

**CoE/EE 460**  
Spring 2001 : Lockwood

Homework #4: Due Thursday, March 8, 2001  
at 5:00pm, in EE homework box

Name:	
-------	--

1. Consider a function called `Is_Equal` that compares two SOP terms. The function should return true if and only if all of the bits in the dontcare mask are identical and if the values for the bits that we do care about match.

- For Example, the following functions return:

`Is_Equal(010-,010-)=1`

`Is_Equal(--10,--10)=1`

`Is_Equal(010-,111-)=0`

`Is_Equal(010-,01-0)=0`

(a) Attach your solution on a separate page.

(b) Assuming that the value and mask bits are stored in a data structure like:

```
unsigned int value; // up to 32-bits: 0=false, 1=true
```

```
unsigned int mask; // up to 32-bits: 0=Care, 1=DontCare
```

Write functional code that determines the return value for the following inputs:

- `SOP1.value=1001, SOP1.mask=0100`
- `SOP2.value=1101, SOP2.mask=0100`

(c) Likewise, Determine the return value for the following inputs:

- `SOP1.value=1001, SOP1.mask=0100`
- `SOP2.value=1101, SOP2.mask=0100`

2. Consider a function called `One_Difference` that compares two SOP terms.

- For terms that have an identical dontcare masks AND have values that we care about but differ by only term: `One_Difference` should return a bitmask that specifies the bit that differs.
- For terms with different dontcare masks or for terms that differ in unmasked value bits by more than one value: `One_Difference` should return zero.

For Example, the following function calls return:

- `One_Difference(010-,110-)=1000`
- `One_Difference(-010,-000)=0010`
- `One_Difference(010-,111-)=0000`
- `One_Difference(010-,01-0)=0000`

(a) Attach your solution on a separate page.

(b) Use your function to determine the return value for the following input:

- `SOP1.value=1001, SOP1.mask=0100`
- `SOP2.value=1010, SOP2.mask=0100`

(c) Use your function to determine the return value for the following input:

- `SOP1.value=1001, SOP1.mask=1000`
- `SOP2.value=1101, SOP2.mask=0100`

3. Using the programming environment of your choice, extend the program that you developed for homework 3 to perform a reduction of minterms to a sum of prime implicants. As with homework 3, your program should accept a list of implicants in the same SOP format. The output should be a similarly-formatted list of the prime implicants.

- Example 1

- The input:  
4  
0000  
0001  
0100  
0110  
done
- Should reduce to:  
000-  
01-0  
0-00

- Example 2

- The Input:  
4  
1111  
1011  
0-11  
done
- Should reduce to:  
--11

- Hints

- Use your program from homework 3 to expand implicants into minterms.
- Implement a data structure which can store a group (or list) of implicants.
- Organize the groups of implements in a table.
  - \* Let the row number determine the **one-count** (ranging from 0 to n)
  - \* Let the column number determine **dontcare\_count** (also ranging from 0 to n).
- Begin by inserting the original minterms into the appropriate rows in the 0'th column of the table.
- Apply the tabular method to join terms that differ by only 1 value.

- (a) Submit your original source code.
  
- (b) Verify your program operates on the example problems.
  
- (c) Verify your program operates on larger problems.
  
- (d) Attach a sheet with the output of your program for:

```
25
01111111100000000001--111
0111111110000---0001--111
011110000--0000001111111
0111111110000000000111111
1100000110100000000---111
done
```

(e) Using a timer (such as the stopwatch or the Unix 'time' command), determine the number of seconds (or fraction thereof) required to execute your program for the 25-variable function given in the last exercise.

(f) Identify the processor type (Example Intel PIII or AMD Athlon) and the Speed (in MegaHertz) of the PC that you used to generate the results. Alternatively, name the CEC machine that you used to run your code.

- Machine make (Processor):
- Machine speed (MHz):
- -or- Machine name (for CEC machines):

(g) Determine the time it takes to run your program on the following function:

```
8
-----
done
```

Processing Time:

(h) Determine the time it takes to run your program on the following function:

```
10
-----
done
```

Processing Time:

(i) Explain why your program may run slowly on the previous two functions.

(j) Based on your results for the two calculations above, estimate the amount of time required to solve a worst-case function of 12 variables. Explain the relationship between the running time and the number of variables.