ECE 333
Winter 2003-2004
Exam #2

This exam is closed-notes and closed-book. You may use your brain and a writing instrument.

"To the best of my knowledge and on my honor as a student and as a professional, I attest that I have not acted in any way that could be considered academic misconduct (e.g., cheating) either in the preparation for or in the taking of this exam."

Signature:

Key

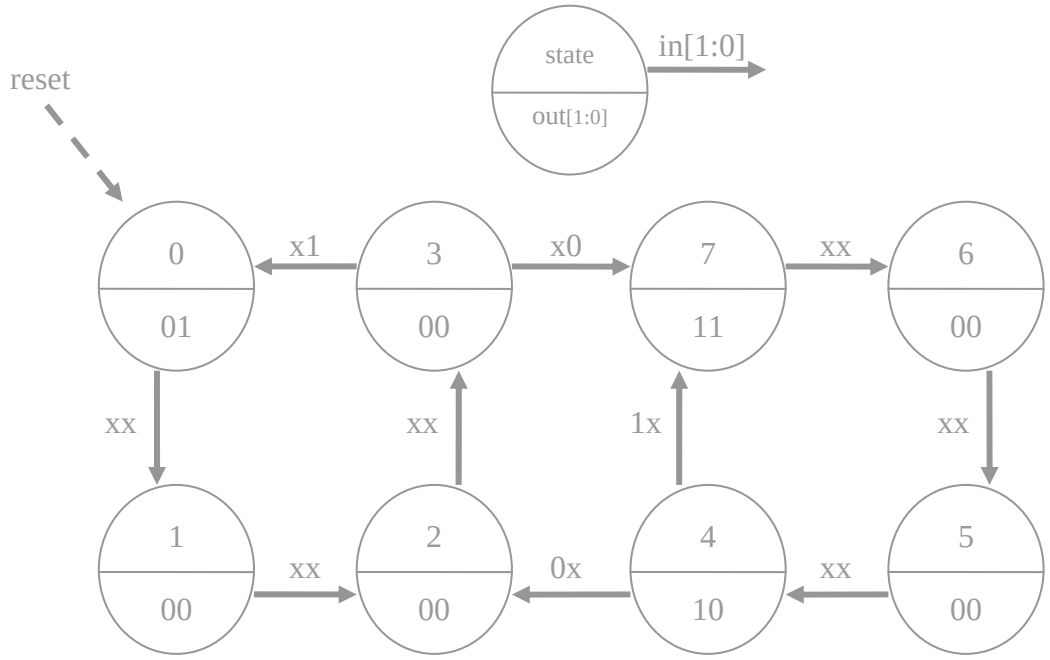
Name:

Key

Point allocation:

Page	Earned Points	Possible Points
1		50
2	0	0
3		30
4		20
Total		100

1. [50 pts] Compiled code is provided for the following state diagram:



The interface for this module is

module p1(clk, reset, in, out);

clk and **reset** are both 1-bit inputs, **in** is a 2-bit input, and **out** is a 2-bit output. **reset** is *asynchronous*.

WRITE a test bench that COMPLETELY checks the finite state machine's functionality. USE the testbench to run a waveform simulation. ANNOTATE the waveform simulation.

You must make a decision about whether or not the provided code is working correctly. You have control over the test bench and simulation. If you decide that it is not working correctly, your annotated diagram should corroborate your claim and you should document ***every aspect*** of its incorrect behavior. Similarly, if you decide that it is working correctly, your annotated diagram should corroborate your claim for ***every aspect*** of the correct behavior.

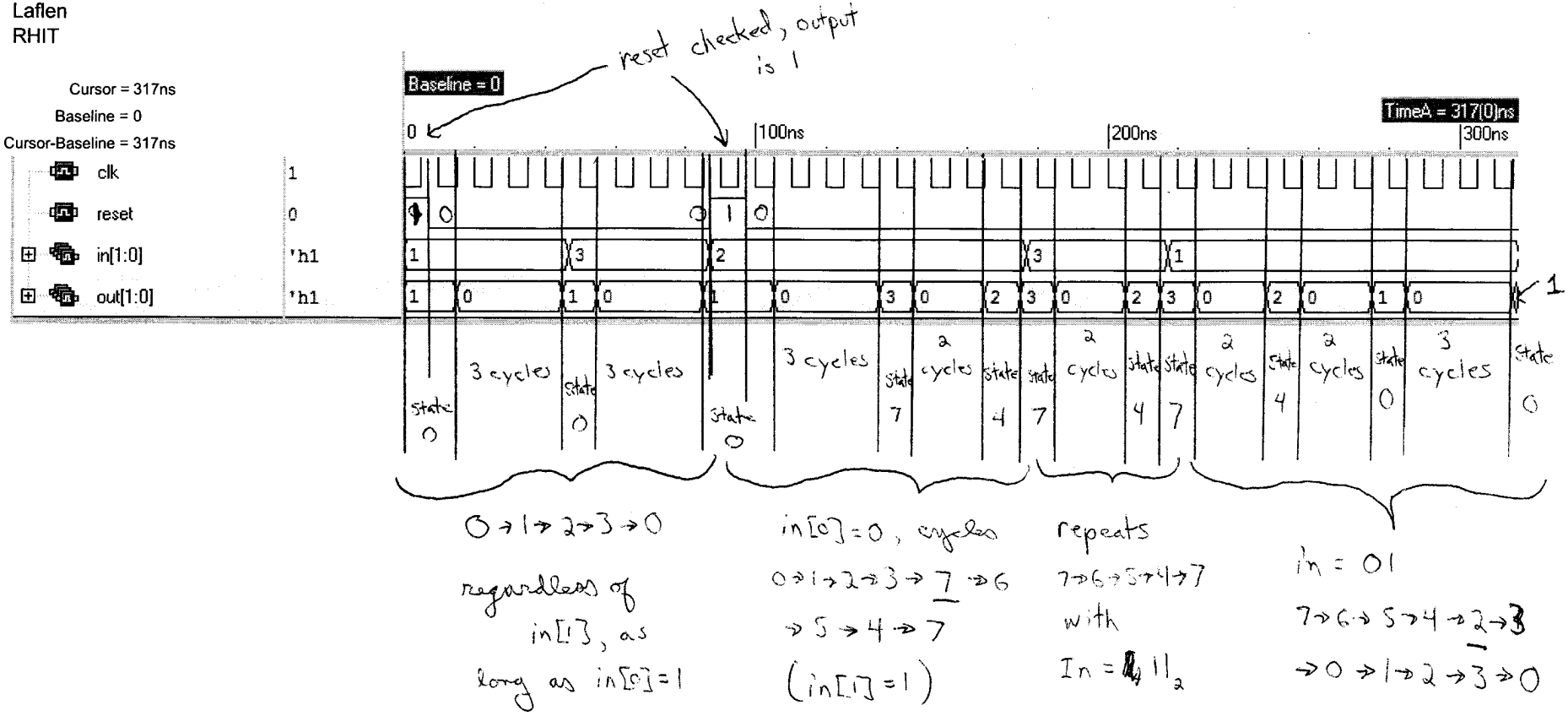
To earn full credit for this problem, turn in the following with your exam:

1. an annotated waveform simulation testing the functionality of the provided module
2. a copy of the final testbench code used to generate your waveform
3. your analysis of the provided module – does it perform according to the specification?
4. make sure that your name is clearly on each additional submitted page
5. staple everything together into your test document

```
module p1_TB;


// declare inputs to test circuit here
reg clk, reset;
reg [1:0] in;


// declare outputs from test circuit here
wire [1:0] out;


// instatiate one instance of test circuit here
p1 test_p1( .clk(clk), .reset(reset), .in(in), .out(out) );


// define clock behavior here
always #5 clk <= ~clk;


// main test loop
initial begin
   $shm_open( "p1_test_waves.shm" );
   $shm_probe( "AC" );

   // initialize inputs
   clk = 0;
   reset = 1;
   in = 1;

   // write your test(s) here
   #7 reset = 0; // offset timing by 2 units from rising clock edge

   // while in == 1, should continue looping 0-1-2-3-0
   #40
   in = 2'b11;
   #40

   in = 2'b10;
   reset = 1;
   #10 reset = 0;

   // observe behavior 0-1-2-3-7-6-5-4-7-6-5-...
   #80
   in = 2'b11;
   #40

   // observe switch back 7-6-5-4-2-3-0-1-2-3-...
   in = 2'b01;
   #100

   $finish;
end

endmodule
```

ECE 333 Winter Exam #2 Problem #1 Solution Waveform

Laflen
RHIT



- machine has correct outputs for given loops/cycles
- it is possible that the machine has extra loops (more bits of state), but I am reasonably certain it is correct.

(In fact, It Is)

2. [30 pts] Write Verilog code for a clock divider circuit that outputs a slow-clock with a frequency of one-fifth (1/5) the input clock frequency.

Inputs to the circuit are:

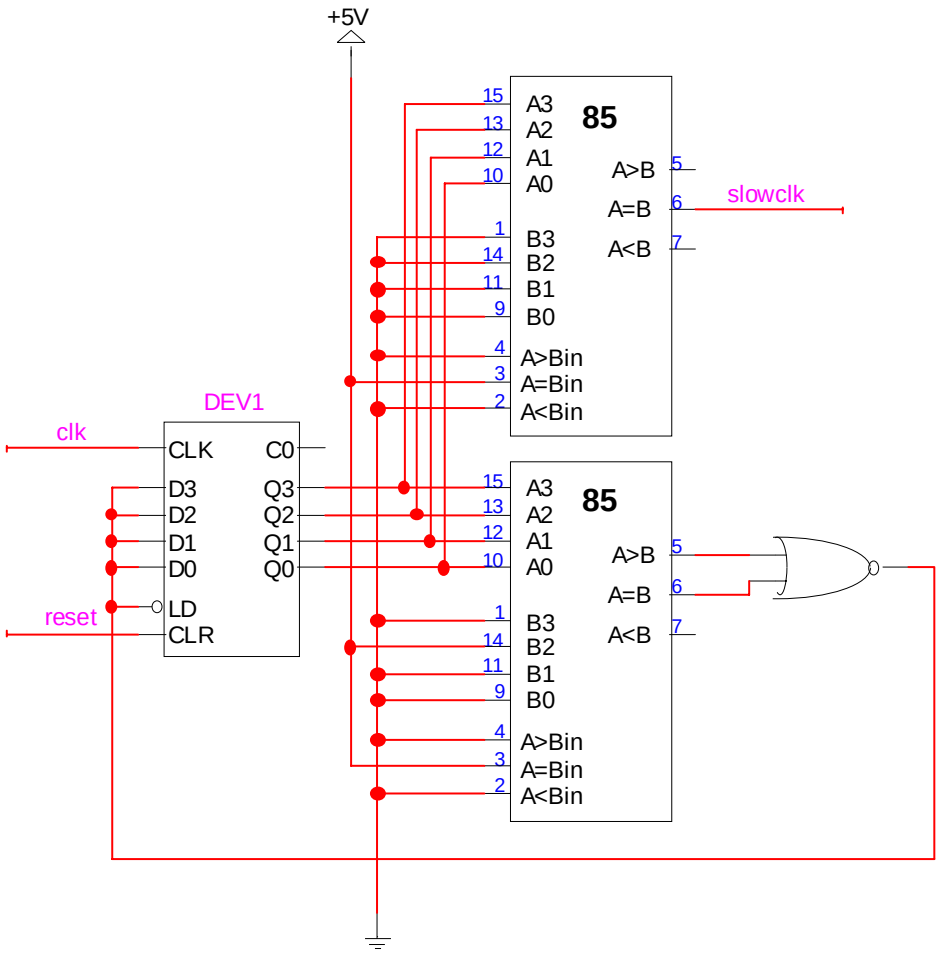
- **clk**: the fast input clock
- **reset**: an asynchronous reset

Output from the circuit is:

- **slowclk**: the 1/5 frequency slower clock. NOTE: slowclk does not need to have a specific duty cycle.

Space is provided on the next page for your Verilog code.

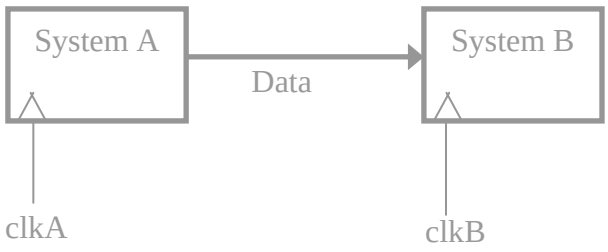
Space is provided below for you to draw a middle-level schematic of your circuit. In the schematic, include logic blocks like mux's, adders, registers, counters, decoders, and comparators. **Although you will receive most of the credit in this problem for correct Verilog, you will not receive *any* partial credit for *incorrect* Verilog code if you skip the schematic step.**



```
module clockDivFifth( clk, reset, slowclk );

input clk, reset;
output slowclk;

reg slowclk;
reg [3:0] count;
reg syncReset;

// counter
always @ ( posedge clk, posedge reset )
     if( reset )
          count <= 0;
     else if( syncReset )
          count <= 0;
     else
          count <= count + 1;

// comparator for slowclk
always @ ( count )
     if( count == 0 )
          slowclk <= 1;
     else
          slowclk <= 0;

// comparator for syncReset
always @ ( count )
     if( count >= 4 )
          syncReset <= 1;
     else
          syncReset <= 0;

endmodule
```

[20 pts] Complete the following short-answer questions:

(A) (5 pts) For the following system:



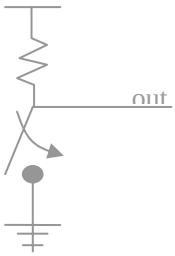
Explain the potential problems that can arise within this type of inter-system communication

clkA and clkB can be out of phase, with/without frequency mismatches. This can lead to set-up and hold time violations, resulting in metastability and data corruption.

(B) (5 pts) Suggest a solution to the potential problem in the above system

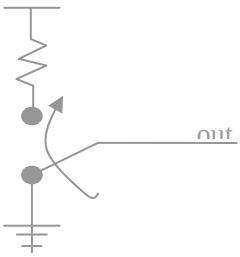
A two-register input layer should be placed on the data-input line to system B. This will prevent metastability within the system itself. However, protocol must be established to handle the asynchronous behavior – either system B must acknowledge the potential to miss incoming packets or some form of inter-system handshaking should be established.

(C) (5 pts) When the following circuit bounces, what are its possible intermediate values (*i.e.*, what does it bounce between?)



The "out" line will always be connected to either source or ground.

(D) (5 pts) When the following circuit bounces, what are its possible intermediate values (*i.e.*, what does it bounce between?)



The "out" line could potentially be disconnected from both source and ground. In this case it is "floating."