MASSACHUSETTS INSTITUTE OF TECHNOLOGY
DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

6.111 Introductory Digital Systems Laboratory
Fall 2006

Midterm Exam: November 1, 2006

Name	*Score*
SOLUTIONS	*(out of 100)*

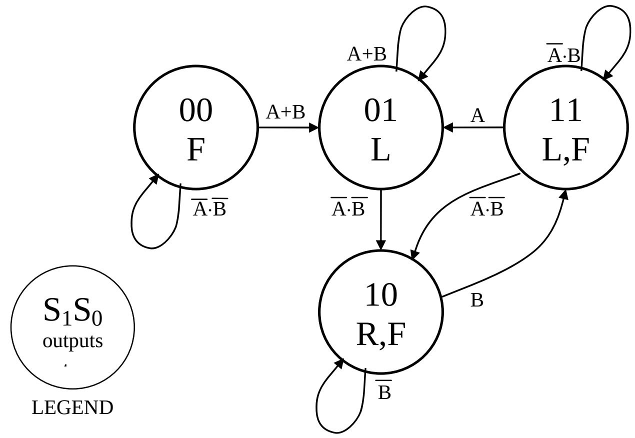
This is a closed book exam. Calculators can be used (but I don't think you'll need one).

Please write your answers legibly in the spaces provided. You can use the backs of the pages for scratch work.

Problem	**Score**
#1	
#2	
#3	
#4	

Problem 1. (30 Points)

A state transition diagram for a 4-state FSM is shown below. The FSM has two one-bit inputs (A and B) and three one-bit outputs (F, L and R). The current state is represented as a two-bit value S_1S_0.



(A) (16 Points) Please <u>neatly</u> fill in the truth table for the FSM's combinational logic below: using S_1, S_0, A and B as the inputs, and using next_S_1, next_S_0, F, L and R as the outputs.

S_1	S_0	A	B	$next\_S_1$	$next\_S_0$	F	L	R
0	0	0	0	0	0	1	0	0
0	0	0	1	0	1	1	0	0
0	0	1	0	0	1	1	0	0
0	0	1	1	0	1	1	0	0
0	1	0	0	1	0	0	1	0
0	1	0	1	0	1	0	1	0
0	1	1	0	0	1	0	1	0
0	1	1	1	0	1	0	1	0
1	0	0	0	1	0	1	0	1
1	0	0	1	1	1	1	0	1
1	0	1	0	1	0	1	0	1
1	0	1	1	1	1	1	0	1
1	1	0	0	1	0	1	1	0
1	1	0	1	1	1	1	1	0
1	1	1	0	0	1	1	1	0
1	1	1	1	0	1	1	1	0

(B) (14 Points) Give minimal sum-of-product expressions for next_S_1 and next_S_0**.** Hint: Use Karnaugh maps (you can use the backs of the exam pages for scratch work).

Minimal sum-of-products expression for next_S_1: $S_1\overline{S}_0 + S_1\overline{A} + S_0\overline{A}\overline{B}$

Minimal sum-of-products expression for next_S_0: $B + \overline{S}_1 A + S_0 A$

Problem 2. (20 Points)

Hoping to get a head start on her final project, Vera Log downloaded the following Verilog module from the net. It's supposed to implement a 4-bit two's complement adder using the propagate and generate signals Vera learned about in lecture.

```
module adder4(a,b,cin,s,cout);
  input [3:0] a,b;
  input cin;
  output [3:0] s;

  wire [3:0] a,b,s,c,p,g;
  wire cout;

  always @ (a or b) begin
    g <= a & b;
    p <= a ^ b;
    c <= g | (p & {c[2:0],cin});
    s <= p | {c[2:0],cin};
    cout <= c[3];
  end
endmodule
```

After a quick glance Vera sees that the module has many Verilog and logic errors and is hoping someone knowledgeable will fix it up. Help Vera out by rewriting the module, correcting the errors as you go.

Rewrite the module below, correcting all the errors.

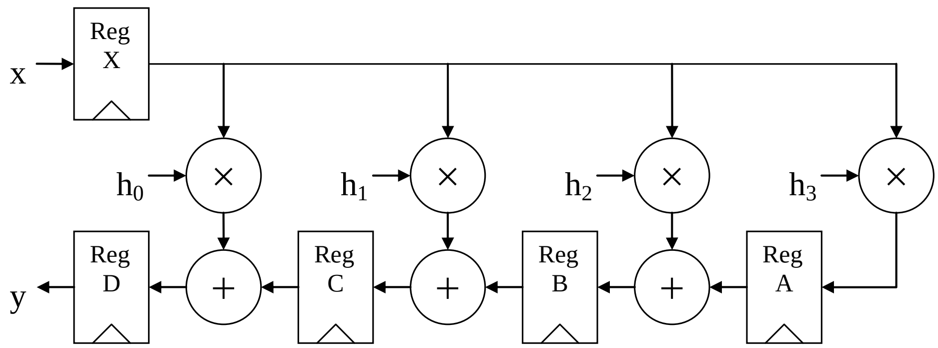
```
module adder4(a,b,cin,s,cout);
  input [3:0] a,b;
  input cin;
  output [3:0] s;
  output cout;

  wire [3:0] a,b;  // this declaration is not needed…
  reg [3:0] s,c,p,g;
  reg cout;

  always @ (a or b or cin) begin
    g = a & b;
    p = a ^ b;
    c = g | (p & {c[2:0],cin});
    s = p ^ {c[2:0],cin};
    cout = c[3];
  end
endmodule
```

Problem 3. (25 Points)

The schematic below implements a 4-tap FIR filter in its transposed form. The incoming data (x) is a stream of 16-bit two's complement numbers and the coefficients are 10-bit two's complement numbers.



(A) (6 Points) What are the minimum and maximum possible values for a coefficient?

Minimum coefficient value: -512
Maximum coefficient value: 511

(B) (5 Points) One of the coefficients has the value -42. What's the appropriate binary representation for this value?

Appropriate binary representation for coefficient of -42: 1111010110

(C) (8 Points) Indicate the appropriate width in bits for each of the registers A – D assuming that we don't want to lose any arithmetic precision (i.e., we don't want to round, shift or truncate any of the intermediate results or the final answer).

Most positive value for y = 4*(-32768)*(-512).
Most negative value for y = 4*(32767)*(-512).
Only need 28 bits for final value to represent actual range of output values.

Width of Register A: 26
Width of Register B: 27
Width of Register C: 28
Width of Register D: 28 or 29

(D) (6 Points) What's the minimum clock period for which the circuit will still work correctly given the following timing specifications for the components:

component	t_{CD}	t_{PD}	t_{SETUP}	t_{HOLD}
Register	0.7ns	1.1ns	0.6ns	0.4ns
Adder	0.2ns	2.1ns	--	--
Multiplier	0.2ns	4.5ns	--	--

Minimum clock period: 8.3ns

Problem 4. (25 Points)

Consider the following Verilog module that iteratively computes the square root of an 8-bit integer value.

```
module sqrt(clk,data,start,answer,done);
  input clk,start;
  input [7:0] data;
  output [3:0] answer;
  output done;

  reg [3:0] answer;
  reg busy;
  reg [1:0] bit;

  wire [3:0] trial;
  assign trial = answer | (1 << bit);  // << is left shift

  always @ (posedge clk) begin
    if (busy) begin
      if (bit == 0) busy <= 0;
      else bit <= bit - 1;
      if (trial*trial <= data) answer <= trial;
    end
    else if (start) begin
      busy <= 1;
      answer <= 0;
      bit <= 3;
    end
  end

  assign done = ~busy;
endmodule
```

Please neatly complete the timing diagram below as the module computes the square root of 169.