SiFive Core IP FPGA Eval Kit User Guide
v3p0

SiFive Core IP FPGA Eval Kit User Guide

Proprietary Notice

Release Information

Version	Date	Changes
v3p0	Feb 28, 2019	Updated for 19.02 Core IP Release • Supports v19.02 Core IP Package • Added descriptions for Arty-100T • Added Chapters for E2, E3/S5, E7/S7 MCS Images
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 Core IP FPGA Eval Kit. To learn more about the functionality of your specific Core IP please read the appropriate Core IP Manual.

This guide will help you download and flash the Core IP FPGA Eval Kit image to an 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 Core IP 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 v19.02 of the Core IP FPGA Eval Kit has the following limitations compared with the fully functional Core 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 IP RTL from sifive.com.

Chapter 2

Required Hardware

The Core IP FPGA Eval Kit requires the following hardware:

2.1 Xilinx Arty A7 Artix-7 FPGA Evaluation Kit

The Arty A7 is a Xilinx FPGA development board for makers and hobbyists. The Arty A7 comes in two FPGA variants: The Arty A7-35T features Xilinx XC7A35TICSG324-1L. The Arty A7-100T features the larger Xilinx XC7A100TCSG324-1. Both can be purchased from Digilent or Avnet.

http://www.xilinx.com/products/boards-and-kits/arty.html
https://store.digilentinc.com/

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 Core IP Arty FPGA Dev Kit has a standard JTAG debugging interface, and the tools included with the Core IP FPGA Eval Kit 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/

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 Core IP FPGA Eval 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 Core IP 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 3.1. Note that the Olimex ARM-USB-TINY-H and the PMOD header on the Arty Board have different numbering schemes. Figures 3.2 and 3.3 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

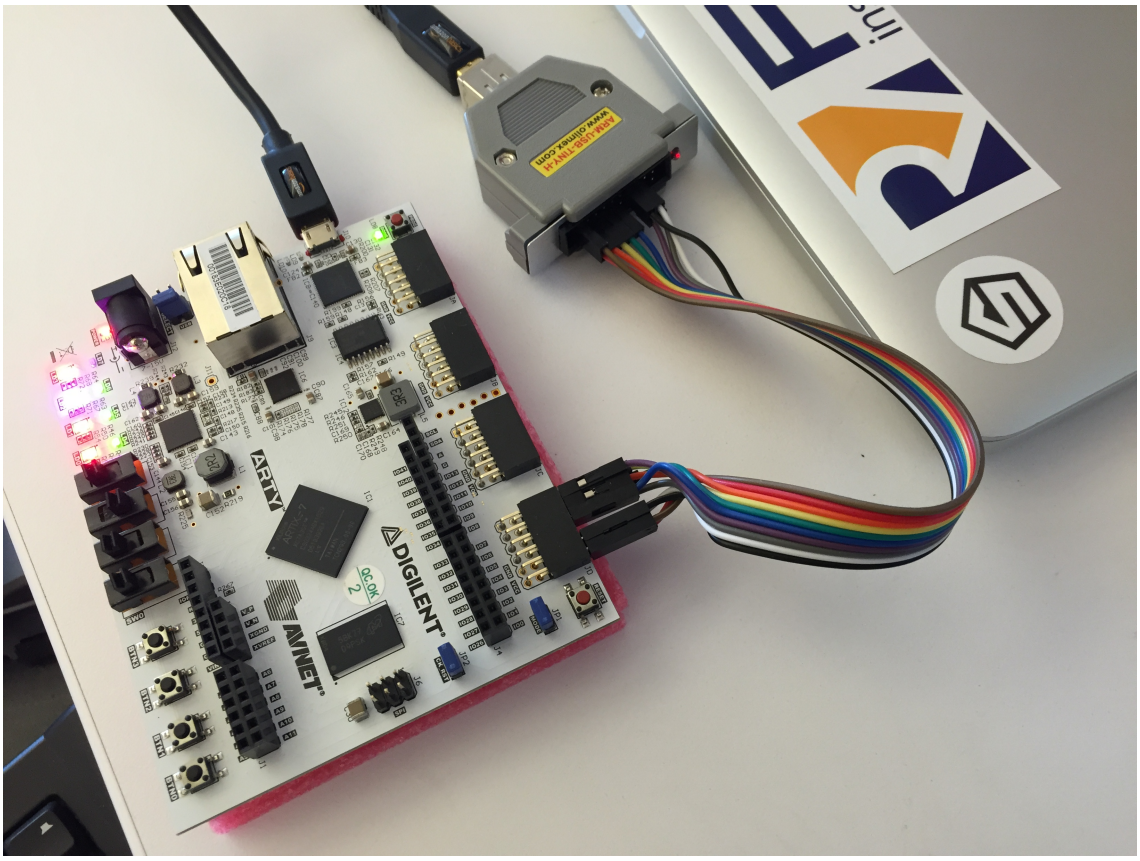

Figure 3.4: Photo of the Arty Board showing USB and Debug Connections

Chapter 4

FPGA Flash Programming File

The Xilinx Artix-7 35T or 100T FPGA configures on power-on from an on-board 16MB QuadSPI Flash.

To program the Arty Board, locate the desired MCS file within your Core IP package. For example,

`sifive_coreip_[XX]_FPGA_Evaluation_Arty_[35|100]T_v[XXX]_rc[xx].mcs`

The Xilinx Vivado Design Suite is used for flash programming. Both the Vivado Lab Edition and WebPACK Edition 2018.2 support Artix-7 devices free of charge.

4.1 Programming the Arty 35T SPI Flash

To program the Arty 35T SPI Flash with Vivado take the following steps:

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 parameters:

Part	m25ql128
Manufacturer	Micron
Alias	n25q12833vspi-x1_x2_x4
Family	mt25ql
Type	spi
Density	128
Width	x1_x2_x4

6. Click OK to “Do you want to program the configuration memory device now?”
7. Add the MCS file

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.

4.2 Programming the Arty 100T SPI Flash

To program the Arty 100T SPI Flash with Vivado take the following steps:

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 parameters:

Part	s25fl128sxxxxxx0
Manufacturer	Spansion
Alias	s25fl127s
Family	s25flxxxs
Type	spi
Density	128
Width	x1_x2_x4

6. Click OK to “Do you want to program the configuration memory device now?”
7. Add the MCS file
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

5.1 Serial Setup

Using a terminal emulator such as GNU screen on Linux or a terminal on Windows, open a console connection from the host computer to the Core IP 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/ttyUSB2 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}=="6011", MODE="664", GROUP="plugdev"

SUBSYSTEM=="tty", ATTRS{idVendor}=="0403",
  ATTRS{idProduct}=="6011", 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 /dev/tty/USB2 /dev/tty/USB3
```

(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/ttyUSB2`.

```
> ls -l /dev/ttyUSB2
crw-rw-r-- 1 root plugdev 188, 1 Nov 28 00:00 /dev/ttyUSB1
```

4. Add yourself to the `plugdev` group to eliminate the need to sudo for access to the device. 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.

5.1.1 Reset and boot

The FPGA Core IP Eval Kit's boot code contains a jump to the external SPI Flash as described in the DTS file.

For example,an E2/E3 or S5 Core IP 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.

A S7 or E7 Core IP FPGA Eval Kit will default to the QSPI to boot at address 0x20400000.

5.1.2 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.

When Switch 0 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.

5.2 Default Demo Program

For Core IP 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 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.

5.2.1 Terminal Log

If you have your serial setup correctly, your terminal will display a figure similar to the below. (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


    BUILD TIME : Feb 28 2019 : 00:00:00

Welcome to the E21 Core IP FPGA Evaluation Kit!
```

Chapter 6

Software Development Flow

SiFive supports several methods of obtaining the software development toolchain. Freedom Studio is an Eclipse-Based IDE which bundles everything you need into one download. You can also compile the source yourself and run command line tools with the Freedom E SDK..

These different development versions will all install the same set of tools, but the versions, install paths and associated software libraries and examples are different for each.

6.1 Supported Platforms

Freedom Studio is supported on Linux, macOS, and Windows.

6.2 Software Development Using Freedom Studio IDE

SiFive recommends software development for the Core IP 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/boards

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

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

6.3 Software Development Using Freedom E SDK Command Line Tools

Freedom-E-SDK is a public github repository, maintained by SiFive Inc, that makes it easy to get started developing software for SiFive's Freedom and RISC-V Core IP Platforms. The SDK supports a wide array of SiFive Core IPs, SoCs and Emulation environments.

https://github.com/sifive/freedom-e-sdk

This section describes how to setup the toolchain and configure the SDK. The section also walks through building an example program and executing it in the RTL testbench included in a SiFive Core IP deliverable. In addition, the section will walk through how to import custom BSPs and build a program using the custom BSP target.

6.3.1 Setting Up Freedom-E-SDK

Prerequisites

To use this SDK, you will need the following software available on your machine:

```
GNU Make
Git
```

Toolchain Prerequisites

To build examples and programs, you will need the following software installed on your machine:

- RISC-V GNU Toolchain
- RISC-V OpenOCD (for use with development board and FPGA targets)

Pre-built versions of these softwares can be found on the SiFive Website.

https://www.sifive.com/boards

The pre-built tools have been carefully packaged to support both RISCV 32bit & 64bit ISAs and work on Linux, macOS, and Windows hosts.

Download the toolchain your platform, and unpack it to your desired location. Then, use the RISCV_PATH and RISCV_OPENOCD_PATH variables when using the tools. For example,

```
> cp openocd-<date>-<platform>.tar.gz /my/desired/location/
> cp riscv64-unknown-elf-gcc-<date>-<platform>.tar.gz /my/desired/location
> cd /my/desired/location
> tar -xvf openocd-<date>-<platform>.tar.gz
> tar -xvf riscv64-unknown-elf-gcc-<date>-<platform>.tar.gz
> export RISCV_OPENOCD_PATH=/my/desired/location/openocd
> export RISCV_PATH=/my/desired/location/riscv64-unknown-elf-gcc-<date>-<version>
```

6.3.2 Cloning the Repository

The Freedom-E-SDK repository can be cloned by running the following commands:

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

The recursive option is required to clone the git submodules included in the repository. If at first you omit the recursive option, you can achieve the same effect by updating submodules using the command:

```
> git submodule update --init --recursive
```

6.4 Freedom E SDK Arty BSP

The Freedom Metal Compatibility Library layer uses, the board support package files, to provide the hardware abstraction layer. These bsp files can be found under the bsp folder in Freedom-E-SDK and are encapsulated entirely within each target directory. The supported targets are:

SiFive Freedom E310 Arty

- freedom-e310-arty

SiFive CoreIP Arty FPGA Evaluation targets

- coreip-eXX-arty
- coreip-sXX-arty

For example, the board support files may consist of the following:

`***design.dts***`

The DeviceTree description of the target. This file is used to parameterize the Freedom Metal library to the target device. It is included as reference so that users of Freedom Metal are aware of what features and peripherals are available on the target.

`***metal.h***`

The Freedom Metal machine header which is used internally to Freedom Metal to instantiate structures to support the target device.

`***metal.lds***`

The linker script for the target device.

`openocd.cfg (for development board and FPGA targets)`

Used to configure OpenOCD for flashing and debugging the target device.

`***settings.mk***`

Used to set march and mabi arguments to the RISC-V GNU Toolchain.

6.5 Example Programs

Some example programs can be found under software:

`***hello***`

Prints ”Hello, World!” to stdout, if a serial device is present on the target.

```
***return-pass***
```

Returns status code 0 indicating program success.

```
***return-fail***
```

Returns status code 1 indicating program failure.

```
***example-itim****
```

Demonstrates how to statically link application code into the Instruction Tightly Integrated Memory (ITIM) if an ITIM is present on the target.

```
***software-interrupt***
```

Demonstrates how to register a handler for and trigger a software interrupt

```
***timer-interrupt***
```

Demonstrates how to register a handler for and trigger a timer interrupt

```
***local-interrupt***
```

Demonstrates how to register a handler for and trigger a local interrupt

```
***example-pmp***
```

Demonstrates how to configure a Physical Memory Protection (PMP) region

6.6 Using the Freedom E SDK

6.6.1 Building an Example

To compile a bare-metal RISC-V program:

```
make BSP=metal PROGRAM=timer-interrupt TARGET=coreip-s51-arty software
```

The square brackets in the above command indicate optional parameters for the Make invocation.

6.6.2 Uploading to the Target Board

```
make BSP=metal [PROGRAM=hello] [TARGET=coreip-s51-arty] upload
```

6.6.3 Debugging a Target Program

```
make BSP=metal [PROGRAM=hello] [TARGET=coreip-s51-arty] debug
```

6.6.4 Cleaning a Target Program Build Directory

```
make BSP=metal [PROGRAM=hello] [TARGET=coreip-s51-arty] clean
```

6.6.5 Create a Standalone Project

You can export a program to a standalone project directory using the standalone target. The resulting project will be locked to a specific TARGET.

STANDALONE_DEST is a required argument to provide the desired project location.

```
make standalone BSP=metal [PROGRAM=hello] [TARGET=coreip-s51-arty]
  STANDALONE_DEST=/path/to/desired/location
```

Once the standalone project is created, it can be compiled simply by typing make.

```
cd /path/to/desired/location standalone
make
```

Run make help for more commands.

Chapter 7

E2 Core IP FPGA Eval Kit MCS Image Contents

Figure 7.1 shows a block diagram of the E2 Core IP FPGA Eval Kit.

The evaluation kit includes an evaluation Core IP along with additional peripherals and I/Os to allow software prototyping.

7.1 Core IP FPGA Eval Kit Memory Map

The FPGA design on the Core IP FPGA Eval Kit has an evaluation version of the SiFive E2 Core IP, as well as peripheral devices which are not included with the Core IP deliverable. These devices allow you to perform basic I/O to prototype and benchmark some basic applications.

Please refer to the Device Tree file, (.dts) for details of the Memory Map.

7.2 Core IP FPGA Eval Kit Clock and Reset

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

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 Core IP FPGA Eval Kit . This is combined with the `io_ndreset` to drive the `reset` input to the Core IP.

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

7.3 Core IP 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 7.1. 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 Core IP FPGA Eval Kit and Local Interrupt sources are provided in Table 7.2

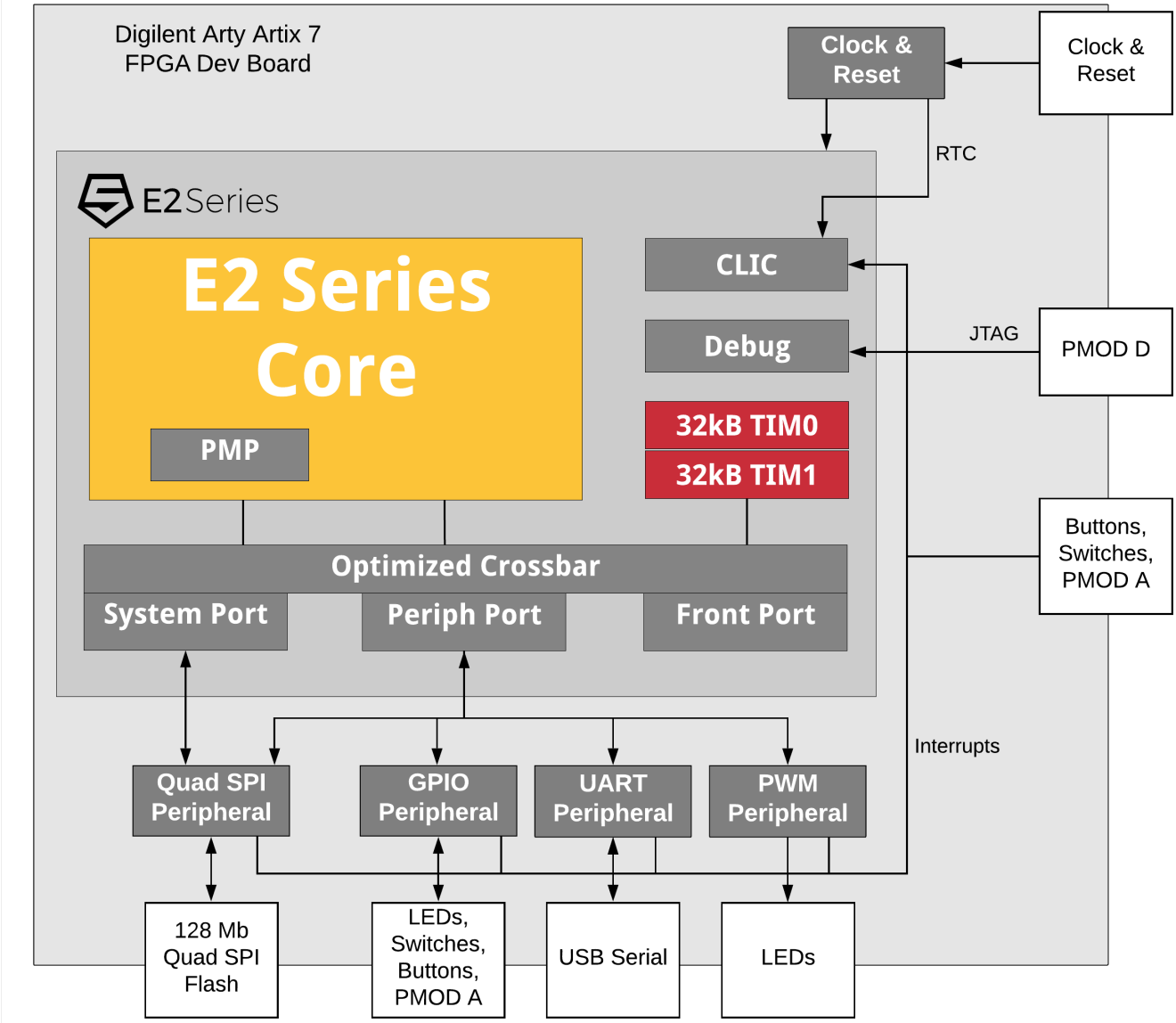

Figure 7.1: E2 Core IP FPGA Eval Kit Block Diagram

Table 7.1: Core IP 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 IP	System Reset	LED[4]	
System Indicators	Debugger SRST_n	LED[5]	
	dmactive	LED[6]	
	internal Reset	LED[7]	

Table 7.2: Core IP 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

E3 / S5 Core IP FPGA Eval Kit MCS Image Contents

Figure 8.1 shows a block diagram of the E3 / S5 Core IP FPGA Eval Kit.

The evaluation kit includes an evaluation Core IP along with additional peripherals and I/Os to allow software prototyping.

8.1 Core IP FPGA Eval Kit Memory Map

The FPGA design on the Core IP FPGA Eval Kit has an evaluation version of the SiFive E3 / S5 Core IP, as well as peripheral devices which are not included with the Core IP deliverable. These devices allow you to perform basic I/O to prototype and benchmark some basic applications.

Please refer to the Device Tree file, (.dts) for details of the Memory Map.

8.2 Core IP FPGA Eval Kit Clock and Reset

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

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 Core IP FPGA Eval Kit . This is combined with the `io_ndreset` to drive the `reset` input to the Core IP.

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

8.3 Core IP 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 8.1. 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 Core IP FPGA Eval Kit and Local Interrupt sources are provided in Table 8.2

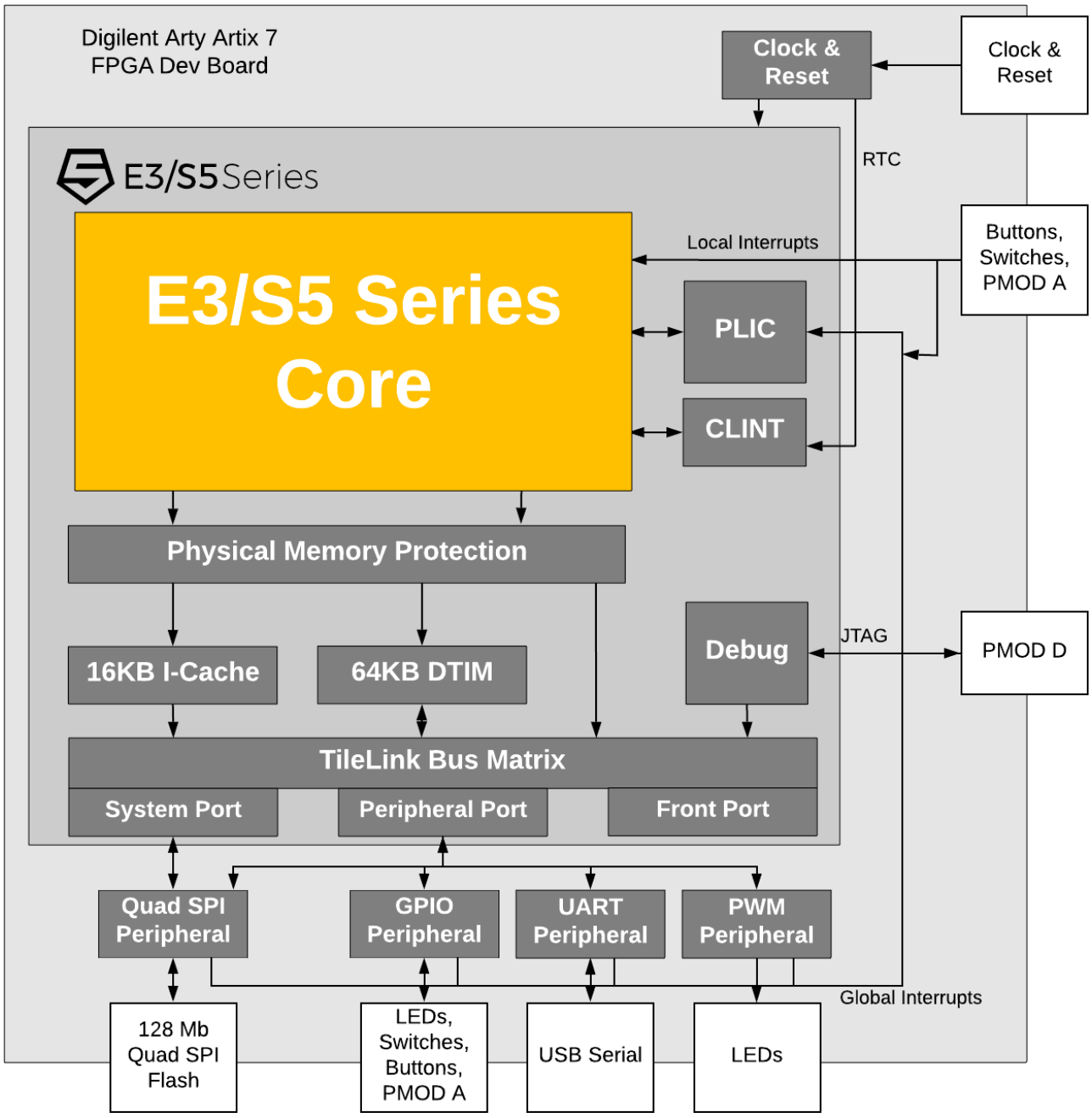

Figure 8.1: E3 / S5 Core IP FPGA Eval Kit Block Diagram

Table 8.1: Core IP 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 IP	System Reset	LED[4]	
System Indicators	Debugger SRST_n	LED[5]	
	dmactive	LED[6]	
	internal Reset	LED[7]	

Table 8.2: Core IP 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 9

E7 / S7 MCS Image Contents

Figure 9.1 shows a block diagram of the E7 / S7 Core IP FPGA Eval Kit. The evaluation kit includes an evaluation Core IP along with additional peripherals and I/Os to allow software prototyping.

9.1 Core IP FPGA Eval Kit Memory Map

The FPGA design on the Core IP FPGA Eval Kit has an evaluation version of the SiFive E7 / S7 Core IP, as well as peripheral devices which are not included with the Core IP deliverable. These devices allow you to perform basic I/O to prototype and benchmark some basic applications.

Please refer to the Device Tree file, (.dts) for details of the Memory Map.

9.2 Core IP FPGA Eval Kit Clock and Reset

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

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 Core IP FPGA Eval KitṪhis is combined with the `io_ndreset` to drive the `reset` input to the Core IP.

9.3 Core IP 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 9.1. 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 Core IP FPGA Eval Kit and Local Interrupt sources are provided in Table 9.2

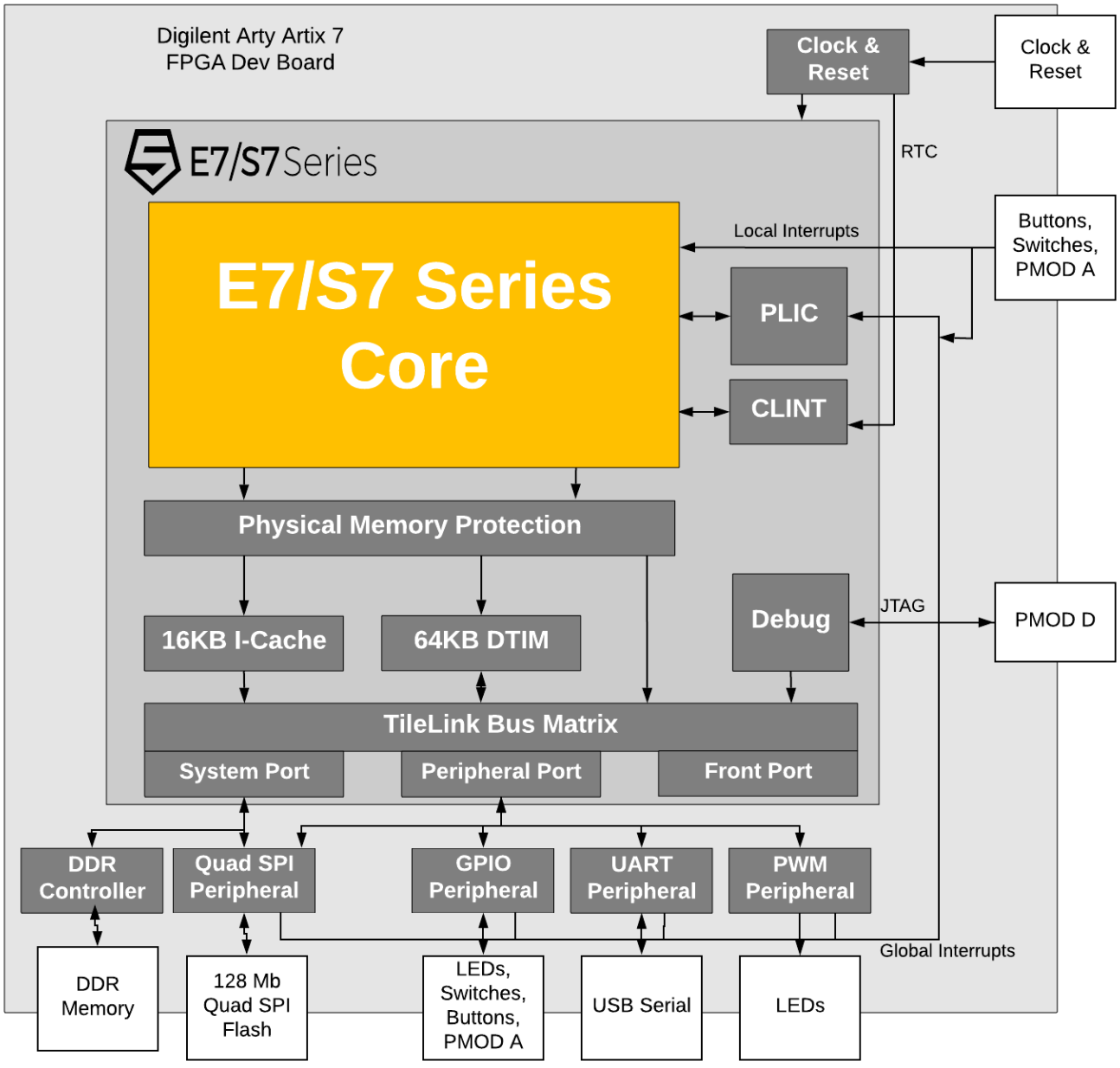

Figure 9.1: E7 / S7 Core IP FPGA Eval Kit Block Diagram

Table 9.1: Core IP 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 IP	System Reset	LED[4]	
System Indicators	Debugger SRST_n	LED[5]	
	dmactive	LED[6]	
	internal Reset	LED[7]	

Table 9.2: Core IP 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 10

For More Information

Additional information, the latest version of this guide, and supporting files can be found at
https://www.sifive.com
More information about RISC-V in general is available at
http://riscv.org
SiFive thoughts, ideas, and news at
https://www.sifive.com/blog/
Webinars at
https://info.sifive.com/risc-v-webinar