

# CoE/EE460 Switching Theory

## Lecture 9

John Lockwood

Washington University

Spring 2001

<http://www.arl.wustl.edu/~lockwood/class/coe460/>

## Announcements

- Homework 3 : Due Wednesday
  - Solutions will be posted on Wednesday
- Programming Review
  - Prerequisite for CoE/EE460 is EE260M
  - Prerequisite for EE260M is CS101 or 136.

## Suggestions for Software Implementation

- Create plan for the software design
  - Define: Class / Data structures
  - Define: Functions / Methods
- Implement one component at a time
- Debug each component
- Display intermediate steps

## Cygwin Tool Environment

- Provides
  - 32-bit C/C++ GNU software
    - gcc / g++ / gdb / etc.
  - Runs on Win95/98/ME/NT/2000
  - Apps source-code compatible with tools in Linux/UNIX
- Download from
  - <http://www.cygwin.com/>
- Freeware (GPL)



## Implementation Example (1)

- Implement program
- Compile
  - `g++ -o <output> <source.cpp>`
- Generate input
  - Edit data input file
- Run Program
- Generate Minterms

```
lockwood@WJL ~/coe460/mp/mp4
$ g++ -o parsevar parsevar.cpp
lockwood@WJL ~/coe460/mp/mp4
$ cat datain
3
---
done
lockwood@WJL ~/coe460/mp/mp4
$ parsevar.exe <datain
Function has 3 variables

==== Generate Minterms ====
000
001
010
011
100
101
110
111
```

## Implementation Example (2)

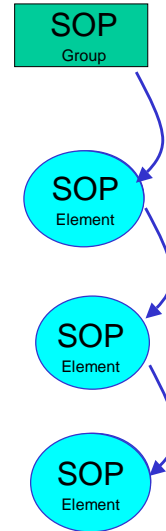
- Minterm Expansion Example
  - 3 variable function = “---”
  - Expand into Minterms
    - (Same as before)
- Classify Minterms by `one_count`
- Insert `sop_terms` into Table
  - Rows : `one_count`
  - Columns : `dontcare_count`
  - Avoid duplicates

```
lockwood@WJL ~/coe460/mp/mp4
==== Count one terms ====
000
one_count=0
001
one_count=1
010
one_count=1
011
one_count=2
100
one_count=1
101
one_count=2
110
one_count=2
111
one_count=3

==== Create Table ====
Table[0][0] =
000
Table[0][1] =
001
100
Table[0][2] =
011
101
110
Table[0][3] =
111
Table[1][0] =
Table[1][1] =
Table[1][2] =
Table[1][3] =
Table[2][0] =
Table[2][1] =
Table[2][2] =
Table[2][3] =
Table[3][0] =
Table[3][1] =
Table[3][2] =
Table[3][3] =
```

## SOP Group

- SOP Group = Collection of SOP Elements
  - Allow scanning of elements
  - Allow insertion of unique members
    - Avoid duplicates
  - Allow modification of group members.



## Implementation Example (3)

- Combine Groups in Table
  - That differ by a single value
  - Unmark primes that combine
  - Insert result into next column
- Display table

```
==== Merge Groups in Table ====
Merge[0][0] with [1][0] to get [0][1]
Compare:
000
001
Appending to Table[0][1]
00-
Compare:
000
010
Appending to Table[0][1]
000
Compare:
100
Appending to Table[0][1]
== Round[0] ==
100
Table[0][0]
Table[0][1]
Table[0][2]
011
110
Table[0][3] =
111
Table[1][0] =
00-
000
-00
Table[1][1]
Table[1][2]
Table[1][3]
Table[2][0]
Table[2][1]
Table[2][2]
Table[2][3]
Table[3][0]
Table[3][1]
Table[3][2]
Table[3][3]
Merge[1][0] with [2][0] to get [1][1]
Compare:
001
011
Appending to Table[1][1]
0-1
Compare:
101
101
Appending to Table[1][1]
-01
Compare:
001
110
Compare:
:
```

## SOP Group Matrix : In class example

- For each  $DC \in \{0..3\}$
- For each  $OC \in \{0 .. DC-1\}$ 
  - Compare
    - SOP\_Group (DC,OC) to SOP\_Group (DC,OC+1)
  - If (Difference=1 term),
    - Insert to SOP\_Group (DC+1,OC)
    - Unmark Primes

DontCare Count

	DC <sub>0</sub>	DC <sub>1</sub>	DC <sub>2</sub>	DC <sub>3</sub>	DC <sub>4</sub>
OC <sub>0</sub>	0000 000- 0-00				
OC <sub>1</sub>	0001 0100	00-1 -001 -100			
OC <sub>2</sub>	0011 1001 1100				
OC <sub>3</sub>					
OC <sub>4</sub>	1111				

One Count

## Comparing SOP Groups

- For each  $SOP_i \in \text{Group 1}$   
 For each  $SOP_j \in \text{Group 2}$   
 Compare  $SOP_i$  to  $SOP_j$

