ECE 274 – Digital Logic
Final Sample Problems

1.) Define SRAM, DRAM, ROM, EPROM, EEPROM, and Flash memories.

2.) Define the terms “optimization” and “tradeoff,”

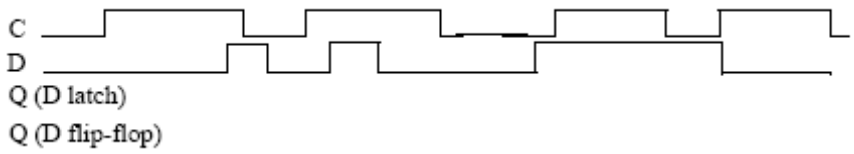
3.) Convert the following two’s complement binary numbers to decimal numbers:

a. 1010
b. 01000000
c. 11001100
d. 11111
e. 010111011001

4.) Convert the following decimal numbers to 8-bit two’s complement binary numbers:

a. -23
b. 87
c. 123
d. -101

5.) Compare the behavior of D latch and D flip-flop devices by completing the timing diagram below. Provide a brief explanation of the behavior of each device.



6.) A wristwatch display can show one of four items: the time, the alarm, the stopwatch, or the date, controlled by two signals s1 and s0 (00 displays the time, 01 the alarm, 10 the stopwatch, and 11 the date -- assume s1s0 control an N-bit-wide mux that passes through the appropriate register). Pressing a button B (which sets B=1) sequences the display to the next item (if the presently displayed item is the date, the next item is the current time). Create a state diagram for an FSM describing this sequencing behavior, having an input bit B, and two output bits s1 and s0. Be sure to only sequence forward by one item each time the button is pressed, regardless of how long the button is pressed -- in other words, be sure to wait for the button to be released after sequencing forward one item. Use short but descriptive names for each state. Make displaying the time the initial state.

7.) Design a 4-bit carry-ripple style magnitude comparator that has two outputs, a greater-than or equal-to output *gte*, and a less-than or equal-to output *lte*. Be sure to clearly show the equations used in developing the individual 1-bit comparators and how they are connected to form the 4-bit circuit.

8.) Use magnitude comparators and logic to design a circuit that outputs 1 when an 8-bit input is between 75 and 100, inclusive

9.) Design an 8-bit register with 2 control inputs s1 and s0, 8 data inputs I7..I0, and 8 data outputs Q7..Q0. s1s0=00 means maintain the present value, s1s0=01 means load, and s1s0=10 means clear. s1s0=11 means to swap the high nibble with the low nibble (a nibble is 4 bits), so 11110000 would become 00001111, and 11000101 would become 01011100.

10.) Design a special multiplier circuit that can multiply its 16-bit input by 1, 2, 4, 8, or 16, or 32, specified by three inputs a, b, c (abc=000 means no multiply, abc=001 means multiply by 2, abc=010 means by 4, abc=011 means by 8, abc=100 means by 16, abc=101 means by 32). *Hint*: Use a predefined component.

11.) Create a high-level state machine for a simple data encryption/decryption device. If a bit-input b is 1, the device stores the data from a 32-bit input I as what is known as an offset value. If b is 0 and another bit-input e is 1, then the device "encrypts" its input I by adding the stored offset value to I, and outputs this encrypted value over a 32-bit output J. If instead another bit-input d is 1, the device should "decrypt" the data on I by subtracting the offset value before outputting the decrypted value over J. Be sure to explicitly handle all possible combinations of the three input bits.

12.) (a) Create a high-level state machine that adds each register of one 128x8 register file to the corresponding registers of another 128x8 register file, storing the results in a third 128x8 register file. The system should only begin the addition when a bit input add is 1, and should not perform the addition again until it has finished adding (only adding again if add is 1). (b) Extend this system to either add or subtract, using an additional bit-input op, where op=1 means add, and op=0 means subtract.

13.) Using the RTL design method shown in Table 5.1, create an RTL design that computes the sum of all positive numbers within a 512-word register file *A* consisting of 32-bit numbers stored in two's complement form.

14.) Convert the following C code, which calculates the maximum difference between any two numbers within an array *A* consisting of 256 8-bit values, into a high-level state machine.

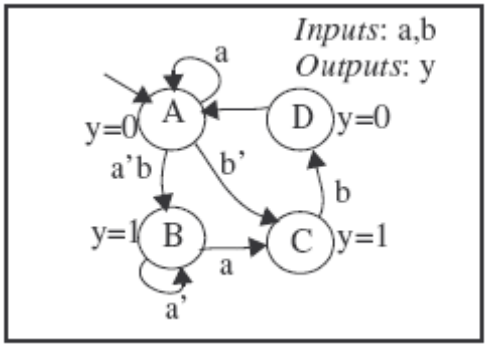
```
Inputs: byte a[256], bit go
Outputs: byte max_diff, bit done
MAX_DIFF:
  while(1) {
    while(!go);
    done = 0;
    i = 0;
    max = 0;
    min = 255; // largest 8-bit value
    while( i < 256 ) {
      if( a[i] < min ) {
        min = a[i];
      }
      if( a[1] > max ) {
        max = a[i];
      }
      i = i + 1;
    }
    max_diff = max - min;
    done = 1;
  }
```

15.) Perform two-level logic size optimization for the function F(a,b,c,d) = ab + a'b'd' using a K-map. Express the answer as sum-of-products.

16.) For the function F(a,b,c) = a'c + ac + a'b, determine all prime implicants and all essential prime implicants of the function.

17.) Use repeated application of the expand operation to heuristically minimize the function F(a,b,c,d,e) = abcde + abcde' + abcd'e'. Try expanding each term for each variable.

18.) Convert the following Moore FSM to the nearest Mealy FSM equivalent.



19.) Describe the difference between full-custom IC and standard cell ASIC designs.

20.) Show how to implement on two 3-input 2-ouput lookup tables the following functions: F(a,b,c,d) = abc + d and G = a'. You must implement both F and G with only two lookup tables connected as follows.