








This version (23 Feb 2017 12:29) was **approved** by larsc.
The [Previously approved version](#) (22 Feb 2017 18:16) is available. 

AD-FMCADC4-EBZ FMC Board

Introduction

The  [EVAL-AD-FMCADC4-EBZ](#) is a high speed four channel data acquisition board featuring two  [AD9680](#) dual channel ADC at 1000 [MSPS](#) (1240 [MSPS](#)) and four [ADA4961](#)  low distortion, 3.2 [GHz](#), RF DGA driving each converter. The FMC form factor supports the JESD204B high speed serial interface. All clocking and channel synchronization is provisioned on-board using the [AD9528](#)  [AD9528](#) clock generator. This board meets most of the FMC specifications in terms of mechanical size, mounting hole locations etc., for further details, please refer to the FMC specification.

Although this board does meet most of the FMC specifications, it is not meant as a  [commercial off the shelf](#) (COTS) board. If a commercial, ready to go integrate product is required, please refer to one of the many FMC manufacturers.










[ADI](#) also provides reference designs (HDL and software) for this board to work with commonly available Altera and Xilinx development boards.

Hardware

The AD-FMCADC4-EBZ board's primary purpose is to demonstrate the capabilities of the devices on board quickly and easily by providing a seamless interface to an FMC carrier platform and running the reference design on the carrier FPGA. The board is designed to self power and self clock when connected to the FMC carrier. The analog signals (up to four) are connected to J301A, J301B, J301C and J301D. This rapid prototyping board can also be synchronized across channels.

Devices

The FMC board includes the following products by Analog Devices:

-  [AD9680](#) 14-bit dual channel ADC with sampling speeds of up to 1250 [MSPS](#), with a  [JESD204B](#) digital interface.
-  [ADA4961](#) Low Distortion, 3.2 [GHz](#), RF Digital Gain Amplifier.
-  [AD9528](#) JESD204B Clock Generator with 14 LVDS Outputs
-  [ADP2384](#) 20 [V](#), 4 [A](#), Synchronous, Step-Down DC-to-DC Regulator
-  [ADP7104](#) is a 20V, 500mA, low noise, CMOS LDO
-  [ADM7154](#) 600 [mA](#), Ultra Low Noise, High PSRR, RF Linear Regulator
-  [ADM7172](#) 6.5 [V](#), 2 [A](#), Ultralow Noise, High PSRR, Fast Transient Response CMOS LDO
-  [ADP1741](#) is a 2A, low V_{in} , low dropout, CMOS linear regulator



Top View

Table of Contents

- ♦ [AD-FMCADC4-EBZ FMC Board](#)
 - ♦ [Introduction](#)
 - ♦ [Hardware](#)
 - ♦ [Devices](#)
 - ♦ [Clocking](#)
 - ♦ [Analog Front End](#)
 - ♦ [Revision A](#)
 - ♦ [Running No-OS Application & Changing Sampling Rate to 1.24GHz](#)
 - ♦ [Downloads \(Hardware\)](#)
 - ♦ [Downloads \(HDL\)](#)
 - ♦ [FMCADC4](#)
 - ♦ [Help & Support](#)
 - ♦ [Downloads \(Linux\)](#)





Bottom View

Clocking

The AD-FMCADC4-EBZ includes an on-board 80MHz reference oscillator from Crystek. This feature can be disconnected and an external reference can be applied through J901. When referencing the schematic make sure the proper component changes are made in order to directly route the input into the AD9528.

Analog Front End

The AD-FMCADC4-EBZ uses a passive front end designed for very wide bandwidth. A single ended input needs to be provided to the analog inputs mentioned earlier. A 1:2 impedance ratio broadband balun then converts the input signal differentially to the ADA4961 inputs and has a 1.6GHz bandwidth at -3dB. Each channel amplifier can be adjusted independently in terms of gain.

Revision A

The revision A board supports amplifier gain control via spi. After power-up, the gain of the amplifier defaults to an attenuated state. Use a low jitter, low noise signal source with a level at -20dBm to the analog inputs (J301-A/B/C/D). Apply a signal source no greater than -10dBm to achieve full-scale of the converter when maximum gain of the amplifier is applied.

Running No-OS Application & Changing Sampling Rate to 1.24GHz

The HDL reference design is built around a processor as in an embedded system. You may use either Linux or No-OS software to demonstrate the design (details in the downloads section). In order to run the HDL with the No-OS application, first we need to build the HDL bit file and software elf file.

At the time of this writing, we are using the 'dev' branch for both. The [HDL user guide](#) contains the instructions to build the bit file. **Please make sure you use the 'dev' branch (checkout dev right after cloning).**

Once the bit file is ready, follow these instructions to build the elf file. This assumes you are following our directory structures. If you are not, just get the idea from here and port it to your environment. However you have to figure out things on your own.

1. Clone [No-OS](#) repository
2. Checkout the 'dev' branch (git checkout dev)
3. Change the directory to `ad-fmcadc4-ebz/zc706`.
4. Make the elf file by running `make HDF-FILE=~/projects/fmcadc4/zc706/fmcadc4_zc706.sdk/system_top.hdf`

The make will build the default 'hello-world', but we only need the bsp and I am no fan of eclipse, hence this method. If you are more comfortable with the [GUI](#), import all the files (or folders) that the make uses.

A typical run looks like this:

```
[~/github/noos/ad-fmcadc4-ebz/zc706]> make HDF-FILE=~/github/hdl/projects/fmcadc4/zc706/fmcadc4_zc706.sdk/system_top.hdf
xsct -s ../../scripts/xilinx_xsct.tcl ~/github/hdl/projects/fmcadc4/zc706/fmcadc4_zc706.sdk/system_top.hdf >> xilinx_xsct.log 2
>&1

arm-xilinx-eabi-gcc -DXILINX -Ibsp/ps7_cortexa9_0/include -I. -I../../common_drivers/adc_core -I../../common_drivers/jesd204b_
gt -I../../common_drivers/jesd204b_v51 -I../../common_drivers/xilinx_platform_drivers -I../../drivers/ad9528 -I../../drivers/ad
9680 -Os -ffunction-sections -fdata-sections -o zc706.elf sw/src/platform.c ../ad_fmcadc4_ebz.c ../../common_drivers/adc_core/a
dc_core.c ../../common_drivers/jesd204b_gt/jesd204b_gt.c ../../common_drivers/jesd204b_v51/jesd204b_v51.c ../../common_drivers/
xilinx_platform_drivers/platform_drivers.c ../../drivers/ad9528/ad9528.c ../../drivers/ad9680/ad9680.c -Lbsp/ps7_cortexa9_0/lib
/ -Tsw/src/lscript.ld -Wl,--start-group,-lxil,-lgcc,-lc,--end-group
[~/github/noos/ad-fmcadc4-ebz/zc706]
```

Start an [UART](#) terminal.

```
[USB0]
port = /dev/ttyUSB0
speed = 115200
bits = 8
stopbits = 1
parity = none
crlfauto = True ## if not set, expect non-aligned text
```

The folder contains a zc706.tcl file that you can launch with xmd. You can also run it using Vivado or SDK - up to you.

```
[~/github/noos/ad-fmcadc4-ebz/zc706]> xmd -tcl zc706.tcl
rlwrap: warning: your is 'xterm' but rlwrap couldn't find it in the terminfo database. Expect some problems.

Xilinx Microprocessor Debugger (XMD) Engine
XMD v2015.2 (64-bit)
SW Build 1266856 on Fri Jun 26 16:35:25 MDT 2015
** Copyright 1986-2015 Xilinx, Inc. All Rights Reserved.

Executing user script : zc706.tcl
Configuring Device 2 (xc7z045) with Bitstream -- hw/system_top.bit

.....10.....20.....30.....40.....50.....6
0.....70.....80.....90.....Done
Successfully downloaded bit file.

JTAG chain configuration
-----
Device  ID Code      IR Length  Part Name
  1      4ba00477         4      arm_dap
  2      23731093         6      xc7z045

JTAG chain configuration
-----
Device  ID Code      IR Length  Part Name
  1      4ba00477         4      arm_dap
  2      23731093         6      xc7z045

-----
Enabling extended memory access checks for Zynq.
Writes to reserved memory are not permitted and reads return 0.
To disable this feature, run "debugconfig -memory_access_check disable".

-----

CortexA9 Processor Configuration
-----
Version.....0x00000003
User ID.....0x00000000
No of PC Breakpoints.....6
No of Addr/Data Watchpoints.....4

Connected to "arm" target. id = 64
Starting GDB server for "arm" target (id = 64) at TCP port no 1234
Processor stopped

Processor Reset .... DONE

Downloading Program -- zc706.elf
  section, .text: 0x00100000-0x0010656b
  section, .init: 0x0010656c-0x00106583
  section, .fini: 0x00106584-0x0010659b
  section, .rodata: 0x0010659c-0x00106927
  section, .data: 0x00106928-0x00106e9b
  section, .eh_frame: 0x00106e9c-0x00106e9f
  section, .mmu_ttbl: 0x00108000-0x0010bfff
  section, .ARM.exidx: 0x0010c000-0x0010c007
  section, .init_array: 0x0010c008-0x0010c00f
  section, .fini_array: 0x0010c010-0x0010c013
  section, .bss: 0x0010c014-0x0010c0a7
  section, .heap: 0x0010c0a8-0x0010e0af
  section, .stack: 0x0010e0b0-0x001118af
Download Progress..10.20.30.40.50.60.70.80.90.Done
Setting PC with Program Start Address 0x00100000
Processor started. Type "stop" to stop processor

RUNNING> Disconnected from Target 64

Disconnected from Target 352
```

The following messages should appear on the terminal.

```
AD9680 PLL is locked.
AD9680 successfully initialized.
AD9680 PLL is locked.
AD9680 successfully initialized.
JESD204B successfully initialized.
ADC Core Initialized (1000 MHz).
ADC Core Initialized (1000 MHz).
Initialization done.

Capture done.
```

A brief background information on what is happening. Let's look at the `No-OS` main function. First, it configures and sets the `GPIO` based on the board.

```
adc4_gpio_ctl(GPIO_DEVICE_ID);
```

The clock chip is programmed to output the desired clocks and sys-ref signals. The default setting is 1GHz for the AD9680 and 500MHz for the FPGA.

```
ad9528_setup(SPI_DEVICE_ID, 0, ad9528_pdata_lpc);
```

The transceiver cores are initialized. Here only DRP access is possible. If you are planning to change the transceivers, this is where they should be.

```
jesd204b_gt_initialize(FMCADC4_GT_BASEADDR, 8);
```

The AD9680 devices are initialized (checking the `PLL` status)

```
ad9680_setup(SPI_DEVICE_ID, 1);
ad9680_setup(SPI_DEVICE_ID, 2);
```

The design uses Xilinx's JESD IP- it needs to be programmed to match the device settings (frame count, byte count, scrambling and such).

```
jesd204b_setup(AD9680_JESD_BASEADDR, jesd204b_st);
```

After the above setup, bring the transceivers up, here we check for everything on the link, starting from the `PLL` locked to `SYNC` deasserted.

```
jesd204b_gt_setup(ad9680_gt_link);
```

The individual AD9680 cores are brought out of reset.

```
adc_setup(ad9680_0, 2);
adc_setup(ad9680_1, 2);
```

The ADC has a PRBS generator at the sample level that can be monitored in the FPGA. This is a robust way to confirming the link status. The software monitors this and reports any errors.

This is setting the PRBS generator in the device.

```
ad9680_spi_write(1, AD9680_REG_DEVICE_INDEX, 0x3);
ad9680_spi_write(1, AD9680_REG_ADC_TEST_MODE, 0x05);
ad9680_spi_write(1, AD9680_REG_OUTPUT_MODE, 0);
ad9680_spi_write(2, AD9680_REG_DEVICE_INDEX, 0x3);
ad9680_spi_write(2, AD9680_REG_ADC_TEST_MODE, 0x05);
ad9680_spi_write(2, AD9680_REG_OUTPUT_MODE, 0);
```

This is setting the PRBS monitors in the FPGA.

```
adc_pn_mon(ad9680_0, 2, 1);
adc_pn_mon(ad9680_1, 2, 1);
```

If you don't see any other messages in the `UART` other than the ones mentioned above- all is well. You can open up Vivado and see things in ILA also.

Let's see now how we can change the sampling rate to 1.24 GHz. The AD9680 maximum sampling rate is 1.25GHz. However the board uses a 80MHz crystal as the reference clock to AD9528. Unless you change it, this limits the maximum clock output on the banks to 1.24GHz. Also note that the Kintex 7 SOC on ZC706 is a -2 device. The maximum lane rate is limited to 10Gbps. However, it should be possible to over clock the transceiver (but do so at your own risk). Officially, you must get a -3 device to run the link at 12.4Gbps.

Going back to our program.

```
#ifdef MODE_1_24G
    ad9528_pdata_lpc.pll2_ndiv_a_cnt = 1;
    ad9528_pdata_lpc.pll2_ndiv_b_cnt = 23;
    ad9528_pdata_lpc.pll2_n2_div = 31;
    ad9528_pdata_lpc.pll2_vco_diff_m1 = 3;
#endif
```

Let's define that macro somewhere on that file.

```
[~/github/noos/ad-fmcd4-ebz/zc706]> gitdiff.pl ../ad_fmcd4_ebz.c
71a72
> #define MODE_1_24G
```

And re-run the make.

```
[~/github/noos/ad-fmcd4-ebz/zc706]> make HDF-FILE=~/.github/hdl/projects/fmcd4/zc706/fmcd4_zc706.sdk/system_top.hdf
arm-xilinx-eabi-gcc -DXILINX -Ibsp/ps7_cortexa9_0/include -I. -I../common_drivers/adc_core -I../common_drivers/jesd204b_
gt -I../common_drivers/jesd204b_v51 -I../common_drivers/xilinx_platform_drivers -I../drivers/ad9528 -I../drivers/ad
9680 -Os -ffunction-sections -fdata-sections -o zc706.elf sw/src/platform.c ../ad_fmcd4_ebz.c ../common_drivers/adc_core/a
dc_core.c ../common_drivers/jesd204b_gt/jesd204b_gt.c ../common_drivers/jesd204b_v51/jesd204b_v51.c ../common_drivers/
xilinx_platform_drivers/platform_drivers.c ../drivers/ad9528/ad9528.c ../drivers/ad9680/ad9680.c -Lbsp/ps7_cortexa9_0/lib
/ -Tsw/src/lscript.ld -Wl,--start-group,-lxil,-lgcc,-lc,--end-group
```

The UART should now show this.

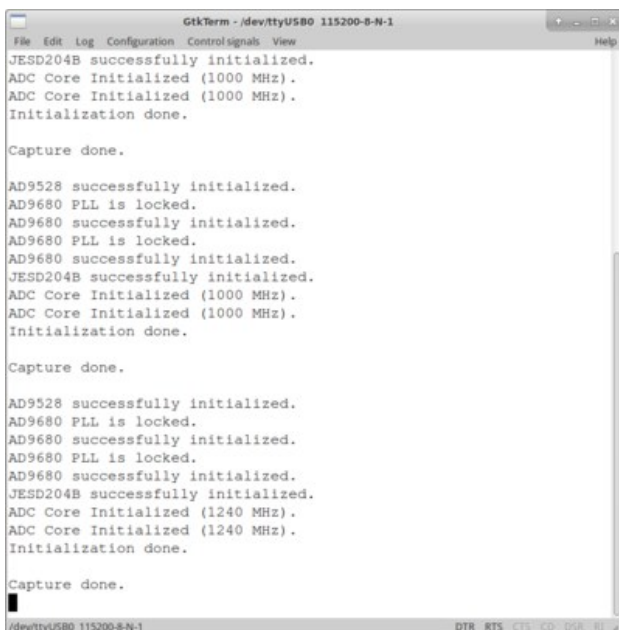
```
AD9528 successfully initialized.
AD9680 PLL is locked.
AD9680 successfully initialized.
AD9680 PLL is locked.
AD9680 successfully initialized.
JESD204B successfully initialized.
ADC Core Initialized (1240 MHz).
ADC Core Initialized (1240 MHz).
Initialization done.

Capture done.
```

The clocks reported by the core is 1240MHz instead of the previous 1000MHz. There are no error messages and PRBS locks. This is all there is to it. There is no need for HDL modifications. However, if you ran into trouble here are a couple of things to try.

- If you have ran this back to back- try running 1.24GHz option from power up.
- Modify the HDL to use a -3 device and change the constraints to run at 12.40Gbps
- Upgrade the device on board to a -3 device.

Here is the UART window screen capture.



```
GtkTerm - /dev/ttyUSB0 115200-8-N-1
File Edit Log Configuration Control signals View Help
JESD204B successfully initialized.
ADC Core Initialized (1000 MHz).
ADC Core Initialized (1000 MHz).
Initialization done.

Capture done.

AD9528 successfully initialized.
AD9680 PLL is locked.
AD9680 successfully initialized.
AD9680 PLL is locked.
AD9680 successfully initialized.
JESD204B successfully initialized.
ADC Core Initialized (1000 MHz).
ADC Core Initialized (1000 MHz).
Initialization done.

Capture done.

AD9528 successfully initialized.
AD9680 PLL is locked.
AD9680 successfully initialized.
AD9680 PLL is locked.
AD9680 successfully initialized.
JESD204B successfully initialized.
ADC Core Initialized (1240 MHz).
ADC Core Initialized (1240 MHz).
Initialization done.

Capture done.
```

The application leaves the device in a ramp pattern, and if you are looking at the data using ILA should see it. If you would like to switch it to the analog input, do the following. In this case I am changing only the fourth channel (SMA - J301D).

```

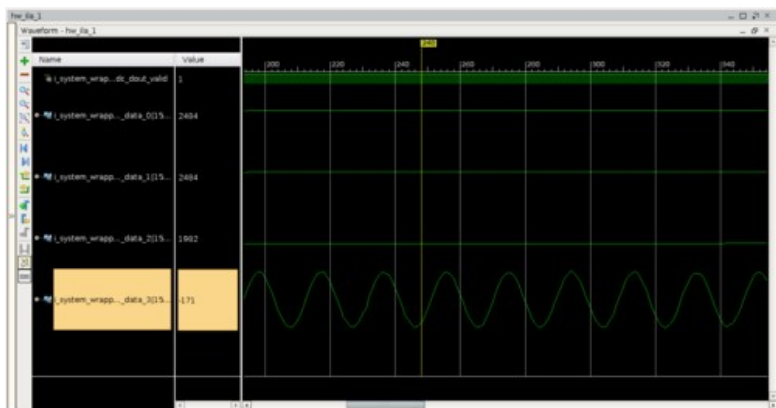
ad9680_spi_write(1, AD9680_REG_DEVICE_INDEX, 0x3);
ad9680_spi_write(1, AD9680_REG_ADC_TEST_MODE, 0x0F);
ad9680_spi_write(1, AD9680_REG_OUTPUT_MODE, 0x1);

ad9680_spi_write(2, AD9680_REG_DEVICE_INDEX, 0x3);
ad9680_spi_write(2, AD9680_REG_ADC_TEST_MODE, 0x0F);
ad9680_spi_write(2, AD9680_REG_OUTPUT_MODE, 0x1);

ad9680_spi_write(2, AD9680_REG_DEVICE_INDEX, 0x2);
ad9680_spi_write(2, AD9680_REG_ADC_TEST_MODE, 0x00);
ad9680_spi_write(2, AD9680_REG_OUTPUT_MODE, 0x1);
adc_write(ad9680_1, ADC_REG_CHAN_CNTRL(1), 0x51);

```

Here is the ILA plot screen capture.



Downloads (Hardware)

Rev A:



- [Schematic](#)
- [Bill of Materials](#)
- [PCBoard Fab Drawing](#)
- [PCBoard Gerber files](#)

Downloads (HDL)

FMCADC4

Hardware	Project	Carriers	Library Cores
----------	---------	----------	---------------

AD-FMCADC4-EBZ	fmcadc4	zc706	
--------------------------------	-------------------------	-----------------------	--

			axi_ad9680
--	--	--	----------------------------

			axi_dmac
--	--	--	--------------------------

			util_axis_ffio
--	--	--	--------------------------------

			util_axis_resize
--	--	--	----------------------------------

			util_cpack
--	--	--	----------------------------

			axi_jesd_gt
--	--	--	-----------------------------

			util_jesd_gt
--	--	--	------------------------------

			util_bsplitt
--	--	--	------------------------------

			util_mfifo
--	--	--	----------------------------

			axi_clkgen
--	--	--	----------------------------

			axi_hdmi_tx
--	--	--	-----------------------------

			axi_spdif_tx
--	--	--	------------------------------

			axi_adcffio
--	--	--	-----------------------------



Help & Support



- The [carriers](#) (abbreviations can be found [here](#)) are commonly available FPGA evaluation boards.
- The [HDL user guide](#) contains all the documentation, build instructions and register map tables.
- The following quick links allows you to browse the github repository for a list of current [branches](#), [library components](#), and [projects](#).
- Questions? We can help with [FPGA questions](#), [Linux driver questions](#), [No-OS Drivers](#).

Downloads (Linux)

- [JESD204B Linux Driver](#)
- [AD9680-ADA4961 Linux driver](#)
- [ZC706 Linux image](#)

resources/eval/user-guides/ad-fmcdac4-ebz.txt · Last modified: 22 Feb 2017 19:23 by larsc

15,000

Problem Solvers

4,700+

Patents

125,000

Customers

50+

Years

Ahead of What's Possible

ADI enables our customers to interpret the world around us by intelligently bridging the physical and digital with unmatched technologies that sense, measure and connect. We collaborate with our customers to accelerate the pace of innovation and create breakthrough solutions that are ahead of what's possible.

[See the Innovations](#)

Analog Devices. Dedicated to solving the toughest engineering challenges.

SOCIAL



QUICK LINKS

- | | |
|---|--|
| About ADI | Analog Dialogue |
| Careers | Contact us |
| Investor Relations | News Room |
| Quality & Reliability | Sales & Distribution |

LANGUAGES

- [English](#)
- [简体中文](#)
- [日本語](#)
- [Русский](#)

NEWSLETTER

Interested in the latest news and articles about ADI products, design tools, training and events? Choose from one of our 12 newsletters that match your product area of interest, delivered monthly or quarterly to your inbox.

[Sign Up](#)