Name: Ariel Ohayon ID: #206912354

Verilog – Work 1

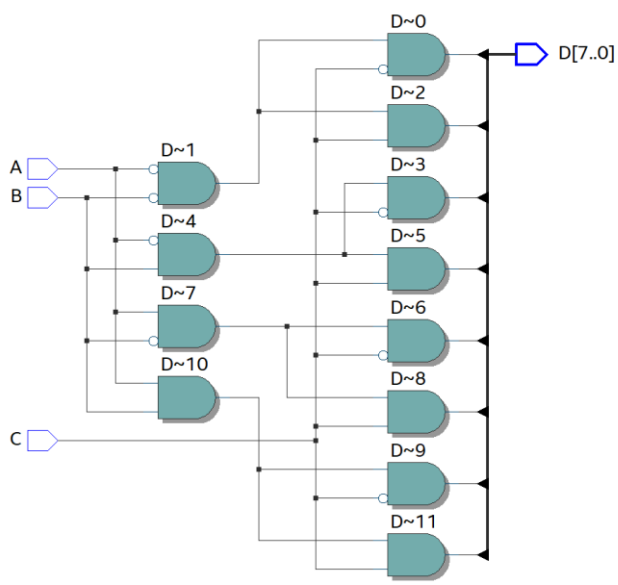
Part A

In this exercise we need to design a decoder 3x8 (3 – inputs, 8(=2^3) - outputs).

1. Dataflow modeling design – The design contents series of assigns instructions that translated to logic gates directly by the bitwise operands.
   The design of decoder3x8 in Verilog:

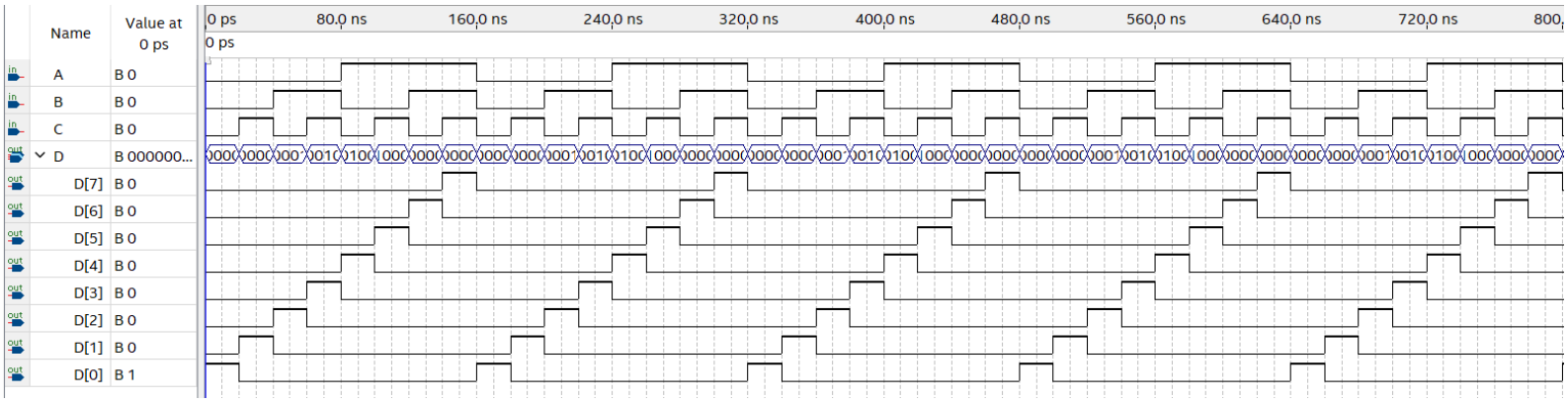
```
// Dataflow modeling Design

module Decoder_3x8(D,A,B,C);
output  [7:0] D;
input                 A,B,C;

assign D[0] = (~A) & (~B) & (~C);       // 000
assign D[1] = (~A) & (~B) &  (C);       // 001
assign D[2] = (~A) &  (B) & (~C);       // 010
assign D[3] = (~A) &  (B) &  (C);       // 011
assign D[4] =  (A) & (~B) & (~C);       // 100
assign D[5] =  (A) & (~B) &  (C);       // 101
assign D[6] =  (A) &  (B) & (~C);       // 110
assign D[7] =  (A) &  (B) &  (C);       // 111

endmodule
```

RTL:



Name: Ariel Ohayon ID: #206912354

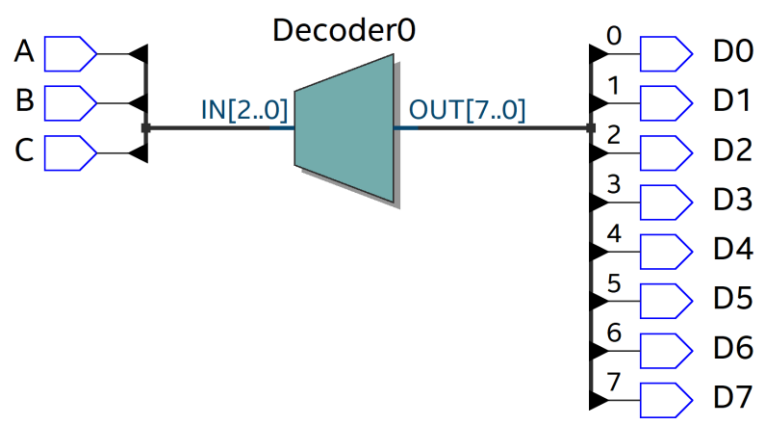
Waveform signals:



As we can see in the waveform we get the correct signals for all of the possible states of the component.

2. In this exercise we need to design the same component with the behavioral modeling method.
   - The first behavioral model design we need to design the outputs of the component define as single outputs (and not as [7:0] output signal).
     In addition we were required to design the component with case statements.
     The behavioral modeling must include sensitivity list (always @(…)).
     The design in Verilog in behavioral modeling:

```
//Behaviorial modeling Design.
module Decoder3x8_ver1_Behave (D0,D1,D2,D3,D4,D5,D6,D7,A,B,C);
output reg D0,D1,D2,D3,D4,D5,D6,D7;
input  A,B,C;    //A - MSB, C - LSB
reg [2:0] in;
always @ (A,B,C)
begin
in = {A,B,C};
D0 = 0;
D1 = 0;
D2 = 0;
D3 = 0;
D4 = 0;
D5 = 0;
D6 = 0;
D7 = 0;
case (in)
3'b000: D0 = 1'b1;
3'b001: D1 = 1'b1;
3'b010: D2 = 1'b1;
3'b011: D3 = 1'b1;
3'b100: D4 = 1'b1;
3'b101: D5 = 1'b1;
3'b110: D6 = 1'b1;
3'b111: D7 = 1'b1;
endcase
end
endmodule
```



The waveform I get in this design is the same waveform that I get in the dataflow design, the only change is in the definition of the output signal.

- The second behavioral modeling based on single output that define as vector signal ([7:0] output).
  The second Verilog design:

```
module Decoder3x8_ver2_Behave (out,in);

output reg [7:0] out;
input  [2:0] in ;

       always @(in)
       begin
              case (in)
                     3'b000: out = 8'b00000001;
                     3'b001: out = 8'b00000010;
                     3'b010: out = 8'b00000100;
                     3'b011: out = 8'b00001000;
                     3'b100: out = 8'b00010000;
                     3'b101: out = 8'b00100000;
                     3'b110: out = 8'b01000000;
                     3'b111: out = 8'b10000000;
              endcase
       end

endmodule
```

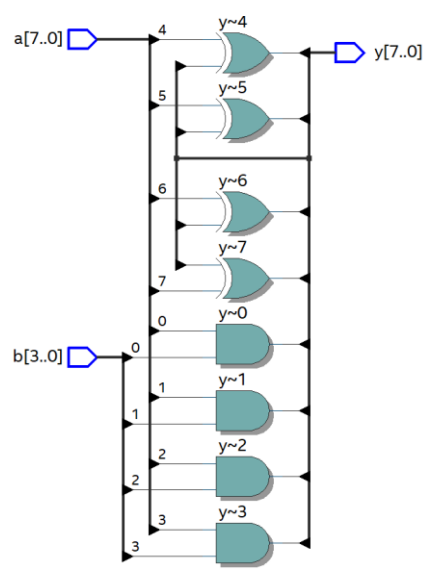
Also here, I got the same RTL as the previous behavioral design and the same waveform.

Part B

In this Part I required to design a basic logic component.
The design of this system is need to performed by using the operators only on parts of the vector input and output signals.

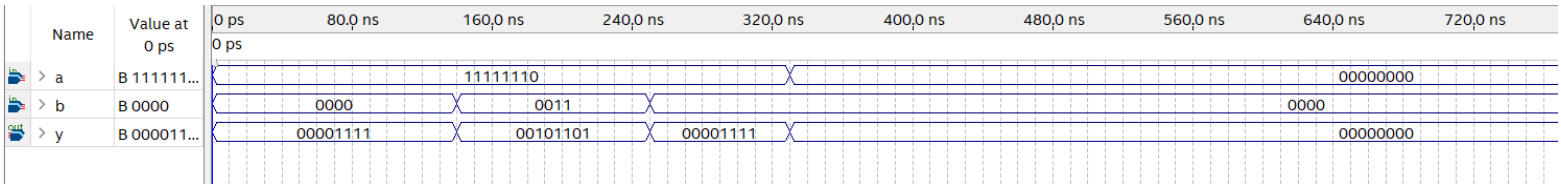
The Verilog design:

```
module sys (y,a,b);

output [7:0] y;
input  [7:0] a;
input  [3:0] b;

assign y[7:4] = a[3:0] & b;
assign y[3:0] = a[7:4] ^ y[7:4];
endmodule
```



Name: Ariel Ohayon ID: #206912354

Waveform:



The results show in the figure is check for only a few states of the system (not full testbench).

In the first state:
input a = 11111110,
input b = 0000
and we got that the output y = 00001111.
This state is correct because: b = 0000 so the output signals of the and gates have to be ‘0’ (the MSB nibble of y), because of that we get in the first input of the xor gate (by the feedback) zeros and the second input of the xor gates come from the MSB nibble of the input x (“1111”), and as we can see in the figure the results are really the results of the xor gates (0 xor 1 = 1).
And as you can see the results in all of the 4 states (by the same calculation) are corresponding to the requirements.

GITHUB URL: https://github.com/Ariel-Ohayon/HIT-Verilog/tree/main/Works/Work1