Problem Set 5

Magson Gao (magson@mit.edu)
6.111 Fall 2018

September 23, 2018

1 Problem A

Key:

● State/Output ↷ a Required Input Event a **(Default: any)**

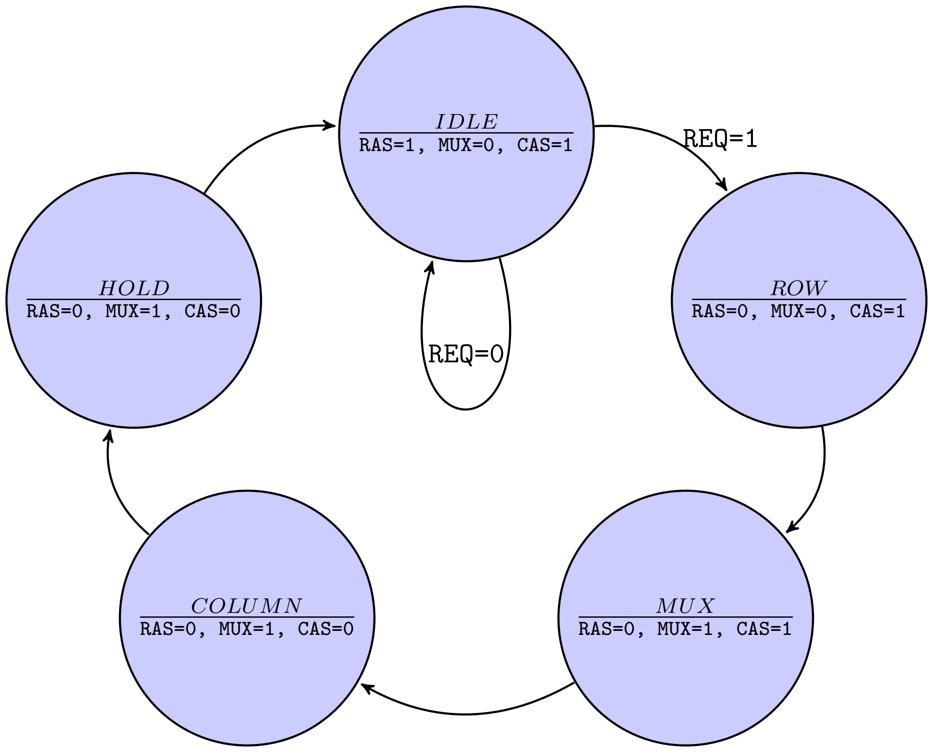


Figure 1.1: The finite state machine diagram for the given system.

2 Problem B

Code:

```
module mem_controller(
            input        clk_in,
            input        req_in,
            output       ras_out,
            output       mux_out,
            output       cas_out
          );

  localparam STATE_IDLE = 3'b000;
  localparam STATE_ROW  = 3'b001;
  localparam STATE_MUX  = 3'b011;
  localparam STATE_COL  = 3'b010;
  localparam STATE_HOLD = 3'b110;

  reg [2:0] state = STATE_IDLE;

  assign ras_out = (state == STATE_IDLE);
  assign mux_out = ~((state == STATE_IDLE) | (state == STATE_ROW));
  assign cas_out = ((state == STATE_IDLE) | (state == STATE_ROW) | (state == STATE_MUX));

  // state machine
  always @(posedge clk_in)
  begin
    case (state)
      STATE_IDLE : state <= req_in ? STATE_ROW : STATE_IDLE;
      STATE_ROW  : state <= STATE_MUX;
      STATE_MUX  : state <= STATE_COL;
      STATE_COL  : state <= STATE_HOLD;
      STATE_HOLD : state <= STATE_IDLE;
    endcase
  end

endmodule
```

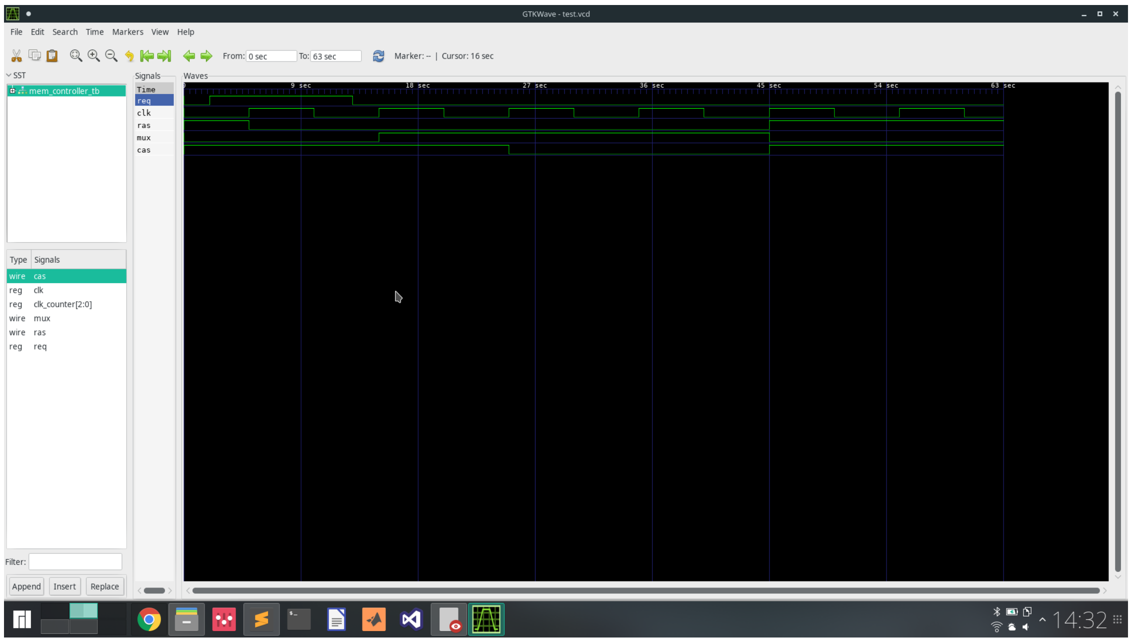
Testbench:

```
`define assert(signal, value) \
        if (signal !== value) \
        begin \
            $display("ASSERTION FAILED in %m at time %0t: signal != value", $time); \
        end

module mem_controller_tb;
  reg         clk;
  reg         req;
  wire        ras;
  wire        mux;
  wire        cas;
  reg [2:0]   clk_counter;

  initial
  begin
      $dumpfile("test.vcd");
    $dumpvars(0,mem_controller_tb);
    clk         = 0;
    clk_counter = 0;
    req         = 0;

    #2;
    req = 1;

    #11;
    req = 0;
    #50;
    $finish;
  end


  mem_controller m1(
            .clk_in    (clk),
            .req_in    (req),
            .ras_out   (ras),
            .mux_out   (mux),
            .cas_out   (cas)
            );

  always @(posedge clk)
  begin
    if ((clk_counter > 0) | ((req == 1) & (clk_counter == 0)))
    begin
      clk_counter = clk_counter + 1;

      if (clk_counter == 1)
      begin
        `assert(ras, 0);
        `assert(mux, 0);
        `assert(cas, 1);
      end
      else if (clk_counter == 2)
      begin
        `assert(ras, 0);
        `assert(mux, 1);
        `assert(cas, 1);
      end
      else if (clk_counter == 3)
      begin
        `assert(ras, 0);
        `assert(mux, 1);
        `assert(cas, 0);
      end
      else
      begin
        `assert(ras, 0);
        `assert(mux, 1);
        `assert(cas, 0);
        clk_counter = 0;
      end
    end
  end

  always #5 clk = ~clk;

endmodule
```

3 Problem C

It is glitch free as the states are encoded using grey codes, thus all change of states except from state hold to state idle requires only one bit change. For state hold to state idle there are two possible glitches: 110 to 100 to 000 or 110 to 010 to 000. In both cases the intermediate meta-state has identical output values to the initial hold state. Thus no output glitches occur.

4 Problem D


GTKWave - test.vcd
File Edit Search Time Markers View Help
From: 0 sec
To: 63 sec
Marker: -- | Cursor: 16 sec
SST
mem_controller_tb
Type Signals
wire cas
reg clk
reg clk_counter[2:0]
wire mux
wire ras
reg req
Filter:
Append Insert Replace
Signals
Time
req
clk
ras
mux
cas
Waves
9 sec
18 sec
27 sec
36 sec
45 sec
54 sec
63 sec
14:32