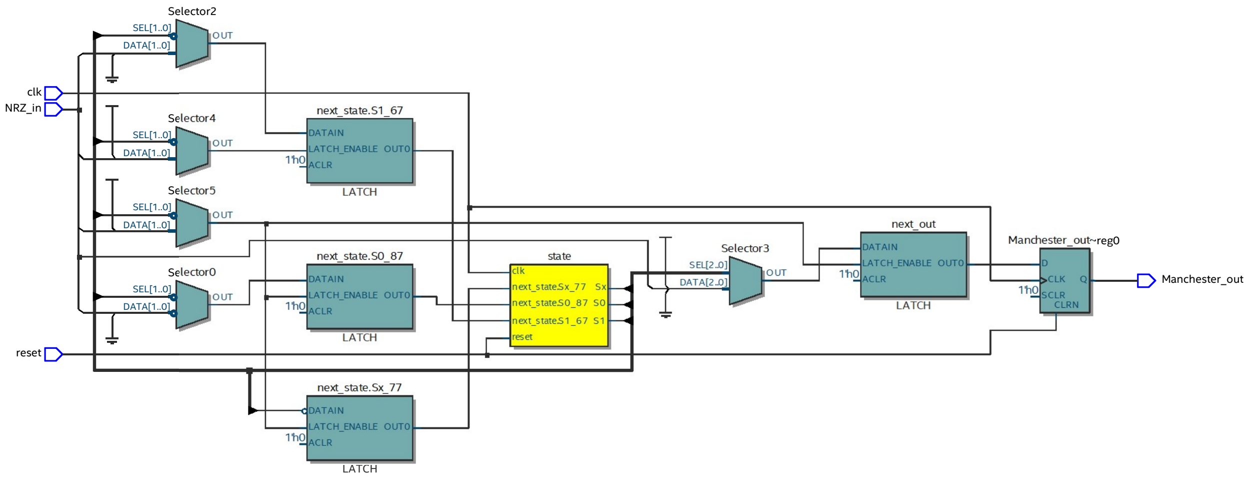
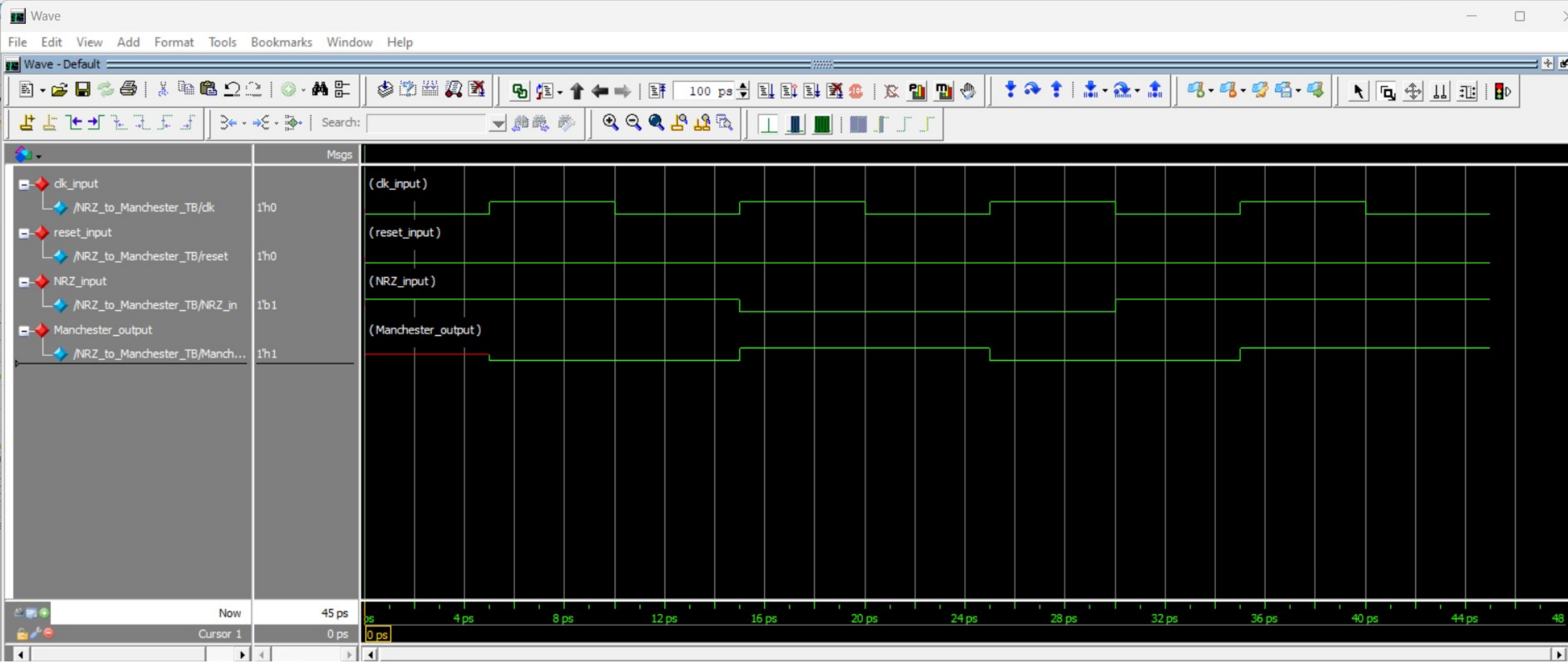
```
module NRZ_to_Manchester (Manchester_out,
                          NRZ_in,
                          clk,
                          reset
                          );

output Manchester_out;
input wire NRZ_in, clk, reset;

reg [1:0] state, next_state;
reg     next_out, Manchester_out;      //to assign values within always block

parameter Sx = 2'b01;         //waiting for new NRZ input
parameter S0 = 2'b00;         //An NRZ 0 is being converted to 01
parameter S1 = 2'b11;         //An NRZ 1 is being converted to 10

//sequential logic updating the state
always @ (posedge clk or posedge reset)      //asynchronous reset
   if (reset) begin state <= Sx;
               Manchester_out <= 1'b0;
            end
   else       begin state <= next_state;
                Manchester_out <= next_out;
            end

//combinational logic to find the next_state and the Manchester_out
always @ *           //if the state or the NRZ_in change
   case (state)
      Sx: if (NRZ_in) begin
                     next_state = S1;
                     next_out   = 1'b1;
                  end
         else     begin
                     next_state = S0;
                     next_out   = 1'b0;
                  end
      S0: if (NRZ_in) begin
                     next_state = Sx;        //NRZ_in has to be 0
                     next_out   = 1'b1;
                  end
      S1: if (NRZ_in) begin
                     next_state = Sx;        //NRZ_in has to be 1
                     next_out   = 1'b0;
                  end
             default:  begin
                     next_state = Sx;
                     next_out   = 1'b0;
                  end
   endcase
endmodule
```

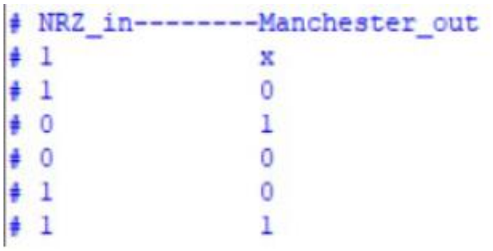

Selector2
SEL[1..0]
DATA[1..0]
OUT
clk
NRZ_in
next_state.S1_67
DATAIN
LATCH_ENABLE OUT0
1'h0
ACLR
LATCH
Selector4
SEL[1..0]
DATA[1..0]
OUT
Selector5
SEL[1..0]
DATA[1..0]
OUT
next_state.S0_87
DATAIN
LATCH_ENABLE OUT0
1'h0
ACLR
LATCH
Selector0
SEL[1..0]
DATA[1..0]
OUT
state
clk
next_state.Sx_77 Sx
next_state.S0_87 S0
next_state.S1_67 S1
reset
Selector3
SEL[2..0]
DATA[2..0]
OUT
next_out
DATAIN
LATCH_ENABLE OUT0
1'h0
ACLR
LATCH
Manchester_out~reg0
D
CLK
Q
1'h0
SCLR
CLRN
Manchester_out
reset
next_state.Sx_77
DATAIN
LATCH_ENABLE OUT0
1'h0
ACLR
LATCH

```
module NRZ_to_Manchester_TB ();

//define the registers and wires
reg clk, reset;
reg NRZ_in;

wire Manchester_out;

//define the unit under test UUT
NRZ_to_Manchester UUT (
                       .Manchester_out(Manchester_out),
                       .NRZ_in(NRZ_in),
                       .clk(clk),
                       .reset(reset)
                      );

//instantiate the clk signal
initial
   begin
      clk = 1'b0;
      forever #5 clk = ~clk;     //10ns clk period
   end

//instantiate the reset signal
initial
   begin
           reset = 1'b0;       //togel the reset signal on
           NRZ_in  = 2'b01;
      #100 reset = 1'b1;      //toggle the reset signal off
   end

//instantiate all the posibble states for PAM4 with time intervals
initial
   begin
      NRZ_in = 2'b01; #15;
      NRZ_in = 2'b00; #15;
      NRZ_in = 2'b11; #15;

      $stop;
   end

//display the results
initial begin
  $display("NRZ_in--------Manchester_out");
  $monitor("%b    %b ",NRZ_in, Manchester_out);
 end
endmodule
```



```
clk_input
  /NRZ_to_Manchester_TB/clk           1'h0
reset_input
  /NRZ_to_Manchester_TB/reset         1'h0
NRZ_input
  /NRZ_to_Manchester_TB/NRZ_in        1'b1
Manchester_output
  /NRZ_to_Manchester_TB/Manch...      1'h1

Now        45 ps
Cursor 1    0 ps      0 ps
```

Output table



```
# NRZ_in--------Manchester_out
# 1             x
# 1             0
# 0             1
# 0             0
# 1             0
# 1             1
```

NRZ to Manchester Summary

This portion takes in NRZ data and outputs Manchester code