Rose-Hulman Institute of Technology
Computer Science and Software Engineering
Profs. Archana Chidanandan and Larry Merkle

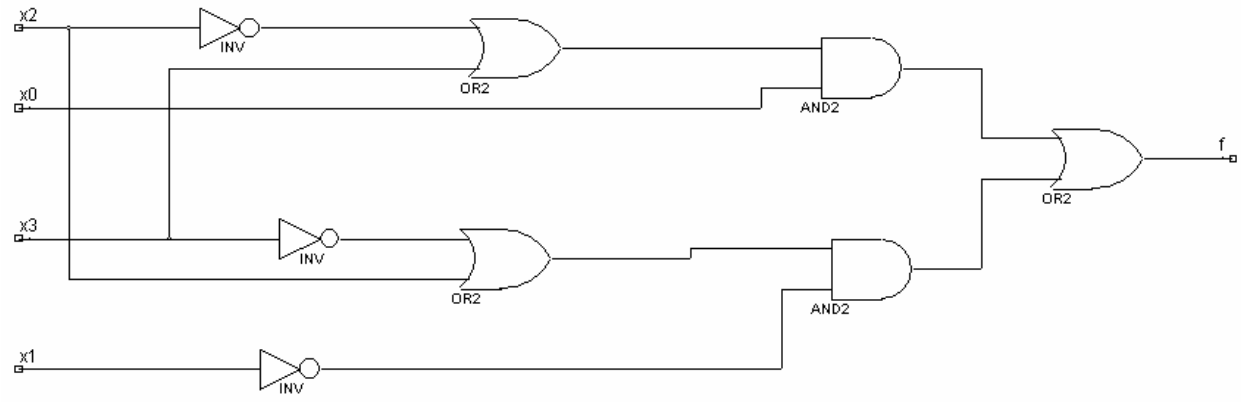
Homework 0 (Skills Review) - Solutions
Maximum points : 20

Combinational Logic

1. Design a digital logic circuit that implements the 4-input Boolean function described by the truth table below. You may use three inverters to complement the inputs, and up to five other gates. *Hint*: Consider using a Karnaugh map.

x_0	x_1	x_2	x_3	$f(x_0x_1x_2x_3)$
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

One possible solution:



Rose-Hulman Institute of Technology
Computer Science and Software Engineering
Profs. Archana Chidanandan and Larry Merkle

2. Explain why each of the following statements about full adders is false (assume that the inputs are A, B, and CI, and that the outputs are D and CO).

    a. D is true if and only if exactly one of the inputs is 1.
    b. CO is true if and only if exactly two of the inputs are 1.

The following is the truth table for the full adder:

CI	A	B	CO	D
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

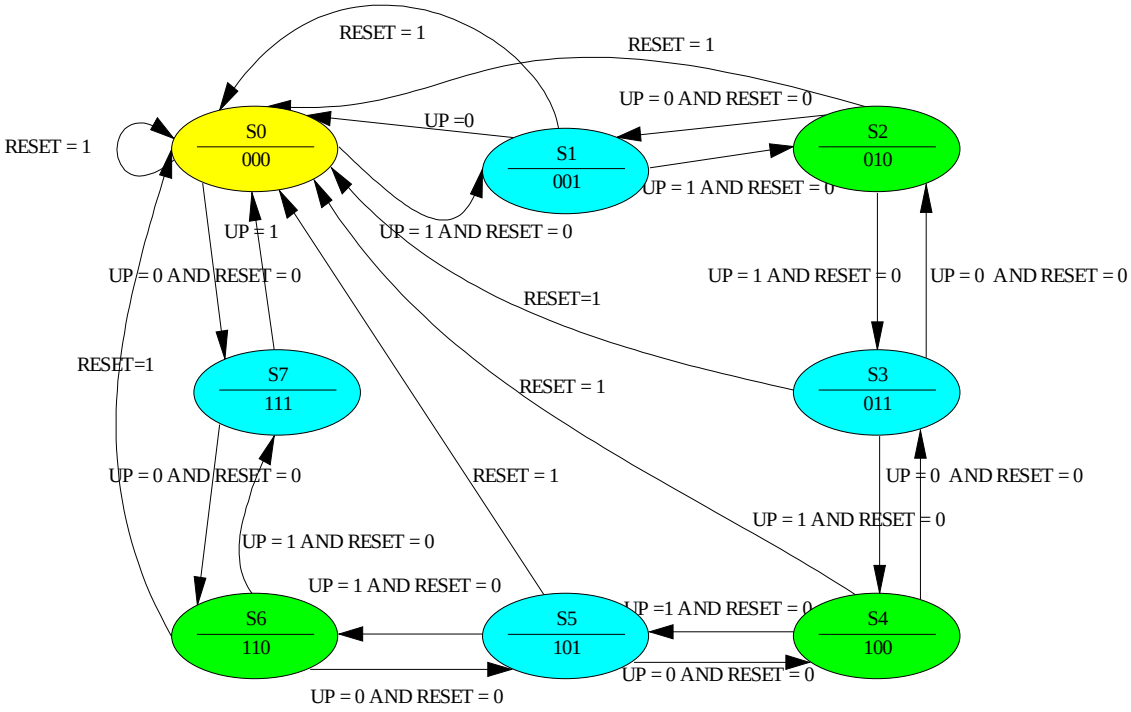
    a. It is evident from the truth table that D is 1 when CI, A and B are all 1.
    b. Also, when CI, A and B are all 1, CO is 1, thus negating the statement that CO is true if and only if exactly two of the inputs are 1.

Sequential Logic

3. Obtain the state diagram for a 3-bit up-down counter. The inputs are:
    UP = 1 (count up)
    = 0 (count down)

    RESET = 1 (set the counter to 000)
    = 0 (continue counting)

    (See next page for solution.)



2's Complement Representation

4. Find the 8-bit 2's complement representation of -55.

   To determine the 2's complement of a number:

   a. If the number is positive, then the 2's complement representation is simply the binary representation of the number, with the leftmost bit (MSB), being the sign bit and equal to zero.
   b. If the number is negative, then flip each bit of the binary representation of the number (bit-wise NOT) and then add 1.

   $55_{10} = 00110111_2$

   To obtain -55,

   $$\begin{array}{r} 11001000 \\ +\quad\quad 1 \\ \hline 11001001_2 = -55_{10} \end{array}$$

Note: A shortcut to determine the 2's complement for a negative number: Examine the bits of the number from right to left. Look for the first occurrence of '1'. Skip this bit and flip all the other bits to the left of this bit. The result is the 2's complement of the negative number.

Rose-Hulman Institute of Technology
Computer Science and Software Engineering
Profs. Archana Chidanandan and Larry Merkle

Hexadecimal Representation

5. Find the hexadecimal form of the 8-bit 2's complement representation of -55.

   The hexadecimal representation is a compact representation of a binary number.
   To determine the hexadecimal equivalent of an 8-bit 2's complement number:
   a. Group the bits from right to left into 4-bit groups.
   b. Each 4-bit value is then replaced, by the equivalent hexadecimal value.
      Therefore, $-55_{10} = \underline{11001001}_2 = C9_{16}$

Data Types

6. Find the range of integers that can be expressed in 16-bit 2's complement representation.

   The range of integers that can be represented in n-bit 2's complement representation is -2^{n-1} to $+2^{n-1}-1$. Therefore, for n = 16, the range is -2^{15} to $+2^{15}-1$

7. Write a Java code fragment that determines the largest number is an array of 10 integers. It should also determine the index of the largest value in the array.

```
count = 10;            //The number of elements in the array
index = 0;             //Index to the array
maxIndex = -1;         //Stores the index of the largest
                       //element in the array
largest = -1;          //Stores the values of the largest
                       //element of the array

while (i != count) {   //If there are elements to read from
                       //the array
   if( largest < A[i] ){ //if the current array element is
                       // larger than "largest"
        largest = A[i];
        maxIndex = i;
    }
  i++;                 //increment "i" to read the next
                       //element in the array
}
```

Control Structures

8. Write a Java code fragment that implements the behavior shown in the following table:

`this.cond1()`	`this.cond2()`	Behavior
`false`	`false`	`this.method1()` executes, and then `this.method2()` executes.
`false`	`true`	`this.method1()` executes, but `this.method2()` does not.

`true`	`false`	`this.method2()` executes, but `this.method1()` does not.
`true`	`true`	Neither `this.method1()` nor `this.method2()` executes.

```
if (!this.cond1() && !this.cond2()){
     this.method1();
     this.method2();
}
if(!this.cond1() && this.cond2()){
     this.method1();
} else if ( this.cond1() && !this.cond2())
     this.method2();
}
```

The above code fragment will also ensure that neither `this.method1()` nor `this.method2()` will be executed if both `this.cond1()` and `this.cond2()` return true.

9. Write a Java code fragment such that `this.method1()` executes exactly *N* times, where *N* is a non-negative integer. You may **not** use a `while` loop.

One possible solution:

```
for ( int i = 0; i < N; i++) {
     this.method1();
}
```

10. Given the following Java code fragment, what output would result from evaluating the expression `this.myMethodB()`?

```
private int myInField = 1;
private int myOutField = 2;

public int myMethodA( int myParameter ) {
   this.myInField = myParameter;
   return this.myOutField;
}

public void myMethodB() {
   int myLocal = 3;
   myLocal = this.myMethodA( 4 );
   System.out.print( this.myInField + ", ");
   System.out.print( myLocal );
}
```

Output:

```
4,2
```