Zynq-7000 EPP ZC702 Base Targeted Reference Design

User Guide

UG925 (v1.0) June 21, 2012

XILINX®

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
06/21/12	1.0	Initial Xilinx release.

Table of Contents

Chapter 1

Introduction

This user guide describes the Base Targeted Reference Design (TRD) based on Zynq™-7000 Extensible Processing Platform (EPP) architecture. The TRD is included with the ZC702 Evaluation kit. To learn more about the ZC702 Evaluation kit and how to evaluate different demonstrations based on Zynq-7000 EPP architecture, refer to UG926, *Zynq-7000 EPP ZC702 Evaluation Kit Getting Started Guide* [Ref 1].

This document explains the functional architecture of the hardware and software components of the TRD. Also provided are register descriptions for IP/logic implemented in programmable logic (PL), Base TRD package directory structure, and pointers to enable the user to further develop embedded platforms based on Zynq-7000 EPP architecture.

The Base Targeted Reference Design

The Base TRD showcases various features and capabilities of the Zynq Z-7020 EPP device for the embedded domain in a single package using a Xilinx standard Zynq Linux-based video pipeline design.

The Base TRD consists of two processing elements: The Zynq-7000 EPP processing system (PS) and a video accelerator based on PL. The EPP allows the user to implement a specific functionality either as a software program running on the Zynq-7000 EPP PS or as a hardware design inside the PL. The Base TRD demonstrates an optimization of how the user can seamlessly switch between a software or a hardware implementation, contributing to ease of use. The TRD also demonstrates the value of offloading computation-intensive tasks onto PL, thereby freeing the CPU resources available for user-specific applications.

Software developers can leverage the Base TRD and start programming right away using the widely known Eclipse-based integrated development environment (IDE), GNU compiler tool chain, Linux operating system (OS), and libraries. Embedded hardware designers now have immediate access to the industry standard ARM® Cortex™-A9 core processor system and video IPs running on PL out of the box. In addition, the TRD also provides customers with access to the various PL-based video components VDMA, TPG, VTC, Sobel filter, and the Xylon logiCVC-ML display controller [Ref 28]. These IPs are part of the video pipeline implemented in the PL.

The Base TRD consists of a PS based on the embedded ARM Cortex-A9 core processor, a video processing pipeline implemented in PL, and a Linux-based software application that includes a Qt-based graphical user interface (GUI) [Ref 29] to provide user control and monitoring. The Linux-based software platform and software application run on the ARM Cortex-A9 cores. The software application works in tandem with hardware and provides the user the choice of offloading computation-intensive processing to the PL-based hardware subsystem.

The Zynq-7000 EPP Base TRD reference design (Figure 1-1) consists of these components:

- A dual ARM Cortex-A9 core processor-based embedded Linux OS, board support package (BSP), and U-Boot boot loader
- PL-based hardware IPs that enable acceleration of computation-intensive video processing tasks
- Linux-based application software components to configure and control the PL-implemented hardware processing IPs and the data flow between PL-based video components and the PS

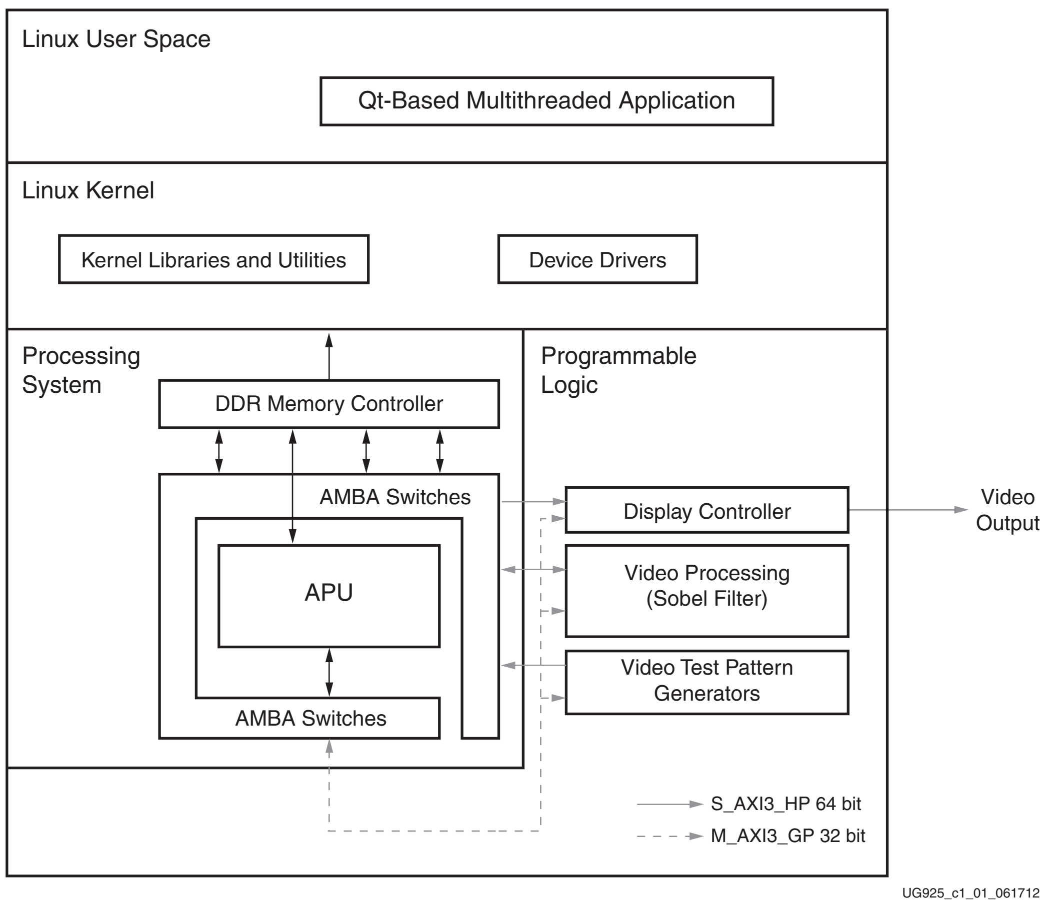


Figure 1-1: **Zynq-7000 EPP Base TRD System Block Diagram**

The TRD deliverables include source code for RTL design and software packages such as the Linux OS, device drivers, the application, and the GUI.

Base TRD Key Features

Components in the Base TRD are further described in this section.

The Base TRD processing system (PS) includes:

- A dual ARM Cortex-A9 core
- ARM AMBA® AXI interconnect
- Multi-protocol, 32-bit DDR DRAM controller
- 1 GB DDR3 running at 533 MHz
- Standard peripheral interfaces including USB, Ethernet, UART, I2C, SD MMC, and GPIO

Base TRD programmable logic includes:

- Two AXI interconnects, 64-bit wide at 150 MHz
- One AXI interconnect, 32-bit wide at 75 MHz
- AXI VDMA(s)
- A full HD video input and output interface
- A Sobel accelerator
- Two AXI Performance Monitors

Base TRD software includes:

- Xilinx Zynq-7000 standard Linux kernel (based on Open Source Linux 3.x)
- Linux device drivers for TRD-specific IPs
- Qt-based Linux application demonstrating the video processing pipeline

List of Acronyms

Table 1-1 lists acronyms used in this document.

Table 1-1: **Acronyms**

Acronym	Definition
AFI	AXI FIFO interface
APU	Application processor unit
BSP	Board support package
DTB	Device tree binary
DTS	Device tree source

Table 1-1: **Acronyms *(Cont'd)***

Acronym	Definition
EDK	Embedded Development Kit
EPP	Extensible processing platform
FMC	FPGA mezzanine card
FPS	Frames per second
FSBL	First-stage boot loader
GIC	General interrupt controller
GUI	Graphical user interface
HD	High definition
IDE	Integrated development environment
IOP	Input/output peripherals
IP	Intellectual property
KFLOPS	kilo floating-point operations per second
OCM	On-chip memory
OS	Operating system
PL	Programmable logic (inside the Zynq-7000 EPP)
PS	Processing system
RTL	Register transfer level
SD	Secure Digital
SD MMC	Secure Digital Multimedia Card
SDK	Software Development Kit
TDP	Targeted Design Platform
TPG	(Video) Test Pattern Generator
TRD	Targeted Reference Design
TTC	Triple-timer counter
VDMA	Video direct memory access
VTC	Video timing controller
XPS	Xilinx Platform Studio
ZC702	Platform development board based on the Zynq Z-7020 EPP device
Zynq Z-7020	An implementation of the Zynq-7000 EPP with a fixed feature set and PL capabilities

Functional Description

This chapter describes the Zynq-7000 ZC702 Base Targeted Reference Design (TRD) hardware design, software system, and video demonstration application components. It also describes how data flows through the various connected IPs and includes information about the flow of application control.

Hardware Architecture

The block diagram for the Base TRD is shown in Figure 2-1. This design has two parts:

- Processing system (PS)
- Video IPs and custom logic implemented in programmable logic (PL)

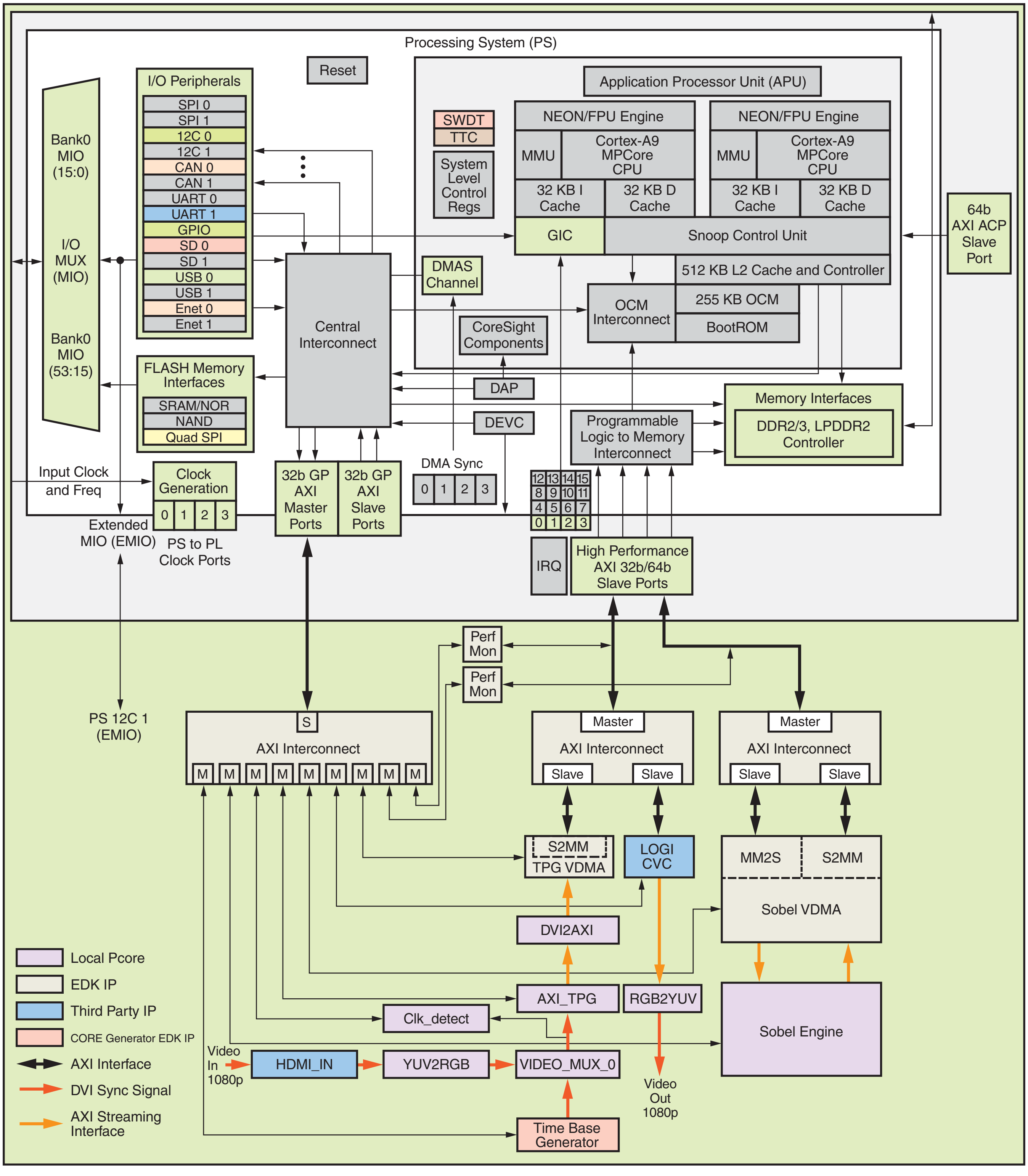


Figure 2-1: **Zynq-7000 EPP Base TRD Hardware Block Diagram**

This system is implemented in a Zynq-7000 EPP device (xc7z020clg484-1) using ISE® Design Suite, version 14.1 tools.

The PL hardware utilization for the implemented design is shown in Table 2-1.

Table 2-1: **PL Hardware Utilization for Device xc7z020clg484-1[(1)]**

FPGA Components	Total Available	Used	% of Used
LUTs	53,200	16,366	30
I/Os	200	51	25
FPGA Logic Memory			
RAMB36E1/FIFO36E1s	140	24	17
RAMB18E1/FIFO18E1s	280	15	5

Notes:

1. The figures provided here are only indicative of nature and can vary between different tool chain versions.

The PL-implemented video IP and custom logic address map is shown in Table 2-2.

Table 2-2: **FPGA Logic Address Map for the Zynq-7000 ZC702 Base TRD**

Instance	Peripheral	Base Address	High Address
CLK_DETECT_0	clk_detect_v1_00_a	0x40060000	0x4006FFFF
VTC_0	axi_vtc_v3_00_a	0x40070000	0x4007FFFF
TPG_0	axi_tpg_v2_00_a	0x40080000	0x4008FFFF
TPG_VDMA	axi_vdma_v5_01_a	0x40090000	0x4009FFFF
SOBEL_ENGINE	sobel_filter_top_v1_00_a	0x400c0000	0x400cFFFF
		0x400d0000	0x400dFFFF
SOBEL_VDMA	axi_vdma_v5_01_a	0x400b0000	0x400bFFFF
LOGICVC_0	LogiCVC_v2_04_a	0x40030000	0x4003FFFF
PERF_MON_HP2	axi_perf_mon_v1_00_a	0x400E0000	0x400EFFFF
PERF_MON_HP0	axi_perf_mon_v1_00_a	0x400F0000	0x400FFFFF

System Configuration

Processing System

This design makes full use of these four major components in the PS:

- Application processor unit (APU)
- Interconnect
- Input/output peripherals (IOP)

- Memory interfaces

This section describes some of the features of the PS used in this design. For detailed information about the complete feature set including a functional description, see UG585, *Zynq-7000 Extensible Processing Platform Technical Reference Manual* [Ref 2].

APU

The APU includes the dual ARM Cortex-A9 core processor, snoop control unit (SCU), L2 cache controller, on-chip memory (OCM), 8-channel DMA, system watchdog timer (SWDT), and triple timer controller (TTC) blocks.

Cortex-A9 Core - The ARM Cortex-A9 core processor implements the ARMv7 architecture and runs 32-bit ARM instructions, 16-bit and 32-bit Thumb instructions, and 8-bit Java byte codes in the Jazelle state. The media processing engine implements ARM NEON coprocessor technology, a single instruction multiple data (SIMD) architecture that adds instructions targeted at audio, video, 3D graphics, image, and speech processing.

General Interrupt Controller - The GIC collects interrupts from various sources and distributes these interrupts to each of the ARM cores. The interrupt distributor holds the list of pending interrupts for each ARM Cortex-A9 core processor and then selects the highest priority interrupt before issuing it to the Cortex-A9 processor interface. Interrupts of equal priority are resolved by selecting the lowest ID. A total of 64 shared peripheral interrupts (PL interrupts + PS I/O peripheral interrupts) are supported, starting from ID 32. Table 2-3 lists interrupt IDs for interrupts coming from PL.

Table 2-3: **Interrupt IDs for PL-Generated Interrupts**

Interrupt line	ID	Type	Instance
CVC_DISPLAY_interrupt	91	Level	CVC_DISPLAY
TPG_VDMA_s2mm_introut	90	Level	TPG_VDMA
SOBEL_VDMA_s2mm_introut	89	Level	SOBEL_VDMA
SOBEL_VDMA_mm2s_introut	88	Level	SOBEL_VDMA

Interconnect

The interconnect unit connects all PS and PL master and slave devices. There are a total of six Advanced eXtensible Interface (AXI) slave ports dedicated for AXI masters residing in the PL, and four of these ports contain deep FIFOs to improve data throughput. Two AXI master ports provide access to AXI slaves in the PL. In this design, masters in PL are connected through two AXI slave ports with deep FIFOs. One AXI master port is used to access registers in AXI slave IPs in PL.

An advanced peripheral bus (APB) master port is provided for accessing software programmable registers of all PS modules. The top level switch is AXI3-compliant, the soft IPs provided by Xilinx are AXI4-compliant, and the soft AXI interconnect IP provides protocol bridging as needed.

S_AXI_HP - The high performance slave AXI interfaces (S_AXI_HP) connect the PL to AFI blocks in the PS. The PL has four AXI masters out of which two are connected to the S_AXI_HP0 port and two are connected to the S_AXI_HP2 port. The HP port enables a high throughput data path between AXI masters in the programmable logic and the processing system's DDR3 memory. The main aim of the AXI FIFO interface (AFI) units is to smooth out this variable latency, allowing the ability to stream data continuously from DDR to the PL masters and from the PL masters to DDR. The PL-side interface of AFI runs on the clock coming from the PL. In this design, a 150 MHz clock is connected from the PL side. The DDR-side clock is running on 2/3 of the DDR_CLK (533 MHz). The high performance AXI interface module provides several "hooks" to assist in bandwidth management of masters connected to different PL ports. Controlling issuance capability available from the PL port is one of the hooks exercised in this design to obtain a fair share of bandwidth between two masters, SOBEL VDMA, and the display controller.

M_AXI_GP - This AXI master port interfaces with AXI slave IPs in PL through an AXI Lite interconnect. The CPU manages initializing and controlling the video pipeline through this port.

IOP - The IOP unit includes communication peripherals. GPIO, Ethernet, USB, I2C, and SD controllers from the PS are used extensively in this design.

GPIO - The 64-bit general purpose input/outputs (GPIOs) are connected to the PL through the extendable multiplexed I/O (EMIO) interface. Sixty-four bits are divided into two banks, each of 32 bits. Because each GPIO bit can be dynamically configured as input or output, GPIO bits are used in this design for a variety of functions. Table 2-4 lists the GPIO bit and purpose in design.

Table 2-4: **GPIO Bits Functional Description**

GPIO Bit Number	Net Name	Purpose
0	ps7_0_GPIO_O[0]	Resets video receiving block dvi2axi bridge
1	ps7_0_GPIO_O[1]	Resets Sobel engine block
3	ps7_0_GPIO_O[3]	Selects a line for the video multiplexer—either the external video source or the internally generated test pattern
6	ps7_0_GPIO_O[6]	IMAGEON FMC I2C multiplexer reset
7	ps7_0_GPIO_O[7]	Onboard I2C multiplexer reset
2, 4, 5	N/A	N/A

Memory Interfaces

The memory interfaces unit includes the DDR memory controller and nonvolatile memory controllers. The DDR memory controller includes a 4-port arbiter. One AXI port is dedicated for ARM CPU access and two ports are dedicated for high performance AXI interface master devices in the programmable logic. The remaining port is shared by all other AXI masters. In this design, DDR3 is configured to run at 533 MHz, and the AXI interface is running at 355 MHz.

PL Clocks

The PS provides four (FCLKCLK[3:0]) fully programmable clocks to the PL. These clocks are routed directly to PL clock buffers to serve as a frequency source for the PL. The clock generator module in PL gets a 100 MHz clock from FCLKCLK[0].

PL Reset

The PS provides four (FCLKRESETN[3:0]) fully programmable reset signals to the PL. These signals are asynchronous to PS clocks. The PL logic reset block in this design receives input from FCLKRESETN[0] and generates necessary reset signals for the design implemented in PL.

Programmable Logic

Clocking

The FPGA logic design has three clock domains: AXI MM (memory-mapped) interconnect, AXI register interface, and video clock. These domains run at 150 MHz, 75 MHz, and 148.5 MHz, respectively.

The clock generator module receives a 100 MHz input clock from the PS FCLKCLK[0] and generates 75 MHz and 150 MHz. The AXI Lite interconnect works on 75 MHz. Apart from the AXI Lite interconnect, the register interface of AXI VDMA, AXI TPG, logiCVC-ML, AXI VTC, sobel_filter, perf_monitor, and clk_detect cores are driven by the 75 MHz clock.

Two instances of the AXI_MM interconnect connected to the HP port of the PS run on 150 MHz. The S2MM (stream to memory map) and MM2S (memory map to stream) channels of VDMAs are running at 150 MHz. The 150 MHz clock also drives the logiCVC-ML memory read interface.

The video clock comes from the external clock synthesizer or from the AVNET FMIO card. BUFGMUX dynamically selects which video clock source drives logic running on the video clock domain. The AXI TPG, AXI VTC, dvi2axi, logiCVC-ML, and rgb2YCbCr 4:2:2 blocks run on the video clock.

Table 2-5 lists system clocks.

Table 2-5: **System Clocks**

Clock Signal	Source	Frequency	Use
FPGA_CLK	PS - FCLK_CLK0	100 MHz	Input clock to clock generator
clk_75mhz	Internal mixed-mode clock manager (MMCM)	75 MHz	Slave clock for AXI Lite interconnect, generated by clock generator
clk_150mhz	Internal MMCM	150 MHz	Clock for AXI MM interconnect, generated by clock generator
VIDEO_CLK_P, VIDEO_CLK_N	External video clock coming from clock synthesizer on board	148.5 MHz	Clock for display controller and video receiving block
fmc_imageon__in_0_clk_pin	External video clock coming from AVNET FMIO card	148.5 MHz	Clock for video receiving modules

Based on user clock configuration inputs, the clock generator determines the correct configuration of the PLLs.

Table 2-6 shows clock requirements of master and slave peripherals connected in system and their connection.

Table 2-6: **PL Clock Configuration**

Component	Frequency (MHz)	Phase	Buffered	Connection
clock_generator_0				
• CLKIN	100		Yes	FPGA_CLK
• CLKOUT0	75		Yes	clk_75mhz
• CLKOUT1	150		Yes	clk_150mhz
VIDEO_MUX_0				
• video_clk_1	148.5	0	Yes	VIDEO_CLK
• video_clk_2	148.5	0	Yes	fmc_imageon_hdmi_in_0_clk_pin
• video_clk	148.5	0	Yes	video_clk_int
Processor				
eps7_0				
• FCLK_CLK0	100	0	Yes	FPGA_CLK
• M_AXI_GP0_ACLK	75	0	Yes	clk_75mhz
• S_AXI_HP0_ACLK	150	0	Yes	clk_150mhz
• S_AXI_HP2_ACLK	150	0	Yes	clk_150mhz
Buses				
axi4_0				

Table 2-6: **PL Clock Configuration *(Cont'd)***

Component	Frequency (MHz)	Phase	Buffered	Connection
• INTERCONNECT_ACLK	150	0	Yes	clk_150mhz
axi4_1				
• INTERCONNECT_ACLK	150	0	Yes	clk_150mhz
axi4_lite				
• INTERCONNECT_ACLK	150	0	Yes	clk_150mhz
Peripherals				
Proc_sys_reset_1				
• Slowest_sync_clk	75	0	Yes	clk_75mhz
LOGICVC_0				
• S_AXI_ACLK	75	0	Yes	clk_75mhz
• mclk	150	0	Yes	clk_150mhz
• vclk	148.5	0	Yes	VIDEO_CLK
SOBEL_ENGINE				
• SYS_CLK	150	0	Yes	clk_150mhz
• s_axi_CONTROL_BUS_ACLK	75	0	Yes	clk_75mhz
• s_axi_SOBEL_CONTROL_ACLK	75	0	Yes	clk_75mhz
SOBEL_SWRST_FF				
• Clk	75	0	Yes	clk_75mhz
SOBEL_VDMA				
• m_axi_mm2s_aclk	150	0	Yes	clk_150mhz
• m_axi_s2mm_aclk	150	0	Yes	clk_150mhz
• m_axis_mm2s_aclk	150	0	Yes	clk_150mhz
• s_axi_lite_aclk	75	0	Yes	clk_75mhz
• s_axis_s2mm_aclk	150	0	Yes	clk_150mhz
TPG_SWRST_FF				
• Clk	75	0	Yes	clk_75mhz
TPG_VDMA				
• m_axi_s2mm_aclk	150	0	Yes	clk_150mhz
• s_axi_lite_aclk	75	0	Yes	clk_75mhz
• s_axis_s2mm_aclk	150	0	Yes	clk_150mhz
PERF_MON_HP0				
• AXI_CLK	150	0	Yes	clk_150mhz
• S_AXI_ACLK	75	0	Yes	clk_75mhz

Table 2-6: **PL Clock Configuration *(Cont'd)***

Component	Frequency (MHz)	Phase	Buffered	Connection
PERF_MON_HP2				
• AXI_CLK	150	0	Yes	clk_150mhz
• S_AXI_ACLK	75	0	Yes	clk_75mhz
VTC_0				
• video_clk_in	148.5	0	Yes	video_clk_int
TPG_0				
• clk	148.5	0	Yes	video_clk_int
• S_AXI_ACLK	75	0	Yes	clk_75mhz
CLK_DETECT_0				
• DUT_CLK	148.5	0	Yes	video_clk_int
• S_AXI_ACLK	75	0	Yes	clk_75mhz
HDMI_IN				
• clk	148.5	0	Yes	fmc_imageon_hdmi_in_0_clk_pin
YUV2RGB				
• clk	148.5	0	Yes	fmc_imageon_hdmi_in_0_clk_pin

Reset

The proc_sys_reset module implements a reset scheme. Input to the proc_sys_reset core is generated by PS FCLK_RESET0. The polarity of input reset to this block is indicated by parameter C_EXT_RESET_HIGH. In this design, C_EXT_RESET_HIGH is set to 0 as reset generated by PS is active-Low. This block generates various types of resets, such as reset for interconnect, peripheral reset, and so on. All the blocks in the PL are driven by interconnect reset, which is active-Low in polarity.

For detailed information about the complete feature set and a functional description of the proc_sys_reset IP, refer to DS406, *LogiCORE IP Processor System Reset Module Product Specification* [Ref 3].

AXI Interconnect

FPGA logic design has two interconnects for AXI memory-mapped masters and one interconnect for the AXI register interface.

AXI memory-mapped interconnects are connected to masters like AXI_VDMA and logiCVC-ML. Slaves connected to these interconnects includes HP0 and HP2 ports of Zynq-7000 EPP PS. This interconnect operates at 150 MHz and the data width is 64-bit wide. The read/write acceptance and issuance are set to 8. The acceptance and issuance helps improve system performance. The PS HP port can accept a maximum burst length of 16.

This imposes a limitation on getting minimum acceptable bandwidth for every master in a multi-master system. The optimum setting of issuance and acceptance reduces throttle on the bus and compensates for long latencies.

The AXI register interface is clocked at 75 MHz. The Zynq-7000 EPP PS GP0 port acts as master on this interconnect and connected slaves have register maps. AXI TPG and AXI VTC are examples of slaves connected to this interconnect. The operations of the video pipeline are controlled by registers inside every IP. Depending upon data flow required in the video pipeline, the processor writes these registers through the AXI Lite interconnect. The AXI Lite interconnect accepts write or read transfers from the CPU, performs address decoding, selects a particular slave, and establishes a communication channel between the CPU and the slave device.

For detailed information about the complete feature set and a functional description of the AXI Interconnect IP, refer to DS768, *LogiCORE IP AXI Interconnect* [Ref 4].

AXI VDMA

AXI VDMA has an AXI streaming interface on one side and an AXI memory-mapped interface on the other side. The VDMA has two channels: MM2S (memory-mapped to streaming) and S2MM (streaming to memory-mapped). The MM2S channel reads the number of data beats programmed through the C_MM2S_MAX_BURST_LENGTH parameter and presents it to the slave device connected through the streaming interface. The data width of the streaming interface can be different than the memory-mapped interface and controlled through C_M_AXIS_MM2S_TDATA_WIDTH. The data width of the S2MM memory-mapped interface is controlled by the C_M_AXI_MM2S_DATA_WIDTH parameter.

The S2MM channel receives data from the master device connected through the streaming interface. The C_S_AXIS_S2MM_TDATA_WIDTH parameter decides the width of the streaming interface. Data received on the streaming interface is then written into the system memory through the memory-mapped interface. The C_M_AXI_S2MM_DATA_WIDTH parameter decides the data width of the memory-mapped interface and C_S2MM_MAX_BURST_LENGTH governs the burst length of the write transaction.

In this design, the streaming interface data width is set to 32-bit wide and the memory-mapped interface is configured as 64-bit wide. The AXI VDMA is used in simple register direct mode, which removes the area cost of the scatter gather feature. Initialization, status, and management registers in the AXI VDMA core are accessed through an AXI4-Lite slave interface. To get the best possible throughput for AXI VDMA instances, the maximum burst length is set to 16. In addition, the master interfaces have a read and write issuance of 8 and a read and write FIFO depth of 512 to maximize throughput. The line buffers inside the AXI VDMA for the read and write sides are set to 4K deep and the store and forward feature of the AXI VDMA are enabled on both channels to improve system performance and reduce the risk of system throttling.

For detailed information on the complete feature set and a functional description of AXI VDMA IP, refer to PG020, *LogiCORE IP AXI Video Direct Memory Access Product Guide* [Ref 5].

AXI VTC

The AXI VTC is a general purpose video timing generator and detector. The input side of this core automatically detects horizontal and vertical synchronization pulses, polarity, blanking timing, and active video pixels. This information can be used by application software to take various decisions and for configuration of the video pipeline. In the current design, application software measures resolution of external video and then decides whether to switch to the external video source or not. The same feature can be expanded in the future to configure the video pipeline based on input resolution.

The output side of the core generates the horizontal and vertical blanking and synchronization pulses. The width and interval of these pulses are configured through the AXI Lite interface. The AXI TPG block generates a video test pattern based on video timing pulses generated by AXI VTC. In this design, VTC is used to generate video timing signals to match Full HD (1080p60) video format.

The video timing generator/detector block and AXI Lite interface of this core work on a single clock domain, that is, the video clock.

For detailed information on the complete feature set and a functional description of AXI VTC IP, refer to PG016, *LogiCORE IP Video Timing Controller Product Guide* [Ref 6].

Clock Detect

The video pipeline in this design can take video input either from an external video source or from the internally generated video test pattern. There is a possibility that the user selects external video mode and the FMC is not present. This scenario can lead to application failure and might drag the design into an unrecoverable state. This core gives provision to the application to avoid such a scenario. The core determines the frequency of the incoming video signal, and based on this value, the application decides whether the incoming video rate is suitable for the application's functioning or not.

The management and configuration of core is controlled through the AXI register interface. The sampling duration register holds the number of cycles for which the internal cycle counter is On. This core has two counters: The sampling counter works on the AXI Lite clock and the cycle counter runs on the video clock. Both counters start counting after the EN bit in the control register is set. The sampling counter generates load pulse upon reaching the terminal count specified by the sampling duration register. The cycle counter state is then loaded into the cycle counter register. Both counters clear their states and start counting again until the EN bit in the control register is set to `0`.

The register value of the cycle count can be used by the application to decide whether a video clock source is present or not, and if present, what the frequency of the video clock is. The value of cycle count read from the register also helps determine the input video standard, for example, if the cycle count is around 25 MHz, it is VGA mode. If it is 75 MHz, it is in HD ready (720p) mode, and if 148 MHz, it is Full HD (1080p).

AXI TPG

The AXI TPG contains an AXI register interface to access slave control registers from a processor. This IP can generate patterns like color bars, horizontal and vertical burst patterns, and zone plates. The generation of pattern is controlled through the pattern control register. It also enables the overlay of a box on a selected pattern. The motion control register controls the speed at which the box moves over a selected pattern. In this design, a zone plate pattern is used with a moving box. The size of the zone plate is controlled through zplate hdelta and zplate vdelta registers. The box size register controls the size of the box, and the color of the box is selected by the box color register. The width and height of the pattern is equal to 1920 x 1080, selected through the line length and frame height register.

logiCVC-ML

The logiCVC-ML is a multi-layer video display controller from Xylon [Ref 28]. The logiCVC-ML controller refreshes the display image by reading the video memory and converting the read data into a data stream acceptable for the display interface. It generates control signals for the display, and supports multiple layers with video processing functions such as alpha blending, transparency, and move around.

For detailed information about the complete feature set and a functional description of logiCVC-ML IP, refer to the Xylon data sheet [Ref 28].

AXI Performance Monitor

The AXI Performance Monitor can monitor and analyze system behavior on the AXI interface. This core is used in the Base TRD to measure read and write throughput on AXI slave ports of the PS (HP0 and HP2), which are used to access DDR memory from PL. The core consists of the AXI4-Lite interface to configure and control the core.

This core is configured to measure the read and write throughput by counting the number of transactions per second. When the configured time interval expires, measured throughput in bytes is loaded into a register and read by the software application.

Two AXI Performance Monitors are instantiated to measure read and write throughput of HP0 and HP2 simultaneously.

fmc imageon hdmi in

This IP core receives video from IMAGEON FMC, in YCbCr 4:2:2 format, with embedded vblank and hblank signals, and extracts blanking information.

YCbCr 4:2:2 RGB

The IMAGEON FMC provides video in YCbCr 4:2:2 format and all the video processing is done in RGB format.

This IP converts the video received by fmc imageon hdmi in IP in YCbCr 4:2:2 to RGB 4:4:4 format.

This core consists of two IPs: Chroma resampler (converts YCbCr from 4:2:2 to 4:4:4) and YCbCr 4:2:2 to RGB. These two cores are generated by the CORE Generator™ tool, and their netlists are used to make the core.

The croma resampler IP is generated with these settings:

- Maximum number of columns: **1920**
- Maximum number of rows: **1080**
- Fixed coefficient low pass filtering

The YCbCr 4:2:2 to RGB IP is generated with these settings:

- Standard selection: **HD ITU 709 1250 PAL**
- Input range selection: **0** to **255** for computer graphics
- Input data width: **8**
- Output data width: **8**
- Coefficient bits: **18**

Software Architecture

This section explains the software architecture for the Zynq-7000 ZC702 Base TRD. Figure 2-2 illustrates a top level view of the software architecture.

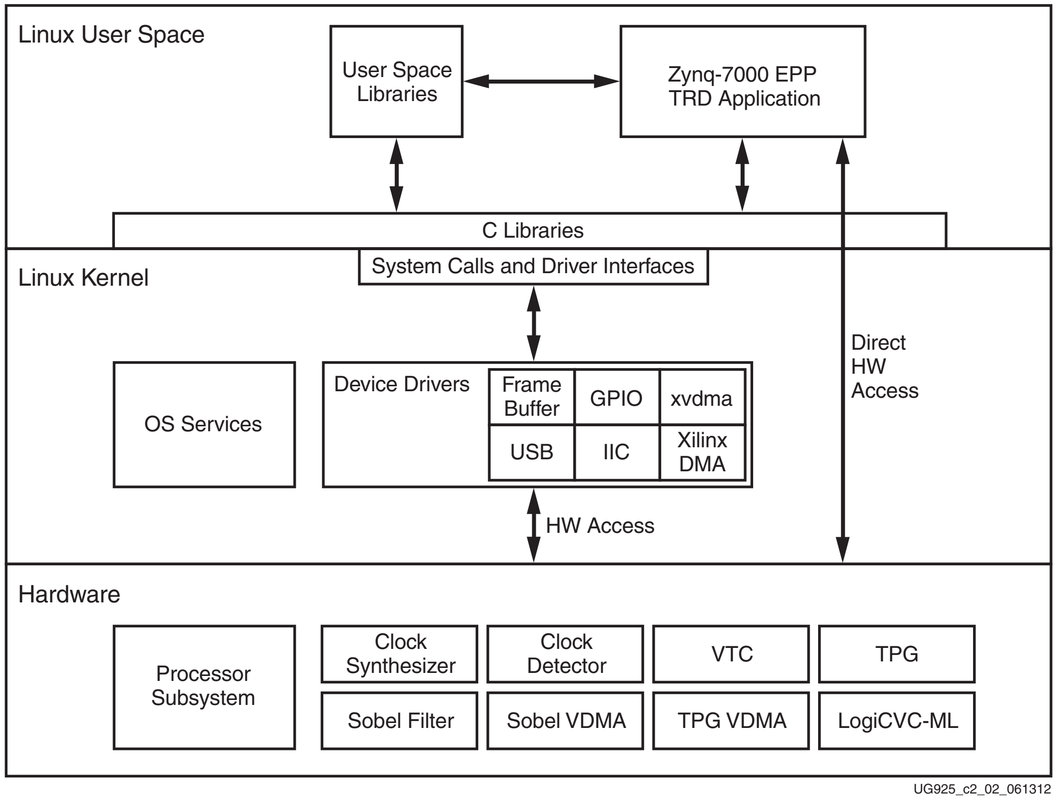


Figure 2-2: **Software Architecture: Top Level View**

A multithreaded Linux application is responsible for running the TRD demonstration. This application uses the Qt-based GUI, which is displayed through Display Monitor, to obtain user inputs. Depending on the inputs, it calls device drivers to configure the hardware and to enable a particular data path. Figure 2-2 illustrates the various data paths.

Three major software components are involved in the Base TRD:

- Boot loader
- Xilinx Linux kernel
- Application

Boot Loader

A two-stage boot loader is used for the Zynq-7000 EPP Linux boot-up. The FSBL is responsible for initializing required hardware and loads the second-stage boot loader, U-Boot, which is responsible for loading kernel images in the DDR memory.

The FSBL source code is generated through the Xilinx SDK tool, depending on the hardware design specification. That source code is modified for the Base TRD to initialize the onboard HDMI out chip.

U-Boot is an open source universal boot loader used across various embedded platforms. The source code, customized for Zynq-7000 EPP Linux, is available on the Xilinx Open Source ARM git Repository [Ref 7].

Refer to *Xilinx Zynq Embedded Processors Base TRD* wiki pages [Ref 33] to build the FSBL and U-Boot.

Xilinx Linux Kernel

The Xilinx Linux kernel is based on the mainline open source kernel git tree, adding support for a variety of Xilinx IP core drivers and reference boards. The source code is available on the Xilinx Open Source ARM git Repository [Ref 7].

The Xilinx Linux kernel is extended (patched) to support IPs specific to this Base TRD. Patching and building the Linux kernel is explained in *Xilinx Zynq Embedded Processors Base TRD* wiki pages [Ref 33].

Table 2-7 lists kernel drivers used for the Base TRD.

Table 2-7: **Linux Kernel Drivers Used by the Base TRD**

Linux Driver	Function	Called By
Frame buffer	Drives the display controller (logiCVC-ML) to display the application UI and control data path	Application
Frame buffer	Frame buffer console display	Linux console
PS - IIC	Controls the clock generator to generate the desired clock for interfaces	Frame buffer driver, depending on the resolution to be used
	Configures the HDMI IN interface on the FMIO card	Application
PS - GPIO	Provides Reset signals to VDMA IPs	Application
XVDMA	Controls VDMA IP for various data flows	Application
Xilinx DMA	Provides core DMA functionality and the actual hardware interface	XVDMA driver

All of the drivers in Table 2-7 except the XVDMA driver come with the standard Xilinx Linux kernel. So the XVDMA driver is patched into the kernel.

Frame Buffer Driver

Linux provides a standard frame buffer, which is hardware-independent, and the application can use this buffer without knowing the underlying display controller. The Xylon frame buffer driver for CVC IP is registered with the standard frame buffer driver to provide support for the logiCVC-ML display controller.

The Xylon frame buffer driver is compiled with the kernel and probes for the hardware and resolution specification by scanning the `dtb` file at boot. If there is no entry for logiCVC-ML IP in the `dtb` file, then the driver does not load itself.

The application uses the standard Linux frame buffer driver, which has a character driver interface.

For this Base TRD, `/dev/fb0` is the node that is populated. To find the exact device node, iteratively match the **id** field of structure fb_fix_screeninfo acquired by the FBIOGET_FSCREENINFO IOCTL call for each device node populated. If the **id** field is *Xylon FB*, the frame buffer driver is for Xylon logiCVC-ML.

Note: An IOCTL (input/output control) is a general purpose Linux system call used for implementing the interface between a user application and a device driver.

More details on the frame buffer driver is available in kernel documentation at `<Linux Kernel source>/Documentation/fb/framebuffer.txt`.

PS-IIC Driver

The IIC bus is used for configuring the HDMI IN controller and the onboard clock synthesizer. The I2C driver is a standard Linux I2C driver populated as `/dev/i2c-0`. Standard I2C IOCTL calls are used for configuring the I2C devices.

PS-GPIO Driver

The GPIO SYSFS interface is used for GPIO configuration. The GPIO SYSFS interface allows the user to control I/O pins using files under the `/sys` directory.

When the system boots, all GPIO pins are owned by the kernel. The pins do not show up in the SYSFS system until they are exported. To export them, write the pin number (say 54) to the file `/sys/class/gpio/export`.

This results in the pseudo files for pin 54 showing up under `/sys/class/gpio/gpio54` as:

```
/sys/class/gpio/gpio54/direction
/sys/class/gpio/gpio54/value
```

The user can set the direction by writing **`in`** or **`out`** to the `direction` file. For the `out` direction, writing **`0`** or **`1`** on the corresponding `value` file resets or sets the pin, respectively. Similarly, the user can read the `value` file for the `in` direction.

XVDMA Driver

XVDMA is a character driver used for configuring and controlling video DMA transactions for both TPG and Sobel hardware. XVDMA internally calls the Xilinx DMA driver to complete the task and interrupt handling.

The device node that uses the XVDMA driver is `/dev/xvdma`. The following IOCTLs are defined for the XVDMA driver and contain the corresponding IOCTL arguments to be used:

- XVDMA_GET_NUM_DEVICES
  - This call obtains the number of VDMA probed and available for use.
  - Argument: Address of unsigned int [unsigned int *].
    - This gets filled up with the number of VDMA when the call returns.
- XVDMA_GET_DEV_INFO
  - This call gives device information like channel number, for the given VDMA ID.
  - Argument: Address of struct xvdma_dev [struct xvdma_dev *].
    - Before calling, the **device_id** field of this structure should be filled with VDMA ID. On return the rest of the structure is filled by the driver.
- XVDMA_DEVICE_CONTROL
  - This call sets the VDMA channel configuration.
  - Argument: Address of struct xvdma_chan_cfg [struct xvdma_chan_cfg *].
    - Before calling, this structure should be filled with required channel configurations.
    - To reset VDMA, only fill **`chan = <channel id>`** and **`config.reset = 1`** fields of structure.
- XVDMA_PREP_BUF
  - This call sets the buffer configurations.
  - Argument: Address of struct xvdma_buf_info [struct xvdma_buf_info *].
  - Before calling, this structure should be filled with required buffer configurations.
- XVDMA_START_TRANSFER
  - This call triggers the VDMA transfer.
  - Argument: Address of struct xvdma_transfer [struct xvdma_transfer *].
  - Before calling, this structure should be filled. The structure specifies the channel ID and whether the call is synchronous or asynchronous.
- XVDMA_STOP_TRANSFER
  - This call stops the VDMA.
  - Argument: Address of the unsigned int variable [unsigned int *].
  - Before calling, this int variable should be filled with the channel ID.

Structures used for the above IOCTL are defined in the `driver_include.h` file, which is part of the application source code provided with the Base TRD. The file contains comments explaining each field of the structure.

Xilinx DMA

This driver is not used by the application. It is internally called by XVDMA to perform operations on video DMA hardware.

Application

Figure 2-3 describes various components of the application.

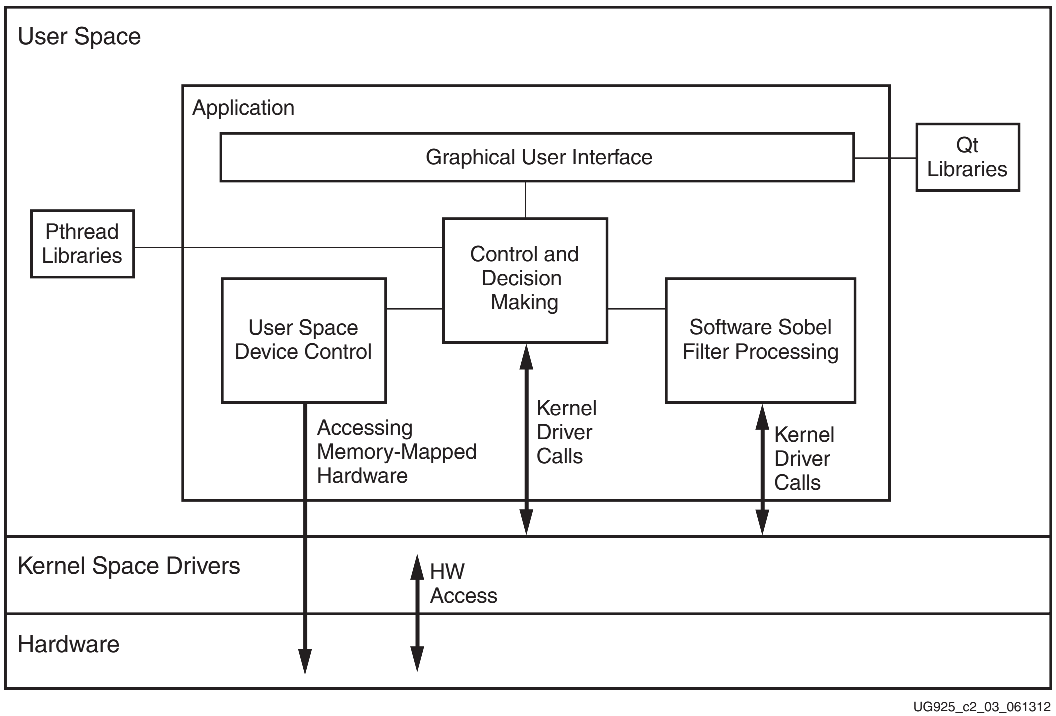


Figure 2-3: **Application Functional Blocks**

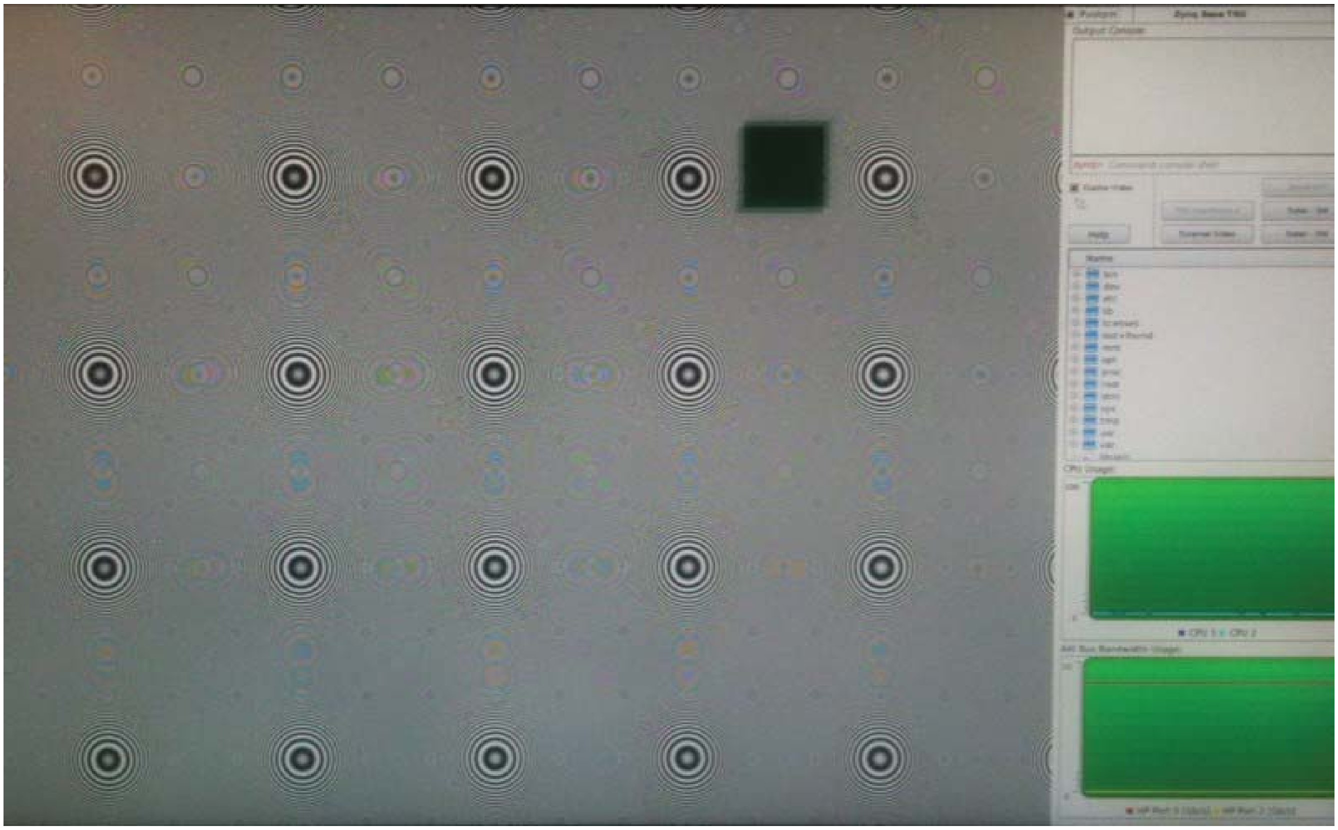
The application is divided into the following functional blocks:

- GUI
- Control and decision making
- User space device control
- Software Sobel filter processing

The first three components run in one thread of the application, while the software Sobel filter runs in a second separate thread.

Graphical User Interface

The GUI for this Base TRD is designed using the Qt framework (see Figure 2-4).



UG925_c2_04_061212

Figure 2-4: **GUI for TRD Application**

The main tasks the GUI performs include:

- Getting user inputs
- Plotting graphs
- Displaying the video area

The user input and graph fade away after three seconds of no activity. These reappear with a left-click. These can also be pinned on the display by selecting the **Pushpin** checkbox on the top right, so that these are always visible.

Get User Inputs

The Qt framework provides for having the keyboard and mouse as input devices (it internally uses Linux USB-HID class drivers). The input from the user includes **video Enable/Disable**, **Pattern Select**, and **Mode Select** for the video pipeline.

Plot Graph

Two graphs are plotted using the Qt framework. The first graph demonstrates CPU utilization for each ARM core, and the second demonstrates memory bandwidth utilization.

In the CPU utilization graph, the horizontal axis is for time and the vertical axis is for the percentage of CPU utilization.

In the memory bandwidth graph, the horizontal axis is for time and the vertical axis is for MB of read and write transactions.

Display Video Area

This is the full screen area, where the output of the video pipeline is displayed.

Control and Decision Making

This block receives input from the GUI and maintains the state transition for the complete application. It communicates with all other blocks of the application and with the kernel drivers to change the state of hardware.

The following hardware is configured through this control block using the kernel drivers mentioned in Xilinx Linux Kernel:

- VDMA (using the XVDMA driver)
- VDMA reset and multiplexer switching for external live video (using the GPIO driver)
- FMIO initialization for the HDMI IN (using the I2C driver)
- logiCVC-ML control (using the frame buffer driver)

There are six combinations of the data flow:

1. The TPG creates and writes the pattern in reserved video memory in DDR. The display controller displays video memory (TPG pattern). See Figure 2-5.

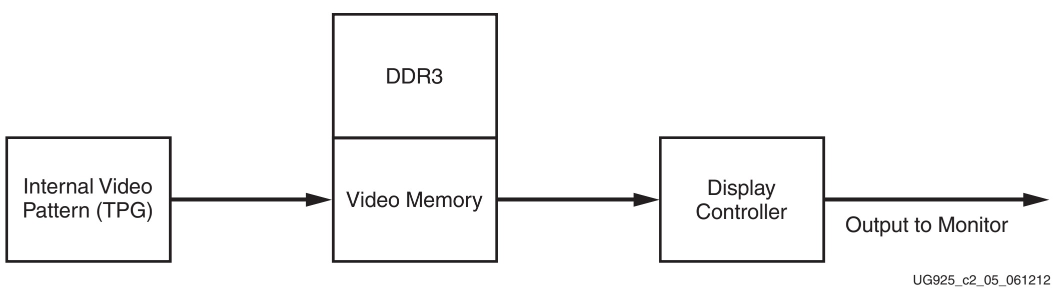


Figure 2-5: **Video Display Pipeline Data Flow**

2. The TPG creates and writes the pattern in intermediate DDR memory. The software Sobel filter reads the intermediate DDR memory, detects the edge, and writes the filtered image in the reserved video memory in DDR. The display controller displays video memory (filtered TPG pattern). See Figure 2-6.

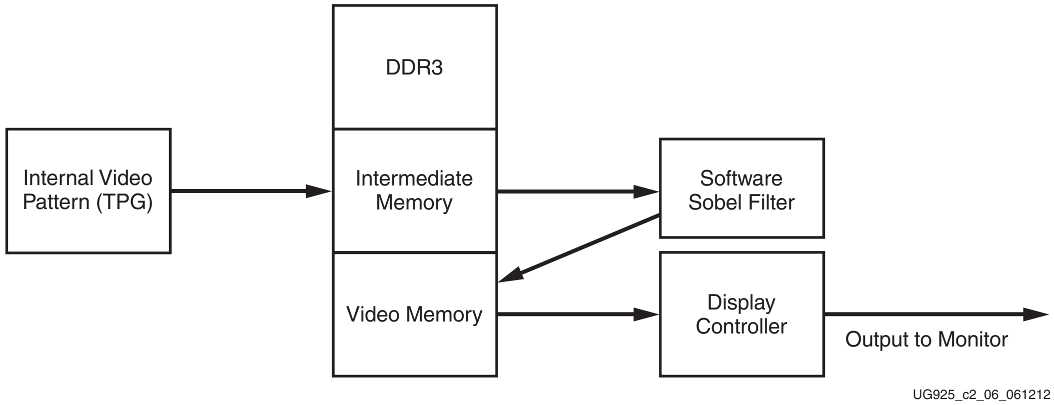


Figure 2-6: **Video Processing and Display Pipeline Data Flow**

3. The TPG creates and writes the pattern in intermediate DDR memory. The hardware Sobel filter reads the intermediate DDR memory, detects the edge, and writes the filtered image in the reserved video memory in DDR. The display controller displays video memory (filtered TPG pattern).

Similarly, there are three more cases (4, 5, and 6) where instead of an internally generated pattern, external live video is used (see Figure 2-7 and Figure 2-8).

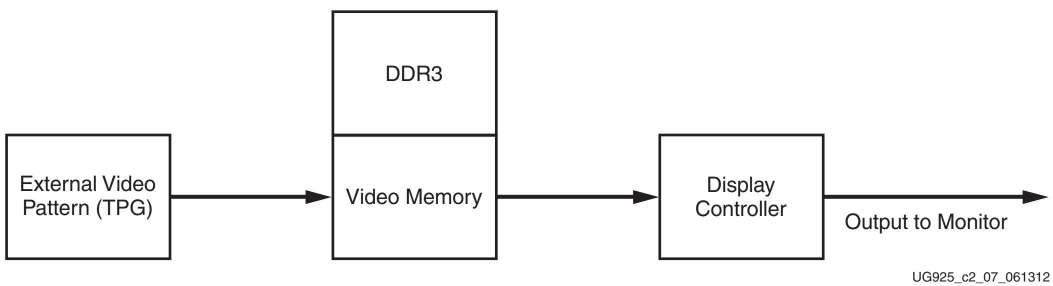


Figure 2-7: **Video Display Pipeline Data Flow**

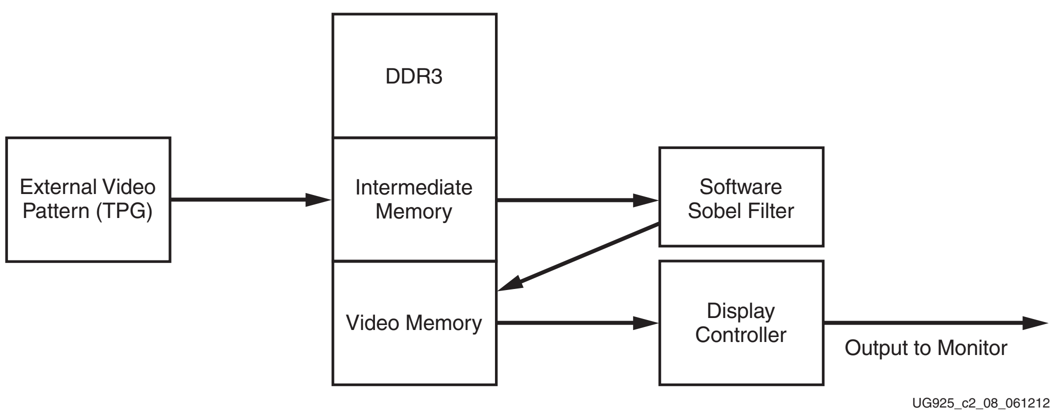


Figure 2-8: **Video Processing and Display Pipeline Data Flow**

User Space Device Control

All the IPs need not have a kernel space driver, especially when they are memory-mapped and do not use interrupt. Such IPs can be configured and controlled from the user space by mapping the physical address range to the virtual address space in the user space.

These IPs are configured and controlled from the user space:

- TPG
- Clock detector
- Video timing controller
- Sobel filter

The APIs to configure these IPs are explained in the `udriver.h` file, which is part of the application source code provided with the Base TRD.

Software Sobel Filter Processing

Here is the implementation of the Sobel filter algorithm for edge detection. It runs in a separate thread and is turned On and Off by the controlling block. This is compiled with the Neon flag enabled and with the highest compiler optimization level (O3). It takes the buffer address of the original frame as input and writes the filter image on the buffer address provided as output.

The following API is used for software Sobel filter processing:

```
void sw_sobel_processing (unsigned long in_buffer, unsigned long out_buffer)
```

Register Description

This appendix describes the details about configuration and control registers most commonly accessed by the Linux driver and application. The registers implemented in hardware are memory-mapped to the PS address range directly.

TPG Registers

The TPG registers are used to control various internal features within the TPG. The relative address of the TPG pattern selection register is `0x00`. Table A-1 describes this register's structure.

Base address	`0x40080000`
Version	2.0.0a

Pattern Selection Register

Relative address	`0x00`

Table A-1: **Pattern Selection Register**

Bit Position	Mode	Default Value	Description
31:14	-	-	Reserved
13	RW	0x0	H blank polarity
12	RW	0x0	V blank polarity
11	RW	0x0	H sync polarity
10	RW	0x0	V sync polarity
9	RW	0x0	Enable box
8:6	RW	0x0	Component mask
5	RW	0x0	CbCr polarity

Table A-1: **Pattern Selection Register *(Cont'd)***

Bit Position	Mode	Default Value	Description
4	RW	0x0	Enable X Hairs
3:0	RW	0x0	0000 - input pass through 0001 - horizontal ramp 0010 - vertical ramp 0011 - temporal ramp 0100 - red 0101 - green 0110 - blue 0111 - black 1000 - white 1001 - bars 1010 - ZonePlate/Sweep

Motion Control Register

The relative address of the TPG motion control register is `0x04`. Table A-2 describes this register's structure.

Relative address `0x04`

Table A-2: **Motion Control Register**

Bit Position	Mode	Default Value	Description
31:9	-	-	Reserved
8:1	RW	0x0	Motion Speed
0	RW	0x0	Motion Enable

Frame Size Register

The relative address of the TPG generated video frame size configuration register is `0x0C`. Table A-3 describes this register's structure.

Relative address `0x0C`

Table A-3: **Frame Size Register**

Bit Position	Mode	Default Value	Description
31:17	-	–	Reserved
27:16	RW	0x0	Frame height

Table A-3: **Frame Size Register *(Cont'd)***

Bit Position	Mode	Default Value	Description
15:12	-	–	Reserved
11:0	RW	0x0	Frame length

Z Plate H Delta Register

The relative address of the TPG Z-plate H-delta configuration register is 0x10. Table A-4 describes this register's structure.

Relative address 0x10

Table A-4: **Z Plate H Delta Register**

Bit Position	Mode	Default Value	Description
31:16	RW	0x0	Z Plate H Delta Start
15:0	RW	0x0	Z Plate H Delta

Z Plate V Delta Register

The relative address of the TPG Z-plate V-delta configuration register is 0x14. Table A-5 describes this register's structure.

Relative address 0x14

Table A-5: **Z Plate V Delta Register**

Bit Position	Mode	Default Value	Description
31:16	RW	0x0	Z Plate V Delta Start
15:0	RW	0x0	Z Plate V Delta

Box Size

The relative address of the TPG generated video box size configuration register is 0x18. Table A-6 describes this register's structure.

Relative address 0x18

Table A-6: **Box Size Register**

Bit Position	Mode	Default Value	Description
31:12	-	–	Reserved
11:0	RW	0x0	Box size

Box Color

The relative address of the TPG generated video box color configuration register is 0x1C. Table A-7 describes this register's structure.

Relative address 0x1C

Table A-7: **Box Color Register**

Bit Position	Mode	Default Value	Description
31:24	-	–	Reserved
23:0	RW	0x0	Box color

Clock Detect Registers

The clock detector's registers are used to detect the video clock.

Base address 0x40060000

Version 2.0.0a

Event Count Enable Register

The relative address of the clock detector's event count enable register is 0x00. Table A-8 describes this register's structure.

Relative address 0x00

Table A-8: **Event Count Enable Register**

Bit Position	Mode	Default Value	Description
31:1	-	–	Reserved
0	RW	0x0	Event count enable

Clock Count Register

The relative address of the clock detector's clock count register is `0x08`. Table A-9 describes this register's structure.

Relative address `0x08`

Table A-9: **Clock Count Register**

Bit Position	Mode	Default Value	Description
31:0	R	0x0	Clock counter value

Event Duration Register

The relative address of the clock detector's event duration register is `0x18`. Table A-10 describes this register's structure.

Relative address `0x18`

Table A-10: **Event Duration Register**

Bit Position	Mode	Default Value	Description
31:0	RW	0x0	After reaching this value, the event counter starts re-counting from zero. During the re-count, the clock counter is enabled.

Directory Structure

This appendix describes the directory structure and organization of the files and folders delivered with the package.

Included Files and Systems

Figure B-1 gives a top level view of the directories and files included in the Zynq-7000 ZC702 EPP device Base TRD package. Table B-1 summarizes how the directories provided relate to this user guide. This user guide uses the directories in the extracted Base TRD zipped package and assumes that the directories and files are placed or copied onto the user's local computer.

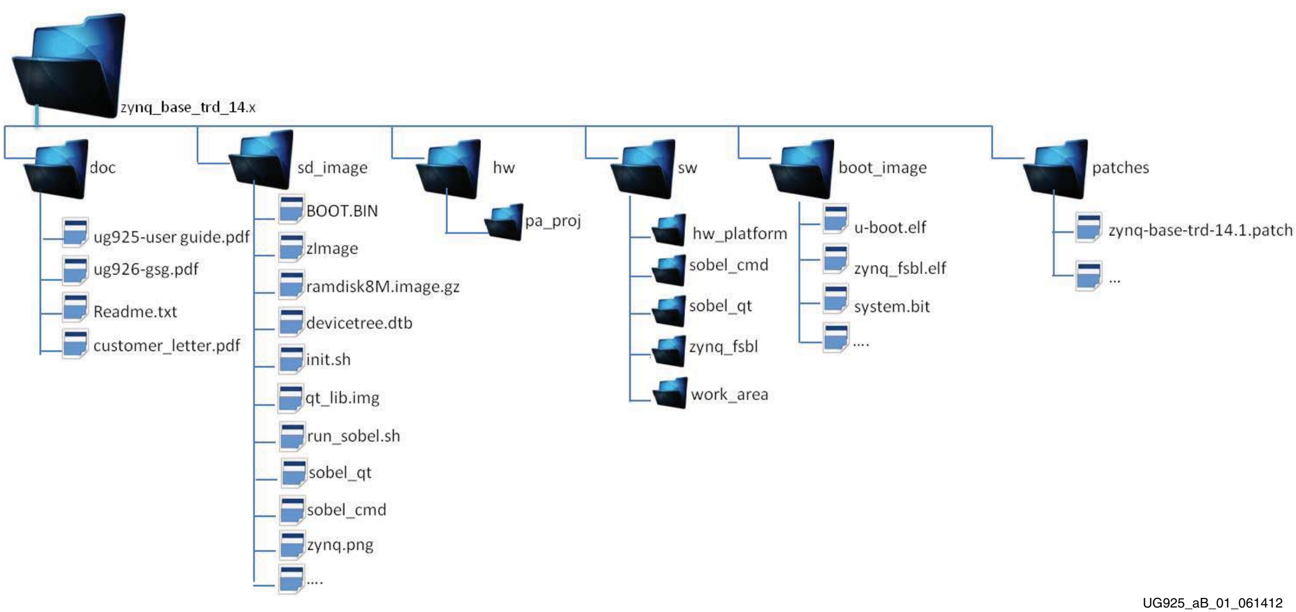


Figure B-1: **Directory Structure**

Table B-1: **Explanation of Directories in the Zynq-7000 ZC702 Base TRD File System**

Directory	Purpose
doc	This directory contains the documents provided with the Zynq-7000 ZC702 Base TRD, including this user guide.
sd_image	This directory contains a pre-built hardware design and software executables which provide a quick way to run the video demonstration. The application binary images are compatible to boot from the SD MMC and run the Linux application.
hw	This directory includes the Zynq-7000 ZC702 Base TRD hardware design.
sw	This directory includes the Base TRD applications source code and hardware platform exported from the PlanAhead™ tool. The included ISE-SDK project can be used to build the application.
boot_image	This directory includes the hardware design bitfile and other Zynq system configuration executables in binary.
patches	This directory includes Base TRD Linux patches to the Xilinx standard Zynq-7000 EPP Linux kernel.

Appendix C

Install the Zynq-7000 EPP Design and Development Environment

Install the Xilinx ISE Design Suite

Refer to UG798, *Xilinx Design Tools: Installation and Licensing Guide* [Ref 8] to install and license the Xilinx ISE Design Suite® tool.

Set Up Linux System Software Development Tools

To download and set up the ARM GNU tool chain for the Zynq-7000 EPP Linux system software and application development, refer to the Xilinx ARM GNU Tools wiki page [Ref 9]. The tool suite download requires a valid, registered Xilinx user login name and password.

Set Up git Tools

Git is a free software tool for managing distributed version control and the development of software project files. A git clone is a full-fledged repository with complete history and full revision tracking capabilities. Git gives the developer a local copy of the entire development project files and their existing (published) history on a git server within a repository organized as a set of directories. A user must clone a file or folder before using the same items from the project repository.

The user needs to have a git client installed on the Linux development PC. Refer to the Xilinx Using Git website [Ref 10] for working with Xilinx git and to section 5.1 of UG821, *Zynq-7000 EPP Software Developers Guide* [Ref 11] for working with Xilinx git Linux repositories.

Additional Resources

Xilinx Resources

To search the Answer database of silicon, software, and IP questions and answers, or to create a technical support WebCase, see the Xilinx Support website at:

www.xilinx.com/support

For a glossary of technical terms used in Xilinx documentation, see:

http://www.xilinx.com/company/terms.htm

To find additional documentation, see the Xilinx website at:

http://www.xilinx.com/support/documentation/index.htm

References

These documents provide supplemental material useful with this guide.

Zynq-7000 EPP Documents

1. UG926, *Zynq-7000 EPP ZC702 Evaluation Kit Getting Started Guide*
2. UG585, *Zynq-7000 Extensible Processing Platform Technical Reference Manual*
3. DS406, *LogiCORE IP Processor System Reset Module Product Specification*
4. DS768, *LogiCORE IP AXI Interconnect*
5. PG020, *LogiCORE IP AXI Video Direct Memory Access Product Guide*
6. PG016, *LogiCORE IP Video Timing Controller Product Guide*
7. Xilinx Open Source ARM git Repository
   http://git.xilinx.com/
8. UG798, *Xilinx Design Tools: Installation and Licensing Guide*

9. Xilinx ARM GNU Tools
http://wiki.xilinx.com/zynq-tools
10. Using Git
http://wiki.xilinx.com/using-git
11. UG821, *Zynq-7000 EPP Software Developers Guide*
12. Xilinx Zynq-7000 EPP website:
http://www.xilinx.com/products/silicon-devices/epp/zynq-7000
13. Zynq-7000 Extensible Product Platform Product Table:
http://www.xilinx.com/publications/prod_mktg/zynq7000/Zynq-7000-combined-product-table.pdf
14. Zynq-7000 EPP ZC702 Evaluation Kit: http://www.xilinx.com/ZC702
15. DS190, *Zynq-7000 Extensible Processing Platform Overview*
16. git: the fast version control system home page
http://git-scm.com/
17. Zynq Linux: Downloading the Kernel Tree
http://xilinx.wikidot.com/zynq-linux#toc7
18. Zynq Linux: Configuring and Building the Linux Kernel
http://xilinx.wikidot.com/zynq-linux#toc8
19. Xilinx Open Source Linux
http://wiki.xilinx.com/open-source-linux
20. Xilinx Device Tree Generator
http://xilinx.wikidot.com/device-tree-generator
21. Device Tree general information
http://devicetree.org/Main_Page
22. AMBA AXI4-Stream Protocol Specification
http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ihi0051a/index.html
23. PCI-SIG Documentation
http://www.pcisig.com/specifications
24. Xilinx PlanAhead Design and Analysis Tool website
http://www.xilinx.com/tools/planahead.htm
25. UG821, *Zynq-7000 EPP Software Developers Guide*
26. UG873, *Zynq Concepts, Tools, and Techniques Guide*
27. UG673, *Quick Front-to-Back Overview Tutorial: PlanAhead Design Tool*

Additional Useful Documents

Documents associated with other software, tools, and IP used by the Base TRD are available at these vendor or public websites:

28. Xylon IP Cores - logiCVC-ML Compact Multilayer Video Controller description
www.logicbricks.com/Products/logiCVC-ML.aspx
29. Qt Online Reference Documentation. Qt is a toolkit for creating GUIs.
http://doc.qt.nokia.com/
30. logiCVC-ML Compact Multilayer Video Controller Data Sheet
http://www.logicbricks.com/Documentation/Datasheets/IP/logiCVC-ML_hds.pdf
31. Silicon Labs CP210x USB to UART Bridge VCP Drivers
http://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx

Additional Useful Sites for Boards and Kits

More information on Zynq-7000 family boards, FMC extension cards, and other kits based on Zynq-7000 architecture is available here.

32. Xilinx Zynq-7000 EPP Boards and Kits
http://www.xilinx.com/products/boards_kits/zynq-7000.htm
33. Xilinx Zynq Embedded Processors Base TRD wiki page
http://wiki.xilinx.com/zynq-base-trd