

CPE 322: SIM 1

NAME: Dan Otieno.

SECTION #: 01.

DUE DATE: 03/25/2022.

DESCRIPTION:

This assignment required simulating the logic designs in Homework 1 using Questa and Verilog. The 3 different simulation types included are behavioral implementations of the Mealy and Moore models, along with a structural implementation of the Mealy model.

The testbench file that was used to simulate outputs for all implementations is below:

Testbench Code:

```
// Testbench for subtractor.v
`timescale 1ns/100ps
module tb_subtractormealy;

reg a=0, b=0, clk=1,en=0,reset=0;
wire d;

initial
begin

    // Not testing the en input in this example
    // setting it to 1 so it responds to each
    // positive edge
    en = 1;

    /* perform subtraction 0110 - 0101
    Result = 0001*/

    // reset back to state 0
    // time = 0ns
    reset = 1;
        a = 0;
        b = 0;

    // time = 10ns
    #10
    reset = 0;
        a = 0;        // LSB input, a = 0;
        b = 1;        // LSB input, b = 1;

    // time = 20ns
    #10
```

```

reset = 0;
a = 1;           // Next input, a = 1;
                b = 0;           // Next input, b = 0;

// time = 30ns
#10
reset = 0;
a = 1;           // Next input, a = 1;
                b = 1;           // Next input, b = 1;

// time = 40ns
#10
reset = 0;
a = 0;           // MSB input, a = 0;
                b = 0;           // MSB input, b = 0;

/*Subtract 0110_1100 from 1010_0110 (8-bit binary subtraction)
Result = 0011_1010*/

// reset back to state 0
// time = 50ns
#10
reset = 1;
a = 0;
                b = 0;

// time = 60ns
#10
reset = 0;
a = 0;           // LSB input, a = 0;
                b = 0;           // LSB input, b = 0;

// time = 70ns
#10
reset = 0;
a = 1;           // Next input, a = 1;
                b = 0;           // Next input, b = 0;

// time = 80ns
#10
reset = 0;
a = 1;           // Next input, a = 1;
                b = 1;           // Next input, b = 1;

```

```
// time = 90ns
#10
reset = 0;
a = 0;          // MSB input, a = 0;
                b = 1;          // MSB input, b = 1;
```

```
// time = 100ns
#10
reset = 0;
a = 0;          // Next input, a = 0;
                b = 0;          // Next input, b = 0;
```

```
// time = 110ns
#10
reset = 0;
a = 1;          // Next input, a = 1;
                b = 1;          // Next input, b = 1;
```

```
// time = 120ns
#10
reset = 0;
a = 0;          // Next input, a = 0;
                b = 1;          // Next input, b = 1;
```

```
// time = 130ns
#10
reset = 0;
a = 1;          // MSB input, a = 1;
                b = 0;          // MSB input, b = 0;
```

```
end
```

```
// set up a free running clock with period 10 ns
always
begin
    clk = #5 ~clk;
end
```

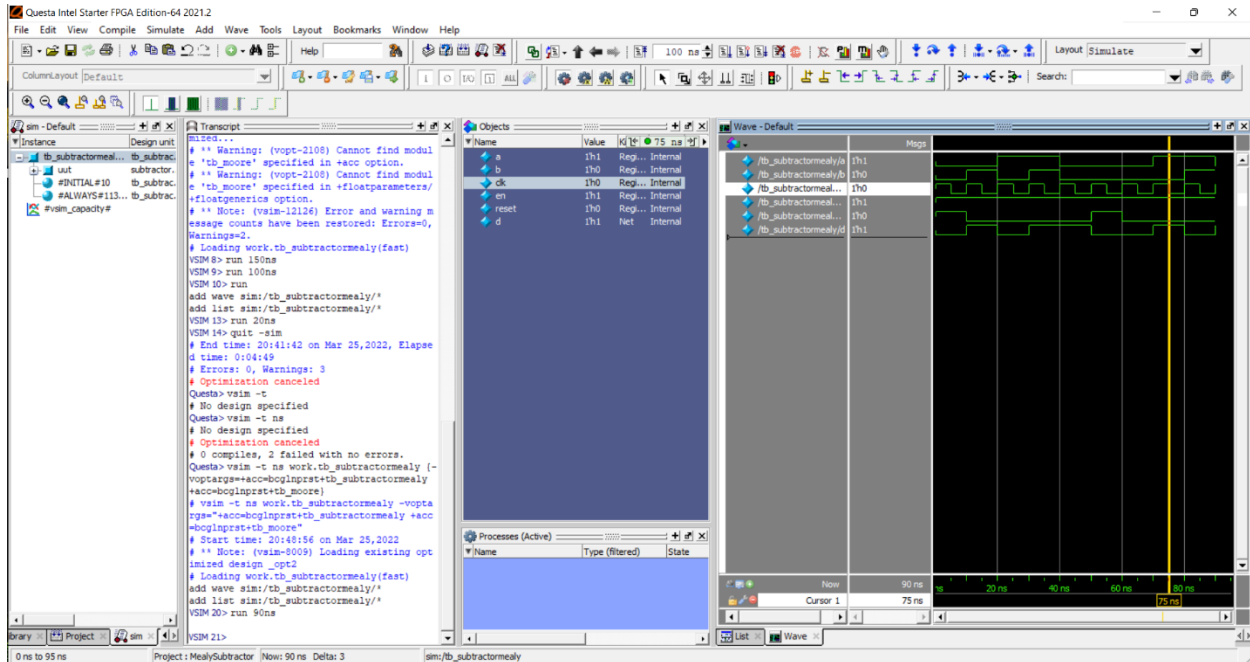
```
// instantiate the negator as the unit under test (uut), this part was modified per question.
subtractormealy uut (.A(a),.B(b),.CLK(clk),.En(en),.Reset(reset),.D(d));
```

```
endmodule
```

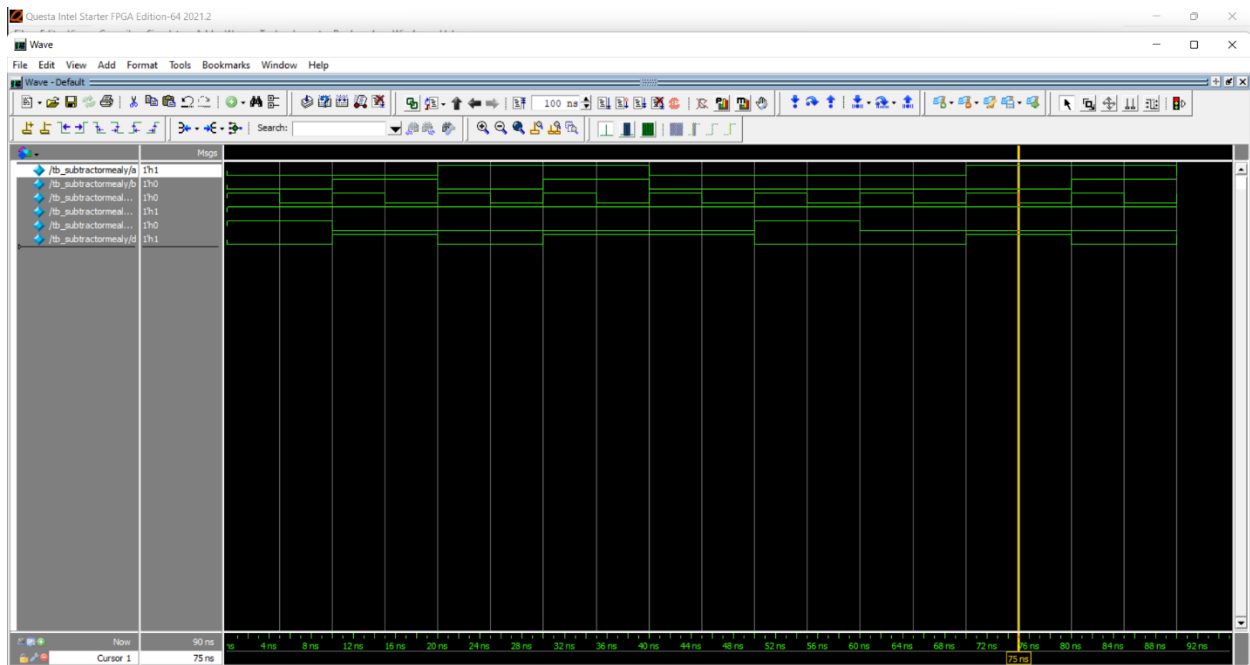
PART 1: BEHAVIORAL SIMULATION FOR MEALY USING MODELSIM (QUESTA).

Below are the codes and screenshots of the simulation in Questa.

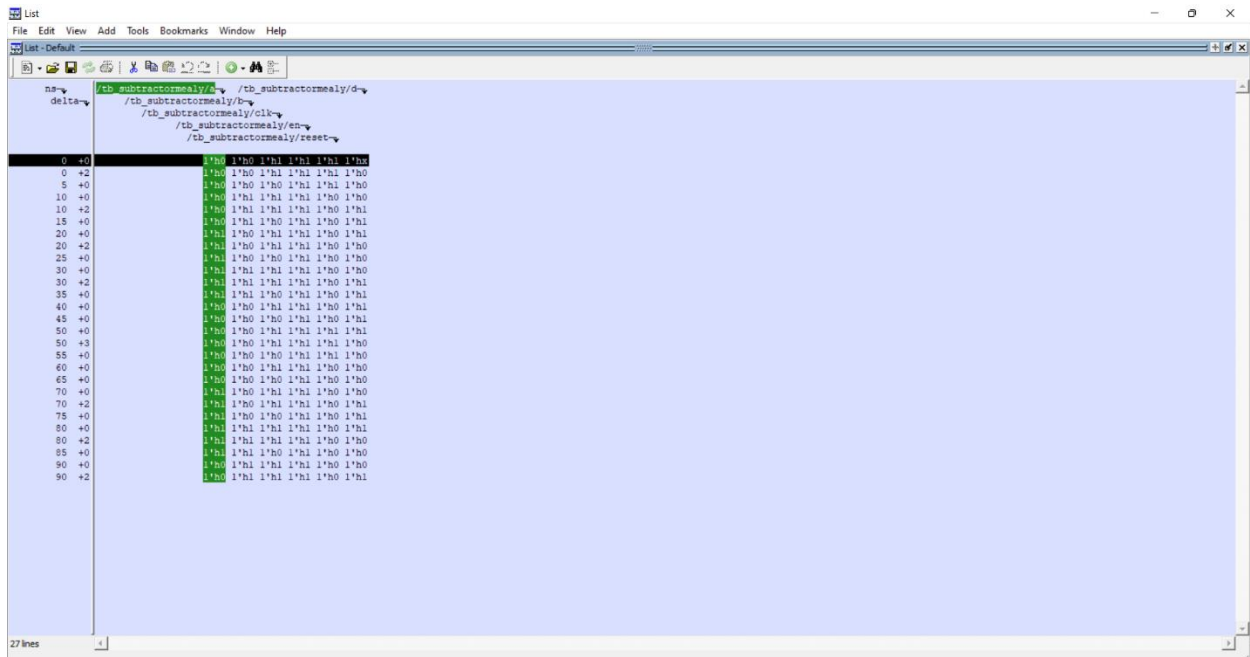
Questa Transcript Window (Part 1):



Mealy Behavioral waveform:



Mealy Behavioral List:



Mealy Behavioral Implementation Verilog Source code:

```
/* This is a behavioral model of a Mealy state machine.
```

```
- Dan Otieno
```

```
*/
```

```
module subtractormealy(input A, B, CLK, En, Reset, output reg D);
```

```
    reg [2:0] State=0, Nextstate=0;
```

```
    always @(State, A, B)          // Combinational Circuit
```

```
        case(State)
```

```
            0 : if(A == 0 && B == 0) begin Nextstate = 0;D = 0;          end
```

```
                else if(A == 0 && B == 1) begin Nextstate = 1;D = 1; end
```

```
                else if(A == 1 && B == 0) begin Nextstate = 0;D = 1; end
```

```
                else if(A == 1 && B == 1) begin Nextstate = 0;D = 0; end
```

```
            1 : if(A == 0 && B == 0) begin Nextstate = 0;D = 1;          end
```

```
                else if(A == 0 && B == 1) begin Nextstate = 1;D = 0; end
```

```
                else if(A == 1 && B == 0) begin Nextstate = 1;D = 0; end
```

```
                else if(A == 1 && B == 1) begin Nextstate = 1;D = 1; end
```

```
            default : begin Nextstate = 0; D = 0; end
```

```
        endcase
```

```
    always @(posedge CLK) // State Register portion of design.
```

```
        if (Reset)
```

```
            State = 0; // synchronous Reset
```

```
        else if (En)
```

```
            State = Nextstate;
```

```
endmodule
```

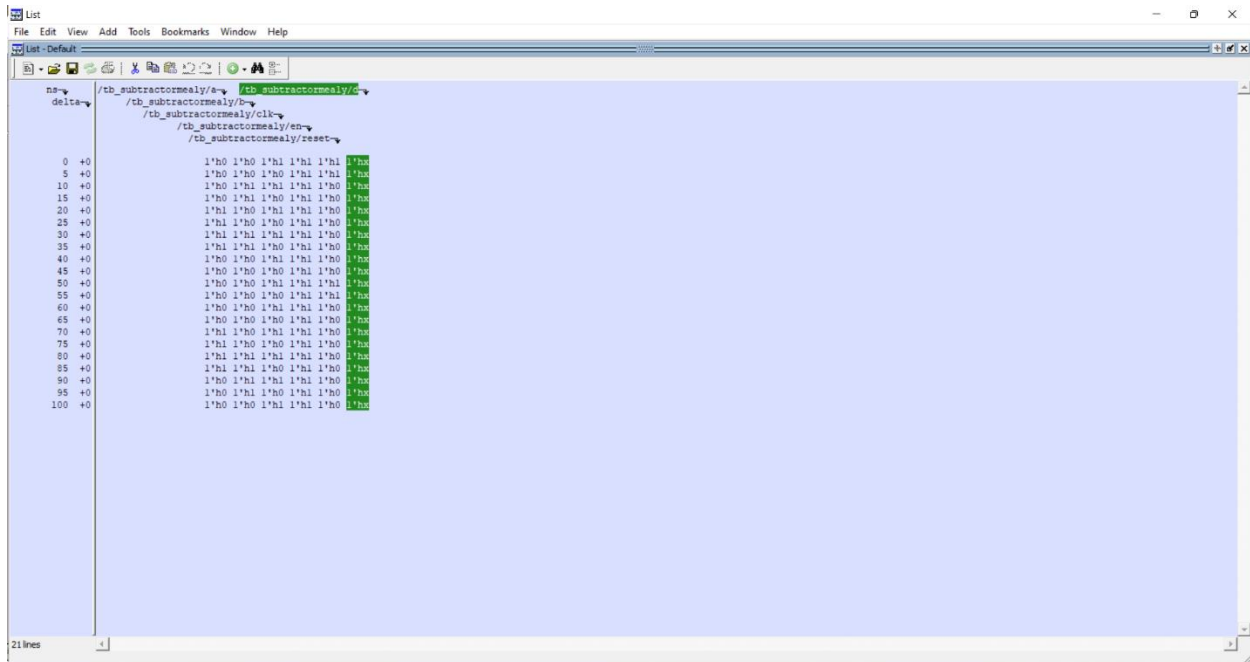
PART 2: MEALY STRUCTURAL IMPLEMENTATION SIMULATION.

The structural implementation simulation screenshots and code are shown below:

The screenshot displays the Questa Intel Starter FPGA Edition-64 software interface. The main window is divided into several panes:

- Transcript Window:** Shows the simulation results, including the opening of the "tb_subtractormeanly.v" file, the compilation process, and the simulation execution. The transcript indicates that the simulation was successful and that the design is being optimized.
- Wave Window:** Displays a timing diagram for the simulation. The diagram shows the signals over time, with a time scale ranging from 0 ns to 31 ns. The signals are labeled with names like "tb_subtractormeanly/a", "tb_subtractormeanly/b", "tb_subtractormeanly/c", "tb_subtractormeanly/d", "tb_subtractormeanly/en", "tb_subtractormeanly/reset", and "tb_subtractormeanly/d".
- Objects Window:** Lists the objects in the simulation, including the design files and the simulation results.
- Wave-Default Window:** Shows the default wave settings for the simulation.

The interface also includes a menu bar at the top with options like File, Edit, View, Compile, Simulate, Add, Wave, Tools, and Window. The bottom status bar shows the current time (0 ns to 31 ns) and the project name (Project: mealy_struct).



Mealy Structural Verilog Source code:

```

/* Dan Otieno.
- Structural Implementation of Mealy Subtractor.
- 03/20/22 */

module subtractormealy_struct(input      A, B, CLK, En, Reset, output D);
    wire A1,B1,A_N,B_N,C1,C2,C3,C4,C5,C6,D0,Q,Q_N;

    not    I1(A_N,A1);
    not    I2(B_N,B1);
    and    G1(C1,A_N,B1,Q_N);
    and    G2(C2,B_N,A1,Q_N);
    and    G3(C3,A_N,B_N,Q);
    and    G4(C4,B1,A1,Q);
    and    G5(C5,A1,Q);
    and    G6(C6,B1,A_N);
    or     G7(D0,C5,C6);
    or     G8(D,C1,C2,C3,C4);

    d_ff   FF1(CLK,D0,Q_N,Q);

endmodule

module d_ff(input clk, d0, output reg q_n, output q);

```

initial

```
q_n = 0;
```

always @(posedge clk)

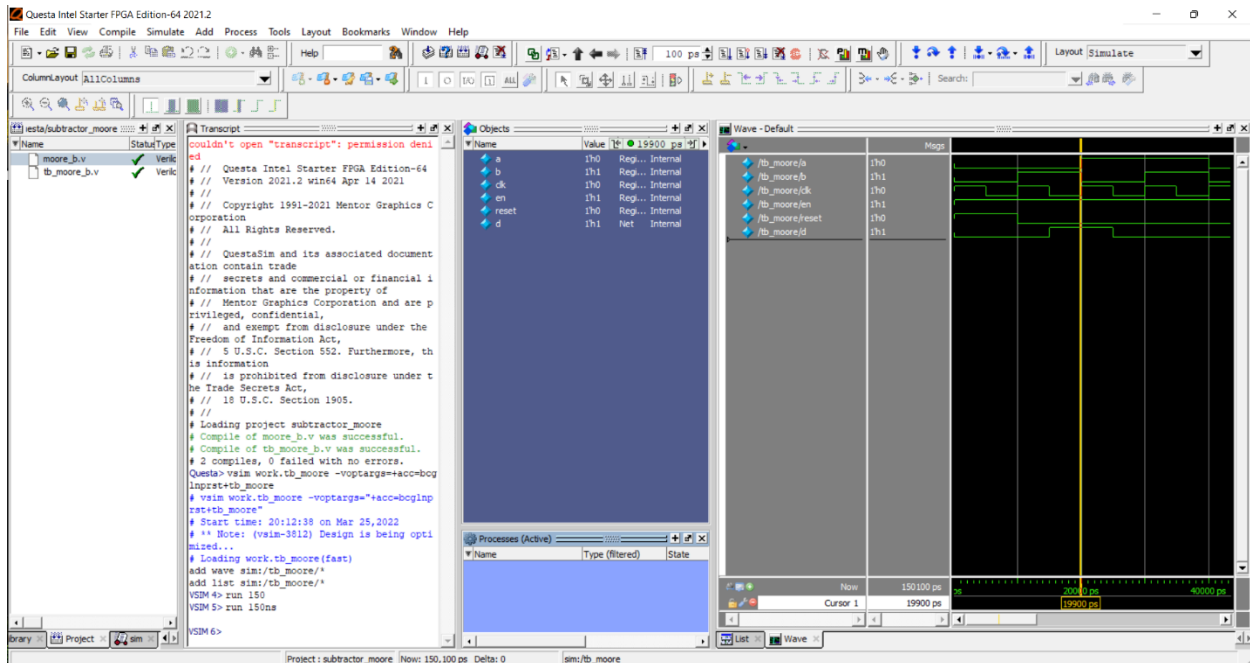
```
q_n = d0;
```

```
assign q = ~q_n;
```

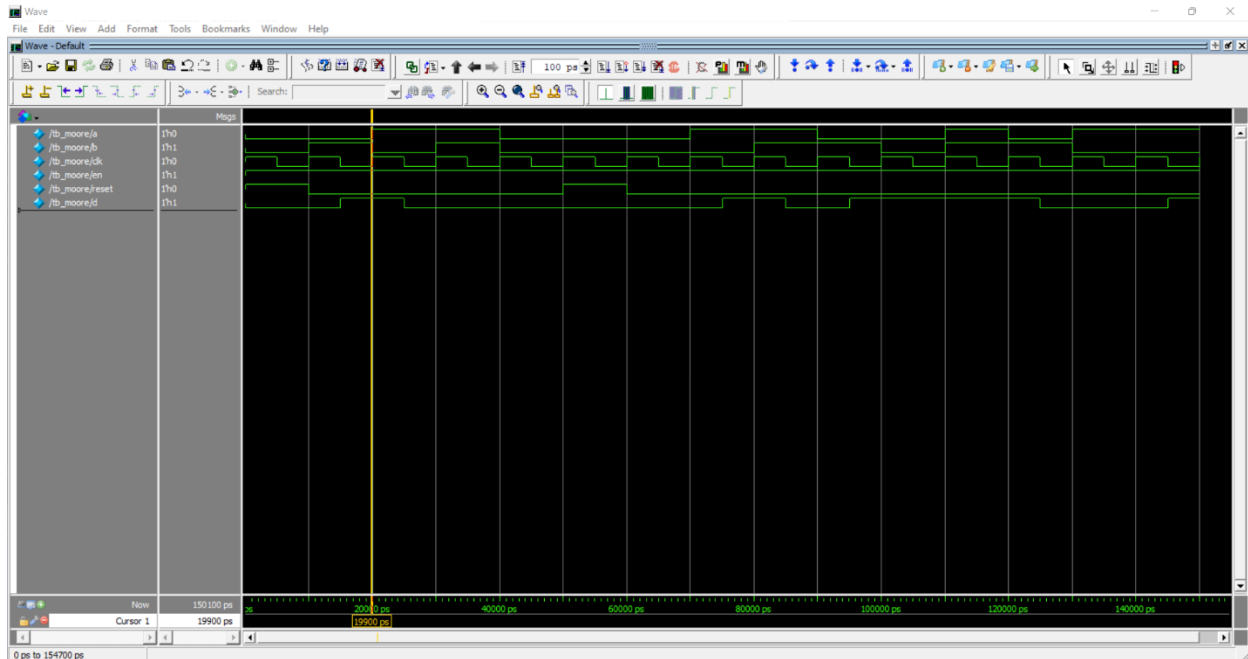
endmodule

PART 3: BEHAVIORAL SIMULATION FOR MOORE USING MODELSIM (QUESTA).

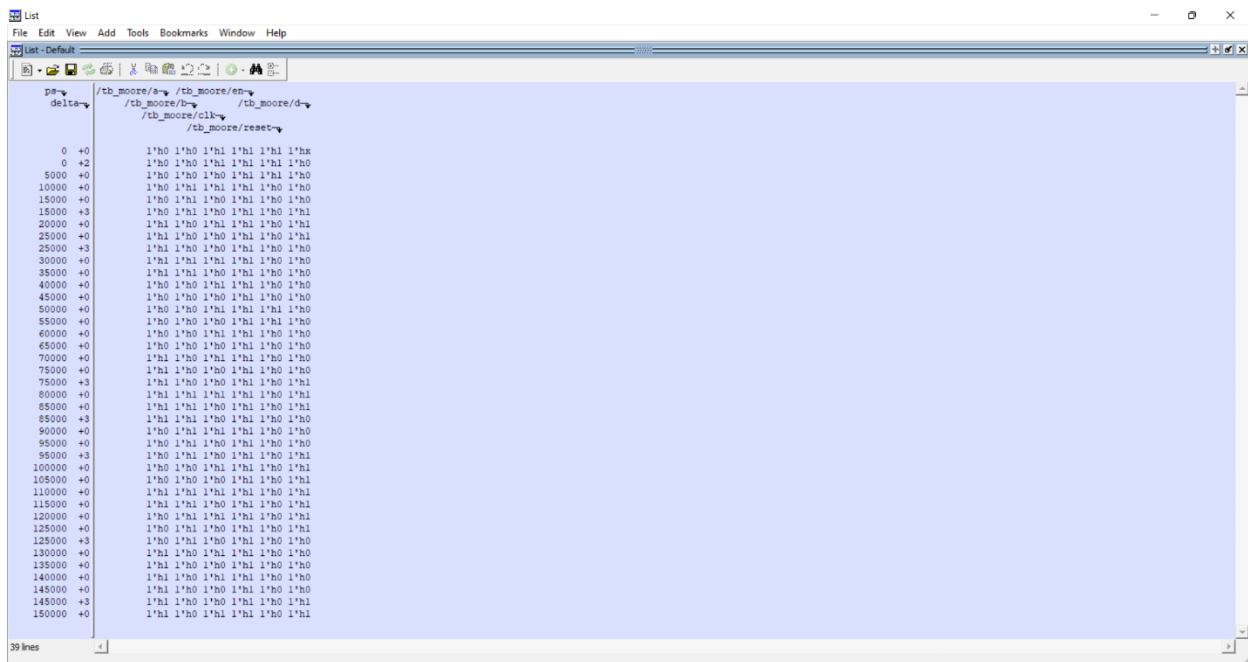
Moore transcript window:



Moore Behavioral Waveform:



Moore Behavioral List output:



Moore Behavioral Verilog source code:

```
/* Moore Implementation of subtractor
behavioral model*/
module moore_b(input A, B, CLK, En, Reset, output reg D);

    reg[3:0]    State=0, Nextstate=0;
```

```

//combinational network 1.
always @ (State, A, B)
    case (State)
        0      :    begin
                        D=0;
                        if(!A && !B) Nextstate=0;
                        else if(!A && B) Nextstate=3;
                        else if(A && !B) Nextstate=1;
                        else Nextstate=0;
                    end

        1      :    begin
                        D=1;
                        if(!A && !B) Nextstate=0;
                        else if(!A && B) Nextstate=3;
                        else if(A && !B) Nextstate=1;
                        else Nextstate=0;
                    end

        2      :    begin
                        D=0;
                        if(!A && !B) Nextstate=3;
                        else if(!A && B) Nextstate=2;
                        else if(A && !B) Nextstate=0;
                        else Nextstate=3;
                    end

        3      :    begin
                        D=1;
                        if(!A && !B) Nextstate=3;
                        else if(!A && B) Nextstate=2;
                        else if(A && !B) Nextstate=0;
                        else Nextstate=3;
                    end

        default:    Nextstate=0;
    endcase

//state reg portion.
always @ (negedge CLK)
    if(Reset) State = 0;
    else if(En) State = Nextstate;

endmodule

```

