Computer Science and Engineering 2021.03

Sample Final Test

Answer all questions in the space provided

Student Last Name: _____

Student Given Name: _____

Student Id. No: _____

Question	Value	Score
1	30	
2	20	
3	50	

Question 1. [30 points, 10 such questions]

1. [3 points] In the standard carry-lookahead we make use of two signals g and p. What is their proper name?
g for generate
p for propagate

2. [3 points] What is the disadvantage of one-bit branch prediction.
mispredicts twice the branch in an inner for-loop

3. [3 points] What are the three types of hazards in pipelines.
Structural hazards
Data hazards
Control hazards

4. [3 points] What is the simplest hardware technique to reduce the stalls due to hazards like
```
    slli    x5, x5, 3
    add     x5, x5, x10
```
forwarding (aka bypassing)

5. [3 points] What is the basic principle that guides the design of memory hierarchies
Make the common case fast

6. [3 points] Inside what Verilog contruct can we have a `repeat` statement.
inside an always or an initial

7. [3 points]

8. [3 points]

9. [3 points]

10. [3 points]

Question 2.

[20 points]

1. [7 points]

2. [7 points] Write the truth table for the following logic function

$$F = A + BC + B'C'$$

```
ABC    F
000    1
001    0
010    0
011    1
100    1
101    1
110    1
111    1
```

Question 3.

[50 points]

1. [20 points] Assume you are given a module called `incr` that implements a combinational circuit that has a four bit input and a four bit output. The output is the input plus one. Write a simple Verilog module that has an one bit input C, another one bit input `pulse` and a four bit output Z. The output is normally zero when the input C is zero, but when the input C becomes one, then at the next sixteen pulses (`pulse`) the circuit counts from zero to fifteen and back to zero (i.e. 0, 1, 2, ..., 14, 15, 0, 1...)

```verilog
module cntr(C, pulse, Z);
   input C, pulse;
   output reg [3:0] Z;

   wire [3:0]                      Zin;

   incr myincr(Zin,Z);

   always @(posedge pulse)
     if (C)
       Z <= Zin;

   always @(posedge C)
     Z <= 4'b0000;
endmodule // cntr
```

2. [7 points] Write three versions of a module that implements a half adder using

(1)   the `always` construct of Verilog

(2)   the `assign` construct

(3)   the primitive gates AND, OR, NOT.

```
module hadd1(S, Cout, a, b);
   output S, Cout;
   input  a, b;

   assign S    = a^b;
   assign Cout = a&b;
endmodule // hadd1

module hadd2(S, Cout, a, b);
   output S, Cout;
   input  a, b;

   xor (S, a, b);
   and (Cout, a, b);
endmodule // hadd1

module hadd3(S, Cout, a, b);
   output reg S, Cout;
   input  a, b;

   always @(a,b)
     begin
        S    = a^b;
        Cout = a&b;
     end
endmodule // hadd1
```