2nd Programming Assignment, Arrays and Module Composition

- See runtime outputs in the DEMO folder. To copy files from it: "cp ~changw/html/137/prg/2/demo/* ."

- The goal of this assignment is to practice the use of array syntax to declare a set of wires or registers, and the composition of hardware modules by incorporating decoders into multiplexers (instead of repeating the same code statements in the previous assignment).

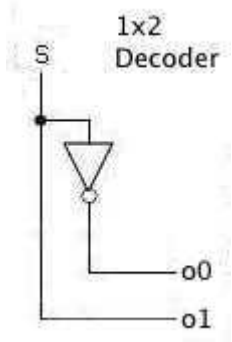
- Two example programs are given for **`1x2 decoder`** and **`2x1 multiplexer`**. The other three circuits will be for you to work on and submit.

- Submit three source files: ***d2x4.v***, ***m4x1.v***, and ***adder.v*** into the folder that has your name which is under the ***2ndPrgAssig*** folder which is under the 137 dropbox area on host Voyager as before. Do not submit any **a.out** or other files such as your runtime output.

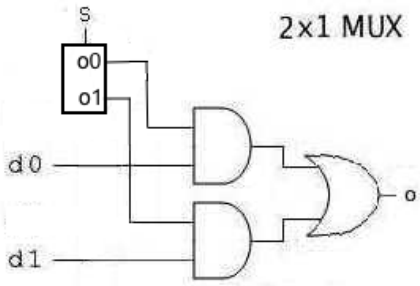
- If you need to resubmit after a correction, use **`mkdir`** command at the **`smb`** prompt to make a new folder under the designated folder (in your named folder). Use names such as ***V2*** (for version 2) and then execute *smb* command: **`cd V2`** (change directory into V2) and issue the "put" command again. The *V2* folder should be created in your named folder where you were supposed to submit one set of files. Use V3 as further newly-corrected files needed, etc.

- At the start of each program, type in your name in a commented line.

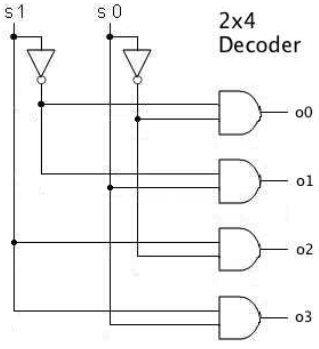
- To connect to the working server, launch two **`PuTTy`** terminals (for editing and compiling/running) from a Windows PC to server ***atoz, sp1, sp2, or sp3*** (via ***titan.ecs.csus.edu*** or ***athena.ecs.csus.edu*** first if you are not already in the ECS computer network, e.g., making connection from at home or other public university network areas.)



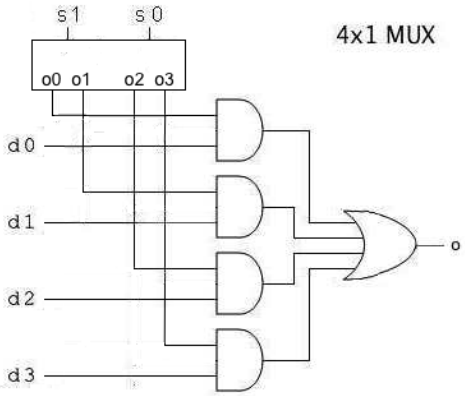
Example Code



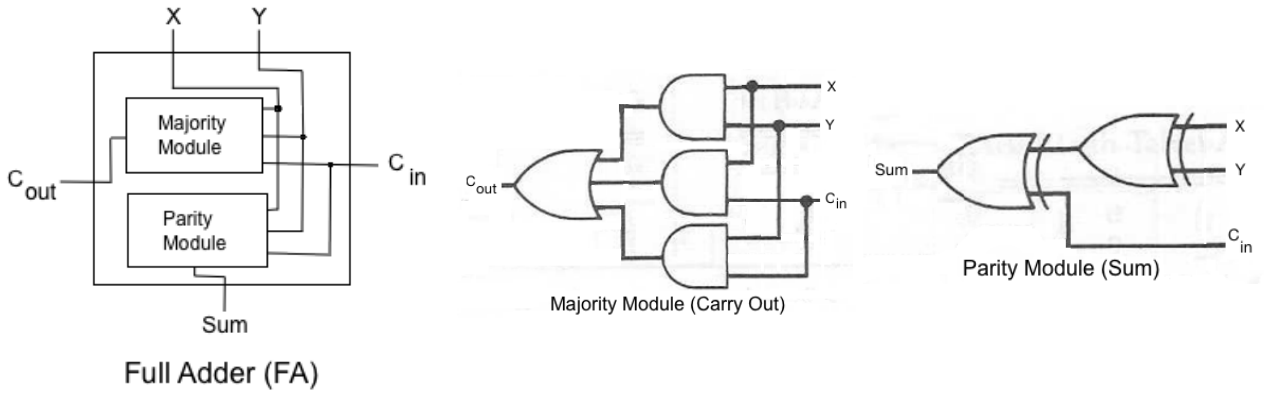
Example Code



Code/submit: 1. d2x4.v
Runtime output should be similar to that in the demo folder.



Code/submit: 2. m4x1.v
Runtime output should be similar to that in the demo folder.



A Full Adder can be composed by a parity function (sum) and a majority function (carry-out):

C_{in}	X	Y	->	C_{out}	Sum
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

Code/submit: 3. adder.v which outputs the above truth table.

- For good file organization skills, after logging in your ECS Linux account, issue shell commands such as *mkdir ...* to make work directories (folders) and work your files in there. Learn to use shell commands and programming editor such as vi, will greatly enhance one's developer skills. Very helpful and concise learning materials are listed below.

  - Useful Linux and vi Commands
  - Compile Your Verilog Programs
  - Access Dropbox from Shell
  - Simple Verilog Handbook
  - Complete Verilog Manual
  - More Program Examples