SiFive E31/E51 Core Complex FPGA Eval Kit User Guide v2p0

SiFive E31/E51 Core Complex FPGA Eval Kit User Guide

Proprietary Notice

Release Information

Version	Date	Changes
v2p0	Feb 2, 2018	Updated to match v2p0 of the Evaluation MCS: • FPGA Eval includes ITIM • DTIM size increased to 64kiB • Number of HWBP increased to 8 • Added User Mode Support • Updated various links
v1p0	May 4, 2017	First release

Contents

List of Figures

Chapter 1

Introduction

1.1 About this Document

This document gives necessary information for a user of the SiFive E31/E51 Core Complex FPGA Eval Kit. To learn more about the functionality of the E31 and E51 Core Complex, please read the appropriate Core Complex Manual.

This guide will help you download and flash the E31/E51 Core Complex FPGA Eval Kit image to a FPGA development board. It will help you install software tools to allow you to write, upload, and debug code on the Eval Kit. It also contains information about what is contained in the MCS file for the E31/E51 Core Complex FPGA Eval Kit.

1.2 About this Release

This Eval Kit allows you to prototype and benchmark your target RISC-V software without modifying, integrating, or synthesizing any Verilog code.

This release is intended for evaluation purposes only.

1.3 Evaluation Version Limitations

Version v2p0 of the E31/E51 Core Complex FPGA Eval Kit has the following limitations compared with the fully functional Core Complex IP:

- DTIM is limited in size to 64kB.
- Peripheral Bus, System Bus, and Front Bus are not exported for additional user connections. The evaluation can utilize the peripherals included on the FPGA.
- Not all Local and Global interrupts are exported at the top level.

To target a different FPGA platform or perform synthesis or simulation, you may obtain an Evaluation Version of the Core Complex IP RTL from `sifive.com`.

Chapter 2

Required Hardware

The E31/E51 Core Complex FPGA Eval Kit requires the following hardware:

2.1 Xilinx Artix-7 35T Arty FPGA Evaluation Kit

The Arty is a Xilinx FPGA development board for makers and hobbyists. It can be purchased from Digilent, Avnet, or Digi-Key.

`http://www.xilinx.com/products/boards-and-kits/arty.html`
`https://store.digilentinc.com/arty-board-artix-7-fpga-development-board-for-makers-and-hobbyis`

2.2 USB A to Micro-B Cable

Any standard USB Type A Male to Micro-B Male cable can be used to interface with the Arty. Note that the Arty kit does not include one.

`http://store.digilentinc.com/usb-a-to-micro-b-cable/`

2.3 Olimex ARM-USB-TINY-H Debugger

The Olimex ARM-USB-TINY-H is a hardware JTAG debugger. The Freedom E310 Arty FPGA Dev Kit has a standard JTAG debugging interface, and the tools included with the Freedom E SDK have been tested using the Olimex ARM-USB-TINY-H. It can be purchased from Olimex or Digi-Key.

`https://www.olimex.com/Products/ARM/JTAG/ARM-USB-TINY-H/`
`http://www.digikey.com/product-detail/en/olimex-ltd/ARM-USB-TINY-H/1188-1013-ND/3471388`

2.4 USB A to B Cable

Any standard USB Type A Male to B Male cable can be used to interface to the Olimex ARM-USB-TINY-H Debugger. Note that the package does not include one. These are available from a variety of sources, including Digi-Key.

`http://www.digikey.com/product-detail/en/assmann-wsw-components/AK672-2-1/AE1462-ND/930247`

2.5 Male-To-Female Jumper Cables (10)

The connection between the Olimex ARM-USB-TINY-H and Freedom E310 Arty FPGA Dev Kit requires 10 connections. These can be made with Male-to-Female jumper cables. These cables are available from Adafruit in convenient rip-apart ribbon cables:

`https://www.adafruit.com/products/826`

Chapter 3

Board Setup

3.1 Connecting the USB Interface

Connect the USB Type A to Micro-B cable between the USB-JTAG port (J10) of the Arty and the host machine. This provides UART console access to the E31/E51 Core Complex FPGA Eval Kit as well as a 5V power source for the board. This is also the interface by which the FPGA fabric will be programmed.

3.2 Connecting the Debugger

The debugger is essential for downloading and debugging code with your SDK. The software will be downloaded to SPI Flash, so it will be retained. Without the debugger you can only flash the FPGA image and run the included demo program, you cannot change the software which executes.

Connect the Olimex ARM-USB-TINY-H with the USB Type A to B cable to the host machine. Then connect the Olimex ARM-USB-TINY-H debugger to PMOD header JD using the 10 jumper cables. The pinout is as shown in Figure **??**. Note that the Olimex ARM-USB-TINY-H and the PMOD header on the Arty Board have different numbering schemes. Figures 3.2 and **??** clarify the different pinouts for the two connectors.

Figure 3.4 shows what the board looks like with all the debug connections in place.

Note: It is important to connect to PMOD header JD (not JA, JB, or JC). JD was selected over the other PMOD headers to avoid damage to the Arty board in the event of mismatched connections.

Signal Name	ARM-USB-TINY-H Pin Number	Suggested Jumper Color	Freedom E310 Arty Dev Kit JD Pin Number
VREF	1	red	12
VREF	2	brown	6 (“VCC”)
nTRST	3	orange	2
TDI	5	yellow	7
TMS	7	green	8
TCK	9	blue	3
TDO	13	purple	1
GND	14	black	5 (“GND”)
nRST	15	grey	9
GND	16	white	11

Figure 3.1: Debugging Connections between Olimex ARM-USB-TINY-H and Arty Board’s PMOD header JD

	1 : VREF (red)	2 : VREF (brown)
	3 : nTRST (orange)	4
	5 : TDI (yellow)	6
	7 : TMS (green)	8
NOTCH	9 : TCK (blue)	10
NOTCH	11	12
	13 : TDO (purple)	14 : GND (black)
	15 : nRST (grey)	16 : GND (white)
	17	18
	19	20
	LED	

Figure 3.2: Debug Connections To the Olimex ARM-USB-TINY-H

square pad	1 : TDO (purple)	7 : TDI (yellow)
	2 : nTRST (orange)	8 : TMS (green)
	3 : TCK (blue)	9 : nRST (grey)
	4	10
“GND”	5 : GND (black)	11 : GND (white)
“VCC”	6 : VREF (brown)	12 : VREF (red)

Figure 3.3: Debug Connections to the Arty Board JD PMOD Header

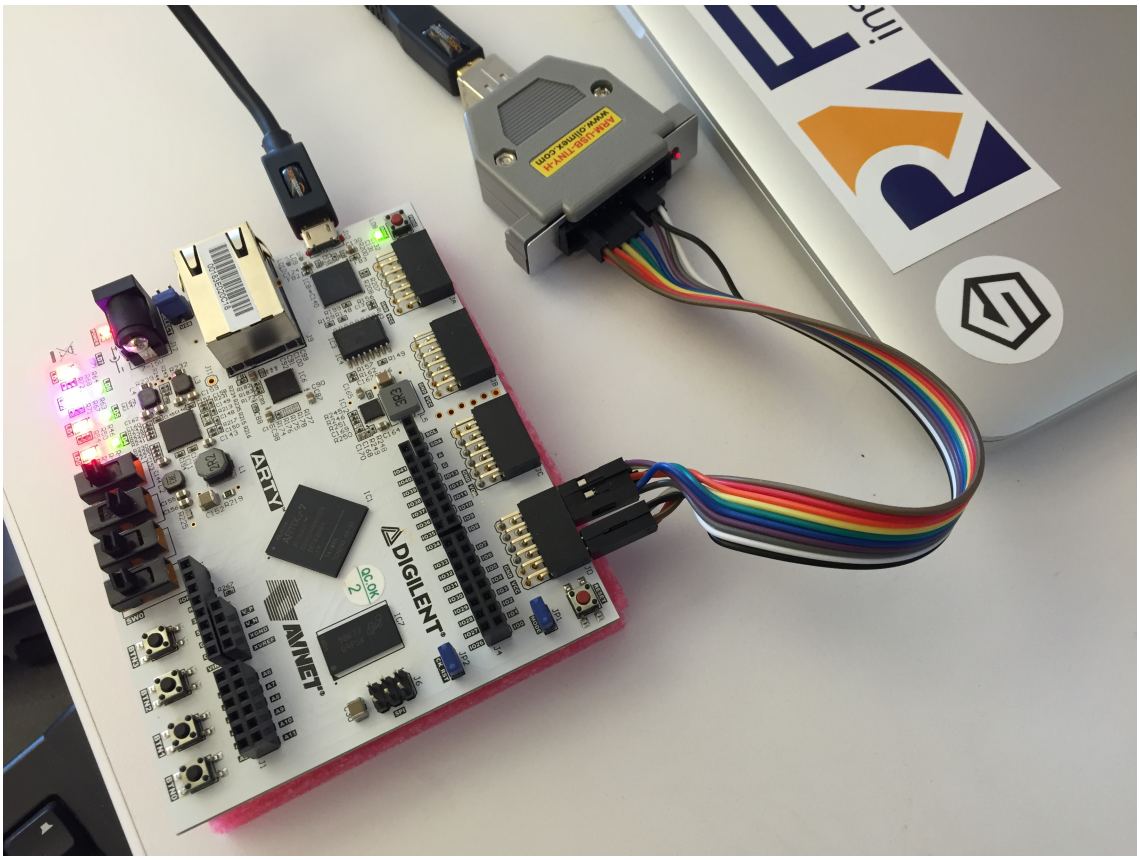

Figure 3.4: Photo of the Arty Board showing USB and Debug Connections

Chapter 4

FPGA Flash Programming File

The Xilinx Artix-7 XC7A35T FPGA configures on power-on from an on-board quad SPI flash chip.

To obtain the E31/E51 Core Complex FPGA Eval Kit SPI flash programming file, first claim it from the following URL:

https://dev.sifive.com/risc-v-core-ip/evaluate/fpga/

You will be asked to sign up for a free account or log into an existing one. After you are logged in, the file will appear in your personal dashboard under the name SiFive E31/E51 Core Complex FPGA Eval Kit. Click on it to view details, then click “Download Files” to download the release tarball. Unpack the tarball to extract the MCS file to use to program your FPGA board.

The Xilinx Vivado Design Suite is used for flash programming. Both the Vivado Lab Edition and WebPACK Edition 2016.1 support Artix-7 devices free of charge.

To program the SPI flash with Vivado:

1. Launch Vivado
2. Open Hardware Manager
3. Open target board
4. Right click on the FPGA device and select ”Add Configuration Memory Device”
5. Select the following SPI flash paramters:

Part	n25q128-3.3v
Manufacturer	Micron
Family	n25q
Type	spi
Density	128
Width	x1_x2_x4

6. Click OK to “Do you want to program the configuration memory device now?”
7. Add sifive_coreip_E31_FPGA_Evaluation_v2p0_0.mcs (i.e. the MCS file which you downloaded and unpacked).

8. Select OK
9. Once the programming completes in Vivado, press the “PROG” Button on the Arty Board to load the image into the FPGA.

Chapter 5

Boot and Run

The MCS file includes a simple demo program. This program is loaded to the SPI Flash along with the FPGA image. With Switch 0 set to the "Off" position (towards the edge of the board), on reset the Core Complex will execute a simple demo program. This program prints a message over the UART and uses the PWM peripheral to change RGB LED 1. This program will be overwritten in the SPI Flash when you program new software into the board with the SDK, but the FPGA image will not be modified. Source for this program is included in the SDK.

Using a terminal emulator such as GNU screen on Linux, open a console connection from the host computer to the E31/E51 Core Complex FPGA Eval Kit.

Set the following parameters:

Speed	115200
Parity	None
Data bits	8
Stop bits	1
Hardware Flow	None

For example, on Linux using GNU Screen:

```
sudo screen /dev/ttyUSB1 115200
```

You can use `Ctrl-a k` to "kill" (exit) the running screen session.

Depending on your setup, you may need additional drivers or permissions to communicate over the USB port.

If you are running on Ubuntu-style Linux, the below is an example of steps you may need to follow to access your dev kit without `sudo` permissions:

1. With your board's debug interface connected, make sure your device shows up with the `lsusb` command:

    ```
    > lsusb
    ...
    ```

```
Bus XXX Device XXX: ID 0403:6010 Future Technology Devices
International, Ltd FT2232C Dual USB-UART/FIFO IC
```

2. Set the udev rules to allow the device to be accessed by the `plugdev` group:

```
> sudo vi /etc/udev/rules.d/99-openocd.rules
```

Add the following lines and save the file (if they are not already there):

```
# These are for the HiFive1 Board
SUBSYSTEM=="usb", ATTR{idVendor}=="0403",
  ATTR{idProduct}=="6010", MODE="664", GROUP="plugdev"

SUBSYSTEM=="tty", ATTRS{idVendor}=="0403",
  ATTRS{idProduct}=="6010", MODE="664", GROUP="plugdev"

# These are for the Olimex Debugger for use with E310 Arty Dev Kit

SUBSYSTEM=="usb", ATTR{idVendor}=="15ba",
  ATTR{idProduct}=="002a", MODE="664", GROUP="plugdev"

SUBSYSTEM=="tty", ATTRS{idVendor}=="15ba",
  ATTRS{idProduct}=="002a", MODE="664", GROUP="plugdev"
```

3. See if your board shows up as a serial device belonging to the `plugdev` group:

```
> ls /dev/ttyUSB*
/dev/ttyUSB0 /dev/ttyUSB1
```

(If you have other serial devices or multiple boards attached, you may have more devices listed). For serial communication with the UART, you will always want to select the higher number of the pair, in this example `/dev/ttyUSB1`.

```
> ls -l /dev/ttyUSB1
crw-rw-r-- 1 root plugdev 188, 1 Nov 28 12:53 /dev/ttyUSB1
```

4. Add yourself to the `plugdev` group. You can use the `whoami` command to determine your user name.

```
> whoami your_user_name > sudo usermod -a -G plugdev your_user_name
```

5. Log out and log back in, then check that you're now a member of the plugdev group:

```
> groups
...  plugdev ...
```

Now you should be able to access the serial (UART) and debug interface without sudo permissions.

5.1 Terminal Log

If you have your serial setup correctly, this is what you will see on your terminal (you may need to hit the 'Reset' button to restart the program):

```
               SIFIVE, INC.

         55555555555555555555555
        5555                   5555
       5555                     5555
      5555                       5555
     5555       55555555555555555555
    5555       555555555555555555555
   5555                             5555
  5555                               5555
 5555                                 5555
555555555555555555555555555           55555
 55555           555555555           55555
   55555           55555           55555
     55555           5           55555
       55555                   55555
         55555               55555
           55555           55555
             55555       55555
               55555   55555
                 555555555
                   55555
                     5


Welcome to the E31 Core IP FPGA Evaluation Kit!
```

Chapter 6

Software Development Flow

The E31/E51 Core Complex FPGA Eval Kit's reset vector is set using Switch 0 on the board. When the switch is "Off" (set towards the edge of the board), the reset vector is set to 0x40400000, which is mapped to the external SPI Flash on the board. When the switch is "On" (set away from the edge of the board), the reset vector is set to 0x00000000. This will cause the core to simply wait for the debugger to load a program. You can change the program which the Eval Kit runs by using the debug/programming interface to flash a new compiled program into the DTIM or SPI Flash.

SiFive provides both precompiled binaries as well as source for the software development toolchain. In addition, SiFive supports the Eclipse-Based Freedom Studio IDE for software development.

6.1 Software Development Using Freedom Studio IDE

SiFive recommends software development for the E31/E51 Core Complex FPGA Eval Kit with the Eclipse-based Freedom Studio IDE. Freedom Studio is supported for Windows, macOS, and Linux. When using this method, the precompiled tools and drivers are automatically installed, you do not need to download or install it seperately to get tools and example code.

You can obtain Freedom Studio from the SiFive website:

`https://www.sifive.com/product/tools/`

More information on how to use it can be found in the Freedom Studio Manual:

`https://www.sifive.com/documentation/tools/freedom-studio-manual`

6.2 Software Development Using Command Line Tools

6.2.1 Obtaining the Freedom E SDK Toolchain

The Freedom E Software Development Kit provides everything required to compile, customize, and debug C, C++ and/or RISC-V assembly programs: GCC 7.1.0 cross-compilation toolchain, RISC-V enabled GDB and OpenOCD, etc. The SDK also includes example code projects. The toolchain can either be compiled from source in the SDK, or downloaded from SiFive's website as precompiled binary packages.

Obtaining the Freedom E SDK Precompiled Binaries

Precompiled binaries can be obtained from SiFive's website:

`https://www.sifive.com/product/tools/`

To get example projects without downloading all of the toolchain sources, you can do a shallow clone of the Freedom E SDK:

```
git clone https://github.com/sifive/freedom-e-sdk.git
```

Compiling the Freedom E SDK Tools from Source

This step is not necessary if you have downloaded the precompiled binaries as described in the previous section.

To clone the Freedom E SDK git repository:

```
git clone --recursive https://github.com/sifive/freedom-e-sdk.git
```

Install all the necessary packages described in the repository's `README.md` file.

To build the software toolchain:

```
cd freedom-e-sdk
make tools BOARD=coreplexip-e31-arty
```

To keep your software toolchain up to date with the upstream repository:

```
cd freedom-e-sdk
git pull origin master
git submodule update --init --recursive
make tools BOARD=coreplexip-e31-arty
```

6.2.2 Compiling Software Programs

To build a C program that will be loaded by the debugger/programmer into the SPI Flash, use the Freedom E SDK to compile. Examples are provided in the `software/` directory. To build the program:

```
cd freedom-e-sdk
make software PROGRAM=demo_gpio BOARD=coreplexip-e31-arty
```

To compile the Dhrystone benchmark instead:

```
cd freedom-e-sdk
make software PROGRAM=dhrystone BOARD=coreplexip-e31-arty
```

6.2.3 Uploading Software Programs

To upload the program to the SPI flash, connect the board's debug interface as described in Chapter 3. Then execute:

```
cd freedom-e-sdk
make upload PROGRAM=<your desired program> BOARD=coreplexip-e31-arty
```

6.2.4 Debugging Running Programs

To debug your program with GDB, connect your board and launch OpenOCD in one terminal window:

```
cd freedom-e-sdk
make run_openocd BOARD=coreplexip-e31-arty
```

In a second terminal window, launch the debugger, which will automatically connect to the running OpenOCD target:

```
cd freedom-e-sdk
make run_gdb PROGRAM=<your desired program> BOARD=coreplexip-e31-arty
```

This will automatically launch OpenOCD and GDB, connect to the board, and halt the currently running program. You can step through the running program with `stepi`, or load the new program using `load`. The usual suite of GDB commands are available to set breakpoints, examine and modify memory, continue execution, etc.

Chapter 7

E31/E51 Core Complex FPGA Eval Kit MCS Image Contents

Figure 7.1 shows a block diagram of the E31/E51 Core Complex FPGA Eval Kit. The evaluation kit includes an evaluation E31 or E51 Core Complex along with additional peripherals and I/Os to allow software prototyping.

7.1 E31/E51 Core Complex FPGA Eval Kit Memory Map

The FPGA design on the E31/E51 Core Complex FPGA Eval Kit has an evaluation version of the SiFive E31/E51 Core Complex, as well as peripheral devices which are not included with the Core Complex deliverable. These devices allow you to perform basic I/O to prototype and benchmark some basic applications.

The peripherals are connected within the FPGA fabric at memory locations which correspond to the Periphery Bus and System Bus as shown in Table 7.1. The Front Bus is not connected on the E31/E51 Core Complex FPGA Eval Kit.

7.2 E31/E51 Core Complex FPGA Eval Kit Clock and Reset

The E31/E51 Core Complex FPGA Eval Kit has a 100MHz input to the FPGA. This is used to derive the Core Complex's `io_coreClock` at 65 MHz, and the `clock` (peripheral clock) at 32.5 MHz. The `io_rtcToggle` is driven at approximately 32kHz.

The system reset driven by the Reset Button on the evaluation board is combined with the external debugger's `SRST_n` pin as a full system reset for the E31/E51 Core Complex FPGA Eval KitṪhis is combined with the `io_ndreset` to drive the `reset` input to the Core Complex.

The reset vector is set with Switch 0. Leave the switch in the "Off" position to execute from SPI Flash.

7.3 E31/E51 Core Complex FPGA Eval Kit Pinout

The peripherals perform I/O functionalities and are also used to demonstrate the use of Global Interrupts. The peripheral devices are connected to hardware on the Arty development board as described in Table 7.2. More information about the Peripheral Devices can be found in the SiFive

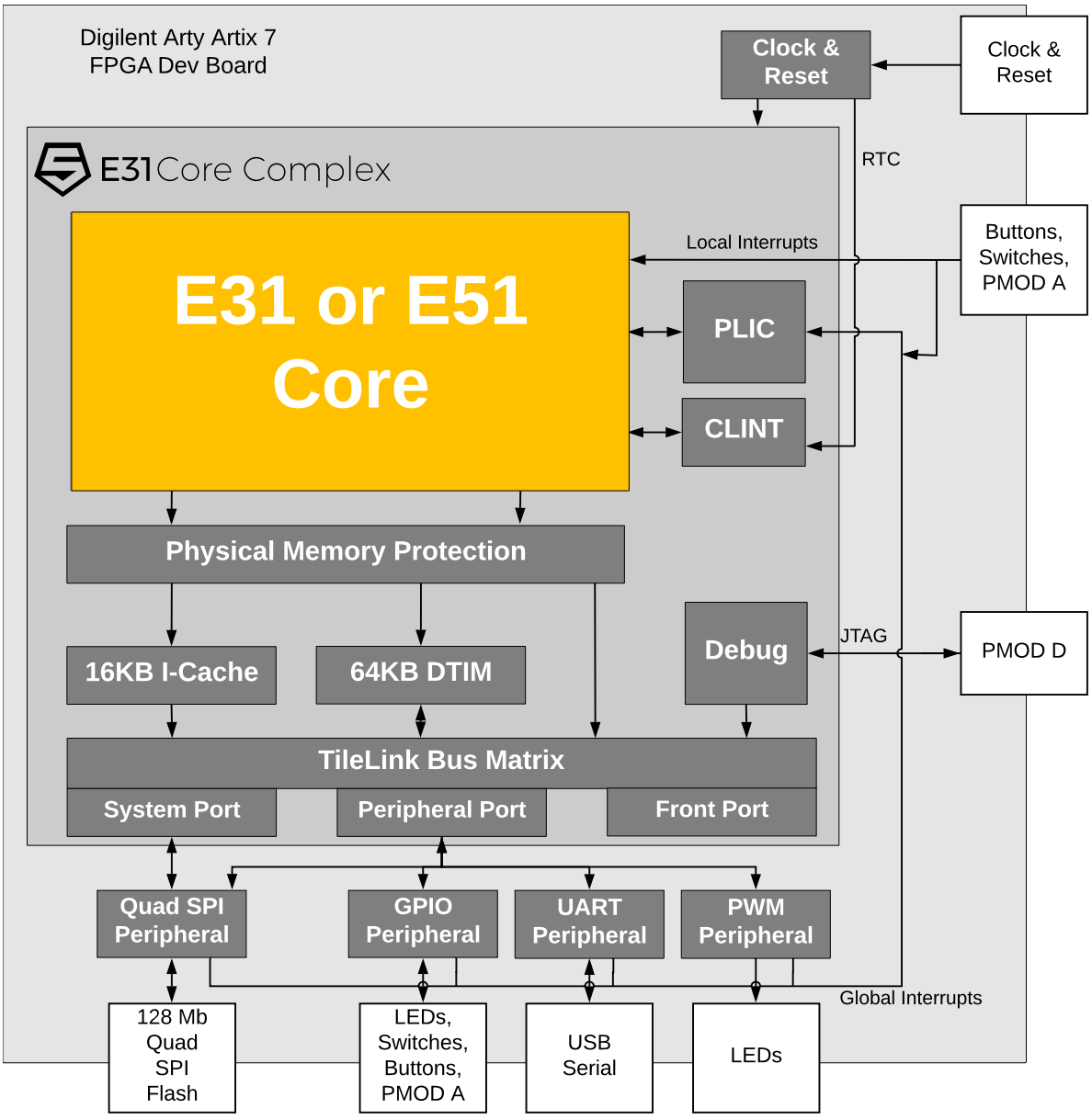

Figure 7.1: E31/E51 Core Complex FPGA Eval Kit Block Diagram

Freedom E300 Platform Reference Manual. Some inputs are wired directly as Global Interrupts, while others go through the GPIO peripheral.

In addition, some board I/Os are configured as Local Interrupt sources. The mapping between hardware on the E31/E51 Core Complex FPGA Eval Kit and Local Interrupt sources are provided in Table 7.3

Base	Top	Description	Notes
0x0000_0000	0x0000_00FF	*Reserved*	
0x0000_0100		Halt Notification	
0x0000_0104		Start Notification	
0x0000_0108		Resume Notification	
0x0000_010C		Exception Notification	Debug (4 KiB)
0x0000_0110	0x0000_02FF	*Reserved*	
0x0000_0300	0x0000_03FF	Debug RAM ($\leq$ .25 KiB)	
0x0000_0400	0x0000_07FF	Debug Flags ($\leq$ 1 KiB)	
0x0000_0800	0x0000_0FFF	Debug ROM ($\leq$ 2 KiB)	
0x0000_1000	0x01FF_FFFF	*Reserved*	
0x0200_0000	0x0200_FFFF	Core Complex-Local Interrupts (CLINT) ($\leq$64 KiB)	on core complex Devices (224 MiB)
0x0201_0000	0x07FF_FFFF	*Reserved*	
0x0800_0000	0x0800_1FFF	ITIM (8 KiB)	
0x0800_2000	0x0BFF_FFFF	*Reserved*	
0x0C00_0000	0x0FFF_FFFF	Platform-Level Interrupt Control (PLIC) (64 MiB)	
0x1001_0000	0x1FFF_FFFF	*Reserved*	
0x2000_0000	0x2000_0FFF	UART Peripheral	
0x2000_1000	0x2000_1FFF	*Reserved*	
0x2000_2000	0x2000_2FFF	GPIO Peripheral (16 pins)	Off Core Complex address space accessed via System and Peripheral busses
0x2000_3000	0x2000_3FFF	*Reserved*	
0x2000_4000	0x2000_4FFF	Quad SPI Flash Control	
0x2000_5000	0x2000_5FFF	8-bit, 4-comparator PWM	
0x2000_3000	0x3FFF_FFFF	*Reserved*	
0x4000_0000	0x5FFF_FFFF	Memory Mapped Quad SPI Interface	
0x6000_0000	0x7FFF_FFFF	*Reserved*	
0x8000_0000	0x8000_FFFF	Data Tightly Integrated Memory (DTIM) (64kB)	
0x8001_0000	0xFFFF_FFFF	*Reserved*	

Table 7.1: E31/E51 Core Complex FPGA Eval Kit Physical Memory Map

Table 7.2: E31/E51 Core Complex FPGA Eval Kit GPIO Offset to Board Pin Number

Peripheral	Peripheral Offset	Connections	Global Interrupt Number
UART	UART TX/RX	To USB Serial	1
	SWITCH 0	Direct Global	2
	SWITCH 1	Interrupts	3
	SWITCH 2		4
	SWITCH 3		5
Quad SPI	all QSPI	To Quad SPI Flash	6
GPIO	GPIO[0]	LED 0 RED	7
	GPIO[1]	LED 0 GREEN	8
	GPIO[2]	LED 0 BLUE	9
	GPIO[3]	SWITCH 3	10
	GPIO[4]	BUTTON 0	11
	GPIO[5]	BUTTON 1	12
	GPIO[6]	BUTTON 2	13
	GPIO[7]	BUTTON 3	14
	GPIO[8]	PMOD A[0]	15
	GPIO[9]	PMOD A[1]	16
	GPIO[10]	PMOD A[2]	17
	GPIO[11]	PMOD A[3]	18
	GPIO[12]	PMOD A[4]	19
	GPIO[13]	PMOD A[5]	20
	GPIO[14]	PMOD A[6]	21
	GPIO[15]	PMOD A[7]	22
PWM/Counter	PWM CMP[0]		23
	PWM CMP[1]	LED 1 RED	24
	PWM CMP[2]	LED 1 GREEN	25
	PWM CMP[3]	LED 1 BLUE	26
Non core complex	System Reset	LED[4]	
System Indicators	Debugger SRST_n	LED[5]	
	dmactive	LED[6]	
	internal Reset	LED[7]	

Table 7.3: E31/E51 Core Complex FPGA Eval Kit Local Interrupts Mapping

Hardware Input	Local Interrupt Number (Index in `mip`, `mie`, registers)
Switch 0	16
Switch 1	17
Switch 2	18
Switch 3	19
Button 0	20
Button 1	21
Button 2	22
Button 3	23
PMOD A[0]	24
PMOD A[1]	25
PMOD A[2]	26
PMOD A[3]	27
PMOD A[4]	28
PMOD A[5]	29
PMOD A[6]	30
PMOD A[7]	31

Chapter 8

For More Information

Additional information, the latest version of this guide, and supporting files can be found at `https://dev.sifive.com`