EECS 151/251A Homework 3

Due Friday, Sept 25th, 2020

2020-10-03

For this HW Assignment

You will be asked to write several Verilog modules as part of this HW assignment. You will need to test your modules by running them through a simulator. A useful tool is https://www.edaplayground.com, a free, online Verilog simulator.

For all problems, include your Verilog code, test bench, and test results (including the simulation output and a waveform). Also explain what aspects of your design are being verified by your testbench.

Problem 1: Logic Simplification [5 pts]

Take this truth table consisting of 4 input variables(A, B, C, D) and 1 output(Out):

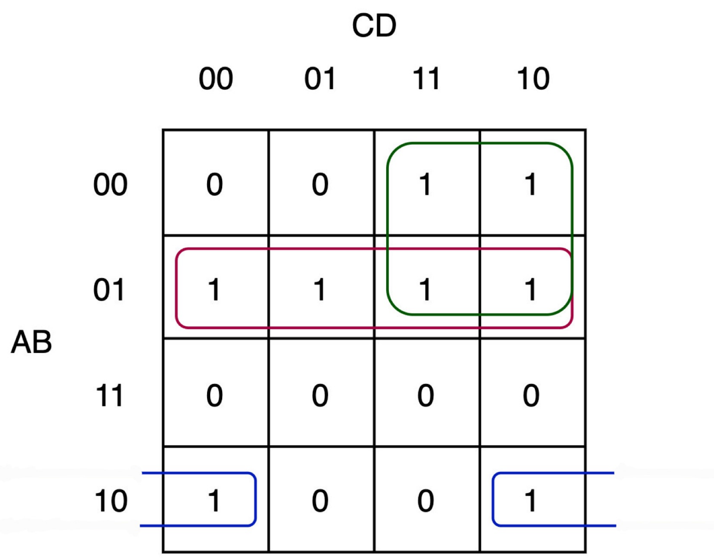
A	**B**	**C**	**D**	**Out**
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

1. Write a sum-of-products directly from the truth table. [1 pt]
2. Use a Karnaugh Map to simplify the logic and write the simplified sum-of-products and product-of-sums representations. [2 pts]
3. Using the simplified sum-of-products representation, draw the circuit that implements this function. Transform this circuit such that it is made up only of inverters and NAND gates. All gates should have 2 inputs and 1 output. [2 pts]

Solution:

1. $\bar{A}\bar{B}C\bar{D} + \bar{A}\bar{B}CD + \bar{A}B\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}BC\bar{D} + \bar{A}BCD + A\bar{B}\bar{C}\bar{D} + A\bar{B}C\bar{D}$

2.



AB \ CD	00	01	11	10
00	0	0	1	1
01	1	1	1	1
11	0	0	0	0
10	1	0	0	1

SoP: $\bar{A}B + A\bar{B}\bar{D} + \bar{A}C$
PoS: $(\bar{A} + \bar{B})(\bar{A} + \bar{D})(A + B + C)$
3.

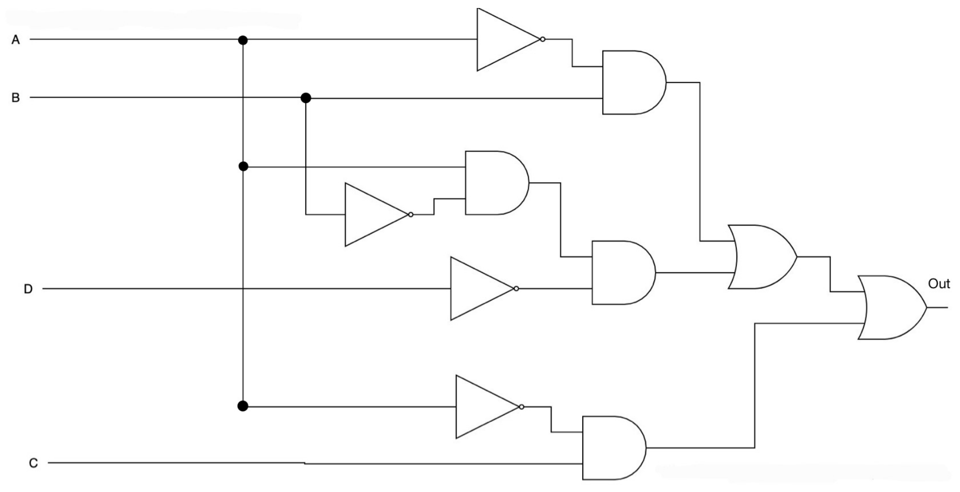


Figure 1: Direct implementation of the simplified SoP representation

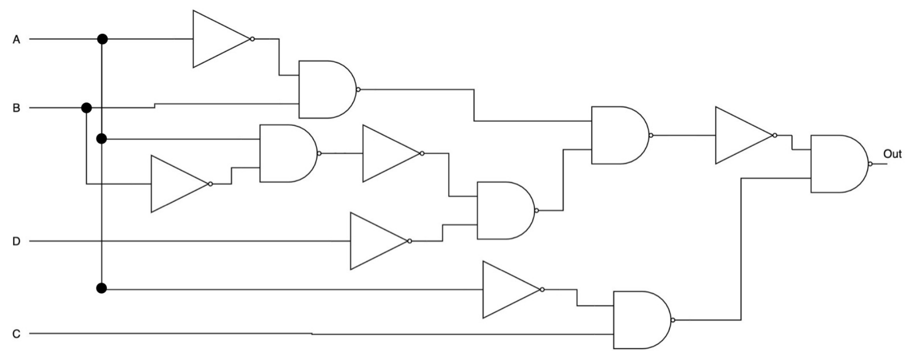


Figure 2: NAND-only implementation of the simplified SoP representation

Problem 2: Combinational Logic [4 pts]

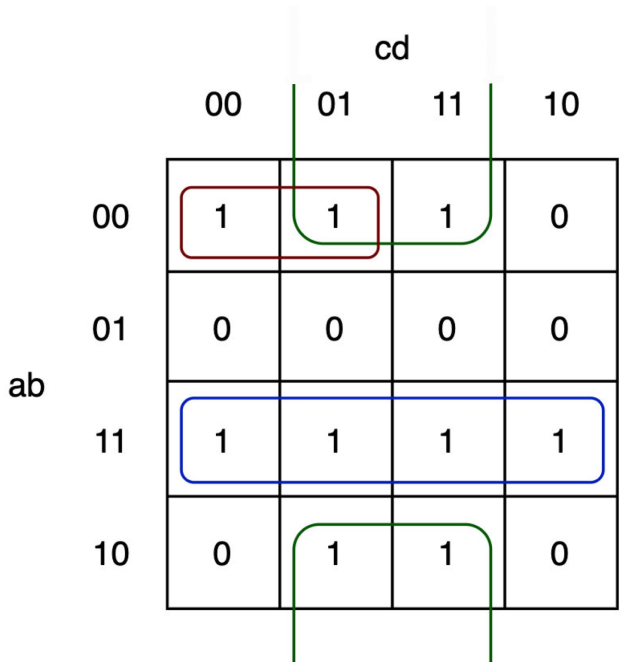
Consider the following Boolean function:

$$\bar{a}\bar{b}\bar{c}\bar{d} + \bar{a}\bar{b}\bar{c}d + \bar{a}\bar{b}cd + a\bar{b}d + \overline{(\bar{a}+\bar{b}+c)} + \overline{(\bar{a}+\bar{b}+\bar{c})}$$

1. Use a K-map to simplify. Show your work. [1 pts]
2. Use a Boolean algebra to simplify. Show your work. [1 pts]
3. Using the simplified sum-of-products representation, draw the circuit that implements this function. Then bubble push to transform the circuit into products-of-sum form. All gates should have 2 inputs and 1 output. [2 pts]

Solution:

1.



ab \ cd	00	01	11	10
00	1	1	1	0
01	0	0	0	0
11	1	1	1	1
10	0	1	1	0

$ab + \bar{a}\bar{b}\bar{c} + \bar{b}d$

2. $\bar{a}\bar{b}\bar{c}\bar{d} + \bar{a}\bar{b}\bar{c}d + \bar{a}\bar{b}cd + a\bar{b}d + \overline{(\bar{a}+\bar{b}+c)} + \overline{(\bar{a}+\bar{b}+\bar{c})}$
$= \bar{a}\bar{b}\bar{c}\bar{d} + \bar{a}\bar{b}\bar{c}d + \bar{a}\bar{b}\bar{c}d + \bar{a}\bar{b}cd + a\bar{b}d + \overline{(\bar{a}+\bar{b}+c)} + \overline{(\bar{a}+\bar{b}+\bar{c})}$
$= \bar{a}\bar{b}\bar{c}(\bar{d}+d) + \bar{a}\bar{b}(\bar{c}+c)d + a\bar{b}d + ab\bar{c} + abc$
$= \bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}d + a\bar{b}d + ab(\bar{c}+c)$
$= \bar{a}\bar{b}\bar{c} + (\bar{a}+a)\bar{b}d + ab$
$= \bar{a}\bar{b}\bar{c} + \bar{b}d + ab$

3.

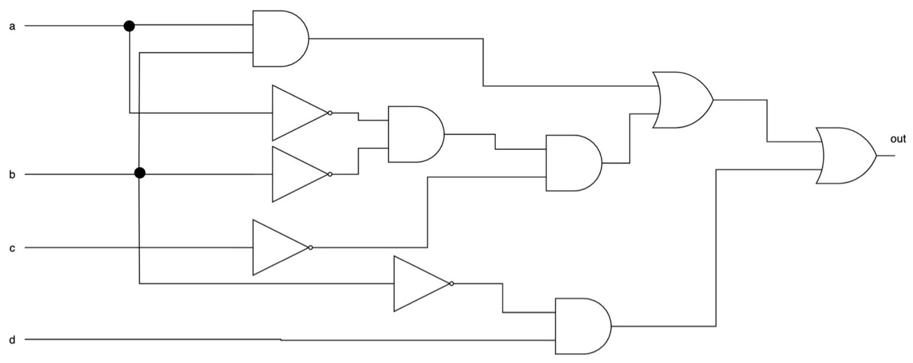


Figure 3: Direct implementation of the simplified SoP representation

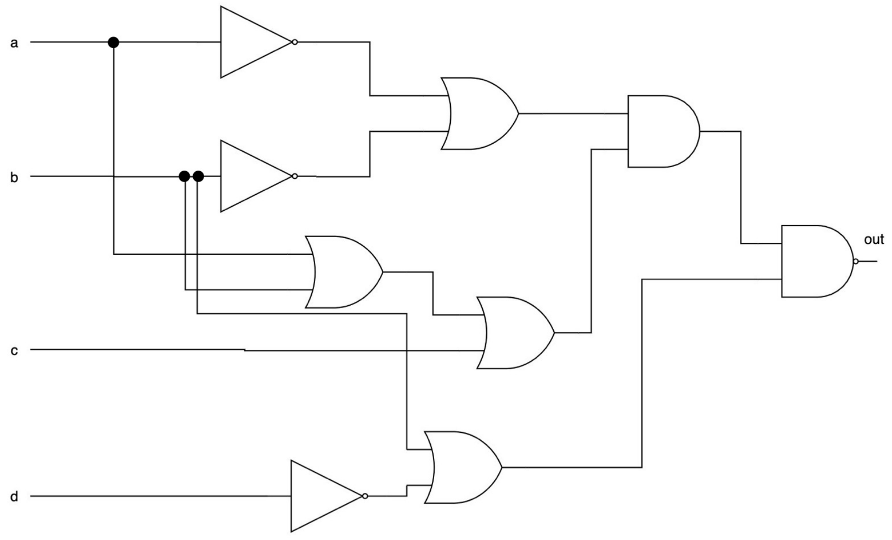


Figure 4: PoS representation after bubble pushing

Problem 3: FSM DNA sequencing[6 pts]

Help your TAs design a DNA sequencing machine! We want the machine to be able to detect some DNA sequences of bases(A, C, G, T) that we are interested in. The machine receives a one-hot encoded ACGT input every clock cycle, and outputs whether the sequence has been detected. `OUT` is pulled high for 1 clock cycle to indicate a pattern is detected, and then pulled low to prepare for the next match. The machine also has a `RESET` button that lets the user interrupt and start over at any time.

sequence to detect: `ACCTG`
inputs: `ACGT(4 bits), RESET(1 bit)`
output: `OUT`

1. Draw the state diagram of this circuit, marking the transition conditions and output values. Your implementation should be in the style of a Moore machine. [1 pts]
2. Write the Verilog that corresponds to your circuit in part a). Simulate your circuit using both the given and other sequences. Show the waveform. [2 pts]
3. Draw the state diagram of this circuit as a Mealy state machine. [1 pts]
4. Write the Verilog that corresponds to your circuit in part (c). Simulate your circuit using both the given and other sequences. Show the waveform. [2 pts]

Solution:

1.

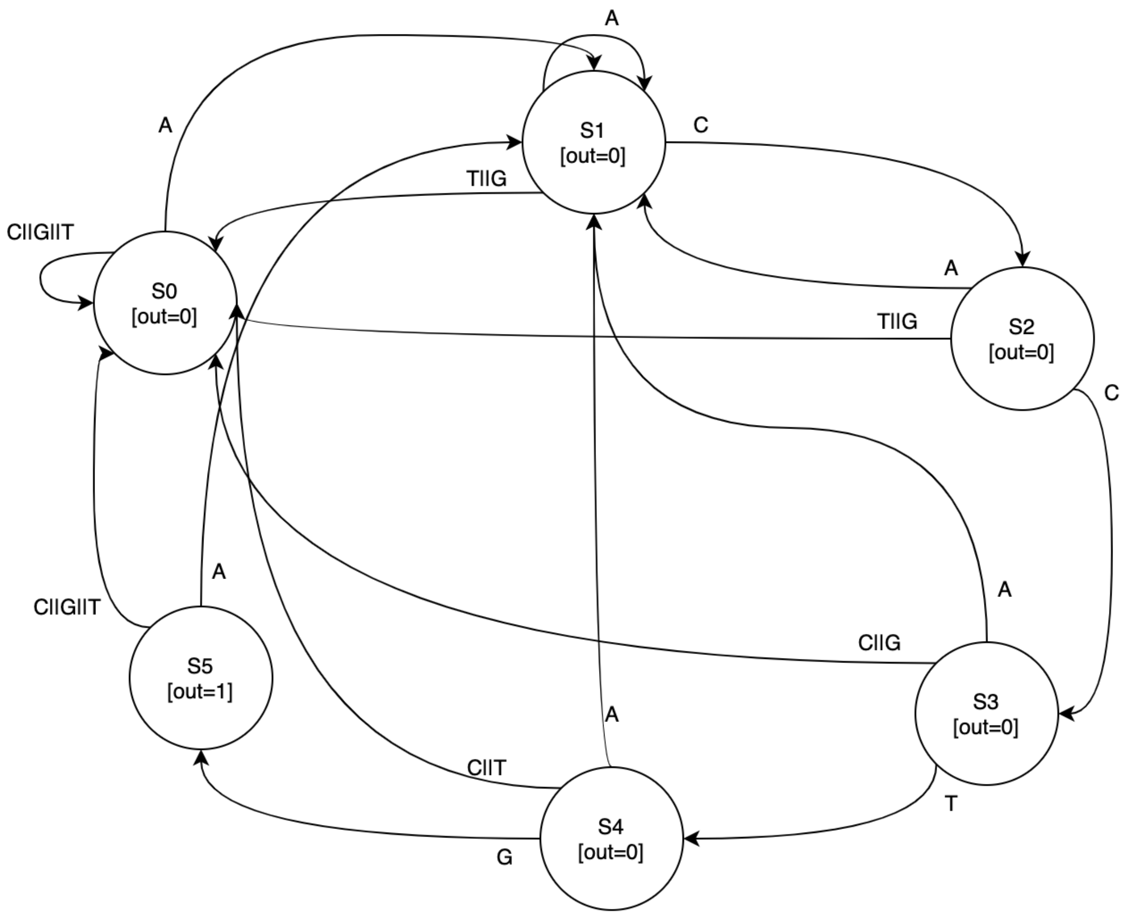


Figure 5: Moore FSM

2. **Design:**

```
module moore (input clk,
                input rst,
                input [3:0] acgt,
                output out
               );

  localparam S0 = 0;
  localparam S1 = 1;
  localparam S2 = 2;
  localparam S3 = 3;
  localparam S4 = 4;
  localparam S5 = 5;

  reg [2:0] current_state, next_state;

  assign out = (current_state == S5) ? 1'b1 : 1'b0;

  always @(posedge clk) begin
    if (rst) current_state <= S0;
    else current_state <= next_state;
  end

  always @(*) begin
    case (current_state)

      S0:
        begin
          if (acgt == 4'b1000) next_state = S1;
          else next_state = S0;
              end

       S1:
         begin
           if (acgt == 4'b0100) next_state = S2;
           else if (acgt == 4'b1000) next_state = S1;
           else next_state = S0;
         end

        S2:
          begin
            if (acgt == 4'b0100) next_state = S3;
           else if (acgt == 4'b1000) next_state = S1;
           else next_state = S0;
          end

              S3:
          begin
            if (acgt == 4'b0001) next_state = S4;
           else if (acgt == 4'b1000) next_state = S1;
           else next_state = S0;
          end

               S4:
          begin
            if (acgt == 4'b0010) next_state = S5;
           else if (acgt == 4'b1000) next_state = S1;
           else next_state = S0;
```

```
          end

         S5:
          begin
            if (acgt == 4'b1000) next_state = S1;
                        else next_state = S0;
          end

              default:
          begin
                        next_state = S0;
          end
    endcase
  end
endmodule
```

Testbench:

```
`timescale 1ns/1ns

module moore_test;
  reg clk, rst;
  reg [3:0] acgt;
  wire out;

  initial clk = 0;

  moore dut(.clk(clk), .rst(rst), .acgt(acgt), .out(out));

  always #(2) clk <= ~clk;
  initial begin
    $dumpfile("dump.vcd");
    $dumpvars(1, moore_test);

    clk = 0;

    rst = 1;
    #4;
    rst = 0;

    // sequence acctg
    acgt = 4'b1000;
    #4;
    acgt = 4'b0100;
    #4;
    acgt = 4'b0100;
    #4;
    acgt = 4'b0001;
    #4;
    acgt = 4'b0010;
    #4;

    // sequence cccat
    acgt = 4'b0100;
    #4;
    acgt = 4'b0100;
    #4;
    acgt = 4'b0100;
    #4;
    acgt = 4'b1000;
    #4;
    acgt = 4'b0001;
    #4;
    $finish();

  end
endmodule
```

Waveform:

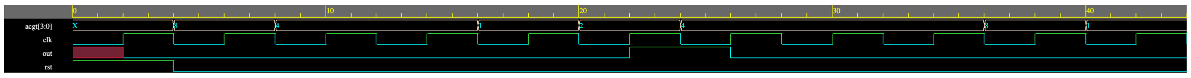


Figure 6: Moore FSM testbench waveform

3.

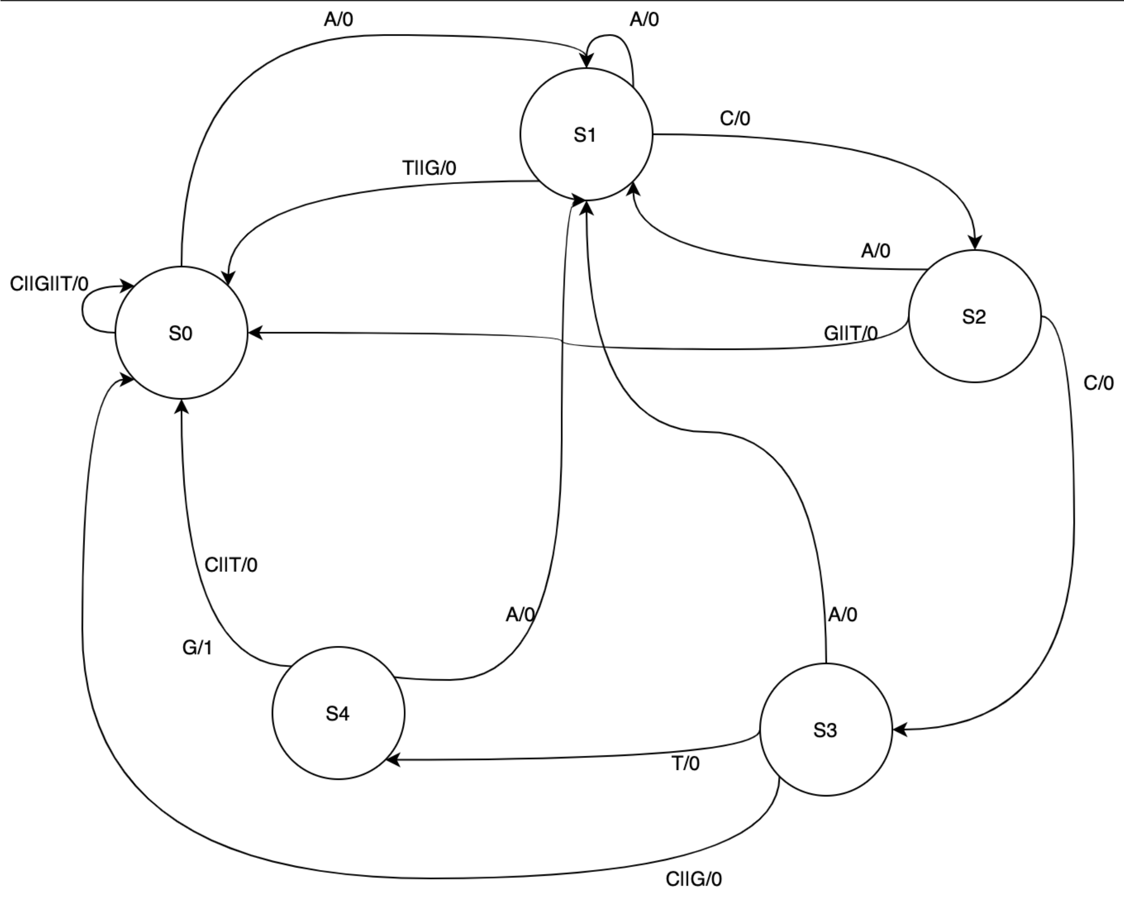


Figure 7: Mealy FSM

4. **Design:**

```
module mealy (input clk,
                 input rst,
                 input [3:0] acgt,
                 output reg out
                );

  localparam S0 = 0;
  localparam S1 = 1;
  localparam S2 = 2;
  localparam S3 = 3;
  localparam S4 = 4;

  reg [2:0] current_state, next_state;
  reg out_sig;
```

```
always @(posedge clk) begin
      if (rst) current_state <= S0;
  else current_state <= next_state;
  out <= out_sig;
end

always @(*) begin
  case (current_state)

    S0:
      begin
        out_sig = 1'b0;
        if (acgt == 4'b1000) next_state = S1;
        else next_state = S0;
            end

     S1:
       begin
         out_sig = 1'b0;
         if (acgt == 4'b0100) next_state = S2;
         else if (acgt == 4'b1000) next_state = S1;
         else next_state = S0;
       end

      S2:
        begin
          out_sig = 1'b0;
          if (acgt == 4'b0100) next_state = S3;
         else if (acgt == 4'b1000) next_state = S1;
         else next_state = S0;
        end

            S3:
        begin
          out_sig = 1'b0;
          if (acgt == 4'b0001) next_state = S4;
         else if (acgt == 4'b1000) next_state = S1;
         else next_state = S0;
        end

             S4:
        begin
          if (acgt == 4'b0010) begin
            out_sig = 1'b1;
            next_state = S0;
          end
```

```
              else if (acgt == 4'b1000) begin
                out_sig = 1'b0;
                next_state = S1;
              end
             else begin
               out_sig = 1'b0;
               next_state = S0;
             end
            end

                default:
            begin
                          next_state = S0;
              out_sig = 1'b0;
            end
      endcase
    end
  endmodule
```

Testbench:

```
`timescale 1ns/1ns

module mealy_test;
  reg clk, rst;
  reg [3:0] acgt;
  wire out;

  initial clk = 0;

  mealy dut(.clk(clk), .rst(rst), .acgt(acgt), .out(out));

  always #(2) clk <= ~clk;
  initial begin
    $dumpfile("dump.vcd");
    $dumpvars(1, mealy_test);

    clk = 0;

    rst = 1;
    #4;
    rst = 0;

    // sequence acctg
    acgt = 4'b1000;
    #4;
    acgt = 4'b0100;
    #4;
    acgt = 4'b0100;
    #4;
    acgt = 4'b0001;
    #4;
    acgt = 4'b0010;
    #4;

    // sequence cccat
    acgt = 4'b0100;
    #4;
    acgt = 4'b0100;
    #4;
    acgt = 4'b0100;
    #4;
    acgt = 4'b1000;
    #4;
    acgt = 4'b0001;
    #4;
    $finish();

  end
endmodule
```

Waveform:

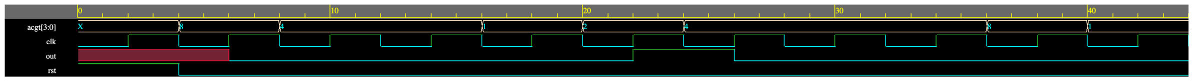


Figure 8: Mealy FSM testbench waveform