

## Programmable Logic Devices: Stage A

**Acknowledgements:** Developed by Bassam Matar, Faculty at Chandler-Gilbert Community College, Chandler, Arizona.

**Lab Summary:** This lab will present design entry, simulation, and prototyping with tools that are provided by Altera for this purpose. We will show how a complete simple design circuit of the half adder can be directly entered into Quartus® II for synthesis, post synthesis simulation, timing analysis, and device programming. We will show the implementation of more complex designs in future labs by running them through the design flow illustrated in this lab.

**Lab Goal:** The goal of this lab is to determine a truth table of a half adder using Altera software.

### **Learning Objectives**

1. Create a half adder project in Quartus® II (web edition).
2. Use the Quartus® Block Editor to enter a graphical design in Quartus® II.
3. Compile and simulate the half adder design.
4. Program an Altera CPLD with the half adder design.
5. Test the design of half adder on a CPLD test board to determine its truth table.
6. Create, compile, and simulate a half adder symbol in Quartus® II (web edition).
7. Create, compile, and simulate an incrementer-4 symbol in Quartus® II.
8. Use the Incrementer-4 to find Two's complement representations.

**Grading Criteria:** Your grade will be determined by your instructor.

**Time Required:** 6 - 7 hours

### **Special Safety Requirements**

Static electricity can damage the CPLD device used in this lab. Use appropriate ESD methods to protect the devices. Be sure to wear a grounded wrist-strap at all times while handling the electronic components in this circuit. The wrist strap need not be worn after the circuit construction is complete.

No serious hazards are involved in this laboratory experiment, but be careful to connect the components with the proper polarity to avoid damage.



### **Lab Preparation**

- Read the WRE PLD Narrative Module.
- Read this document completely before you start on this experiment.
- Acquire required test equipment.
- Print out the laboratory experiment procedure that follows.

### **Equipment and Materials**

Each team of students will need the test equipment, tools, and parts specified below. Students should work in teams of two.

Test Equipment and Power Supplies	Quantity
The following items from the UP2 Educational Kit: <ul style="list-style-type: none"> <li>• Altera UP-2 circuit board with ByteBlaster Download Cable</li> <li>• Quartus<sup>®</sup> II Web Edition software</li> <li>• AC adapter, minimum output: 7 VDC, 250 mA DC</li> </ul>	1
ESD Anti-static Wrist Strap	1
#22 Solid-core wire	As needed
Wire Strippers	1

### **Additional References:**

1. *Digital Design and Implementation with Field Programmable Devices* textbook found with UP2 educational kit.
2. UP2 educational kit data sheet found on Altera web site: [www.altera.com](http://www.altera.com)

### **Introduction**

This is the first of three stages of software/hardware work required for creating a microprocessor project. Stages A and B are completed in this WRE module. Stage C will be completed in the WRE Micro and Embedded Controllers Part 2. You will advance from knowing very little about digital design and the use of Altera software to the point where you can build a microprocessor and understand some of its fundamental principles.

In Stage A, you will combine simple Boolean gates to form larger, more useful circuits. Also, you will systematically design some other elementary functions into circuits built from basic gates and devices. In Stage B, you will combine the circuits built in stages A and the current stage to form an ALU that will handle all the data manipulation operations in a microprocessor. In Stage C, you will add in memory and show how the memory can be linked to an ALU to perform actions on stored data such as addition, subtraction, and some logic operation. The data manipulations are controlled by the user (i.e. Brainless Microprocessor).



## Stage A

In this stage, you will first load and license the Altera Quartus® II software. Then you will be given some small circuits of limited functionality and shown how they can be used to construct bigger circuits of greater functionality. This is one of the most important ideas of digital logic design.

### Basic Arithmetic Functions

The three basic logic gates (AND, OR, and NOT) that you studied in your class perform the basic functions of Boolean logic. Computers need to do other operations as well. Now, we will provide you with circuits, built using the basic Boolean gates that perform some commonly encountered operations. These circuits even have names because they are used so frequently.

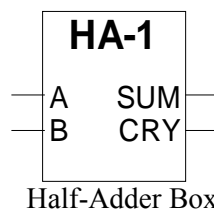
In this module, we will use the full adder, half adder, and increment circuits as the basis for arithmetic.

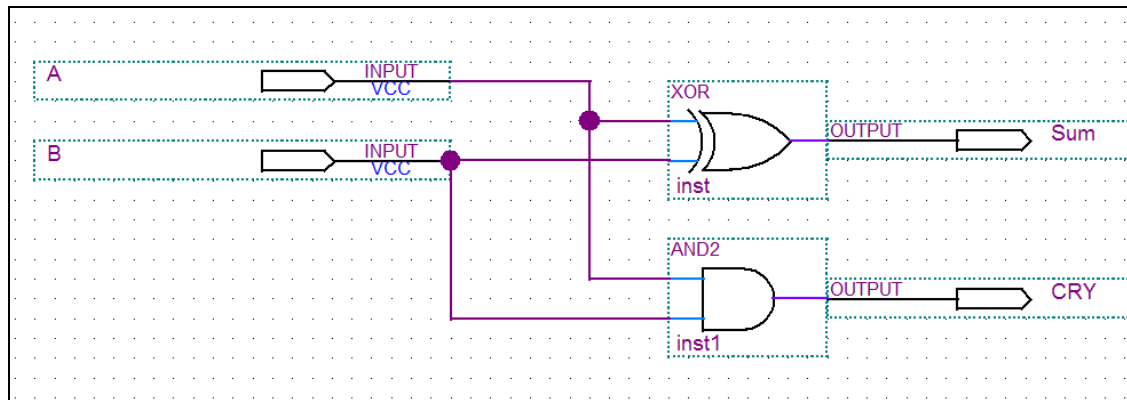
- The **full adder** has two numbers and a 1-bit carry as inputs, and a single number and 1-bit carry as outputs. It adds the two input numbers and the carry to get the single sum result and carry out. Two simpler circuits can be built that resemble a full adder with some missing input or output.
- The **half adder** adds two binary inputs but does not have a carry input.
- The **increment** is also similar to the full adder but is missing one of the two number inputs. As its name implies, the increment is useful when you may want to add 1 to a number. To do this, you just set the carry into a 1.

In this stage, we will build, test, and store some basic circuits for performing arithmetic. These circuits are the smallest half adder and full adder. Circuits that are more complicated can be built using these two circuits as building blocks. Therefore, we first need working copies of the simple circuits. Below are the circuits you should enter into Altera Quartus® II and the truth table of the circuits. The circuit is shown on the next page. We will write a (-1) after the names of each of these devices to indicate that these are the 1 bit (hence -1) version of them. We will develop circuits in Stage B in order to control the flow of data throughout the system.

#### HALF ADDER-1:

INPUTS		OUTPUTS	
A	B	SUM	CRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1





Half-Adder Circuit

In the above circuit, we show a box with input and output pins. These boxes are the basis for our hierarchical design approach. They allow us to suppress the detail of how the circuits work and just describe the function. The box alone is a much simpler appearing object than the associated circuits of Boolean gates. Therefore, the first thing that we need to do in this lab is to use Altera Quartus® II to build the Boolean gate circuits. The second thing we will do is put these circuits into the appropriately labeled box. In other words, we want to build each of the circuits above, test them according to their truth tables to see if they work, and then put the circuit inside their box (picture). This becomes absolutely necessary if we want to keep track of what we are doing in the larger circuit.

## UP2 Development Board

A simulation model is a soft model of hardware that is being designed while a prototype is a hard model of the hardware under design. A development board includes one or more devices used as prototypes and provides easy programming and interfaces for utilization of such devices. Such a board provides basic I/O devices and interfaces for the peripherals that may be used with the prototype devices.

A digital designer develops the design using simulators at various levels of abstraction, synthesis tools, placement programs, and other related tools with a development board. When the design is complete, devices on a development board are programmed and the actual peripherals are connected to the prototype device using the interfaces provided on the board. This provides an exact hardware model of the design that can be tested for actual physical conditions.

Because of the growth of programmable devices, development boards have become very popular, and are available for various devices, various degrees of complexity, and many common interfaces. It is not unusual for a company with a product that is being distributed to a limited number of customers to use a development board in the product that is being shipped. This way, the system will be tested and examined on-site and the final board will be generated after all design and board layout bugs are fixed.

Altera's UP2 development board is mainly designed for educational purposes. In a college setting, this board is used for laboratories related to courses on digital system design, computer architecture, and peripheral. In spite of its main educational target, UP2, with its popular MAX 7000S CPLD and FLEX 10K FPGA is a very useful development board for industrial settings.

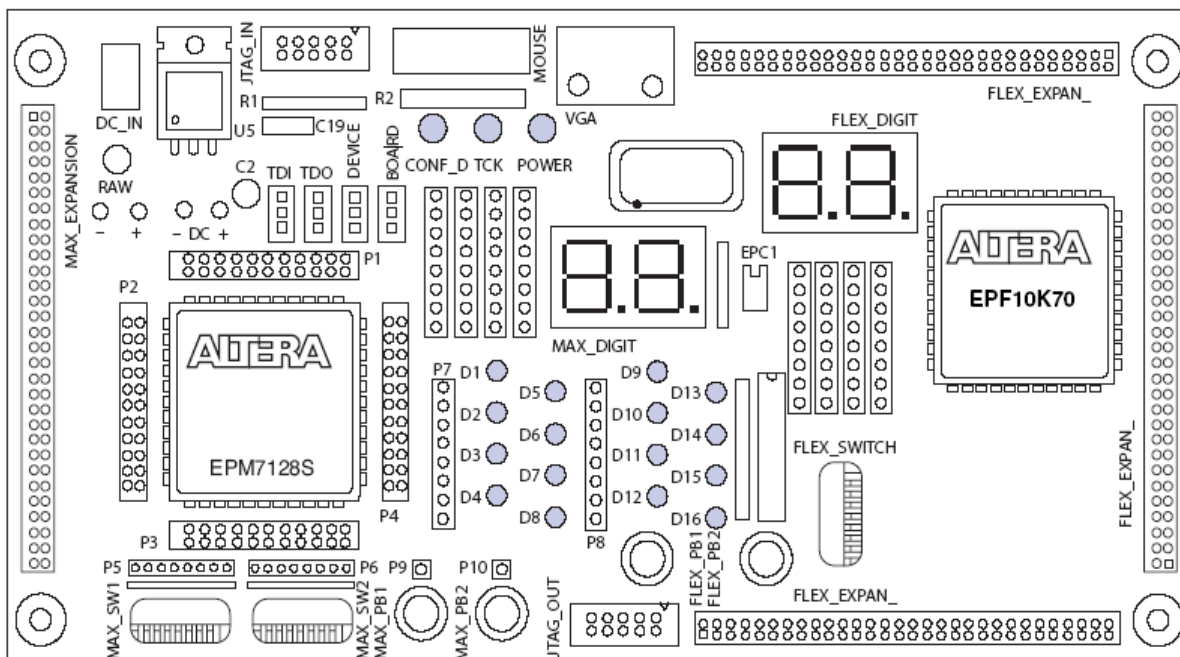


Altera's "University Program UP2 Development Kit's datasheet is a document that describes details of this board.

## UP2 General Features

The features that we use in the development of our brainless microprocessor projects are described in this section.

The UP2 development board, shown below, has an EPM7128SLC84-7 CPLD and an EPF10K70RC240-4 FPGA. The board has an interface for ByteBlaster II device programming hardware that can be used to program the on-board devices. The board provides power and clock for its CPLD and FPGA.



**CPLD:** The EPM7128S device, a member of the MAX 7000S family, is based on erasable programmable read-only memory (EEPROM) elements. The EPM7128S device features a socket-mounted 84-pin plastic j-lead chip carrier (PLCC) package and has 128 macrocells. With a capacity of 2,500 gates and a simple architecture, the EPM7128S device is ideal for introductory circuit designs as well as larger combinational and sequential logic functions.

**FPGA:** The EPF10K70 device is based on SRAM technology. It is available in a 240-pin RQFP package and has 3,744 logic elements (LEs) and nine embedded array blocks (EABs). With 70,000 typical gates, the EPF10K70 device is ideal for intermediate to advanced digital design courses, including computer architecture, communications, and DSP applications.



**ByteBlaster II:** Designs can be easily and quickly downloaded into the UP2 board using the ByteBlaster II download cable, which is a hardware interface to a standard parallel port. This cable sends programming or configuration data between the device programming software (programmer part of Quartus<sup>®</sup> II) and the UP2 Education Boards. Because design changes are downloaded directly to the devices on the board, prototyping is easy and multiple design iterations can be accomplished in quick succession.

**JTAG Input Header:** The 10-pin female plug on the ByteBlaster II download cable connects with the JTAG\_IN 10-pin male header on the UP Education Board. The board provides power and ground to the ByteBlaster II download cable. Data is shifted into the devices via the TDI pin and shifted out of the devices via the TDO pin. In all of our configurations, we use the ByteBlaster II in the Joint Test Action Group (JTAG) operating mode.

**Jumpers:** The UP Education Board has four three-pin jumpers (TDI, TDO, DEVICE, and BOARD) that set the JTAG configuration. The JTAG chain can be set for a variety of configurations (i.e., to program only the EPM7128S device, to configure only the FLEX 10K device, to configure and program both devices, or to connect multiple UP Education Boards together). In all of our work with UP2, we only use one board and only one of the programming devices.

**Supply Power:** The DC\_IN power input accepts a 2.5-mm x 5.55-mm female connector. The acceptable DC input is 7 to 9 V at a minimum of 350 mA. The RAW power input consists of two holes for connecting an unregulated power source. After being regulated by the board hardware, proper DC voltage is applied to both devices on the board.

**Oscillator:** The UP Education Board contains a 25.175-MHz crystal oscillator. The output of the oscillator drives a global clock Input on the EPM7128S device (pin 83) and a global clock input on the FLEX 10K device (pin 91).

**EPM7128S CPLD Device:** Resources for the UP2 CPLD device include prototyping headers, switches, push-buttons, LEDs, and seven-segment displays. Female connectors surrounding this device give access to its pins for connecting to on-board or external resources.

**Prototyping Headers:** The EPM7128S prototyping headers are female headers that surround the device and provide access to the device's signal pins. The 21 pins on each side of the 84-pin PLCC package connect to one of the 22-pin, dual-row 0.1-inch female headers. The pin numbers for the EPM7128S device are printed on the UP2 Education Board (an "X" Indicates an unassigned pin).



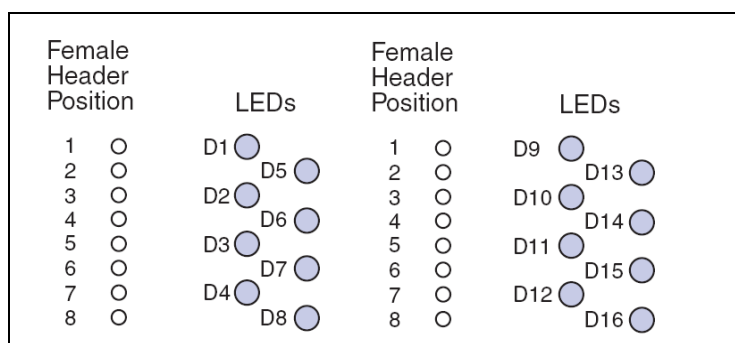
The table lists the pin numbers for the four female headers: P1, P2, P3, and P4. The power, ground, and JTAG signal pins are not accessible through these headers.

P1		P2		P3		P4	
Outside	Inside	Outside	Inside	Outside	Inside	Outside	Inside
75	76	12	13	33	34	54	55
77	78	14	15	35	36	56	57
79	80	16	17	37	38	58	59
81	82	18	19	39	40	60	61
83	84	20	21	41	42	62	63
1	2	22	23	43	44	64	65
3	4	24	25	45	46	66	67
5	6	26	27	47	48	68	69
7	8	28	29	49	50	70	71
9	10	30	31	51	52	72	73
11	X	32	X	53	X	74	X

**Push-buttons and Switches:** Max\_PB1 and MAX\_PB2 are two push-buttons that provide active-low signals and are pulled-up through 10k ohms resistors. Pins from the EPM7128S device are not pre-assigned to these push-buttons. Connections to these signals are made by inserting one end of the hook-up wire into the push-button female header. The other end of the hook-up wire should be inserted into the appropriate female header assigned to the I/O pin of the EPM7128S device.

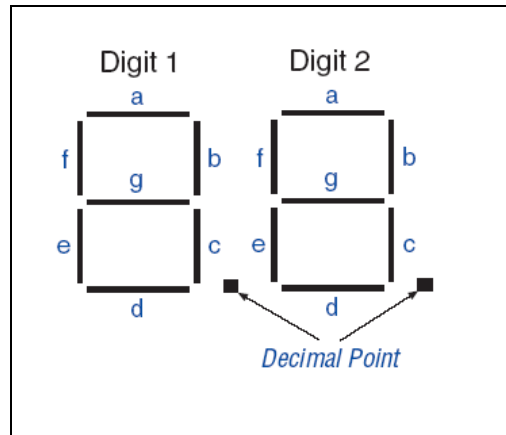
MAX\_SW1 and MAX\_SW2 each contain eight switches that provide logic-level signals. These switches are pulled-up through 10k ohms resistors. Pins from the EPM7128S device are not pre-assigned to these switches. Connections to these signals are made by inserting one end of the hook-up wire into the female header aligned with the appropriate switch. The other end of the hook-up wire is inserted into the appropriate female header assigned to the I/O pin of the EPM7128S device. The switch output is set to logic 1 when the switch is open and set to logic 0 when the switch is closed.

**LEDs and Displays:** The UP2 Education Board contains 16 LEDs (shown below) that are pulled-up with a 330-Ω resistor. An LED is illuminated when a logic 0 is applied to the female header associated with the LED. LEDs D1 through D8 are connected in the same sequence to the female headers (i.e., D1 is connected to position 1, and D2 is connected to position 2, etc.). LEDs D9 through D16 are connected in the same sequence to the female headers (i.e., D9 is connected to position 1, and D10 is connected to position 2, etc.).





MAX\_DIGIT is a dual-digit, seven-segment display connected directly to the EPM7128S device. Each LED segment of the display can be illuminated by driving the connected EPM7128S device I/O pin with logic 0. The figure below shows the display segments and their connections to EPM7128S pins.

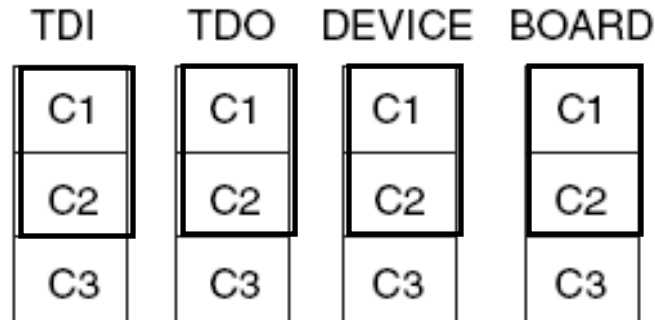


Display Segment	Pin for Digit 1	Pin for Digit 2
a	58	69
b	60	70
c	61	73
d	63	74
e	64	76
f	65	75
g	67	77
Decimal point	68	79





**Programming:** EPM7128S of the UP2 is programmed by the ByteBlaster II hardware connected to the JTAG terminal of the board. Board jumpers must be set as shown below figure in order to program this device.



Desired Action	TDI	TDO	DEVICE	BOARD
Program EPM7128S device only	C1 & C2	C1 & C2	C1 & C2	C1 & C2

**EPF10K70 FPGA Device:** Resources for the UP2 FPGA device include switches, push-buttons, seven-segment displays, a VGA connector, and a keyboard/mouse interface connector. Pins from the EPF10K70 device are pre-assigned to these resources. For connection to other peripherals, expansion pins on the sides of the UP2 board should be used.

**Push Buttons and Switches:** FLEX\_PB 1 and FLEX\_PB2 are two push buttons that provide active-low signals to two general-purpose I/O pins on the FLEX 10K device. FLEX\_PB1 connects to pin 28, and FLEX\_PB2 connects to pin 29. Each push button is pulled-up through a 10k ohms resistor. FLEX\_SW1 contains eight switches that provide logic-level signals to eight general-purpose I/O pins on the FLEX 10K device. An input pin is set to logic 1 when the switch is open and set to logic 0 when the switch is closed. The figure below shows FLEX pin connections to the FLEX switch set.

Switch	FLEX 10K Pin
FLEX_SWITCH-1	41
FLEX_SWITCH-2	40
FLEX_SWITCH-3	39
FLEX_SWITCH-4	38
FLEX_SWITCH-5	36
FLEX_SWITCH-6	35
FLEX_SWITCH-7	34
FLEX_SWITCH-8	33



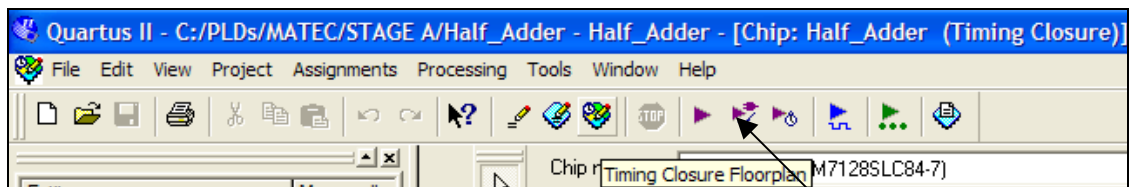
**FLEX\_DIGIT Display:** FLEX\_DIGIT is a dual-digit, seven-segment display connected directly to the FLEX 10K device. Each LED segment on the display can be illuminated by driving the connected FLEX 10K device I/O pin with logic 0.

Display Segment	Pin for Digit 1	Pin for Digit 2
a	6	17
b	7	18
c	8	19
d	9	20
e	11	21
f	12	23
g	13	24
Decimal point	14	25

Additional information on the configuration is available at the Altera web site at <http://www.altera.com/literature/univ/upds.pdf>

## Configure Devices

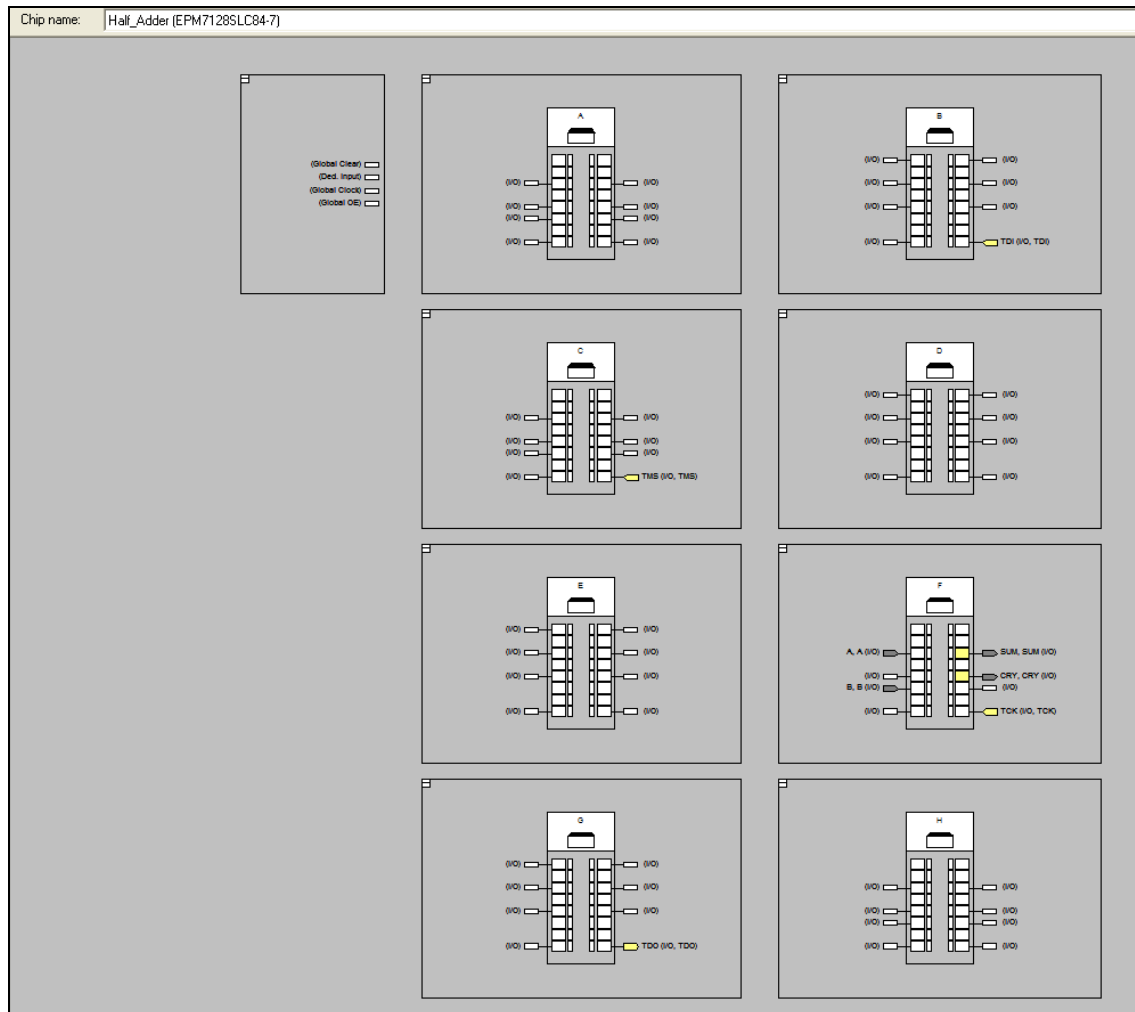
Quartus® II enables you to inspect the timing of your configured devices, see their floor plan, and in some cases, change the way device cells are used. Three icons in the *Standard tool* bar are used to activate *Timing Closure Floor*, *Last Compilation Floor plan* and *Chip Editor Applications*.



*Timing Closure Window*



The figure below shows the Timing Closure window that resulted from compilation of the half adder circuit targeting the EPM7128SLC84-7 CPLD used in this lab.



As shown, there are eight PLDs in this CPLD (Refer to your textbook for more information). These PLDs are numbered A to H from the upper left to lower right CPLD.

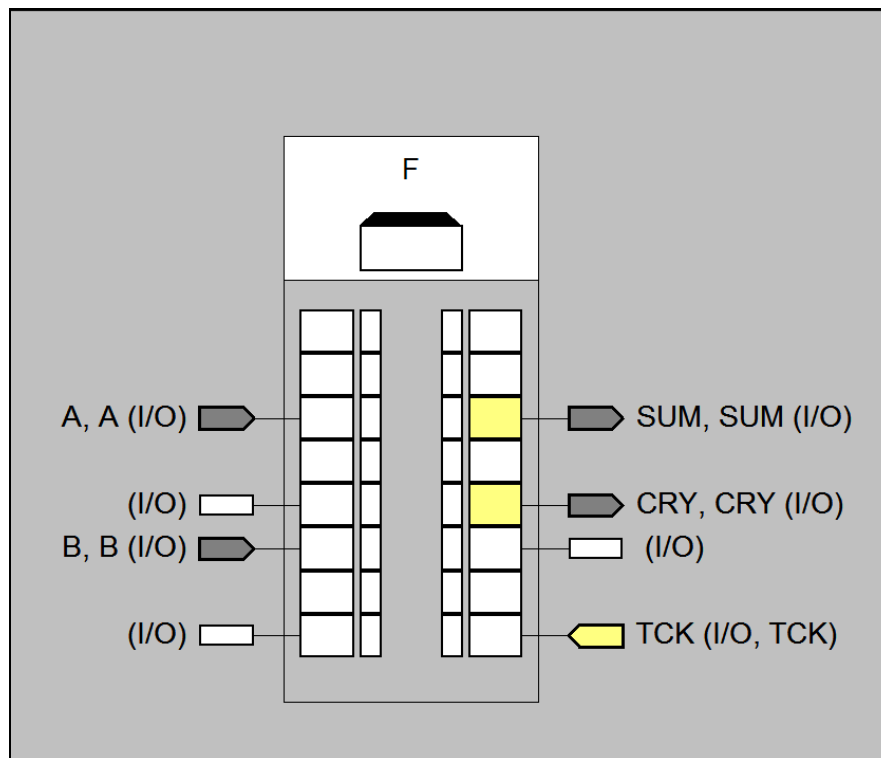
For navigating in this floorplan, there is an associated tool bar that is shown below.



This tool bar becomes active when a floorplan of the device is displayed. Zooming on the floorplan of the above figure enables us to see the actual IO pins used for our design, macrocells that have been utilized, and their timings.



The figure below shows PLD "F" that is used for implementing a half adder circuit.



Other features on the Timing Closure window include display of critical times, connection counts, delays in and out of cells, node equations, user specified pin assignments, fitter pin assignments, and other parameters that show how the compiler and the device programmer have implemented our design in our target device. This information is displayed by selecting appropriate icons from the **Floorplan Editor** tool bar.



## **Lab Procedure: Installing and Licensing Altera Quartus® II**

Before we can build the circuit, we have to load and license the Altera Quartus® II software. The software will not run without the license from Altera. NOTE: The Altera web site changes frequently. These instructions are accurate as of May 2006. The section labels may change but you should be able to access the software needed.

1. Start your internet browser and access Altera web site at [www.Altera.com](http://www.Altera.com).
2. Select *Download the Free Quartus® II v6.0 Web Edition*.
3. If there is no place to select the Download from the first page of the web site:
  - a. Select *University Program* from the **Education and Events** section.
  - b. Select *Design Software* from the **Education Materials** section.
  - c. Select *Quartus® II Software for Education* from the paragraph titled **Altera University Program Design Software**.
  - d. Select *Quartus® II Web Edition Software* from **Table 1**.
  - e. Select *Download the Free Quartus® II Software Web Edition Software*.
4. Click on **Licensing** on the left hand side of below figure under **Download & Licensing**. When you request a license for an Altera software product from the Altera web site, the license file is sent via e-mail and includes installation instructions.
5. Click on Quartus® II Web Edition Software Free
6. Enter the appropriate information as shown below
7. Once you hit Submit Request, you will be asked to enter your network interface card (NIC) number as shown below.

MAX+PLUS II EDA Support

- View by Vendor
- View by Tool
- View by Function

Download & Licensing

- Download
- Licensing

### Registration Information

Enter your network interface card (NIC) number:

Please indicate how this software will be used. Note this information is for statistical purposes only.\*

☐ Standard: usage will be partially or entirely for commercial/industrial purposes

☐ Academic: will be used only for academic/education/hobby purposes

If you have a preferred distributor, enter the name here



8. You can obtain your NIC number by executing the Command prompt in your window.
  - a. Click on **Start** and select **Run**.
  - b. Type **command** in the box and click **OK**.
  - c. At the command prompt, type `ipconfig/all`. It will show you a list of the network settings for your computer as shown below for my computer.

```

C:\WINDOWS\system32\command.com
Microsoft(R) Windows DOS
(C)Copyright Microsoft Corp 1990-2001.

C:\DOCUMENT1\BMAIAR>ipconfig/all

Windows IP Configuration

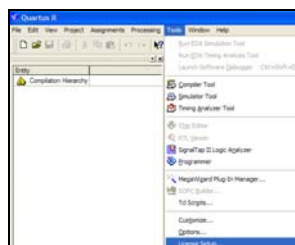
Host Name . . . . . : en4025047
Primary Dns Suffix . . . . . :
Node Type . . . . . : Hybrid
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No
DNS Suffix Search List. . . . . : eas.asu.edu
asu.edu

Ethernet adapter Local Area Connection:

Connection-specific DNS Suffix . : ducy.asu.edu
Description . . . . . : Broadcom NetXtreme 57xx Gigabit Controller
Physical Address. . . . . : 00-0F-1F-DC-D1-ED
Dhcp Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes
IP Address. . . . . : 149.169.40.254
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 149.169.40.1
DHCP Server . . . . . : 129.219.110.130
DNS Servers . . . . . : 129.219.17.5
                        129.219.13.81
                        129.219.17.200
Primary WINS Server . . . . . : 149.169.31.3
Secondary WINS Server . . . . . : 149.169.31.9
Lease Obtained. . . . . : Tuesday, June 14, 2005 7:17:30 AM
Lease Expires . . . . . : Thursday, June 16, 2005 7:17:30 AM

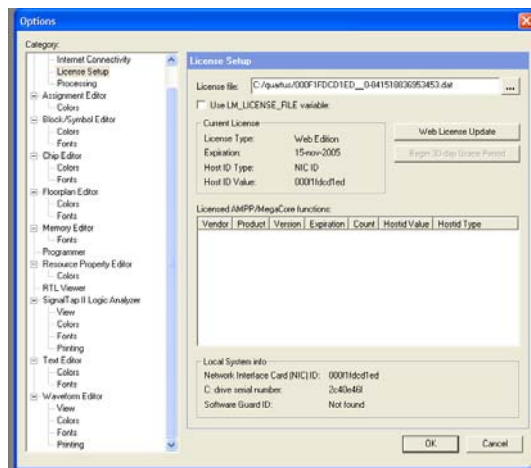
C:\DOCUMENT1\BMAIAR>
  
```

- d. Write down the Physical Address of your computer. It is 16 digits number and it is unique to your computer.
  - e. Type **Exit** at the command prompt to close the window.
9. Type your NIC number and select the option **Academic**, *it will be used only for academic/education/hobby purposes* as shown below and click continue.
10. You will be prompt to take survey.
11. A license file will be mailed to you immediately to the email address you specify in the form of step 3. The tile of the email should be ***“Your Altera Development Tools License”***. The email includes the following sections:
  - a. Extensive Learning Resources
  - b. License File Installation Instructions
  - c. Support Information
12. The following instruction should be included in your email:
  - a. PC Instructions:
    - i) Save your license file text to your computer's hard drive. Altera recommends saving the file into your `c:/quartus` directory with a `".dat"` file name extension.
  - b. In the Quartus® II software, choose License Setup (Tools Menu).





- c. In the License File box, type or browse to the full path and file name of the file you saved in step 1



- d. Once you click OK, the Quartus® II Web Edition software is now licensed.  
 e. If you have problems, use AN 340 to troubleshoot your licensing setup.
13. Install the driver for Altera ByteBlaster by typing the following address in your browser:  
<http://www.altera.com/support/software/drivers/dri-index.html>
14. Follow the instruction to install the driver for your appropriate window operating system.

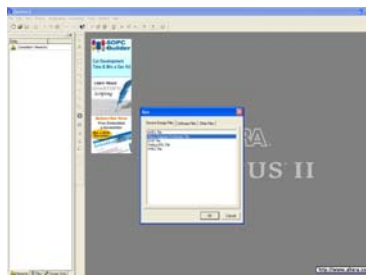
### **Lab Procedure: Building the HALF ADDER-1**

In this portion of the lab, you will use Altera Quartus® II to build the HALF ADDER-1 circuit. The steps involved are design entry, compilation, simulation, and downloading.

NOTE: At the conclusion of the lab, you will be asked to comment on why certain steps are required. Be sure to take notes on these questions as they appear in the lab procedure.

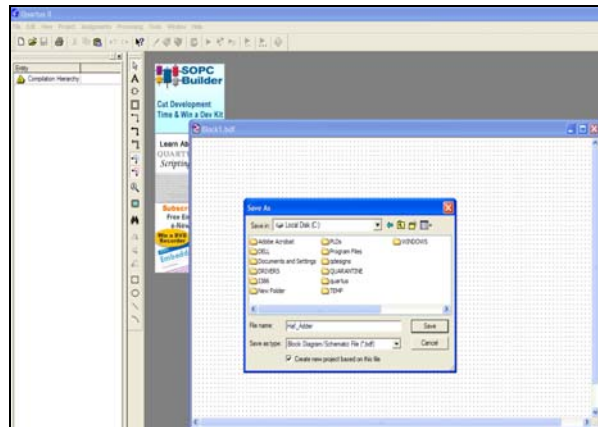
### **Design Entry Instructions**

1. Open Quartus® II software
  - a. Select Start →
  - b. All Programs →
  - c. Altera →
  - d. Quartus® II Web Edition
2. From Quartus® Main Menu Select **block diagram/schematic file** from **New** option under **File** menu and select OK.

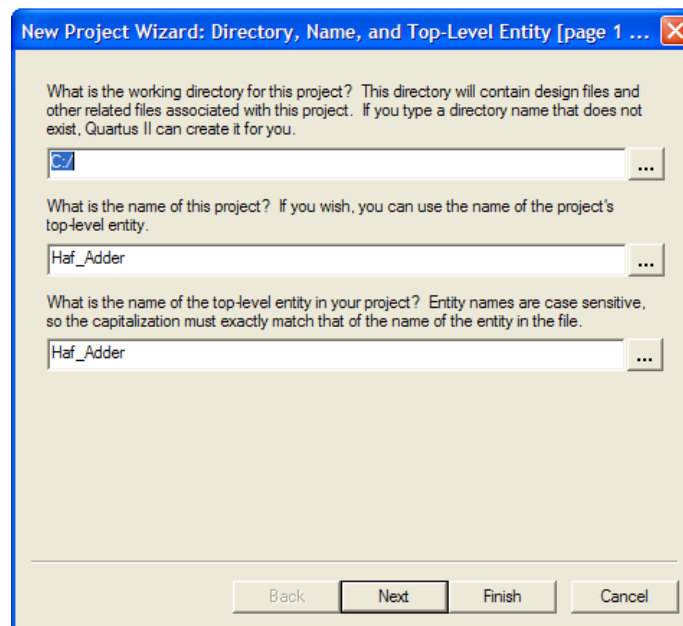




3. Save the file (Half\_Adder.bdf) and create a new project:
  - a. Select **File**
  - b. Select **Save As** from the *main menu*.



4. Select **Next** and verify that the New Project Wizard Directory menu reflects the correct directory and project name as shown below and then select **Finish**.



5. Enter the required logic symbols for the half adder circuit as shown in figure 1 from the primitives' subdirectory.
6. Click on the symbol tool as shown below:

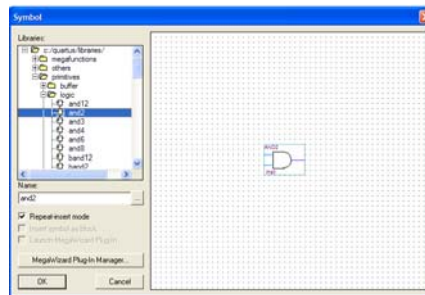


**Symbol Tool**

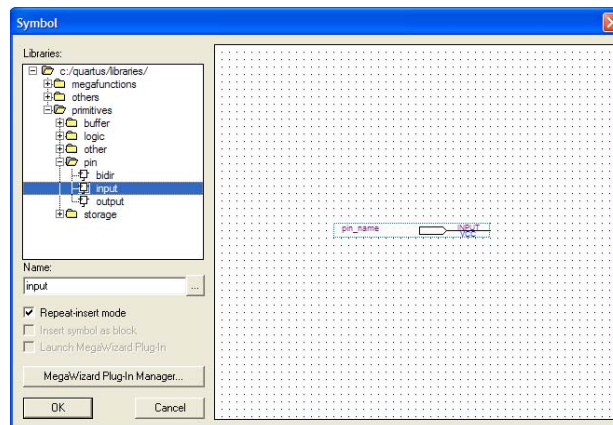




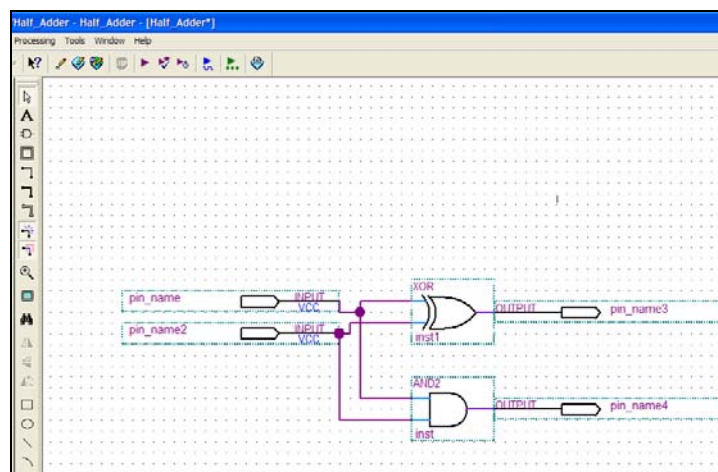
7. Next Expand the *C:/Quartus/libraries/*, then expand the *primitives subdirectory* and select *and2* and *xor* from the *logic sub-directory* as shown below.



8. Enter the input (A,B) and output (Sum, CRY) pins from the primitives' subdirectory as shown below:



9. Connect the XOR and AND gates to their inputs and outputs pins. To connect components, click over one end of the one component and drag a line to one end of a second component. When you drag the line, use the horizontal and vertical grids to help you align connections properly. When this is completed, you should have the circuit below:

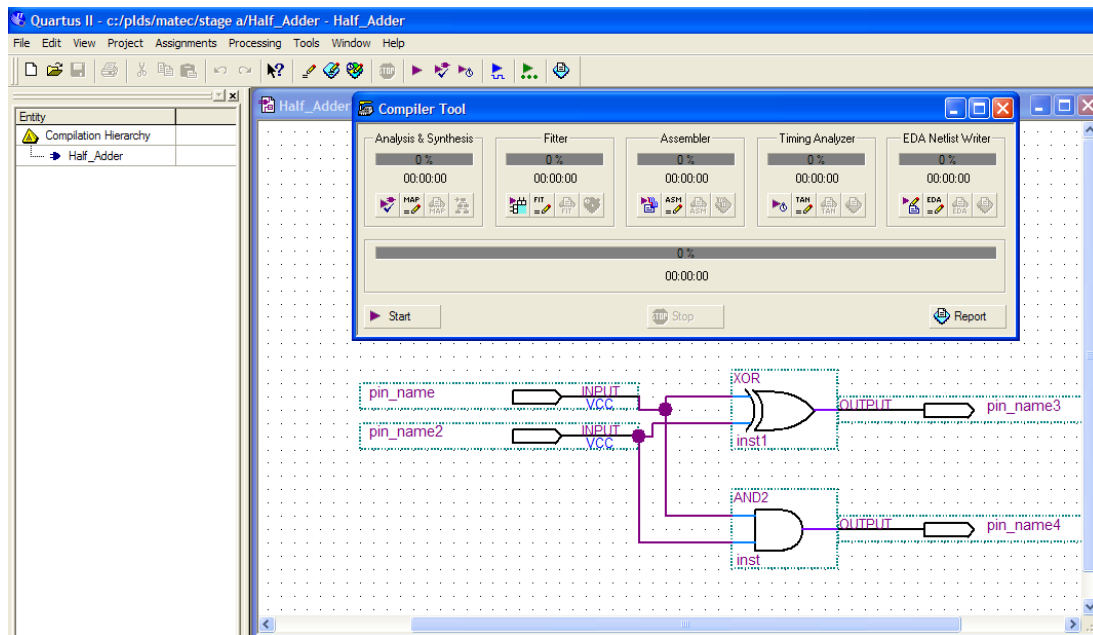
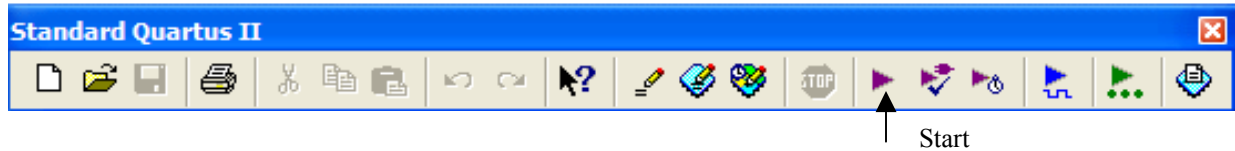




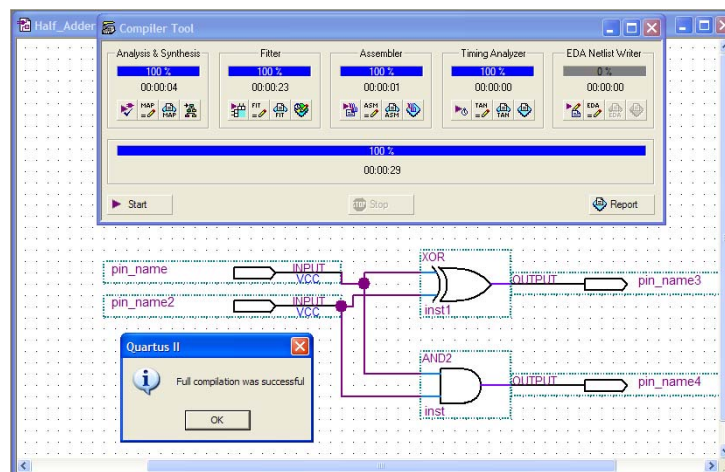
### Compilation Instructions:

10. Compile the half adder circuit.

- From the **tools menu**, select the **compiler** tool and hit **Start**.
- Or you click on the **start compilation** button on the Quartus® II toolbar as shown below:



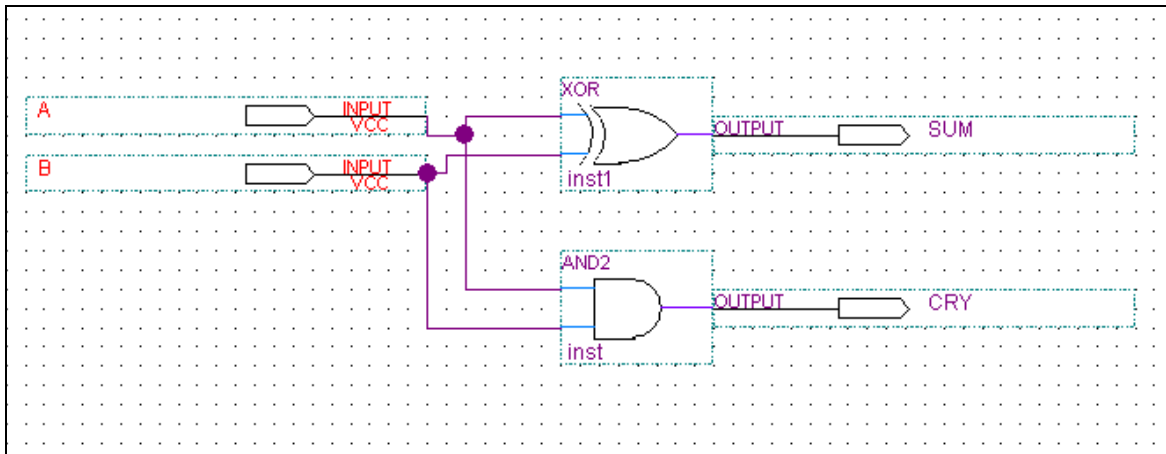
- The compiler will take place and either signal the successful compilation or indicate errors. If there are any errors, make the appropriate corrections to the circuit and recompile it again.





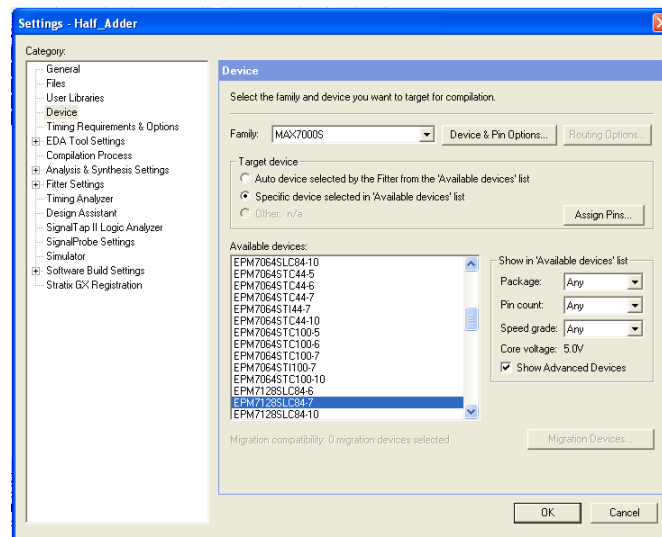
# 11. Test the circuit.

- Rename input and output ports to names as listed in our truth table. Use A and B for inputs, Sum and Cry for outputs.
- To name a pin, double click it to open its **Pin Properties** window or right-click it and select properties from the pull-down menu that shows up.
- When this is completed, save your circuit and make sure it is named the same as your project, i.e. **Half\_Adder**. Also, make sure you run the compiler one more time from **tools** menu.



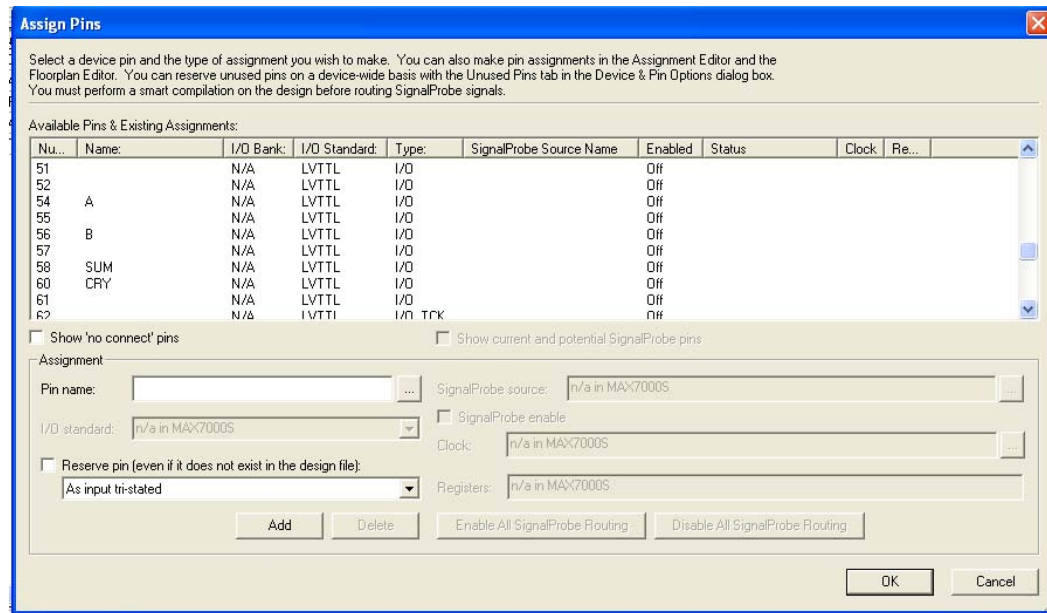
12. Device Configuration: For our half-adder, use the MAX7000s device family, and in this family of devices we use the EPM7128SLC84-7 CPLD. We have selected this device because it is one of the two programmable devices on Altera's UP2 development board. The figure below shows the project definition page in which the project is defined.

13. From Assignments menu, select Assign pins and then select the appropriate options as shown below:

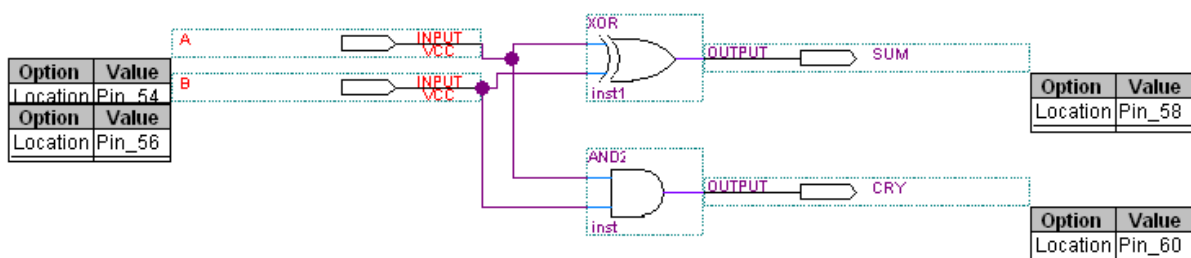




14. Assign Pins to the inputs and outputs ports of our half adder circuit to the pins of the CPLD. To do this, select **Assign Pins** from **Assignments** menu. The corresponding window is shown below:



- Scroll down the device's pin list and click on the pin that is being assigned to a circuit node.
- Then in the assignment area, next to the Pin Name type the name of the node of your circuit.
  - Type the name of the node in association of the highlighted pin number.
  - Assign A, B, Sum and CRY to device pins 54, 56, 58, and 60 respectively. The reason for this selection becomes clear after we discuss the UP2 board in the next section.
- The circuit now shows the appropriate pins for inputs and outputs as shown below:



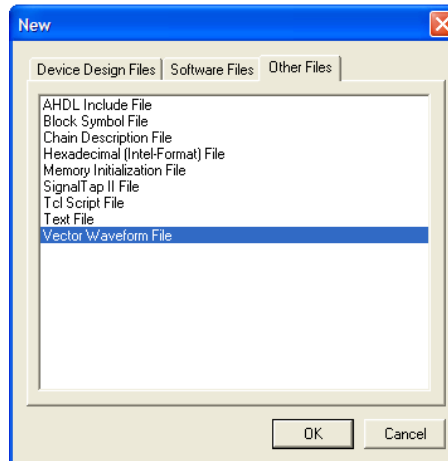
15. Compile your circuit one more time to make sure there are no errors.



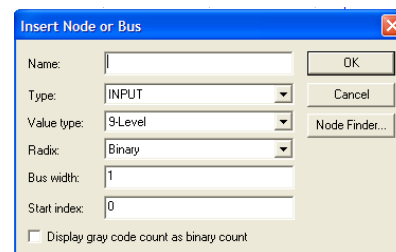
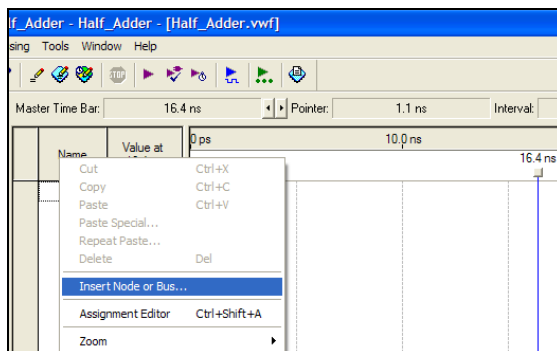
## Half Adder Simulation Instructions

The simulation is often referred to as post-synthesis, because it simulates the actual gates and cells of the target device (the device that will be programmed).

16. Before we start our simulation, we have to specify input values for the half adder (A and B).
  - a. To do this, from the file menu, select **New** and then **Vector Waveform File** in the **Other Files** folder to open the waveform editor window as shown below and click OK.



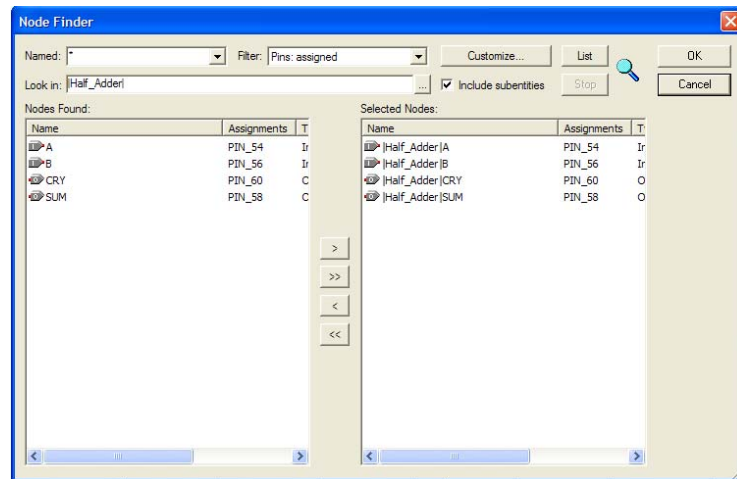
- b. The waveform file is initially blank and we need to enter our node names and their associated waveforms. Save the waveform as Half\_Adder.vwf
17. Use the right mouse button and select the **Insert Node or Bus**. This brings up the **Insert Node or Bus** window as shown below:



18. Click on Node Finder and select Pins: Assigned and click List to view all the nodes that apply to our circuit (A, B, SUM and CRY).
19. Next highlight all the nodes and click the >> symbol button to copy all nodes to the Selected Nodes list as shown below.

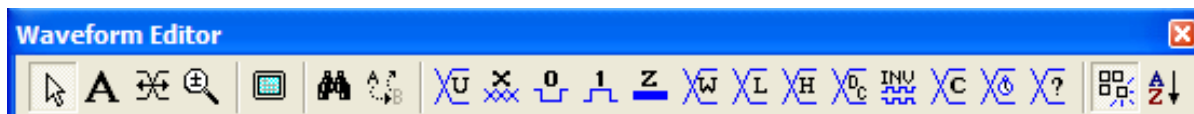


20. Click OK to accept all the nodes inputs and outputs.

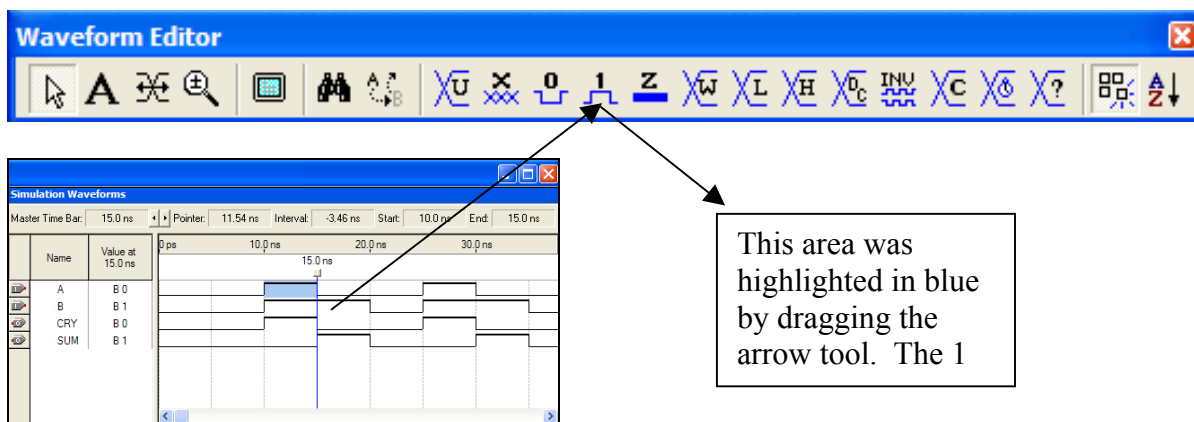


21. Defining the waveforms for our inputs:

- a. In order to define waveforms for our inputs, we can take advantage of the tools provided by the **Waveforms Editor tool bar** shown below.



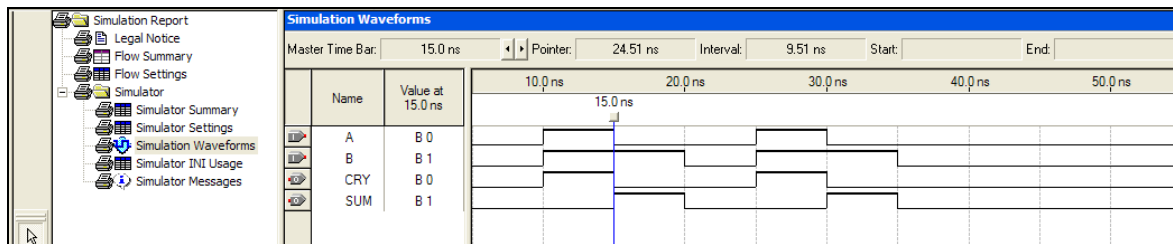
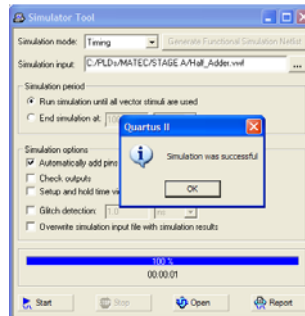
- b. Select the **Waveform Editing Tool** (the icon with crossing waveforms and a two-headed arrow inside it). When this is selected, you can use your mouse to paint your waveform in the waveform area of below figure to paint 0s and 1s.
- c. Alternatively, you can select the arrow in the Waveform Editor, and in your waveform area paint a portion of the waveform. The painted area of the waveform will highlight. Then select one of the icons to set a value in the highlighted as shown below:





## 22. Run the timing simulation.

- From the main menu Quartus® II screen menu, select **Tools** then **Simulator Tool**. The simulator tool will appear as shown below.
- On the Simulator Tool, select **Timing** in the Simulation mode selection box.
- Click Start and upon completion, the *simulation was successful* prompt appears.

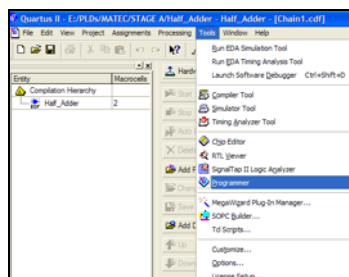


NOTE: When you run your timing simulation, you might see a delay from the inputs A, B to the outputs CRY and SUM. This delay is due to the delays of logic cells of our CPLD and the delay of its IO cells. A successful simulation run verifies the correctness of our design of the half adder as well as the timing of the input waveforms. If we change the waveforms too fast, the outputs cannot catch up and we will end up with had-to-justify output waveforms.

## 23. Programming the target PLD

- The necessary files for this purpose are generated in Quartus® II after a design has been successfully compiled. MAX 7000S devices use the *pof* (Programmer Object File) programming file format and the FLEX 10K device use the *sof* (SRAM Object File).
- The files that are generated by the compiler include all necessary configuration data for the appropriate PLDs.

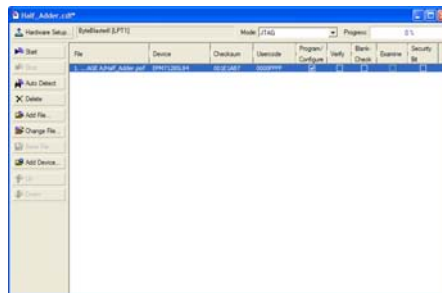
24. To start the device programming process, select the Programmer from the **tool** menu as shown below.







25. This brings up the device configuration window, as shown below.

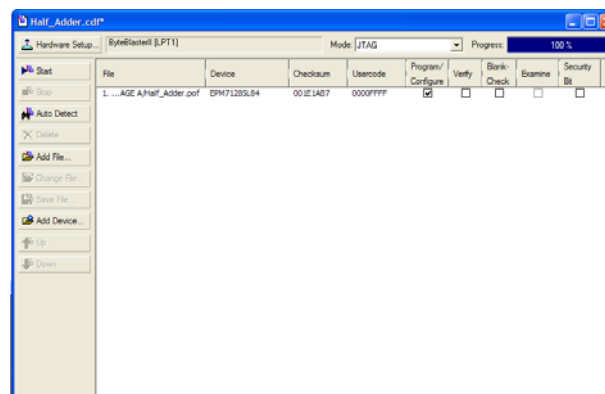


26. Before you click start, make sure that the development board containing the target device has been properly configured for programming.

- a. For our half-Adder circuit, the ByteBlasterII has been selected and assigned to port LPT1 using JTAG (Joint Test Action Group) mode.
- b. If the Quartus® II environment is being used for the first time, setup the programmer hardware.
  - i) Click on *Hardware Setup*.
  - ii) In the Hardware Setup window that opens, add your hardware, select it, and close the window.
  - iii) In our Half\_Adder circuit, we have specified ByteBlasterII that is connected to the LPT1 port of our computer. ByteBlasterII is a device programmer by Altera for Altera devices, and connects to the parallel printer port of your PC. The UP2 development board uses this hardware for programming its devices.
  - iv) The selected Mode of programming is JTAG, which is the way ByteBlasterII connects to UP2. Always check your development board data sheet prior to downloading.

**NOTE:** If your circuit file name (Half\_Adder.cdf) is not displayed under file, then you are still in evaluation mode license of the software. You need to go to the Altera web site and follow the instructions to update your license.

27. After you have completed your setup in the configuration window, select Start to download. *Program and configure* are terms synonymous with *download*.







28. After the downloading is completed, as indicated in the progress bar by 100% on the blue bar graph, the object design is ready for testing on the circuit board. Keep in mind that the relatively simple Half\_Adder logic circuit used for illustration will use only a very small fraction of the total capacity of the target device.

### Testing the Half-Adder Circuit in the ALTERA UP2 Development Board

#### Device Programming

The two figures under Device Programming show the Quartus® II programming window that is used to program an EPM7128S device in JTAG programming mode. The device is connected to the LPT1 port through the ByteBlaster II programming cable.

In this section, we will show how the UP2 development board should be setup for its EPM7128S device to be programmed with the Half\_Adder example of this lab.

29. Connect the LPT1 side of ByteBlaster II to your computer and the JTAG end of it to the JTAG\_IN of the UP2 board to program the EPM7128S device of the UP2 board.
30. Set the jumpers on the UP2 according to Jumper Setup under (Programming) for programming the MAX device.

TDI	TDO	DEVICE	BOARD
C1	C1	C1	C1
C2	C2	C2	C2
C3	C3	C3	C3

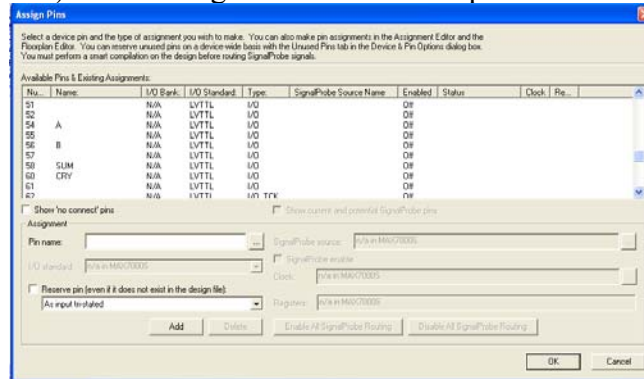
Desired Action	TDI	TDO	DEVICE	BOARD
Program EPM7128S device only	C1 & C2	C1 & C2	C1 & C2	C1 & C2

31. Click on the *Start Programming* icon of the Programming tool bar to program the EPM7128S devices with the Half\_Adder.pof file that corresponds to our Half\_Adder design circuit.

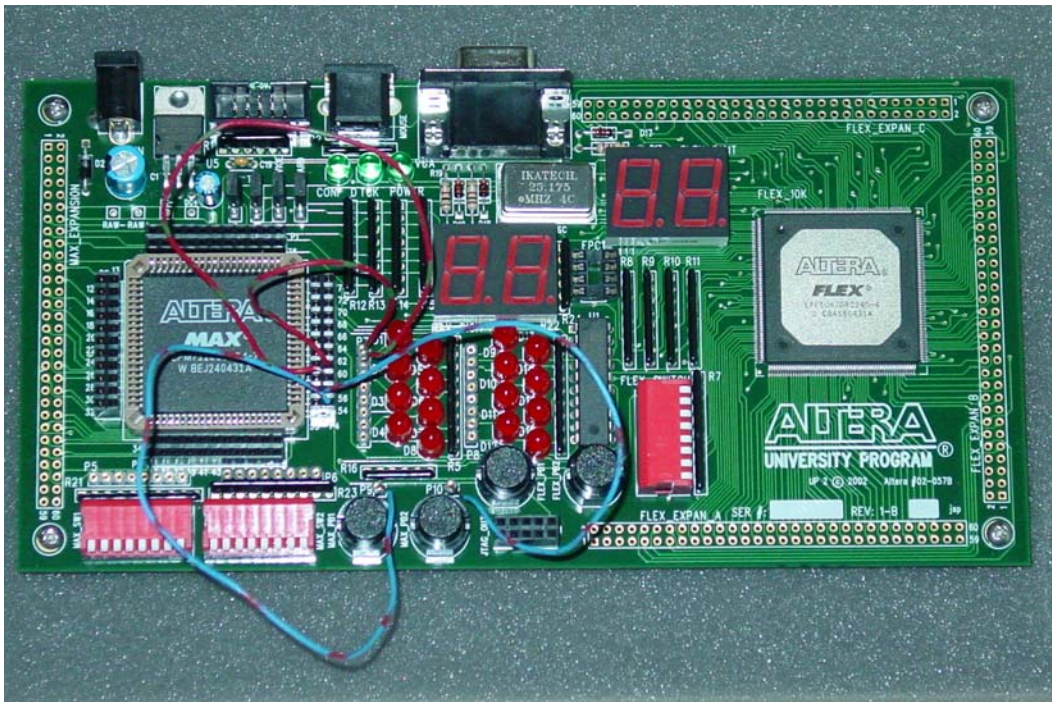


## Testing the Design

After performing the above steps, our Half\_Adder project implemented on the MAX 7000S device of the UP2 board is ready to be tested. As discussed before, push-buttons and LEDs are not pre-assigned to EPM7128S pins. The figure below shows that input ports (A and B) of our example design are assigned to device pins 54 and 56, and the outputs (SUM and CRY) of our design are connected to pins 58 and 60.



32. Make the following connections as shown in the figure below.
  - a. Connect UP2 push- buttons and the D2 LED to pins 54 and 56.
  - b. Connect the UP2 D3 LED to pins 58 and 60 respectively.



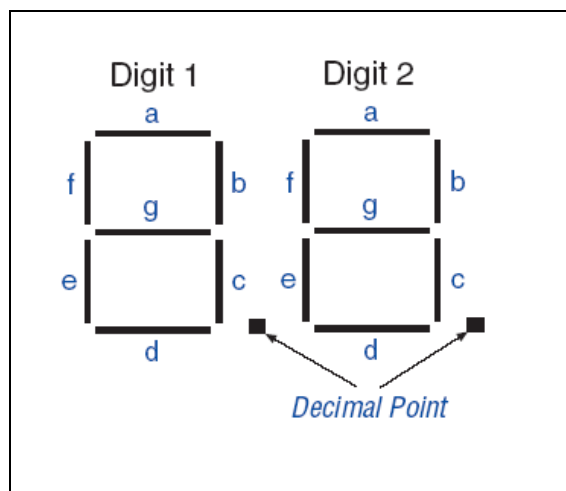


33. With these connections, we are ready to test the EPM7128S implementation of our *Half\_Adder* design. Since the push-buttons are normally 1, the output CRY of our Half\_Adder design becomes 1 when the push-buttons are not pressed. However, since an LED is illuminated when logic 0 is applied to it, the D2 and D3 LEDs are on when both push-buttons are pressed. By pressing one or both push-buttons, the output LEDs will be illuminated or off (indicating the appropriate outputs for SUM and CRY).

**Note: Input, Bit 0 = Push-Button is pressed. For either Max\_PB1 or Max\_PB2**

INPUTS		OUTPUTS	
Push-Button A	Push-Button B	D1 LED SUM	D2 LED CRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

34. Notice that Segment *b* of MAX display digit 1 is also illuminated when a both push-buttons are not pressed. This is because Pin 60 EPM7128S is pre-assigned to this display segment and it is our CRY output of the Half\_Adder circuit.





Display Segment	Pin for Digit 1	Pin for Digit 2
a	58	69
b	60	70
c	61	73
d	63	74
e	64	76
f	65	75
g	67	77
Decimal point	68	79

### Putting the Half Adder into a Box

We want to be able to put the circuit into a “black box” or icon and still be able to see the internal guts of it. We want to represent our half-adder with a device so we can use it to design the Incrementer-4 circuit that is needed for our Arithmetic Logic Unit (ALU) circuit.

The procedure for putting the circuit in a box is summarized below:

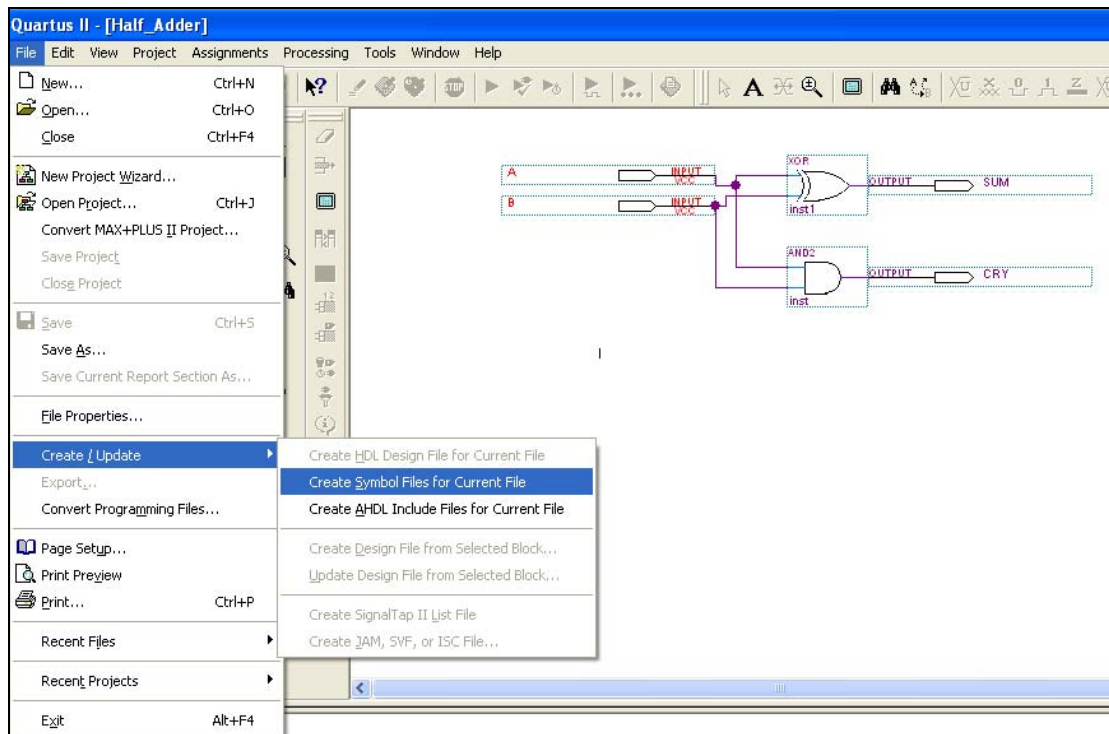
The first thing that is needed is to make your circuit an element of a larger circuit by developing a picture or symbol of what it should look like.

The picture needs to identify your circuit well enough that you will not forget what it is. It also needs to be as compact as possible, so it does not use too much valuable screen space. It also should adhere to some sort of standard, so other people could look at the picture and know what kind of circuit it is. Finally, your picture should also have line spacing compatible with Altera, so the pins that you have pictured coming out of the device are able to connect smoothly with Altera software lines.

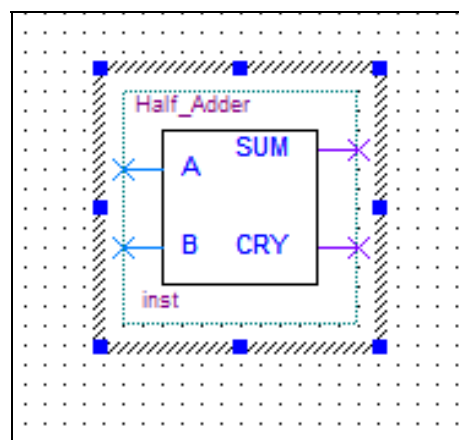


### Lab Procedure: Putting the Half Adder into a Box

1. Open the Half\_Adder schematic (Half\_adder.bdf)
2. Create the Half\_Adder device by selecting **File**→**Create/Update** and then select “*Create Symbol File for Current File*” while the circuit for the half\_adder is still open.



3. You should get the following symbol:

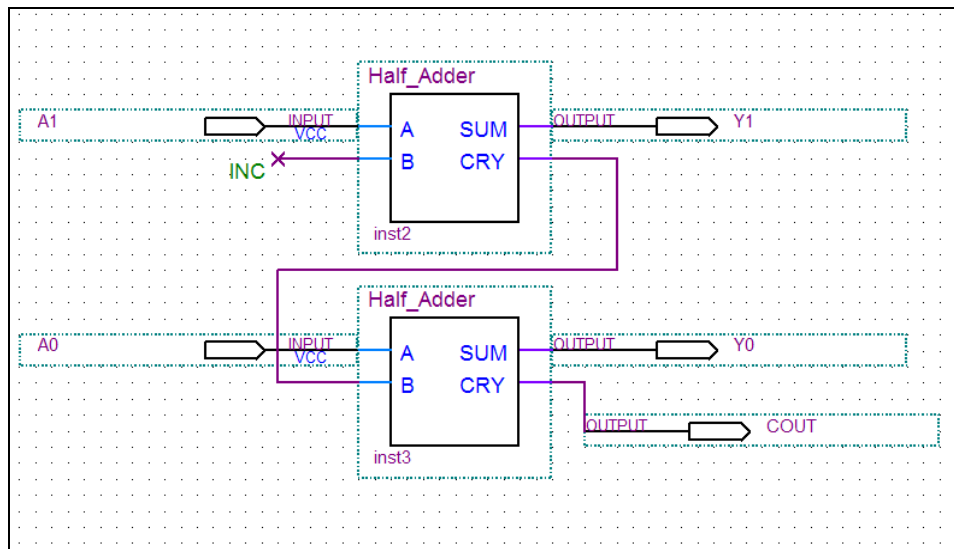


4. Test the device and works according the truth table of Half\_Adder circuit. Remember, we will use the half\_adder device instead of the circuit to create larger designs.
5. For any changes to the locations of pins, you can always select **Edit**→**Edit Symbol** from the main menu.



## Creating a 4-bit INCREMENTER

Another arithmetic operation is the **INCREMENT** operation, where we increase the input by 1. A **HALFADDER-1 (-1 means one bit)** actually is an **INCREMENTER by 1**, with the carry input (B), serving as a control input bit which tells the circuit whether or not to increment the number. There is a one bit output and a carry output. Similarly, an **INCREMENTER by 2** can be formed out of two **HALFADDER-1's** as follows:



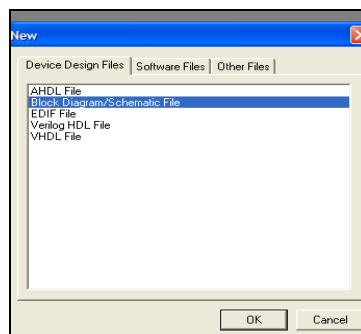
INCREMENT-2

In a similar fashion, we can use four half adders to form an **INCREMENT-4**. An **INCREMENT-4** should have one four-bit binary number and a carry as input. It should also have one four-bit binary number and a carry as output. If the input carry is low, the output number will be the same as the input number. If the input carry is high, the output will be more than the input.

This is an excellent example of how we create and build a larger circuit from a smaller one.

### Build an INCREMENT-4

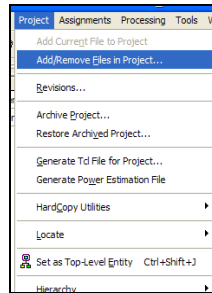
6. From the main menu, select File, New and click on Block Diagram/Schematic File as shown below:



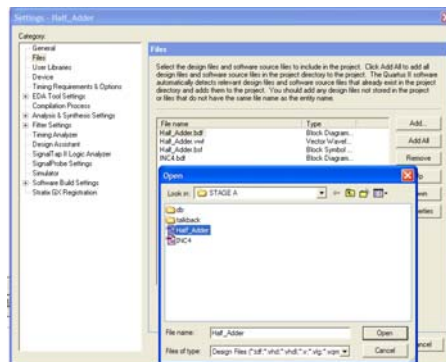
7. Save the new schematic as INC4.bdf (INC4=Incrementer 4).
8. Save the project as (INC4.qpf).



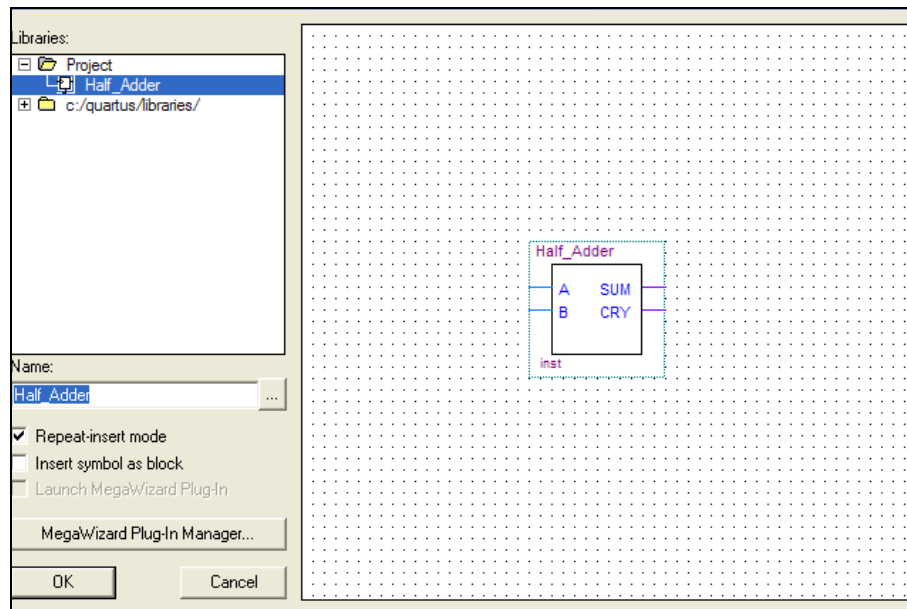
9. From the Project pull-down menu, select Add/Remove Files in Project as shown below to add the half\_adder sub circuit to the incrementer 4.



- Click on ADD and browse for your half\_adder.bdf.
- Click Open as shown below. The Project will be included in the new project.



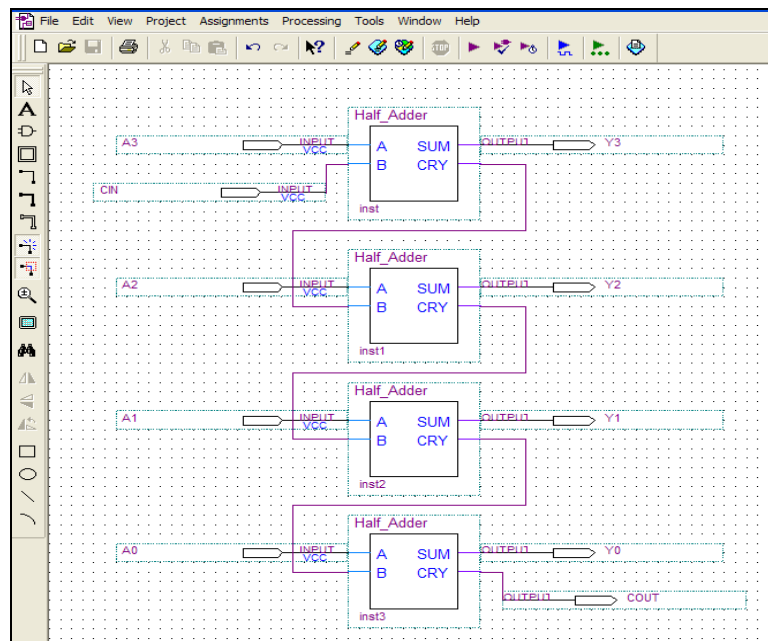
10. Click on Symbol Tool and select the Half Adder as shown below:







11. Click OK and insert 4 of the half\_adder symbols on the screen and connect them as shown below.

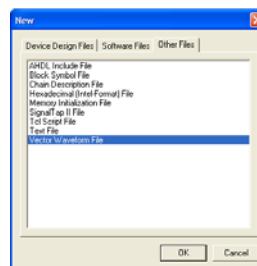


12. Insert *inputs* pins  $A_0$ ,  $A_1$ ,  $A_2$  and  $A_3$  and  $C_{IN}$  (carry-in), and outputs pins ( $Y_0$ ,  $Y_1$ ,  $Y_2$ ,  $Y_3$  and  $C_{out}$  (carry-out)) and connect them using the Orthogonal node tool as shown above.
13. Compile the incrementer 4 circuit using the same step illustrated in previous discussion. The compiler will indicate if there is any problem with your design work.
14. The flow summary should show that the INC4 design uses 5 macrocells of a MAX 7000S device.

### Test the INCREMENT-4 using Hexadecimal Numbers

Since 4-bit binary numbers run from 0 to 15 in decimal form, a convenient way to test 4-bit circuits is to use base 16 or Hexadecimal arithmetic. A hexadecimal capability is included in Quartus II Altera software.

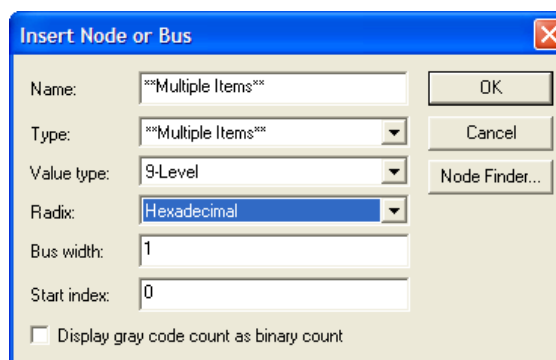
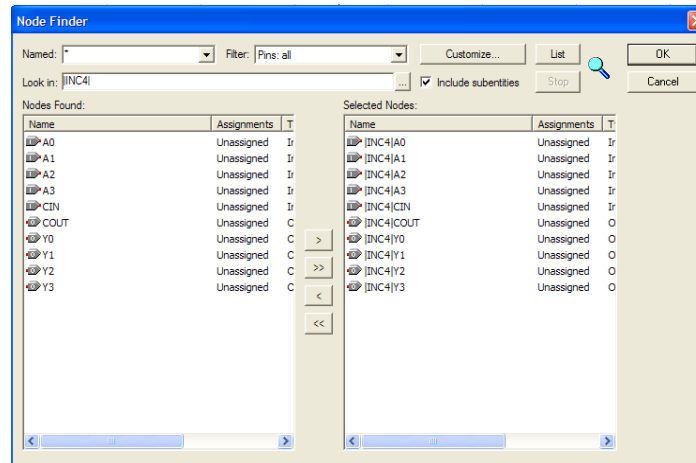
15. Using Hex arithmetic, verify that the **INCREMENT-4** works in the desired fashion. In your comment section of the questions, explain why 32 tests are needed. Record 16 combinations of the 32 of your test results and include them in your report.
  - a. Select *New* and then *Vector Waveform File* in the *Other Files* folder to open the waveform editor window as shown below and click OK.



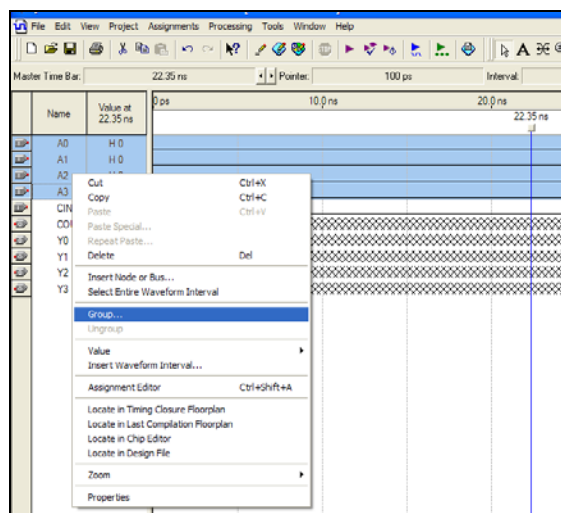




- b. Use Node finder as illustrated in previous exercise list all inputs and outputs pins as shown below and then click OK.

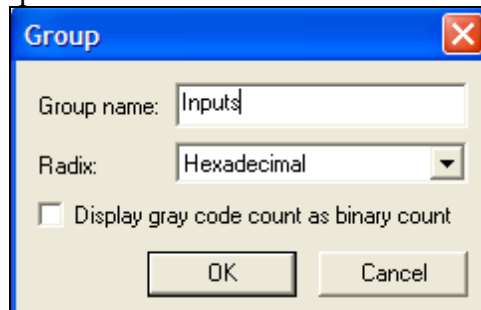


- c. Group the 4 inputs (A0...A3) by highlighting all 4 inputs, click the right key mouse and select Group.

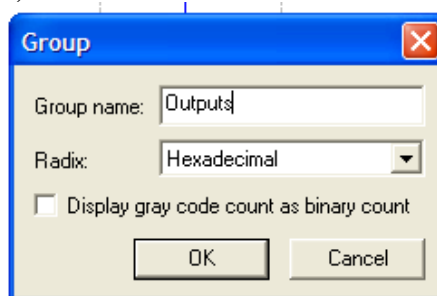




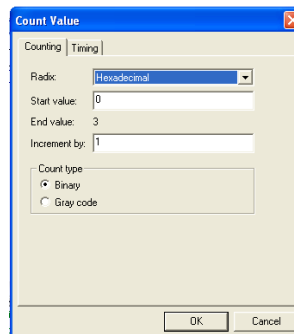
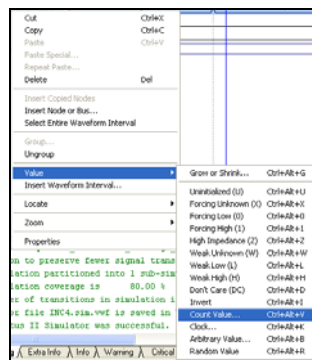
- d. Select a name for your group of inputs.



- e. Do the same for the outputs (Y0...3).

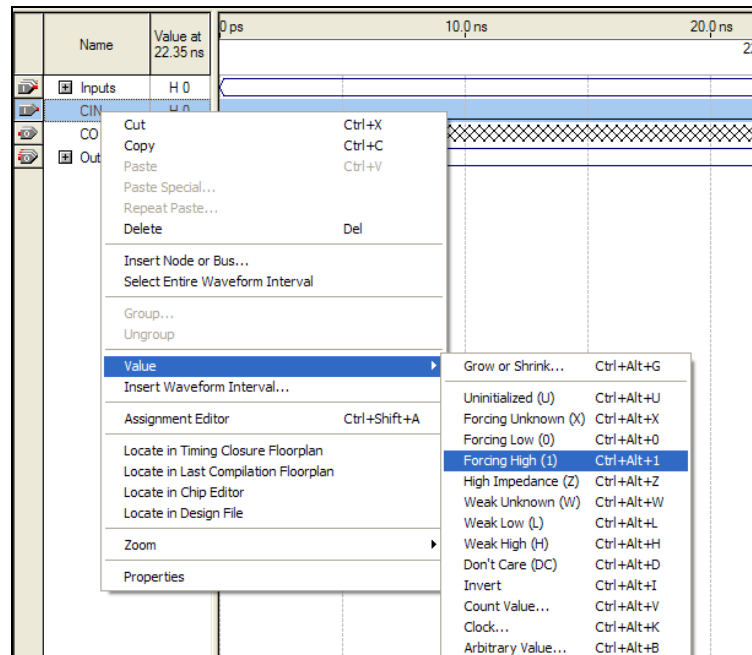


- f. For the Inputs, highlight the inputs to make it count from 1-F (Hex. values).  
g. Click the right clock mouse and select count values as shown below.



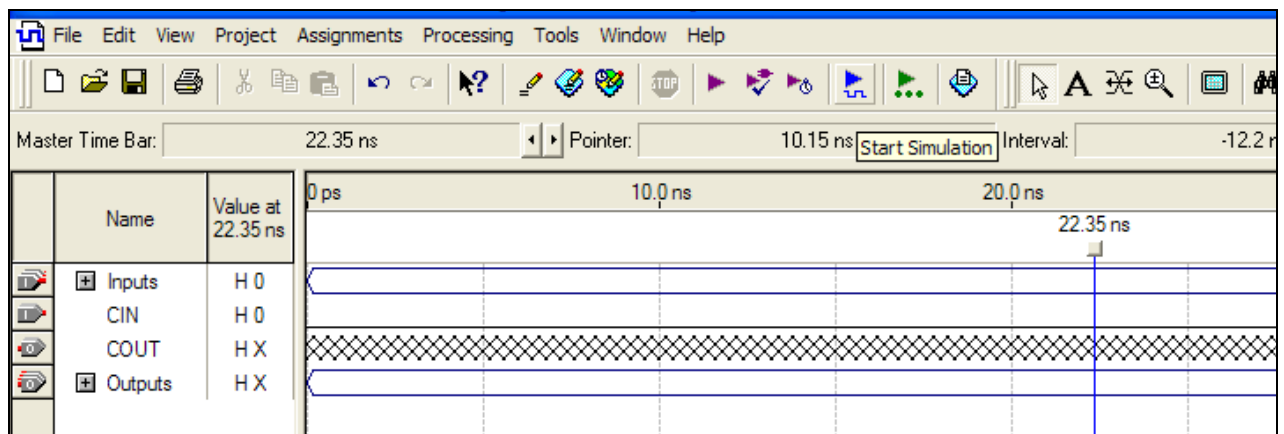


- i) For the  $C_{in}$ , make the input always high by highlighting the input and select *Forcing High (1)* under value.



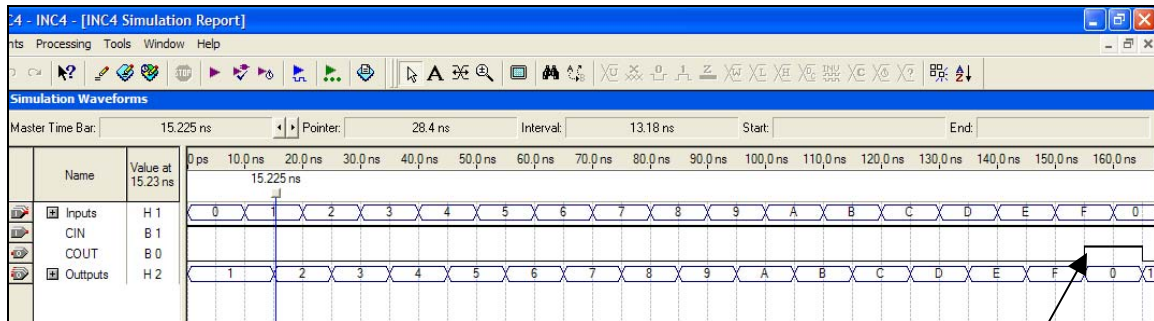
- h. Run the timing simulation by clicking on the start icon.

Note: Make sure you give the simulation file a name that is the same when you created the circuit.



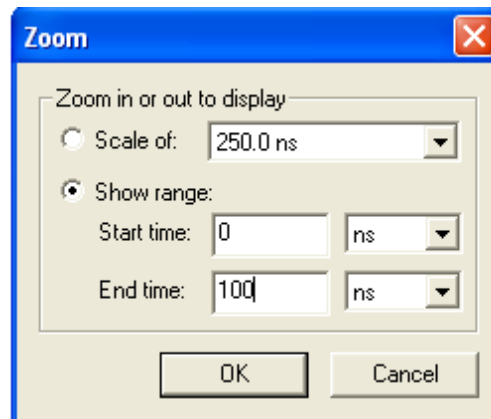


- i. Once your simulation is completed, you should be able to observe, the operation incrementer 4. Given each input a hexadecimal number and  $C_{in}=1$ , we should get an increment of 1 for every input. Notice the  $C_{out}$  output when you add F+1 as shown below. Notice, the time delay is about 5 ns as was indicated by the compiler.



F+1=0 with Cout =1

- j. You can zoom to the right values by clicking the right hand mouse and select the appropriate range for your zoom:

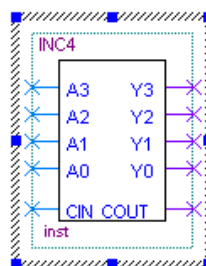
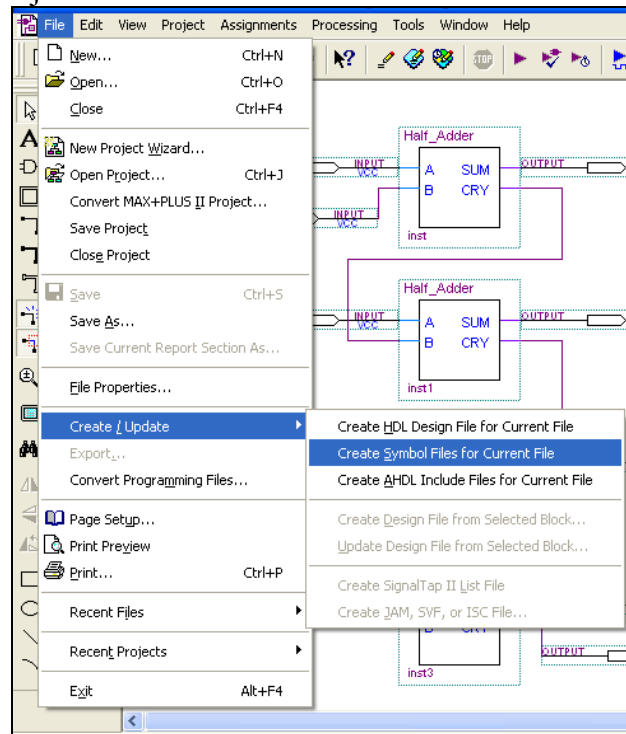




## Making an INCREMENT-4 Device/Symbol

16. Repeat the procedure used before to develop Half\_Adder Symbol.

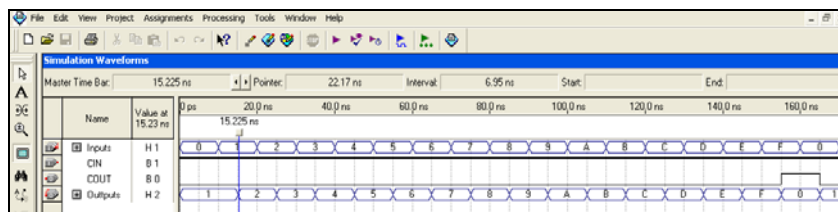
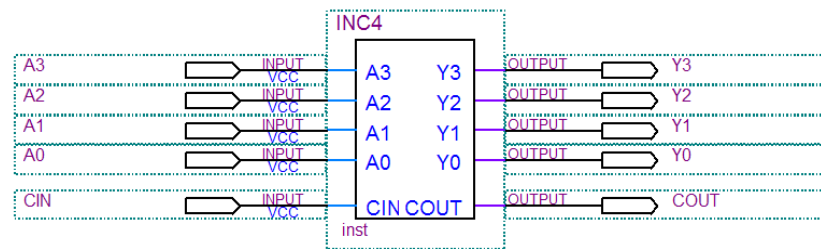
- Select from **File**  $\Rightarrow$  **Create/Update** from the main menu. Select **Create Symbol for current file** from the new dialog box and create a box for the circuit.
- Save the inc4 box to your project.



Incrementer Symbol

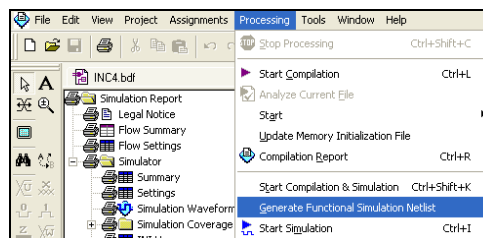


17. Test the incrementer symbol:

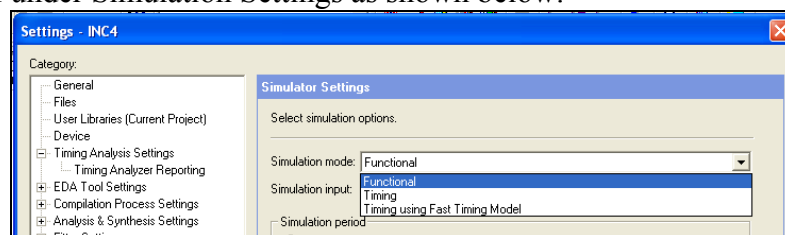


a. In the test, the time delay is about 5 ns

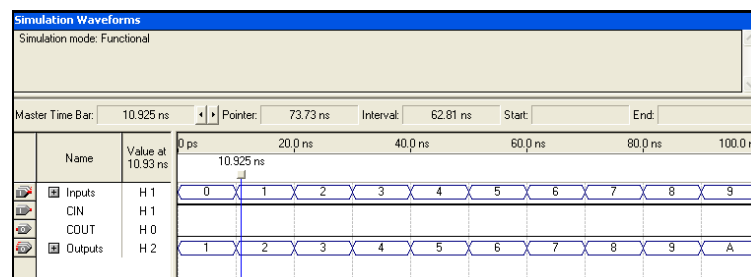
- i) To get rid of the timing delay, create “**Generate Functional Simulation Netlist**” from **Processing** main menu as shown below.



ii) Select Functional under Simulation Settings as shown below:



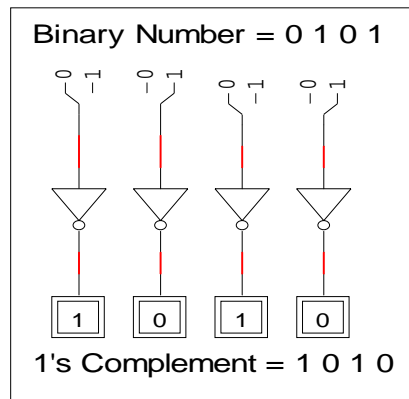
iii) Here are the results with **NO** timing delay:





## Using the INCREMENT-4 to find Two's Complement Representations

The 1's and 2's complement of a binary number are important because they permit the representation of negative numbers. The method of 2's complement arithmetic is commonly used in computers to handle negative numbers. The 1's complement of a binary number is found by simply inverting all the binary numbers. This is done by changing them from 1's to 0's or from 0's to 1's as shown in this example.



The 2's Complement of a binary number is simply found by adding 1 to the Least Significant Bit (LSB) of the 1's complement.

$$2's \text{ Complement} = (1's \text{ Complement} + 1)$$

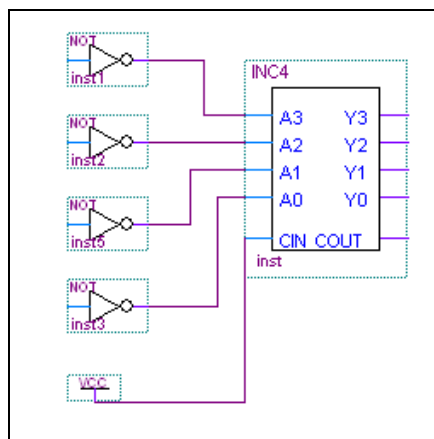
Example:

2's Complement of the above binary number (0101) is:

$$\begin{array}{r} 1 \ 0 \ 1 \ 0 \\ + \quad \quad 1 \\ \hline 1 \ 0 \ 1 \ 1 \end{array}$$

1's Complement  
2's Complement

The following circuit should do a two's complement of a binary number:



Note that by connecting INC to VCC, INC is always a 1.

18. Build the circuit in Quartus® II Altera and test whether it really performs a two's complement. It is up to you to decide what a good test is but be sure to record the details of your tests. You will use a similar circuit for two's complement toward the end of the lab.

**Questions:**

1. Use the timing analysis of the half adder that you have created in the this lab to fill up the following truth table:

INPUTS		OUTPUTS	
A	B	SUM	CRY
0	0		
0	1		
1	0		
1	1		

2. Switch the inputs on the trainer for the Half-Adder where input A is attached to pin 56 and the B input to pin 54. Verify the truth table of the Half-Adder.
3. Use the incrementer-4 device or circuit to find the **One's** complement of the following Hexadecimal numbers:
- | <u>HEX:</u> | <u>Binary</u> |
|-------------|---------------|
| a. A        | 1010          |
| b. F        | 1111          |
| c. C        | 1100          |
| d. 7        | 0111          |
4. Use the incrementer-4 device or circuit to find the **two's** complement of the following Hexadecimal numbers:
- A
  - F
  - C
  - 7





Use the rest of the page for your observation of how the Half-Adder programmed using a PLD. Be sure to note components that are used and answer the questions asked during the lab.