Integrator/LM-XCV600E+ Integrator/LM-EP20K600E+
User Guide

Copyright © 2000-2001. All rights reserved.

Release Information

<table>
<thead>
<tr>
<th>Date</th>
<th>Issue</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>24 November 2000</td>
<td>A</td>
<td>New document.</td>
</tr>
<tr>
<td>11 December 2000</td>
<td>B</td>
<td>Errors in switch settings corrected in Sections 4.3 and 5.3.</td>
</tr>
<tr>
<td>23 July 2001</td>
<td>C</td>
<td>Incorrect dimension in figure B-1 corrected.</td>
</tr>
</tbody>
</table>

Proprietary Notice

Words and logos marked with ® or ™ are registered trademarks or trademarks owned by ARM Limited, except as otherwise stated below in this proprietary notice. Other brands and names mentioned herein may be the trademarks of their respective owners.

Neither the whole nor any part of the information contained in, or the product described in, this document may be adapted or reproduced in any material form except with the prior written permission of the copyright holder.

The product described in this document is subject to continuous developments and improvements. All particulars of the product and its use contained in this document are given by ARM in good faith. However, all warranties implied or expressed, including but not limited to implied warranties of merchantability, or fitness for purpose, are excluded.

This document is intended only to assist the reader in the use of the product. ARM Limited shall not be liable for any loss or damage arising from the use of any information in this document, or any error or omission in such information, or any incorrect use of the product.

Conformance Notices

This section contains conformance notices.

Federal Communications Commission Notice

This device is test equipment and consequently is exempt from part 15 of the FCC Rules under section 15.103 (c).

Confidentiality Status

This document is Open Access. This document has no restriction on distribution.

Product Status

The information in this documents is Final (information on a developed product).
Web Address

http://www.arm.com
Contents
Integrator/LM-XCV600E+

Preface
About this document ................................................................. viii
Further reading .............................................................................. x
Feedback ....................................................................................... xi

Chapter 1 Introduction
1.1 About the ARM Integrator/LM logic module ......................... 1-2
1.2 Logic module architecture ..................................................... 1-5
1.3 Links, indicators, and switches ............................................... 1-6
1.4 Differences between core and logic modules ......................... 1-8
1.5 Care of modules ................................................................. 1-9

Chapter 2 Getting Started
2.1 Using a bench power supply .................................................. 2-2
2.2 Using the logic module with an Integrator motherboard ......... 2-3
2.3 Setting the DIP switches ........................................................ 2-5
2.4 Using Multi-ICE or other JTAG equipment ......................... 2-6

Chapter 3 Hardware Description
3.1 FPGA ............................................................................. 3-2
Contents

3.2 System bus interface ................................................................. 3-3
3.3 Clock control ............................................................. 3-4
3.4 Reset control ............................................................... 3-9
3.5 JTAG support ............................................................ 3-11
3.6 Memory ................................................................. 3-19
3.7 LEDs and switches ......................................................... 3-20
3.8 Prototyping and expansion .................................................... 3-21
3.9 Debug connectors ......................................................... 3-22

Chapter 4 Configuring Altera Logic Modules
4.1 Altera FPGA configuration system architecture ........................................ 4-2
4.2 Altera FPGA tool flow .............................................................. 4-4
4.3 Configuring the Altera FPGA from flash ............................................... 4-7
4.4 Loading new Altera FPGA configurations ............................................ 4-9
4.5 Reprogramming the PLD ............................................................. 4-11

Chapter 5 Configuring Xilinx Logic Modules
5.1 Xilinx FPGA configuration system architecture ........................................ 5-2
5.2 Xilinx FPGA tool flow .............................................................. 5-4
5.3 Configuring the Xilinx FPGA from flash ............................................... 5-6
5.4 Loading new Xilinx FPGA configurations ............................................ 5-8
5.5 Reprogramming the PLD ............................................................. 5-11

Chapter 6 Supplied FPGA Examples
6.1 About the FPGA configuration examples ............................................. 6-2
6.2 Example 2 programmer’s reference .................................................... 6-5

Appendix A Signal Descriptions
A.1 EXPA ................................................................. A-2
A.2 EXPB ................................................................. A-5
A.3 EXPIM ................................................................. A-10
A.4 Diagnostic connectors ............................................................ A-13

Appendix B Mechanical Specification
B.1 Mechanical details ............................................................. B-2
Preface

This preface introduces the ARM Integrator/LM logic modules and their reference documentation. It contains the following sections:

- *About this document* on page viii
- *Further reading* on page x
- *Feedback* on page xi.
About this document

This document describes how to set up and use the ARM Integrator/LM-XCV600E+ and Integrator/LM-EP20K600E+ logic modules.

Intended audience

This document has been written for experienced hardware and software developers as an aid to developing ARM-based products using these ARM Integrator logic modules as a standalone development system or with an Integrator motherboard.

Organization

This document is organized into the following chapters:

Chapter 1 Introduction
Read this chapter for an introduction to the logic module.

Chapter 2 Getting Started
Read this chapter for a description of how to set up and start using the logic module.

Chapter 3 Hardware Description
Read this chapter for a description of the hardware architecture of the logic module. This includes clocks, resets, and debug features.

Chapter 4 Configuring Altera Logic Modules
Read this chapter for a description of how an Altera FPGA is configured at power-up, the configuration options available, and how to download your own FPGA configurations.

Chapter 5 Configuring Xilinx Logic Modules
Read this chapter for a description of how a Xilinx FPGA is configured at power-up, the configuration options available, and how to download your own FPGA configurations.

Chapter 6 Supplied FPGA Examples
Read this chapter for a description of the example FPGA configurations supplied with the logic module. This chapter provides information essential for understanding the memory map and register functions necessary to support the logic module as part of an Integrator development system.
Appendix A Signal Descriptions
Refer to this appendix for connector pinouts.

Appendix B Mechanical Specification
Refer to this appendix for mechanical details of the logic module.

Typographical conventions
The following typographical conventions are used in this book:

*italic* Highlights important notes, introduces special terminology, denotes internal cross-references, and citations.

**bold** Highlights interface elements, such as menu names. Denotes ARM processor signal names. Also used for terms in descriptive lists, where appropriate.

```
monospace
```
Denotes text that can be entered at the keyboard, such as commands, file and program names, and source code.

```
monospace
```
Denotes a permitted abbreviation for a command or option. The underlined text can be entered instead of the full command or option name.

```
monospace italic
```
 Denotes arguments to commands and functions where the argument is to be replaced by a specific value.

```
monospace bold
```
Denotes language keywords when used outside example code.
Further reading

This section lists related publications by ARM Limited and other companies that provide additional information and examples.

ARM publications

The following publications provide information about related ARM products and toolkits:

- *ARM Integrator/AP User Guide* (ARM DUI 0098)
- *ARM Integrator/SP User Guide* (ARM DUI 0099)
- *ARM Multi-ICE User Guide* (ARM DUI 0048)
- *AMBA Specification* (ARM IHI 0011)
- *ARM Architectural Reference Manual* (ARM DDI 0100)
- *ARM Firmware Suite Reference Guide* (ARM DUI 0102)
- *ADS Tools Guide* (ARM DUI 0067)
- *ADS Debuggers Guide* (ARM DUI 0066)
- *ADS Debug Target Guide* (ARM DUI 0058)
- *ADS Developer Guide* (ARM DUI 0056)

Other publications

The following publication provides information about the clock controller chip used on the Integrator modules:

- *MicroClock OSCaR User Configurable Clock Data Sheet* (MDS525), MicroClock Division of ICS, San Jose, CA.
Feedback

ARM Limited welcomes feedback both on the ARM Integrator/LM and on the documentation.

Feedback on this document

If you have any comments about this document, please send email to errata@arm.com giving:
• the document title
• the document number
• the page number(s) to which your comments refer
• an explanation of your comments.

General suggestions for additions and improvements are also welcome.

Feedback on the ARM Integrator/LM-XCV600E+ and LM-EP20K600E+

If you have any comments or suggestions about this product, please contact your supplier giving:
• the product name
• an explanation of your comments.
Chapter 1
Introduction

This chapter provides an introduction to the ARM Integrator/LM logic modules. It contains the following sections:

- About the ARM Integrator/LM logic module on page 1-2
- Logic module architecture on page 1-5
- Links, indicators, and switches on page 1-6
- Differences between core and logic modules on page 1-8
- Care of modules on page 1-9.
1.1 About the ARM Integrator/LM logic module

The Integrator/LM logic module is designed as a platform for developing Advanced Microcontroller Bus Architecture (AMBA™) Advanced System Bus (ASB), Advanced High-performance Bus (AHB), and Advanced Peripheral Bus (APB) peripherals for use with ARM cores.

You can use the logic module in three ways:

- as a standalone system
- with an Integrator core module, and an Integrator/AP or Integrator/SP motherboard
- as a core module with either Integrator/AP or Integrator/SP motherboard if a synthesized ARM core, such as the ARM7TDMI-S, is programmed into the FPGA.
- stacked without a motherboard, if one module in the stack provides system controller functions of a motherboard.

Figure 1-1 on page 1-3 and Figure 1-2 on page 1-4 show the layout of the logic module.

There are two main variants of this logic module:

- the Integrator/LM-EP20K600E+ is fitted with an Altera Apex FPGA
- the Integrator/LM-XCV600E+ is fitted with a Xilinx Virtex E FPGA.

The functionality of the logic module is defined by a configuration image loaded into the FPGA at power-up. Two FPGA configuration examples are preloaded into flash to get you started with AMBA AHB or ASB designs. You can also download your own configurations using Multi-ICE® or using other tools supported by the FPGA manufacturer.
Figure 1-1 Integrator/LM-XCV600E+ layout

Module/motherboard connector

Trace connector

Multi-ICE connector

Link 3

CONFIG link

Link 1

Link 2

User LEDs

Interface module connector

Power connector (standalone)

Status LEDs

Module/motherboard connector

Prototyping grid

ZBT SSRAM

Flash memory

Logic analyzer connector

FPGA

General purpose switches

Mode switches

XChecker download connector

Push button

Interface module connector

ZBT SSRAM

Flash memory

Logic analyzer connector

FPGA

General purpose switches

Mode switches

XChecker download connector

Push button

Figure 1-1 Integrator/LM-XCV600E+ layout
Figure 1-2 Integrator/LM-EP20K600E+ layout
1.2 Logic module architecture

Figure 1-3 shows the architecture of the logic module.

The logic module comprises the following:

- Altera or Xilinx FPGA
- configuration PLD and flash memory for storing FPGA configurations
- 1MB ZBT SSRAM
- clock generators and reset sources
- switches
- LEDs
- prototyping grid
- JTAG, Trace, and logic analyzer connectors
- system bus connectors to a motherboard or other modules.

These components are discussed in detail in Chapter 3 Hardware Description.
1.3 Links, indicators, and switches

The logic module provides:

- CONFIG link
- nine user-definable surface-mounted LEDs
- user-definable push button
- 4-way mode switch and 8-way user definable switch.

You can also add your own links, switches, and small integrated circuits to the prototyping grid if required.

1.3.1 CONFIG link

The CONFIG link enables configuration mode. Configuration mode changes the JTAG signal routing and is used to download new PLD or FPGA configurations.

1.3.2 LEDs summary

The LEDs are listed in Table 1-1 on page 1-6.

<table>
<thead>
<tr>
<th>Name</th>
<th>Color</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFGLED</td>
<td>Orange</td>
<td>Configuration mode. This LED is lit when the CONFIG link is fitted.</td>
</tr>
<tr>
<td>POWER</td>
<td>Green</td>
<td>Power supply OK. This LED is lit when 3.3V power is supplied to the board.</td>
</tr>
<tr>
<td>FPGA_OK</td>
<td>Green</td>
<td>FPGA is configured. This LED is lit when power is supplied and the FPGA has loaded its configuration.</td>
</tr>
<tr>
<td>PROGFLASH</td>
<td>Orange</td>
<td>PLD is in flash programming mode (Altera only).</td>
</tr>
<tr>
<td>LED[7:0]</td>
<td>Green</td>
<td>These are general purpose LEDs connected to FPGA pins. Drive LOW to light.</td>
</tr>
<tr>
<td>LED8</td>
<td>Red</td>
<td>General purpose LED that you can use to signal error conditions. This is connected to an FPGA pin and must be driven LOW to light.</td>
</tr>
</tbody>
</table>

1.3.3 Switches

The 8-way DIP switch (S2) and push button (S3) provide general purpose inputs to the FPGA.
The 4-way DIP switch (S1) is used to select which of the FPGA configuration images stored in flash is used when the logic module powers up (see Chapter 4 Configuring Altera Logic Modules for Altera types, or Chapter 5 Configuring Xilinx Logic Modules for Xilinx types).
1.4 Differences between core and logic modules

Core and logic modules handle the interrupt signals differently. Core modules must receive interrupts, but logic modules, that implement peripherals, generate interrupts.

The signals on HDRB and EXPB concerned with interrupts are different, as shown in Table 1-2 on page 1-8. All signals on these pins must be driven open-collector (open-drain) to prevent conflict when logic and core modules are connected together in the same stack.

Appendix A Signal Descriptions provides a full description of all the connector pins.

<table>
<thead>
<tr>
<th>HDRB Label</th>
<th>Description</th>
<th>EXPB Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nFIQ0</td>
<td>Fast interrupt to module 0</td>
<td>E16</td>
<td>Not used</td>
</tr>
<tr>
<td>nFIQ1</td>
<td>Fast interrupt to module 1</td>
<td>E17</td>
<td>Not used</td>
</tr>
<tr>
<td>nFIQ2</td>
<td>Fast interrupt to module 2</td>
<td>E18</td>
<td>Not used</td>
</tr>
<tr>
<td>nFIQ3</td>
<td>Fast interrupt to module 3</td>
<td>E19</td>
<td>Not used</td>
</tr>
<tr>
<td>nIRQ0</td>
<td>Interrupt to module 0</td>
<td>IRQSRC0</td>
<td>Interrupt source from module 0 to interrupt controller</td>
</tr>
<tr>
<td>nIRQ1</td>
<td>Interrupt to module 1</td>
<td>IRQSRC1</td>
<td>Interrupt source from module 1 to interrupt controller</td>
</tr>
<tr>
<td>nIRQ2</td>
<td>Interrupt to module 2</td>
<td>IRQSRC2</td>
<td>Interrupt source from module 2 to interrupt controller</td>
</tr>
<tr>
<td>nIRQ3</td>
<td>Interrupt to module 3</td>
<td>IRQSRC3</td>
<td>Interrupt source from module 3 to interrupt controller</td>
</tr>
</tbody>
</table>

You can use the logic module to implement a synthesized processor, such as an ARM7TDMI-S, in which case it functions as a core module. As a core module, it receives interrupts, and is installed in the HDRA/B stack.

Also, on the Integrator/AP, the 32 GPIO lines are routed to the EXPB connector, but not the HDRB connector. The GPIO signals are not available on the Integrator/SP.
1.5 Care of modules

This section contains advice about how to prevent damage to your Integrator modules.

—— Caution ————

To prevent damage to your logic module, observe the following precautions:

- When removing a core or logic module from a motherboard, or when separating modules, take care not to damage the connectors. Do not apply a twisting force to the ends of the connectors. Loosen each connector first before pulling on both ends of the module at the same time.

- Use the logic module in a clean environment and avoid debris fouling the connectors on the underside of the PCB. Blocked holes result in damage to connectors on the motherboard or module below. Visually inspect the module to ensure that connector holes are clear before mounting it onto another board.

- Observe ElectroStatic Discharge (ESD) precautions when handling any Integrator board.
Introduction
Chapter 2
Getting Started

This chapter describes how to set up and start using the logic module. It contains the following sections:

- Using a bench power supply on page 2-2
- Using the logic module with an Integrator motherboard on page 2-3
- Setting the DIP switches on page 2-5
- Using Multi-ICE or other JTAG equipment on page 2-6.
2.1 Using a bench power supply

To power the logic module as a standalone system, connect a bench power supply capable of supplying +3.3V and +5V using the screw terminals shown in Figure 2-1 on page 2-2. You must apply and remove the 3.3V and 5V supplies simultaneously.

--- Caution ---
You must take care to wire the supply correctly, because there is no reverse-polarity protection. The power terminals are marked clearly on the PCB.

--- Figure 2-1 Power supply screw terminals ---
2.2 Using the logic module with an Integrator motherboard

The logic module and core modules can be mounted onto an Integrator/AP or Integrator/SP motherboard, as described in:
- Mounting the logic module on an Integrator/AP on page 2-3
- Mounting on an Integrator/SP on page 2-4.

--- Note ---
The logic module can be configured to support stacking without a motherboard (see Module stacking options on page 3-15).

2.2.1 Mounting the logic module on an Integrator/AP

The Integrator/AP provides two module mounting positions. These are used as follows
- logic modules mount onto the connectors EXPA and EXPB
- core modules mount onto the connectors HDRA and HDRB.

Figure 2-2 on page 2-3 shows an example system a core module and four logic modules attached to an Integrator/AP (see the Integrator/AP User Guide for more details).

![Figure 2-2 Assembled Integrator/AP development system](image)
Note

Logic modules can be mounted on the HDRA and HDRB connector without causing damage. However, there are differences in the routing of some signals on the HDRB and EXPB that affect the operation of the module (see Differences between core and logic modules on page 1-8).

Fitting procedure

Caution

To prevent damage to the Integrator/AP and modules:
- Power down before fitting or removing modules.
- Do not exceed four modules in one stack.
- Do not exceed a combined total of five modules on the Integrator/AP.

Fit a logic module as follows:

1. Place the Integrator/AP on a firm level surface.
2. Align connectors EXPA and EXPB on the logic module with the corresponding connectors on the Integrator/AP.
3. Press firmly on both ends of the logic module so that both connectors close together at the same time.
4. Repeat steps 2 and 3 for additional modules.

2.2.2 Mounting on an Integrator/SP

The Integrator/SP provides one mounting position which means that core and logic modules are mounted in a single stack. This limits the total number of modules that can be fitted to four.

There are differences in the routing of some signals that affect the operation of the logic module if it is mounted on the Integrator/SP. This particularly applies to the interrupt signal routing (see Differences between core and logic modules on page 1-8).
2.3 Setting the DIP switches

When the logic module powers up in user mode, the FPGA loads configuration data from flash memory. The flash memory is preloaded with two example configuration images. These are selected as follows:

- The 4-way DIP switch (S1) is used by the preloaded PLD configuration to select the FPGA configuration.
- If the logic module is mounted on a motherboard, the signals CFGSEL [1:0] from the motherboard can be used to select the appropriate FPGA configuration to support operation with an AHB or ASB motherboard.

For a full description of FPGA configuration image selection, see Configuring the Altera FPGA from flash on page 4-7 for Altera types, or Configuring the Xilinx FPGA from flash on page 5-6 for Xilinx types.

The 8-poles DIP switch (S2) is intended for general-purpose use after configuration.
2.4 Using Multi-ICE or other JTAG equipment

JTAG equipment, such as Multi-ICE, is connected to the 20-way box header, as shown in Figure 2-3 on page 2-6. When multiple core or logic modules are stacked on a motherboard, the JTAG equipment is always connected to the top module in the stack. Refer to Reset control on page 3-9 for a description of the JTAG system.

Note

The logic module programming utility requires Multi-ICE release 1.4 or above. Refer to the Multi-ICE User Guide for details of how to use Multi-ICE.
Chapter 3

Hardware Description

This chapter describes logic module hardware and contains the following sections:

- **FPGA** on page 3-2
- **System bus interface** on page 3-3
- **Clock control** on page 3-4
- **Reset control** on page 3-9
- **JTAG support** on page 3-11
- **Memory** on page 3-19
- **LEDs and switches** on page 3-20
- **Prototyping and expansion** on page 3-21
- **Debug connectors** on page 3-22.
3.1 FPGA

The two types of logic are module fitted with either a Xilinx Virtex an Altera Apex FPGA. The assignment of the input/output banks and JTAG implementation are described in the following sections:

- FPGA bank assignment on page 3-2
- JTAG and the FPGA on page 3-2.

For information about how the FPGA is configured, see Chapter 4 Configuring Altera Logic Modules, or Chapter 5 Configuring Xilinx Logic Modules.

For information about the configurations supplied with your logic module, see Chapter 6 Supplied FPGA Examples.

3.1.1 FPGA bank assignment

The FPGA input/output pins are organized into eight banks. Most of the input/output pins are used to support the logic module when it is configured to operate with an Integrator motherboard or core module. Support for prototyping is also provided as follows:

- two banks of input/output signals are connected to the interface module connector EXPIM (see Interface module connector on page 3-21)
- one bank of the input/output signals is connected to the prototyping grid as well as the EXPIM connector (see Prototyping and expansion on page 3-21).

3.1.2 JTAG and the FPGA

The FPGA contains a hardware JTAG TAP controller. You can use this TAP controller to download new FPGA configurations. In addition, a number of input/output pins are reserved for a virtual TAP controller synthesized into the FPGA configuration. You can use the virtual TAP controller to access devices that are synthesized in the FPGA.

The CONFIG link is used to route the JTAG connector to the hardware TAP controller or the virtual TAP controller (see JTAG support on page 3-11).
3.2 System bus interface

The system bus interface connects the logic module with other Integrator modules. This must be implemented according to the AHB or ASB specifications. Example configurations are supplied with the logic module to get you started and to enable you to develop with various bus configurations (see Chapter 6 Supplied FPGA Examples).

In a conventional AMBA system, a single central decoder is used to provide a select signal \((DSEL_x \text{ for ASB})\) for each slave on the bus. However, in the Integrator family this scheme is varied. Each module is responsible for providing its own select signals. This provides greater flexibility and improves performance.

The Integrator memory map defines the address space of each module depending on its position within the stack and on whether it is mounted in the HDRA/HDRB or EXPB stack. When a module is not present, the central decoder on the motherboard provides a default response for bus transfers in the unoccupied address space. This default response is switched off when the module is present.

The scheme requires the central decoder to detect which modules are present and for each module to detect its position in the stack, and in which stack it is mounted. A module must respond to all memory accesses within its allocated address space but not to accesses outside of its allocated space.

The signals \(ID[3:0]\), \(nPPRES[3:0]\), and \(nEPRES[3:0]\) and a signal rotation scheme are used by modules to determine their position in the stack and to signal their presence to the central decoder. A logic module can determine its position from \(ID[3:0]\), and therefore its address range. Table 3-1 on page 3-3 shows addresses for modules in either stack position.

<table>
<thead>
<tr>
<th>ID[3:0]</th>
<th>Module ID</th>
<th>EXPA/EXPB</th>
<th>HDRA/HDRB</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1101</td>
<td>3 (top)</td>
<td>0xF0000000</td>
<td>0xB0000000</td>
<td>256MB</td>
</tr>
<tr>
<td>1011</td>
<td>2</td>
<td>0xE0000000</td>
<td>0xA0000000</td>
<td>256MB</td>
</tr>
<tr>
<td>0111</td>
<td>1</td>
<td>0xD0000000</td>
<td>0x90000000</td>
<td>256MB</td>
</tr>
<tr>
<td>1110</td>
<td>0 (bottom)</td>
<td>0xC0000000</td>
<td>0x80000000</td>
<td>256MB</td>
</tr>
</tbody>
</table>
3.3 Clock control

The FPGA has four dedicated clock inputs for use in user mode. The function and control of the clock signals are described in the following sections:

- Clock architecture on page 3-4
- Programming the on-board clock generators on page 3-7.

3.3.1 Clock architecture

Figure 3-1 on page 3-4 shows the architecture of the clock subsystem.
The ICS525 devices are two programmable oscillator devices. These are supplied with a 24MHz reference clock and the frequency of their output clocks are configured by setting signal levels on their divider input pins. All divider inputs are connected to the FPGA. Pull-down resistors on the divider inputs ensure that the oscillator outputs default to 4.8MHz if the FPGA is not configured.

The FPGA configuration examples supplied with the logic module provide two registers, LM_OSC1 and LM_OSC2, which control the divider inputs (see Example 2 programmer's reference on page 6-5).

The output clocks from the IC525 devices are fed to two buffers. Both are provided with enable inputs (PWRDN_CLK1 and PWRDN_CLK2) from the FPGA. The buffer for CLK1 defaults to ON and the buffer for CLK2 defaults to OFF.

Note

The CLK2 buffer should only be enabled if the logic module is used to clock other modules when stacked without a motherboard (see Module stacking options on page 3-15). CLK2 to the FPGA is always enabled.

The system clocks from the Integrator motherboard are controlled by similar oscillators on the motherboard. See the user guide for your motherboard for more information.

3.3.2 Clock signal summary

Table 3-2 on page 3-5 provides a summary of the clock signals on the logic module.

<table>
<thead>
<tr>
<th>Clock name</th>
<th>Clock source</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSCLK</td>
<td>Motherboard system clock</td>
</tr>
<tr>
<td>CLK1</td>
<td>On-board clock generator (programmable)</td>
</tr>
<tr>
<td>CLK2</td>
<td>On-board clock generator (programmable)</td>
</tr>
<tr>
<td>IM_CLK</td>
<td>Clock supplied from an interface module</td>
</tr>
<tr>
<td>CCLK (Xilinx)</td>
<td>Configuration clock supplied by the PLD to the FPGA during FPGA configuration</td>
</tr>
<tr>
<td>DCLK (Altera)</td>
<td></td>
</tr>
</tbody>
</table>
Table 3-2 Logic module clock signals (continued)

<table>
<thead>
<tr>
<th>Clock name</th>
<th>Clock source</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCLK</td>
<td>This signal provides a clock signal to the ZBT SSRAM</td>
</tr>
<tr>
<td>PWRDNCLK1</td>
<td>This signal can be used to enable or disable the CLK1_3:0 and CLK1 outputs</td>
</tr>
<tr>
<td>PWRDNCLK2</td>
<td>This signal can be used to enable or disable the SYSCLK[3:0] outputs to HDRB</td>
</tr>
</tbody>
</table>
3.3.3 Programming the on-board clock generators

The two clock generators are independently programmable and produce frequencies in the range 1MHz to 160MHz. Each clock is controlled by an associated set of input signals `CTRLCLKx[18:0]` that are assigned to the control inputs of the oscillators as shown in Table 3-3 on page 3-7.

<table>
<thead>
<tr>
<th>Signals</th>
<th>Control parameter</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTRLCLKx[18:16]</td>
<td>Output divider</td>
<td>S[2:0]</td>
</tr>
<tr>
<td>CTRLCLKx[15:9]</td>
<td>Reference divider</td>
<td>R[6:0]</td>
</tr>
<tr>
<td>CTRLCLKx[8:0]</td>
<td>VCO divider</td>
<td>V[8:0]</td>
</tr>
</tbody>
</table>

The reference divider and VCO divider are used to calculate the output frequency using the following formula:

\[
\text{Frequency} = 48\text{MHz} \cdot \frac{(V[8:0]+8)}{(R[6:0]+2) \cdot S}
\]

The output divider S can be assigned any of the values shown in Table 3-4.

<table>
<thead>
<tr>
<th>S</th>
<th>S[2:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>001</td>
</tr>
<tr>
<td>4</td>
<td>011</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
</tr>
<tr>
<td>6</td>
<td>111</td>
</tr>
<tr>
<td>7</td>
<td>101</td>
</tr>
<tr>
<td>8</td>
<td>010</td>
</tr>
<tr>
<td>9</td>
<td>110</td>
</tr>
<tr>
<td>10</td>
<td>000</td>
</tr>
</tbody>
</table>
The following operating range limits must be observed:

\[ \frac{(V[8:0] + 8)}{(R[6:0] + 2)} \cdot 10MHz < 48MHz \]

\[ R[6:0] < 118 \]

--- **Note** -----

You can calculate values for the clock control signals using the ICS525 calculator on the Integrated Circuit Systems website at:

3.4 Reset control

The logic module provides three predefined reset signals and a push button that you can use to assert a reset. Figure 3-2 on page 3-9 shows the architecture of the reset system.

3.4.1 JTAG test reset (nTRST)

The JTAG test reset signal, nTRST, is an active LOW open-collector signal. It is connected to an FPGA input/output pin to provide a reset input to the TAP controllers. There are three possible sources of the nTRST signal:
- Multi-ICE connector
- Trace (embedded trace macrocell) connector
- motherboard or core module on the EXPB connectors
3.4.2 Multi-ICE system reset (nSRST)

The Multi-ICE system reset signal, nSRST, is a bidirectional, active LOW, open-collector signal. It can be driven by JTAG equipment to reset the logic module. Some JTAG equipment monitors this line to sense when the module has been reset by the user. The nSRST signal connects to an FPGA input/output pin, and is present on Multi-ICE, Trace, and EXPB connectors in a similar way to the nTRST signal.

——— Note ————

On the Integrator/AP, the expansion connector (EXPB) nSRST signal is completely separate from the core module (HDRB) nSRST signal (see the Integrator/AP User Guide for more details).

3.4.3 Motherboard reset (nSYSRST)

The motherboard reset signal, nSYSRST, is driven by the motherboard system controller, and is routed to an FPGA input/output pin on the logic module (see the Integrator/AP User Guide for further information).
3.5 JTAG support

The logic module provides support for programming using JTAG. The Multi-ICE and Trace connectors provide access to the FPGA and PLD TAP controllers. The JTAG hardware is connected to the top board in the stack.

The routing of the JTAG signals depends on the following factors:
- whether the logic module is being used standalone or is mounted on a motherboard
- whether the logic module is in configuration mode or user mode
- whether the board is in flash program mode (Altera type only).

The CONFIG link allows you to select between two JTAG modes:
- configuration mode, used for in-system reprogramming of the FPGA or PLD
- user mode.

Note
The Integrator/AP provides logic module and core module mounting positions. The JTAG signals in these two positions are completely isolated. When the logic module has been programmed, downloading and debugging of ARM code is performed using the JTAG connector on the core module.

3.5.1 Configuration mode

Figure 3-3 on page 3-11 shows the JTAG routing on the logic module in configuration mode.

Select configuration mode by fitting the CONFIG link. Fitting the CONFIG link on the top module in a stack selects configuration mode on all of the modules in the same stack. The CONFIG LED is lit on all modules in the same stack.
In configuration mode, the FPGA and the PLD are connected into the TDI-TDO chain. This allows the FPGA, PLD, and flash memory to be configured or programmed using the JTAG port.

--- Note ---
If more than one module is present when the stack is in configuration mode, reduce the JTAG TCK speed to 1MHz or below to ensure reliable operation (see the *Multi-ICE User Guide*).

### 3.5.2 Flash program mode (Altera only)

Figure 3-4 on page 3-12 shows the JTAG routing for flash program mode.

![Figure 3-4 Flash program mode JTAG routing](image)

Select flash program mode by inserting the CONFIG link and setting S1[4] to the OPEN position.

--- Note ---
The JTAG TCK speed must be set to 1MHz to ensure reliable operation.

### 3.5.3 User mode

Figure 3-5 on page 3-12 shows the JTAG routing for user mode.

![Figure 3-5 User and configuration mode JTAG routing](image)
Select user mode by removing the CONFIG link. This is the default mode.

In user mode, the JTAG signals are connected to FPGA input/output pins provided to enable you to implement a virtual TAP controller. This facility is provided for FPGA designs that require a TAP controller, for example, designs that include a synthesized processor.

In user mode, the four standard JTAG signals, along with RTCK and nTRST, are routed to the virtual TAP controller on the FPGA. The hardware FPGA TAP controller and the PLD are switched out of the TDI-TDO path. RTCK is a Multi-ICE specific signal used to support adaptive clocking (see Using Multi-ICE adaptive clocking on page 3-16).

If your design (or any other module in the same stack) does not implement a TAP controller, then you can ignore these connections. This is usually the case when prototyping AMBA peripherals on the Integrator/AP with the logic module in the expansion position.

Note
If there is another module in the stack that requires JTAG support, the FPGA design must route TDI to TDO and TCK to RTCK for the JTAG system to work correctly.

3.5.4 Using JTAG with a multi-module Integrator system

Figure 3-6 on page 3-14 shows the JTAG data routing for two logic modules and a motherboard in user mode.

Routing switches on the logic module are controlled by the signal nMBDET from the motherboard. If the logic module is used standalone, nMBDET is pulled HIGH and the JTAG path is confined to components on the logic module.

If the logic module is mounted on a motherboard, either directly or on top of another module, as in Figure 3-6 on page 3-14, nMBDET is pulled LOW by the motherboard and the TDI-TDO data path is routed first down to the motherboard and then up through each module in turn. A maximum of four modules can be stacked in this way. This can be a combination of core and logic modules.

The JTAG port does not normally operate if the logic module is stacked without a motherboard. This is because nMBDET is pulled HIGH, isolating the logic module JTAG from other modules.
This logic module can be configured to allow certain core module types to be stacked on it without a motherboard being present (see Module stacking options on page 3-15). Refer to the documentation for your core module for information about support for this mode of operation.

Figure 3-6 JTAG data path (user mode)
3.5.5 Module stacking options

The logic module provides three stacking options that can be selected by moving a surface-mount link (LK3). The link ensures that the TDI/TDO and TCK/RTCK signals are correctly routed through the stack for each configuration.

Table 3-5 on page 3-15 shows the link position used to select the different stacking options.

<table>
<thead>
<tr>
<th>Position</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>B-C</td>
<td>Normal (default)</td>
</tr>
<tr>
<td>A-B</td>
<td>LM at bottom of stack with no motherboard</td>
</tr>
<tr>
<td>C-D</td>
<td>LM at positions 1, 2, or 3 with no motherboard</td>
</tr>
</tbody>
</table>

The stacking options are:

**normal**  The normal option allows the module to be used standalone or with a motherboard.

**Logic module at bottom of the stack and no motherboard**

This option uses a logic module at the bottom of a stack of one or more other logic modules. One logic module must provide the system control function (for example, a system bus arbiter) normally provided by the motherboard.

To use this option:

- on the logic module at the bottom of the stack, set LK3 to A-B (see Table 3-5 on page 3-15).
- on any other logic modules, set LK3 to the C-D position.
- on one logic module, program and enable the CLK2 clock generator (see Clock architecture on page 3-4).

**Core module at bottom of the stack and no motherboard**

This option uses a core module at the bottom of a stack of one or more other modules. One logic module must be included that provides the system control function (for example, a system bus arbiter) normally provided by the motherboard.
Note

Module stacking without a motherboard is supported by later core module types that have a link similar to LK3 on the logic module. At the time of publication supporting core modules are:

- Integrator/CM9x6E-S (rev C and later)
- Integrator/CM9x0T-ETM (rev C and later)
- Integrator/CM10200 (rev C and later).

For up to date information about core module support for this stacking option, refer to the ARM website.

To use this option:

- on the core module at the bottom of the stack, set the link to the appropriate position (see the user guide for your core module).
- on any logic modules, set LK3 to the C-D position.
- on one logic module, program and enable the CLK2 clock generator (see Clock architecture on page 3-4).

3.5.6 Using Multi-ICE adaptive clocking

To use Multi-ICE with adaptive clocking, ensure the following:

- **TCK** is returned on **RTCK** to the Multi-ICE connector.
- The **TCK** signal to any logic module in a stack below the top board is driven by the **RTCK** output of the board above. The signal is then routed down through other modules to the motherboard and then back up to the Multi-ICE connector.

See the Multi-ICE User Guide for more information.
3.5.7 JTAG signal descriptions

Table 3-6 on page 3-17 provides a summary of the JTAG signals.

Table 3-6 JTAG and related signals

<table>
<thead>
<tr>
<th>Signal</th>
<th>Description</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDI</td>
<td>Test Data In</td>
<td>This signal is routed down the stack of modules to the motherboard and</td>
</tr>
<tr>
<td></td>
<td>(from JTAG tool)</td>
<td>then up through any connected JTAG device on each module in the stack</td>
</tr>
<tr>
<td></td>
<td></td>
<td>and returned to the Multi-ICE connector as TDO.</td>
</tr>
<tr>
<td>TDO</td>
<td>Test Data Out</td>
<td>This signal is the return path of the data input signal TDI. The logic</td>
</tr>
<tr>
<td></td>
<td>(to JTAG tool)</td>
<td>module connects to the TDO signal from the module beneath using the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TDO_Below pin on the EXPB socket. The signal from this pin is</td>
</tr>
<tr>
<td></td>
<td></td>
<td>routed through TAP controllers in devices on the logic module as TDI</td>
</tr>
<tr>
<td></td>
<td></td>
<td>and is then routed to the next module up the stack on the TDO pin of the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXPB plug. The length of track driven by the last component in the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>chain is kept as short as possible.</td>
</tr>
<tr>
<td>TCK</td>
<td>Test Clock</td>
<td>This signal synchronizes all JTAG transactions. TCK connects to all</td>
</tr>
<tr>
<td></td>
<td>(from JTAG tool)</td>
<td>JTAG components in the TDI-TDO chain. It makes use of series</td>
</tr>
<tr>
<td></td>
<td></td>
<td>termination resistors on stubs to reduce reflections and maintain good</td>
</tr>
<tr>
<td></td>
<td></td>
<td>signal integrity. TCK flows down the stack of modules and connects to</td>
</tr>
<tr>
<td></td>
<td></td>
<td>each JTAG component, but if there is a device in the scan chain that</td>
</tr>
<tr>
<td></td>
<td></td>
<td>synchronizes TCK to some other clock, then all down-stream devices</td>
</tr>
<tr>
<td></td>
<td></td>
<td>are connected to the RTCK signal on that component (see RTCK below).</td>
</tr>
<tr>
<td>TMS</td>
<td>Test Mode Select</td>
<td>This signal controls transitions in the tap controller state machine. TMS</td>
</tr>
<tr>
<td></td>
<td>(from JTAG tool)</td>
<td>connects to all JTAG components in the scan chain as the signal flows</td>
</tr>
<tr>
<td></td>
<td></td>
<td>down the module stack.</td>
</tr>
<tr>
<td>RTCK</td>
<td>Return TCK</td>
<td>Some devices sample TCK (for example, a synthesizable core with only</td>
</tr>
<tr>
<td></td>
<td>(to JTAG tool)</td>
<td>one clock), and this delays the time at which a component actually</td>
</tr>
<tr>
<td></td>
<td></td>
<td>captures data. RTCK is used to return the sampled clock to the JTAG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>equipment, so that the TCK is not advanced until the synchronizing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>device has captured the data. In adaptive clocking mode, Multi-ICE must</td>
</tr>
<tr>
<td></td>
<td></td>
<td>detect an edge on RTCK before changing TCK. In a multiple-device</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JTAG chain, the RTCK output from a component connects to the TCK input</td>
</tr>
<tr>
<td></td>
<td></td>
<td>of the down-stream device. The RTCK signal on the module</td>
</tr>
<tr>
<td></td>
<td></td>
<td>connectors HDRB/EXPB returns TCK to the JTAG equipment. If there are</td>
</tr>
<tr>
<td></td>
<td></td>
<td>no synchronizing components in the TDI-TDO chain then, it is not</td>
</tr>
<tr>
<td></td>
<td></td>
<td>necessary to use the RTCK signal and it is connected to ground on the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>motherboard.</td>
</tr>
</tbody>
</table>

Copyright © 2000-2001. All rights reserved.
Table 3-6 JTAG and related signals (continued)

<table>
<thead>
<tr>
<th>Signal</th>
<th>Description</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>nRTCKEN</td>
<td>Return TCK enable</td>
<td>This active LOW signal is driven by any module that requires RTCK to be routed back to the JTAG equipment. If nRTCKEN is HIGH the motherboard drives RTCK LOW. If nRTCKEN is LOW, the motherboard drives the TCK signal back up the stack to the JTAG equipment. The logic module drives this signal LOW when it is not in configuration mode. This signal is left unconnected by modules that do not require adaptive clocking.</td>
</tr>
<tr>
<td>nCFGEN</td>
<td>Configuration enable</td>
<td>This active LOW signal is used to put the boards into configuration mode. In configuration mode all FPGAs and PLDs are connected to the TDI-TDO chain so that they can be configured by the JTAG equipment.</td>
</tr>
<tr>
<td>FPGA_DONE</td>
<td>All FPGAs are configured</td>
<td>This open-collector signal indicates when all FPGAs in the system have configured. This signal is not part of the JTAG scheme, but is relevant to how the boards are reset and, therefore, has an effect on nSRST. The signal is routed between all FPGAs in the system through a pin on the HDRB/EXPB connectors. The master reset controller on the motherboard senses this line and holds all the boards in reset (by driving nSRST LOW) until all the FPGAs are configured. It is essential that a pull-up is added to the FPGA input/output pad during synthesis.</td>
</tr>
</tbody>
</table>

Note

This note refers to Xilinx types only. The signal naming on this logic module differs slightly from the rest of the Integrator system. The logic module provides separate LOCAL_DONE and GLOBAL_DONE signals to make the scope of each signal clear. When GLOBAL_DONE reaches the EXPB connector it is known as FPGADONE to the rest of the Integrator system.
3.6 Memory

The logic module provides 1MB of ZBT SSRAM and 4MB of flash memory.

3.6.1 SSRAM

A 256K x 32-bit ZBT-SSRAM (Micron part number MT55LC256K32F) is provided with address, data, and control signals routed to the FPGA. The address and data lines to the SSRAM are completely separate from the AMBA buses.

3.6.2 Flash memory

This is used for FPGA configuration, and must not be used for any other purpose. Configuration is managed by the configuration PLD.
3.7 LEDs and switches

This section describes the LEDs and switches on the logic module.

3.7.1 LEDs

There are eight general-purpose green LEDs. These are lit by driving the associated LED output pin LOW. A red LED is also provided to indicate a user-defined error condition.

The Example 2 FPGA configuration supplied with the logic module provides the register LM_LEDS to control the LEDs (see User LEDs control register on page 6-9).

The location of the LEDs is illustrated in Figure 1-1 on page 1-3.

3.7.2 DIP switches

The logic module provides two DIP switches:

- a 4-way DIP switch used to select the FPGA image
- an 8-way DIP switch on the module that is provided for user-defined operation

The configuration PLD monitors the 4-way DIP switch and uses the settings to select an FPGA image from the flash memory. See Configuring the Altera FPGA from flash on page 4-7 or Configuring the Xilinx FPGA from flash on page 5-6.

The Example 2 FPGA configuration supplied with the logic module provides the register LM_SW to allow you to read the settings of the 8-way switch (see Switches register on page 6-10).

Note

The FPGA pins that are used to monitor the switches must always be configured as an input or high impedance. This is because the signals are grounded when the switch is in the ON position.

3.7.3 Push button

The push button is general purpose switch that provides an active LOW input to an input/output pin on the FPGA. It can be used, for example, to implement a reset button if the FPGA is configured to supports this.
3.8 Prototyping and expansion

The logic module allows expansion using the interface module connector or the prototyping grid.

3.8.1 Interface module connector

The logic module provides the general-purpose interface module connector EXPIM to enable you to add an interface module to the system. The connector provides access to FPGA input/output banks 0 and 1 (Xilinx) or 5 and 6 (Altera) plus a number of control signals. This facility enables you to add additional hardware, such as interface circuitry and connectors (see EXPIM on page A-10 for pinout details).

Some of the signals to the EXPIM connectors are also routed to the prototyping grid.

3.8.2 Prototyping grid

The prototyping grid consists of an 16 x 17 grid of 0.1 inch pitch plated-through holes. The holes are labelled A to P, and 0 to 16.

The holes on the left side of the grid are connected to FPGA input/output pins from the FPGA and various other signals. Power and ground signals are provided around an area of open circuit holes on the right side. The screen printing on the module indicates the power and ground holes.

You can use the prototyping grid to mount small components, wire to off-board circuitry, or to mount connectors.

All of the signals from FPGA bank 0 (Xilinx) or 5 (Altera) are routed to the prototyping grid so that you can utilize the various input/output standards supported by the FPGA.

3.8.3 Voltage selection

The output voltages for banks 0 and 1 (Xilinx), or 5 and 6 (Altera) are user selectable. LK1 selects the voltage for bank 0 (Xilinx) or 5 (Altera). LK2 selects the voltage for bank 1 (Xilinx) or 6 (Altera). There are three options:

- 3.3V supplied from the logic module (default position)
- 1.8V supplied from the logic module
- user-supplied voltage from prototyping hole G9 for bank 0/5 or H9 for bank 1/6.

The first two options are selected by moving the soldered link to the appropriate position, indicated by the screen printing on the module. The third option is selected by removing the link completely (see Table A-5 on page A-11).
3.9 Debug connectors

The logic module provides a logic analyzer connector and a Trace connector.

3.9.1 Logic analyzer connector

A 38-way Mictor connector is provided for debugging or monitoring purposes. Two 16-bit channels and clocks are routed directly to FPGA pins. These input/output pins are shared with the prototyping area.

3.9.2 Trace connector

The trace connector is intended for use with FPGA configurations that implement a synthesized ARM processor core with an Embedded Trace Macrocell (ETM). In this application the logic module is generally used standalone, or as a core module.

A 38-way Mictor connector is used. You can use it to route additional signals to a logic analyzer in a non-ETM FPGA design.
Chapter 4
Configuring Altera Logic Modules

This chapter describes how the Altera FPGA is configured at power-up, the configuration options available, and how to download your own FPGA configurations. It contains the following sections:

- *Altera FPGA configuration system architecture* on page 4-2
- *Altera FPGA tool flow* on page 4-4
- *Configuring the Altera FPGA from flash* on page 4-7
- *Loading new Altera FPGA configurations* on page 4-9
- *Reprogramming the PLD* on page 4-11.
4.1 Altera FPGA configuration system architecture

When the logic module is powered up in user mode, the FPGA loads configuration data to its internal configuration memory. Figure 4-1 on page 4-2 shows the architecture of the FPGA configuration system.

![Figure 4-1 FPGA configuration architecture](image)

The Altera logic module type has three programming/configuration modes that are selected by the S1[4] and by the CONFIG link, as shown in Table 4-1 on page 4-2.

<table>
<thead>
<tr>
<th>S1[4]</th>
<th>CONFIG</th>
<th>Mode selected</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPEN</td>
<td>OUT</td>
<td>Byte streamer mode (user mode)</td>
</tr>
<tr>
<td>OPEN</td>
<td>IN</td>
<td>ByteblasterMV mode</td>
</tr>
<tr>
<td>CLOSED</td>
<td>OUT</td>
<td>Not used</td>
</tr>
<tr>
<td>CLOSED</td>
<td>IN</td>
<td>PLD operates in flash programmer mode</td>
</tr>
</tbody>
</table>
4.1.1 Byte streamer mode

This is the normal operating mode and is selected by setting S1[4] to OPEN and leaving the CONFIG link OUT. In this mode, S1[3:0] are used to determine which configuration image stored in flash is selected (see Configuring the Altera FPGA from flash on page 4-7).

4.1.2 ByteblasterMV mode

This mode is used to download volatile FPGA images using the Altera ByteblasterMV cable (refer to Altera documentation). Select ByteblasterMV mode by setting S1[4] to OPEN and fitting the CONFIG link.

4.1.3 Flash programmer mode

Flash programmer mode is used to download new configurations into the flash memory. Select this mode by setting S1[4] to CLOSED and fitting the CONFIG link.
4.2  Altera FPGA tool flow

Preparing FPGA configuration files entails two steps:

1.  Synthesis.
2.  Place and route.

Figure 4-2 on page 4-4 illustrates the basic tool flow process.
4.2.1 Synthesis

The synthesis stage of the tool flow takes the HDL files (either VHDL, Verilog, or a combination) and compiles them into a netlist targeted at a particular technology. In the case of this type of logic module, the target technology is Altera Apex. There are several synthesis tools available for both Windows and UNIX platforms, that provide support for a variety of programmable logic vendors. Synthesis information is supplied either through a GUI front end or command-line script. The information typically includes:

- a list of HDL files
- the target technology
- required optimization, such as area or delay
- timing and frequency requirements
- required pull-ups or pull-downs on the FPGA input/output pads
- output drive strengths.

Refer to the documentation for your particular software tool for further information.

A common netlist file format produced by synthesis is *Electronic Data Interchange Format* (EDIF) (for example, filename.edf). This file is used by the next stage of the tool flow, which is place and route.

4.2.2 Place and route

Place and route for this logic module type is performed using the Altera Quartus place and route tool. This produces a .rbf file that is used to program the FPGA in flash program mode (see *Flash program mode (Altera only)* on page 3-12. A .sof file is also produced for use with the ByteBlasterMV cable.

The Altera targeted .edf is aimed at a particular device, and takes into account the device size, package type, and speed grade. However, to ensure that Quartus generates a file that operates correctly with the logic module, use the following settings:

1. From the *Processing* menu, select the *Compiler settings* option.

2. Select the *Chips & Devices* tab, and then click the *Device & Pin Options* button. The *Device & Pin Options* dialog is displayed. Do the following (in any order).

   - Select the *General* tab and check the *Enable INIT_DONE output* box.
   - Select the *Configuration* tab, set *Configuration scheme* to *Passive Parallel Synchronous (for flash configuration)*, and set all *Dual-purpose pins usage after configuration* options to OFF.
   - If you are using Multi-ICE to program in flash programing mode, select *Programming* files tab and check the *Raw Binary File* box.
Select the **Reserve all unused pins** tab and set all unused pins as inputs, tri-stated.

Use the default for all other settings.

Signal names from the top-level HDL are mapped onto actual device pins by a user compiler setting file `.csf`. You can also specify the timing requirements within this file.

**Note**

The `pinout.csf` file for the complete logic module FPGA pin allocation is supplied on the CD. This is intended as a starting point for any design, and will need to be edited before using in the place and route process.
4.3 Configuring the Altera FPGA from flash

The FPGA is configured from the flash when S1[4] is set to OPEN and the CONFIG link is OUT. The flash memory has space to store up to four configurations for EP20K600E or up to two configurations for EP20K1000E types. The configuration image is selected when the logic module is powered according to the setting of S1[3] and the CFGSEL[1:0] signals from the motherboard:

- If S1[3] is OPEN, then S1[1] and S1[2] are used to select the image
- If S1[3] is CLOSED and there is motherboard present, the signals CFGSEL[1:0] are used to select the image. If there is no motherboard present, then S1[1] and S1[2] are used to select the image.

Table 4-2 on page 4-7 and Table 4-3 on page 4-7 show the FPGA image selection options.

### Table 4-2 Image selection for the EP20K600E FPGA

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x000000</td>
<td>x</td>
<td>CLOSED</td>
<td>CLOSED</td>
<td>OPEN</td>
<td>OPEN</td>
</tr>
<tr>
<td>1</td>
<td>0x100000</td>
<td>x</td>
<td>CLOSED</td>
<td>OPEN</td>
<td>OPEN</td>
<td>OPEN</td>
</tr>
<tr>
<td>2</td>
<td>0x200000</td>
<td>x</td>
<td>OPEN</td>
<td>CLOSED</td>
<td>OPEN</td>
<td>OPEN</td>
</tr>
<tr>
<td>3</td>
<td>0x300000</td>
<td>x</td>
<td>OPEN</td>
<td>OPEN</td>
<td>OPEN</td>
<td>OPEN</td>
</tr>
</tbody>
</table>

### Table 4-3 Image selection for the EP20K1000E FPGA

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x000000</td>
<td>xx</td>
<td>CLOSED</td>
<td>x</td>
<td>OPEN</td>
<td>OPEN</td>
</tr>
<tr>
<td>1</td>
<td>0x200000</td>
<td>xx</td>
<td>OPEN</td>
<td>x</td>
<td>OPEN</td>
<td>OPEN</td>
</tr>
<tr>
<td>0</td>
<td>0x000000</td>
<td>0x</td>
<td>CLOSED</td>
<td>x</td>
<td>CLOSED</td>
<td>OPEN</td>
</tr>
<tr>
<td>1</td>
<td>0x200000</td>
<td>1x</td>
<td>OPEN</td>
<td>x</td>
<td>CLOSED</td>
<td>OPEN</td>
</tr>
</tbody>
</table>
Note

The switch labels used in the Table 4-2 on page 4-7 and Table 4-3 on page 4-7 refer to the markings on the switch. These correspond with the register bits named \texttt{SW[3:0]} in the configuration examples.

The positions of the switches \texttt{S1[3:1]} have no effect on the flash programming operation, only image selection on power-up.

See \textit{Example 2 programmer's reference} on page 6-5 for a description of the example images stored in flash when the logic module is shipped.
4.4 Loading new Altera FPGA configurations

You can program the FPGA by writing configuration data into the flash memory using Multi-ICE, or directly using the Altera ByteblasterMV cable.

To reconfigure the FPGA, the logic module must be in CONFIG mode. This is enabled by fitting the CONFIG link (J11). The CFGLED is lit as an indication that configure mode is selected.

To enable flash program mode, the CONFIG link must be fitted and S1[4] must be CLOSED.

Note
The CONFIG LED lights when S1[4] is CLOSED. However the CONFIG link must be fitted for flash program mode to work.

4.4.1 Reconfiguring the FPGA directly with JTAG

Use the DOWNLOAD connector (J10) with the Altera ByteblasterMV cable. Refer to Altera documentation for the use and operation of this tool.

4.4.2 Downloading new the FPGA configurations into flash

The flash memory on the logic module configures the FPGA during power-up if the CONFIG link is not fitted and the PLD is in byte streamer mode. The progcards utility is used to program the flash and can be optionally used again to verify the flash image against a bit file. Flash programming requires Progcards 2.00 or later.

Note
Multi-ICE 2.00 and later autodetects the logic module. If you are using Multi-ICE 1.4, you must manually configure the Multi-ICE server. You will need to ensure that the \irlength.arm file in the Multi-ICE directory contains the following lines, as appropriate for the FPGA fitted to your logic module:

\begin{verbatim}
EP20K600E=10 ; for a 600E size FPGA
EP20K1000E=10 ; for a 1000E size FPGA
ARMFLASH=5 ; needed for all FPGA sizes
\end{verbatim}

Refer to the \textit{Multi-ICE User Guide} for further information about using Multi-ICE. Manual configuration files are provided on the CD supplied with the logic module.
To load a new configuration into the FPGA:

1. Produce a `<filename>.rbf` file.
2. Produce a `<filename>.brd` for your design. This is a configuration file for progcards.exe.
3. Put the logic module in flash program mode by fitting the CONFIG link and setting S1[4] to CLOSED.
4. Configure the Multi-ICE server using a configuration file. For example, \LM-ep20k600e\configure\ep20kV600e_flash_program.cfg.

**Note**

The Altera-equipped logic module requires the JTAG TCK signal to operate at 1MHz maximum for flash programming. The Multi-ICE autodetect feature works with this logic module, but you must reduce the clock speed from the 10MHz default. Alternatively, load Multi-ICE with a manual configuration file (for example, ep20k600e_flash_program.cfg). This sets TCK to 1MHz.

5. Run the progcards utility. All .brd files present in the current directory that match the TAP configuration are offered as options.
6. Remove the CONFIG link and set S1[4] to OPEN.
7. Power the logic module down.
8. Power the logic module up again.
4.5 Reprogramming the PLD

The logic module is supplied with the PLD already programmed.

--- Caution ---

You are advised not to reprogram this device with any image other than those provided.

---

Program the PLD as follows:

1. Put the logic module into configuration mode by fitting the CONFIG link (J11) and power-up.

2. Start the Multi-ICE server and load the configuration file for your logic module. For example:
   
   ```
   <CDdrive>:\LM-ep20k600e\configure\ep20k600e_pld_program.cfg
   ```

3. Start a command prompt and move to the directory `\LM-ep20k600e\configure\`

4. Run the `progcards` utility by typing: `progcards <ret>`

5. Choose the required PLD image. If there is only one suitable match for your hardware, programming starts immediately with no menu being displayed.
Chapter 5
Configuring Xilinx Logic Modules

This chapter describes how the Xilinx FPGA is configured at power-up, the configuration options available, and how to download your own FPGA configurations. It contains the following sections:

- Xilinx FPGA configuration system architecture on page 5-2
- Xilinx FPGA tool flow on page 5-4
- Configuring the Xilinx FPGA from flash on page 5-6
- Loading new Xilinx FPGA configurations on page 5-8
- Reprogramming the PLD on page 5-11.
5.1 Xilinx FPGA configuration system architecture

At power-up the FPGA loads configuration data to its internal configuration memory. Figure 5-1 on page 5-2 shows the architecture of the FPGA configuration system.

The FPGA provides two configuration modes and the logic module provides three ways to load configuration data into the FPGA. The configuration modes are selected by the mode pins (M[2:0]) on the FPGA. The values of M1 and M2 are fixed but the M0 is controlled by the CONFIG link, as shown in Table 5-1 on page 5-2.

![Figure 5-1 FPGA configuration architecture](image)

**Figure 5-1 FPGA configuration architecture**

<table>
<thead>
<tr>
<th>M0</th>
<th>Mode selected</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Slave serial mode, CONFIG link fitted</td>
</tr>
<tr>
<td>0</td>
<td>Select MAP mode, CONFIG link removed</td>
</tr>
</tbody>
</table>

**Table 5-1 FPGA programming modes**
Note

The signals \texttt{FD[7:0]} are connected with \texttt{SD[7:0]}. The signals are used for flash data transfers during FPGA configuration and for configuration downloads into flash. At all other times the signals are used for SSRAM transfers. All designs must drive \texttt{FnOE} and \texttt{FnWE} HIGH to inhibit reads and writes to the flash.

### 5.1.1 Select MAP mode

This mode is the normal FPGA configuration mode. The FPGA configuration is loaded from flash memory and the process is managed by the configuration \textit{Programmable Logic Device} (PLD). The flash must contain valid configuration data and the CONFIG link must not be fitted.

The flash memory can store multiple configuration images. The image is selected either by the DIP switch or by \texttt{CFGSEL[1:0]} from the motherboard (see \textit{Configuring the Xilinx FPGA from flash} on page 5-6).

### 5.1.2 Slave serial mode

You can use slave serial mode to configure the FPGA with the XChecker tool (refer to the Xilinx documentation for information about using this tool). This mode is selected by fitting the CONFIG link.

### 5.1.3 Boundary scan programming (JTAG)

You can use the Multi-ICE JTAG port to download configurations when the CONFIG link is fitted (see \textit{Reconfiguring the FPGA directly with JTAG} on page 4-9).
5.2 Xilinx FPGA tool flow

Preparing FPGA configuration files entails two steps:

1. Synthesis.
2. Place and route.

Figure 5-2 illustrates the basic tool flow process.
5.2.1 Synthesis

The synthesis stage of the tool flow takes the HDL files (either VHDL, Verilog, or a combination) and compiles them into a netlist targeted at a particular technology. In the case of Xilinx Virtex, there are several synthesis tools available for both Windows and UNIX platforms, that provide support for a variety of programmable logic vendors.

Synthesis information is supplied either through a GUI front end, or in the form of a command-line script. The information typically includes:
- a list of HDL files
- the target technology
- required optimization, such as area or delay
- timing and frequency requirements
- required pull-ups or pull-downs on the FPGA input/output pads
- output drive strengths.

Refer to the documentation for your particular software tool for further information.

A common netlist file format produced by synthesis is Electronic Data Interchange Format (EDIF) (for example, filename.edf). This file is used by the next stage of the tool flow, place and route.

5.2.2 Place and route

Place and route for this logic module type is performed using Xilinx-specific software. This produces a .bit file that is used to program the FPGA. The .edf file is aimed at a particular device, taking into account the device size, package type, and speed grade.

_____ Note _______

Always specify CCLK as the start up clock for your design. The progcards utility automatically sets the startup clock to the JTAG clock option when you program the FPGA directly. Selecting CCLK ensures that the process always works for download into the FPGA or into flash.

_____ Note _______

Signal names from the top-level HDL are mapped onto actual device pins by a user constraints file .ucf. You can also specify the timing requirements within this file.

_____ Note _______

The pinout.uct file for the complete logic module FPGA pin allocation is supplied on the CD. This is intended as a starting point for any design, and must be edited before use in the place and route process.
5.3 Configuring the Xilinx FPGA from flash

The flash memory has space to store up to four configurations for XCV600E or XCV1000E FPGA types, or up to two configurations for XCV1600E or XCV2000E types. The configuration image is selected when the logic module is powered according to the setting of S1[3] and the CFGSEL[1:0] signals from the motherboard:

- if S1[3] is OPEN, then S1[1] and S1[2] are used to select the image
- If S1[3] is CLOSED and there is motherboard present, the signals CFGSEL[1:0] are used to select the image. If there is no motherboard present, then S1[1] and S1[2] are used to select the image.

Table 5-2 on page 5-6 and Table 5-3 on page 5-6 show the FPGA image selection options.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x000000</td>
<td>xx</td>
<td>CLOSED</td>
<td>CLOSED</td>
<td>OPEN</td>
<td>x</td>
</tr>
<tr>
<td>1</td>
<td>0x100000</td>
<td>xx</td>
<td>CLOSED</td>
<td>OPEN</td>
<td>OPEN</td>
<td>x</td>
</tr>
<tr>
<td>2</td>
<td>0x200000</td>
<td>xx</td>
<td>OPEN</td>
<td>CLOSED</td>
<td>OPEN</td>
<td>x</td>
</tr>
<tr>
<td>3</td>
<td>0x300000</td>
<td>xx</td>
<td>OPEN</td>
<td>OPEN</td>
<td>OPEN</td>
<td>x</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x000000</td>
<td>00</td>
<td>x</td>
<td>x</td>
<td>CLOSED</td>
<td>x</td>
</tr>
<tr>
<td>1</td>
<td>0x100000</td>
<td>01</td>
<td>x</td>
<td>x</td>
<td>CLOSED</td>
<td>x</td>
</tr>
<tr>
<td>2</td>
<td>0x200000</td>
<td>10</td>
<td>x</td>
<td>x</td>
<td>CLOSED</td>
<td>x</td>
</tr>
<tr>
<td>3</td>
<td>0x300000</td>
<td>11</td>
<td>x</td>
<td>x</td>
<td>CLOSED</td>
<td>x</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x000000</td>
<td>xx</td>
<td>CLOSED</td>
<td>x</td>
<td>OPEN</td>
<td>x</td>
</tr>
<tr>
<td>1</td>
<td>0x200000</td>
<td>xx</td>
<td>OPEN</td>
<td>x</td>
<td>OPEN</td>
<td>x</td>
</tr>
<tr>
<td>0</td>
<td>0x000000</td>
<td>0x</td>
<td>CLOSED</td>
<td>x</td>
<td>CLOSED</td>
<td>x</td>
</tr>
<tr>
<td>1</td>
<td>0x200000</td>
<td>1x</td>
<td>OPEN</td>
<td>x</td>
<td>CLOSED</td>
<td>x</td>
</tr>
</tbody>
</table>
Note

The switch labels used in the Table 5-2 on page 5-6 and Table 5-3 on page 5-6 refer to the markings on the switch. These correspond with the register bits named \( S[3:0] \) in the configuration examples.

The positions of the switches have no effect on the flash programming operation, only image selection on power-up.

See Chapter 6 *Supplied FPGA Examples* for a description of the example images stored in flash when the logic module is shipped.
5.4 Loading new Xilinx FPGA configurations

You can program the FPGA in two ways:

- writing configuration data directly to the FPGA using Multi-ICE, or the Xilinx XChecker cable
- writing configuration data to the flash memory using Multi-ICE.

To reconfigure the FPGA, the logic module must be in CONFIG mode. This is enabled by fitting the CONFIG link (J11). The CFGLED is lit as an indication that configure mode is selected.

For a description of CONFIG mode, see Configuration mode on page 3-11.

5.4.1 Reconfiguring the FPGA directly

Using JTAG to program the FPGA is fast, but the configuration is lost when the power supply is removed. Programming takes between 10 and 15 seconds to complete using Multi-ICE on a fast computer (for example, a 400MHz Windows computer).

Using Xilinx XChecker cable

Use the DOWNLOAD connector (J5) with the Xilinx XChecker cable. Refer to Xilinx documentation for the use and operation of this tool.

--- Note ---
A 3.3V voltage adapter must be used with Virtex E devices.
---

Using Multi-ICE

You can reprogram the FPGA using Multi-ICE. A Multi-ICE client application called progcards is provided to read .bit files and configure the FPGA using the Multi-ICE hardware. You must use a board file (.brd) to tell the progcards utility about the method of programming. Examples are provided on the CD supplied with the logic module.

--- Note ---
The progcards utility requires Multi-ICE release 1.4 or later.
---

For a full description of this utility, refer to the document file progcards.pdf on the supplied CD.
To load a new configuration into the FPGA:

1. Produce a `<filename>.bit` file for your design.

2. Produce a `<filename>.brd` for your design. This is a configuration file for `progcards.exe`.

3. Put the logic module in configuration mode by fitting the CONFIG link.

4. Configure the Multi-ICE server using a configuration file. For example, `LMXCV2000e.cfg`.

5. Run the `progcards` utility. All `.brd` files present in the current directory that match the TAP configuration are offered as options.

Note

Multi-ICE 2.00 and later autodetects the logic module. If you are using Multi-ICE 1.4, you must manually configure the Multi-ICE server. Ensure that the `irlength.arm` file in the Multi-ICE directory contains one of the following lines, as appropriate for the FPGA fitted to your logic module:

```
XCV600E=5 ;for 600E size FPGA
XCV1000E=5 ;for 1000E size FPGA
XCV1600E=5 ;for 1600E size FPGA
XCV2000E=5 ;for 2000E size FPGA
```

Refer to the *Multi-ICE User Guide* for further information about using Multi-ICE. Manual configuration files are provided on the CD supplied with the logic module.

### 5.4.2 Downloading new the FPGA configurations into FLASH

The flash memory on the logic module configures the FPGA on power-up when the CONFIG link is not fitted. The `progcards` utility is used to program the flash. It first loads a flash programmer design into the FPGA, then writes the bit file to the flash memory. You can use the `progcards` utility to verify the flash image against a bit file.

The steps in writing a bit file to flash are similar to those described in *Reconfiguring the FPGA directly* on page 5-8. The only difference is the contents of the `.brd` file (examples are provided on the CD).

To load the FPGA configuration from flash:

1. Remove the CONFIG link.
2. Power the logic module down.
3. Power the logic module up again.
5.5 Reprogramming the PLD

The logic module is supplied with the PLD already programmed.

--- Caution ---

You are advised not to reprogram this device with any image other than those provided.

Program the PLD as follows:

1. Put the logic module into configuration mode by fitting the CONFIG link (J11) and power-up.

2. Start the Multi-ICE server and the load the configuration file for your logic module. For example: `<CDdrive>:\LM-XCV600e\configure\LXCV2000e.cfg`

3. Start a command prompt and move to the directory `<CDdrive>:\LM-XCV600e\configure`

4. Run the progcards utility.

5. Choose the required PLD image. If there is only one suitable match for your hardware, programming starts immediately with no menu being displayed.
Chapter 6
Supplied FPGA Examples

This chapter describes the FPGA configurations supplied with the logic module. It contains the following sections:

- About the FPGA configuration examples on page 6-2
- Example 2 programmer’s reference on page 6-5.
6.1 About the FPGA configuration examples

The logic module is supplied with two example FPGA configurations for each FPGA type. These are supplied to allow you to gain experience with synthesis, design, and place and route for the logic module. VHDL and Verilog versions of the examples are provided.

6.1.1 Example 1

Example 1 provides an entry to designing with the logic module as a standalone platform. It is intended to verify that the correct methods are being used for synthesis, place and route, and programming. This example flashes the LEDs, with frequency controlled by S2[1:0], and places a binary count on the logic analyzer connector.

6.1.2 Example 2

Two versions of Example 2 are provided to support the following implementations:

- AHB motherboard and AHB peripherals
- ASB motherboard and AHB peripherals

Note

The two versions of Example 2 are preloaded into flash (see Configuring the Altera FPGA from flash on page 4-7 for Altera types or Configuring the Xilinx FPGA from flash on page 5-6).

All versions of Example 2 are intended for use with the Integrator/AP motherboard with the logic module fitted in the expansion position. The interrupt request signal from the logic module is routed to the interrupt controller on the Integrator/AP.

The Example 2 configurations are built from common blocks with a top level specific to the bus implementation. Each version includes:

- a ZBT SSRAM controller
- an AHB to APB bridge
- an APB register peripheral
- an APB interrupt controller
- an address decoder.

Figure 6-1 on page 6-3 shows the examples without a system bus bridge.
Figure 6-1 Example without an ASB to AHB bridge

Figure 6-2 on page 6-3 shows the example with a system bus bridge.

Figure 6-2 Example with an ASB to AHB bridge
Table 6-1 on page 6-4 provides a summary description of the supplied HDL files. A more detailed description of each HDL block is included within the files in the form of comments.

### Table 6-1 HDL file descriptions

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASBAHBTop</td>
<td>These files are the top-level HDL that instantiate all of the high-speed peripherals, decoder, and all necessary support and glue logic to make a working system. The files are named so that, for example, ASBAHBTop.vhd is the top level for AHB peripherals connected to an ASB system bus.</td>
</tr>
<tr>
<td>AHBASBTop</td>
<td></td>
</tr>
<tr>
<td>ASB2AHB</td>
<td>This is the bridge required to connect AHB peripherals to an ASB Integrator system.</td>
</tr>
<tr>
<td>AHBDecoder</td>
<td>The decoder block provides the high-speed peripherals with select lines. These are generated from the address lines and the module ID (position in stack) signals from the motherboard. The decoder blocks also contain the default slave peripheral to simplify the example structure. The Integrator family of boards uses a distributed address decoding system (see Example 2 memory map on page 6-6).</td>
</tr>
<tr>
<td>AHBMuxS2M</td>
<td>This is the AHB multiplexor that connects the read data buses from all of the slaves to the AHB master(s).</td>
</tr>
<tr>
<td>AHB2BTRAM</td>
<td>High-speed peripherals require that SSRAM controller block supports word, halfword, and byte operations to the SSRAM on the logic module.</td>
</tr>
<tr>
<td>AHB2APB</td>
<td>This is the bridge blocks required to connect APB peripherals to the high-speed AMBA AHB bus. They produce the peripheral select signals for each of the APB peripherals.</td>
</tr>
<tr>
<td>AHBAPBSys</td>
<td>The components required for an APB system are instantiated in this block. These include the bridge and the APB peripherals. This file also multiplexes the APB peripheral read buses and concatenates the interrupt sources to feed into the interrupt controller peripheral.</td>
</tr>
<tr>
<td>APBRegs</td>
<td>The APB register peripheral provides memory-mapped registers that you can use to: • configure the two clock generators (protected by the LM_LOCK register) • write to the user LEDs • read the user switch inputs. It also latches the pressing of the push button to generate an expansion interrupt.</td>
</tr>
<tr>
<td>APBIntcon</td>
<td>The APB interrupt controller contains all of the standard interrupt controller registers and has an input port for four APB interrupts. (The example only uses one of them. The remaining three are set inactive in the AHBAPBSys block.) Four software interrupts are implemented.</td>
</tr>
</tbody>
</table>

---

**Note**

The files listed in Table 6-1 on page 6-4 are supplied in both Verilog (.v) and VHDL (.vhd) versions.
6.2 Example 2 programmer’s reference

The software sources and precompiled .axf file for this example demonstrate the operation of the SSRAM controller and APB peripherals. They are common for both versions of Example 2.

There are separate project files for both the Software Development Toolkit v2.51 and the ARM Developer Suite v1.0.1 and above.

6.2.1 Software description

There are four source files included in Example 2:

- logic.c The main C code.
- logic.h Constants.
- platform.h Constants.
- rw_support.s Assembler functions for SSRAM testing.

After the FPGAs have been configured, indicated by the FPGA_OK LED being lit, you can download and execute the example software on the core module.

The example code operates as follows:

1. Determines DRAM size on the core module and sets up the system controller.
2. Checks that the logic module is present in the AP expansion position.
3. Reports module information.
4. Sets the logic module clock frequencies.
5. Tests SSRAM for word, halfword, and byte accesses.
6. Flashes the LEDs.
7. Remains in a loop that displays the switch value on the LEDs.
Example 2 sets up the memory map for the logic module as shown in Figure 6-3 on page 6-6. This shows the locations to which logic modules are assigned by the main address decoder on the motherboard. The diagram also shows how Example 2 decodes the address space for the logic module when it is LM0.

**Note**

The Integrator system implements a distributed address decoding scheme in which each core or logic module is responsible for decoding its own address space. It is important when implementing a logic module design, to ensure that the module responds to all memory accesses in the appropriate memory region (see System bus interface on page 3-3).
6.2.3 Example 2 APB register peripheral

Table 6-2 on page 6-7 shows the mapping of the logic module registers. The addresses shown are offsets from the base addresses shown in Figure 6-3 on page 6-6.

Table 6-2 Logic module registers

<table>
<thead>
<tr>
<th>Offset address</th>
<th>Name</th>
<th>Type</th>
<th>Size</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000000</td>
<td>LM_OSC1</td>
<td>Read/write</td>
<td>19</td>
<td>Oscillator divisor register 1</td>
</tr>
<tr>
<td>0x00000004</td>
<td>LM_OSC2</td>
<td>Read/write</td>
<td>19</td>
<td>Oscillator divisor register 2</td>
</tr>
<tr>
<td>0x00000008</td>
<td>LM_LOCK</td>
<td>Read/write</td>
<td>17</td>
<td>Oscillator lock register</td>
</tr>
<tr>
<td>0x0000000C</td>
<td>LM_LEDS</td>
<td>Read/write</td>
<td>9</td>
<td>User LEDs control register</td>
</tr>
<tr>
<td>0x00000010</td>
<td>LM_INT</td>
<td>Read/write</td>
<td>1</td>
<td>Push button interrupt register</td>
</tr>
<tr>
<td>0x00000014</td>
<td>LM_SW</td>
<td>Read</td>
<td>8</td>
<td>Switches register</td>
</tr>
</tbody>
</table>

Oscillator divisor registers

The oscillator registers control the frequency of the clocks generated by the two clock generators (see Clock control on page 3-4).

Before writing to the oscillator registers, you must unlock them by writing the value 0x0000A05F to the LM_LOCK register. After writing the oscillator register, relock them by writing any value other than 0x0000A05F to the LM_LOCK register.
Table 6-3 on page 6-8 describes the oscillator register bits.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Function</th>
<th>Default</th>
</tr>
</thead>
</table>
| 18:16 | OD | Read/write | Output divider: 000 = divide by 10  
001 = divide by 2  
010 = divide by 8  
011 = divide by 4  
100 = divide by 5  
101 = divide by 7  
110 = divide by 9  
111 = divide by 6. | 110 |
| 15:9 | RDW | Read/write | Reference divider word. Defines the binary value of the R[6:0] pins of the clock generator. | 0111110 |
| 8:0 | VDW | Read/write | VCO divider word. Defines the binary value of the V[8:0] pins of the clock generator. | 000000100 |

**Note**

The reset value of this register sets the oscillators to 1MHz.

For information about setting the clock frequency, see *Clock control* on page 3-4.
Oscillator lock register

The lock register is used to control access to the oscillator registers, allowing them to be locked and unlocked. This mechanism prevents the oscillator registers from being overwritten accidently. Table 6-4 on page 6-9 describes the lock register bits.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>LOCKED</td>
<td>Read</td>
<td>This bit indicates if the oscillator registers are locked or unlocked:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 = unlocked</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 = locked.</td>
</tr>
<tr>
<td>15:0</td>
<td>LOCKVAL</td>
<td>Read/write</td>
<td>Write the value 0x0000A05F to this register to enable write accesses</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>to the oscillator registers.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Write any other value to this register to lock the oscillator registers.</td>
</tr>
</tbody>
</table>

User LEDs control register

The LEDs register is used to control the user LEDs (see LEDs summary on page 1-6). Writing a 0 to a bit lights the associated LED.

Push button interrupt register

The push button interrupt register contains 1 bit. It is a latched indication that the push button has been pressed. The output from this register is used to drive an input to the interrupt controller. Table 6-5 on page 6-9 describes the operation of this register.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>LM_INT</td>
<td>Read</td>
<td>This bit when SET is a latched indication that the push button has been pressed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Write</td>
<td>Write 0 to this register to CLEAR the latched indication.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Writing 1 to this register has the same effect as pressing the push button.</td>
</tr>
</tbody>
</table>
Switches register

This register is used to read the setting of the 8-way DIP switch. A 0 indicates that the associated switch element is CLOSED (ON).

6.2.4 Example 2 interrupt controller

The interrupt control registers are listed in Table 6-6 on page 6-10.

Table 6-6 Interrupt controller registers

<table>
<thead>
<tr>
<th>Register name</th>
<th>Address offset</th>
<th>Access</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LM_ISTAT</td>
<td>0x1000000</td>
<td>Read</td>
<td>8 bits</td>
<td>Interrupt status register</td>
</tr>
<tr>
<td>LM_IRSTAT</td>
<td>0x1000004</td>
<td>Read</td>
<td>8 bits</td>
<td>Interrupt raw status register</td>
</tr>
<tr>
<td>LM_IENSET</td>
<td>0x1000008</td>
<td>Read/write</td>
<td>8 bits</td>
<td>Interrupt enable set</td>
</tr>
<tr>
<td>LM_IENCLR</td>
<td>0x100000C</td>
<td>Write</td>
<td>8 bits</td>
<td>Interrupt enable clear</td>
</tr>
<tr>
<td>LM_SOFTINT</td>
<td>0x1000010</td>
<td>Write</td>
<td>4 bits</td>
<td>Software interrupt register</td>
</tr>
</tbody>
</table>

The interrupt controller provides three registers for controlling and handling interrupts. These are:

- status register
- raw status register
- enable register, which is accessed using the enable set and enable clear locations.

The way that the interrupt enable, clear, and status bits function for each interrupt is illustrated in Figure 6-4 on page 6-11 and described in the following subsections. This figure shows the control for one interrupt bit. The logic module interrupts are routed to the system interrupt controller on the motherboard to one of the EXPINT[3:0] interrupts, depending on the position of the logic module in the stack.
**Interrupt status register**

The status register contains the logical AND of the bits in the raw status register and the enable register.

**Interrupt raw status register**

The raw status register indicates the signal levels on the interrupt request inputs. A bit set to 1 indicates that the corresponding interrupt request is active.

**Interrupt enable set**

Use the enable set locations to set bits in the enable register as follows:
- Set bits in the enable register by writing to the ENSET location:
  - 1 = SET the bit
  - 0 = leave the bit unchanged.
- Read the current state of the enable bits from the ENSET location.

**Interrupt enable clear**

Use the clear set locations to set bits in the enable register as follows:
- Clear bits in the enable register by writing to the ENCLR location:
  - 1 = CLEAR the bit
0 = leave the bit unchanged.

**Software interrupt register**

This register is used by software to generate interrupts.

**Interrupt register bit assignment**

The bit assignments for the status, raw status, and enable registers are shown in Table 6-7 on page 6-12.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:5</td>
<td>-</td>
<td>Spare</td>
</tr>
<tr>
<td>4</td>
<td>PBINT</td>
<td>Push button interrupt</td>
</tr>
<tr>
<td>3:0</td>
<td>SOFTINT[3:0]</td>
<td>Interrupt generated by writing to the LM_SOFTINT location</td>
</tr>
</tbody>
</table>
Appendix A
Signal Descriptions

This appendix describes the Integrator/LM interface connectors and signal connections. It contains the following sections:

- **EXPA** on page A-2
- **EXPB** on page A-5
- **EXPIM** on page A-10
- **Diagnostic connectors** on page A-13.
Figure A-1 on page A-2 shows the pin numbers of the EXPA plug and socket. All pins on the EXPA socket are connected to the corresponding pins on the EXPA plug.
A.1.1 AHB signal descriptions

The signals present on the pins labeled A[31:0], B[31:0], C[31:0], and D[31:0] are described in Table A-1 on page A-3 for an AHB system bus.

Table A-1 Bus bit assignment (for an AMBA AHB bus)

<table>
<thead>
<tr>
<th>Pin label</th>
<th>Signal (AHB)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A[31:0]</td>
<td>HADDR[31:0]</td>
<td>System address bus</td>
</tr>
<tr>
<td>B[31:0]</td>
<td>Not used</td>
<td>-</td>
</tr>
<tr>
<td>C[31:16]</td>
<td>Not used</td>
<td>-</td>
</tr>
<tr>
<td>C15</td>
<td>HMASTLOCK</td>
<td>Locked transaction</td>
</tr>
<tr>
<td>C12</td>
<td>HREADY</td>
<td>Slave ready</td>
</tr>
<tr>
<td>C11</td>
<td>HWRITE</td>
<td>Write transaction</td>
</tr>
<tr>
<td>C[1:0]</td>
<td>HTRANS[1:0]</td>
<td>Transaction type</td>
</tr>
<tr>
<td>D[31:0]</td>
<td>HDATA[31:0]</td>
<td>System data bus</td>
</tr>
</tbody>
</table>
A.1.2 ASB signal description

The signals present on the pins labeled A[31:0], B[31:0], C[31:0], and D[31:0] are described in Table A-2 on page A-4 for an ASB system.

Table A-2 Bus bit assignment (for an AMBA ASB bus)

<table>
<thead>
<tr>
<th>Pin label</th>
<th>Signal (ASB)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A[31:0]</td>
<td>BA[31:0]</td>
<td>System address bus</td>
</tr>
<tr>
<td>B[31:0]</td>
<td>Not used</td>
<td>-</td>
</tr>
<tr>
<td>C[31:16]</td>
<td>Not used</td>
<td>-</td>
</tr>
<tr>
<td>C15</td>
<td>BLOK</td>
<td>Locked transaction</td>
</tr>
<tr>
<td>C14</td>
<td>BLAST</td>
<td>Last response</td>
</tr>
<tr>
<td>C13</td>
<td>BERROR</td>
<td>Error response</td>
</tr>
<tr>
<td>C12</td>
<td>BWAIT</td>
<td>Wait response</td>
</tr>
<tr>
<td>C11</td>
<td>BWRITE</td>
<td>Write transaction</td>
</tr>
<tr>
<td>C10</td>
<td>Not used</td>
<td>-</td>
</tr>
<tr>
<td>C7</td>
<td>Not used</td>
<td>-</td>
</tr>
<tr>
<td>C4</td>
<td>Not used</td>
<td>-</td>
</tr>
<tr>
<td>C[1:0]</td>
<td>BTRAN[1:0]</td>
<td>Transaction type</td>
</tr>
<tr>
<td>D[31:0]</td>
<td>BD[31:0]</td>
<td>System data bus</td>
</tr>
</tbody>
</table>
A.2 EXPB

The EXPB plug and socket have slightly different pinouts. A signal rotation scheme is used to route some of the signals to specific logic modules (see Through-board signal routing on page A-7).

A.2.1 EXPB socket pinout

Figure A-2 on page A-5 shows the pin numbers of the socket EXPB on the underside of the logic module.
A.2.2 EXPB plug pinout

Figure A-3 on page A-6 shows the pin numbers of the EXPB plug on the top of the logic module.
A.2.3 Through-board signal routing

The signals on the pins labeled H[31:0] are cross-connected between the plug and socket so that the signals are rotated through the stack in groups of four. This ensures that each module in the stack connects to one specific signal in each group. For example, the first block of four are connected as shown in Table A-3 on page A-7.

<table>
<thead>
<tr>
<th>Plug</th>
<th>Connects to</th>
<th>Socket</th>
</tr>
</thead>
<tbody>
<tr>
<td>H0</td>
<td>connects to</td>
<td>H1</td>
</tr>
<tr>
<td>H1</td>
<td>connects to</td>
<td>H2</td>
</tr>
<tr>
<td>H2</td>
<td>connects to</td>
<td>H3</td>
</tr>
<tr>
<td>H3</td>
<td>connects to</td>
<td>H0</td>
</tr>
</tbody>
</table>

The signals on the pins labeled F[31:0] and J[16:0] on the socket are routed to the same pins on all modules in the stack and so connect to the corresponding pins on the plug.
A.2.4 EXPB signal descriptions

Table A-4 on page A-8 describes the signals on the pins labeled H[31:0], J[16:0], and F[31:0] for an AHB system bus.

<table>
<thead>
<tr>
<th>Pin label</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>H[31:28]</td>
<td>SYSCLK[3:0]</td>
<td>System clock to each core logic module</td>
</tr>
<tr>
<td>H[27:24]</td>
<td>nEPRES[3:0]</td>
<td>Logic module present</td>
</tr>
<tr>
<td>H[23:20]</td>
<td>nIRQSRC[3:0]</td>
<td>Interrupt request from logic module 3, 2, 1, and 0 respectively</td>
</tr>
<tr>
<td>H[19:16]</td>
<td></td>
<td>Not connected</td>
</tr>
<tr>
<td>H[15:12]</td>
<td>ID[3:0]</td>
<td>Logic module stack position indicator</td>
</tr>
<tr>
<td>H[11:8]</td>
<td>SLOCK[3:0]</td>
<td>System bus lock from processor 3, 2, 1, and 0 respectively (not used in ASB).</td>
</tr>
<tr>
<td>H[3:0]</td>
<td>SREQ[3:0]</td>
<td>System bus request</td>
</tr>
<tr>
<td>J16</td>
<td>nRTCKEN</td>
<td>RTCK AND gate enable</td>
</tr>
<tr>
<td>J13</td>
<td>nCFGEN</td>
<td>Sets motherboard into configuration mode</td>
</tr>
<tr>
<td>J12</td>
<td>nSRST</td>
<td>Multi-ICE reset (open collector)</td>
</tr>
<tr>
<td>J11</td>
<td>FPGADONE</td>
<td>Indicates when FPGA configuration is complete (open collector)</td>
</tr>
<tr>
<td>J10</td>
<td>RTCK</td>
<td>Returned JTAG test clock</td>
</tr>
<tr>
<td>J9</td>
<td>nSYSRST</td>
<td>Buffered system reset</td>
</tr>
<tr>
<td>J8</td>
<td>nTRST</td>
<td>JTAG reset</td>
</tr>
<tr>
<td>J7</td>
<td>TDO</td>
<td>JTAG test data out. The TDO signal is routed through devices on each board as it passes up through the stack. For a description of how the JTAG signals are routed, see JTAG support on page 3-11. For a description of the JTAG signals, see JTAG signal descriptions on page 3-17.</td>
</tr>
</tbody>
</table>
J6  TDI  JTAG test data in. The TDI signals is routed straight down to the motherboard at the bottom of the stack. From the motherboard it becomes TDO.
For a description of how the JTAG signals are routed, see JTAG support on page 3-11. For a description of the JTAG signals, see JTAG signal descriptions on page 3-17.

J5  TMS  JTAG test mode select

J4  TCK  JTAG test clock. The TCK signal is routed through devices on each module as it passes down through the stack.
For a description of how the JTAG signals are routed, see JTAG support on page 3-11. For a description of the JTAG signals, see JTAG signal descriptions on page 3-17.

J[3:1]  MASTER[2:0]  Master ID. Binary encoding of the master currently performing a transfer on the bus. Corresponds to the module ID and to the HBUSREQ and HGRANT line numbers.

J0  nMBDET  Motherboard detect pin

F[31:0]/GPIO [31:0]  If the logic module is mounted in the EXPA/EXPB position on an Integrator/AP, these pins connect to the GPIO bus on the Integrator/AP. This bus is routed between the system controller FPGA on the motherboard and the FPGA on the logic module. These signals are available for your own applications.
If the logic module is mounted in the HDRA/HDRB position on the motherboard, these pins connect to the F bus that is routed between any modules in the stack. There are no signals from the motherboard present on these pins.
A.3 EXPIM

These connectors are the same type as those used for EXPA. Figure A-4 on page A-10 shows the pin numbers for EXPIM.
These connector provides expansion connections to the FPGA and to some of the plated through holes in the prototyping area. Signals are routed as follows:

- FPGA connections are routed to the same pins on the plug and the socket
- connections to the prototyping area are routed to pins on the plug only
- a number of fixed and configurable power supply rails are routed to pins on the plug only.

Table A-5 on page A-11 shows the signals for both Altera and Xilinx logic module types.

<table>
<thead>
<tr>
<th>Label</th>
<th>Altera</th>
<th>Xilinx</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXP85</td>
<td>nPOR</td>
<td>nPOR</td>
<td>Power on reset (plug). Not connected (socket).</td>
</tr>
<tr>
<td>EXP87</td>
<td>nPBUTT</td>
<td>nPBUTT</td>
<td>Push button S3 (plug). Not connected (socket).</td>
</tr>
<tr>
<td>EXP88</td>
<td>FAST2</td>
<td>FAST2</td>
<td>FAST2 input to the FPGA.</td>
</tr>
<tr>
<td>EXP89</td>
<td>IM_PROTO0</td>
<td>IM_PROTO0</td>
<td>Prototyping area A10 (plug). Not connected (socket).</td>
</tr>
<tr>
<td>EXP91</td>
<td>IM_PROTO1</td>
<td>IM_PROTO1</td>
<td>Prototyping area B10 (plug). Not connected (socket).</td>
</tr>
<tr>
<td>EXP92</td>
<td>IM_PROTO2</td>
<td>IM_PROTO2</td>
<td>Prototyping area C10 (plug). Not connected (socket).</td>
</tr>
<tr>
<td>EXP93</td>
<td>IM_CLK</td>
<td>IM_CLK</td>
<td>Clock input to the FPGA</td>
</tr>
<tr>
<td>EXP95</td>
<td>IM_PROTO3</td>
<td>IM_PROTO3</td>
<td>Prototyping area D10 (plug). Not connected (socket).</td>
</tr>
<tr>
<td>EXP96</td>
<td>IM_PROTO4</td>
<td>IM_PROTO4</td>
<td>Prototyping area E10 (plug). Not connected (socket).</td>
</tr>
<tr>
<td>EXP97</td>
<td>VCCO_5</td>
<td>VCCO_0</td>
<td>Configurable voltage power supply rail (plug). Not connected (socket).</td>
</tr>
<tr>
<td>EXP98</td>
<td>VCCO_5</td>
<td>VCCO_0</td>
<td>Configurable voltage power supply rail (plug). Not connected (socket).</td>
</tr>
<tr>
<td>EXP185</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>EXP187</td>
<td>IM_PROTO5</td>
<td>IM_PROTO5</td>
<td>Prototyping area F10 (plug). Not connected (socket).</td>
</tr>
<tr>
<td>EXP188</td>
<td>IM_PROTO6</td>
<td>IM_PROTO6</td>
<td>Prototyping area G10 (plug). Not connected (socket).</td>
</tr>
<tr>
<td>EXP189</td>
<td>CLK1_0</td>
<td>CLK1_0</td>
<td>Clock signal from the CLK1 buffer (plug), see Clock control on page 3-4. Not connected (socket).</td>
</tr>
</tbody>
</table>
### Table A-5 EXPIM signal description (continued)

<table>
<thead>
<tr>
<th>Label</th>
<th>Altera</th>
<th>Xilinx</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXP191</td>
<td>CLK1_1</td>
<td>CLK1_1</td>
<td>Clock signal from the CLK1 buffer (plug), see Clock control on page 3-4. Not connected (socket).</td>
</tr>
<tr>
<td>EXP192</td>
<td>IMPROTO7</td>
<td>IMPROTO7</td>
<td>Prototyping area H10 (plug). Not connected (socket).</td>
</tr>
<tr>
<td>EXP193</td>
<td>IMPROTO8</td>
<td>IMPROTO8</td>
<td>Prototyping area A11 (plug). Not connected (socket).</td>
</tr>
<tr>
<td>EXP195</td>
<td>IMPROTO9</td>
<td>IMPROTO9</td>
<td>Prototyping area B12 (plug). Not connected (socket).</td>
</tr>
<tr>
<td>EXP196</td>
<td>FAST3</td>
<td>IMPROTO10</td>
<td>FAST2 is from the FPGA (plug, Altera only) Prototyping area C12 (plug, Xilinx only) Not connected (socket)</td>
</tr>
<tr>
<td>EXP197</td>
<td>VCCO_6</td>
<td>VCCO_1</td>
<td>Configurable voltage power supply rail (plug). Not connected (socket).</td>
</tr>
<tr>
<td>EXP198</td>
<td>VCCO_6</td>
<td>VCCO_1</td>
<td>Configurable voltage power supply rail (plug). Not connected (socket).</td>
</tr>
</tbody>
</table>
A.4 Diagnostic connectors

This section provides details about the logic analyzer and Trace connectors. Figure A-5 on page A-13 shows the pin numbers of this type of connector.

Figure A-5 Diagnostic connector pin locations
Table A-6 on page A-14 shows the pinout of the Trace type B connector.

<table>
<thead>
<tr>
<th>Channel</th>
<th>Pin</th>
<th>Pin</th>
<th>Channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>No connect</td>
<td>1</td>
<td>2</td>
<td>No connect</td>
</tr>
<tr>
<td>No connect</td>
<td>3</td>
<td>4</td>
<td>No connect</td>
</tr>
<tr>
<td>GND</td>
<td>5</td>
<td>6</td>
<td>TRCCLK</td>
</tr>
<tr>
<td>DBGREQ</td>
<td>7</td>
<td>8</td>
<td>DBGACK</td>
</tr>
<tr>
<td>nSRST</td>
<td>9</td>
<td>10</td>
<td>EXTTRIG</td>
</tr>
<tr>
<td>TDO</td>
<td>11</td>
<td>12</td>
<td>VDD (3.3V)</td>
</tr>
<tr>
<td>RTCK</td>
<td>13</td>
<td>14</td>
<td>VDD (3.3V)</td>
</tr>
<tr>
<td>TCK</td>
<td>15</td>
<td>16</td>
<td>TRCPKT7</td>
</tr>
<tr>
<td>TMS</td>
<td>17</td>
<td>18</td>
<td>TRCPKT6</td>
</tr>
<tr>
<td>TDI</td>
<td>19</td>
<td>20</td>
<td>TRCPKT5</td>
</tr>
<tr>
<td>nTRST</td>
<td>21</td>
<td>22</td>
<td>TRCPKT4</td>
</tr>
<tr>
<td>TRCPKT15</td>
<td>23</td>
<td>24</td>
<td>TRCPKT3</td>
</tr>
<tr>
<td>TRCPKT14</td>
<td>25</td>
<td>26</td>
<td>TRCPKT2</td>
</tr>
<tr>
<td>TRCPKT13</td>
<td>27</td>
<td>28</td>
<td>TRCPKT1</td>
</tr>
<tr>
<td>TRCPKT12</td>
<td>29</td>
<td>30</td>
<td>TRCPKT0</td>
</tr>
<tr>
<td>TRCPKT11</td>
<td>31</td>
<td>32</td>
<td>TRCSYNC</td>
</tr>
<tr>
<td>TRCPKT10</td>
<td>33</td>
<td>34</td>
<td>PIPESTAT2</td>
</tr>
<tr>
<td>TRCPKT9</td>
<td>35</td>
<td>36</td>
<td>PIPESTAT1</td>
</tr>
<tr>
<td>TRCPKT8</td>
<td>37</td>
<td>38</td>
<td>PIPESTAT0</td>
</tr>
</tbody>
</table>
A.4.2 Logic analyzer connector

Table A-7 on page A-15 shows the pinout of the logic analyzer connector J7.

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal</th>
<th>Pin</th>
<th>Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>GND</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>LA_ACLK</td>
<td>6</td>
<td>LA_BCLK</td>
</tr>
<tr>
<td>7</td>
<td>LA_A15</td>
<td>8</td>
<td>LA_B15</td>
</tr>
<tr>
<td>9</td>
<td>LA_A14</td>
<td>10</td>
<td>LA_B14</td>
</tr>
<tr>
<td>11</td>
<td>LA_A13</td>
<td>12</td>
<td>LA_B13</td>
</tr>
<tr>
<td>13</td>
<td>LA_A12</td>
<td>14</td>
<td>LA_B12</td>
</tr>
<tr>
<td>15</td>
<td>LA_A11</td>
<td>16</td>
<td>LA_B11</td>
</tr>
<tr>
<td>17</td>
<td>LA_A10</td>
<td>18</td>
<td>LA_B10</td>
</tr>
<tr>
<td>19</td>
<td>LA_A9</td>
<td>20</td>
<td>LA_B9</td>
</tr>
<tr>
<td>21</td>
<td>LA_A8</td>
<td>22</td>
<td>LA_B8</td>
</tr>
<tr>
<td>23</td>
<td>LA_A7</td>
<td>24</td>
<td>LA_B7</td>
</tr>
<tr>
<td>25</td>
<td>LA_A6</td>
<td>26</td>
<td>LA_B6</td>
</tr>
<tr>
<td>27</td>
<td>LA_A5</td>
<td>28</td>
<td>LA_B5</td>
</tr>
<tr>
<td>29</td>
<td>LA_A4</td>
<td>30</td>
<td>LA_B4</td>
</tr>
<tr>
<td>31</td>
<td>LA_A3</td>
<td>32</td>
<td>LA_B3</td>
</tr>
<tr>
<td>33</td>
<td>LA_A2</td>
<td>34</td>
<td>LA_B2</td>
</tr>
<tr>
<td>35</td>
<td>LA_A1</td>
<td>36</td>
<td>LA_B1</td>
</tr>
<tr>
<td>37</td>
<td>LA_A0</td>
<td>38</td>
<td>LA_B0</td>
</tr>
</tbody>
</table>
A.4.3 Multi-ICE (JTAG)

Figure A-6 on page A-16 shows the pinout of the Multi-ICE connector J12. For a detailed description of the JTAG signals, see JTAG signal descriptions on page 3-17.

Figure A-6 Muti-ICE connctor pinout
Appendix B

Mechanical Specification

This appendix contains the specifications for the logic module. It contains the following section:

- Mechanical details on page B-2.
B.1 Mechanical details

The logic module is designed to be stackable on a number of different motherboards. Figure B-1 on page B-2 shows the mechanical outline of the logic module and indicates the position of pin 1 on the Samtec connectors. (All dimensions are in mm.)
Figure B-2 on page B-3 shows how the pins of the Samtec connectors are numbered.

![Diagram showing pin numbering for Samtec connectors](image_url)

**Figure B-2** Samtec connector pin numbering
# Index

The items in this index are listed in alphabetical order, with symbols and numerics appearing at the end. The references given are to page numbers.

## A
- Adaptive clocking 3-16
- Altera place and route 4-5
- Architecture
  - clock 3-4
  - FPGA configuration 4-2, 5-2
  - reset 3-9
  - system 1-5

## B
- Boundary scan programming 5-3

## C
- Care of modules 1-9
- Clock control 3-4
- Clock signals 3-5
- CONFIG link 1-6, 4-2, 4-9, 5-2
- Configuration

## D
- Altera FPGA 4-2
- Xilinx FPGA 5-2
- Configuration Example 1 6-2
- Configuration mode 3-11
- Configuration modes, FPGA 5-2
- Connecting Multi-ICE 2-6
- Connectors
  - DOWNLOAD 4-9
  - EXPA 2-3
  - EXPB 2-3
  - HDRA 2-3
  - logic analyzer A-15
  - Multi-ICE A-16
  - Trace A-14
  - Control
    - clocks 3-7
    - interrupts 6-11
    - reset 3-9
  - Core and logic module differences 1-8

## D
- Diagnostic connectors A-13
- DIP switches, setting 2-5
- Document confidentiality status ii
- DOWNLOAD connector 4-9

## E
- Electromagnetic conformity ii
- Electronic Data Interchange Format 4-5, 5-5
- Enable register, interrupt 6-11
- Example memory map 6-6
- EXPA connector 2-3
- EXPB connector 2-3

## F
- FCC notice ii
- Fitting modules to motherboard 2-4
Index

Flash memory 3-19
Flash program mode 3-12
FPGA configuration 2-5
    Altera 4-2
    Xilinx 5-2
FPGA configuration examples 6-2
FPGA configuration image selection 5-6
FPGA input/output pins 3-2
FPGA place and route 4-4, 5-4
FPGA programming 4-9
FPGA synthesis 4-4, 5-4
FPGA, description 3-2

H
HDL files 4-5, 5-5
HDRA connectors 2-3, 2-4
HDRA pinout A-2
HDRB connectors 2-4
HDRB plug pinout A-6
HDRB signals A-8
HDRB socket pinout A-5

I
ICSS525 3-5
Indicators 1-6
Input/output, FPGA 3-2
Integrator memory map 6-6
Integrator/LM-XCV400+ layout 1-3, 1-4
Interrupt control 6-11
Interrupt pins 1-8
Interrupt register bit assignment 6-12
Interrupt status 6-11

J
JTAG and the FPGA 3-2
JTAG data path 3-14
JTAG signal descriptions 3-17
JTAG signal routing 3-11, 3-12
JTAG support 3-11
JTAG test reset 3-9
JTAG, with multi-module system 3-13

L
LEDs summary 1-6
Links 1-6, 3-15
Logic analyzer connectors A-15
Logic module base address decodes 3-3
Logic module registers 6-7
Logic modules, attaching 2-3
Logic modules, maximum number 2-4

M
Map mode 5-3
Maximum number of modules 2-4
Memory map, example 6-6
Module fitting 2-4
Modules, combined maximum 2-4
Motherboard configuration signals 2-5
Motherboard reset 3-10
Multi-ICE adaptive clocking 3-16
Multi-ICE connector 3-9
Multi-ICE (JTAG) connector A-16
Multi-ICE, connecting 2-6

N
Notices, FCC ii

O
Oscillator divisor registers 6-7
Oscillator lock register 6-9

P
Physical layout 1-3, 1-4
Pinout
    HDRA A-2
    HDRB plug A-6
    HDRB socket A-5
Pinouts
    logic analyzer connector A-15
    Trace A-14
Place and route 4-4, 5-4

R
Raw status register, interrupt 6-11
Registers
    LM_IENCLR 6-10
    LM_IENSET 6-10
    LM_INT 6-7
    LMIRSTAT 6-10
    LM_INSTAT 6-10
    LMLED 6-7
    LM_LOCK 6-7
    LM_OSC1 6-7
    LM_OSC2 6-7
    LM_SOFTINT 6-10
    LM_SW 6-7
Related publications x
Reset architecture 3-9
Reset control 3-9

S
Select MAP mode 4-2, 5-2
Selecting the FPGA image 5-6
Setting the DIP switches 2-5
Signals
    clocks 3-5
    JTAG 3-17
Slave serial mode 4-2, 5-2, 5-3
SSRAM 3-19
stacking 3-15
Stacking link 3-15
Stacking modules 3-15
Status register 6-11
Switches 1-6
Switches register 6-10
Index

Synthesis  4-5, 5-5
Synthesis, FPGA  4-4, 5-4
System architecture  1-5
System bus interface  3-3
System reset  3-10

T
Through-board signals  A-7
Tool flow
  Altera  4-4
  Xilinx  5-4
Trace  3-9
Trace connector  A-14

U
User LEDs control register  6-9
User mode, JTAG  3-12