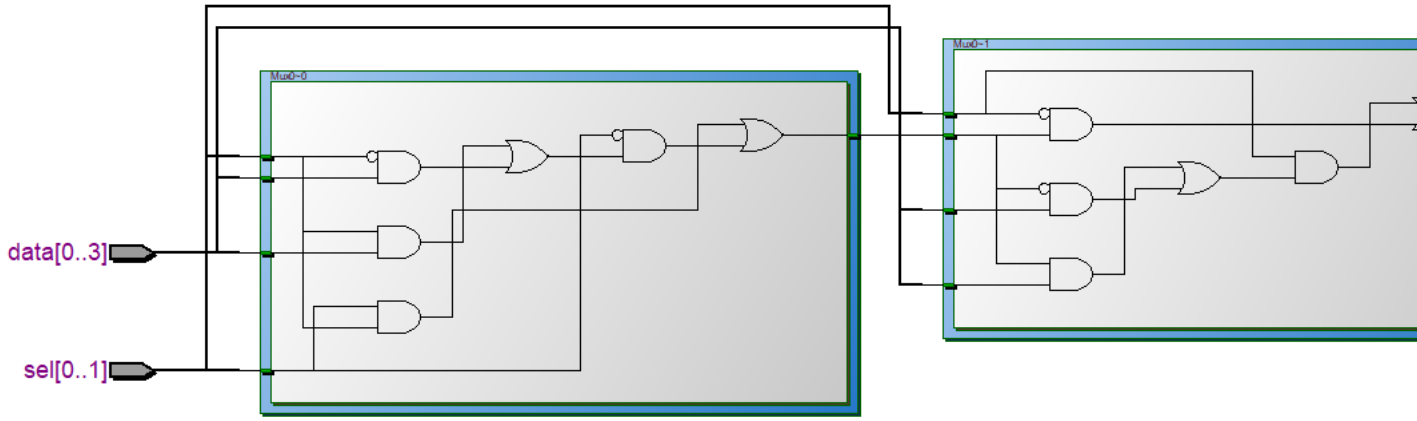
Code

```
module sneh20(
   input wire [3:0] data,
   input wire [1:0] sel,
   output reg out
);

always @(*) begin
   case(sel)
      2'b00: out = data[0];
      2'b01: out = data[1];
      2'b10: out = data[2];
      2'b11: out = data[3];
      default: out = 0;
   endcase
end

endmodule
```

```
134  module sneh20(
135      input wire [3:0] data,
136      input wire [1:0] sel,
137      output reg out
138  );
139
140  always @(*) begin
141      case(sel)
142          2'b00: out = data[0];
143          2'b01: out = data[1];
144          2'b10: out = data[2];
145          2'b11: out = data[3];
146          default: out = 0;
147      endcase
148  end
149
150  endmodule
```

Output 1)Cyclone II (RTL View), 2) Cyclone II (TTL View)





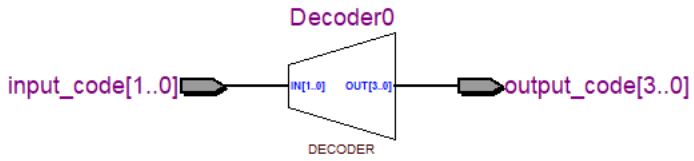
Q5 . Repeat the experiment execution flow for 2x4 Decoder.

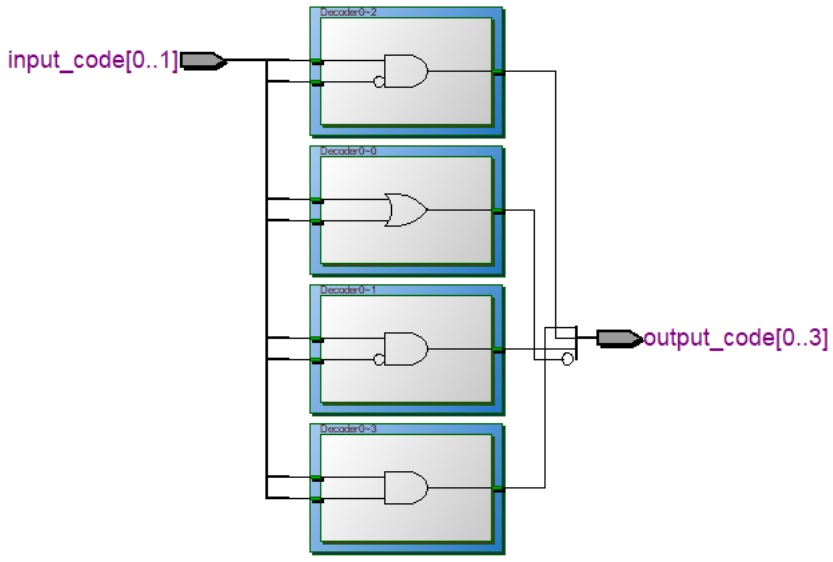
Code

```
module sneh20(
   input wire [1:0] input_code,
   output reg [3:0] output_code
);

always @(*) begin
   case(input_code)
      2'b00: output_code = 4'b0001;
      2'b01: output_code = 4'b0010;
      2'b10: output_code = 4'b0100;
      2'b11: output_code = 4'b1000;
      default: output_code = 4'bxxxx;
   endcase
end

endmodule
```

```
152 module sneh20(
153     input wire [1:0] input_code,
154     output reg [3:0] output_code
155 );
156
157 always @(*) begin
158     case(input_code)
159         2'b00: output_code = 4'b0001;
160         2'b01: output_code = 4'b0010;
161         2'b10: output_code = 4'b0100;
162         2'b11: output_code = 4'b1000;
163         default: output_code = 4'bxxxx;
164     endcase
165 end
166
167 endmodule
168
```

Output 1)Cyclone II (RTL View), 2) Cyclone II (TTL View)





Conclusion :-In this experiment we learnt that IF statements are synthesized by generating a multiplexer for each variable assigned within the if statement .We also

learnt to implement different logic circuits using the behavioural style of modelling and we got to know that the behavioural modelling is the fastest among all the other modelling styles .