4FM

Getting Started Guide

Microsoft Windows

4DSP LLC

Email: support@4dsp.com

Table of Contentswi

Revision History

Date	Revision	Revision
2013-04-26	- Replaced Visual Studio 2008 portions with Visual Studio 2012 - Corrected a few typos.	4.5
2013-06-11	Added FMC667 support	4.6
2013-08-22	Improved Bit2Hex	4.7
2013-09-18	- Replaced ModelSim with ISim - Added 4FM Linux installer path - Removed the reference firmware matrix - Added 4FM FMC Analyzer (plugin and application)	4.8
2013-10-18	- Replaced old StellarIP with the new StellarIP. - Modified training material sections. - Reshaped chapter ordering.	4.9
2013-10-23	Added more cards support in 4FM FMC Analyzer	4.10
2013-11-18	Added importing legacy StellarIP design file importing	4.11
2013-12-10	Added FM780/VP780 in the manual	4.12
2014-02-14	Added paths to Linux Calibration for FMC126 as well as path to the Zynq on Linux elements	4.13
2014-03-13	Added more available firmware/FMC cards to the FMC Analyzer selection	4.14
2014-03-21	- Reworded several descriptions - Corrected a few typos	4.15

1 Introduction

The 4FM “Getting Started Guide” describes the steps that a user must perform in order to use the 4FM development kit.

Microsoft Windows 7 is required to install/use the development kit. The BSP is not tested by 4DSP under Windows 2000 or Windows XP anymore because most of our customers are using Windows 7.

The 4FM SDK is a set of tools and interfaces designed primarily to communicate with the various hardware manufactured by 4DSP. Additionally, Xilinx firmware projects can be created from the 4DSP firmware packages using the StellarIP software.

The 4FM SDK also includes support for the FMC product line, and specific chapters are dedicated to their description.

2 Requirements and handling instructions

2.1 Hardware requirements and handling instructions

Please refer to the user manual shipped with the hardware you purchased and strictly follow the handling instructions. Faulty handling may seriously damage the card and void the warranty. The most important of these handling instructions concerns the allowed operating temperatures. The devices should not be used at a temperature exceeding 4DSP specifications.

2.2 Software requirements

- 4DSP's StellarIP software relies on Xilinx software tools to prepare and compile Xilinx projects. An operating Xilinx ISE design suite is required.
- ISim is required to simulate projects created by StellarIP and compiled using Xilinx tools. Different simulators can be used to simulate the design, but the various ISim macros should therefore be rewritten in consequence.
- Xilinx simulation libraries for other simulators should be compiled using Xilinx's "Simulation Library Compilation Wizard".
- 4FM SDK requires Windows 7.
- 4FM and ETHAPI APIs require Visual Studio 2012 to be properly installed and configured.
- The various pieces of documentation that accompany your 4DSP product are provided as a set of PDF documents. There are no hardcopies of the documentation.

2.3 Software requirements (Visual Studio 2012)

To compile an application using 4DSP APIs, you must first integrate the API into Visual Studio as described below.

Launch Visual Studio and open a project from the “FILE” menu.

Then, open the “Property Manager” view from the “VIEW” > “Other Windows” menu.

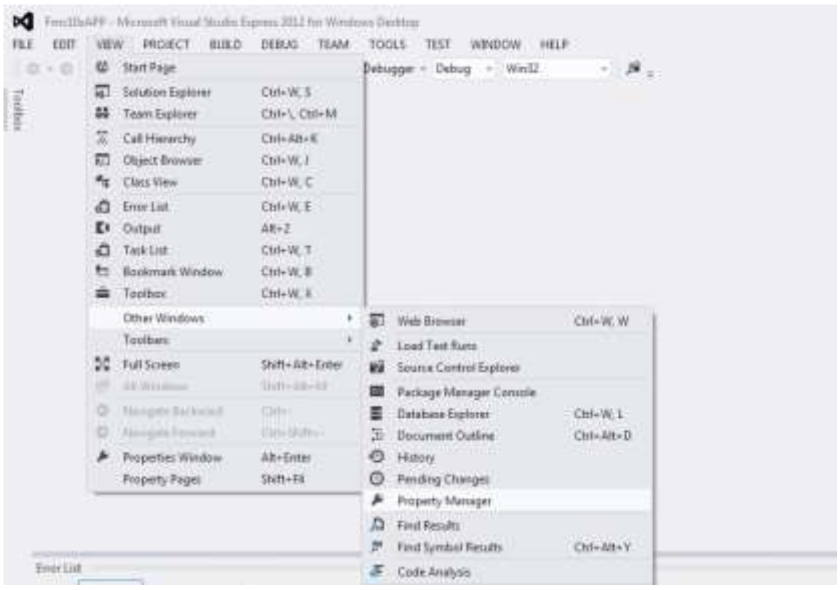


Figure 1: The "Property Manager" view is available from the "VIEW" menu.

Double-clicking on “Microsoft.Cpp.Win32.user” from either “Debug|Win32” or “Release|Win32” opens the “Microsoft.Cpp.Win32.user Property Pages” where you can configure both “Include Directories” and “Library Directories”.

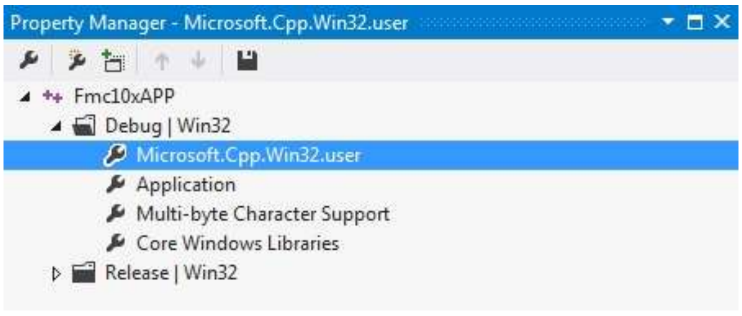


Figure 2: The "Property Manager" view

The 4FM SDK Core installer installs these two folders at the following paths:

(32-bit system)

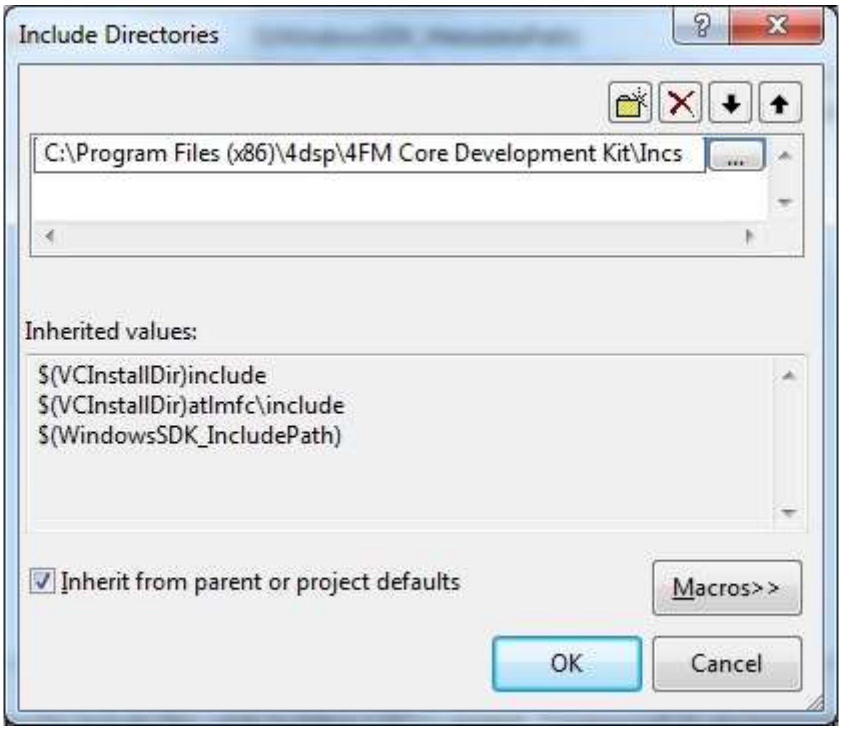
Libraries: *C:\Program Files\4DSP\4FM Core Development Kit\Libs*

Includes: *C:\Program Files\4DSP\4FM Core Development Kit\Incs*

(64-bit system):

Libraries: *C:\Program Files (x86)\4DSP\4FM Core Development Kit\Libs*

Includes: *C:\Program Files (x86)\4DSP\4FM Core Development Kit\Incs*



C:\Program Files (x86)\4dsp\4FM Core Development Kit\Incs

$(VCInstallDir)include
$(VCInstallDir)atlmfc\include
$(WindowsSDK_IncludePath)

☑ Inherit from parent or project defaults

Figure 3: Define a folder for "Include Files"

3 Software Installation

The 4FM installer is available as a .zip archive from our website. Download and decompress the file onto your hard drive.

You will be asked to browse for a license file during the installation. The license file should have been sent to you in an email. It is not possible to continue without a license file.

4DSP device drivers for PCI (PMC/XMC) and Ethernet are digitally signed and can be installed and loaded on both 32 and 64-bit Windows 7 variants.

3.1 4FM Linux (Optional)

After installation of the 4FM BSP, the 4DSP BSP archive for Linux is available for customers who have purchased the Linux BSP at the following paths:

(32-bit system): *C:\Program Files\4dsp\4FM Core Development Kit\Linux*

(64-bit system): *C:\Program Files (x86)\4dsp\4FM Core Development Kit\Linux*

3.2 4FM FMC126 Calibration Linux (Optional)

After installation of the 4FM BSP, the Calibration archive for Linux is available for customers who have purchased the FMC126 calibration at the following paths:

(32-bit system): *C:\Program Files\4dsp\4FM Core Development Kit\Linux*

(64-bit system): *C:\Program Files (x86)\4dsp\4FM Core Development Kit\Linux*

3.3 FMC On Zynq – Linux (Optional)

After installation of the 4FM BSP, the Zynq on Linux material is available for customers who have purchased the FMC126 calibration at the following paths:

For ZC706:

(32-bit system): *C:\Program Files\4dsp\FMC Board Support Package\Linux on Zynq\ZC706*

(64-bit system): *C:\Program Files (x86)\4dsp\FMC Board Support Package\Linux on Zynq\ZC706*

For ZC702:

(32-bit system): *C:\Program Files\4dsp\FMC Board Support Package\Linux on Zynq\ZC702*

(64-bit system): *C:\Program Files (x86)\4dsp\FMC Board Support Package\Linux on Zynq\ZC702*

Follow the directions found in the *FMC on ZC706-Linux* user manual for more information.

4 Development kit components description

A few components are available after a successful installation. The license file browsed during the installation phase dictates which components are installed. A component can be a different application or a plug-in. A plug-in is a software component that is executed by the GUI control application. The components are described below.

4.1 Software Overview

This section gives a hierarchical view of all the software components installed for both PCI and Ethernet based devices.

4.1.1 PCI/PCI Express installation

After installing the BSP with a license file for a PCI/PCI Express product, the following components will be installed. A PCI bus is available for most 4DSP hardware as well as for Xilinx products such as the ML605, KC705, VC707, and VC709.

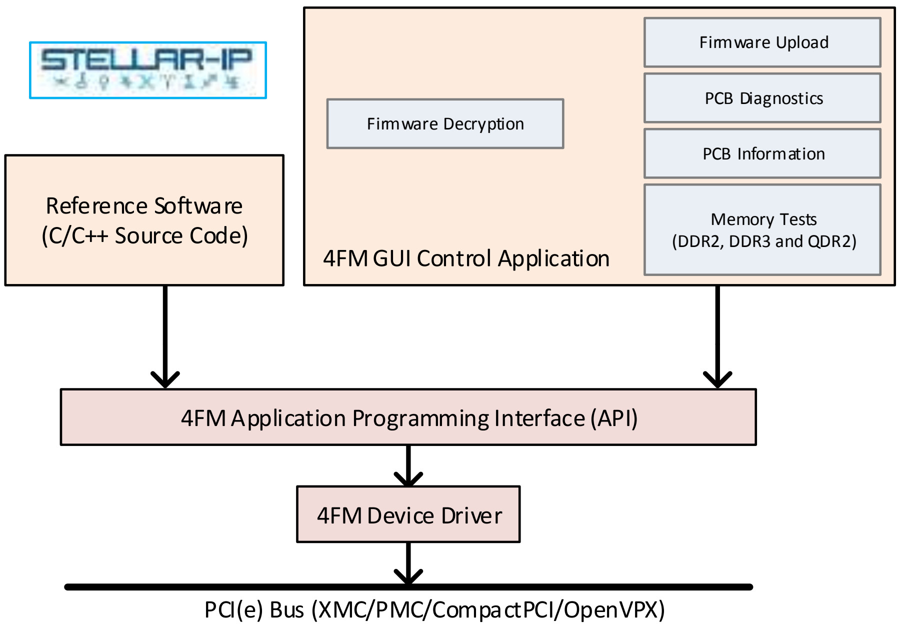

STELLAR-IP
Firmware Upload
PCB Diagnostics
PCB Information
Memory Tests (DDR2, DDR3 and QDR2)
Firmware Decryption
4FM GUI Control Application
Reference Software (C/C++ Source Code)
4FM Application Programming Interface (API)
4FM Device Driver
PCI(e) Bus (XMC/PMC/CompactPCI/OpenVPX)

Figure 4: Components for a PCI/PCIe installation

4.1.2 Ethernet installation

After installing the BSP with a license file for an Ethernet product, the following components will be installed. The Ethernet interface is mainly used by Xilinx hardware such as VC707, KC705, ML605, SP601, and SP605. 4DSP's FC6301 CompactPCI hardware also takes advantage of the Ethernet interface.

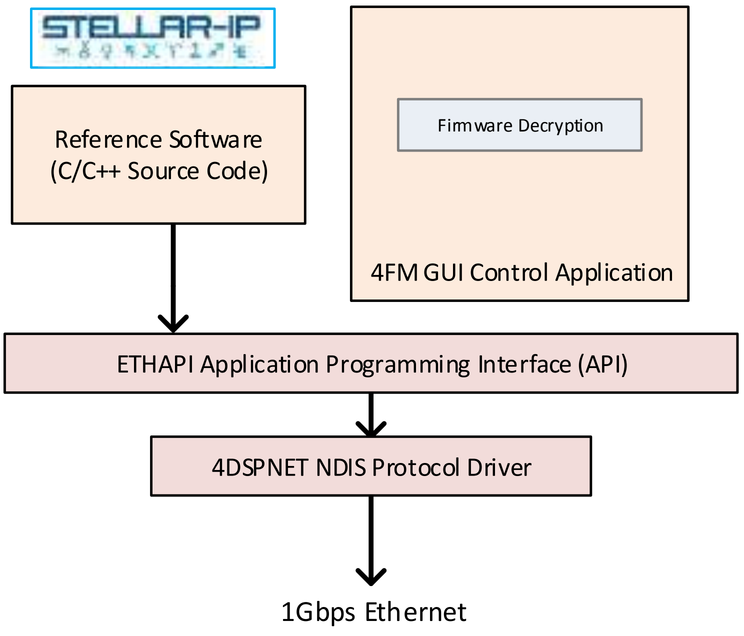

STELLAR-IP
Reference Software (C/C++ Source Code)
Firmware Decryption
4FM GUI Control Application
ETHAPI Application Programming Interface (API)
4DSPNET NDIS Protocol Driver
1Gbps Ethernet

Figure 5: Components for an Ethernet installation

4.1.3 TCP/IP installation

After installing the BSP with a license file for a TCP/IP product, the following components will be installed. The TCP/IP interface is mainly used by Xilinx hardware as ZC702, ZC706, and Zedboard.

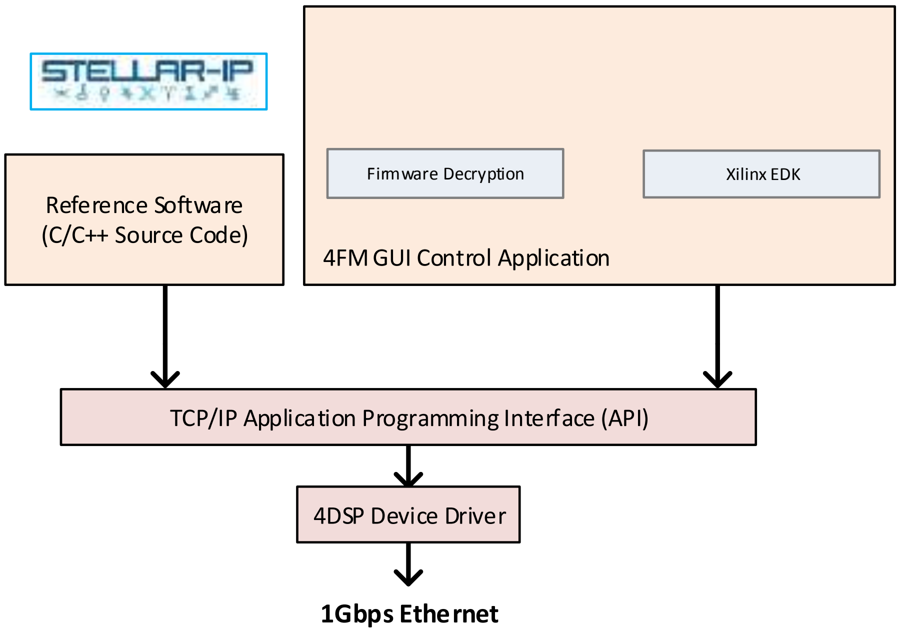

STELLAR-IP
Reference Software (C/C++ Source Code)
Firmware Decryption
Xilinx EDK
4FM GUI Control Application
TCP/IP Application Programming Interface (API)
4DSP Device Driver
1Gbps Ethernet

Figure 6: Components for a TCP/IP installation

4.2 StellarIP overview

The staff at 4DSP has been using a tool called StellarIP internally for quite a while. StellarIP simplifies the design process by providing well-structured, reusable firmware blocks. The advantages of StellarIP are:

- Helps to create well-structured FPGA designs (called “constellation” in StellarIP)
- Improves reusability of firmware blocks (called “star” in StellarIP)
- Has a higher level of abstraction while creating a new constellation (FPGA design)
- Speeds up the design cycle by automating recurring tasks:
  - automated top-level VHDL file creation
  - automated ISE project file creation

It is important to remember the following terminology when working with StellarIP:

-	**Star**	A functional block with a specific task.
-	**Wormhole**	A connection between two stars. A wormhole comprises one or more signals in either one or both directions.
-	**Constellation**	A collection of stars that forms the top level of a firmware design.

StellarIP does not understand HDL or C, but requires the user to interconnect a star on a schematic. StellarIP is able to create a project for the graphical Xilinx ISE and Xilinx Vivado tools.

In other words, the schematic drawing defines a constellation, which is the top level of a firmware. A constellation is made of stars (functional blocks provided by 4DSP and/or created by the user). The stars are interconnected using wormholes. Wormholes are generic communication interfaces defined by 4DSP and/or the user.

Additional information about StellarIP can be obtained from 4DSP's website on the page: FPGA Development Tool (Stellar IP) | 4DSP, LLC. Please refer to the StellarIP user manual for complete details.

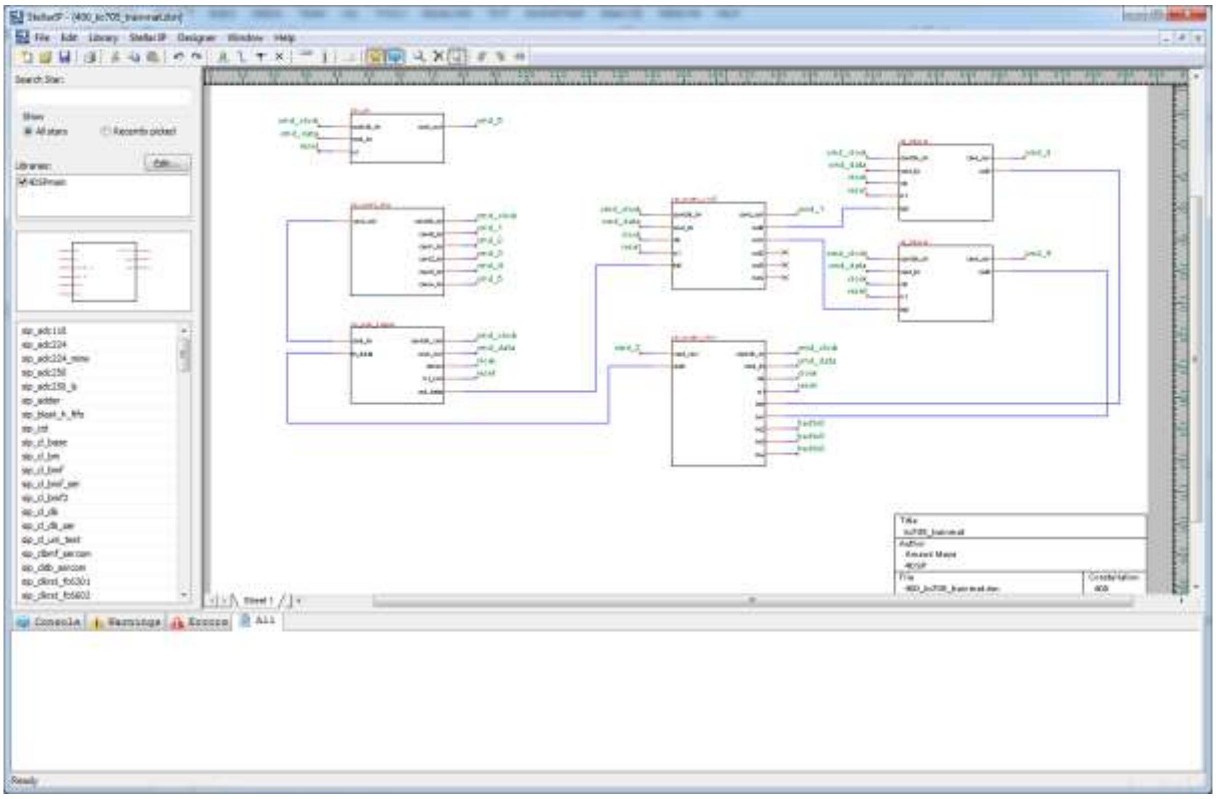


Figure 7: StellarIP

4.3 Firmware overview

4DSP provides firmware source code as an encrypted firmware package with the .4ff file extension (4DSP Firmware File). The firmware indicated by a license file will be automatically decrypted by the 4FM installer during the post-installation phase. A standalone firmware file decryption utility is available within the 4FM GUI Control Application software.

A decrypted folder contains a constellation description, a star library including documentation for every star, and documentation describing the constellation.

The decrypted firmware is converted to a Xilinx ISE project file by StellarIP. After decryption, the flow is a standard FPGA flow that Xilinx users are familiar with.

An overview of StellarIP can be found in Chapter 4.2 StellarIP overview. A detailed description of decrypted firmware architecture can be found in Annex 4 - Firmware .4FF file.

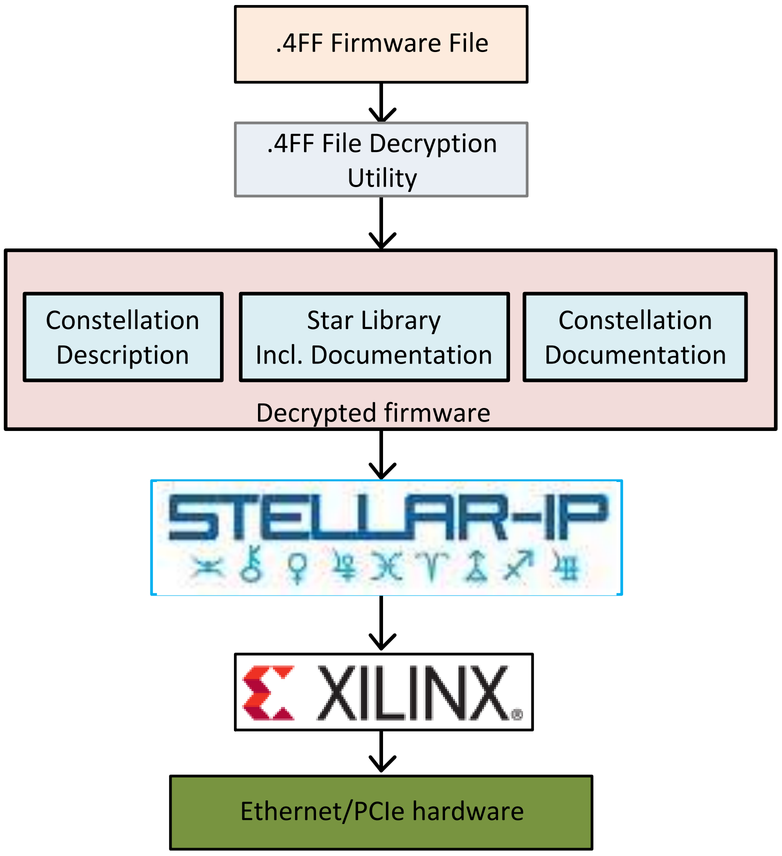

.4FF Firmware File
.4FF File Decryption Utility
Constellation Description
Star Library Incl. Documentation
Constellation Documentation
Decrypted firmware
STELLAR-IP
XILINX
Ethernet/PCIe hardware

Figure 8: Firmware compilation chain

4.4 Documentation

Various documents such as “User Guides” are installed by the SDK installer. They can be found at the following paths:

(32-bit system): *C:\Program Files\4dsp\4FM Core Development Kit\Documentation*

(64-bit system): *C:\Program Files (x86)\4dsp\4FM Core Development Kit\Documentation*

Additionally, documentation for the FMC product line can be found at the following paths:

(32-bit system): *C:\Program Files\4dsp\FMC Board Support Package\Documentation*

(64-bit system): *C:\Program Files (x86)\4dsp\ FMC Board Support Package\Documentation*

***Note**: firmware documentation is part of the firmware package, please refer to the files 4FM Firmware and Annex 4 - Firmware .4FF for more information about the firmware packages.*

4.5 4FM Firmware (Design Files)

The firmware packages are automatically decrypted by the installer and extracted at the following paths:

(32-bit system): *C:\Program Files\4dsp\Common\Firmware\Extracted*

(64-bit system): *C:\Program Files (x86)\4dsp\Common\Firmware\Extracted*

This folder contains one folder per firmware allowed by your license file. Each firmware subfolder includes documentation; StellarIP firmware and its associated test bench; and the recovery files. The recovery files are programming files that are ready to be uploaded to the hardware.

***Note**: if the firmware was not decrypted automatically by the installer, please refer to Annex 1 – Manual Firmware Installation for more information about manually extracting the firmware.*

4.6 4FM Firmware (Recovery Files)

The firmware recovery files are installed with the other BSP components. This provides the user with firmware files ready to be uploaded to the hardware. These files are located at the following paths:

(32-bit system): *C:\Program Files\4dsp\Common\Firmware\Recovery*

(64-bit system): *C:\Program Files (x86)\4dsp\Common\Firmware\Recovery*

4.7 4DSP Drivers

A driver is the main interface between the hardware and the software. This chapter provides some information about the device drivers, the 4FM device driver, and the 4DSPNET protocol driver.

4DSP drivers are digitally signed to allow these software components to be installed for both the 32-bit and 64 bit version of Windows 7. A digitally signed driver is required to load and run flawlessly under 64-bit environments.

4.7.1 4FM PCI/PCIe driver

This is a well-optimized kernel mode driver. An API is available to communicate with the device driver from the user application space.

The device driver is automatically installed by the SDK installer. It can be installed manually if you need to transfer the driver to a target system. The driver files are located at the following paths after a default installation:

(32-bit system): *C:\Program Files\4dsp\4FM Core Development Kit\Driver*

(64-bit system): *C:\Program Files (x86)\4dsp\4FM Core Development Kit\Driver*

4.7.2 4DSPNET NDIS Protocol driver

This driver implements a minimalistic protocol over Ethernet. There is no specific handshaking or data recovery implemented as compared to TCP/IP, for instance.

The device driver is automatically installed by the SDK installer. The driver can be installed manually if you need to transfer the driver to a target system. The driver files are located at the following paths after a default installation:

(32-bit system): *C:\Program Files\4dsp\FMC Board Support Package \Driver*

(64-bit system): *C:\Program Files (x86)\4dsp\FMC Board Support Package\Driver*

4.8 4DSP APIs

Application programming interfaces (API) are provided by 4DSP to bring board level control to the user application space. These APIs are compatible with Microsoft Visual Studio. 4DSP recommends and supports Microsoft Visual Studio 2012.

Other Microsoft Visual Studio versions might be used but downloading and installing the Visual Studio 2012 redistributables package is required. The redistributables package is available as a free download from Microsoft.

4.8.1 4FM (PCI API)

The API communicates with the hardware device through the 4FM device driver installed by the SDK. The API provides all the C functions one might want to call from an application. This API is meant to communicate with 4DSP's PCI or PCIe devices.

The available functions allow obtaining diagnostics, reading/writing firmware registers, and receiving/sending data using direct memory access (DMA) operations.

Note: *Please consult the "4FM Programmers Guide" PDF document for a more detailed description of the 4FM API.*

4.8.2 ETHAPI API (Ethernet API)

The API communicates with the Window's NDIS layer and communicates with the NDIS Ethernet protocol driver. The protocol is minimalistic, so no data recovery or integrity checks are implemented.

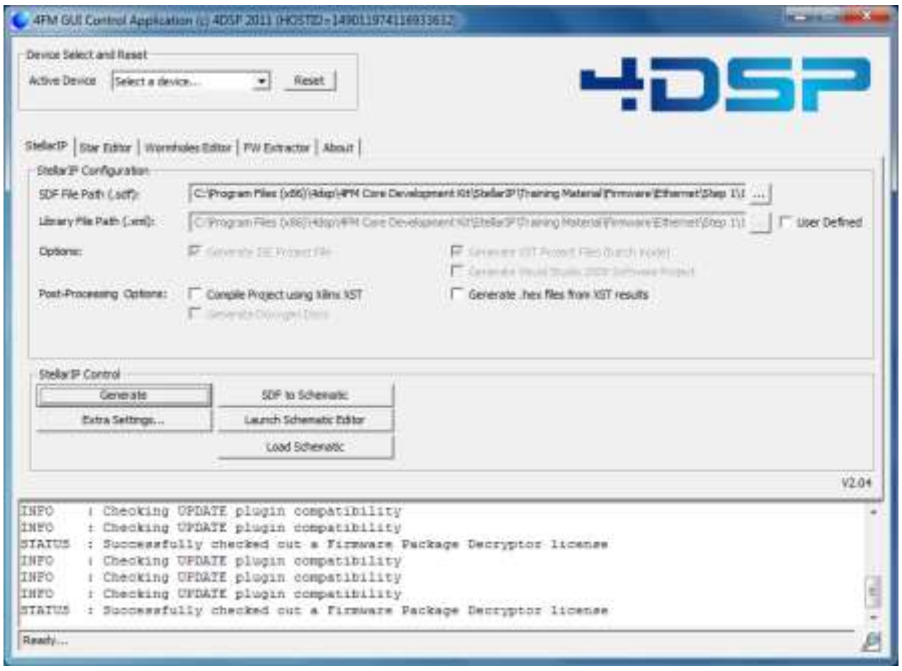
The available functions allow reading/writing firmware registers and receiving/sending data through the Ethernet interface.

Note: *Please consult the "doxyethapi" CHM document for a more detailed description of the ETHAPI API.*

4.9 4FM GUI Control application

The 4FM GUI is a graphical user interface engine that displays plug-in components. It has a few general controls for selecting a device, and it displays the status of various plug-ins.

Controls and details for each plug-in can be accessed by clicking the corresponding tab in the GUI Control Application. Figure 9 shows the application with four plug-ins loaded.


4FM GUI Control Application (c) 4DSP 2011 (HOSTID=149011974116933632)

Device Select and Reset
Active Device: Select a device...

StellarIP Configuration
SDF File Path (.sdf): C:\Program Files (x86)\4dsp\4FM Core Development Kit\StellarIP\Training Material\Firmware\Ethernet\Step 1\
Library File Path (.xml): C:\Program Files (x86)\4dsp\4FM Core Development Kit\StellarIP\Training Material\Firmware\Ethernet\Step 1\
User Defined: ☐
Options: [✓ Generate ISE Project File] [✓ Generate XST Project Files (batch mode)] ☐ Generate Visual Studio 2008 Software Project
Post-Processing Options: ☐ Compile Project using Xilinx XST ☐ Generate .hex files from XST results ☐ Generate Doxygen Docs

V2.04

```
INFO    : Checking UPDATE plugin compatibility
INFO    : Checking UPDATE plugin compatibility
STATUS  : Successfully checked out a Firmware Package Decryptor license
INFO    : Checking UPDATE plugin compatibility
INFO    : Checking UPDATE plugin compatibility
INFO    : Checking UPDATE plugin compatibility
STATUS  : Successfully checked out a Firmware Package Decryptor license
```

Ready...

Figure 9: Main graphical user input dialog

The upper “Device Select and Reset” area allows a user to select a device (FM489, FM680, FM780, FC6301, VP680, VP780, etc.) to control from the “Active Device” dropdown menu. Additionally, a reset command can be sent to the hardware device by clicking the “Reset” button. **Note that devices communicating over Ethernet, such as the ML605, cannot be controlled by this software.**

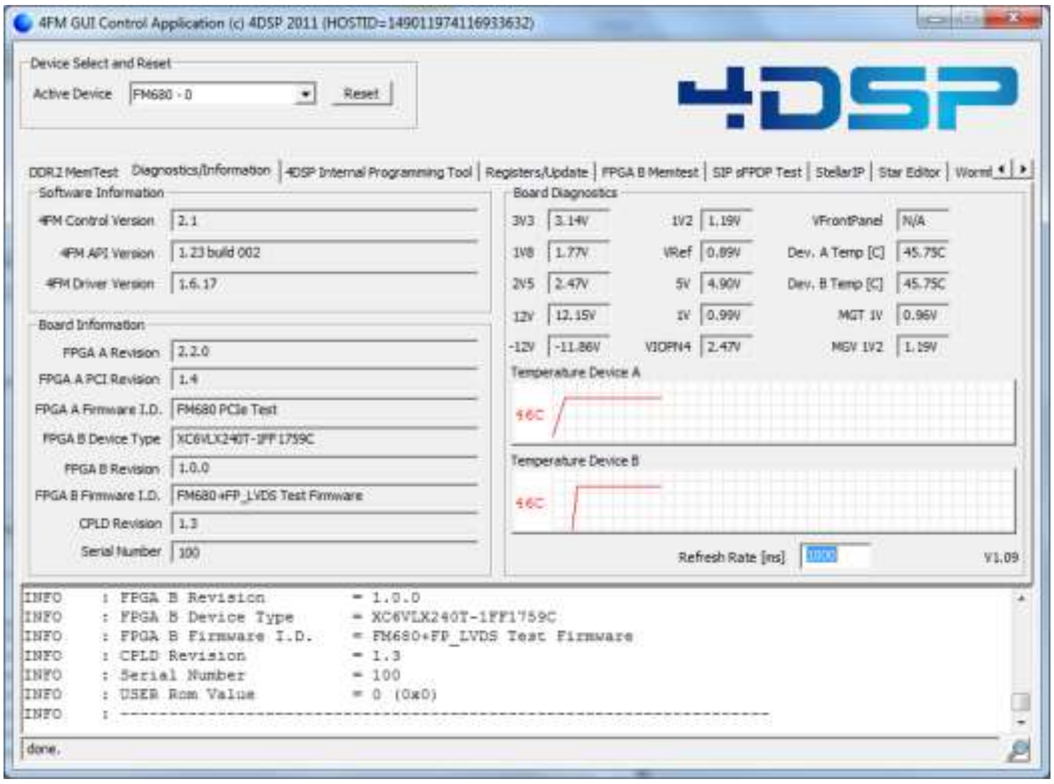
The central “plug-in” area displays all the plug-ins loaded by the graphical user interface. Each plug-in has its own tab. Figure 9 shows the 4FM GUI Control Application with the StellarIP plug-in loaded and active.

The lower “debug” area is a text field that displays various debug messages sent by the plug-ins. StellarIP typically displays crucial information in this area which can be very useful to diagnose malformed StellarIP descriptions.

At the bottom is the “Status” area which displays the current status of the graphical user interface. Clicking the magnifying glass button on the right opens an “extended debug view” which is useful for checking the “debug” logs. This window can be resized as desired.

4.9.1 4FM Diagnostics/Information plug-in (PCI/PCIe hardware)

Figure 10 shows the diagnostics/information plug-in, which gives the user a quick overview of the hardware and software status. A short description of the different status fields follows below.



Active Device: FM680 - 0

4FM Control Version	2.1
4FM API Version	1.23 build 002
4FM Driver Version	1.6.17

FPGA A Revision	2.2.0
FPGA A PCI Revision	1.4
FPGA A Firmware I.D.	FM680 PCIe Test
FPGA B Device Type	XC6VLX240T-1FF1759C
FPGA B Revision	1.0.0
FPGA B Firmware I.D.	FM680+FP_LVDS Test Firmware
CPLD Revision	1.3
Serial Number	100

3V3	3.14V	1V2	1.19V	VFrontPanel	N/A
1V8	1.77V	VRef	0.89V	Dev. A Temp [C]	45.75C
2V5	2.47V	5V	4.90V	Dev. B Temp [C]	45.75C
12V	12.15V	1V	0.99V	MGT 1V	0.96V
-12V	-11.86V	VIOFN4	2.47V	MGV 1V2	1.19V

Refresh Rate [ms] 1000

```
INFO    : FPGA B Revision          = 1.0.0
INFO    : FPGA B Device Type       = XC6VLX240T-1FF1759C
INFO    : FPGA B Firmware I.D.     = FM680+FP_LVDS Test Firmware
INFO    : CPLD Revision            = 1.3
INFO    : Serial Number            = 100
INFO    : USER Rom Value           = 0 (0x0)
INFO    : -------------------------------------------------------------------
```

done.

Figure 10: Diagnostics/Information plug-in view

Software information:

- Graphical User Interface engine version (4FM Control Version)
- Application Programming Interface version (4FM API Version)
- Device driver version (4FM Driver Version)

Board information:

- FPGA A firmware revision (FPGA A Revision) is the firmware version currently loaded in FPGA A.
- FPGA A PCI revision is the version of the PCI interface module also in FPGA A.
- FPGA A firmware I.D. is an identification number in FPGA A firmware.
- FPGA B device type is the FPGA type for FPGA B. FPGA B revision is the version string for the firmware currently active in FPGA B.
- FPGA B firmware I.D. is an identification number in FPGA B firmware.
- CPLD revision is the CPLD firmware’s version.
- Serial Number is the board serial number programmed by the factory before the board is shipped to a customer.

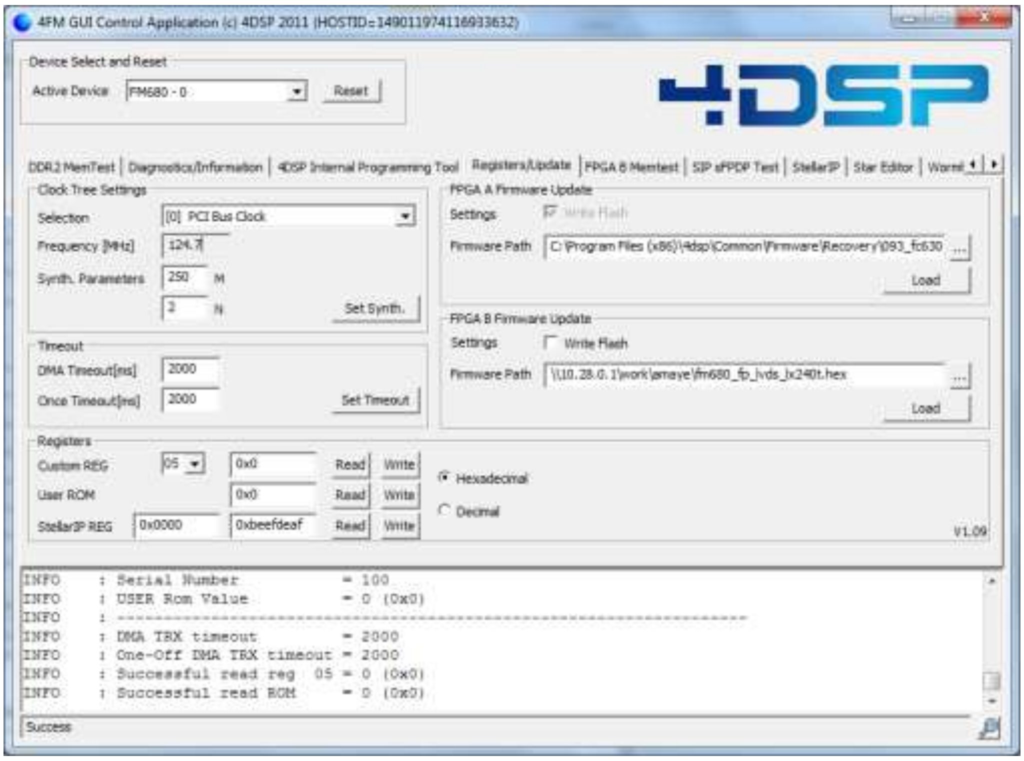
Board diagnostics:

- Voltage readings of the different power rails.
- FPGA A temperature and FPGA B temperature for 4DSP hardware with two FPGAs.
- FPGA temperatures graph.

Note*: Devices communicating over Ethernet, such as the ML605, are* ***not*** *supported by this plug-in!*

4.9.2 4FM Registers/Update plug-in (PCI/PCIe hardware)

This plug-in allows basic interaction with the hardware as well as the loading of new FPGA configurations to the onboard flash. For some boards it is possible to directly upload a new configuration to the FPGA B device.


4FM GUI Control Application (c) 4DSP 2011 (HOSTID=14901197411693632)

Device Select and Reset — Active Device: FM680 - 0

Clock Tree Settings — Selection: [0] PCI Bus Clock; Frequency [MHz]: 124.7; Synth. Parameters: 250 M, 2 N

Timeout — DMA Timeout[ms]: 2000; Once Timeout[ms]: 2000

Registers — Custom REG: 05, 0x0; User ROM: 0x0; StellarIP REG: 0x0000, 0xbeefdeaf; [✓ Hexadecimal] Decimal

FPGA A Firmware Update — Settings: [✓ Write Flash]; Firmware Path: C:\Program Files (x86)\4dsp\Common\Firmware\Recovery\093_fc630

FPGA B Firmware Update — Settings: Write Flash; Firmware Path: \\10.28.0.1\work\amaye\fm680_fp_lvds_lx240t.hex

```
INFO    : Serial Number          = 100
INFO    : USER Rom Value         = 0 (0x0)
INFO    : -----------------------------------------------------------------
INFO    : DMA TBX timeout        = 2000
INFO    : One-Off DMA TBX timeout = 2000
INFO    : Successful read reg  05 = 0 (0x0)
INFO    : Successful read ROM    = 0 (0x0)
```

Success

Reset
DDR2 MemTest | Diagnostics/Information | 4DSP Internal Programming Tool | Registers/Update | FPGA B Memtest | SIP sFPDP Test | StellarIP | Star Editor | Wormi
Set Synth.
Set Timeout
Read
Write
Load
V1.09

Figure 11: Registers/Update plug-in view

Clock Tree Settings:

This area displays frequencies from the clock tree. Choose a clock source using from "Selection" list to display the current frequency for the selected clock. Additionally, the frequency and synthesizer parameters can be adjusted to change the onboard clock synthesizer frequency.

Timeout:

This area allows modification of the default DMA timeout value. The value must be entered in milliseconds. "Once Timeout" sets a timeout value in milliseconds for the next operation only.

Registers:

- "Custom REG" - Reading and writing custom registers (Only available on FM48x hardware series)
- "User ROM" - Reading and writing the 32-bit wide, non-volatile user ROM value

- "StellarIP REG" - Reading and writing StellarIP registers. The text field on the left represents an address and the text field on the right side represents the actual value. A StellarIP register is 32-bit wide and can transfer value to a specific address in the firmware register space.

FPGA A/B Firmware Update:

These areas allow firmware to be programmed into either FPGA A or FPGA B on the 4DSP hardware. Browse to a **.hex** file (or **.pof** for Altera hardware) and press the "Load" button to upload a firmware into the hardware.

Check the "Write Flash" box to program the selected firmware into FLASH memory instead of only programming the FPGA. A power cycle is required after programming the FLASH in order to load the new configuration into the FPGA.

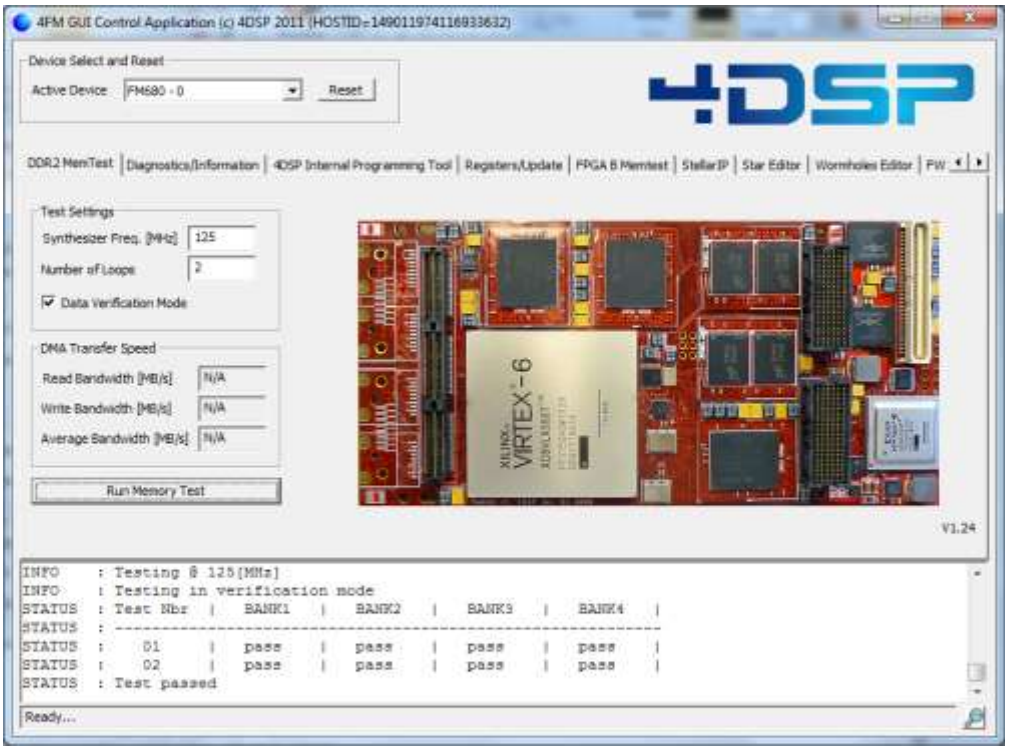
Leave the "Write Flash" check box empty to directly program the FPGA. The new Firmware is ready after programming and does not require a power cycle. This type of programming is volatile and the uploaded firmware will not persist after a power cycle.

Note 1*: Direct FPGA B (to FPGA device) update/upload is not supported by default on FM680 devices. Only a FM680 board with a CPLD version* ***1.3*** *and above can be directly updated from the software. FPGA B update/upload is not supported for VP780, VP680, and FC6301 devices.*

Note 2*: Devices communicating over Ethernet, such as the ML605, are* ***not*** *supported by this plug-in!*

4.9.3 4FM Memory Test plug-ins (PCI/PCIe hardware)

There are several plug-ins to test the BLAST memory on our boards. Three plug-ins are available (DDR2, DDR3, and QDR2 tests) and the correct one is loaded for a given hardware. For example, if you have a board with both DDR2 and QDR2 BLASTs, then two plug-ins will be loaded in the graphical user interface. All three plug-ins have the same interface.


4FM GUI Control Application (c) 4DSP 2011 (HOSTID=149011974116933632)
Device Select and Reset
Reset
DDR2 MemTest | Diagnostics/Information | 4DSP Internal Programming Tool | Registers/Update | FPGA B Memtest | StellarIP | Star Editor | Wormholes Editor | FW
Test Settings
DMA Transfer Speed
Run Memory Test
XILINX VIRTEX-6
V1.24

Active Device: FM680 - 0
Synthesizer Freq. [MHz]: 125
Number of Loops: 2
☑ Data Verification Mode
Read Bandwidth [MB/s]: N/A
Write Bandwidth [MB/s]: N/A
Average Bandwidth [MB/s]: N/A

```
INFO     : Testing @ 125[MHz]
INFO     : Testing in verification mode
STATUS   : Test Nbr  |   BANK1    |   BANK2    |   BANK3    |   BANK4    |
STATUS   : -------------------------------------------------------------
STATUS   :    01     |   pass     |   pass     |   pass     |   pass     |
STATUS   :    02     |   pass     |   pass     |   pass     |   pass     |
STATUS   : Test passed
```

Ready...

Figure 12: DDR2 memory test plug-in view

Test Settings:

This area allows configuration of the operating frequency and the number of loops for the test. Typically the reference firmware allows the memory test to run with a base frequency of 130 MHz.

DMA Transfer Speed:

This area displays the direct memory access speed for the last test. Read Bandwidth (from hardware to host), Write Bandwidth (from host to hardware), and Average Bandwidth are displayed. The "Data Verification Mode" box must be unchecked in order to obtain DMA transfer speed information.

***Note 1**: DDR3 memory banks operate at **twice** the synthesizer frequency. If you set the clock synthesizer to 200 MHz in the DDR3 memory test plug-in, that causes the DDR3 to be clocked at 400 MHz.*

Note 2*: Devices communicating over Ethernet, such as the ML605, are* ***not*** *supported by this plug-in!*

Note 3*: DDR3 memory banks run at a fixed 250MHz clock on FM780 and VP780 regardless of the synthesizer frequency.*

4.9.4 4FM Firmware Installer Plug-in

This plug-in does not require any hardware be attached to the computer. The firmware packages are shipped as an encrypted firmware file. This plug-in allows the retrieval of firmware sources from one of the available firmware packs.

It is normally not required to decrypt the firmware manually unless you have received a dedicated firmware package from 4DSP.

The installer automatically decrypts and extracts the firmware you are licensed to use during the installation process. Please refer to Chapter 4.5 4FM Firmware for more information about this.

This plug-in comes with a firmware package containing all the general firmware included in the standard board support package. The firmware provides reference designs that allow access to the different peripherals attached to the FPGAs on the PCB.

The database also includes the recovery files which can be uploaded to the hardware. A demo license grants access to the recovery files, but not the firmware sources.

4.9.5 4FM FMC Analyzer Plug-in and Application

The 4FM FMC Analyzer Plug-in is a plugin for the 4FM GUI application. This plug-in launches the 4FM FMC Analyzer Application. The 4FM FMC Analyzer allows for real-time acquisition and the display of ADC data on a select number of 4DSP's FMC cards.

The 4FM FMC Analyzer software is written using existing reference applications that are available to customers with a small modification for the initialization and real-time data acquisition. It demonstrates the capabilities of 4DSP's hardware and firmware reference design as it applies to real-time data acquisition.

4.9.5.1 Installation

The 4FM FMC Analyzer is installed during the 4FM driver installation if the necessary license(s) are detected.

4.9.5.2 Hardware and Firmware Requirements

The 4FM FMC Analyzer uses a software library for FFT power spectrum calculation of 10 acquisitions per second with FFT size up to 8192 samples. Because it is calculating FFT power spectrum in either floating-point or double-precision while providing a graphical display of the data, the minimum system requirement is a Windows 7 PC with multicore (i3 and above), at least 4GB of RAM, and a graphics card with current driver.

The software requires the user to have access to the 4DSP hardware, the 4DSP firmware, and its appropriate license for the particular FMC card and the accompanying FPGA carrier card.

The software currently supports the following FMC cards on specific carrier cards (with firmware constellation ID):

FMC Type	ML605 (Ethernet)	VC707 (Ethernet)	ML605 (PCI)	KC705	VP680	VP780	PC720
FMC150	Yes - 88		Yes - 367	Yes - 156	Yes - 150		Yes – 366
FMC116	Yes - 198			Yes - 225	Yes - 239		Yes - 311
FMC30RF	Yes - 227			Yes - 227			
FMC110	Yes - 078		Yes – 386	Yes - 250	Yes - 148		Yes - 307
FMC160		Yes -417					
FMC164/168	Yes - 363			Yes - 377 and 361	Yes - 364		Yes - 387
FMC176	Yes - 347			Yes -275	Yes-375	Yes-294	Yes – 308
FMC108	Yes – 88			Yes - 260	Yes-151		
FMC126	Yes -89			Yes - 253	Yes - 149	Yes -353	Yes - 309
FM204	Yes -95			Yes - 251	Yes-153		

4.9.5.3 Getting Started with the 4DSP Analyzer

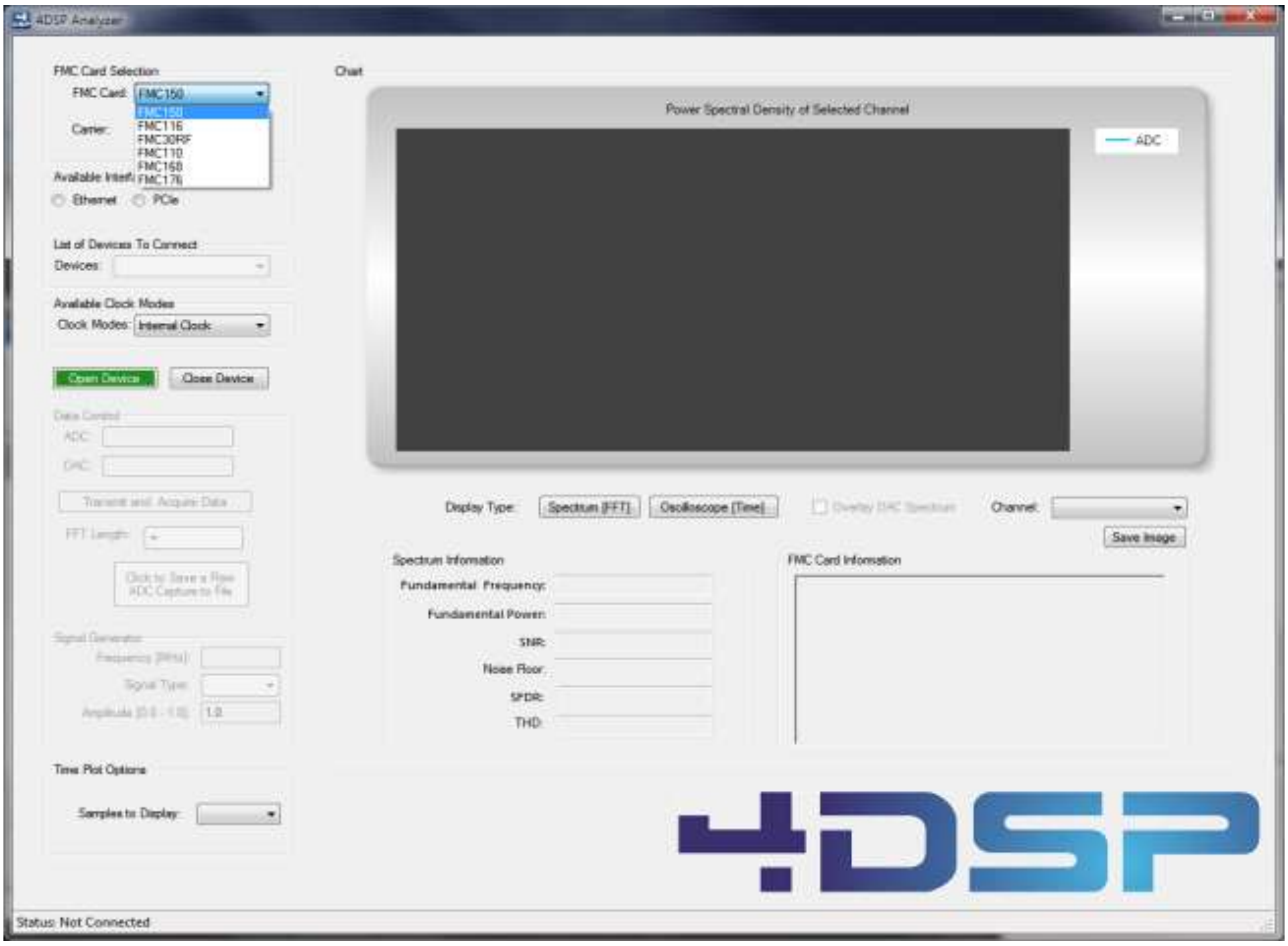
4.9.5.3.1 Configuring the Hardware

In this example, the ML605 with a FMC150 is used.

Connect the ML605 to the FMC150 on the LPC side. Connect the Ethernet cable from the PC to the ML605 and the USB cable to the debug port. Load the firmware bitstream with a constellation ID of 88 for the ML605 with FMC150.

4.9.5.3.2 4DSP Analyzer Software

The first step is to start the 4DSP Analyzer. The following display should appear on the screen:



FMC Card: FMC150

Clock Modes: Internal Clock

Amplitude (0.0 - 1.0): 1.0

Status: Not Connected

Figure 13: Initial 4DSP Analyzer Screen

4.9.5.3.3 Card Selection

You will notice that most of the screen is grayed out after the analyzer is opened. The software will dynamically show the available options depending on the user selections. The first step is to choose the appropriate card:

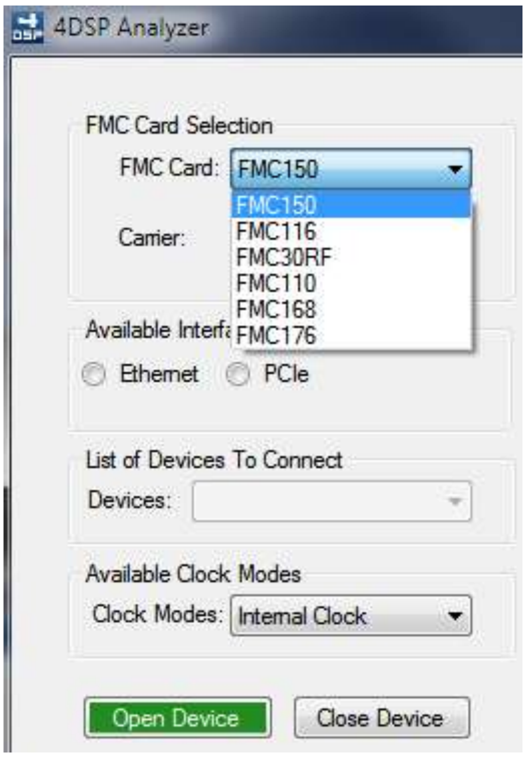


Figure 14: FMC Card Selection

4.9.5.3.4 Interface and Device Selection

The software shows the available interface options for the card selected. For the FMC150 on the ML605, we support both the Ethernet (constellation ID 88) and PCIe (constellation ID 367), so two radio buttons are shown:

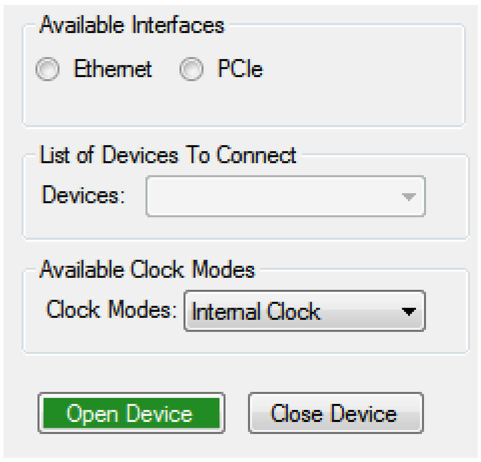


Figure 15: Interfaces

Select the Ethernet interface:

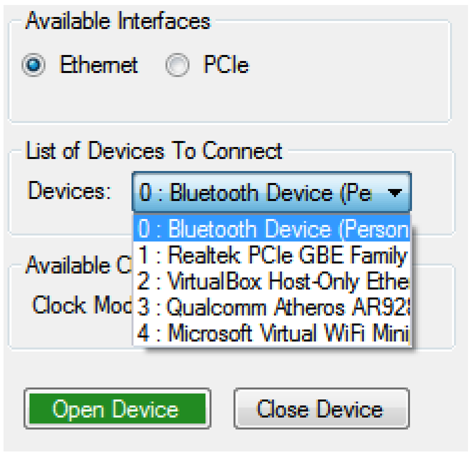

Available Interfaces
Ethernet
PCIe
List of Devices To Connect
Devices: 0 : Bluetooth Device (Pe
0 : Bluetooth Device (Person
1 : Realtek PCIe GBE Family
2 : VirtualBox Host-Only Ethe
3 : Qualcomm Atheros AR92
4 : Microsoft Virtual WiFi Mini
Available C
Clock Mod
Open Device
Close Device

Figure 16: Available Ethernet Devices

The available Ethernet devices are retrieved using the reference software function call ***(sipif_get_deviceenumeration***). This calls the 4DSP driver to get a list of available Ethernet devices. This list is similar to the one generated when running ***ipconfig*** from the command prompt. The "List of Devices To Connect" shown in Figure 16 depends on what devices are connected on your computer.

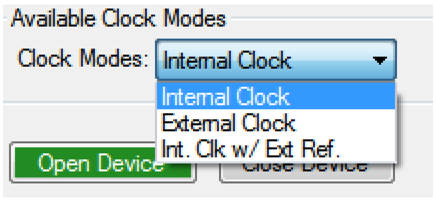


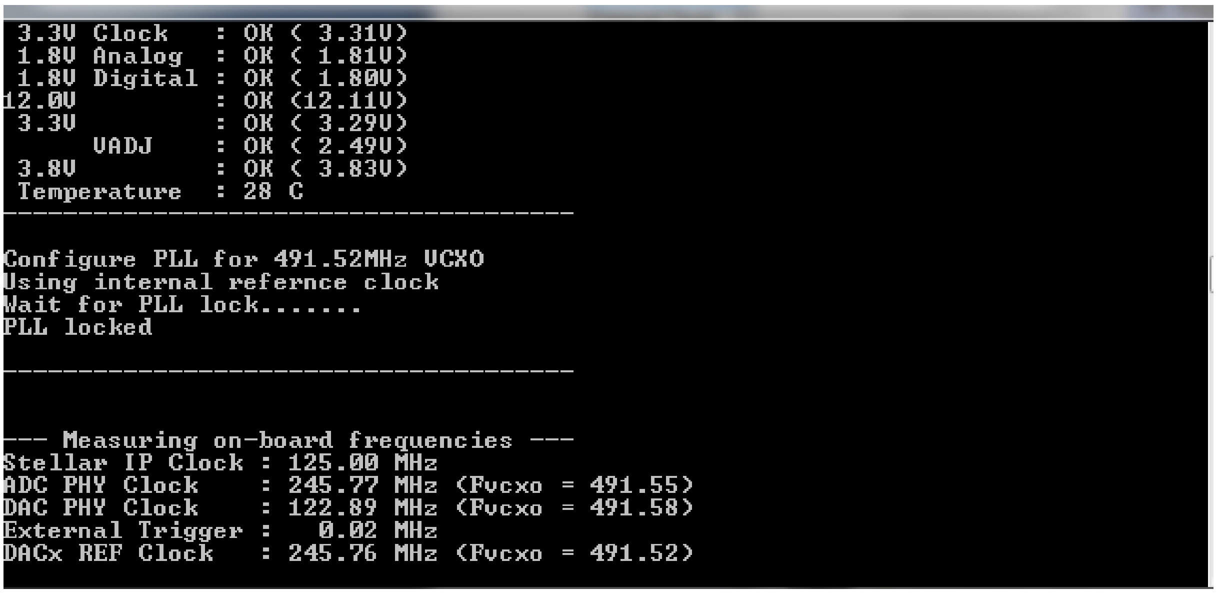
Figure 17: Available Clock Modes

After selecting the card, the "Clock Modes" selection box is available to choose a clock mode for the FMC card. This usually includes both *Internal* and *External* clock modes. Select the appropriate clock mode to operate.

4.9.5.3.5 Connect/Open the Device

The "Open Device" button is used to initialize the FPGA with the appropriate settings and to set up the various PLL, Clock settings, etc. This software is the same as the reference software provided with the 4FM BSP.

Clicking the "Open Device" button will open a console screen.



```
 3.3V Clock    : OK ( 3.31V)
 1.8V Analog   : OK ( 1.81V)
 1.8V Digital  : OK ( 1.80V)
12.0V          : OK (12.11V)
 3.3V          : OK ( 3.29V)
      VADJ     : OK ( 2.49V)
 3.8V          : OK ( 3.83V)
 Temperature   : 28 C
--------------------------------------

Configure PLL for 491.52MHz VCXO
Using internal refernce clock
Wait for PLL lock.......
PLL locked

--------------------------------------


--- Measuring on-board frequencies ---
Stellar IP Clock : 125.00 MHz
ADC PHY Clock    : 245.77 MHz (Fvcxo = 491.55)
DAC PHY Clock    : 122.89 MHz (Fvcxo = 491.58)
External Trigger :   0.02 MHz
DACx REF Clock   : 245.76 MHz (Fvcxo = 491.52)
```

The GUI now shows the various ADC and DAC options if the board is enabled for one or the other or both. For the FMC150, the options include both ADC and DAC. The GUI will also display the number of available channels. For the FMC150, the number of channels is two each.

If the opening and initialization of the device fails for any reason, the GUI will return the appropriate error. Possible error conditions can include:

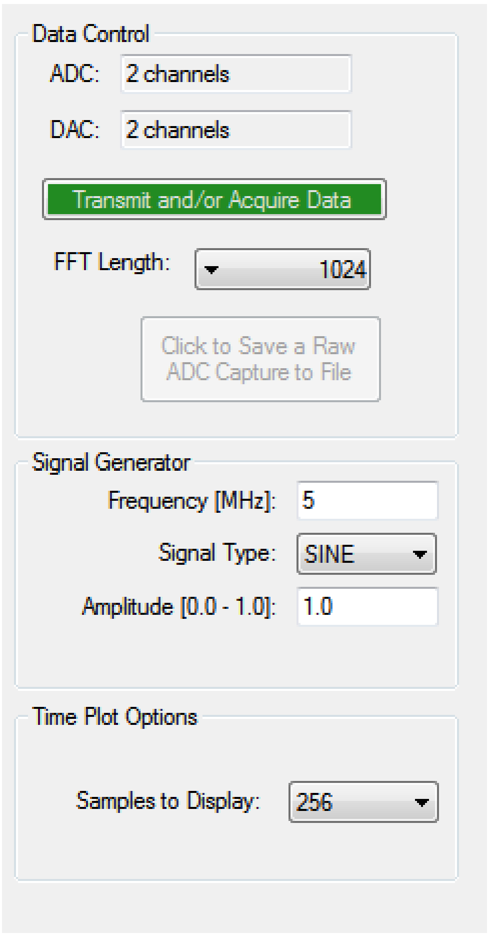
- Using the wrong device index or device type (PCI versus Ethernet)
- Wrong firmware is loaded on the card
- Mounting the FMC card onto the wrong slot (such as mounting FMC150 on the HPC rather than the LPC)

The console will also show where the failure occurs –if the wrong firmware with incorrect constellation ID is used, the console will typically display that as well.

4.9.5.4 Graphical Interface

4.9.5.4.1 Data Control

After the device is successfully opened, the Data Control fields display the number of available ADC and DAC channels. Click the **Green** button to transmit or acquire the data. FFT Length can be selected from the dropdown list.



Data Control
ADC: 2 channels
DAC: 2 channels
Transmit and/or Acquire Data
FFT Length: 1024
Click to Save a Raw ADC Capture to File

Signal Generator
Frequency [MHz]: 5
Signal Type: SINE
Amplitude [0.0 - 1.0]: 1.0

Time Plot Options
Samples to Display: 256

Figure 18: Data Control, Signal Generator, and Time Plot Options

4.9.5.4.2 Signal Generator

The software offers the following type of signal generation for boards with DAC control:

- SINE wave – The software will attempt to generate a sine wave signal near the requested frequency. The size of the burst is dependent upon the FMC card being used. To generate a perfect sine wave cycle, the frequencies will be rounded to the nearest multiple of the digital waveform that is available with a given FMC. For example, the burst size on the DAC of the FMC150 is 16k samples. The start and end of the sample must therefore be aligned or the resulting spectrum will contain harmonics and distortion because the reference firmware repeats the same burst. The signal is generated to -1 and +1 in floating-point/double-precision and translated to the appropriate alignment for the reference firmware.

- SQUARE wave – Similar to sine wave, the software will generate a square wave signal that is near the frequency selected by the user. The restriction is tighter because a SQUARE wave requires that the low and high signals have the same number of samples in order to have a consistent 50% duty cycle (i.e. 1 for low and 1 for high, or 2 for low and 2 for high).

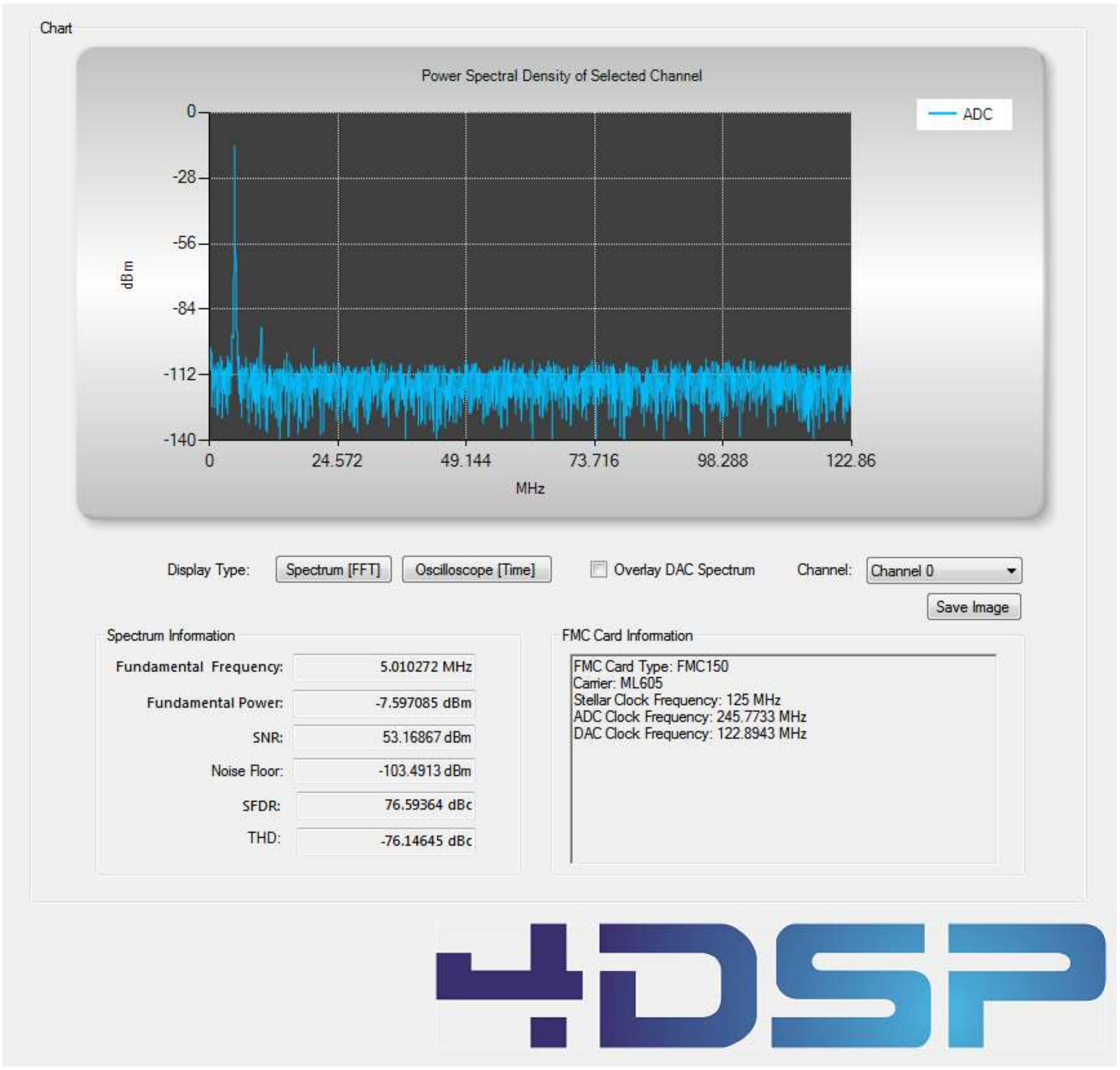
- SAW wave – The “sawtooth” waveform is generated in a way similar to the square waveform. Each period must be consistently the same size in order for the data to align in a burst.

The Amplitude selection allows the user to adjust the output waveform by a factor. If it is 1.0, a sine wave will be generated to have -1.0 to 1.0 equivalent. This is then translated to sine “short” array (16-bit signed array with a certain alignment and precision) and piped through the interface to the FPGA.

We will now connect the DAC outputs to the ADC inputs to demonstrate the software capability of the 4DSP Analyzer. Alternatively, the user can connect the DAC outputs and ADC inputs to a spectrum analyzer or signal generator of their choice.

4.9.5.4.3 Graphical Display

Select Frequency of 5 MHZ, Signal Type of SINE, and Amplitude 1.0. Select FFT Length of 8192 and click on the green **Transmit** button. The GUI now shows a real-time spectrum display:



Fundamental Frequency:	5.010272 MHz
Fundamental Power:	-7.597085 dBm
SNR:	53.16867 dBm
Noise Floor:	-103.4913 dBm
SFDR:	76.59364 dBc
THD:	-76.14645 dBc

FMC Card Type: FMC150
Carrier: ML605
Stellar Clock Frequency: 125 MHz
ADC Clock Frequency: 245.7733 MHz
DAC Clock Frequency: 122.8943 MHz

Figure 19: Spectrum Analyzer Display

Note: The signal generator is now locked so that the user can't modify it. The Frequency is changed to 5.0031 MHz because this is the nearest signal frequency we can generate while preserving an integer number of periods.

The graphical interface shows dynamic FFT spectrum plot with real-time spectrum information, including fundamental frequency, fundamental power (normalized to 0 dB), SNR, and other features. The following features are available:

- Spectrum (FFT)
- Oscilloscope (Time): Displays the last N samples based on the Time Plot Options. The software will attempt to sync the phase to minimize jitter.

- Overlay DAC Spectrum: This option overlays the DAC spectrum if the DAC is available.
- Channel: Select the appropriate channel to sample.
- Save Image: This button allows the user to save the Chart display into a JPEG file.

On the *Data Control* side, there is an option to save the RAW ADC Capture to a file. This is similar to the reference software function that saves a single burst of raw 16-bit ADC data into a text file.

The FMC Card Information box shows the device type and the relevant sampling frequencies of each card. This is important because for cards such as the FMC150, the DAC and ADC clock frequencies are not the same. Please be sure to refer to the hardware documentation on the card itself. Each card also has a potentially different range of voltages to sample. On the FMC150, the DAC outputs a 1 Vpp signal, but the ADC sampling range is 2Vpp. The effect can be seen in the time domain plot:

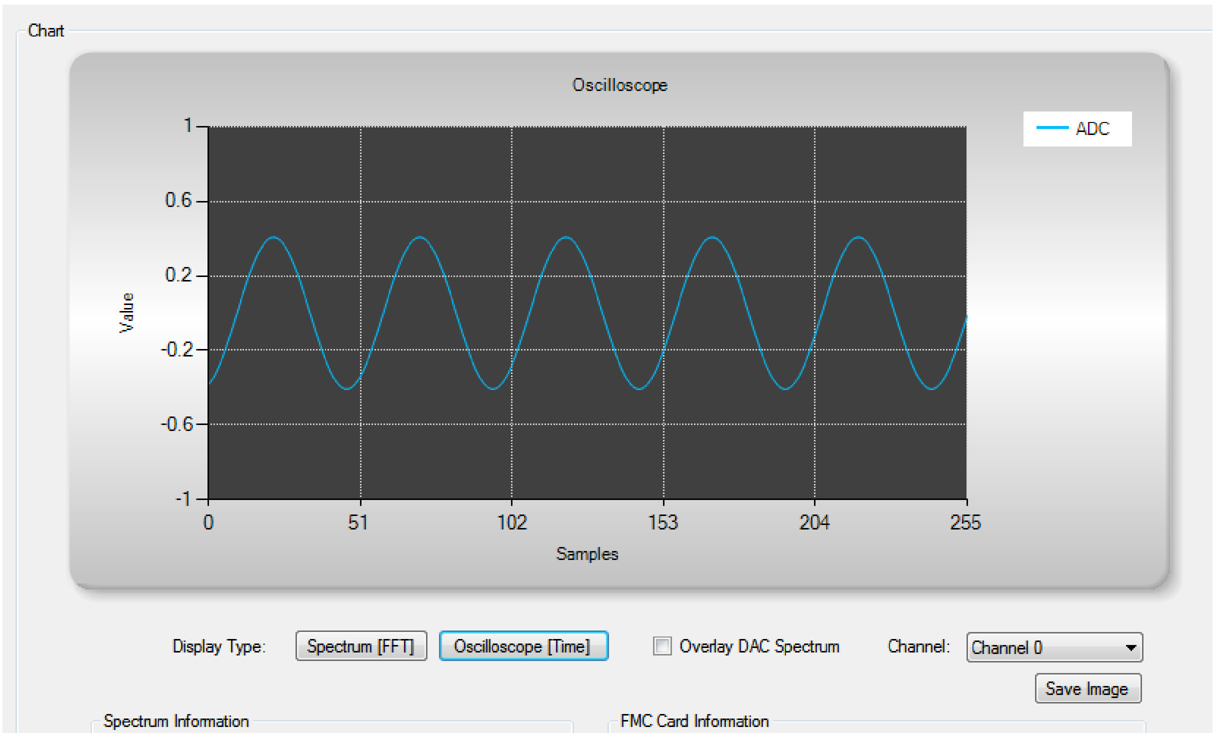


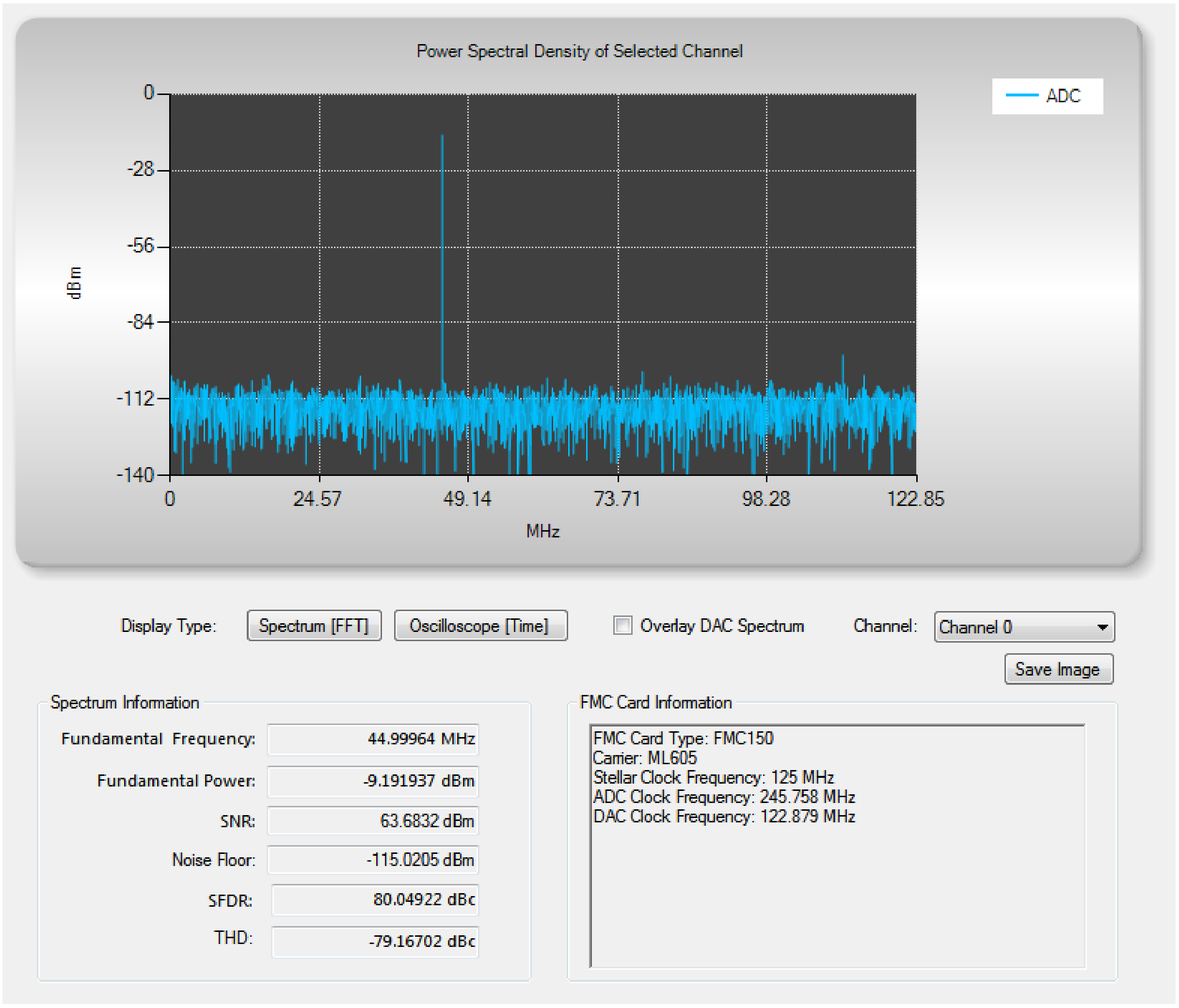
Figure 20: Time Domain Plot

The card is set up to generate -1 to +1 waveform at full amplitude. It is using a loopback cable to route the DAC signal to the ADC signal. Because the FMC150 ADC is 2 Vpp and the DAC is 1 Vpp, we only get half of the waveform range on the ADC. This effectively reduced the power of the signal as indicated in the FFT spectrum plot seen above in Figure 20.

4.9.5.4.4 Advanced Options

One of the more advanced options is the ability to overlay DAC spectrum. This is particularly useful if the user is doing a loopback test. The display shows the spectrum of the actual burst, adjusts it to the spectrum analyzer display on the ADC side, and matches the frequency side by side.

Figure 21 shows an example of the FFT Spectrum with a 45 MHZ signal sent and looped back on the FMC150:

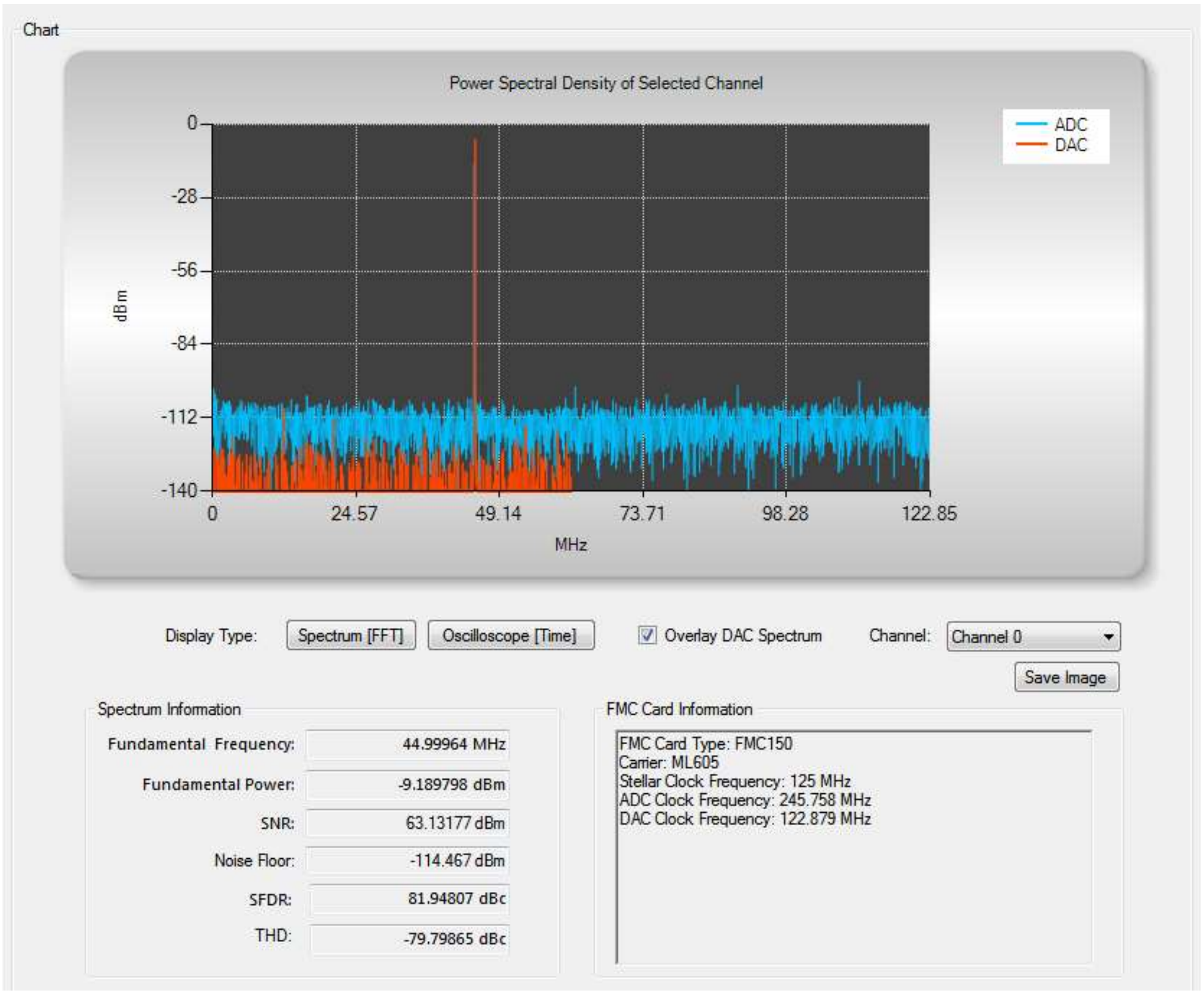


Fundamental Frequency:	44.99964 MHz
Fundamental Power:	-9.191937 dBm
SNR:	63.6832 dBm
Noise Floor:	-115.0205 dBm
SFDR:	80.04922 dBc
THD:	-79.16702 dBc

FMC Card Type: FMC150
Carrier: ML605
Stellar Clock Frequency: 125 MHz
ADC Clock Frequency: 245.758 MHz
DAC Clock Frequency: 122.879 MHz

Figure 21: 45 MHz Sinewave Spectrum

Figure 22 shows the display with the *Overlay DAC Spectrum* option checked:



☑ Overlay DAC Spectrum

Channel: Channel 0

Fundamental Frequency:	44.99964 MHz
Fundamental Power:	-9.189798 dBm
SNR:	63.13177 dBm
Noise Floor:	-114.467 dBm
SFDR:	81.94807 dBc
THD:	-79.79865 dBc

FMC Card Type: FMC150
Carrier: ML605
Stellar Clock Frequency: 125 MHz
ADC Clock Frequency: 245.758 MHz
DAC Clock Frequency: 122.879 MHz

Figure 22: Overlay DAC Spectrum

The series in *orange* is the DAC spectrum output. Since the spectrum is as close to a perfect sine wave as it allows, the noise floor will typically be less. Also, the DAC is only sampling at 122.879 MHz, giving a Nyquist frequency of 61.44 MHz, or about half of the ADC spectrum.

4.9.5.4.5 Examples of Different Signals

Saw wave at 3.84 MHz:

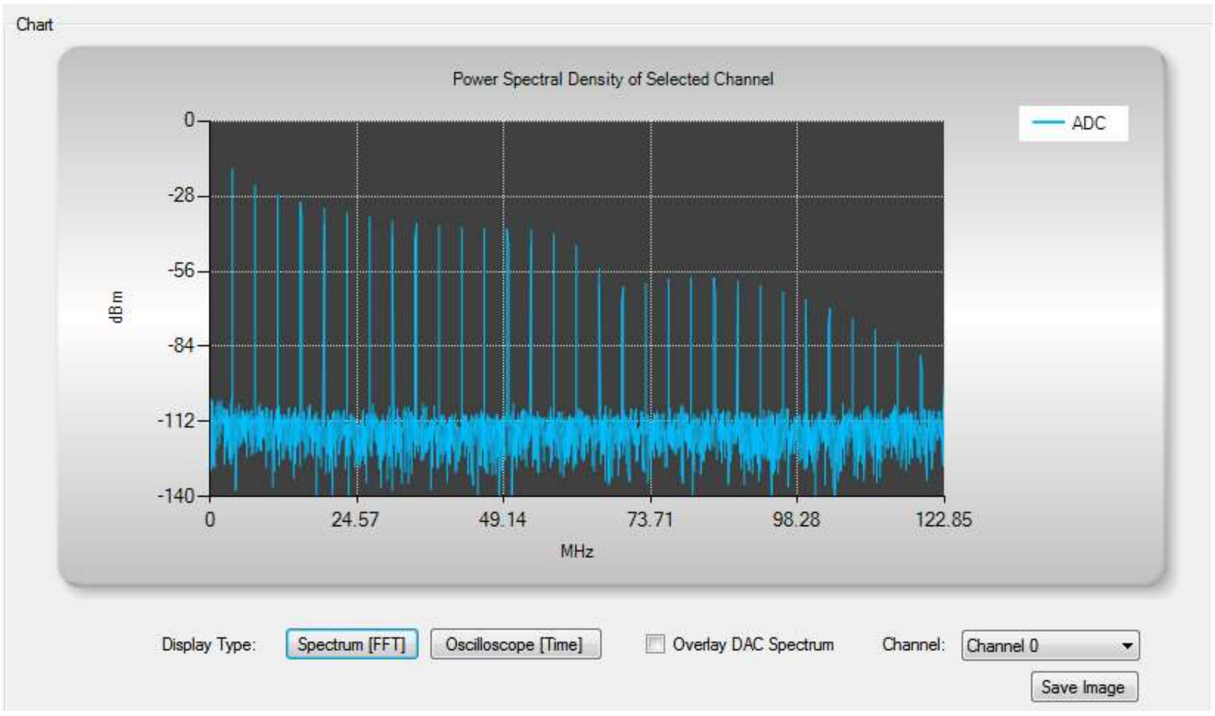


Figure 23: Saw Wave FFT Spectrum

Square wave:

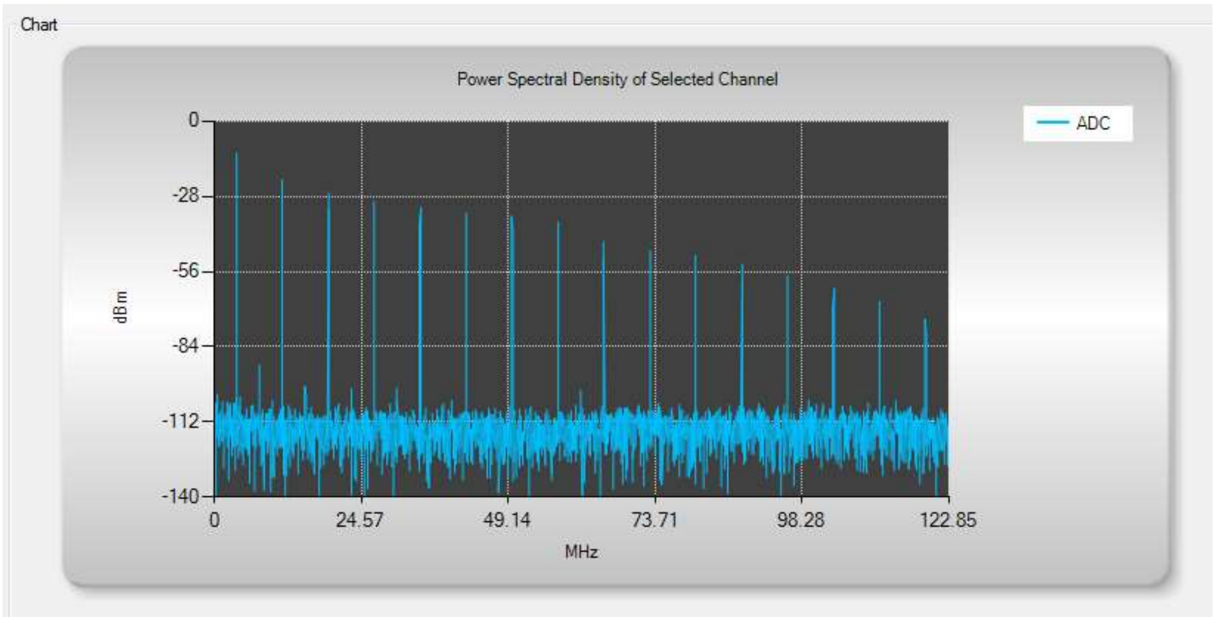


Figure 24: Square Wave FFT Spectrum

4.9.5.4.6 Spectral Plots of Various FFT Sampling Sizes

The following examples show the various sampling sizes in the FFT, using the "Save Image" button to create and save the files.

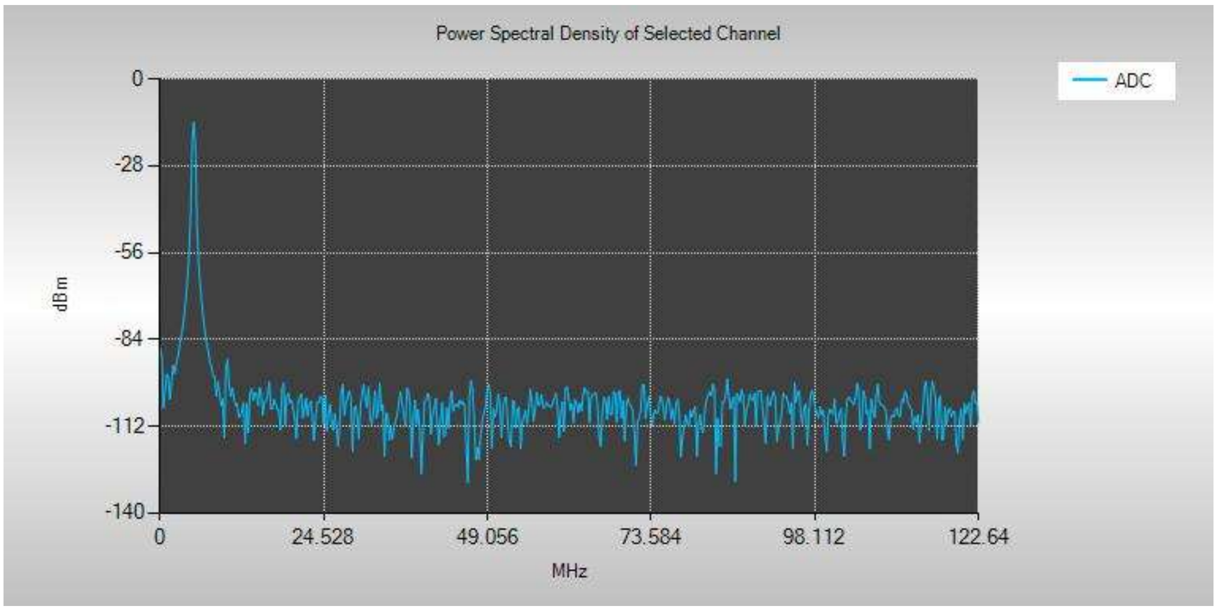


Figure 25: Spectral Plot with 1024 FFT Samples

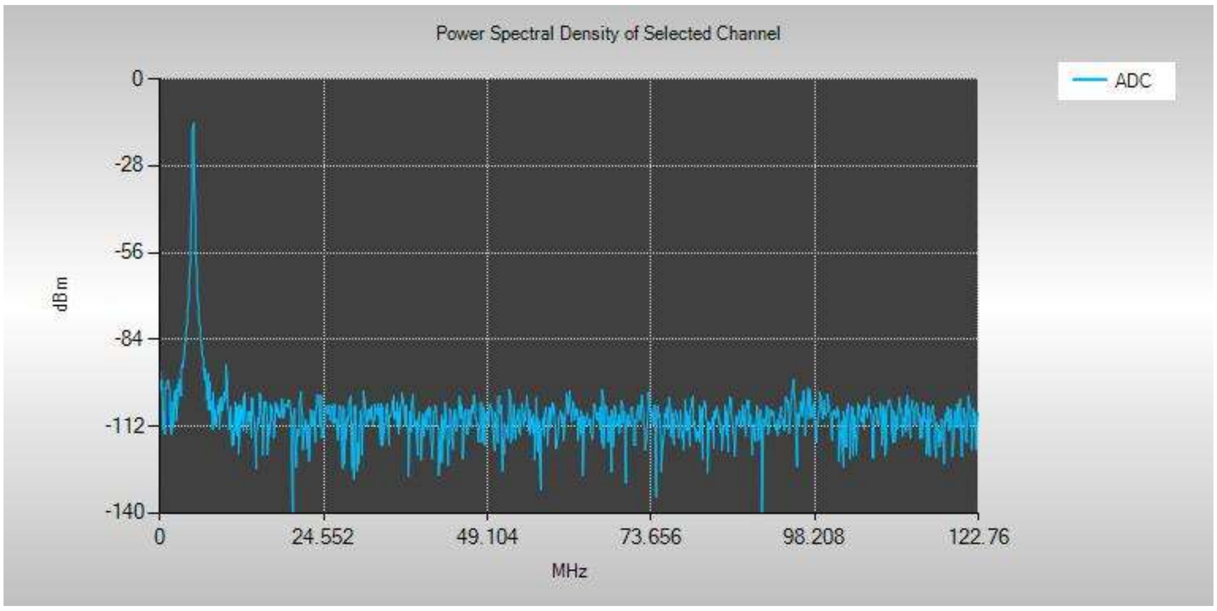


Figure 26: Spectral Plot with 2048 FFT Samples

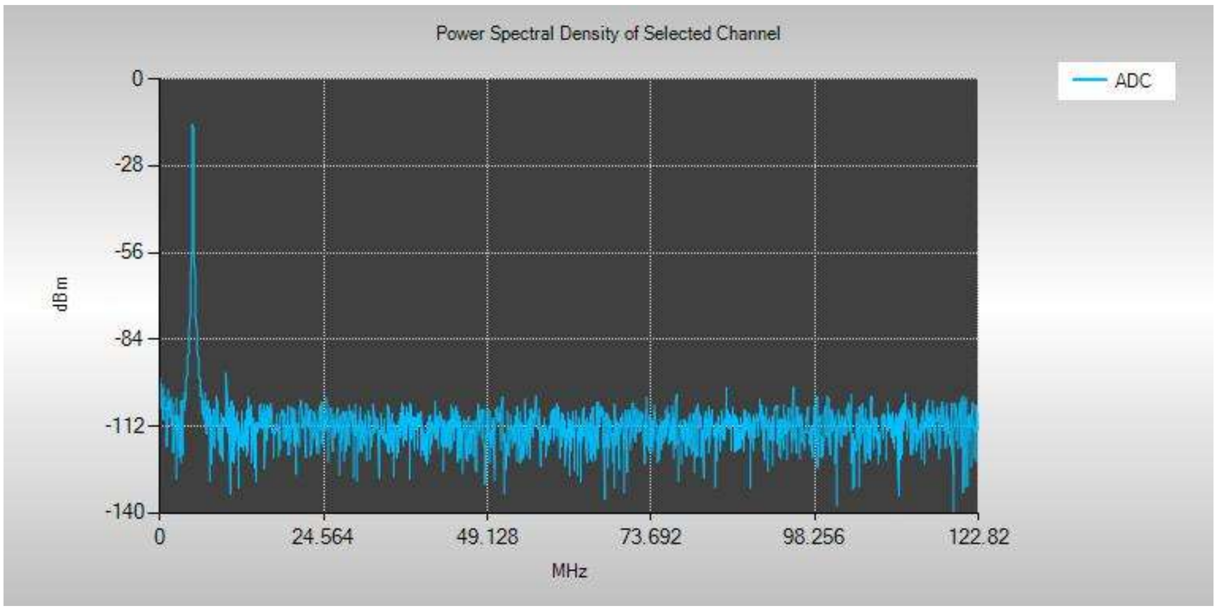


Figure 27: Spectral Plot with 4096 FFT Samples

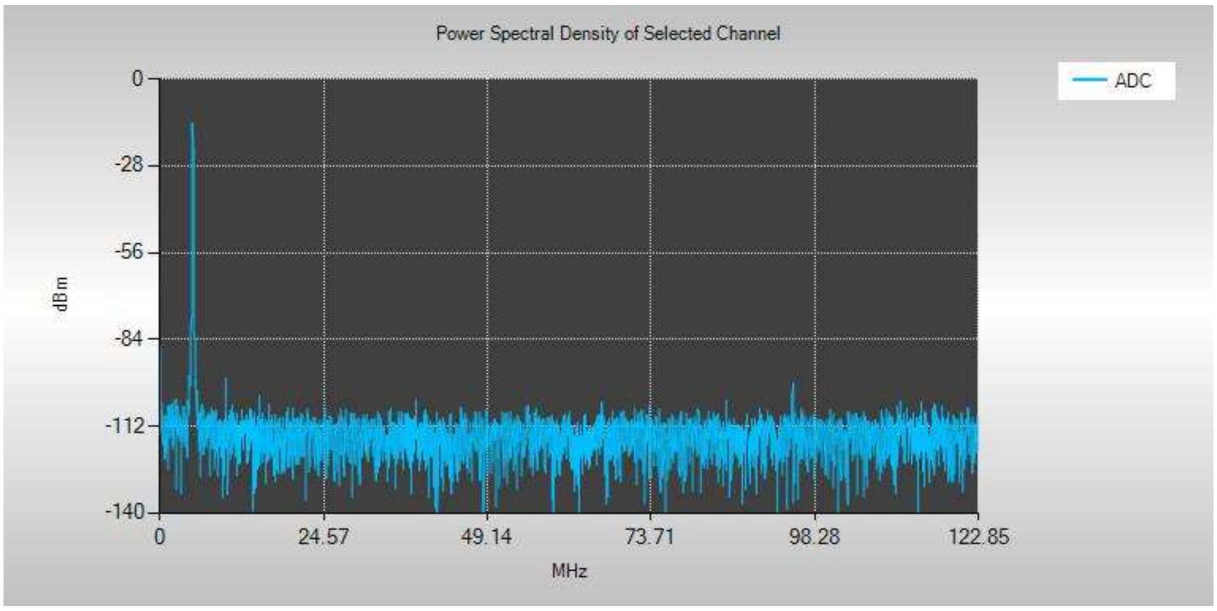


Figure 28: Spectral Plot with 8192 FFT Samples

5 StellarIP

This chapter provides step-by-step instructions on how to use StellarIP. Each section is dedicated to a specific task, and they should be taken in the correct order.

This chapter uses a pre-created design that is installed along with the 4DSP development kit. This design is called Training material and is described in Chapter 10 Training material description.

At the time of writing, only training material for KC705 exists. The design can be modified to support other hardware types. Advanced designs are provided by 4DSP which target the specific hardware you have purchased from 4DSP.

***NOTE**: Each constellation has a document describing the top level design. The description of the individual stars is located in the "doc" subfolder of each individual star. It is also possible to access the star document by right-clicking on a star from StellarIP and choosing "open documentation".*

5.1 Firmware project creation

The purpose of this section is to describe the steps required to create a firmware project that can be compiled using the Xilinx ISE tool.

We will use the "kc705_trainmat" StellarIP project to generate the Xilinx project file and simulate the design using ISim.

The same steps apply to the board specific constellations that are delivered with the BSP. In order to generate a programming file for your specific hardware, you should follow the steps outlined in this chapter but target another .dsn file (.\251_kc705_fmc204\kc705_fmc204.dsn, for example).

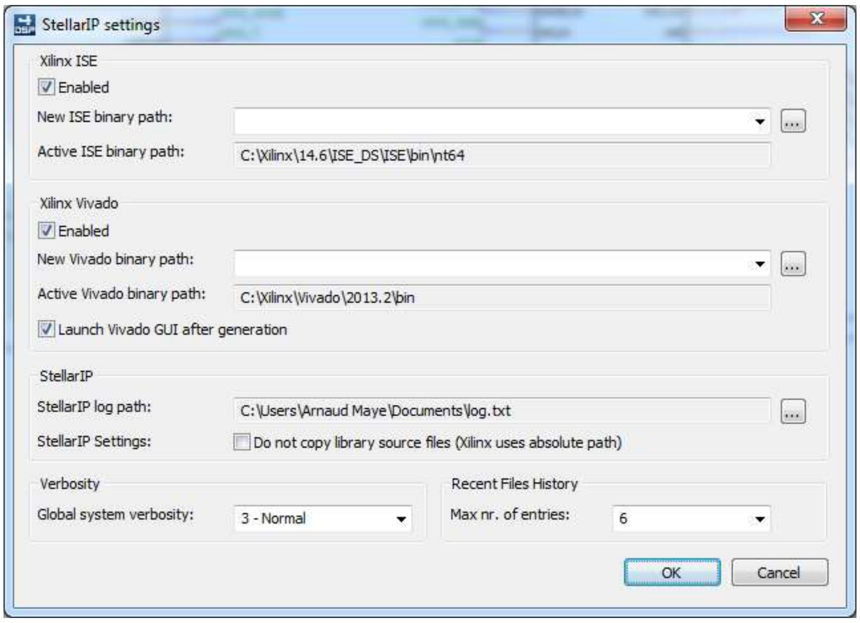
After these steps are completed, the user will have a complete design with vhdl files that can be modified as needed. It is not necessary to use StellarIP after this point if it is not needed.

***Note**: This chapter assumes 'C:\Program Files (x86)\4dsp\Common\Firmware\Extracted\400_kc705_trainmat' and children have been copied to a local folder. This chapter will use **{path}** to represent the actual path where the folder has been copied. For example, if you have copied 'C:\Program Files (x86)\4dsp\Common\Firmware\Extracted\400_kc705_trainmat' to C:\, **{path}** is then C:\. Make sure you have read/write access to the target folder as the tool expects complete read/write/modify access to this folder.*

5.1.1 StellarIP configuration (Path to ISE / Vivado)

StellarIP uses Xilinx tools in order to create project files. The path to these executable/batch files needs to be set in StellarIP.

The “StellarIP settings” dialog can be accessed from the “Stellar IP” menu by choosing “Settings”. Figure 29 shows how the dialog should look when properly configured.


StellarIP settings

Xilinx ISE
- [✓] Enabled
- New ISE binary path:
- Active ISE binary path: C:\Xilinx\14.6\ISE_DS\ISE\bin\nt64

Xilinx Vivado
- [✓] Enabled
- New Vivado binary path:
- Active Vivado binary path: C:\Xilinx\Vivado\2013.2\bin
- [✓] Launch Vivado GUI after generation

StellarIP
- StellarIP log path: C:\Users\Arnaud Maye\Documents\log.txt
- StellarIP Settings: [ ] Do not copy library source files (Xilinx uses absolute path)

Verbosity
- Global system verbosity: 3 - Normal

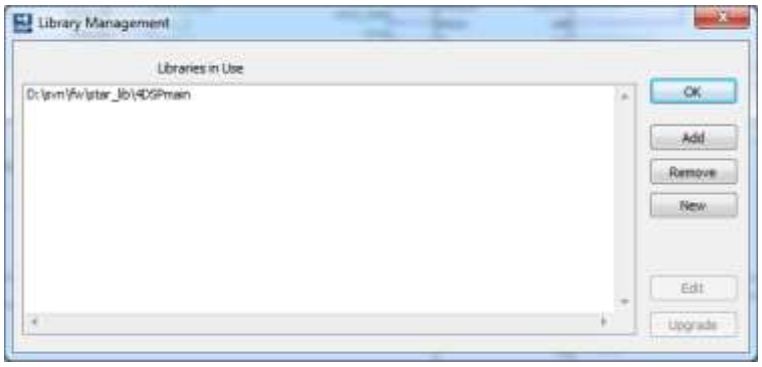
Recent Files History
- Max nr. of entries: 6

Figure 29: Settings dialog

5.1.2 StellarIP configuration (Path to StellarIP Library)

StellarIP uses a library containing the star information. It is required to browse to the library to successfully create an ISE/Vivado project. You can launch the “Library Management” dialog from the “Library” menu. Choose “Manage” to display the dialog.

Press the “Add” button and browse to the actual library file found under ***{path}\star_lib***. This library is called *4DSPmain.TCLib.* You can add the library to the “Libraries in Use” list by pressing the “Open” button.


Library Management

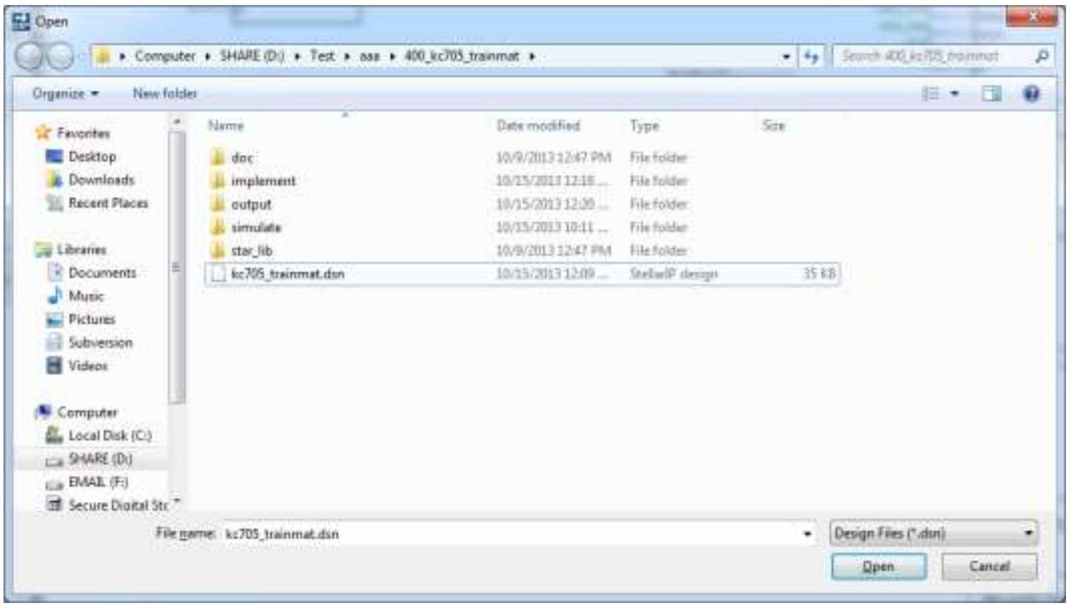
Libraries in Use

D:\svn\fw\star_lib\4DSPmain

Figure 30: Library Management

5.1.3 Open a design schematic

Browse to the design schematic using “Open” from the “File” menu. Browse to the file called “kc705_trainmat.dsn” which is found under ***{path}.***



Name	Date modified	Type	Size
doc	10/9/2013 12:47 PM	File folder	
implement	10/15/2013 12:18 ...	File folder	
output	10/15/2013 12:20 ...	File folder	
simulate	10/15/2013 10:11 ...	File folder	
star_lib	10/9/2013 12:47 PM	File folder	
kc705_trainmat.dsn	10/15/2013 12:09 ...	StellarIP design	35 KB

File name: kc705_trainmat.dsn
Design Files (*.dsn)

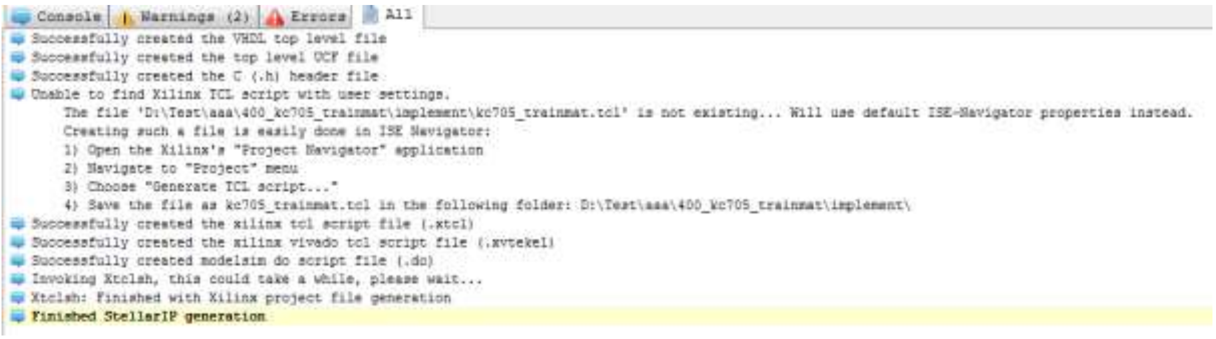
Figure 31: Browsing to the designs schematic

The design schematic will then be displayed in StellarIP.

5.1.4 Generate the ISE/Vivado project

To generate the project, simply press “CTRL-G” on your keyboard or choose “Generate” from the “Stellar IP” menu.

The debug/info control on the graphical user interface will display the status in real time. After the process is completed, the output should match Figure 32:



```
Successfully created the VHDL top level file
Successfully created the top level UCF file
Successfully created the C (.h) header file
Unable to find Xilinx TCL script with user settings.
    The file 'D:\Test\aaa\400_kc705_trainmat\implement\kc705_trainmat.tcl' is not existing... Will use default ISE-Navigator properties instead.
    Creating such a file is easily done in ISE Navigator:
    1) Open the Xilinx's "Project Navigator" application
    2) Navigate to "Project" menu
    3) Choose "Generate TCL script..."
    4) Save the file as kc705_trainmat.tcl in the following folder: D:\Test\aaa\400_kc705_trainmat\implement\
Successfully created the xilinx tcl script file (.xtcl)
Successfully created the xilinx vivado tcl script file (.xvtekel)
Successfully created modelsim do script file (.do)
Invoking Xtclsh, this could take a while, please wait...
Xtclsh: Finished with Xilinx project file generation
Finished StellarIP generation
```

Figure 32: Successful Stellar IP generation

5.1.5 Examine StellarIP results

StellarIP has created a complete Xilinx ISE project that includes a few files worth mentioning.

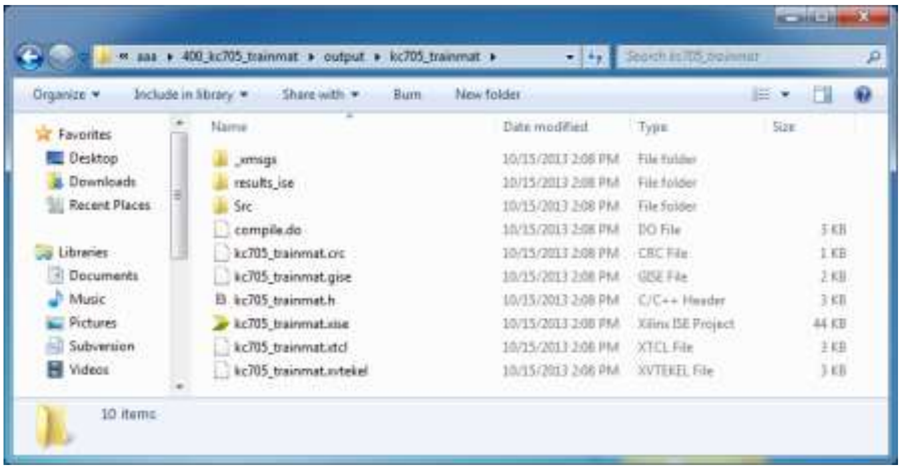
In Chapter 5.1.3 Open a design schematic, we browsed to a path containing a StellarIP Schematic File (.dsn). The StellarIP schematic was located on the root of the firmware folder and the StellarIP output is created in the “Output” folder.

By default, the output folder is found at the following path:

“{***path}****output*\***kc705_trainmat***\”

Available after StellarIP generation (only relevant files are described):

- ***kc705_trainmat.xise*** is Xilinx ISE Navigator files. These files can be loaded into Xilinx Project Navigator like any other Xilinx ISE project.
- ***kc705_trainmat.h*** is a software header with addresses of all the stars of our constellation. In other words, this is the constellation’s address mapping table.
- ***Src*** is a subfolder where all the VHDL sources are located.
- ***Compile.do*** is a file that can be called from modelsim to compile the complete FPGA design into the user work library.



Name	Date modified	Type	Size
_xmsgs	10/15/2013 2:08 PM	File folder	
results_ise	10/15/2013 2:08 PM	File folder	
Src	10/15/2013 2:08 PM	File folder	
compile.do	10/15/2013 2:08 PM	DO File	5 KB
kc705_trainmat.crc	10/15/2013 2:08 PM	CRC File	1 KB
kc705_trainmat.gise	10/15/2013 2:08 PM	GISE File	2 KB
kc705_trainmat.h	10/15/2013 2:08 PM	C/C++ Header	3 KB
kc705_trainmat.xise	10/15/2013 2:08 PM	Xilinx ISE Project	44 KB
kc705_trainmat.xtcl	10/15/2013 2:08 PM	XTCL File	3 KB
kc705_trainmat.xvtekel	10/15/2013 2:06 PM	XVTEKEL File	3 KB

Figure 33: Output folder created by StellarIP

5.2 Open and compile the project in Xilinx ISE

From the start menu, go to “Xilinx ISE Design Suite” > “ISE Design Tools” > “Project Navigator.” Browse to the .xise file located in “*{path}**output****kc705_trainmat****output*”

Xilinx ISE Navigator loads the project and provides a known and proven compilation flow.

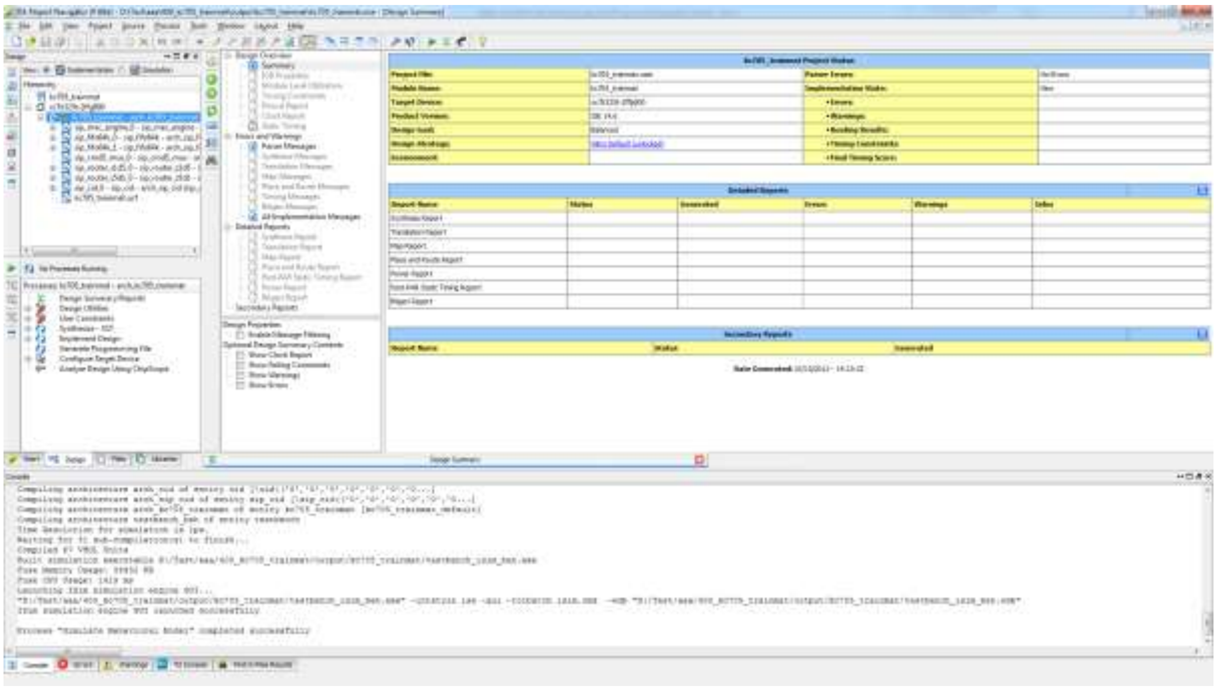


Figure 34: Design successfully compiled using ISE Project Navigator

5.3 Simulating StellarIP results

From the start menu, go to “Xilinx Design Tools” > “ISE Design Suite {version}” > “ISE Design Tools” > “Project Navigator.” Browse to the .xise file located in “***{path}****output****kc705_trainmat****output*”.

Open the Tcl Console (View → Panels → Tcl Console) and run the following command:

- source ../../simulate/isim/isim.tcl

From the Design view, switch from Implementation to Simulation view. Click the top module “testbench” and run “Simulate Behavioral Model”.

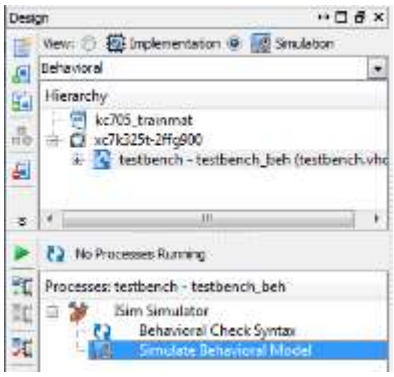

Design
View: Implementation Simulation
Behavioral
Hierarchy
kc705_trainmat
xc7k325t-2ffg900
testbench - testbench_beh (testbench.vhd
No Processes Running
Processes: testbench - testbench_beh
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model

Figure 35: Simulation setup

Please wait for the design compilation to finish and then add some signals to the waveform view. Right-click on the module and select "Add To Wave Window".

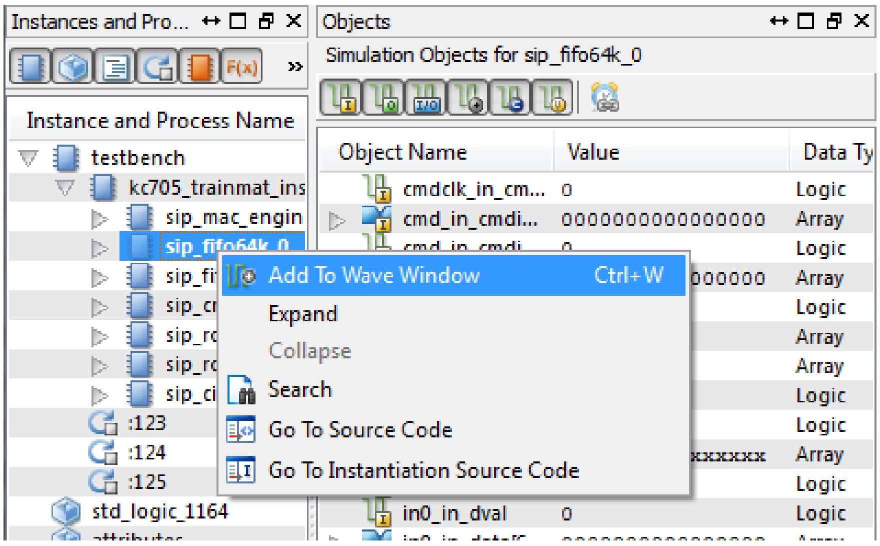


Figure 36: Adding signals to the wave window

You can set the simulation time in the toolbar and click run, or you can simply type "run 200 us" in the Console window. Once simulation is complete, you can again type "run 200 us" to move the simulation up to 400 microseconds. Additional information is available in the ISim console window.

Once the test bench has completed, the file output.txt found in "*{path}**output****kc705_trainmat**\" can be opened. It will contain the data that was received from the design. The simulation transfers 512 bytes so the 512 bytes in output.txt file should be equal to the first 512 bytes in the input.txt file.

Note*: input.txt is larger than output.txt.*

5.4 Modifying simulation behavior

The test bench used for this simulation actually reads a file from the ISim folder. This file is called “sip_cmd.sip” and can be found at the following location:

{path}*simulate**isim*

This file is a script that can be changed to modify the simulation behavior.

The following commands are available in the script:

WAIT = {value}	Wait for the specified amount of time in microseconds (us).
REG_GET = {address}	Read a 32-bit value at a given address.
REG_SET = {address} {value}	Write a 32-bit value at a given address.
DMAPUSH = {size}	Send an amount of bytes using DMA operation. (*Note that the data is actually read from the input.txt file.)*
DMAPULL = {size}	Receive an amount of byte using DMA operation. (*Note that the received data is written to the output.txt file.)*

6 StellarIP Star Creation

A comprehensive set of ready-to-use stars targeting the hardware that was purchased is available to the user. Many of these stars are generic and can be used on 4DSP or third-party hardware. Memory controllers, data routers, command routers, and daughter board interface stars are typically provided to end users as part of the BSP.

In most cases the user will want to add their own processing to the constellation. To do this, custom stars must be created. This chapter explains the steps to create a FIFO star with 128KB storage space.

6.1 Updating existing sip_fifo64k star to a sip_fifo128k star (VHDL/Netlist)

1) Copy the contents of folder sip_fifo64k from your star_lib into a newly created folder sip_fifo128k in your star_lib folder.
2) With the Xilinx tool Coregenerator open the file \sip_fifo128k\vhdl\xilinx \corgen.cgp. Open the *fifo64k_fifo_64x8K* IP core and change the target name to *fifo128k_fifo_64x16K.* Double the FIFO storage space from 8k to 16k in the Coregen settings. Generate this new FIFO design.
3) Remove the files *fifo64k_fifo_64x8K.ngc*, *fifo64k_fifo_64x8K.xco* and *fifo64k_fifo_64x8K.vhd* from the folder \sip_fifo128k\vhdl\xilinx.
4) Rename the files \sip_fifo128k\vhdl \fifo64k.vhd, \sip_fifo128k\vhdl \fifo64k_regs.vhd and \sip_fifo128k\vhdl fifo64k_stellar_cmd.vhd to fifo128k.vhd, fifo128k_regs.vhd, and fifo128k_stellar_cmd.vhd respectively.
5) Rename the files \sip_fifo128k\ sip_files\sip_fifo64k.lst, \sip_fifo128k\ sip_files\sip_fifo64k.nfo, and \sip_fifo128k\ sip_files\sip_fifo64k.vhd to sip_fifo128k.lst, sip_fifo128k.nfo, and sip_fifo128k.vhd respectively.
6) Modify the contents of sip_fifo128k.nfo by changing the first line to read 00D4 instead of 003D. This is the star identifier number. The other line is the version and it can remain as 1.0.
7) Modify paths in sip_fifo128k.lst by changing 64k to 128k and 64x8K to 64x16K.
8) Modify the contents of sip_fifo128k.vhd by changing 64k to 128k and 64x8K to 64x16K.

At this stage you have a complete sip_fifo128k folder in your star_lib folder.

6.2 Adding sip_fifo128k star in StellarIP

The easiest way to create a new star is to reuse an existing star and all of its related attributes. This method is called “cloning a star” and is available in StellarIP. This example will modify the name of the original star but not its attributes. It is possible to add, remove, or modify input/output ports from a cloned star if modifications are required.

Click the “Edit” button from the star picker dialog located on the left side of the StellarIP window. This will display the complete library. The names and previews of the symbols are visible.

Scroll down to sip_fifo64k, select this star, and choose “Clone Star” from the right-click menu as seen in Figure 37.

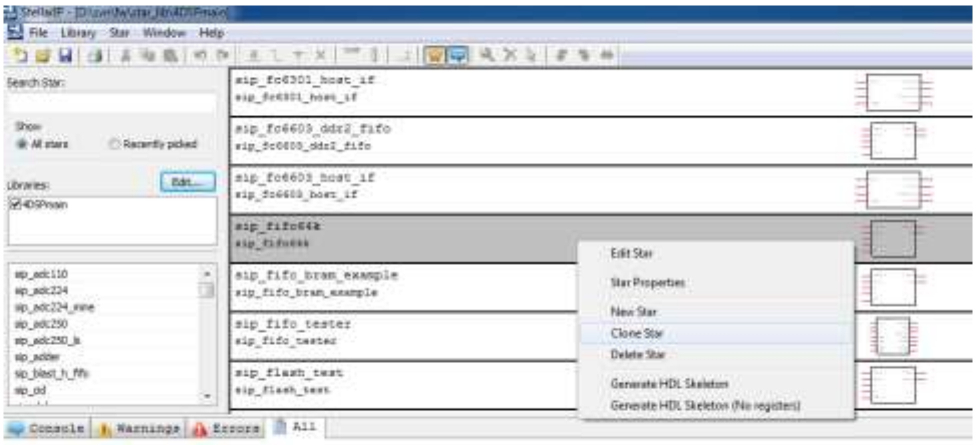


Figure 37: Cloning a star from the library view

A new dialog will be displayed where you can choose a new name for the cloned star. Enter sip_fifo128k into the dialog as seen in Figure 38.

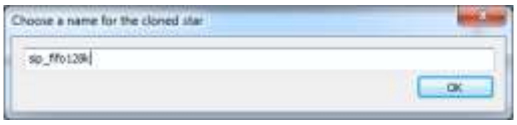
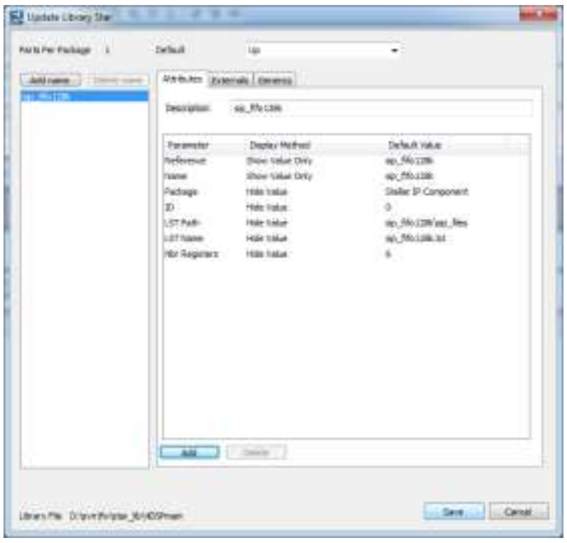


Figure 38: Choosing a new name for the clone

A new dialog will be displayed with a summary of the attributes that were all copied from the parent, as seen in Figure 39. Click the “Save” button.



Parts Per Package 1 Default Up

sip_fifo128k

Description: sip_fifo128k

Parameter	Display Method	Default Value
Reference	Show Value Only	sip_fifo128k
Name	Show Value Only	sip_fifo128k
Package	Hide Value	Stellar IP Component
ID	Hide Value	0
LST Path	Hide Value	sip_fifo128k\sip_files
LST Name	Hide Value	sip_fifo128k.lst
Nbr Registers	Hide Value	6

Library File D:\svn\fw\star_lib\4DSPmain

Figure 39: The star attributes dialog

The newly created star is now part of the library and can be used in the schematic editor.

7 Creating a new schematic

In the Chapter 6 StellarIP Star Creation we created a new star based on an existing star. We will now create a new schematic design based on the current kc705_tainmat by cloning a schematic design to a new name.

7.1 Open the current design

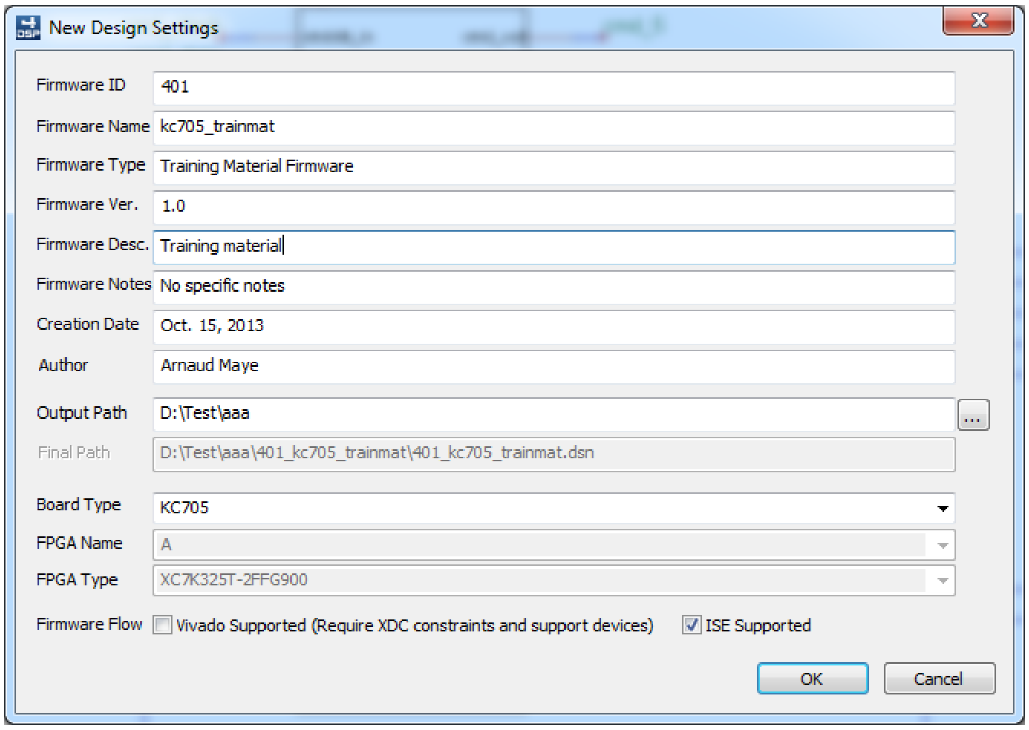
Browse to the design schematic using “Open” from the “File” menu. Select the file called “kc705_trainmat.dsn”. This file is found under ***{path}.***

7.2 Clone the current design

Choose “Clone” from the “File” menu. You can then adjust the actual settings required for the cloned design. Configure the “New Design Settings” dialog as seen in Figure 40 with the exception of the “Output Path” field. Make sure to verify that *your* “Output Path” field defaults to the location where the initial schematic design is located.

Press the “OK” button to create the new design based on the initial design.

Note that the “Author” and “Creation Date” field are automatically populated by StellarIP.


New Design Settings

Firmware ID	401
Firmware Name	kc705_trainmat
Firmware Type	Training Material Firmware
Firmware Ver.	1.0
Firmware Desc.	Training material
Firmware Notes	No specific notes
Creation Date	Oct. 15, 2013
Author	Arnaud Maye
Output Path	D:\Test\aaa
Final Path	D:\Test\aaa\401_kc705_trainmat\401_kc705_trainmat.dsn
Board Type	KC705
FPGA Name	A
FPGA Type	XC7K325T-2FFG900
Firmware Flow	☐ Vivado Supported (Require XDC constraints and support devices) ☑ ISE Supported

Figure 40: Settings for our cloned design

7.3 Modify our clone

Now that our new star has been created and our design is cloned, we can add the new star to the new design.

7.3.1 Changing the cmd_mux star

Every star in the design with registers implemented must be connected to the command mux star. Failing to connect the command bus will prevent the register from communicating with this star. Because there are no more free ports on the cmd_mux_star, we need to replace sip_cmd5_mux star with a sip_cmd10_mux star in the schematic.

1) Delete the sip_cmd5_mux star.
2) Make some room for the sip_cmd10_mux star.
3) Create a net on cmd5_in port and name it cmd_6.
4) Create nets on cmd6_in to cmd9_in and name all these nets "tiedto0". This is required because input wormholes cannot be left unconnected.

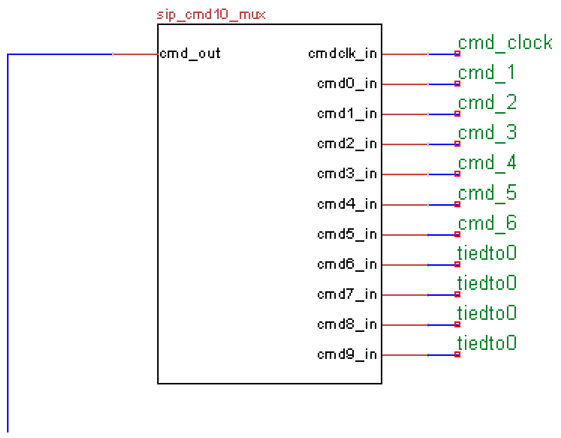


Figure 41: sip_cmd10_mux when properly connected

7.3.2 Adding sip_fifo128k star in the design

From the star picker on the left side of the StellarIP window, scroll down to the sip_fifo128k star and double-click the corresponding line. You can place the new star where you want in your schematic.

Please connect the star as the other sip_fifo64k in the design. In0 and out0 of the new star are connected to the data routers. Make sure everything is connected as shown in Figure 42.

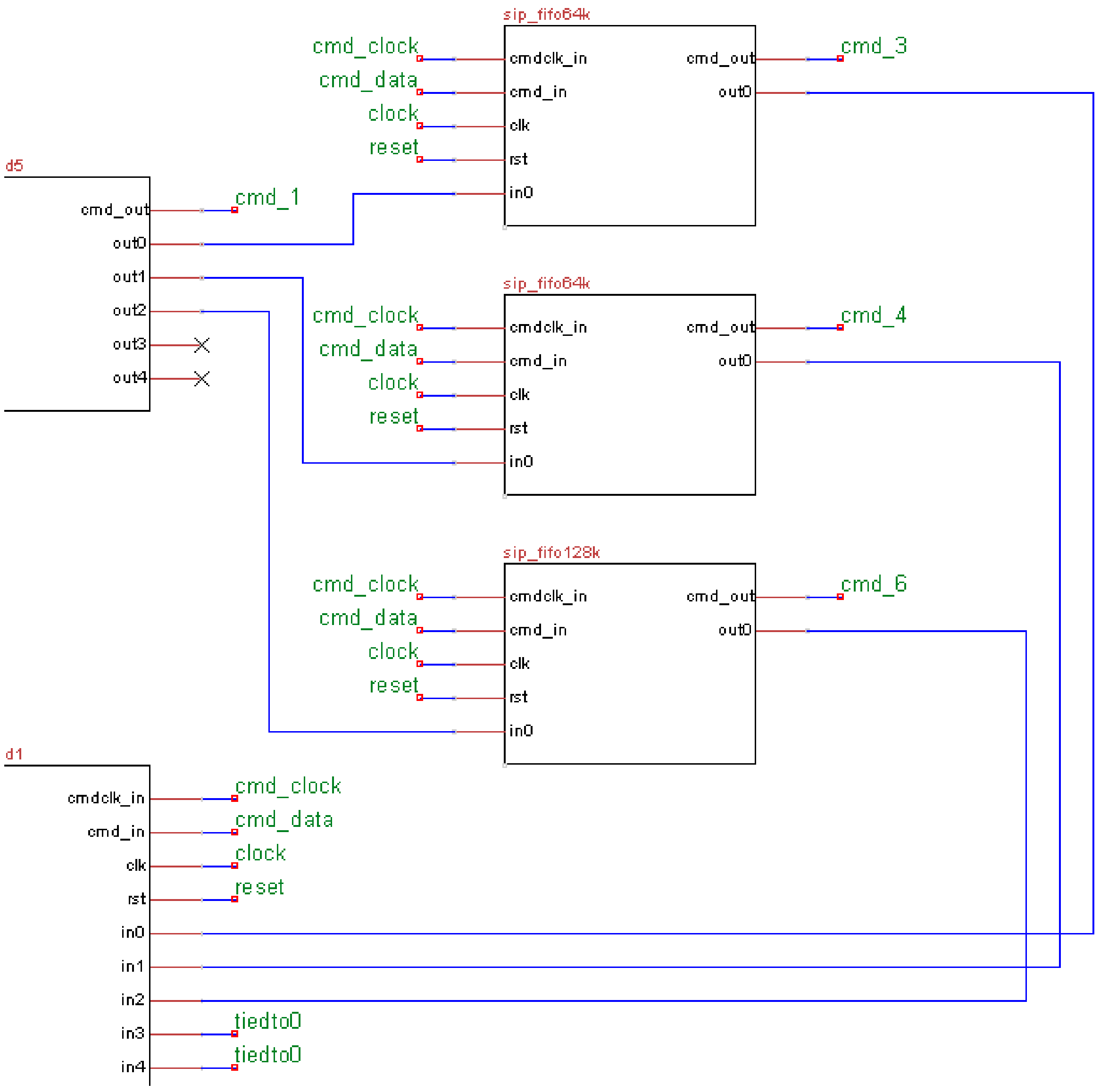


Figure 42: Design connections after adding sip_fifo128k

7.3.3 Generating the new design

To generate the project, simply press “CTRL-G” on your keyboard or choose “Generate” from the “Stellar IP” menu.

7.3.4 Verifying our modifications

A simple way to verify if our star has been properly inserted by StellarIP is to check the project’s header file (.h). This file includes definitions for all the stars including their addresses.

We can see in the constellation header (kc705_trainmat.h) that the “sip_fifo128k” star is part of our design. We can also see that the ID and version match what we have previously placed into the .nfo file. The start address is 0x2410, and the end address is 0x2415.

7.3.5 What’s next?

Steps from Chapter 5.2 Open and compile the project in Xilinx ISE can be followed for the newly created design.

8 Importing StellarIP legacy design and libraries

The design entry was different in earlier StellarIP versions; the libraries were in *xml* format and the design descriptions were ASCII files with an *sdf* extension. The new StellarIP libraries are in *tclib* format and the design description is a schematic design with a *dsn* extension.

This chapter describes the steps required to convert a legacy design into a design fully supported by the new StellarIP versions.

8.1 Remove the current library

Choose “Manage” from the “Library” menu. Remove a library by selecting the library and pressing the “Remove” button. Repeat this procedure as many times as required until there are no more libraries shown in the library management dialog.

8.2 Preparing the legacy firmware

The target firmware should be copied from the place it was decrypted by the installer to a path without space characters such as “C:\fw\”. Write access is mandatory for this operation to succeed. In this example we use 112_ml605_fmc104 firmware and we copy the firmware folder on the D disk. The resulting path is D:\112_ml605_fmc104. Note that any legacy firmware can be upgraded that way and not only firmware provided by 4DSP; any firmware created by our customers can also be upgraded the same way.

8.3 Import the legacy library

Choose “Import Legacy Library” from the “File” menu. Browse for the XML library file found in the firmware. The path we use for this example is D:\112_ml605_fmc104, so the XML is located under D:\112_ml605_fmc104\star_lib\library.xml.

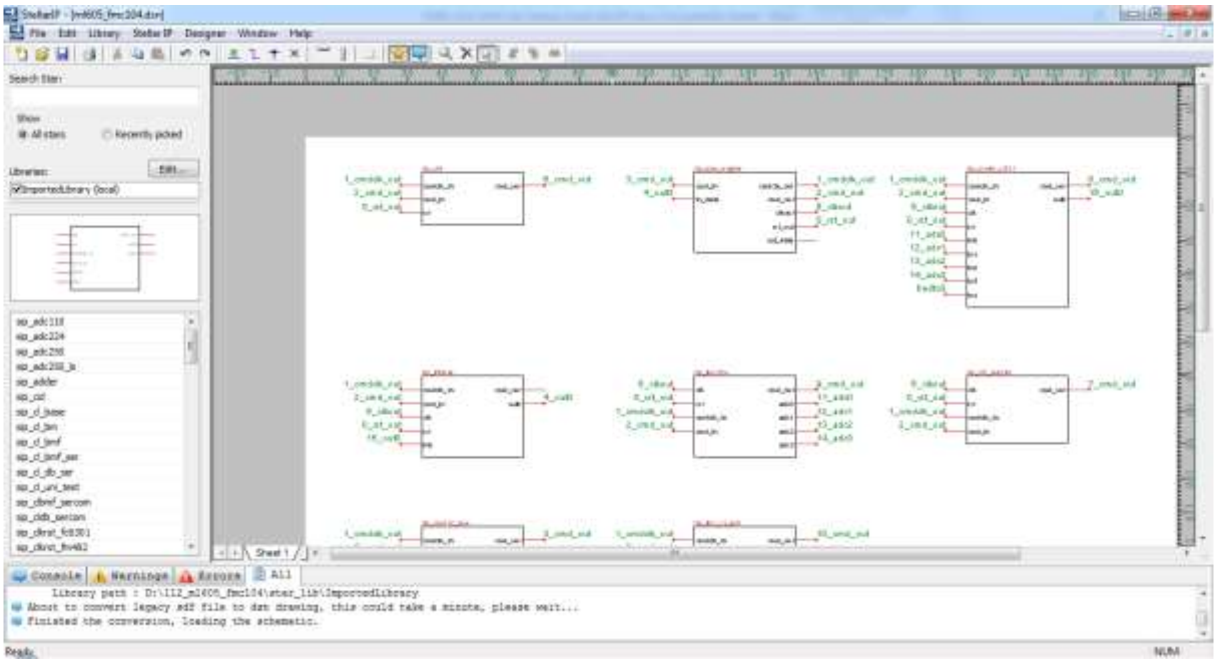
StellarIP will prompt for the library name which defines the way the imported library will be named and appear in the star picker. We simply press “OK” to keep the default “ImportedLibrary” name.

The process can take a couple of minutes. The actual time required depends on the size of the legacy library. StellarIP will indicate completion by writing a “Successfully finished importing” line in the output control. Sometimes unchecking and rechecking the imported library is required to get the library to display properly in the star picker.

8.4 Import the legacy design

Now that the library has been properly imported we can import the legacy design file. Choose “Import Legacy Design” from the “File” menu. Browse for the *sdf* description file found in the firmware. The path we use for this example is D:\112_ml605_fmc104, so the *sdf* is located under D:\112_ml605_fmc104\implement\ml605_fmc104.sdf. Answer “Yes” when prompted by StellarIP about where to place the converted design file.

StellarIP then detects a local library and offers the choice to use either the local library or the currently selected library. Answer “Yes” at this time.



```
Library path : D:\112_ml605_fmc104\star_lib\ImportedLibrary
About to convert legacy sdf file to dsn drawing, this could take a minute, please wait...
Finished the conversion, loading the schematic.
```

Figure 43: StellarIP after a successful import

9 Post compilation steps (PCI/PCIe hardware).

In Chapter 5 StellarIP (Firmware project creation) we generated a project using StellarIP and compiled the project using "Xilinx Project Navigator". In this chapter we will explain how to upload the firmware to the hardware. Two options are available: The first is to directly upload the firmware via the JTAG chain using "Xilinx Impact" and a JTAG programmer. The second method is to upload the firmware into the FLASH memory. Uploading a .bit file using a JTAG is outside the scope of this manual, so only the second option is described below.

9.1.1 Convert the .bit file into a .hex file (bit2hex)

The previously created .bit file must be converted to a programming file (.hex) before it can be uploaded into the flash device. A bit file conversion tool is available and has been installed/referenced by the installer.

Browse to the *C:\Program Files\4dsp\4FM Core Development Kit\Bins* folder. Drag and drop your .bit file onto "bit2hex" icon.

The bit2hex application will generate a programming file using Xilinx iMPACT. This file will be created in the folder where the initial .bit file is located. In the same folder, it will also place a log file generated by iMPACT containing additional information on the previous generation process.

When the bit2hex application is run for the first time, it will not have the path to Xilinx iMPACT in its settings yet. It will first attempt to find the path by itself by looking in the standard Xilinx installation locations. From the valid search results, bit2hex will automatically select the most recent version of Xilinx. In that case, bin2hex will function directly out of the box with no configuration required. If bin2hex cannot find impact.exe in any of the usual Xilinx installation locations, it will ask the user to select it manually from an open-file-dialog. Impact.exe can usually be found in a directory that looks like or is a variation on this: 'C:\Xilinx\14.4\ISE_DS\ISE\bin\nt64\impact.exe'. Usually only the version number differs, and bin2hex detects that. If Xilinx is installed in a completely different directory, bin2hex will not detect it.

You can also manually change the Xilinx path later by double-clicking bit2hex.exe instead of dragging and dropping to it. This method triggers a prompt to select the Xilinx path using the same open-file-dialog mentioned before. This is handy when you want to use an older or newer installation of Xilinx without removing the current one from your system.

If the Xilinx path stored in the bin2hex setting somehow becomes invalid (by registry corruption, the renaming of a related file or folder, or by de-installation), bin2hex will clear its settings and treat the situation in the same way as when it ran for the first time. It will look for the newest valid version by itself and select it silently. If that yields no result, you will be prompted to set the path with the open-file-dialog.

9.1.2 Convert the .bit file into a .hex file (shell extension)

On Windows 7 (32-bit), the SDK includes a kernel extension that can be used to easily convert a .bit file into a .hex file. It is available when right-clicking a .bit file and choosing “Convert to programming file (.hex)”.

The .bit file will be converted to a .hex file, and the .bit file will be deleted. Note that this process can take up to two minutes, and no specific information will be displayed on the screen during this time.

9.1.3 Upload the firmware into hardware

Now that we have a .hex programming file, we will use the 4FM GUI Control application and the “Registers/Update” plug-in.

We want to upload a firmware into FPGA A or FPGA B depending on which FPGA you are targeting. 4DSP carriers such as VP680, VP780, FC6301, and FC6603 have only one FPGA, so the hex file should be programmed in FPGA A. On carriers such as FM482, FM680, and FM780, only FPGA B should be programmed. FPGA A on these boards should not be changed unless explicitly requested by 4DSP.

Browse to the newly created .hex file and press the “Load” button.

After the operation is completed successfully, perform a power cycle of the computer. Start the 4FM GUI control application after rebooting, select the hardware, and switch to the “Diagnostics/Information” plug-in. The FPGA B firmware ID has changed and now reflects step 1 of the training material as expected.

9.2 Test the firmware using StellarIP’s Script Parser (PCI/PCIe hardware)

During our simulation we have looked at and modified the “sip_cmd.sip” file in Chapter 5.4 Modifying simulation behavior.

4DSP provides customers with an application that parses this file and communicates with the hardware through the API and the device driver. It provides a convenient way to quickly inspect firmware behavior without coding a line of software.

This is a command line application referenced by the SDK installer. It can be called from any folder in your system.

The syntax of this application is as follows:

```
“4FM SIP Script Parser.exe” {Device Type} {Device Number} {Script Name}
```

- ***Device Type*** is the type of device you purchased from 4DSP (FM489, FM680, FC6301).
- ***Device Number*** is always ‘0’ (zero) unless you have several 4DSP devices in your system.
- ***Script Name*** is the name of the script to be parsed by the application.

Open a command prompt window and change directory (cd) to the ISim folder:

```
cd {path}\output\{projectname}\simulate\isim
```

Launch the application:

```
"4FM SIP Script Parser.exe" {deviceype} 0 sip_cmd.sip
```

9.3 Test the firmware using training material’s reference application

The training material is accompanied by reference software. This reference software is actually reading/writing registers and receiving/sending data.

They can be loaded and compiled using Microsoft Visual Studio 2012. The reference firmware targets the initial training material and not the one where we added a sip_fifo128k star.

The software project can be found at the following path:
*C:\Program Files\4dsp\4FM Core Development Kit\StellarIP\Training Material\*Software

Pressing CTRL-F5 in Visual Studio will compile and execute the application.

10 Training materials

Training material is provided to the customers as an example with both a firmware and a software project. The firmware is meant to be processed by StellarIP and the software is meant to be compiled by Microsoft Visual Studio.

Please consult the *cd400_kc705_trainmat.pdf* document to understand more about the firmware architecture.

The software flow is as follows:

1) Open the hardware device.
2) Retrieve the constellation toplogy using cid_init().
3) Obtain addresses of all the stars in the firmware using several calls to cid_getstaroffset().
4) Create random data patterns in two buffers.
5) Configure the data router to route data to/from the first sip_fifo64k star using sxdx_configurerouter().
6) Send the random data using sipif_writedata().
7) Get the data back using sipif_readdata().
8) Configure the data router to route data to/from the second sip_fifo64k star using sxdx_configurerouter().
9) Send the random data using sipif_writedata().
10) Get the data back using sipif_readdata().
11) Compare input/output buffers for both FIFO transfers.
12) Report back to the console.

11 Evaluating FMC AD/DA product using Xilinx Development kits

11.1 VC707, KC705, ML605, SP601, SP605

1) Make sure the development board is powered down.
2) For ML605, make sure the PHY jumper setting on the development board selects GMII/MII interface mode. This is the factory default jumper setting.

   For VC707 and KC705, make sure the mode switch settings on the development boards select the JTAG interface mode.
3) Plug the FMC card into the development board. Use the LPC connector for LPC boards or use the HPC connector for HPC boards.

   Note that the HPC connector on KC705 does not support the HB signal pins. In order to use a fully compliable HPC connector, FMC700 is required.

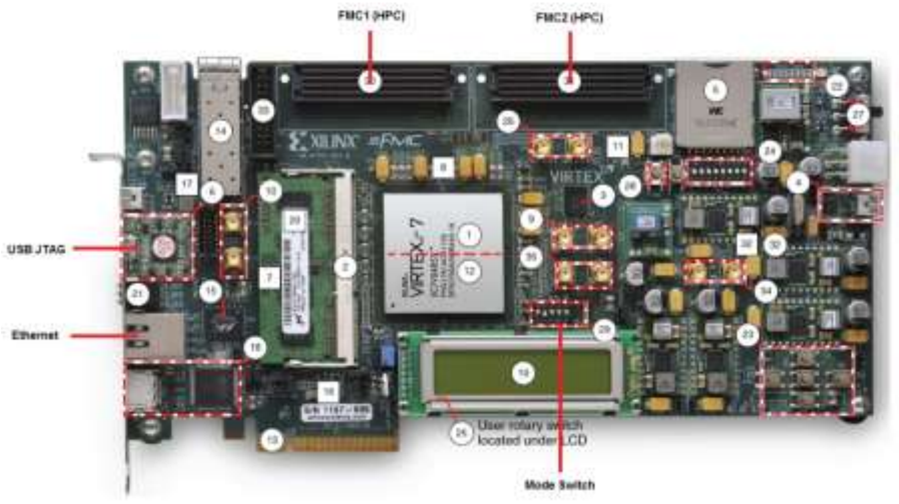


Figure 44: VC707 development board

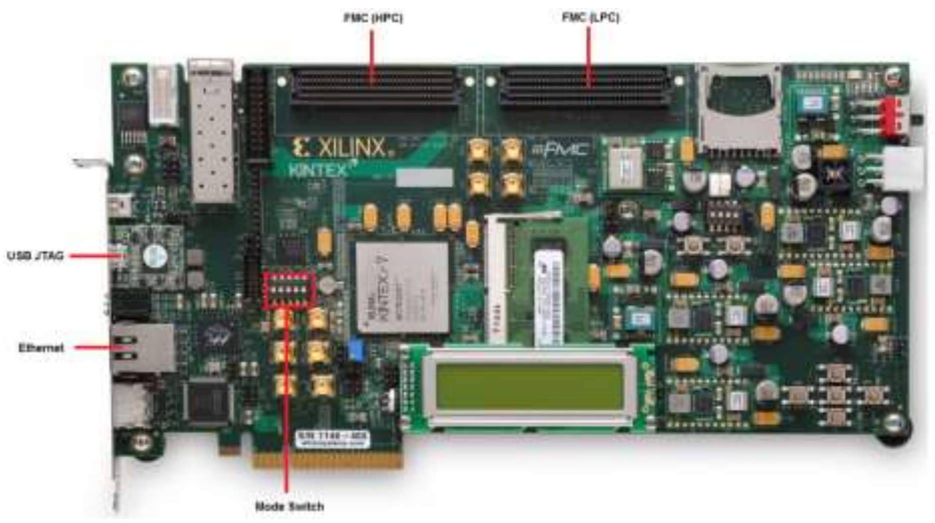


Figure 45: KC705 development board

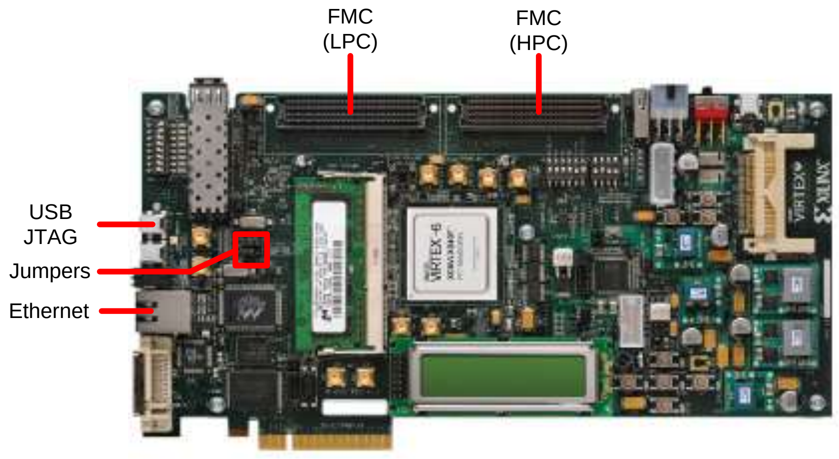


Figure 46: ML605 development board

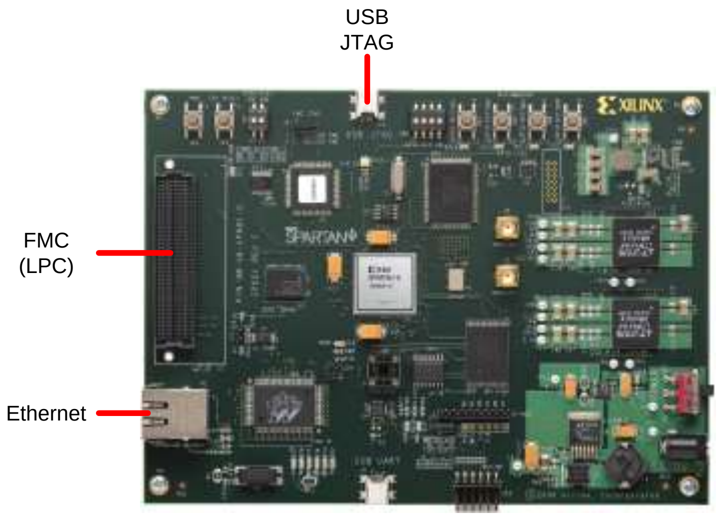


Figure 47: SP601 development board

4) Connect the USB cable included with the development board to the USB JTAG connection on the development board. Connect the other end of the cable to your host computer.
5) Connect the Ethernet cable included with the development board to the Ethernet connection on the development board. Connect the other end of the cable to your host computer (only direct connection **without** Ethernet switch/hub is supported).
6) Power up the development board.

!	**Make sure there is proper air flow across the FMC card when power is supplied to the board. A minimum airflow of 300 LFM is recommended. 4DSP's warranty does not cover boards on which the maximum allowed temperature has been exceeded.**

7) Load the reference firmware from "C:\Program Files\4dsp\Common\Firmware\Recovery".
8) Xilinx iMPACT can be used to directly load the .bit files to the FPGA using Boundary Scan. Choose the bit file that reflects your development board and FMC card, for example:
    a. "vc707_fmc150.bit" must be used for evaluation of the FMC150 on a VC707 development board.
    b. "kc705_fmc150.bit" must be used for evaluation of the FMC150 on a KC705 development board.
    c. "ml605_fmc150.bit" must be used for evaluation of the FMC150 on a ML605 development board.
    d. "sp601_fmc104.bit" must be used for evaluation of the FMC104 on a SP601 development board.
9) From a command window, browse to "C:\Program Files\4DSP\FMC Board Support Package\Bins" and run the application that matches your FMC card. Refer to Table 1. A list of Ethernet devices is shown. Look for the Ethernet device that connects to the development board.
10) Call the application. Here is the application usage:

```
FMCxxxApp.exe {interface type} {device type} {device index} {clock mode}
```

- "interface type" is either 0 (PCI) or 1 (Ethernet).
- "device type" is a board such as the VC707.
- "device index" is the NDIS device index found on 9).
- "clock mode" can be either 0 (Internal clock) or 1 (External Clock).

11) The sample data is stored in the folder "C:\Program Files\4DSP\FMC Board Support Package\Bins", where an ASCII file and a binary file are created for each channel (if applicable).
12) The DAC outputs will show a sine wave (if applicable).

FMC series	FMC Types	Application
FMC10x series	FMC108, FMC107, FMC104, FMC103	Fmc10xApp.exe
FMC110 series	FMC110	Fmc110App.exe
FMC11x series	FMC116, FMC112	Fmc116App.exe
FMC12x series	FMC122, FMC125, FMC126	Fmc12xApp.exe
FMC150 series	FMC150	Fmc150App.exe
FMC176 series	FMC176	Fmc176App.exe
FMC204 series	FMC204	Fmc204App.exe
FMC210 series	FMC210	Fmc210App.exe
FMC230 series	FMC230	Fmc230App.exe

Table 1: Application overview

Note: If the access rights are not correct, the application will not store any buffer to hard disk. If this happens, copy the compiled application somewhere that you have write access. Typically the Desktop is a place where every user has write access by default.

11.2 VC709

1) Make sure the development board is powered down.
2) Make sure the configuration switch (SW11) setting on the board selects JTAG interface mode. This is the factory default setting.
3) Plug the FMC card into the development board. Use the HPC connector for HPC boards or LPC boards.

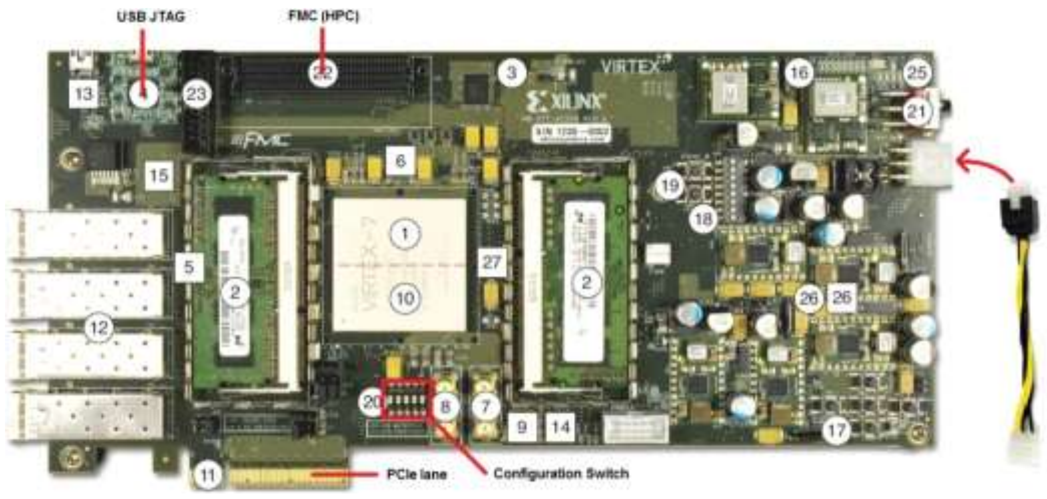


Figure 48: VC709 development board

4) Connect the USB cable included with the development board to the USB JTAG connection on the development board. Connect the other end of the cable to your host computer.
5) Connect the ATX adapter cable to J18 on the VC709 board. Plug the VC709 board into the PCIe connector.
6) Power up the development board.

!	**Make sure there is proper air flow across the FMC card when power is supplied to the board. A minimum airflow of 300 LFM is recommended. 4DSP's warranty does not cover boards on which the maximum allowed temperature has been exceeded.**

7) Load the reference firmware from "C:\Program Files\4dsp\Common\Firmware\Recovery".
8) Xilinx iMPACT can be used to directly load the .bit files to the FPGA using Boundary Scan. Choose the bit file that reflects your development board and FMC card, for example:
    a. "vc709_fmc230.bit" must be used for evaluation of the FMC230 on a VC709 development board.
9) **Do a soft-reboot.**
10) From a command window, browse to "C:\Program Files\4DSP\FMC Board Support Package\Bins" and run the application that matches your FMC card. Refer to Table 1 in Section 11.1.
11) Call the application. Here is the application usage:

```
FMCxxxApp.exe {interface type} {device type} {device index} {clock mode}
```

- "interface type" is either 0 (PCI) or 1 (Ethernet).
- "device type" is a board such as the VC709.
- "device index" is the NDIS device index found on 10).
- "clock mode" can be either 0 (Internal clock) or 1 (External Clock).

12) The sample data is stored in the folder "C:\Program Files\4DSP\FMC Board Support Package\Bins", where an ASCII file and a binary file are created for each channel (if applicable).
13) The DAC outputs will show a sine wave (if applicable).

11.3 ZC702, ZC706, Zedboard

1) Make sure the development board is powered down.
2) For ZC702 and ZC706, make sure the JTAG select switch setting on the board selects JTAG interface mode and the mode switch sets "00000".

   For Zedboard, make sure the mode jumper setting on the board selects JTAG interface mode.

3) Plug the FMC card into the development board. Use the HPC connector for HPC boards or the LPC connector for LPC boards. Note that ZC706's HPC connector does not support HA and HB signal pins.

For ZC702 and ZC706, set GPIO_DIP_SW0 to "0" on SW12. For Zedboard, set SW0 to "0". This switch acts as a FMC PRSNT signal. These switches act as a FMC PRSNT signal.

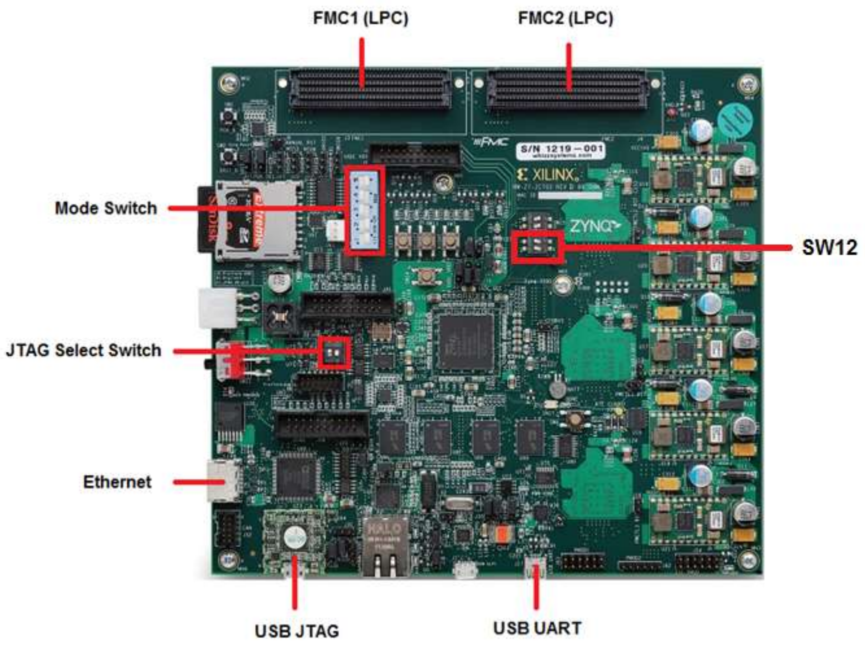


Figure 49: ZC702 development board

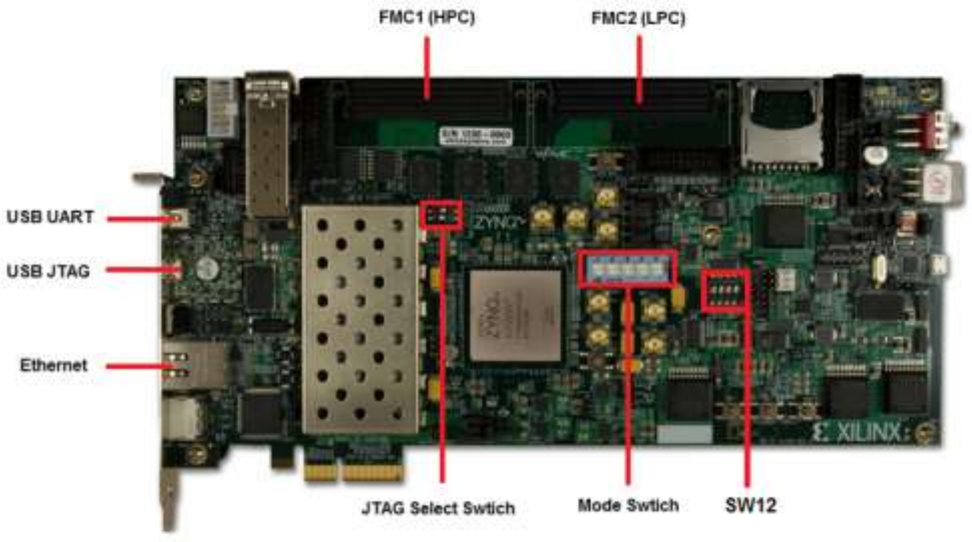


Figure 50: ZC706 development board

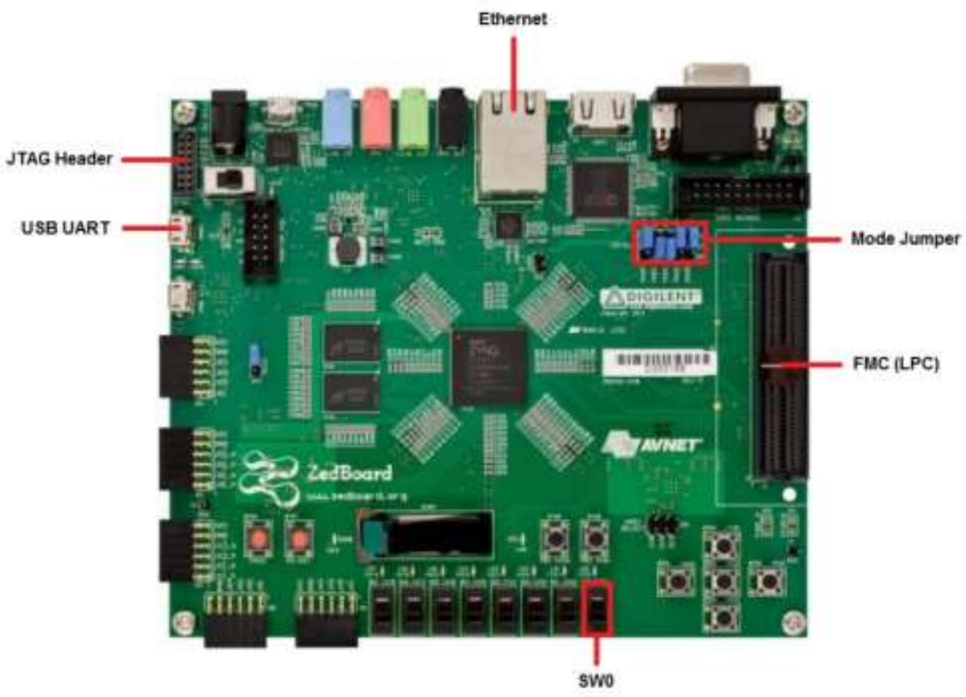


Figure 51: Zedboard development board

4) For ZC702 and ZC706, connect the USB cables included with the development board to the USB JTAG connection and the USB to UART connection on the development board.

   For Zedboard, connect the USB cable included with the development board to the USB to UART connection and connect the Xilinx Platform Cable USB that is NOT included in the board package to the Xilinx JTAG header on the development board.

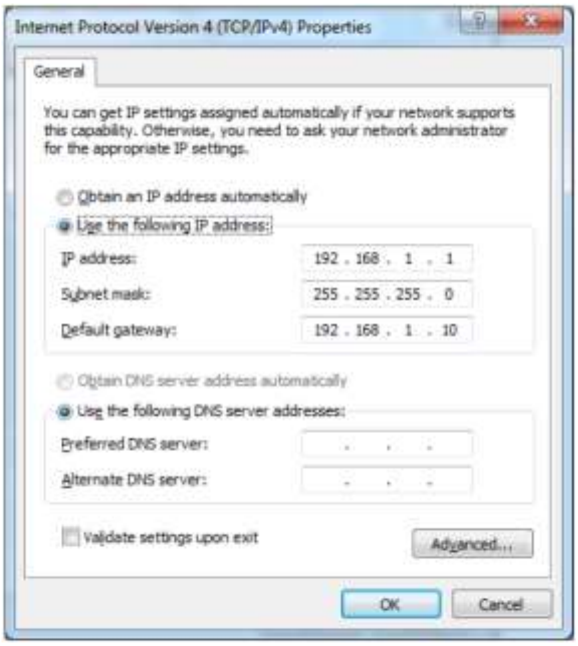
5) Install USB UART driver. Please refer to the Xilinx and Zedboard HW User Guides.

6) Connect the Ethernet cable included with the development board to the Ethernet connection on the development board. Connect the other end of the cable to your host computer (only direct connection **without** Ethernet switch/hub is supported).

7) Power up the development board.

!	**Make sure there is proper air flow across the FMC card when power is supplied to the board. A minimum airflow of 300 LFM is recommended. 4DSP's warranty does not cover boards on which the maximum allowed temperature has been exceeded.**

8) Open the Network and Sharing Center on Windows 7. Click "Change adapter settings". Open Properties on Local Area Connection. Click "Internet Protocol Version 4 (TCP/IPv4)" and open Properties. TCP/IPv4 must set as seen in Figure 52.


Internet Protocol Version 4 (TCP/IPv4) Properties

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

○ Obtain an IP address automatically
● Use the following IP address:

IP address:	192 . 168 . 1 . 1
Subnet mask:	255 . 255 . 255 . 0
Default gateway:	192 . 168 . 1 . 10

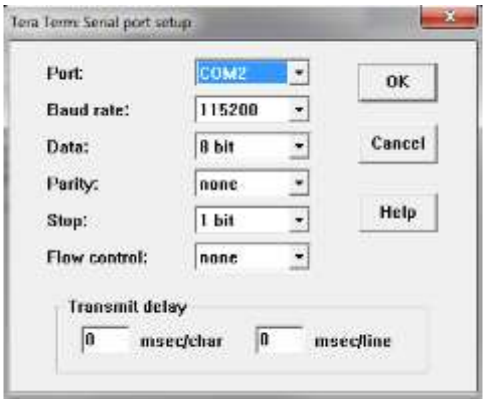
○ Obtain DNS server address automatically
● Use the following DNS server addresses:

Preferred DNS server:	. . .
Alternate DNS server:	. . .

☐ Validate settings upon exit

Figure 52: TCI/IP settings on the host computer

9) Open the terminal emulator and connect it to the port that is connected to USB UART. Tera Term is used in this example. Please refer the settings used in Figure 53.


Tera Term: Serial port setup

Port:	COM2
Baud rate:	115200
Data:	8 bit
Parity:	none
Stop:	1 bit
Flow control:	none

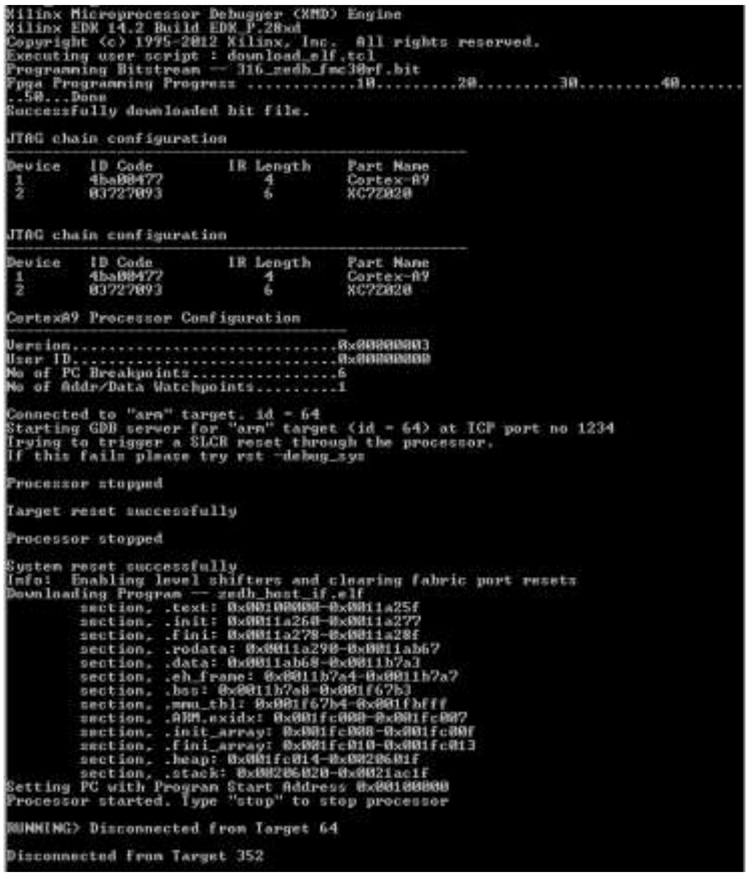
Transmit delay: 0 msec/char 0 msec/line

Figure 53: Serial port setup

10) Open the **ISE Command Prompt** and go to the reference firmware in “C:\Program Files\4dsp\Common\Firmware\Recovery”. The following files are required to upload a bit file.
    - xxx_xxx_xxx.bit
    - ps7_init.tcl
    - go.bat
    - download_elf.tcl
    - xxx_host_if.elf

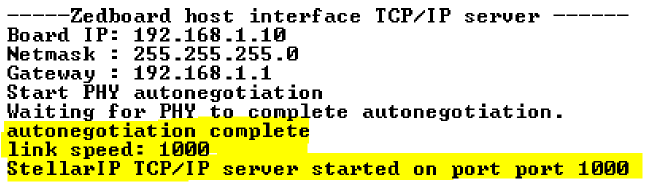
    a. “zc702_fmc30rf.bit” must be used for evaluation of the FMC30RF on a ZC702 development board.
    b. “zc706_fmc30rf.bit” must be used for evaluation of the FMC30RF on a ZC706 development board.
    c. “zedb_fmc30rf.bit” must be used for evaluation of the FMC30RF on a Zedboard development board.

11) Type “go *.bit”. The programming file is being uploaded as shown in Figure 54. Wait until TCP/IP server has started as shown in Figure 55.



```
Xilinx Microprocessor Debugger (XMD) Engine
Xilinx EDK 14.2 Build EDK_P.28xd
Copyright (c) 1995-2012 Xilinx, Inc.  All rights reserved.
Executing user script : download_elf.tcl
Programming Bitstream -- 316_zedb_fmc30rf.bit
Fpga Programming Progress ............10.........20.........30.........40.......
..50...Done
Successfully downloaded bit file.

JTAG chain configuration
--------------------------------------------------
Device   ID Code        IR Length    Part Name
 1       4ba00477           4        Cortex-A9
 2       03727093           6        XC7Z020

JTAG chain configuration
--------------------------------------------------
Device   ID Code        IR Length    Part Name
 1       4ba00477           4        Cortex-A9
 2       03727093           6        XC7Z020

CortexA9 Processor Configuration
-------------------------------------
Version.............................0x00000003
User ID.............................0x00000000
No of PC Breakpoints.................6
No of Addr/Data Watchpoints..........1

Connected to "arm" target. id = 64
Starting GDB server for "arm" target (id = 64) at TCP port no 1234
Trying to trigger a SLCR reset through the processor.
If this fails please try rst -debug_sys

Processor stopped

Target reset successfully

Processor stopped

System reset successfully
Info:  Enabling level shifters and clearing fabric port resets
Downloading Program -- zedb_host_if.elf
        section, .text: 0x00100000-0x0011a25f
        section, .init: 0x0011a260-0x0011a277
        section, .fini: 0x0011a278-0x0011a28f
        section, .rodata: 0x0011a290-0x0011ab67
        section, .data: 0x0011ab68-0x0011b7a3
        section, .eh_frame: 0x0011b7a4-0x0011b7a7
        section, .bss: 0x0011b7a8-0x001f67b3
        section, .mmu_tbl: 0x001f67b4-0x001fbfff
        section, .ARM.exidx: 0x001fc000-0x001fc007
        section, .init_array: 0x001fc008-0x001fc00f
        section, .fini_array: 0x001fc010-0x001fc013
        section, .heap: 0x001fc014-0x0020601f
        section, .stack: 0x00206020-0x0021ac1f
Setting PC with Program Start Address 0x00100000
Processor started. Type "stop" to stop processor

RUNNING> Disconnected from Target 64

Disconnected from Target 352
```

Figure 54: Output of the ISE Command Prompt



```
-----Zedboard host interface TCP/IP server ------
Board IP: 192.168.1.10
Netmask : 255.255.255.0
Gateway : 192.168.1.1
Start PHY autonegotiation
Waiting for PHY to complete autonegotiation.
autonegotiation complete
link speed: 1000
StellarIP TCP/IP server started on port port 1000
```

Figure 55: Output of the terminal emulator

12) From a command window browse to "C:\Program Files\4DSP\FMC Board Support Package\Bins" and run the application that matches your FMC card. Refer to Table 1 in Section 11.1.
13) Call the application. Here is the application usage:

```
FMCxxxApp.exe {interface type} {device type} {device index} {clock mode}
```

- "interface type" is either 0 (PCI), 1 (Ethernet), or 2(TCP/IP).
- "device type" is a TCP/IP address (192.168.1.10).
- "device index" is the port (1000).
- "clock mode" can be either 0 (RX out / TX out), 1 (RX out / TX in), or 2 (RX in / TX out).

14) The sample data is stored in the folder "C:\Program Files\4DSP\FMC Board Support Package\Bins", where an ASCII file and a binary file is created for each channel (if applicable).
15) The DAC outputs will show a sine wave (if applicable).

11.4 Documentation

For each board type, a user manual is installed in "C:\Program Files\4DSP\FMC Board Support Package\Docs".

The firmware (constellation) documentation as well as star (IP core) documentation is found in the .4ff files. More information about firmware and 4ff files can be found in Chapter 4.3 Firmware overview.

11.5 Reference Firmware

Please contact 4DSP sales or visit www.4dsp.com for the reference firmware availability.

12 Evaluating FMC DSP products using Xilinx Development kits

12.1 Quick start

This section provides a short introduction to what the development framework looks like for the FMC645 and FMC667 DSP products. A description of the steps required to get started with the framework is also included.

A typical FMC645 development system is comprised of a host computer, an ML605/VC707 evaluation kit from Xilinx, and an FMC645 mezzanine board.

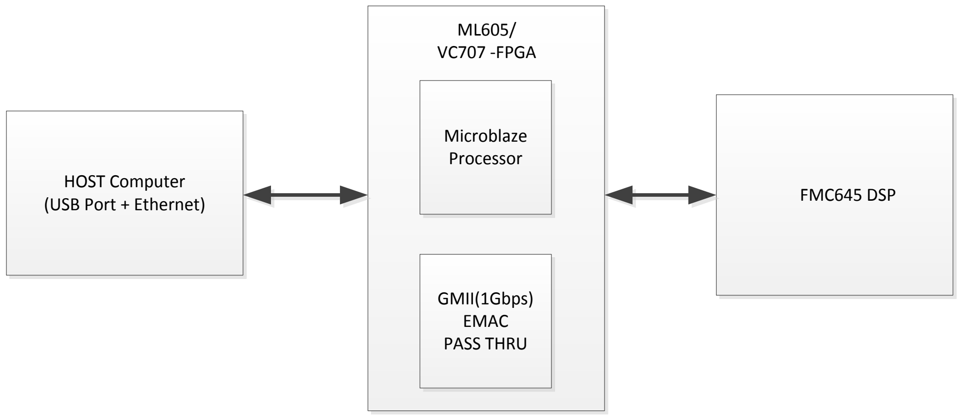


Figure 56: Elements constituting the FMC645-DSP environment

A typical FMC667 development system is comprised of a host computer, a KC705/VC707 evaluation kit from Xilinx, and an FMC667 mezzanine board.

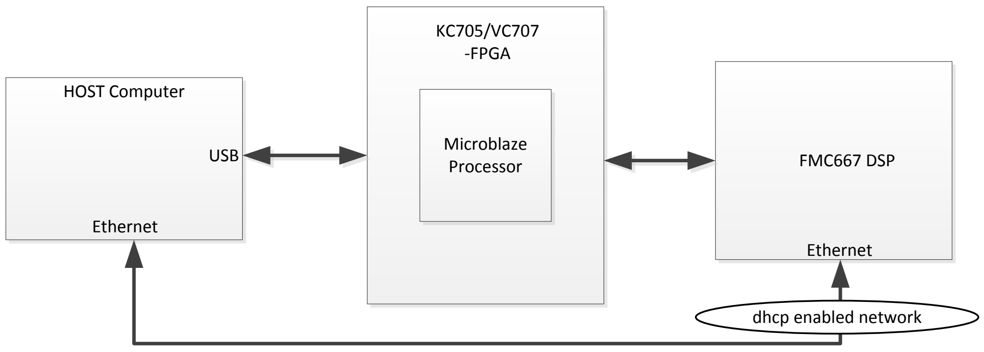
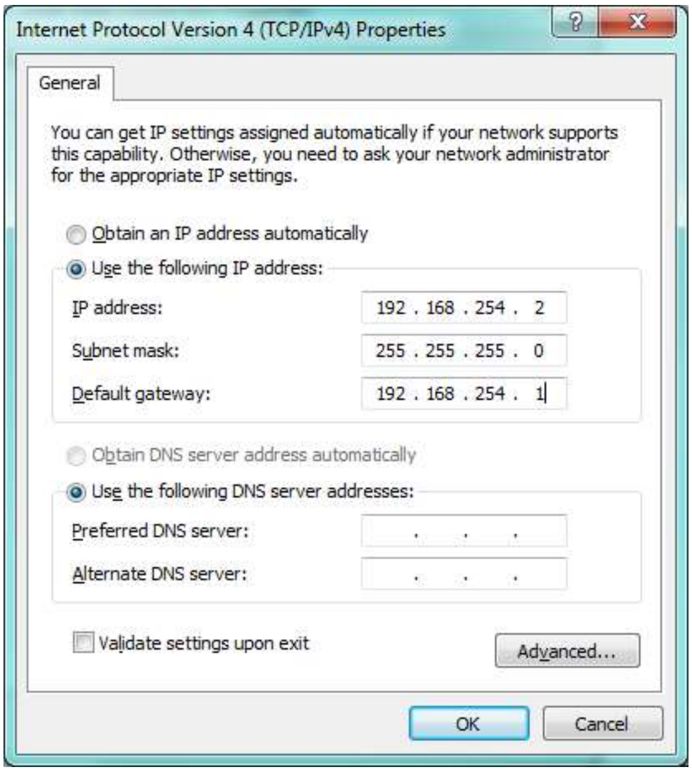


Figure 57: Elements constituting the FMC667-DSP environment

Note that the framework expects the host computer to have a 1Gbps Ethernet port (10Mbps and 100Mbps are not supported by the ML605 FPGA firmware).

12.1.1 FMC645 TCP/IP settings

The TCP/IP settings in Windows should be changed for the Ethernet interface set to receive connection to the test hardware. The settings should match those seen in Figure 58. Note that this interface **must** support 1Gbps (one Gigabit per second), as the firmware does not support 10/100Mbps speeds.



Internet Protocol Version 4 (TCP/IPv4) Properties

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

Obtain an IP address automatically [✓ Use the following IP address:]

IP address:	192 . 168 . 254 . 2
Subnet mask:	255 . 255 . 255 . 0
Default gateway:	192 . 168 . 254 . 1

Obtain DNS server address automatically [✓ Use the following DNS server addresses:]

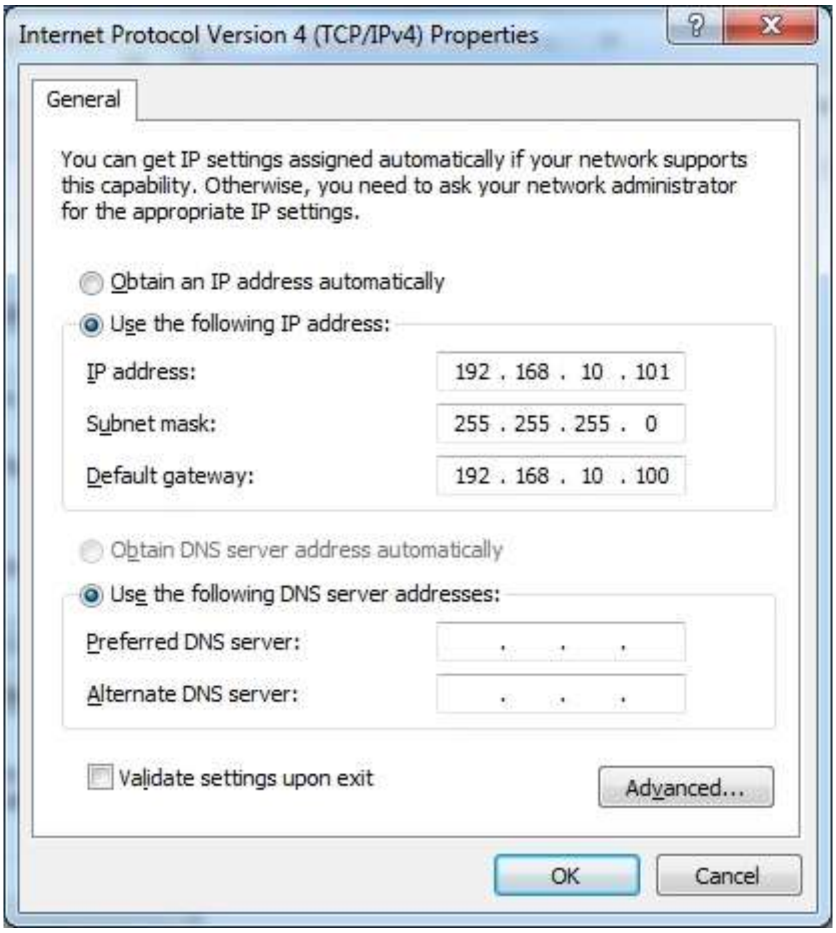
Preferred DNS server:	. . .
Alternate DNS server:	. . .

☐ Validate settings upon exit

Figure 58: Valid TCP/IPv4 settings for the framework

12.1.2 FMC667 TCP/IP settings

The TCP/IP settings in Windows should be changed for the Ethernet interface set to receive connection to the test hardware. The settings should match those seen in Figure 59. Note that this interface **must** support 1Gbps (one Gigabit per second) as the firmware does not support 10/100Mbps speeds.


Internet Protocol Version 4 (TCP/IPv4) Properties

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

Obtain an IP address automatically [✓ Use the following IP address:]

IP address: 192 . 168 . 10 . 101
Subnet mask: 255 . 255 . 255 . 0
Default gateway: 192 . 168 . 10 . 100

Obtain DNS server address automatically [✓ Use the following DNS server addresses:]

Preferred DNS server: . . .
Alternate DNS server: . . .

Validate settings upon exit

Figure 59: Valid TCP/IPv4 settings for the framework

Note that the virtual com driver required by the CP210X chip on the Xilinx development kits should be downloaded from the Silicon Labs website. A direct link to the actual installer is CP210x driver download. Please reboot your host computer if prompted to do so.

12.1.3 Prepare the KC705 for FMC667 test setup (FMC667 and KC705 ONLY)

The KC705 requires a FLASH programming file (.mcs) to configure its on-board voltage level to work with the FMC667. 4DSP has created an FPGA design that will communicate with the voltage controller that makes the VADJ on the KC705. After the firmware loads, it will change the default 2V5 to become 1V8.

The required FLASH programming file (.mcs) provided by 4DSP is found in the same folder as the FPGA programming file. These items are found in the recovery portion of the firmware file installed by this installer. The default path for these recovery files is 'C:\Program Files\4DSP\Common\Firmware\Recovery\356_kc705_fmc667_lpc'.

More information about firmware and recovery files can be found in Chapter 4.6 4FM Firmware (Recovery Files).

Procedure for the FMC667 and KC705:

1) Configure KC705's SW13 dip switch block as follows (SW13.1=Off, SW13.2=Off, SW13.3=Off, SW13.4=On, and SW13.5=Off).
2) Attach a USB cable from KC705's JTAG port to the host computer.
3) Power up KC705.
4) Open Xilinx iMPACT and choose "Boundary Scan" from the left pane.
5) Right-click on the right pane and choose "Initialize Chain" from the context menu.
6) Double-click SPI/BPI above the Kintex 7 device on the chain representation.
7) In the "ADD Prom File" dialog, browse to the correct FLASH programming file (.mcs) and press OK.
8) In the "Select Attached SPI/BPI" dialog, choose "BPI PROM", "28F00AP30", "16", and "25:24". Press OK.
9) Click on the "FLASH" device above Kintex 7 device and double-click "Program" from the "iMPACT Processes" on the left pane.
10) Leave all the default settings in the "Device Programming Properties" dialog and press OK.
11) Wait for about 20-25 minutes for the FLASH upload process to be completed.

12.1.4 Prepare and connect Xilinx Development kit to the host computer

FMC645:

Three cables should be connected between the ML605/VC707 and the host computer:

- A USB cable between ML605/VC707's "JTAG" and the host computer.
- A USB cable between ML605/VC707's "UART" and the host computer.
- An Ethernet cable between ML605/VC707's Ethernet connector and the host computer. The Ethernet interface on the host computer should support 1Gbps as 10/100Mbps are not supported by the FPGA firmware.

The FMC645 mezzanine should be attached to the ML605's **HPC** FMC connector or the VC707's **FMC 1 HPC (J35)** FMC connector.

FMC667:

Two cables should be connected between the KC705/VC707 and the host computer and one between the host computer and the FMC667:

- A USB cable between KC705/VC707's "JTAG" and the host computer.
- A USB cable between KC705/VC707's "UART" and the host computer.
- An Ethernet cable between the FMC667 and the host computer.

The FMC667 mezzanine should be attached to the KC705's **HPC** FMC connector or the VC707's **FMC 1 HPC (J35)** FMC connector.

12.1.5 Programming the Xilinx development board FPGA device

The FPGA on the development board under test should be programmed with a firmware in order to operate properly.

The required FPGA programming file (.bit) provided by 4DSP is found in the recovery portion of the firmware files installed by this installer. The default path for the recovery files is:

FMC645 + ML605: 'C:\Program Files\4DSP\Common\Firmware\Recovery\196_ml605_fmc645'

FMC645 + VC707: 'C:\Program Files\4DSP\Common\Firmware\Recovery\255_vc707_fmc645'

FMC667 + KC705: 'C:\Program Files\4DSP\Common\Firmware\Recovery\356_kc705_fmc667_lpc'

FMC667 + VC707: 'C:\Program Files\4DSP\Common\Firmware\Recovery\355_vc707_fmc667_lpc'

Xilinx iMPACT should be used to "upload" the FPGA programming file (.bit) to the FPGA.

More information about firmware and recovery files can be found in Chapter 4.6 4FM Firmware (Recovery Files).

12.1.6 Programming ML605 Flash device (Optional)

If the FLASH device is not programmed, then iMPACT should be used to program the FPGA programming file (.bit) into the FPGA each time the ML605 is power cycled. This is equivalent to repeating the instructions in Section 12.1.5 Programming ML605 FPGA device (Virtex 6 LX240T).

The required FLASH programming file (.mcs) provided by 4DSP is found in the same folder as the FPGA programming file. These items are found in the recovery portion of the firmware file installed by this installer. The default path for these recovery files is 'C:\Program Files\4DSP\Common\Firmware\Recovery\196_ml605_fmc645'.

More information about firmware and recovery files can be found in Chapter 4.6 4FM Firmware (Recovery Files).

Procedure for the FMC645 and ML605:

1) Configure ML605's S2 dip switch block as the following (S2.1=Off, S2.2=On, S2.3=Off, S2.4=On, S2.5=Off, and S2.6=Off).
2) Attach a USB cable from ML605's JTAG port to the host computer.
3) Power up ML605.
4) Open Xilinx iMPACT and choose 'Boundary Scan' from the left pane.
5) Right-click on the right pane and choose "Initialize Chain" from the contextual menu.
6) Double-click SPI/BPI above "xc6lv240t" on the chain representation.
7) In the "ADD Prom File" dialog, browse to the correct FLASH programming file (.mcs) and press OK.
8) In the "Select Attached SPI/BPI" dialog, choose "BPI PROM", "28F256P30", "16", and "NOT USED". Press OK.
9) Click on the "FLASH" device above "xc6vlx240t" and double-click "Program" from the "iMPACT Processes" on the left pane.
10) Leave all the default settings in the "Device Programming Properties" dialog and press OK.
11) Wait for about 20-25 minutes for the FLASH upload process to be completed.

12.1.7 Verify FPGA firmware initialization

After programming the development board's FPGA using iMPACT (or having the FPGA configuration stored into ML605 FLASH for the FMC645), it is mandatory to check for proper firmware initialization. A wrong firmware initialization indicates that the FPGA firmware is not ready for operation. If you see an incorrect LEDs status, press "CPU RST" to reset the complete Xilinx development board and restart the FPGA firmware initialization. The switch for "CPU RST" is as follows for each development board:

- ML605:- SW10
- KC705:- SW7
- VC707:- SW8

Note that an FMC645/FMC667 mezzanine board should be attached to the HPC connectors (as outlined in Chapter 9.1.3) for the GPIO LEDs to display a proper initialization status. Examples of the correct LED sequence for each development board are shown below in Figure 60. The same sequence applies for all development boards and both mezzanine cards. The FPGA firmware initialization status is displayed on the LEDs of each development board. It is important that these LEDs display "Ready" status.

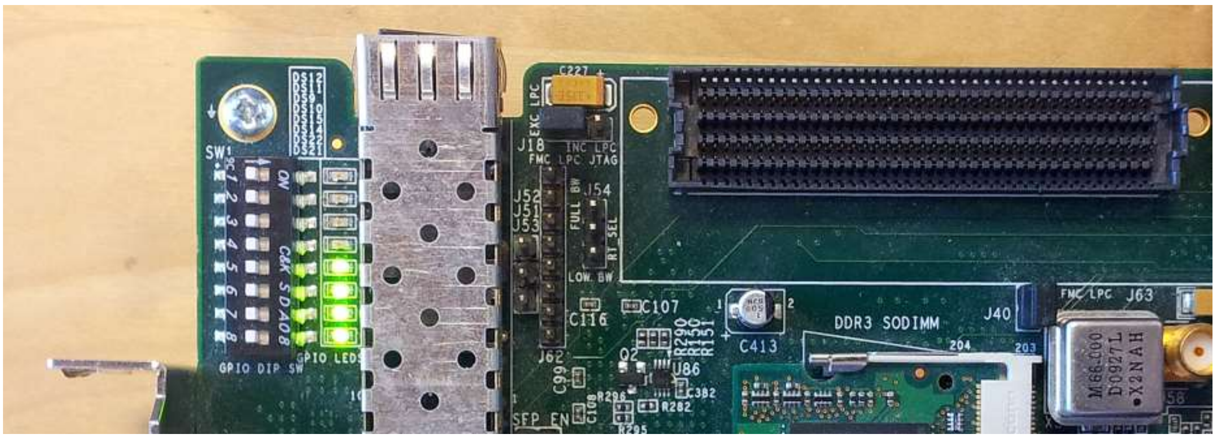


Figure 60: ML605 GPIO LEDs showing status of a proper firmware initialization with FMC645

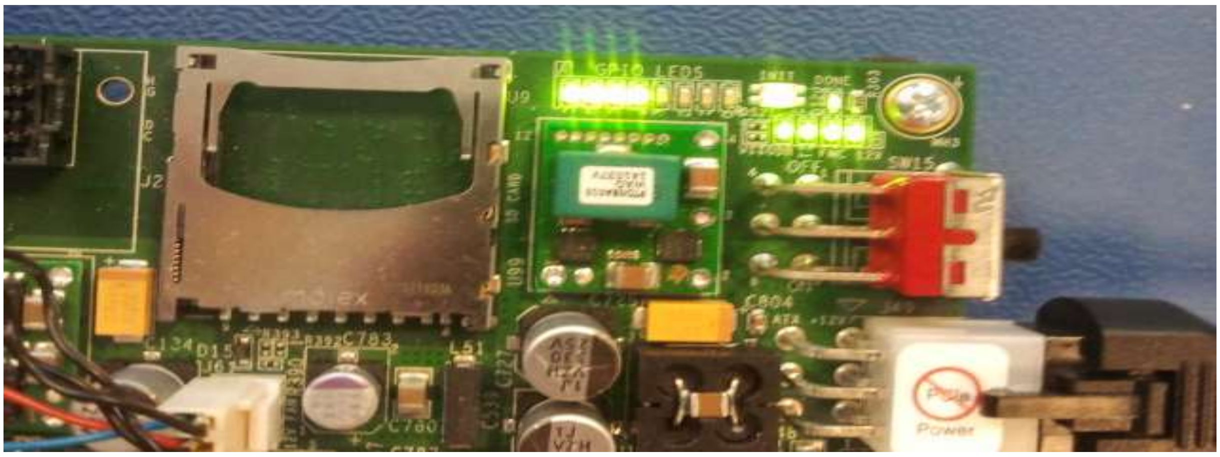


Figure 61: KC705 GPIO LEDs showing status of a proper firmware initialization with FMC667

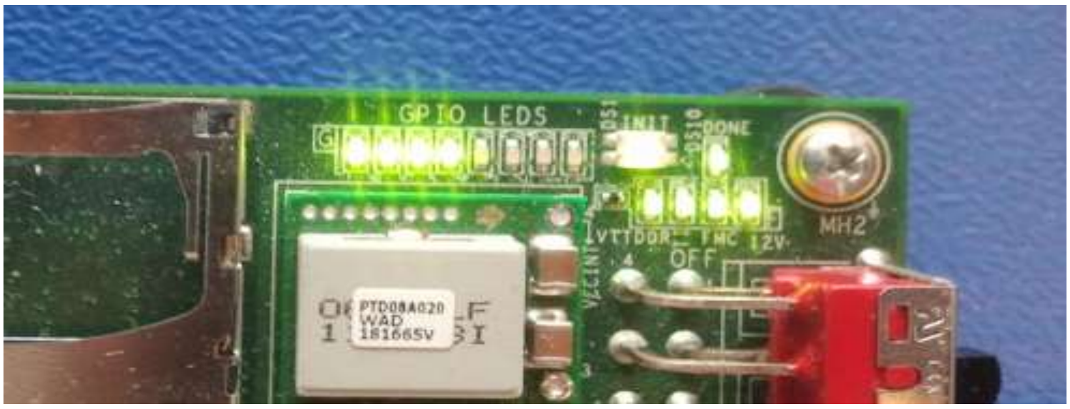


Figure 62: VC707 GPIO LEDs showing status of a proper firmware initialization with FMC667

The information displayed on the VC707 LCD shown in Figure 63 is an example of what should be expected on the KC705 and VC707 development boards after proper initialization.

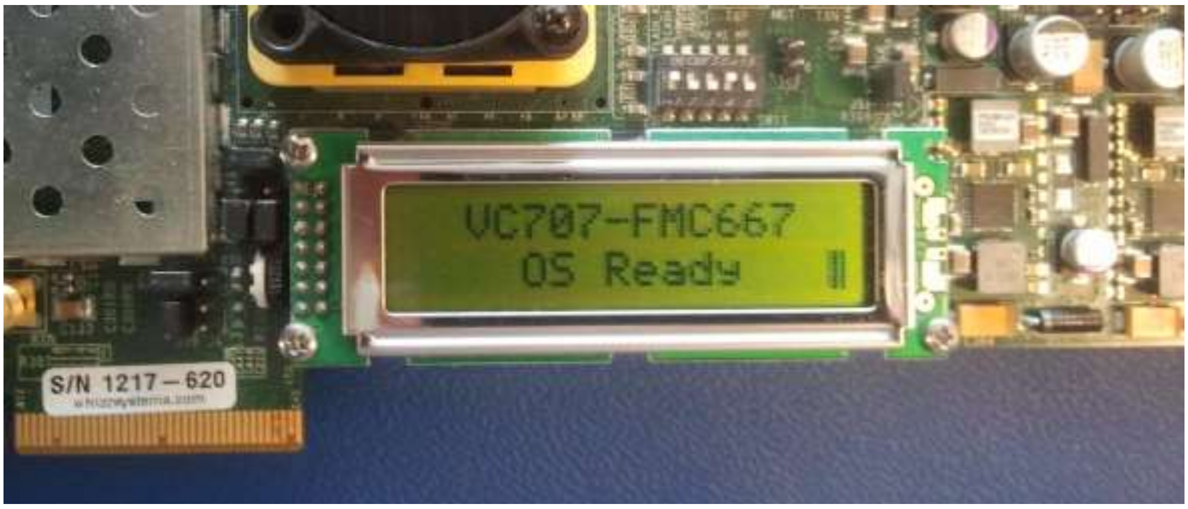
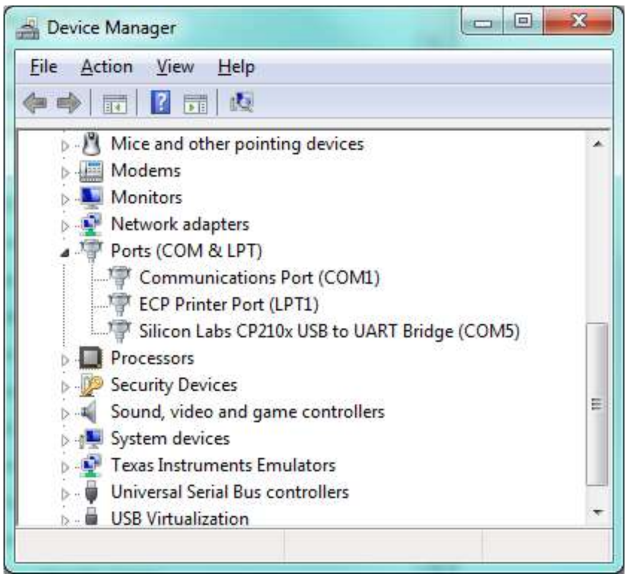


Figure 63: VC707 LCD showing status of a proper firmware initialization with FMC667

12.1.8 Check which virtual COM (UART) port is featured by the device driver

While following the instructions in Chapter 12.1.1 Prepare the host computer, we installed a virtual COM port driver downloaded from the Silicon Labs website. The driver will kick in as soon the development board is powered up and a USB cable is connected between the development board's "UART" connector and the host computer.

The Windows device manager in the "Ports (COM & LPT)" category indicates which COM port the development board is mapped to through the virtual COM port. In our case the COM port name is "COM5" as seen in Figure 64. This is the actual COM port argument we will be passing to the test application later on.



Mice and other pointing devices
Modems
Monitors
Network adapters
Ports (COM & LPT)
Communications Port (COM1)
ECP Printer Port (LPT1)
Silicon Labs CP210x USB to UART Bridge (COM5)
Processors
Security Devices
Sound, video and game controllers
System devices
Texas Instruments Emulators
Universal Serial Bus controllers
USB Virtualization

Figure 64: Which COM port is featured by the system

12.1.9 Description of the FMC645 and FMC667 Test Peripheral Applications

4DSP provides reference applications to check all the FMC645 and FMC667 interfaces[1]. The DSP device is in charge of testing various DSP interfaces. It is controlled by the host computer through a processor implemented in the development board FPGA. The required application is provided as source code and as a precompiled executable. The precompiled binary is the component we are going to use in the next chapter. Complete execution of the application takes 5 to 10 minutes.

The FMC645 application tests the following interfaces:

- I2C
- PCI
- GPIO
- EMIFA
- MCBSP
- ETHERNET
- SRIO
- DDR2

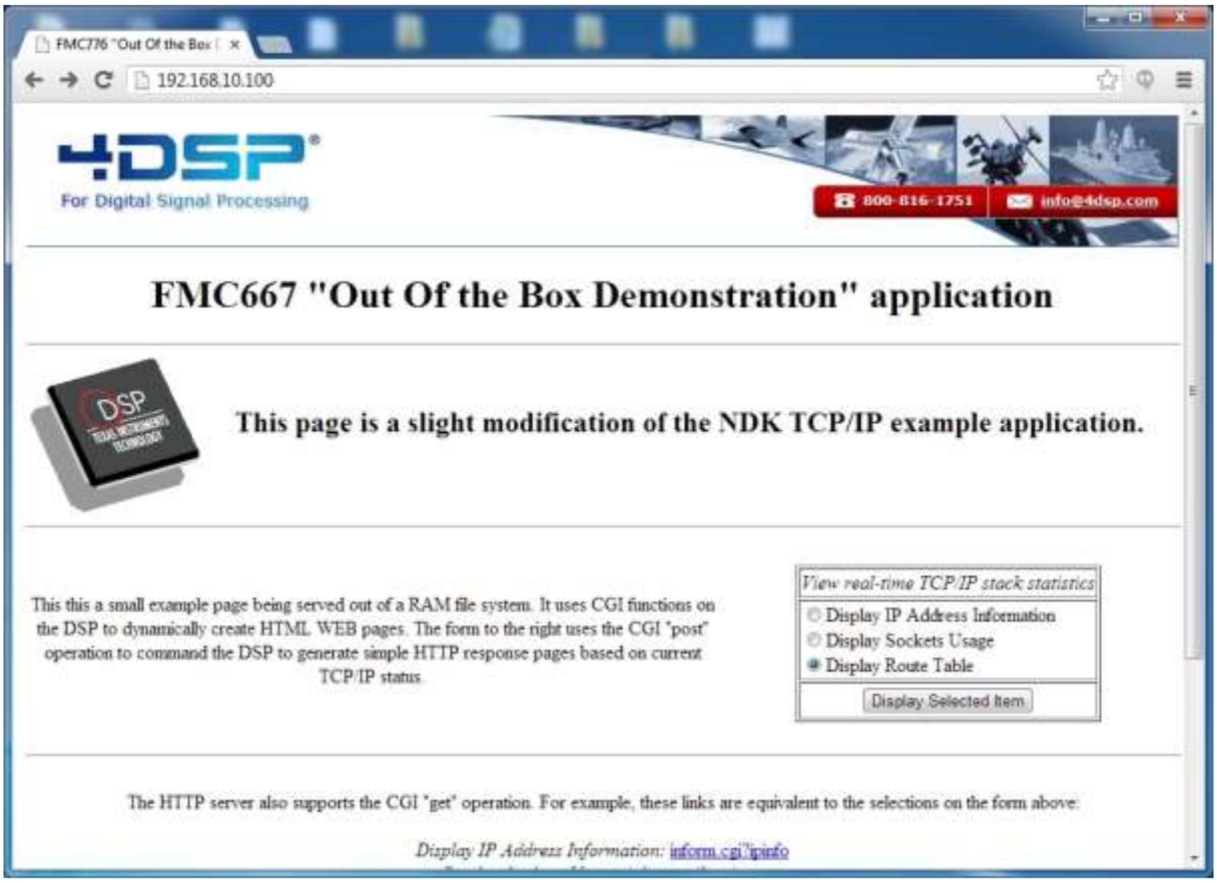
The FMC667 application tests the following interfaces:

- I2C
- PCIe
- GPIO/BOOT
- EMIF16
- TIMER INPUT
- DDR3
- ETHERNET

[1] The FMC667 reference application can have reduced functionality. There are LPC and HPC support available. Please check which Board Support Package has been provided with the Hardware.

12.1.10 Test the FMC667 with embedded web server

A web server is pre-loaded into the flash memory of the FMC667 by 4DSP. Make sure the Ethernet settings are configured for the FMC667 as described above. Open a browser window and go to “http://192.168.10.100”. The following HTML page should load from the DSP on the FMC667:



FMC667 "Out Of the Box Demonstration" application

This page is a slight modification of the NDK TCP/IP example application.

This this a small example page being served out of a RAM file system. It uses CGI functions on the DSP to dynamically create HTML WEB pages. The form to the right uses the CGI "post" operation to command the DSP to generate simple HTTP response pages based on current TCP/IP status.

View real-time TCP/IP stack statistics

- ○ Display IP Address Information
- ○ Display Sockets Usage
- ● Display Route Table

The HTTP server also supports the CGI "get" operation. For example, these links are equivalent to the selections on the form above:

Display IP Address Information: inform.cgi?ipinfo

12.1.11 Executing the FMC645/FMC667 Test Peripheral Application

In a command prompt window, navigate (cd) to FMC board support package binary folder and execute the application. The default location is C:\Program Files\4DSP\FMC Board Support Package\Bins.

- cd C:\Program Files\4DSP\FMC Board Support Package\Bins
- fmc645_microblaze_test_peripherals.exe **COM5**

  OR
- cd C:\Program Files\4DSP\FMC Board Support Package\Bins
- fmc667_microblaze_test_peripherals.exe **COM5**

Note that COM5, passed as an argument, should be replaced by the actual COM port found while following the instructions in Chapter 12.1.8 Check which virtual COM (UART) port is featured by the device driver.

An example of the correct console output can be seen in the Annexes for the FMC645 on the ML605 and the FMC667 on the VC707 (Annex 5 – fmc645 test peripherals log and Annex 6 – fmc667 test peripherals log).

12.1.12 Obtaining more information, technical support

Please refer to the FMC645 or the FMC667 user manuals for a detailed description of the BSP and detailed instructions on how to compile the FPGA firmware, the MicroBlaze software, and the DSP software.

In case of technical problems, please register to the 4DSP support forum (4DSP Forum) and indicate the serial number of your mezzanine card.

13 Annex 1 – Manual Firmware Installation

The firmware is automatically decrypted and extracted by the installer during the installation. Instructions for manually decrypting the firmware are provided below, if needed.

Note that the firmware package contains a lot of different firmware. For example, if you purchased an FM482, you will not have a license line for FC6301 and will not be able to decrypt the FC6301 firmware.

Do not attempt to install any other firmware that is not specific to your hardware.

Note that this step does not cover uploading a firmware into the hardware but only the decryption and extraction process. Without a decryption license for a specific firmware, you are only able to access recovery files. These files are ready to be uploaded in the firmware.

The recovery firmware files have a **.hex** extension and are located at the following paths:

C:\Program Files\4DSP\Common\Firmware\Recovery (32-bit)

C:\Program Files (x86)\4DSP\Common\Firmware\Recovery (64-bit)

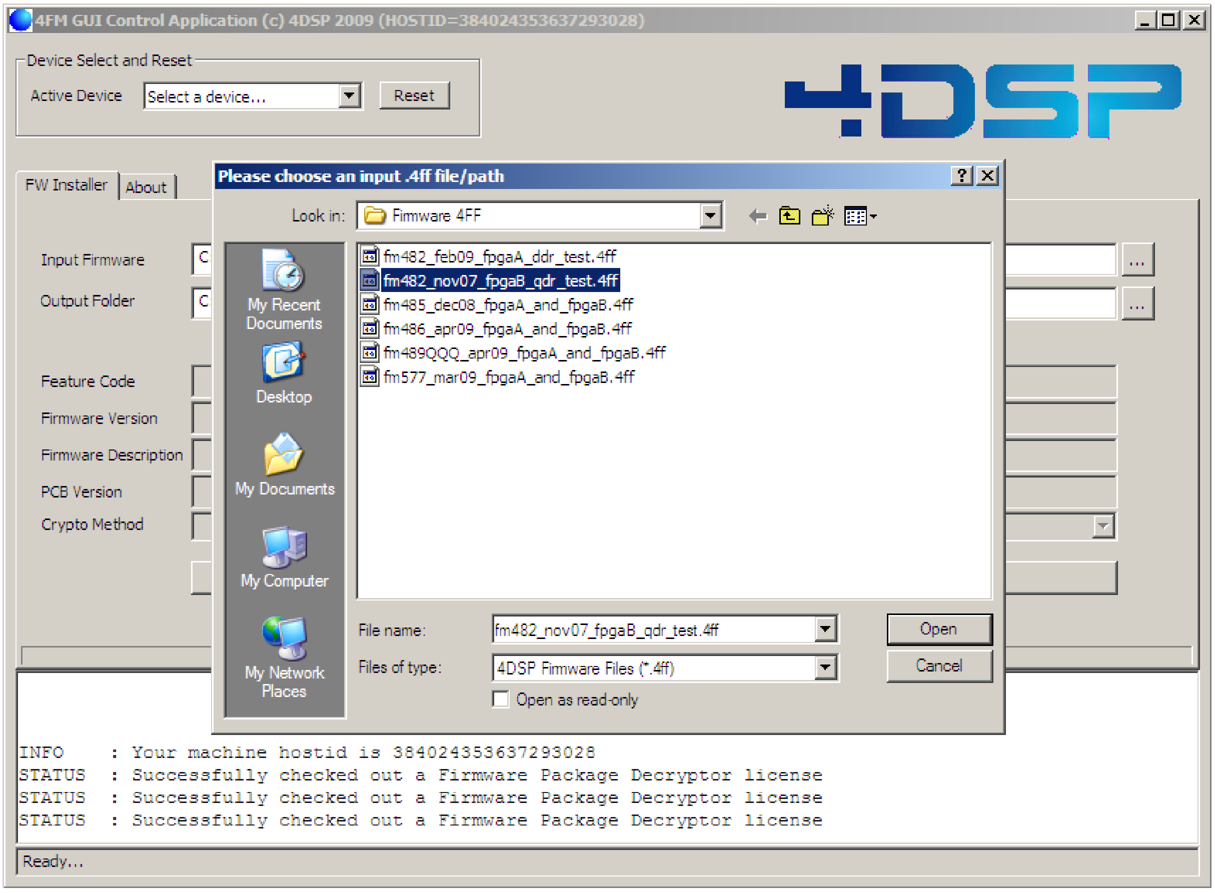
The encrypted firmware packages have a **.4ff** extension and are located at the following paths:

C:\ Program Files\4DSP\Common\Firmware\ (32-bit)

C:\ Program Files (x86)\4DSP\Common\Firmware\ (64-bit)

13.1 Select a input firmware package file

Open the 4FM GUI, select your device from the dropdown menu, and wait for the plug-ins to load. Select the “FW Installer” tab, locate “Input Firmware”, and click the button marked “…” on the right side. Choose a firmware package file. In this example, we are installing firmware for the FM482. We have the correct license code for FM482.

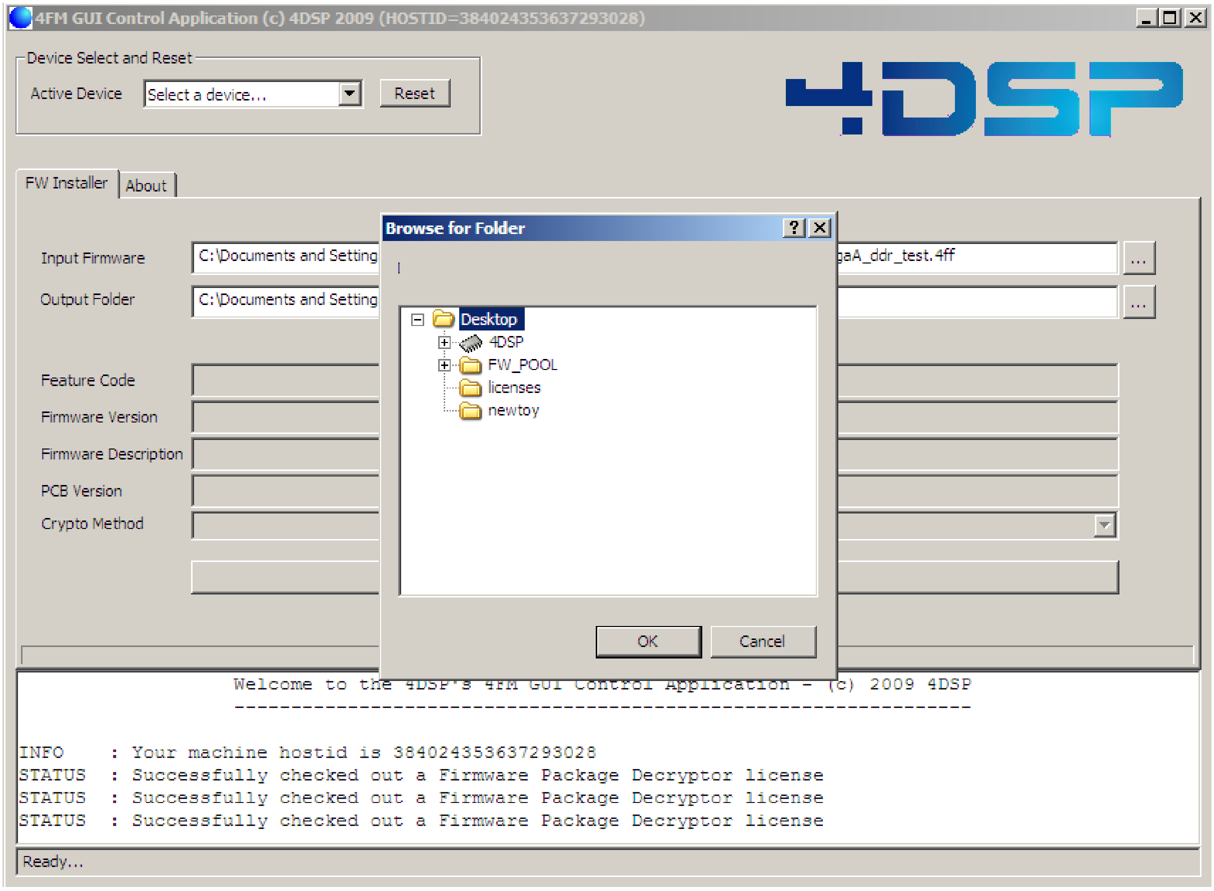

4FM GUI Control Application (c) 4DSP 2009 (HOSTID=384024353637293028)
Device Select and Reset
Active Device
Select a device...
Reset
FW Installer
About
Input Firmware
Output Folder
Feature Code
Firmware Version
Firmware Description
PCB Version
Crypto Method
Please choose an input .4ff file/path
Look in:
Firmware 4FF
My Recent Documents
Desktop
My Documents
My Computer
My Network Places
fm482_feb09_fpgaA_ddr_test.4ff
fm482_nov07_fpgaB_qdr_test.4ff
fm485_dec08_fpgaA_and_fpgaB.4ff
fm486_apr09_fpgaA_and_fpgaB.4ff
fm489QQQ_apr09_fpgaA_and_fpgaB.4ff
fm577_mar09_fpgaA_and_fpgaB.4ff
File name:
fm482_nov07_fpgaB_qdr_test.4ff
Files of type:
4DSP Firmware Files (*.4ff)
Open as read-only
Open
Cancel
Ready...

```
INFO    : Your machine hostid is 384024353637293028
STATUS  : Successfully checked out a Firmware Package Decryptor license
STATUS  : Successfully checked out a Firmware Package Decryptor license
STATUS  : Successfully checked out a Firmware Package Decryptor license
```

Figure 65: Browsing to a firmware file

13.2 Select an output folder

Locate “Output Folder” and click the button marked “…” on the right side. Choose an output folder where the firmware will be created. All firmware will be installed in a subfolder automatically created by the installer. Note: you do not need to create a subfolder by yourself.



Input Firmware: C:\Documents and Setting … gaA_ddr_test.4ff

Output Folder: C:\Documents and Setting

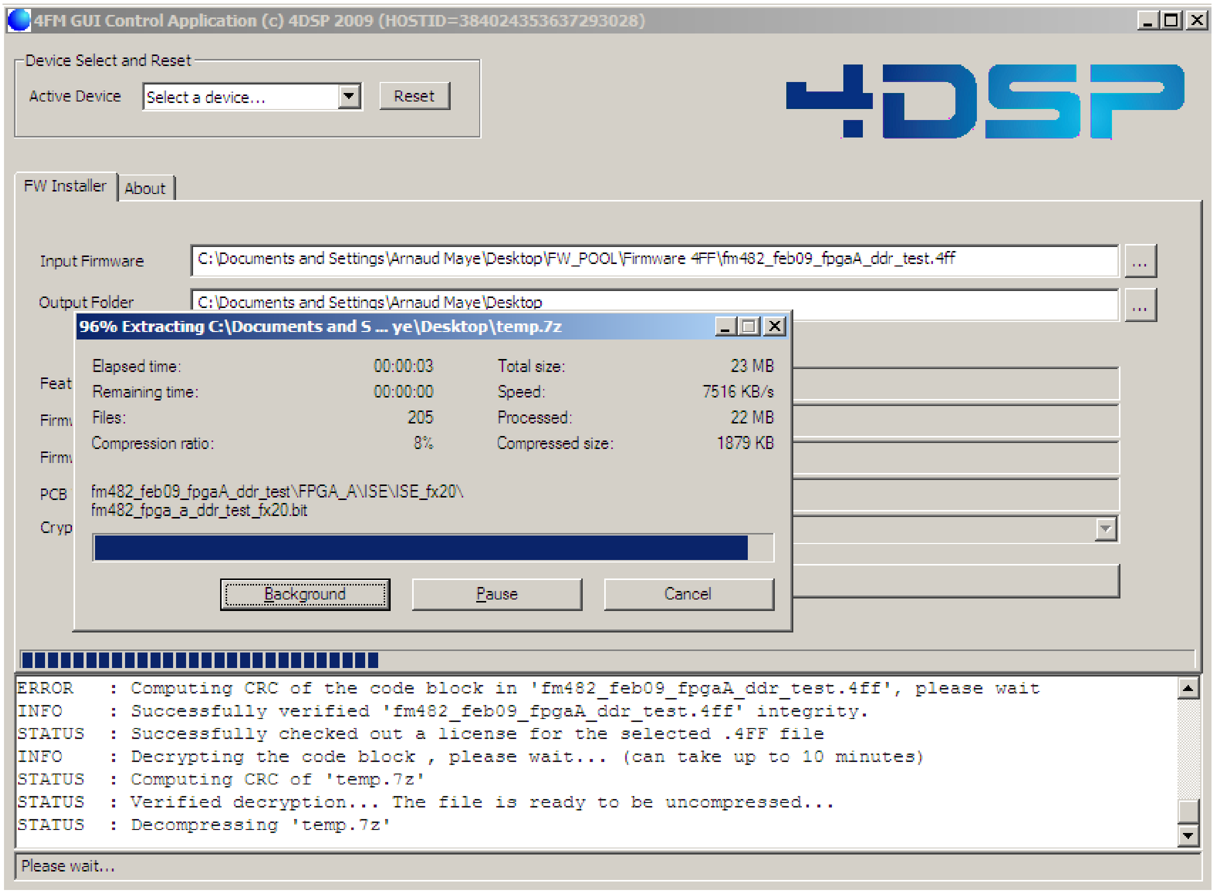
```
                Welcome to the 4DSP's 4FM GUI Control Application - (c) 2009 4DSP
                -----------------------------------------------------------------

INFO    : Your machine hostid is 384024353637293028
STATUS  : Successfully checked out a Firmware Package Decryptor license
STATUS  : Successfully checked out a Firmware Package Decryptor license
STATUS  : Successfully checked out a Firmware Package Decryptor license
```

Figure 66: Browsing to an output folder

13.3 Install the firmware

After selecting an input and an output file, launch the installation process by pressing the “Install The Firmware” button.

The progress is updated in real time in the debug area. After a while, the decompression starts and the status display shown in Figure 67 can be seen:


4FM GUI Control Application (c) 4DSP 2009 (HOSTID=384024353637293028)
Device Select and Reset
Active Device
Select a device...
Reset
FW Installer
About
96% Extracting C:\Documents and S ... ye\Desktop\temp.7z
Background
Pause
Cancel
Please wait...

Input Firmware: C:\Documents and Settings\Arnaud Maye\Desktop\FW_POOL\Firmware 4FF\fm482_feb09_fpgaA_ddr_test.4ff
Output Folder: C:\Documents and Settings\Arnaud Maye\Desktop

Elapsed time:	00:00:03	Total size:	23 MB
Remaining time:	00:00:00	Speed:	7516 KB/s
Files:	205	Processed:	22 MB
Compression ratio:	8%	Compressed size:	1879 KB

fm482_feb09_fpgaA_ddr_test\FPGA_A\ISE\ISE_fx20\fm482_fpga_a_ddr_test_fx20.bit

```
ERROR   : Computing CRC of the code block in 'fm482_feb09_fpgaA_ddr_test.4ff', please wait
INFO    : Successfully verified 'fm482_feb09_fpgaA_ddr_test.4ff' integrity.
STATUS  : Successfully checked out a license for the selected .4FF file
INFO    : Decrypting the code block , please wait... (can take up to 10 minutes)
STATUS  : Computing CRC of 'temp.7z'
STATUS  : Verified decryption... The file is ready to be uncompressed...
STATUS  : Decompressing 'temp.7z'
```

Figure 67: The firmware installer decompresses the data

13.4 In case of trouble

If the installation is not successful for some reason, please pay attention to the event log in the main debug area. Make sure your license file is installed properly.

For example, if you are trying to install an FM482 firmware, then you need to have an FM482 line in your license file.

The license should resemble the one below, but the SIGN, KEY, and other values will be unique to your license code. In order to install a firmware, the license line `NOTICE=` field should be set to `SOURCE`, and the `KEY=` field should have a 64-digit key.

```
PRODUCT=FM482 VERSION=2009.4 SN=12345678 CUSTNR=0 DELIVERYNR=0 NOTICE=SOURCE
KEY=123460937B26B5765426AD4CDF79E198863EBC51863EF65A7EA6F75B4D962151 EXPIRY=NEVER
HOSTID=ANY SIGN="AC3I 6PAA SVQJ WZY3 BBYY 9WRS XT5F SBKB 4CGR ZPKX P24J NQ25 8TT7
PNYE SPAS BM6Q 52TF NXRN AEES 5Y53 NREB GUAS X32D Q35H HWAG 96IR KEAR M95U 9SAC
22YI 7NUA 6E8K 5GSR RJ28 G6EJ C3RW IT65 JUEB I7JV H2QW G4WQ MFKE H57R IPFT BKII
FBKJ S2DD 2WGD 7WUE CH6Q MFIB TJE5 9PHA"
```

A successful installation will generate a log of events such as those seen in the following example:

```
INFO    : Processing firmware package 'fm482_feb09_fpgaA_ddr_test.4ff', please
wait..
STATUS  : Reading from file 'fm482_feb09_fpgaA_ddr_test.4ff'
STATUS  : Closing 'C:\Documents and Settings\Arnaud Maye\Desktop\FW_POOL\Firmware
4FF\fm482_feb09_fpgaA_ddr_test.4ff'
STATUS  : Computing CRC of file 'fm482_feb09_fpgaA_ddr_test.4ff', please wait
INFO    : Successfully verified 'fm482_feb09_fpgaA_ddr_test.4ff' integrity.
STATUS  : DeObfsucating 'fm482_feb09_fpgaA_ddr_test.4ff'
ERROR   : Computing CRC of the code block in 'fm482_feb09_fpgaA_ddr_test.4ff',
please wait
INFO    : Successfully verified 'fm482_feb09_fpgaA_ddr_test.4ff' integrity.
STATUS  : Successfully checked out a license for the selected .4FF file
INFO    : Decrypting the code block , please wait... (can take up to 10 minutes)
STATUS  : Computing CRC of 'temp.7z'
STATUS  : Verified decryption... The file is ready to be uncompressed...
STATUS  : Decompressing 'temp.7z'
STATUS  : The firmware is ready to be used
```

14 Annex 2 – License File details

14.1 Introduction

All development kits from version 2 and up require a license for each software component in order to run. When the core graphical user interface tries to load a plug-in, the plug-in will first check to see that the information in the license line matches the license code stored in the license file. If the correct license code cannot be found, the plug-in will not load.

The same mechanism is used to install the firmware packages. If you do not have a license for a specific firmware package, you cannot install it. The license code is specific to the firmware.

For each different firmware package, a “DEMO” license does not allow you to install a firmware package. However a “SOURCES” license allows a firmware package to be installed.

A license line can be either floating (can be used on any computer) or locked to a specific computer.

4DSP’s license policy allows customers to obtain either floating or locked license files. Please contact the 4DSP sales department if the license you have received does not suit your requirement.

If you have received a development kit on loan or on rent, then it is likely that your license has an expiration date. The plug-ins will refuse to load after the expiration date has passed, and you must contact 4DSP support to extend the evaluation period.

14.2 License format

This is a sample license. Any modification to the license file will render the license invalid. You are required to have one of these lines per product.

```
PRODUCT=FM482 VERSION=2009.4 SN=12345678 CUSTNR=0 DELIVERYNR=0 NOTICE=SOURCE
KEY=12345678 EXPIRY=NEVER HOSTID=ANY SIGN="AC3I … 9PHA"
```

The NOTICE, HOSTID, and EXPIRY fields contain codes indicating the type of license you have received. At the time of writing, 4DSP is issuing the following types of license codes:

14.2.1 The NOTICE= field

- DEMO. This is a demo license file. These licenses are generally locked to a computer hardware ID and have an expiration date. A DEMO license does not support the installation of any firmware. However, you are allowed to use all the installed recovery files without any restriction.
- SOURCE. This license allows the installation and decryption firmware package files. These licenses are typically sold on a floating basis and have no expiration date.

14.2.2 The HOSTID= field

- ANY. In this case the license can be used on any computer. However, please consult the EULA to ensure you are allowed to do that, especially if you have an agreement for a maximum number of seats with 4DSP.
- {ID}. If a code is there instead of ANY, then your license will only operate on the machine with the same ID code. The computer ID is displayed in the main graphical user interface caption (HOSTID=nnnnnn).

14.2.3 The EXPIRY= field

- NEVER. This license code does not expire.
- {DATE}. This license operates until {DATE} is reached.

15 Annex 3 - Compiling the 4FM Core Example

15.1 SDK Integration in Visual Studio 2012

In order to compile an application using this SDK, you must integrate the SDK in Visual Studio.

Please refer to Chapter 2.3 Software requirements (Visual Studio 2012) for more information on how to integrate 4FM SDK in Visual Studio.

15.2 Compiling in Visual Studio

The 4FM.vcxproj Visual Studio project file can be opened and compiled like any other application source.

16 Annex 4 - Firmware .4FF file

The firmware packages are shipped as 4DSP Firmware Files with the .4ff file extension. These files can be seen as an archive. They are heavily encrypted and signed. Only a user with a license for this specific firmware may “decrypt” .4ff files.

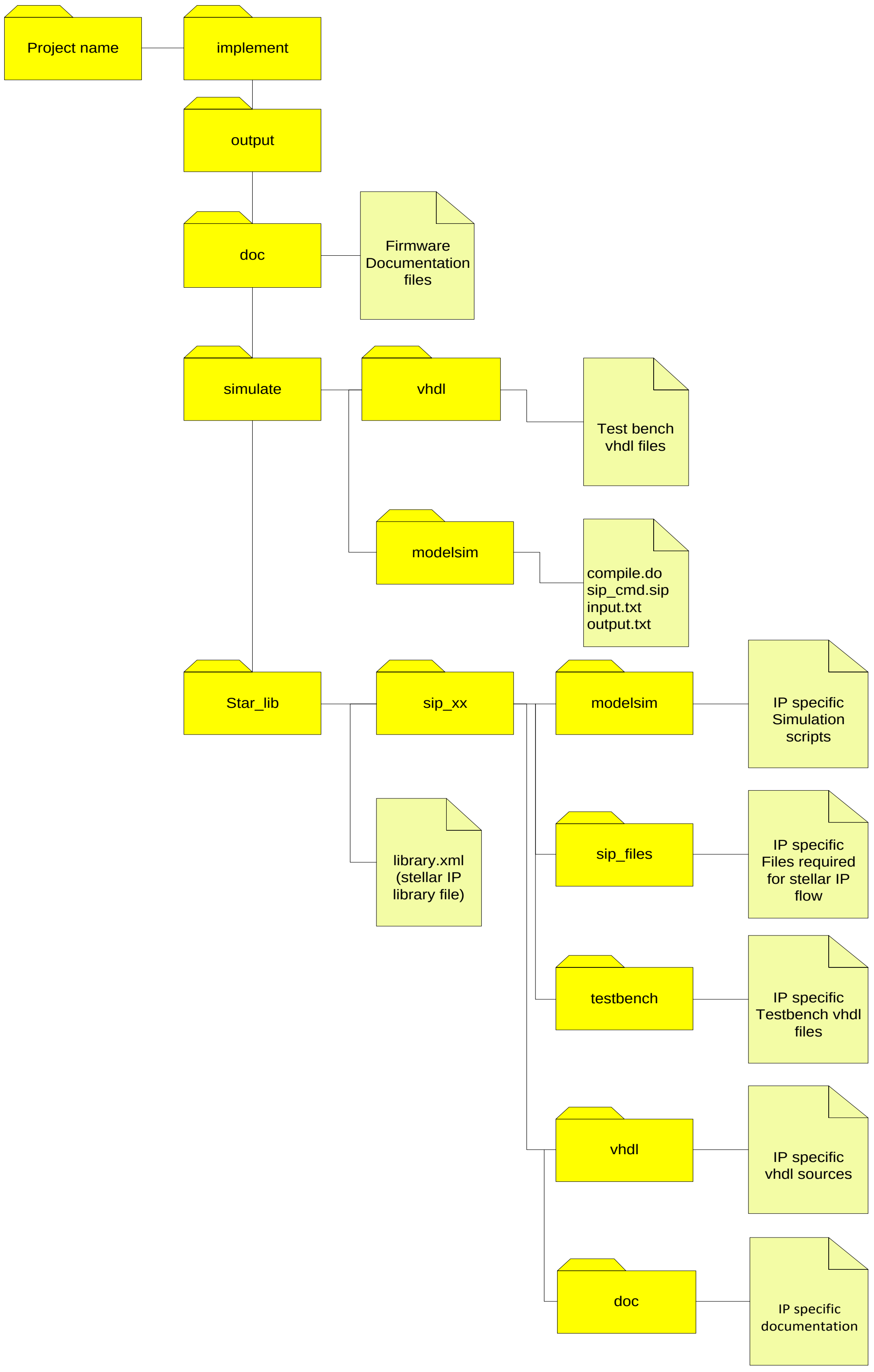
Please refer to the Annex 1 – Manual Firmware Installation chapter to understand how to decrypt a firmware using 4DSP tools.

16.1 4FF content

A decrypted firmware is a folder with several subfolders inside:

- A “doc” folder containing the firmware documentation
- An “implement” folder containing the actual StellarIP source files for the firmware
- An “output” folder (empty or not). This folder will be populated when the firmware is processed by StellarIP.
- A “simulate” firmware containing the simulation files for the firmware.
- A “star_lib” folder containing the firmware IP cores (star) provided by 4DSP for this specific firmware
- A release_notes.txt file

16.2 4FF content (Hierarchical view)


Project name
implement
output
doc
Firmware Documentation files
simulate
vhdl
Test bench vhdl files
modelsim
compile.do
sip_cmd.sip
input.txt
output.txt
Star_lib
sip_xx
modelsim
IP specific Simulation scripts
library.xml (stellar IP library file)
sip_files
IP specific Files required for stellar IP flow
testbench
IP specific Testbench vhdl files
vhdl
IP specific vhdl sources
doc
IP specific documentation

17 Annex 5 – fmc645 test peripherals log

```
Welcome to the 4DSP FMC645 Peripherals test application!
-------------------------------------------------------------------------
Opening communication ports (Only UART): Done



-------------------------------------------------------------------------
Checking StellarIP firmware attributes
Firmware ID            : 196
Firmware SW Build Code : 1332412368
Firmware FW Version    : 1
Firmware FW Build Code : 0
Number of FW stars     : 5
sip_ub_fmc645 address  : 0x0

StellarIP attribute Test results : Test Succeeded!



-------------------------------------------------------------------------
Testing I2C Bus on FMC645

I2C Test results : Test Succeeded!



-------------------------------------------------------------------------
Now that I2C is verified, let’s check board diagnostics, please wait...



Checking Board Diagnostics:
Temp ADT       OK (38C)
Temp TMSC6455  OK (38C)
3V3  on FMC645 OK (3.2V)
1V8  on FMC645 OK (1.8V)
1V25 on FMC645 OK (1.2V)
1V25 on FMC645 OK (1.2V)
2V5  on FMC645 OK (2.5V)
3V3  on FMC645 OK (3.2V)
0V9  on FMC645 OK (0.9V)
Diagnostics Test results : Test Succeeded!



-------------------------------------------------------------------------
Now that I2C is verified, let’s check 24LC02B chip, please wait...
sip_fmc645_test address  : 0xc0009000
Writing 256 bytes to the EEPROM
```

```
..............................................................................................
..............................................................................................
..........................................................
Verifying 256 bytes in the EEPROM
..............................................................................................
..............................................................................................
..........................................................
EEPROM verification finished with (0) error(s)
24LC02B Test results : Test Succeeded!


------------------------------------------------------------------------
Testing I2C communication from host to the TMS320C6455 device...
 Uploading the DSP software task to the DSP, please wait...(100%)
I2C Test results :
Test Succeeded!


------------------------------------------------------------------------
Measuring frequencies
Stellar IP Clock : 100.00 MHz
DSP CLKIN1       :  40.00 MHz
DSP CLKIN2       :  25.00 MHz
DSP CLK 50MHZ    :  50.00 MHz
DSP CLK 125MHZ   : 125.00 MHz
DSP SYSCLK4      :  75.00 MHz (dsp clock = 1200.00 MHz)


------------------------------------------------------------------------
Testing GPIO on the FMC645...
 DSP Software about to be uploaded to DSP is 'fmc645_gpio_ack.hex'
 Argument1 (8 Bit) about to be passed to DSP is '0x00'
 Argument2 (8 Bit) about to be passed to DSP is '0x00'
 Uploading the DSP software task to the DSP, please wait...(100%)
 Waiting DSP to advertise himself as ready... Done!
 Sending first argument to DSP software... Done!
 Sending second argument to DSP software... Done!
 Telling DSP to start executing the DSP software... Done!
 Wait DSP software to indicate end of execution... Done!
 Reading error code register 1 from the DSP software... Done!
 Reading error code register 2 from the DSP software... Done!
 Status code 1 received from DSP software '0x0'
 Status code 2 received from DSP software '0x0'
GPIO Test results: Status = Test Succeeded!


------------------------------------------------------------------------
Testing MCBSPs on the FMC645...
 DSP Software about to be uploaded to DSP is 'fmc645_mcbsp_ack.hex'
 Argument1 (8 Bit) about to be passed to DSP is '0x00'
 Argument2 (8 Bit) about to be passed to DSP is '0x00'
 Uploading the DSP software task to the DSP, please wait...(100%)
 Waiting DSP to advertise himself as ready... Done!
 Sending first argument to DSP software... Done!
 Sending second argument to DSP software... Done!
 Telling DSP to start executing the DSP software... Done!
 Wait DSP software to indicate end of execution... Done!
 Reading error code register 1 from the DSP software... Done!
 Reading error code register 2 from the DSP software... Done!
 Status code 1 received from DSP software '0x0'
 Status code 2 received from DSP software '0x0'
MCBSP Test results: Status = Test Succeeded!



------------------------------------------------------------------------
Testing SRIO on the FMC645...
 DSP Software about to be uploaded to DSP is 'fmc645_srio_ack.hex'
 Argument1 (8 Bit) about to be passed to DSP is '0x00'
 Argument2 (8 Bit) about to be passed to DSP is '0x00'
 Uploading the DSP software task to the DSP, please wait...(100%)
 Waiting DSP to advertise himself as ready... Done!
 Sending first argument to DSP software... Done!
 Sending second argument to DSP software... Done!
 Telling DSP to start executing the DSP software... Done!
 Wait DSP software to indicate end of execution... Done!
 Reading error code register 1 from the DSP software... Done!
 Reading error code register 2 from the DSP software... Done!
 Status code 1 received from DSP software '0x0'
 Status code 2 received from DSP software '0x0'
SRIO Test results: Status = Test Succeeded!



------------------------------------------------------------------------
Testing EMIFA on the FMC645...
 DSP Software about to be uploaded to DSP is 'fmc645_emifa_ack.hex'
 Argument1 (8 Bit) about to be passed to DSP is '0x00'
 Argument2 (8 Bit) about to be passed to DSP is '0x00'
 Uploading the DSP software task to the DSP, please wait...(100%)
 Waiting DSP to advertise himself as ready... Done!
 Sending first argument to DSP software... Done!
 Sending second argument to DSP software... Done!
 Telling DSP to start executing the DSP software... Done!
 Wait DSP software to indicate end of execution... Done!
 Reading error code register 1 from the DSP software... Done!
 Reading error code register 2 from the DSP software... Done!
 Status code 1 received from DSP software '0x4'
 Status code 2 received from DSP software '0x4'
```

```
EMIFA Test results: Status = Test Succeeded!


------------------------------------------------------------------------
Testing DDR2 Memory on the FMC645 (512MBytes)...
 DSP Software about to be uploaded to DSP is 'fmc645_ddr2_ack.hex'
 Argument1 (8 Bit) about to be passed to DSP is '0x40'
 Argument2 (8 Bit) about to be passed to DSP is '0x00'
 Uploading the DSP software task to the DSP, please wait...(100%)
 Waiting DSP to advertise himself as ready... Done!
 Sending first argument to DSP software... Done!
 Sending second argument to DSP software... Done!
 Telling DSP to start executing the DSP software... Done!
 Wait DSP software to indicate end of execution... Done!
 Reading error code register 1 from the DSP software... Done!
 Reading error code register 2 from the DSP software... Done!
 Status code 1 received from DSP software '0x3'
 Status code 2 received from DSP software '0x0'
DDR2 Test results: Status = Test Succeeded!


------------------------------------------------------------------------
Now that DSP is hot after DDR2 test, let’s check board diagnostics, please wait...


Checking Board Diagnostics:
Temp ADT       OK (39C)
Temp TMSC6455  OK (39C)
3V3  on FMC645 OK (3.2V)
1V8  on FMC645 OK (1.8V)
1V25 on FMC645 OK (1.2V)
1V25 on FMC645 OK (1.2V)
2V5  on FMC645 OK (2.5V)
3V3  on FMC645 OK (3.2V)
0V9  on FMC645 OK (0.9V)
Diagnostics Test results : Test Succeeded!


------------------------------------------------------------------------
Testing EMAC on the FMC645...

Before to continue further, please connect an ethernet
cable between ML605 and your host...
Configure your HOST TCPIP/V4 Settings as the following:
IP ADDRESS :    192.168.254.2 (Host address)
GATEWAY    :    192.168.254.1 (DSP address)
NETMASK    :    255.255.255.0
```

```
As soon this is done please press RETURN to continue

 Uploading the DSP software task to the DSP, please wait...(100%)
Waiting 25 seconds to making sure the link is up...
%%%%%%%%%%%%%%%%%%%%%%%%%
Trying to connect to the DSP using TCP/IP: Done
Writing 10 registers to the DSP address space (Via TCPIP), please wait...
Reading 10 registers back from the DSP address space (Via TCPIP), please wait...
0x00000000 : 0x00000000
0x00000001 : 0x00000001
0x00000002 : 0x00000002
0x00000003 : 0x00000003
0x00000004 : 0x00000004
0x00000005 : 0x00000005
0x00000006 : 0x00000006
0x00000007 : 0x00000007
0x00000008 : 0x00000008
0x00000009 : 0x00000009
Comparing 10 registers read back from the Microblaze address space, please wait...
0x00000000 : 0x00000000 (OK)
0x00000001 : 0x00000001 (OK)
0x00000002 : 0x00000002 (OK)
0x00000003 : 0x00000003 (OK)
0x00000004 : 0x00000004 (OK)
0x00000005 : 0x00000005 (OK)
0x00000006 : 0x00000006 (OK)
0x00000007 : 0x00000007 (OK)
0x00000008 : 0x00000008 (OK)
0x00000009 : 0x00000009 (OK)
Ethernet Test results :
Test Succeeded!
You should be able to ping 192.168.254.1 (the DSP)



------------------------------------------------------------------------
Finished, let’s check board diagnostics again, please wait...


Checking Board Diagnostics:
Temp ADT        OK (40C)
Temp TMSC6455   OK (40C)
3V3  on FMC645 OK (3.2V)
1V8  on FMC645 OK (1.8V)
1V25 on FMC645 OK (1.2V)
1V25 on FMC645 OK (1.2V)
2V5  on FMC645 OK (2.5V)
```

```
3V3  on FMC645 OK (3.2V)
0V9  on FMC645 OK (0.9V)
Diagnostics Test results : Test Succeeded!

Finished
```

18 Annex 6 – fmc667 test peripherals log

```
Welcome to the 4DSP FMC667 Peripherals test application!
```

TO BE ADDED