XAPP593 (v1.0) September 16, 2011

DisplayPort Sink Reference Design

Authors: Arun Ananthapadmanaban and Vamsi Krishna

Summary

This application note describes the implementation of a DisplayPort™ sink core and policy maker reference design targeted for the Spartan®-6 FPGA Consumer Video Kit (CVK) [Ref 1].

Introduction

The purpose of this reference design is to implement the DisplayPort sink design and associated software policy maker in a MicroBlaze™ processor. The reference design is a loop-through system that receives video from a DisplayPort source via the receive link, buffers the video data, and retransmits it over the DisplayPort transmit link.

The policy maker performs several tasks such as initialization of GTP transceiver links, probing of registers, and other features useful for bring-up and use of the core. The application controls both the sink and source of the reference design and communicates with the monitor (sink) connected on the transmit port of the reference design using the auxiliary channel.

The reference design included with this application note encompasses the DisplayPort source and sink cores generated from the Xilinx® CORE Generator™ tool, policy maker, and a frame buffer logic using external memory. This application note focuses on the reference hardware and the policy maker implementation. The block diagram of the test system is shown in Figure 1.

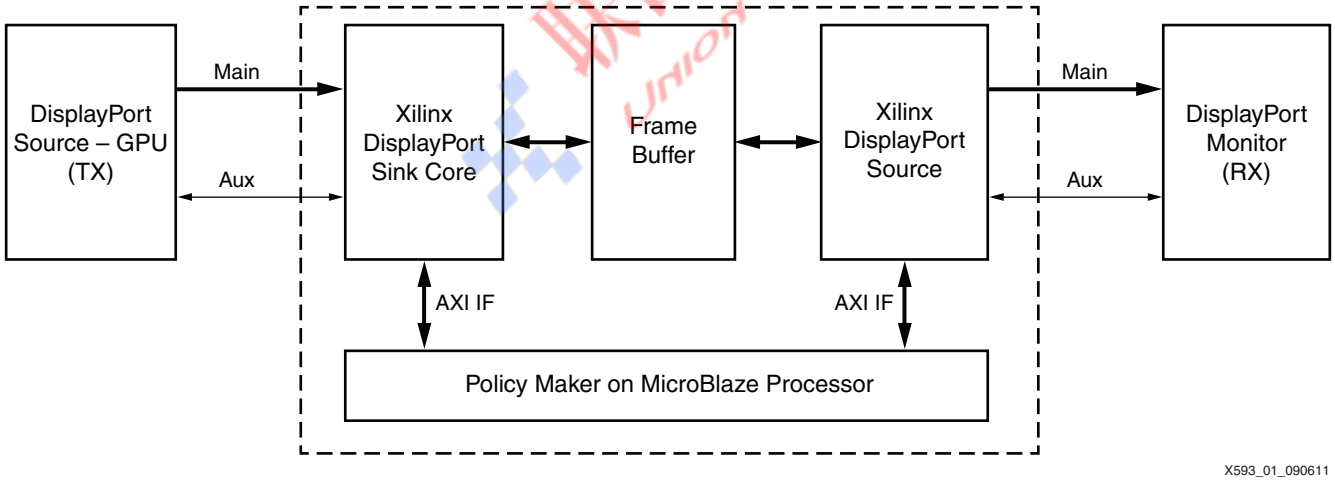


Figure 1: **System Block Diagram**

This application note allows the user to connect a graphics processing unit (GPU) to the Xilinx DisplayPort sink core, and loops the video to a monitor using the DisplayPort source core.

Hardware Implementation

The system consists of DisplayPort sink and source cores, a video timing 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.

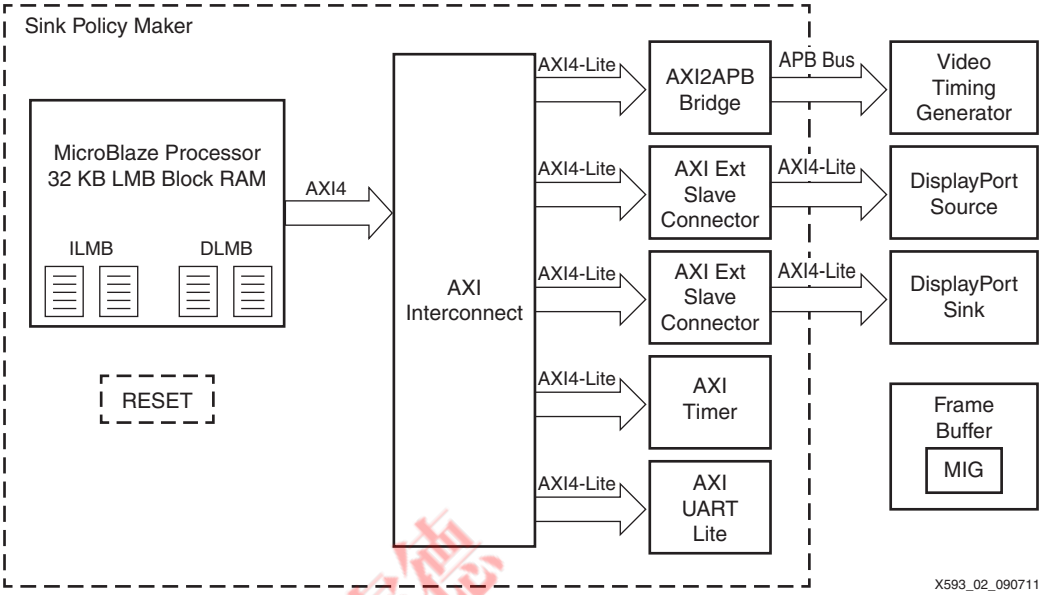


Figure 2: **EDK System Block Diagram**

The source and sink cores are 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 2].

The sink core is the master of the system in the sense that the link configuration and video rates are defined by what the sink core receives. The data from the sink core is copied to external DRAM via the frame buffer and Memory Interface Generator (MIG) logic. The control path functions are handled by the MicroBlaze policy maker, whose main function is to keep track of the sink core status and configure the source core with an identical configuration. The policy maker functions also include programming the video timing generator with the received video attributes and synthesis of a video clock for the source core.

The video timing is generated by the video timing generator while the data is read out of the stored frames in the external DRAM to create a video loop-through system.

Clocking

The TED CVK 1.0 consists of these clocks:

- Input clocks:
  - GTP reference clocks of 135 MHz for high bit rate (HBR) and 81 MHz for reduced bit rate (RBR)
  - System clock of 200 MHz
- Derived clocks:
  - AXI and MicroBlaze processor clock at 40 MHz derived from 200 MHz reference clock
  - RX video clock at 100 MHz derived from 200 MHz reference clock (DMA mode)

- DRAM clock at 200 MHz derived from 200 MHz reference clock
- TX video clock synthesized using DCM_CLKGEN from link clock

Frame Buffer

The frame buffer block interfaces to the sink and source video interface, and uses a MIG core to connect to the external DRAM memory. The MIG core is generated from the CORE Generator tool. For details, refer to the *Spartan-6 FPGA Memory Controller User Guide* [Ref 3]. The frame buffer uses the external DRAM to buffer four video frames and handles a mismatch in the sink and source video data rate by manipulating the frame pointer of the source video data to implement a circular buffer. The frame buffer:

- Maintains four frames of data in external memory.
- Packs pixel data on each pixel interface to optimize DRAM utilization. This packing is based on the video bits per color (BPC) configuration.
  - For 6, 8, and 10 bits, two pixels are packed into one 64-bit word of the memory. Thus, the memory QWORD count is half of the horizontal resolution (HRES).
  - For 12 and 16 bits, one pixel and one component of the next pixel is packed into one 64-bit word of the memory. Thus, the memory QWORD count is 3/4 of the HRES.
- Handles frame synchronization between the sink and source cores.
  - Frame sync logic adjusts TX frame pointers to adjust rate mismatches between the sink and source cores. If the sink core video rate is faster than the source core, the logic skips a frame while reading out. If the sink core video rate is slower than the source core, the logic repeats a frame while reading out.

Frame CRC

The system also has a CRC8 engine on the output of the sink core (pixel data) and another at the input of the source core. These cyclic redundancy check (CRC) engines compute frame CRC per color component on a per-pixel interface basis. The computed CRC values can be used to debug any pixel data mismatch. Matching CRC computed across the frame buffer, i.e., on the sink output and source input, guarantees data integrity of the frame buffer logic and external memory.

Table 22 and Table 23 list reference CRC values for link layer compliance (LLC) patterns (LLC Ramp) at various resolutions.

Software Implementation

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 addresses of these DisplayPort cores are referenced multiple times throughout the software as `XILINX_DISPLAYPORT_TX_BASE_ADDRESS` and `XILINX_DISPLAYPORT_RX_BASE_ADDRESS`, which are in turn referenced from `sys_defs.h`.

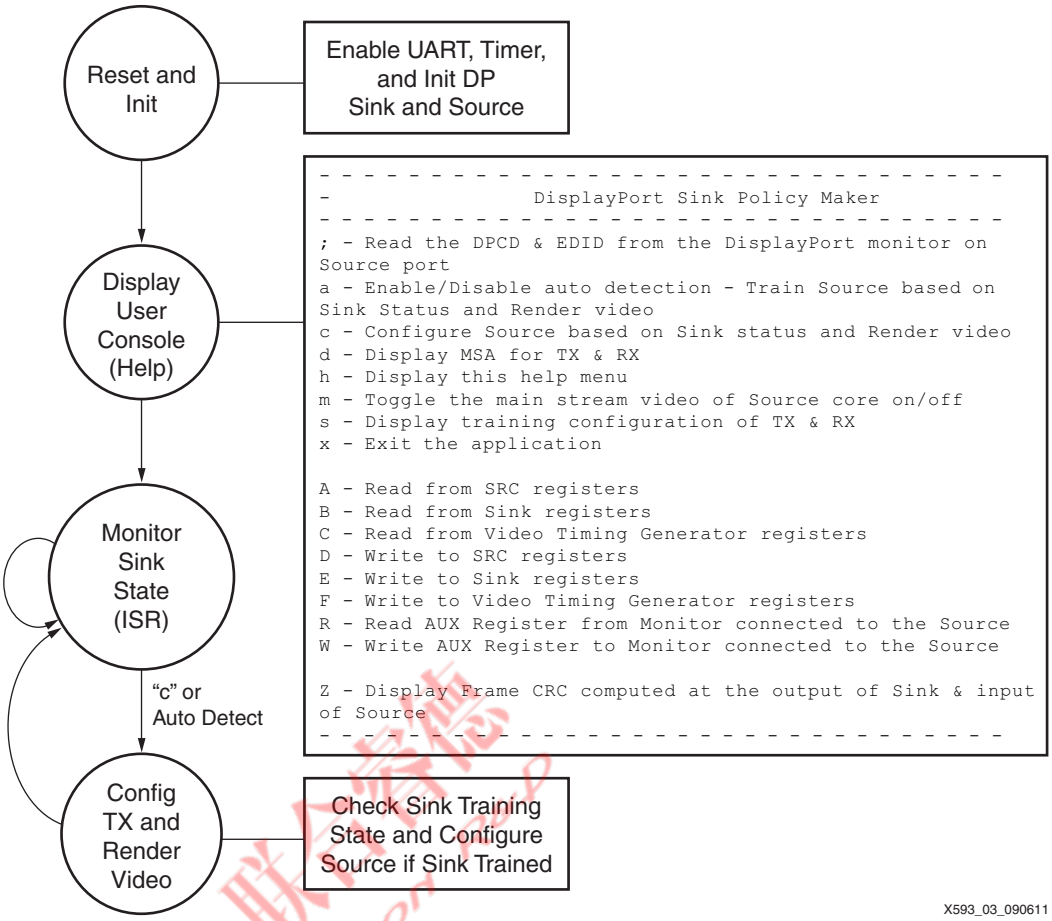


Figure 3: **Software Flow Diagram**

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, then checks the DisplayPort cores in the system to ensure that both a DisplayPort source and sink core are present. The DisplayPort core type is read from address offset `0xFC` of the DisplayPort sink and source cores.

For the physical layer (PHY) module:

1. Put the PHY into reset.
2. Wait for the PHY to be ready.
3. Bring the PHY out of reset.

For the sink core:

1. Disable the receiver.
2. Set the clock divider.
3. Set the minimum voltage swing required for training (based on the GTP PHY).
4. Enable the receiver.
5. Unmask all interrupts.

For the source core:

1. Disable the transmitter.
2. Set the clock divider.

3. Set the DisplayPort clock speed.
4. Enable the transmitter.
5. Unmask all interrupts.

At this point, the init_platform function in `main.c` has completed. Next, the xilcccAppInit (`xil_ccc_app.c`) function is called from `main.c`. Inside xilcccAppInit, the dplpmInitLinkPolicyMaker (`displayport_lpm.c`) function is called. The dplpmInitLinkPolicyMaker function is responsible for setting the hardware capabilities of the system. These settings control the start-up functionality of the transmitter before the terminal is active.

Display User Console

The user console is displayed after the system is initialized. However, the DisplayPort link is not yet active. Because the sink core is the master, the link becomes active only if the sink is trained. The software comes up with auto detection mode disabled, i.e., the system does not expect a monitor connected to the source port (Table 1). The functionality of each terminal command is described in the Command Processor, page 6 section.

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

Displayed on Terminal

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
-                DisplayPort Sink Policy Maker                          -
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
; - Read the DPCD & EDID from the DisplayPort monitor on Source port
a - Enable/Disable auto detection - Train Source based on Sink Status and Render video
c - Configure Source based on Sink status and Render video
d - Display MSA for TX & RX
h - Display this help menu
m - Toggle the main stream video of Source core on/off
s - Display training configuration of TX & RX
x - Exit the application

A - Read from SRC registers
B - Read from Sink registers
C - Read from Video Timing Generator registers
D - Write to SRC registers
E - Write to Sink registers
F - Write to Video Timing Generator registers
R - Read AUX Register from Monitor connected to the Source
W - Write AUX Register to Monitor connected to the Source

Z - Display Frame CRC computed at the output of Sink & input of Source
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Monitor Sink State, Configure, and Render Video (Auto Detect Mode)

This part of the code works on polling the interrupt status register of the sink core. The basic functions are based on the various interrupts as shown in Figure 4, and are implemented as a simple state machine in the `dpSinkISR`() routine. Figure 4 shows the switching of the routine based on interrupt status.

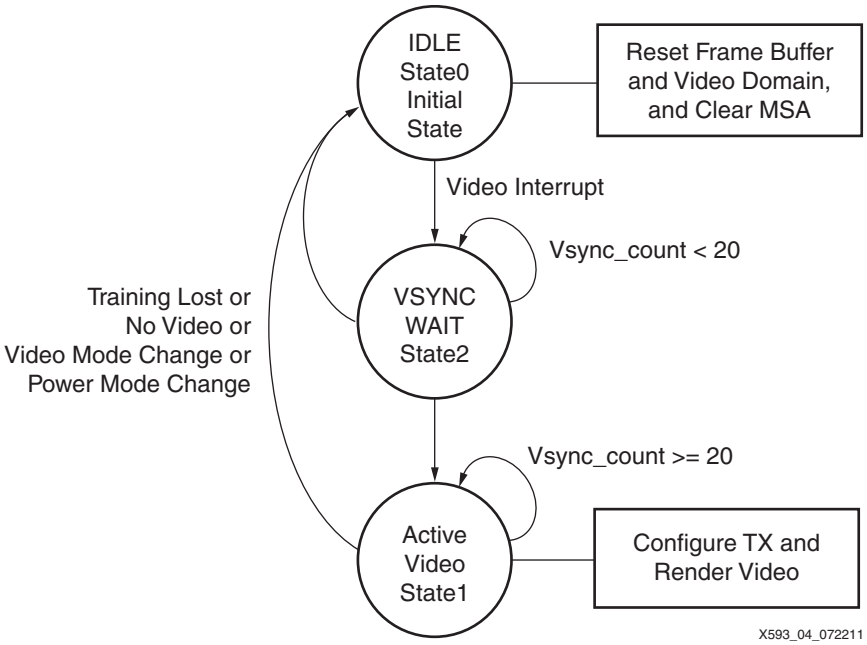


Figure 4: **Sink Interrupt Handler**

Types of interrupts include:

- No video interrupt: This interrupt indicates that the sink has detected that the main link has no video information. The routine disables the timing generator and frame buffer on this interrupt.
- Power mode interrupt: Power-down/power-up mode information is displayed. The PHY is put into power down or brought up by the link layer hardware, and there is no specific software action required, except that the software state machine is moved to *not trained* and *no video* when powered down.
- Video interrupt: When auto detect mode is enabled, the sink interrupt handler enters the vsync wait state on seeing a video interrupt. In this state, the routine waits for a pre-determined number of vertical blanking interrupts before moving to an active video state.

Command Processor

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

; — Read the DPCD and EDID from the DisplayPort Monitor on Source Port

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 is an example of what would be displayed when the command is executed. The command also dumps raw EDID data on the terminal. This EDID data 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: **DPCD Status Description**

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 (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	For full details, refer to the *VESA DisplayPort Standard* [Ref 4].
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:		
Has EDID	: No	

Table 3: **DPCD Status Description *(Cont'd)***

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

a — Enable/Disable Auto Detection - Train Source based on Sink Status and Render Video

This command enables or disables the ability to auto detect a monitor connected on the source core. By default, auto detection is disabled. When auto detection mode is enabled, the software that tracks the status in the sink core auto configures the source core with the link configuration and video configuration (see Table 4).

Table 4: **Terminal Display on Auto Detect Mode**

Displayed on Terminal When Video Mode is Changed and Auto Detect is Enabled
Sink :: Detected video mode change ..Entering VSYNC wait state = = = = = = = = = = TX set to RX video parameters = = = = = = = = = = = = = = = = = = = = = = Video Clock Synthesis = = = = = = = = = = = = Required Vid_freq = 81000, Computed Vid_freq = 81000, Err = 0, Ref_freq = 81000 M = 257 D = 257 = = = = = = = = = = = = Source TU management cfg = = = = = = = = = = = link_freq = 162, lanes = 4, link bandwidth = 648000 KBps vid_clk = 162000, hres = 1600, vres = 1200, bpc = 8, video bandwidth req = 486000 KBps tu_size = 64, avg_bytes_per_tu = 48000, frac = 0, min_bytes_per_tu = 48 = = = = = = = = = = = Render Video ..... = = = = = = = = = = = = = =

c — Configure Source based on Sink Status and Render Video

This command checks the sink training and video status and configures the source core to render video. This is a manual mode of the **a** command. This command displays video-related information on the terminal as shown in Table 5.

Table 5: **Terminal Display on Manual Configuration of Source - Monitor**

Displayed on the Terminal
= = = = = = = = = = TX set to RX video parameters = = = = = = = = = = = = = = = = = = = = = = Video Clock Synthesis = = = = = = = = = = = = Required Vid_freq = 65001, Computed Vid_freq = 65000, Err = 1, Ref_freq = 81000 M = 65 D = 81 = = = = = = = = = = = = Source TU management cfg = = = = = = = = = = = link_freq = 162, lanes = 4, link bandwidth = 648000 KBps vid_clk = 65001, hres = 1024, vres = 768, bpc = 8, video bandwidth req = 195003 KBps tu_size = 64, avg_bytes_per_tu = 19259, frac = 259, min_bytes_per_tu = 19 = = = = = = = = = = = Render Video ..... = = = = = = = = = = = = =

d — Display MSA for RX and 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 6 and Table 7.

Table 6: **Terminal Display for Display MSA Command**

Displayed on Terminal

```
SINK MSA Received Values
    HRES        : 1024
    HPOL        : 1
    HSWIDTH     : 136
    HTOTAL      : 1344
    VHEIGHT     : 768
    VSPOL       : 1
    VSWIDTH     : 6
    VTOTAL      : 806
    MISC0       : 0020
    MVID        : 13146
    NVID        : 32768
    VBID        : 0
 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 Timing Generator config
    VPOL        : 1
    HPOL        : 1
    DEPOL       : 0
    VSWIDTH     : 6
    VB          : 29
    VF          : 3
    VRES        : 768
    HSWIDTH     : 136
    HB          : 160
    HF          : 24
    HRES        : 1024
```

Table 7: **MSA Command Description**

Displayed on Terminal		Description
Main Stream Attributes RX (SINK MSA Received Values)		
HRES	: 1920	Active video horizontal resolution.
HPOL	: 0	Polarity of VSYNC and HSYNC, 1 = High, 0 = Low.
HSWIDTH	: 44	Number of clock cycles HSYNC is asserted.
HTOTAL	: 2200	Total number of clock cycles for front porch + HSYNC + back porch + active video.
VHEIGHT	: 1080	Active video vertical resolution (line count).

Table 7: **MSA Command Description *(Cont'd)***

Displayed on Terminal		Description
VSPOL	: 0	Polarity of VSYNC.
VSWIDTH	: 5	Number of HSYNCs that the VSYNC is asserted.
VTOTAL	: 1125	Number of HSYNCs of the frame including blanking times.
MISC0	: 0x0020	Miscellaneous 0 register.
MVID	: 30038	PLL multiplier for stream clock recovery.
NVID	: 32768	PLL divider for stream clock recovery.
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	: 0x0000	Miscellaneous 1 register from DisplayPort specification.
User Pixel Width	: 2	Refer to *LogiCORE IP DisplayPort v2.3 User Guide* [Ref 6] 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 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. (HRES x 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.

Table 7: **MSA Command Description *(Cont'd)***

Displayed on Terminal		Description
HSWIDTH	: 22	HSYNC width.
HB	: 74	Horizontal back porch.
HF	: 44	Horizontal front porch.
HRES	: 960	Horizontal resolution.

h - Display this help menu

This command displays the help menu, as shown in Table 5, page 8.

m - Toggle the main stream video of Source core on/off

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

s — Display Training Configuration of TX & RX

This command displays the DisplayPort sink training information and configuration data of the DisplayPort monitor connected to the source port Table 8.

Table 8: **Terminal Display for DPCD Status Command**

Displayed on Terminal
[SINK] Training Status Link BW Set : 0x06 Lane Count Set : 0x04 DPCD_LANE01 : 0x77 DPCD_LANE23 : 0x77 [SOURCE] Monitor Training Status: 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 [SOURCE] 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 9 provides detailed descriptions of the training configuration registers of the TX and RX.

Table 9: **Sink and Source Status Command Description**

Displayed on Terminal		Description
[SINK] Training Status (Sink Core Status)		
Link BW Set	: 0x06	LINK_BW_SET: Main link bandwidth 0xA = HBR 2.7G 0x6 = RBR 1.62G

Table 9: **Sink and Source Status Command Description *(Cont'd)***

Displayed on Terminal		Description
Lane Count Set	: 0x04	LANE_COUNT_SET: Lane count
DPCD_LANE01	: 0x77	LANE0_1_STATUS: Lane0 and Lane1 status
DPCD_LANE23	: 0x77	LANE2_3_STATUS: Lane2 and Lane3 status
[SOURCE] 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 has changed status. Bit 7 = LINK_STATUS_UPDATED Link status and adjust request updated since 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 9: **Sink and Source Status Command Description *(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 as in 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 9: **Sink and Source Status Command Description *(Cont'd)***

Displayed on Terminal		Description
(0x0102) Training Pattern Set	: 0x00	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 zeros. 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. Bit 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 is identical to that of TRAINING_LANE0_SET.)
(0x0105) Training Lane 2 Set	: 0x10	TRAINING_LANE2_SET (Bit definition is identical to that of TRAINING_LANE0_SET.)

Table 9: **Sink and Source Status Command Description *(Cont'd)***

Displayed on Terminal		Description
(0x0106) Training Lane 3 Set	: 0x10	TRAINING_LANE3_SET (Bit definition is identical to that of TRAINING_LANE0_SET.)
(0x0107) Downspread Ctrl	: 0x00	DOWNSPREAD_CTRL: Down-spreading control Bit 3:0 = RESERVED. Read all 0s. Bits 4 = SPREAD_AMP Spreading amplitude 0 = No downspread 1 = Equal to or less than 0.5% down spread

x — Exit the Application

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

A — Read from SRC Registers

This command allows direct read access to the registers inside the DisplayPort source LogiCORE IP (Table 10). See the Source Core Architecture chapter of *LogiCORE IP DisplayPort v2.3 User Guide* [Ref 6] for a description of each register.

Table 10: **Terminal Display for Read Source Register Command**

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

B — Read from Sink Registers

This command allows direct read access to the registers inside the DisplayPort sink LogiCORE IP (Table 11). A description of each register appears in the Sink Configuration Space chapter of *LogiCORE IP DisplayPort v2.3 User Guide* [Ref 6].

Table 11: **Terminal Display for Read Sink Register Command**

Displayed on the Terminal
Enter 4 hex characters: Sink Read address 0x000c Sink Read Addr C4A0000C Read Data: 0003

C — Read from Video Timing Generator Registers

This command allows direct read access to the registers inside the timing generator (Table 12). Refer to Table 19 for a description of each of the registers. Frame buffer control registers are also part of this register map.

Table 12: **Terminal Display for Read Video Timing Generator Register Command**

Displayed on the Terminal
Enter 4 hex characters: Video Timing Gen Read address 0x0100 Video Timing Gen Read Addr CCA00100 Read Data: 0981

D — Write to SRC Registers

This command allows direct write access to the registers inside the DisplayPort source LogiCORE IP (Table 13).

Table 13: **Terminal Display For Write to Source Register Command**

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

E — Write to Sink Registers

This command allows direct write access to the registers inside the DisplayPort sink LogiCORE IP (Table 14).

Table 14: **Terminal Display for Write to Sink Register Command**

Displayed on the Terminal
Enter 4 hex characters: Sink Write address 0x000c
Enter 4 hex characters: Sink Write data 0x0003
Sink Write Addr C4A0000C Write Data: 0003

F — Write to Video Timing Generator Registers

This command allows direct write access to the registers inside the timing generator (Table 15).

Table 15: **Terminal Display for Write to Video Timing Generator Register Command**

Displayed on the Terminal
Enter 4 hex characters: Video Timing Gen Write address 0x0100
Enter 4 hex characters: Video Timing Gen Write Data 0x0981
Video Timing Gen Write Addr CCA00100 Write Data: 0981

R — Read AUX Register from Monitor Connected to the Source

This command allows read access to the DisplayPort configuration data register space of the DisplayPort monitor connected to the source using the AUX channel (Table 16). Detailed descriptions of the registers can be found in the DisplayPort specification in the address mapping for the DPCD table [Ref 4].

Table 16: **Terminal Display for Aux Read from Monitor on Source Command**

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

W — Write AUX Register to Monitor Connected to the Source

This command allows write access to the DisplayPort configuration data register space of the DisplayPort monitor, connected to the source using the AUX channel (Table 17). Detailed descriptions of the registers can be found in the DisplayPort specification in the address mapping for the DPCD table [Ref 4].

Table 17: **Terminal Display for Aux Write to Monitor on Source Command**

Displayed on the Terminal
Enter 4 hex characters: Aux Write Address 0x0100
Enter 2 hex characters: Aux Write Data 0x06
Aux Write Addr 0100 Write Data: 06

Z — Display Frame CRC Computed at the Output of Sink and Input of Source

This command reads out the frame CRC8 computed at the video interfaces of the DisplayPort sink and source cores (Table 18).

Table 18: **Terminal Display and Description for Display Frame CRC Command**

Displayed on Terminal	Description
HRES 1024, VRES 768, MISCO 20, ERROR 0 CRC values Pixel 0 interface RX Rcrc 0x54DF, Gcrc 0xBE68 Bcrc 0xADDA TX Rcrc 0x54DF, Gcrc 0xBE68 Bcrc 0xADDA CRC values Pixel 1 interface RX Rcrc 0x783D, Gcrc 0xDCCE Bcrc 0x3F82 TX Rcrc 0x5387, Gcrc 0x71B3 Bcrc 0x8E36	RX CRC corresponds to the frame CRC values seen on sink video outputs. TX CRC corresponds to the frame CRC values seen on source video inputs. CRC engines are available on a per pixel interface basis.

Video Timing Generator and Frame Buffer Control Registers

The video timing generator and frame buffer control registers map is shown in Table 19.

Table 19: **Video Timing 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
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
0x100	R/W	Frame buffer and timing control register. Bit 0 = Enable frame buffer Bit 1 = RX video reset Bit 2 = DRAM reset Bit 8 = Enable frame synchronization logic. This logic matches frame rates (due to video clock ppm variations) between the sink and source cores by adjusting the frame start pointer of the source.(1) Bit 9 = When set to 1, this register forces frame buffer logic to start buffering unconditionally Bit 10 = When set to 1, this register configures frame buffer to skip four RX video frames before buffering data to memory. Bit 11 = Use RX VSYNC leading edge for timing and control. Bit 12 = Use RX HSYNC leading edge for timing and control.

Table 19: **Video Timing Generator Registers *(Cont'd)***

Register Address	Read/Write	Description
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
0x300	R/W	Bits 15:0 = Frame buffer base address MSB. Base address of the DRAM memory. Set to 0x0 by default. Usage: base_addr = {Frame buffer base address MSB[15:0], 16'b0}
0x304	R/W	Bits 15:0 = Frame offset address MSB. Address offset between frames. Usage: frame_offset = {Frame offset address MSB[15:0], 16'b0}
0x308	R/W	Bits 15:0 = Line offset. Address offset between lines of a frame.
0x30c	R	Frame buffer status. MIG and frame buffer status. Bit 0 = p0_mcb_cmd_full Bit 1 = p0_mcb_wr_underrun Bit 2 = p2_mcb_cmd_full Bit 3 = p2_mcb_wr_underrun Bit 4 = p1_mcb_cmd_full Bit 5 = p1_mcb_rd_empty Bit 6 = rxp_fifo_overflow Bit 7 = p3_mcb_cmd_full Bit 8 = p3_mcb_rd_empty Bit 9 = txp_fifo_underflow Bit 10 = p1_mcb_rd_overflow Bit 11 = p3_mcb_rd_overflow Bit 14:12 = frame_sync_count Bit 15 = calib_done
0x310	R/W	Frame buffer BPC. Same as BPC definition on MISC0.
0x31c	R/W	Memory QWORD count – Number of 64-bit words per line of data, after packing and based on BPC For BPC 6, 8, and 10: TX HRES QWORD count = HRES/2 For BPC 12 and 16: TX HRES QWORD count = HRES x 3/4 (2)
0x400 (3)	R	Sink frame CRC for R component, on Pixel-0 interface
0x404	R	Sink frame CRC for G component, on Pixel-0 interface
0x408	R	Sink frame CRC for B component, on Pixel-0 interface
0x40c	R	Sink frame CRC for R component, on Pixel-1 interface
0x410	R	Sink frame CRC for G component, on Pixel-1 interface
0x414	R	Sink frame CRC for B component, on Pixel-1 interface
0x420	R	Source frame CRC for R component, on Pixel-0 interface
0x424	R	Source frame CRC for G component, on Pixel-0 interface
0x428	R	Source frame CRC for B component, on Pixel-0 interface

Table 19: **Video Timing Generator Registers *(Cont'd)***

Register Address	Read/Write	Description
0x42c	R	Source frame CRC for R component, on Pixel-1 interface
0x420	R	Source frame CRC for G component, on Pixel-1 interface
0x424	R	Source frame CRC for B component, on Pixel-1 interface

Notes:

1. Frame synchronization logic adjusts TX frame pointers to adjust rate mismatches between the sink and source cores. If the sink core video rate is faster than the source core, the logic skips a frame while reading out. If the sink core video rate is slower than the source core, the logic repeats a frame while reading out.
2. The frame buffer packs pixel data based on BPC to write to a 64-bit DDR2 DRAM interface. For 6, 8, and 10 bits, two pixels are packed into one 64-bit word of the memory. Thus, the QWORD count is half of the HRES. For 12 and 16 bits, one pixel and one component of the next pixel are packed into one 64-bit word of the memory. Thus, the QWORD count is 3/4 of the HRES.
3. CRC-8 engines are present at the output of the sink core and input of the source core on a per-pixel interface. Matching the CRC computed across the frame buffer (on the sink output and source input) guarantees data integrity of the frame buffer logic and external memory.

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 transmitter (GPU), such as a laptop with DisplayPort output
- DisplayPort receiver device, such as a monitor
- Platform cable USB JTAG programmer
- DisplayPort cable and two USB cables

The setup is shown in Figure 5.

Figure 5: **Hardware Setup**

Ensure that all necessary cables and jumpers are set correctly on the board, as shown in Table 20 and Figure 6.

Table 20: **CVK Jumper Settings**

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

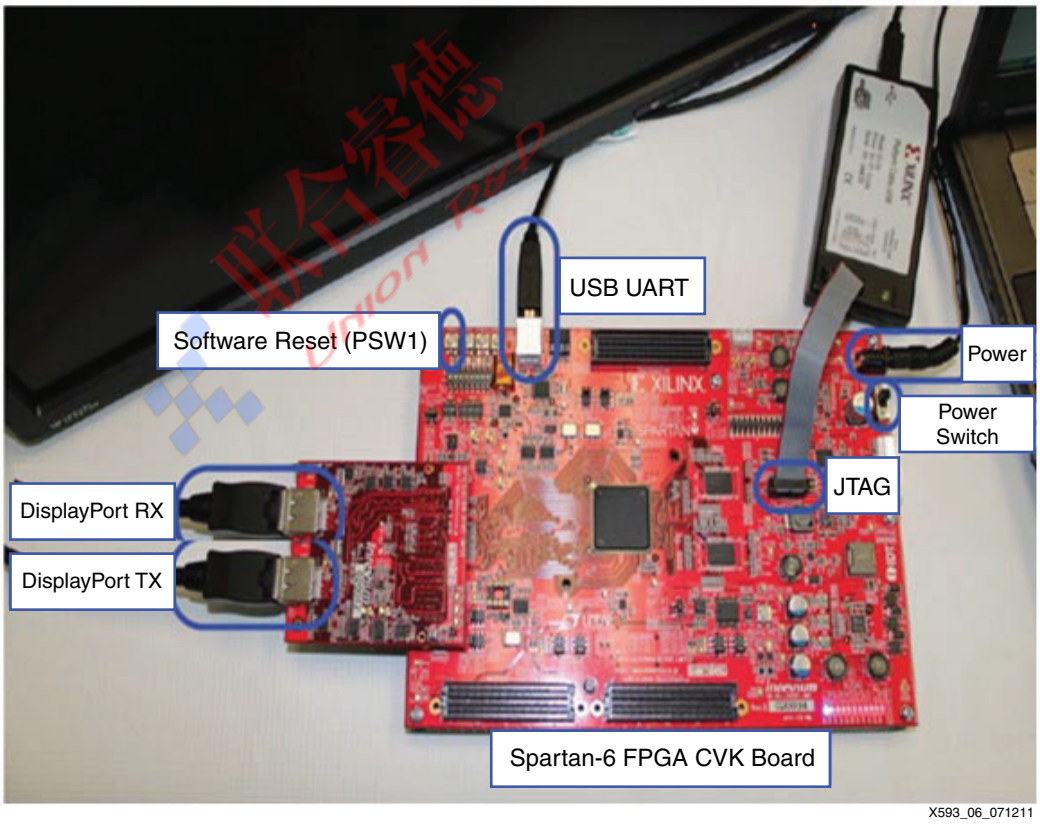


Figure 6: **CVK 1.0 Board Setup**

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 designs. With these files, the user can bring up the systems and ensure connectivity. In the reference design files accompanying this application note, the files are located in `XAPP593/s6/CVK1.0/sdk_workspace/hw_platform_0/download.bit`.

The working systems and the extended displays are shown in Figure 7.

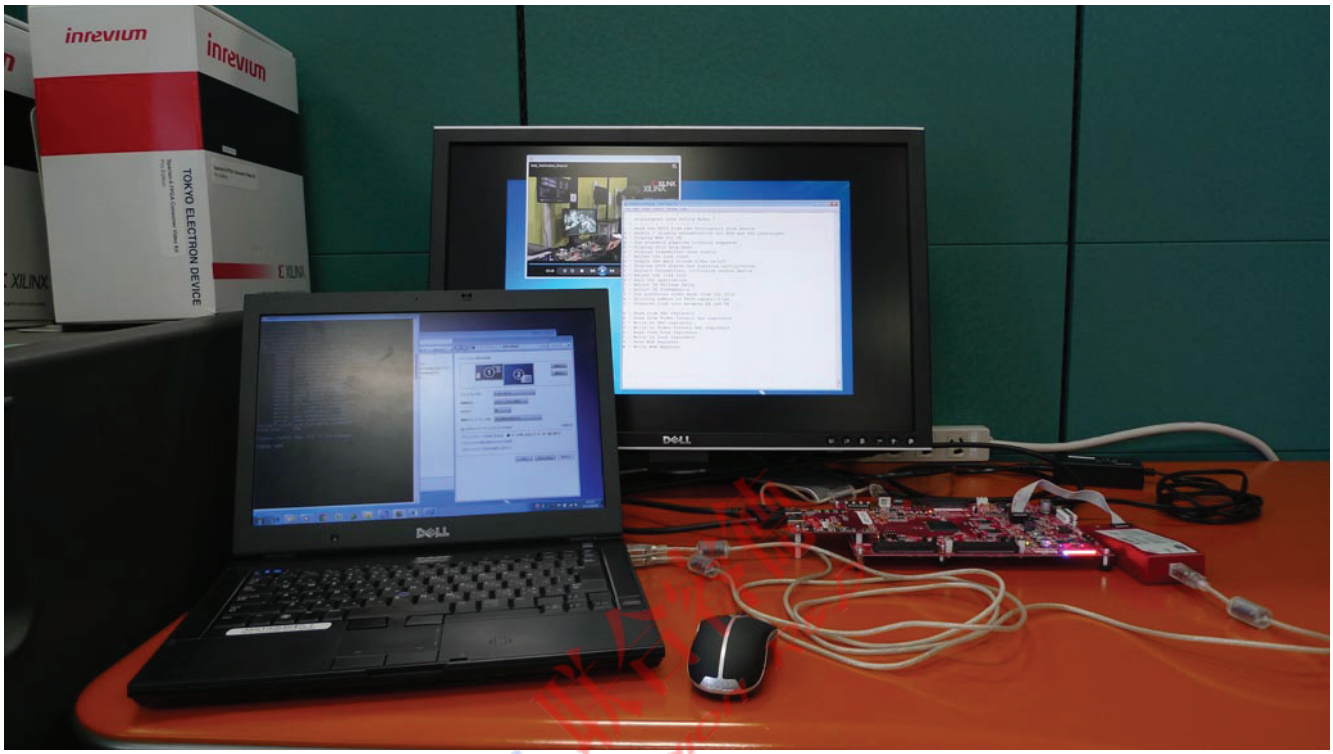


Figure 7: **Working System**

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_sink_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 8 shows the directory hierarchy and the most important files in the directories.

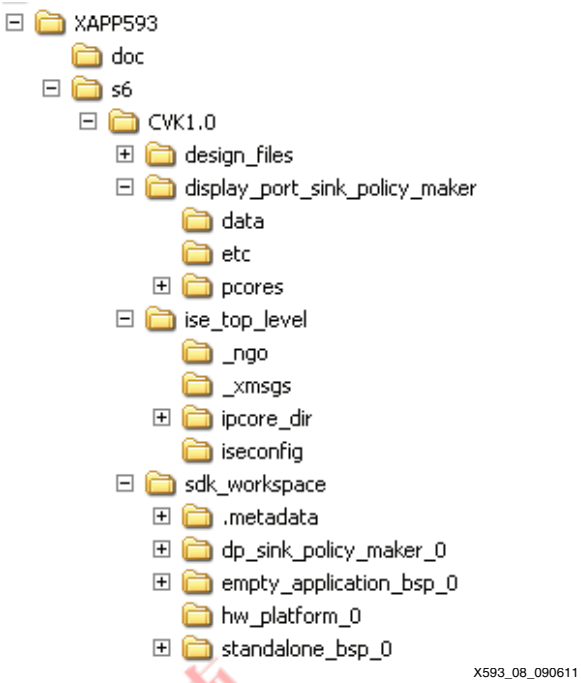


Figure 8: **Directory Structure**

Generating the Design

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

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 9. Remember the location of the XAPP folder, because it is referenced multiple times throughout the use of this application note. After creating the directories, copy the contents of the original `XAPP/design_files` directory to the newly created `design_files` directory.

Figure 9: **Directory Structure Setup**

Step 2: Create the ISE Tools System

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

Set the **Location:** and **Working Directory:** to **`mydirectory/XAPP/ise_top_level`**, where `mydirectory` is the location where the XAPP folder was placed in Step 1: Set up the Directory Structure. Set the **Top-level source type** to **HDL**, as shown in Figure 10.

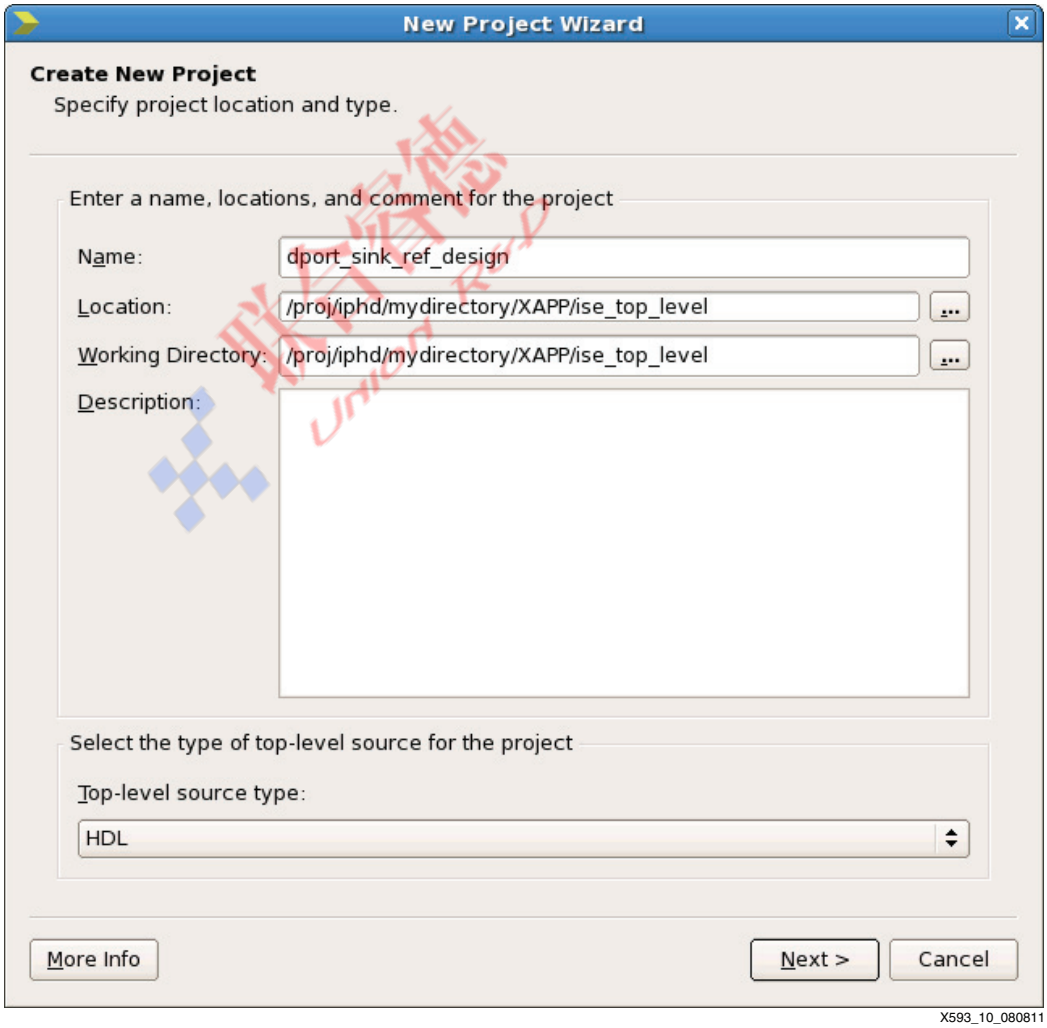


Figure 10: **ISE Tools New Project Wizard**

Set the device and project properties to select the correct device for the CVK board (Figure 11).

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

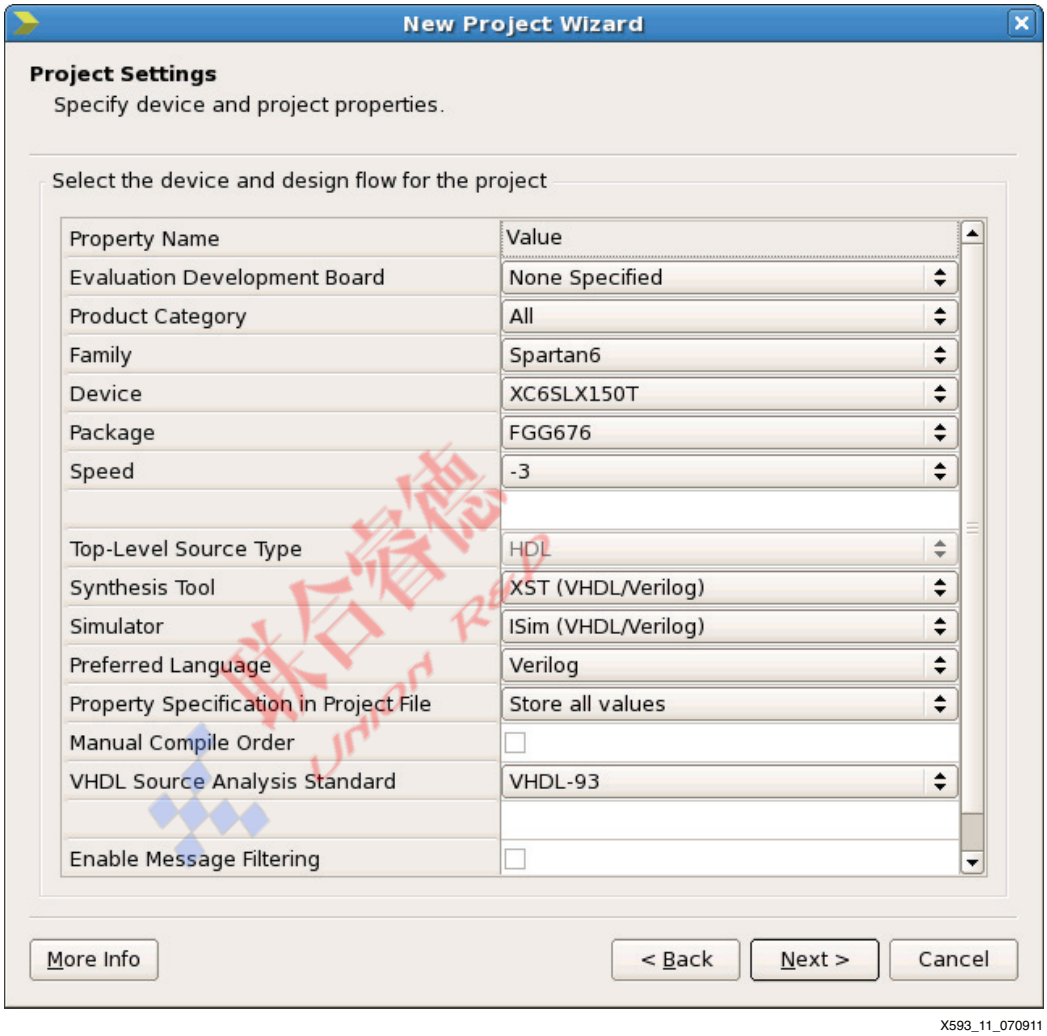


Figure 11: **ISE Tools Project Settings**

Click **Next**, 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 are created using the CORE Generator tool, as described in the next step.

Set up a new project for the CORE Generator tool to point to the `ipcore_dir` in the project, as shown in Figure 12.

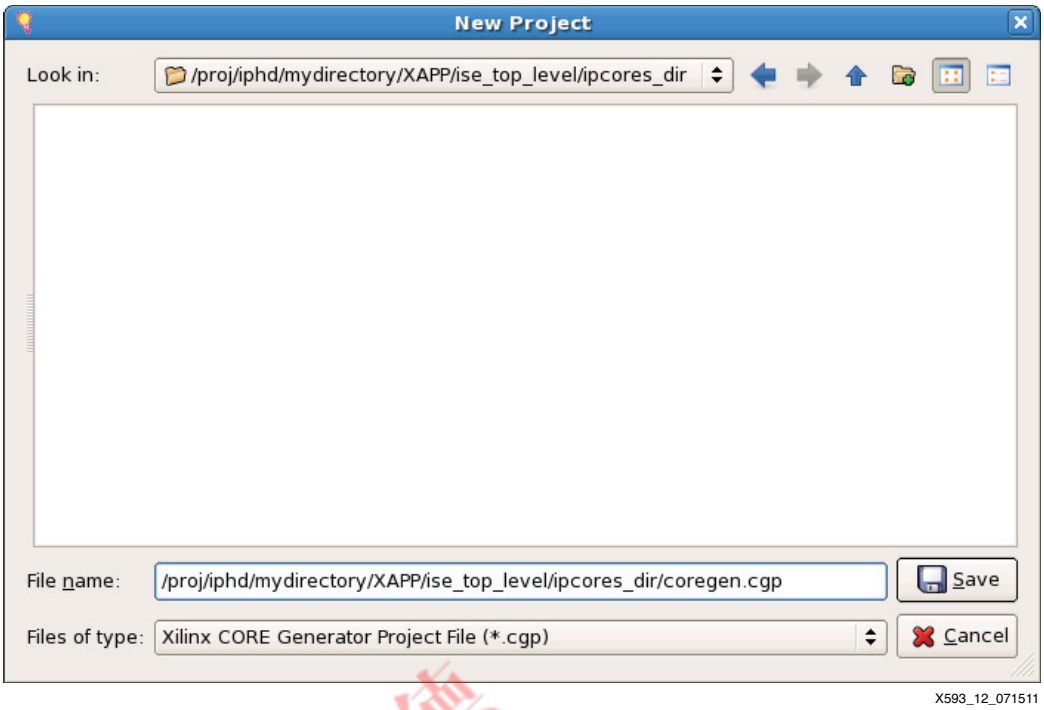


Figure 12: **CORE Generator Tool - New Project Setup**

Set up the project options for the CORE Generator tool—the same project options as the ones for the ISE software—to generate the core (Figure 13).

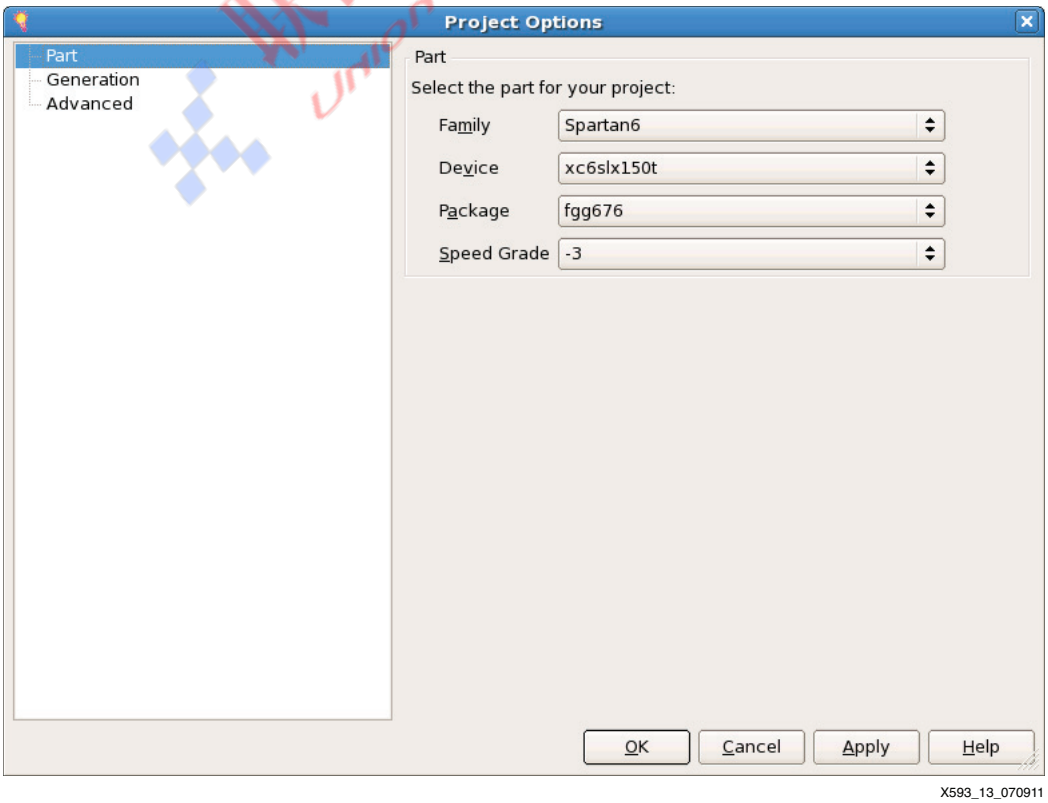


Figure 13: **CORE Generator Tool - New Project Options**

Step 3: Generate and Integrate the Cores

Open the CORE Generator tool from the ISE tools by clicking on **Tools** → **CORE Generator**. When the CORE Generator tool opens, navigate to the DisplayPort version 2.3 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 core (Figure 14):

1. Enter the **Component Name** as **displayport_v2_3.**
2. Select **Transmit Source Core** from the data flow direction radio buttons.
3. Set **Number of Lanes** to **4.**
4. Click **Generate**.

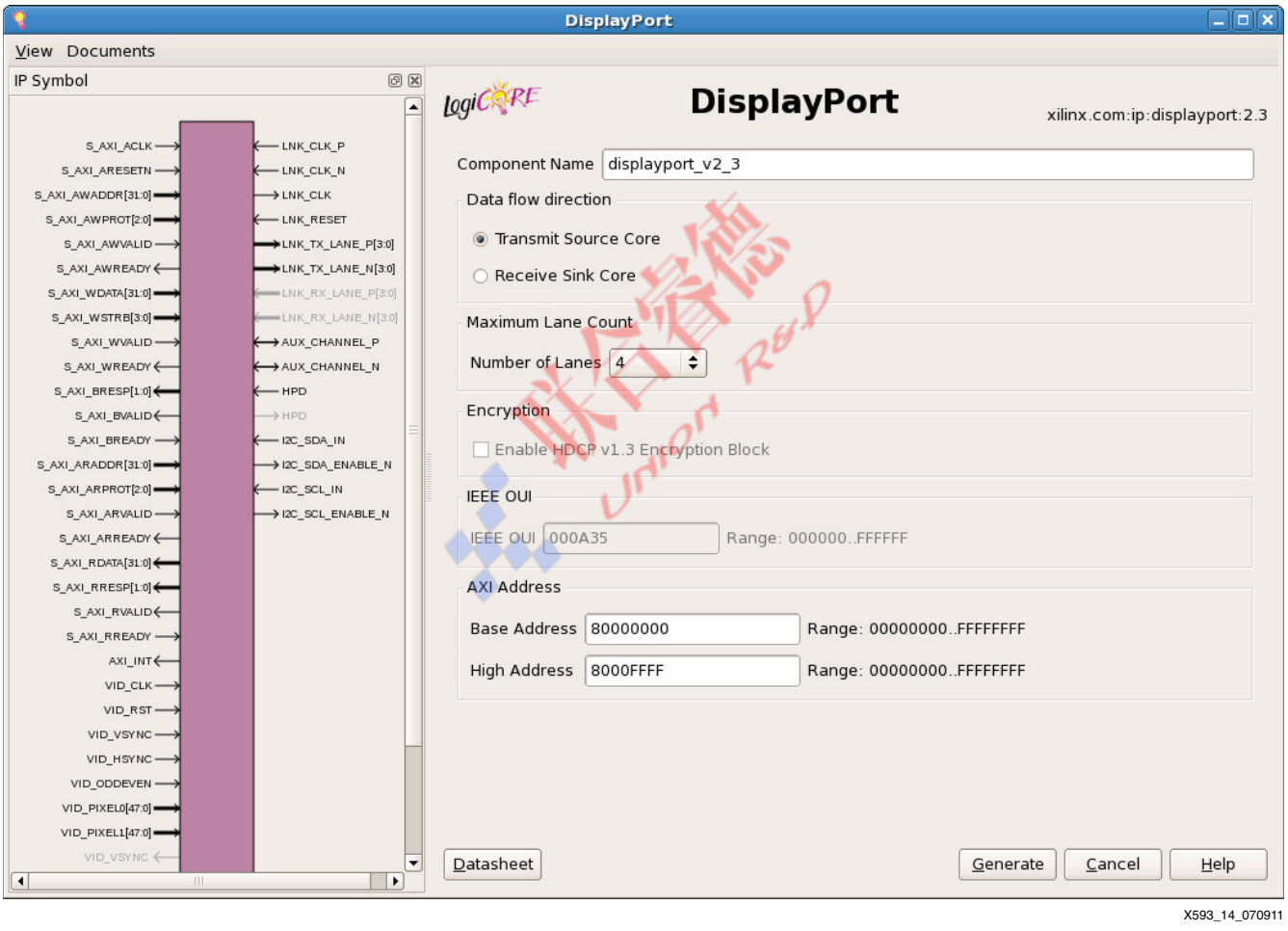


Figure 14: **DisplayPort TX LogiCORE IP Generation**

To generate the RX core (Figure 15):

1. Enter the **Component Name** as **displayport_v2_3_rx**.
2. Select **Receive Sink Core** from the data flow direction radio buttons.
3. Set **Number of Lanes** to **4.**
4. Click **Generate**.
5. After the RX core is generated, close the CORE Generator software.

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.

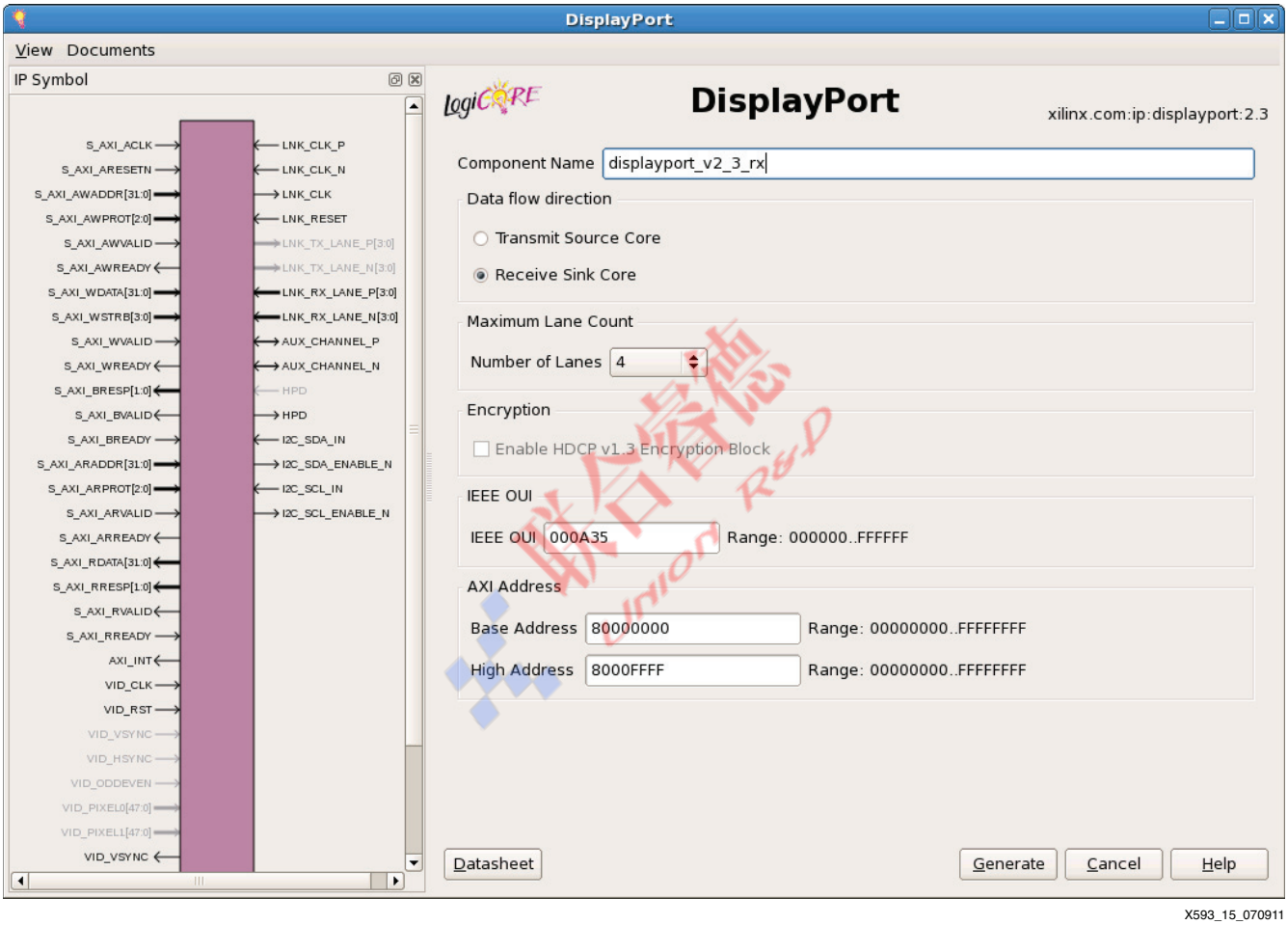


Figure 15: **DisplayPort RX LogiCORE IP Generation**

After the core is generated, it can be integrated into the ISE tools system. The generated core is located in:

```
mydirectory/XAPP/ise_top_level/ipcore_dir/displayport_v2_3
mydirectory/XAPP/ise_top_level/ipcore_dir/displayport_v2_3_rx
```

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_txrx_exdes.v`.

The new example design contains the MicroBlaze processor-based policy maker from this application note. The design files also provide the MIG core. The user can decide to regenerate the MIG core from the CORE Generator tool using the similar flow. The frame buffer logic, CRC engines, and clocking modules are also sourced from the design files.

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

```
mydirectory/XAPP/design_files/clk_wiz_200m.v
mydirectory/XAPP/design_files/dcmspi.v
mydirectory/XAPP/design_files/displayport_txrx_exdes.v
mydirectory/XAPP/design_files/displayport_txrx.ucf
mydirectory/XAPP/design_files/vid_clkgen.v
mydirectory/XAPP/design_files/patgen/regs.v
mydirectory/XAPP/design_files/patgen/timing.v
mydirectory/XAPP/design_files/patgen/video_pat_gen.v
mydirectory/XAPP/design_files/frame_buffer/crc_16_comp.v
mydirectory/XAPP/design_files/frame_buffer/frame_buf_top.v
mydirectory/XAPP/design_files/frame_buffer/rx_video_pack.v
mydirectory/XAPP/design_files/frame_buffer/tx_dfifo.v
mydirectory/XAPP/design_files/frame_buffer/tx_video_unpack.v
mydirectory/XAPP/design_files/frame_buffer/vid_crc_16.v
mydirectory/XAPP/design_files/frame_buffer/video_to_mc_bridge.v
mydirectory/XAPP/design_files/frame_buffer/mig_top_200m/user_design/rtl/infrastructure.v
mydirectory/XAPP/design_files/frame_buffer/mig_top_200m/user_design/rtl/memc_wrapper.v
mydirectory/XAPP/design_files/frame_buffer/mig_top_200m/user_design/rtl/mig_top_200m.v
mydirectory/XAPP/design_files/frame_buffer/mig_top_200m/user_design/rtl/mcb_controller/iodrp_controller.v
mydirectory/XAPP/design_files/frame_buffer/mig_top_200m/user_design/rtl/mcb_controller/iodrp_mcb_controller.v
mydirectory/XAPP/design_files/frame_buffer/mig_top_200m/user_design/rtl/mcb_controller/mcb_raw_wrapper.v
mydirectory/XAPP/design_files/frame_buffer/mig_top_200m/user_design/rtl/mcb_controller/mcb_soft_calibration_top.v
mydirectory/XAPP/design_files/frame_buffer/mig_top_200m/user_design/rtl/mcb_controller/mcb_soft_calibration.v
mydirectory/XAPP/design_files/frame_buffer/mig_top_200m/user_design/rtl/mcb_controller/mcb_ui_top.v
mydirectory/XAPP/design_files/frame_buffer/mem/line_fifo.v
mydirectory/XAPP/design_files/frame_buffer/mem/line_fifo.ngc
mydirectory/XAPP/design_files/frame_buffer/mem/line_fifo_tx.v
mydirectory/XAPP/design_files/frame_buffer/mem/line_fifo_tx.ngc
mydirectory/XAPP/ise_top_level/ipcore_dir/displayport_v2_3/source/dport_txrx_phy.v
mydirectory/XAPP/ise_top_level/ipcore_dir/displayport_v2_3/source/dport_txrx.v
mydirectory/XAPP/ise_top_level/ipcore_dir/displayport_v2_3/source/s6_gt_tile.v
mydirectory/XAPP/ise_top_level/ipcore_dir/displayport_v2_3/source/s6_gt_wrapper.v
mydirectory/XAPP/ise_top_level/ipcore_dir/displayport_v2_3/source/displayport_v2_3_tx.v
mydirectory/XAPP/ise_top_level/ipcore_dir/displayport_v2_3/source/dport_txlink_top.v
mydirectory/XAPP/ise_top_level/ipcore_dir/displayport_v2_3/dport_txlink_top.ngc
mydirectory/XAPP/ise_top_level/ipcore_dir/displayport_v2_3_rx/dport_rxlink_top.ngc
mydirectory/XAPP/ise_top_level/ipcore_dir/displayport_v2_3_rx/source/dport_rxlink_top.v
mydirectory/XAPP/ise_top_level/ipcore_dir/displayport_v2_3_rx/example_design/iic_edid_rom.vhd
mydirectory/XAPP/ise_top_level/ipcore_dir/displayport_v2_3_rx/example_design/iic_rom.vhd
```

The TXRX PHY module (`dport_txrx_phy.v`) requires define files from both TX and RX core. The user can copy these files to `ise_top_level` to resolve define values. The define files are:

```
mydirectory/XAPP/ise_top_level/ipcore_dir/displayport_v2_3/source/dport_tx_defs.v
mydirectory/XAPP/ise_top_level/ipcore_dir/displayport_v2_3_rx/source/dport_rx_defs.v
mydirectory/XAPP/ise_top_level/ipcore_dir/displayport_v2_3_rx/source/dport_rx_dpcd_defs.v
```

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.

Note: The sink device EDID structure advertises 1600x1200 as the maximum supported resolution.

The EDID structure present in `mydirectory/XAPP/ise_top_level/ipcore_dir/displayport_v2_3_rx/example_design/iic_edid_rom.vhd` can be modified to change the maximum supported resolution. For more details on the EDID data structure, refer to the VESA Enhanced EDID Standard [Ref 7].

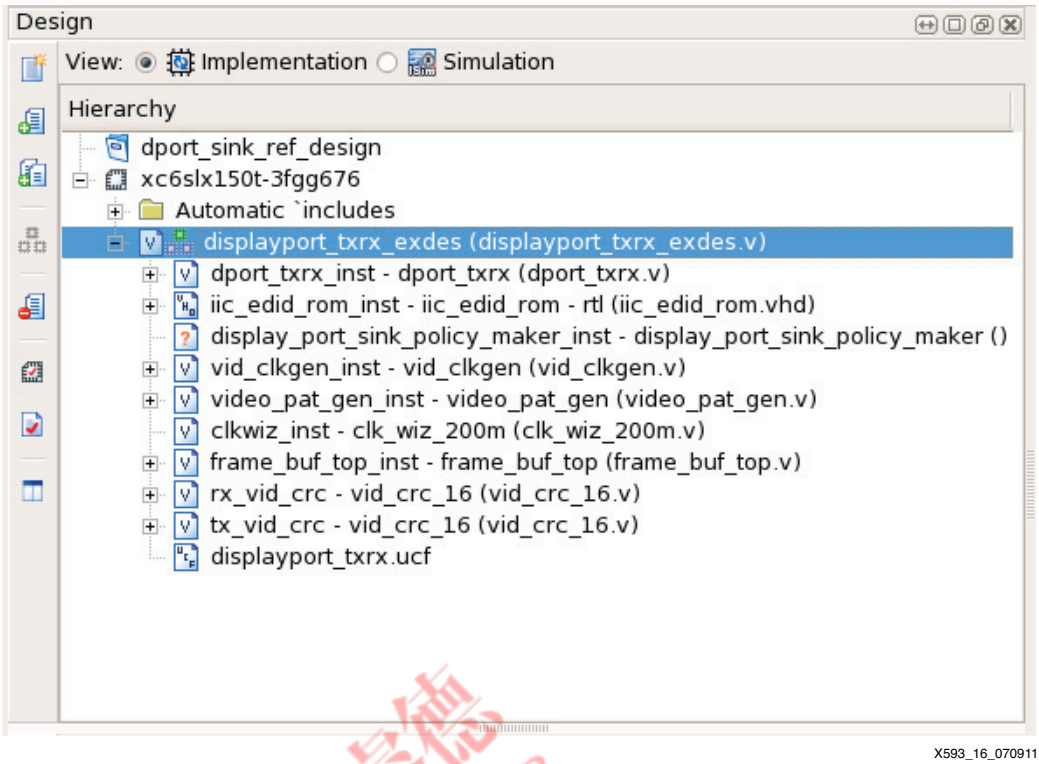


Figure 16: **Partial ISE Tools System**

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 software and select **New Source.** Select **Embedded Processor** as the source type. Set the File name to **display_port_sink_policy_maker** and the Location to **mydirectory/XAPP.** Ensure the **Add to project** box is checked, where *mydirectory* is the directory where the XAPP folder was placed in Step 1: Set up the Directory Structure. Select **Next**, then **Finish**. This launches Xilinx Platform Studio (XPS). If prompted to create a system using Base System Builder, select **No** (Figure 17).

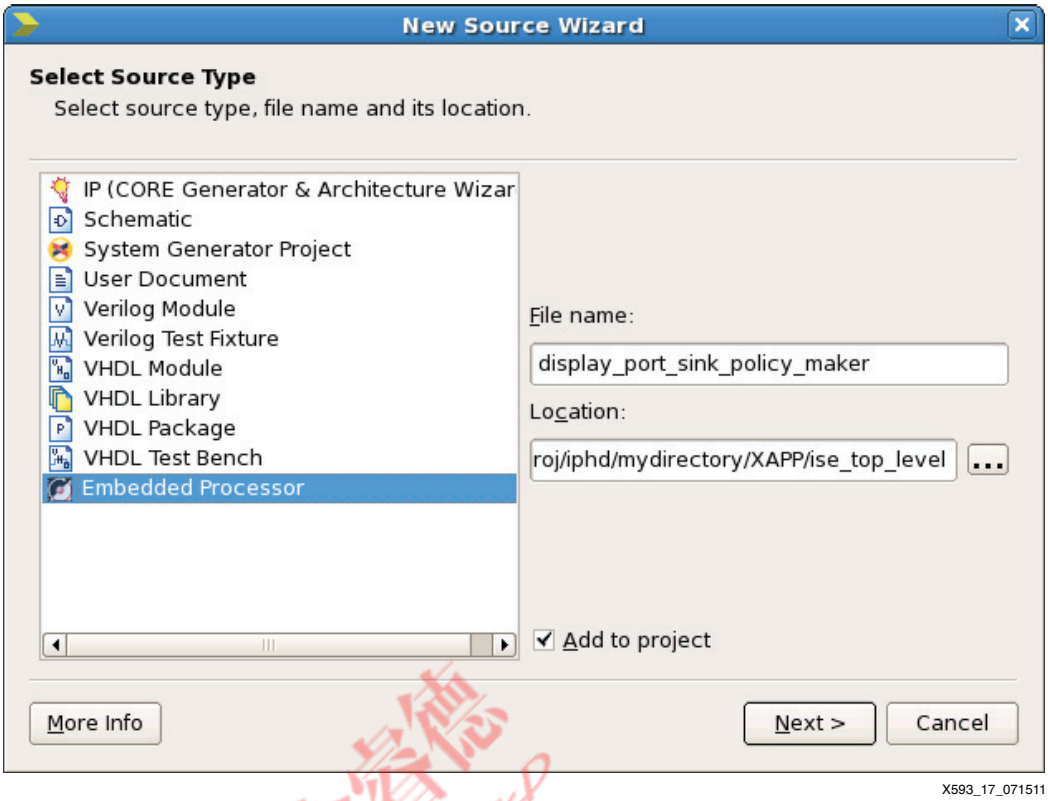


Figure 17: **ISE Tools New Source Wizard for EDK Insertion**

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

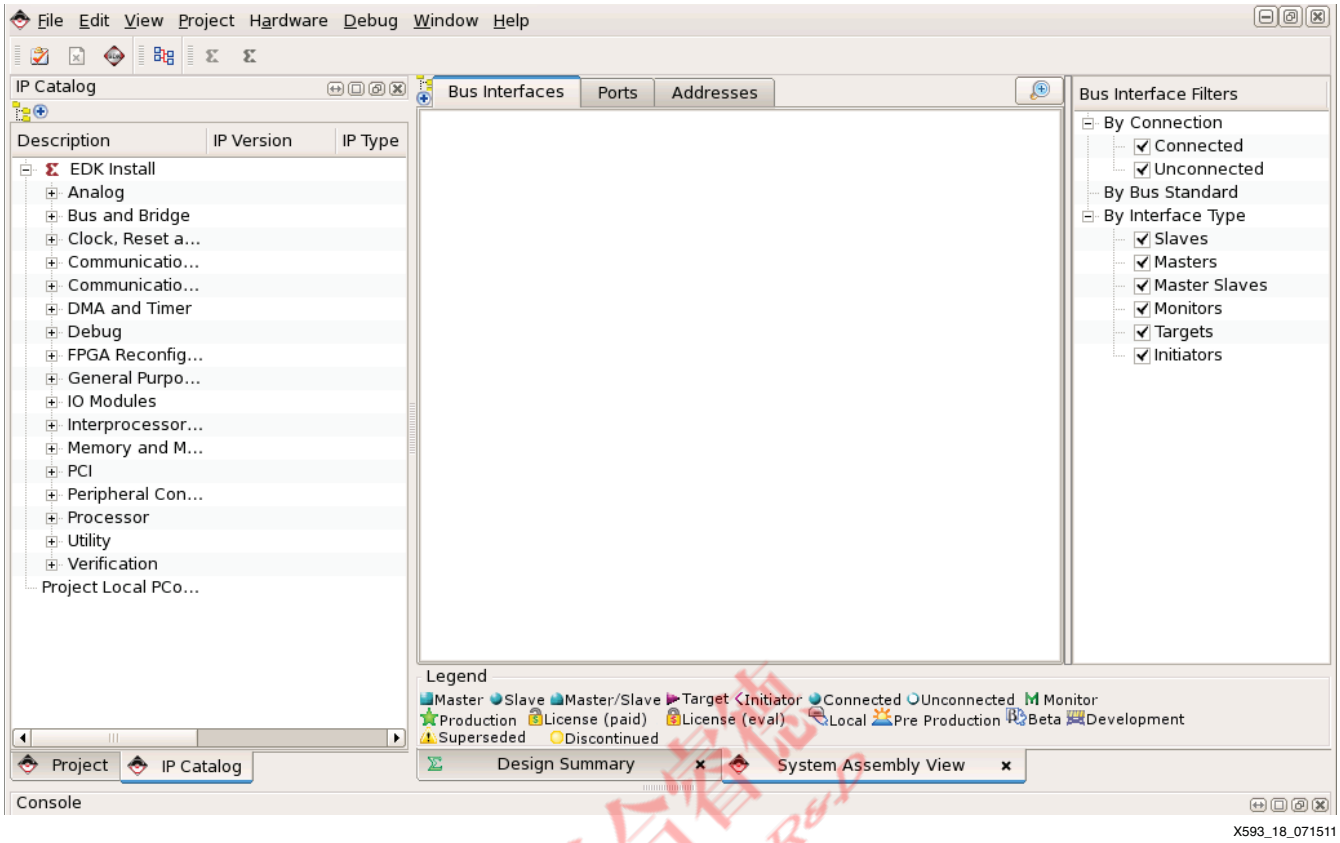


Figure 18: **EDK IP Catalog Selection**

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_sink_policy_maker/pcores/` to the current design `mydirectory/XAPP/display_port_sink_policy_maker/pcores`.

After the directory is 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 to see when the IP is added. Next, add cores listed in Table 21 to the system by locating them in the IP Catalog and double-clicking them. In the System Assembly View, click the IP name and rename each one to the names given in Table 21.

Table 21: **EDK Required IP**

IP Location	Version	Name
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`
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`

Table 21: **EDK Required IP *(Cont'd)***

IP Location	Version	Name
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
Project Local pcores → USER → AXI EXTERNAL SLAVE CONNECTOR	1.00.a	axi_ext_slave_conn_1

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-downs 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 the EDK 13.2 Documentation [Ref 5].

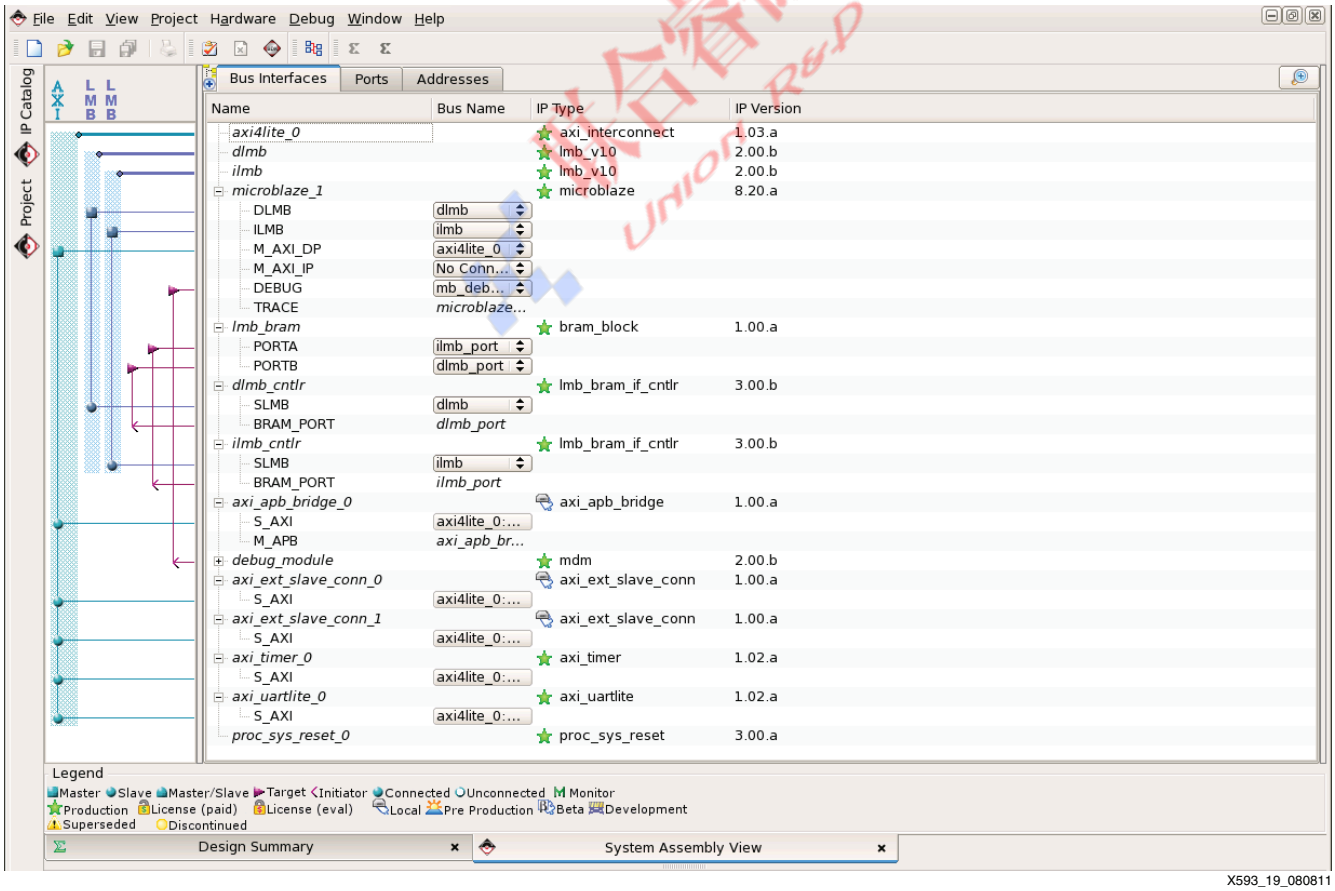


Figure 19: **EDK Bus Connectivity**

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 pre-configured or automatically configured. However, the MicroBlaze processor and the UART Lite IPs need to be configured for this system.

MicroBlaze Processor 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 on the **axi_uartlite_0** IP, select the **User** tab, and set the **UART Lite Baud Rate** to **115200.** Set **Use Parity** to **FALSE,** and click **OK** as shown in Figure 21.

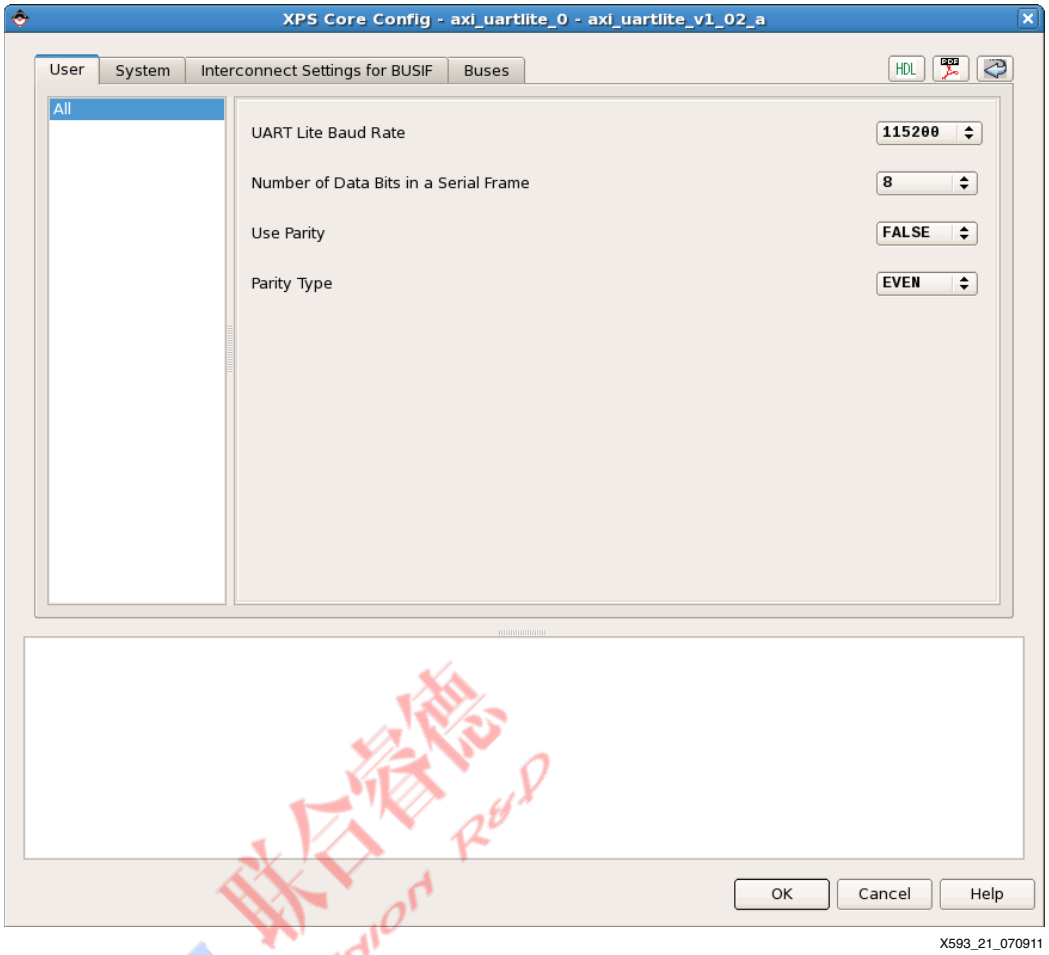


Figure 21: **RS-232 UART Configuration**

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.

Double-click the **axi_ext_slave_conn1** IP and select **AXILITE** as the C_S_AXI_PROTOCOL. Configure C_S_AXI_RNG00_BASEADDR and C_S_AXI_RNG00_HIGHADDR to **0xC4A00000** and **0xC4A0FFFF**, 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 and Figure 23 on the line following the PARAMETER VERSION. Save the file, return to the System Assembly View, and click on the **Ports** tab.

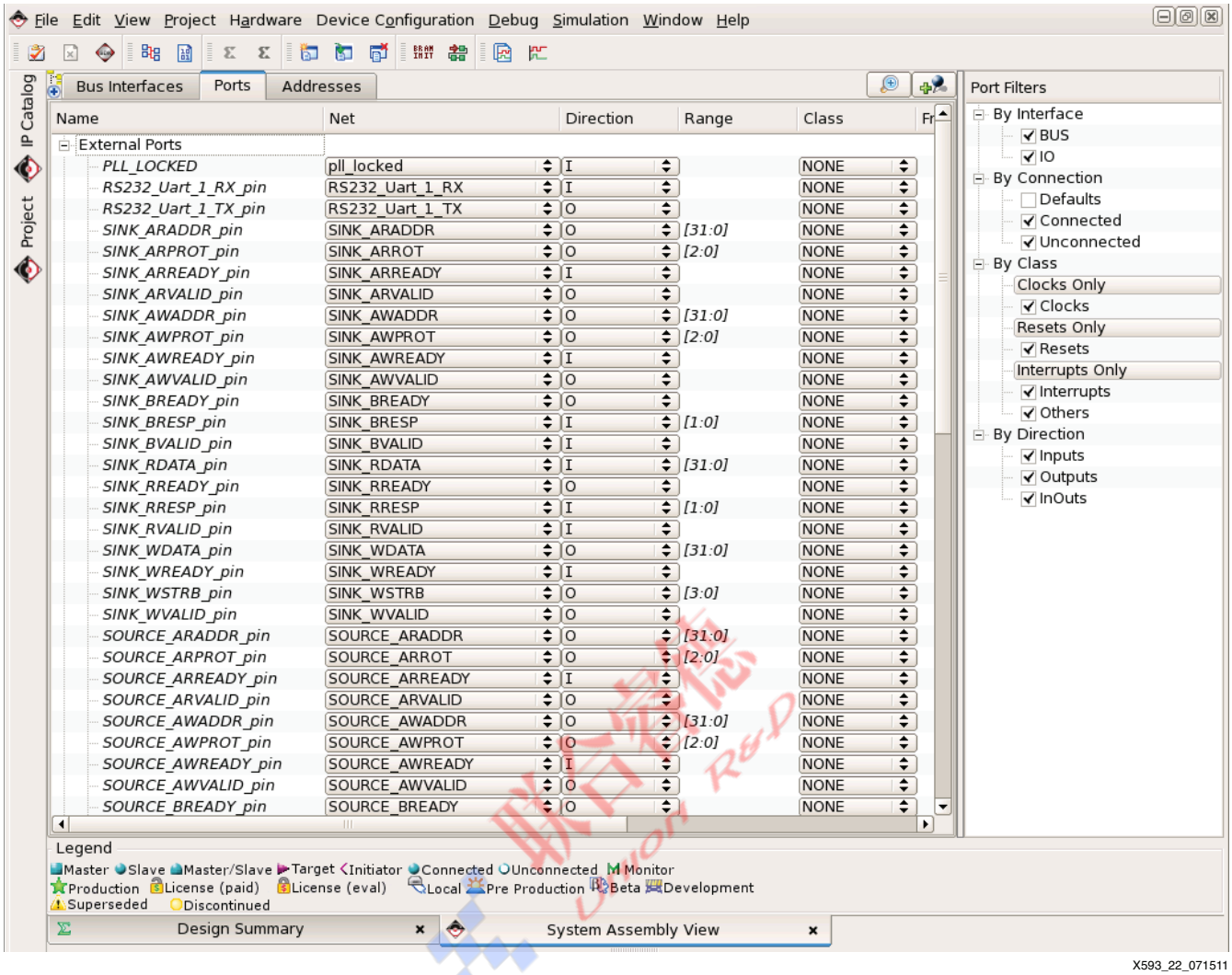


Figure 22: **EDK External Port Connections**

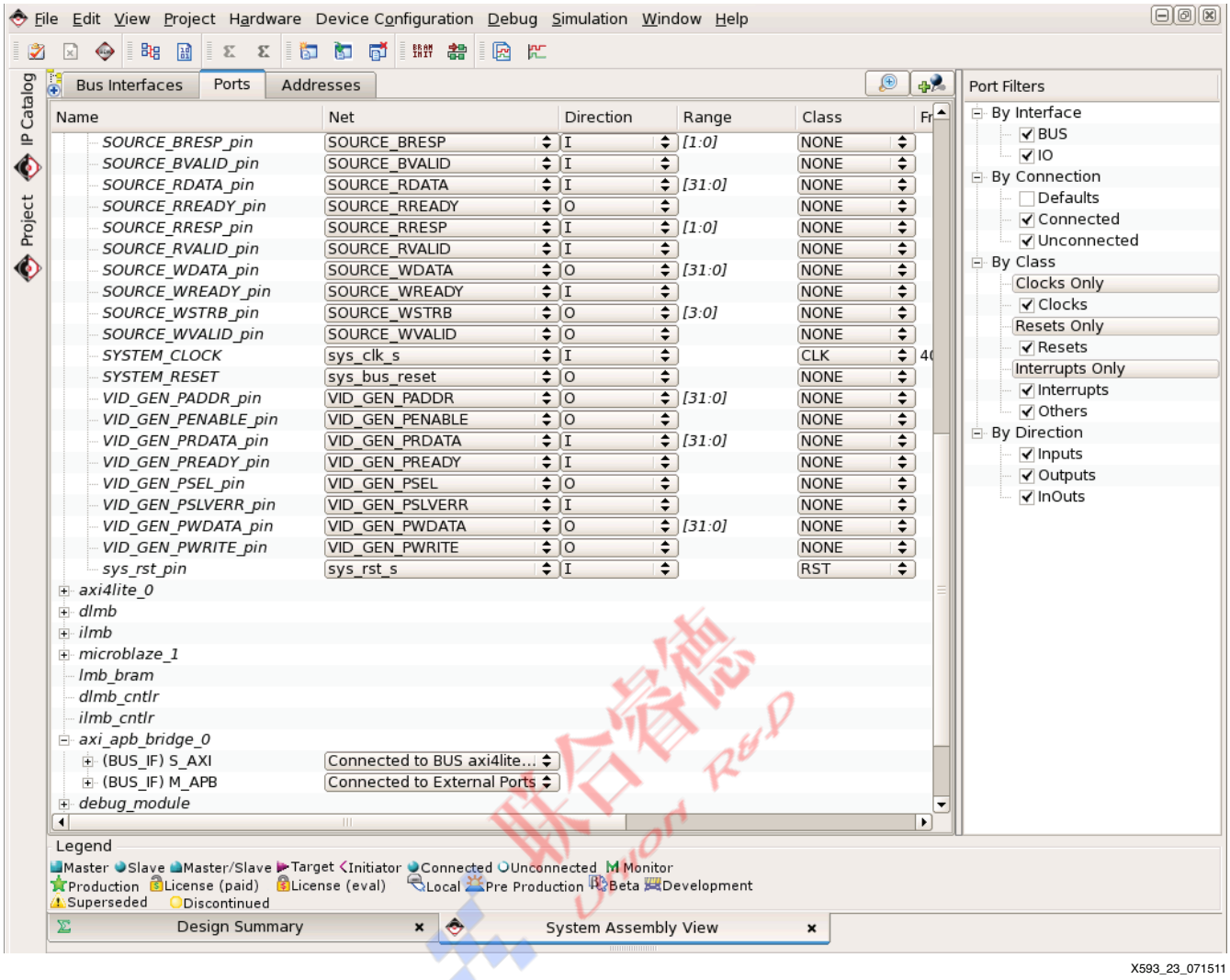


Figure 23: **EDK External Port Connections (Continued)**

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 need to be changed to reflect that.

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

XPS can now be closed, and work can resume in the ISE software. When returning to the ISE software, the `display_port_sink_policy_maker_inst` should be populated with the EDK project as shown in Figure 24.

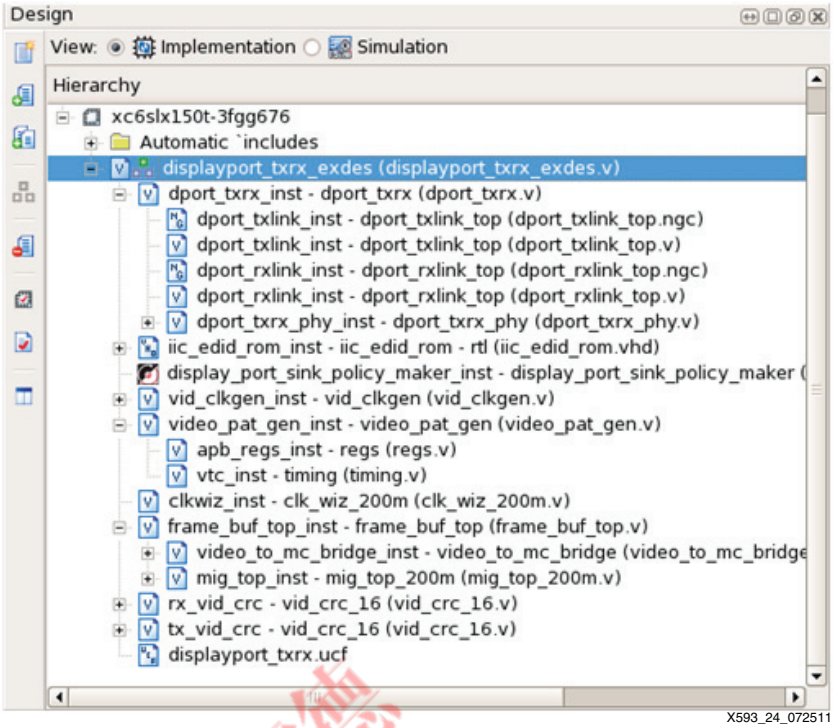
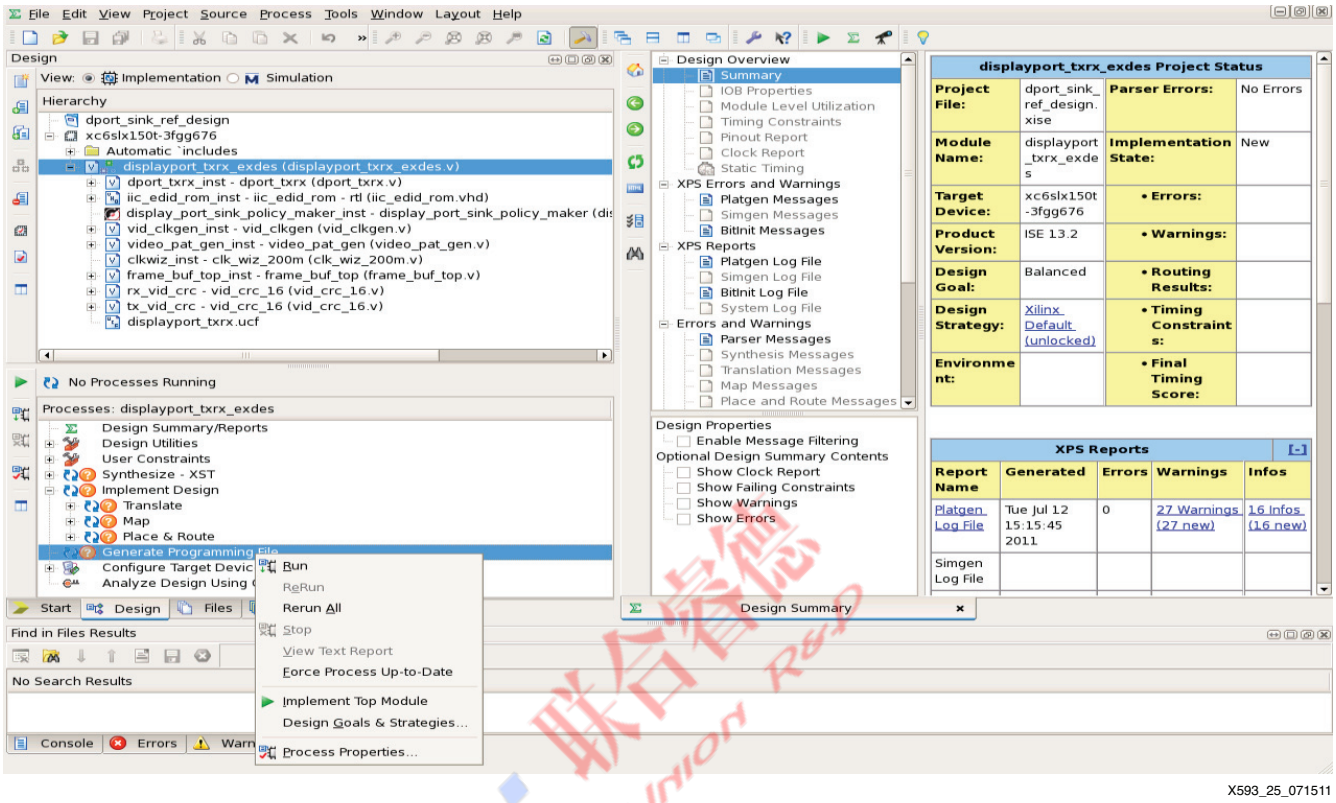


Figure 24: **Complete ISE Tools System**

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 software as part of the ADD file (Figure 25).



X593_25_071511

Figure 25: **Implementation Properties**

The design is now ready to be built. Double-click on **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 the 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_sink_policy_maker_inst** and double click **Export Hardware Design to SDK with Bitstream** from the bottom-left pane, as shown in Figure 26.

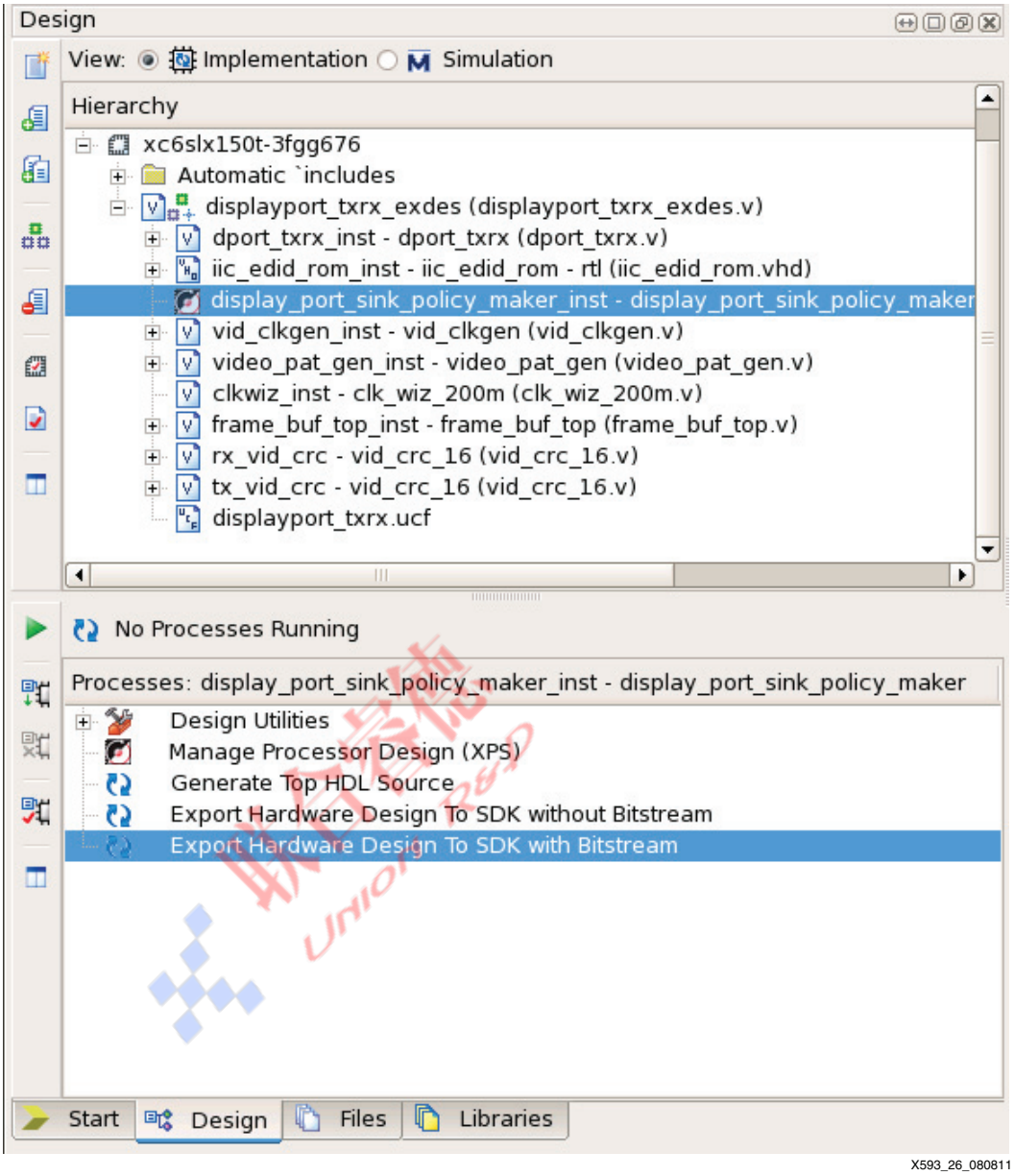


Figure 26: **Export Hardware Design to SDK**

Double-click **Export Hardware Design to SDK.** This creates an XML file in `mydirectory/display_port_sink_policy_maker/SDK/SDK_Export/hw/` named `display_port_sink_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_sink_policy_maker/SDK/SDK_Export/hw/display_port_sink_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 Figure 27.
7. Click **OK**.

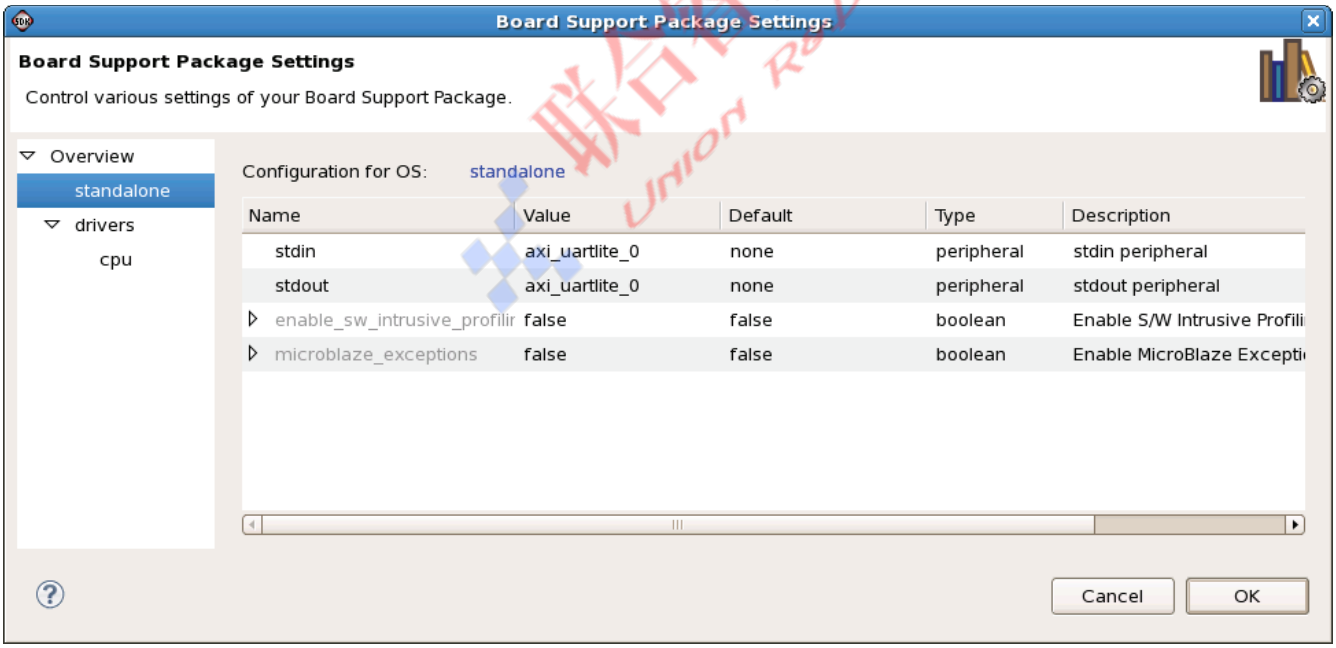


Figure 27: **Board Support Package Settings**

Xilinx C Project

To create the C Project:

1. Click **File** → **New** → **Xilinx C Project**.
2. Set the project name to **dp_sink_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**.

Add the Code

SDK should now have three projects in the project explorer, as shown in Figure 28.

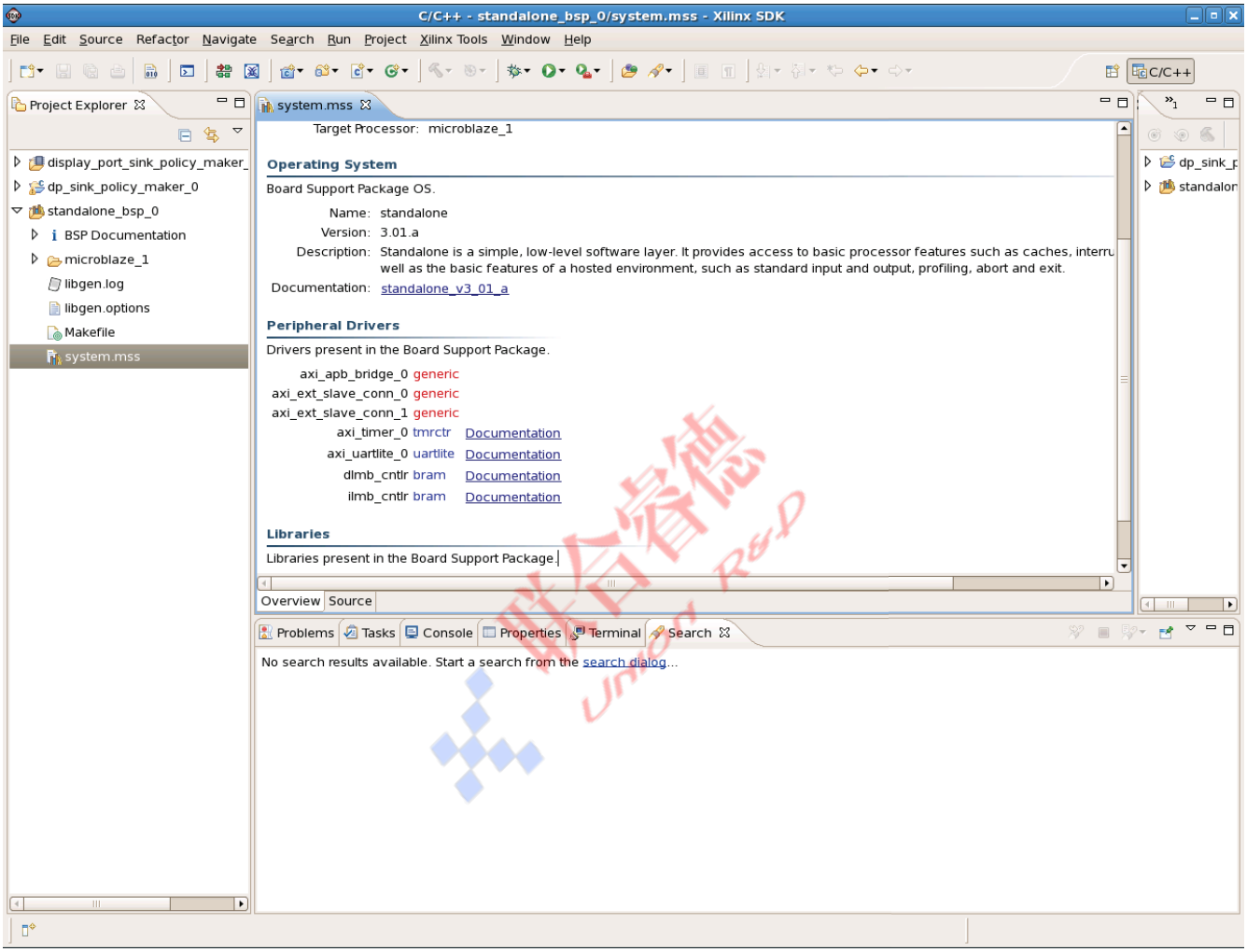


Figure 28: **SDK Project Explorer**

Using a console or folder browser, copy the source files (`*.c` and `*.h`) from the reference design folder `XAPP/sdk_workspace/dp_sink_policy_maker_0/src` to `mydirectory/XAPP/sdk_workspace/dp_sink_policy_maker_0/src`.

Go back to SDK and refresh the `dp_sink_policy_maker_0/src` folder by clicking on it in the Project Explorer window and pressing **F5**. The source should automatically compile and place `dp_sink_policy_maker_0.elf` in the `mydirectory/XAPP/sdk_workspace/dp_sink_policy_maker_0/Debug` directory.

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_sink_policy_maker_0** project, and select **C/C++ Build Settings**. Next, click the **Build Steps** tab and, as shown in Figure 29, set the **Command:** in the Post-build steps to:

```
data2mem -bm ../../hw_platform_0/system.bmm -bt
../../hw_platform_0/system.bit -bd dp_sink_policy_maker_0.elf -o b ../../
hw_platform_0/download.bit
```

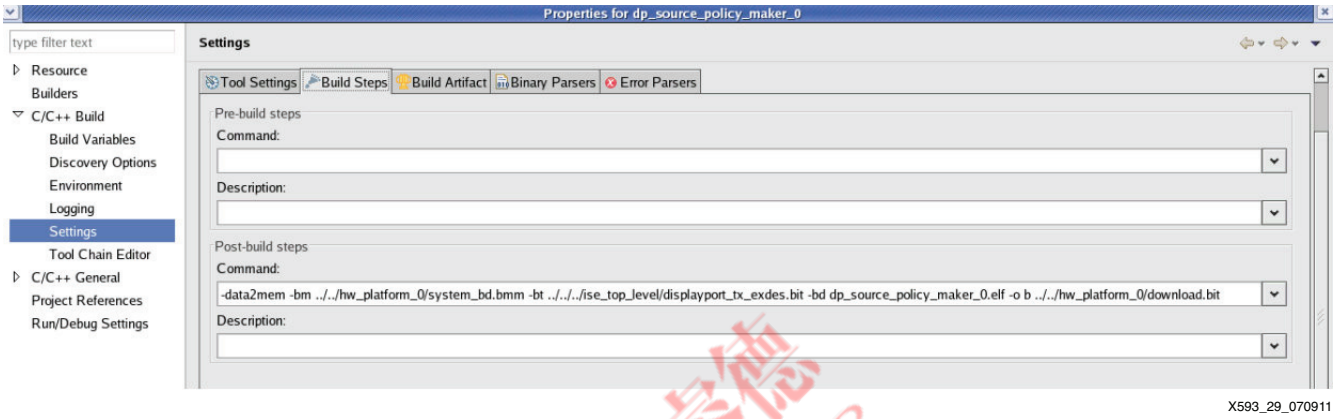


Figure 29: **SDK Post-Build Steps**

These steps create a new `download.bit` file in the `mydirectory/XAPP/sdk_workspace/hw_platform_0` directory every time the software is rebuilt. The download.bit file can now be downloaded to the Spartan-6 FPGA on the CVK board. Refer to Setup and Usage, page 19 for more information about using the reference design.

Reference CRC Values

Table 22 and Table 23 list frame reference CRC values for single pixel and dual pixel modes, respectively.

Table 22: **Single Pixel Mode Reference CRC Values**

Resolution	BPC	CRC-Red	CRC-Green	CRC-Blue
640 x 480	6	0x5792	0xCB6D	0x4CD
640 x 480	8	0xFD7	0x83E	0x7C2A
640 x 480	10	0x1DA0	0x8BAC	0xF83D
1024 x 768	6	0xD2BE	0xC858	0x767C
1024 x 768	8	0xAFF5	0x68A4	0x0734
1024 x 768	10	0xD9DF	0x7BB7	0x8C67
1280 x 768	6	0xC918	0x99F0	0x3108
1280 x 768	8	0xc7AE	0x1532	0x29BC
1280 x 768	10	0x5BCB	0xa9ED	0x8F5E
1280 x 1024	6	0xA16D	0xC633	0x7E77
1280 x 1024	8	0xD0A7	0x73B0	0x0699

Table 22: **Single Pixel Mode Reference CRC Values *(Cont'd)***

Resolution	BPC	CRC-Red	CRC-Green	CRC-Blue
1280 x 1024	10	0xA6FF	0x314C	0x5E19
1600 x 1200	6	0xD71D	0xBFAD	0x186F
1600 x 1200	8	0x1D5F	0xC4E5	0xEC7C
1600 x 1200	10	0xA090	0xEE03	0xDDB4

Table 23: **Dual Pixel Mode Reference CRC Values**

Resolution	BPC	CRC-Red 0	CRC-Green 0	CRC-Blue 0	CRC-Red1	CRC-Green1	CRC-Blue 1
640 x 480	6	0x83E3	0x420A	0xD6D9	0xBD9A	0xC852	0xDF4F
640 x 480	8	0x432F	0xC2BF	0x2C9D	0x2CB0	0xE029	0xEEFA
640 x 480	10	0x667D	0x7453	0x3716	0xF425	0x7BCC	0x2F0E
1024 x 768	6	0x49D5	0xFFD7	0xC928	0xC006	0x1788	0xB136
1024 x 768	8	0x6BA2	0xDF31	0x290B	0xE9D5	0x4525	0xF70E
1024 x 768	10	0xBF1a	0xBFD0	0x3FF0	0x71C9	0x16E5	0x0797
1280 x 768	6	0x185C	0xEAC2	0xF04E	0x7676	0xFD9D	0x2255
1280 x 768	8	0x5F4B	0xD585	0x0E4A	0x84C3	0x7051	0x9ACF
1280 x 768	10	0x5ABB	0xD9C1	0x1235	0x42A9	0x3E86	0xB3DA
1280 x 1024	6	0x08F4	0xA9B2	0x52B6	0x673B	0xBAB6	0x9CED
1280 x 1024	8	0xE190	0x78CC	0xCB76	0x5A60	0x7C0D	0x58E3
1280 x 1024	10	0xAD68	0x7BC3	0x25AE	0xD124	0x824E	0x0BC4
1600 x 1200	6	0x060F	0x8513	0x3B62	0x24C0	0x528F	0x35C6
1600 x 1200	8	0x9330	0xDAF8	0x504D	0x3B80	0xEF1F	0x53E4
1600 x 1200	10	0x3EEC	0xBE96	0xEF70	0x0C41	0xA65C	0xDFEF

Conclusion

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

Reference Design

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

https://secure.xilinx.com/webreg/clickthrough.do?cid=174274

The checklist in Table 24 indicates the tool flow and verification procedures used for the reference design.

Table 24: **Reference Design Matrix**

Parameter	Description
General	
Developer Name	Xilinx
Target Devices (Stepping Level, ES, Production, Speed Grades)	Spartan-6 FPGA (XC6SLX150T-FGG676-3)

Table 24: **Reference Design Matrix *(Cont'd)***

Parameter	Description
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
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

Reference Design Footprint

Table 25 lists 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. Implementations with fewer features are possible with less block RAM utilization.

Table 25: **Reference Design Footprint**

IP	LUTs	Flip-Flops	Block RAMs
axi_apb_bridge_0_wrapper	144	141	0
axi_timer_0_wrapper	217	271	0
axi_uartlite_0_wrapper	85	103	0
axi4lite_0_wrapper	500	602	0
proc_sys_reset_0_wrapper	69	55	0
lmb_bram_wrapper	0	0	32
ilmb_cntlr_wrapper	2	6	0
dlmb_cntlr_wrapper	2	6	0
dlmb_wrapper	1	1	0
ilmb_wrapper	1	1	0
debug_module_wrapper	89	80	0

Table 25: **Reference Design Footprint *(Cont'd)***

IP	LUTs	Flip-Flops	Block RAMs
`microblaze_1_wrapper`	1,239	1,167	0
`display_port_sink_policy_maker`	2,349	2,433	32
Reference design including DP core, frame buffer, MIG core, and associated logic	21,385	13,536	85

References

1. *TED Spartan-6 FPGA Consumer Video Kit 2.0*
   http://www.xilinx.com/products/devkits/TB-6S-CVK.htm
2. AMBA Protocol Specifications Document Set
   http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.set.amba/index.html
3. UG388, *Spartan-6 FPGA Memory Controller User Guide*
4. VESA DisplayPort Standard v1.1a
   http://www.vesa.org
5. EDK 13.2 Documentation
   http://www.xilinx.com/support/documentation/dt_edk_edk13-2.htm
6. UG767, *LogiCORE IP DispayPort v2.3 User Guide*
7. VESA Enhanced EDID Standard (Release A, Revision 2 - September 25, 2006)
   http://www.vesa.org

Revision History

The following table shows the revision history for this document.

Date	Version	Description of Revisions
09/16/11	1.0	Initial Xilinx release.

Notice of Disclaimer