Abdalrahman Ali Eltaher

Group 2

Assignment_1

- **Q[1]**
  - Code:

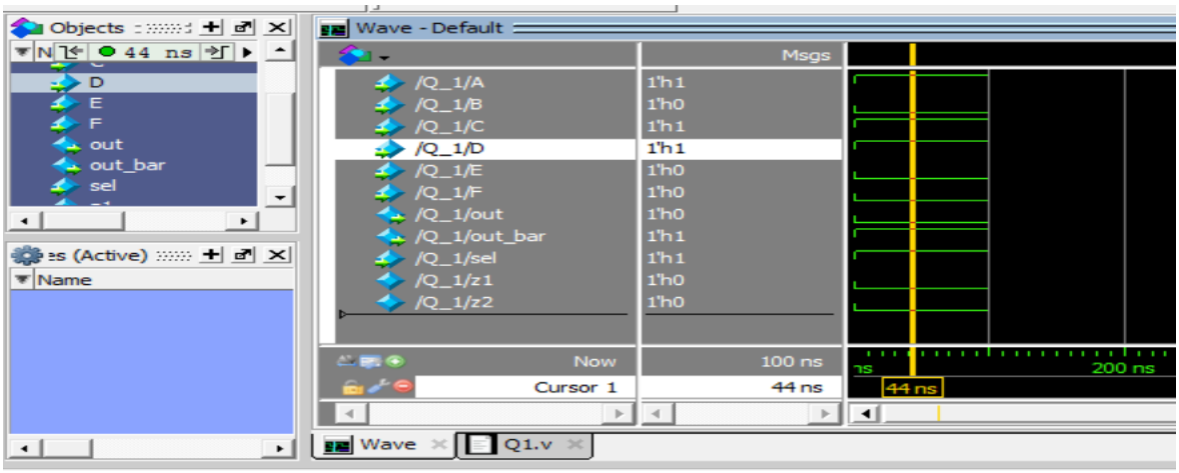
```
Q1.v
E: > Deploma karem > [not_extra]Assignmen_1 > Q1.v
 1  module Q_1(A,B,C,D,E,F,out,out_bar,sel);
 2  input A,B,C,D,E,F,sel ;
 3  output out,out_bar ;
 4  wire z1,z2;
 5  assign z1=A & B & C;
 6  assign z2 = D ~^ E ~^ F;
 7  assign out = (sel == 1)? z2 : z1;
 8  assign out_bar = (sel == 1)? ~z2 : ~z1;
 9
10  endmodule
```

  - Wave:

$\because A = 1 \ , B = 0 \ \text{and} \ C = 1 \ \therefore z1 = 0$

$\because D = 1 \ , E = 0 \ \text{and} \ F = 0 \ \therefore z2 = 0$

$\because sel = 1 \ \therefore \ out = 0 \ \text{and} \ out\_bar = 1$



Q[2]

➢Code:

exercise.v | Q2.v | Settings

E: > Deploma karem > [not_extra]Assignmen_1 > Q2.v

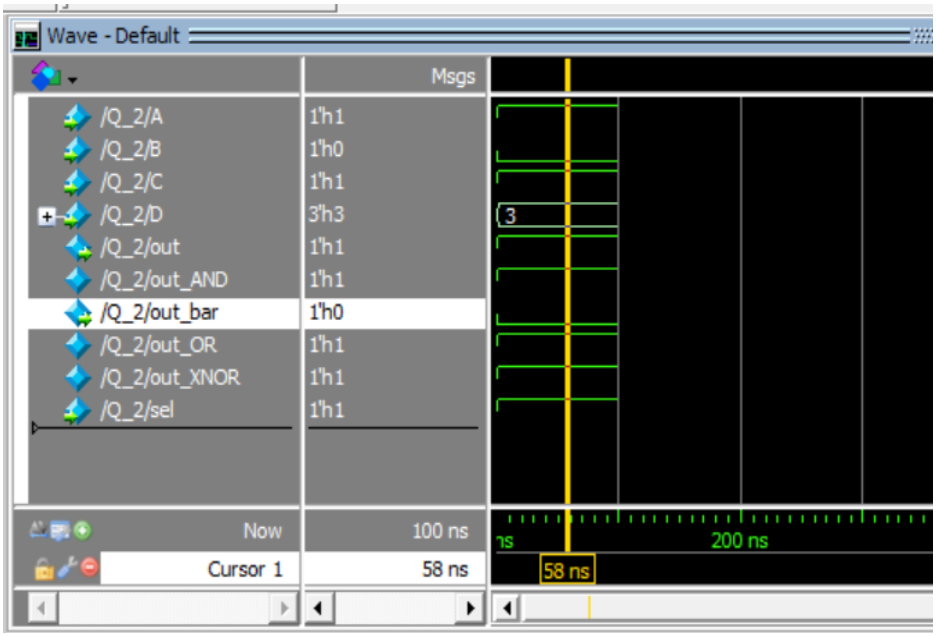
```
1   module Q_2(A,B,C,sel,D,out,out_bar);
2   input[2:0] D;
3   input A,B,C,sel;
4   output reg out,out_bar ;
5   reg  out_AND,out_OR,out_XNOR ;
6   always @(*) begin
7       out_AND = D[0] & D[1];
8       out_OR = D [2] | out_AND ;
9       out_XNOR = ~ (A ^ B ^ C ) ;
10      if (sel == 1) begin
11          out = out_XNOR ;
12          out_bar = ~out_XNOR;
13      end
14      else begin
15          out = out_OR ;
16          out_bar = ~ out_OR;
17      end
18
19  end
20
21  endmodule
```

➢Wave:

- The input D is 3’b011 and The input A=1,B=0 and C =1.

  ∴ out_AND = 1 ,out_OR =1 and out_XNOR =1

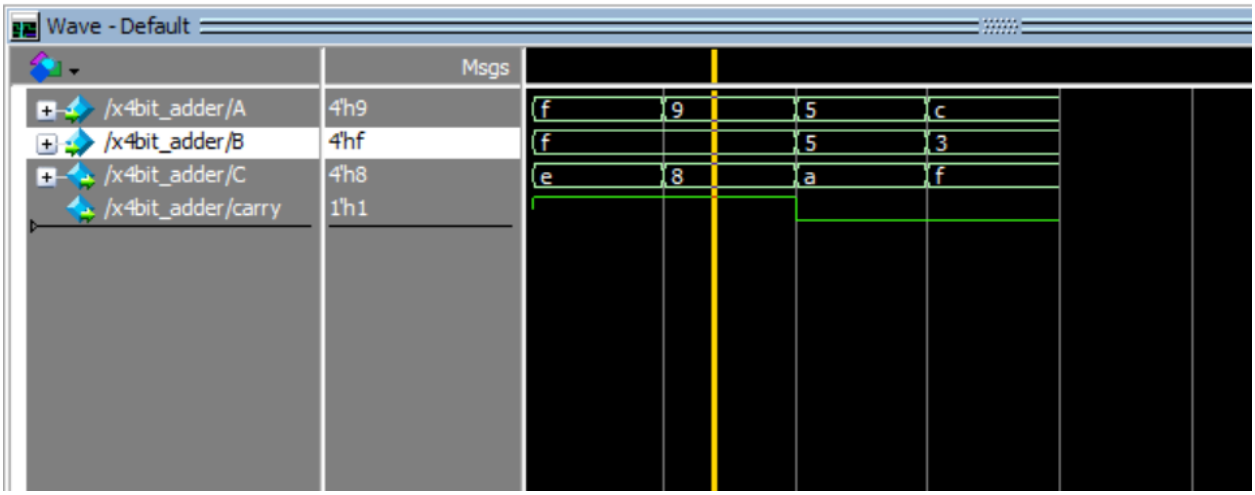
  ∵ sel =1 ∴ out = out_XNOR =1 and out_bar = not out_XNOR =0



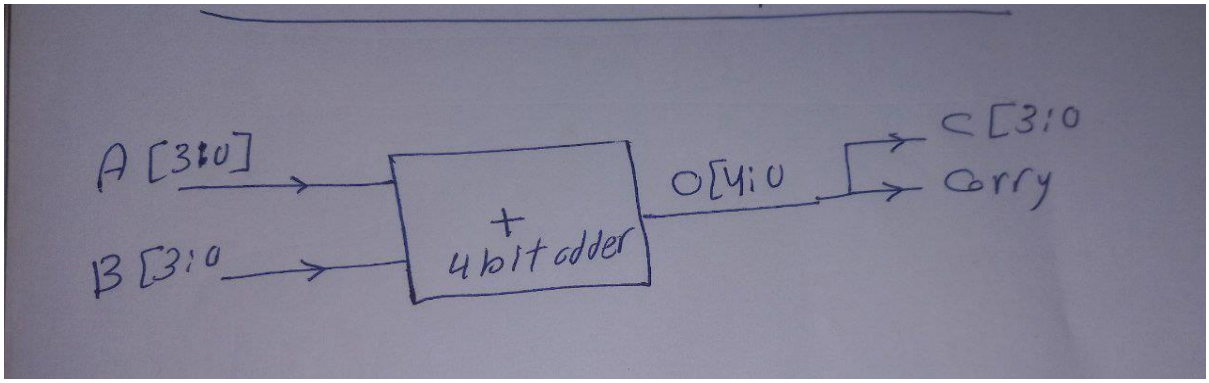
- **Q[3]**
  - Code:

```
oma karem > [not_extra]Assignmen_1 > Q3.v
module x4bit_adder(A,B,C,carry);
input [3:0] A,B;
output [3:0] C ;
output carry ;
assign {carry,C} = A + B ;
endmodule
```

  - Wave:



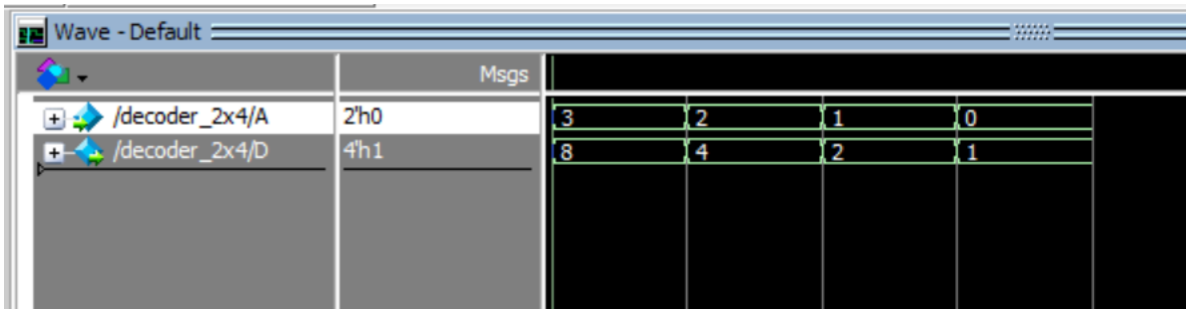
  - Schematic:



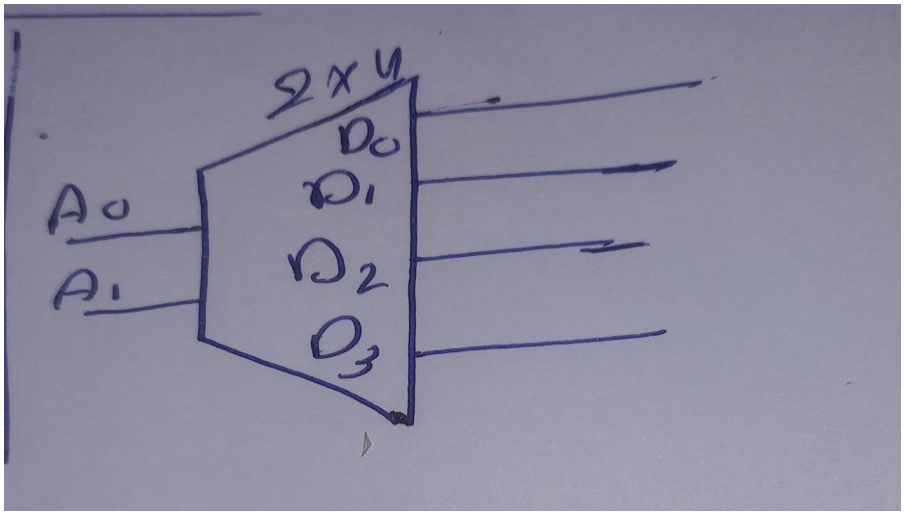
- **Q[4]**
  - Code:

```
1  module decoder_2x4(A,D);
2  input [1:0]A ;
3  output [3:0]D;
4  assign D = (A == 2'b00)? 4'b0001 :(A == 2'b01)? 4'b0010 :(A == 2'b10)? 4'b0100 : 4'b1000 ;
5  endmodule
```

  - Wave:



  - Schematic:

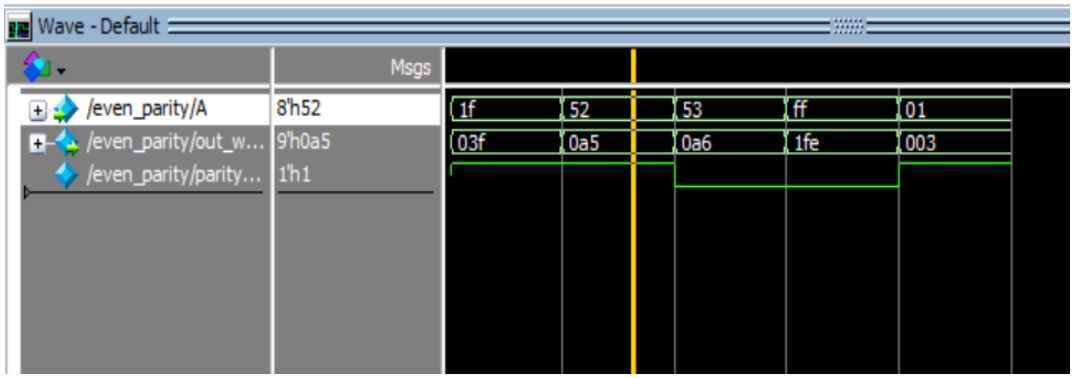


- **Q[5]**
  - Code:

E: > Deploma karem > [not_extra]Assignmen_1 > Q5.v

```
1  module even_parity(A, out_with_parity );
2  input[7:0] A ;
3  output [8:0]  out_with_parity ;
4  wire parity_bit;
5  assign parity_bit = ^ A;
6  assign out_with_parity ={A,parity_bit};
7  endmodule
```

  - Wave:
    - Even Parity If number of 1s is even, parity bit value is 0. If number of 1s is odd, parity bit value is 1.



- **Q[6]**

➢Code:

```
module seven_segment(A,B,opcode,Result,a,b,c,d,e,f,g,enable);
parameter N =4 ;
input [N-1:0] A ,B ;
input [N-3:0] opcode ;
input enable ;
output reg [N-1:0] Result ;
output reg a,b,c,d,e,f,g;
always @(*) begin
    // ALU operation
    case (opcode)
        2'b00: Result = A + B ;
        2'b01: Result = A | B ;
        2'b10: Result = A - B ;
        2'b11: Result = A ^ B ;
    endcase
// 7_segment
if (enable == 1'b1) begin
    case ( Result)
        4'b0000:begin a=1'b1;b= 1'b1;c=1'b1;d=1'b1;e=1'b1;f=1'b1;g=1'b0;end
        4'b0001:begin a=1'b0;b= 1'b1;c=1'b1;d=1'b0;e=1'b0;f=1'b0;g=1'b0;end
        4'b0010:begin a=1'b1;b= 1'b1;c=1'b0;d=1'b1;e=1'b1;f=1'b0;g=1'b1;end
        4'b0011:begin a=1'b1;b= 1'b1;c=1'b1;d=1'b1;e=1'b0;f=1'b0;g=1'b1;end
        4'b0100:begin a=1'b0;b= 1'b1;c=1'b1;d=1'b0;e=1'b0;f=1'b1;g=1'b1;end
        4'b0101:begin a=1'b1;b= 1'b0;c=1'b1;d=1'b1;e=1'b0;f=1'b1;g=1'b1;end
        4'b0110:begin a=1'b1;b= 1'b0;c=1'b1;d=1'b1;e=1'b1;f=1'b1;g=1'b1;end
        4'b0111:begin a=1'b1;b= 1'b1;c=1'b1;d=1'b0;e=1'b0;f=1'b0;g=1'b0;end
        4'b1000:begin a=1'b1;b= 1'b1;c=1'b1;d=1'b1;e=1'b1;f=1'b1;g=1'b1;end
        4'b1001:begin a=1'b1;b= 1'b1;c=1'b1;d=1'b1;e=1'b0;f=1'b1;g=1'b1;end
        4'b1010:begin a=1'b1;b= 1'b1;c=1'b1;d=1'b0;e=1'b1;f=1'b1;g=1'b1;end
        4'b1011:begin a=1'b0;b= 1'b0;c=1'b1;d=1'b1;e=1'b1;f=1'b1;g=1'b1;end
        4'b1100:begin a=1'b1;b= 1'b0;c=1'b0;d=1'b1;e=1'b1;f=1'b1;g=1'b0;end
        4'b1101:begin a=1'b0;b= 1'b1;c=1'b1;d=1'b1;e=1'b1;f=1'b0;g=1'b1;end
        4'b1110:begin a=1'b1;b= 1'b0;c=1'b0;d=1'b1;e=1'b1;f=1'b1;g=1'b1;end
        4'b1111:begin a=1'b1;b= 1'b0;c=1'b0;d=1'b0;e=1'b1;f=1'b1;g=1'b1;end
    endcase
    end
    else begin
      a=1'b0;b= 1'b0;c=1'b0;d=1'b0;e=1'b0;f=1'b0;g=1'b0;
    end
end
endmodule
```

➢Wave: