

# Atlys BSB Support Files for AXI-based EDK 13\_2..14\_2 Designs



1300 NE Henley Court, Suite 3  
Pullman, WA 99163  
(509) 334 6306 Voice | (509) 334  
6300 Fax

Revision: September 13, 2012

## Overview

These files will integrate Base System Builder (BSB) support for the Atlys Spartan-6 field programmable gate array (FPGA) Development Board with tools from the Xilinx Embedded Development Kit (EDK). These support files include board definition files and Digilent custom core support files that will help customers create AXI-based MicroBlaze embedded designs in the BSB.

Operators can use these files to enable the BSB to create Platform Studio projects initialized with properly configured cores for controlling on-board peripherals. This reference describes the procedures and important information for:

- I. Utilizing the BSB Support Files
- II. Creating a Base System using BSB Wizard
- III. Making Connections for the Custom Cores
- IV. Using the Digilent HDMI controller

## I. Utilizing the BSB Support Files

Before beginning to use the BSB Support Files, Digilent recommends that customers first uninstall any older versions of the AXI IPCore support files and download the support files for the most current version of the Xilinx EDK labeled 1\_33.

Users that employ Digilent products may already have the most current version of the Digilent AXI IPCore Support Files. These files have to be installed only once per EDK version and are valid for all applicable Digilent boards. For instance, if customers have installed another Digilent board (e.g. the Nexys3) the Digilent AXI IPCore Support Files will provide the current version of EDK to support all other Digilent boards.

Users must follow steps 1-1 through 1-6 to use supported Xilinx BSB peripherals.

- 1-1. Install the Digilent AXI IPCore Support Files by running the batch file “..\Digilent\_AXI\_IPCore\_Support\_v\_1\_33\ inst\_uninst.bat” and then following the on-screen instructions.
- 1-2. Begin by creating the Base System using the Base System Builder Wizard. See section II “[Creating a Base System using BSB Wizard](#)” for more details. *Note: Table one outlines the peripherals Xilinx BSB currently supports.* To use the custom core supported peripherals go to “System Assembly View” and follow steps 1-3 through 1-6.
- 1-3. Make the AXI clock connections for each custom core. For details on making AXI clock connections refer to the section 3a “[Connecting AXI Clock Signals to the custom cores.](#)”

- 1-4. Connect the external ports for each custom core. Refer to section 3b [“Making external port connections for the custom cores”](#) for more details
- 1-5. For custom cores having I/O signals, check and correct the I/O signal names in order to match them to the signal names in the .ucf file. See section 3c [“Checking I/O signal naming for the \\_pin suffix”](#) for more details.
- 1-6. Users have the option to connect each core having interrupt output signals to the interrupt controller. For more information on this see the sections 2a [“Configuring and removing peripherals”](#) and then 3d [“Connecting internal and interrupt signals for custom cores”](#).

Table two outlines the current peripherals custom cores support. By default, users select custom cores and can remove any core in the BSB Wizard. Go to the “Select and Configure Peripherals” window in the BSB Wizard to select or remove cores. For details, see section 2a [“Configuring and removing peripherals”](#)

Users can generate the bitstream and export their project to the SDK after completing steps 1-1 to 1-6.

Table 1  
Peripherals Supported by Xilinx Cores

Peripheral	Supported Interface	Core name(s)	Notes
128MB DDR (cached)	AXI4	axi_s6_ddrx	--
8 User Switches	AXI4-Lite	axi_gpio	--
5 User Push Buttons	AXI4-Lite	axi_gpio	--
8 LED outputs	AXI4-Lite	axi_gpio	--
UART	AXI4-Lite	axi_uartlite/ axi_uart16550	--
16-MB Quad-SPI PCM	AXI4-Lite	axi_quad_spi	There is an alternative custom core named “Quad-SPI controller”. Users can find this core in the EDK under “Project Peripheral Repository 0/Digilent.” This core allows users to change the SPI mode to single, dual, or quad in the runtime
10/100/1000 Mbps PHY	AXI4-Lite	axi_ethernet	Requires license; exclusive to axi_ethernetlite
10/100 Mbps PHY	AXI4-Lite	axi_ethernetlite	Exclusive to axi_ethernet

Table 2  
Peripherals Supported by Custom Cores

Peripheral	Supported Interface	Core name(s)	Notes
AC-97	AXI4-Lite	d_ac97_axi	Custom core
HDMI	AXI4-Lite, AXI4-Stream	axi_hdmi	Custom core; Not supported in BSB; Supported only with EDK v13.4 and above
USB-EPP	AXI4-Lite	d_usb_epp_dstm_axi	Custom core

*Note: For additional information on using the cores above, please refer to their PDF datasheets found at [www.xilinx.com](http://www.xilinx.com) under the link "Product Support and Documentation."*

## II. Creating a Base System using BSB Wizard

*Note: The example screenshots in this section are not accurate for EDK versions older than 14.2. However, the data users enter into the text boxes and combo boxes is very similar.*

- 2-1. Start Xilinx Platform Studio and create a new project using the Base System Builder by selecting "Create New Project Using Base System Builder." Select "Create a New Project" and the "Create New XPS Project Using BSB Wizard" window should open.
- 2-2. Select the "AXI System" (#1). Click on the BROWSE button beside the "Project File" and select a folder location for the *system.xmp* project file (#2). (See figure one for a screen shot of this window)

Digilent Inc. does **NOT** recommended users assign a path for the project folder that contains spaces (e.g. "My Documents") because these spaces may affect the functionality of the Linux-based EDK and SDK tools.

Digilent Inc. also recommends users assign an empty folder to store the entire EDK hardware project. Dedicating a folder to storage makes it easier to copy and archive the project later. Users may, at their discretion, utilize the same folder for their project workspace. The default project description file name is "system.xmp." Users also have the option to give the project description file any name, as long as the file extension remains and there are no spaces.

- 2-3. Click on the BROWSE button beside the "Set Project Peripheral Repository Search Path" box and browse to the path containing the ".lib" subfolder from the BSB Support Files folder. Select the ".lib" subfolder and click OK.

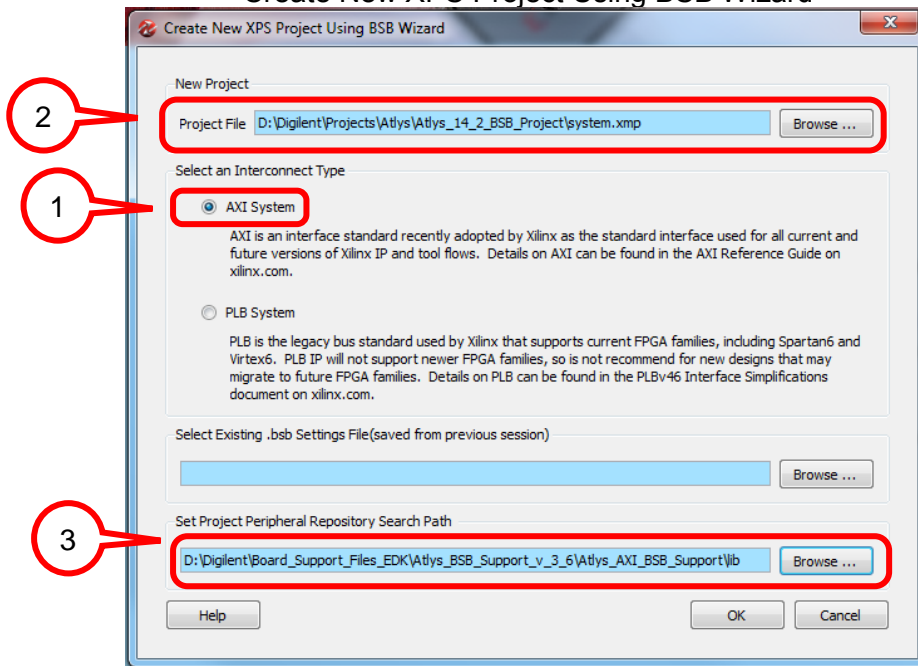
*Note: Users may assign fields in any order until they finalize them by clicking the OK button in the "Create New XPS Project Using BSB Wizard" window.*

- 2-4. After clicking OK the next window will allow the user to select the default Digilent Spartan-6 Atlys as the development board. (See figure two) Upon selecting the Digilent Spartan-6 Atlys ensure the fields are correct and click "NEXT."

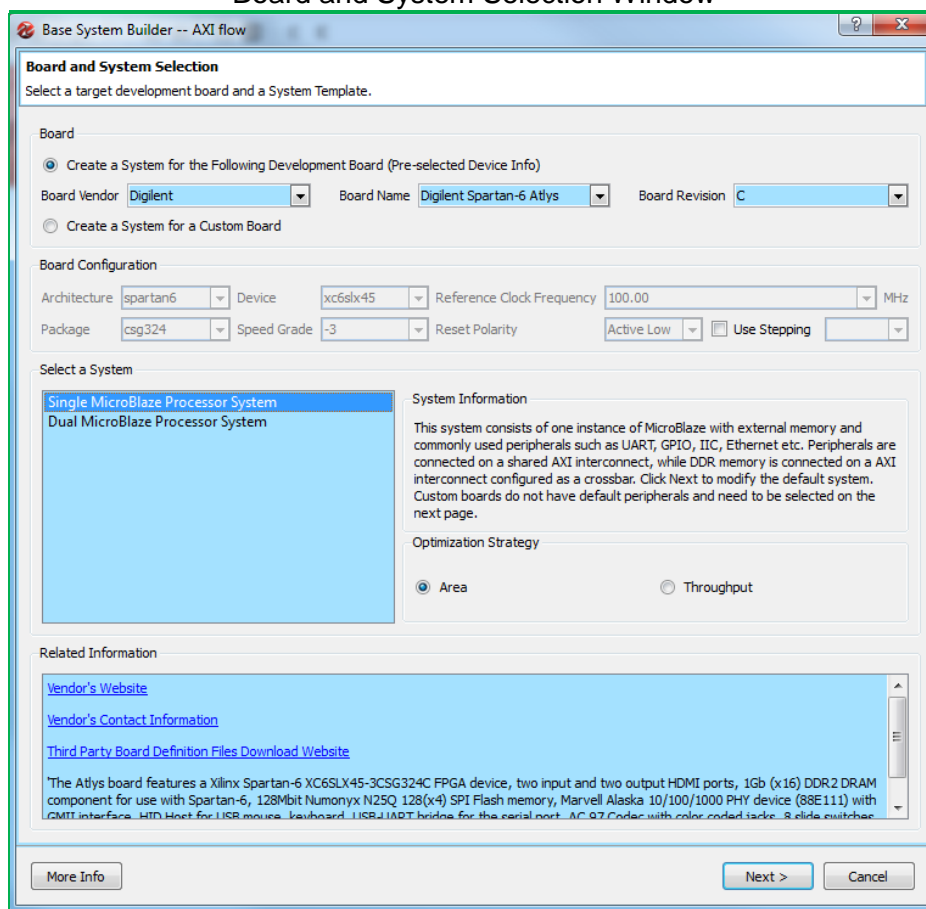
The next window is the "Processor, Cache and Peripheral Configuration" window. Users can select the peripherals from this window, configure some of them and remove other unwanted

peripherals. See section 2a “[Configuring and removing peripherals](#)” for information on removing peripherals.

**Figure 1**  
 “Create New XPS Project Using BSB Wizard”



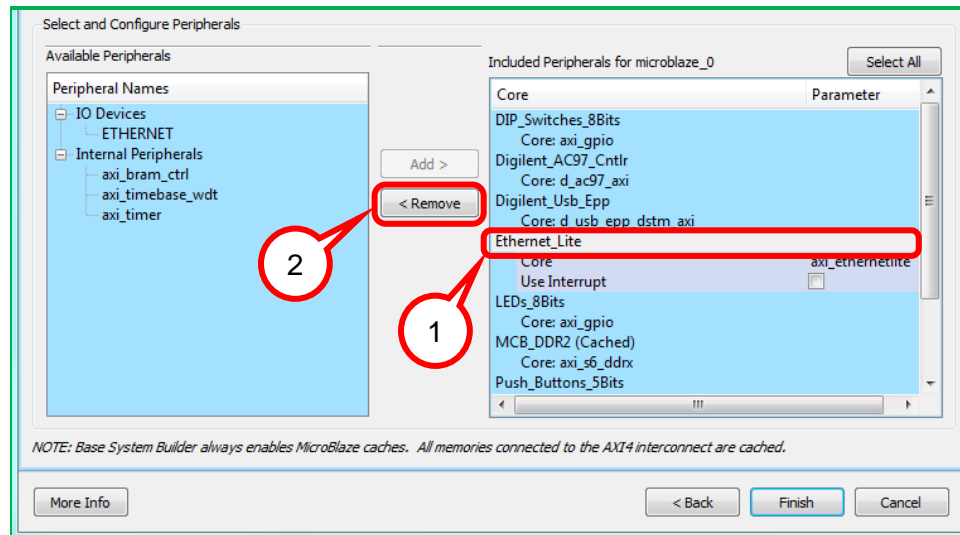
**Figure 2**  
 Board and System Selection Window



## 2a. Configuring and removing peripherals

Users can remove a peripheral by clicking on the specific peripheral (#1) and then selecting “Remove” (#2). (See figure three for details)

**Figure 3**  
Removing a Peripheral in BSB

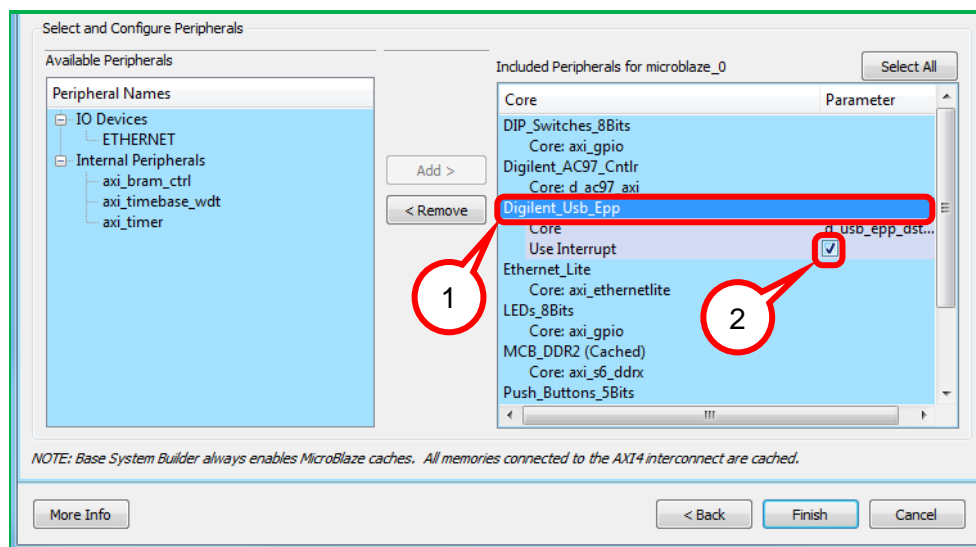


When users select a peripheral the system shows those parameters, if any, that they can configure in BSB. For example, figure four shows the selection of the interrupt option for the Digilent USB-EPP interface. Selecting the “Use Interrupt” option for any peripheral with the option will make BSB add an interrupt controller to the system. If no other peripheral has this option, users must follow the steps in 2a to use interrupts with the Digilent USB-EPP interface.

Because the Digilent USB-EPP interface is a custom core, users will have to manually connect the interrupt signal to the interrupt controller. Refer to section 3a [“Connecting internal and interrupt signals”](#) for additional information.

Users must click FINISH after they have selected and configured their desired cores. Clicking FINISH generates the Base System and the Xilinx Platform Studio (XPS) brings up the “System Assembly View” to show the Bus Connections for the available peripherals.

**Figure 4**  
 Selecting the interrupt option for the Digilent USB-EPP interface

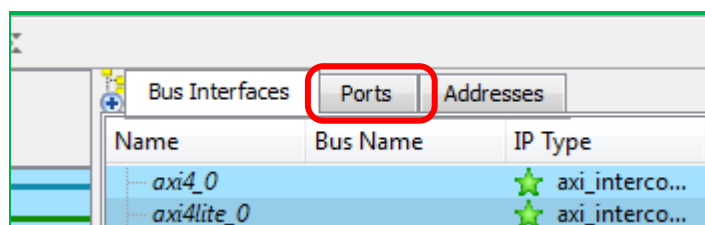


### III. Making Connections for the Custom Cores

#### 3a. Connecting AXI Clock Signals to the custom cores

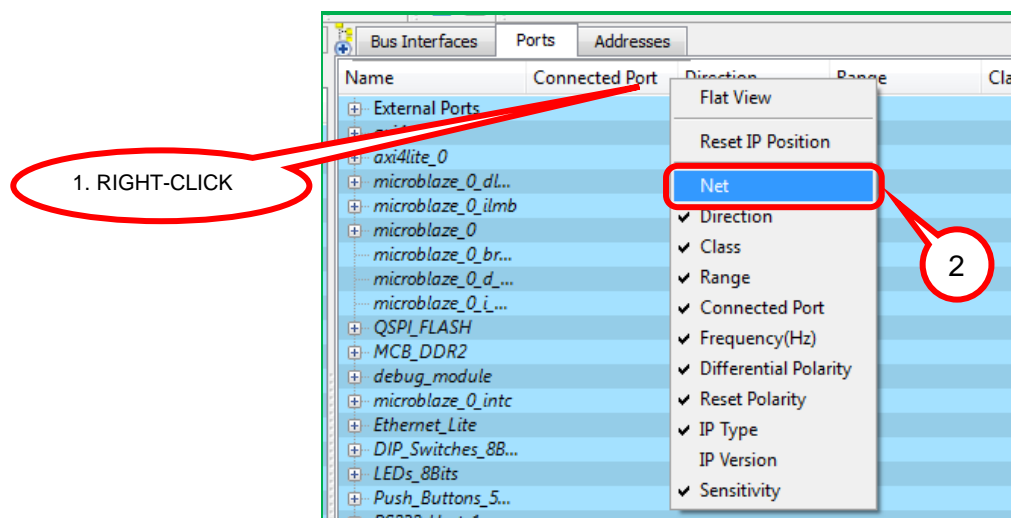
3a-1 In System Assembly View, select the “Ports” tab. (See figure five)

**Figure 5**  
 Selecting the Ports View



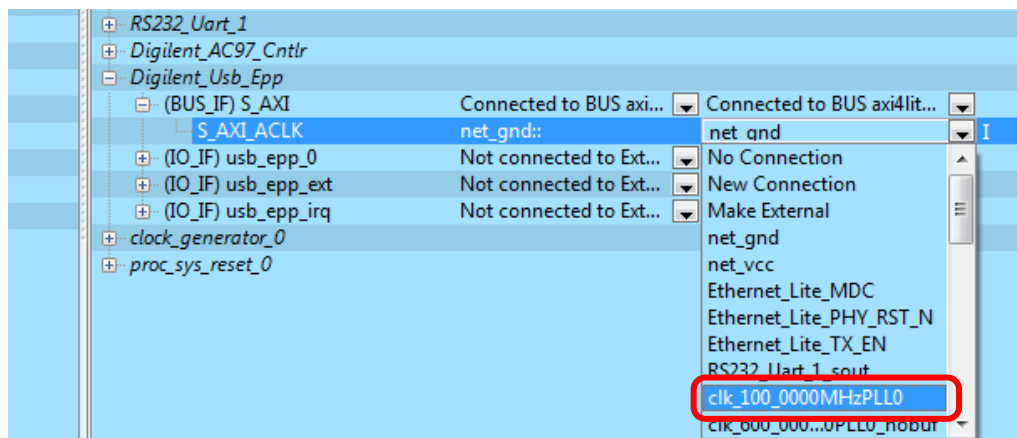
3a-2 If the Ports view does not show the “Net” column, users can reveal the column by right-clicking on the “Ports View” header and selecting “Net” (#1 and #2). Older versions of the EDK should show the “Net” column by default. Adjust the width of the columns to see the full signal names. (See figure six)

**Figure 6**  
Showing the Net column



- 3a-3 Expand “Digilent\_Usb\_Epp” and then the “(BUS\_IF) S\_AXI.” Select the connection signal “clk\_100\_0000MHzPLL0” for the “S\_AXI\_ACLK” in the “Net” column. (See figure seven)

**Figure 7**  
Connecting the AXI clock signal for a custom core



*Note: The signal “clk\_100\_0000MHzPLL0” is both the processor clock and the AXI clock as long as the “Processor, Cache and Peripheral Configuration” window (its lower part is shown in figure three) clock frequency did not change from 100 MHz. Run “clk\_frequencyMHzPLLnumber” to name the system clock signal.*

- 3a-4 Repeat Step 3 above for “Digilent\_AC97\_Cntlr” if this core is also present in the system.

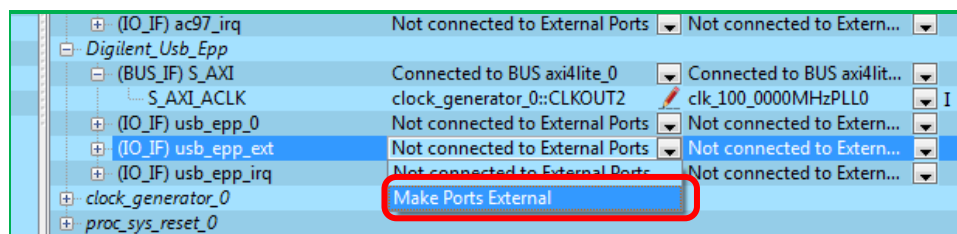
### 3b. Making external port connections for the custom cores

- 3b-1. For “Digilent\_Usb\_Epp” select the signal group – IO Interface “(IO\_IF) usb\_epp\_ext” and expand the combo box in the Connected Ports column. After expanding the



combo box select the “Make Ports External.” (See figure eight). This process should connect all of the core ports to external FPGA ports.

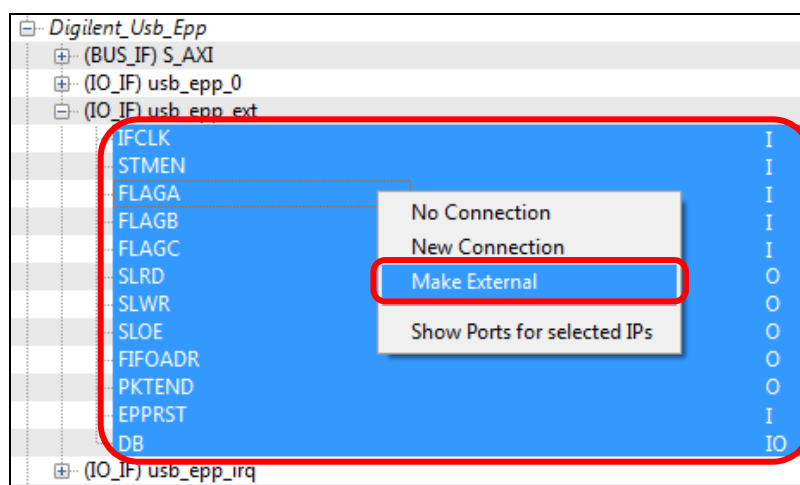
**Figure 8**  
Making an external connection



3b-2. Repeat the step above for “Digilent\_AC97\_Cntlr” group “(IO\_IF) ac97\_ext,” if this core is also present in the system.

*Note: Older versions of EDK might not have the “Connected Port” column available. If the “Connected Port” is unavailable expand the corresponding signal group and select all of the signals while holding the CTRL or SHIFT keys down. Then right-click on the selected signals and choose “Make External.” (See figure nine)*

**Figure 9**  
Making external connections in older versions of EDK



### 3c. Checking I/O signal naming for the *\_pin* suffix

When operators make an I/O signal external, some versions of EDK add the suffix “\_pin” to the signal name. However, in the BSB generated .ucf file the signal names might be without the “\_pin” suffix. Users must ensure the external signal names in both files match. For the Atlys board, the DB signal of the “Digilent\_Usb\_Epp” peripheral might also add the “\_pin” suffix.

Do the following steps 3c-1 through 3c-3 to check for I/O signal naming.

3c-1. Find Ports View, scroll up, and expand “External Ports.” Check for the ports “Digilent\_Usb\_Epp\_DB” or “Digilent\_Usb\_Epp\_DB\_pin.” (See figure ten)

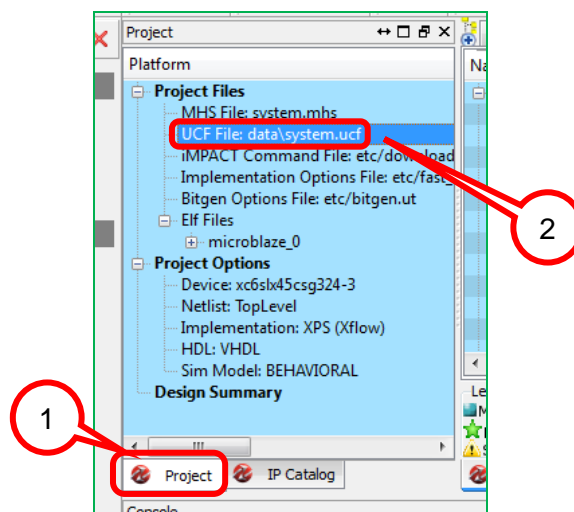


Figure 10  
Viewing external port names

Name	Connected Port	N
External Ports		
DIP_Switches_8Bits_TRL_I	DIP_Switches_8Bits::[gpio_0]::GPIO	DI
Digilent_AC97_Cntlr_BITCLK_pin	Digilent_AC97_Cntlr::[ac97_ext]::...	Di
Digilent_AC97_Cntlr_RESET_N_pin	Digilent_AC97_Cntlr::[ac97_ext]::...	Di
Digilent_AC97_Cntlr_SDATA_IN_pin	Digilent_AC97_Cntlr::[ac97_ext]::...	Di
Digilent_AC97_Cntlr_SDATA_OUT_pin	Digilent_AC97_Cntlr::[ac97_ext]::...	Di
Digilent_AC97_Cntlr_SYNC_pin	Digilent_AC97_Cntlr::[ac97_ext]::...	Di
Digilent_Usb_Epp_DB_pin	Digilent_Usb_Epp::[usb_epp_ext]::...	Di
Digilent_Usb_Epp_EPPRST_pin	Digilent_Usb_Epp::[usb_epp_ext]::...	Di

- 3c-2. Expand the “Project” Files (#1) and double click on the .ucf file “data\system.ucf” (#2) to open it. (See figure eleven)

Figure 11  
Opening the .ucf file



- 3c-3. Locate the “NET Digilent\_Usb\_Epp\_DB[]” lines in the .ucf file. (See figure twelve)  
*Note: The “Net” names do not contain the \_pin suffix.*

There are two ways to add the “\_pin” suffix to the “Net” files. One method is to add the “\_pin” suffix in the .ucf file after “\_DB.” For example, Digilent\_Usb\_Epp\_DB[0] becomes Digilent\_Usb\_Epp\_DB\_pin[0]. The other method of removing the “\_pin” is to rename the .ucf files in Port View. It is easier to remove the “\_pin” suffix in Port View.

*Note: Selecting an external port name in Port View will automatically edit the port name.*

Figure 12  
Viewing .ucf file constraints

```

15 NET Digilent_AC97_Cntlr_SDATA_OUT_pin LOC = "N16" | IOSTANDARD = "
16 NET Digilent_AC97_Cntlr_SYNC_pin LOC = "U17" | IOSTANDARD = "LVCMO
17 NET Digilent_Usb_Epp_DB[0] LOC = "A2" | IOSTANDARD = "LVCMOS33";
18 NET Digilent_Usb_Epp_DB[1] LOC = "D6" | IOSTANDARD = "LVCMOS33";
19 NET Digilent_Usb_Epp_DB[2] LOC = "C6" | IOSTANDARD = "LVCMOS33";
20 NET Digilent_Usb_Epp_DB[3] LOC = "B3" | IOSTANDARD = "LVCMOS33";
21 NET Digilent_Usb_Epp_DB[4] LOC = "A3" | IOSTANDARD = "LVCMOS33";
22 NET Digilent_Usb_Epp_DB[5] LOC = "B4" | IOSTANDARD = "LVCMOS33";
23 NET Digilent_Usb_Epp_DB[6] LOC = "A4" | IOSTANDARD = "LVCMOS33";
24 NET Digilent_Usb_Epp_DB[7] LOC = "C5" | IOSTANDARD = "LVCMOS33";
25 NET Digilent_Usb_Epp_EPPRST_pin LOC = "A5" | IOSTANDARD = "LVCMOS3
  
```

### 3d. Connecting internal and interrupt signals for custom cores

Extensible Provisioning Protocol (EPP) requests for the Digilent USB-EPP interface come from the USB port. Because EPP is time-out based, the processor must answer in at least 100 milliseconds or the system application will signal a timeout. Digilent recommends that users handle EPP requests with an interrupt service routine, instead of continuously polling the interface status.

The demonstration applications include examples for using the USB-EPP interface in both polling and interrupt mode.

Users must connect the interrupt request signal for the Digilent USB-EPP to either an interrupt controller or the Microblaze processor interrupt input to use interrupt service routines.

If users select the “Use Interrupt” option for any core in BSB, then the BSB will add an interrupt controller to the system. Otherwise, operators must manually add and connect interrupt controllers to the system.

Two steps must be taken to connect the interrupt output of a custom core to the interrupt controller. First, users must make a new connection for the custom core interrupt output, and then give a signal name for this connection. Then add the new signal to the interrupt controller’s “Intr” pin input.

Any version of Xilinx EDK newer than version 13.4 forbids making new connections for interrupt outputs in Ports View. Operators must make new connections for interrupt outputs in the Microprocessor Hardware Specification (MHS) file. See steps 3d-1 through 3d-5 for further information on making new connections for interrupt outputs.

#### Making new connections in the MHS file:

- 3d-1. Double-click on “system.mhs.” Users can find the “system.mhs” file above the “system.ucf” file. (See figure eleven)
- 3d-2. After Double-clicking on “system.mhs” locate the line “BEGIN d\_usb\_epp\_dstm\_axi” and insert the line “PORT IRQ\_EPP = Digilent\_Usb\_Epp\_DB\_IRQ\_EPP” before the “END” statement. (See figure thirteen)

**Figure 13**

Adding a new connection to the interrupt output of the Digilent Usb-Epp interface

```

345 BEGIN d_usb_epp_dstm_axi
346 PARAMETER INSTANCE = Digilent_Usb_Epp
347 PARAMETER HW_VER = 1.00.a
348 PARAMETER C_BASEADDR = 0x7bc00000
349 PARAMETER C_HIGHADDR = 0x7bc0ffff
350 BUS_INTERFACE S_AXI = axi4lite_0
351 PORT S_AXI_ACLK = clk_100_0000MHzPLL0
352 PORT IFCLK = Digilent_Usb_Epp_IFCLK
353 PORT STMEN = Digilent_Usb_Epp_STMEN
354 PORT FLAGA = Digilent_Usb_Epp_FLAGA
355 PORT FLAGB = Digilent_Usb_Epp_FLAGB
356 PORT FLAGC = Digilent_Usb_Epp_FLAGC
357 PORT SLRD = Digilent_Usb_Epp_SLRD
358 PORT SLWR = Digilent_Usb_Epp_SLWR
359 PORT SLOE = Digilent_Usb_Epp_SLOE
360 PORT FIFOADR = Digilent_Usb_Epp_FIFOADR
361 PORT PKTEND = Digilent_Usb_Epp_PKTEND
362 PORT EPPRST = Digilent_Usb_Epp_EPPRST
363 PORT DB = Digilent_Usb_Epp_DB
364 PORT IRQ_EPP = Digilent_Usb_Epp_DB_IRQ_EPP
365 END
  
```

*Note: Users can add any name for the new connection. However, Digilent recommends following the “core\_name\_port\_name” naming convention*

3d-3. Save the MHS file. Upon saving XPS will ask to reload the design. Choose to reload. After reloading click on the “System Assembly View” and expand the signal group “(IO\_IF) usb\_epp\_irq” in Port View for the “Digilent\_Usb\_Epp” peripheral, *Note: A new connection will appear in the Net column for the IRQ\_EPP port.*

3d-4. Go back to the “system.mhs” file and locate the line “BEGIN axi\_intc.”

- If the user selected no other cores with interrupt in BSB and did not add another interrupt connection in System Assembly view, then the computer will not show the Intr pin port of the “axi\_intc” core. If the Intr pin port is not visible, then add “PORT Intr = Digilent\_Usb\_Epp\_DB\_IRQ\_EPP” before the “END” statement.
- If users have another core using interrupts, then they should already see the port “Intr.” For example, if operators in BSB select the “Use Interrupt” option for the “Ethernet\_Lite” core, then the interrupt output should already be connected. In this case the line should look like “PORT Intr = Ethernet\_Lite\_IP2INTC\_Irpt.”
- If the interrupt is already connected change the line to “PORT Intr = Digilent\_Usb\_Epp\_DB\_IRQ\_EPP & Ethernet\_Lite\_IP2INTC\_Irpt.” (See figure fourteen)

**Figure 14**

Adding a new Interrupt Connection to the interrupt controller

```

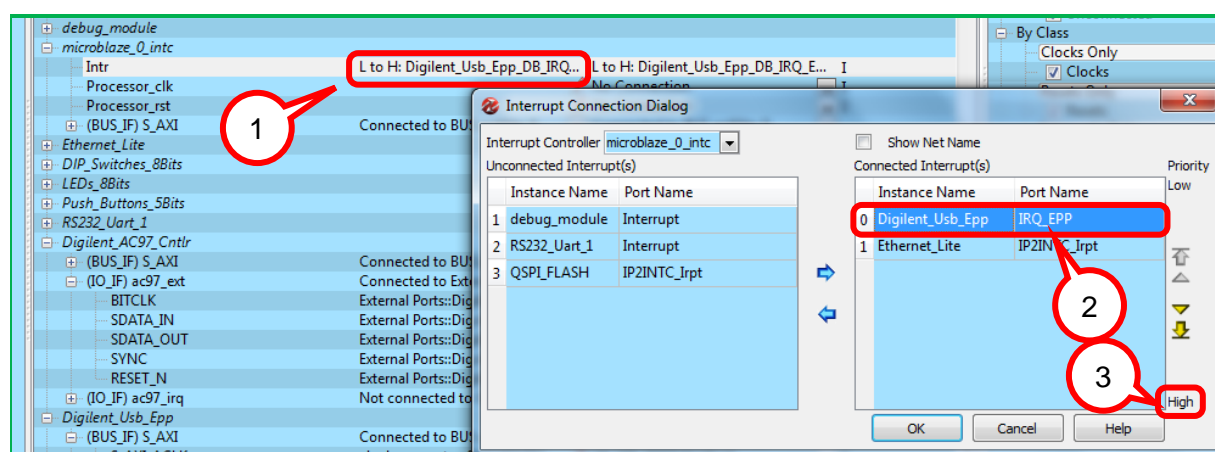
86
87 BEGIN axi_intc
88 PARAMETER INSTANCE = microblaze_0_intc
89 PARAMETER HW_VER = 1.02.a
90 PARAMETER C_BASEADDR = 0x41200000
91 PARAMETER C_HIGHADDR = 0x4120ffff
92 BUS_INTERFACE S_AXI = axi4lite_0
93 BUS_INTERFACE INTERRUPT = microblaze_0_interrupt
94 PORT S_AXI_ACLK = clk_100_0000MHzPLL0
95 PORT Intr = Digilent_Usb_Epp_DB_IRQ_EPP & Ethernet_Lite_IP2INTC_Irpt
96 END
  
```

Note: The "&" sign in figure fourteen serves the same concatenation function as in the VHDL language. The interrupt priority order starts from right to left, with the rightmost being the highest priority.

3d-5. Save the MHS file again and choose Reload.

- Click on the "System Assembly View" and in "Port View" look for the "microblaze\_0\_intc core" and expand it.
- Click on the "Intr" signal in the Connected Ports column (#1).
- Check the pop-up window to ensure that the system recognizes "Digilent\_Usb\_Epp\_DB\_IRQ\_EPP" as an interrupt input signal (#2). Also, if users have more than one interrupt signal connected, check for the interrupt signals priority (#3) in the signal order of the MHS file. (See figure fifteen.)
- The MHS file allows users to edit more than one line in the same time before saving. Therefore, Step 3d-3 may be done together with Step 3d-5.

**Figure 15**  
Checking for interrupt inputs and its priorities for an interrupt controller



Users can make all of the connections presented in section three by editing the MHS file. However, operators may find that Port View is easier and more error-proof than making connections in the System Assembly View.

Be careful when editing the MHS file. Improper connections may lead to bitstream generation failures. Also, syntax errors might cause EDK project corruption. (I.e. the situation that XPS closes the project and announces MHS errors, being unable to open it until the errors are corrected)

Digilent recommends users make a backup copy of their MHS file before editing it. Users may backup their MHS File by either issuing a *File -> Save As...* command or by making a backup copy of the "system.mhs file" in a file explorer. Operators can find the "system.mhs file" in the root of the project directory.

Users can find more information about the MHS file syntax in the "Xilinx Platform Specification Manual" UG642 chapter on "Microprocessor Hardware Specification" (MHS), available at [www.xilinx.com](http://www.xilinx.com).

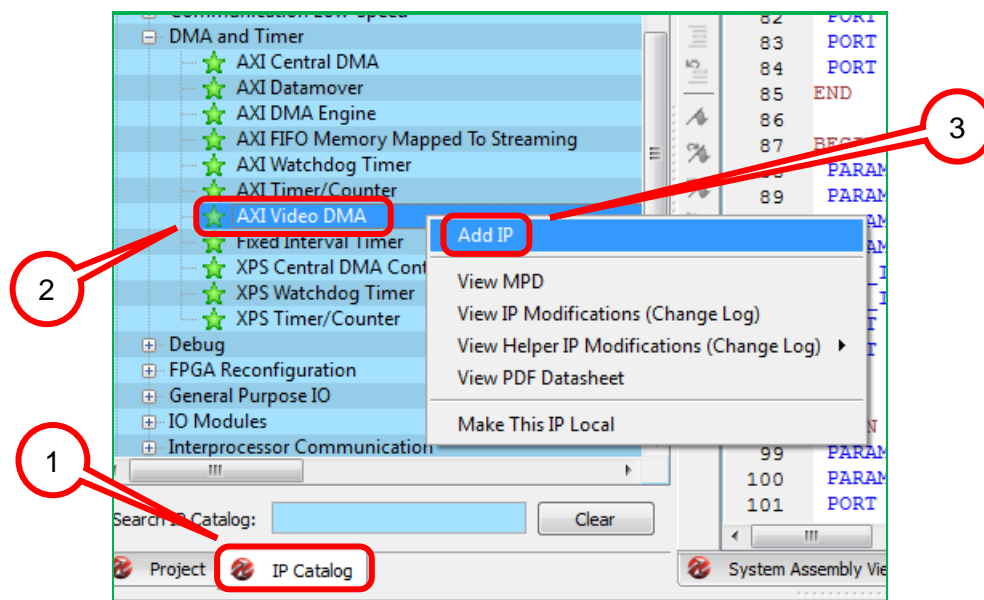
*Note: For older versions of EDK such as 13.2 and earlier, interrupt outputs for custom cores allow new connections. After users make a new connection, they can then connect the interrupt output of a custom core through the wizard shown in figure fifteen by selecting the specific signal and clicking the right facing arrow “->.”*

## IV. Using the Digilent HDMI controller

In order to use the Digilent HDMI controller, users must first connect the AXI Video DMA core. BSB does not include this core because of the large number of tedious manual connections and configurations. However, EDK versions 13.4 and higher support the AXI Video DMA core. Follow steps 4-1 to 4-5 to use the AXI Video DMA core.

- 4-1. Adding an “AXI Video DMA” core: Click on the “IP Catalog” and expand the “DMA and Timer group” (#1). Right-click on “AXI Video DMA” core (#2). Select the “Add IP” option (#3) to open the the XPS Core Config window. (See figure sixteen)

**Figure 16**  
Adding a core from the IP catalog

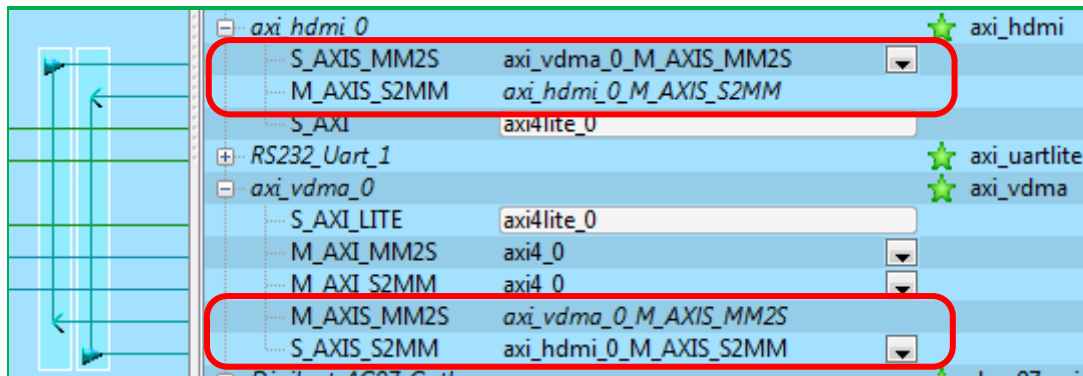


- 4-2. Go to the user tab of the “XPS Core Configuration” window:
- Check the “Enable Asynchronous Clocks” option. For newer versions, expand the “MM2S Channel Options” group. For older versions, the “Primary Clock” is the Asynchronous option.
  - Set the “MM2S Video Line Buffer Depth” to 1024 Bytes
  - Set the “MM2S Video Line Buffer Threshold” to 512 Bytes. For older versions it is the “Almost Empty Threshold” and for newer versions expand the “S2MM Channel Options” group.
  - Set the “S2MM Video Line Buffer Depth” to 1024 Bytes

- Option 1: If the user only needs the HDMI transmitter then use only the “Enable Channel” option from the “S2MM Channel Options.” For older versions uncheck the “Include S2MM Channel.” Also, set the HDMI transmitter option “Use HDMI Receiver” to FALSE.
  - Option 2: If the user uses only the HDMI receiver then use the option “Enable Channel” from the “S2MM Channel Options.” For older versions uncheck the “Include MM2S Channel.” Also, change the HDMI receiver option “Use HDMI Transmitter” to FALSE.
- 4-3. Add an “AXI HDMI Receiver/Transmitter” core in the same manner as step 4-1.
- 4-4. Go to “System Assembly View” and then “Bus Interfaces.” Interconnect the MM2S and S2MM buses between the “axi\_hdmi\_0” and “axi\_vdma\_0.” (See figure seventeen)

**Figure 17**

Interconnecting the MM2S and S2MM buses between the HDMI transmitter and the Video DMA

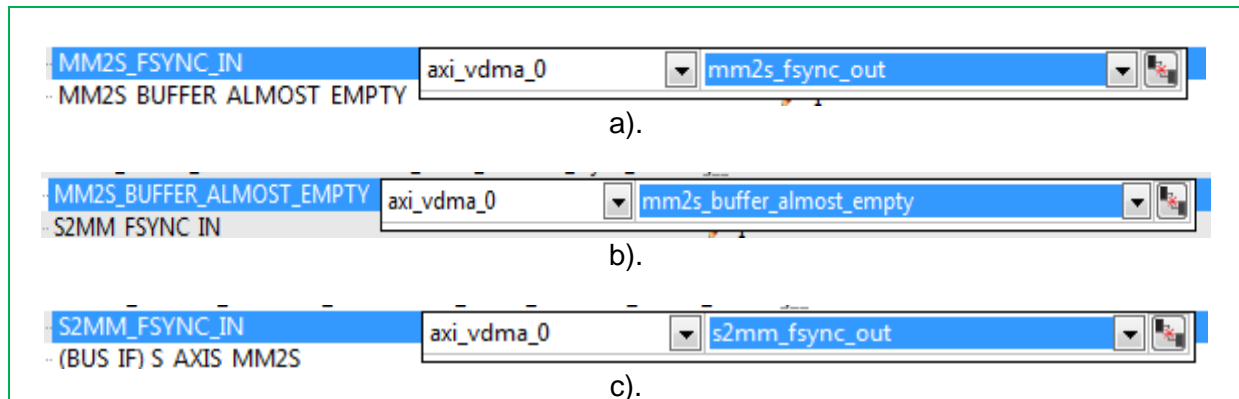


- 4-5. Go to the Ports View and expand the “axi\_hdmi\_0” instance. Click on the pen icon for the port “MM2S\_FSYNC\_IN” and select core “axi\_vdma\_0” signal “mm2s\_fsync\_out.”

In a similar manner, connect port “MM2S\_BUFFER\_ALMOST\_EMPTY” to signal “mm2s\_buffer\_almost\_empty” of “axi\_vdma\_0” and port “S2MM\_FSYNC\_IN” to signal “s2mm\_fsync\_out” of “axi\_vdma\_0.” Users should refer to figure eighteen for details on port connection.



**Figure 18**  
Connecting additional signals on the Digilent HDMI Controller



- 4-6. Open the “system.mhs” link from the “Project” tab and locate the line “BEGIN axi\_hdmi.” Add the following lines before the “END” statement (See figure nineteen for screenshot):

```
PORT S_AXIS_MM2S_ACLK = S_AXIS_MM2S_ACLK_int
PORT M_AXIS_S2MM_ACLK = M_AXIS_S2MM_ACLK_int
PORT ACLK = clk_100_0000MHzPLL0
```

**Figure 19**  
Connecting clock signals for the Digilent HDMI Controller in the MHS file

```
384 BEGIN axi_hdmi
385     PARAMETER INSTANCE = axi_hdmi_0
386     PARAMETER HW_VER = 1.00.a
387     PARAMETER C_BASEADDR = 0x7e220000
388     PARAMETER C_HIGHADDR = 0x7e22ffff
389     BUS_INTERFACE S_AXI = axi4lite_0
390     BUS_INTERFACE M_AXIS_S2MM = axi_hdmi_0 M_AXIS_S2MM
391     BUS_INTERFACE S_AXIS_MM2S = axi_vdma_0 M_AXIS_MM2S
392     PORT S_AXI_ACLK = clk_100_0000MHzPLL0
393     PORT S_AXIS_MM2S_ACLK = S_AXIS_MM2S_ACLK_int
394     PORT M_AXIS_S2MM_ACLK = M_AXIS_S2MM_ACLK_int
395     PORT ACLK = clk_100_0000MHzPLL0
396     PORT TMD5_RX_CLK_P = axi_hdmi_0 TMD5_RX_CLK_P
397     PORT TMD5_RX_CLK_N = axi_hdmi_0 TMD5_RX_CLK_N
398     PORT TMD5_RX_2_P = axi_hdmi_0 TMD5_RX_2_P
399     PORT TMD5_RX_2_N = axi_hdmi_0 TMD5_RX_2_N
400     PORT TMD5_RX_1_P = axi_hdmi_0 TMD5_RX_1_P
```

- 4-7. Look for the line “BEGIN axi\_vdma” and add the following lines (See figure twenty for a screenshot):

```
PORT m_axis_mm2s_aclk = S_AXIS_MM2S_ACLK_int
PORT s_axis_s2mm_aclk = M_AXIS_S2MM_ACLK_int
```



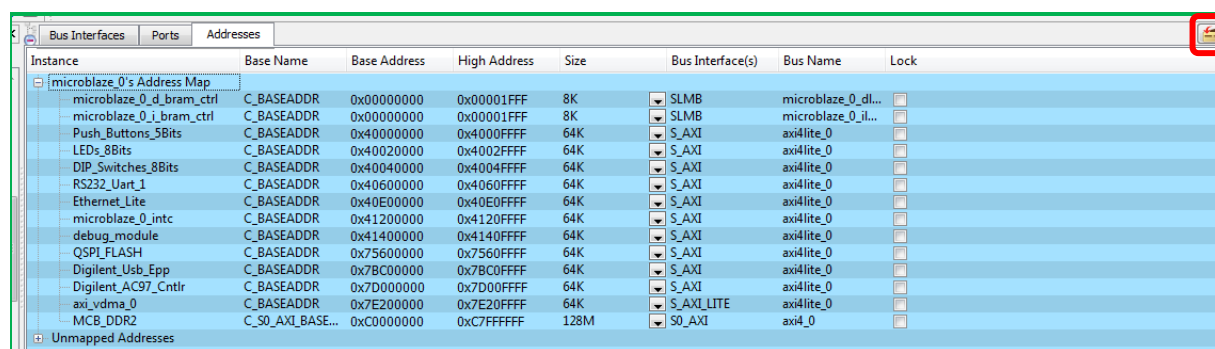
**Figure 20**  
Connecting clock signals for the Video DMA in the MHS file

```

360 BEGIN axi_vdma
361     PARAMETER INSTANCE = axi_vdma_0
362     PARAMETER HW_VER = 4.00.a
363     PARAMETER C_ENABLE_VIDPRMTR_READS = 0
364     PARAMETER C_MM2S_LINEBUFFER_DEPTH = 1024
365     PARAMETER C_S2MM_LINEBUFFER_DEPTH = 1024
366     PARAMETER C_PRMRY_IS_ACLK_ASYNC = 1
367     PARAMETER C_BASEADDR = 0x7e200000
368     PARAMETER C_HIGHADDR = 0x7e20ffff
369     BUS_INTERFACE S_AXI_LITE = axi4lite_0
370     BUS_INTERFACE M_AXI_MM2S = axi4_0
371     BUS_INTERFACE M_AXI_S2MM = axi4_0
372     BUS_INTERFACE S_AXIS_S2MM = axi_hdmi_0_M_AXIS_S2MM
373     BUS_INTERFACE M_AXIS_MM2S = axi_vdma_0_M_AXIS_MM2S
374     PORT s_axi_lite_aclk = clk_100_0000MHzPLL0
375     PORT m_axi_mm2s_aclk = clk_100_0000MHzPLL0
376     PORT m_axi_s2mm_aclk = clk_100_0000MHzPLL0
377     PORT m_axis_mm2s_aclk = S_AXIS_MM2S_ACLK_int
378     PORT s_axis_s2mm_aclk = M_AXIS_S2MM_ACLK_int
379     PORT mm2s_fsync_out = axi_vdma_0_mm2s_fsync_out
  
```

- 4-8. Go to the “Addresses” tab in the “System Assembly View” and click on the “Regenerate addresses” icon to generate addresses for the axi\_hdmi core. (See figure twenty-one) Regenerating the addresses should remove the “Unmapped Addresses” group.

**Figure 21**  
Regenerate address space for the axi\_hdmi core



- 4-9. From the “Project” tab open “system.ucf.” Go to the downloaded AXI Board Support Files, find the “../Atlys\_AXI\_BSB\_Support/lib/Digilent/boards/Digilent\_Atlys/data” folder and use a text editor to open the “Digilent\_HDMI\_axi\_hdmi\_v1\_00\_a.ucf” file. Users can use XPS to open the specific file and copy its contents to the end of the “system.ucf” file. (See figure twenty-two). Do not forget to add an extra line to the last .ucf line.
- 4-10. Finally, save the .ucf file. Users can now begin generating bitstream for their project.

Figure 22  
Adding the .ucf constraints to the "system.ucf" file

```

59 NET RS232_Uart_1_sout LOC = "B16" | IOSTANDARD = "LVCMOS33";
60 NET rzq IOSTANDARD = "LVCMOS18_JEDEC";
61 NET zic IOSTANDARD = "LVCMOS18_JEDEC";
62 #
63 # additional constraints
64 #
65
66 NET "GCLK" TNM_NET = sys_clk_pin;
67 TIMESPEC TS_sys_clk_pin = PERIOD sys_clk_pin 100000 kHz;
68
69 ##### HDMI Core constraints
70 # Overwrite existing VCCAUX setting for TMDS interfaces
71 CONFIG VCCAUX = 3.3;
72
73 #####
74
75 NET "**TMDS_TX_0_N*" LOC = "C8" | IOSTANDARD = "TMDS_33";
76 NET "**TMDS_TX_0_P*" LOC = "D8" | IOSTANDARD = "TMDS_33";
77 NET "**TMDS_TX_1_N*" LOC = "A7" | IOSTANDARD = "TMDS_33";
78 NET "**TMDS_TX_1_P*" LOC = "C7" | IOSTANDARD = "TMDS_33";
79 NET "**TMDS_TX_2_N*" LOC = "A8" | IOSTANDARD = "TMDS_33";
80 NET "**TMDS_TX_2_P*" LOC = "B8" | IOSTANDARD = "TMDS_33";
81 NET "**TMDS_TX_CLK_N*" LOC = "A6" | IOSTANDARD = "TMDS_33";
82 NET "**TMDS_TX_CLK_P*" LOC = "B6" | IOSTANDARD = "TMDS_33";
83 NET "**TMDS_RX_0_N*" LOC = "K18" | IOSTANDARD = "TMDS_33";
84 NET "**TMDS_RX_0_P*" LOC = "K17" | IOSTANDARD = "TMDS_33";
85 NET "**TMDS_RX_1_N*" LOC = "L18" | IOSTANDARD = "TMDS_33";
86 NET "**TMDS_RX_1_P*" LOC = "L17" | IOSTANDARD = "TMDS_33";
87 NET "**TMDS_RX_2_N*" LOC = "J18" | IOSTANDARD = "TMDS_33";
88 NET "**TMDS_RX_2_P*" LOC = "J16" | IOSTANDARD = "TMDS_33";
89 NET "**TMDS_RX_CLK_N*" LOC = "H18" | IOSTANDARD = "TMDS_33";
90 NET "**TMDS_RX_CLK_P*" LOC = "H17" | IOSTANDARD = "TMDS_33";
91 NET "**TMDS_RX_SCL*" LOC = "M16" | IOSTANDARD = "I2C";
92 NET "**TMDS_RX_SDA*" LOC = "M18" | IOSTANDARD = "I2C";
93
94 #####
95
96 NET "**SIG_PLL0?CLKOUT2" TNM_NET = globclk;
97 NET "**Inst_DynClkGen?PllOut_x1" TNM_NET = hdmipclk;
98
99 #####
100 # Timing Constraints
101 #####
102
103 TIMESPEC TS_path1 = FROM globclk TO hdmipclk 14 ns;
104 TIMESPEC TS_path2 = FROM hdmipclk TO globclk 10 ns;

```