PHY 440 Lab14: Programmable Logic Design Techniques II

Almost all digital signal processing requires that information is recorded, possibly manipulated and then stored in some way. Then it makes sense to design small modules that perform the basic signal processing operations. Such modules can be combined into more complex designs performing more complex operations etc. The result is a design with a hierarchy of levels. Building such designs is much easier than building a large circuit by stitching together a large number of primitive gates.

In this Lab you will design a module performing a frequently-used operation such as "1-bit" addition. Then you will use it to create a "4-bit" adder. Also, you will use other already designed modules to create a relatively complex circuit for recording the number of events – an "8-bit" up counter. *To be successful in this Lab you should review DH p.270-p.284 and the Guides to the Xilinx software and hardware posted on the PHY440 WWW site.*

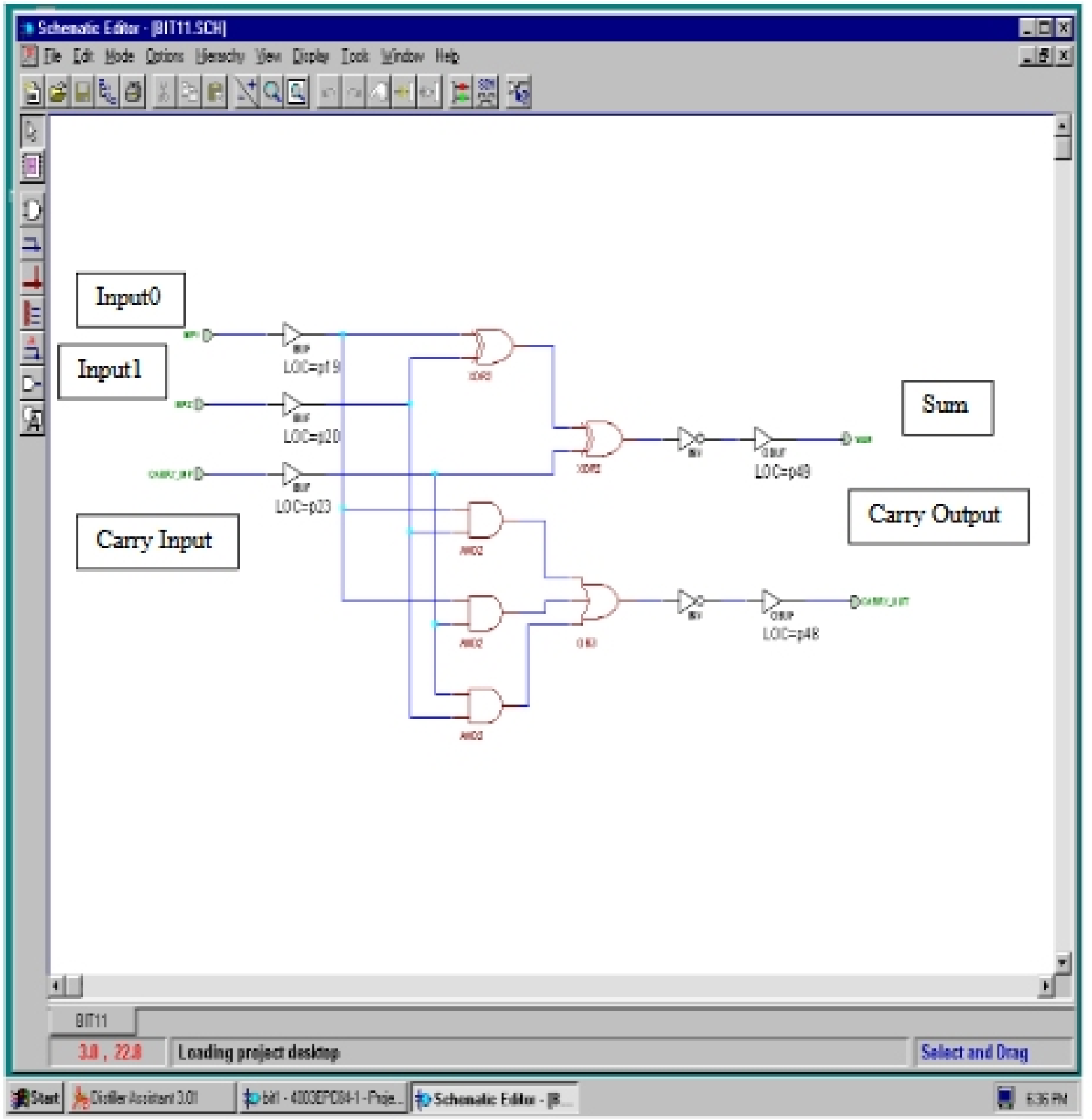
Problem 1. Create a digital circuit that accepts two "1-bit" binary numbers and outputs an "1-bit sum", i.e. an "1-bit adder". Simulate the design, download it to the FPGA demo board and test it.

As specified, the circuit has to have at least two inputs (one for each 1-bit number) and one output (for the 1-bit sum). But it is possible that the sum will not fit into1 bit. For example, adding two 1-bit numbers like 1 and 1, results in a sum 10(2), which requires 2 bits to represent it. For this reason, an extra *carry* output has to be added to indicate when this *overflow* occurs. And if the adder can have a *carry output*, then it makes sense that it should also have a *carry input*. The truth table for such a "1-bit" adder with *carry* input and output is as follows:

Input 1	Input 2	Carry Input	Sum Output	Carry Output
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Transform the truth table into a gate-level logic design using relevant gate primitives from the Schematic Library of Xilinx Foundation Series Software.

For those of you having difficulties in creating design of their own, a "hint" is given below:



Suppose you have designed and simulated the "1-bit" adder successfully. Now you have to assign the three inputs of the circuit to three pins of the XC4003 FPGA which can be driven by the SW3-1 to SW3-8 switch(es) on the board. You have pins # 19,20,23,24,25,26,27 and 28, correspondingly, at your disposal. The outputs may be assigned to any of the pins # 49,48,47,46,45,50 and 51which drive a corresponding segment of a single 7-segment LED on the board. Since each LED segment is turned "on" by driving the corresponding pin to logic "0" you may invert the output signal (as done in the circuit above). Then you will have a LED segment "on" when the output of the summer is "1" and not "0".

Now you may wish to create a module that can be saved and used in other designs.
First, get rid of the inverters, the input (IBUF) and output buffers (OBUF) and the associated pins.
Then select **Hierarchy -> Create Macro Symbol from Current Sheet** menu item.
Select a proper name for your adder and put in relevant comments. Then save it.
It will be appended to the **SC** library and so can be used in future designs.

Problem 2. Create a design that accepts two 4-bits numbers (A3..A0 and B3..B0) and outputs a 4-bit sum (Sum3..Sum0). Make use of your "1-bit" adder module. Simulate the design, download it to the Xilinx demo board and test it. An example of such a "4-bit" adder is given below. This time you may associate the outputs with pins # 61,62,65,66, 57,58,59 and 60 which drive 8 LED bars (D9-D16 row) on the Xilinx board. Please, note that the pin ordering is important. Pins # 61 and # 60 should be associated with the most and least significant bit of an 8-bit number, respectively. You may invert the output signal to have the LED segments "on" when the output of the summer is "1". Also, you may associate pins 19,20,23 and 24 with the bits A3..0, and pins 25,26,27 and 28 with the bits B3..0.