Massachusetts Institute of Technology

Department of Electrical Engineering and Computer Science
6.111 - Introductory Digital Systems Laboratory

Problem Set 2 Solutions

Issued: February 19, 2008

Problem 1: Counters

a) **74LS393** is an **asynchronous 4-bit counter**, implemented with **4 serial T-registers**. The outputs of registers are connected to the inputs of the subsequent registers. Thus, MSBs change only after LSBs change. and clk-to-MSB delay is four times of clk-to-q delay of each register. **74LS163**, on the hand, is a **synchronous 4-bit counter**, implemented by **4 parallel J-K registers**. All output bits change at the rising edge of clock and thus clk-to-MSB delay is similar to clk-to-q delay of individual register.

b) The N^{th} output of ripple counters has twice the time period of $(N-1)^{th}$ output, and thus the N^{th} output of ripple counters is 2^N slower than the input clock. Thus, to create a 421.875kHz clock,

$$\frac{27\,MHz}{421.875\,kHz} = 64 = 2^6$$

we must divide the input clock by two for six times, requiring two 74LS163. Cascade the two 4-bit counters as done in Lab 1 and probe at Qb of the second counter as shown below:

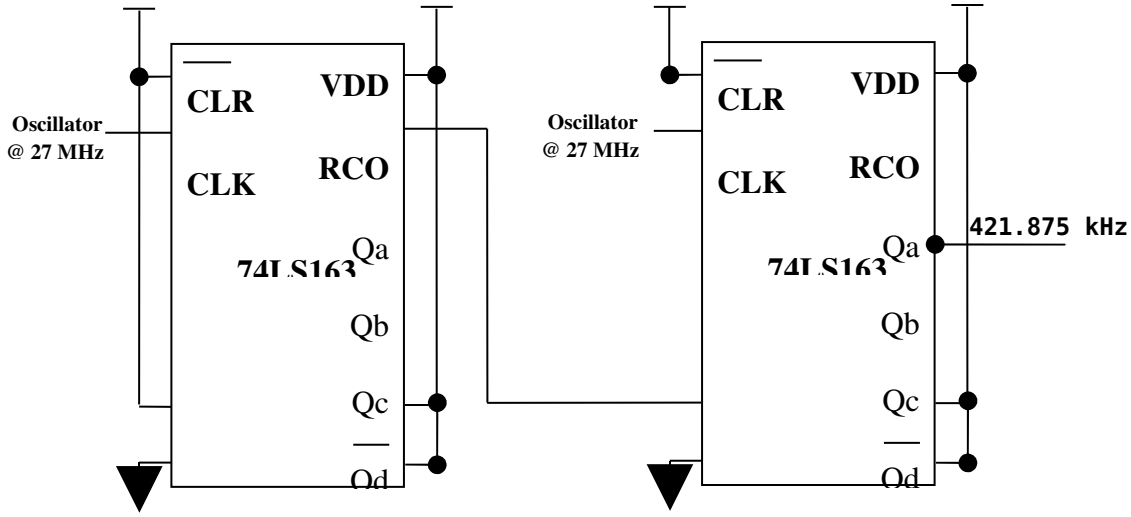


Figure 1: 74LS163 connections to provide a 421.875kHz clock

c) 27000000 is not an exact multiple of twos and thus we cannot use the method used in part b). Instead, using 74LS163s, we count up to 27000000 and pulse the output then to generate 1Hz clock. Since $2^{24} < 27000000 < 2^{25}$ and there are four bits per counter, we need **7 74LS163s** to produce a 1 Hz clock.

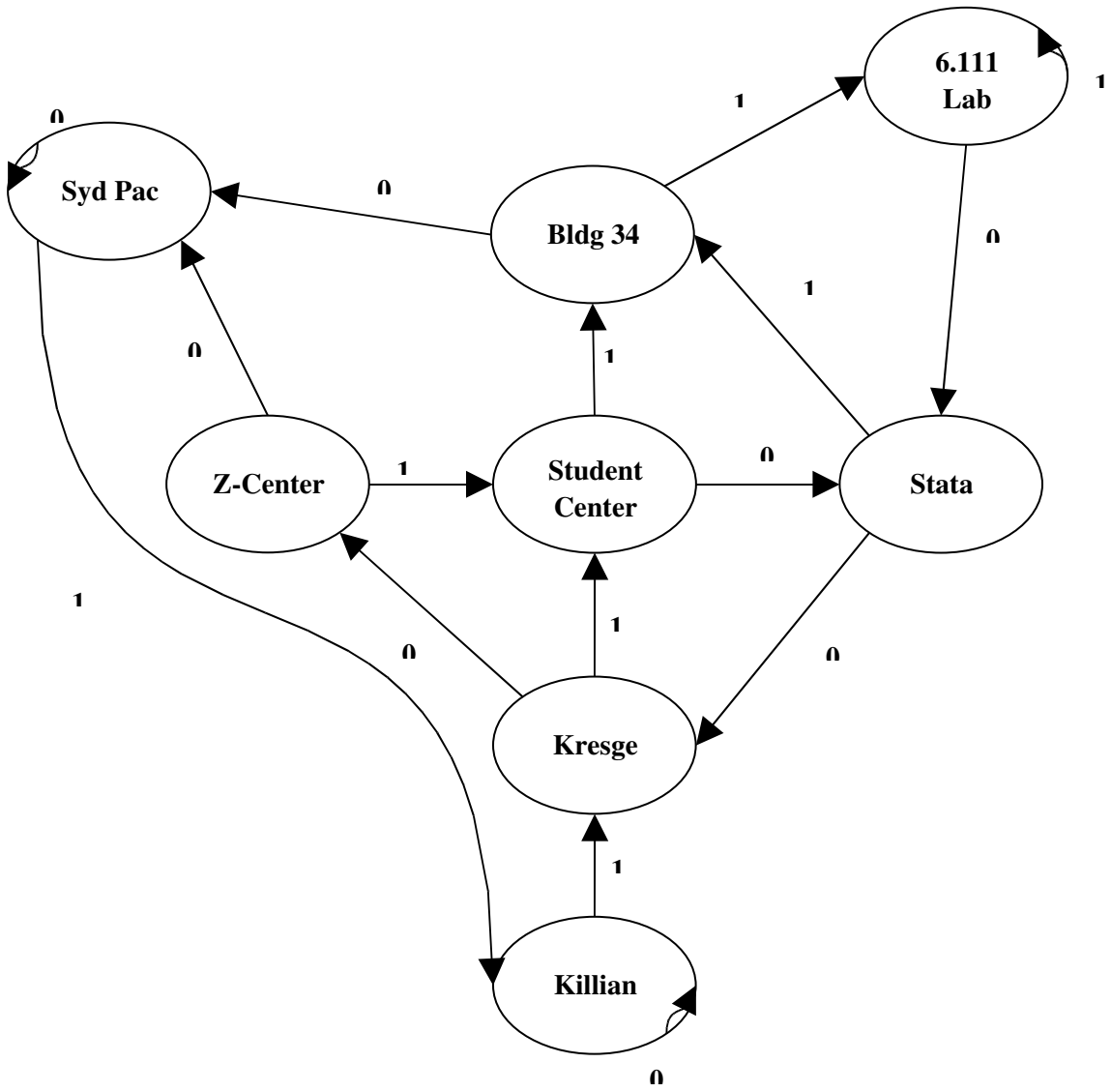
d) Verilog Code for Counter Module:

```
`timescale 1ns/10ps

module counter(clk, reset, enable);

   input clk, reset;
   output enable;

   reg enable;
   reg [24:0] cnt;       //2^24 < 27MHz < 2^25

   always @ (posedge clk)
   begin
         if (reset == 1)
         begin
               cnt <= 25'd0;
               enable <= 1'b0;
         end
         else if(cnt == 25'd26999999)
         // else if(cnt == 25'd1) //for faster simulation
         begin
               enable <= 1'b1;
               cnt <= 25'd0;
         end
```

```
            else
            begin
                    cnt <= cnt + 1;
                    enable <= 1'b0;
            end //else
        end //always

    endmodule
```

Problem 2: Finite State Machines (FSM)

a)



b) **6.111 Lab**

c) **6.111 Lab**

d) Verilog Code for stata_FSM module:

```
`timescale 1ns/10ps

module stata_fsm(clk, fsm_reset, fsm_input, state);

// System Clk for state transition
input clk;

// Global Reset signal
input fsm_reset;
input fsm_input;

output [2:0] state;

// internal state
reg [2:0] state;
reg [2:0] nextstate;

//State Declarations:
parameter KILLIAN = 0;
parameter KRESGE = 1;
parameter ZCENTER= 2;
parameter SYDPAC = 3;
parameter STUDENTCENTER = 4;
parameter BUILDING34 = 5;
parameter LAB = 6;
parameter STATACENTER = 7;

always @ (posedge clk)
begin
    if (fsm_reset) state <= KILLIAN;
    else state <= nextstate;
end

always @ (state or fsm_input)
begin
    nextstate = 3'b000;
    case (state)

    KILLIAN: if (fsm_input) nextstate = KRESGE;
    else nextstate = KILLIAN;

    KRESGE: if (fsm_input) nextstate = STUDENTCENTER;
    else nextstate = ZCENTER;

    ZCENTER: if (fsm_input) nextstate = STUDENTCENTER;
    else nextstate = SYDPAC;

    SYDPAC: if (fsm_input) nextstate = KILLIAN;
```

```
        else nextstate = SYDPAC;

        STUDENTCENTER: if (fsm_input) nextstate = BUILDING34;
        else nextstate = STATACENTER;

        BUILDING34: if (fsm_input) nextstate = LAB;
         else nextstate = SYDPAC;

        STATACENTER: if (fsm_input) nextstate = BUILDING34;
         else nextstate = KRESGE;

        LAB: if (fsm_input) nextstate = LAB;
        else nextstate = STATACENTER;

    endcase //
  end //
endmodule
```

Problem 3: Verilog Testbench

Verilog Code for Top Module:

```
`timescale 1ns/10ps

module top(clk, reset, fsm_input, fsm_reset, state);

 input clk, reset, fsm_input, fsm_reset;
 output [2:0] state;
 wire enable;

 counter c1(clk, reset, enable);
 stata_fsm s1(enable, fsm_reset, fsm_input, state);

endmodule
```

Verilog Code for Test Bench Module:

```
`timescale 1ns/10ps

module testbench;

 reg clk, reset, fsm_input, fsm_reset;
 wire [2:0] fsm_state;
 wire [2:0] fsm;
 integer i;

// Instantiate the top module:
top t1(.clk(clk), .reset(reset), .fsm_input(fsm_input), .fsm_reset(fsm_reset),
```

```
        .state(fsm_state));

// Specify clk to have 27 MHz:
always #0.01852 clk = ~clk;

initial
begin
//Initial signal:
  clk =0;
  reset =0;
  fsm_input = 0;
  fsm_reset =0;

// Reset counter first:
  #0.03704 reset = 1;

  #0.03704 reset = 0;

// Reset FSM:
  #0.01852  fsm_reset = 1;

  #0.01852  fsm_input = 1;

  #0.01852  fsm_reset = 0;

  #0.51865  fsm_input = 0;

// Is Stata staying in 6.111Lab for three clock cycles?

  #0.01852  fsm_reset = 1;

  #0.03704  fsm_reset = 0;

// Let's toggle between 0 and 1 for 8 clock cycles
// until Stata comes back to Killian completing a tour
// without visiting 6.111 Lab!

   #0.03704 fsm_input = 1;

   #0.07048 fsm_input = 0;

for (i= 1; i<=3; i=i+1)
begin
   #0.08334 fsm_input = 1;

   #0.08334 fsm_input = 0;
end
end

endmodule
```

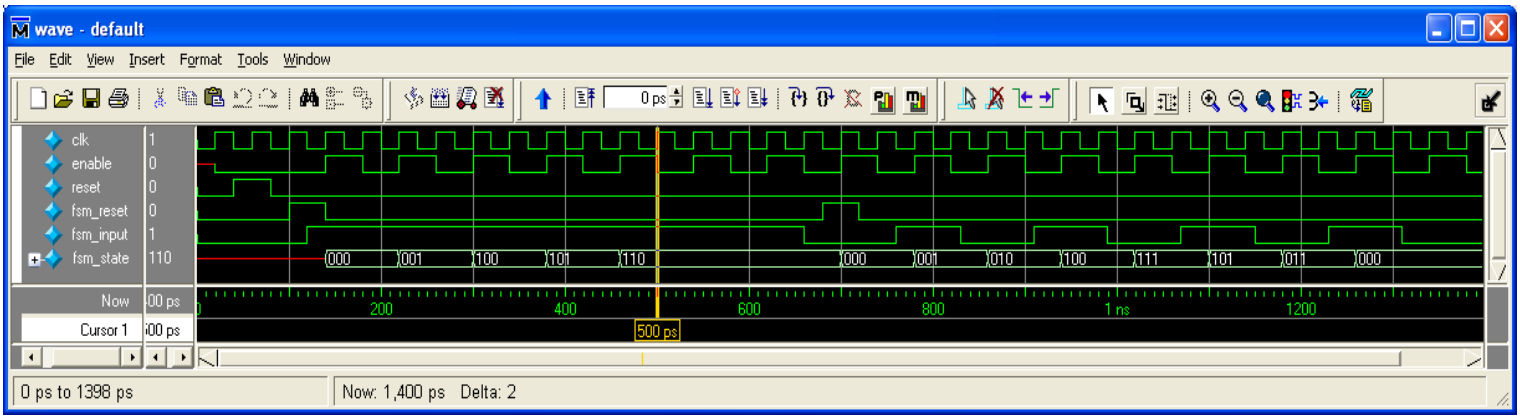


Figure 2: Testbench verifies that Stata goes from Killian -> Kresge -> Student Center -> Building 34 -> 6.111 Lab and gets stuck for three clock cycle. Then, Stata gets out completing a round tour without visiting 6.111

Problem 4: Memory Tester

$\overline{E1}$	E2	$\overline{G}$	$\overline{W}$	Mode	Output	Cycle
H	X	X	X	N/S	High-Z	-
X	L	X	X	N/S	High-Z	-
L	H	H	H	Output Disabled	High-Z	-
L	H	L	H	Read	Dout	Read Cycle
L	H	L*	L	Write	High-Z	Write Cycle
L	H	X	L	Write	High-Z	Write Cycle

* Mode is **write** and thus assume **~G** goes low coincident with or after **~W** goes low.