department of
signal processing

http://sp.utia.cz

Application Note

ÚTIA Akademie věd České republiky
Ústav teorie informace a automatizace AV ČR, v.v.i.

e•MMC AXIS Controller Interfacing MTFC32GJWDQ-4M 32 GB Memory on Xilinx KC705 Board

Lukáš Kohout, Department of Signal Processing, UTIA AV ČR, v.v.i.

kohoutl@utia.cas.cz

Revision history

Rev.	Date	Author	Description
0	27.04.2015	L.Kohout	Initial Release
1	22.06.2015	L.Kohout	Section 3.2 / Figure 10. Section 6 performance update according to faster memories.
2			

Content

Acknowledgement

This work has been supported from project IDEAS, project number ENIAC JU 304603 and MSMT 7H12004.

1 Introduction

This application note describes an interfacing of e•MMC 32 GB non-volatile memory MTFC32GJWDQ-4M [1] by e•MMC AXIS controller on Xilinx KC705 FPGA board [2]. The controller is designed to supports MMC4.51 standard [3] on Xilinx FPGAs and SOCs of family 7 (Artix-7, Kintex-7, Virtex-7, Zynq-7000, etc). An application demonstrates writing and reading video data to the e•MMC memory.

2 Description

Figure 1 shows a simplified block diagram of the application. Video signal comes from image sensor (Vita 2000 [4]), signal parameters are 1920x1080p60 coded in color Bayer matrix. This signal is transformed to YCrCb 4:4:4 color space. To get monochrome signal (gray scale) only luminance component Y is stored to the input frame buffer, in the text this frame buffer will be marked as FB_IN. To display signal like this, it has to be complemented with a chrominance component (constant 128); output HDMI chip on ZC702 board uses YCrCb 4:2:2 color space.

At e•MMC write operation, FB_IN is read. Write speed varies with the e•MMC memory capability, write scenario is *write the data as fast as possible* without any respect to the real frame rate. Evaluation of the writing speed is described in Section 6.2.

e•MMC read operation invokes reading memories content. This data are complemented with SOF (Start of Frame) and EOL (End of Line) markers, and these video data are stored in the output frame buffer (FB_OUT). Read scenario is the same as the write one, *read the data as fast as possible* without any respect to the real frame rate. Evaluation of the e•MMC reading speed is described in Section 6.1 in more detail.

Video data stored in FB_OUT are directly outputs by HDMI ipcore and can be viewed on the monitor screen. All data paths of the video signal are shown in Figure 2. It should be noted that e•MMC memories cannot provide a full-duplex access, so the demonstrator does not perform writing and reading operation at the same time. The only way how to see the data that are currently being written to the memory, is to redirect FB_OUT input side to the FB_IN output side.

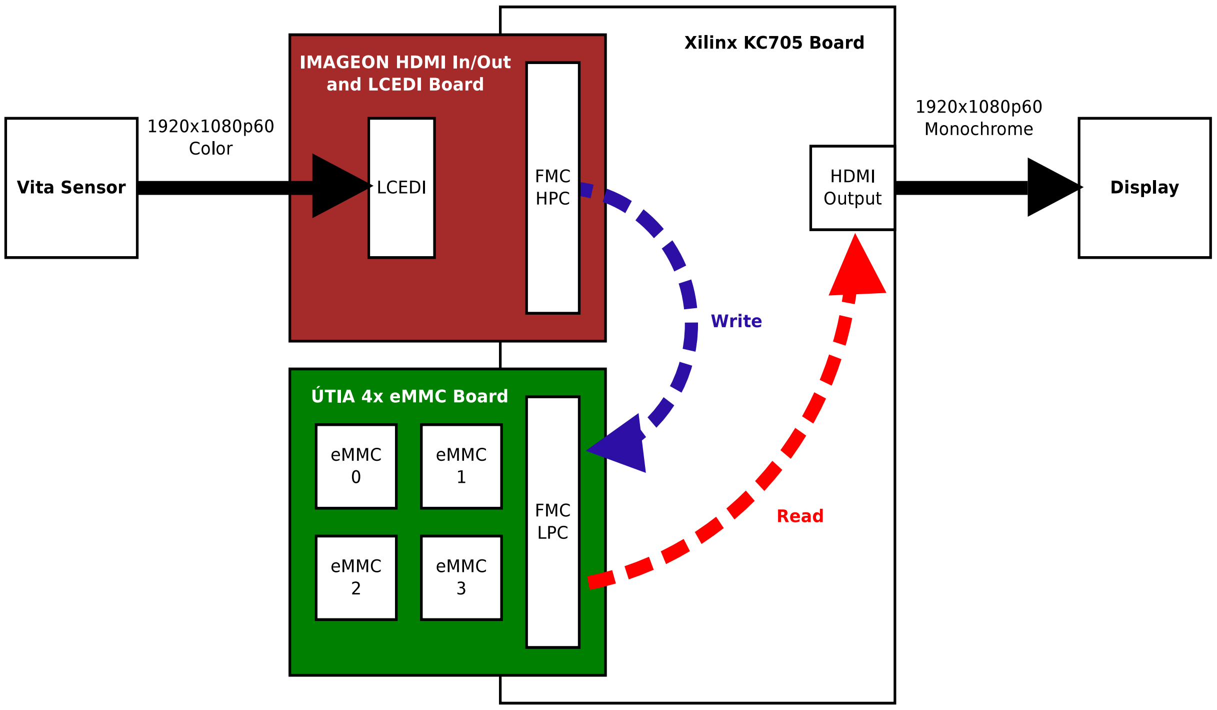


Figure 1: Demonstrator block diagram.

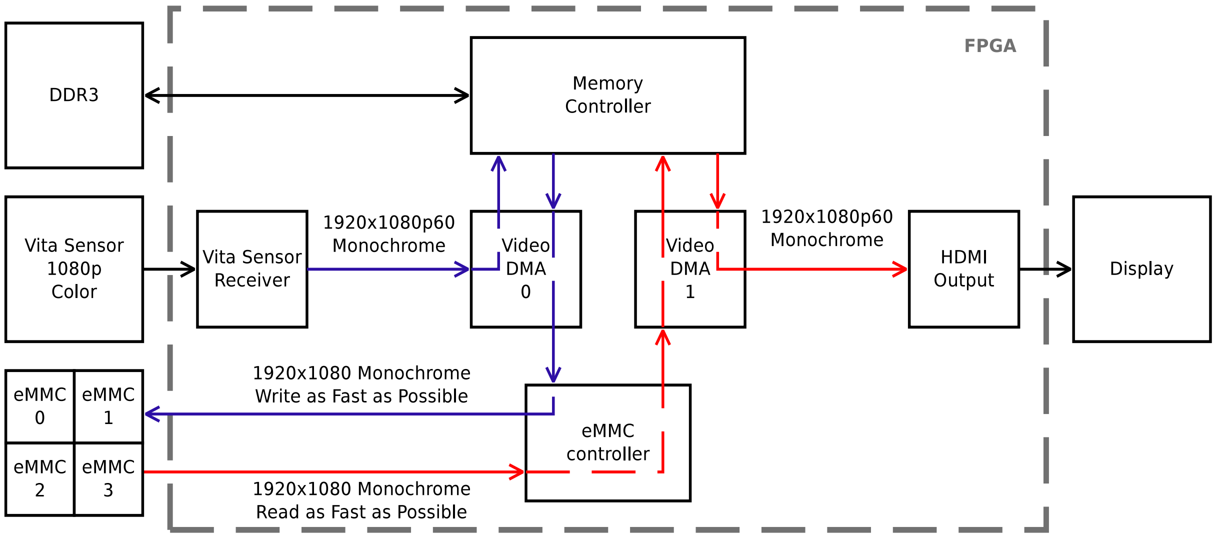


Figure 2: Demonstrator video data paths.

3 Implementation

This application is implemented on Xilinx KC705 board [2] with two additional boards. The first one is AVNET IMAGEON video I/O board [5], it provides an input camera interface. The second board is ÚTIA 4x e•MMC expansion card, the board was developed in ÚTIA and it is FMC module interfacing four MTFC32GJWDQ-4M [1] modules (Figure 3, Appendix A). Whole demonstrator set can be seen in Figure 4. The FPGA side of the application is based on common Vivado 2013.4 ipcore set, ipcores provided with the IMAGEON board and e•MMC AXIS controller developed in ÚTIA. The e•MMC controller is described in Section 3.1 in more details.

3.1 e•MMC AXIS Controller

This section describes an e•MMC AXIS controller. The controller is designed to support MMC 4.51 standard on Xilinx FPGAs and SOCs of family 7 (Artix-7, Kintex-7, Virtex-7, Zynq-

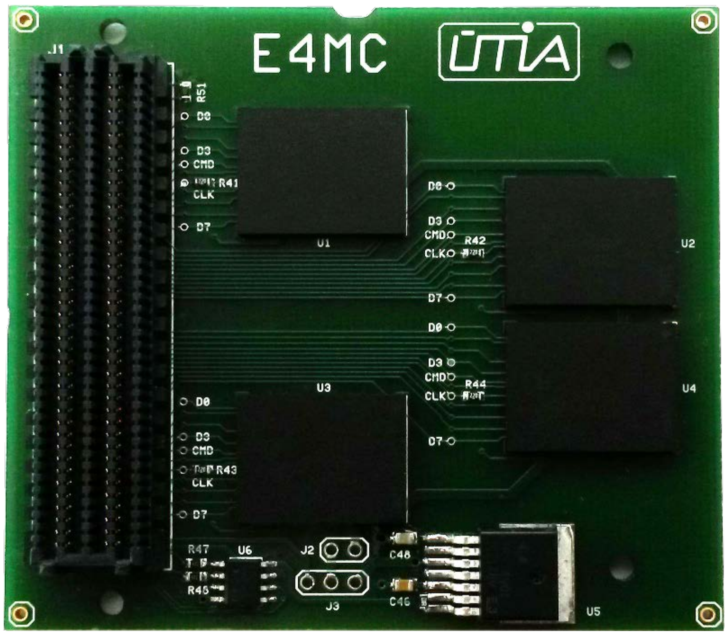


Figure 3: ÚTIA 4x eMMC board.

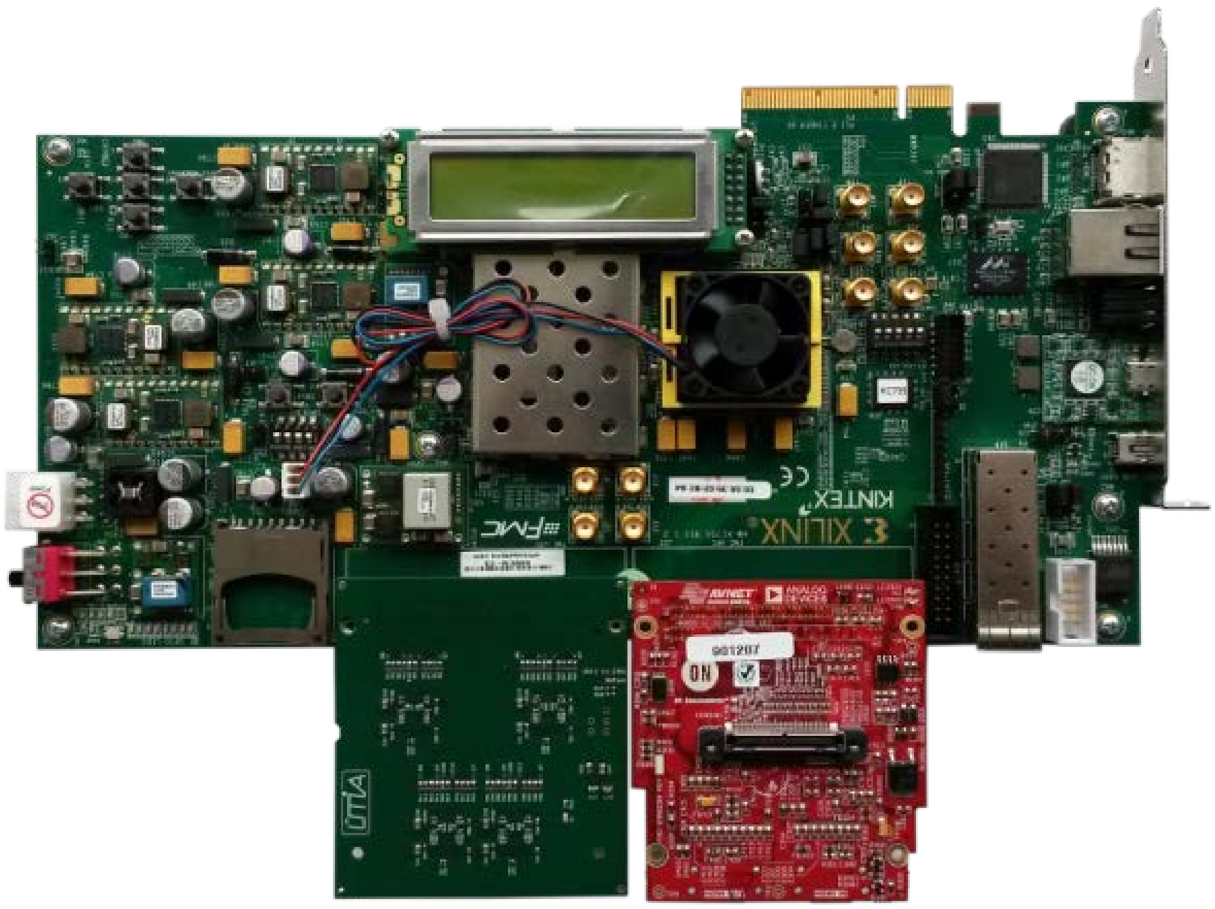


Figure 4: Set of Xilinx KC705 board, IMAGEON video I/O board and 4x e•MMC Board.

7000, etc). MMC-specific features are in the list below:

- JEDEC/MMC standard version 4.51-compliant (JEDEC Standard No. 84-B451) [1].
- Advanced 11-signal interface.
- 1b, 4b, 8b selectable data bus width.
- SDR mode up to 52 MHz lock speed.
- DDR mode is not supported.
- HS200 mode is not supported.
- SPI mode is not supported.

The controller is an ipcore designed with Xilinx Vivado 2013.4 tool to have AXI-Lite interface to control it and AXIS (AXI Stream) interface to transfer data. A simplified block diagram of the controller is shown in Figure 5. The core provides configurable number of e•MMC interfaces, 1 up to 4. Each e•MMC interface requires its own controller to be able run in parallel. The Inner structure of the controller bound to one e•MMC interface is shown in Figure 6.

AXIS interface is intended for video data transmitting, hence the data are supplemented by start of frame and end of line markers. The video data can be also accessed by AXI-Lite interface. In this case, only one e•MMC interface can be accessed at the time.

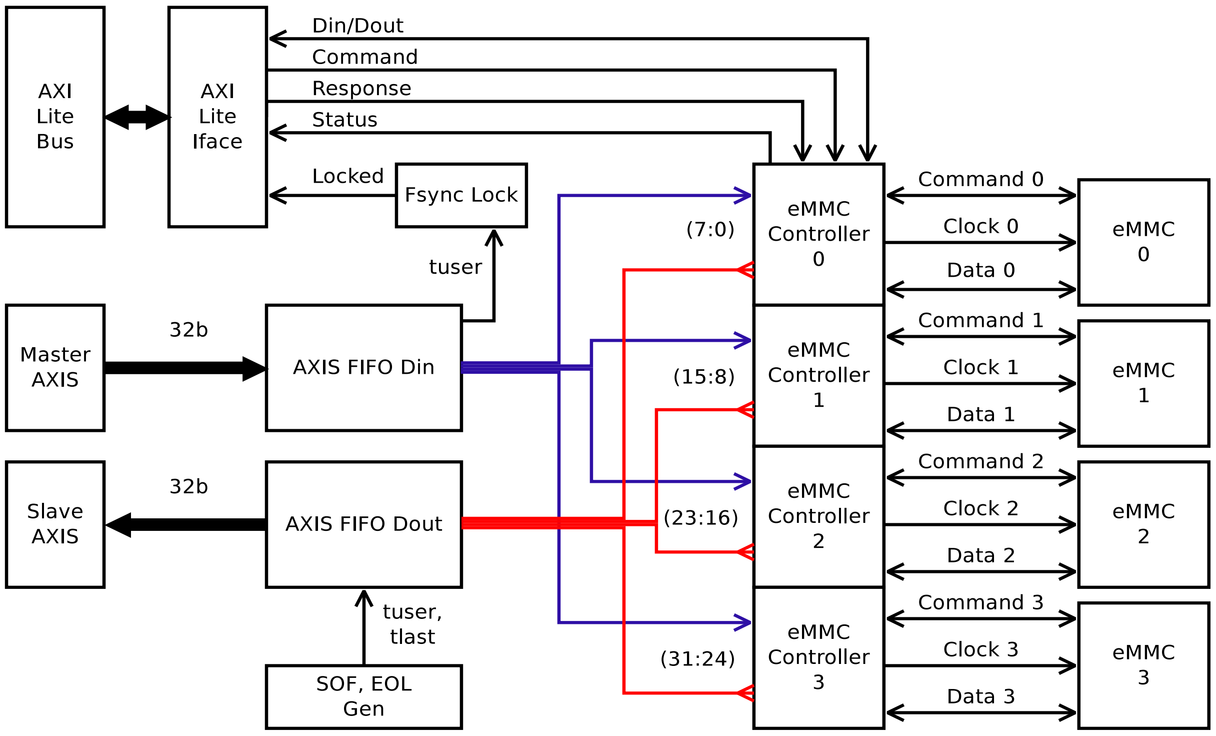


Figure 5: Block diagram of the e•MMC AXIS controller.

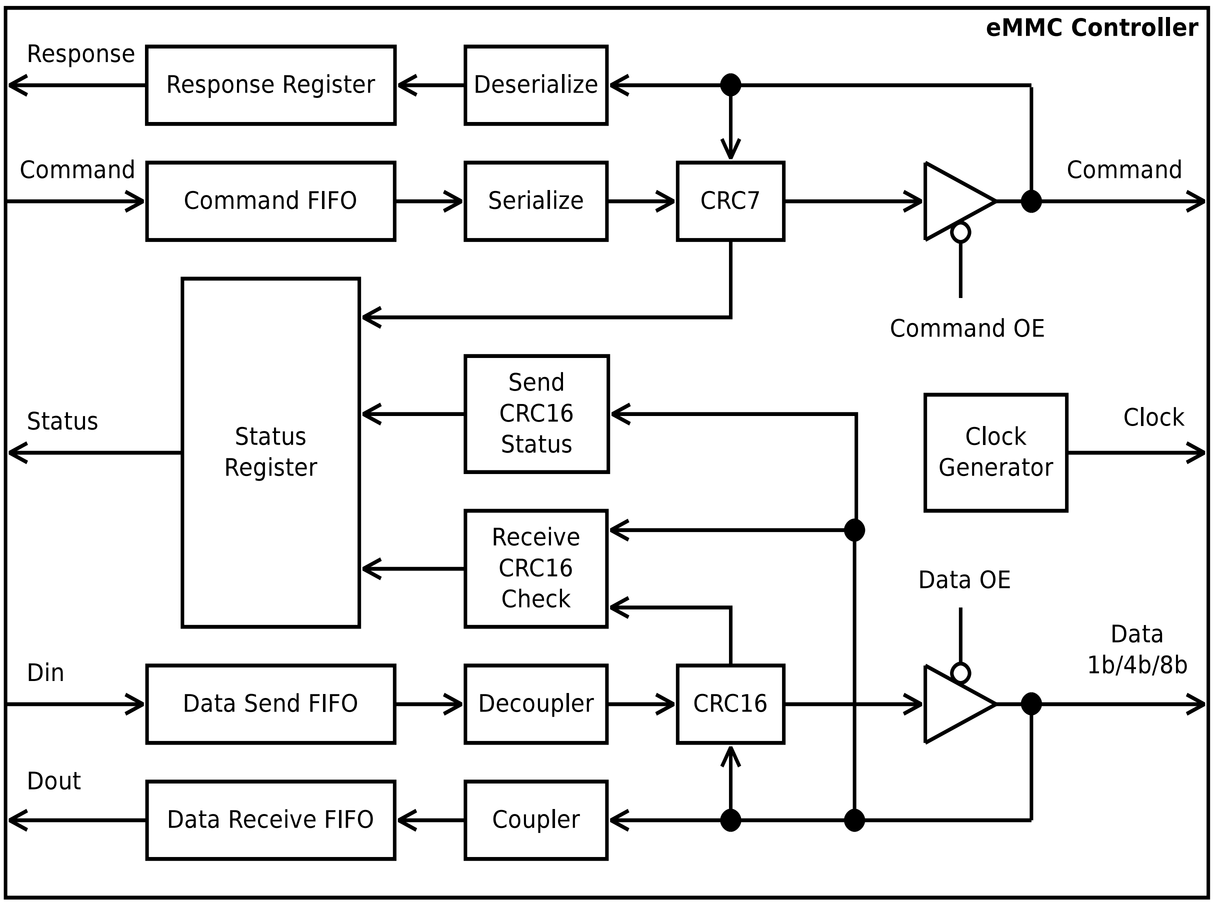


Figure 6: Inner structure of one e•MMC interface controller.

Table 1: e•MMC AXIS Controller Ports.

Port	Type	Direction	Width	Description
S_AXI_Lite	Bus			Slave AXI bus interface.
s_axi_aclk	std_logic	in	1	Slave AXI bus clock.
axi_aresetn	std_logic	in	1	Slave AXI bus reset, active low.
axis_aresetn	std_logic	in	1	AXI Stream reset, active low.
S_AXIS	AXI Stram IO			Slave AXI Stream interface.
s_axis_aclk	std_logic	in	1	Slave AXI Stream clock.
M_AXIS	AXI Stram IO			Master AXI Stream interface.
m_axis_aclk	std_logic	in	1	Master AXI Stream clock.
clk	std_logic	in	1	e•MMC interface fast clock.
ioclk_0	std_logic	out	1	e•MMC_0 output clock.
cmd_io_0	std_logic	inout	1	e•MMC_0 command line.
d_io_0	std_logic_vector	inout	1/4/8	e•MMC_0 data bus.
rstn_0	std_logic	out	1	e•MMC_0 output reset, active low.
ioclk_1	std_logic	out	1	e•MMC_1 output clock.
cmd_io_1	std_logic	inout	1	e•MMC_1 command line.
d_io_1	std_logic_vector	inout	1/4/8	e•MMC_1 data bus.
rstn_1	std_logic	out	1	e•MMC_1 oput put reset, active low.
ioclk_2	std_logic	out	1	e•MMC_2 output clock.
cmd_io_2	std_logic	inout	1	e•MMC_2 command line.
d_io_2	std_logic_vector	inout	1/4/8	e•MMC_2 data bus.
rstn_2	std_logic	out	1	e•MMC_2 output reset, active low.
ioclk_3	std_logic	out	1	e•MMC_3 output clock.
cmd_io_3	std_logic	inout	1	e•MMC_3 command line.
d_io_3	std_logic_vector	inout	1/4/8	e•MMC_3 data bus.
rstn_3	std_logic	out	1	e•MMC_3 output reset, active low.

3.1.1 Ports and Generics

Table 1 summarizes e•MMC AXIS controller ports, in Table 2 there are described configurable parameters (generics) of the controller.

3.1.2 Registers

e•MMC AXIS controller is driven by SW accessible registers, the address map of them is in Table 3. Tables 4 – 24 describe the meaning of each register item.

Table 2: eMMC AXIS Controller Generics.

Generic	Type	Values	Description
C_FAMILY	string	auto	FPGA Family. Autogenerated from project properties.
C_ACLK_FREQ	natural	auto	AXI-Lite interface clock frequency in Hz, it is send FIFO and receive FIFO clock. Autogenerated from s00_axi_aclk port.
C_FAST_CLK_FREQ	natural	auto	Fast clock frequency in Hz of emmc interface. Autogenerated from clk port.
C_SLOW_CLK_FREQ	natural	100 - 400 kHz	Slow clock frequency in Hz of emmc interface.
C_EMMC_IO_CNT	natural	1 - 4	Number of emmc interfaces.
C_DIO_WIDTH	natural	1/4/8	emmc data port width.
C_S_AXI_DATA_WIDTH	integer	32	AXI-Lite data width.
C_S_AXI_ADDR_WIDTH	integer	6	AXI-Lite address width.
C_S_AXIS_TDATA_WIDTH	integer	32/24/16/8	Slave AXI Stream data width.
C_M_AXIS_TDATA_WIDTH	integer	32/24/16/8	Master AXI Stream data width.

Table 3: e•MMC AXIS Controller Register Summary.

Register Name	Address	Width	Type	ResetValue	Description
CMD_Reg	0x00000000	8	rw	0x00	Command register.
CTRL1_Reg	0x00000001	8	rw	0x00	First control register.
CTRL2_Reg	0x00000002	16	rw	0x0000	Second control register.
ARG_Reg	0x00000004	32	rw	0x00000000	Commanda rgument register.
RESP0_Reg	0x00000008	32	rw	0x00000000	Response_0 register.
RESP1_Reg	0x0000000C	32	rw	0x00000000	Response_1 register.
RESP2_Reg	0x00000010	32	rw	0x00000000	Response_2 register.
RESP3_Reg	0x00000014	32	rw	0x00000000	Response_3 register.
BLK_SIZE_Reg	0x00000018	16	rw	0x0000	Block size register.
BLK_CNT_Reg	0x0000001A	16	rw	0x0000	Block count register.
D_SEND_0_Reg	0x0000001C	8	rw	0x00	e•MMC_0 data to send register.
D_SEND_1_Reg	0x0000001D	8	rw	0x00	e•MMC_1 data to send register.
D_SEND_2_Reg	0x0000001E	8	rw	0x00	e•MMC_2 data to send register.
D_SEND_3_Reg	0x0000001F	8	rw	0x00	e•MMC_3 data to send register.
D_RECV_0_Reg	0x00000020	8	ro	0x00	e•MMC_0 received data register.
D_RECV_1_Reg	0x00000021	8	ro	0x00	e•MMC_1 received data register.
D_RECV_2_Reg	0x00000022	8	ro	0x00	e•MMC_2 received data register.
D_RECV_3_Reg	0x00000024	8	ro	0x00	e•MMC_3 received data register.
STATUS_Reg	0x00000024	32	ro	0x0000C403	Status register.
CTRL3_Reg	0x00000028	32	rw	0x00000000	Third control register.

Table 4: Command Register Details.

Field Name	Bits	Type	Reset Value	Description
Start_Bit	7	rw	0b0	Command start bit, always '0'.
Direction	6	rw	0b0	Direction on command line, '1' = send command, '0' receive response. Set always to '1'.
CMD_ID	5:0	rw	0b000000	Command number.

Table 5: Control Register 1 Details.

Field Name	Bits	Type	Reset Value	Description
Host_Rst	7	rw	0b0	Reset the controller, active high.
Device_Rst	6	rw	0b0	Reset emmc device, active high.
Mode_Bus	5:4	rw	0b00	Set bus width, "00" = 1b, "01" = 4b, "10" = 8b, "11" = not defined.
Mode_Data	3:2	rw	0b00	Set data direction, "00" = no data, "01" = receive the data, "10" = send the data, "11" = not defined.
Mode_Speed	1:0	rw	0b00	Set output clock parameters. "00" = low speed (up to 400 kHz), strobe the data on the falling edge. "01" = normal speed (up to 52 MHz), strobe the data on the falling edge. "10" = high speed, strobe the data on the rising edge. "11" = not defined.

Table 6: Control Register 2 Details.

Field Name	Bits	Type	Reset Value	Description
Reserved	15			
CRC_Clr	14	rw	0b0	Clear CRC error statuses (send and receive).
CMD12_Offset	13:1	rw	0x000	CMD12 starting point within the last block of the multiple block data transfer. It is an offset from the block beginning in bytes.
CMD12_auto	0	rw	0b0	Setting to '1' enables CMD12 auto-generation when the last block of the multiple block data transfer is active.

Table 7: Command Argument Register Details.

Field Name	Bits	Type	Reset Value	Description
Arg	31:0	rw	0x00000000	Command argument.

Table 8: Command Response_0 Register Details.

Field Name	Bits	Type	Reset Value	Description
Resp0	31:0	rw	0x00000000	Part of the command response, bytes 3:0.

Table 9: Command Response_1 Register Details.

Field Name	Bits	Type	Reset Value	Description
Resp1	31:0	rw	0x00000000	Part of the command response, bytes 7:4.

Table 10: Command Response_2 Register Details.

Field Name	Bits	Type	Reset Value	Description
Resp2	31:0	rw	0x00000000	Part of the command response, bytes 11:8.

Table 11: Command Response_3 Register Details.

Field Name	Bits	Type	Reset Value	Description
Resp3	31:0	rw	0x00000000	Part of the command response, bytes 15:12.

Table 12: Block Size Register Details.

Field Name	Bits	Type	Reset Value	Description
Reserved	15:13			
Blk_Size	12:0	rw	0b0000000000000	Block size in bytes, for the minimal and maximal block sizes see Table 14. Be aware of block size alignment to 4B. Keep the register value unchanged during the data transfer

Table 13: Block Count Register Details.

Field Name	Bits	Type	Reset Value	Description
Blk_Cnt	15:0	rw	0x0000	Number of blocks within a multiple block data transaction. 0x0000 = no data transfer, 0x0001 = 1 block, 0x0002 = 2 blocks, . 0xFFFF = 65535 blocks. Keep the register value unchanged during the data transfer.

Table 14: Block Size Constraints.

	Send Single Block CMD24	Send Multiple Block CMD25	Receive Single Block CMD17	Receive Multiple Block CMD18
Min	12 B	40 B	12 B	64 B
Max	4084 B	4084 B	4084 B	4084 B

Table 15: e•MMC_0 Data to Send Register Details.

Field Name	Bits	Type	Reset Value	Description
D_Send_0	7:0	rw	0x00	e•MMC_0 data to send, writing this register feeds the send data FIFO of the e•MMC_0 controller. Before writing ths register select corresponding controller (Ctrl_Sel parameter) and check the FIFO is not full (D_Send_Rdy parameter) or it is empty enough to write whole block into (D_Send_Blk_Rdy parameter).

Table 16: e•MMC_1 Data to Send Register Details.

Field Name	Bits	Type	Reset Value	Description
D_Send_1	7:0	rw	0x00	e•MMC_1 data to send, writing this register feeds the send data FIFO of the e•MMC_1 controller. Before writing this register select corresponding controller (Ctrl_Sel parameter) and check the FIFO is not full (D_Send_Rdy parameter) or it is empty enough to write whole block into (D_Send_Blk_Rdy parameter).

Table 17: e•MMC_2 Data to Send Register Details.

Field Name	Bits	Type	Reset Value	Description
D_Send_2	7:0	rw	0x00	e•MMC_2 data to send, writing this register feeds the send data FIFO of the e•MMC_2 controller. Before writing this register select corresponding controller (Ctrl_Sel parameter) and check the FIFO is not full (D_Send_Rdy parameter) or it is empty enough to write whole block into (D_Send_Blk_Rdy parameter).

Table 18: e•MMC_3 Data to Send Register Details.

Field Name	Bits	Type	Reset Value	Description
D_Send_3	7:0	rw	0x00	e•MMC_3 data to send, writing this register feeds the send data FIFO of the e•MMC_3 controller. Before writing this register select corresponding controller (Ctrl_Sel parameter) and check the FIFO is not full (D_Send_Rdy parameter) or it is empty enough to write whole block into (D_Send_Blk_Rdy parameter).

Table 19: e•MMC_0 Received Data Register Details.

Field Name	Bits	Type	Reset Value	Description
D_Recv_0	7:0	ro	0x00	e•MMC_0 received data, after reading this register the received data FIFO shifts the next entry out. Before reading the register select corresponding controller (Ctrl_Sel parameter) and check the FIFO is not empty (D_Recv_Valid parameter) or the FIFO contains whole data block (D_Recv_Valid_Blk parameter).

Table 20: e•MMC_1 Received Data Register Details.

Field Name	Bits	Type	Reset Value	Description
D_Recv_1	7:0	ro	0x00	e•MMC_1 received data, after reading this register the received data FIFO shifts the next entry out. Before reading the register select corresponding controller (Ctrl_Sel parameter) and check the FIFO is not empty (D_Recv_Valid parameter) or the FIFO contains whole data block (D_Recv_Valid_Blk parameter).

Table 21: e•MMC_2 Received Data Register Details.

Field Name	Bits	Type	Reset Value	Description
D_Recv_2	7:0	ro	0x00	e•MMC_2 received data, after reading this register the received data FIFO shifts the next entry out. Before reading the register select corresponding controller (Ctrl_Sel parameter) and check the FIFO is not empty (D_Recv_Valid parameter) or the FIFO contains whole data block (D_Recv_Valid_Blk parameter).

Table 22: e•MMC_3 Received Data Register Details.

Field Name	Bits	Type	Reset Value	Description
D_Recv_3	7:0	ro	0x00	e•MMC_3 received data, after reading this register the received data FIFO shifts the next entry out. Before reading the register select corresponding controller (Ctrl_Sel parameter) and check the FIFO is not empty (D_Recv_Valid parameter) or the FIFO contains whole data block (D_Recv_Valid_Blk parameter).

Table 23: Status Register Details.

Field Name	Bits	Type	Reset Value	Description
Reserved	31:18			
S_AXIS_Locked	17	ro	0b0	Data on slave AXIS interface are synchronized to start of frame. It indicates that the data written to memory always start with first pixel.
D_Recv_Valid_Blk	16	ro	0b0	Received data valid block, this flag indicates that the received data FIFO contains whole data block of size given by Blk_Size parameter.
D_Send_Blk_Rdy	15	ro	0b1	Send data block ready, this flag indicates that the send data FIFO is empty enough to write a block of data. Blk_Size parameter defines the size of the block.
D_Done	14	ro	0b1	Data transaction finished flag. It is active high when no data are send or receive by the core.
Recv_CRC_Err	13	ro	0b0	Data receive CRC error flag. It is asserted high when the received data CRC and the calculated CRC do not match.
Send_CRC_Err	12	ro	0b0	Data send CRC error. This flag is asserted high when the the e•MMC device returns a negative CRC status token ("101"). The positive CRC status token ("010") sets this flag to low. The CRC status token is returned after each block sending.
D_Recv_Valid	11	ro	0b0	Valid data in the received data FIFO. The FIFO is not empty.
D_Send_Rdy	10	ro	0b1	Send data FIFO can accept another data to write. The FIFO is not full.
CMD_Resp_CRC	9:3	ro	0b0000000	Command response CRC value. Calculated CRC on received command response.
CMD_Resp_Valid	2	ro	0b0	Valid response. This flag indicates the validity of the value stored in the response registers (0, 1, 2, and 3).
CMD_Rdy	1	ro	0b1	Ready to accept another command request. The command FIFO is not full.
CMD_Done	0	ro	0b1	Single command has been sent.

Table 24: Control Register 3 Details.

Field Name	Bits	Type	Reset Value	Description
Reserved	31:28			
AXIS_Lock_Rst	27	rw	0b0	Reset slave AXIS locked flag. The lock reset request invokes a resynchronization to next start of frame.
Ctrl_Sel	26:23	rw	0b0000	Select the controller to be used. A bit position in this configuration vector corresponds to selected e•MMC interface.
Res_Y	22:12	rw	0b00000000000	Set number of lines for SOF-EOL generator of the master AXIS interface.
Res_X	11:1	rw	0b00000000000	Set number of pixels for SOF-EOL generator of the master AXIS interface.
AXIS_En	0	rw	0b0	Enable AXIS interface.

3.2 System Overview

The demonstrator has been designed and implemented in Xilinx Vivado 2013.4 tool. System overview is shown in Figure 7. Figure 8 shows PROCESSING_SYSTEM hierarchical block

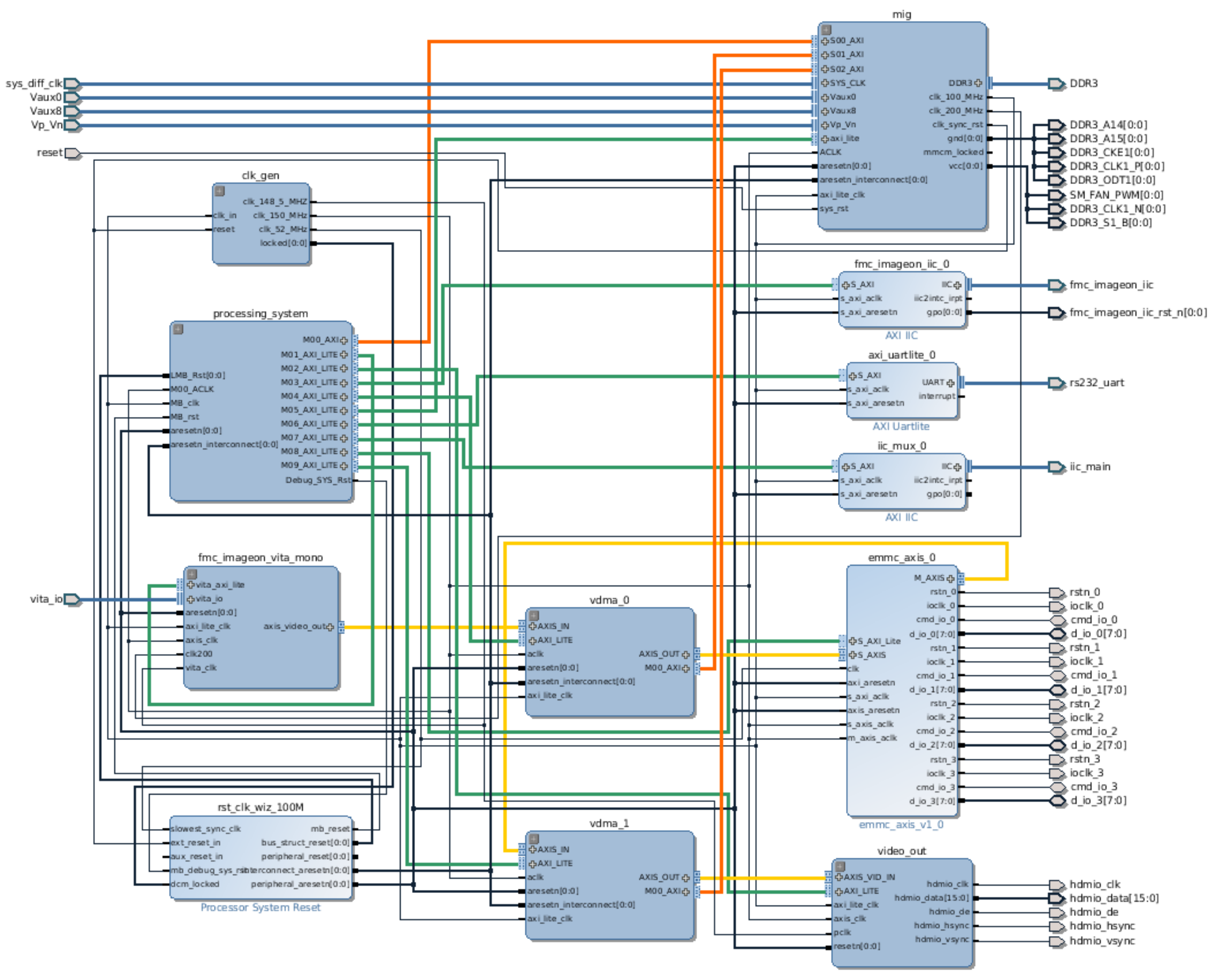


Figure 7: Vivado snapshot - system overview.

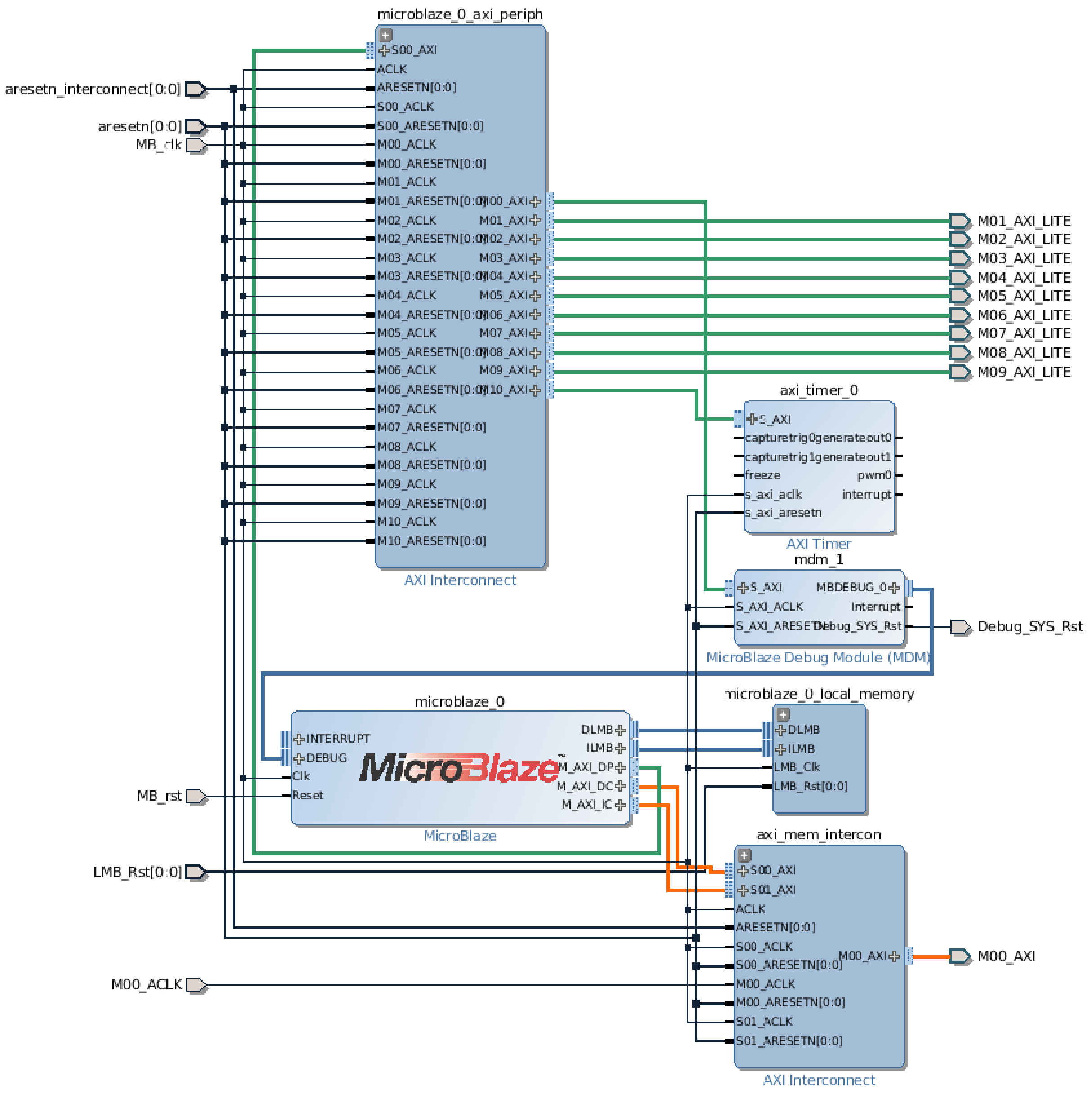


Figure 8: Vivado snapshot - Porcessing System block detail.

detail, it encapsulates MicroBlaze soft-core processor, processor local memory, debug module, peripheral interconnection, timer and processor cache interconnection. Clock frequency of the system is 100 MHz, but external MicroBlaze cache interconnection uses 150 MHz clock.

Figure 9 reveals surrounding blocks of the Memory Interconnection Generator (MIG). Apart from MIG, the main part of it is AXI interconnection that delimits clock domains between MIG inputs and all blocks requesting DDR3 memory access. The MIG input interface (AXI interconnection output) is 512b wide and it runs at 200 MHz. Rest of the system (AXI interconnection inputs) communicates at 150 MHz clock speed and data interfaces are 32b wide.

FMC_IMAGEON_VITA_MONO (Vita Receiver) hierarchical block structure is shown in Figure 10, it contains Vita Sensor Receiver, Video to AXIS Converter, Color Filter Array Interpolator CFA, RGB to YCrCb Converter and AXIS Subset Converter block. The receiver gets input video signal from the sensor coded in Bayer matrix, the signal is converted to RGB

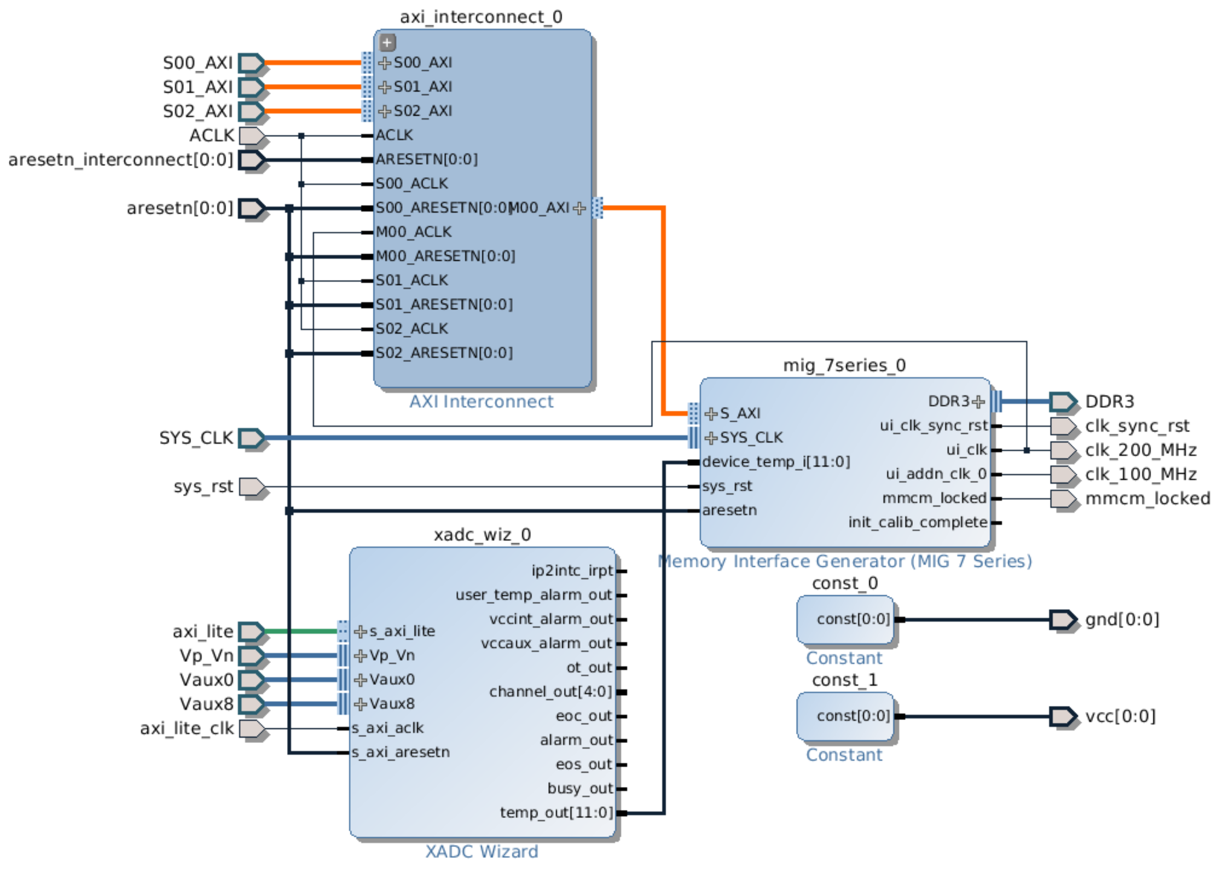


Figure 9: Vivado snapshot - Memory interface Generator (MIG) block detail.

with CFA block, then the signal is converted to YCrCb 4:4:4 color space. Input video signal parameters are 1920x1080p60 (pixel clock is 148.5 MHz). An output AXI Stream (AXIS) transfers only luma component Y to VDMA_0 block. The AXIS runs at 150 MHz.

Figure 11 shows Video DMA (VDMA_0 and VDMA_1) block detail, Video DMA controls frame buffer access, frame buffer(s) is located in main DDR3 memory. It uses 3 frame buffers with input/output interlock mechanism to avoid a tearing effect. Video data are synchronized with Start of Frame flag. AXIS interfaces run at 150 MHz clock speed as well as the memory interconnection. VDMA_0 Input AXIS port has 8b width (only luminance component Y is stored) but output AXIS port has 32b (4 pixels at the time, 1 pixel per e•MMC memory running in parallel). VDMA_1 block has this backwards, 32b at the input side and 8b on its output AXIS.

Video Output hierarchical block is shown in Figure 12. It contains Video Timing Controller (VTC), AXIS to Video Converter and HDMI Output interface ipcore. VTC generates output video data timing (HSYNC, VSYNC, DE) with respect to output image parameters, 1920x1080p60. AXIS to Video Converter reads AXIS with video data from VDMA_1 and required timing from VTC, it synchronizes video data with the timing. HDMI Output block supplements video signal with chrominance component (constant 128) and outputs the signal to the external HDMI chip.

EMMC_AXIS ipcore provides an interfacing of e•MMC memories, it reads AXIS from VDMA_0 to write data to the e•MMC memory, and generates AXIS to the VDMA_1 to display video data read from the e•MMC memory. This controller is described in Section 3.1 in more details.

The system also contains supporting blocks needed to configure external chips (FMC_IMAGEON_IIC and IIC_MUX), or provide user interface (UART_LITE ipcore). There are also clock generator and reset generator.

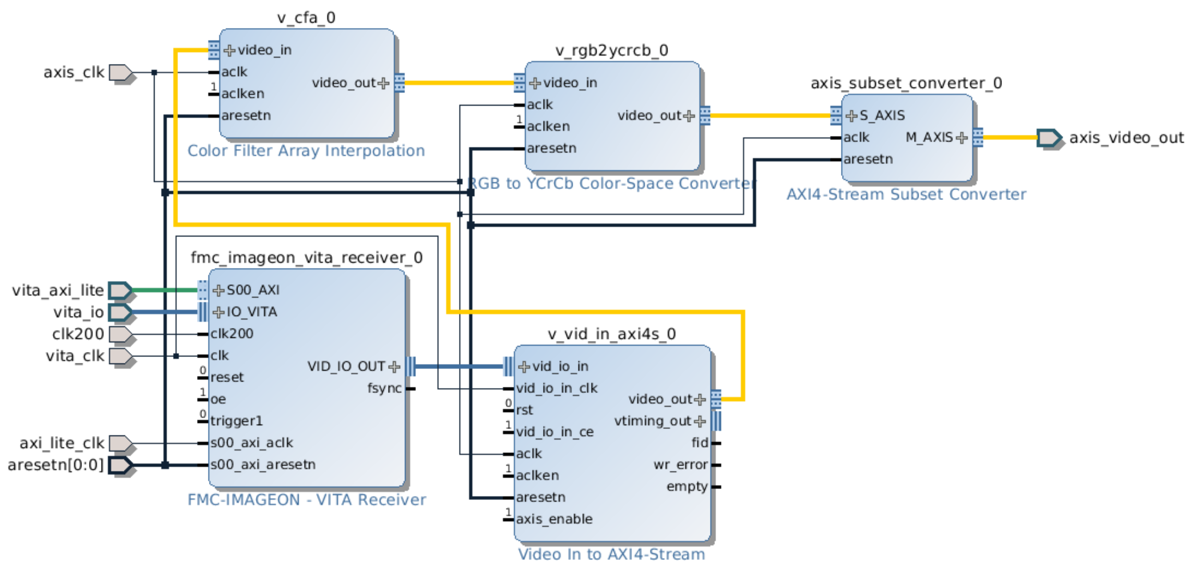


Figure 10: Vivado snapshot - Vita Receiver.

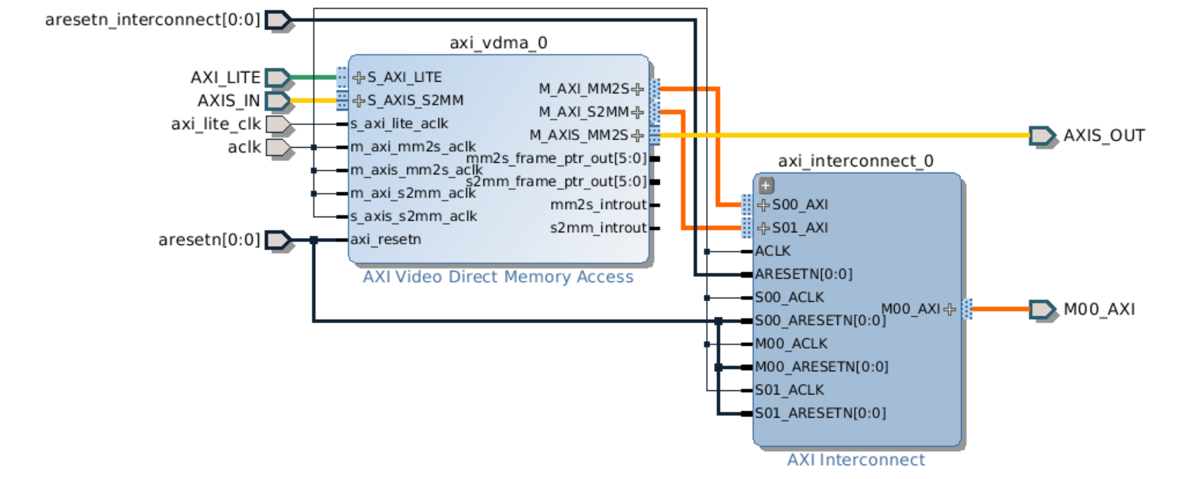


Figure 11: Vivado snapshot - VDMA_0 and VDMA_1 blocks detail.

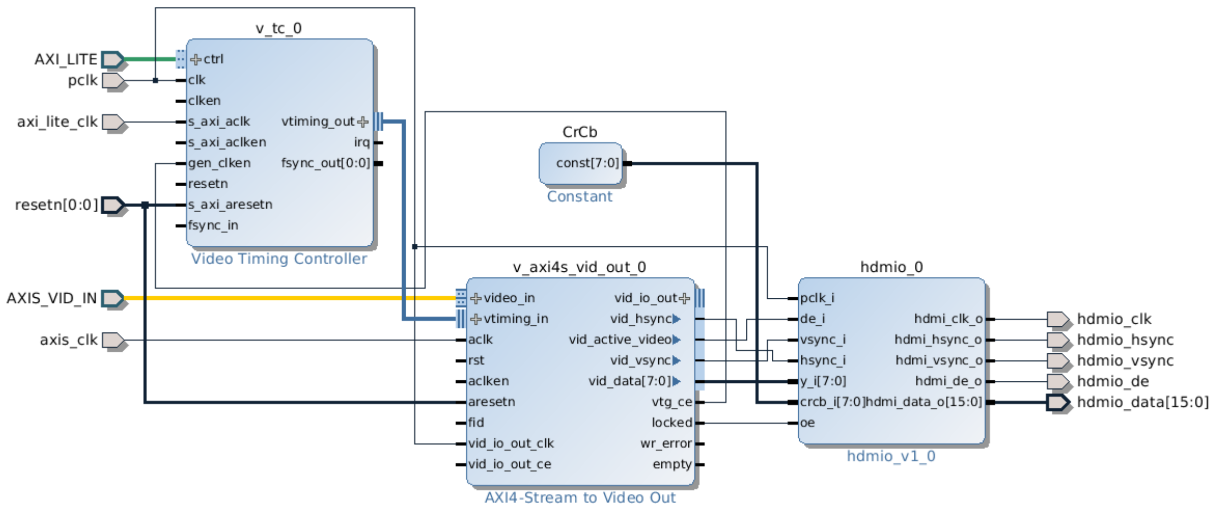


Figure 12: Vivado snapshot - Video Out block detail.

4 Initial e•MMC Setup

Initial setup for each e•MMC memory consists of steps enumerated in the list below:

1. Initialize e•MMC AXIS controller (host) and the memory (device).
2. Read OCR, CID and CSD registers from the memory.
3. Set bus width to 8b mode.
4. Set bus speed to 52 MHz.
5. Read EXT_CSD register.

Until the bus speed is switched to 52 MHz, the communication runs in slow mode using 300 kHz clock. Reading OCR, CID and CSD registers uses solely command line, but the reading ECSD register uses all of 8 data lines to obtain its content.

The structure of OCR, CID and CSD registers is listed in text below. Information decoded from these registers corresponds to structure presented in MMC System Specification version 3.31 [6]. As the MMC standard in version 3.31 does not specify ESCD register, its value is listed without the structure description. It should be noted that the OCR, CID and CSD structure could be slightly different compare to MMC 4.51 [3].

4.1 Operation Condition register OCR

Value:

```
C0FF8080

Power ON status:      1
2.7-3.6V:             0x1FF
2.0-2.6V:             0x00
1.65-1.95V:           1
```

4.2 CardID Register CID

Value:

```
0x00FE014E 4D4D4333 32471088 4458EA11

MID:                  0xFE
OID:                  0x014E
PNM:                  MMC32G
PRV:                  1.0
PSN:                  2554616042
MTD:                  2014, January
```

4.3 Card Specific Data Register CSD

Value:

```
00D06E01 320F5913 FFFFFFFF FF924000

CSD_STRUCTURE:       Reserved
SPEC_VERS:           Reserved
TAAC:                Time unit - 1 ms, Mult factor - 6.0
NSAC:                1 * 100 CC
TRAN_SPEED:          Freq unit - 10 Mb/s, Mult factor - 2.5
CCC:                 Supported command classes - 0 2 4 5 6 7
READ_BL_LEN:         512B
READ_BL_PARTIAL:     0
WRITE_BLK_MISALIGN:  0
READ_BLK_MISALIGN:   0
DSR_IMP:             1
C_SIZE:              4095
VDD_R_CURR_MIN:      100 mA
VDD_R_CURR_MAX:      200 mA
VDD_W_CURR_MIN:      100 mA
VDD_W_CURR_MAX:      200 mA
C_SIZE_MULT:         512
ERASE_GRP_SIZE:      31
ERASE_GRP_MULT:      31
WP_GRP_SIZE:         32 GB
WP_GRP_ENABLE:       1
DEFAULT_ECC:         0
R2W_FACTOR:          16
WRITE_BL_LEN:        512 B
WRITE_BL_PARTIAL:    0
CONTENT_PROT_APP:    0
FILE_FORMAT_GRP:     0
COPY:                0
PERM_WRITE_PROTECT:  0
FILE_FORMAT:         HDD-like FS with partition table
ECC:                 None (default)
```

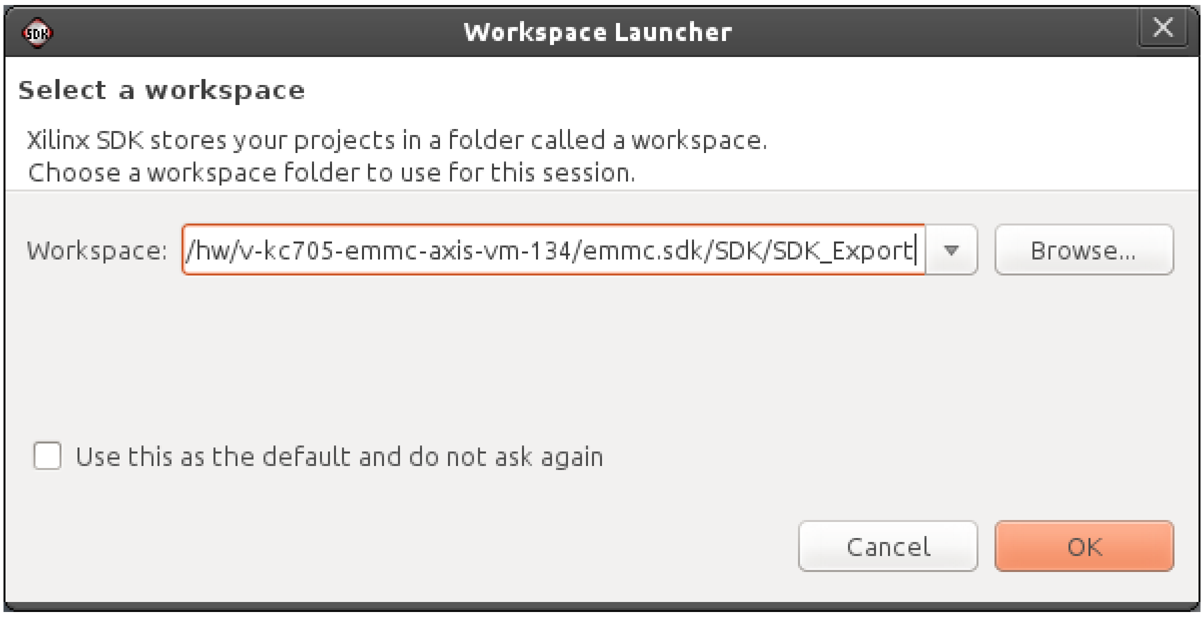
4.4 Extended Card Specific Data Register ECSD

Value (bytes 511 downto 0):

```
00 00 00 00 00 00 00 01 03 01 3F 3F 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 19 FF 00 00 00 00 00 32 00
04 09 09 00 00 00 00 0F 55 06 09 07 00 80 07 10
01 01 04 06 09 00 10 00 03 A2 00 00 00 08 08 08
08 08 08 00 02 02 05 05 01 04 0F 17 00 02 00 06
00 00 00 00 00 00 01 00 01 00 00 00 00 00 00 00
00 00 00 00 00 00 00 01 1F 05 00 00 00 00 00 03
00 01 D1 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

5 Quickstart

1. Plug ÚTIA 4x e•MMC board into FMC LPC connector on KC705 board.
2. Plug AVNET IMAGEON video I/O board into FMC HPC on KC705 board.
3. Connect Vita 2000 sensor with AVNET IMAGEON video I/O board, use LCEDI connector.
4. Connect JTAG cable.
5. Plug USB UART cable in. Serial terminal settings:
    - Baudrate: 115200
    - Data bits: 8
    - Stop bits: 1
    - No parity
    - No flow control


Workspace Launcher

Select a workspace

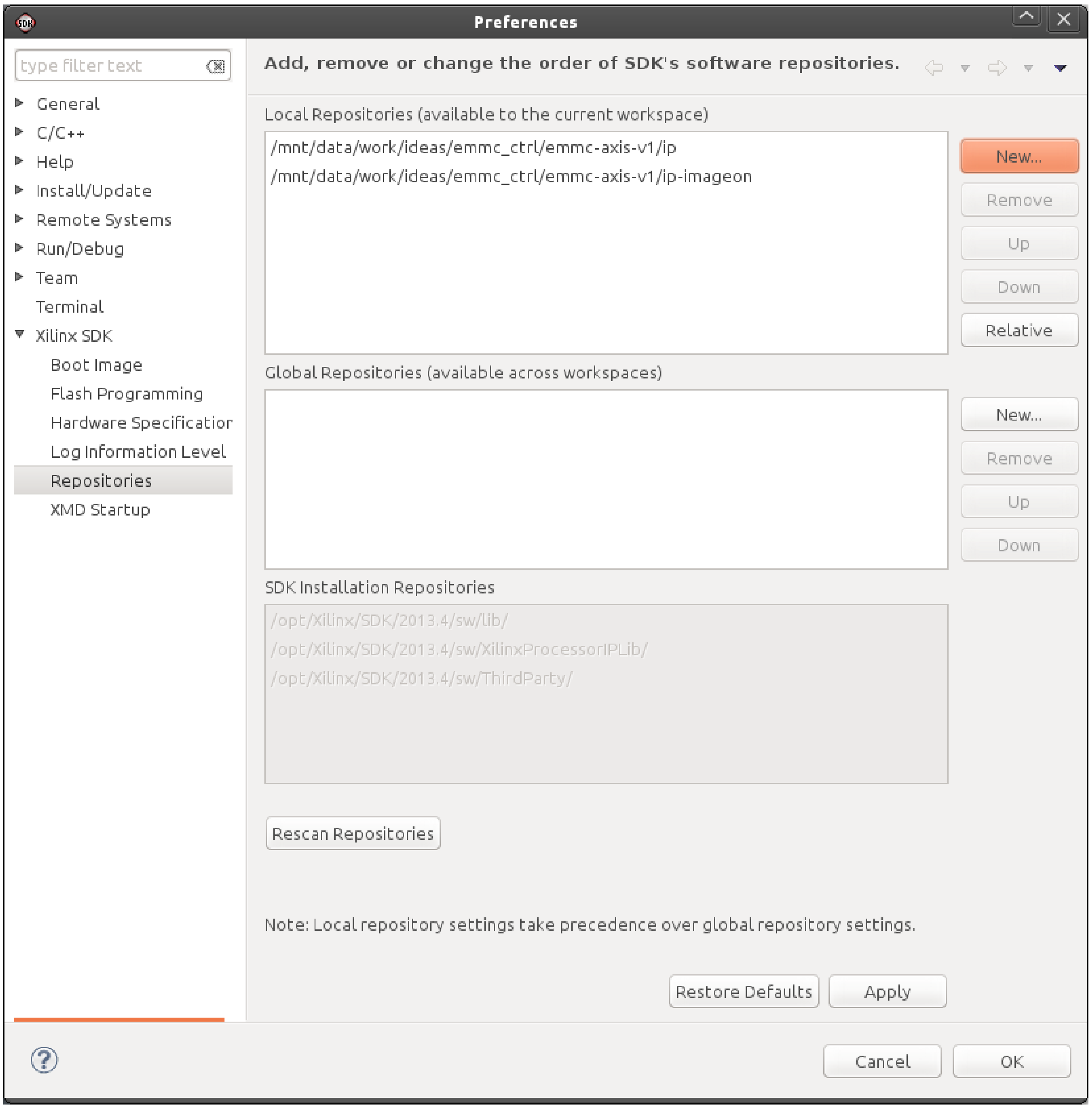
Xilinx SDK stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: /hw/v-kc705-emmc-axis-vm-134/emmc.sdk/SDK/SDK_Export

Figure 13: SDK workspace directory.

6. Connect LCD monitor, use HDMI connector on KC705 board, do not connect HDMI output on AVNET IMAGEON video I/O board. The monitor should be capable of showing 1920x1080p60 image.
7. Switch KC705 board on.
8. Unzip package *emmc-axis-v1.zip*. The package content is listed in Section 7.
9. Start Xilinx SDK 2013.4 tool.
10. Set SDK workspace directory to *emmc-axis-v1/hw/v-kc705-emmc-axis-vm-134/emmc.sdk/SDK/SDK_Export* (Figure 13).
11. Set paths to IP repositories, menu: *Xilinx Tools→Repositories.* If there any path definitions exist, remove them. Set new paths to repositories *emmc-axis-v1/ip* (e•MMC AXIS core) and *emmc-axis-v1/ip-imageon* (IMAGEON cores). It is recommended to use *ip-imageon* repository located in the package because it contains fixed version of the *Vita IP Core* (Figure 14).
12. Clean *emmc-vita-mono* project, menu: *Project→Clean→Clean all projects.* This recompiles the project.
13. Download bitstrem, menu: *Xilinx Tools→Program FPGA*.
14. Run compiled SW application, menu: *Run→Run.*
15. Observe the serial terminal.
16. Char ‘v’ starts writing video from the camera to the e•MMC momeries, char ‘V’ starts reading the memories. An application menu is listed below
    - C - Camera submenu
    - c - Select eMMC controller 0 to 3
    - v - Write video data via AXIS
    - V - Read video data via AXIS
    - s - Prepare random data to write
    - w - Write 1MB
    - r - Read 1MB and compare
    - W - Write 32MB - 512B

- R - Read 32MB - 512B and compare
- p - Print write buffer
- P - Print Read buffer
- 1 - Set 1b data bus
- 4 - Set 4b data bus
- 8 - Set 8b data bus
- S - Set low speed (300 kHz)
- F - Set high speed (50 MHz)
- m - Print this menu
- x - Quit


Preferences

Add, remove or change the order of SDK's software repositories.

Local Repositories (available to the current workspace)

/mnt/data/work/ideas/emmc_ctrl/emmc-axis-v1/ip
/mnt/data/work/ideas/emmc_ctrl/emmc-axis-v1/ip-imageon

Global Repositories (available across workspaces)

SDK Installation Repositories

/opt/Xilinx/SDK/2013.4/sw/lib/
/opt/Xilinx/SDK/2013.4/sw/XilinxProcessorIPLib/
/opt/Xilinx/SDK/2013.4/sw/ThirdParty/

Note: Local repository settings take precedence over global repository settings.

Figure 14: SDK IP repositories.

6 Performance

This section evaluates performance of four e•MMC MTFC32GJWDQ-4M memories running parallel in sequential read and write operations. e•MMC interfaces run at 52 MHz, data bus width is set to 8b. Figures in this section (oscilloscope snapshots) show only command line (CMD) and first data line (DATA0) of the e•MMC interface.

6.1 Read Operation

Figure 15 shows plot related to measurement of a time needed to read 31.999 MB chunk of video data from one e•MMC memory. The time is measured from the start bit of the multiple read block command CMD18 to stop bit of the stop transmission command CMD12. The time is 0.690 s, it means that the throughput of the sequential read is 46.4 MB/s. As the demonstrator uses four e•MMC memories, the maximal throughput of the read operation is 4 * 46.4 MB/s = 185 MB/s. This number corresponds to reading of 93 frames/s. One frame size is 1920 * 1080 = 2073600 B.

6.2 Write Operation

Figure 16 shows plot related to measurement of a time needed to write 31.999 MB chunk of video data. The time is measured from the start bit of the multiple write block command CMD25 to stop bit of the stop transmission command CMD12. The time is 0.780 s, it implies that the throughput of the sequential write is 41 MB/s. Nevertheless, MTFC32GJWDQ-4M memory does not provide stable throughput on writing operation. This behavior can be seen in Figure 17, the memory indicates its busy state by pulling DATA0 line down. In this case, the measured time is 1.91 s which corresponds to 16.75 MB/s.

In case the memory would not stall and using four memories, maximal achievable throughput is 4 * 41 MB/s = 164 MB/s. It corresponds to writing of 82 frames/s. The worst case of the throughput is 4 * 16.75 MB/s = 67 MB/s. This number means 33 frames/s.

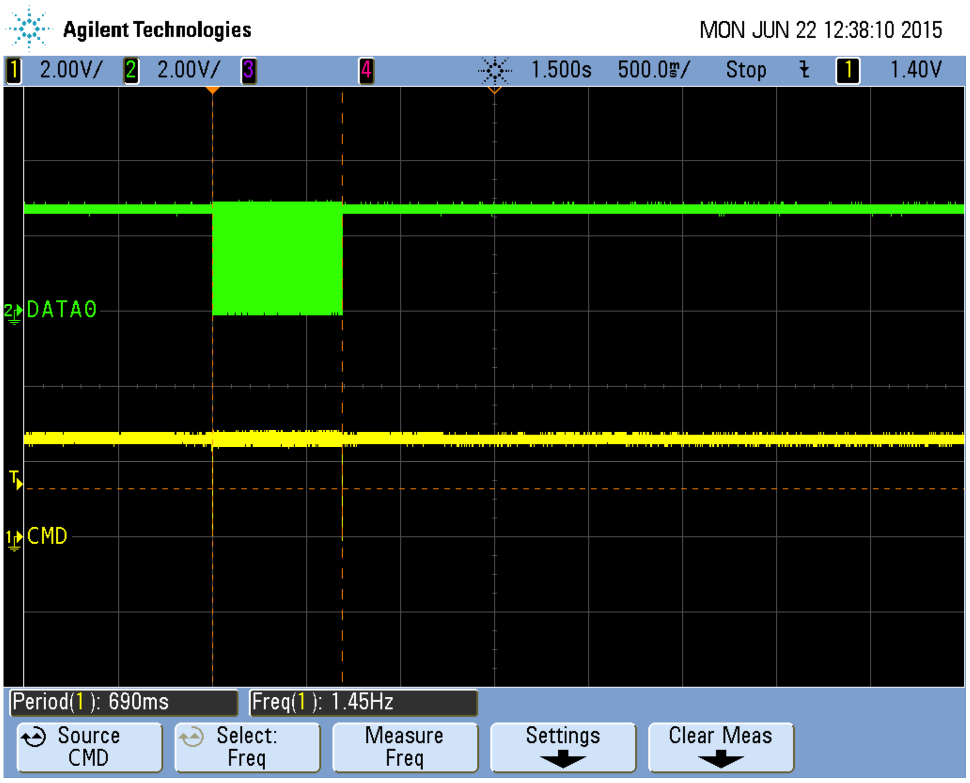


Figure 15: Oscilloscope snapshot - read operation.

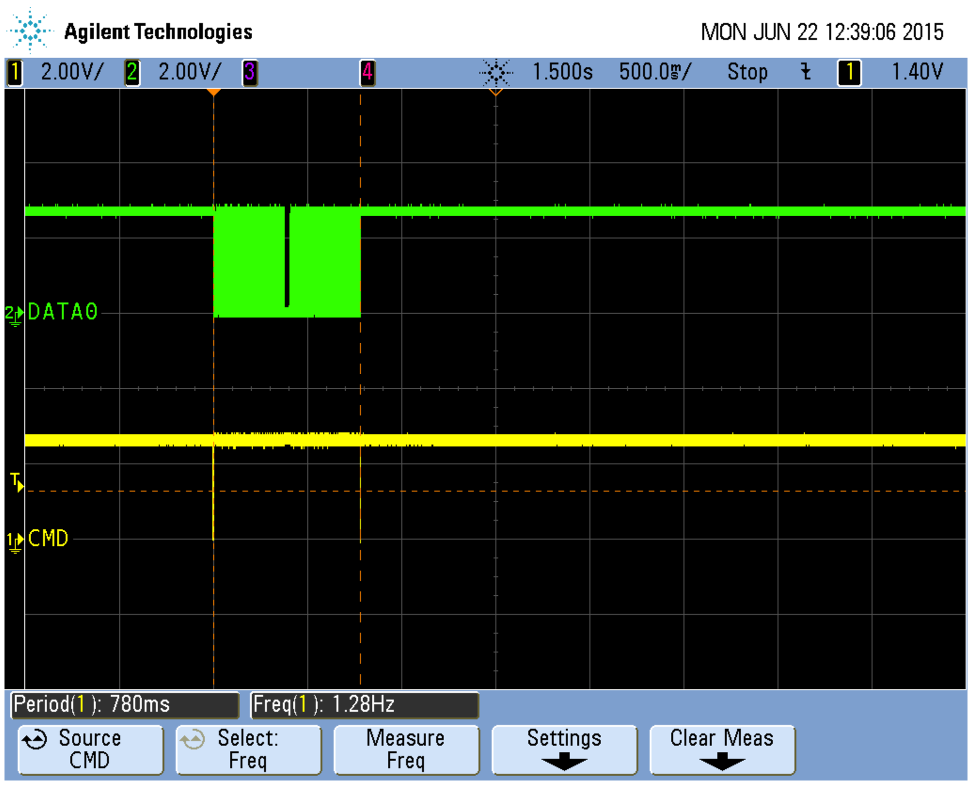


Figure 16: Oscilloscope snapshot - write operation.

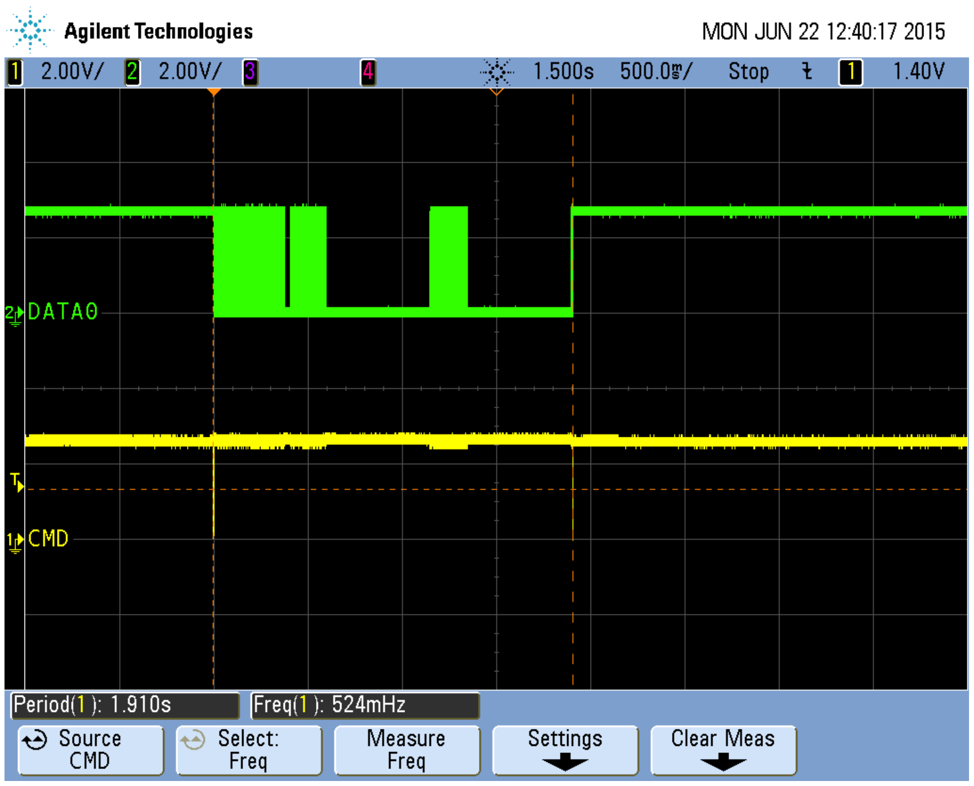


Figure 17: Oscilloscope snapshot - busy state during write operation.

7 Package

```
emmc-axis-v1
|- doc\
|- hw\
|    `- v-kc705-emmc-axis-vm-v2-134\
|- ip\
|    |- emmc_axis_1_0\
|    `- hdmio_1_0\
|- ip-imageon\
|    |- drivers\
|    |- fmc_imageon_vita_receiver_v2\
|    |- interfaces\
|    |- ip_projects\
|    `- sw_services\
|- pcb\
|    |- layers\
|    `- sch\
`- sw\
     |- emmc-axis-tpg\
     `- emmc-axis-vita-mono\
```

8 References

[1] Micron, „MTFC32GJWDQ-4M AIT Z," [Online]. Available: http://www.micron.com/parts/nand-flash/managed-nand/mtfc32gjwdq-4m-ait-z.

[2] XILINX, „KC705 Evaluation Board User Guide UG810 (v1.6.1)," 13 April 2015. [Online]. Available: http://www.xilinx.com/support/documentation/boards_and_kits/kc705/ug810_KC705_Eval_Bd.pdf.

[3] JEDEC, „Embedded Multimedia Card (e•MMC), Electrical Standard 4.51, JESD84-B451," 2011. [Online]. Available: http://www.jedec.org/sites/default/files/docs/JESD84-B451.pdf.

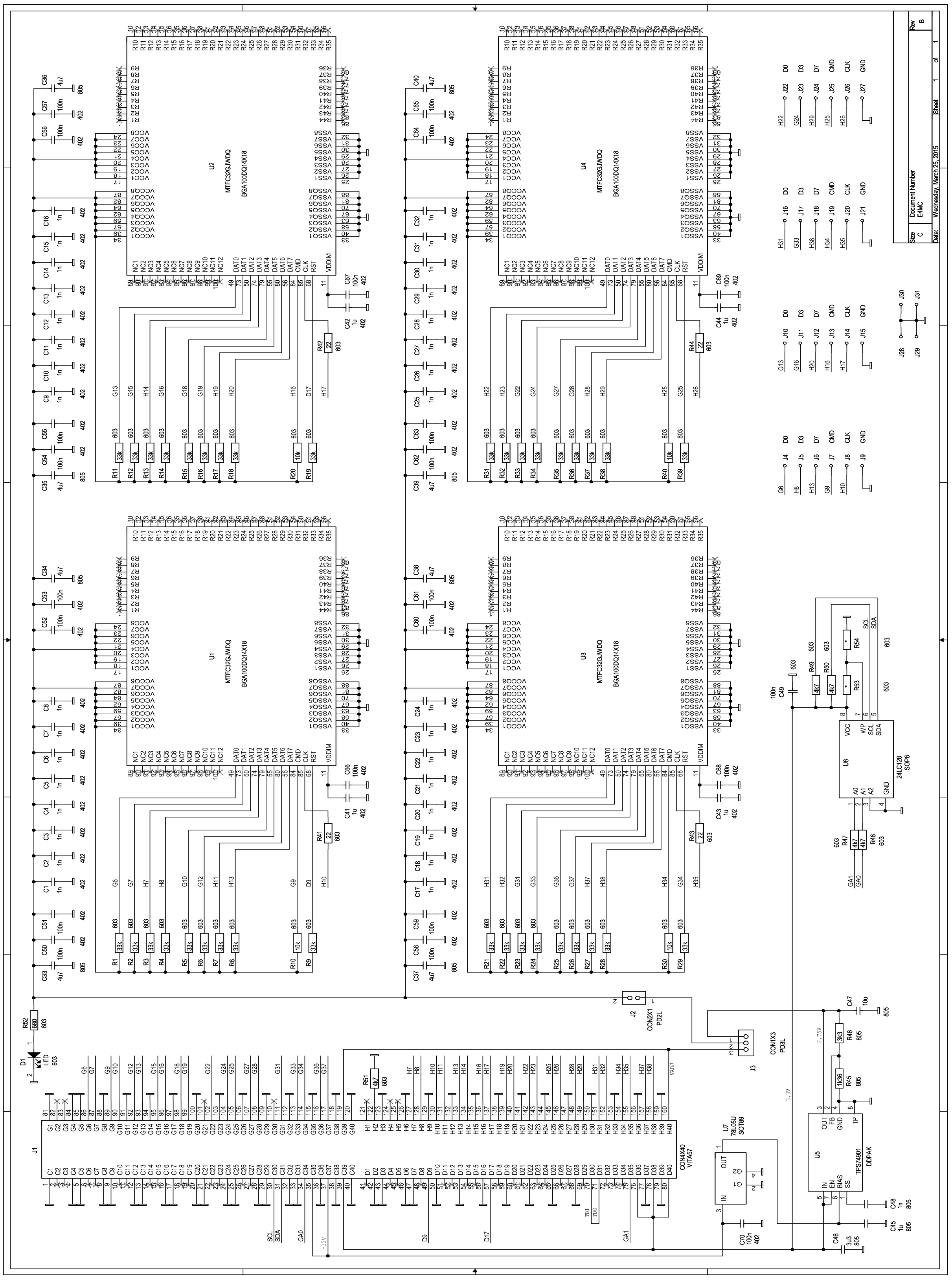
[4] Industries Semiconductor Components, „VITA 2000 2.3 Megapixel 92 FPS Global Shutter CMOS Image Sensor," June 2013. [Online]. Available: http://www.onsemi.com/pub/Collateral/NOIV1SN2000A-D.PDF.

[5] AVNET, „HDMI Input/Output FMC Module," [Online]. Available: http://www.em.avnet.com/en-us/design/drc/Pages/HDMI-Input-Output-FMC-module.aspx.

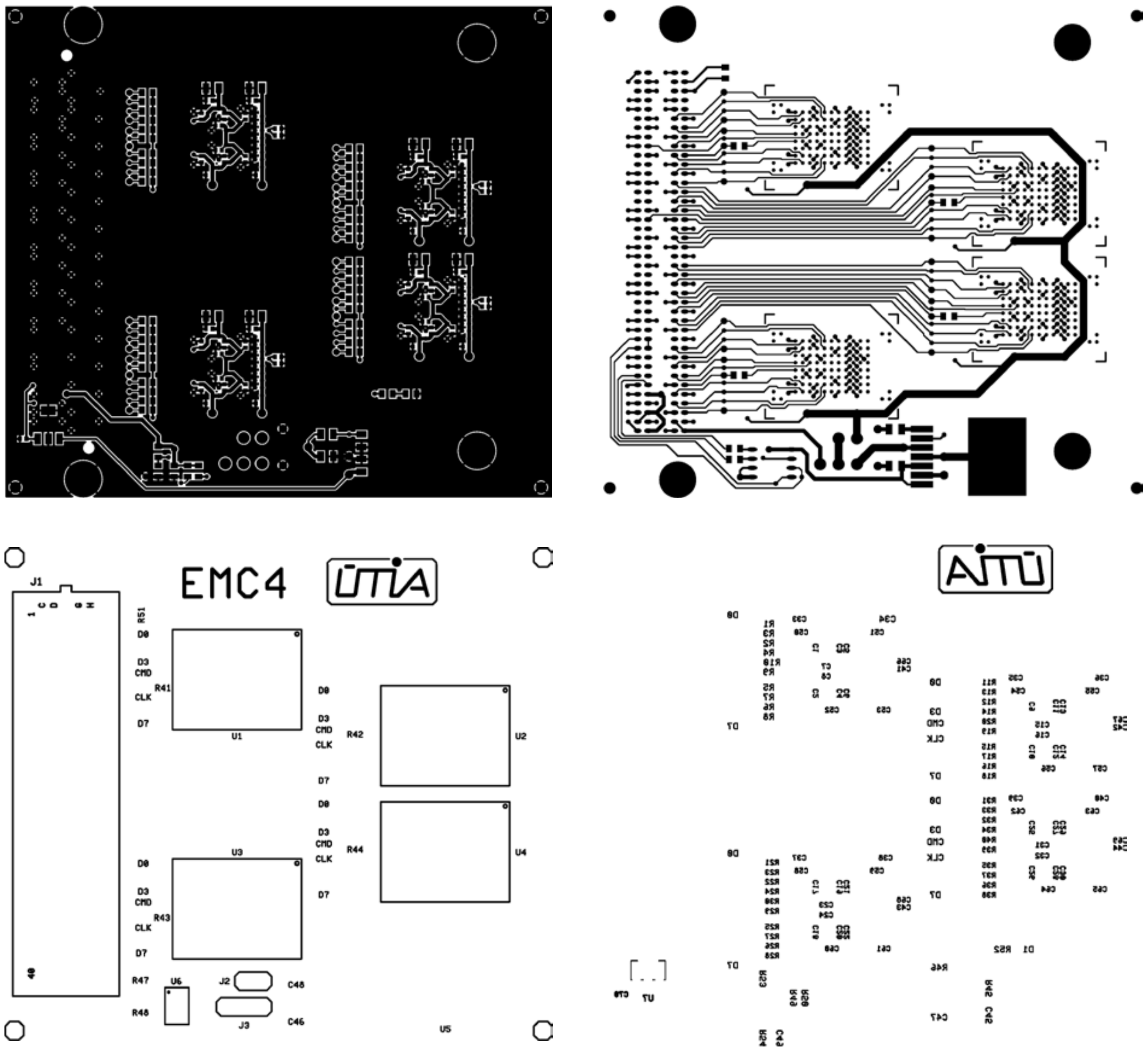
[6] MMCA Technical Committee, „The MultiMediaCard - System Specification Version 3.31," 2003. [Online]. Available: http://read.pudn.com/downloads92/ebook/361855/MMC-System-Spec-v3.31-free.pdf.

A. ÚTIA 4x e•MMC Board

A.1 Schematic



A.2 PCB



A.3 BOM

Item	Quantity	Reference	Part	Package
1	32	C1,C2,C3,C4,C5,C6,C7,C8,C9,C10, C11,C12,C13,C14,C15,C16,C17,C18, C19,C20,C21,C22,C23,C24,C25,C26, C27,C28,C29,C30,C31,C32,C48	1n	402
2	8	C33,C34,C35,C36,C37,C38,C39,C40	4u7	805
3	4	C41,C42,C43,C44	1u	402
4	1	C45	1u	805
5	1	C46	3u3	805
6	1	C47	10u	805
8	1	C49	100n	603
9	21	C50,C51,C52,C53,C54,C55,C56,C57, C58,C59,C60,C61,C62,C63,C64,C65, C66,C67,C68,C69,C70	100n	402

Item	Quantity	Reference	Part	Package
10	1	D1	LED	603
11	1	J1	CON4X40	VITA57
12	1	J2	CON2X1	PD2L
13	1	J3	CON1X3	PD3L
14	36	R1,R2,R3,R4,R5,R6,R7,R8,R9,R11, R12,R13,R14,R15,R16,R17,R18,R19, R21,R22,R23,R24,R25,R26,R27,R28, R29,R31,R32,R33,R34,R35,R36,R37, R38,R39	33k	603
15	4	R10,R20,R30,R40	10k	603
16	4	R41,R42,R43,R44	22	603
17	1	R45	1k36 (1K5||15K)	805
18	1	R46	3k3	805
19	5	R47,R48,R49,R50,R51	4k7	603
20	1	R52	680	603
21	2	R53(JMP1),R54(JMP2)	0	
22	4	U1,U2,U3,U4	MTFC32GJWDQ	BGA100DQ14X18
23	1	U5	TPS74901	DDPAK
24	1	U6	24LC128	SOP8
25	1	U7	78L05U	SOT89