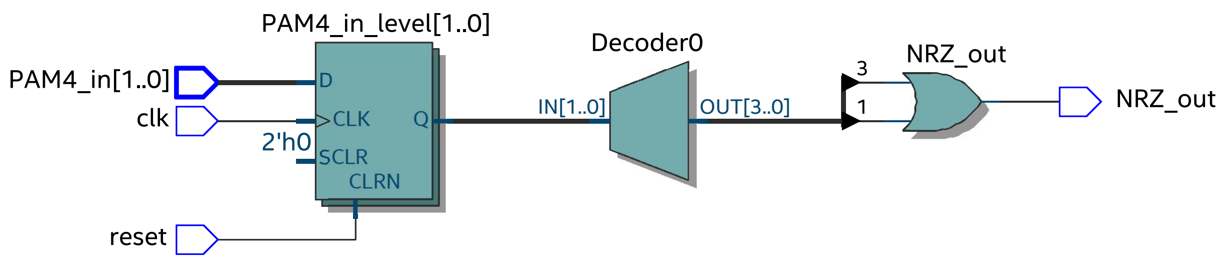
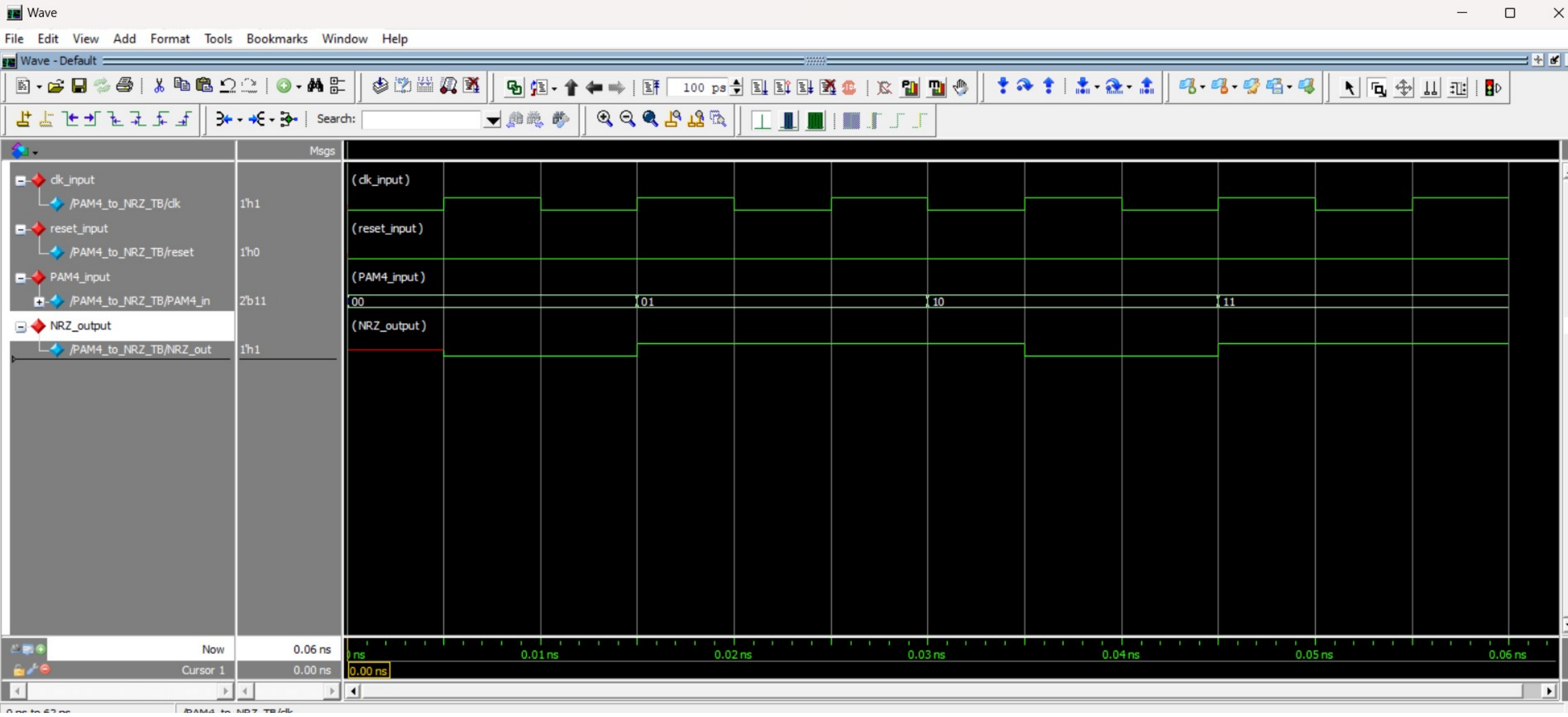
```
/*---------------------------------------------------------------------------------
Name Lamin Jammeh
CLass: EE417 Summer 2024
Lesson 04 HW Question 5 Q2 Part1
Group: Ron Kalin/ Lamin Jammeh
Project Description: the portion takes in PAM4_in with 2 bits input and 4 different
levels and converts it to NRZ
---------------------------------------------------------------------------------*/

module PAM4_to_NRZ (
                     NRZ_out,
                     PAM4_in,
                     clk,
                     reset
                     );
//assigen the inputs and putputs as registers and wires
input wire clk, reset;           //clk and reset signal
input wire [1:0] PAM4_in;           //2 bits PAM4_in input signal
output reg NRZ_out;

//4 different PAM4_in levels
reg [1:0] PAM4_in_level;

//diffrent PAM4_in states
parameter S0 = 2'b00;
parameter S1 = 2'b01;
parameter S2 = 2'b10;
parameter S3 = 2'b11;

//sequential logic updating the state
always @ (posedge clk or posedge reset)      //asynchronous reset
   if (reset) PAM4_in_level <= S0;
   else       PAM4_in_level <= PAM4_in;

 // Next state logic and output logic
    always @(*) begin
        case (PAM4_in_level)
            S0:   NRZ_out = 1'b0;
            S1:   NRZ_out = 1'b1;
            S2:   NRZ_out = 1'b0;
            S3:   NRZ_out = 1'b1;
        endcase
    end

endmodule
```

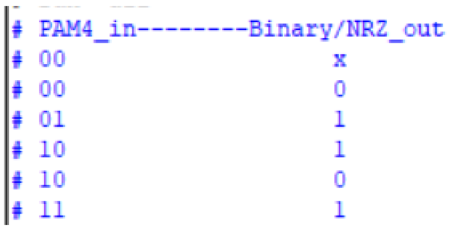

PAM4_in_level[1..0]
PAM4_in[1..0]
clk
reset
D
CLK
Q
2'h0
SCLR
CLRN
Decoder0
IN[1..0]
OUT[3..0]
3
1
NRZ_out
NRZ_out

```verilog
/*--------------------------------------------------------------------------------
Name Lamin Jammeh
CLass: EE417 Summer 2024
Lesson 04 HW Question 5 Q2 Part1
Group: Ron Kalin/ Lamin Jammeh
Project Description: test bench for part1
--------------------------------------------------------------------------------*/

module PAM4_to_NRZ_TB ();

//define the registers and wires
reg clk, reset;
reg [1:0] PAM4_in;
wire NRZ_out;

//define the unit under test UUT
PAM4_to_NRZ UUT (
                 .NRZ_out(NRZ_out),
                 .PAM4_in(PAM4_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
          PAM4_in  = 2'b00;
      #100 reset = 1'b1;      //toggle the reset signal off
   end

//instantiate all the posibble states for PAM4_in with time intervals
initial
   begin
      PAM4_in = 2'b00; #15;
      PAM4_in = 2'b01; #15;
      PAM4_in = 2'b10; #15;
      PAM4_in = 2'b11; #15;

      $stop;
   end

//display the results
initial begin
  $display("PAM4_in--------Binary/NRZ_out");
  $monitor("%b           %b ",PAM4_in, NRZ_out);
 end
endmodule
```



Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
100 ps
Search:
Msgs
clk_input
/PAM4_to_NRZ_TB/clk
1'h1
reset_input
/PAM4_to_NRZ_TB/reset
1'h0
PAM4_input
/PAM4_to_NRZ_TB/PAM4_in
2'b11
NRZ_output
/PAM4_to_NRZ_TB/NRZ_out
1'h1
(clk_input)
(reset_input)
(PAM4_input)
00
01
10
11
(NRZ_output)
Now
0.06 ns
Cursor 1
0.00 ns
0 ns
0.01 ns
0.02 ns
0.03 ns
0.04 ns
0.05 ns
0.06 ns
0.00 ns

Output Table

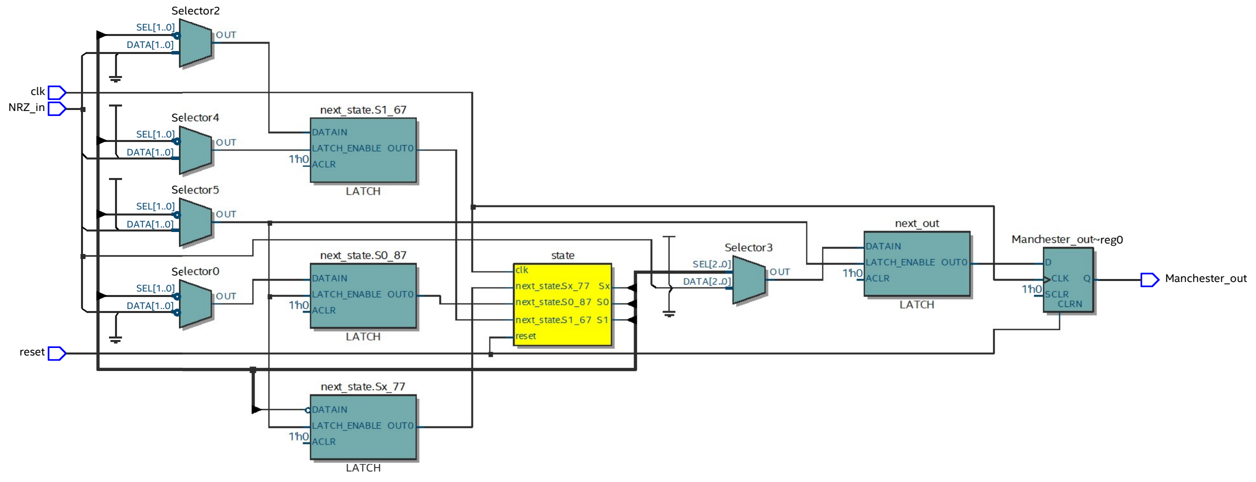


```
# PAM4_in--------Binary/NRZ_out
# 00                    x
# 00                    0
# 01                    1
# 10                    1
# 10                    0
# 11                    1
```

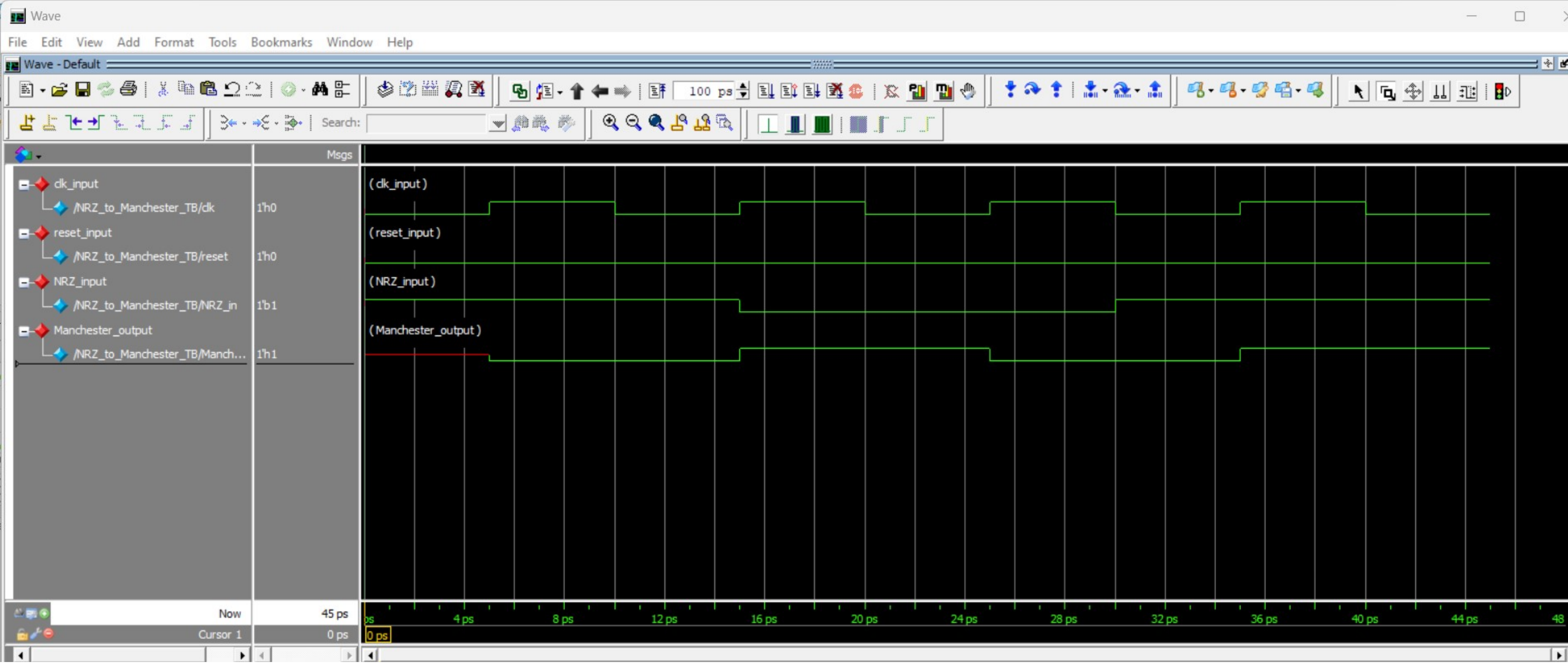
PAM4 to NRZ Summary

- This portion of the design takes in PAM4 data and converts it to NRZ data.
- The PAM4 has four levels of input S0, S1, S2, and S3 corresponding to [00,01,10,11]

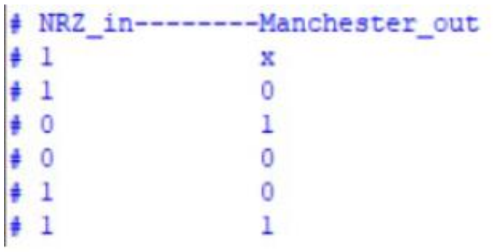
```verilog
module NRZ_to_Manchester (Manchester_out,
                          NRZ_in,
                          clk,
                          reset
                          );

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
Selector4
Selector5
Selector0
next_state.S1_67
DATAIN
LATCH_ENABLE OUT0
ACLR
1'h0
LATCH
next_state.S0_87
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]
next_out
Manchester_out~reg0
D
CLK
Q
SCLR
CLRN
Manchester_out
reset
next_state.Sx_77

```verilog
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
```


Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
Search:
Msgs
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
(clk_input)
(reset_input)
(NRZ_input)
(Manchester_output)
Now
45 ps
Cursor 1
0 ps
ps
4 ps
8 ps
12 ps
16 ps
20 ps
24 ps
28 ps
32 ps
36 ps
40 ps
44 ps
48
0 ps

Output table

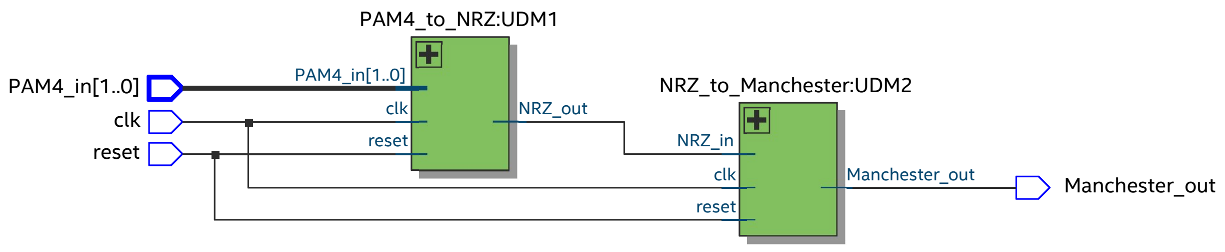


```
# NRZ_in--------Manchester_out
# 1             x
# 1             0
# 0             1
# 0             0
# 1             0
# 1             1
```

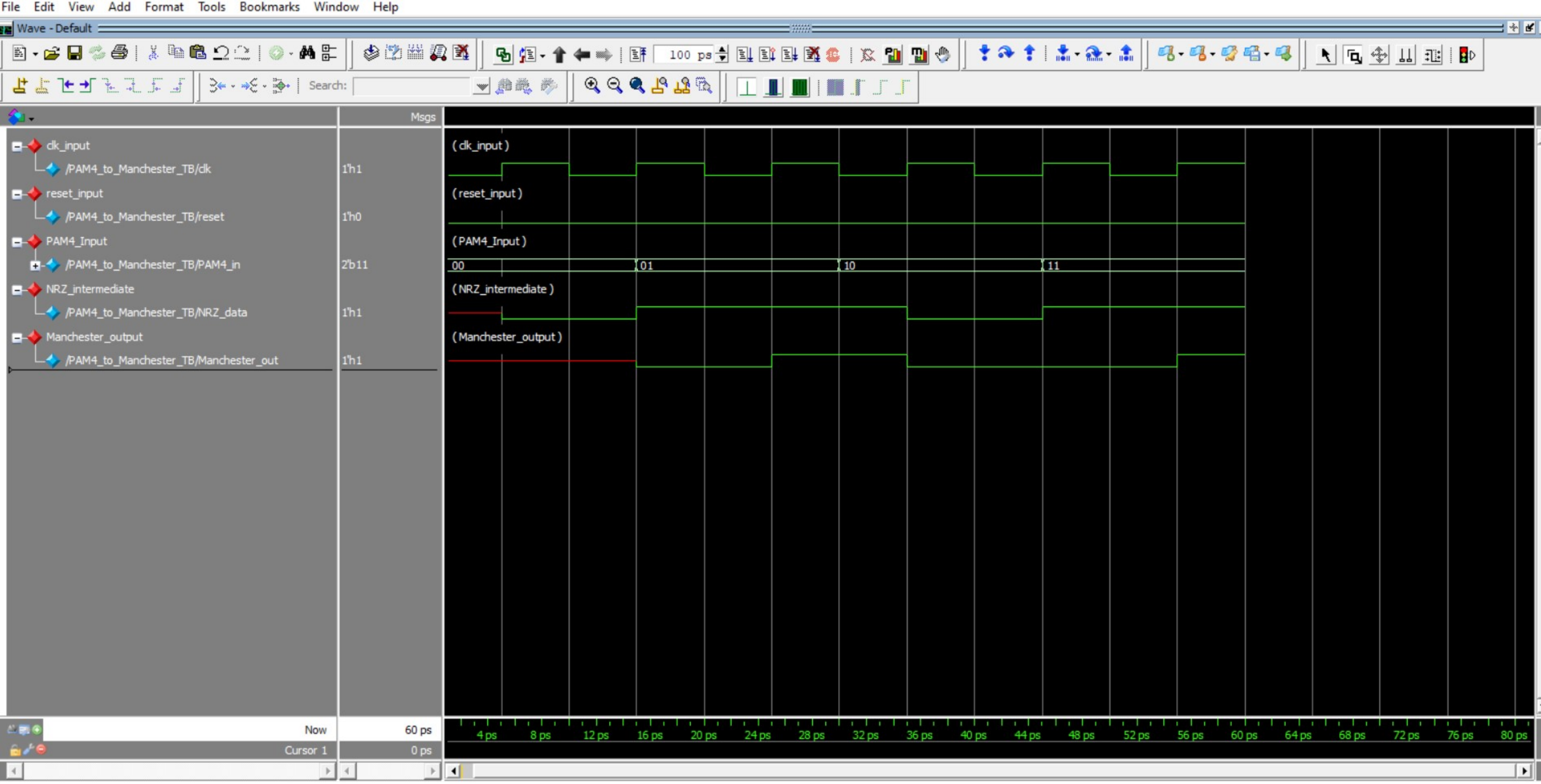
NRZ to Manchester Summary

This portion takes in NRZ data and outputs Manchester code

```verilog
module PAM4_to_Manchester (
                           Manchester_out,
                           PAM4_in,
                           clk,
                           reset
                           );

//assigen the inputs and outputs as registers and wires
input  wire       clk, reset;           //clk and reset signal
input  wire [1:0] PAM4_in;             //2 bits PAM4_in input signal
output wire       Manchester_out;      //declare as wire since it will be connected to module

//internal probe
wire NRZ_data;               //output signal from the PAM4_to NRZ module

//instantiate the PAM4-NRZ and NRZ-Manchester modules as user define modules (UDM1 and UDM2)
PAM4_to_NRZ UDM1 (
                  .NRZ_out(NRZ_data),                          //note NRZ_out becomes 
NRZ_data
                  .PAM4_in(PAM4_in),
                  .clk(clk),
                  .reset(reset)
                 );

NRZ_to_Manchester UDM2 (
                        .Manchester_out(Manchester_out),
                        .NRZ_in(NRZ_data),                     //note NRZ_in becomes NRZ_data
                        .clk(clk),
                        .reset(reset)
                       );

endmodule
```

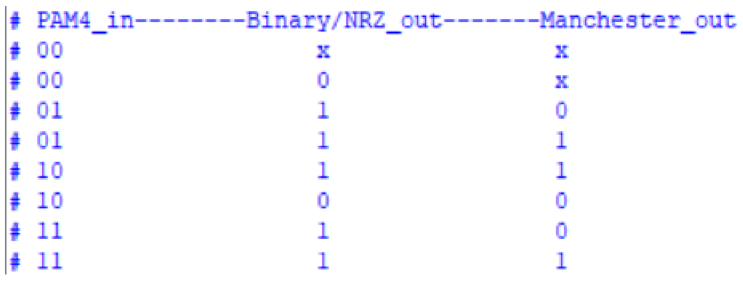

PAM4_to_NRZ:UDM1
PAM4_in[1..0]
clk
reset
PAM4_in[1..0]
clk
reset
NRZ_out
NRZ_to_Manchester:UDM2
NRZ_in
clk
reset
Manchester_out
Manchester_out

```verilog
module PAM4_to_Manchester_TB ();

//define the registers=wire and wires=registers from desing-to-testbench
reg       clk, reset;              //clk and reset signal
reg [1:0] PAM4_in;                //2 bits PAM4_in input signal
wire       Manchester_out;

//define the unit under test UUT
PAM4_to_Manchester UUT (
                        .Manchester_out(Manchester_out),
                        .PAM4_in(PAM4_in),
                        .clk(clk),
                        .reset(reset)
                        );

//monitor internal probe
assign NRZ_data = UUT.UDM1.NRZ_out;

//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
        PAM4_in  = 2'b00;
      #100 reset = 1'b1;      //toggle the reset signal off
   end

//instantiate all the posibble states for PAM4_in with time intervals
initial
   begin
      PAM4_in = 2'b00; #15;
      PAM4_in = 2'b01; #15;
      PAM4_in = 2'b10; #15;
      PAM4_in = 2'b11; #15;

      $stop;
   end

//display the results
initial begin
  $display("PAM4_in--------Binary/NRZ_out-------Manchester_out");
  $monitor("%b          %b           %b ",PAM4_in, NRZ_data,
Manchester_out);
 end
endmodule
```



```
clk_input
  /PAM4_to_Manchester_TB/clk               1'h1
reset_input
  /PAM4_to_Manchester_TB/reset             1'h0
PAM4_Input
  /PAM4_to_Manchester_TB/PAM4_in           2'b11
NRZ_intermediate
  /PAM4_to_Manchester_TB/NRZ_data          1'h1
Manchester_output
  /PAM4_to_Manchester_TB/Manchester_out    1'h1

Now      60 ps
Cursor 1  0 ps
```

Output table



```
# PAM4_in--------Binary/NRZ_out-------Manchester_out
# 00                  x                 x
# 00                  0                 x
# 01                  1                 0
# 01                  1                 1
# 10                  1                 1
# 10                  0                 0
# 11                  1                 0
# 11                  1                 1
```

PAM4 to Manchester out

- This module instantiates 2 different modules
- The final modules take PAM4 data converts the data to NRZ
- The NRZ data is converter to Manchester code as final output