

# Sundance Multiprocessor Technology Limited Product Specification

Form : QCF54  
Template Date :  
5 November 2014

<b>Product Name:</b>	EMC <sup>2</sup>
<b>Subsystem:</b>	EMC <sup>2</sup>
<b>Product Number:</b>	EMC <sup>2</sup>
<b>Document Issue Number:</b>	1
<b>Issue Date:</b>	29 June 2016
<b>Original Author:</b>	E. Wheatley

## EMC<sup>2</sup> System Manual

Sundance Multiprocessor Technology Ltd, Chiltern House,  
Waterside, Chesham, Bucks. HP5 1PS.

This document is the property of Sundance and may not be copied  
nor communicated to a third party without prior written  
permission.

© Sundance Multiprocessor Technology Limited 2009



## Revision History

Issue	Changes Made	Date	Initials
1	Original	16/6/2016	EW
2	Changed PC IP address. Updated the screen shot of the GUI. Added the output resolution and the video and resolution settings.	29/6/2016	EW
3	More details added	05/7/2016	EW

## Table of Contents

<b>1</b>	<b>EMC<sup>2</sup> System Overview .....</b>	<b>4</b>
<b>2</b>	<b>System Data Flows.....</b>	<b>5</b>
2.1	Firmware.....	5
2.2	UDP server.....	9
2.3	Petalinux.....	9
2.4	UDP Client .....	9
<b>3</b>	<b>Software/firmware installation .....</b>	<b>10</b>
<b>4</b>	<b>Running the demo .....</b>	<b>11</b>
4.1	Hardware.....	11
4.1.1	Hardware requirement .....	11
4.1.2	Hardware setup.....	12
4.2	Software.....	12
<b>5</b>	<b>Building the demo.....</b>	<b>16</b>
5.1	Platform hardware .....	16
5.2	Firmware.....	16
5.3	MicroBlaze application and UDP server .....	17
5.4	Boot.bin.....	17
5.5	Petalinux.....	18
5.6	Uboot environment .....	19
<b>6</b>	<b>References .....</b>	<b>19</b>

## Table of Figures

Figure 1: System data flow.....	5
Figure 2: Registers.....	6
Figure 3: MicroBlaze commands.....	7
Figure 4: HDMI setup registers.....	8
Figure 5: Installation structure .....	11
Figure 6: Hardware setup .....	12
Figure 7: Serial console settings .....	13
Figure 8: UDP client running.....	14
Figure 9: UDP server running .....	15
Figure 10: Demo running.....	16

# 1 EMC<sup>2</sup> System Overview

The demonstration runs on a stand-alone EMC<sup>2</sup> Development Platform which is a PCIe/104 OneBank™ board with dual ARM9 CPU, a reconfigurable FPGA Logic and an interface to CPU specific I/O features. The EMC<sup>2</sup> is a carrier board for a Trenc compatible SoC module.

In this system, the EMC<sup>2</sup> Development Platform is controlled by a Graphical User Interface host application over an Ethernet connection for live video. The Ethernet interface is made available on an add-on board called SEIC (Sundance External Interface Connector).

The purpose of this demo is to allow real-life data, in this case a video-stream from a HDMI Output to be loaded into the Zynq's DDR memory and then displayed again on a second HDMI-Input device (typically a monitor). In this example, a Xilinx 32-bit MicroBlaze CPU controls the transfer of data between the HDMI input, the DDR3 memory and the HDMI output.

In this demonstration, a VITA57.1 FMC® compatible Daughter Card is plugged to the EMC<sup>2</sup>-DP to provide HDMI input/output capabilities. The input video is stored to DDR3 memory and the output video is read from DDR3 memory. A MicroBlaze 32-bit soft-core processor is implemented in the Zynq PL to control the HDMI interface to have access to the DDR3 memory and the video data for processing.

The data are stored in the DDR memory in 3 buffers arranged in a circular manner.

The input and output HDMI rate are identical.

The memory read and write are alternated to provide data for HDMI output while storing the HDMI input data. The memory read is performed ahead of the memory write.

The GUI host application controls the EMC<sup>2</sup>-DP via the 1GB Ethernet port using a write/read API. The Ethernet link is used to communicate with the ARM processor core running Linux that, in turn, communicates with the MicroBlaze processor to configure the system and drive its memory accesses remotely.

The host application can also provide any data to write to memory to be processed and then read it back.

For example, it can load a picture to display on the HDMI output and read a picture captured on the HDMI input.

A UDP server runs in Linux on the PS and the host application runs the UDP client. The MicroBlaze controls the VDMA IP core to perform the read and write operations requested by the host. The EMC<sup>2</sup>-DP boots from SD card so the application code runs at power up and could be used to configure the system from Ethernet.

From the graphic interface, the control commands can be sent manually. But more efficiently, the host application can load a set of commands from an xml file and send it to the system. Typically, a set of commands will be:

- Read from memory (with the option to display data in the GUI and/ or save it to a file),
- write to memory (the data can be manually entered or come from a file),
- Sleep.

An XML command file can also be created or modified from the graphic interface.

The use of XML files facilitates the system configuration but also makes it easier and quicker to repeat operations like running full system tests. Therefore, the system is very flexible and can be controlled remotely using the Ethernet port.

## 2 System Data Flows

The embedded system operates under the control of a windows GUI by sending commands to start the acquisition on the HDMI input and display data on the HDMI output.

Thus this system consists of 4 main software/firmware parts:

- The firmware for the Zynq PL (Programmable Logic)
- The UDP server on the Zynq PS (Processing System)
- The Petalinux project to run Linux on the Zynq.
- The UDP client in shape of a GUI in Windows.

The figure below illustrates the system data flow.

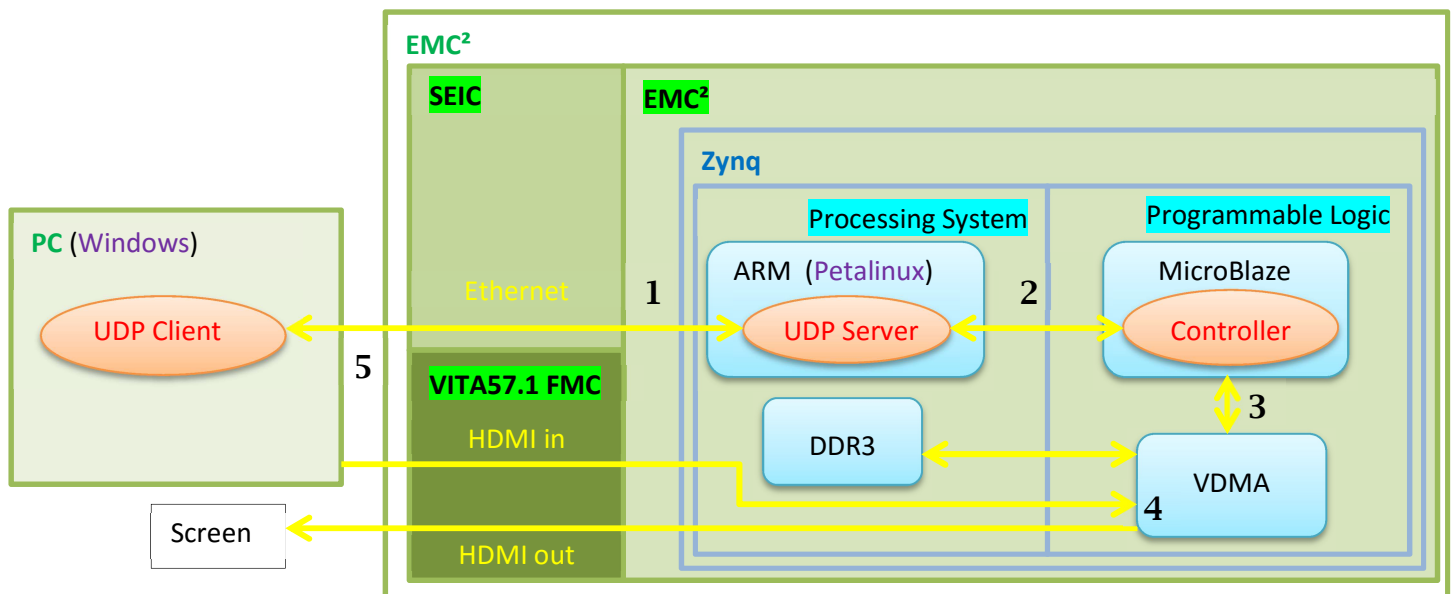


Figure 1: System data flow

1. The UDP Client sends a request to the UDP server via Ethernet.
2. The UDP server relays the request to the MicroBlaze.
3. The MicroBlaze controls the VDMA IP core to perform the request.
4. The VDMA IP core initializes and setups the HDMI input and output according to the MicroBlaze's commands.
5. The result of the request is sent back to the UDP Client.

### 2.1 Firmware

The firmware was created with the IP integrator in Vivado. The firmware is assembled from standard Xilinx IP and Avnet FMC\_Imageon IP.

The HDMI in and HDMI out IP cores from Avnet (FMC-Imageon) used the AXI4-stream Video protocol. The output resolution is set at 1280x720 and the video frequency is set at 65MHz.

The Xilinx® LogiCORE™ IP AXI VDMA core provides the high-bandwidth direct memory access between the DDR memory and the HDMI peripherals.

The software for the MicroBlaze runs as a bare-metal application and has no external dependencies.

Some addresses in the DDR memory has been reserved for specific purposes. The Microblaze on the PL and the UDP server communicates with each other using these addresses. The Linux kernel has been compiled as such as it isn't aware of these addresses and therefore can't run on this part of the memory. This was necessary to avoid a system crash.

These registers are defined in the header file "*source\SDK\include\EMC2\_HDMI\_demo\_cmd.h*" included in the source code of both the MicroBlaze and the UDP server. Also in this header file are the valid requests the MicroBlaze can receive.

The table below describes these addresses:

Register address	value	description
CMD_ADDR	0x30000000	Address where the UDP server writes the request ( <b>Error! Reference source not found.</b> ) the MicroBlaze has to perform. The MicroBlaze keeps polling this address until it sees a new request. Then the MicroBlaze executes the command. Once completed, the MicroBlaze changes CMD_ADDR to COMMAND_COMPLETE in order to acknowledge the UDP server's query. The UDP will modify the register CMD_ADDR, only after the MicroBlaze has successfully completed its task. Thanks to this handshake between the PL and the PS all the commands are process in the order they are received and none are lost.
DATA_ADDR	0x10000000	Address where the HDMI data are transferred to and from. The data are stored in 3 buffers in a circular manner. Each buffer contains a video frame.
ERROR_ADDR	0x30000010	
REG_ADDR	0x30000100	Address of the register containing the information about the HDMI.

Figure 2: Registers

The MicroBlaze can control the HDMI input and HDMI out by requesting a string of actions to be performed. The MicroBlaze does so by writing to the register CMD\_ADDR. The table below lists the valid MicroBlaze requests:

Command name	Value	Description
FMC_IMAGEON	1	Command to initialise the routine for XPS_IIC implementation and the FMC_IMAGEON driver
VIDEO_CLK_INIT	2	Command to initialize the Video Clock Synthesizer. The CDCE913 has 3 outputs which are configured as follows: * Y1 => 74.25 MHz * Y2 => off * Y3 => off
VIDEO_CLK_CONFIG	3	Command to configure the Video Clock Synthesizer's Y1 output.
HDMII_INIT	4	Command to initialize the HDMI Input Interface.
HDMIO_INIT	5	Command to initialize the HDMI Output Interface.
VDMA_OUT_INIT	7	Command to setup the read channel and start the DMA engine to transfer.
VTC_OUT_CONFIG	8	Command to initialize and configure the VTC generator.
HDMII_GET_LOCK	9	Command to check the HDMI input signal is locked.
VIDEO_IN_INFO	10	Command to write the HDMI input settings to the information register.
VTC_IN_RESET	11	Command to initialize and reset the VTC detector.
VDMA_IN_STOP	12	Command to stop the DMA engine to write.
VDMA_IN_INIT	13	Command to setup the write channel and start the DMA engine to transfer.
VDMA_INIT	14	Command to initialize DMA engines.
VDMA_OUT_STOP	15	Command to stop the DMA engine to read.
VDMA_IN_START	16	Command to start the DMA engine to write.
VDMA_OUT_START	17	Command to start the DMA engine to read.

Figure 3: MicroBlaze commands

It is also possible for the MicroBlaze to read the HDMI input/output settings. The table below describes the HDMI settings registers.

Address	Parameter	Address	Parameter
---------	-----------	---------	-----------

0x30000100	Video clock <sup>1</sup>	0x30000148	Input locked
0x30000104	Output resolution <sup>2</sup>	0x3000014C	Input resolution <sup>2</sup>
0x30000108	Output width	0x30000150	Input width
0x3000010C	Output height	0x30000154	Input height
0x30000110	Output HDMI	0x30000158	Input HDMI
0x30000114	Output encrypted	0x3000015C	Input encrypted
0x30000118	Output interlaced	0x30000160	Input interlaced
0x3000011C	Output colour depth	0x30000164	Input colour depth
0x30000120	Output horizontal active video	0x30000168	Input horizontal active video
0x30000124	Output horizontal front porch	0x3000016C	Input horizontal front porch
0x30000128	Output horizontal synchronisation width	0x30000170	Input horizontal synchronisation width
0x3000012C	Output horizontal synchronisation polarity	0x30000174	Input horizontal synchronisation polarity
0x30000130	Output horizontal back porch	0x30000178	Input horizontal back porch
0x30000134	Output vertical active video	0x3000017C	Input vertical active video
0x30000138	Output vertical front porch	0x30000180	Input vertical front porch
0x3000013C	Output vertical synchronisation width	0x30000184	Input vertical synchronisation width
0x30000140	Output vertical synchronisation polarity	0x30000188	Input vertical synchronisation polarity
0x30000144	Output vertical back porch	0x3000018C	Input vertical back porch

<sup>1</sup> Video Clock: 0=25.175MHz, 1=27MHz, 2=40MHz, 3=65MHz, 4=74.25MHz, 5=110MHz, 6=148.5MHz, 7=162MHz

<sup>2</sup> Resolution: 0=VGA, 1=480P, 2=576P, 3=SVGA, 4=XGA, 5=720P, 6=SXGA, 7=1080P, 8=UXGA

Figure 4: HDMI setup registers

The firmware and MicroBlaze software for the Zynq (xc7z015clg485-1) were developed in Vivado 2015.2 and its SDK. The PC software environment is Windows 7 64-bit or a later version.

The firmware source code is located in the folder: “source\Vivado\EMC2\_HDMI\_demo\_bit”.



The MicroBlaze source code is located in the folder: “*source\SDK\EMC2\_HDMI\_demo\_app*”.

## 2.2 UDP server

The UDP server runs on Linux on the Zynq of the EMC<sup>2</sup>.

Once the UDP server has successfully initialised a datagram socket, it waits for a request from the UDP client. Once received, the request is parsed into a readable format: command, address, size and data.

**command:** it can be of 2 types:

- a write command (“w”) when the UDP client requests to write *size* of *data* to *address*.
- a read command (“r”) when the UDP client requests to read *size* of *data* from *address*.

**address:** the DDR memory address to transfer the data to or from.

**size:** the size of the data to transfer.

**data:** data to transfer. The data can be of 2 types:

- a 32-bit word which is a command for the MicroBlaze to execute. In that case, the command is written to a specific registry which is an address of the DDR reserved for that purpose only.
- a video frame

Then the UDP server will access Linux’s memory resources so it can map the physical addresses to virtual addresses.

In the case of a write request, the UDP server will send to the client an “*Acq*” message if the request was successful.

The UDP server software has to be manually started from a serial console after each system reboot.

The UDP server is written in C and was developed and compiled with Xilinx SDK.

The UDP server source code is located in the folder: “*source\SDK\UDP\_Server*”.

## 2.3 Petalinux

Petalinux 2014.4 is running on the ARM. Petalinux was setup and built on a 64-bit Ubuntu 14.04 LTS machine (kernel 3.16.0-30-generic).

The Petalinux source code is located in the folder: “*EMC2\_Demo\source\Petalinux*”.

## 2.4 UDP Client

A Graphical User Interface is used to interface to the EMC<sup>2</sup> Development Platform.

The GUI is a user-friendly way to control the HDMI interface on the EMC<sup>2</sup>-DP through the Ethernet link.

Through this GUI, data transfers to and from the DDR memory inside the platform are possible. Therefore, the GUI acts as a client and the Linux application on the

ARM processor acts as a server. The network protocol UDP is used for this communication.

A request is made of a type, an address, a length, a file or data, an IP address and a port number.

The GUI uses XML files to load a group of commands, enabling the user to access the platform in a quick and constant manner.

The GUI host application has been developed in C++ in the QT5 software environment to make the deployment to other platforms possible.

The GUI source code is located in the folder: "EMC2\_Demo\source\UDP\_Client\Demo".

### 3 Software/firmware installation

All the files necessary to run or rebuild the EMC<sup>2</sup> demo are included in the compressed file *EMC<sup>2</sup>\_demo.zipx*.

Once uncompressed, the folder structure is has described in "Figure 5: Installation structure".

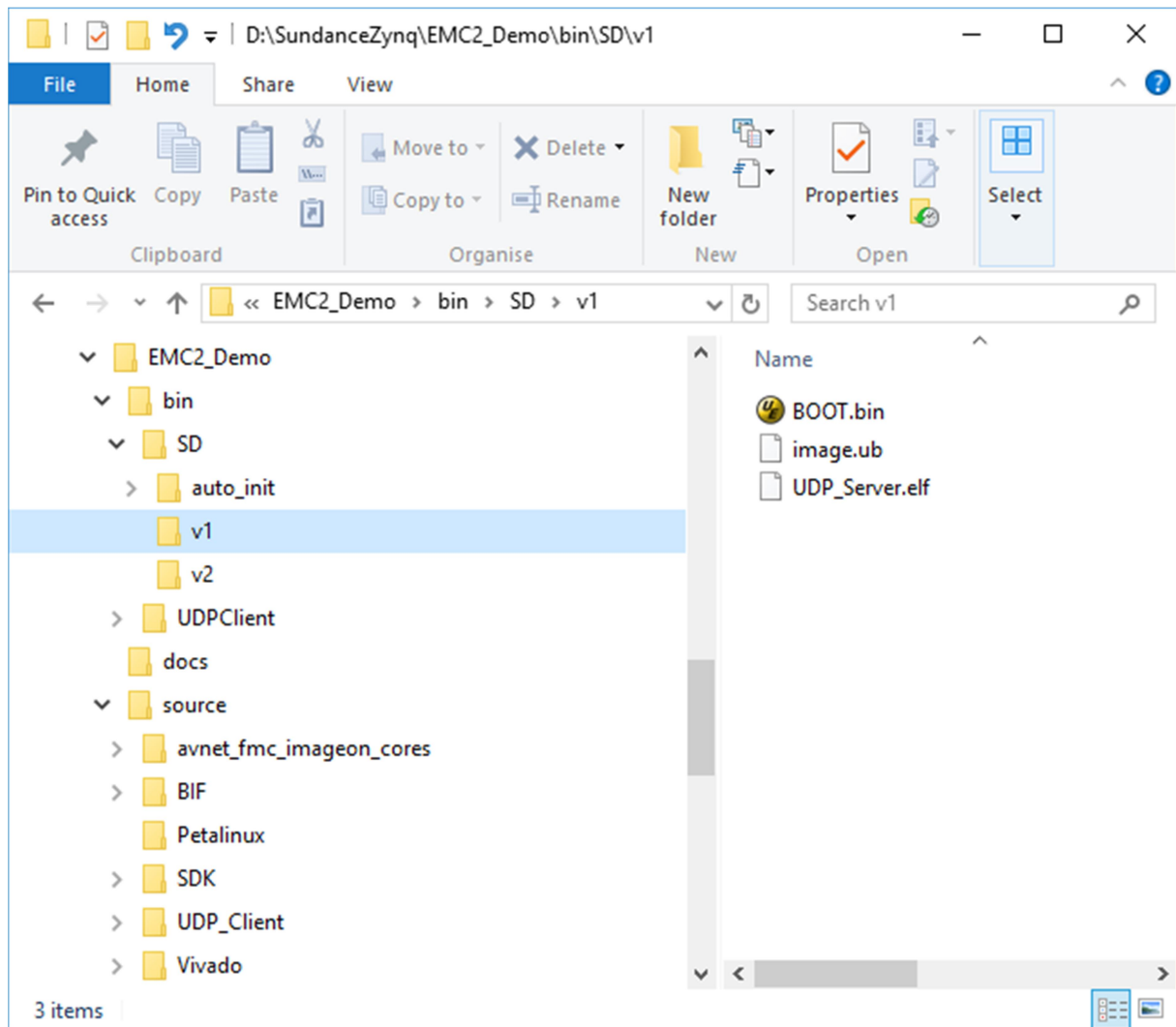


Figure 5: Installation structure

The folder “*bin*” contains all the compiled and generated files necessary to run the demo.

The folder “*source*” contains all the source files necessary to rebuild the demo.

Some subfolder are called “*V1*” or “*V2*”, they referred respectively to version 1 or version 2 of the EMC<sup>2</sup> platform.

## 4 Running the demo

### 4.1 Hardware

#### 4.1.1 Hardware requirement

For the demo you will need

- an EMC<sup>2</sup> board with a SEIC and a FMC Imageon board,
- a HDMI screen
- a computer running Windows 8
- cables: power, Ethernet, mini USB to USB, 2x HDMI.

### 4.1.2 Hardware setup

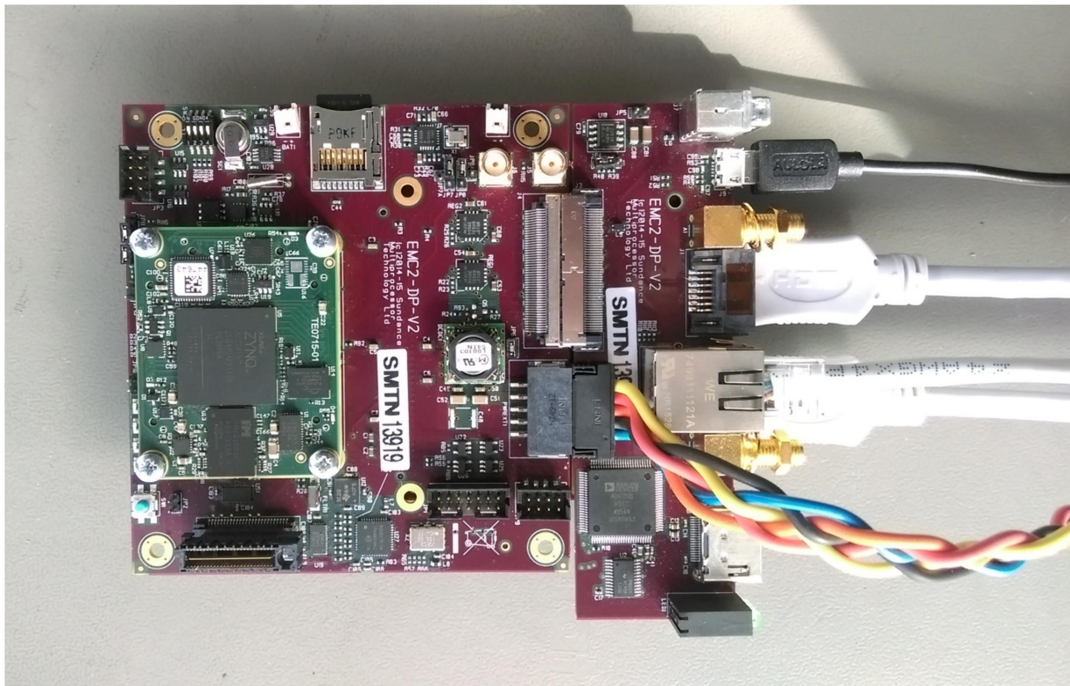


Figure 6: Hardware setup

1. Connect the power cable to the EMC<sup>2</sup> board but don't switch the power on for now.
2. Connect the USB cable between the EMC<sup>2</sup> board and the computer.
3. Connect the Ethernet cable between the EMC<sup>2</sup> board and the computer.
4. On the computer go to:  
"Control panel" -> "Network and sharing centre" -> "Change adapter settings"  
- > "Ethernet" -> "Properties" -> "Internet Protocol 4" and change the IP address to "192.168.0.14".
5. Connect the computer screen to the HDMI in on the FMC board and the other monitor to the HDMI out on the FMC board.

### 4.2 Software

- 1- Copy to the micro SD card (previously formatted as explained here: <http://www.xilinx.com/Prepare+Boot+Medium>) the files: *image.ub*, *BOOT.bin* and *UDP\_Server.elf* from the "bin/SD" folder.  
Note that the "V1" folder contains the files for the EMC<sup>2</sup>\_DP version 1, the "V2" folder the files for the EMC<sup>2</sup>-DP version 2 and the folder "auto-init" the files where the HDMI input and output are configured at boot up.
- 2- Insert the micro SD card and switch the power on to boot the EMC<sup>2</sup> board.
- 3- Open a serial console (for example Putty) with the following settings:  
Speed: 115200  
Data bits: 8  
Stop bits: 1

Parity: *None*  
Flow control: *None*

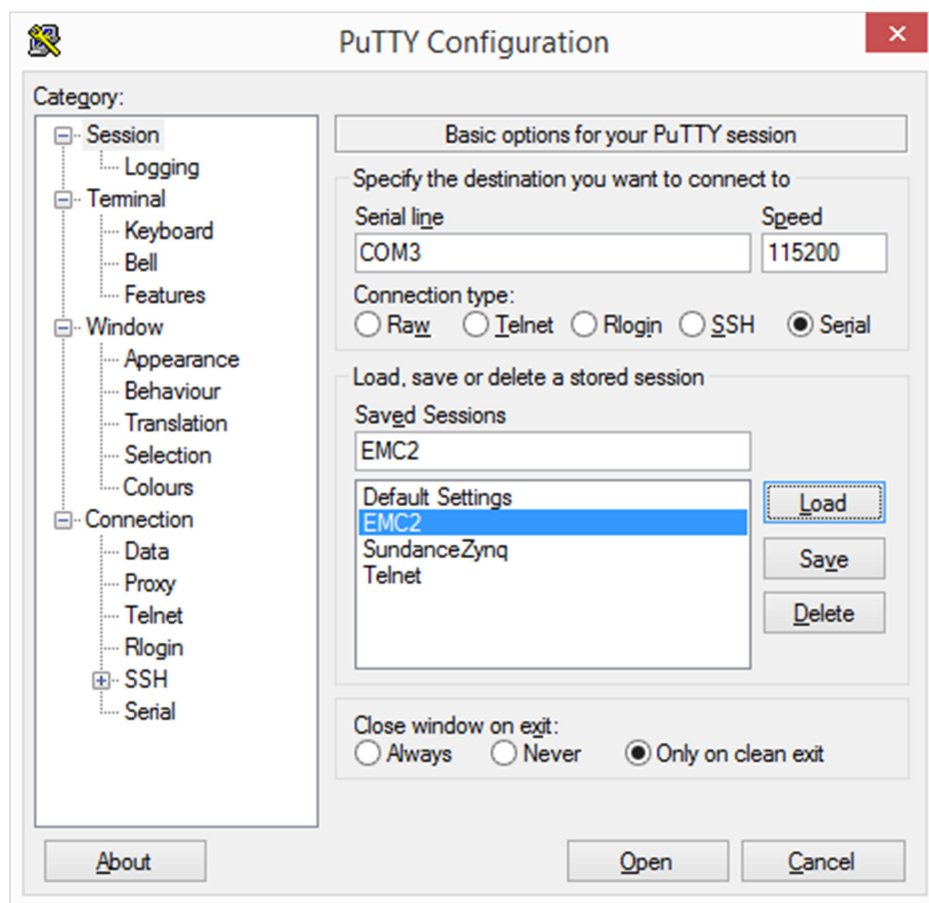
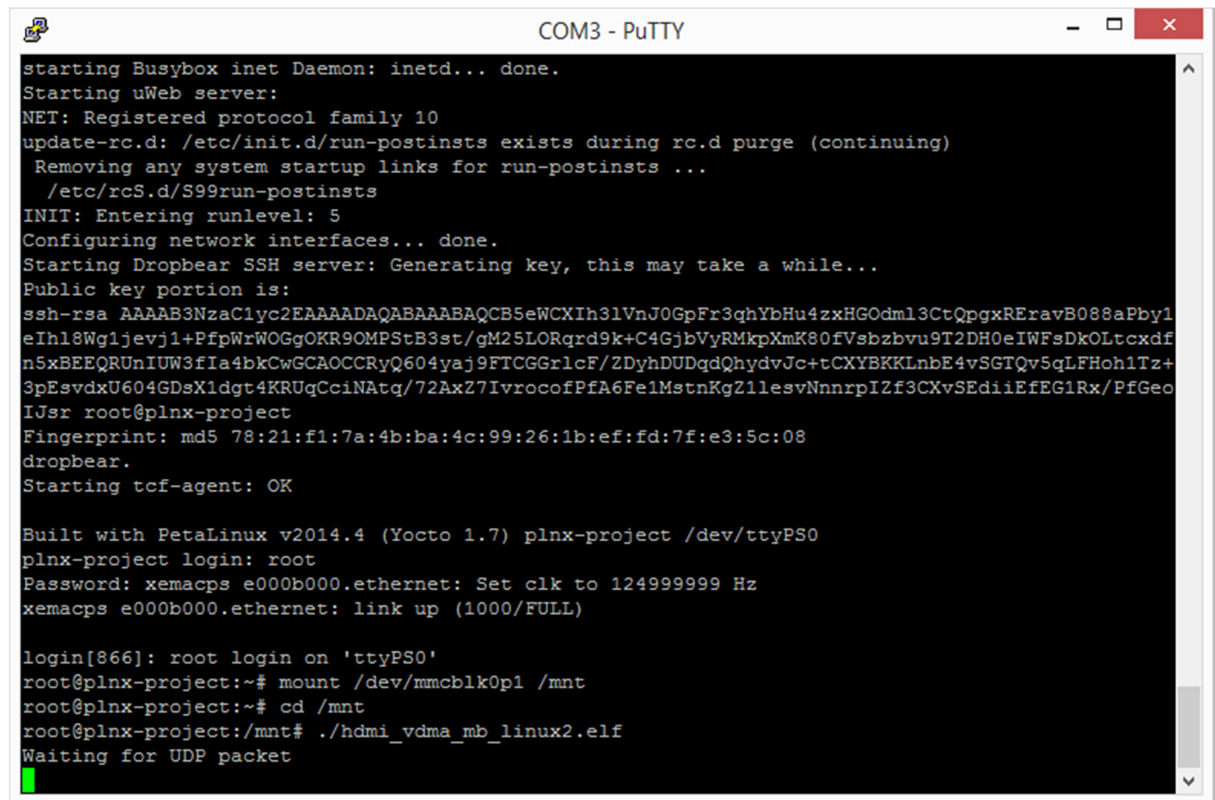


Figure 7: Serial console settings

- 4- In the serial console, log on in Linux with the following credentials:  
Name: *root*  
Password: *root*
- 5- In the serial console type the following to run the UDP server:  

```
>>mount /dev/mmcbk0p1 /mnt  
>>cd /mnt  
>>./UDP_Server.elf
```



```
starting Busybox inet Daemon: inetd... done.
Starting uWeb server:
NET: Registered protocol family 10
update-rc.d: /etc/init.d/run-postinsts exists during rc.d purge (continuing)
Removing any system startup links for run-postinsts ...
/etc/rcS.d/S99run-postinsts
INIT: Entering runlevel: 5
Configuring network interfaces... done.
Starting Dropbear SSH server: Generating key, this may take a while...
Public key portion is:
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCB5eWCXIh3lVnJ0GpFr3qhYbHu4zxHGOdml3CtQpgxREravB088aPby1
eIh18Wg1jevjl+PfpWrwOGgOKR9OMPStB3st/gM25LORqrd9k+C4GjbVyRMkpXmK80fVsbzbvu9T2DH0eIWfsDkOLtcxdf
n5xBEEQRUnIUW3fIa4bkCwGCAOCCryQ604ya9FTCGGr1cF/ZDyhDUDqdQhydVJc+tCXyBKKLnBE4vSGTQv5qLFHoh1Tz+
3pEsvdxU604GDsXldgt4KRUqCciNatq/72AxZ7IvrocOfPfA6Fe1MstnKgZ1lesvNnnrpIZf3CXvSEdiiEfeG1Rx/PfGeo
IJsr root@plnx-project
Fingerprint: md5 78:21:f1:7a:4b:ba:4c:99:26:1b:ef:fd:7f:e3:5c:08
dropbear.
Starting tcf-agent: OK

Built with PetaLinux v2014.4 (Yocto 1.7) plnx-project /dev/ttyPS0
plnx-project login: root
Password: xemacps e000b000.ethernet: Set clk to 124999999 Hz
xemacps e000b000.ethernet: link up (1000/FULL)

login[866]: root login on 'ttyPS0'
root@plnx-project:~# mount /dev/mmcblk0p1 /mnt
root@plnx-project:~# cd /mnt
root@plnx-project:/mnt# ./hdmi_vdma_mb_linux2.elf
Waiting for UDP packet
```

Figure 8: UDP client running

6- On the computer open the UDP client “*bin/SmtUDPCClient/SmtUDPCClient.exe*”

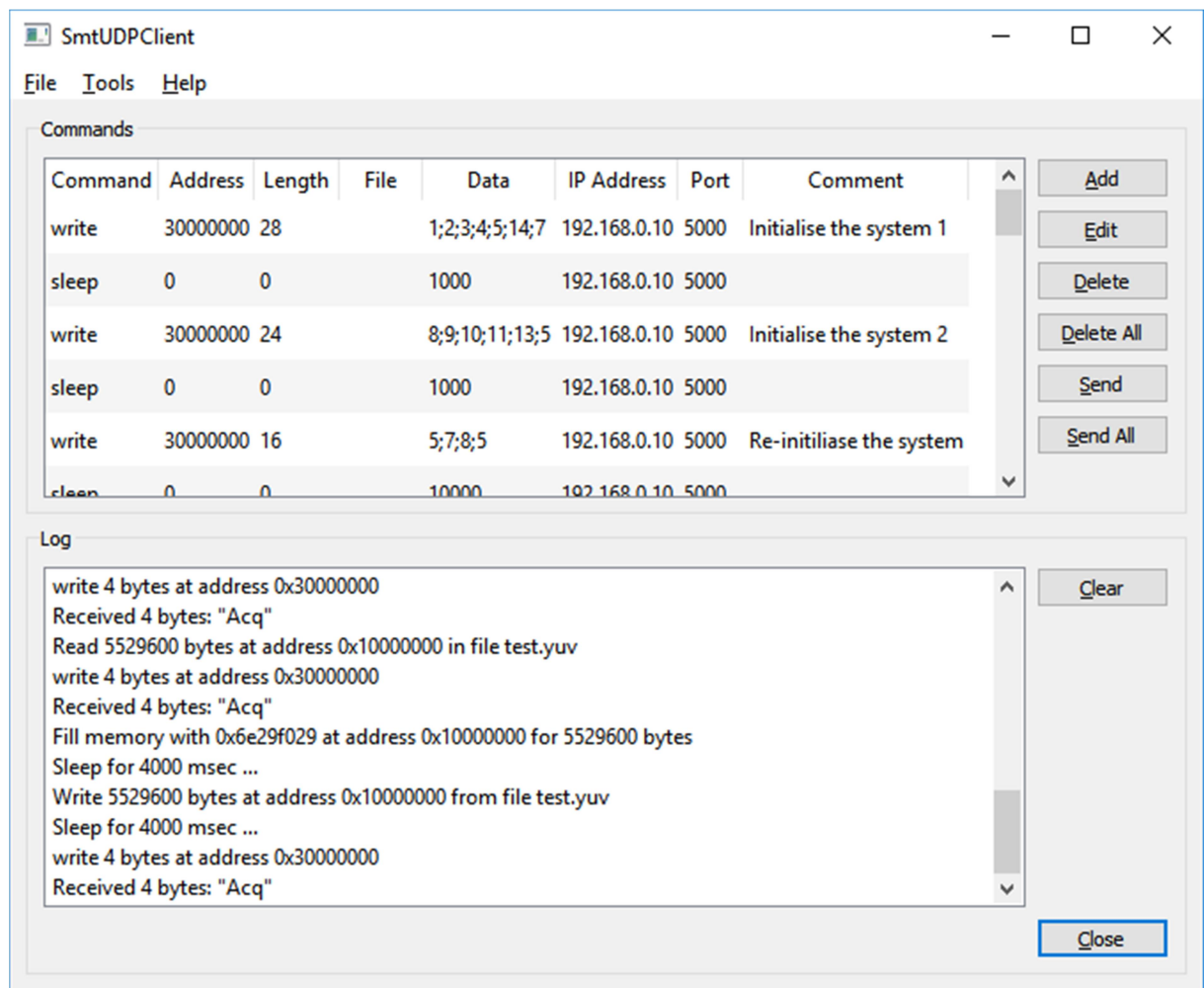


Figure 9: UDP server running

- 7- In the GUI go to "File" -> "Load file" and browse to the file *bin/SmtUDPClient/RunTest.xml*
- 8- Press the button "Send All". A series of test should be performed. Including HDMI input displayed on the HDMI output, also colour written in DDR through Ethernet being displayed on the HDMI out screen. The video output resolution is set at 1280x720.



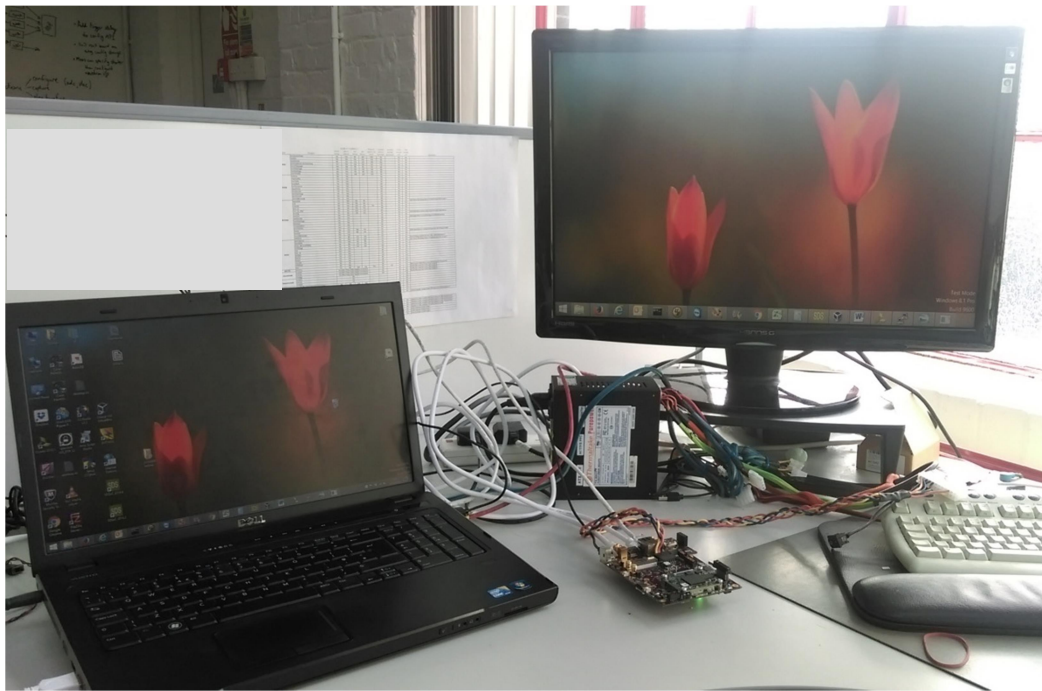


Figure 10: Demo running

Note: The file *commands.xml* can be used to display in the UDP client a set of commands to send to the MicroBlaze. For example, commands to read the HDMI input and output status and information.

## 5 Building the demo

All the source files and the project files are included in this demo. However if a project rebuild is needed, please follow the steps below.

### 5.1 Platform hardware

The EMC<sup>2</sup> platform needs to be added to the Xilinx folder.

To do so:

- Go to the folder “C:\Xilinx\ Vivado\2015.2\data\boards\board\_files”
- Copy the folder “EMC2\_Demo\platform”

### 5.2 Firmware

To rebuild the bitstream, open the project file “EMC2\_Demo\source\Vivado\EMC2\_HDMI\_demo\_bit\EMC2\_HDMI\_demo\_bit.xpr” in Vivado.

Next set the Vivado Repositories:

- go to “Project settings”->”IP”
- Add *avnet\_fmc\_imageon\_cores* folder to the local repositories then press OK



Then select “*build Bitstream*”.

To build the bitstream with the MicroBlaze application embedded inside, open Xilinx SDK, go to “*Xilinx Tools*”-> “*Program FPGA*”. In the “*Program FPGA*” window, in “*Software Configuration*”, select the *EMC2\_HDMI\_demo\_app.elf* file for the microBlaze processor.

The resulting bitstream file can be found in the platform directory: *EMC2\_Demo\source\SDK\EMC2\_HDMI\_demo\_platform*.

### 5.3 MicroBlaze application and UDP server

There are 2 options for the system initialisation: at boot time or manually. For the system to get initialize at boot up, the value of “*#if*” need to be changed to 1 in the MicroBlaze source file: “*EMC2\_Demo\source\SDK\EMC2\_HDMI\_demo\_app\src\fmc\_imageon\_hdmi\_framebuffer.c*”:

```
// auto initialize the system
#ifdef 0
    if (test<17) {
        if (test==7) {
            while (sleep!=10000000) sleep++;
            sleep = 0;
        }

        if (test==13) {
            while (sleep!=10000000) sleep++;
            sleep = 0;
        }
        *pCmd= pCmdTest[test];
        test++;
    }
#endif
```

To rebuild the MicroBlaze application, the FSBL (First Stage Boot Loader) and the UDP server, open the XILINX SDK workspace which is in the folder *EMC2\_Demo\source\SDK*.

In Xilinx SDK, you need to add the FMC Imageon repository. To do so:

- Go to “*Xilinx Tools*” -> “*Repositories*”
- Add *avnet\_fmc\_imageon\_cores* folder to the local repositories then press OK

Then select the FMC Imageon library:

- Right click the project board support package (*EMC2\_HDMI\_demo\_app\_bsp*)
- Select *Overview* in the top left corner of the BSP window
- Select the *fmc\_iic\_sw* and *fmc\_imageon\_sw* libraries.

Go to “*Project*” -> “*Build All*”.

### 5.4 Boot.bin

To build the “*boot.bin*” file:

- Open Xilinx SDK

- Go to “Xilinx Tools” -> “Create Zynq Boot Image”
- Select “Import from existing BIF file”
- In “Import BIF file path” browse to “EMC2\_Demo\source\BIF\v2\output.bif”
- Select the right files in that order: FSBL.elf, \*.bit, u-boot.elf
- Then press “Create Image”.

## 5.5 Petalinux

To rebuild the Petalinux project you need a Linux machine.

Create a folder “EMC2\_HDMI\_demo” and a subfolder “hwdef”. In the subfolder “hwdef”, copy the hardware description file exported by Vivado “design\_1\_wrapper.hdf”

Go to the folder where Petalinux is installed and type:

```
>>source settings.sh
```

In a terminal window, cd to the EMC2\_HDMI\_demo folder, then create a Petalinux project and configure it:

```
>>petalinux-create --type project --template zynq --name plnx-project
```

```
>>cd hwdef
```

```
>>petalinux-config --get-hw-description -p ../plnx-project
```

In “Subsystem AUTO hardware settings” select “Ethernet settings” then uncheck “Obtain IP address automatically” and set it to “192.168.0.10”

In the folder, “EMC2\_HDMI\_demo/plnx-project/subsystem/linux/configs”, add the Ethernet configuration in the file “system-top.dts” as such:

```
/dts-v1/;
/include/ "system-conf.dtsi"
/ {
    chosen {
        bootargs = "console=ttyPS0,115200 root=/dev/mmcblk0p2 rw
earlyprintk rootfstype=ext4 rootwait devtmpfs.mount=1";
    };
};

&gem0 {
    phy-handle = <&phy0>;
    ps7_ethernet_0_mdio: mdio {
        #address-cells = <1>;
        #size-cells = <3>;
        phy0: phy@0 {
            compatible = "marvell,88e1512";
            device_type = "ethernet-phy";
            reg = <0>;
        };
    };
};
```

Back in the terminal windows, build the project:

```
>>petalinux-build
```

The file “*image.ub*” is in the folder “*EMC2\_HDMI\_demo/plnx-project/images.linux*”.

## 5.6 Uboot environment

In order for the system to run properly the following uboot environment variables need to be set.

```
kernel_img=image.ub
```

```
netstart=0x01000000
```

```
sdboot=echo boot Petalinux; mmcinfo && fatload mmc 0 ${netstart}  
${kernel_img} && bootm
```

## 6 References

Petalinux: <http://www.xilinx.com/products/design-tools/embedded-software/petalinux-sdk.html>

Vivado : <http://www.xilinx.com/products/design-tools/vivado.html>

SDK : <http://www.xilinx.com/products/design-tools/embedded-software/sdk.html>

QT: <http://www.qt.io>

FMC-IMAGEON from Avnet : <https://products.avnet.com/shop/en/ema/kits-and-tools/development-kits/3074457345623596557>

Trenz SoC ZYNQ module : <http://www.trenz-electronic.de/products/fpga-boards/trenz-electronic/te0715-zynq.html>

EMC2 : <http://www.sundance.technology/som-carriers/pc104-boards/emc2-dp/>

Source files : [http://ftp2.sundance.com/Pub/Support\\_Files/Tulipp/EMC2\\_Demo.zip](http://ftp2.sundance.com/Pub/Support_Files/Tulipp/EMC2_Demo.zip)