# CMPE 415

Name:

## This exam has 24 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) Very briefly distinguish between PLDs and ASICs, i.e., what are the trade-offs with regard to chip speed, design effort, design size, and configurability.

2) (4 pts) From the designer's/user's perspective, i.e., your perspective, briefly describe the basic characteristics of fusible-link and mask programmable technologies. How are they different at the most fundamental level with regard to using them for applications?

3) (4 pts) Briefly describe, one sentence each, the trade-offs between FLASH and SRAM technologies.

4) (4 pts) Briefl y distinguish standard cell technology from gate array technology, i.e., how are they different at the very lowest levels, e.g., fabrication level?

5) (4 pts) Briefl y describe how FPGAs "fill the gap" between PLDs on one side and ASICs (gate arrays, std cells, full custom) on the other?

6) (4 pts) With respect to design tool complexity within CAD systems, why are 'FPGA-ASIC hybrids' more difficult to deal with than 'FPGA-*structured ASIC* hybrids'?

7) (4 pts) Identify two drawbacks of SRAM-based FPGAs with regard to system (board-level) design and security?

8) (3 pts) List three advantages that companies give for their antifuse-based FPGAs over SRAM based versions.

9) (9 pts) For each implementation technology of FPGAs, i.e., SRAM, antifuse and EEPROM/FLASH, answer the following questions.

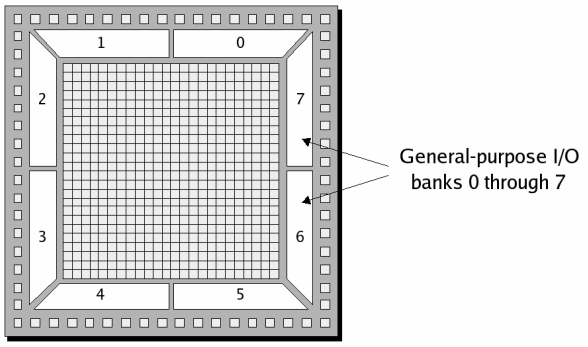
Reprogramming Speed: (fast, etc.)
- SRAM:
- Anitfuse:
- EEPROM/FLASH:

Size of configuration cell:(large, etc.)
- SRAM:
- Anitfuse:
- EEPROM/FLASH:

Technology node: (current node, etc)
- SRAM:
- Anitfuse:
- EEPROM/FLASH:

10) (4 pts) Briefl y distinguish between hard and soft microprocessor cores, i.e., list 1 advantage and disadvantage of each, with respect to the other.

11) (4 pts) Why do vendors provide 'banks' of general purpose I/O, i.e., what capability does this provide?




12) (4 pts) Distinguish between *soft IP* and *firm IP*.

13) (4 pts) Identify the following code fragment as behavioral algorithmic, behavioral RTL or structural. Briefl y explain your choice.

```
module 4bit_adder (a, b, c_in, sum, c_out);
  output [3:0] sum;
  output c_out;
  input [3:0] a, b;
  input c_in;

  assign {c_out, sum} = a + b + c_in;
endmodule
```

14) (4 pts) Structural Verilog code can be *explicit* or *implicit*. What keyword does Verilog use to define *implicit* structural code and what is its name?

15) (4 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
```

16) (4 pts) Identify 3 common ways to assign values to variables of type *net* within Verilog.

17) (3 pts) Indicate "yes" or "no" in the following table, which defines how *net* and *register* variables can be used in module I/O ports.

| Variable type | input | output | inout |
|---|---|---|---|
| net | | | |
| register | | | |

18) (4 pts) What feature(s) of the following code identify this code as using *procedural continuous assignment* as opposed to *continuous assignment*?

```
module mux4_PCA (a, b, c, d, select, y_out)
    input a, b, c, d;
    input [1:0] select;
    output y_out;
    reg y_out;

    always @ (select)
        if (select == 0) assign y_out = a; else
        if (select == 1) assign y_out = b; else
        if (select == 2) assign y_out = c; else
        if (select == 3) assign y_out = c; else y_out = 1'bx;
endmodule
```

19) (4 pts) Name 3 ways in which you can suspend a behavior.

20) (4 pts) Modify the following code in the simplest way possible to make the FF asynchronous.

```
module df_behav (data, clk, q, q_bar, set, reset)
    input data, clk, set, reset;
    output q, q_bar;
    reg q;

    assign q_bar = ~q;

    always @(posedge clk)
        begin
            if (reset == 0 ) q = 0;
            else if (set == 0) q = 1;
            else q = data;
        end
endmodule
```

21) (4 pts) Give the final values for *A* and *B* after the following code sequences execute.

```
initial
    begin
    A = 0;
    B = 1;
    ...
    A <= B;
    B <= A;
    end
```

```
initial
    begin
    A = 1;
    B = 0;
    ...
    A = B;
    B = A;
    end
```

22) (4 pts) Assuming the first stmt executes with t_sim = 0, what is the value of t_sim when the value of *B* and *D* are sampled in the following code fragments?

```
...                                  ...
#2 A = #5 B;  // t_sim =             A <= #5 B; // t_sim =
C = D; // t_sim =                    #1 C <= D; // t_sim =
...                                  ...
```

23) (4 pts) In class, we decided that the following code fragment with intra-assignment delay better modeled real hardware (over the version that used inter-assignment delay.) What problem still remains and how can this code be changed to improve hardware modeling?

```
module bit_or8_gate4(y, a, b)
    input [7:0] a, b;
    output [7:0] y;
    reg [7:0] y;

    always @(a or b) begin
        y = #5 a | b;
    end
endmodule
```

24) (5 pts) Draw the waveforms associated for *wave1* and *wave2* generated by the following code.

```
module multiple_no_block_2;
  reg wave1, wave2;
  initial begin
      #5 wave1 = 0;
          wave2 = 0;
          wave1 <= #5 1;
          wave2 <= #10 1;
          wave2 <= #20 0;
     #10 wave1 = 1;
          wave1 <= #5 0;
        end
 endmodule
```