# Atlys BSB Support Files for AXI-based EDK 13.3 Designs

**Revision:** December 15, 2011

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

## Note:

**This document and the board support files refer to AXI – based EDK 13.3 designs only. For EDK 13.2 designs use the board support files and document from "Atlys_BSB_Support_v_3_2.zip", downloadable from the Digilent website.**

## Overview

This package will integrate board support for the Atlys Spartan-6 FPGA Development Board into Xilinx EDK tools. It includes board definition files for creating AXI-based MicroBlaze embedded designs in the Base System Builder (BSB). It also includes cores for custom peripherals such as the Digilent USB-EPP interface, the 16MB Quad SPI Flash Memory, HDMI and AC97. 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. The currently supported cores are outlined in Table 1.

**Table 1. BSB Supported Peripherals**

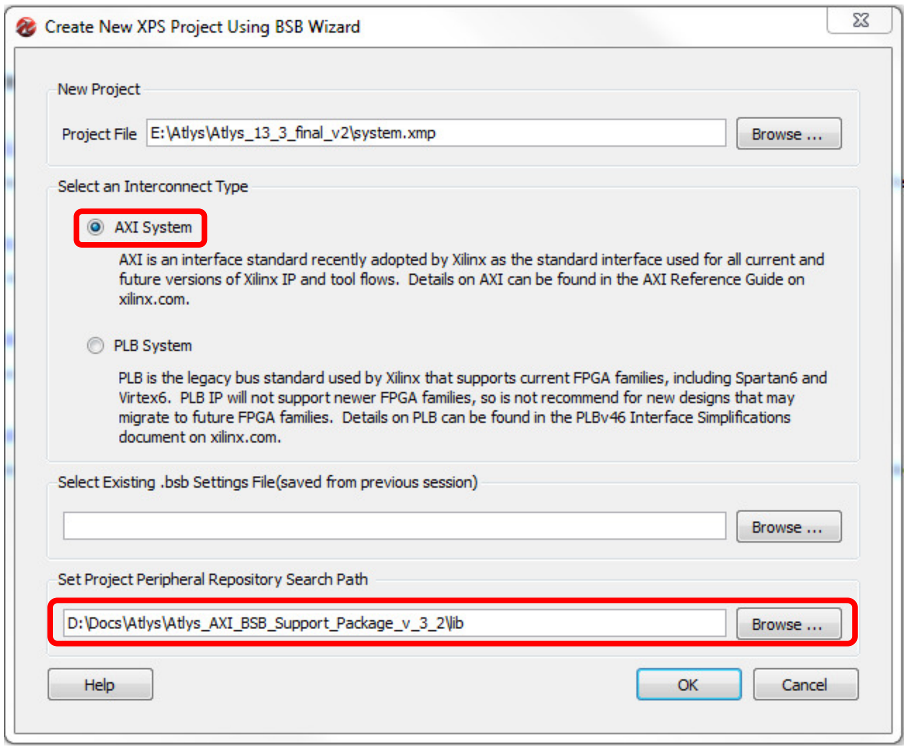
| Peripheral | Supported Interface | Core name(s) | Notes |
|---|---|---|---|
| 128MB DDR (cached) | AXI4 | axi_s6_ddrx | -- |
| 8 User Switches | AXI4-Lite | axi_gpio | -- |
| 5 User Push Buttons | AXI4-Lite | axi_gpio | -- |
| 8 LED outputs | AXI4-Lite | axi_gpio | -- |
| UART | AXI4-Lite | axi_uartlite/axi_uart16550 | -- |
| 16-MB Quad-SPI PCM | AXI4-Lite | d_qspi_axi | Custom core; supports 1X, 2X and 4X modes |
| AC-97 | AXI4-Lite | d_ac97_axi | Custom core |
| HDMI | AXI4-Lite, AXI4-Stream | axi_hdmi | Custom core |
| USB-EPP | AXI4-Lite | d_usb_epp_dstm_axi | Custom core |
| 10/100/1000 Mbps PHY | AXI4-Lite | axi_ethernet | Requires license |
| 10/100 Mbps PHY | AXI4-Lite | axi_ethernetlite | -- |

*For additional information on using these cores, please refer to their PDF datasheets*

## Using the BSB Support Files

**Note:** In order to use the custom cores from the BSB Support Files, first install the "Digilent_AXI_IPCore_Support" package (from Digilent website).
The "Digilent_AXI_IPCore_Support" adds core and bus definitions in IPXACT standard to the EDK installation directory.

1. Start Platform Studio and create a new project using Base System Builder by selecting "Create New Project Using Base System Builder". Choose AXI system in the "Create New XPS Project Using BSB Wizard" window.
2. 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, then press OK.
   The BSB window should look like in figure below:


Create New XPS Project Using BSB Wizard
New Project
Browse ...
Select an Interconnect Type
Select Existing .bsb Settings File(saved from previous session)
Browse ...
Set Project Peripheral Repository Search Path
Browse ...
Help
OK
Cancel

Project File E:\Atlys\Atlys_13_3_final_v2\system.xmp

[✓ 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.

D:\Docs\Atlys\Atlys_AXI_BSB_Support_Package_v_3_2\lib

**Figure 1. BSB window with specifying the Peripheral Repository Search Path**

Click OK. You should now be able to select the Digilent Spartan-6 Atlys as your development board further in the Board Selection window.

## Using Interrupt for the Digilent USB-EPP interface

EPP requests for the USB-EPP interface come from the USB port. If there is no answer in 100 ms, the PC application will signal a timeout. Therefore, it is recommended that Epp requests are handled with an interrupt service routine instead of continuously polling the interface status.

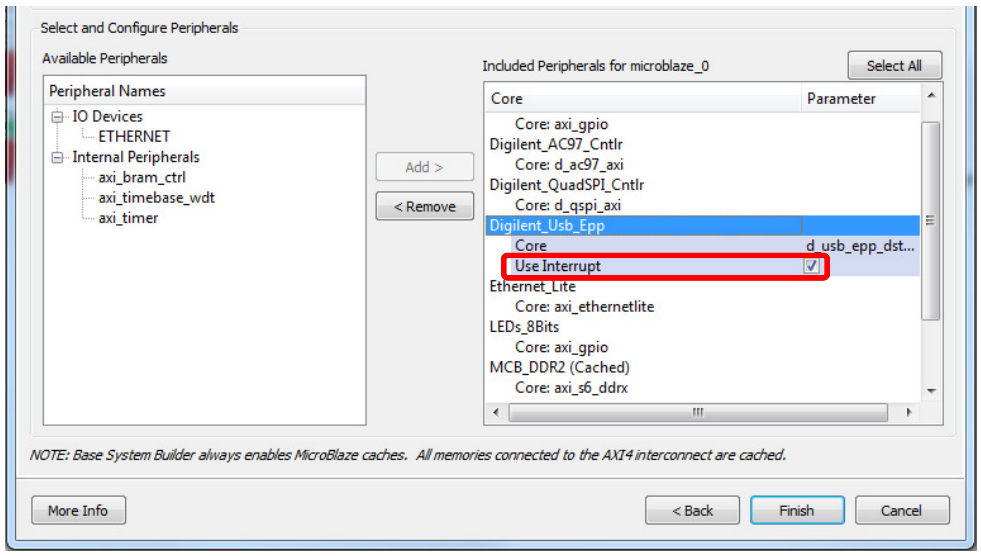
The demo applications include examples for using the USB-EPP interface in both polling and interrupt mode.

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.

For example, to select the "Use Interrupt" option for the Digilent_Usb_Epp peripheral in BSB, in the "Peripheral Configuration" window click on the "Digilent_Usb_Epp" peripheral and select "Use Interrupt" like is shown in Figure 2 below.

Note that because the USB-EPP interface is a custom core, its interrupt output is not automatically connected by BSB and the user will have to make the connection manually, as described in the section "Using the Digilent USP-EPP interface controller" below.


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 |
|---|---|
| Core: axi_gpio | |
| Digilent_AC97_Cntlr | |
| Core: d_ac97_axi | |
| Digilent_QuadSPI_Cntlr | |
| Core: d_qspi_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 | |

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

**Figure 2. Selecting the "Use Interrupt" option for the Digilent_Usb_Epp interface in Base System Builder**

## Using the custom cores

Although the "Digilent_AXI_IPCore_Support" package installs the IPXACT definitions for custom cores, the internal and external connections for the custom cores are not automatically made by BSB (except for AXI4-Lite bus connections). Therefore, the user must manually make the connections when using any of the custom cores. The following sections describe how to manually create these connections for each custom core.

## Using the Digilent USB-EPP interface controller

If the Digilent USB-EPP interface controller is present in the system, the following connections have to be made:

a. Go to the "Ports" tab in XPS and expand *Digilent_Usb_Epp* core instance. Make the ports under "(IO_IF) usb_epp_ext" external by selecting all signals (with Ctrl key pressed) and right clicking over the selection.

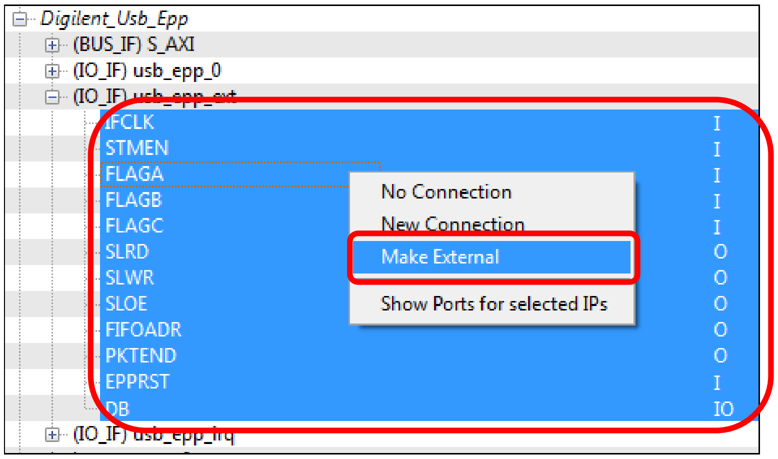
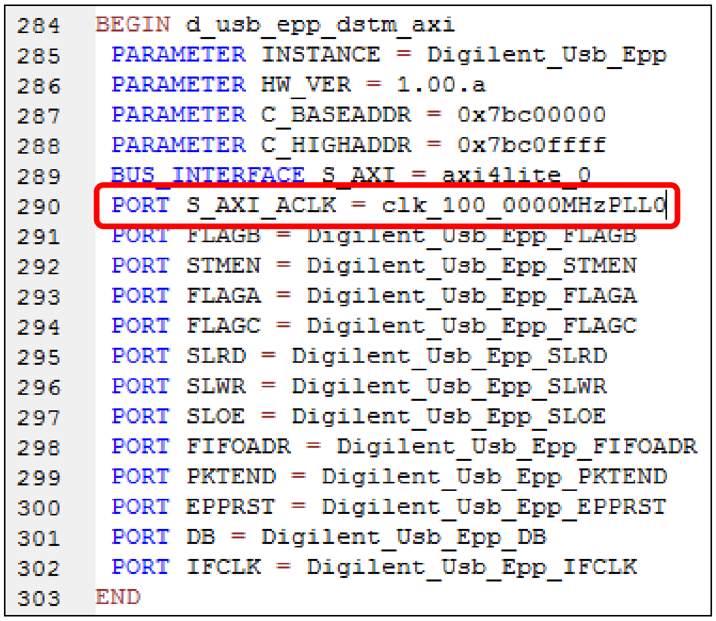


Figure 3. Making external connections on Digilent_Usb_Epp

b. From the "Project" tab open *system.mhs* and scroll down to where d_usb_epp_dstm_axi controller is. Replace "**PORT S_AXI_ACLK = net_gnd**" with "**PORT S_AXI_ACLK = clk_100_0000MHzPLL0**".



```
BEGIN d_usb_epp_dstm_axi
 PARAMETER INSTANCE = Digilent_Usb_Epp
 PARAMETER HW_VER = 1.00.a
 PARAMETER C_BASEADDR = 0x7bc00000
 PARAMETER C_HIGHADDR = 0x7bc0ffff
 BUS_INTERFACE S_AXI = axi4lite_0
 PORT S_AXI_ACLK = clk_100_0000MHzPLL0
 PORT FLAGB = Digilent_Usb_Epp_FLAGB
 PORT STMEN = Digilent_Usb_Epp_STMEN
 PORT FLAGA = Digilent_Usb_Epp_FLAGA
 PORT FLAGC = Digilent_Usb_Epp_FLAGC
 PORT SLRD = Digilent_Usb_Epp_SLRD
 PORT SLWR = Digilent_Usb_Epp_SLWR
 PORT SLOE = Digilent_Usb_Epp_SLOE
 PORT FIFOADR = Digilent_Usb_Epp_FIFOADR
 PORT PKTEND = Digilent_Usb_Epp_PKTEND
 PORT EPPRST = Digilent_Usb_Epp_EPPRST
 PORT DB = Digilent_Usb_Epp_DB
 PORT IFCLK = Digilent_Usb_Epp_IFCLK
END
```

Figure 4. Connecting the clock

If the "Use Interrupt" option was selected in BSB for the Digilent USB-EPP controller interface, the USB-EPP Interrupt output has to be connected to the interrupt controller:

c. On the "Ports" tab, in XPS, expand *microblaze_intc_0* instance and click on the "Net" of "Intr" port. An "Interrupt Connection Dialog" appears. Select the interrupt port: **Digilent_Usb_Epp_IRQ_EPP**, shown in Figure 5 below, clock on the arrow to add the connection then press OK.

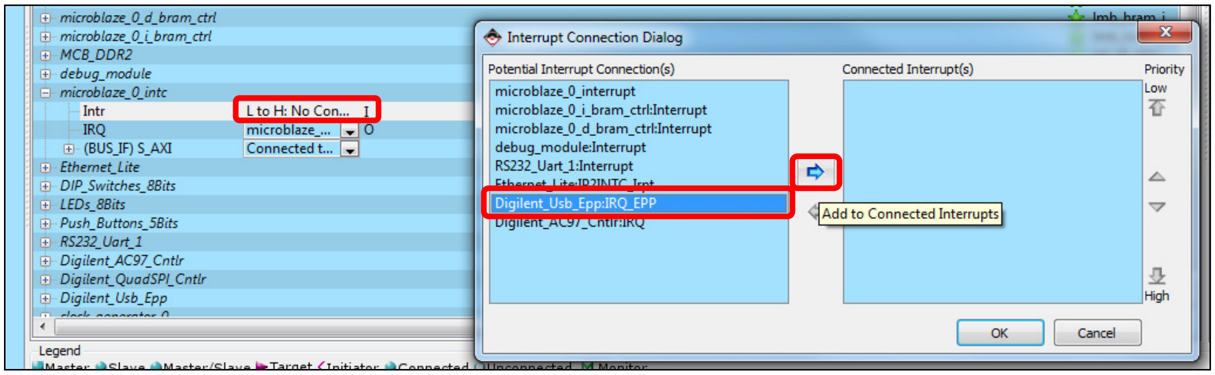


Figure 5. Connecting interrupt port to the interrupt controller

## Connecting Digilent Quad-SPI controller

a. Expand *Digilent_QuadSPI_Cntlr* core instance. Make external the ports under “(IO_IF) qspi_ext” by selecting all signals (with Ctrl key pressed) and right clicking over the selection.

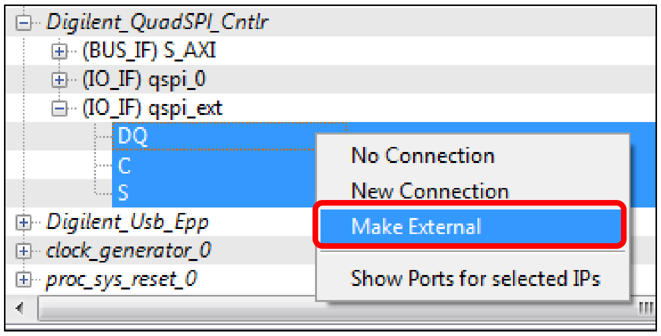
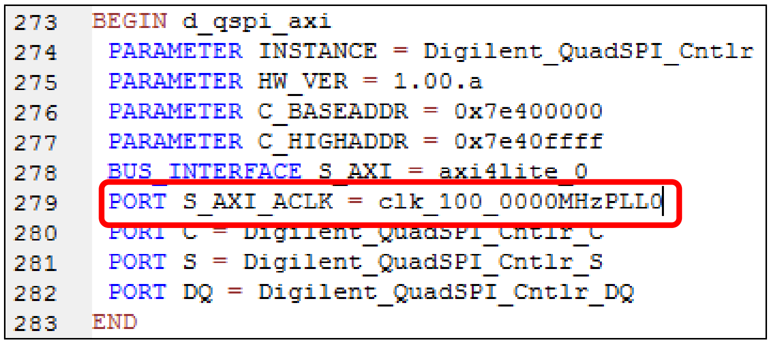


Figure 6. Making external connections on Digilent_QuadSPI_Cntlr

b. From the “Project” tab open *system.mhs* and scroll down to where d_qspi_axi controller is. Replace “**PORT S_AXI_ACLK = net_gnd**” with “**PORT S_AXI_ACLK = clk_100_0000MHzPLL0**”.



```
BEGIN d_qspi_axi
 PARAMETER INSTANCE = Digilent_QuadSPI_Cntlr
 PARAMETER HW_VER = 1.00.a
 PARAMETER C_BASEADDR = 0x7e400000
 PARAMETER C_HIGHADDR = 0x7e40ffff
 BUS_INTERFACE S_AXI = axi4lite_0
 PORT S_AXI_ACLK = clk_100_0000MHzPLL0
 PORT C = Digilent_QuadSPI_Cntlr_C
 PORT S = Digilent_QuadSPI_Cntlr_S
 PORT DQ = Digilent_QuadSPI_Cntlr_DQ
END
```

Figure 7. Connecting the clock

## Connecting Digilent AC97 controller

a. Expand *Digilent_AC97_Cntlr* core instance. Make external the ports under "(IO_IF) ac97_ext" by selecting all signals (with Ctrl key pressed) and right clicking over the selection.

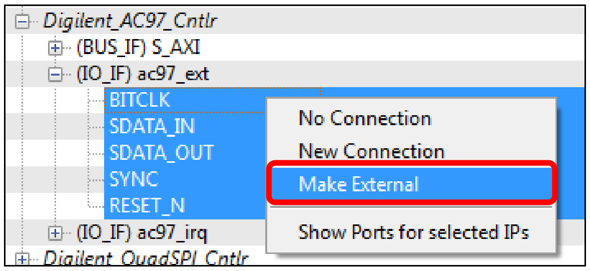
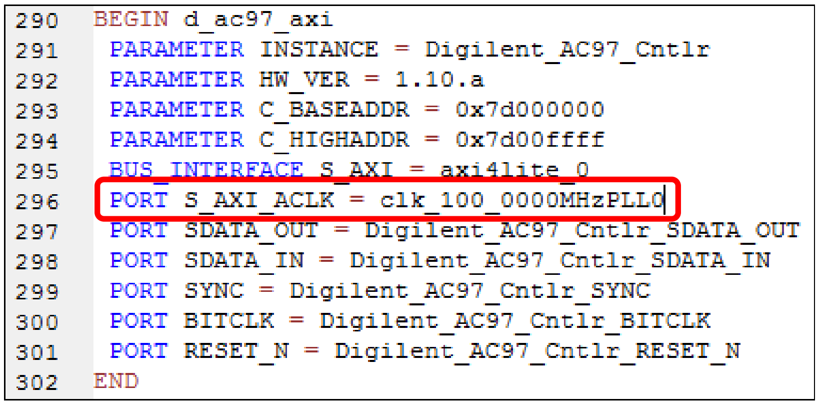


**Figure 8. Making external connections on Digilent_AC97_Cntlr**

b. From the "Project" tab open *system.mhs* and scroll down to where d_ac97_axi controller is. Replace "**PORT S_AXI_ACLK = net_gnd**" with "**PORT S_AXI_ACLK = clk_100_0000MHzPLL0**".



```
BEGIN d_ac97_axi
 PARAMETER INSTANCE = Digilent_AC97_Cntlr
 PARAMETER HW_VER = 1.10.a
 PARAMETER C_BASEADDR = 0x7d000000
 PARAMETER C_HIGHADDR = 0x7d00ffff
 BUS_INTERFACE S_AXI = axi4lite_0
 PORT S_AXI_ACLK = clk_100_0000MHzPLL0
 PORT SDATA_OUT = Digilent_AC97_Cntlr_SDATA_OUT
 PORT SDATA_IN = Digilent_AC97_Cntlr_SDATA_IN
 PORT SYNC = Digilent_AC97_Cntlr_SYNC
 PORT BITCLK = Digilent_AC97_Cntlr_BITCLK
 PORT RESET_N = Digilent_AC97_Cntlr_RESET_N
END
```

**Figure 9. Connecting the clock**

## Using the Digilent HDMI controller

In order to use the Digilent HDMI controller, the AXI Video DMA core must be used:

a. Add *AXI Video DMA* core. In XPS Core Config window, under the "User" tab set "**MM2S Video Line Buffer Depth**" to **1024** Bytes and "**MM2S Video Line Buffer Almost Empty Threshold**" to **512** Bytes. Also, set "**S2MM Video Line Buffer Depth**" to **1024** Bytes. Check "**Primary clock Is Asynchronous**" option too.

   ***Note***: if the user only needs the HDMI transmitter then option "**Include S2MM Channel**" can be unchecked. If the user uses only the HDMI receiver then option "**Include MM2S Channel**" can be unchecked.

b. Add *AXI HDMI Receiver/Transmitter*, under Digilent from the "IP Catalog" tab.

   Note: if the user uses only the HDMI transmitter, then option "**Use HDMI Receiver**" must be set to **FALSE**. If the user only need the HDMI Receiver then option "**Use HDMI Transmitter**" must be set to **FALSE**.

c. In the "Bus Interfaces" tab connect bus "**M_AXIS_S2MM**" of *axi_hdmi_0* to "**S_AXIS_S2MM**" of *axi_vdma_0* and "**M_AXIS_MM2S**" of *axi_vdma_0* to "**S_AXIS_MM2S**" of *axi_hdmi_0* instance.

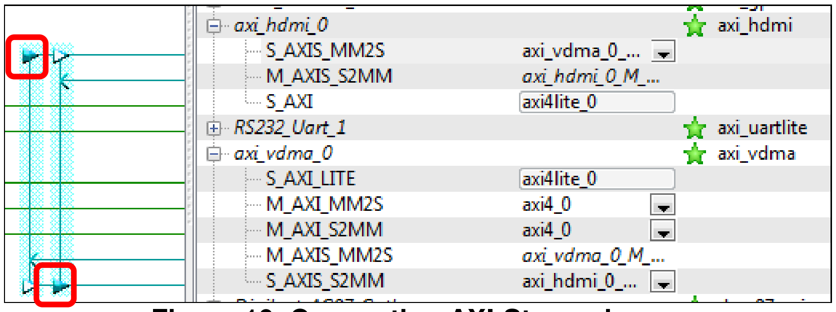


**Figure 10. Connecting AXI-Stream buses**

d. Go to the "Ports" tab in XPS and expand *axi_hdmi_0 instance*. Click on the pen on port "**MM2S_FSYNC_IN**" and select core "**axi_vdma_0**", signal "**mm2s_fsync_out**". Connect port "**MM2S_BUFFER_ALMOST_EMPTY**" to signal "**mm2s_buffer_almost_empty**" of "**axi_vdma_0**", and "**S2MM_FSYNC_IN**" to signal "**s2mm_fsync_out**" of "**axi_vdma_0**".

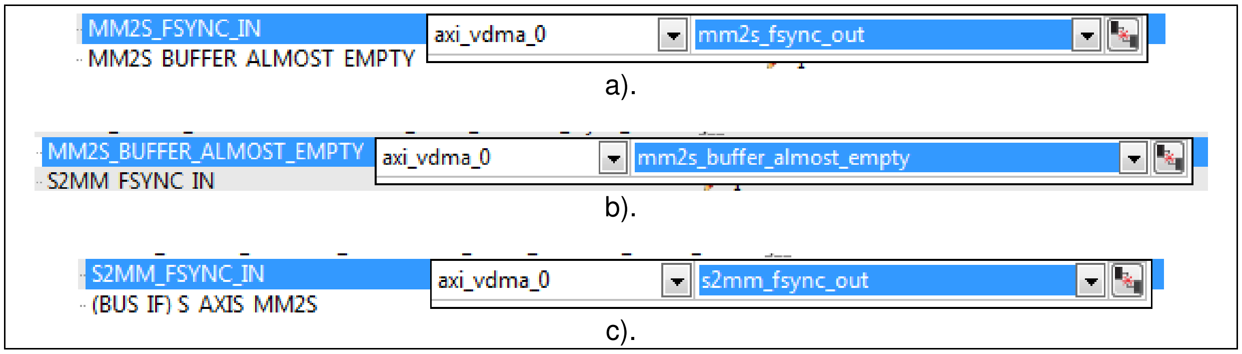
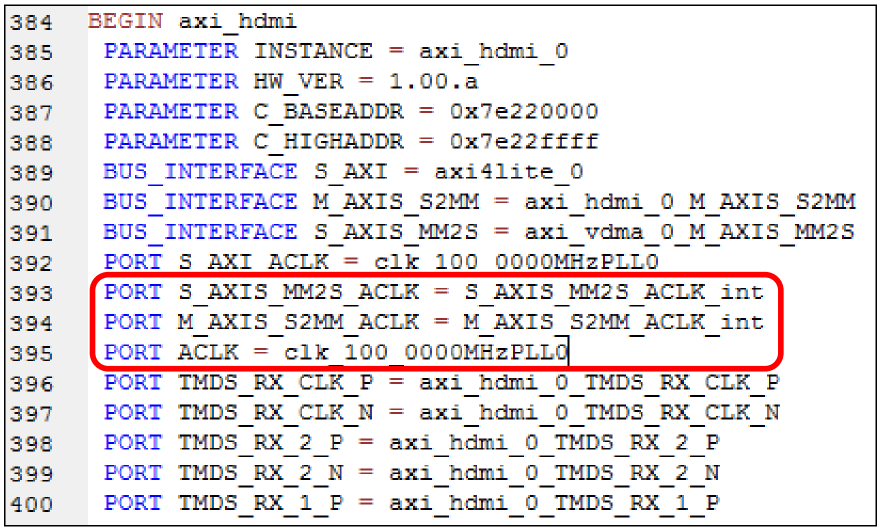


**Figure 11. Connecting additional signal on Digilent HDMI**

e. From the "Project" tab open *system.mhs* and scroll down to where axi_hdmi controller is located. Add the following lines:

```
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
```

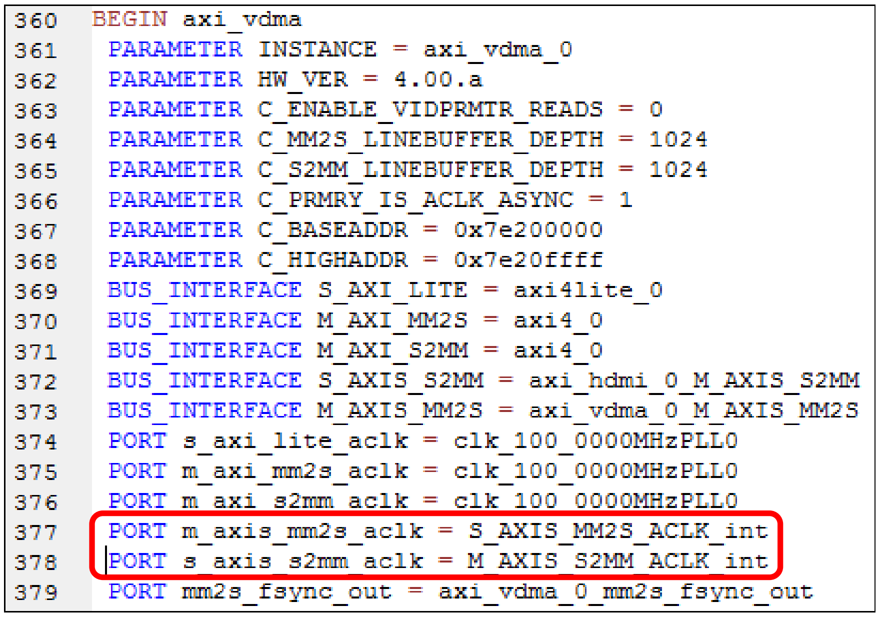


```
BEGIN axi_hdmi
 PARAMETER INSTANCE = axi_hdmi_0
 PARAMETER HW_VER = 1.00.a
 PARAMETER C_BASEADDR = 0x7e220000
 PARAMETER C_HIGHADDR = 0x7e22ffff
 BUS_INTERFACE S_AXI = axi4lite_0
 BUS_INTERFACE M_AXIS_S2MM = axi_hdmi_0_M_AXIS_S2MM
 BUS_INTERFACE S_AXIS_MM2S = axi_vdma_0_M_AXIS_MM2S
 PORT S_AXI_ACLK = 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 12. Creating new clock connections**

f. Scroll to the axi_vdma core, in *system.mhs* and add the following lines:

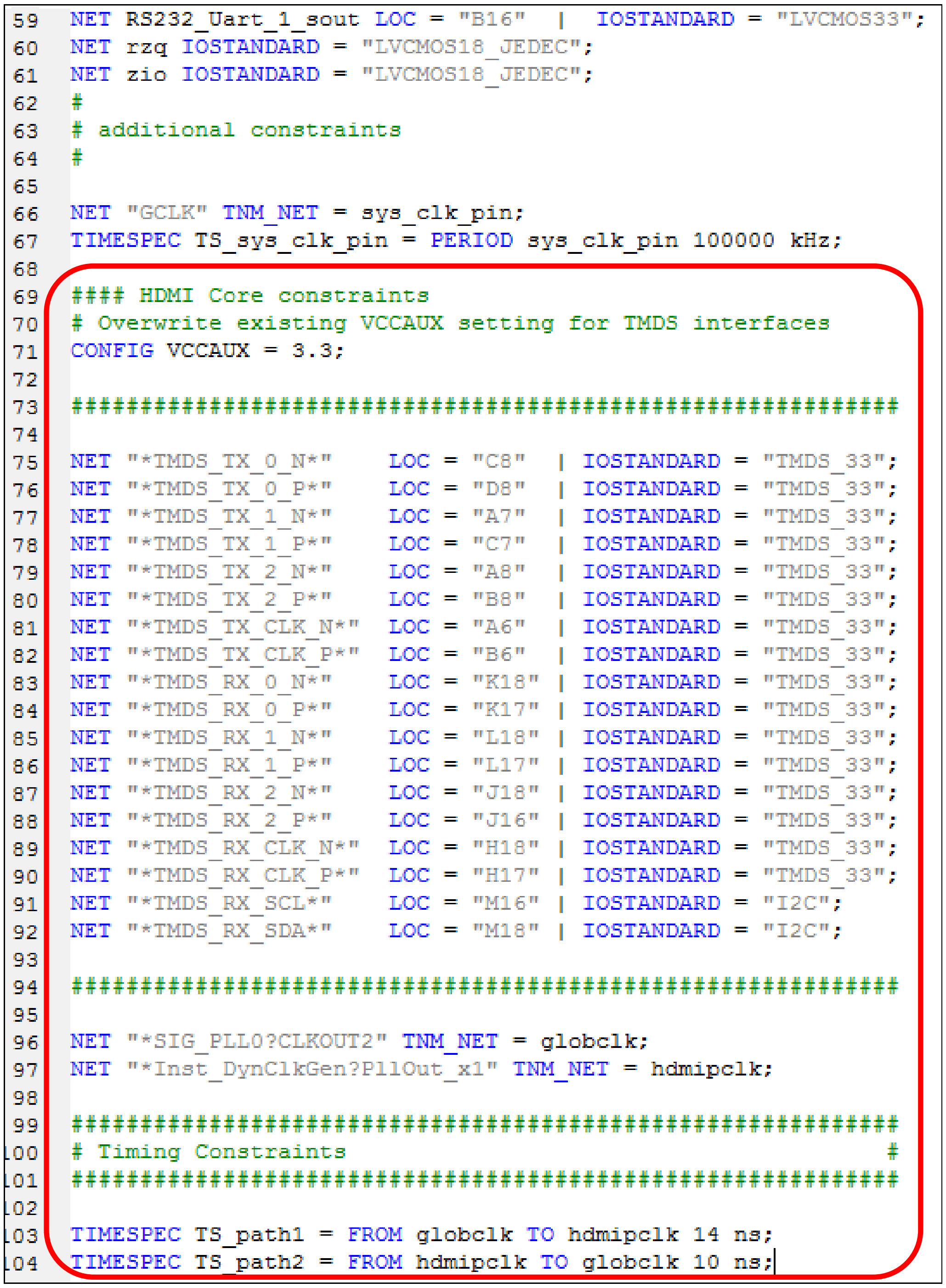
```
PORT m_axis_mm2s_aclk = S_AXIS_MM2S_ACLK_int
PORT s_axis_s2mm_aclk = M_AXIS_S2MM_ACLK_int
```



```
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 13. Connecting the clocks to VDMA**

g. From the “Project” tab open *system.ucf*. From the downloaded “Atlys_BSB_Support/Atlys_AXI_BSB_Support/lib/Digilent/boards/Digilent_Atlys/data” open *Digilent_HDMI_axi_hdmi_v1_00_a.ucf* file and copy its contents to *system.ucf*.



```
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 14. Adding additional constraints**