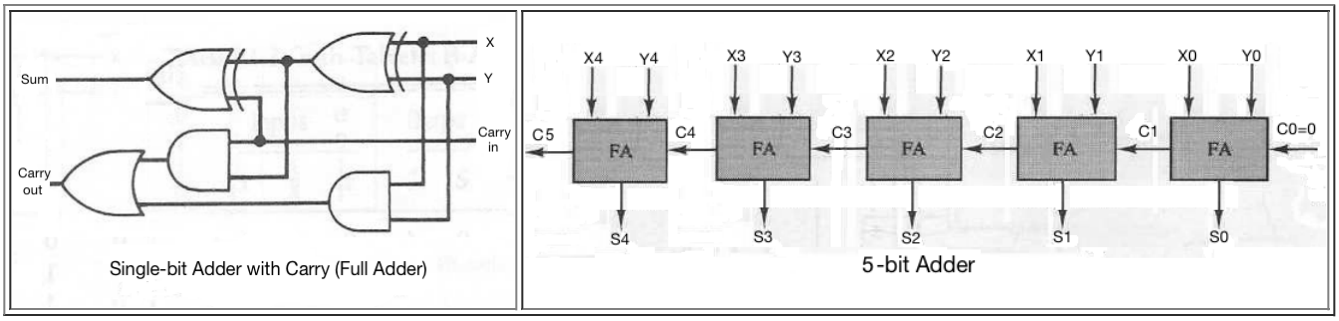
CSc 137 Verilog Programming #2: 5-Bit Adder

1. In this assignment we write a Verilog program to simulate a 5-bit adder. Submit only one program file ***adder.v*** to folder ***2ndPrgAssig*** located under your named folder in the dropbox area on host *Voyager*. Again, *mkdir V2* folder (for version 2) under your named folder to resubmit a new version if needed, and *V3* for yet another new version, etc. Misplaced files may cause point deduction.

2. A single-bit adder with a carry-in bit is called a *Full Adder* (FA). See the left diagram below. To compose a 5-bit adder, 5 FA's are linked together as shown by the right diagram below. The first carry-in bit **C0** is zero.



3. Copy and run the demo executable:

```
atoz% cp -p ~changw/html/137/prg/2/demo-a.out  .
atoz% demo-a.out
Enter X:
02
Enter Y:
05
X = 2 (00010)  Y = 5 (00101)
Result= 7 (00111)  C5 = 0

atoz% demo-a.out
Enter X:
17
Enter Y:
19
X = 17 (10001)  Y = 19 (10011)
Result= 4 (00100)  C5 = 1
```

4. The ASCII value obtained from a keyboard input must be converted, e.g., the character *'3'* will be read by the program as 51 (its ASCII table order) so subtract 48 from it to get the value it represents.

5. The skeleton of your program (5-bit adder) may look like:

```
// (MY NAME)
// adder.v, 137 Verilog Programming Assignment #2
module TestMod;                      // the "main" thing
   parameter STDIN = 32'h8000_0000; // I/O address of keyboard input channel

   reg [7:0] str [1:3]; // typing in 2 chars at a time (decimal # and Enter key)
   reg [4:0] X, Y;      // 5-bit X, Y to sum
   wire [4:0] S;        // 5-bit Sum to see as result
   wire C5;             // like to know this as well from result of adder

   instantiate the big adder module (giving X and Y as input, getting S and C5 as output)

   initial begin
      prompt for entering X: $display("Enter X: ");
      get 1st character:        --> str[1] = $fgetc(STDIN);
      get 2nd character:        --> str[2] = $fgetc(STDIN);
```

```
         and the ENTER key:        --> ...
         convert str to value for X:
            str[1] - 48 first, then times 10, then + str[2] - 48

         do the above to get input and convert it to Y

         #1; // wait until adder gets them processed
         $display X and Y (run demo to see display format)
         and
         $display S and C5 (run demo to see display format)
      end
   endmodule

   module BigAdder(X, Y, S, C5);
      input [4:0] X, Y;   // two 5-bit input items
      output ...          // S should be similar
      output ...          // another output for a different size

      ...                 // declare temporary wires

      ... (get an instance of a full adder, C0 is 0)
      ... (get another full adder...)
      ... (get another full adder...)
      ... (get another full adder...)
      ... (get another full adder...)
   endmodule

   module FullAdderMod(...); // single-bit adder module
      ... code the full adder according to the diagram above
   endmodule
```

- A List of Useful Linux and vi Commands
- Access Dropbox from a Linux Shell