Final CMPE 415

Name:

This exam has 23 questions

You must show all of your work -- partial credit may be given to partially correct answers, while answers with no justification may not receive full points. Use the back of the exam sheets if you need extra space.

WARNING: KEEP YOUR EYES ON YOUR OWN PAPER. CHEATING OF ANY SORT WILL CAUSE YOU TO FAIL THIS COURSE.

1) (4 pts) List four of the five major markets that FPGAs are having an impact on, i.e., are eating into.

2) (4 pts) Briefly describe how mask-programmable devices are "customized" for a particular application.

3) (4 pts) Identify an important advantage of EPROM/EEPROM/FLASH technologies over fusible link technologies.

4) (4 pts) Why is it necessary to include two transistors per cell in EEPROM technologies?

5) (4 pts) Identify one distinguishing feature in each of PROM, PLA and PALs, i.e., how are they different from each other?

6) (4 pts) Identify which one of the following is more sensible: embedding an FPGA in an ASIC or embedding an ASIC in an FPGA.

7) (4 pts) Briefly distinguish (one sentence) between MUX-based and LUT-based logic blocks, i.e., how are they used to implement logic?

8) (4 pts) Name the structure used inside the FPGA to allow the FPGA to be programmed.

9) (4 pts) Name the attractive feature of "parallel load with FPGA as master" configuration mode over "serial load with FPGA as master".

10) (4 pts) Name, do NOT describe, four differences in the design philosophy of ASIC and FPGAs.

11) (4 pts) Of the 10 differences in design philosophy between ASIC and FPGAs discussed in class, identify 2 that are "free" for the FPGA designer, i.e., he/she does NOT need to worry about, that are NOT free for the ASIC designer.

12) (4 pts) Briefly describe the two additional steps required in FPGA schematic-level flows over those required for ASICs at this level.

13) (4 pts) Name the additional step required in ASIC or FPGA flows that is required when using an HDL-based design flow.

14) (4 pts) Briefly describe two characteristics of gain-based SVP with respect to path delay and gate size characteristics of the generated netlist.

15) (4 pts) Name the problem that FPGA-based SVPs are designed to handle, i.e., what would designers like to avoid when making a small change to their design?

16) (5 pts) Structural Verilog code can be *explicit* or *implicit*. What keyword does Verilog use to define *implicit* structural code and what is its name?

17) (5 pts) Identify the following code as RTL or algorithmic behavioral code.

```
module compare_2_algo (A_lt_B, A_gt_B, A_eq_B, A, B);
  input [1:0] A,B;
  output A_lt_B, A_gt_B, A_eq_B;
  reg A_lt_B, A_gt_B, A_eq_B;

  always @ (A or B)
    begin
      A_lt_B = 0; A_gt_B = 0; A_eq_B = 0;
      if (A == B) A_eq_B = 1;
      else if (A > B) A_gt_B = 1;
      else A_lt_B = 1;
    end
endmodule
```

18) (5 pts) Give a brief description of the meaning of x and z as defined in simulation within Verilog?

19) (5 pts) In the following explicit style of FSM, are the outputs synchronous or asynchronous?

```
module FSM_style2 (...)
   input ...;
   output ...;
   parameter size = ...;
   reg [size-1 : 0] state, next_state;

   assign the_outputs = ... // a function of state and inputs

   always @ ( state or the_inputs )
      // decode next_state with case or if stmt

   always @ (negedge reset or posedge clk)
      if (reset == 1'b0) state <= start_state;
      else state <= next_state;
endmodule
```

20) (5 pts) Give 2 of the 6 "rules of thumb" associated with synthesis of combinational logic.

21) (5 pts) What variables need to be in the event control expression in order for the following always block to be synthesized as combinational logic.

```
module or_nand_2 (enable, x1, x2, x3, x4, y);
   input enable, x1, x2, x3, x4;
   output y;
   reg y;
   always @( ... )
      begin
         y = ~(enable & (x1 | x2) & (x3 | x4));
      end
endmodule
```

22) (5 pts) Is the following code synthesized as combinational or sequential logic? Why?

```
module mux_latch( y_out, sel_a, sel_b, data_a, data_b);
  input sel_a, sel_b, data_a, data_b;
  output y_out;
  reg y_out;

  always @(sel_a or sel_b or data_a or data_b)
    case ({sel_a, sel_b})
      2'b10: y_out = data_a;
      2'b01: y_out = data_b;
    endcase
endmodule
```

23) (5 pts) In order to avoid priority structure, the case items in a case structure must be mutually exclusive. Is this true for the following code?

```
module example(Data, Code);
  input [7:0] Data;
  output [2:0] Code;
  reg [2:0] Code;

  always @(Data)
    begin
      if (Data[7]) Code = 7; else
      if (Data[6]) Code = 6; else
      if (Data[5]) Code = 5; else
      if (Data[4]) Code = 4; else
      if (Data[3]) Code = 3; else
      if (Data[2]) Code = 2; else
      if (Data[1]) Code = 1 else
      if (Data[0]) Code = 0; else
                   Code = 3'bx;
    end
endmodule
```