XAPP493 (v2.0) September 16, 2011

DisplayPort Source Core Reference Design

Author: Arun Ananthapadmanaban and Vamsi Krishna

Summary

This application note describes the implementation of a DisplayPort™ source core and policy maker reference design targeted for the Spartan®-6 FPGA Consumer Video Kit (CVK) [Ref 1] and Virtex®-6 FPGA ML605 Evaluation Kit [Ref 2] with Avnet Digital Visual Interface (DVI) I/O FPGA Mezzanine Connector (FMC) Module [Ref 3].

Introduction

The purpose of the reference design is to implement the DisplayPort source design and associated software policy maker in a MicroBlaze™ processor. The DisplayPort source Policy Maker communicates to both the transmitter (source) and the receiver (sink) to perform several tasks such as initialization of GTP or GTX transceiver links, probing of registers, and other features useful for bring-up and use of the core. In this specific reference system, the DisplayPort source Policy Maker is implemented at the source side. The mechanism for this communication to the sink side is over the auxiliary channel, as described in the VESA DisplayPort Standard [Ref 4].

The reference design included with the application note encompasses the DisplayPort source policy maker, a DisplayPort source core generated using the CORE Generator™ tool, and a video pattern generator to create video data for transmission over the DisplayPort link. The focus of this application note is on the reference hardware and the policy maker implementation. The block diagram of the test system is shown in Figure 1.

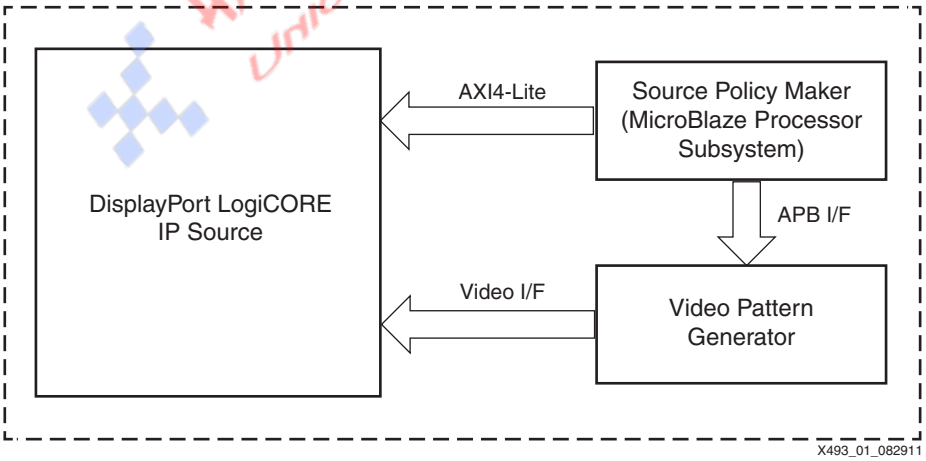


Figure 1: **System Block Diagram**

Hardware Implementation

The system consists of a DisplayPort source core, a video pattern generator, and a minimal MicroBlaze processor to implement the policy maker. The policy maker is implemented in stand-alone C code running on the MicroBlaze processor. The block diagram for the MicroBlaze processor system is shown in Figure 2.

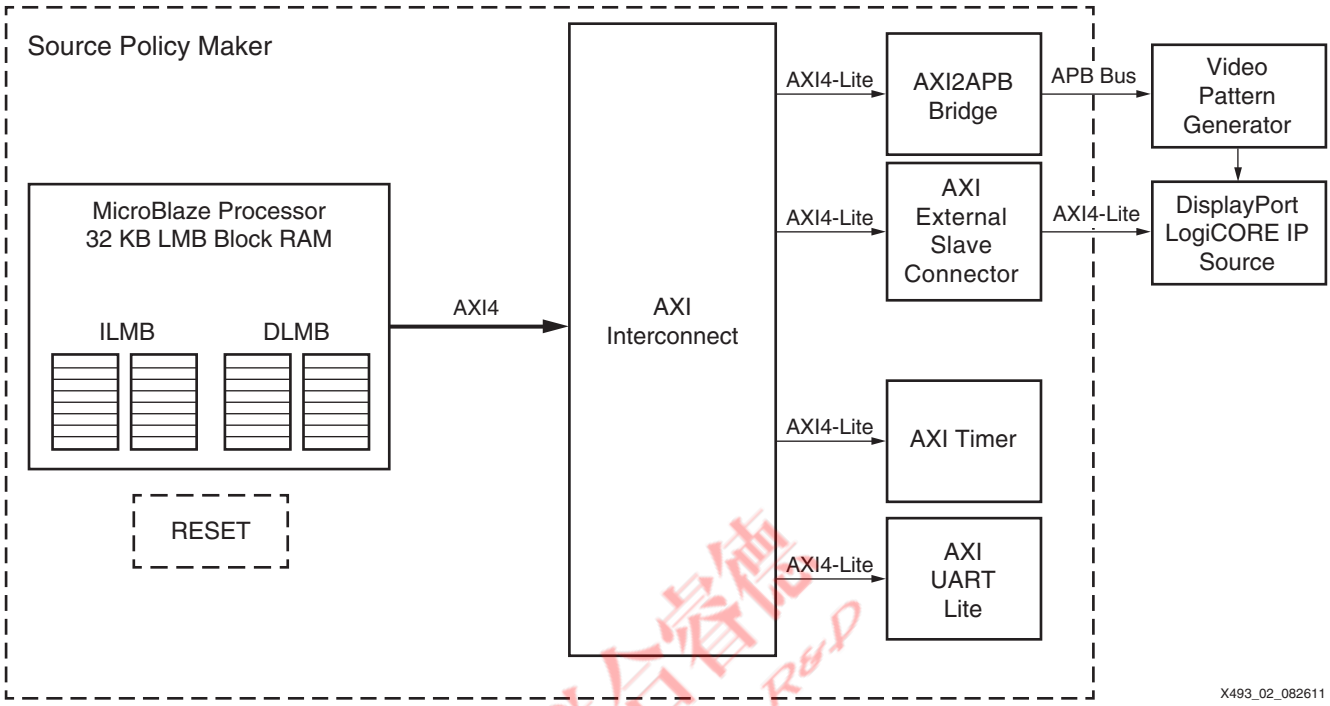


Figure 2: **EDK System Block Diagram**

The DisplayPort source core is generated from the CORE Generator tool. The policy maker design includes a MicroBlaze processor, an AXI Interconnect, an AXI timer, a MicroBlaze Debug Module (MDM) debug core, an AXI UART, and one instantiation of the AXI2APB bridge. The control connection to the DisplayPort IP is done through an AXI4-Lite interface through slave extension modules.

The communications interface between the MicroBlaze processor and the DisplayPort cores is an AXI4-Lite interface, a subset of the ARM® AMBA® 4 specification [Ref 5].

Clocking

The TED CVK 1.0 consists of these clocks:

- Input clocks:
  - GT reference clocks of 135 MHz for 2.7G line rate and 81 MHz for 1.62G line rate
  - System clock of 200 MHz
- Derived clocks:
  - AXI and MicroBlaze processor clock at 40 MHz derived from 200 MHz reference clock.
  - TX video clock synthesized using DCM_CLKGEN from link clock.

The ML605 with Avnet DVI I/O FMC module consists of these clocks:

- Input clocks:
  - Gigabit transceiver (GT) reference clocks from the Avnet DVI I/O FMC module that are programmed through the IIC interface. Reference clocks are dynamically configured

based on the link rate using the IIC interface through the policy maker. The link rates are 108 MHz for the 2.7G line rate and 81 MHz for the 1.62G line rate.
    - System clock of 200 MHz.
- Derived clocks:
    - The AXI and MicroBlaze processor clock at 40 MHz derived from 200 MHz reference clock.
    - TX video clock synthesized using mixed-mode clock manager (MMCM) from link clock.

Video Pattern Generator

Video timing information and pixel data is generated by the pattern generator. The video timing information is programmed from the policy maker. Standard pixel data patterns are generated based on the register programming from policy maker. These are the standard test patterns available:

- Vesa LLC pattern
- Vesa pattern3 - bars
- Vesa color squares
- Flat red screen
- Flat blue screen
- Flat green screen
- Flat yellow screen
- Color bars

Software Implementation

The software of the DisplayPort source reference design is the primary mechanism for controlling the DisplayPort link. The software provides a flexible terminal interface for testing and debugging the link interface. Figure 3 shows a state diagram of the DisplayPort source policy maker software. The default software included with the reference design runs through the link training and hot plug detection, and provides a UART console interface to provide debug capabilities such as read/write of the DisplayPort AUX registers and mode selection. The software consists of standalone polled C code running on a MicroBlaze processor core.

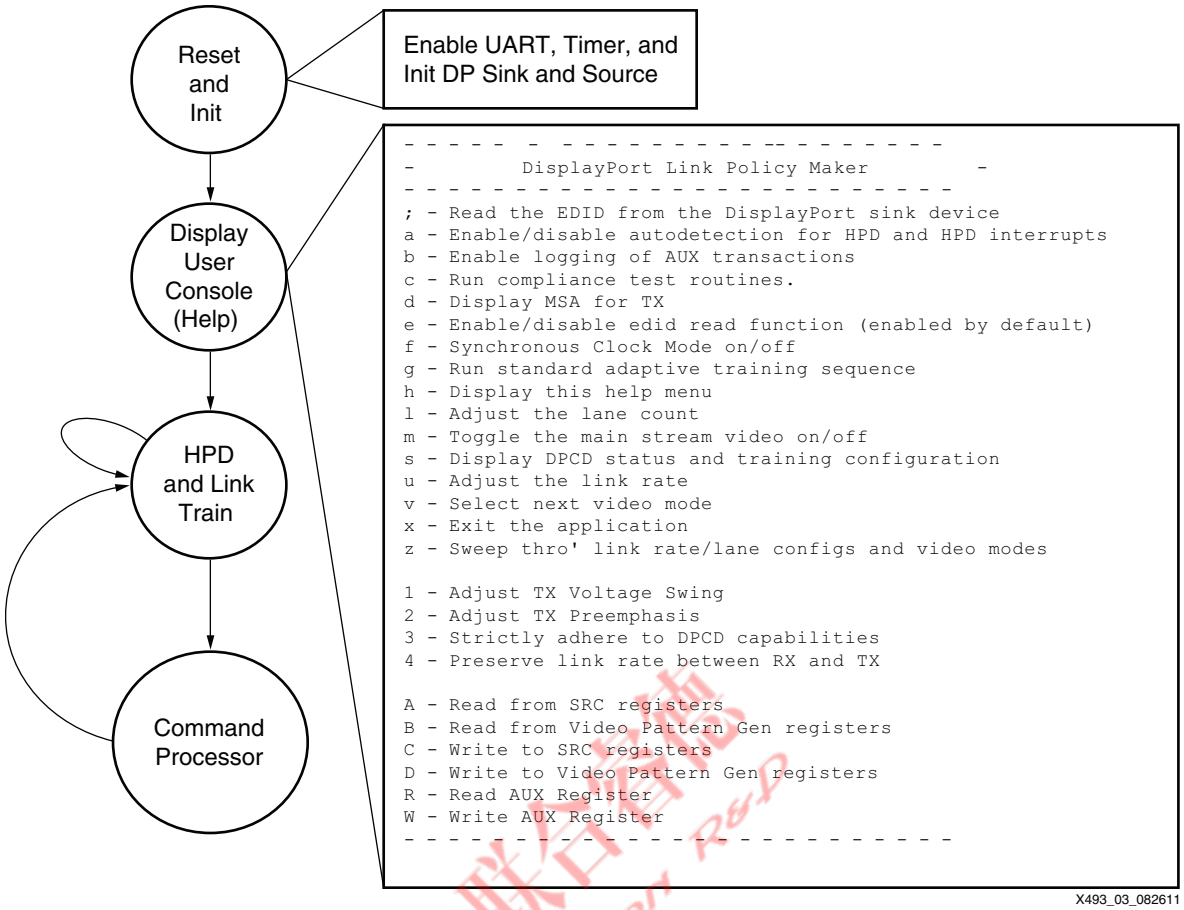


Figure 3: **Software Flow Diagram**

Software Flow

The state diagram in Figure 3 shows the basic structure of the software. This section describes the startup procedure and terminal options in more detail.

The address of this DisplayPort core is referenced multiple times throughout the software as XILINX_DISPLAYPORT_TX_BASE_ ADDRESS, which is in turn referenced from `sys_defs.h`.

Reset and Init

When the bitstream is downloaded to the FPGA, the MicroBlaze processor begins executing. The MicroBlaze processor first initializes its peripherals. It runs a self test on the timer and then checks the DisplayPort core in the system to ensure that it is a DisplayPort source core. The DisplayPort core type is read from address `0xFC` of the DisplayPort core; the value `0xAxxxx` indicates that the core is a source core.

If the DisplayPort core is identified to be a source core, this sequence is executed (`displayport_tx_drv.c`):

1. Disable the transmitter
2. Put the physical layer (PHY) into reset
3. Set the clock divider
4. Set the DisplayPort clock speed
5. Bring the PHY out of reset

6. Wait for the PHY to be ready
7. Enable the transmitter
8. Unmask all interrupts

The code then initializes the voltage swing and pre-emphasis values, but they are not yet transferred to the DisplayPort core. At this point, the init_platform function in `main.c` is complete.

Next, the xilcccAppInit (`xil_ccc_app.c`) function is called from `main.c`. Inside xilcccAppInit, the dplpmInitLinkPolicyMaker (`displayport_lpm.c`) function, which is responsible for setting the hardware capabilities of the transmitter, is called. The parameters passed to the dplpmInitLinkPolicyMaker function are link_rate and lane_count, but many settings are statically assigned within the function. The settings in dplpmInitLinkPolicyMaker control the startup functionality of the transmitter before the terminal is active. For the Virtex-6 FPGA system, the PLL on the FMC card is initialized after the init_platform routine using the AvnetPLLInit() routine.

Display User Console

The user console is displayed after the system is initialized (see Table 1). However, the DisplayPort link is not yet active. The hot-plug detect and link training take place after the console is displayed, and the terminal input command processor cannot begin until link training is complete. In other words, if the DisplayPort cable is not plugged in, or if hot-plug detect did not get established, pressing keys in the terminal does not execute commands.

Note: Keys that are pressed before the hot-plug detect is established are stored in the UART FIFO. The commands execute after the cable is plugged in.

The functionality of each terminal command is described in Command Processor, page 6.

Table 1: **Terminal Display for Command Processor**

Displayed on Terminal

```
- - - - - - - - - - - - - - - - - - - - - - - - - -
-          DisplayPort Link Policy Maker          -
- - - - - - - - - - - - - - - - - - - - - - - - - -
; - Read the EDID from the DisplayPort sink device
a - Enable / disable autodetection for HPD and HPD interrupts
b - Enable logging of AUX transactions
c - Run compliance test routines.
d - Display MSA for TX
e - Enable/disable edid read function (enabled by default)
f - Synchronous Clock Mode on/off
g - Run standard adaptive training sequence
h - Display this help menu
l - Adjust the lane count
m - Toggle the main stream video on/off
s - Display DPCD status and training configuration
u - Adjust the link rate
v - Select next video mode
x - Exit the application
z - Sweep thro' link rate / lane configs and video modes

1 - Adjust TX Voltage Swing
2 - Adjust TX Preemphasis
3 - Strictly adhere to DPCD capabilities
4 - Preserve link rate between RX and TX

A - Read from SRC registers
B - Read from Video Pattern Gen registers
C - Write to SRC registers
D - Write to Video Pattern Gen registers
R - Read AUX Register
W - Write AUX Register
- - - - - - - - - - - - - - - - - - - - - - - - - -
```

Hot-Plug Detect and Link Training

Hot-plug detect occurs every 100 ms when the user terminal is inactive or after a terminal function completes. The 100 ms time-out for checking hot-plug detect is set in `xil_ccc_app.c` at the xil_getc function call **`app_ctrl → cmd_key = xil_getc(100);`**. To change the time-out value, the time-out parameter passed to xil_getc should be changed. To see this hot-plug detect in action, the DisplayPort cable, which is attached to the CVK board, should be connected and disconnected. See Setup and Usage, page 20 for details on cable connectivity setup.

Command Processor

The command processor receives the input from the terminal and executes the desired transaction, as described in this section.

; — Read the EDID from the DisplayPort Sink Device

This function reads the DisplayPort Configuration Data (DPCD) and extended display identification data (EDID) from the sink device through the AUX channel and displays relevant information. Table 2 shows an example of what is displayed when the **`;`** command is executed. The terminal dumps the raw EDID data on the command terminal and can be used for advanced debug.

Table 2: **Terminal Display for DPCD and EDID Command**

Displayed on Terminal
Dump DPCD
DisplayPort Configuration Data
DPCD Revision : 1.1
Max Link Rate : 2.7 Gbps
Max Lane Count : 4
Enhanced Framing : Yes
Max Downspread : 0.5%
Require AUX Handshake : Yes
Number of RX Ports : 2
Main Link ANSI 8B/10B : No
Downstream Port Count : 0
Format Conversion Support : No
OUI Support : No
Receiver Port 0:
Has EDID : No
Uses Previous Port : No
Buffer Size : 0
========== Reading EDID... Start ==========
EDID[000 to 016] 00, FF, FF, FF, FF, FF, FF, 00, 10, AC, 4D, 40, 49, 45, 39, 30
EDID[016 to 032] 0D, 14, 01, 04, A5, 2F, 1E, 78, 3E, EE, 95, A3, 54, 4C, 99, 26
EDID[032 to 048] 0F, 50, 54, A5, 4B, 00, 71, 4F, 81, 80, B3, 00, 01, 01, 01, 01
EDID[048 to 064] 01, 01, 01, 01, 01, 01, 7C, 2E, 90, A0, 60, 1A, 1E, 40, 30, 20
EDID[064 to 080] 36, 00, DA, 28, 11, 00, 00, 1A, 00, 00, 00, FF, 00, 52, 38, 38
EDID[080 to 096] 30, 4B, 30, 33, 4D, 30, 39, 45, 49, 0A, 00, 00, 00, FC, 00, 44
EDID[096 to 112] 45, 4C, 4C, 20, 50, 32, 32, 31, 30, 0A, 20, 20, 00, 00, 00, FD
EDID[112 to 128] 00, 38, 4B, 1E, 53, 10, 00, 0A, 20, 20, 20, 20, 20, 20, 00, 99
========== Reading EDID...Done ==========

Table 3 provides descriptions of some of the DPCD capabilities.

Table 3: **Terminal Display for DPCD and EDID Command**

Displayed on Terminal	Description
DisplayPort Configuration Data	
DPCD Revision : 1.1	The revision number of the DisplayPort Sink.
Max Link Rate : 2.7 Gbps	The maximum capable link rate of the sink device. This is 2.7 Gb/s or 1.62 Gb/s.
Max Lane Count : 4	Lane count can be 1, 2, or 4.
Enhanced Framing : Yes	Framing mode support.
Max Downspread : 0.5%	Spread Spectrum support.
Require AUX Handshake : Yes	See *VESA DisplayPort Standard v1.1a* [Ref 4] for full details.
Number of RX Ports : 1	
Main Link ANSI 8B/10B : No	
Downstream Port Count : 0	
Format Conversion Support : No	
OUI Support : No	
Receiver Port 0:	

Table 3: **Terminal Display for DPCD and EDID Command *(Cont'd)***

Displayed on Terminal	Description
Has EDID : No Uses Previous Port : No Buffer Size : 0	

a — Enable/Disable Autodetection for HPD and HPD Interrupts

This function enables or disables the ability to autodetect hot-plug detect. Autodetection is enabled by default. This can be seen by connecting and disconnecting the DisplayPort cable while the application is running. As mentioned in Hot-Plug Detect and Link Training, page 6, hot-plug detect is checked periodically.

Table 4 is an example of what is displayed when the **a** command is executed. Each time the command is executed, autodetection toggles ON or OFF.

Table 4: **Terminal Display in Auto-Detect Mode**

Displayed on Terminal	Description
TX Autodetection is disabled	The code does not dynamically detect when the cable is connected or disconnected.

TX Autodetection is enabled	The code dynamically detects when the cable is connected or disconnected.
	When the cable is disconnected, this output is displayed:
	`##############################################`
	`######  Detected Dis-Connection Event!  ######`
	`##############################################`
	When the cable is reconnected, this output is displayed:
	`##########################################`
	`######  Detected Connection Event!  ######`
	`##########################################`

b — Enable Logging of AUX Transactions

This command enables/disables aux log. When enabled, the policy maker software displays the aux transaction on the console, as shown in Table 5.

Table 5: **Terminal Display of AUX Transactions**

Displayed on Terminal (with AUX Logging Enabled)
AUX Write: Address 0100, Bytes 01, Data 0x06 AUX Read : Address 0100, Bytes 01, Data 0x06 AUX Write: Address 0101, Bytes 01, Data 0x81 AUX Write: Address 0107, Bytes 01, Data 0x00 AUX Write: Address 0103, Bytes 04, Data 0x00 0x00 0x00 0x00 AUX Write: Address 0102, Bytes 01, Data 0x21 AUX Write: Address 0103, Bytes 04, Data 0x00 0x00 0x00 0x00 AUX Read : Address 0202, Bytes 06, Data 0x00 0x00 0x00 0x00 0xCC 0xCC AUX Write: Address 0103, Bytes 04, Data 0x38 0x38 0x38 0x38 AUX Read : Address 0202, Bytes 06, Data 0x01 0x00 0x00 0x00 0xCC 0xCC

c — Run Compliance Test Routines

This command provides some options required for compliance testing, namely, link training at various rates and lane counts, pattern type, bits per color, etc. The **c** command displays another submenu as described in Table 6.

Table 6: **Terminal Display Run Compliance Test Routines**

Displayed on Terminal
Run compliance test automation routines. Choose test option 1 --> train link @1.62G 1 lane 2 --> train link @1.62G 2 lane 3 --> train link @1.62G 4 lane 4 --> train link @2.7G 1 lane 5 --> train link @2.7G 2 lane 6 --> train link @2.7G 4 lane b --> set bits per color of video m --> set video resolution p --> set video pattern

For example, to train at the 1.62G line rate with one lane, enter **c** followed by **1**.

To change display patterns, enter **c** followed by **p**. This invokes the submenu shown in Table 7 from which the desired pattern can be selected.

Table 7: **Terminal Display to Choose Video Pattern to be Displayed**

Displayed on Terminal
Choose Video pattern 0 --> Color Bars 1 --> Vesa LLC pattern 2 --> Vesa Pattern3 - bars 3 --> Vesa Color Squares 4 --> Flat Red screen 5 --> Flat Blue screen 6 --> Flat Green screen 7 --> Flat Yellow screen Others --> select option from PSW2

To change the bits per color, enter **c** followed by **b**. This invokes the submenu shown in Table 8 from which the desired bits per color option can be selected.

Table 8: **Terminal Display to Choose Bits per Color**

Displayed on Terminal
Choose Video Bits per color option 0 --> 6 bpc 1 --> 8 bpc 2 --> 10 bpc 3 --> 12 bpc 4 --> 16 bpc

To switch to a specific resolution, enter **c** followed by **m**. This invokes the submenu shown in Table 9 from which the desired resolution can be selected.

Table 9: **Terminal Display to Choose Bits per Color**

Displayed on Terminal
Choose Video Resolution option 0 --> 640x480 1 --> 800x600 2 --> 1024x768 3 --> 1280x1024 4 --> 1600x1200 5 --> 1920x1200(RB)

d — Display MSA for TX

This command displays the main stream attribute (MSA) values for the DisplayPort source LogiCORE IP and the video pattern generator settings, as described in Table 10.

Table 10: **Terminal Display for MSA Command**

Displayed on Terminal	
Main Stream Attributes TX	
Clocks, H Total	: 1344
Clocks, V Total	: 806
Polarity (V / H)	: 3
HSync Width	: 136
VSync Width	: 6
Horz Resolution	: 1024
Vert Resolution	: 768
Horz Start	: 296
Vert Start	: 35
Misc0	: 0x00000021
Misc1	: 0x00000000
User Pixel Width	: 1
M Vid	: 13146
N Vid	: 32768
Transfer Unit Size	: 64
User Data Count	: 1535
Video Pattern Generator Config	
VPOL	: 1
HPOL	: 1
DEPOL	: 0
VSWIDTH	: 6
VB	: 29
VF	: 3
VRES	: 768
HSWIDTH	: 136
HB	: 160

Table 10: **Terminal Display for MSA Command *(Cont'd)***

Displayed on Terminal	
HF	: 24
HRES	: 1024

Table 11 provides detailed descriptions of the MSA attributes.

Table 11: **MSA Command Description**

Displayed on Terminal		Description
Main Stream Attributes TX (Source MSA transmitted Values)		
Clocks, H Total	: 2200	Total number of clock cycles for Horizontal Front Porch + HSync + Back Porch + Active Video
Clocks, V Total	: 1125	Total number of clock cycles for Vertical Front Porch + VSync + Back Porch + Active Video
Polarity (V / H)	: 0	Polarity of VSYNC and HSYNC, 1 = High, 0 = Low
HSync Width	: 44	Number of clock cycles HSYNC is asserted
VSync Width	: 5	Number of HSYNCS that the VSYNC is asserted
Horz Resolution	: 1920	Active-video horizontal resolution
Vert Resolution	: 1080	Active-video vertical resolution (line count)
Horz Start	: 192	Number of clocks between start of HSYNC and start of active video. The front porch is determined by this number: Front porch = H Total – Horz Resolution – Horz Start
Vert Start	: 41	Number of clocks between start of VSYNC and start of active video. The front porch is determined by this number: Front porch = V Total – Vert Resolution – Vert Start
Misc0	: 0x0021	Miscellaneous 0 register from DisplayPort specification
Misc1	: 0x00000000	Miscellaneous 1 register from DisplayPort specification
User Pixel Width	: 2	Refer to data sheet for options and details
M Vid	: 30038	PLL multiplier for stream clock recovery
N Vid	: 32768	PLL divider for stream clock recovery
Transfer Unit Size	: 64	TRANSFER_UNIT_SIZE. This sets the size of a transfer unit in the transmitter framing logic. This number should be in the range of 32 to 64 and is set to a fixed value that depends on the inbound video mode.
User Data Count	: 2879	This register is used to translate the number of pixels per line to the native internal 16-bit datapath. It should be set to (HRES * bits per pixel/16) – 1.
Video Timing Generator Configuration (Note this is dual pixel mode)		
VPOL	: 0	VSYNC polarity for the pattern generator
HPOL	: 0	HSYNC polarity for the pattern generator
DEPOL	: 0	Data Enable polarity for the pattern generator
VSWIDTH	: 5	VSYNC width
VB	: 36	Vertical back porch
VF	: 4	Vertical front porch
VRES	: 1080	Vertical resolution
HSWIDTH	: 22	HSYNC width
HB	: 74	Horizontal back porch
HF	: 44	Horizontal front porch
HRES	: 960	Horizontal resolution

e — Enable/Disable EDID Read Function

This command allows user to enable/disable read of EDID from the sink device.

f — Synchronous Clock Mode On/Off

Synchronous clocking mode can be turned on and off using this command. Because this command affects MSA values, video parameters are reprogrammed but remain in the previous mode (Table 12).

Table 12: **Terminal Display for Synchronous Clocking Mode Selection**

Displayed on Terminal
MISC0[0]Synchronous Clock enabled Setting mode to VIDEO_MODE_640_480_60_32, 6 Bits Per Color MISC0[0]Synchronous Clock disabled Setting mode to VIDEO_MODE_640_480_60_32, 6 Bits Per Color

g — Run Standard Adaptive Training Sequence

This command runs a full training sequence on the link, including hot-plug detect, link setup, and link training.

h — Display this Help Menu

This command displays the help menu, as shown in Table 1, page 6.

l — Adjust the Lane Count

This command toggles the transmitter between 1, 2, or 4 lanes each time **l** (lower-case L) is pressed. The possible outputs are shown in Table 13. Some sinks support lane count reduction without retraining while some require retraining.

Table 13: **Lane Count Status Display**

Displayed on Terminal
Lane count set to 1 Lane count set to 2 Lane count set to 4

m — Toggle the Main Stream Video On/Off

This command toggles the main link on or off each time **m** is pressed.

s — Display DPCD Status and Training Configuration

This command displays the training information and configuration data of the DisplayPort monitor connected to the source port (Table 14).

Table 14: **Terminal Display for DPCD Status Command**

Displayed on Terminal
Lane 0/1 Status : 0x77 Lane 2/3 Status : 0x77 Lane Align Status : 0x01 Sink Status : 0x01 Adjustment Request 0/1 : 0x00 Adjustment Request 2/3 : 0x00 Training Config: (0x0100) Link Bandwidth Setup : 0x06 (0x0101) Lane Count Set : 0x84 (0x0102) Training Pattern Set : 0x00 (0x0103) Training Lane 0 Set : 0x00 (0x0104) Training Lane 1 Set : 0x00 (0x0105) Training Lane 2 Set : 0x00 (0x0106) Training Lane 3 Set : 0x00 (0x0107) Downspread Ctrl : 0x00

Table 15 provides a detailed description of the various fields of the DPCD status command.

Table 15: **DPCD Status Display**

Displayed on Terminal	Description
Training Status (monitor connected to source port)	
Lane 0/1 Status = 0x77	LANE0_1_STATUS: Lane0 and Lane1 status Bit 0 = LANE0_CR_DONE Bit 1 = LANE0_CHANNEL_EQ_DONE Bit 2 = LANE0_SYMBOL_LOCKED Bit 3 = RESERVED. Read 0. Bit 4 = LANE1_CR_DONE Bit 5 = LANE1_CHANNEL_EQ_DONE Bit 6 = LANE1_SYMBOL_LOCKED Bit 7 = RESERVED. Read 0.
Lane 2/3 Status = 0x77	LANE2_3_STATUS (Bit definition identical to that of LANE0_1_STATUS)
Lane Align Status = 0x81	LANE_ALIGN__STATUS_UPDATED Bit 0 = INTERLANE_ALIGN_DONE Bits 5:1 = RESERVED. Read all 0s. Bit 6 = DOWNSTREAM_PORT_STATUS_CHANGED Bit 6 is set when any of the downstream ports change status. Bit 7 = LINK_STATUS_UPDATED Link Status and Adjust Request are updated after the last read. Bit 7 is set when updated and cleared after read.
Sink Status = 0x00	SINK_STATUS Bit 0 = RECEIVE_PORT_0_STATUS 0 = SINK out of synchronization 1 = SINK in synchronization Bit 1 = RECEIVE_PORT_1_STATUS 0 = SINK out of synchronization 1 = SINK in synchronization These status bits are set only when the sink device determines that the received streams are properly regenerated and within the supported stream format range. Bits 7:2 = RESERVED. Read all 0s.

Table 15: **DPCD Status Display *(Cont'd)***

Displayed on Terminal	Description
`Adjustment Request 0/1 = 0x88`	ADJUST_REQUEST_LANE0_1: Voltage swing and equalization setting adjust request for Lane0 and Lane1 Bits 1:0 = VOLTAGE_SWING_LANE0 00 = Level 0 01 = Level 1 10 = Level 2 11 = Level 3 Bits 3:2 = PRE-EMPHASIS_LANE0 00 = Level 0 01 = Level 1 10 = Level 2 11 = Level 3 Bits 5:4 = VOLTAGE_SWING_LANE1 (Same as Lane0) Bits 7:6 = PRE-EMPHASIS_LANE1 (Same as Lane1)
`Adjustment Request 2/3 = 0x88`	ADJUST_REQUEST_LANE2_3 (Bit definitions identical to that of ADJUST_REQUEST_LANE0_1)
`(SOURCE) Training Config: (Source Core)`	
`(0x0100) Link Bandwidth Setup : 0x06`	LINK_BW_SET: Main link bandwidth 0xA: HBR 2.7G 0x6: RBR 1.62G
`(0x0101) Lane Count Set : 0x84`	LANE_COUNT_SET: Main link lane count = value Bits 4:0 = LANE_COUNT_SET 1h = One lane 2h = Two lanes 4h = Four lanes For DPCD Ver.1.0: Bits 7:5 = RESERVED. Read all 0s. For DPCD Ver.1.1: Bits 6:5 = RESERVED. Read all 0s. Bit 7 = ENHANCED_FRAME_EN 0 = Enhanced framing symbol sequence is not enabled. 1 = Enhanced framing symbol sequence for BS, SR, CPBS, and CPSR is enabled.

Table 15: **DPCD Status Display *(Cont'd)***

Displayed on Terminal	Description
(0x0102) Training Pattern Set : 0x00 stopped here	TRAINING_PATTERN_SET Bits 1:0 = TRAINING_PATTERN_SET: Link training pattern setting 00 =Training not in progress (or disabled) 01 =Training Pattern 1 10 =Training Pattern 2 11 = RESERVED Bits 3:2 = LINK_QUAL_PATTERN_SET 00 = Link quality test pattern not transmitted 01 = D10.2 test pattern (unscrambled) transmitted 10 = Symbol Error Rate measurement pattern transmitted 11 = PRBS7 transmitted Bit 4 = RECOVERED_CLOCK_OUT_EN Bit 5 = SCRAMBLING_DISABLE 0 = DisplayPort transmitter scrambles data symbols before transmission 1 = DisplayPort transmitter disables scrambler and transmits all symbols without scrambling For DPCD Version 1.0: Bits 7:6 = Reserved, read as 0s. For DPCD Version 1.1 Bits 7:6 = SYMBOL ERROR COUNT SEL 00 = Disparity error and illegal symbol error 01 = Disparity error 10 = Illegal symbol error 11 = Reserved
(0x0103) Training Lane 0 Set : 0x10	TRAINING_LANE0_SET: Link Training Control_Lane0 Bits 1:0 = VOLTAGE SWING SET 00 = Training with voltage swing level 0 01 = Training with voltage swing level 1 10 = Training with voltage swing level 2 11 = Training with voltage swing level 3 Bit 2 = MAX_SWING_REACHED Set to 1 when the maximum driven current setting is reached. Bits 4:3 = PRE-EMPHASIS_SET 00 = Training without pre-emphasis 01 = Training with pre-emphasis level 1 10 = Training with pre-emphasis level 2 11 = Training with pre-emphasis level 3 Bit 5 = MAX_PRE-EMPHASIS_REACHED Set to 1 when the maximum drive current setting is reached.
(0x0104) Training Lane 1 Set : 0x10	TRAINING_LANE1_SET (Bit definition identical to that of TRAINING_LANE0_SET.)
(0x0105) Training Lane 2 Set : 0x10	TRAINING_LANE2_SET (Bit definition identical to that of TRAINING_LANE0_SET.)
(0x0106) Training Lane 3 Set : 0x10	TRAINING_LANE3_SET (Bit definition identical to that of TRAINING_LANE0_SET.)
(0x0107) Downspread Ctrl : 0x00	DOWNSPREAD_CTRL: Down-spreading control Bits 3:0 = RESERVED. Read all 0s. Bit 4 = SPREAD_AMP Spreading amplitude: 0 = No downspread 1 = Equal to or less than 0.5% downspread

u — Adjust the Link Rate

This function lets the user change the link rate to either the reduced bit rate (RBR) or high bit rate (HBR). When the rate change happens, the sink device needs to be retrained. The policy maker adheres to the DCPD capabilities advertised by the sink. For example, if a sink can handle only RBR, changing the link rate to HBR causes a retrain sequence, but in RBR. To force a link rate, the **u** command must be used in tandem with the **3** command to turn off Adhere to DPCD Capabilities.

v — Select Next Video Mode

This command initializes the video pattern generator and changes the video mode. The command cycles through some defined modes, as shown in Table 16.

Note: The user can add more to xilcccChangeVideoMode(), as required. The bits per color and pattern of display selection is independent of the video resolution.

Table 16: **Video Mode Status Display**

Displayed on Terminal
Mode 0x0, Link Rate 270, Lane Count 4, Video mode ID 640x480_60_P, Bpc 6
Mode 0x1, Link Rate 270, Lane Count 4, Video mode ID 800x600_60_P, Bpc 6
Mode 0x2, Link Rate 270, Lane Count 4, Video mode ID 1024x768_60_P, Bpc 6
Mode 0x3, Link Rate 270, Lane Count 4, Video mode ID 1280x768_60_P, Bpc 6
Mode 0x4, Link Rate 270, Lane Count 4, Video mode ID 1280x800_60_P, Bpc 6
Mode 0x5, Link Rate 270, Lane Count 4, Video mode ID 1280x1024_60_P, Bpc 6
Mode 0x6, Link Rate 270, Lane Count 4, Video mode ID 1440x900_60_P, Bpc 6
Mode 0x7, Link Rate 270, Lane Count 4, Video mode ID 1600x1200_60_P, Bpc 6
Mode 0x8, Link Rate 270, Lane Count 4, Video mode ID 1680x1050_60_P, Bpc 6
Mode 0x9, Link Rate 270, Lane Count 4, Video mode ID 1920x1200_60_P_RB, Bpc 6
Mode 0xA, Link Rate 270, Lane Count 4, Video mode ID 1920x1200_60_P, Bpc 6

x — Exit the Application

This function exits the application loop and returns to main. The processor remains in an infinite loop in main and does nothing more at this point.

z — Sweep thro' Link Rate/Lane Configs and Video Modes

The policy maker software loops to create link rate/lane count combinations and runs through various video modes when this command is used, as shown in Table 17.

Table 17: **Terminal Display for Sweep through Mode Command**

Displayed on Terminal

`- - - - - - - - - - - - - - - - - - - - - - - -`
`Running training on all lane/link combinations & video modes...`
`- - - - - - - - - Status - - - - - - - - - -`
`Lane 0/1 Status = 0x07`
`Lane 2/3 Status = 0x00`
`Lane Align Status = 0x01`
`Sink Status = 0x00`
`Adjustment Request 0/1 = 0x44`
`Adjustment Request 2/3 = 0x44`
`Training Config:`
`(0x0100) Link Bandwidth Setup : 0x06`
`(0x0101) Lane Count Set : 0x81`
`(0x0102) Training Pattern Set : 0x00`
`(0x0103) Training Lane 0 Set : 0x08`
`(0x0104) Training Lane 1 Set : 0x08`
`(0x0105) Training Lane 2 Set : 0x08`
`(0x0106) Training Lane 3 Set : 0x08`
`(0x0107) Downspread Ctrl : 0x00`
`- - - - - - - - - - - - - - - - - - - - - - - -`
 	
`$$$$$$ Link setup to 1 lane(s), link_rate 162 Mbps $$$$$`
`Setting mode to VIDEO_MODE_640_480_60_32, 6 Bits Per Color`
 	
`Press x to exit loop or any key to move to next mode....`

The policy maker waits for user input before moving to the next mode/configuration. The user can choose to exit the loop or continue by selecting **x** or any other key (Table 18).

Table 18: **Terminal Display for Sweep through Video Mode Command**

Displayed on Terminal
`Setting mode to VIDEO_MODE_800_600_60_32, 8 Bits Per Color` `Press x to exit loop or any key to move to next mode....`

1 — Adjust TX Voltage Swing

This command lets the user change the voltage swing setting of the PHY module. The user is prompted to select the required voltage swing, as shown in Table 19.

Table 19: **Terminal Display for Adjust TX Voltage Swing**

Displayed on Terminal

`Please select a voltage swing setting`
`1 - 400 mV`
`2 - 600 mV`
`3 - 800 mV`
`4 - 1200 mV`

2 — Adjust TX Preemphasis

This command lets the user change the pre-emphasis setting of the PHY module. The user is prompted to select the required pre-emphasis, as shown in Table 20.

Table 20: **Terminal Display for Adjust TX Pre-Emphasis**

Displayed on Terminal
Please select a preemphasis setting 1 - 0 dB (1x) 2 - 3.5 dB (1.5x) 3 - 6 dB (2x) 4 - 9.5 dB (3x)

3 — Strictly Adhere to DPCD Capabilities

The use of this mode restricts the policy maker to use the capabilities of the sink device while training. This command lets the user force a specific training configuration by overriding the advertised DPCD capabilities. By default, the policy maker is set to adhere to the DPCD capabilities.

4 — Preserve link rate between RX and TX

This command preserves link rate feature control downshifting of the training rate to RBR if training fails in HBR. By default, the policy maker down shifts the link rate to RBR if HBR training fails. This command can be used to disable this mechanism.

A — Read from SRC Registers

This command allows direct read access to the registers inside the DisplayPort source LogiCORE IP (Table 21). A description of each register can be found in the “Source Core Architecture” chapter of *LogiCORE IP DisplayPort v2.3 User Guide* [Ref 6].

Table 21: **Terminal Display for Read Source Register Command**

Displayed on Terminal
Enter 4 hex characters: Source Read address 0x0000 Source Read Addr C3A00000 Read Data: 0006

B — Read from Video Timing Generator Registers

This command allows direct read access to the registers inside the timing generator (Table 22). Refer to Table 27, page 19 for a description of each register.

Table 22: **Terminal Display for Read Video Pattern Generator Register Command**

Displayed on Terminal
Enter 4 hex characters: Video Pattern Gen Read address 0x0100 Video Pattern Gen Read Addr CCA00100 Read Data: 0001

C — Write to SRC Registers

This command allows direct write access to the registers inside the DisplayPort source LogiCORE IP (Table 23).

Table 23: **Terminal Display for Write to Source Register Command**

Displayed on Terminal
Enter 4 hex characters: Source Write address 0x0000 Enter 4 hex characters: Source Write data 0x0006 Source Write Addr C3A00000 Write Data: 0006

D — Write to Video Pattern Generator Registers

This command allows direct write access to the registers inside the video pattern generator (Table 24).

Table 24: **Terminal Display for Write to Video Pattern Generator Register Command**

Displayed on Terminal

`Enter 4 hex characters: Video Pattern Gen Write address 0x0100`
`Enter 4 hex characters: Video Pattern Gen Write Data 0x0001`
`Video Pattern Gen Write Addr CCA00100 Write Data: 0001`

R — Read AUX Register

This command allows read access to the DisplayPort configuration data register space of the DisplayPort monitor connected to the source using the AUX channel. Detailed descriptions of the registers can be found in the DisplayPort specification in the address mapping for the DPCD table (Table 25).

Table 25: **Terminal Display for AUX Read from Monitor on Source Command**

Displayed on Terminal
`Enter 4 hex characters: Aux Read Address 0x0100` `Aux Read Addr 0100, Read Data: 06`

W — Write AUX Register

This command allows write access to the DisplayPort configuration data register space of the DisplayPort monitor connected to the source using the AUX channel. Detailed descriptions of the registers can be found in the DisplayPort specification in the address mapping for the DPCD table (Table 26).

Table 26: **Terminal Display for AUX Write to Monitor on Source Command**

Displayed on Terminal

`Enter 4 hex characters: Aux Write Address 0x0100`
`Enter 2 hex characters: Aux Write Data 0x06`
`Aux Write Addr 0100 Write Data: 06`

The video pattern generator register map is shown in Table 27.

Table 27: **Video Pattern Generator Registers**

Register Address	Read/Write	Description
0x000	R/W	Bit 0 = Enable video output. Bit 1 = SW reset of the pattern generator.
0x004	R/W	Bit 0 = VSYNC polarity.
0x008	R/W	Bit 0 = HSYNC polarity.
0x00C	R/W	Bit 0 = DE polarity.
0x010	R/W	Bits 8:0 = VSYNC width.
0x014	R/W	Bits 8:0 = Vertical back porch.
0x018	R/W	Bits 8:0 = Vertical front porch.
0x01C	R/W	Bits 10:0 = Vertical resolution.
0x020	R/W	Bits 8:0 = HSYNC width.

Table 27: **Video Pattern Generator Registers *(Cont'd)***

Register Address	Read/Write	Description
0x024	R/W	Bits 8:0 = Horizontal back porch.
0x028	R/W	Bits 8:0 = Horizontal front porch.
0x02C	R/W	Bits 10:0 = Horizontal resolution.
0x104	R/W	Bits 7:0 = TX Video clock M value. Used for video clock synthesis. Video_clock = lnk_clk × M/D.
0x108	R/W	Bits 7:0 = TX Video clock D value. Used for video clock synthesis. Video_clock = lnk_clk × M/D.
0x200	R	Bits 11:0 = VSYNC counter current count.
0x204	R	Bits 11:0 = HSYNC counter current count.
0x208	R	Bits 11:0 = Data enable counter current count.

Setup and Usage

This reference design is targeted at the Spartan-6 FPGA Consumer Video Kit [Ref 1]. To bring up the reference design, this setup is needed:

- PC with at least two USB 1.1 or USB 2.0 ports with ISE Design Suite 13.2 and EDK 13.2 installed.
- Spartan-6 FPGA CVK board and power supply.
- DisplayPort-compliant receiver device, such as a monitor.
- Platform cable USB JTAG programmer.
- DisplayPort cable and two USB cables.

The setup is shown in Figure 4.

Figure 4: **CVK 1.0 Hardware Setup**

Ensure that all necessary cables and jumpers are set correctly on the board, as shown in Table 28 and Figure 5.

Table 28: **CVK 1.0 Jumper Settings**

Jumper Name	Jumper Position
JP2	1-2
JP3	2-3
JP4	1-2
JP5	1-2
JP6	1-2
JP7	1-2
JP8	1-2
JP9	1-2
JP10	1-2
JP11	1-2
JP12	1-2

Figure 5 shows the required connections for a DisplayPort source demonstration of the CVK 1.0 board.

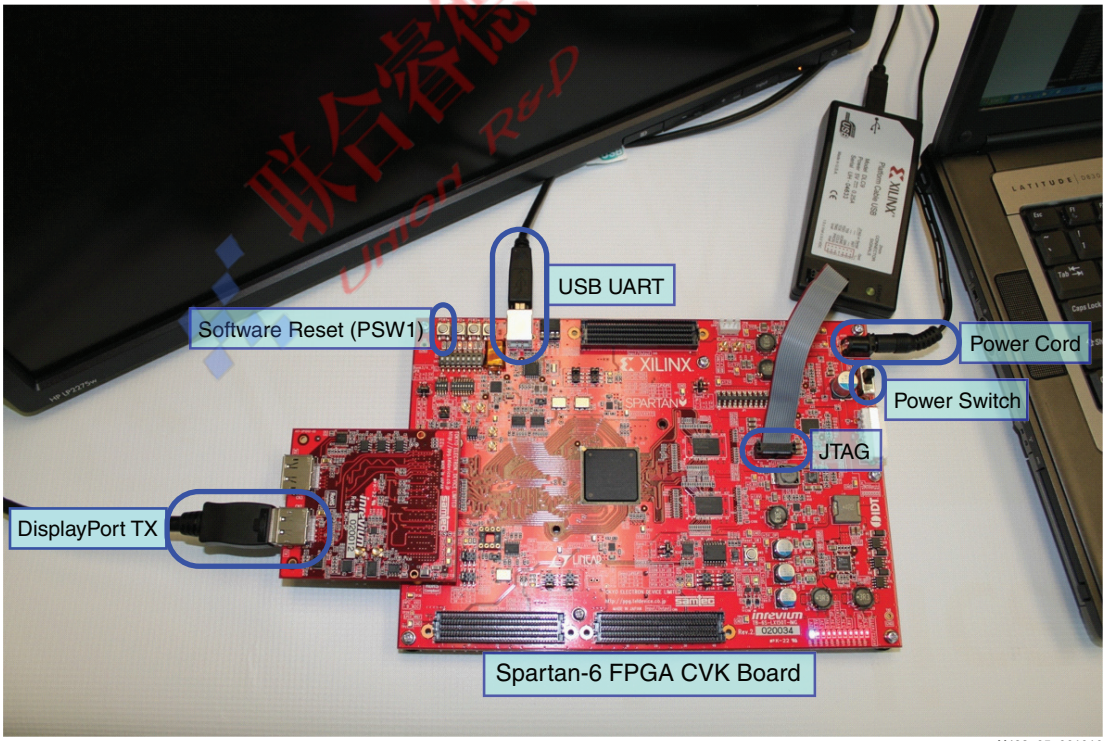


Figure 5: **CVK 1.0 Board Setup**

Figure 6 shows the required connections for a DisplayPort source demonstration of the ML605 board with an Avnet FMC card.

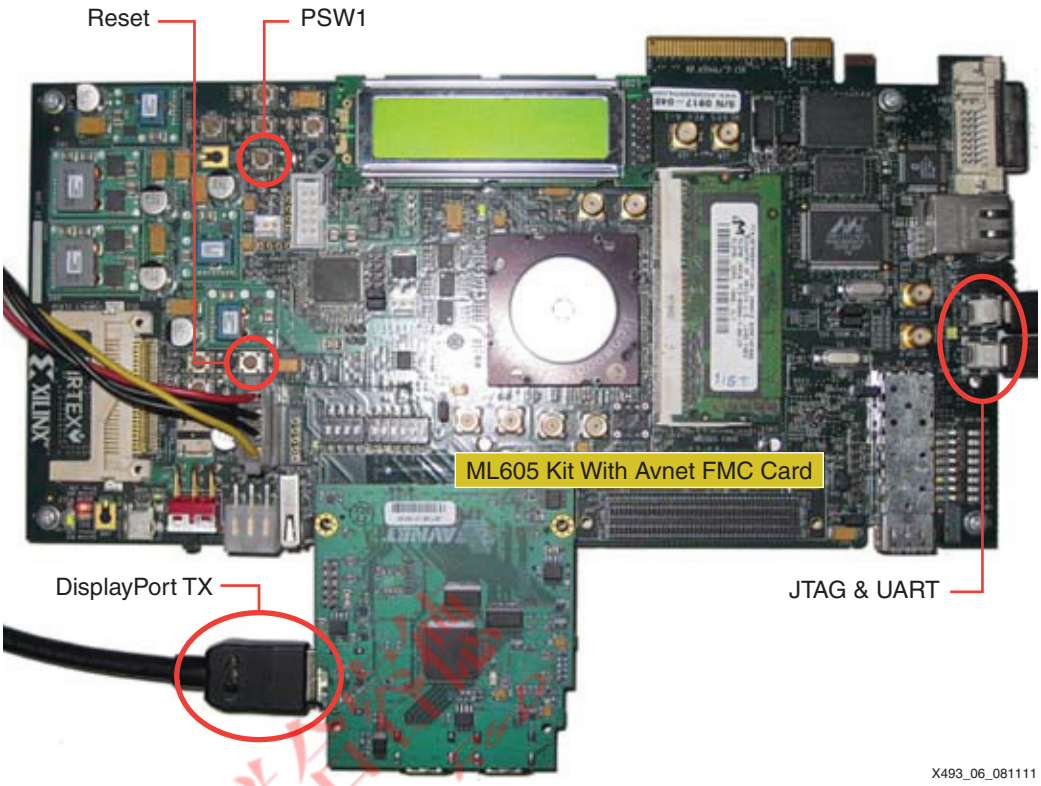


Figure 6: **ML605 Board with Avnet DVI I/O FMC Module Setup**

The USB UART is detected and installed automatically onto a COM port. The device manager should be checked to see which COM port it has been assigned to. On the host computer, a terminal application such as HyperTerminal or PuTTy should be opened in serial mode and connected to the COM port with these settings (for PuTTy):

Baud: 115200
Parity: None

Pre-verified system bitstream files and executable and linkable format (ELF) files are available for both Spartan-6 and Virtex-6 FPGA designs. With these files, the user can bring up the systems and ensure connectivity.

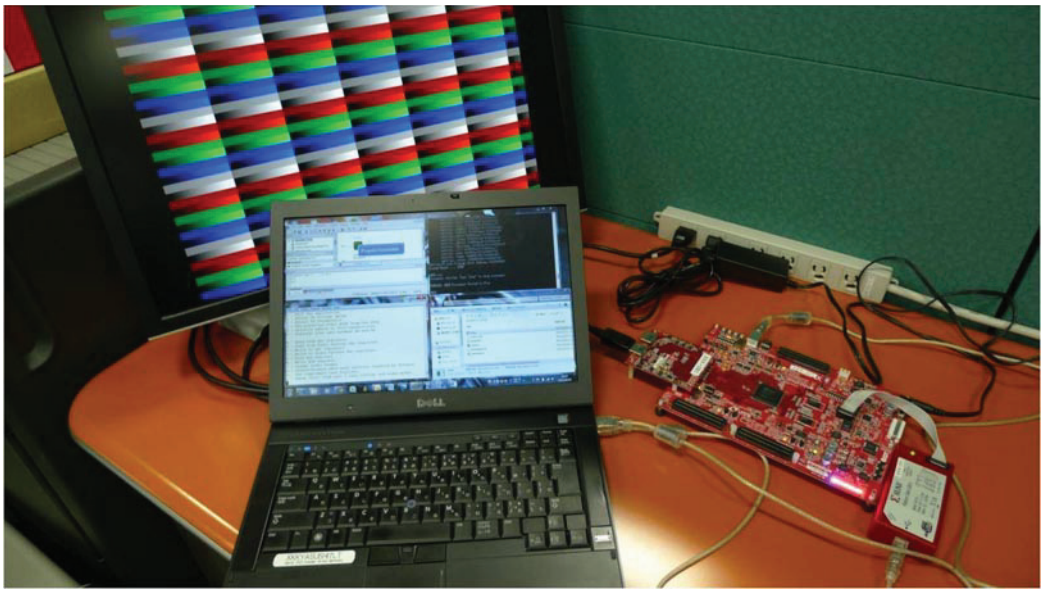
The Spartan-6 FPGA files can be downloaded from:

```
XAPP493/s6/CVK1.0/sdk_workspace/hw_platform_0/download.bit
```

The Virtex-6 FPGA files can be downloaded from:

```
XAPP493/v6/ML605/sdk_workspace/hw_platform_0/download.bit
```

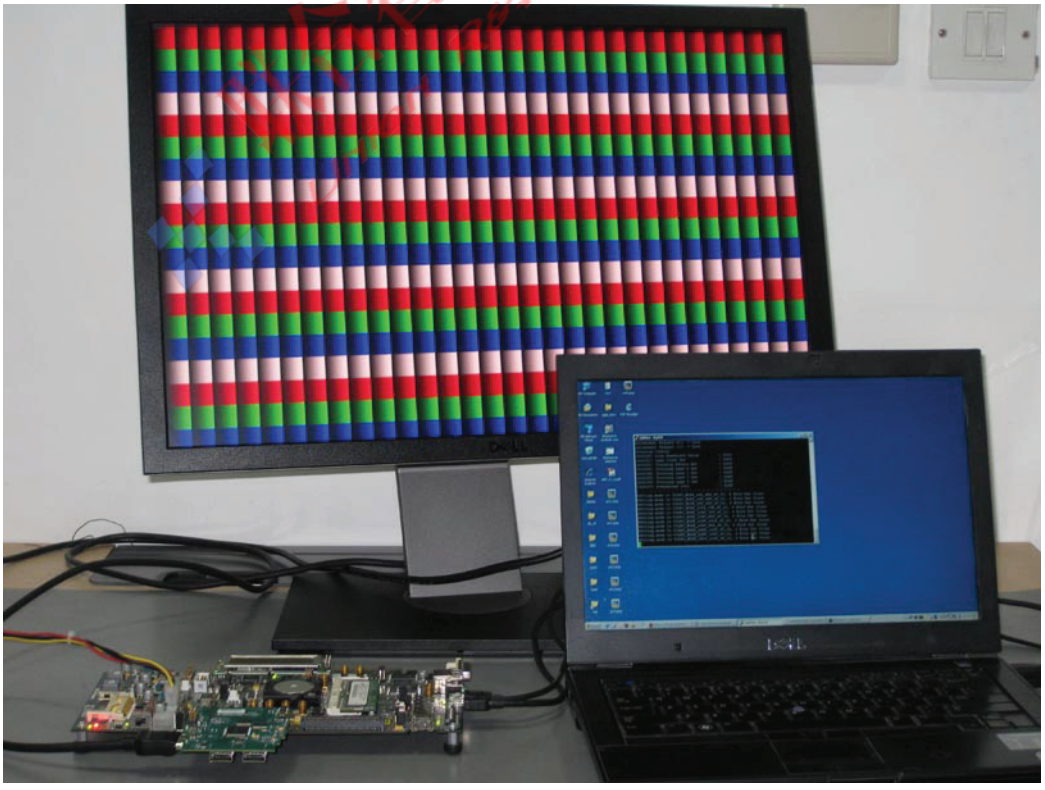
Figure 7 and Figure 8 show the working systems with the pre-verified files on the CVK 1.0 and ML605 boards, respectively.



X493_07_070711

Figure 7: **Working System with CVK 1.0**

Figure 8 shows an example setup of a working system with the ML605 board and the Avnet DVI I/O FMC module.



X493_08_070711

Figure 8: **Working System with ML605 Board and Avnet DVI I/O FMC Module**

Reference Design Files

The reference design is organized into five directories:

- `doc`: Contains documentation relevant to the reference design
- `ise_top_level`: Contains the top-level ISE tools project
- `display_port_source_policy_maker`: Contains the DisplayPort Sink Policy Maker, which is an EDK system
- `design_files`: Contains additional design files not generated by the ISE tools or EDK.
- `sdk_workspace`: Contains the SDK workspace files

Figure 9 shows the directory hierarchy and the most important files in the directories.

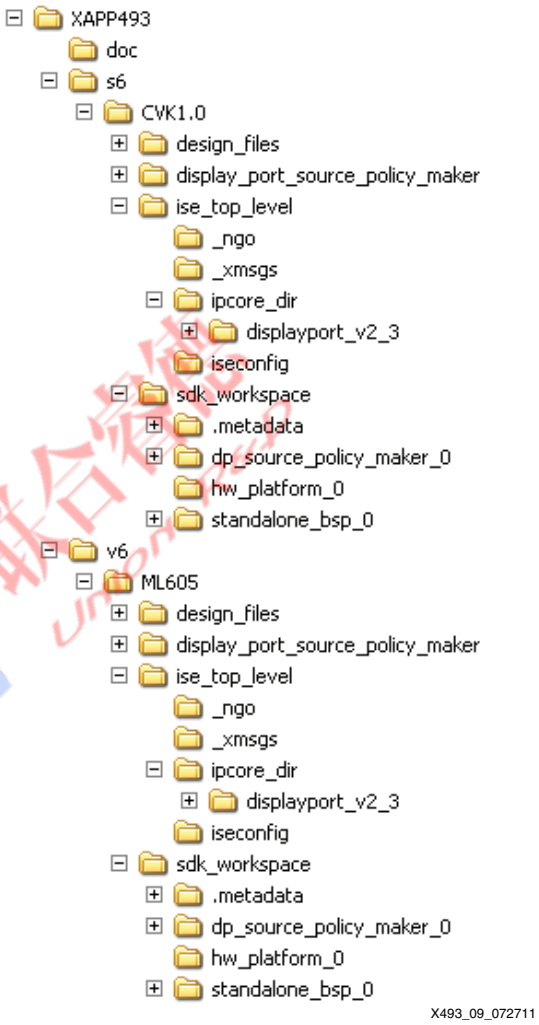


Figure 9: **Directory Structure**

Generating the Design

This section discusses how to recreate the system using the ISE tools, the CORE Generator software, EDK, and SDK. The basic steps to recreate the project are:

The steps shown are for CVK 1.0 using the Spartan-6 device. Similar steps are performed for the Virtex-6 family with the appropriate part name.

Step 1: Set Up the Directory Structure

Setting up the directory structure is very important to maintain readability of the directories and to avoid duplicate files. Create a directory structure as shown in Figure 10. Remember the location of the XAPP folder because it is referenced multiple times throughout the use of this application note.

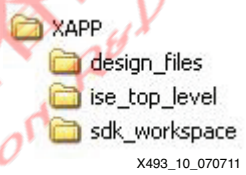


Figure 10: **Directory Structure Setup**

After creating the directories, copy the contents of the original `XAPP/design_files` directory to the newly created `design_files` directory.

Step 2: Create the ISE Tools System

All hardware source files are managed by the ISE tools project to facilitate implementation. Open the ISE tools and create a new project by clicking **File** → **New Project...** and enter the project **Name:** **`dport_source_ref_design`**.

Set **Location:** and **Working Directory:** to **`mydirectory/XAPP/ise_top_level`**, where `mydirectory` is the location where the application note folder is placed in Step 1: Set Up the Directory Structure. Set the **Top-level source type** to **HDL**, as shown in Figure 11.

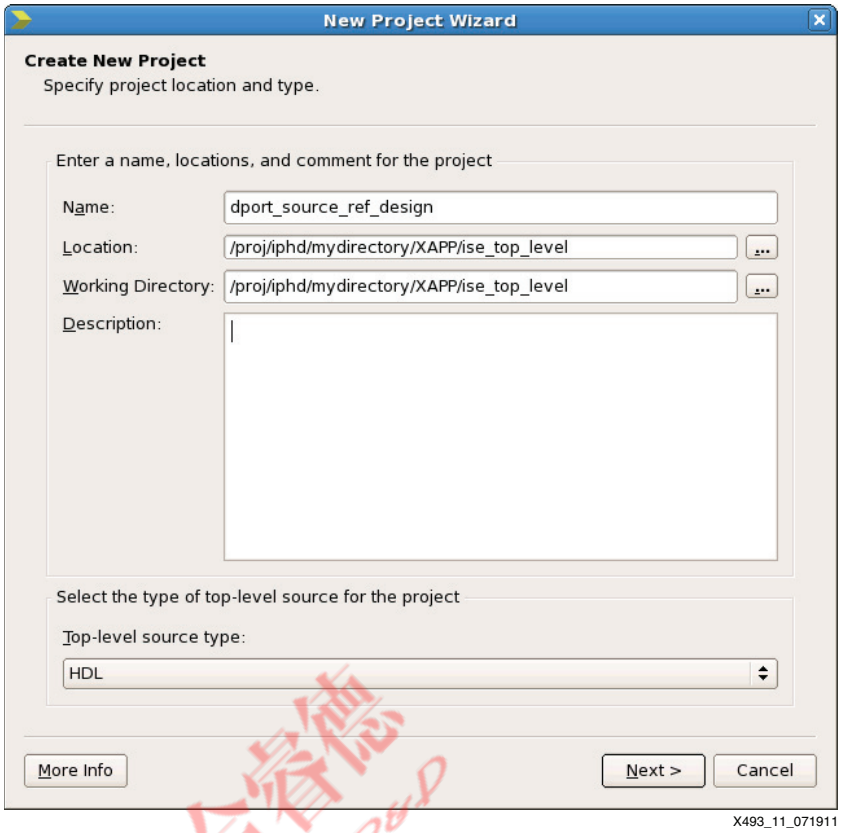


Figure 11: **ISE Tools New Project Wizard**

Set the device and project properties to select the correct device for the CVK board (Figure 12):

- **Family**: **Spartan6**
- **Product**: **XC6SLX150T**
- **Package**: **FGG676**
- **Speed**: **-3**

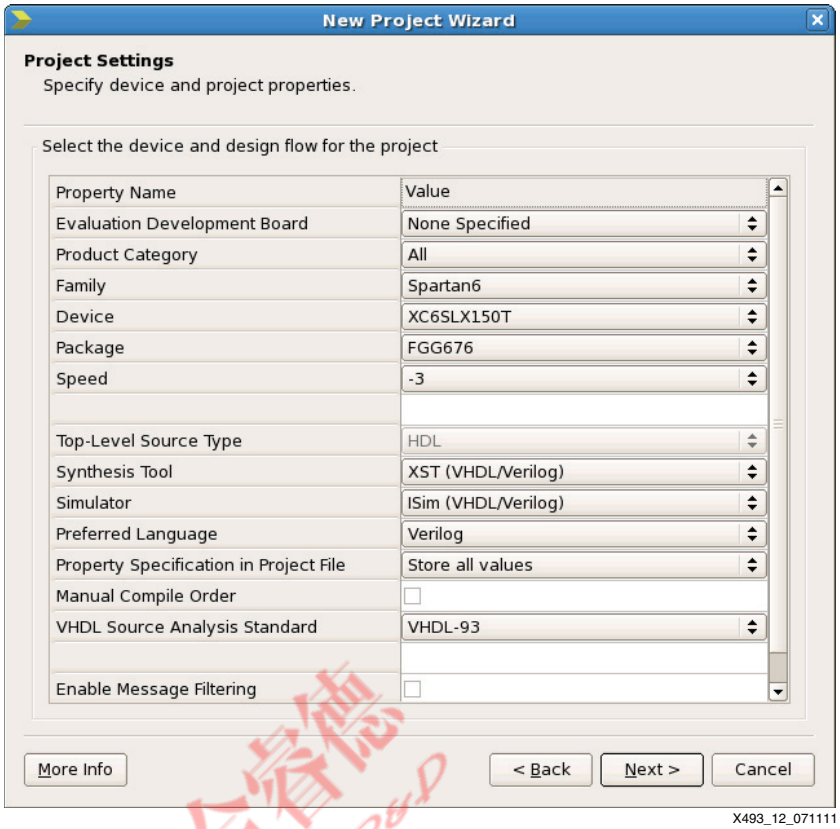


Figure 12: **ISE Tools Project Settings**

Click **Next**, and then **Finish** to create the ISE tools system.

The system is now ready to have source files added to it. The first files to be added are the DisplayPort transmitter and receiver files. These files are created using the CORE Generator software, as described in the next step.

Step 3: Generate and Integrate the Cores

Open the CORE Generator software from the ISE tools by clicking on **Tools** → **CORE Generator…**. Set up a new project for the CORE Generator software, which points to `ipcore_dir` in the project, as shown in Figure 13.

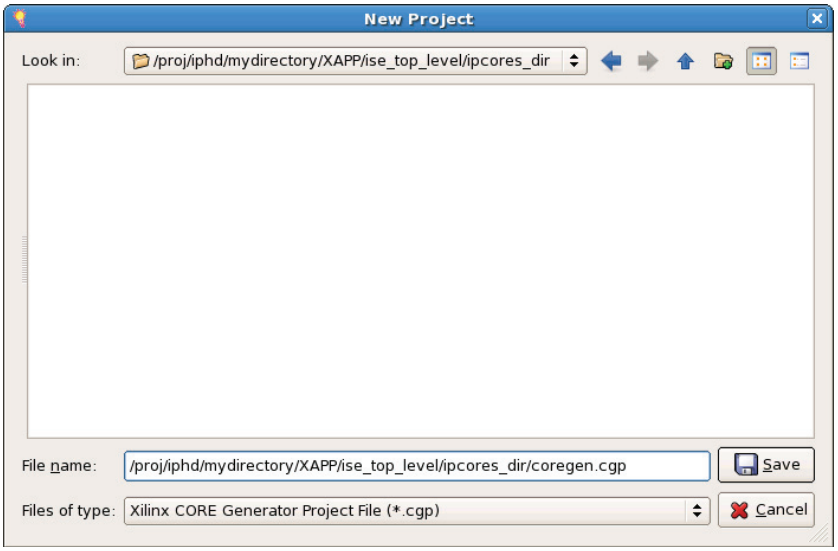


Figure 13: **CORE Generator Software - New Project Setup**

Set up the project options for the CORE Generator software—the same project options as the ones for the ISE tools—to generate the core (Figure 14).

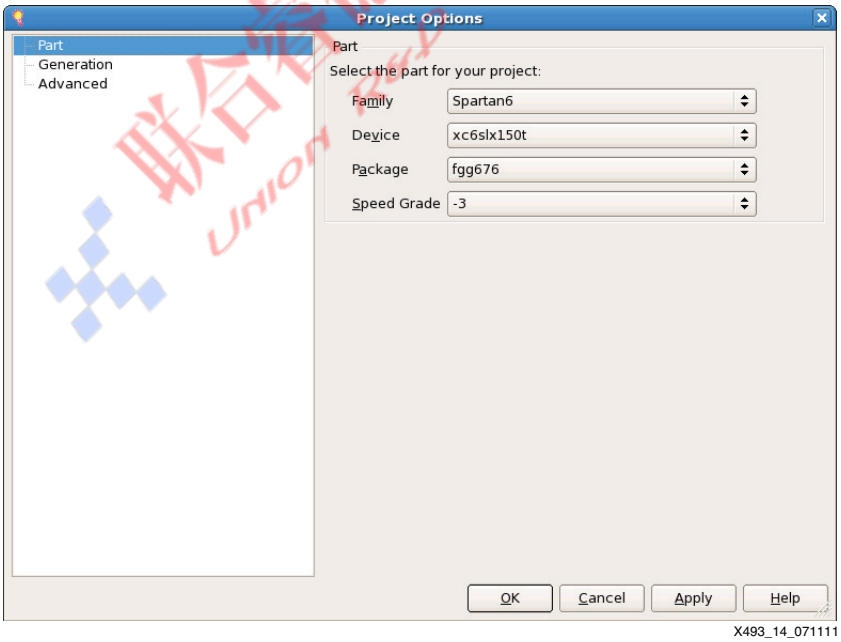


Figure 14: **CORE Generator Software - New Project Options**

In the CORE Generator software, navigate to the DisplayPort version 2.3 core found in the **Standard Bus Interfaces** → **DisplayPort** directory and double-click the entry.

> ***Note:*** If the core is not available, a license might be required. Information on obtaining a license can be found at http://www.xilinx.com/products/ipcenter/ipaccess_fee.htm.

To generate the transmitter and receiver cores, set the options as shown below (Figure 15):

- Set **Component Name** to **displayPort_v2_3**
- Select **Transmit Source Core** from the data flow direction radio buttons
- Set **Number of Lanes** to **4**
- Click **Generate**

Close the CORE Generator software after generation of the core is completed.

Note: AXI address information in the DisplayPort GUI need not be updated because the system memory map address is assigned while building the EDK system.

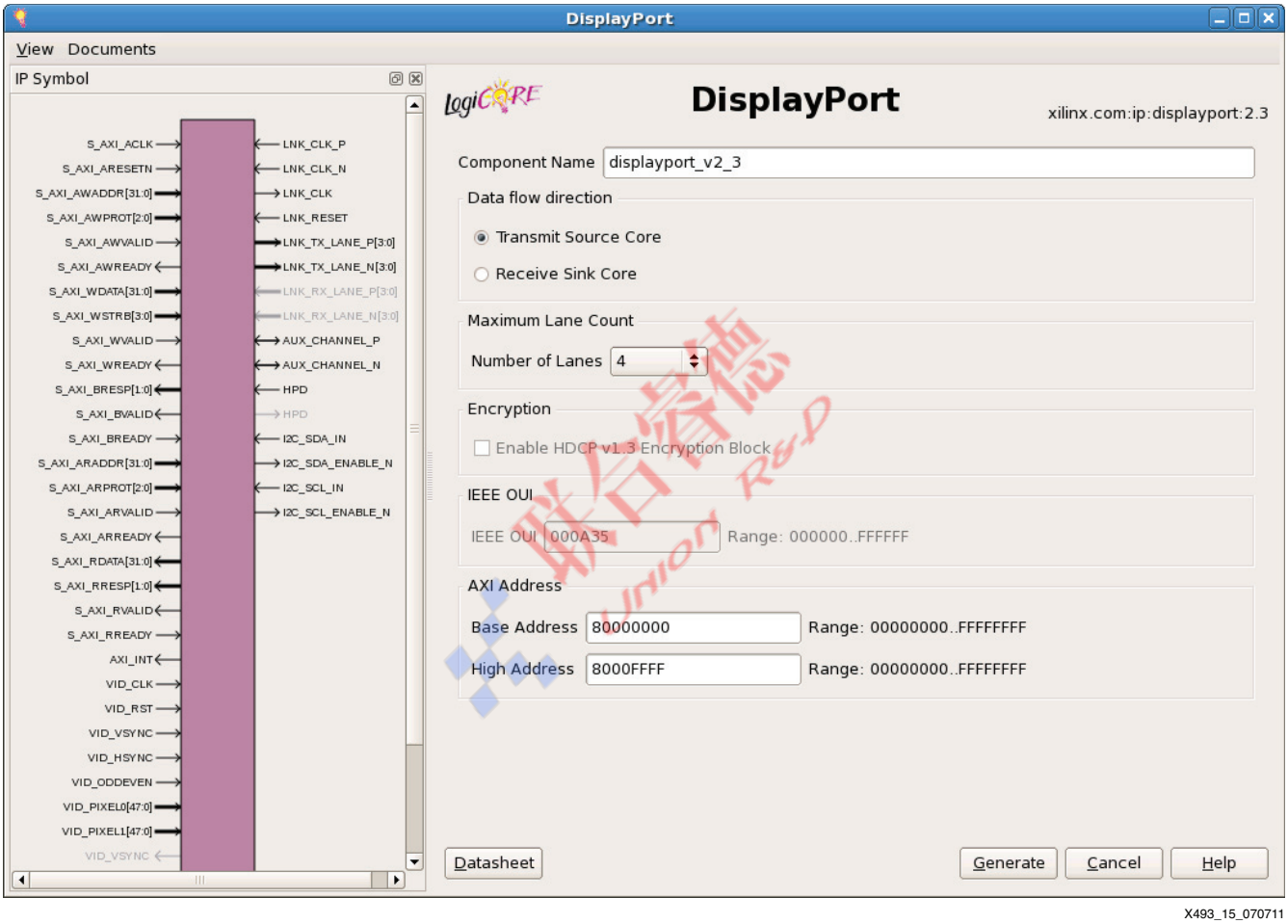


Figure 15: **DisplayPort LogiCORE IP Generation**

After the core is generated, it can be integrated into the ISE tools system. The generated core is in the `mydirectory/XAPP/ise_top_level/ipcore_dir/displayport_v2_3` directory. The example design provided with the LogiCORE IP is not used. Instead, it is replaced by the example design provided at `XAPP/design_files/displayport_tx_exdes.v`.

The new example design contains the MicroBlaze processor-based policy maker from this application note. The cyclic redundancy check (CRC) engines and clocking modules are also sourced from design files.

Right-click in the Hierarchy window within the ISE tools and select **Add Source**. Add these sources to the project:

```
mydirectory/XAPP/design_files/dcmspi.v
```

```
mydirectory/XAPP/design_files/displayport_tx_exdes.v
mydirectory/XAPP/design_files/patgen/crc_16_comp.v
mydirectory/XAPP/design_files/patgen/dp_test_pattern.v
mydirectory/XAPP/design_files/patgen/hdclrbar.v
mydirectory/XAPP/design_files/patgen/regs.v
mydirectory/XAPP/design_files/patgen/timing.v
mydirectory/XAPP/design_files/patgen/vid_crc_16.v
mydirectory/XAPP/design_files/patgen/video_pat_gen.v
mydirectory/XAPP/design_files/vid_clkgen.v
mydirectory/XAPP/ipcore_dir/displayport_v2_3/source/displayport_v2_3_tx.v
mydirectory/XAPP/ipcore_dir/displayport_v2_3/source/dport_txlink_top.v
mydirectory/XAPP/ipcore_dir/displayport_v2_3/source/dport_tx_phy.v
mydirectory/XAPP/ipcore_dir/displayport_v2_3/source/s6_gt_tile.v
mydirectory/XAPP/ipcore_dir/displayport_v2_3/source/s6_gt_wrapper.v
mydirectory/XAPP/design_files/defines.v
mydirectory/XAPP/design_files/displayport_tx.ucf
```

Note: For the Virtex-6 family, the following Virtex-6 FPGA GT wrapper files must be added instead of the Spartan-6 FPGA wrapper files:
`mydirectory/XAPP/ipcore_dir/displayport_v2_3/source/v6_gtx_wrapper_gtx.v` and `mydirectory/XAPP/ipcore_dir/displayport_v2_3/source/v6_gtx_wrapper.v`.

Note: DisplayPort NGC files can be directly added as source files to the project, or a macro search path can be used to point to the DisplayPort NGC directory. The DisplayPort NGC can be found in `mydirectory/XAPP/ipcore_dir/displayport_v2_3/dport_txlink_top.ngc`.

At this stage, the project should appear as shown in Figure 16. All necessary sources except the EDK subsystem have been added to the project.

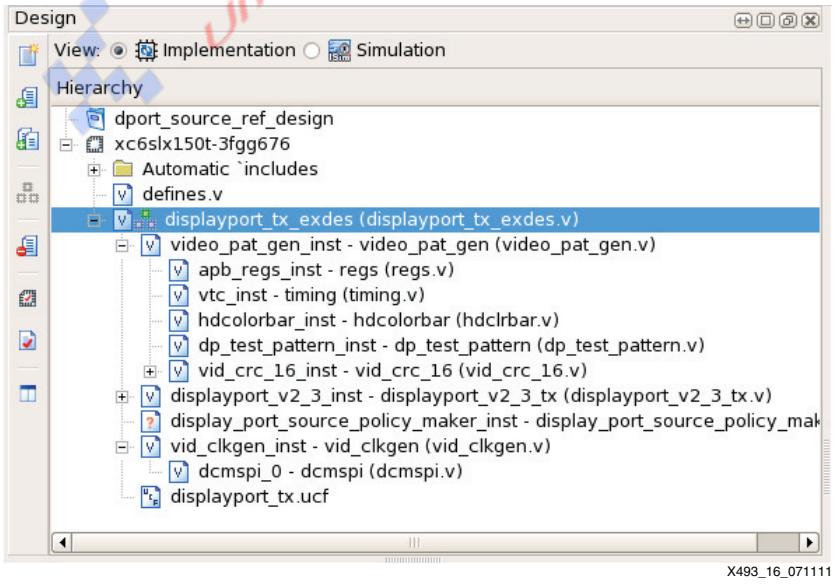


Figure 16: **Partial ISE Tools System**

Step 4: Create an EDK System

To add an EDK system to the ISE tools project, right-click in the Hierarchy window within the ISE tools and select **New Source**. In the New Source Wizard (Figure 17), select **Embedded Processor** as the source type. Set the File name to `display_port_source_policy_maker` and the Location to `mydirectory/XAPP`, where mydirectory is the directory where the XAPP folder was placed in Step 1: Set Up the Directory Structure. Ensure that the **Add to project** box is checked. Select **Next**, then **Finish**. Xilinx Platform Studio (XPS) is launched. If prompted to create a system using Base System Builder, select **No**.

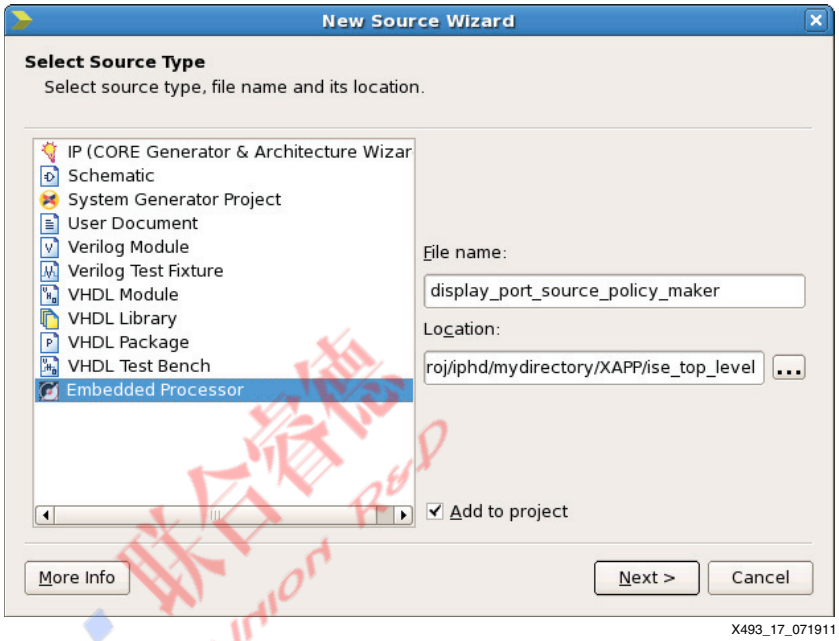


Figure 17: **ISE Tools New Source Wizard for EDK Insertion**

XPS should now be active, and an empty system should be shown. Open the IP Catalog, as shown in Figure 18. From this screen, all required IP is added to the system.

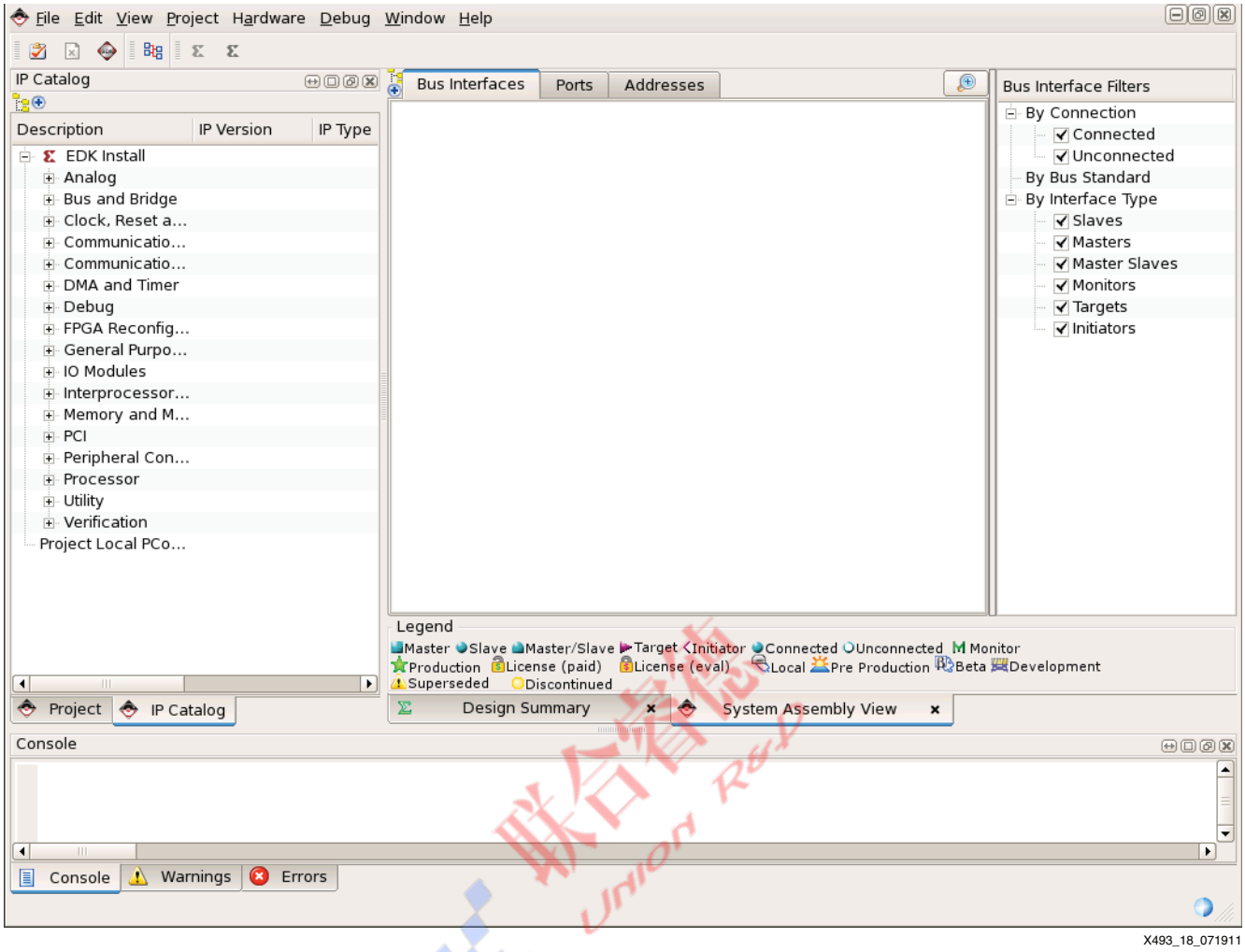


Figure 18: **EDK IP Catalog Selection**

Project Local pcores should be empty, and the AXI2APB bridge and AXI external slave connector IPs must be added. Copy the `axi_apb_bridge_v1_00_a` and `axi_ext_slave_conn_v1_00_a` folders from the reference design files in `XAPP/display_port_source_policy_maker/pcores/` to the current design `mydirectory/XAPP/display_port_source_policy_maker/pcores`. After the directory has been copied, click **Project** → **Rescan User Repositories** to refresh the IP Catalog. If asked to create a project using BSB wizard, select **No**.

Add the IP

Before adding the IP, click the System Assembly View tab to be able to see the IP as it is added. Add the cores listed in Table 29 to the system by locating them in the IP catalog and double-clicking them. After the IP names appear in the System Assembly View, rename them as shown in Table 29.

Table 29: **EDK Required IP**

IP Location	Version	Rename to:
Processor → MicroBlaze	8.20.a	`microblaze_1`
Bus and Bridge → AXI Interconnect	1.03.a	`axi4lite_0`
Bus and Bridge → Local Memory Bus (LMB) 1.0	2.00.b	`ilmb`

Table 29: **EDK Required IP *(Cont'd)***

IP Location	Version	Rename to:
Bus and Bridge → Local Memory Bus (LMB) 1.0	2.00.b	dlmb
Memory and Memory Controller → LMB BRAM Controller	3.00.b	ilmb_cntlr
Memory and Memory Controller → LMB BRAM Controller	3.00.b	dlmb_cntlr
Memory and Memory Controller → Block Ram (BRAM) Block	1.00.a	lmb_bram
Communication Low-Speed → AXI UART (Lite)	1.02.a	axi_uartlite_0
Debug → MicroBlaze Debug Module (MDM)	2.00.b	debug_module
Clock, Reset and Interrupt → Processor System Reset Module	3.00.a	proc_sys_reset_0
DMA and Timer → AXI Timer/Counter	1.02.a	axi_timer_0
Project Local pcores → USER → AXI APB Bridge	1.00.a	axi_apb_bridge_0
Project Local pcores → USER → AXI EXTERNAL SLAVE CONNECTOR	1.00.a	axi_ext_slave_conn_0
Communication Low-Speed → AXI IIC Interface(1)	1.01.a	iic_master

Notes:

1. IIC interface is required for the ML605 board with Avnet DVI I/O FMC module-based designs.

Connect the Buses

Navigate to the **Bus Interfaces** tab of the **System Assembly View** and connect all of the AXI bus connections, LMB bus connections, and DEBUG bus connections, as shown in Figure 19. This can be done by clicking the circles, squares, and triangles on the left side of the system assembly view or by using the pull-down menus in the Bus Name column. It might be useful to click the **+** located next to the **Bus Interfaces** tab to see the bus connectivity more clearly.

Note: For more details on using EDK, refer to EDK 13.2 documentation [Ref 7].

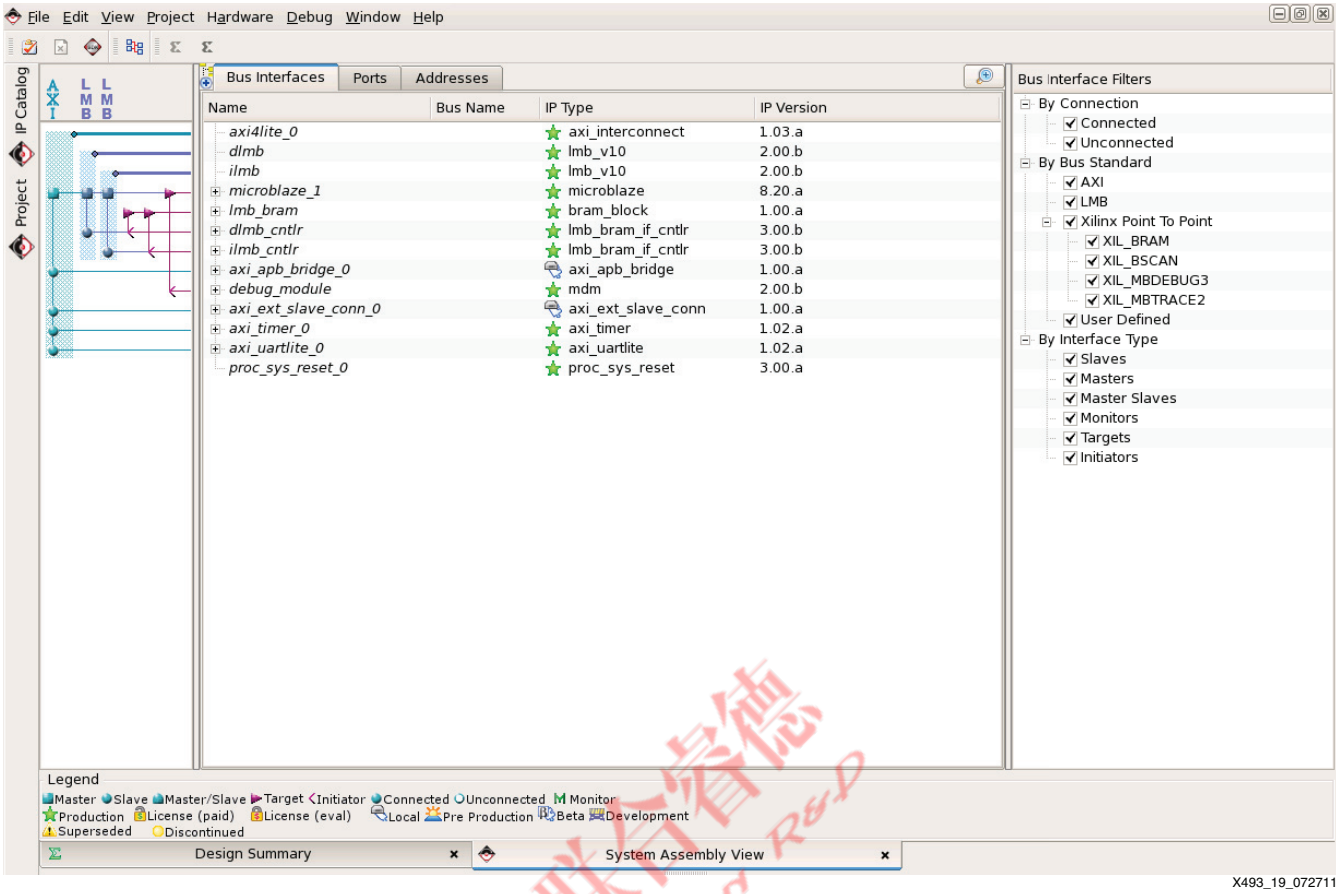


Figure 19: **EDK Bus Connectivity**

After all of the buses are connected, click the **Addresses** tab and click **Generate Addresses**.

Note: Ensure that `dlmb_cntlr` and `ilmb_cntlr` have a base address of `0x00000000` and a size of 64K.

Now that the bus connections are made, each IP needs to be configured. Navigate to the **Bus Interfaces** tab to begin configuring IP. Many of the IPs in the system are preconfigured or automatically configured. However, the MicroBlaze processor and UART Lite IPs must be configured for this system.

MicroBlaze Configuration

Double-click the **microblaze_1** IP, select **Minimum Area** and click **OK**, as shown in Figure 20. Select the **Enable Debug** option, and on Page 3 of the MicroBlaze wizard, select the **AXI bus interface** option.

Figure 20: **MicroBlaze Processor Configuration**

RS-232 UART Configuration

Double-click the **axi_uartlite_0** IP and select the **User** tab. Set the **UART Lite Baud Rate** to **115200**, set **Use Parity** to **FALSE**, and click **OK**. See Figure 21.

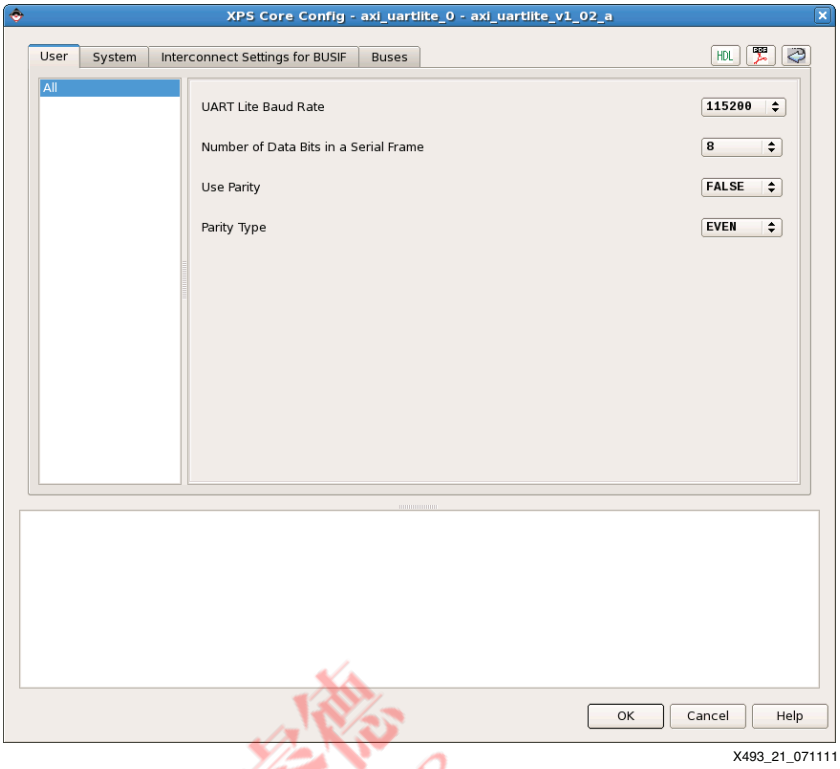


Figure 21: **RS-232 UART Configuration**

Debug Module Configuration

Double-click the **debug_module** IP, select the **UART** configuration, and deselect the **Enable JTAG UART** checkbox.

AXI External Slave Connector Module Configuration

Double-click the **axi_ext_slave_conn0** IP and select **AXILITE** as the C_S_AXI_PROTOCOL. Configure C_S_AXI_RNG00_BASEADDR and C_S_AXI_RNG00_HIGHADDR to **0xC3A00000** and **0xC3A0FFFF**, respectively, to create a 64K memory map range for the slave connected to the extender.

Port Connections

Now that the majority of the system is internally connected and configured, the external connections need to be added. To do this, open the `.mhs` file found on the **Project** tab and add the port declarations shown in Figure 22 on the line following PARAMETER VERSION.

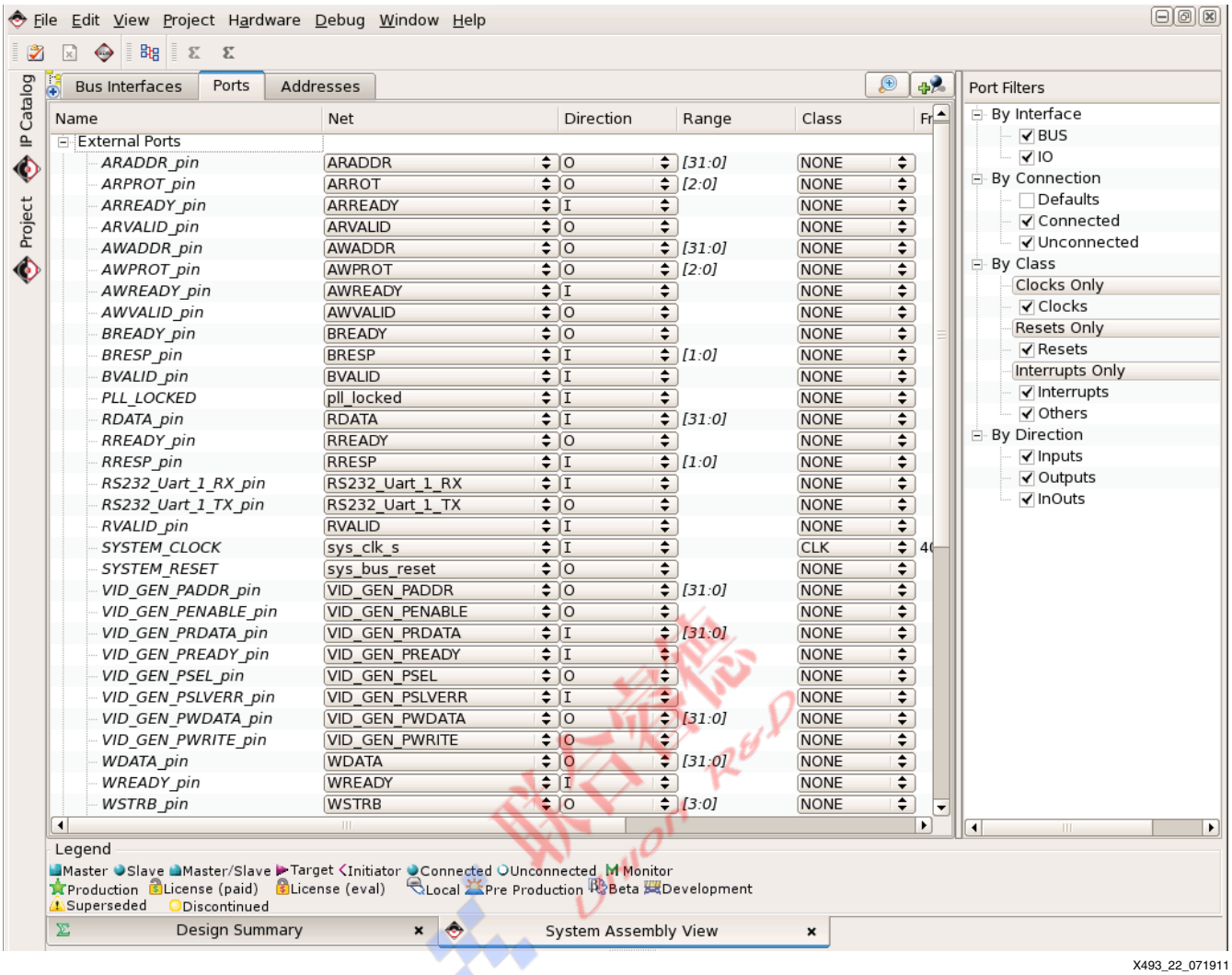


Figure 22: **EDK External Port Connections**

Save the file, return to the **System Assembly View**, and click the **Ports** tab.

On the **Ports** tab, expand the **External Ports** to see the newly added ports. The `sys_rst_pin` port has a reset polarity of 1, and the `SYSTEM_CLOCK` port has a frequency of 40 MHz. If the reset polarity or clock frequency change, these parameters must be changed to reflect that.

After all of the connections have been made, the `.mhs` file should appear very similar to the `.mhs` included in the reference design (`XAPP/display_port_source_policy_maker/display_port_source_policy_maker.mhs`). The XPS portion is now complete and is ready to be integrated into the top-level design.

XPS can be closed at this point and work can resume in the ISE tools. When returning to the ISE tools, `display_port_source_policy_maker_inst` should be populated with the EDK project as shown in Figure 23.

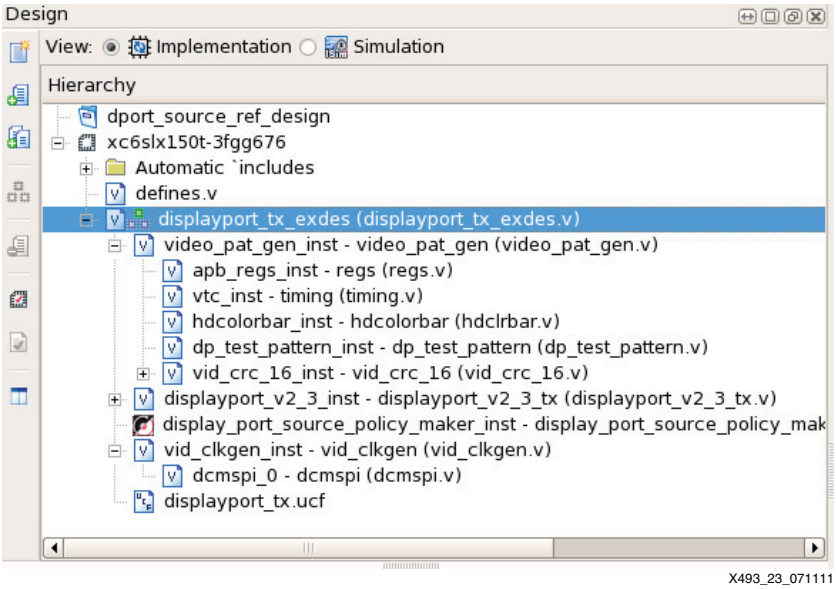


Figure 23: **Complete ISE Tools System**

Step 5: Generate the Bitstream

With the project now containing all required source files, the bitstream can be generated and the platform can be exported to SDK. The DisplayPort cores are generated as NGC files and these are read into the ISE tools as part of the ADD file (Figure 24).

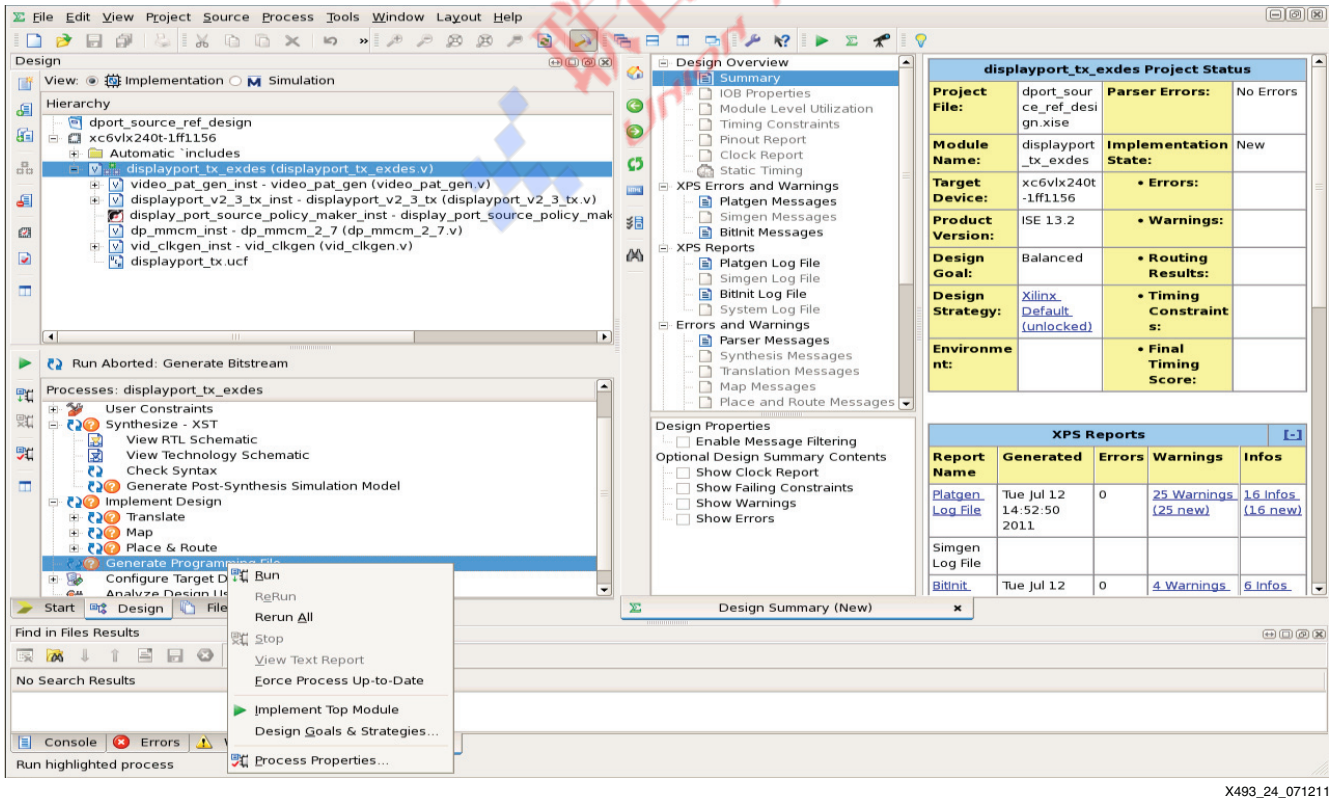


Figure 24: **Implementation Properties**

The design is now ready to be built. Double-click **Generate Programming File**. The design should run through synthesis, implementation, and bitstream generation.

The base hardware system is now built, but it contains no software for the MicroBlaze processor. To add software to the MicroBlaze processor, an SDK project must be created. First, however, the hardware design needs to be exported so that SDK has a reference system.

From the ISE tools project navigator, select **display_port_source_policy_maker_inst** and double click **Export Hardware Design to SDK with Bitstream** from the bottom-left pane, as shown in Figure 25.

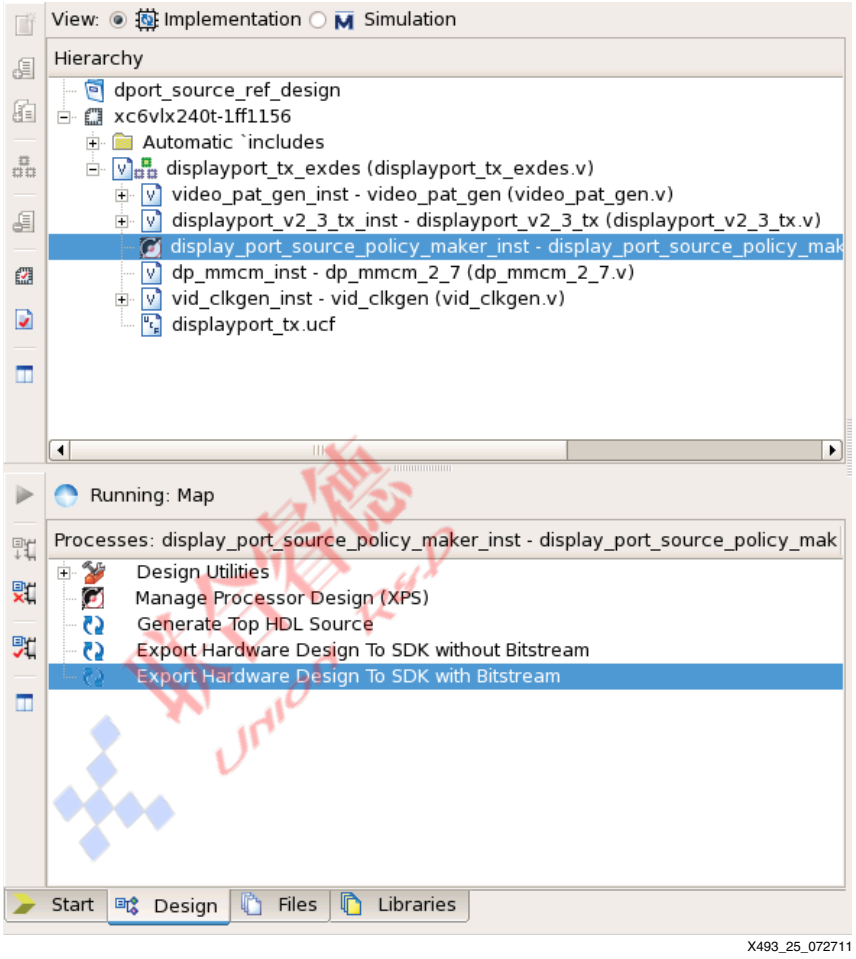


Figure 25: **Export Hardware Design to SDK**

Double-click **Export Hardware Design to SDK**. This creates an XML file in `mydirectory/display_port_source_policy_maker/SDK/SDK_Export/hw/` named `display_port_source_policy_maker.xml`. This XML file represents the EDK system and is used by SDK to create a hardware platform.

Step 6: Create an SDK Project

Open SDK and set the Workspace to `mydirectory/XAPP/sdk_workspace`.

In SDK, three components are needed to create a software project: a hardware specification, a board support package, and a C or C++ project. The three components are created as described below.

Xilinx Hardware Platform Specification

To create the hardware specification:

1. Click **File** → **New** → **Xilinx Hardware Platform Specification**.

2. Set the project name to `hw_platform_0`.
3. Set the target hardware specification to `mydirectory/XAPP/display_port_source_policy_maker/SDK/SDK_Export/hw/display_port_source_policy_maker.xml`.
4. Click **Finish**.

Xilinx Board Support Package

To create the board support package:

1. Click **File** → **New** → **Xilinx Board Support Package**.
2. Set the project name to `standalone_bsp_0`.
3. Set the hardware platform to `hw_platform_0`.
4. Set the board support package OS to `standalone`.
5. Click **Finish**.
6. In the Board Support Package Settings, ensure that `stdin` and `stdout` are set to `axi_uartlite_0`, as shown in Figure 26, then click **OK**.

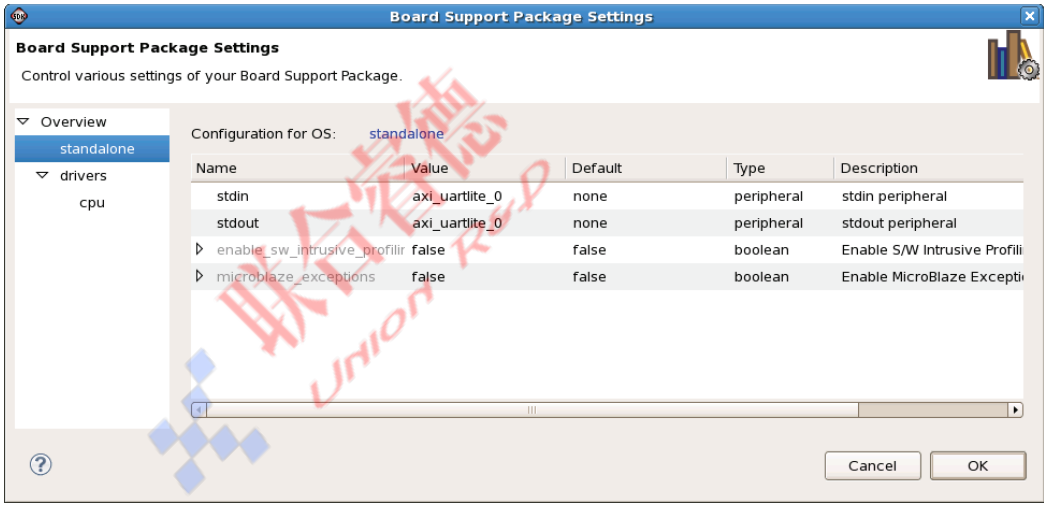


Figure 26: **Board Support Package Settings**

Xilinx C Project

To create the C project:

1. Click **File** → **New** → **Xilinx C Project**.
2. Set the project name to `dp_source_policy_maker_0`.
3. Set the project template to `Empty Application`.
4. Click **Next**.
5. Click the **Target an existing Board Support Package** radio button and select `standalone_bsp_0`.
6. Click **Finish**.

Adding the Code

SDK should now have three projects in the Project Explorer, as shown in Figure 27.

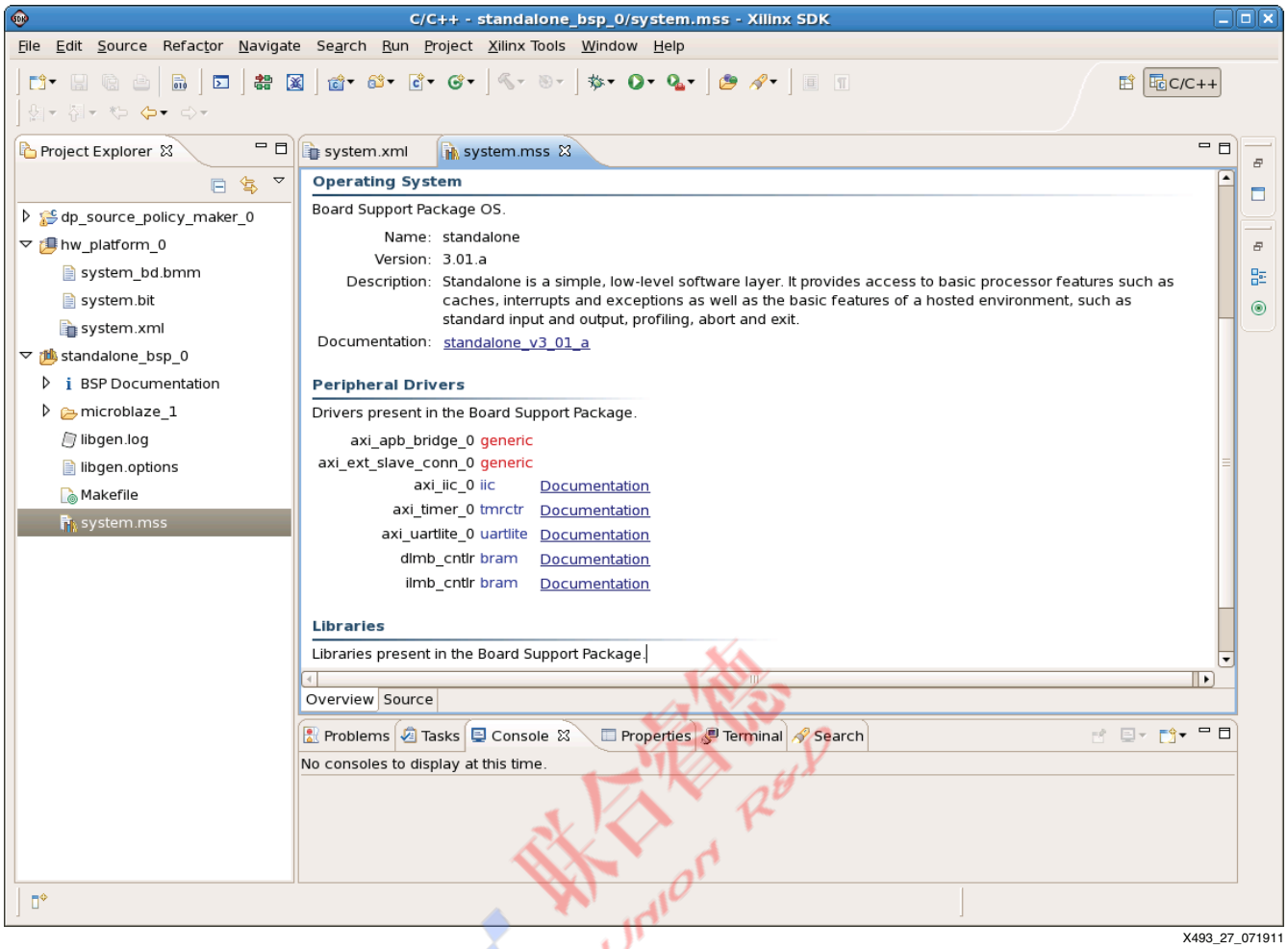


Figure 27: **SDK Project Explorer**

Using a console or folder browser, copy the source files (`*.c` and `*.h`) from the reference design folder `XAPP/sdk_workspace/dp_source_policy_maker_0/src` to `mydirectory/XAPP/sdk_workspace/dp_source_policy_maker_0/src`.

Go back to SDK and refresh the `dp_source_policy_maker_0/src` folder by clicking it in the Project Explorer window and pressing **F5**. The source should automatically compile and place `dp_source_policy_maker_0.elf` in the `mydirectory/XAPP/sdk_workspace/dp_source_policy_maker_0/Debug` directory.

Note: Set GCC Optimization to -1 by right-clicking the **dp_source_policy_maker_0** project, select **C/C++ Build Settings**, select **MicroBlaze gcc compiler** > **optimization**, and set the optimization level to **Optimize -01**.

Step 7: Update the Bitstream

There are several ways to update the bitstream with processor data. The method used for this application note is a post-processing step in SDK.

From the project explorer in SDK, right-click the `dp_source_policy_maker_0` project and select **C/C++ Build Settings**. Next, click the **Build Steps** tab and, as shown in Figure 28, set the **Command:** in the post-build steps to:

```
data2mem -bm ../../hw_platform_0/system.bmm -bt
../../hw_platform_0/system.bit -bd dp_source_policy_maker_0.elf -o b
../../hw_platform_0/download.bit
```

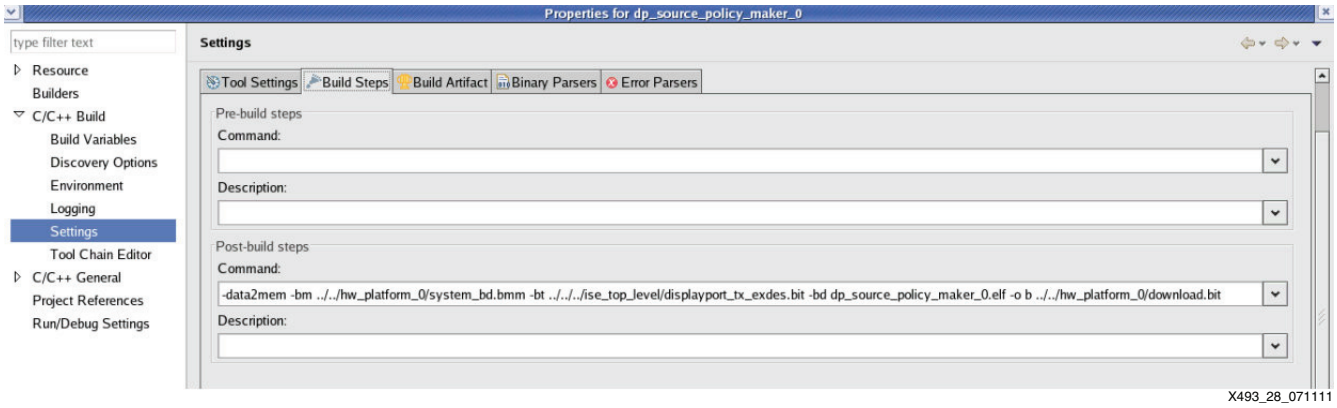


Figure 28: **SDK Post-Build Steps**

This creates a new `download.bit` file in `mydirectory/XAPP/sdk_workspace/hw_platform_0` every time the software is rebuilt. The `download.bit` can now be downloaded into the targeted FPGA system. Refer to Setup and Usage, page 20 for more information about using the reference design.

Reference Design

The reference design files for this application note can be downloaded at:

https://secure.xilinx.com/webreg/clickthrough.do?cid=147817

The checklist in Table 30 indicates the tool flow and verification procedures used for the reference design.

Table 30: **Reference Design Matrix**

Parameter	Description
General	
Developer Name	Xilinx
Target Devices (Stepping Level, ES, Production, Speed Grades)	Spartan-6 FPGA (XC6SLX150T-FGG676-3) Virtex-6 FPGA ((XC6VLX240T-FF1156-1)
Source Code Provided?	Yes
Source Code Format	Verilog, C
Design Uses Code or IP from Existing Reference Design, Application Note, 3rd party, or CORE Generator™ Software?	Yes DisplayPort LogiCORE IP, v2.3 with AXI
Simulation	
Functional Simulation Performed?	Yes
Timing Simulation Performed?	No
Testbench Provided for Functional and Timing Simulations?	No
Testbench Format	Verilog
Simulator Software and Version	ModelSim 6.6d
SPICE/IBIS Simulations?	No
Implementation	
Synthesis Software Tools and Version	ISE software, v13.2

Table 30: **Reference Design Matrix *(Cont'd)***

Parameter	Description
Implementation Software Tools and Version	EDK 13.2 SDK 13.2
Static Timing Analysis Performed?	Yes
Hardware Verification	
Hardware Verified?	Yes
Hardware Platform Used for Verification	Spartan-6 FPGA Consumer Video Kit Virtex-6 FPGA ML605 Evaluation Kit with Avnet DVI I/O FMC Module

Table 31 shows the resource utilization of the default policy maker reference design. The number of block RAMs used is for a more full-featured, user-driven console-based policy maker. Less featured software implementations are possible with less block RAM utilization.

Table 31: **Reference Design Footprint of the CVK 1.0 System**

IP	LUTs	Flip-Flops	Block RAMs
axi4lite_0_wrapper	461	590	0
axi_apb_bridge_0_wrapper	143	141	0
axi_timer_0_wrapper	212	264	0
axi_uartlite_0_wrapper	84	105	0
debug_module_wrapper	89	79	0
dlmb_wrapper	1	1	0
dlmb_cntlr_wrapper	2	6	0
ilmb_wrapper	1	1	0
ilmb_cntlr_wrapper	2	6	0
lmb_bram_wrapper	0	0	16
microblaze_1_wrapper	1,239	1,139	0
proc_sys_reset_0_wrapper	69	56	0
display_port_source_policy_maker	2,552	2,746	16
Reference design including DP core associated logic	10,868	6,140	20

References

1. *Spartan-6 FPGA Consumer Video Kit*
   http://www.xilinx.com/products/devkits/TB-6S-CVK.htm
2. *ML605 Evaluation Kit*
   http://www.xilinx.com/support/documentation/ml605.htm
3. *Avnet DVI I/O FMC Module*
   http://www.em.avnet.com/evk/home/0,1719,RID%253D0%2526CID%253D58704%2526CCD%253DUSA%2526SID%253D32214%2526DID%253DDF2%2526LID%253D32232%2526BID%253DDF2%2526CTP%253DEVK,00.html
4. *VESA DisplayPort Standard v1.1a*
   http://www.vesa.org

5. *AMBA Protocol Specifications Document Set*
http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.set.amba/index.html
6. UG767, *LogiCORE IP DisplayPort v2.3 User Guide*
7. EDK 13.2 Documentation
http://www.xilinx.com/support/documentation/dt_edk_edk13-2.htm

Conclusion

The reference design guides the user to create a DisplayPort source system using ISE, EDK, and SDK tools. Detailed descriptions of the feature-rich policy maker software are provided to enable the user with various features of the DisplayPort designs. Pre-verified system files provided with the application note help to ensure system connectivity and fast system bring-up.

Revision History

The following table shows the revision history for this document.

Date	Version	Description of Revisions
07/21/10	1.0	Initial Xilinx release.
09/16/11	2.0	Major updates throughout.

Notice of Disclaimer