VadaTech FMC228

FPGA Reference Design Manual

May 24, 2017

Version 1.2.1

vadatech inc

THE POWER OF VISION

Notice

While reasonable efforts have been made to assure the accuracy of this document, VadaTech, Inc. assumes no liability resulting from any omissions in this document or from the use of the information obtained herein. VadaTech reserves the right to revise this document and to make changes periodically and the content hereof without obligation of VadaTech to notify any person of such revision or changes.

Electronic versions of this material may be read online, downloaded for personal use, or referenced in another document as a URL to the VadaTech Incorporated Web site. The text itself may not be published commercially in print or electronic form, edited, translated, or otherwise altered without the permission of VadaTech, Inc.

It is possible that this publication may contain reference to or information about VadaTech products (machines and programs), programming, or services that are not available in your country. Such references or information must not be construed to mean that VadaTech intends to announce such products, programming, or services in your country.

Revision History

Doc Rev	Description of Change	Revision Date
1.2.1	Added AMC535 and AMC536 carrier support in Quartus 16.2.1	5/24/2017
1.2.0	Added AMC593 carrier support. Updated Vivado design to 2016.4	5/9/2017
1.1.1	1. Corrected ADC signal name in section 3.5.7	6/6/2016
1.1.0	1. Added ordering option B=1 support in Vivado reference designs (AMC516 and AMC502)	5/19/2016
1.0.2	1. Removed blank page from document + fixed typo	5/18/2016
1.0.1	1. Added Altera support for amc532_xxx_22x_xxx_fmc228	5/18/2016
1.0.0	1. Split out manual from original AMC516 FPGA reference design user manual 2. Changed amc516_x1x_22x-xxx_fmc228 reference design to amc516_fmc228. Added ORDERING_OPTION_B generic in the top module to automatically initialize the board based on the option 3. Added amc502_fmc228 reference design	4/14/2016

Table of Contents

List of Figures

List of Tables

1 Document Overview

This document describes the available VadaTech FPGA carrier board reference designs for the FMC228 board. The reference designs demonstrate the FMC228's functionalities using one or more of VadaTech's FPGA carrier boards. These reference designs facilitate factory testing, customer acceptance testing, hardware debugging, and act as references for further customer development.

FPGA/software development on the customer's FPGA carrier board is expected to be performed at the customer's site to add any additional application-specific functionality to the FMC228 board.

1.1 Applicable Products

- VadaTech FMC228

1.2 Document References

- VadaTech AMC516 Datasheet (http://www.vadatech.com)
- VadaTech AMC516 Hardware Reference Manual (http://www.vadatech.com)
- VadaTech AMC516 FPGA Reference Design Manual (http://www.vadatech.com)
- VadaTech AMC516 Software User Manual (http://www.vadatech.com)
- VadaTech AMC502 Datasheet (http://www.vadatech.com)
- VadaTech AMC502 Hardware Reference Manual (http://www.vadatech.com)
- VadaTech AMC502 FPGA Reference Design Manual (http://www.vadatech.com)
- VadaTech AMC502 Software User Manual (http://www.vadatech.com)
- VadaTech AMC532 Datasheet (http://www.vadatech.com)
- VadaTech AMC532 Hardware Reference Manual (http://www.vadatech.com)
- VadaTech AMC532 FPGA Reference Design Manual (http://www.vadatech.com)
- VadaTech AMC535 Datasheet (http://www.vadatech.com)
- VadaTech AMC535 Hardware Reference Manual (http://www.vadatech.com)
- VadaTech AMC535 FPGA Reference Design Manual (http://www.vadatech.com)
- VadaTech AMC536 Datasheet (http://www.vadatech.com)
- VadaTech AMC536 Hardware Reference Manual (http://www.vadatech.com)
- VadaTech AMC536 FPGA Reference Design Manual (http://www.vadatech.com)
- VadaTech FMC228 Datasheet (http://www.vadatech.com)
- VadaTech FMC228 Hardware Reference Manual (http://www.vadatech.com)
- PICMG® AMC.0 AdvancedMC Mezzanine Module (http://www.picmg.org)
- PICMG® AMC.1 AdvancedMC PCI Express and AS (http://www.picmg.org)
- PICMG® AMC.2 AdvancedMC Ethernet (http://www.picmg.org)

- PICMG® AMC.3 AdvancedMC Storage (http://www.picmg.org)
- PICMG® AMC.4 AdvancedMC Serial RapidIO (http://www.picmg.org)
- Xilinx Virtex-7 Datasheets and User's Guides
- Xilinx Vivado Documentation
- Xilinx IP Documentation
- Altera Stratix-5 Datasheets and User's Guides
- Altera Quartus Documentation
- Altera IP Documentation
- ANSI/VITA 57.1 FPGA Mezzanine Card (FMC) Standard (http://www.vita.com)

1.3 Acronyms Used in this Document

Acronym	Description
ADC	Analog to Digital Converter
AMC	Advanced Mezzanine Card
BAR	Base Address Register
BIST	Built-In Self Test
BSP	Board Support Package
C2M	Carrier-to-Mezzanine (signal)
CGND	Chassis Ground
CLK	Clock
CPU	Central Processing Unit
DAC	Digital to Analog Converter
DIP	Dual In-line Package
DMA	Direct Memory Access
DMUX	De-multiplexer
FMC	FPGA Mezzanine Card
FPGA	Field Programmable Gate Array
FRU	Field Replaceable Unit
GbE	Gigabit Ethernet
GND	Signal Ground
GSPS	Giga Samples Per Second
HPC	High Pin Count (FMC connector)
ioctl	Input/Output/Control
IP	Intellectual Property
IPMI	Intelligent Platform Management Interface
JSM	JTAG Switch Module
JTAG	Joint Test Action Group
LED	Light Emitting Diode
LPC	Low Pin Count (FMC connector)
LVCMOS	Low-Voltage Complementary Metal Oxide Semiconductor
LVDS	Low Voltage Differential Signaling
M2C	Mezzanine-to-Carrier (signal)
MAC	Media Access Controller
MB	Megabyte (2^20 bytes)
MIG	Memory Interface Generator

M-LVDS	Multi-point Low Voltage Differential Signaling
MMC	Module Management Controller (IPMI controller of AMC)
MUX	Multiplexer
n.c.	No connection
PCI	Peripheral Component Interconnect
PCIe	Peripheral Component Interconnect Express
PICMG	PCI Industrial Computer Manufacturer's Group
PLL	Phase Locked Loop
SERDES	Serializer/Deserializer
TCLK	Telephony Clock
VADJ	Adjustable Voltage (power rail)
VIO	I/O Voltage (power rail)
VREF	Reference Voltage (power rail)

Table 1: Acronyms

2 Xilinx 7-Series FPGA Carrier Board Reference Designs

The following reference design projects are provided.

FPGA Carrier Board	FPGA / Speed Grade	Name	Signature ID	Image ID	Note
AMC516	Xilinx Virtex-7 X690T-2 (or faster)	AMC516_FMC228	0xA516	0xF228	1. FPGA Speed Grade 2 (or faster) is required to achieve 10Gbps SERDES line rate
AMC502	Xilinx Kintex-7 K420T-2 (or faster)	AMC502_FMC228	0xA525	0xF228	1. FPGA Speed Grade 2 (or faster) is required to achieve 10Gbps SERDES line rate 2. Two Vivado project files are provided: one for FMC228 on FMC0; the other for FMC228 on FMC1.
AMC593	Xilinx Kintex Ultrascale KU115-2 (or faster)	AMC593_FMC228	0xA593	0xF228	1. FPGA Speed Grade 2 is specified since AMC593 currently doesn't have ordering option for speed grade 1. 2. Two Vivado project files are provided: one for FMC228 on FMC0; the other for FMC228 on FMC1.

Table 2: Xilinx 7-Series FPGA Carrier Board Reference Design Projects

2.1 Reference Design Projects Naming Convention

The FPGA reference design name follows the convention for single FMC carrier boards:

'VadaTech single FMC carrier board' + '_FMC228'

For dual FMC carrier boards, two projects are provided for each FMC connector:

'VadaTech dual FMC carrier board' + '_FMC228' + '_F0'
'VadaTech dual FMC carrier board' + '_FMC228' + '_F1'

The projects always use generic in the design's top module to indicate the supported carrier FPGA speed grade.

2.2 Identify Project Images Loaded in the FPGA

All carrier board reference designs implement the following common registers. On most VadaTech carrier boards with PPC CPU option, a script is provided to read those common registers. Please refer to individual carrier board's FPGA reference design manual for details.

Common Register
Signature ID
IMAGE ID
VERSION 1
VERSION 2

Table 3: Reference Design Common Registers

On some reference designs, the above common registers may be mapped to another base address too, and can be read out using the provided application software in addition to the script. In this case, the common register read values should be identical.

2.3 Building Xilinx 7-Series Carrier Boards Reference Designs

The designs use **Vivado** and the pre-compiled images make use of hardware evaluation licenses where necessary rather than paid licenses if applicable. VadaTech does not provide licenses for the Vivado tool nor the IP cores from Xilinx. Please contact Xilinx directly for licensing, IP core, or device-specific support.

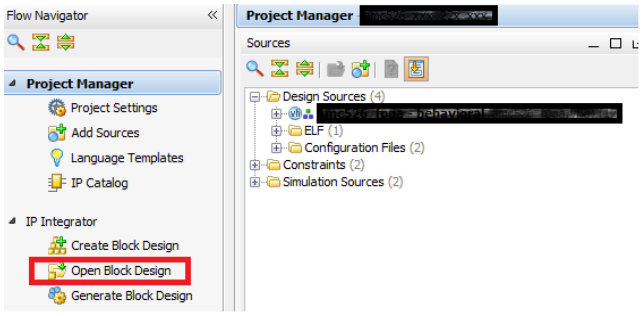
NOTE: When building the images make sure to build the out-of-context runs first before performing the main run, otherwise the required sub-components won't be available on your machine and the main run will fail.

LEGAL NOTICE: The VadaTech custom VHDL code included in this reference design is the intellectual property of VadaTech Incorporated. Permission is granted to use the VadaTech custom VHDL code royalty-free in customer FPGA carrier board designs targeting the VadaTech FMC228 card only. Redistribution to third parties or use of this code for any other purpose is strictly prohibited.

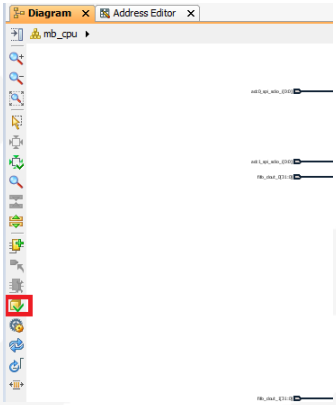
2.3.1 Build reference designs with Xilinx Block Design (IP integrator)

For designs using Xilinx Block Design (IP integrator), please follow the steps below before trying to run synthesis/implementation. The steps will generate all necessary block design files for one time only and will not be required in subsequent builds.

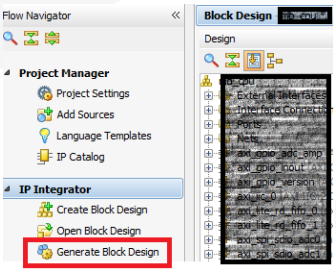
1. Open the project in Vivado
2. Open Block Design



3. In the opened Diagram window, click ‘Validate Design’ icon or press ‘F6’



4. If a window pops up, click ‘Rerun Validate Design’ in the pop up window. If not, go to step 5.
5. Generate Block Design

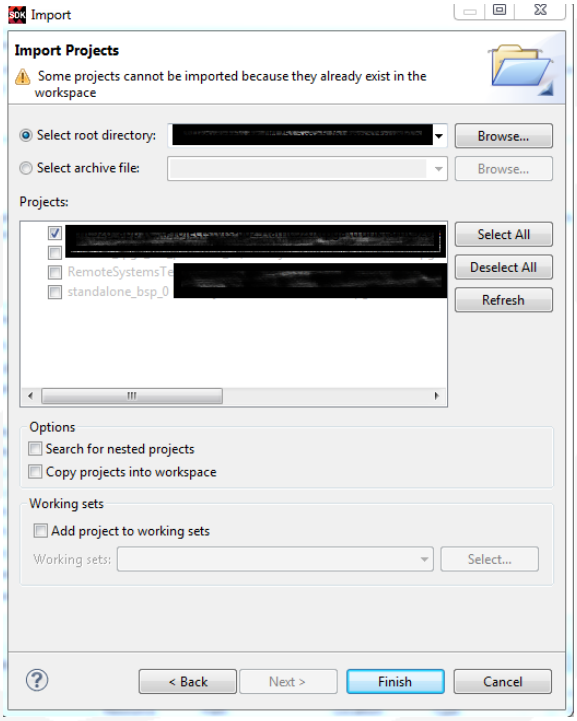


2.3.2 Build SDK project

For a reference design using SDK, a prebuilt .elf file is provided. The reference design can be synthesized and implemented without using Xilinx SDK software with the default .elf file. However, if a user would like to modify the SDK project, please make sure the Xilinx SDK software is installed on the computer.

1. Open the project in Vivado.
2. Validate and generate block design
3. Synthesize the project

4. Click the Vivado menu item: File->Export->Export Hardware. Browse to the reference design's ~/sources/sdk folder and click 'Select'. Do not use the default <local to project> option.
5. Click the Vivado menu item: File->Launch SDK. Choose the same ~/sources/sdk folder for both 'Exported location' and 'Workspace'. Do not use the default <local to project> options.
6. The Xilinx SDK software should be opened now.
7. Click the SDK menu item: File->New->Board Support Package. Click 'Finish' to accept default settings in the pop-up window.
8. Click 'OK' in the 'Board Support Package Settings' pop-up window.
9. Click the SDK menu item: File->Import. In the 'Import' pop-up window, expand 'General' tab and choose 'Existing Projects into Workspace'. Click 'Next'.
10. In the 'Import Projects' page, click 'Browse...' button to open pop-up window 'Browse for Folder' and click 'OK' to accept the default settings.
11. Click 'Finish' and the project will be automatically built.



2.3.3 Build reference designs under Vivado for dual carrier boards

The two Vivado project files (*_f0.xpr and *_f1.xpr) provided for dual carrier boards share the same IP core files. Please don't synthesize both projects at the same time at the same location. VadaTech recommends to copy the reference design to another location to build both projects at the same time.

2.4 Building Xilinx 7-Series Application Software

The application software is typically located at the project's ~/sources/sdk/fmc228_tool/src folder.

Whereas possible, all FMC reference designs try to instantiate a MicroBlaze soft CPU core in addition to other interfaces to communicate with the designs. To build application software running on other interfaces, a Makefile with 3 targets is provided (replace 'xxx' with the carrier board's actual name):

ACCESS_INTERFACE	Description	Target name	Note
P2040_LB	PPC local bus to the FPGA	fmc228_tool_lb	Running on the carrier board's PPC CPU. Requires carrier board's lb driver at /dev/amcxxx_fpga_lb
PCIE_DEV_DRV	PCIe interface to the PPC CPU	fmc228_tool_drv	Running on the carrier board's PPC CPU. Requires pcie driver (provided): /dev/vt_simple_pcie_drv
PCIE_MMAP	Backplane PCIe interface	fmc228_tool_mmap	Running on any CPU board in the same chassis. No driver is required. Needs 'pciutils' package to run. 'pciutils-devel' package to build.
MicroBlaze		N/A	Running on the carrier board's FPGA. A front panel FPGA RS-232 console is required.

Table 4: Reference Design Control Interfaces

Some control interfaces may not be available for a given carrier board. For example, a carrier board without PPC CPU option can only use the MicroBlaze (provided a front panel FPGA RS-232 is available) and the backplane PCIe interfaces; a carrier board with PPC CPU option typically doesn't have a FPGA RS-232 console on the front panel and therefore the MicroBlaze interface is not available.

To build the targets fmc228_tool_drv and/or fmc228_tool_lb for running on a PPC CPU, the FPGA carrier board's BSP source (cross compiler) for the PPC CPU must be properly installed on a Linux computer (host) and path to the cross compiler must be correctly specified in the Makefile.

To build the MicroBlaze application, Vivado SDK software is required.

2.4.1 Building Application Software Running on a PPC CPU

For carrier boards with PPC CPU option, please following the detailed procedure below (replace xxx with the carrier board's actual name):

1. Download and extract the application software source (typically located at reference design's ~/sources/sdk/fmc228_tool/src) from the reference design project file to a location on the host

2. Build the carrier board's (AMCxxx) BSP source as described in AMCxxx Software User Manual on the host

3. Enter to the BSP's folder on the host

   # cd amcxxx/

4. Copy the whole software directory '/src' in step 1 to the BSP's folder on the host

5. Enter to the source dir

   # cd src/

6. Edit the Makefile to change the CC variable as follows if not so

   CC=../tmp/sysroots/$(shell uname -m)-linux/usr/bin/ppce500mc-fsl-linux/powerpc-fsl-linux-gcc --sysroot=../tmp/sysroots/p2041rdb/

7. Compile

   # make

8. Copy the executable from the Linux machine to the FPGA carrier board's CPU system (target) to execute using SCP command

2.4.2 Building and installing PCIe driver for a PPC CPU

After steps in 2.4.1 is done, if the design supports PCIe link to the PPC CPU, please following the detailed procedure below (replace xxx with the carrier board's actual name):

1. Enter the driver directory on the host

   # cd p2040_pcie_driver/

2. Build the driver

   # make

3. Copy the driver module to the target

4. Install the driver on the target

2.4.3 Building application software on a backplane CPU

If the design can be controlled via the backplane PCIe interface from a host CPU, please following the procedure below to build the application software on the host system:

1. On the host Linux system, make sure the package 'pciutils-devel' is installed.
2. Copy all SDK application source files under the ~/src directory except 'lscript.ld' 'platform.c', 'platform.h' and 'platform_config.h' from the reference design to a directory on the host computer.
3. Compile

   # make

2.5 FMC228 Main Design

The main design demonstrates the following functionalities of the FMC228 card:

- SPI/SDIO registers read/write to LMK04828, HMC835 and AD9234s on the FMC228 card
- Two ADCs (AD9234) on 1GHz sampling w/o decimation.
- Front panel Trigger_In to Trigger_Out loop back
- Front panel CLK IN port
- Front panel LEDs turn-on test
- LMK04828 status LEDs
- HMC835 lock status LEDs

The top module of the main design is fmc228.vhd. It uses AXI4-Lite as the control interface and claims 64k bytes AXI space. Two AXI4-Lite slave interfaces are provided since the PPC CPU local bus to AXI4-Lite Bridge (fsl_elbc_axi4lite_bridge) doesn't have wait state and can't go through two AXI crossbar switches.

2.5.1 FMC228 main design reference design block diagram

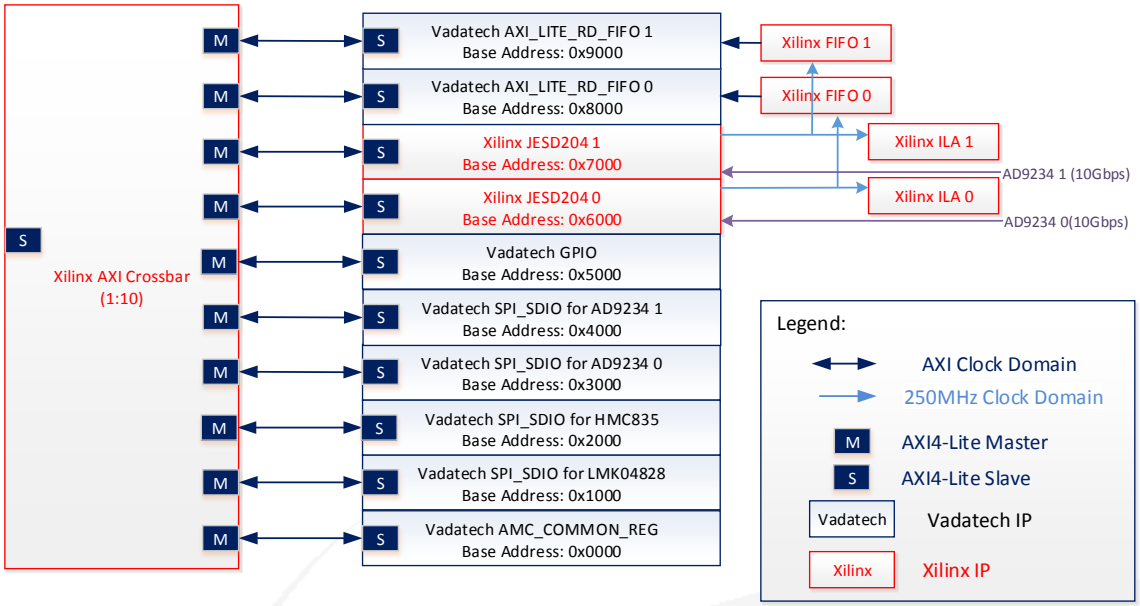


Figure 1: FMC228 Main Design Block Diagram

2.5.2 Clocks

The following clocks are used in the reference design:

Clock Name	Frequency	Source	Description
axi_aclk	Depends	Depends	This clock is used for all AXI4-Lite interfaces
jesd204b_core_clk	250MHz	JESD204 core clock from FMC228 LMK04828	This clock is generated in the JESD204 core. It is used only inside the JESD204 core.
rx_aclk	250MHz	JESD204 core	JESD204 AXI4-Stream clock

Table 5: FMC228 Main Design Clocking

2.5.3 AXI address map

The reference design has 64K AXI memory space mapped as followings:

Base Address	Instantiated Core	Core Address Width
0x0000	AMC_COMMON_REG	12
0x1000	Vadatech AXI_SPI_SDIO	7
0x2000	Vadatech AXI_SPI_SDIO	7
0x3000	Vadatech AXI_SPI_SDIO	7
0x4000	Vadatech AXI_SPI_SDIO	7
0x5000	Vadatech GPIO	12
0x6000	Xilinx JESD204 Core 0	12
0x7000	Xilinx JESD204 Core 1	12
0x8000	Vadatech AXI_LITE_RD_FIFO 0	12
0x9000	Vadatech AXI_LITE_RD_FIFO 1	12

Table 6: FMC228 Main Design AXI Address Map

2.5.4 Vadatech cores

- FSL_LBC_TO_AXI4_LITE_BRIDGE

  The core implements a simple Freescale mpc85xx eLBC bus to AXI4-Lite bridge with 32-bit addresses. Please refer to the VHDL file for limitations.

- AMC_COMMON_REG

  The core implements the AMC carrier board's common registers listed in Table 3.

- AXI_SPI_SDIO

The core connects the AXI4-Lite interface to serial peripheral interface (SPI/SDIO) slave devices. It supports bi-directional data line and multiple channels. The core can be dynamically configured to suit different address and data bits width.

- AXI_LITE_RD_FIFO

  The core provides a simple AXI-Lite interface to read FIFO data.

- AXI_GPIO_32

  This core provides 32 32-bit GPIOs on AXI-Lite interface. The core is used for miscellaneous controls and information that cannot be generalized/grouped into an independent core. It provides reading/writing over various on-board IC pins and some FPGA internal signals such as clock enables and resets. It also contains read-only identification information such as version, signature etc. There is also a scratch register present in the core which can be used for bus testing.

  Please refer to the VHDL source code for detailed register map.

2.5.5 Xilinx core

The following core from Xilinx is used:

- Xilinx LogiCORE IP AXI Interconnect Product Guide (PG059)

2.5.6 JESD204 link parameters for the ADCs

The FMC228 board supports maximum of 4 JESD204 lanes for each ADC. The serial link parameters that are used in the reference design are:

Parameters	Description	Value
Fs	ADC sampling frequency	1GHz
D	Decimation Factor	1
HD	High Density	1
L	Number of serial lanes	4
F	Number of octets per frame	1
M	Number of converters	2
Bit Rate	JESD204 serial link bit rate	10Gbps

Table 7: FMC228 Main Design Serial Link Parameters

2.5.7 Capturing ADC data using ILA

In addition to capture ADC data using the application software, the reference design instantiates two Xilinx ILA cores to capture ADC raw data directly for debug purpose.

The ILA cores only appear in the Vivado Hardware Manager panel after the JESD204 core clocks are stable in the FPGA. And this requires sending initialization command 'i' after the FPGA is configured.

There's one ILA core for each ADC. To capture data from ADC 0 (Analog IN 0 port on the front panel), send the following command (example) from the Vivado Tcl Console after the hw_ila_1 core waveform is captured on the Vivado Hardware Manager panel:

```
write_hw_ila_data c:/data/fmc228_ila_1.zip hw_ila_data_1
```

The example command saves the captured compressed data to c:/data directory. The user is free to choose the location and file name for the captured data.

After unzipping the compressed file, the waveform file waveform.cvs is available for processing.

2.6 FMC228 Application Software

The usage information for the fmc228_tool_* is shown below:

```
FMC228 on AMC Carrier Tool v2
Usage: <cmd> [<opts>]
CMDs:
    i [b0|b1]                       - Initialize the card to default values
                                      b0: on-board sampling clk; b1: external
    c [file]                        - Capture ADC data
    d <g|j0|j1>                     - Dump FPGA register contents
    w <addr> <val>                  - Write value at fpga address
    r <addr>                        - Read back at fpga address
    dd <a0|a1|h|l>                  - Dump device's register contents
    wd <a0|a1|h|l> <addr> <val>     - Write value to the device's address
    rd <a0|a1|h|l> <addr>           - Read value from the device's address
    rs <fmc|fp>                     - Select 10MHz ref clock input for lmk04828
Device Key:
      g     = GPIO
      j0    = JESD204 0
      j1    = JESD204 1
      a0    = AD9234 0
      a1    = AD9234 1
      h     = HMC835
      l     = LMK04828
```

Figure 2: FMC228 Application Tool Usage Information

When using the FPGA RS-232 console for MicroBlaze (if applicable with the carrier board's ordering option), the command 'c' doesn't have the option to save to a file.

2.6.1 Board initialization

The board must be initialized first by sending the command 'i [ordering option B]' after the FPGA is configured.

If ordering option B=0 (use on-board PLL for the ADC sampling clock), the initialization function expects that the carrier board is configured to output a 10MHz reference clock to the FMC_CLK2_BIDIR pin which is connected to the CLKIN0 input of the LMK04828 on the fmc228 board. If no such a clock is present, the ADC sampling clocks will be referenced to the FMC228 on-board 100MHz VCXO. In this case, the LMK04828 PLL1 will be out of lock and the initialization command will report the LMK04828 is not locked. However, the board will still be functioning. If a phase locked sampling clock is desired, a high quality reference clock to the LMK04828 from either the front panel (LMK04828 CLKIN1) or the FMC FMC_CLK2_BIDIR (LMK04828 CLKIN0) must be provided. Please refer to the carrier board's manual on how to configure the carrier board to send such a reference clock to the FMC_CLK2_BIDIR pin.

If ordering option B=1 (use external ADC sampling clock via front panel CLK IN port), a 1GHz clock with appropriate amplitude (9dBm typical, refer to the hardware reference manual for valid range) should be provided.

2.6.2 Command examples

Comand examples running on PPC CPU using CPU LB:

- **`./fmc228_tool_lb i b0`**

  Initialize all components on the FMC228 for on-board sampling clock

- **`./fmc228_tool_lb i b1`**

  Initialize all components on the FMC228 for external sampling clock

- **`./fmc228_tool_lb i`**

  Initialize all components based on the value of generic ORDER_OPTION_B specified in the top VHD module

- **`./fmc228_tool_lb d j0`**

  Dump all registers in AXI space of the Xilinx JESD204 core for the 1st AD9234 ADC to the screen

- **`./fmc228_tool_lb dd l`**

  Dump all LMK04828 registers to the screen

- **`./fmc228_tool_lb r 0x5FFC`**

  Read the 32-bit register values at the AXI address 0x5FFC (0x5000 base + 0xFFC offset = GPIO core register SIG). The read back value should be 0xF228AYYY (Image ID + Signature) where YYY is the AMC carrier board's name.

- **`./fmc228_tool_lb rd l 4`**

  Read LMK04828 regsiter value at its address 0x4 (ID_PROD[15:8]). The read back value shoud be 0xD0.

- **`./fmc228_tool_lb rs fp`**

Select the front panel CLK IN port as the LMK04828 PLL1 reference clock. The front panel CLK IN port is connected to the LMK04828 CLKIN1 pins. A valid 10MHz clock must present on the front panel CLK IN port before executing this command.

- **`./fmc228_tool_lb wd a0 0xa 0xbf`**

  Write 0xBF to the ADC0's register 0xA (scratch pad)

- **`./fmc228_tool_lb c`**

  Capture 512 ADC samples from all 4 channels to the screen

2.7 AMC502_FMC228 Reference Design Project

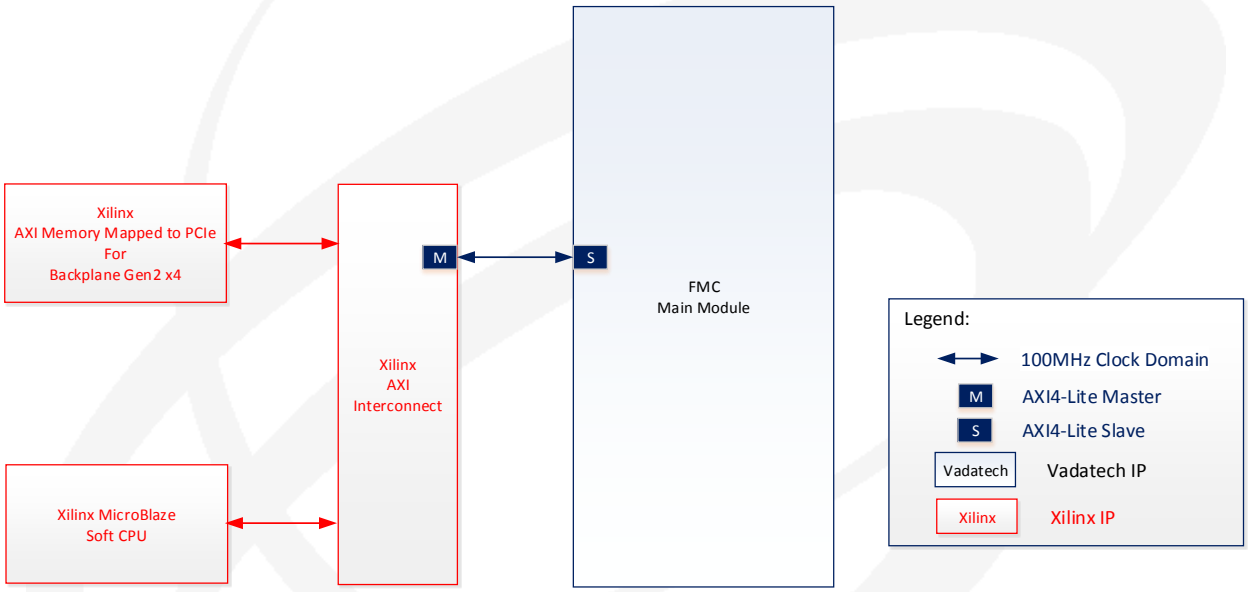


Figure 3: AMC502_FMC228 Reference Design Block Diagram

The design implements 2 interfaces to control the FMC228 main design. Please refer to the AMC502 manauls and reference designs on how to control the telco clocks on the AMC502 using FPGA to provide a 10MHz clock to the FMC_CLK2_BIDIR pins.

2.8 AMC516_FMC228 Reference Design Project

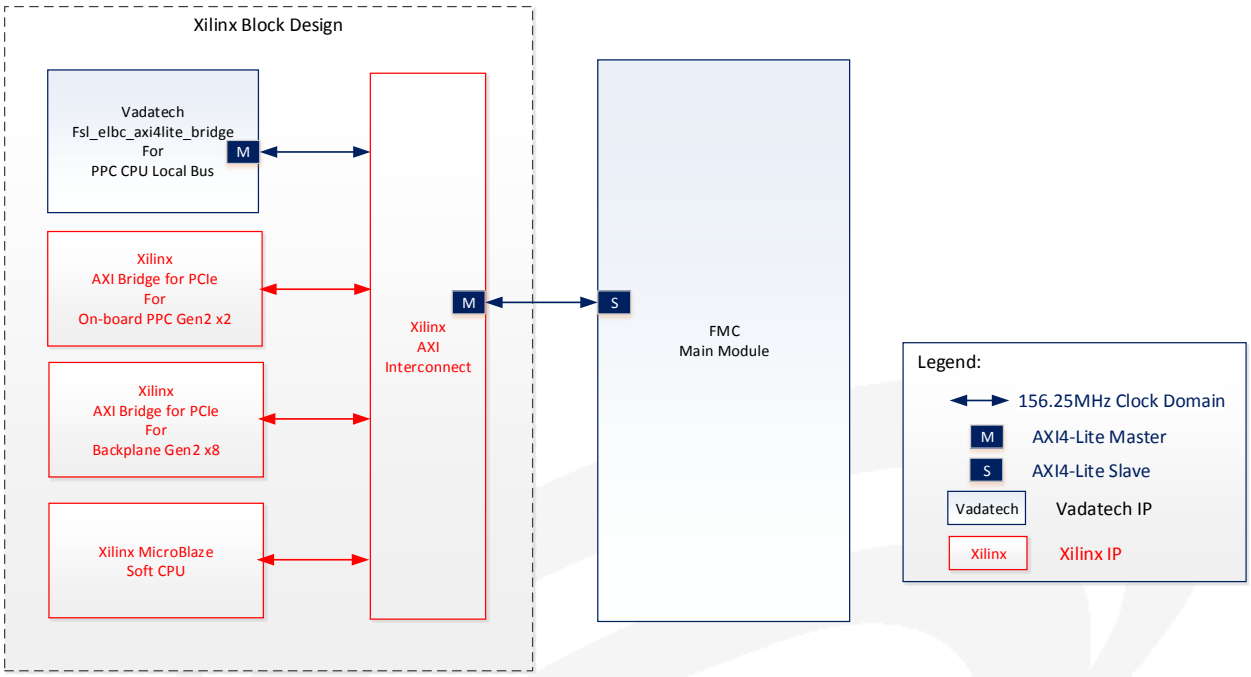


Figure 4: AMC516_FMC228 Reference Design Block Diagram

The design implements 4 interfaces to control the FMC228 main design. A script file 'testclocks_10mhz_to_FMC_CLK2_BIDIR_amc516.sh' is provided to demo how to output a 10MHz frequency to the FMC_CLK2_BIDIR pins from the on-board PPC CPU (ordering option B=1). The 10MHz clock is generated by the Quad PLL on the AMC516 and is for demonstration purpose only. In the real application, please feed a high quality reference clock to the LMK04828 from either the front panel (CLKIN1) or the FMC connector (CLKIN0).

For ordering option B=0, please refer to the AMC516 manauls and reference designs on how to control the telco clocks on the AMC516 using FPGA.

2.9 AMC593_FMC228 Reference Design Project

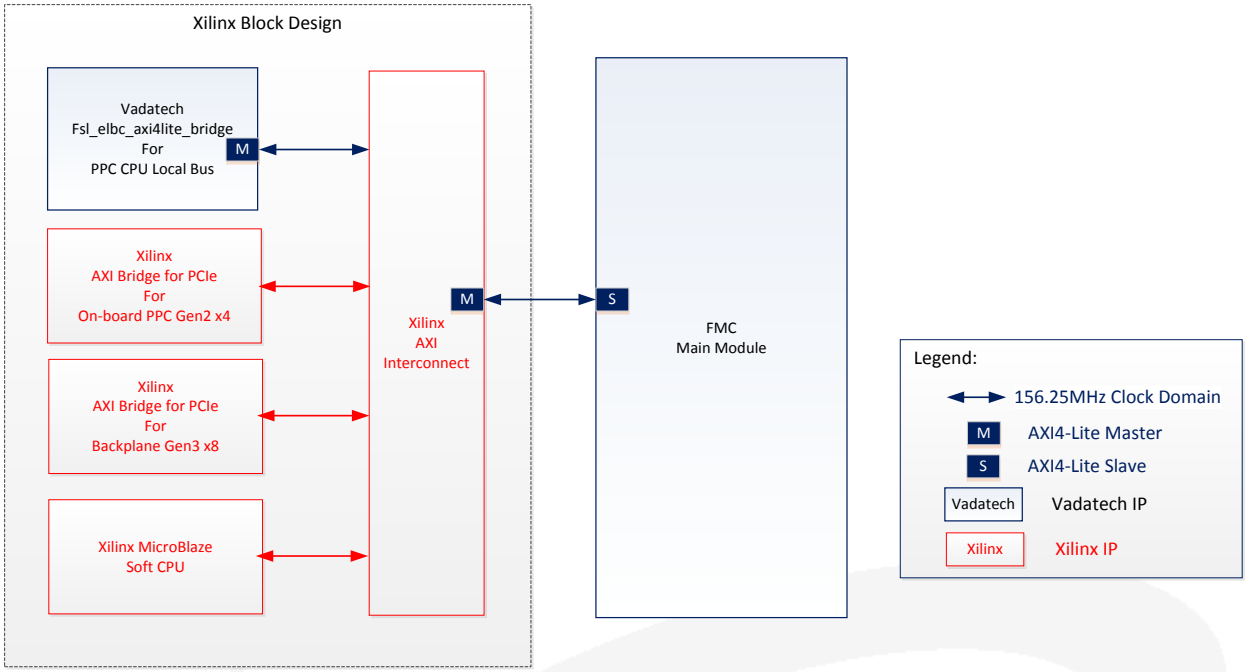


Figure 5: AMC593_FMC228 Reference Design Block Diagram

The design implements 4 interfaces to control the FMC228 main design. Please refer to the AMC593 manauls and reference designs on how to control the telco clocks on the AMC593.

3 Altera 5-Series and 10-Series FPGA Carrier Board Reference Designs

The following reference design projects are provided.

FPGA Carrier Board	FPGA / Speed Grade	Name	Signature ID	Image ID	Note
AMC532	Altera Stratix-5 5SGXEA (all speed grades)	AMC532-XXX-22X-XXX_FMC228	0xA532	0xF228	1.The application software is running on the Altera FPGA's NIOS-2 processor 2.Tested with 5SGXEA7N2F45C2
AMC535	Altera Arria-10 SOC SX600 (all speed grades)	AMC535-XXX-X2X-XXX_FMC228	0xA535	0xF228	1.The application software is running on the Altera FPGA's NIOS-2 processor 2.Tested with 10AS066K2F40E2SG
AMC536	Altera Arria-10 GX1150 (all speed grades)	AMC536-XXX-X2X-XXX_FMC228	0xA536	0xF228	1.The application software is running on the Altera FPGA's NIOS-2 processor 2.Tested with 10AX115N2F40E2SG

Table 8: Altera 5-Series and 10-Series FPGA Carrier Board Reference Design Projects

3.1 Reference Design Projects Naming Convention

The FPGA reference design name follows the convention for single FMC carrier boards:

'VadaTech single FMC carrier board' + '_FMC228'

For dual FMC carrier boards, two projects are provided for each FMC connector:

'VadaTech dual FMC carrier board' + '_FMC228' + '_F0'
'VadaTech dual FMC carrier board' + '_FMC228' + '_F1'

The projects always use generic in the design's top module to indicate the supported carrier FPGA speed grade.

3.2 Identify Project Images Loaded in the FPGA

All carrier board reference designs implement the following common registers. On most VadaTech carrier boards with PPC CPU option, a script is provided to read those common registers. Please refer to individual carrier board's FPGA reference design manual for details.

Common Register
Signature ID
IMAGE ID
VERSION 1
VERSION 2

Table 9: Reference Design Common Registers

On some reference designs, the above common registers may be mapped to another base address too, and can be read out using the provided application software in addition to the script. In this case, the common register read values should be identical.

3.3 Building Altera 5-Series and 10-Series Carrier Boards Reference Designs

The designs use **Quartus** and the pre-compiled images make use of hardware evaluation licenses where necessary rather than paid licenses if applicable. VadaTech does not provide licenses for the Quartus tool nor the IP cores from Altera. Please contact Altera directly for licensing, IP core, or device-specific support.

NOTE: It is recommended to use the Front Panel FPGA RS-232 Port and a terminal program such as Tera Term for all NIOS-II host CPU menu driven designs described in this document.

LEGAL NOTICE: The VadaTech custom VHDL code included in this reference design is the intellectual property of VadaTech Incorporated. Permission is granted to use the VadaTech custom VHDL code royalty-free in customer FPGA carrier board designs targeting the VadaTech FMC228 card only. Redistribution to third parties or use of this code for any other purpose is strictly prohibited.

3.4 Building Altera 5-Series and 10-Series Application Software

The application software is located in the project's ~/application_software folder.
Please follow the detailed procedure below to build application software on the Altera NIOS-II Processor (replace the xxx with actual carrier board name).

1. Use Quartus II to open the main QPF project file
2. Open the NIOS II Software Build Tools for Eclipse GUI from the Quartus II Tools menu.
3. Create a new bsp template, select the project's SOPC file, select a Blank Project template, enter a project name such as amcxxx_fmc228, then click Finish

4. Import or copy the .c and .h files in the application_software/src subdirectory to the Eclipse GUI's Project Explorer main project folder (amcxxx_fmc228)

5. Right click on the _bsp portion of the project (amcxxx_fmc228_bsp), then right click NIOS, then open the BSP Editor

6. Under the Main tab select uart_fpga_cpu for stdin, stdout, and stderr. This will allow you to connect the AMC carrier board's front panel FPGA CPU to a PC COM port and use a terminal program such a Tera Term (recommended) to run the NIOS-II menu based software

7. Click on the Generate BSP button, then exit the BSP Editor and select Save if prompted

8. Right click on the main project name and select Build Project

   NOTE: Skip steps 10, 11, and 12 if not embedding the newly built software into the FPGA configuration SOF file.

9. Right click on the main project name (amcxxx_fmc228) and select Make Targets, click on Build, select the mem_init_generate radio button, then click the Build button

10. In the Quartus Assignments menu select Settings and include the newly created meminit.qip file

11. Quartus compile the project

12. Download the project SOF file using a USB Blaster though JTAG using the Quartus II Programmer. The NIOS-II menu system should appear on the PC terminal with the original project's software or with the newly built software embedded in the SOF using steps 10-12. See section 3.6 of this document for additional information.

13. For software development using the NIOS II Software Build Tools for Eclipse GUI, you can download and test each new software build by right clicking on amcxxx_fmc228 in the Project Explorer, then right clicking on Run As and selecting NIOS II Hardware. The NIOS-II software elf file should download to the FPGA though the USB Blaster and the NIOS-II menu system for the newly built software should appear on the PC terminal. See section 3.6 of this document for additional information.

3.5 FMC228 Main Design

The main design demonstrates the following functionalities of the FMC228 card:

- SPI/SDIO registers read/write to LMK04828, HMC835 and AD9234s on the FMC228 card
- Two ADCs (AD9234) on 1GHz sampling w/o decimation.
- Front panel Trigger_In to Trigger_Out loop back
- Front panel LEDs turn-on test
- LMK04828 status LEDs
- HMC835 lock status LEDs

The top module of the main design is amc532_xxx_22x_xxx_fmc228.vhd for the AMC532, amc535_xxx_x2x_xxx_fmc228.vhd for the AMC535, and amc536_xxx_x2x_xxx_fmc228.vhd for the AMC536. This reference design is for testing the FMC228 FPGA mezzanine card only. It doesn't implement any interface on the backplane AMC ports. It only supports the NIOS CPU interface for register reads/writes.

3.5.1 FPGA reference design block diagram

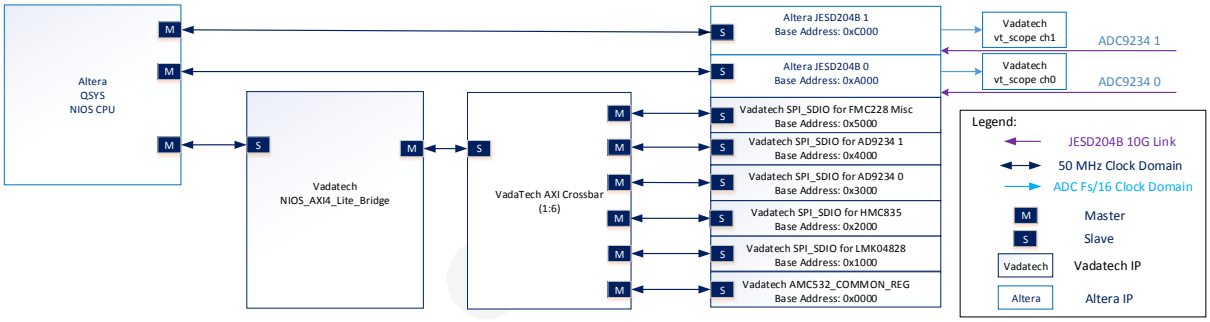


Figure 6: AMC Carrier Board FMC228 Reference Design Block Diagram

Application software is provided to demo how to output a 10MHz frequency to the FMC_CLK2_BIDIR pins. The 10MHz clock is generated by the Quad PLL on the AMC carrier board and is for demonstration purpose only. In the real application, please feed a high quality reference clock to the LMK04828 from either the front panel (CLKIN1) or the FMC connector (CLKIN0).

For FMC228 ordering option B=0, please refer to the respective AMC carrier board manuals and reference designs on how to control the telco clocks on the AMC carrier board using the FPGA.

3.5.2 Clocks

The following clocks are used in the reference design:

Clock Name	Frequency	Source	Description
nios_clk	50 MHz	AMC carrier board's clock generator	This clock is used for all NIOS and AXI4-Lite interfaces
fmc_gbtclk0	250MHz	JESD204B core 0 Gigabit receiver CDR training reference clock	JESD204B core 0 Gigabit receiver CDR training reference clock
fmc_gbtclk1	250MHz	JESD204B core 1 Gigabit receiver CDR training reference clock	JESD204B core 1 Gigabit receiver CDR training reference clock
j0_rx_aclk	250MHz	JESD204B 0 CDR	JESD204B ADC 0 data stream clock
j1_rx_aclk	250MHz	JESD204B 1 CDR	JESD204B ADC 1 data stream clock

Table 10: AMC Carrier Board Reference Design Clocking

3.5.3 Address map

Base Address	Instantiated Core	Core Address Width
0x0000	AMC_COMMON_REG	12
0x1000	Vadatech AXI_SPI_SDIO	7
0x2000	Vadatech AXI_SPI_SDIO	7
0x3000	Vadatech AXI_SPI_SDIO	7
0x4000	Vadatech AXI_SPI_SDIO	7
0x5000	Vadatech GPIO	12
0xA000	Altera JESD204 Core 0	8
0xC000	Altera JESD204 Core 1	8

Table 11: AMC Carrier Board Reference Design Address Map

3.5.4 Vadatech cores

- NIOS_AXI4_LITE_BRIDGE

  The core implements a simple Altera NIOS bus to AXI4-Lite bridge with 32-bit addresses. Please refer to the VHDL file for limitations.

- AMC_COMMON_REG

  The core implements the AMC Carrier Board's FMC228 reference design common registers listed in Table 12.

Local Bus Offset	Mnemonic	Description
0x00000000	BIT0	Read-only test pattern 0x0001
0x00000002	BIT1	Read-only test pattern 0x0002
0x00000004	BIT2	Read-only test pattern 0x0004
0x00000006	BIT3	Read-only test pattern 0x0008
0x00000008	BIT4	Read-only test pattern 0x0010
0x0000000A	BIT5	Read-only test pattern 0x0020
0x0000000C	BIT6	Read-only test pattern 0x0040
0x0000000E	BIT7	Read-only test pattern 0x0080
0x00000010	BIT8	Read-only test pattern 0x0100
0x00000012	BIT9	Read-only test pattern 0x0200
0x00000014	BIT10	Read-only test pattern 0x0400
0x00000016	BIT11	Read-only test pattern 0x0800
0x00000018	BIT12	Read-only test pattern 0x1000
0x0000001A	BIT13	Read-only test pattern 0x2000
0x0000001C	BIT14	Read-only test pattern 0x4000
0x0000001E	BIT15	Read-only test pattern 0x8000
0x00000020	SCRATCH	Read/write scratch register
0x000003F8	IMAGE_ID	Read-only image identifier
0x000003FA	VERSION1	Read-only major/minor version number (0xMMNN)
0x000003FC	VERSION2	Read-only patch/revision version number (0xPPRR)

0x000003FE	SIGNATURE	Read-only signature (0xAYYY) YYY=532, 535, or 536

Table 12: FPGA Reference Design Common Registers

- AXI_SPI_SDIO

  The core connects the AXI4-Lite interface to serial peripheral interface (SPI/SDIO) slave devices. It supports bi-directional data line and multiple channels. The core can be dynamically configured to suit different address and data bit widths.

- VT_SCOPE

  The core provides a simple interface to capture real-time ADC waveform data for display in the Altera SignalTap GUI and for export to the PC file system.

- FMC228_MISC

  The core connects the AXI4-Lite interface to miscellaneous controls and information that cannot be generalized/grouped into an independent core. It provides reading/writing over various on-board IC pins and some FPGA internal signals such as clock enables and resets. It also contains read-only identification information such as version, signature etc. There is also a scratch register present in the core which can be used for bus testing.

  Please refer to the VHDL source code for detailed register map.

3.5.5 Altera core

The following core from Altera is used:

- JESD204B IP Core User Guide (UG-01142)

3.5.6 JESD204 link parameters for the ADCs

The FMC228 board supports maximum of 4 JESD204 lanes for each ADC. The serial link parameters that are used in the reference design are:

Parameters	Description	Value
Fs	ADC sampling frequency	1GHz
D	Decimation Factor	1
HD	High Density	1
L	Number of serial lanes	4
F	Number of octets per frame	1
M	Number of converters	2
Bit Rate	JESD204 serial link bit rate	10Gbps

Table 13: FMC228 Main Design Serial Link Parameters

3.5.7 Capturing ADC data

For real-time DSP applications, big-endian blocks of 4 data samples streaming at Fs/16 are available from ADC0 as VHDL signals j0_a_data and j0_b_data which are clocked by j0_rx_aclk, and from ADC1 as VHDL signals j1_a_data and j1_b_data which are clocked by j1_rx_aclk. Each ADC provides two channels of streaming 1 GSPS data.

For diagnostic purposes, the reference design instantiates the VadaTech vt_fifo and vt_scope IP cores, to capture raw ADC streaming data for observation in SignalTap, and for exporting to a `.csv` file. vt_fifo gets both ADC channel's A & B data onto the same clock tree, and vt_scope maps the ADC's A & B streams of 4 sample data blocks each to a single data stream. For AMC535 and AMC536 to simplify meeting timing in vt_scope, after vt_fifo the adc data is passed through a barrel shifter, to clock vt_scope and SignalTap at half the data rate of j0_rx_aclk. The user can observe an output similar to an oscilloscope as shown in Figure 7.

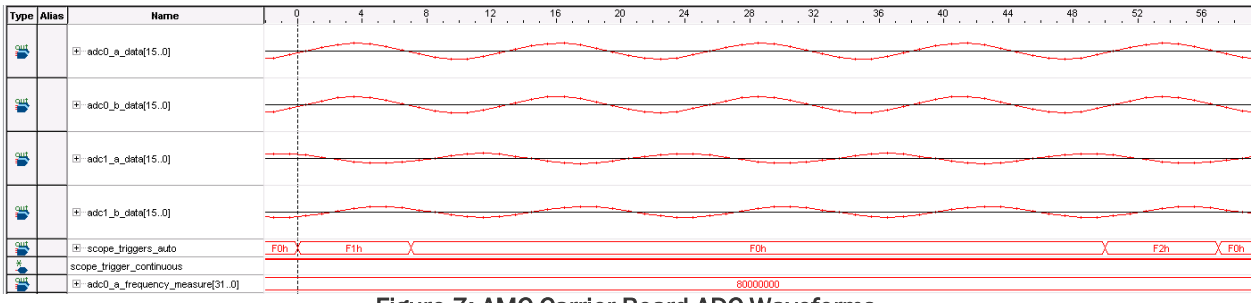


Figure 7: AMC Carrier Board ADC Waveforms

The user can also export the ADC data stream for evaluation purposes using the Quartus II GUI's File-Export menu, before building their own applications. It is recommended to set the SignalTap Bus Display Format for adc0_a_data, adc0_b_data, adc1_a_data, and adc1_b_data to "Signed Decimal in Two's Complement" before exporting.

If a stable waveform is not observable, confirm the hardware setup and then select the **`Init and Sync`** menu items:

```
Select a menu option below:
        Init & Sync
              A) FMC228 Initialize
              B) Synchronize Clock Generator
```

3.5.8 Running the reference design

The design uses **Quartus 15.1** software for AMC532, and **Quartus 16.2.1** software for AMC535 and AMC536, and uses time-limited evaluation licenses where necessary rather than paid licenses. VadaTech does not provide licenses for the Quartus II tool nor the IP cores from Altera. Please contact Altera directly for licensing, IP cores, or Stratix-5 or Arria-10 device-specific support.

1) Power up the AMC Carrier Board with the FMC228 mounted.

2) Use the Quartus II Programmer to configure the respective AMC carrier board's FPGA with **`amc532_xxx_22x_xxx_fmc228.sof`**, **`amc535_xxx_x2x_xxx_fmc228.sof`**, or **`amc536_xxx_x2x_xxx_fmc228.sof`**.

3) Observe an output similar to below:

```
AMC532 FPGA FMC228 Reference Design Software 1.0.0 R0
Initializing LMK04828
Checking if the LMK04828 is locked
Initializing the HMC835LP6GE chip
Checking if HMC_ADC is locked
Initializing AD9234 ADC 0
Initializing AD9234 ADC 1
Initializing JESD204B Cores
Enabling JESD204B clocks
JESD204B core 0 linked
JESD204B core 1 linked
```

```
Select a menu option below:
        Init & Sync
               A) FMC228 Initialize
               B) Synchronize Clock Generator
        HMC835
               C) Dump
               D) Read SDIO
               E) Write SDIO
        LMK04828
               F) Dump
               G) Read SDIO
               H) Write SDIO
        AD9234 (ADC0)
               I) Dump
               J) Read SDIO
               K) Write SDIO
               L) Test Mode
               M) Receiver Tuning
        AD9234 (ADC1)
               N) Dump
               O) Read SDIO
               P) Write SDIO
               Q) Test Mode
               R) Receiver Tuning
        FPGA
               S) Read
               T) Write
               U) Dump JESD204B Device 0
               V) Dump JESD204B Device 1
               W) Dump FMC228_MISC
        Reference Clock Source
               X) FPGA
               Y) Front Panel Clock In
>>
```

For the **Read**, **Write**, **Read SDIO**, **Write SDIO**, and **Receiver Tuning** menu items, the user can complete keyboard input with **CTRL ENTER**.

3.6 FMC228 Application Software

All FMC reference designs come with application software.

The Altera 5-Series and 10-Series FPGA Carrier Boards use an Altera NIOS-II soft CPU core provided by the FPGA's manufacturer and is instantiated in the FPGA. The FPGA manufacturer's SDK software is required to build the application. See Building Altera 5-Series and 10-Series Application Software step 13 of this document for additional information.

3.6.1 Board Initialization

The board is automatically initialized by software after the FPGA is configured. By default, the carrier board is configured to output a 10MHz reference clock to the FMC_CLK2_BIDIR pin which is connected to the CLKIN0 input of the LMK04828 on the fmc228 board.

If a phase locked sampling clock is desired, a high quality reference clock to the LMK04828 from either the front panel (LMK04828 CLKIN1) or the FMC FMC_CLK2_BIDIR (LMK04828 CLKIN0) must be provided. Please refer to the carrier board's manual on how to configure the carrier board to send such a reference clock to the FMC_CLK2_BIDIR pin.

3.6.2 NIOS-II Host CPU

The design includes an application software tool to control the FPGA reference design from the NIOS CPU, and is located in the **`application_software`** subdirectory. See section 3.4 for additional information.

Selected menu descriptions and examples:

- **`FMC228 Initialize`**

  Initialize all components on the FMC228. This command is automatically called after device configuration or when the software is downloaded, see section 3.4 for details.

- **`Synchronize Clock Generator`**

  Resynchronize the JESD204B links

- **`Dump JESD204B Device 0`**

  Dump all registers of the Altera JESD204B core for the 1st AD9234 ADC to the screen

- **`Dump JESD204B Device 1`**

  Dump all registers of the Altera JESD204B core for the 2nd AD9234 ADC to the screen

- **`LMK04828 Dump`**

  Dump all LMK04828 registers to the screen

- **`HMC835 Dump`**

  Dump all HMC835 registers to the screen

- **`Read`**

Read the 32-bit register values at the NIOS-II address. For example, reading FMC228_MISC register value at address 0x5FFC (0x5000 base + 0xFFC offset = FMC228_MISC core register SIG should be 0xF228A532 (Image ID + Signature) for the AMC532 carrier board.

```
FPGA
       Read
       Write
       Dump JESD204B Device 0
       Dump JESD204B Device 1
       Dump FMC228_MISC

    Enter value in hex then press CTRL ENTER
    Press CTRL ENTER without entering a hex value to accept the
    {default}

    Address {0x0} [0-DFFF]: 5FFC
    Read value = 0xF228A532 @ register address 0x4FFC
```

- **Read SDIO**

Read a register from an FMC228 device on the SPI bus. For example, reading LMK04828 register value at address 0x4 (ID_PROD[15:8]) should read back 0xD0

```
LMK04828
       Dump
       Read SDIO
       Write SDIO

    Address {0x0} [0-1FFF]: 4
    Read value = 0x000000D0 @ register address 0x0004 from device
    LMK04828
```

- **Reference Clock Source**

The software during initialization defaults to outputting a 10MHz frequency to the CLKIN0 of the LMK04828 on the fmc228 board.

Select either the front panel or the FPGA as the LMK04828 PLL1 reference clock source. It is recommended to select **FMC228 Initialize** then **Synchronize Clock Generator** after changing the reference clock source.

For example, choose the **Front Panel Clock In** menu item in bold below:

```
Reference Clock Source
       FPGA
       Front Panel Clock In
```

The front panel CLK IN port is connected to the LMK04828 CLKIN1 pin. A valid 10MHz 3.3V compatable LVCMOS/LVTTL clock must be present on the front panel CLK IN port before executing this command.

NOTE: If no front panel clock source is provided, the ADC sampling clocks will be referenced to the FMC228 on-board 100MHz VCXO. In this case, the LMK04828 PLL1 will be out of lock and the initialization command will report the LMK04828 is not locked. However, the board will still be functioning.

- **Write SDIO**

  Write a register to an FMC228 device on the SPI bus. For example, write 0xBF to the ADC0's register 0xA (scratch pad)

```
ADC9234 (ADC 0)
       Dump
       Read SDIO
       Write SDIO
       Test Mode

    Address {0x0} [0-7FFF]: A
    Data {0x0} [0-FF]: BF
    Wrote 0x000000BF @ 0x000A to device ADC9234_0
```

4 Appendix A: Vadatech Cores Register Specification

4.1 Vadatech AXI_SPI_SDIO Core Register Definition

This core implements a SPI/SDIO master controller which supports maximum of 32 slaves with the same protocol. The protocol format is controlled by register instead of parameterized in the HDL code to offer the higher application layer users a better understanding on how to use this module.

The core can be used in two modes: SPI and SDIO

The SPI mode uses four logic signals with dedicated date in and out lines: SCLK, MOSI, MISO and SS.

The SDIO mode uses three logic signals with shared data in/out line: SCLK, SDIO and SS

The mode is selected by how the core is connected on a specific design. The core itself doesn't care which mode is selected.

Alternative naming conversions used in IC product datasheets are (from the slave's point of view):

- SCLK: SCK, CLK
- MOSI: SDI, DIN, DATA
- MISO: SDO
- SS: nCS, CSB, CS, CS*, CS#, LE

Offset	Mnemonic	Description
0~1Bh	N/A	Reserved
1Ch	GIER	Global interrupt enable register
20h	ISR	Interrupt status register
24h	N/A	Reserved
28h	IER	Interrupt enable register
2C~3Fh	N/A	Reserved
40h	SRR	Software reset register
44~5Fh	N/A	Reserved
60h	SPICR	Control register
64h	SPISR	Status register
68h	SPIDTR	Data transmit register
6Ch	SPIDRR	Data receive register
70h	SPISSR	Slave select register

Table 14: Vadatech SPI_SDIO Register Map

4.1.1 GIER: Global interrupt enable register

Register Addrss: 1Ch

Bit(s)	Field	Default Value	Access Type	Description
31	GIE	0	R/W	Global Interrupt Enable. Allows passing all individually enabled interrupts to the interrupt controller. When set to: 0 = Disabled. 1 = Enabled.
30-0	Reserved	N/A	N/A	Reserved

4.1.2 ISR: Interrupt status register

Register Address: 20h

Bit(s)	Field	Default Value	Access Type	Description
31-1	Reserved	N/A	N/A	Reserved
0	DONE	1	RO	Indicates if a transaction is finished. When Writing SPIDTR register, this bit is set to 0. After all data are shifted out, this bit is set to 1

4.1.3 IER: Interrupt enable register

Register Address: 28h

Bit(s)	Field	Default Value	Access Type	Description
31-1	Reserved	N/A	N/A	Reserved
0	DONE	0	RO	Allows generating an interrupt after a transaction is finished. When set to: 0 = Disabled 1 = Enabled

4.1.4 SRR: Software reset register

Register Address: 40h

Bit(s)	Field	Default Value	Access Type	Description
31-4	Reserved	N/A	N/A	Reserved
3-0	RKEY	N/A	WSC	Write 0xA to this field to cause a soft reset for four AXI clock cycles

4.1.5 SPICR: Control register

Register Address: 60h

Bit(s)	Field	Default Value	Access Type	Description
31:28	Reserved	N/A	N/A	Reserved
27	RBPOL	0	R/W	Read back polarity. 0 = Read back happens on the SCLK assertion edge (from CPOL to ~CPOL transition) 1 = Read back happens on the SCLK de-assertion edge (from ~CPOL to CPOL transition) In SDIO mode, this bit is typically set to the same value as the CPHA bit. In SPI mode, some slaves may require this bit to set a different value then CPHA.
26	CPOL	0	R/W	The SCLK polarity in idle state when SS is de-asserted (high)
25	CPHA	0	R/W	SCLK phase control 0 = The core outputs the first data bit when the SS is asserted. A half SCLK cycle later, the first SCLK assertion edge is sent out. The slave devices latch data on the SCLK assertion edge. The core sends out the rest of data bits on the SCLK de-assertion edge. The core doesn't change the SDIO line on the last SCLK de-assertion edge. 1 = The core sends out the first SCLK assertion edge together with the first data bit a half SCLK cycle later after the SS is asserted. The slave devices latch data on SCLK de-assertion edge. In either case, the core de-asserts the SS line a half SCLK cycle later after the last SCLK de-assertion edge.

24	RW_RD_CS	1	R/W	R/W bit level for read. In most cases, '1' for read and '0' for write.
23:22	Reserved	N/A	N/A	Reserved
21:16	RW_POS	0	R/W	R/W bit position in the transmit data. The R/W bit has to be shifted out before the data bits, so the valid range for this field is TRAN_WIDTH to (DATA_POS+1). If the first bit to shift out is the R/W bit, then this value should be TRAN_WIDTH since the position is 1 based (The LSB of the transmit data is 1, and the MSB of the transmit data is TRAN_WIDTH). When sending out data from the SPIDTR register, the core compares the bit value of the transmit data at the RW_POS location with the RW_RD_CS value, if they match, the core will assert a tri-state control line @ DATA_POS location. A FPGA design should connect this tri-state control line to the output enable port of the SDIO if in SDIO mode. If in SPI mode, this tri-state control line should be left unconnected. If RW_POS = 0, the tri-state control line is disabled.
15:14	Reserved	N/A	N/A	Reserved
13:8	DATA_POS	8h	R/W	Data bits start position in the transmit/receive data. Data bits should be the last to shift out when write in SDIO mode (after address bits and R/W bit). When RW_POS = 0, data bits can be shifted out before the address bits. This field is used to control when to tri-state of the SDIO line. Valid range: TRAN_WIDTH <= DATA_POS <= 1
7:6	Reserved	N/A	N/A	Reserved
5:0	TRAN_WIDTH	10h	R/W	Total bits to transfer. 8<= TRAN_WIDTH <= 32 When RW_POS ≠ 0: TRAN_WIDTH = ADDR_WIDTH + DATA_WIDTH + 1 When RW_POS = 0: TRAN_WIDTH = ADDR_WIDTH + DATA_WIDTH

4.1.6 SPISR: Status register

Register Address: 64h
This is the same as the ISR register

Bit(s)	Field	Default Value	Access Type	Description
31-1	Reserved	N/A	N/A	Reserved
0	DONE	1	RO	Indicates if the transaction is finished. After writing the most significant byte of the SPIDTR register, this bit is set to 0 and the core begins to shift out the SPIDTR register data and shift in data to the SPIDRR register. After all data bits are shifted out/in, this bit is set to 1

4.1.7 SPIDTR: Data transmit register

Register Address: 68h

Bit(s)	Field	Default Value	Access Type	Description
31 - TRAN_WIDTH	TXDATA	0	WO	Data to transfer. MSB will be shifted out first. Data writing to this field will immediately be shifted out if the SPISR.DONE is '1' and the most significant byte is written. Data writing to this field when SPISR:DONE is '0' will be discarded. If the AXI4 wstrb (byte enable) signal is used, the most significant byte must be written last. The data in this register will be shifted out right after the most significant byte is written.
[TRAN_WIDTH–1] - 0	Reserved	N/A	N/A	Reserved

4.1.8 SPIDRR: Data receive register

Register Address: 6Ch

DATA_WIDTH = DATA_POS

Bit(s)	Field	Default Value	Access Type	Description
31 - DATA_WIDTH	Reserved	N/A	N/A	Reserved
[DATA_WIDTH–1] - 0	DATA	0	RO	Data received

4.1.9 SPISSR: Slave select register

Register Address: 70h

The SPI slave select register (SPISSR) contains an active-high, one-hot encoded slave select vector SS of length N, where N is the number of slaves. The SS vector occupies the right-most bits of the register. At most, one bit can be asserted high. This bit denotes the slave with which the local master communicates.

Bit(s)	Field	Default Value	Access Type	Description
31 – N	Reserved	N/A	N/A	Reserved
[N-1] – 0	SS	1	R/W	Active-high, one-hot encoded slave select vector of length N bits. The slaves are numbered right to left starting at zero with the LSB. The slave numbers correspond to the indexes of the signal SS.