Note: The time estimate provided indicates the amount of time in which you are expected to be able to complete each problem in order to allow for sufficient time to complete all problems. Furthermore, these problems are only sample problems and are NOT an indication of the subset of material you will need to know for the final.

1.) (5 minutes) Create a 2-bit carry-lookahead adder with two 2-bit inputs, a and b, a 2-bit output, s, and a carry output co and provide the logic equations for the outputs of the carry-lookahead logic. You do not need to provide the detailed internal logic gates or equations for the other components within the carry-lookahead adder.

2.) (4 minutes) For the function F(a,b,c) = b’c’ + a’b’c + abc + abc’, determine all prime implicants and all essential prime implicants of the function. Perform two-level logic size optimization for the function and express the answer as sum-of-products.

3.) (2 minutes) Define SRAM and very briefly describe the internal storage cell within SRAM.

4.) (2 minutes) Define FPGA and briefly describe the benefits and drawbacks of FPGAs compared to at least one other implementation option.

5.) (10 minutes) Design a high-level state machine for an event timer. The event timer has a 1-bit input e that will be 1 for one clock cycle indicating when each distinct event occurs. The event time has two 8-bit outputs, et and ot. The et output will output the elapsed time in clock cycles between the last two events. The ot output will output the number of times the measured elapsed time between the last two events exceeded a user-defined threshold specified by an 8-bit input th. Ensure that your event timer design works correctly regardless of how fast events occur. Note: You do not need to convert the state diagram to a datapath and controller.

6.) (15 minutes) Convert the following C-like code to a high-level state machine. Be sure to clearly specify the inputs, outputs, and local registers of your high-level state machine. Using the RTL design method, create a datapath for the high-level state machine and convert the high-level state machine to an FSM.

```
Inputs: byte A[256], bit go
Outputs: byte avg, bit done
AVERAGE:
  while(1) {
    while(!go);
    done = 0;
    avg=0;
    sum=0;
    i=0;
    while ( i < 256 ) {
      sum = sum + A[i];
      i = i + 1;
    }
    avg = sum / 256;
    done = 1;
  }
```

7.) (5 minutes) Using a one-hot state encoding, create a state table and determine the Boolean equations for the following FSM.

```
Inputs: x (bit)
Outputs: to (2 bits)
```

8.) (10 minutes) Configure the FPGA fabric shown below to implement a 2-bit carry-ripple adder with two 2-bit inputs, a and b, a 2-bit output, s, and a carry output co. All inputs and outputs should be external inputs and outputs.
9.) (7 minutes) Convert the following Mealy FSM to its nearest equivalent Moore FSM using the fewest number of states as possible. The FSM has three outputs, $x$, $y$, and $z$, that are assumed to be assigned a value of 0 in every state in which an explicit assignment is not shown.

A diagram is shown with states S0 and S1, transitions labeled with inputs $a$, $a'$, $b$, $b'$, and outputs $x$, $y$, $z$.