Atlys BSB Support Files for AXI-based EDK 13_2..14_3 Designs

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

Revision: October, 30th, 2012

Overview

This package will integrate Base System Builder (BSB) support for the Atlys Spartan-6 FPGA Development Board into Xilinx EDK tools. It includes board definition files and Digilent custom core support files for creating AXI-based MicroBlaze embedded designs in BSB.

With these files the BSB can be used to create Platform Studio projects initialized with cores that are properly configured to control the on-board peripherals.

Contents:

1. Checklist for using the BSB Support Files

In order to use the peripherals fully supported by Xilinx BSB only, do the following steps:

1. Install the Digilent AXI IPCore Support Files. This can be done by running ..\Digilent_AXI_IPCore_Support_v_1_34\ inst_uninst.bat and following the on-screen instructions.
   Notes:
   a. The current version of the Digilent AXI IPCore Support Files is 1_34. If you previously installed an older version of the AXI IPCore support files for the current version of EDK, it is recommended to uninstall those files first.
   b. The Digilent AXI IPCore Support Files has to be installed only once per EDK version and is valid for all of the Digilent boards.
   Therefore, if using another Digilent board, for example, Nexys3, the Digilent AXI IPCore Support Files do not need to be installed again for the current version of EDK

2. Create the Base System using the Base System Builder Wizard. For details, see chapter "**Creating a Base System using BSB Wizard**"

Note: The currently supported peripherals by Xilinx BSB are outlined in Table 1 below.

Additionally, in order to use the peripherals supported by custom cores, in System Assembly View do the following steps:

3. For each custom core, make the AXI clock connections. For details, see chapter "**Connecting AXI Clock Signals to the custom cores**".

4. For each custom core, connect its external ports. For details, see chapter "**Making external port connections for the custom cores**".

5. For custom cores having I/O signals, check and correct the I/O signal names in order to match to the signal names in the .ucf file. For details, see chapter "**Checking I/O signal naming for the _pin suffix**".

6. Optionally, for each core having interrupt output signal, connect this signal to the interrupt controller. For details, see chapters "**Configuring and removing peripherals**", then "**Connecting internal and interrupt signals for custom cores**".

Notes:

a) The current peripherals supported by custom cores are outlined in Table 2 below.
b) By default, the custom cores are selected in BSB Wizard. However, any core can be removed in BSB Wizard, in the "Select and Configure Peripherals" window. For details, see chapter "Configuring and removing peripherals"

After the steps above are completed, the bitstream can be generated and the project can be exported to SDK.

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". The core can be find in EDK in the "Project Peripheral Repository 0/Digilent" The core allows changing the SPI mode (single, dual, 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

For additional information on using the cores above, please refer to their PDF datasheets

2. Creating a Base System using BSB Wizard

Note: For EDK versions older than 14.3, the screenshots below may differ. However, the relevance of the data entered into the text boxes and combo boxes remains the same.

Start Xilinx Platform Studio and create a new project using Base System Builder by selecting “Create New Project Using Base System Builder”. The “Create New XPS Project Using BSB Wizard” will appear, as shown in Figure 1 below.
In this window:

1. Make sure that “AXI System” is selected

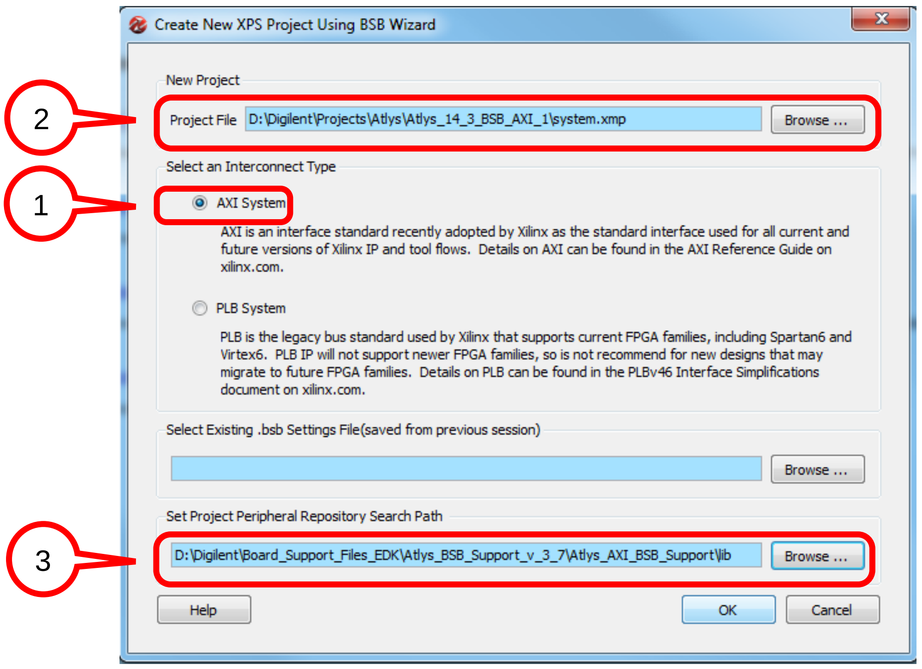
2. Click on the “Browse” button beside the “Project File” and select a folder where the system.xmp project file will be located.

   Notes:

   a. It is **NOT** recommended to use a path which contains spaces, such as “My Documents” for the project folder, because a folder like this might affect the functionality of the EDK and SDK tools, from which many are linux-based.
   b. It is recommended to use an empty folder. The whole EDK hardware project will be stored in the same folder, therefore it is easier to archive and copy the whole project. Later, if preferred, SDK can also be set to use as project workspace a subfolder in the project root folder.
   c. By default, the project description file is “system.xmp”. However, it can be given any name if the file extension is kept and obviously, does not contain spaces.

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 AXI Support Files folder, select the **.\lib** subfolder and click OK.

   Note: Obviously, the steps above can be made in any order until you don’t click the OK button of the “Create New XPS Project Using BSB Wizard” window.


Create New XPS Project Using BSB Wizard

New Project

Project File: D:\Digilent\Projects\Atlys\Atlys_14_3_BSB_AXI_1\system.xmp

Select an Interconnect Type

[✓ AXI System]

AXI is an interface standard recently adopted by Xilinx as the standard interface used for all current and future versions of Xilinx IP and tool flows. Details on AXI can be found in the AXI Reference Guide on xilinx.com.

PLB System

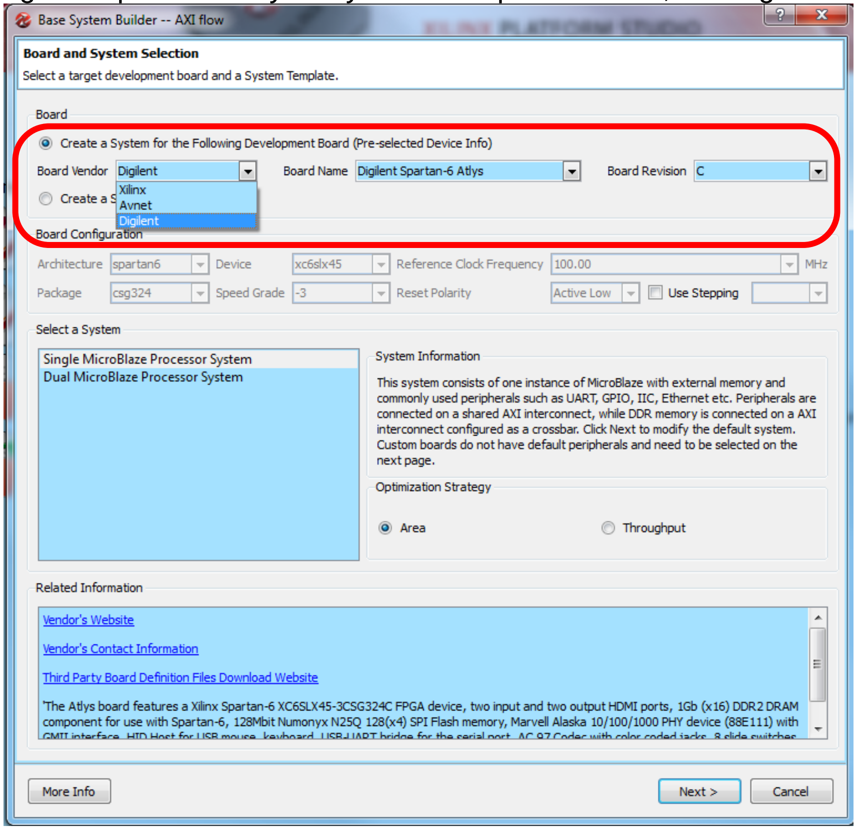
PLB is the legacy bus standard used by Xilinx that supports current FPGA families, including Spartan6 and Virtex6. PLB IP will not support newer FPGA families, so is not recommend for new designs that may migrate to future FPGA families. Details on PLB can be found in the PLBv46 Interface Simplifications document on xilinx.com.

Select Existing .bsb Settings File(saved from previous session)

Set Project Peripheral Repository Search Path: D:\Digilent\Board_Support_Files_EDK\Atlys_BSB_Support_v_3_7\Atlys_AXI_BSB_Support\lib

Figure 1. BSB window with specifying the project type, project location and the Peripheral Repository Search Path

After you click OK, in the next window you should now be able to select the Digilent as vendor and the Digilent Spartan-6 Atlys as your development board, see Figure 2.



Base System Builder -- AXI flow

Board and System Selection

Select a target development board and a System Template.

Board

Create a System for the Following Development Board (Pre-selected Device Info)

Board Vendor: Digilent (Xilinx, Avnet, Digilent) — Board Name: Digilent Spartan-6 Atlys — Board Revision: C

Board Configuration

Architecture: spartan6 — Device: xc6slx45 — Reference Clock Frequency: 100.00 MHz

Package: csg324 — Speed Grade: -3 — Reset Polarity: Active Low — Use Stepping

Select a System

Single MicroBlaze Processor System

Dual MicroBlaze Processor System

System Information

This system consists of one instance of MicroBlaze with external memory and commonly used peripherals such as UART, GPIO, IIC, Ethernet etc. Peripherals are connected on a shared AXI interconnect, while DDR memory is connected on a AXI interconnect configured as a crossbar. Click Next to modify the default system. Custom boards do not have default peripherals and need to be selected on the next page.

Optimization Strategy

[✓ Area] Throughput

Related Information

Vendor's Website

Vendor's Contact Information

Third Party Board Definition Files Download Website

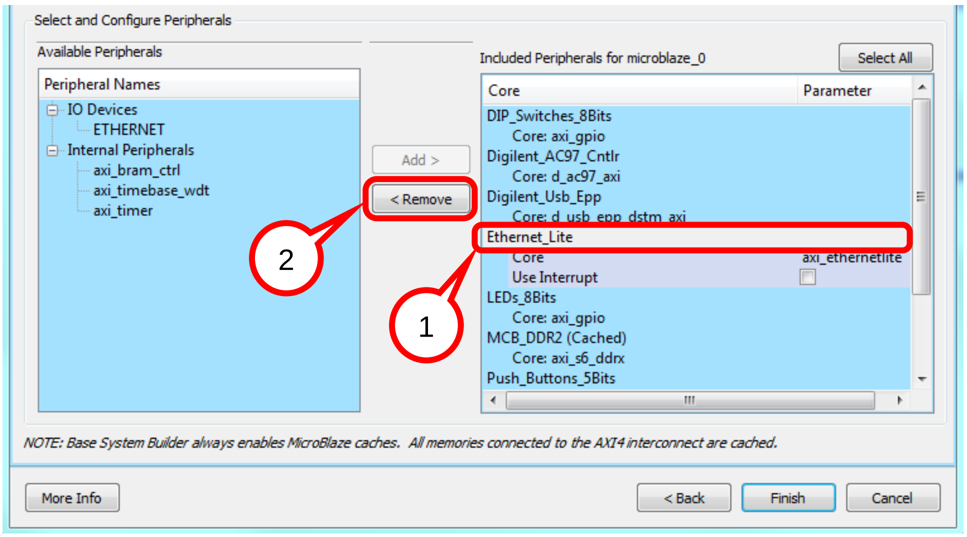
'The Atlys board features a Xilinx Spartan-6 XC6SLX45-3CSG324C FPGA device, two input and two output HDMI ports, 1Gb (x16) DDR2 DRAM component for use with Spartan-6, 128Mbit Numonyx N25Q 128(x4) SPI Flash memory, Marvell Alaska 10/100/1000 PHY device (88E111) with

Figure 2. Board and System Selection Window

Click "**Next**". The following window is the "Processor, Cache and Peripheral Configuration" window. In this one the peripherals can be selected, some of the peripherals can be configured and unwanted peripherals can be removed.

Configuring and removing peripherals

In order to remove a peripheral, simply click on the specific peripheral and then click "Remove", see Figure 3 for details:



Select and Configure Peripherals

Available Peripherals

Peripheral Names
- IO Devices
  - ETHERNET
- Internal Peripherals
  - axi_bram_ctrl
  - axi_timebase_wdt
  - axi_timer

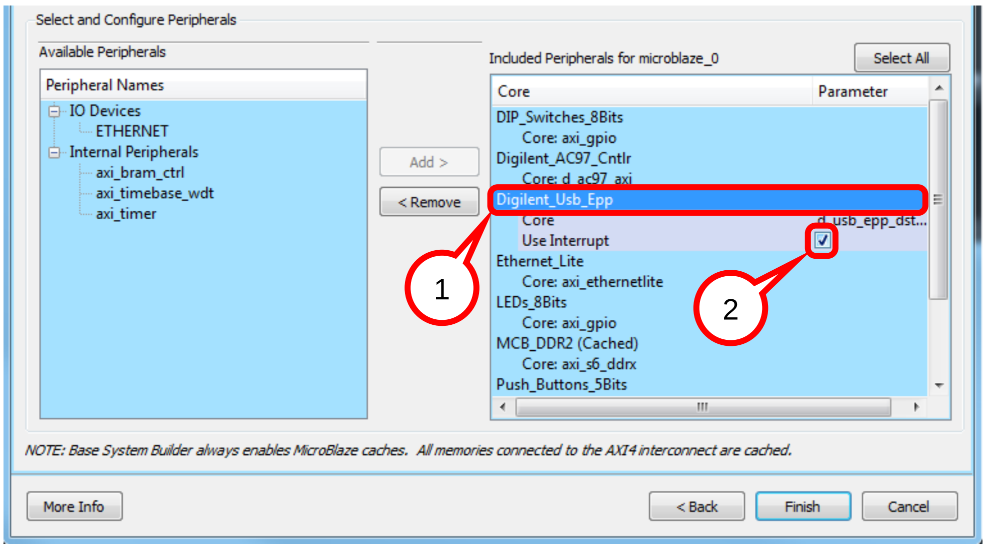
Included Peripherals for microblaze_0

Core	Parameter
DIP_Switches_8Bits	
Core: axi_gpio	
Digilent_AC97_Cntlr	
Core: d_ac97_axi	
Digilent_Usb_Epp	
Core: d_usb_epp_dstm_axi	
Ethernet_Lite	
Core	axi_ethernetlite
Use Interrupt	
LEDs_8Bits	
Core: axi_gpio	
MCB_DDR2 (Cached)	
Core: axi_s6_ddrx	
Push_Buttons_5Bits	

NOTE: Base System Builder always enables MicroBlaze caches. All memories connected to the AXI4 interconnect are cached.

Figure 3. Removing a Peripheral in BSB

Note: When a peripheral is selected, its parameters that can be configured in BSB, if any, are shown. For example, Figure 4 shows the example to select the interrupt option for the Digilent USB-EPP interface:


Select and Configure Peripherals

Available Peripherals

Peripheral Names
- IO Devices
  - ETHERNET
- Internal Peripherals
  - axi_bram_ctrl
  - axi_timebase_wdt
  - axi_timer

Included Peripherals for microblaze_0

Core	Parameter
DIP_Switches_8Bits	
Core: axi_gpio	
Digilent_AC97_Cntlr	
Core: d_ac97_axi	
Digilent_Usb_Epp	
Core	d_usb_epp_dst...
Use Interrupt	✓
Ethernet_Lite	
Core: axi_ethernetlite	
LEDs_8Bits	
Core: axi_gpio	
MCB_DDR2 (Cached)	
Core: axi_s6_ddrx	
Push_Buttons_5Bits	

NOTE: Base System Builder always enables MicroBlaze caches. All memories connected to the AXI4 interconnect are cached.

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

Selecting the “Use Interrupt” option for any peripheral having this option will make BSB to add an interrupt controller to the system. Therefore, if no other peripheral has this option, in order to use interrupts with the Digilent USB-EPP interface, the steps above needs to be made.

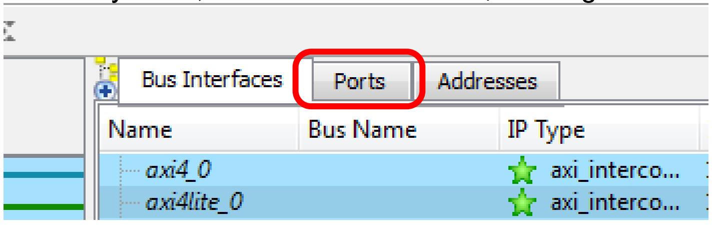
However, because the Digilent USB-EPP interface is a custom core, its interrupt signal will have to be manually connected to the interrupt controller. See chapter “Connecting internal and interrupt signals” later in this document.

After the needed cores are selected and/or configured, click “Finish”. The Base System is generated and Xilinx Platform Studio (XPS) will bring up the System Assembly View showing the Bus Connections of the various peripherals.

3. Making Connections for the Custom Cores

Connecting AXI Clock Signals to the custom cores

1. In System Assembly View, select the “Ports” tab, see Figure 5 below:



Name	Bus Name	IP Type
axi4_0		axi_interco...
axi4lite_0		axi_interco...

Figure 5. Selecting the Ports View

2. If the “Net” column is not sown in the Ports view (by default, is shown in older EDK versions), show it by right-clicking on the Ports View header and selecting “Net”, as shown in the figure below:

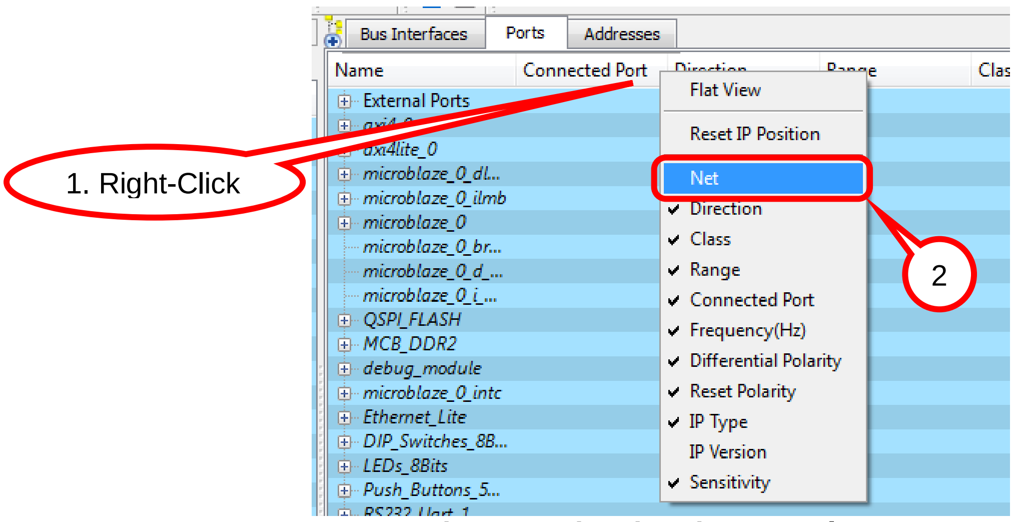


Figure 6. Showing the Net column

Drag the width of columns as appropriate in order to see the full signal names.

3. Expand “Digilent_Usb_Epp”, then “(BUS_IF) S_AXI”. In the Net column select for S_AXI_ACLK the signal “clk_100_0000MHzPLL0”, as shown in the figure below:

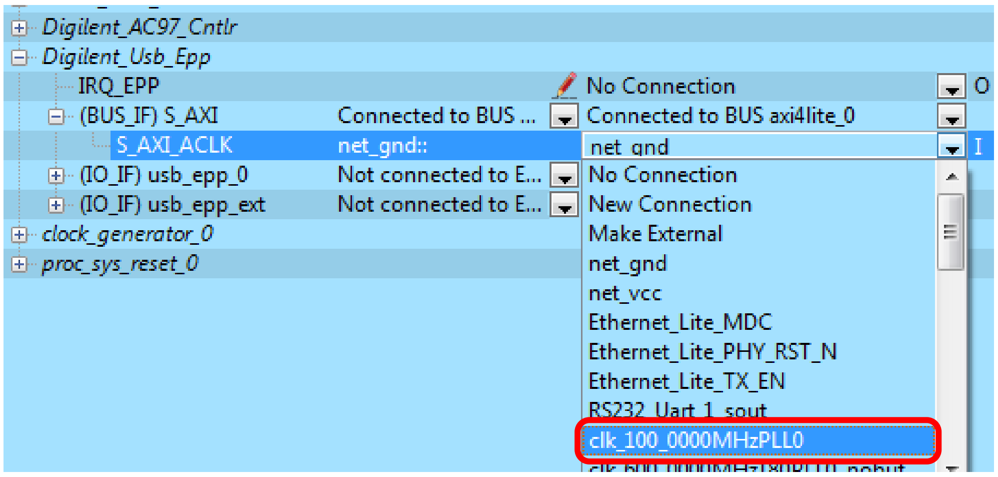


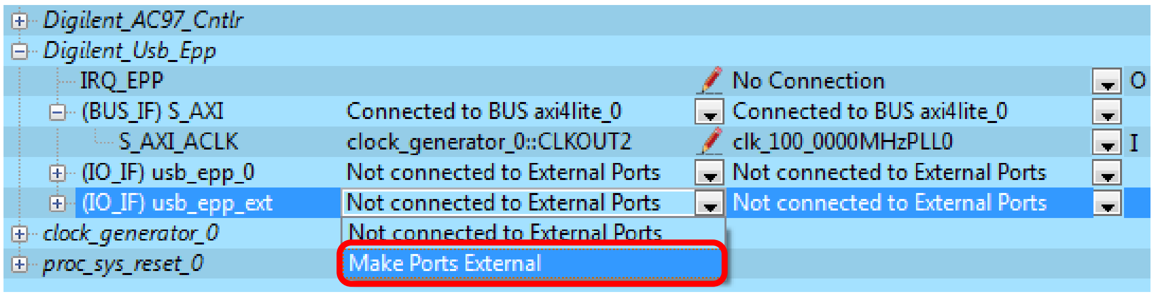
Figure 7. Connecting the AXI clock signal for a custom core

Note: The signal “clk_100_0000MHzPLL0” is both the processor clock and the AXI clock, assuming that in the “Processor, Cache and Peripheral Configuration” window (its lower part is shown in Figure 3) the clock frequency did not change from 100 MHz. Usually, the system clock signal is named in the “clk_frequencyMHzPLLnumber”.

4. Repeat Step 3 above for “Digilent_AC97_Cntlr”, if this core is also present in the system.

Making external port connections for the custom cores

1. For Digilent_Usb_Epp, select the signal group – IO Interface “(IO_IF) usb_epp_ext” and, in the Connected Ports column, expand the combo box, then select “Make Ports External”, as shown in the figure below. This will connect all of the core ports to external FPGA ports.

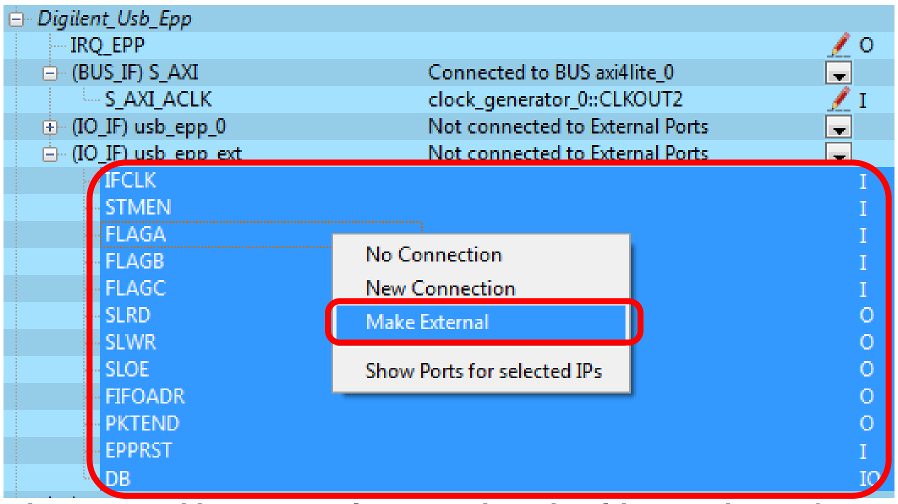


Digilent_AC97_Cntlr			
Digilent_Usb_Epp			
IRQ_EPP		No Connection	O
(BUS_IF) S_AXI	Connected to BUS axi4lite_0	Connected to BUS axi4lite_0	
S_AXI_ACLK	clock_generator_0::CLKOUT2	clk_100_0000MHzPLL0	I
(IO_IF) usb_epp_0	Not connected to External Ports	Not connected to External Ports	
(IO_IF) usb_epp_ext	Not connected to External Ports	Not connected to External Ports	
clock_generator_0			
proc_sys_reset_0			

Figure 8. Making an external connection

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.
In this case expand the corresponding signal group and select all of the signals by holding the CTRL or SHIFT keys down. Then right-click on the selection and choose “Make External”, as in the figure below:



Digilent_Usb_Epp		
IRQ_EPP		O
(BUS_IF) S_AXI	Connected to BUS axi4lite_0	
S_AXI_ACLK	clock_generator_0::CLKOUT2	I
(IO_IF) usb_epp_0	Not connected to External Ports	
(IO_IF) usb_epp_ext	Not connected to External Ports	
IFCLK		I
STMEN		I
FLAGA		I
FLAGB		I
FLAGC		I
SLRD		O
SLWR		O
SLOE		O
FIFOADR		O
PKTEND		O
EPPRST		I
DB		IO

Figure 9. Making external connections in older versions of EDK

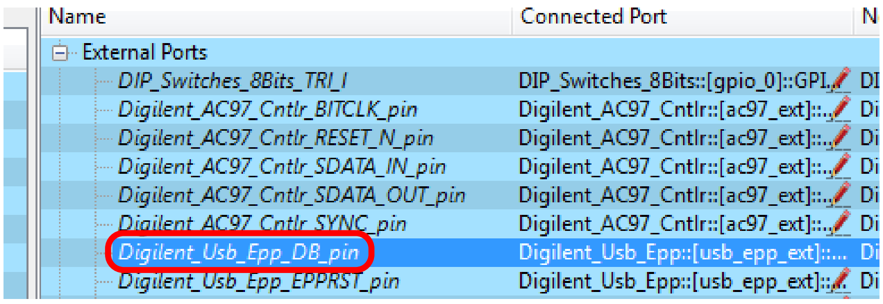
Checking I/O signal naming for the _pin suffix

When an I/O signal is made external, some versions of EDK add the “_pin” suffix to the signal name. However, in the .ucf file generated by BSB the signal names might be without the “_pin” suffix. Obviously, the external signal names will have to match in both files.

For the Atlys board, this case might happen for the DB signal of the Digilent_Usb_Epp peripheral.

In order to check for I/O signal naming, do the following steps:

1. In Ports View, scroll up and expand “External Ports”. Check for the port named “Digilent_Usb_Epp_DB” or “Digilent_Usb_Epp_DB_pin”, as shown in the figure below:



Name	Connected Port	N
External Ports		
DIP_Switches_8Bits_TRI_I	DIP_Switches_8Bits::[gpio_0]::GPI	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

Figure 10. Viewing external port names

2. Select the “Project” tab then double-click on the “system.ucf” file, as shown in the figure below. The .ucf file will open.

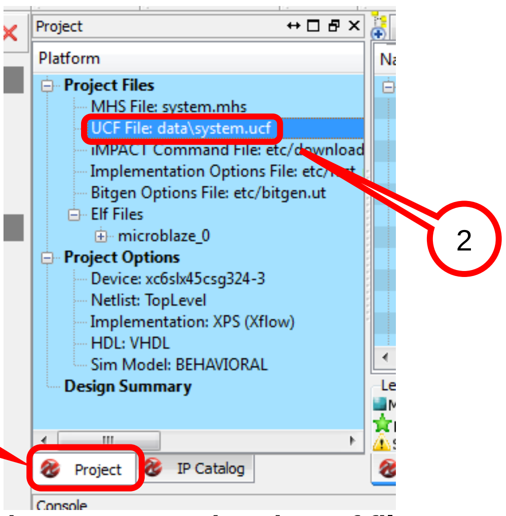
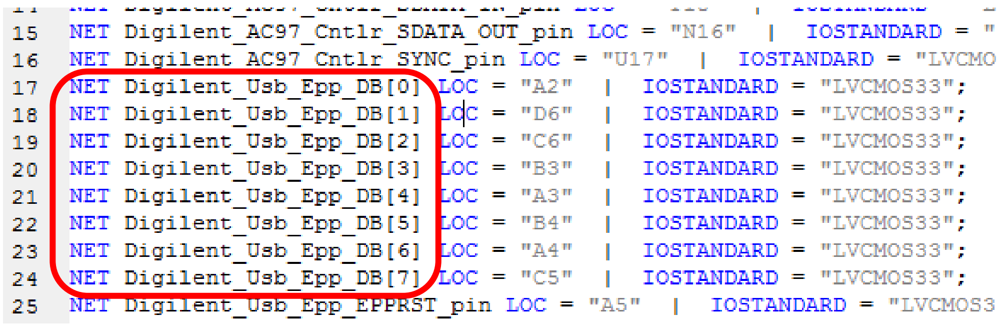


Figure 11. Opening the ucf file

3. Locate the “NET Digilent_Usb_Epp_DB[]” lines in the .ucf file, as shown in the figure below. Note that the net names do not contain the “_pin” suffix.



```
NET Digilent_AC97_Cntlr_SDATA_OUT_pin LOC = "N16"  | IOSTANDARD = "
NET Digilent_AC97_Cntlr_SYNC_pin LOC = "U17"  | IOSTANDARD = "LVCMO
NET Digilent_Usb_Epp_DB[0] LOC = "A2"  | IOSTANDARD = "LVCMOS33";
NET Digilent_Usb_Epp_DB[1] LOC = "D6"  | IOSTANDARD = "LVCMOS33";
NET Digilent_Usb_Epp_DB[2] LOC = "C6"  | IOSTANDARD = "LVCMOS33";
NET Digilent_Usb_Epp_DB[3] LOC = "B3"  | IOSTANDARD = "LVCMOS33";
NET Digilent_Usb_Epp_DB[4] LOC = "A3"  | IOSTANDARD = "LVCMOS33";
NET Digilent_Usb_Epp_DB[5] LOC = "B4"  | IOSTANDARD = "LVCMOS33";
NET Digilent_Usb_Epp_DB[6] LOC = "A4"  | IOSTANDARD = "LVCMOS33";
NET Digilent_Usb_Epp_DB[7] LOC = "C5"  | IOSTANDARD = "LVCMOS33";
NET Digilent_Usb_Epp_EPPRST_pin LOC = "A5"  | IOSTANDARD = "LVCMOS3
```

Figure 12. Viewing ucf constraints

In this case, either add the “_pin” suffix in the .ucf file after “_DB”, so, for example, “Digilent_Usb_Epp_DB[0]” becomes “Digilent_Usb_Epp_DB_pin[0]”, or, rename the external port to remove the _pin suffix in Port View. Noticeably, removing the “_pin” suffix in Port View is easier.

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

Connecting internal and interrupt signals for custom cores

EPP requests for the Digilent USB-EPP interface come from the USB port. The EPP protocol is time-out based Therefore If the processor does not answer in about 100 ms, the PC application will signal a timeout. So, it is recommended that EPP requests to be handled with an interrupt service routine instead of continuously polling the interface status.

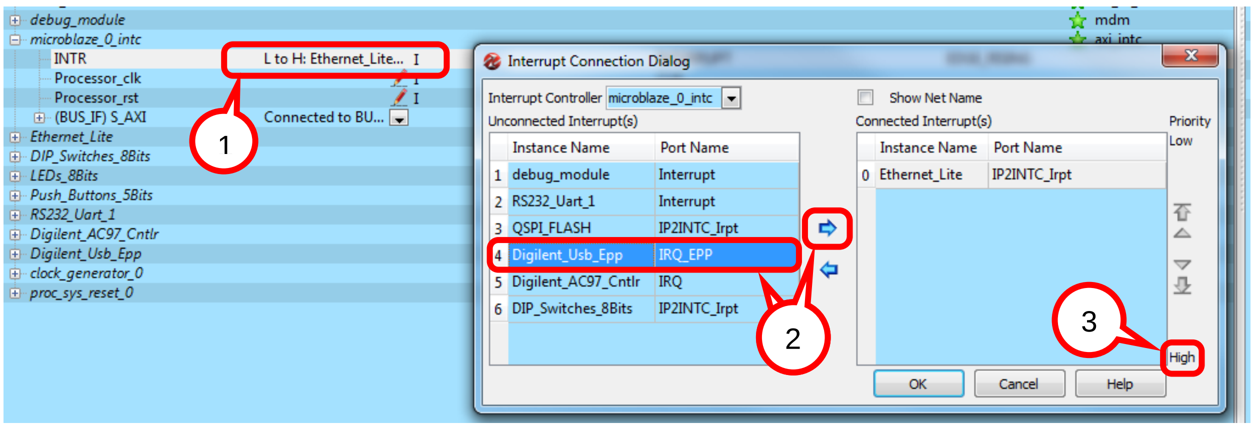
The demo applications include examples for using the USB-EPP interface in both polling and interrupt mode.

In order to use interrupt service routines, the interrupt request signal for the Digilent USB-EPP has to be connected to either an interrupt controller or the Microblaze processor interrupt input.

If the “Use Interrupt” option is selected for any core in BSB, then the Base System Builder will add an interrupt controller to the system. Otherwise, the interrupt controller has to be manually added and connected to the system.

In order to connect the interrupt output of a custom core to the interrupt controller:

1. Click on the System Assembly View and in Port View look for the microblaze_0_intc core then expand it. In the Connected Ports column click on the INTR signal.
2. In the pop-up window in the left side select each interrupt signal that is desired to be connected, then click the right arrow.
3. In the right side use the up and down arrows to set the interrupt priorities, if desired.



Interrupt Controller microblaze_0_intc

Unconnected Interrupt(s)

	Instance Name	Port Name
1	debug_module	Interrupt
2	RS232_Uart_1	Interrupt
3	QSPI_FLASH	IP2INTC_Irpt
4	Digilent_Usb_Epp	IRQ_EPP
5	Digilent_AC97_Cntlr	IRQ
6	DIP_Switches_8Bits	IP2INTC_Irpt

Connected Interrupt(s)

	Instance Name	Port Name
0	Ethernet_Lite	IP2INTC_Irpt

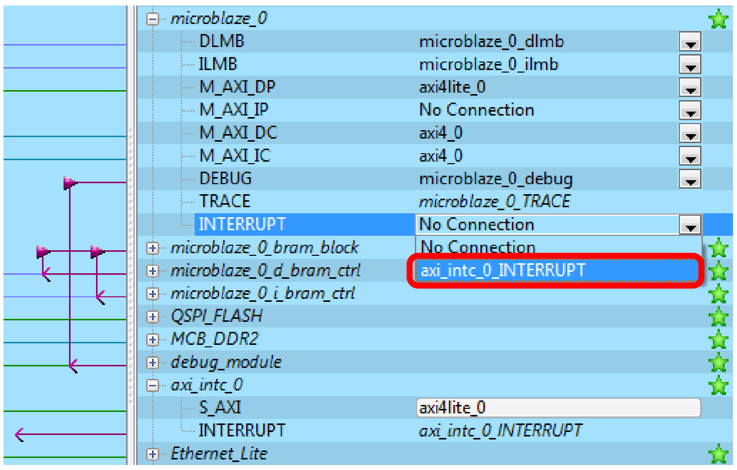
Figure 13. Adding interrupt signals to an interrupt controller and setting the interrupt priorities

Note:

If the BSB wizard added the interrupt controller, its name, by default, will be “microblaze_0_intc”.Howewer, if the interrupt controller is manually added later in System Assembly View, then its name, by default, will be “axi_intc_0”.

If the interrupt controller is manually added to the system, then its Interrupt output will have to be manually connected to the Microblaze processor.

This can be done in System Assembly View, click on the “Bus Interfaces” tab, expand the Microblaze processor (by default “microblaze_0”) and make the connection to the “Interrupt” port, as shown in the figure below:



microblaze_0	
DLMB	microblaze_0_dlmb
ILMB	microblaze_0_ilmb
M_AXI_DP	axi4lite_0
M_AXI_IP	No Connection
M_AXI_DC	axi4_0
M_AXI_IC	axi4_0
DEBUG	microblaze_0_debug
TRACE	microblaze_0_TRACE
INTERRUPT	No Connection
microblaze_0_bram_block	No Connection
microblaze_0_d_bram_ctrl	axi_intc_0_INTERRUPT
microblaze_0_i_bram_ctrl	
QSPI_FLASH	
MCB_DDR2	
debug_module	
axi_intc_0	
S_AXI	axi4lite_0
INTERRUPT	axi_intc_0_INTERRUPT
Ethernet_Lite	

Figure 14. Connecting manually the Interrupt signal from the interrupt controller to the Microblaze processor

Note:

All of the connections presented in the chapters above can be made by editing the MHS file. However, making connections in the System Assembly View, Port View is easier and more error-proof.

Be careful when editing the MHS file. Improper connections can lead to bitstream generation failures. Also, syntax errors might lead to EDK project corruption, i.e. the situation that XPS closes the project and announces MHS errors, not being able to open it until the errors are connected.

Therefore it is recommended to make a backup copy of the MHS file before editing it.

This can be done either by a File -> Save As… command or by making a backup copy of the system.mhs file in a file explorer. The system.mhs file can be found in the root of the directory where the project is located.

More information about the MHS file syntax can be fund in "Xilinx Platform Specification Manual" UG642, chapter "Microprocessor Hardware Specification (MHS), on www.xilinx.com.

4. Using the Digilent HDMI controller

Note: This core is not included in BSB due to many connections and configurations that have to be manually made. Also, this core is supported only in EDK versions 13.4 and above.

In order to use the Digilent HDMI controller, first the AXI Video DMA core must be connected. In order use the core, go through the following steps:

1. Add an "AXI Video DMA" core:
   Click on the "IP Catalog", expand the "DMA and Timer group", select "AXI Video DMA" core, right-click and select "Add IP", as figure below shows:

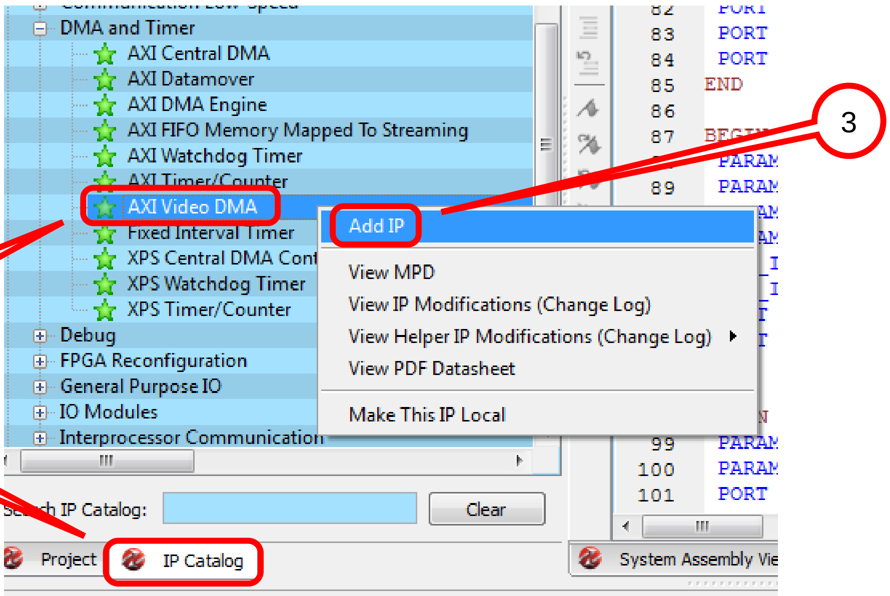


Figure 15. Adding a core from the IP catalog

The XPS Core Config window will appear.

2. In XPS Core Config window, in the "User" tab:
   - Check the "**Enable Asynchronous Clocks**" (for older versions, "**Primary Clock is Asynchronous**") option, do not leave it to "Auto"

   - For newer versions, expand the "**MM2S Channel Options**" group. Older versions will have the parameters under "**VDMA Options**".
     - Set "**Line Buffer Depth**" (older versions: "**MM2S Video Line Buffer Depth**") to **1024** Bytes
     - Set "**MM2S Video Line Buffer Threshold**" (older versions: "**MM2S Video Line Buffer Almost Empty Threshold**") to **512** Bytes

   - For newer versions, expand the "**S2MM Channel Options**" group. Older versions will have the parameter under "**VDMA Options**".
     - Set "**Line Buffer Depth**" (older versions: "**S2MM Video Line Buffer Depth**") to **1024** Bytes

Notes:

a. If the user does not need the HDMI receiver then the option "**Enable Channel**" from the "**S2MM Channel Options**" can be unchecked. For older versions, the "**Include**

S2MM Channel" under "**VDMA options"** can be set to "**FALSE"**. Also the parameter above regarding the S2MM channel does not have to be set.

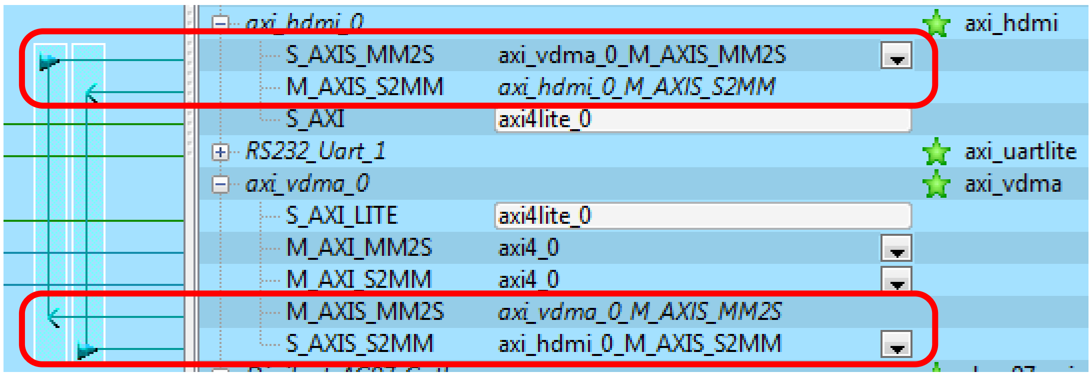
b. If the user does not need the HDMI transmitter then the option "**Enable Channel**" from the "**MM2S Channel Options**" can be unchecked. For older versions, the "**Include MM2S Channel**" under "**VDMA options"** can be set to "**FALSE"**. Also the parameters above regarding the MM2S channel do not have to be set.

3. In the same manner as at Step 1, from the "IP Catalog, Project Peripheral Repository0/Digilent", add an "AXI HDMI Receiver/Transmitter" core.

Notes:

a. If the user does not need the HDMI receiver then set the option "**Use HDMI Receiver Module**" to **FALSE.**
b. If the user does not need the HDMI transmitter then set the option "**Use HDMI Transmitter Module**" to **FALSE.**

4. Go to "System Assembly View", Bus Interfaces View.
Interconnect the MM2S and S2MM buses between the axi_hdmi_0 and axi_vdma_0 cores in the way shown in the picture below:



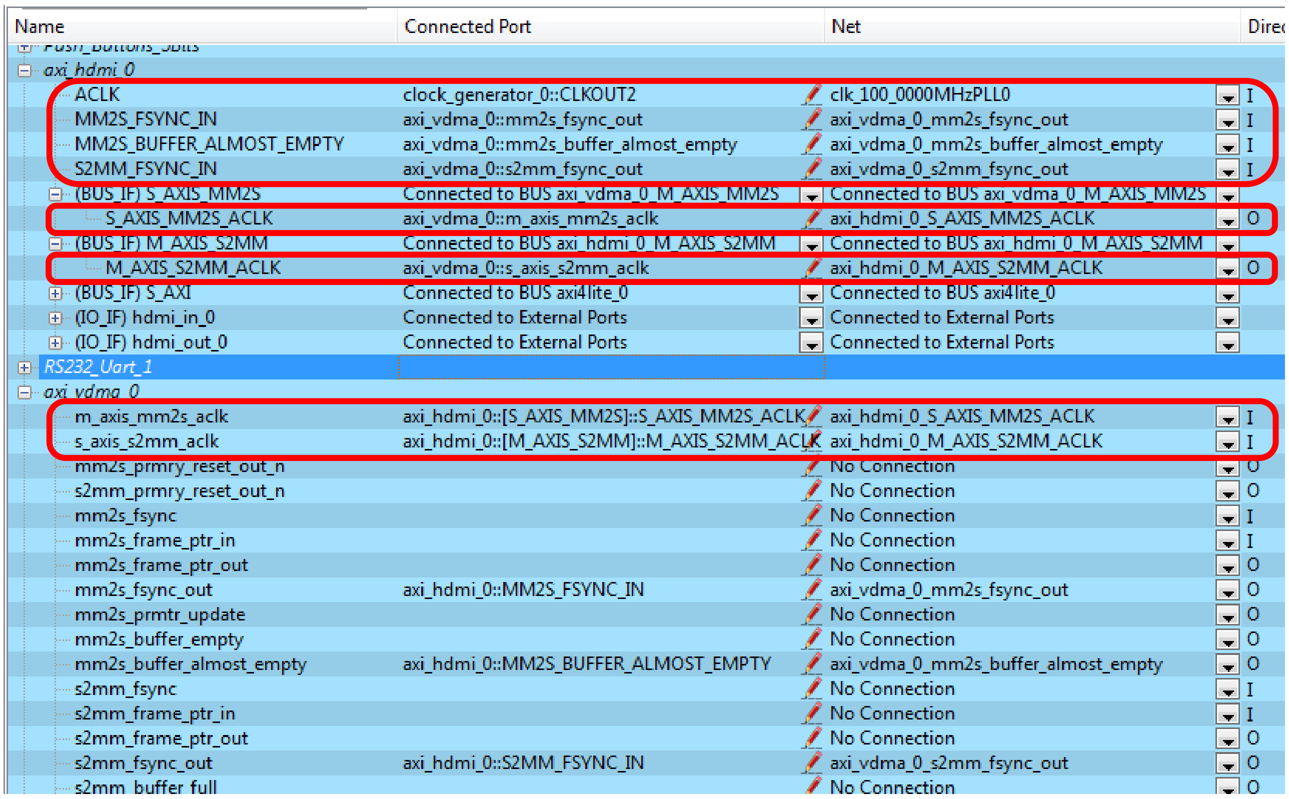
axi_hdmi_0		axi_hdmi
S_AXIS_MM2S	axi_vdma_0_M_AXIS_MM2S	
M_AXIS_S2MM	*axi_hdmi_0_M_AXIS_S2MM*	
S_AXI	axi4lite_0	
RS232_Uart_1		axi_uartlite
axi_vdma_0		axi_vdma
S_AXI_LITE	axi4lite_0	
M_AXI_MM2S	axi4_0	
M_AXI_S2MM	axi4_0	
M_AXIS_MM2S	*axi_vdma_0_M_AXIS_MM2S*	
S_AXIS_S2MM	axi_hdmi_0_M_AXIS_S2MM	

Figure 16. Interconnecting the MM2S and S2MM buses between the HDMI transmitter and the Video DMA

Note:

By default, the Video DMA instance name is set to "axi_vdma_0" and the HDMI Receiver/Transmitter instance name to "axi_hdmi_0" and will be referred further. If the user renames the instances, please refer to them using their new name.

5. Go to the Ports View. Expand the "axi_hdmi_0" instance.
   - Click on the pen icon on the port "**MM2S_FSYNC_IN**" and select core "**axi_vdma_0**", signal "**mm2s_fsync_out**", then click on the check icon to make the connection.
   - In a similar manner, connect port "**MM2S_BUFFER_ALMOST_EMPTY**" to signal "**mm2s_buffer_almost_empty**" of "**axi_vdma_0**".
   - Also connect "**S2MM_FSYNC_IN**" to signal "**s2mm_fsync_out**" of "**axi_vdma_0**", see the figure below for details.
   - Also connect port "**ACLK"** to "**clk_100_0000MHzPLL0"**
   - Expand "**(BUS_IF) S_AXIS_MM2S**" and "**(BUS_IF) M_AXIS_S2MM"** bus interfaces. Click on the combobox arrow in the "**Net**" column at the "**S_AXIS_MM2S_ACLK"** signal and choose "**New Connection"**. Create also a new connection for the "**M_AXIS_S2MM_ACLK"** signal, see the figure below.
6. Expand the "axi_vdma_0" instance.
   Using the combobox arrow, connect "**m_axis_mm2s_aclk"** to **"axi_hdmi_0_S_AXIS_MM2S_ACLK"** and **"s_axis_s2mm_aclk"** to "**axi_hdmi_0_M_AXIS_S2MM_ACLK"** (to the new connections created at step 5), see the figure below:



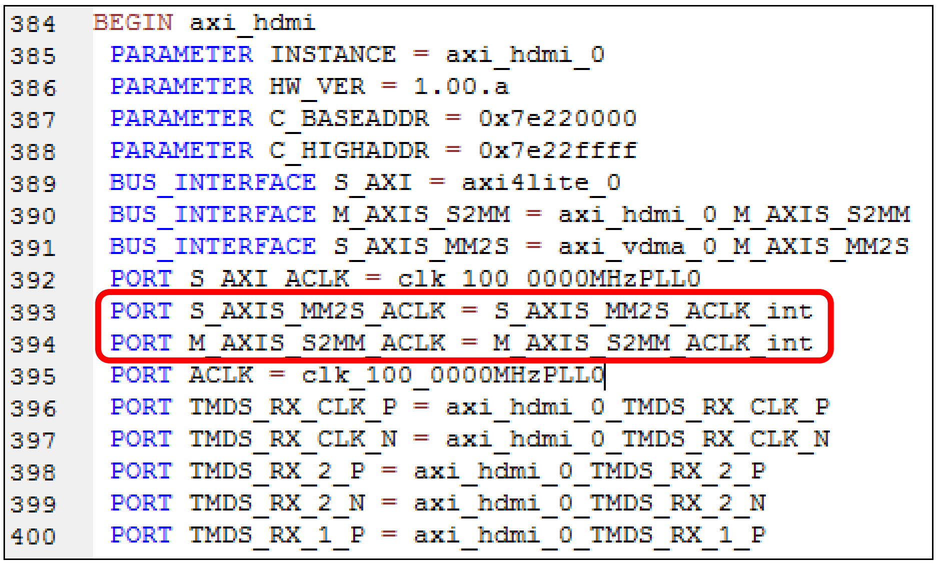
Name	Connected Port	Net	Direc
Push_Buttons_5Bits			
axi_hdmi_0			
ACLK	clock_generator_0::CLKOUT2	clk_100_0000MHzPLL0	I
MM2S_FSYNC_IN	axi_vdma_0::mm2s_fsync_out	axi_vdma_0_mm2s_fsync_out	I
MM2S_BUFFER_ALMOST_EMPTY	axi_vdma_0::mm2s_buffer_almost_empty	axi_vdma_0_mm2s_buffer_almost_empty	I
S2MM_FSYNC_IN	axi_vdma_0::s2mm_fsync_out	axi_vdma_0_s2mm_fsync_out	I
(BUS_IF) S_AXIS_MM2S	Connected to BUS axi_vdma_0_M_AXIS_MM2S	Connected to BUS axi_vdma_0_M_AXIS_MM2S	
S_AXIS_MM2S_ACLK	axi_vdma_0::m_axis_mm2s_aclk	axi_hdmi_0_S_AXIS_MM2S_ACLK	O
(BUS_IF) M_AXIS_S2MM	Connected to BUS axi_hdmi_0_M_AXIS_S2MM	Connected to BUS axi_hdmi_0_M_AXIS_S2MM	
M_AXIS_S2MM_ACLK	axi_vdma_0::s_axis_s2mm_aclk	axi_hdmi_0_M_AXIS_S2MM_ACLK	O
(BUS_IF) S_AXI	Connected to BUS axi4lite_0	Connected to BUS axi4lite_0	
(IO_IF) hdmi_in_0	Connected to External Ports	Connected to External Ports	
(IO_IF) hdmi_out_0	Connected to External Ports	Connected to External Ports	
RS232_Uart_1			
axi_vdma_0			
m_axis_mm2s_aclk	axi_hdmi_0::[S_AXIS_MM2S]::S_AXIS_MM2S_ACLK	axi_hdmi_0_S_AXIS_MM2S_ACLK	I
s_axis_s2mm_aclk	axi_hdmi_0::[M_AXIS_S2MM]::M_AXIS_S2MM_ACLK	axi_hdmi_0_M_AXIS_S2MM_ACLK	I
mm2s_prmry_reset_out_n		No Connection	O
s2mm_prmry_reset_out_n		No Connection	O
mm2s_fsync		No Connection	I
mm2s_frame_ptr_in		No Connection	I
mm2s_frame_ptr_out		No Connection	O
mm2s_fsync_out	axi_hdmi_0::MM2S_FSYNC_IN	axi_vdma_0_mm2s_fsync_out	O
mm2s_prmtr_update		No Connection	O
mm2s_buffer_empty		No Connection	O
mm2s_buffer_almost_empty	axi_hdmi_0::MM2S_BUFFER_ALMOST_EMPTY	axi_vdma_0_mm2s_buffer_almost_empty	O
s2mm_fsync		No Connection	I
s2mm_frame_ptr_in		No Connection	I
s2mm_frame_ptr_out		No Connection	O
s2mm_fsync_out	axi_hdmi_0::S2MM_FSYNC_IN	axi_vdma_0_s2mm_fsync_out	O
s2mm_buffer_full		No Connection	O

Figure 17. Additional signalconnections on the Digilent HDMI Controller and the Video DMA

Note:

Older versions of EDK might not allow new connections in the System Assembly View. In this case the new connections i.e. step 5, last line and step 6 will have to be made using the MHS file, as shown below:

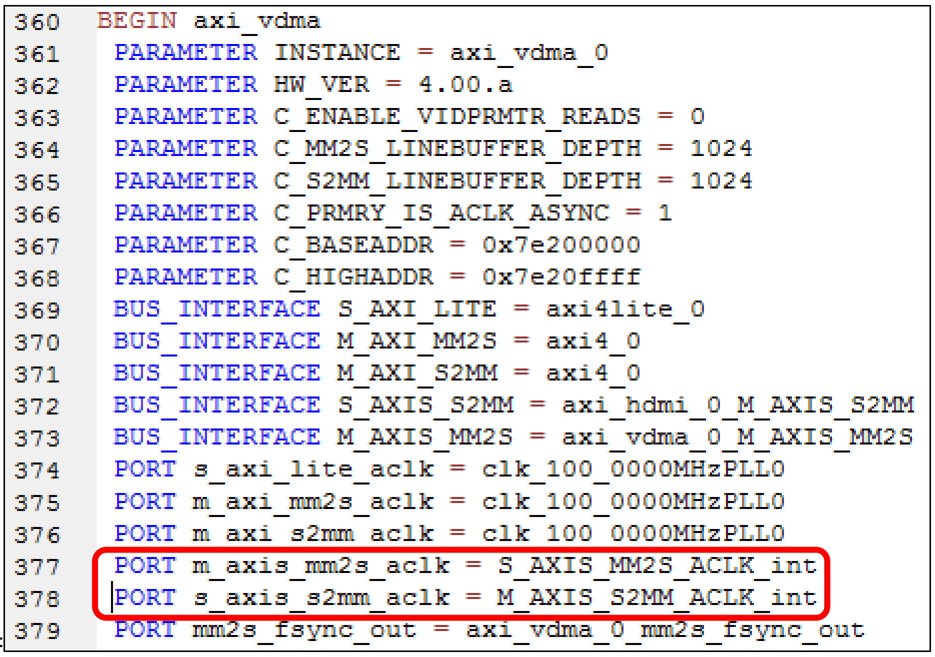
- From the "Project" tab open "system.mhs" and locate the line "BEGIN axi_hdmi". Before the "END" statement, add the following lines:
  `PORT S_AXIS_MM2S_ACLK = S_AXIS_MM2S_ACLK_int`
  `PORT M_AXIS_S2MM_ACLK = M_AXIS_S2MM_ACLK_int`, as shown in the figure below:



```
BEGIN axi_hdmi
 PARAMETER INSTANCE = axi_hdmi_0
 PARAMETER HW_VER = 1.00.a
 PARAMETER C_BASEADDR = 0x7e220000
 PARAMETER C_HIGHADDR = 0x7e22ffff
 BUS_INTERFACE S_AXI = axi4lite_0
 BUS_INTERFACE M_AXIS_S2MM = axi_hdmi_0_M_AXIS_S2MM
 BUS_INTERFACE S_AXIS_MM2S = axi_vdma_0_M_AXIS_MM2S
 PORT S_AXI_ACLK = clk_100_0000MHzPLL0
 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
 PORT TMDS_RX_CLK_P = axi_hdmi_0_TMDS_RX_CLK_P
 PORT TMDS_RX_CLK_N = axi_hdmi_0_TMDS_RX_CLK_N
 PORT TMDS_RX_2_P = axi_hdmi_0_TMDS_RX_2_P
 PORT TMDS_RX_2_N = axi_hdmi_0_TMDS_RX_2_N
 PORT TMDS_RX_1_P = axi_hdmi_0_TMDS_RX_1_P
```

Figure 18. Making new connections for the Digilent HDMI Controller in the MHS file

- Look for the line “BEGIN axi_vdma” and add the following lines:
  PORT m_axis_mm2s_aclk = S_AXIS_MM2S_ACLK_int
  PORT s_axis_s2mm_aclk = M_AXIS_S2MM_ACLK_int, as shown in the figure below:

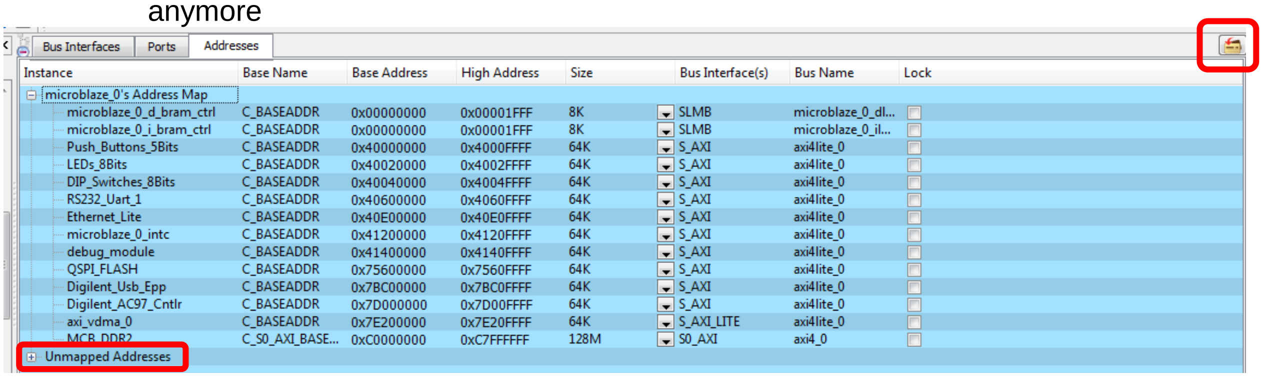


```
BEGIN axi_vdma
 PARAMETER INSTANCE = axi_vdma_0
 PARAMETER HW_VER = 4.00.a
 PARAMETER C_ENABLE_VIDPRMTR_READS = 0
 PARAMETER C_MM2S_LINEBUFFER_DEPTH = 1024
 PARAMETER C_S2MM_LINEBUFFER_DEPTH = 1024
 PARAMETER C_PRMRY_IS_ACLK_ASYNC = 1
 PARAMETER C_BASEADDR = 0x7e200000
 PARAMETER C_HIGHADDR = 0x7e20ffff
 BUS_INTERFACE S_AXI_LITE = axi4lite_0
 BUS_INTERFACE M_AXI_MM2S = axi4_0
 BUS_INTERFACE M_AXI_S2MM = axi4_0
 BUS_INTERFACE S_AXIS_S2MM = axi_hdmi_0_M_AXIS_S2MM
 BUS_INTERFACE M_AXIS_MM2S = axi_vdma_0_M_AXIS_MM2S
 PORT s_axi_lite_aclk = clk_100_0000MHzPLL0
 PORT m_axi_mm2s_aclk = clk_100_0000MHzPLL0
 PORT m_axi_s2mm_aclk = clk_100_0000MHzPLL0
 PORT m_axis_mm2s_aclk = S_AXIS_MM2S_ACLK_int
 PORT s_axis_s2mm_aclk = M_AXIS_S2MM_ACLK_int
 PORT mm2s_fsync_out = axi_vdma_0_mm2s_fsync_out
```

Figure 19. Making the new connections created above for the Video DMA in the MHS file

7. Go to the System Assembly View -> Adresses tab. In older versions of EDK it might happen that adresses for the axi_hdmi_0 core were not automatically generated.

   If there is an “Unmapped Addresses group, then click on the “Regenerate Addresses” icon, as shown in the figure below, in order to generate adresses for the axi_hdmi core, see the figure below.
   After regenerating addresses, the “Unmapped Addresses” group should not be shown anymore

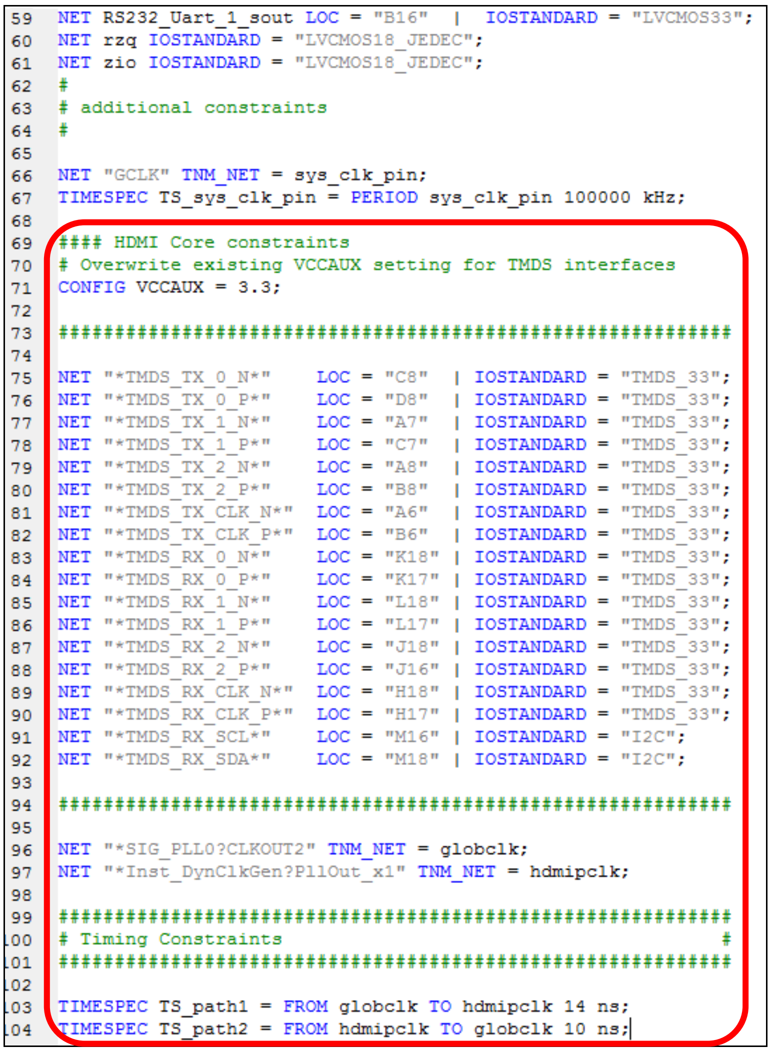


Bus Interfaces | Ports | Addresses

Instance	Base Name	Base Address	High Address	Size	Bus Interface(s)	Bus Name	Lock
microblaze_0's Address Map							
microblaze_0_d_bram_ctrl	C_BASEADDR	0x00000000	0x00001FFF	8K	SLMB	microblaze_0_dl...	☐
microblaze_0_i_bram_ctrl	C_BASEADDR	0x00000000	0x00001FFF	8K	SLMB	microblaze_0_il...	☐
Push_Buttons_5Bits	C_BASEADDR	0x40000000	0x4000FFFF	64K	S_AXI	axi4lite_0	☐
LEDs_8Bits	C_BASEADDR	0x40020000	0x4002FFFF	64K	S_AXI	axi4lite_0	☐
DIP_Switches_8Bits	C_BASEADDR	0x40040000	0x4004FFFF	64K	S_AXI	axi4lite_0	☐
RS232_Uart_1	C_BASEADDR	0x40600000	0x4060FFFF	64K	S_AXI	axi4lite_0	☐
Ethernet_Lite	C_BASEADDR	0x40E00000	0x40E0FFFF	64K	S_AXI	axi4lite_0	☐
microblaze_0_intc	C_BASEADDR	0x41200000	0x4120FFFF	64K	S_AXI	axi4lite_0	☐
debug_module	C_BASEADDR	0x41400000	0x4140FFFF	64K	S_AXI	axi4lite_0	☐
QSPI_FLASH	C_BASEADDR	0x75600000	0x7560FFFF	64K	S_AXI	axi4lite_0	☐
Digilent_Usb_Epp	C_BASEADDR	0x7BC00000	0x7BC0FFFF	64K	S_AXI	axi4lite_0	☐
Digilent_AC97_Cntlr	C_BASEADDR	0x7D000000	0x7D00FFFF	64K	S_AXI	axi4lite_0	☐
axi_vdma_0	C_BASEADDR	0x7E200000	0x7E20FFFF	64K	S_AXI_LITE	axi4lite_0	☐
MCB_DDR2	C_S0_AXI_BASE...	0xC0000000	0xC7FFFFFF	128M	S0_AXI	axi4_0	☐
Unmapped Addresses							

Figure 20. Regenerate address space for the axi_hdmi core

8. From the “Project” tab open “system.ucf”. From the downloaded AXI Board Support Files, from the path **../Atlys_AXI_BSB_Support/lib/Digilent/boards/Digilent_Atlys/data**” folder open the **Digilent_HDMI_axi_hdmi_v1_00_a.ucf** file, using a text editor. Also XPS can be used to open the specific file. After the file is opened, copy its contents to the end of the **system.ucf** file, see the figure below.
   Don’t forget to add an extra line to the last .ucf line.



```
NET RS232_Uart_1_sout LOC = "B16"  |  IOSTANDARD = "LVCMOS33";
NET rzq IOSTANDARD = "LVCMOS18_JEDEC";
NET zio IOSTANDARD = "LVCMOS18_JEDEC";
#
# additional constraints
#

NET "GCLK" TNM_NET = sys_clk_pin;
TIMESPEC TS_sys_clk_pin = PERIOD sys_clk_pin 100000 kHz;

#### HDMI Core constraints
# Overwrite existing VCCAUX setting for TMDS interfaces
CONFIG VCCAUX = 3.3;

##############################################################

NET "*TMDS_TX_0_N*"    LOC = "C8"   | IOSTANDARD = "TMDS_33";
NET "*TMDS_TX_0_P*"    LOC = "D8"   | IOSTANDARD = "TMDS_33";
NET "*TMDS_TX_1_N*"    LOC = "A7"   | IOSTANDARD = "TMDS_33";
NET "*TMDS_TX_1_P*"    LOC = "C7"   | IOSTANDARD = "TMDS_33";
NET "*TMDS_TX_2_N*"    LOC = "A8"   | IOSTANDARD = "TMDS_33";
NET "*TMDS_TX_2_P*"    LOC = "B8"   | IOSTANDARD = "TMDS_33";
NET "*TMDS_TX_CLK_N*"  LOC = "A6"   | IOSTANDARD = "TMDS_33";
NET "*TMDS_TX_CLK_P*"  LOC = "B6"   | IOSTANDARD = "TMDS_33";
NET "*TMDS_RX_0_N*"    LOC = "K18"  | IOSTANDARD = "TMDS_33";
NET "*TMDS_RX_0_P*"    LOC = "K17"  | IOSTANDARD = "TMDS_33";
NET "*TMDS_RX_1_N*"    LOC = "L18"  | IOSTANDARD = "TMDS_33";
NET "*TMDS_RX_1_P*"    LOC = "L17"  | IOSTANDARD = "TMDS_33";
NET "*TMDS_RX_2_N*"    LOC = "J18"  | IOSTANDARD = "TMDS_33";
NET "*TMDS_RX_2_P*"    LOC = "J16"  | IOSTANDARD = "TMDS_33";
NET "*TMDS_RX_CLK_N*"  LOC = "H18"  | IOSTANDARD = "TMDS_33";
NET "*TMDS_RX_CLK_P*"  LOC = "H17"  | IOSTANDARD = "TMDS_33";
NET "*TMDS_RX_SCL*"    LOC = "M16"  | IOSTANDARD = "I2C";
NET "*TMDS_RX_SDA*"    LOC = "M18"  | IOSTANDARD = "I2C";

##############################################################

NET "*SIG_PLL0?CLKOUT2" TNM_NET = globclk;
NET "*Inst_DynClkGen?PllOut_x1" TNM_NET = hdmipclk;

##############################################################
# Timing Constraints                                         #
##############################################################

TIMESPEC TS_path1 = FROM globclk TO hdmipclk 14 ns;
TIMESPEC TS_path2 = FROM hdmipclk TO globclk 10 ns;
```

Figure 21. Adding the ucf constraints to the system.ucf. file

9. Obviously, save the .ucf file. Now you can start generating the bitstream for the project.