Student ID: ____________

CS 151
Quiz 4

Name : __________________ , __________________
      (Last Name)                   (First Name)

Student ID : ______________

Signature : ________________

Instructions:

1. Please verify that your paper contains 9 pages including this cover.
2. Write down your Student-Id on the top of each page of this quiz.
3. This exam is closed book. No notes or other materials are permitted.
4. Total credits of this quiz are 70 points.
5. To receive credit you must show your work clearly.
6. Calculators are NOT allowed.
Q1 [Memory Design] [15 points]

Design a 24K*64-bit RAM using standard 8K*16-bit RAM modules like the one shown below.

Use a minimum number of the following logic components in your design (no other components may be used for this design):

1) Priority Encoder
2) Decoder
3) Multiplexer

You don’t need to connect all input lines (data, addr, rw, en), however, you DO need to label the bit ranges for each data field, and address field as well as the enable signals.
Q2 [Critical Path and Frequency Calculation] [15 points]

Shown below is simple filter composed of 3 input registers (A, B, C) and an output register (O), a series of adders, multipliers and a multiplexer with two select signals.

2a. Determine the register-to-register critical path and the longest path delay for the circuit. Draw a circle around every component in the critical path. [10 points]

Assume no wire delay, and the following component delays:
Multipliers take 5 ns
Adders take 2 ns
Multiplexers take 1 ns
2b. Based on the longest path delay obtained from part 2a, what should the frequency of this circuit be? (Make sure your answer is in GHz, MHz, KHz, or Hz) [5 points]

For the answer see page 4.
Q3 [RTL Design-C-code to Gates] [40 points]

The following C-code tries to count the number of unsigned byte numbers divisible by four from array A.

Inputs: byte A[128], bit start
Outputs: byte mod4_count, bit done

main() {
    short uint i, count, data;
    while(1) {
        while(!start); // wait for start signal
        done = 0;
        i=0;
        count = 0;
        data = 0;
        mod4_count = 0;

        while(i<128) {
            data = A[i];
            if(data%4==0) {
                count = count + 1;
            }
            i = i + 1;
        }

        mod4_count = count;
        done=1;
    }
}

3a. Using the templates shown below, convert EACH of the C-code statements to its high level state machine representation. You should start from the outermost loop. [30 points]
Stmt #1
while(1) {
    (while statements)
}

Stmt #2
while(!start) {
    (while statements)
}

Stmt #3
done=0;

Stmt #4
i=0;

Stmt #5
count=0;

Stmt #6
data=0;

Stmt #7
mod4_count=0;

Stmt #8
while(i<128) {
    (while statements)
}

Stmt #9
data=A[i];

Stmt #10
if(data%4==0) {
    (then statements)
}

Stmt #11
count=count + 1;

Stmt #12
i=i+1;

Stmt #13
mod4_count=count;

Stmt #14
done=1;
3b. Now that you have transformed the C statements to their high level state machine representation, merge them together to represent the final high level state machine. Make sure you remove all unnecessary states. [10 points]
Some states could be combined into one (see the red circles), also, the fist state can be merged with the state containing the condition not start (!start) because the transition will always be 1 (or true), and the other path is never taken, hence, it can be removed. The above SM was sufficient for full credit assuming you merged the states circled.
Final State Machine:

![State Machine Diagram]

- **!start**: Transition to start
- **!i<128**: Transition on input $i < 128$
- **data=A[i]**: Transition on data $A[i]$
- **!(data%4==0)**: Transition on condition $data \% 4 \neq 0$
- **data%4==0**: Transition on data $data \% 4 = 0$
- **count = count + 1**: Transition on count $count = count + 1$
- **i = i + 1**: Transition on index $i = i + 1$
- **done=1**: Transition on condition $done = 1$
- **mod4_count=count**: Transition on result $mod4\_count = count$