Unit / Module Description:	PCIe/104 OneBank + ARM + FPGA + FMC carrier
Unit / Module Number:	EMC2-DP
Document Issue Number:	3.1
Issue Date:	04/03/2019
Original Author:	Timoteo Garcia Bertoa

EMC2-DP V2 STARTER'S GUIDE



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 2019



EMC2-DP

Revision History

Issue	Changes Made	Date	Initials
1.0	First draft.	2/6/16	TG
1.1	Added the board files for EMC ² . Updated information about boot images. Added HDMI test project tutorial.	23/6/16	TG
	Added information about the SEIC connector.		
1.2	Added warning at 1.1.2	04/8/16	TG
1.3	Fixed errors + update for z7030	05/10/16	TG
1.4	Added HDMI SDSoC Platform tutorial	07/10/16	TG
2.0	Added chapter 4.Demos	15/10/17	EW
	Changed the structure of 3.6 and 3.7		
	Added how to create an SDSoC platform in 2016.3 and above		
3.0	Re-worked the whole document, using Vivado 17.4, SDK 17.4, SDSoC 17.4	21/02/19	TG
3.1	Corrected paging and updated pictures	04/03/19	TG

Table of Contents

1	In	troduction6
2	H	ardware6
	2.1	How is the hardware distributed?6
	2.2	How can I connect/disconnect the EMC2-DP and the SoM?9
	2.3	How can I connect the EMC2-DP to the PSU?9
	2.4	How can I configure the I/O Voltages?11
	2.5	How can I configure the board to use PCIe?13
	2.6	How can I boot from flash or SD card?15
	2.7	Information about the SEIC Connector16
3	Sc	oftware
	3.1	How can I use Vivado with the EMC ² ?
	3.2	How can I create a Zynq FSBL?
	3.3	How can I create SD boot file?
	3.4	How can I program the FPGA from Flash in Vivado?
	3.5	How can I create an SDSoC Platform?
	3.6	How can I create/run Sundance demos?41

Table of Figures

Figure 1 - EMC2-DP	6
Figure 2 - TE0712, Xilinx Artix-7 FPGA SoM	7
Figure 3 - TE0715-7030, Xilinx Zynq ARM + FPGA SoM	7
Figure 4 - TE0841, Xilinx Kintex UltraScale SoM	8
Figure 5 - TEO820-ZU4EV, Xilinx Zynq UltraScale+	8
Figure 6 - Power supply	9
Figure 7 - Cable for the PSU	10
Figure 8 - Connecting the EMC ² to the PSU	10
Figure 9 - EMC2-DP fully working	10
Figure 10 - JP7 and JP8	11
Figure 11 - 3.3V selection	11
Figure 12 - 2.5V selection	12
Figure 13 - 1.8V selection	12
Figure 14 - Upstream port selection	13
Figure 15 - JP12 sets the host mode, SW2 selects port 0 as upstream port	14
Figure 16 - Boot mode selection	15
Figure 17 - SEIC extension board	16
Figure 18 - SEIC connector	16
Figure 19 - SEIC Connector Pinout	17
Figure 20 - Create a new project	19
Figure 21 - Board file selection	20
Figure 22 - Project information in Vivado	21
Figure 24 - Board interfaces in IP Integrator	21
Figure 24 - How to use interfaces in IPI	22
Figure 25 - Block generated through the interface	22
Figure 26 - Zynq Ultrascale+Processing System	23
Figure 27 - Zynq Ultrascale+ architecture	24
Figure 28 - Validate design	24
Figure 29 - HDL Wrapper	25
Figure 30 - Generating Bitstream	25
Figure 31 - Export hardware including the bitstream	26
Figure 32 - Xilinx SDK	26
Figure 33 - Create FSBL Application Project	27
Figure 34 - Zynq FSBL project	

Figure 35 - Create image	29
Figure 36 - Project settings	30
Figure 37 - Open target	31
Figure 38 - Selecting memory flash part	32
Figure 39 - Vivado design for the platform	33
Figure 41 - SDSoC Platform Project	34
Figure 41 - Import .dsa file	35
Figure 42 - Platform project opened	35
Figure 43 - System Configuration	36
Figure 44 - Domain Configuration	
Figure 45 - Custom Platform added	37
Figure 46 - Create Application Project	37
Figure 47 - Select Custom Platform	
Figure 48 - Application templates	
Figure 49 - Application window, accelerating function	
Figure 50 - Estimation Performance	40

1 Introduction

This document is a guide for those who own an EMC²-DP V2, and gives an overview of both hardware and software capabilities in order to set the board up to be used in any system.

This guide provides basic steps to create simple projects targeting the different Xilinx tools.

2 Hardware

2.1 How is the hardware distributed?

The EMC2-DP is an FMC carrier board, PC/104 form factor, with a SEIC extension board which provides connectivity for different interfaces such as HDMI, SATA, USB, etc.

The EMC2-DP has a 4 x 5 cm form factor socket, compatible with different SoCs, with Xilinx 7 Series, Zynq 7 series, Ultrascale or Zynq Ultrascale+ devices. https://www.sundance.technology/som-cariers/pc104-boards/emc2-dp/



Figure 1 - EMC2-DP

The user might purchase a module with the EMC2-DP. Some examples are:

• 7 Series FPGA example: <u>https://www.sundance.technology/system-on-modules-som/som-modules/te0712-xilinx-artix-7-fpga-som/</u>



Figure 2 - TE0712, Xilinx Artix-7 FPGA SoM

• Zynq 7 series example: <u>https://www.sundance.technology/system-on-modules-som/som-modules/te0715-7030-xilinx-zynq-arm-fpga-som/</u>



Figure 3 - TE0715-7030, Xilinx Zynq ARM + FPGA SoM

• Ultrascale series example: <u>https://www.sundance.technology/system-on-modules-som/som-modules/te0841/</u>



Figure 4 - TE0841, Xilinx Kintex UltraScale SoM

• Zynq Ultrascale+ series example: <u>https://www.sundance.technology/system-on-modules-som/som-modules/teo820-zu4ev/</u>



Figure 5 - TEO820-ZU4EV, Xilinx Zynq UltraScale+

2.2 How can I connect/disconnect the EMC2-DP and the SoM?

Connecting the module seems trivial, but it's recommended to place it on the carrier, and press down from two opposite corners, so that all the pins make contact at the same time, and none get bent.

In order to disconnect the module, follow a similar procedure, pushing up from the bottom of the carrier. Following the steps described in this video helps to avoid any damage to the hardware:

https://www.youtube.com/watch?v=QVOC8dk5n10&feature=youtu.be

2.3 How can I connect the EMC2-DP to the PSU?

The EMC² works well with any power supply which provides the 12V, 3.3V and 5V necessary for the board to work with all its capabilities. The one we recommend is the power supply <u>FSP300-60GHS</u>.



Figure 6 - Power supply

Connect the power supply to the board, using the following cable:





And connect the cable to the board:

Figure 8 - Connecting the EMC² to the PSU

Turn on the PSU connecting it through the power connector and switching it on:



Figure 9 - EMC2-DP fully working

2.4 How can I configure the I/O Voltages?

WARNING: Never configure the IO Voltages with the board powered up!

This board needs an external supply of 3.3V for most of the components on board, as well as 5V and 12V for the PCIe and FMC ports. The IO voltages for the FPGA banks can be selected through the jumpers shown in the picture.



JP7 and JP8 select the different voltages available as follows:



• 3.3V: Position 1-2

Figure 11 - 3.3V selection

• 2.5V: Position 2-JP7A and 2-JP8A



Figure 12 - 2.5V selection

• 1.8V: Position 2-3



Figure 13 - 1.8V selection

JP7 selects the voltage which feeds mainly the HDMI and SEIC related signals (One of the FPGA banks, while JP8 selects the voltage which feeds mainly the FMC related signals (Two other different FPGA banks).

The FPGA banks, in Xilinx devices, are classified as High Range (HR), High Performance (HP), or High Density (HD). HP banks, for instance, only support up to 1.8V, which means that supplying 3.3V to the FPGA would damage the corresponding bank.

Depending on the SoM used with the carrier board, the specifications for the FPGA might differ, and therefore, there is a risk of damaging the device.

Sundance might provide the EMC2-DP only populating 1.8V selection. If it's not the case, **the user is responsible of ensuring that the voltage is correctly set.**

Contact Sundance if support is needed.

2.5 How can I configure the board to use PCIe?

The EMC2-DP can work in host or add/on mode, depending on the application and the needs of the user.

In case of using the EMC2-DP as host, there is one thing the user must do:

• JP12 must be set, as it is related to the "clock select" pin at the PCIe, and it will provide the 100MHz reference clock.

Figure 15 shows the connectivity.

If the board is used on a stack, it can be used in add/on mode, where JP12 should be unconnected.

To select the upstream port, SW2 should be configured as follows:



Figure 14 - Upstream port selection

(PEX) Port 0: (Pcie) Lane 0: 0000(LLLL): All On
(PEX) Port 4: (Pcie) Lane 1: 0100(LHLL): On-Off-On-On
(PEX) Port 1: (Pcie) Lane 4: 0001(LLLH): Off-On-On-On
(PEX) Port 5: (Pcie) Lane 5: 0101(LHLH): On-Off-On-Off
(PEX) Port 7: (Pcie) Lane 6: 0110(LHHL): Off-Off-Off-On
(PEX) Port 9: (Pcie) Lane 7: 0111(LHHH): On-Off-Off-Off

Where the PEX Port and Pcie Lane are the same thing, but called differently. L corresponds to "On" and H to "Off" from SW2.

Here there is an example of how JP12 and SW2 should be to set the board as host:



Figure 15 - JP12 sets the host mode, SW2 selects port 0 as upstream port

2.6 How can I boot from flash or SD card?

The jumper (JP11) is the boot mode for the "flash devices" to be set from the EMC2-DP.



Figure 16 - Boot mode selection

The positions depending on where the user wants to boot from are:

- Position 1-2, QSPI flash mode
- Position 2-3, SD card mode (closer to JP12)

In Figure 17, Next to JP12, JP11 is set as SD booting mode.

2.7 Information about the SEIC Connector

The EMC²-DP has most of its capabilities available at the extension board, accessible through the SEIC connector.



Figure 17 - SEIC extension board

This connector is labelled as J3 and J4 on the extension board and main board respectively.



Figure 18 - SEIC connector

The pinout of the SEIC is represented in this schematic, where the top pins of both J3 and J4 are connected, as well as the bottom pins.



Figure 19 - SEIC Connector Pinout

3 Software

3.1 How can I use Vivado with the EMC²?

The EMC2-DP has been used in multiple versions of Vivado, from 2015, and it's recommended to use always the newer versions.

Board files from <u>Trenz Electronic</u> can be used depending on the module installed on the carrier.

Also, Sundance Multiprocessor Technology LTD provides documentation and resources in GitHub:

https://github.com/SundanceMultiprocessorTechnology/

Board files from Sundance are accessible at "VCS-1_Board_Files" repository:

https://github.com/SundanceMultiprocessorTechnology/VCS-1_Board_Files

Using board files from Trenz allows the user to have automatic configuration for the Zynq device and external DDR memory, whereas using the Sundance board files also provides some default configurations to use interfaces present in the carrier board, such as LEDs, UARTs, I2C devices, etc.

All the pin locations for the SEIC devices and FMC are described in the <u>EMC² Board</u> <u>IO</u> document, or a Master Pinout spreadsheet located in GitHub with the Board Files.

To include the board files in your system, download the corresponding files, and add them at the installation path of Vivado, normally:

<installationpath>/Xilinx/Vivado/20XX.X/data/boards/board_files/

NOTE: all the examples shown in this guide are using Vivado 17.4, SDK 17.4, SDx 17.4, and EMC2-DP + TE0820-4EV

Having Vivado open, create a new project, either on "Create Project", or File \rightarrow New Project:

Elle Elny Toole Window Hele Q+ Ouick Access			~ ~ @
Quick Start create Project > open Project > Open Example Project >	HLA Editors	New Project $\checkmark \land \land$	N.7030
Tasks Manage IP > Open Hardware Manager > Xillinx Tot Store >			
Learning Center Documentation and Tutorials > Tot console Q X + III B Ⅲ ■ Oppin, Victory t	E XILINX ALL PROGRAMMAELE.		2 – 0 G X
Ifen Lakeolski 44-460 (pening he travet localbost III2/slin_ctf Current by decise (att be discuss state) Then Lakeolski 24-460 (pening he travet (pening he discuss state) Ifen (Lakeols 27-460) byte state (TAG decise index = 0) is not p current by decise (att be discuss and decise index = 0) refrest by decise (att be discuss and decise index = 0) current by decise (att be discuss and decise index = 0) current by decise (att be discuss and decise index = 0) current by decise (att be discuss and decise index = 0) current by decise (att be discuss att be discussed of the discuss att be discussed of the discused of the discussed of t	D1 (?) io ogranned (DONE status = 0) lap_1] Ol	< gack <u>Best</u> Evanh Cancel	

Figure 20 - Create a new project

Click "Next" and select the path of the project.

Choose RTL project and mark the square "Do not specify sources at this time" in case the user doesn't require importing any source and it's a blank project.

Then, when Vivado asks for a device part. Select "Boards", and choose the corresponding board files.

oose a default Xilin	x part or board for your project. This	s can be	changed later.		P		
Select: 🛛 👜 Parts	🔤 Boards						
Filter/ Preview			•	23 64			
Vendor:	All	~	- 2				
Display <u>N</u> ame:		~					
Board Re <u>v</u> :	Latest	~					
	Reset All Filters		1				
Coorsh: O-			8		(an filment) (af feighter (and))		
<u>s</u> earch: <u>e</u>	~						
Display Name			Vendor	Board Rev	Part		
📓 Kintex-Ultrascal	e Alphadata board		alpha-data.com	1.0	💷 xcku060-ttva1156-2-e	^	
📓 ZedBoard Zynq	Evaluation and Development Kit		em.avnet.com	d	@ xc7z020clg484-1		
EMC2-DP-V2 + 1	E0715-30-04		sundance.com	1.0	xc7z030sbg485-1		
EMC2-DP-V2 + 1	E0820-3EG-02		sundance.com	1.0	xczu3eg-sfvc784-1-e		
EMC2-DP-V2 + 1	E0820-4CG-02		sundance.com	1.0	😟 xczu4cg-sfvc784-1-e		
EMC2-DP-V2 + 1	E0820-4EV-03		sundance.com	1.0	😟 xczu4ev-sfvc784-1-e		
						>	
Board Connectors			Target Connection	s			
FMC LPC			FM191-RU			\mathbf{v}	
_							

Figure 21 - Board file selection

Note that at "Board Connectors", FMC_LPC is available, to connect compatible FMC boards, like FM191-RU.

Once the project is opened, the user can see the information about the board and the part used:

Project Summary				? 🗆 🖒 X
Settings Edit				î
Project name: Project location: Product family: Project part: Top module name: Target language: Simulator language:	Zynq_FSBL /home/timin/Vivado_Projects/Starters/Zynq_FSBL Zynq UltraScale+ EMC2-DP-V2 + TE0820-4EV-03 (xczu4ev-sfvc784-1-e) Not defined Verilog Mixed			
Board Part				
Display name: Board part name: Connectors: Repository path: URL: Board overview:	EMC2-DP-V2 + TE0820-4EV-03 sundance.com:emc2-dp_te0820_4ev_1e_fmc_lpc_connector_fm19 FMC_LPC ▶ FM191-RU /home;Xilinx/Vxado/2017.4/data/boards/board_files https://www.sundance.technology/som-cariers/pc104-boards/emc The EMC2-DP is a PCIe/104 OneBank™ Carrier for a Trenz compati expansion for a VTA57.1 FMC™ LPC (/0 board and also i/0 pins. TI "Sundance External Interface Connector - SEIC" contains LEDs, F Ethernet and SATA.	1-ru_fmc_lpc_connector:p 2-zu4ev/ ble SoC Module and has he add-on board, called IS232, USB2.0, HDMI, 1Gb	part0:1.0	
Synthesis		Implementation		
Status: Messages: Part: Strategy: Report Strategy:	Not started No errors or warnings xczu4ev-sfvc784-1-e Vivado Synthesis Defaults Vivado Synthesis Default Reports	Status: Messages: Part: Strategy: Report Strategy:	Not started No errors or warnings xczu4ev-sfvc784-1-e Vivado Implementation Defaults Vivado Implementation Default Reports	

Figure 22 - Project information in Vivado

As soon as the user creates a new block design in IP Integrator, there will be some interfaces available, already set up and ready to use:

Sources Design Signals Board × ? _ Q ₹ ♦ ♦ ↓<	OCK DESIGN - design_1
Q X Image: Clock Sources (0 out of 4 connected) Image: Clock Sources (0 out of 4 connected) AD0 SMA Input Image: AD0 SMA Input AD1 SMA Input Image: AD1 SMA Input SI5338 Clock Outputs Image: Sis338 Clock Outputs FMC (0 out of 6 connected) Image: Image: AD0 SMA Input SI5338 Clock Outputs Image: Imag	ources Design Signals Board × ? _ 🗆 [
 EMC2-DP-V2 + TE0820-4EV-03 Clock Sources (0 out of 4 connected) AD0 SMA Input AD1 SMA Input J5 SMA Input SI5338 Clock Outputs FMC (0 out of 6 connected) ADC UART (fmc_lpc_connector) DIOs on FM191 (fmc_lpc_connector) FMC I2C Bus (fmc_lpc_connector) GPIOs on FM191 (fmc_lpc_connector) UEDs on FM191 (fmc_lpc_connector) UART RQ Interrupts (fmc_lpc_connector) O UART RQ Interrupts (fmc_lpc_connector) O UART RQ Interrupts or Outputs (0 out of 3 connected) VTTL O n-board LEDs PHY LEDS Ethernet Port 	$Q \mid \mathbf{X} \mid \Leftrightarrow \mid \mathbf{\otimes} \mid p^{\oplus} \mid p^{\mathbb{N}} \mid$
 Clock Sources (0 out of 4 connected) AD0 SMA Input AD1 SMA Input J5 SMA Input Si5338 Clock Outputs FMC (0 out of 6 connected) ADC UART (fmc_lpc_connector) DIOs on FM191 (fmc_lpc_connector) FMC I2C Bus (fmc_lpc_connector) FMC I2C Bus (fmc_lpc_connector) GPIOs on FM191 (fmc_lpc_connector) UEDs on FM191 (fmc_lpc_connector) UART RQ Interrupts (fmc_lpc_connector) UART RQ Interrupts (fmc_lpc_connector) On-board LEDs PHY LEDS Ethernet Port Serial Interfaces (0 out of 1 connected) 	EMC2-DP-V2 + TE0820-4EV-03
 AD0 SMA input AD1 SMA input J5 SMA input Si5338 Clock Outputs FMC (0 out of 6 connected) ADC UART (fmc_lpc_connector) DIOs on FM191 (fmc_lpc_connector) FMC I2C Bus (fmc_lpc_connector) FMC I2C Bus (fmc_lpc_connector) GPIOs on FM191 (fmc_lpc_connector) LEDs on FM191 (fmc_lpc_connector) UART RQ Interrupts (fmc_lpc_connector) General Purpose Inputs or Outputs (0 out of 3 connected) LVTTL On-board LEDs PHY LEDS Ethernet Port Serial Interfaces (0 out of 1 connected) 	Clock Sources (0 out of 4 connected)
 AD1 SMA input J5 SMA input Si5338 Clock Outputs FMC (0 out of 6 connected) ADC UART (fmc_lpc_connector) DIOs on FM191 (fmc_lpc_connector) FMC I2C Bus (fmc_lpc_connector) FMC I2C Bus (fmc_lpc_connector) GPIOs on FM191 (fmc_lpc_connector) LEDs on FM191 (fmc_lpc_connector) UART RQ interrupts (fmc_lpc_connector) General Purpose Inputs or Outputs (0 out of 3 connected) LVTTL On-board LEDs PHY LEDS Ethernet Port Serial Interfaces (0 out of 1 connected) 	O AD0 SMA Input
 J5 SMA Input SI5338 Clock Outputs SFMC (0 out of 6 connected) ADC UART (fmc_lpc_connector) DIOs on FM191 (fmc_lpc_connector) FMC I2C Bus (fmc_lpc_connector) GPIOs on FM191 (fmc_lpc_connector) LEDs on FM191 (fmc_lpc_connector) UART RQ Interrupts (fmc_lpc_connector) UART RQ Interrupts (fmc_lpc_connector) General Purpose Inputs or Outputs (0 out of 3 connected) LVTTL On-board LEDs PHY LEDS Ethernet Port Serial Interfaces (0 out of 1 connected) 	○ AD1 SMA Input
 ◇ Si5338 Clock Outputs ✓ ➡ FMC (0 out of 6 connected) ◇ ADC UART (fmc_lpc_connector) ◇ DIOs on FM191 (fmc_lpc_connector) ◇ FMC I2C Bus (fmc_lpc_connector) ◇ GPIOs on FM191 (fmc_lpc_connector) ◇ LEDs on FM191 (fmc_lpc_connector) ◇ UART RQ Interrupts (fmc_lpc_connector) ◇ UART RQ Interrupts or Outputs (0 out of 3 connected) ◇ LVTTL ◇ On-board LEDs ◇ PHY LEDS Ethernet Port ✓ ➡ Serial Interfaces (0 out of 1 connected) 	O J5 SMA Input
 FMC (0 out of 6 connected) ADC UART (fmc_lpc_connector) DIOs on FM191 (fmc_lpc_connector) FMC I2C Bus (fmc_lpc_connector) GPIOs on FM191 (fmc_lpc_connector) LEDs on FM191 (fmc_lpc_connector) UART RQ Interrupts (fmc_lpc_connector) UART RQ Interrupts or Outputs (0 out of 3 connected) LVTTL On-board LEDs PHY LEDS Ethernet Port Serial Interfaces (0 out of 1 connected) 	O Si5338 Clock Outputs
 ADC UART (fmc_lpc_connector) DIOs on FM191 (fmc_lpc_connector) FMC I2C Bus (fmc_lpc_connector) GPIOs on FM191 (fmc_lpc_connector) LEDs on FM191 (fmc_lpc_connector) UART RQ Interrupts (fmc_lpc_connector) UART RQ Interrupts or Outputs (0 out of 3 connected) LVTTL On-board LEDs PHY LEDS Ethernet Port Serial Interfaces (0 out of 1 connected) 	FMC (0 out of 6 connected)
 ◇ DIOS on FM191 (fmc_lpc_connector) ◇ FMC I2C Bus (fmc_lpc_connector) ◇ GPIOS on FM191 (fmc_lpc_connector) ◇ LEDs on FM191 (fmc_lpc_connector) ◇ UART RQ Interrupts (fmc_lpc_connector) ◇ UART RQ Interrupts or Outputs (0 out of 3 connected) ◇ LVTTL ◇ On-board LEDs ◇ PHY LEDS Ethernet Port ✓ Serial Interfaces (0 out of 1 connected) 	► ADC UART (fmc_lpc_connector)
 FMC I2C Bus (fmc_lpc_connector) GPIOs on FM191 (fmc_lpc_connector) LEDs on FM191 (fmc_lpc_connector) UART RQ Interrupts (fmc_lpc_connector) General Purpose Inputs or Outputs (0 out of 3 connected) LVTTL O n-board LEDs PHY LEDS Ethernet Port Serial Interfaces (0 out of 1 connected) 	🔊 DIOs on FM191 (fmc_lpc_connector)
 ◇ GPIOs on FM191 (fmc_lpc_connector) ◇ LEDs on FM191 (fmc_lpc_connector) ◇ UART RQ Interrupts (fmc_lpc_connector) ◇ General Purpose Inputs or Outputs (0 out of 3 connected) ◇ LVTTL ◇ On-board LEDs ◇ PHY LEDs Ethernet Port > Serial Interfaces (0 out of 1 connected) 	FMC I2C Bus (fmc_lpc_connector)
 ◇ LEDs on FM191 (fmc_lpc_connector) ◇ UART RQ Interrupts (fmc_lpc_connector) ✓ a General Purpose Inputs or Outputs (0 out of 3 connected) ◇ LVTTL ◇ On-board LEDs ◇ PHY LEDs Ethernet Port ✓ a Serial Interfaces (0 out of 1 connected) 	🔘 GPIOs on FM191 (fmc_lpc_connector)
 ◇ UART RQ Interrupts (fmc_lpc_connector) ✓ a General Purpose Inputs or Outputs (0 out of 3 connected) ◇ LVTTL ◇ On-board LEDs ◇ PHY LEDS Ethernet Port ✓ a Serial Interfaces (0 out of 1 connected) 	▶ LEDs on FM191 (fmc_lpc_connector)
 General Purpose Inputs or Outputs (0 out of 3 connected) LVTTL On-board LEDs PHY LEDs Ethernet Port Serial Interfaces (0 out of 1 connected) 	OUART RQ Interrupts (fmc_lpc_connector)
 ▷ LVTTL ▷ On-board LEDs ▷ PHY LEDs Ethernet Port ~ Serial Interfaces (0 out of 1 connected) 	General Purpose Inputs or Outputs (0 out of 3 connected)
 ◇ On-board LEDs ◇ PHY LEDs Ethernet Port ✓ Serial Interfaces (0 out of 1 connected) 	IVTTL
♥ PHY LEDs Ethernet Port ♥ Serial Interfaces (0 out of 1 connected)	心 On-board LEDs
✓	ю PHY LEDs Ethernet Port
	Serial Interfaces (0 out of 1 connected)
N RS232	€ RS232

Figure 23 - Board interfaces in IP Integrator

The interfaces available are those ones connected to the FPGA. To use the rest of the capabilities within the PS, they can accessed through the MIO/EMIO pins.

As an example, to select one of the outputs of the clock synthesizer (Si5338), just double click on Si5338 Clock Outputs, at Clock Sources:



Figure 24 - How to use interfaces in IPI

Select "Si5338 Output CLK0" as component mode, and "Clocking Wizard" as the desired IP. Automatically, the input will be assigned to the IP selected.



Figure 25 - Block generated through the interface

Remember that these capabilities are made so that the user can access easily to them without having to constrain them.

3.2 How can I create a Zynq FSBL?

To create a FSBL is necessary to determine the hardware of the system (PS + PL) with Vivado, and export that hardware design to the SDK environment, where it's possible to create an FSBL, which boots the PS, mapping the blocks in the PL into memory (if any).

Following the steps described in the previous section, create a project, having a blank block design in IP Integrator.

Add a Zynq Ultrascale+ MPSoC IP block, and connect *pl_clk0* to *maxihpm0_lpd_aclk*.

Alternatively, with tcl commands:

```
startgroup
create_bd_cell -type ip -vlnv xilinx.com:ip:zynq_ultra_ps_e:3.1
zynq_ultra_ps_e_0
endgroup
connect_bd_net [get_bd_pins zynq_ultra_ps_e_0/pl_clk0] [get_bd_pins
zynq_ultra_ps_e_0/maxihpm0_lpd_aclk]
```



Figure 26 - Zynq Ultrascale+Processing System

Then, click Run Block Automation, and press "OK".

```
apply_bd_automation -rule xilinx.com:bd_rule:zynq_ultra_ps_e -config
{apply_board_preset "1" } [get_bd_cells zynq_ultra_ps_e_0]
```

This will automatically configure the Processing System, as per defined in the board files.



Figure 27 - Zynq Ultrascale+ architecture

It is highly recommended to read the Zynq Ultrascale+ documentation from Xilinx to understand the architecture and be able to configure the device properly.

Click "Validate design" and then "OK"



Figure 28 - Validate design

Create a Wrapper of the design. To do so, go to the "Sources" tab, and right click on the design \rightarrow Create HDL Wrapper



Figure 29 - HDL Wrapper

Generate the Bitstream, tool easy to find in the Flow Navigator.

Bitstream Generation Completed	\otimes				
Bitstream Generation successfully completed.					
Next					
Open Implemented Design					
○ <u>V</u> iew Reports					
🔘 Open <u>H</u> ardware Manager					
O <u>G</u> enerate Memory Configuration File					
Don't show this dialog again					
OK Cancel					

Figure 30 - Generating Bitstream

Export the hardware. Select "File -> Export -> Export Hardware"

4	Export Hardware								
Export hardware platform for software development tools.									
🗹 <u>I</u> nclude b	itstream								
<u>E</u> xport to:	👼 <local proje<="" td="" to=""><th>ect> 🗸</th></local>	ect> 🗸							
?	ОК	Cancel							

Figure 31 - Export hardware including the bitstream

SOK	Zynq	_FSBL.sdk - C/C++ - design_1	_wrapper_hw_pla	atform_0/system	n.hdf - Xilinx SDK			
File Edit Navigate Search Project Run Xilinx Wir	dow Help							
🖻 ~ 🖩 🐚 💩 ~ 🗞 ~ 🔌 🖸 🍀 🗖	■ 🚱 🏘 ~ O ~	∦ ∨ ♥ ∲ ∨ ¢	~					Quick A
Project Explorer 🛛 📄 😫 🔻 🗢 🗖	system.hdf 🛿							- 6
design_1_wrapper_hw_platform_0	design_1_wrapp	er_hw_platform_0 H	lardware P	latform Sp	ecification			
g psu_init_gpl.c	Design Information							
📓 psu_init_gpl.h	Target FPGA Device:	xczu4ev						
🕼 psu_init.c	Part:	xczu4ev-sfvc784-1-e						
📓 psu_init.h	Created With:	Vivado 2017.4						
su_init.html	Created On:	Wed Feb 20 18:34:22 2019						
📝 psu_init.tcl	Address Map for proc	essor psu_cortexa53_[0-3]						
	·							
	Cell	Base Addr	High Addr	Slave I/f	Mem/Reg	Segment	TrustZone	AccessType
	psu_gdma_1	0xfd510000	0xfd51ffff		REGISTER		NonSecure	Read/Write
	psu_gdma_2	0xfd520000	0xfd52ffff		REGISTER		NonSecure	Read/Write
	psu_gdma_3	0xfd530000	0xfd53ffff		REGISTER		NonSecure	Read/Write
	psu_crf_apb	0xfd1a0000	0xfd2dffff		REGISTER		NonSecure	Read/Write
	psu_gdma_4	0xfd540000	0xfd54ffff		REGISTER		NonSecure	Read/Write
	psu_adma_2	0xffaa0000	0xffaaffff		REGISTER		NonSecure	Read/Write

Figure 32 - Xilinx SDK

To create a FSBL, go to File -> New -> Application Project

ok New Project 🗸 🗸 🕅						
Application Project	t ke application project.					
Project name: FSBL						
📒 Use default locati	on					
Location: /home/tin	nin/Vivado_Projects/Starters/Zynq_FSBL/Zyi Browse					
Choose file	system: default 🗸					
OS Platform: stand	dalone v					
Target Hardware						
Hardware Platform:	design_1_wrapper_hw_platform_0 v New					
Processor:	psu_cortexa53_0 v					
Target Software Language:	C ○ C++					
Compiler:	64-bit V					
Hypervisor Guest:	No Y					
Board Support Pack	age: 🔘 Create New 🛛 FSBL_bsp					
	Use existing V					
?	< Back Next > Cancel Finish					

Figure 33 - Create FSBL Application Project

Give a name to the project, for example "FSBL". It's intuitive, as it shows the hardware platform (HDL Wrapper exported previously), the ARM core selected from the Zynq, and the OS platform, although in this case it's a standalone application.

Press next and select "Zynq MP FSBL"



Figure 34 - Zynq FSBL project

Press "Finish" and the FSBL project will be added.

Right click on the project, and build it. That should generate an FSBL.elf file under "Debug", which can be used as bootloader.

Now there is an FSBL that can be used for initialize the PS. It's recommended to check out the file system.mss from the BSP project, because depending on the hardware exported, the user can find different code examples from Xilinx for the different peripherals included in the Zynq processing system (UART, DDR, I2C, USB, etc.).

3.3 How can I create SD boot file?

Assuming and FSBL project has been created for any hardware design:

Click FSBL -> Create Boot Image

The user can now see that SDK tries to export a .bif file, from the .elf file (FSBL project in SDK) and the .bit file (Vivado bitstream file).

SDK	Ci	reate Boot Ima	age		\sim \sim \otimes
Create Boot Image Creates Zynq MP Boot	e I Image in .bin format from given FSBI	L elf and partiti	on files in specified ou	tput folder.	
Architecture: Zynq M	P ~				
Basic Security	Import from existing BIF file				
Output BIF file path:	/home/timin/Vivado_Projects/Starte	ers/Zynq_FSBL/	/Zynq_FSBL.sdk/FSBL/	/bootimage/FSBL.bif	Browse
UDF data:					Browse
Split	Output format: BIN 🗸				
Output path:	/home/timin/Vivado_Projects/Starte	ers/Zynq_FSBL/	/Zynq_FSBL.sdk/FSBL/	'bootimage/BOOT.bin	Browse
Boot image partitions					
File path		Encrypted	Authenticated		Add
(bootloader) /home/t /home/timin/Vivado_	imin/Vivado_Projects/Starters/Zynq_ Projects/Starters/Zynq_FSBL/Zynq_F	none none	none		Delete
?		Preview	BIF Changes	Cancel	ate Image

Figure 35 - Create image

Press "Create Image".

Now at, *<projectpath>/<projectname>/projectname.sdk/FSBL/bootimage/* there should be a .bin file called "BOOT.bin" and a .bif file called "FSBL.bif"

This BOOT.bin file can be placed in a SD card, and selecting the jumpers on the EMC2-DP to boot from SD, described in this document, the Zynq device will be configured when powering up the board.

Remember that in this case it's assumed that the project run in the Zynq is just the FSBL itself. The PS needs to be always booted in order to run any application. To do so, there should be 3 files merged into BOOT.bin: FSBL.elf (bootloader), .bit file (PL) and .elf file (application).

3.4 How can I program the FPGA from Flash in Vivado?

To program the FPGA from flash, the user needs to erase the flash and program the FPGA with a .bin or .mcs file.

Using the FSBL project in Vivado as example:

This project has a .bit file generated, but to program the flash a .bin file is needed. To let Vivado know that, it's possible to do it going to "Project Settings" in the Vivado Flow Navigator, and mark the .bin option in Bitstream options.

0.			
Project Settings	 Bitstream Specify various settings related to writing bitstream 		- 🏊
General			
Simulation	(i) Note: Additional bitstream settings will be available once yo	u open an implem	nented de
Elaboration			
Synthesis	✓ Write Bitstream (write_bitstream)		
Implementation	tcl.pre		
Bitstream	tcl.post		
> IP	-raw_bitfile		
Tool Settings	-mask_file		
Project	-no_binary_bitfile		
IP Defaults	-bin_file		
Source File	-readback_file		
Display	-logic_location_file		
WebTalk	-verbose		
Help	More Options		
 Text Editor 3rd Party Simulators Colors Selection Rules Shortcuts Strategies Remote Hosts Window Behavior 	-bin_file Write a binary bit file without header (.bin).		
?	OK Cancel	<u>Apply</u>	<u>}</u> estore

Figure 36 - Project settings

Now rewrite the bitstream, and open the hardware manager.

Alternatively, the flash memory **can be programmed with an .mcs file**. It can be generated in SDK in the same way the BOOT.bin file is generated for SD booting. Select MCS instead of BIN at the file format options.

Connect the EMC2-DP to the computer through JTAG, and power the board up. Open a new target, and the EMC2-DP will be recognized by the software tool:

A		Оре	en New Hardware Targ	et		\sim \sim \otimes
Select Hardw Select a hardwa clock (TCK) frequ	vare Target re target from t uency. If you do	he list of av not see the	vailable targets, then e expected devices, d	set the appropri ecrease the frec	iate JTAG quency or	
Hardware <u>T</u> a	rgets					
Туре	Name		JTAG Clock Frequency	/		
📓 xilinx_tcf	Digilent/21029	9A56598	15000000	1		
Hardware <u>D</u> e	evices (for unk	Add	Xilinx Virtual Cable (XV	/C) struction Regis	ster (IR) lengt	th)
Name	ID Code	IR Length				
<pre>@ xczu4_0</pre>	04721093	12				
<pre>@ arm_dap_;</pre>	1 5BA00477	4				
Hardware sen	ver: localhost:3	121	Seck N	ext >	inish	Cancel
\bigcirc						

Figure 37 - Open target

The user can see at this point that the hardware manager sees the PL and the PS from the Zynq (FPGA + ARM). Press "Next" and "Finish".

Now, to program the Flash, right click on the FPGA, and select "Add Configuration Memory Device"

Select the right flash memory part, and then, when it's part of the JTAG chain, right click on the flash memory, and select "Program Configuration Memory Device"

A	Program Configuration Memory Device	$\sim \otimes$
Select a configuration	on file and set programming options.	4
Memory Device: <u>C</u> onfiguration file:	mt25qu512-qspi-x8-dual_parallel /home/timin/Vivado_Projects/Starters/Zynq_FSBL_C/Zynq_FSBL_C.sdk/FSBL/bootimage/BOOT.mcs	
Zynq FSBL: /hom Program Operat	e/timin/Vivado_Projects/Starters/Zynq_FSBL_C/Zynq_FSBL_C.sdk/FSBL/Debug/FSBL.elf 🛛 🔊	
Address Range ✓ <u>E</u> rase <u>B</u> lank Check ✓ P <u>r</u> ogram	Configuration File Only	
✓ ⊻erify	OK Cancel App	ly

Figure 38 - Selecting memory flash part

Before pressing "OK", **JP11 should not be in SD boot mode**.

3.5 How can I create an SDSoC Platform?

The steps explained in order to create this platform can be used for any other platforms, changing the paths and names specified. For more detailed information about platforms and libraries, check these documents from Xilinx: UG1146 and UG1236

This platform has been made for a standalone application.

Create a project in Vivado, using IPI. The output clocks of the clock wizard are 100MHz, 200MHz and 300MHz respectively.

The project shown as example has been called "EMC2DP_TE0820_4CG_SDxPlatform"



Figure 39 - Vivado design for the platform

Create a wrapper, generate the bitstream.

In the tcl console, introduce the following commands:

This command specifies the name of the platform, and where is the block design set_property PFM_NAME "sundance.com:EMC2DP_TE0820_4CG:EMC2DP_TE0820_4CG:1.0" [get_files ./EMC2DP_TE0820_4CG_SDxPlatform.srcs/sources_1/bd/design_1/design_1.bd]

Declare the available clocks

set_property PFM.CLOCK { clk_out1 {id "0" is_default "true" proc_sys_reset "proc_sys_reset_0"
} clk_out2 {id "1" is_default "true" proc_sys_reset "proc_sys_reset_1" } clk_out3 {id "2"
is_default "true" proc_sys_reset "proc_sys_reset_2" } } [get_bd_cells /clk_wiz_0]

Declare the AXI available ports for SDSoC to use.

set_property PFM.AXI_PORT { M_AXI_HPM0_FPD {memport "M_AXI_GP"} M_AXI_HPM1_FPD {memport
"M_AXI_GP"} M_AXI_HPM0_LPD {memport "M_AXI_GP"} S_AXI_HPC0_FPD {memport "S_AXI_HPC" sptag
"HPC0"} S_AXI_HPC1_FPD {memport "S_AXI_HPC" sptag "HPC1"} S_AXI_HP0_FPD {memport "S_AXI_HP"
sptag "HP0"} S_AXI_HP1_FPD {memport "S_AXI_HP" sptag "HP1"} S_AXI_HP2_FPD {memport "S_AXI_HP"
sptag "HP2"} S_AXI_HP3_FPD {memport "S_AXI_HP" sptag "HP3"} } [get_bd_cells
/zynq_ultra_ps_e_0]

Declare the interrupts available set intVar []

for {set i 0} {\$i < 8} {incr i} {
lappend intVar In\$i {}
}
set_property PFM.IRQ \$intVar [get_bd_cells /xlconcat_0]</pre>

Specify local cache.
set_property dsa.ip_cache_dir [get_property ip_output_repo [current_project]]
[current_project]

Generate .dsa file write_dsa EMC2DP_TE0820_4CG.dsa -include_bit -force

After this is done, export the hardware, launch SDK, and create an FSBL project, and a Hello World project (it is done in the same way as an FSBL, selecting the corresponding Hello World example).

Build both projects.

Having a .dsa file, create a new project in SDSoC:

Æ	New SDx Project	~ ^ 😣
Project Type		
Choose the project type to create.		
Application Project		
A user application targeting SDSoC, SDAccel or SDK flo	w	
System Project		
Consistent way to manage multiple applications and lik	braries for a system configuration	
Platform Project		
?	< Back Next > Fir	nish Cancel

Figure 40 - SDSoC Platform Project

Specify the path where the .dsa file is stored:

5	New SDx Project	\sim \sim \otimes
Platform Specification		
Specify the DSA and platform	options	
Device Support Archive (DSA):	/home/timin/Vivado_Projects/Starters/EMC2DP_TE0820_4CG_SDxPlatform/EMC2DP_TE08	Browse
Import software platform	components	
O Build software platform co	omponents (Beta)	
?	< Back Next > Finish	Cancel

Figure 41 - Import .dsa file

Create the project.

5		EMC2DP_TE0820_4CG_SDxPlatform -	SDx - EMC2DP_TE0820_4CG/platform.spr - Xilinx SDx	
<u>F</u> ile <u>E</u> dit <u>N</u> avigate	Se <u>a</u> rch <u>P</u> roject <u>R</u> un <u>X</u> ilin	nx <u>W</u> indow <u>H</u> elp		
	~ % ~ * ~ O ~			
ြဲ Project Explorer 🛿	- 0	✓ EMC2DP_TE0820_4CG X	-	
	🗆 😫 🔻 👻	type filter text 🛛 🗈 🕀 🐥 🚿	Platform: EMC2DP_TE0820_4CG	
✓	_4CG	EMC2DP_TE0820_4CG	Name: EMC2DP_TE0820_4CG	
🗸 🗁 dsa			DSA: EMC2DP_TE0820_4CG.dsa	
EMC2DP	_TE0820_4CG.dsa		Description: EMC2DP_TE0820_4CG	
> 🗁 logs				
🗁 resources				
🛹 platform.spr			Samples:	
			Steps to Generate a Platform	
✓ Reports 🖾			1 Define System Configuration 2 Add Processor Group/Define System Configuration	omain
				— J

Figure 42 - Platform project opened

There are some icons at the platform window. Clicking on the "+" icon, add a new System Configuration. The paths for the boot and .bif files are the FSBL project created in SDK.

<i></i>	New System Configuration	~	^	⊗				
System Confi	System Configuration							
😣 A configurati	on already exists with the same name							
Name:	sysConfig							
Display Name:	sysConfig							
Description:	Defines the configuration							
Boot Directory:	ters/Zynq_FSBL/Zynq_FSBL.sdk/FSBL/bootimage	Browse		R				
Bif File:	_4CG_SDxPlatform.sdk/FSBL/bootimage/FSBL.bif	Browse		R				
?	OK	Cano	el					

Figure 43 - System Configuration

Do the same with a new Domain.

5	New Domain in 'sysConfig'	\sim \sim \otimes
Domain		
Create a new domain		
Name:	Domain1	
Display Name:	Domain1	
OS:	standalone 🗸	
Processor:	psu_cortexa53_0 v	
Supported Runtimes:	C/C++ ~	
Linker Script:	2DP_TE0820_4CG_SDxPlatform.sdk/FSBL/src/lscript.lc Brows	se 🖹
Description:	Specifies Standalone	
?	ОК	Cancel



On "Domain", add at "Repositories", a path where examples from Xilinx are stored. They can be downloaded from GitHub:

https://github.com/Xilinx/SDSoC_Examples

Under "Domain", there are some paths to include.

- At "Application Settings", tell the tool where the linker script is. (Hello World Application path)
- At "Board Support Package", include the path of the Hello World .bsp folder
- At "Libraries", include the lib folders of both FSBL and Hello World projects.

The icon that looks like a hammer, (Platform icon), click and select "Generate". This will generate the platform files.

On the same icon, select "Add To Custom Repositories", and the platform will be added to SDSoC.



Figure 45 - Custom Platform added

Create a new project, and this time select Application Project.



Figure 46 - Create Application Project

Select the custom platform.

10			New SDx P	roject			~ ^ 😣
Platform							
The platform defines the hard	ware that will execute your applicat	ion.					
Platforms (5) <u>Filter</u>							
Find:	•						
Name	Board	Family	Part	Version	Vendor	Flow	Location
EMC2DP_TE0820_4CG [c	ustom] EMC2DP_TE0820_4CG	zynquplus	xczu4cg	1.0	sundance.com	SDSoC	/home/timin/Vivado_Projects/Starters/EMC2D
📓 zc702	zc702	zynq	xc7z020	1.0	xilinx.com	SDSoC	\$XILINX_SDX/platforms/zc702/zc702.xpfm
📓 zc706	zc706	zynq	xc7z045	1.0	xilinx.com	SDSoC	\$XILINX_SDX/platforms/zc706/zc706.xpfm
📓 zcu102	zcu102	zynquplus	xczu9eg	1.0	xilinx.com	SDSoC	\$XILINX_SDX/platforms/zcu102/zcu102.xpfm
📓 zed	zed	zynq	xc7z020	1.0	xilinx.com	SDSoC	\$XILINX_SDX/platforms/zed/zed.xpfm
Description	Add Custom Pla	tform) Ma	nage Repos	itories	Add Devices/Pla	forms	
EMC2DP_TE0820_4CG Repository:/EMC2DP_	TE0820_4CG/export/EMC2DP_TE082	10_4CG					
?					<	Back	Next > Finish Cancel
			~				

Figure 47 - Select Custom Platform

The repositories included now appear as possible application templates. Select the matrix multiplication example to accelerate in hardware.

	New SDx Project v ^
mplates	
elect a template to create your project.	
vailable Templates:	
ind: 🛛	02_mmult_hw
Empty Application	This example shows how to perform matrix multiplication on hardware
<pre>/ cpp</pre>	without specifying any optimizations (the same software function is taken
✓ Getting Started Examples	and just targeted to hardware). This results in default random data access
✓ CPU_To_FPGA Examples	compared with the software results for verification and speedup is shown
01_mmult_sw	withrespect to running on hardware versus software.
02_mmult_hw	Keywords:
03_mmult_pipeline	• #pragma HI S loop_tripcount
04_mmult_zero_copy	"pragma neb roop_upcom
05_mmult_array_partition	
Array Partitioning	
Burst Read/Write	
Custom Data Type	
Direct Connection	
DMA SG(scatter-Gather)	
DMA Simple	
Dx Examples SDx Libraries	
	Cancel
	Cancel

Figure 48 - Application templates

An application project is created, and ready to build. Notice that under "Hardware Functions", the user can select which functions will be accelerated in hardware. (PL)

<pre>// EMC2DP_TE0820_40</pre>	G 🕺 Example 🕱		- 0
🔀 Application P	roject Settings	Active build configuration: Debug 🗸	89
General		Options	
Project name:	Example	Data motion network clock frequency (MHz): 300.00 \checkmark	
Project flow:	<u>SDSoC</u>	Generate emulation model Debug V	
Platform: Runtime:	EMC2DP_TE0820_4CG C/C++	Generate bitstream	
System configuratio	n: sysConfig	Generate SD card image	
Domain:	Domain1	Insert AXI performance monitor	
OS:	standalone	Enable event tracing	
		Estimate performance	
		Root function: main	
Hardware Function	S	A 193	×
Name	Clock Frequency (MHz) Path		
🕖 mmult_hw	300.00 src/mmult_accel.cpp		

Figure 49 - Application window, accelerating function

At the Application Project window, select "Release" at "Active build configuration", and, if desired, select "Estimate Performance".

Building might take a long time, as synthesis and implementation is done. In order to avoid this waiting time, prebuilt files can be set up for the custom platform.

Build the project, and bootable files to place in SD card will be generated.

For Linux hosts, it might happen that "gmake" is not installed by default. In order to solve it, just link make to gmake:

sudo ln -s /usr/bin/make /usr/bin/gmake

As it can be seen:

Created: 2	1 Feb 2019 12:39				
<mark>lick Here</mark> to get s	oftware-only applicat	tion performanc	e and speedup		
ote: Performance e	stimation assumes wor	st-case latency of l	Hardware functions,	it also assumes wo	orst-case dat
insfer size for array	s (if transfer size canno	t be determined at	compile time). If th	e Hardware functio	n latency an
ta transfer size at r	un-time is smaller than	such assumptions	s, the performance e	stimation will be m	ore pessimi
an the actual perfo	rmance.				
etails					
erformance estin	nates for 'mmult_hwi	in main.cpp:102	func		
lardware accelerate	ed (Estimated cycles)			1090492	
lardware accelerate	ed (Estimated cycles)			1080482	2
lardware accelerate	ed (Estimated cycles)			1080482	2
lardware accelerate	ed (Estimated cycles) on estimates for Hard	ware functions		1080482	2
lardware accelerato esource utilizatio Resource	ed (Estimated cycles) on estimates for Hard Used	ware functions Total	% Utilization	1080482	2
ardware accelerato esource utilizatio Resource DSP	ed (Estimated cycles) on estimates for Hard Used 4	ware functions Total 728	% Utilization 0.55	1080482	2
esource utilization Resource DSP BRAM	ed (Estimated cycles) on estimates for Hard Used 4 24	ware functions Total 728 128	% Utilization 0.55 18.75	1080482	2
esource utilization Resource DSP BRAM LUT	ed (Estimated cycles) on estimates for Hard Used 4 24 13148	ware functions Total 728 128 87840	% Utilization 0.55 18.75 14.97	1080482	2

Figure 50 - Estimation Performance

3.6 How can I create/run Sundance demos?

Sundance has developed some demos, using HDMI in/out FMC cards, HDMI output interface from the carrier, Camera Link, Applications for robotics applications using VCS-1, etc., for different architectures as Zynq 7 series or Zynq Ultrascale+, which can be found at:

Hackaday projects: <u>https://hackaday.io/SundanceDotCom</u>

GitHub: <u>https://github.com/SundanceMultiprocessorTechnology</u>

Contact Sundance for support