NOTE: This lab assignment does **not** contribute to the overall coursework mark for CS2. However, students are strongly advised to go through the practical as the first formal assignment (which will be handed in Week 4) relies on the use of LogicWorks.

Introduction to LogicWorks

The aim of this warm-up exercise is to introduce the LogicWorks software and use it to design a simple signed/unsigned adder-subtracter.

Part 1: LogicWorks Tutorial

If you are using LogicWorks for the first time you should work through the LogicWorks Tutorial before proceeding further, as some of the terminology and techniques covered will be assumed below.

LogicWorks has a much larger set of available devices than the DigSim software. These are contained in a number of *libraries*, which are available through the *Parts Palette* window. The default libraries (library files have the suffix .CLF) include the following.

1. ALL LIBRARIES contains all the circuits available in LogicWorks.
2. CONNECT includes power and ground connectors, and port connectors, used in designing component sub-circuits
3. 7400DEVS contains a collection of devices from Texas Instruments well-known 7400 series TTL logic. These are simulations of real chips and are therefore more complicated than the devices in, for example, PRIMLOGI.
4. DEMOLIB is used in the tutorial, as you will already have gathered
5. SPICE, MAKEPLD and DISCRETE are used for designs involving analogue and programmable devices, and should not be needed in CS2.

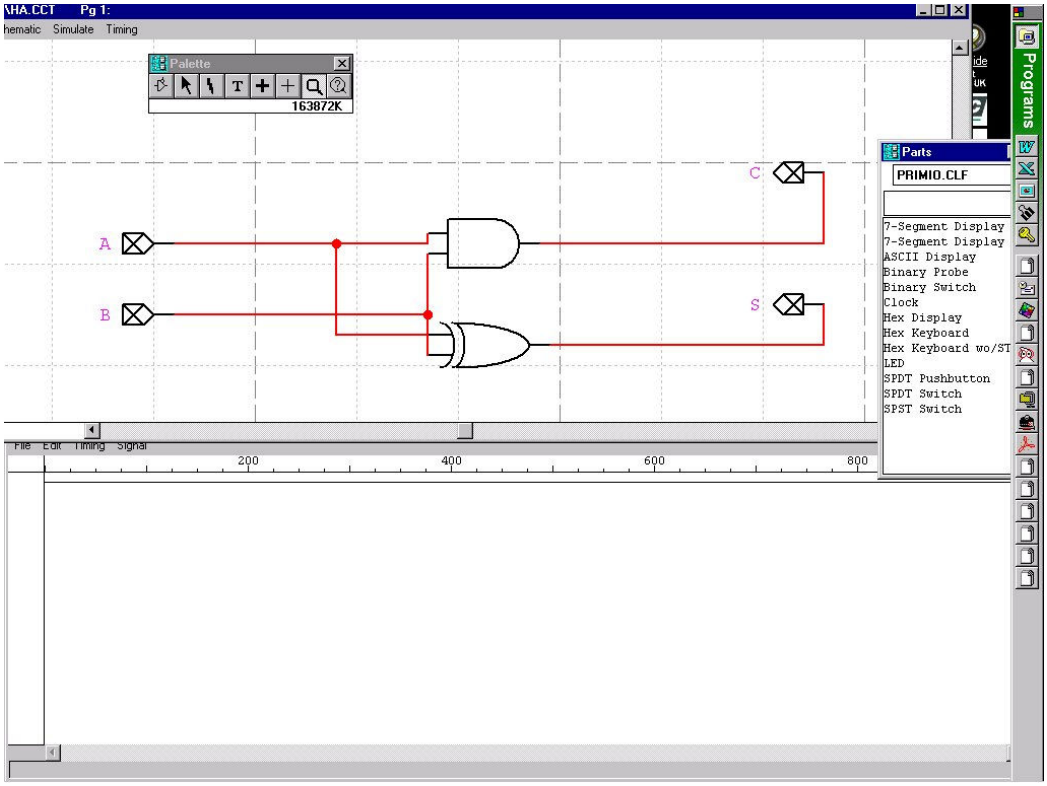
The user can manipulate libraries and create new ones. To do this, right-click on the Parts Palette, to bring up the "Library" menu. The "New Lib…" command can create a new library, while "Lib Maintenance …" allows the copying of devices between libraries, deletion of devices and so on. New libraries are not loaded automatically when LogicWorks starts, but can be accessed using the "Open Lib…" command on the library menu.

In LogicWorks a user can actually create new devices, which can then be stored in a library and thereafter are available for use in new circuit designs. The "Lib Maintenance…" command is extremely powerful, and dangerous. It can be used, for example to delete a device from a library unrecoverably. ***For this reason, you should not use the predefined libraries to store your new devices. Always create a working library to store your own devices.*** You may use "Lib Maintenance…" to copy pre-existing devices which you intend to use, to your working library, if you want the convenience of having all the devices you want to work with, in the same place.

Exercise 1: Making a new chip.

In this exercise, you will manufacture a new device for a simple *half-adder*. If you have not already done so, first create a working library for yourself, using the menu obtained when you right-click on the Parts Palette. Call it something like MYLIB.

Creating a new device is a two-stage process. The first stage involves designing the internal circuit for the device. The second stage involves using the Device Editor to make a suitable symbol, as discussed in the Tutorial.



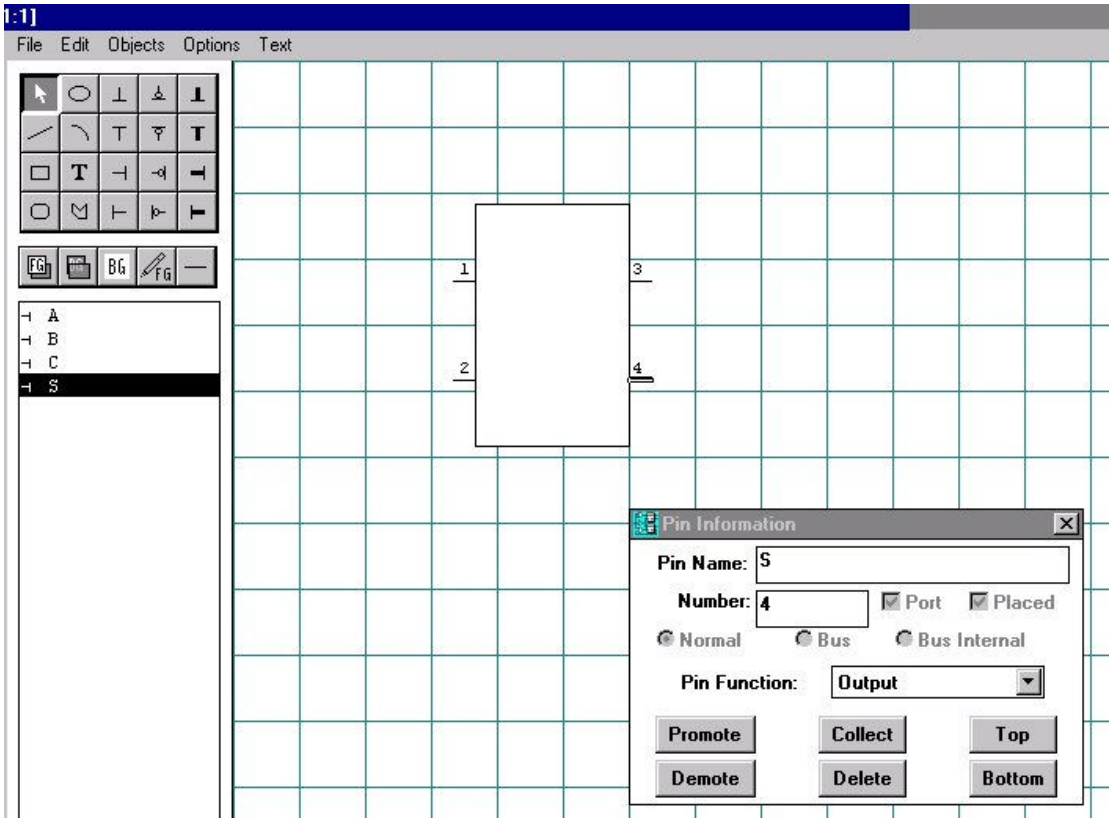
We now construct the circuit for our device (a half adder) in the Schematic window. This requires only the use of a XOR and an AND gate as shown in the screen shot, and will be the *internal circuit* for the new device. Each input to the circuit which will correspond to an input pin on the final device must be attached to an *input port connector*, which can be found in the CONNECT library. Likewise, each output must be connected to an output port connector, as shown. All ports should be labelled.

Once the circuit is complete, save it, but do not close the window. Instead launch the Device Editor by selecting the menu sequence “File: New: Device Symbol”. You can now design a symbol for the part in much the same way as in the LogicWorks Tutorial, but with one important difference—this symbol will have the half-adder circuit attached to it, and will therefore behave just like the half-adder in simulations.

Once the Device Editor is open, select the “Options” menu, and choose the entry "Subcircuit / Part Type". This will invoke a dialogue box with several alternatives. Click on "Create a

subcircuit symbol and select an open circuit to attach to it". A further dialogue box will now invite you to select the circuit in your still-open schematic window. Accept this, and a port list will appear on the left.

Now draw a part that will represent your half-adder in the future. A box with four pins would serve adequately (see Figure below). As you draw pins these are assigned in order to the list of port names on the left. (You may add pin-numbers by double clicking on each port name.) Text can be added as desired with the text tool.



You may now save this part in your working library and close the Device Editor and Schematic Design windows. The new half-adder device is now available for use in new circuits. Test it out using binary switches and probes! It even has the propagation delays inherited from the gate primitives used, as you can verify using the simulator.

Part 2: Design and Build an 8-bit Adder-Subtracter

Now that you have a half-adder chip, you are ready to begin the creative part of this warm-up exercise. The idea is to build a chip that could be used to add or subtract unsigned or signed numbers. To do this, you may need to consult your notes from the Computing Fundamentals 1 (CF1) course. The following steps are intended to guide you through the process.

a) Construct a full-adder, using your half-adder device as a component. Save your full-adder in your personal library.

b) Modify your full-adder to make a full-subtracter. This requires the use of three additional inverters.

c) Modify your full adder so that it has a *function select* line, which allows one to choose whether it should act as an adder or a subtracter. *Hint* this can be done with three additional exclusive-OR gates.

d) Use your 1-bit adder-subtracter to build a 4-bit version, with a function select.

e) Can your 4-bit adder-subtracter be used with two's complement numbers? If not, modify it so that it can be. (Recall that two's complement addition and subtraction is the same as unsigned, except that the overflow test is different. The key is to implement the test for two's complement overflow.

f) Test your adder-subtracter device out. Use a binary switch to select the function, and hex keyboards and displays to provide inputs and outputs

Acceptance Test:

- Show your Tutor the ripple adder-subtracter functioning. He/she will perform an acceptance test using various inputs

- Show a paper copy of your circuit at each stage (a-f). You may use a print out from LogicWorks, or you may draw the designs by hand, but you must make sure all components, lines, chip inputs etc are properly labelled. With each circuit, include a precise description of what it does and a brief indication of the reasoning behind your design.