mikroprojekt

KONDOR AX

Advanced System Development Board

VIDEO DEMO REFERENCE DESIGN GUIDE

28.10.2015.
UM0031
Rev. 1.1

Table of contents

Revision History

Revision	Date	Author	Modification
1.0	15.10.2015.	SH	Initial
1.1	28.10.2015	SH	Release

Related Documents

ID	Code	Description
1	UM0026	KONDOR AX – User Manual
2	UM0027	KONDOR AX – Linux BSP Build Setup Guide
3	UM0030	KONDOR AX – Video Demo Guide

1 Introduction

This document contains information about the video demo for the KONDOR AX - Advanced System Development Board, demonstrating how video data can be transmitted between the i.MX6 Solo ARM processor and the Lattice ECP5 FPGA.

The project files in this demo run in **Diamond 3.5.**

This document focuses on the FPGA part of the demo, or explaining the functionality implemented in the HDL files. For more information about the hardware requirements and demo running instructions consult the KONDOR AX Video Demo Guide document. For pinouts and other board specific information consult the KONDOR AX Board User Manual.

1.1 Directory structure

The demo project is organized in several folders, as shown in Figure 1.

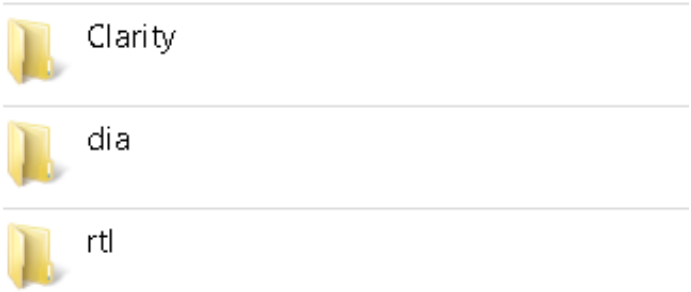

Figure 1: Demo directory structure

IP Cores used in the design are located in the 'Clarity' folder.

The ‘dia’ folder contains the Diamond project files (.ldf, .lpf), implementation files, several modules in netlist format (.ngo).

The ‘rtl’ folder contains the Verilog/VHDL source code files.

2 Functional description

2.1 Overview

The codename of the design for this demo is RD0012_kondor_video_demo.

This demo demonstrates how an application running on the i.MX6 ARM processor can use the PCIe x1 link to send video frames to ECP5 FPGA and then receive them back.

The FPGA stores the frame in LPDDR3 memory. Currently no processing of the video stream is implemented, but this should be easy to add.

Figure 2 shows the block diagram for this design.

The PCIe Endpoint IP and attached logic receive Transport Layer Packets (TLPs) which the i.MX6 sends over the PCIe link. The AXI Transaction Level Completer (AXI TLC) module parses incoming write TLPs and writes them to registers or memory. PCIe read requests are currently not supported.

Transferring the data back from FPGA to CPU memory is implemented by the PCI Endpoint sending write requests to the i.MX6 root complex, which will then write the data to CPU memory.

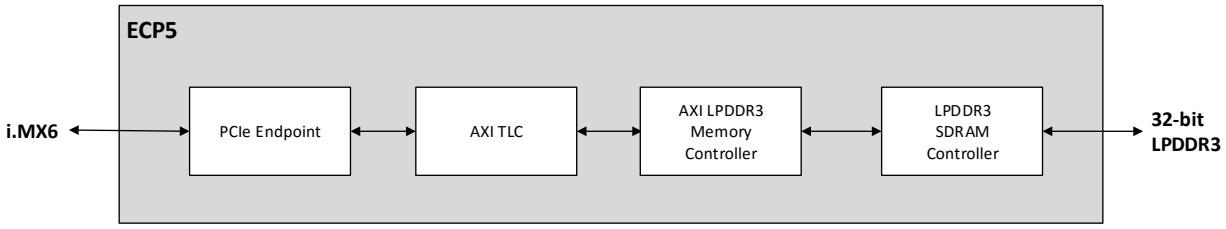

Figure 2: PCI Express Video Demo block diagram

2.2 Clocking

The differential reference clock for PCIe cores is 100 MHz. It is generated by the onboard PLL chip which needs to be configured over I²C. That is why all PCIe

designs include one Lattice Mico32 which serves only to configure the PLL chip. This Mico32 processor is supplied as a netlist

All PCIe-related logic and the AXI bus run on a 125MHz clock supplied by the PCIe Endpoint IP Core.

The LPDDR3 SDRAM Controller uses its dedicated onboard 100MHz reference clock (clk_in) to generate a 300 MHz memory clock, and a 150 MHz system clock (sclk). The

The reset signal (rstn) is generated by a 20-bit wide counter which is initialized on FPGA configuration, and runs on a 38.8 MHz clock generated by an internal oscillator.

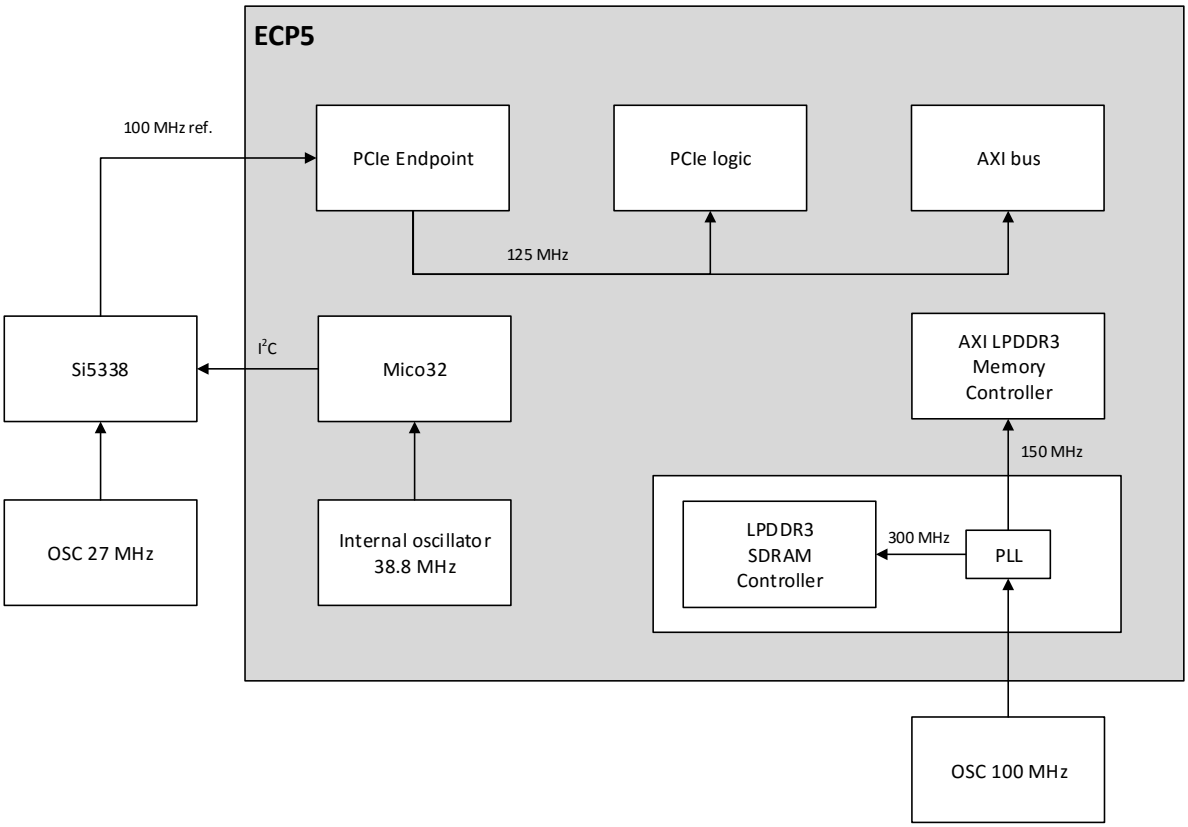

Figure 3 PCI Express Video Demo clocking scheme

2.3 CPU – FPGA Communication

The ECP5 is configured as a PCIe peripheral with two BARs (Base Address Registers). BAR0 is mapped to an 8 MB address region, and BAR1 to a 256 KB

region in the ARM's address space. BAR0 is used for writing to LPDDR3 memory, and due to architecture limitations only 8 MB are currently accessible. Usually BARs are used for accessing control registers, and not for high-bandwidth data transfers. High-bandwidth data transfers over PCIe require the PCIe Endpoint to act as a bus master. This usually means a DMA is needed on the PCIe Endpoint side.

This demo takes a hybrid approach.

The 'write' side is implemented by the CPU's DMA writing the frame into BAR0.

The 'read' side is implemented like this. The CPU requests a memory transfer from the PCIe Endpoint by configuring registers in BAR1. The AXI TLC then generates the appropriate write requests and sends them to the CPU.

BAR1 offset	Function
0x00	32-bit address in CPU address space
0x04	32-bit address on FPGA's AXI bus
0x08	length of block requested (in bytes), can be > 8MB
0x0C	writing any value to this register starts a transfer

Table 1 Read Request Register Map for BAR1

2.4 PCIe x1 interface

The PCIe x1 interface consists of three layers: The Transport, Adaptation and Application Layers.

The Transport Layer takes care of moving Transport Layer Packets across the PCIe link. The functionality of the Transport Layer is handled by the PCIe Endpoint IP Core (pcie_core module).

The Adaptation Layer unpacks PCIe TLPs and presents them to the Application Layer. In other words it translates PCIe transactions into Application Layer transactions (in this case AXI bus transactions). The PCIe specification mandates that all transactions that need a response need to receive one regardless of whether that type of transaction is supported or not. The UR_gen module generates responses for unsupported requests.

The axi_tlc module is the Transaction Layer Completer that converts PCIe transactions into AXI transactions. The ip_tx_arbiter module arbitrates which of the UR_gen and axi_tlc modules will transmit data to the Transport Layer.

For more detail about the PCIe x1 interface, please refer to [UG15].

The axi_intf module handles parsing of incoming write TLPs and writing to LPDDR3 memory. It also generates read commands from BAR1 registers. It pushes read commands to the read_control_fifo and the axi_tlc_cpld module pops them.

The axi_cpld_fifo module reads data from LPDDR3 memory and generates maximum size write TLPs (128 bytes) that are sent back to the CPU. Long write TLPs give good PCIe link performance.

2.5 LPDDR3 interface

The LPDDR3 interface consists of a versatile AXI LPDDR3 controller written in VHDL (iq_mem_axi_2clk_multiport), which translates AXI transactions to memory access commands, and the LPDDR3 SDRAM Controller IP Core from Lattice. The memory is 32-bit wide and is accessed with a 300 MHz DDR clock. This means that on each rising edge of the 150 MHz LPDDR system clock, up to 128 bits of data are available.

For more information about the LPDDR3 SDRAM Controller IP Core and DDR timing, refer to [IPUG110] and [TN1265].

3 Design Information

3.1 Source Code

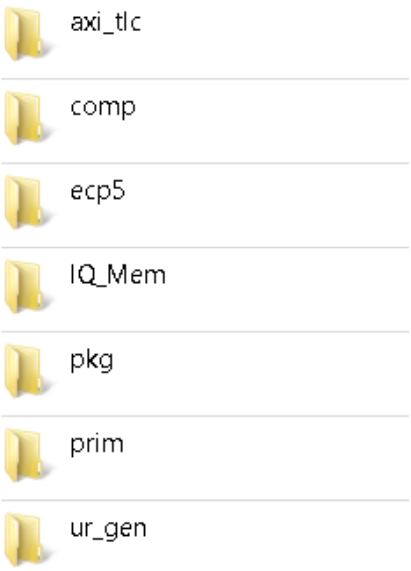

Figure 4 RTL Source Code Structure

The ‘ecp5’ folder contains the top level Verilog file.

The ‘axi_tlc’ folder contains Verilog design files for the AXI TLC.

The ‘ur_gen’ folder contains the Unsupported Request generator logic.

The ‘IQ_Mem’ folder contains the Mikroprojekt’s AXI LPDDR memory controller written in VHDL.

The ‘pkg’, ‘comp’ and ‘prim’ folder include VHDL packages used by the AXI LPDDR memory controller.

3.2 IP versions

IP Name	IP Version
LPDDR3 SDRAM Controller	1.0
PCI Express Endpoint Core	6.1
Dual Clock FIFO	5.8
PLL	5.6
EXTREF	1.1

Table 2 PCI Express Video Demo IP Core versions

3.3 Resource Utilization

Slices	LUTs	Registers	EBRs
11166	17339	11167	21

Table 3 Video Demo Resource Utilization on ECP5

4 PLL Configuration

The name of the PLL configuration design is RD00019_ecp5com_si5338.

The design demonstrates how to generate the mico32_clock.ngo used in the PCIe video demo. This Mico 32 processor is generated with a separate project because of tool limitations that make it impossible to instantiate multiple Mico 32 processors in a single design. Figure 5 gives the Wishbone system configuration and addresses.

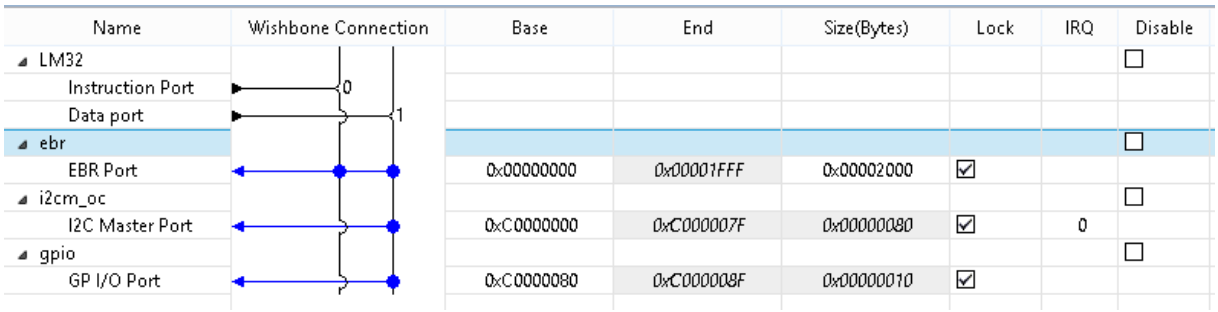

Figure 5 PLL Configuration Wishbone configuration snapshot

The I²C master is used for PLL configuration. The example Si5338 configuration program for PCIe is in the mico32_sw/pcie_test folder.

The GPIO is used for LED indication that the PLL has been successfully configured. In this design LED10 and LED11 are connected to GPIO pins 0 and 1.

In the strategy for the Diamond project the Disable I/O insertion option needs to be set to True, so that the top-level module of this project can be inserted into other designs (`mico32/soc/mico32.v`).

To generate the desired netlists it is enough to only run the synthesis and translate processes in Diamond.

5 References

Lattice Semiconductors:

- *DS1044 ECP5 familiy datasheet*
- *TN1261 ECP5 SERDES/PCS Usage guide*
- *TN1263 ECP5 sysCLOCK PLL/DLL Design and Usage Guide*
- *TN1265 ECP5 High-Speed I/O Interface*
- *Lattice Diamond Tutorial 3.5*
- *Diamond 3.5 User Guide*
- *LatticeMico32 Tutorial*
- *LatticeMico32 Hardware Developer User Guide*
- *Clarity Designer User Manual*
- *FPGA Libraries Reference Guide*
- *IPUG112 PCI Express x1/x2/x4 Endpoint IP Core User's Guide*
- *UG15 PCI Express Basic Demo Verilog Source Code User's Guide*
- *TN1271 PCIe Sample Demo Debugging and Packet Analysis Guide*
- *IPUG110 LPDDR3 SDRAM Controller IP Core User's Guide*

Freescale:

- *IMX6SDLRM i.MX 6Solo/6DualLite Applications Processor Reference Manual*

PCI-SIG:

- *PCI Express Base Specification Revision 3.0*

6 Ordering Information

Please contact us via email **contact@mikroprojekt.hr** about item availability and ordering details.

7 Technical Support Assistance

Basic technical product support is free of charge and available via e-mail to all Mikroprojekt customers, whether they are evaluating or have purchased a Mikroprojekt product.

Basic technical support can be requested by sending an e-mail to **support@mikroprojekt.hr**

Our engineers will reply to your request within 2 working days.

Additionally, Mikroprojekt offers to its customers a premium support package, allowing them to be directly supported by Mikroprojekt engineers. The premium support package consists of 10 hours of live, online support, including:

- Phone Support
- Instant Messaging Support
- TeamViewer VNC Interventions.

Furthermore, the customers who purchase our premium support package benefit from direct design assistance of our engineers.

Contact **sales@mikroprojekt.hr** to order the premium support package.

Terms of use

The reproduction, transmission or use of this document or its contents is not permitted without express written authority. Offenders shall be liable for damages.

All rights, including rights created by patent grant or registration of a utility model or design, are reserved.

Technical data is subject to change at any time.

Contact info