Tutorials for ISE/WebPACK Version 14.7 Schematic Capture

Krista Hill (kmhill@hartford.edu)

Contents

1 Introduction

The Xilinx ISE/WebPACK package allows you to use schematics, hardware description language (HDL) files, and specially designed modules to design logic based systems. The purpose of the tutorials is to get students quickly using the software and associated hardware so the presentation is intentionally brief and to the point. We have found that the sooner students become familiar with CAD tools, the sooner they can start mastering the underlying concepts, that is they can learn to design and use logic circuits. Besides ISE/WebPACK it is important that you consider obtaining the following:

- A Windows 7 compatible PC, Windows 8 should also work. The Linux installation is not presented in this document
- JTAG programming cable used to configure the device, see section 9
- Programmable logic device module such as the Digilent, Inc. [2] CMOD or XMOD [5] CPLD module
- Hardware such as XMOD [5] logic trainer or breadboard. The XMOD logic trainer documentation includes schematics so you can breadboard as much of the trainer as you like.

The tutorials and additional content are also available from our tutorials webpage[4] as well as the XMOD project page[5]

2 Installing Xilinx ISE 14.7

This tutorial outlines how to install ISE/WebPACK version 14.7 on a Windows 7 computer that you have administrative rights to. The release notes for ISE/WebPACK[3] gives all the details regarding supported operating systems and configurations. To summarize a few points:

- Xilinx officially supports 32-bit and 64-bit versions of Windows XP Professional, Windows 7 Professional, Windows Server 2008 (64-bit) only, Red Hat Linux Enterprise Workstation 5 and 6, as well as SUSE Linux Enterprise 11.
- Users of 32-bit Windows who are targeting the largest devices and most complex designs may encounter a memory limitation. For ISE 14.7 to take advantage of the memory increase feature on 32-bit Windows XP Professional systems it is necessary to modify the Windows settings. See the release notes[3].

ISE/WebPACK is also known to run on some other operating systems that are not officially supported by Xilinx. To learn more, Xilinx has a technical support group where users and Xilinx staff discuss topics.

Obtaining ISE

There are at least three ways to obtain the ISE installation files. Note that while there is no charge to use ISE/WebPACK, a license is still necessary and can be obtained at no charge during or after installation.

- Download ISE from Xilinx. The archived installer is large in size (over 6GB) so use a wired Internet connection if you can and give yourself plenty of time. To extract the files see the notes below. Also note that Xilinx has other methods beside downloading files to track the use of ISE/WebPACK.
- To get a copy from a classmate by using a large memory stick (8 GB or larger), be sure to get the multi-file version as the complete archived installer might be too large (over 6GB) for the file system in your memory stick (probably FAT32 file system). Be sure to empty the trash can before copying files to the memory stick.
- Request media from Xilinx, for delivery by mail.

The following outlines how to download ISE and unpack the files:

- Use a browser to open the Xilinx webpage, you may want to consider Internet Explorer.

  http://www.xilinx.com

- In the top-center of the page click the 'Sign-in' link and login. If you don't already have a Xilinx account then in the pop-up window click on the 'Create account' button to create an account so that later you can login. There is no charge for having a Xilinx account.
- To the middle-right of the top of the page click the 'Download' link. In the new page, near the center of the page click on the 'ISE Design Tools' tab. Next, to the left click in ISE version 14.7 and then pick an installer to download.
- If you are only using Windows and have a good Internet connection then pick the 'Full Installer for Windows'. The file will contain a tape archive file (.tar) which is compressed as a GNU zip file (.gz). Besides Windows itself, there are many free programs such as 7-zip (http://www.7-zip.org/) which are known to be able to uncompress and extract the contents of such files.

Running the ISE Installer

The following outlines how to install ISE into a 32 bit or 64 bit Windows 7 computer.

- Once the installer file is uncompressed and the contents are extracted, then open the newly created folder and examine the contained file names.
- Double-click to execute the file 'xsetup'. In the new window, click 'Next' and in each of the following two windows click that you accept the terms of the licenses and click 'Next'.
- In the 'Select Products to Install' select ISE WebPACK and click 'Next'. In the 'Select Installation Options', to save space you can choose to not install WinPCap. Click 'Next'. In the 'Select Destination Directory' just click 'Next'. Finally, in the 'Installation' window click 'Install'.
- After a significant amount of time ISE/WebPACK will install and the License Configuration Window will appear. Click the 'Acquire License' tab, click to select the 'Vivado/ISE WebPack License' and click 'Next'. In the pop-up window click 'Connect Now'. A browser window will open asking you to login to your Xilinx account. In the next window review your information and click 'Next'. Select the ISE WebPACK license option and click 'Generate Node-Locked License'. In the pop-up window and the following window, click 'Next'. Xilinx will send the license in an email attachment, the file is probably named Xilinx.lic. Save the license file to a location such as C:\Xilinx\14.7\
- Back in the License Configuration Window, click the Manage Xilinx Licenses tab and click the 'Load License' button. Navigate to the license file and click 'Open'. A pop-up window will tell you that the file was successfully copied, click 'OK', finally click 'Close'
- Back in the Installer window there might be a message that the environment variables are written to specific files (C:\Xilinx\14.7\ISE_DS\settingsXX.bat where 32 and 64 refer to 32 and 64 bit versions). As soon as the installer is done then find and double-click the 32-bit or 64-bit file appropriate for your machine and save a note just in case there is a need in the future to reference the file. Click the 'Finish' button. The installation is complete

3 PDIR is For Project Folders

Take a moment to consider the storage media where you will work your projects. ISE/WebPACK makes intensive use of storage media and the files are somewhat large so don't consider using a floppy disk.

- If you are using your own computer then the C: drive or other attached hard drive is a good choice.
- For a computer lab ask if there is a networked drive available. The advantages of a networked drive are that it can be accessed from any lab computer and it's likely going to be a RAID system which means it's more reliable than any one disk drive.
- You can perform the tutorials with a Flash memory stick but we discourage you from using one to work significant projects as the write time is slow. A better choice is an external USB hard drive, such devices are available for less than $50 from a number of vendors. You can use a Flash memory stick to archive and transport a project.

Create a Projects Folder

Next, create a folder in the selected drive to store your project folders. We will use PDIR to refer to the folder containing your ISE/WebPACK project folders, on the selected storage media. If your username is UNAME then on your own computer the following could be the path to the folder containing your ISE projects:

```
C:\Users\UNAME\Xilinx
```

For now observe the following points:

- Have all your source files including schematics, description files, and constraint files for a project in that project folder.
- If you are not planning on using a project for some time then in the ISE Project manager be sure to "clean" the project to reduce the size of the project folder by deleting temporary work files. With the mouse select the following and in the pop-up window click OK.

  ```
  Project >> Cleanup Project Files...
  ```

- To save even more space you can archive the project, see the next section.
- Finally a word of caution, the Xilinx tools use many self generated work files and on rare occasion the tools will become confused. If the Xilinx tools start acting particularly odd, then "clean" the project as previously described.

Moving a Project Folder

In time you may want to clear up some significant space from an ISE project or you may want to move a project to a different computer. Rather than just copying an ISE project folder it makes more sense to archive a project to a compressed .zip file.

- Clean up the project folder. Select the following and click OK.

  Project >> Cleanup project files...

- Select a location for the archive

  Project >> Archive...

- In the pop-up window click the three-dot icon (...) and navigate to to the save folder location and click Save. Back in the Project Archive window click OK. After a few minutes the archive process will be complete. To restore your project you can use a program such as 7-zip to uncompress the .zip file.
- With the project archived you can close and then delete the original project file.

4 Configure a CPLD with Simple Gates

This section provides a quick-start for the purpose of configuring a CPLD with simple logic gates and provide a general overview of the process in using ISE/WebPACK. As such, manual wiring can then be used to produce simple logic circuits. For this section you will need the items:

- PC with Xilinx ISE installed, see section 2
- Logic trainer or breadboard and 3.3 Volt power supply
- CPLD module, either CMOD-XC2C64A or XMOD-XC9536XL, see [2] or [5]
- Breadboard wires
- JTAG Programming cable, see appendix section 9

Figure 2 shows the XMOD as well as the CMOD, configured with *AND*, *OR*, as well as *NOT* gates. Besides the actual description file used to describe logic circuitry, it helps to have a .ucf constraints file to assign signals to the CPLD pins. For the purpose of expediency VHDL code is used. The files here should be self explanatory but for more detail see a VHDL reference such as Ashenden [1].

The pinout for the XC9536XL used in the XMOD is close to that of the XC2C64 used in the CMOD so that here a single .ucf suffices. The letters NC mean no-connection and NU mean not-used and are due to the fact that CMOD has one fewer accessible I/O pin than the XMOD. You can use the unused pins for other purposes, for example the XMOD can accommodate a fifth NOT gate.

Figure 1 is the description file expressed with VHDL code. Lines in a VHDL file that start with a pair of hyphens (--) are comment lines. The first few lines of the VHDL file are comments that contain the file name, author name, date, and a statement about the file. Figure 3 is the user constraints file (.ucf) which assigns signals to CPLD pins. Lines in a .ucf file that start with a hash character (#) are comment lines. Likewise, the first few lines of the .ucf file contain the file name, author name, date, and a statement about the file.

```
----------------------------------------------------------------
-- SimpleGates.vhd - YourName - TheDate
-- Describe AND, OR, NOT operations for CMOD/XMOD
----------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity SimpleGates is
  Port ( A_and, B_and, A_or, B_or : in  STD_LOGIC_VECTOR (1 to 4);
         A_not : in  STD_LOGIC_VECTOR (1 to 4);
         F_and, F_or : out  STD_LOGIC_VECTOR (1 to 4);
         F_not : out  STD_LOGIC_VECTOR (1 to 4));
end SimpleGates;
architecture Arch of SimpleGates is
begin
  F_and <= A_and and B_and;
  F_or  <= A_or  or  B_or;
  F_not <= not A_not;
end Arch;
```

Figure 1: Description of simple gates

Preparing a Design

To prepare a design for use with a CPLD we will use ISE to compile the description file with the user constraints file to produce a .jed type image file which represents the configured CPLD. As outlined in section 3 be sure to create a directory for storing your ISE projects. We used PDIR to refer to your ISE projects folder.

- Start ISE/WebPACK.

  `Start >> All Programs >> Xilinx...>> ISE...14.7 >> ISE...Tools >> Project Navigator`

- After a few moments the ISE Project Navigator will open. Click 'OK' to close the 'Tip of the Day' pop-up window, if it appears. Next, in the ISE Project Navigator window start a new project.

  `File >> New Project...`

- In the New Project Wizard enter the following, you can click the three dot browse button (...) to the right of the Location and Working Directory to navigate to each respective directory. Click Next.

`Name:`	`SimpleGates`
`Location:`	PDIR\SimpleGates
`Working Directory:`	PDIR\SimpleGates
`Top-level source type:`	`HDL`

- In the next window enter the following values. The term DEFAULT refers to a given value which and should not be changed. Click 'Next'. Alternatively the XC2C64A used in the CMOD is in the Coolrunner2 CPLDs family and also has the VQ44 package.

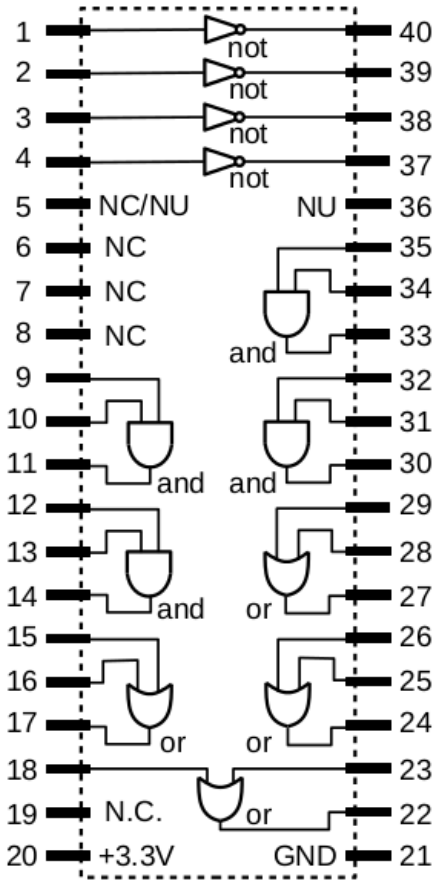

Figure 2: XMOD/CMOD CPLDs configured with simple gates

Evaluation... Board:	None Specified	Synthesis Tool:	XST
Product Category:	General Purpose	Simulator:	ISim
Family:	XC9500XL CPLDs	Preferred Language:	VHDL
Device:	XC9536XL	Property Spec...	DEFAULT
Package:	VQ44	Manual Compile Order:	Not Checked
Speed:	-5	VHDL Source...	DEFAULT
		Enable... Filtering	Not Checked

- The next window provides a summary of the values entered for the new project. Click 'Finish' to create the project and if necessary create a new project folder.
- If you plan to type in files above then for each file select the following to open a new text editor window, then click OK.

  `File >> New >> Text File`
- Once the file is typed in then select the following. Change the type to VHDL or UCF, enter the file name, then click Save.

  `File >> Save As`
- To add a source files to the project open the Add Source window and select:

  `Project >> Add Source...`
- In the ISE Project Navigator window, to the lower left click the 'Design' tab. Just above in the Hierarchy pane click to select 'SimpleGates'. Just below in the Processes pane click the + to expand the 'Implement Design' category. Right click on 'Generate Target Device' and select 'Run'.
- After a few moments either the programming file will be produced or an error will be reported, in which case you will fix the error and try again.

```
#-----------------------------------------------
# SimpleGatesXMOD.ucf - YourName - TheDate
# Pins for XMOD-XC9536XL or CMOD-XC2C64
#-----------------------------------------------
#
# NOT gates
NET "A_not<1>" LOC = "P12"; # Pin-01
NET "F_not<1>" LOC = "P8";  # Pin-40
NET "A_not<2>" LOC = "P13"; # Pin-02
NET "F_not<2>" LOC = "P6";  # Pin-39
NET "A_not<3>" LOC = "P14"; # Pin-03
NET "F_not<3>" LOC = "P5";  # Pin-38
NET "A_not<4>" LOC = "P16"; # Pin-04
NET "F_not<4>" LOC = "P3";  # Pin-37
#
# AND gates
NET "A_and<1>" LOC = "P18"; # Pin-09
NET "B_and<1>" LOC = "P19"; # Pin-10
NET "F_and<1>" LOC = "P20"; # Pin-11
NET "A_and<2>" LOC = "P21"; # Pin-12
NET "B_and<2>" LOC = "P22"; # Pin-13
NET "F_and<2>" LOC = "P23"; # Pin-14
NET "A_and<3>" LOC = "P1";  # Pin-35
NET "B_and<3>" LOC = "P44"; # Pin-34
NET "F_and<3>" LOC = "P43"; # Pin-33
NET "A_and<4>" LOC = "P42"; # Pin-32
NET "B_and<4>" LOC = "P41"; # Pin-31
NET "F_and<4>" LOC = "P40"; # Pin-30
#
# OR gates
NET "A_or<1>"  LOC = "P27"; # Pin-15
NET "B_or<1>"  LOC = "P28"; # Pin-16
NET "F_or<1>"  LOC = "P29"; # Pin-17
NET "A_or<2>"  LOC = "P39"; # Pin-29
NET "B_or<2>"  LOC = "P38"; # Pin-28
NET "F_or<2>"  LOC = "P37"; # Pin-27
NET "A_or<3>"  LOC = "P36"; # Pin-26
NET "B_or<3>"  LOC = "P34"; # Pin-25
NET "F_or<3>"  LOC = "P33"; # Pin-24
NET "A_or<4>"  LOC = "P30"; # Pin-18
NET "B_or<4>"  LOC = "P32"; # Pin-23
NET "F_or<4>"  LOC = "P31"; # Pin-22
#
# End of SimpleGates.ucf
```

Figure 3: Pinout constraints for Simple Gates

The next step is to connect the programmer cable and configure a CPLD. The details are outlined in section 9.

5 ISE/WebPACK Schematic Capture

This section introduces the schematic capture tool. Other parts discuss how to perform a functional simulation and how to configure a CPLD. If you only need to configure a CPLD with some basic gates and are not yet interested in schematic capture then see section 4.

Start and Make A New Project

- To start ISE, on the desktop click left to select the following:

  `Start >> All Programs >> Xilinx...>> ISE...14.7 >> ISE...Tools >> Project Navigator`

- If a 'Tip of the Day' window opens, click 'OK' to close it. In the ISE Project Navigator window, click left to make a new project:

  `File >> New Project...`

- The first New Project Wizard window involves the project location and type. Replace (PDIR) with the actual path to the folder. You can click the three dot browse button (...) to the right of the Location and Working Directory to navigate to each respective directory. Enter the following then click Next.

`Name:`	`fulladd`
`Location:`	(PDIR)`\fulladd`
`Working Directory:`	`(PDIR)\fulladd`
`Top-level source type:`	`Schematic`

- In the next window, for the XMOD-XC9536XC enter the following values. For the CMOD-XC2C64A use the Coolrunner2 CPLDs family and the XC2C64A package. The term `Default` refers to a given value which should not be changed. Click 'Next'.

`Evaluation... Board:`	`None Specified`	`Synthesis Tool:`	`XST`
`Product Category:`	`General Purpose`	`Simulator:`	`ISim`
`Family:`	`XC9500XL CPLDs`	`Preferred Language:`	`VHDL`
`Device:`	`XC9536XL`	`Property Spec...`	`DEFAULT`
`Package:`	`VQ44`	`Manual Compile Order:`	`Not Checked`
`Speed:`	`-5`	`VHDL Source...`	`Default`
		`Enable... Filtering`	`Not Checked`

- The next window is a summary of your choices and values. Review the list then click 'Finish'.

Make a New Schematic

At this moment you should be looking at the ISE/WebPACK project manager window with a project open.

- In the project navigator window, use the mouse to select:

  `Project >> New Source...`

- In the pop-up window click the 'Schematic' source type icon and enter the following field values then click next. Remember that (PDIR) is the actual path to the folder containing your projects.

`File name:`	`fadd`
`Location:`	(PDIR)`\fulladd`

- The next window is a summary of your choices and values. Review the list then click Finish.

- Click on the 'fadd.sch' tab below the schematic pane and then either right click the tab and select 'Float' or in the project window select

  `Window >> Float`

- Double-click in the schematic area and in the pop-up window select a page size that you would like to use. For the tutorial the A size (11 x 8.5 in.) is a good choice.

You are now ready to insert components into the schematic.

Insert Components

- If the Symbols pane and the Options panels are not already visible then select the following and then click the choices to make those panes visible.

  ```
  View >> Panels
  ```

- Next, do the following:

  – In the Categories list select: Logic

  – In the Symbols list select: and2

  Now when you move the mouse in the schematic editor window an instance of a two input AND gate appears. Click left to place the AND gate into your schematic and then click to insert two more AND gates. Continue, inserting an or3 gate and two xor2 gates, so the schematic looks similar to that in Figure 4.

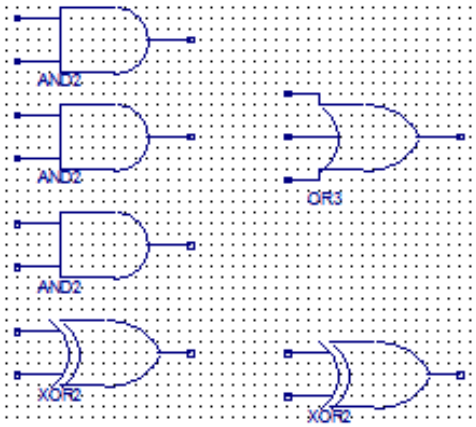

Figure 4: Gates for full-adder

- Be sure to save your work as you go along.

  ```
  File >> Save All
  ```

- To get a better view of the components you can zoom the view in or out. The zoom-to-box choice involves using the mouse to draw a box to zoom to. The zoom-to-full-view zooms out to display the entire schematic. The zoom-to-selected zooms to display selected components. The following choices are also represented with zoom icons. Hovering your mouse pointer over an icon causes a hint to be displayed.

  ```
  View >> Zoom >> In
  View >> Zoom >> Out
  View >> Zoom >> To Box
  View >> Zoom >> To Full View
  View >> zoom >> To Selected
  ```

Figure 5: The various zoom icons

Add Wires

The next step is to add wires and then add I/O markers. The action that the mouse pointer has is related to the tool mode. There are several ways to change to the wire mode. In changing from the normal or select mode to the wire mode the mouse pointer icon changes from a white pointer to cross-hairs.

- Try each of the following to change to the wire mode and then either click the select icon or press the escape key (Esc) to return to the select mode.
  - In the ISE Schematic Editor toolbar select: `Add >> Wire`
  - With the keyboard enter the *Control-W* pair
  - In the schematic area right click and select `Add >> Wire`
  - Click the *Add Wire* icon

(a)Add Wire icon

(b)Select icon

Figure 6: Wire Mode and Select Mode Icons

- Move the cross-hairs to the upper left XOR gate output pin until tiny boxes appear, then left-click the mouse. Move the cross-hairs to the upper input pin of the right XOR gate, then left-click so that a wire is placed.

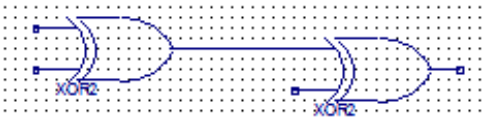

Figure 7: First wire placed

- Add additional wires so the circuit looks like the figure below. To place a bend in the wire being added, left click the mouse. To move a wire, first press the escape key to change back to the select mode. Next, point at the wire click left to grab and then move the wire. To attach a connection to a wire first change to the wire mode, point at where the new connection will be, left click, then move the mouse and finish the wire.

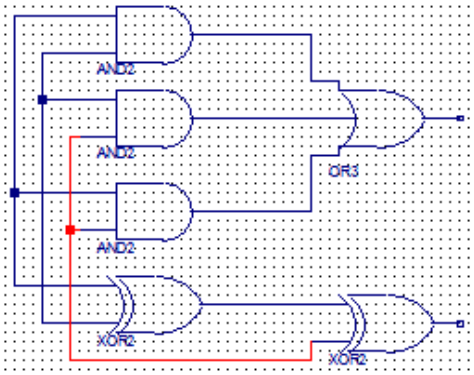

Figure 8: Full-adder wires placed

- To delete or remove a wire first select the wire as described above and then click the delete key.

Add Input/Output Markers

- A dangling wire has a connection on one end only. A red box at the end of a wire indicates that it's not connected to anything. There are two methods to place a dangling wire. Start by changing to the wire mode, using a technique described above.
  - Start a wire without pointing at any pin, that is point away from component pins and left click. Next move the cross-hairs and connect the wire to a component pin.
  - Point at a component pin, left-click to start a wire, place bends in the wire as necessary, then double-click the mouse to end the wire.
- Add dangling wires so the circuit looks like the following:

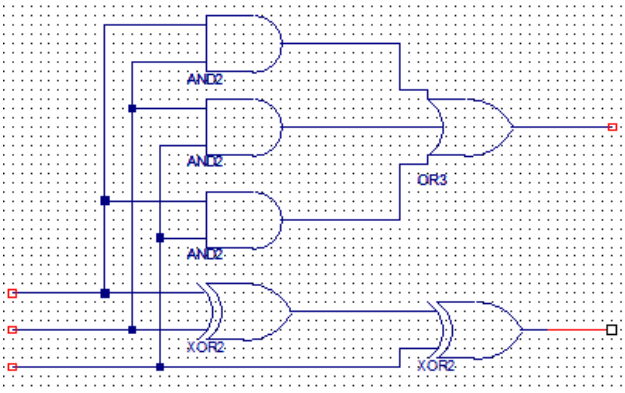

Figure 9: Dangling wires added

- Add I/O markers. To switch to the I/O marker mode do one of the following:
  - In the ISE Schematic toolbar select `Add >> I/O Marker`
  - With the keyboard enter the *Control-G* pair
  - In the schematic area right click and select `Add >> I/O Marker`
  - Click the *Add I/O Marker* icon

Figure 10: Add I/O Marker icon

- When the cursor is in the schematic area it changes to cross-hairs with an attached box. In the Options pane, click the I/O marker direction radio buttons to Add an automatic marker. In most cases the tool correctly infers the marker as being input or output type, but otherwise you can deliberately assign the marker type.
- Point the cross-hairs at the upper left dangling wire end and left click to place an I/O marker. In a similar fashion, attach input markers to other two dangling wires to the left as well as two output markers to the dangling wires to the right.
- Change back to the select mode and then double click the upper left marker. In the pop-up window look in the Category pane and click on 'I/O Markers', increase the font size to a value like 36, and click Apply. In the Category pane, click on 'Nets', then change the name to 'cin' and click OK.
- Likewise assign the names 'ain' and 'bin' to the middle and lower marker. Then assign the names 'cout' and 'sum' to the middle and lower markers to the right.

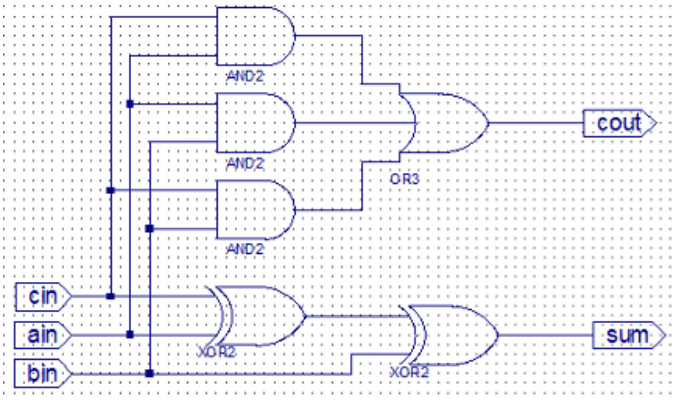

Figure 11: Input and output markers added

Add Internal Signal Names

Later when you perform simulation it will be very helpful to have the names assigned to all the wires inside your schematic.

- Change to the Net name mode. As with adding wires and adding I/O markers there are several ways to change to this mode:
  - In the ISE Schematic toolbar select `Add >> Net Name`
  - With the keyboard enter the *Control-D* pair
  - In the schematic area right click and select `Add >> Net Name`
  - Click the *Add Net Name* icon

Figure 12: Add Net Name icon

- In the Options pane, enter 'c1' in the Name field. Move the mouse into the schematic and click the wire from the top most AND gate output.
- Likewise, assign 'c2', 'c3', and 's1' so the schematic looks like the following. To increase the font size follow the same procedure that you used to modify the I/O markers.

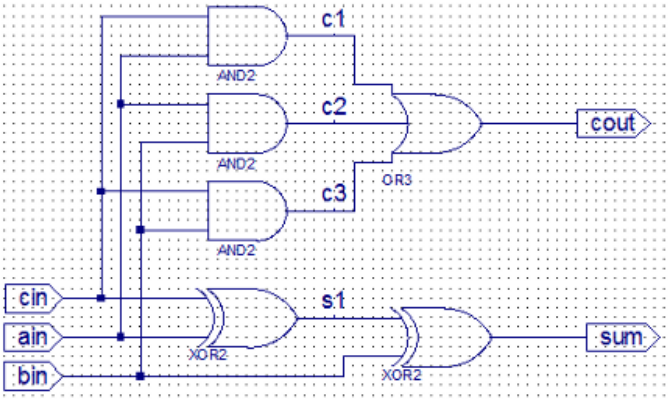

Figure 13: Full adder with names for internal signals

Add a Title Block

- To add a title block to the schematic it helps to first zoom-out fully to display the whole schematic and then follow the same procedure that you used to insert a gate. In the *General* category pick the *title* component and place the title block near the lower right corner of the schematic.
- Double click the title block icon and in the pop-up window enter the following relevant information and then click OK.

  `NameFieldText:` Your Name
  `TitleFieldText:` Full-Adder Circuit
- Take a look at the title block and verify the title and your name

Save and Check the Schematic

- Save your work, select `File >> Save All`
- Dock the schematic editor back into the project window, select `Window >> Dock`
- Under the Options pane or Process pane look for and click on the Design tab. You might have to click the left-arrow that is just to the right of the tabs to bring the Design tab into view.

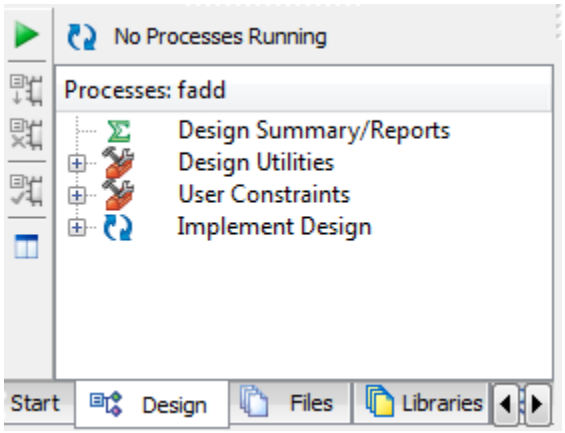

Figure 14: Design Process pane with several tabs

- Look in the Design Hierarchy pane, click the Implementation button and click to select fadd(fadd.sch) if it's not already selected. The three stacked boxes symbol indicates that fadd.sch is the top-level schematic.

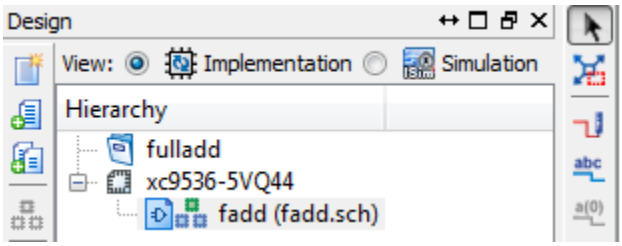

Figure 15: Design Hierarchy pane

- Back in the Design Process pane click the '+' icon that is before 'Design Utilities'
- Double click on 'Check Design Rules' and in a few moments a green check symbol should appear to indicate success. Otherwise, examine the report in the console and make corrections to your schematic.

Your schematic is now complete. Be sure to make a printout. If you want to save disk space or if the Xilinx tools start acting particularly odd, then at the top of the Project Manager window select the following then click OK.

```
Project >> Cleanup Project Files...
```

6 ISE/WebPACK Simulation

This section introduces the ISim simulator included with ISE. At this point you should have a project like that in section 5 which contains the description of a combinational logic circuit. Consideration is given later to state machines. Simulation involves a special file called a *test bench* or *test fixture*. Following version 10.1 32-bit, ISE no longer includes a graphical tool to generate test benches. Rather, an ISE tool generates a skeleton of a test bench so that a minimum of typing is required. Here we consider such a simple VHDL test bench. Xilinx provides application notes[**?**] that may be helpful.

Make a Testbench File

After a skeleton testbench is created you will modify its contents. Be sure that all your work is saved or you may end up with a fairly empty skeleton testbench file. If you are making a testbench for a flip-flop, state machine, or other registered logic then make the skeleton testbench file using the following but making appropriate substitutions. The testbench file name will be different as will the entity name and the signals. At the point where you assign the stimulus input go to section 7.

- With the ISE project manager open select the following:

  ```
  Project >> New Source...
  ```

- In the New Source Wizard pop-up window enter the following. The term `Default` refers to a given value which should not be changed. Click Next.

  ```
  Select source type:   VHDL Test Bench
  File name:            fulladd_tb1
  Location:             Default
  Add to project:       Checked
  ```

- The next window asks you to select a source file. Click to select 'fadd' then click Next

- The next window is a summary of your choices and values. Review the list then click 'Finish'.

- If the test bench file doesn't automatically open then open the file. Under the Options pane or process pane look for and click on the Design tab. You might have to click the left arrow just to the right to bring the Design tab into view.

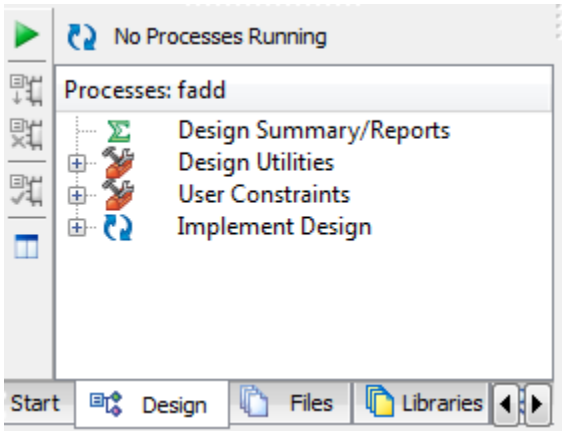

Figure 16: Process pane with tabs

- Look in the Hierarchy pane, click to select the 'Simulation' button, check that the 'Behavioral' choice is made and double click on 'fadd_fadd_sch_tb' so the corresponding test bench file appears in the text editor window.

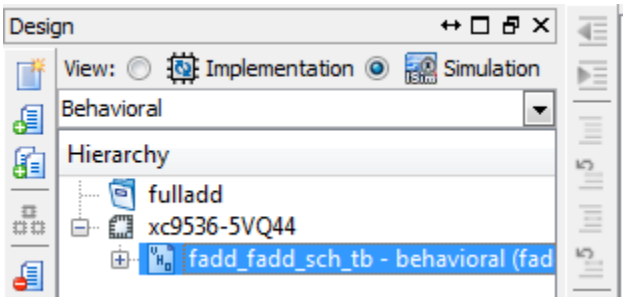

Figure 17: Hierarchy pane for full adder project behavioral simulation

A test bench is analogous to a laboratory test bench, which has signal generators, and test gear. We will use the test bench to describe the inputs and rather than using the test gear to verify the results, we will visually examine the simulator output waveform.

- At the top of the test bench file look for the following generated text . The ellipsis refer to extra content not shown here. The double hyphens -- are used to start a comment that the simulator will ignore. Following the comment block are LIBRARY and USE clauses that we will take as given.

```
-- Vhdl test bench created from schematic...
--
-- Notes:
...
-- stimulus for your design. --
```

Figure 18: Generated opening comment block

- Modify the comment block to be like that in Figure 19, to contain the project name, file name, author name, date, and a statement about what the file was written for. This information is akin to signing your work and will be expected on every test bench that you submit as part of an assignment.

```
-- fulladder - fadd_tb1.vhd - YourName - TheDate
-- This is an example test bench for the tutorial
--
```

Figure 19: Opening comment block for an assignment

- Look for the keyword 'ENTITY' as in the text below. This is where the inputs and outputs are declared in the description of a circuit. Note that a testbench file is referred to as being *enclosed* as it has no overall inputs or outputs.

```
ENTITY fadd_fadd_sch_tb IS
END fadd_fadd_sch_tb;
```

Figure 20: Generated opening comment block

- Next, look for the keyword 'ARCHITECTURE' and following that the keyword 'COMPONENT' starts the description of what the circuit you made looks like from the outside, so to speak. Next, all the signals that connect to the inputs and outputs of your circuit are declared. To make the typing later easier we add a declaration for a signal x. This part of the testbench will look like the following:

```
ARCHITECTURE behavioral OF fadd_fadd_sch_tb IS
  COMPONENT fadd
  PORT( cout  :  OUT   STD_LOGIC;
        bin   :  IN    STD_LOGIC;
        ain   :  IN    STD_LOGIC;
        cin   :  IN    STD_LOGIC;
        sum   :  OUT   STD_LOGIC);
  END COMPONENT;

  SIGNAL cout :  STD_LOGIC;
  SIGNAL bin  :  STD_LOGIC;
  SIGNAL ain  :  STD_LOGIC;
  SIGNAL cin  :  STD_LOGIC;
  SIGNAL sum  :  STD_LOGIC;
  SIGNAL x    :  STD_LOGIC_VECTOR(1 to 3); -- New line
BEGIN
```

Figure 21: Declaration part of test bench

- Following the keyword 'BEGIN' in Figure 21 is where an instance of your circuit is created. The acronym UUT means "Unit Under Test" and refers to your circuit. We will insert a new line of code to look like that in Figure 22.

```
BEGIN
  ain <= x(1); bin <= x(2); cin <= x(3); -- New line
  UUT: fadd PORT MAP(
     cout => cout,
     bin => bin,
     ain => ain,
     cin => cin,
     sum => sum
  );
```

Figure 22: Assign from X and instantiate the Unit Under Test

- The next part assigns the simulation inputs. Look for the keyword 'PROCESS' and its following keyword 'BEGIN'. Each of the following lines after the keword 'BEGIN' in Figure 23 describes one step in the simulation. Each line assigns an input and the wait keyword causes the simulation to proceed for a given amount of simulated time. Insert code so the process looks like the following. Note that double quotes are used to express strings of bits but single quotes are used for individual bits, such as '0' or '1'.

```
-- *** Test Bench - User Defined Section ***
   tb : PROCESS
   BEGIN
      x <= "000"; wait for 100 ns; -- The following
      x <= "001"; wait for 100 ns; -- are new text
      x <= "010"; wait for 100 ns; -- lines, insert
      x <= "011"; wait for 100 ns; -- and/or modify
      x <= "100"; wait for 100 ns; --
      x <= "101"; wait for 100 ns; --
      x <= "110"; wait for 100 ns; --
      x <= "111"; wait;            -- end of input
   END PROCESS; -- *** End Test Bench
END;
```

Figure 23: Testbench simulation steps

Check the Testbench

With the testbench created the next step is to check the file

- Look in the Design Hierarchy pane, click to select the Simulation button, check that hte 'Behavioral' choice is made and double click on 'fadd_fadd_sch_tb' so the corresponding test bench file appears in the text editor window. The Design Hierarchy pane should look like that in Figure 17. ISE provides at least two opportunities to use simulation to check the correctness of a circuit.

  Behavioral - A simulation performed using the VHDL code as-is. No device specific timing information is provided so the simulation looks ideal.

  Post-Route - This is the whole enchilada. This type contains timing information from the implementation and will be the most realistic.

- In the Design Process pane click the '+' symbol to the left of 'ISim Simulator' to expand the category.
- Right click on 'Behavioral Check Syntax' and select 'Run'

After a few moments a green check should appear in the pane, othwise scroll through the comments made in the console below to determine what happened during the check and then correct the testbench file.

Perform the Simulation

- Back in the Design Process pane right click on 'Simlate Behavioral Model' and select 'Process Properties...'. In the pop-up window change only the Simulation Run Time to 800 ns, click OK.
- Right click again on 'Simlate Behavioral Model' and select 'Run'. After a few moments the simulation window will appear. If a Windows Security Alert appears then click 'Allow access.'
- Once the simulation starts, by default it shows you only the last few moments of the simulation. Select:
  `View >> Zoom >> To Full View`
- You will sign your simulation results by inserting a new divider below the signals and typing in your name or nickname. In the Names column, right click and select 'New Divider' then type your name or nickname into the text field. You can drag the vertical dividers as necessary. Your simulation will look similar to the following. This is as far as you need to go to perform the tutorial as an assignment.

Figure 24: First simulation results for fadd

Note that this first simulation display only the top-most inputs and outputs in your circuit. For future assignments such as that involving state machines you will want to examine the actual state and with hierarchy you will want to see internal signals, so it will be necessary to look inside your circuit.

- The 'Instance and Processes' pane, to the left, is used to navigate the levels or hierarchy. Click the right-pointing triangle before 'fadd_fadd_sch_tb' then click on 'UUT'. At this point all the circuit internal signals will be visible in the 'Objects' pane.
- In the Objects pane either drag 'c1' to the waveform window or right-click on the signal and select 'Add to Wave Window'. Add the signals 'c2', 'c3', and 's1' to the waveform window as well.
- In the toolbar click 'Simulation' and note the simulation choices there are as well as the corresponding icons and hot-key combinations. Also note the simulation time to the right of the toolbar. Change the simualtion time from 1.00us to 800ns.

- Restart the simulation then use X-run to run the simulation only for the given simulation time. Your simulation will look like the following. Before closing ISim you can save the configuration for the next time that you start ISim.

```
File >> Save As
```

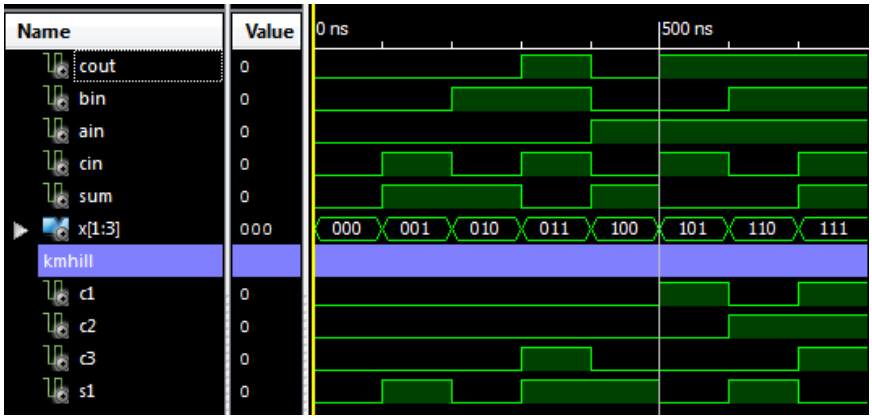

Figure 25: Simulation results for fadd showing internal signals

Be sure to capture, print to paper, or print-to-file your simulation results.

7 Test Bench For Sequential Logic

In modifying a test bench for use with a state machine it is necessary to provide a clock input as well as reset. The test bench for a D-type flip-flop demonstrates all the basic points of such a test bench. Consider the flip-flop description given in Figure 26

. That flip-flop updates state at a rising clock edge which means that the clock signal transitioned from *low* to *high*. The clear and set inputs are *asynchronous* and *positive logic*, which means without regard to the clock and that the high value is signficant. The asynchronous inputs override other behaviors of the flip-flop. The following list summarizes the inputs and ouputs:

- **D** = Synchronous input
- **C** = Asynchronous clear, causes the state to become *low*
- **S** = Asynchronous set, causes the state to become *high*
- **clk** = Clock input
- **Q** = Flip-flop state output

In the simulation in Figure 27 the flip-flop is initially in an unknown state. The set and clear inputs force the state to be 1 and 0, respectively. The flip-flop is also shown loading the value 1 and the value 0, each at a rising clock edge. As a behavioral simulation it contains no device timing information which causes the simulation to appears to have no propagation delay.

To prepare for the simulation first follow the steps above to make the test bench skeleton file DFlipFlop_tb.vhd and then make some changes. As always, start by changing the opening comment block in the code. Feel free to remove extraneous text that interferes with text that you are inserting. Figure 28 lists the signals declared in the test bench. Note that clk is changed to initialize to 1. Remove any remaining declarations for signals and constants. There is no need here for a signal X as in the tutorial.

```
-----------------------------------------
-- DFlipFlop.vhd - YourName - TheDate
-- Behavioral description of D-type FlipFlop
-----------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity DFlipFlop is
    Port ( D,C,S,clk : in  STD_LOGIC;
           Q : out  STD_LOGIC);
end DFlipFlop;
architecture Behavior of DFlipFlop is
  signal QX : STD_LOGIC;
begin
  process(C,S,clk)
  begin
    if C = '1' then
      QX <= '0';
    elsif S = '1' then
      QX <= '1';
    elsif clk'event and clk = '1' then
  QX <= D;
    end if;
  end process;
  Q <= QX;
end Behavior;
```

Figure 26: Description of D-type flip-flop with asynchronous set and clear

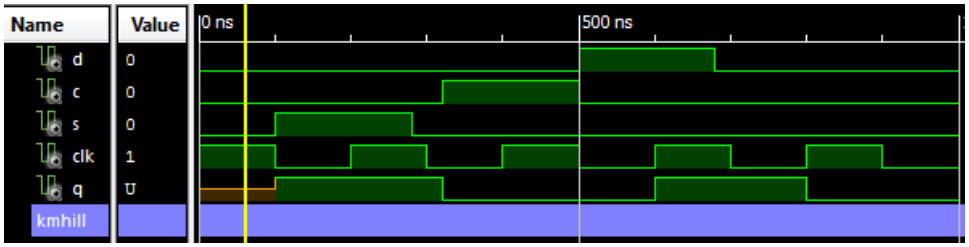

Figure 27: Flip-flop simulation result

```
--Inputs
signal D : std_logic := '0';
signal C : std_logic := '0';
signal S : std_logic := '0';
signal clk : std_logic := '1'; -- Change this line

--Outputs
signal Q : std_logic;
```

Figure 28: Signals declared in test bench

The input is assigned with the following code. If there is a process construct already associated with clk, then remove it. The following code will assign all the input signals to the flip-flop.

```
-- Make the clock signal, new line
clk <= not clk after 100 ns;

-- Stimulus process, insert new lines
stim_proc: process
begin
  wait for 100 ns;
  S <= '1'; wait for 180 ns;
  S <= '0'; wait for 40 ns;
  C <= '1'; wait for 180 ns;
  C <= '0'; D <= '1'; wait for 180 ns;
  D <= '0'; wait;
end process;
```

Figure 29: Stimulus given in test bench

8 CPLD Pins and Synthesis

This section introduces the steps to perform synthesis for a CPLD. At this point you should have a project like that in section 5 which contains the description of a logic circuit. In this part you will synthesize the full-adder for use with an XMOD module which uses a Xilinx XC9536XL CPLD though the assigned pins should also work with a CMOD module which uses a Coolrunner-II CPLD.

Initial Pin Assignment

Here we will make a so-called User Constraints File (.ucf) which is used to assign the input/output signal names to actual pins, which is necessary as otherwise each time the synthesis process is performed. the tools will arbitrarily make an assignment for you. In creating a new .ucf files it helps to use a graphical tool. Newer versions of ISE use a tool called PlanAhead for FPGAs, however for CPLDs we will use PACE.

- If the Design tab is not already selected then look under the Options pane for the Design tab and click it. You might have to click the left arrow (at lower right) to bring the Design tab into view.

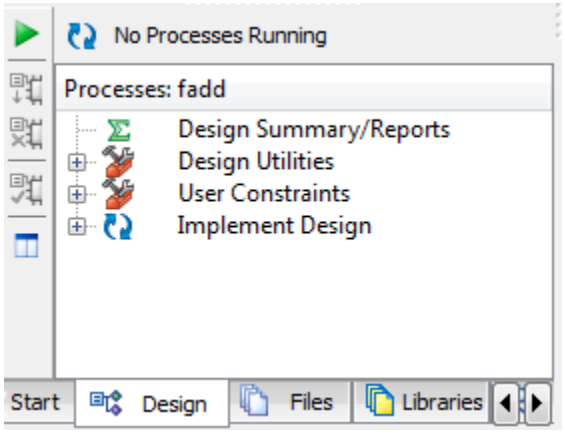

Figure 30: Design Processes pane

- Look in the Design Hierarchy pane, click the 'Implementation' button and then click so the fadd (fadd.sch) is selected.

- Back in the Design Processes pane click the '+' before User Constraints to expand the category
- Either double-click or right click on 'Floorplan IO - Pre-Synthesis' and select Run.
- If a message appears indicating that a .ucf does not exist yet so that a new file will be created, click OK or YES.

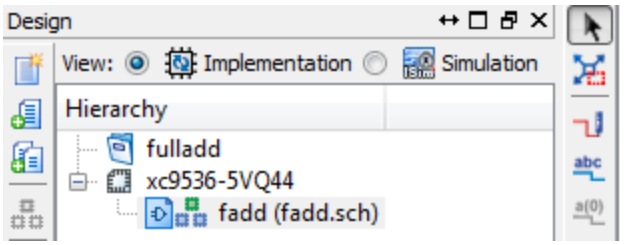

Figure 31: Design Hierarchy pane

The PACE graphical program is used to assign resouces to your design. In opening PACE, ISE performs an analysis step that takes a moment. If there are any issues in your schematic this step may fail in which case, go back to check the schematic. Once PACE opens, change the values in the LOC column to match the following.

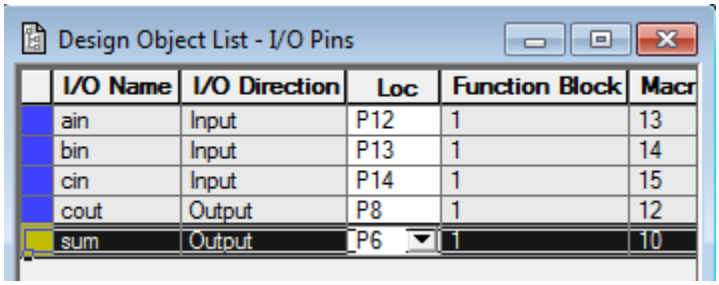

Figure 32: Making pin assignments with PACE

After entering the last LOC number, be sure to press the enter key and then save your work with `File >> Save`. In the pop-up bus delimiter choice window, select the default and then click OK. Finally, close PACE.

Editing the Constraints File

Once a .ucf file is created it may be more convenient to hand edit the file than use PACE, but in particular hand editing is how you insert commets and is how you will sign your work, so to speak. At a minimum be sure to insert your name, the date, and given that you may have different versions of a .ucf file include a description of what makes the file unique.

- Back in the Design Hierarchy pane click the 'Implementation' button and then click the '+' in front of the fadd (fadd.sch) symbol to expand the category then click to select 'fadd.ucf'
- In the Design Processes pane click the '+' in front of 'User Constraints' to expand the category and then either double click or right click on 'Edit Constraints (text)' and select Run.
- Add or replace text as necessary so that the file looks like the following. The '`#`' symbol starts a comment which ends with the line. Each line starting with the '`NET`' is used to assign a signal to a pin. The signal name is in double quotes, the pin name is in double quotes after '`LOC =`' and the line ends with a semicolon. The comments inserted at the end of each line tell us the corresponding module pin is.

A full list of CPLD pins for the XMOD is given with the XMOD CPLD documentation on the XMD web page[5] and is similar to the list for the CMOD, given in documentation from Digilent, Inc.[2].

Synthesizing the Design

Remember that an ISE design identifes the 'top-most' module that will be handled. Look back in the Design Hierarchy pane, three stacked boxes before 'fadd (fadd.sch) indicate that this is the top-most module. Otherwise right click the module symbol to make it your pick for the top-most module.

```
# fadd.ucf, Your Name, The Date
# Description of what makes the file unique
#PACE: Start of Constraints generated by PACE

#PACE: Start of PACE I/O Pin Assignments
NET "ain"  LOC = "P12"  ; # ModPin  1
NET "bin"  LOC = "P13"  ; # ModPin  2
NET "cin"  LOC = "P14"  ; # ModPin  3
NET "cout"  LOC = "P8"  ; # ModPin 40
NET "sum"  LOC = "P6"  ;  # ModPin 39

#PACE: Start of PACE Area Constraints

#PACE: Start of PACE Prohibit Constraints

#PACE: End of Constraints generated by PACE
```

Figure 33: User Constraints file (.ucf)

- If the Design tab is not already selected then under the Options pane look for the Design tab and click it. You might have to click the left arrow (at lower right) to bring the Design tab into view.
- Look in the Design Hierarchy pane, click the 'Implementation' button and then click so the fadd (fadd.sch) is selected.
- In the Process pane, double click on or right click on 'Implement Design' and select Run. After a few moments a green check mark indicates success, congratulations.
- A summary pane appears. If the synthesis failed or warnings appear then click on 'Errors/Warnings' to resolve the situation.
- Click on the 'Design Summary' tab and then under the 'Design Overview' click on 'CPLD Fitter Report.' Examine what resources were used. There should be 3 Input pins, 2 Output pins, 2 macrocells, and no mention of registers or flip-flops.

The next step to implementing a design is to configure an actual device, which is outlined in section 9

9 Configuring a CPLD by using JTAG

The JTAG programming cable includes necessary electronics to interface with a programmable logic device. A JTAG programming cable is a worthwhile investment that can be used with many different devices. Things to consider in making a choice of software and hardware include the following:

- PC interface type, USB or parallel port
- The first version of the programming software that supports your operating system and programming hardware
- Connector interface and protocol support

To keep this presentation simple we will consider only Xilinx iMPACT which is part of Xilinx ISE. Adept is also available from Digilent at no cost but is not described here in detail. For programmer hardware we consider products from Digilent, Inc. As outlined in section 2, this book was written for ISE version 14.7, which is known to directly support all current Digilent Inc programming cables. If you are using an older version of ISE be sure to refer to the choices table. Adept is alterntive software from Digilent Inc that you can use to configure Xilinx devices. The CPLD modules we use allow for 6-pin connectors. Xilinx ISE produces industry standard files which allows for additional options that are not considered here.

As a side note, the action of *programming* a device is used here synonymously with that of *configuring* a device. The ambiguity may arise as devices such as ROMs are used to store the executable code for a program and traditionally a ROM is configured by a device called a programmer. Given the system on a chip (SoC) concept the action of configuing a device can also be thought of as being inclusive of the action of programming a device.

- http://www.digilentinc.com/choosing.cfm

Connecting a Programmer Cable

As shown in Figure 34, to configure any CPLD, besides the programming cable it is necessary to provide power (+3.3V) and ground (GND). If you are using a parallel port interface it may not be obvious if the connection is correct until you actually try to configure a device. The first time a USB device is connected to a given PC it is necessary for the PC to install driver software which provides a communications link to the CPLD. Before the programming cable is recognized it may be necessary to connect the CPLD as discussed here. Once the drivers are installed, if you look in the Windows device manager the programmer cable will be listed among the USB devices.

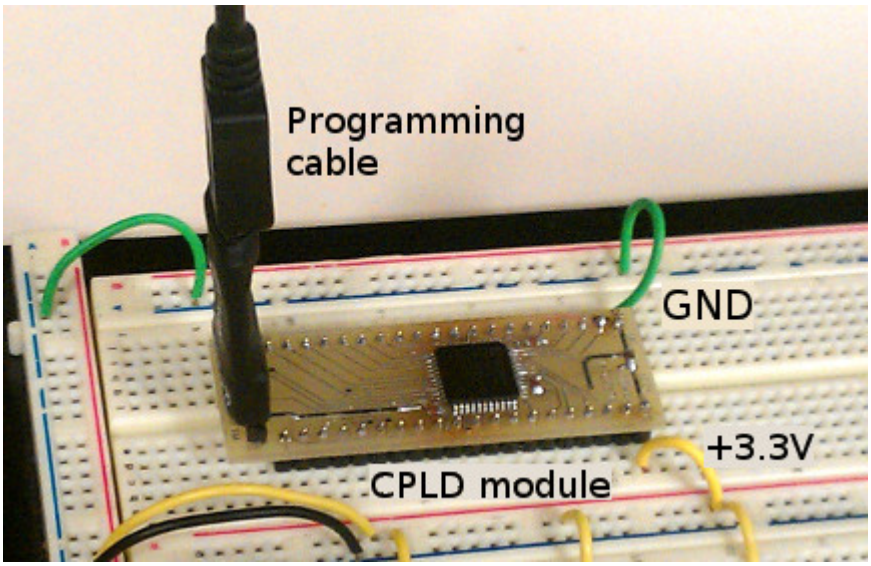

Figure 34: Configuring a CPLD

Using iMPACT to Configure a CPLD

At this point the Synthesize, and Translate, Fit, and Generate Programming File should all be able to run to completion without error so that a final a green check mark appears. In some cases executing Fit might produce a warning but it should still run to completion. The next step is to start iMPACT.

- In the ISE Project Navigator, to the lower left click the 'Design' tab, in the hierarchy pane click the 'Implemenation' choice and select 'SimpleGates', and in the process pane right click on 'Configure Target' and in the pop-up menu click on 'Run' or 'Rerun All'.
- A warning that 'No iMPACT project file exists' should appear, click 'OK'. After a few moments the ISE iMPACT window appears.
- To the upper left double click on 'Boundary Scan' so the larger pane just to the right changes shade. Move the mouse pointer to the larger pane to the right, right click and select 'Add Xilinx Device'. In the pop-up window navigate to the project folder, select the project .jed file, and click 'Open'.
- Graphics showing the CPLD JTAG loop appears in the larger pane. Click to select the CPLD symbol then right click and select 'Program'. In the Device 1 programming properties window click 'OK'. After a few moments of programming a blue message appears indicating that the CPLD is programmed.

You can not remove the JTAG and use the CPLD. Using Adept to configure a CPLD is a two step process. First use iMPACT to create an .svf file which Adept uses to configure the CPLD. For further details, refer to the help Xilinx software manual and Adept documentation.

10 Making and Using Hierarchy Symbols

As a first introduction to hierarchy consider the circuit in Figure 35 that adds a pair of two-bit values to produce a three-bit sum. In adding multi-bit values, each fadd serves to add the bits in one column of values, including the carry-in from the column to the right and producing a carry-out for the column to the left. The lower carry-in bit (cin) is connected to a ground (GND) symbol.

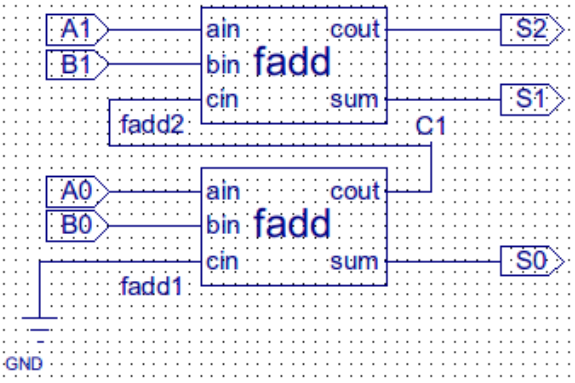

Figure 35: Adder for a pair of two-bit values

To produce the circuit we will start by making a schematic symbol that represents the full-adder (fadd), to use in making the adder circuit in Figure 35. To perform this section you will need skills discussed back in section 5. In particular you need to know how to create a new schematic and use the schematic capture tool.

Create a Symbol

To perform this part of the tutorial you should have a project open with a circuit description entered, tested, and saved. The description can be something other than a schematic, for example a VHDL file can be used. But as an example this section involves creating a fadd symbol to represent the full adder circuit described in sections 5 and 6.

1. In the ISE Design Hierarchy pane for the View select 'Implementation', click to select fadd(fadd.sch) and then in the Design Processes pane click on the '+' symbol before Design Utilities.

2. Double-click on 'Create Schematic Symbol' or right-click on it and select 'Run'. After a few moments a green check mark will appear, otherwise scroll through the comments listed in the console and then fix your design.

3. The fadd symbol that was made is quite arbitrary. We know from experience that drawing and later reading a schematic is greatly eased by simply editing each symbol to adjust the pin placement and font size of the signal names. In the project navigator window select:

   ```
   File >> Open...
   ```

   In the 'Open' pop-up window select 'fadd.sym' so the name appears in the 'Flile name' field, then click Open. The symbol editor will open the newly created fadd symbol.

4. Your symbol should look similar to that in Figure 36. Each dark-red square is called a pin and represents where a connection is made to the symbol. There is a Signal name associated with each pin and a Symbol name for the symbol. The rest is incidental artwork.

5. Adjust the font size of the signal names and then move the pins. One-at-a-time double click on each signal name, in the pop-up window click on the Figures category, then increase the font size to a value like 36, then click OK.

6. One-at-a-time use the mouse to draw a box around each pin and its associated signal name so that both appear selected. Next use the mouse to move the selected part, then click elsewhere. In this case in shuffling the pins were moved closer together. After the shuffling the box was selected and made

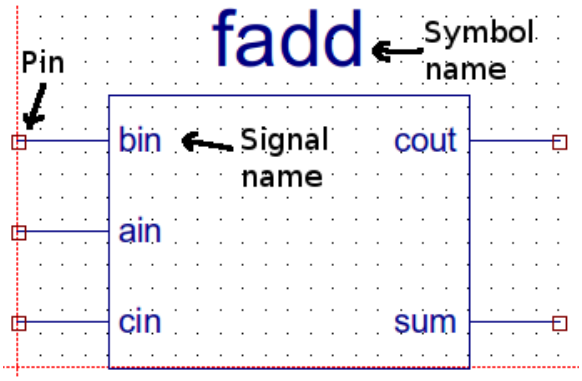

Figure 36: Newly created fadd symbol

shorter and the symbol name was moved inside the box. The symbol was made to look like that in Figure 37.

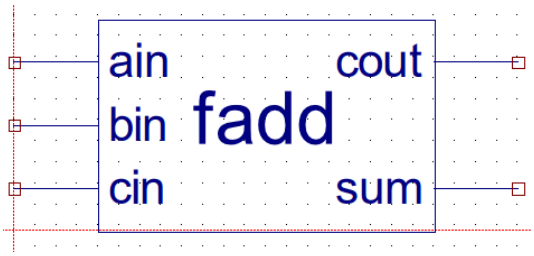

Figure 37: Modified fadd symbol

7. At this point save your work and close the symbol editor

   `File >> Save All`

   `File >> Close`

Using Symbols Without a Bus

Create a new schematic to construct the circuit shown in Figure 35 and save the new schematic as TwoBit.sch. In the schematic capture tool to find the new symbol look in the schematic editor Symbol Category that refers to your project.

Once inserted into the schematic double-click the new symbol and assign a recognizable name for the part designator into the InstName field, click the 'Visible' box, then click OK. You will be able to move the part designator as necessary. To change the font size, double click on the part designator as you would for an internal signal. Once the schematic is drawn, be sure to sign your work by inserting a title box and also testing the correctness of your new circuit.

To write the test bench you can define two-bit a signal for A and another for B, in a manner similar to how you defined the three-bit x signal. In Figure 38(a) note that for a descending sequence the keyword `downto` is used as the keyword `to` is for ascending sequences. The simulation values for A and B are similar and shown in part (b). The user defined section containing some example input is shown in part (c).

The corresponding simulation is shown in Figure 39. Note that while a three-bit value was not mentioned in the test-bench you can group individual bits into a so-called *virtual bus*. Hold down the control key as you select signals so they stay selected, then in the name colomn right click and select 'New Virtual Bus'. To change the new signal namec, click on the name and type a new name in.

```
SIGNAL A,B   :  STD_LOGIC_VECTOR(1 downto 0);    -- New text line
```

(a) New declaration statement for A and B

```
A1 <= A(1); B1 <= B(1); A0 <= A(0); B0 <= B(0); -- New text line
```

(b) New assignments for A and B bits

```
-- *** Test Bench - User Defined Section ***
  tb : PROCESS
  BEGIN
     A <= "00"; B <= "01"; wait for 100 ns; -- New text lines
     A <= "01"; B <= "01"; wait for 100 ns;
     A <= "11"; B <= "01"; wait for 100 ns;
     A <= "10"; B <= "11"; wait;            -- end of input
  END PROCESS; -- *** End Test Bench - User Defined Section ***
```

(c) Example user input section

Figure 38: Outline of changes to test bench for TwoBit.sch

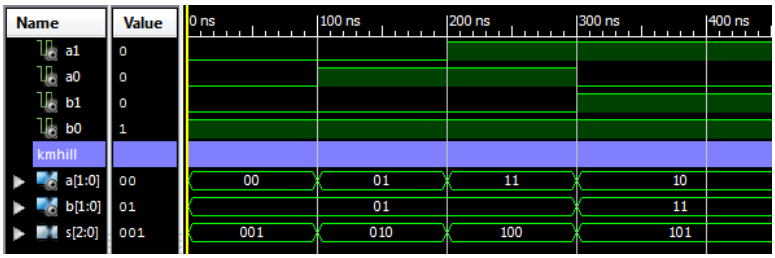

Figure 39: Simulation for TwoBit circuit

Using Symbols With a Bus

What we call a bus actually refers to a bundle of wires that are bundled together. The advantage of using bus signals is the same as why we bundle wires together. It's easier to handle a bundle of like wires that it is to deal with the individual wires. With ISE each wire in a bus has the same base signal name as the bus, along with an index number, so that S(2:0) contains S2, S1, and S0.

To place a bus like that in Figure 40, start by placing a wire some distance from any component. You can click the Add Wire icon as before, or select Add => Wire, then click to start the wire, click to insert a bend, and then double click to end the wire. Next, click the add I/O marker icon or select Add => I/O marker, and look just to the left to the Add I/O Marker Options pane and click "Add an input marker." Next, point and click on the wire end to place the icon. Finally, double click the icon and in the pop-up Object Properties window make any necessary adjustments. With each change that you make, click Apply.

- The port polarity is Input
- Increase the font size to a value such as 36
- Under the nets category change the name to A(1:0)

With the changes made to the icon, click OK. Take a moment to examine the new bus. The bus will appear thicker than regular signal wires, the icon should have the bent end connected to the bus, and the font size should be readable. Note that the range of bits (1:0) is what identifies the wire as a bus, and until then you cannot assign bus taps.

Next, as shown in Figure 41, attach bus taps and wires to attach the inputs. To start, either click the Add Bus Tap icon, or click Add => Bus Icon. In moving the cursor into the schematic, the cursor changes

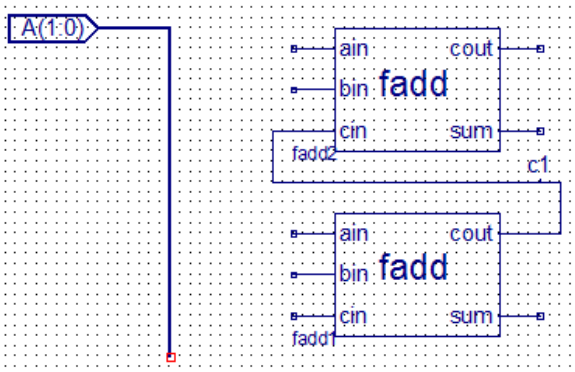

Figure 40: First bus in constructing adder for two-bit values

to a bus tap icon. Just to the left in the Add Bus Tap Options window you can change the orientation of the bus tap. Place a bus tap on the bus, just to the left of each fadd pin labeled ain. Next connect a wire from each bus tap to the corresponding input.

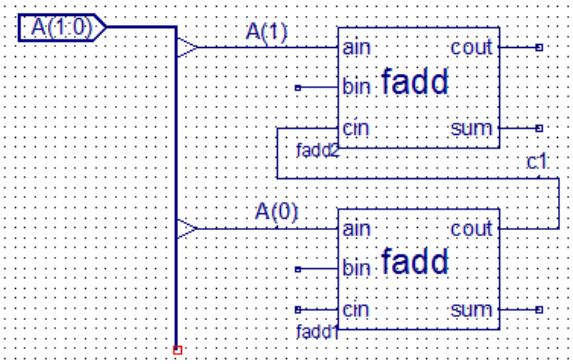

Figure 41: First bus taps in constructiong adder for two-bit values

Once a wire is connected from each bus tap to device pin, identify each tap wire. Either click the Add Net Name icon or select Add => Net Name. Just to the left in the Add Net Name Options window, in the Name field enter A(0) and move the cursor to the lower bus tap wire and click to attach the label. Double click on the label and increase the font size as we usually do. Repeat the above steps to attach A(1) to the upper bus tap wire. In the Add Net Name Options, clicking the right arrow will increase the bus index from 0 to 1.

The final circuit is shown in Figure 42. Connect a ground symbol to the bottom cin pin, as before. Repeat the above steps to create the bus for B(1:0). To the right create the bus S(2:0). Note that for S(2:0) the bent end of the I/O marker is to the right, to signify that the direction is output.

Let's make some observations. To connect a device with a bus pin, a bust tap can be used to connect several wires in a so-called sub-bus. Assigning S(1:0) causes the attached tap wire to respresent a branch from the larger bus. Next, the tap wires must be labeled with the matching bus base name and the index values must be within that of the bus.

To write the test bench there is no need to define additional bus signals. In creating a test bench file the ISE software will generate a skeleton test bench file that has bus signal inputs and outputs. In reviewing the test bench created for this project there should not be any references to clock signals. Figure 43 shows example stimulus that you can use to provide input and Figure 44. In the simulation window, to the left you can click on a bus signal arrow to reveal the individual signals.

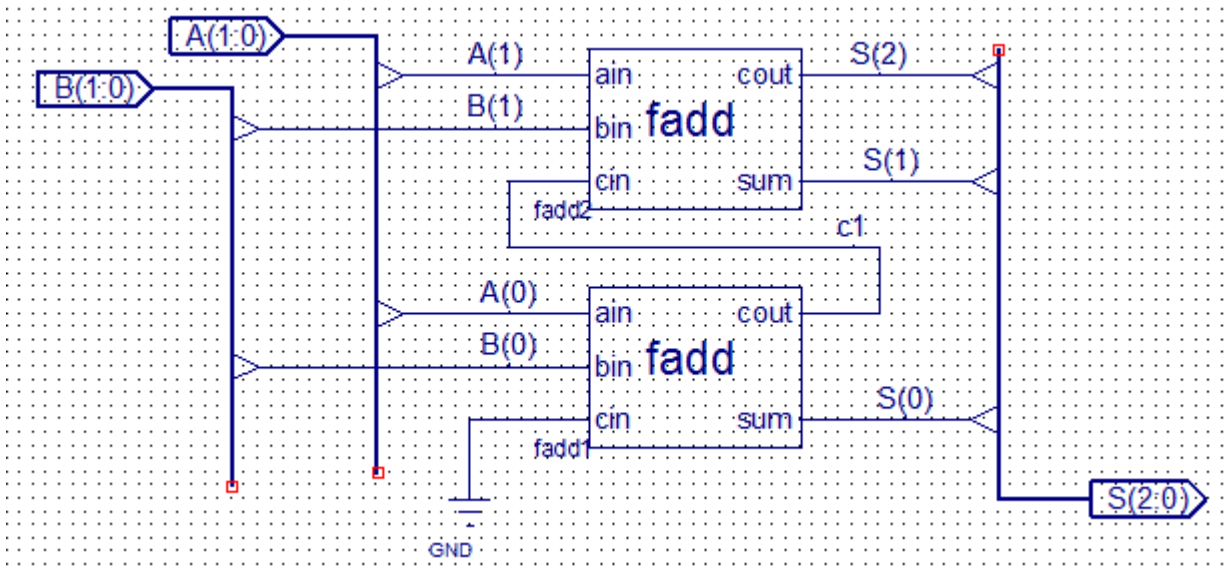

Figure 42: Adder for two-bit values

```
-- *** Test Bench - User Defined Section ***
  tb : PROCESS
  BEGIN
    A <= "01"; B <= "00"; wait for 100 ns;
    A <= "01"; B <= "01"; wait for 100 ns;
    A <= "10"; B <= "01"; wait for 100 ns;
    A <= "10"; B <= "11";
    WAIT; -- will wait forever
  END PROCESS;
```

Figure 43: Example simulation process for test bench

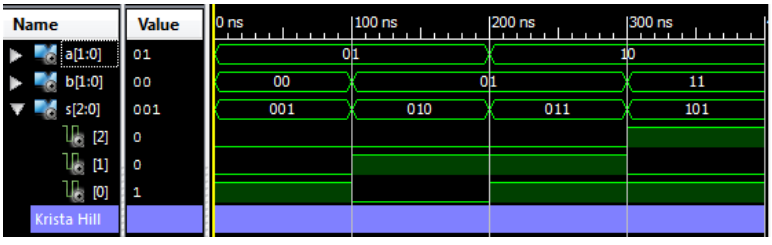

Figure 44: Example simulation of adder for two bit values

11 References

For the full list of pins used with the XMOD CPLD module, refer to the CPLD documentation posted on the XMOD web page[5].

References

[1] P. Ashenden, The Student's Guide to VHDL, second edition (Systems on Silicon), copyright 2008 by Elsevier, Inc.

[2] Digilent Inc., http://www.digilentinc.com/

[3] Xilinx, "ISE Design Suite 14: Release Notes, Installation, and Licensing," UG631 (v14.7) October 2, 2013 http://www.xilinx.com/

[4] ISE Tutorials page, http://uhaweb.hartford.edu/kmhill/suppnotes/isetut/index.htm

[5] XMOD Project, http://uhaweb.hartford.edu/kmhill/projects/xmod/