Integrator/LM-XCV400+

Logic Module

User Guide

ARM

ARM DUI 0130A

Integrator/LM-XCV400+ User Guide

Release information

Change history

Date	Issue	Change
16 February 2000	A	New document

Proprietary notice

Document confidentiality status

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

Product status

The information in this document is Final (information on a developed product).

ARM web address

`http://www.arm.com`

Contents
Integrator/LM-XCV400+ User Guide

About this document vi
Further reading vii
Feedback ix

Chapter 1 Introduction
1.1 About the ARM Integrator/LM-XCV400+ logic module 1-2
1.2 Architecture overview 1-4
1.3 Links, indicators, and switches 1-5

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 DIL switches 2-5
2.4 Using Multi-ICE or JTAG equipment 2-6
2.5 Differences between core and logic modules 2-7
2.6 Care of modules 2-8

Chapter 3 Hardware Description
3.1 Logic module FPGA 3-2
3.2 Bus interfaces 3-5
3.3 Clock control 3-7
3.4 Reset control 3-10
3.5 JTAG support 3-12

3.6 Memory ... 3-17
3.7 LEDs and switches ... 3-18
3.8 Prototyping grid ... 3-19
3.9 Debug connectors ... 3-20

Chapter 4 Designer's Reference
4.1 Tool flow ... 4-2
4.2 Configuring the FPGA from flash ... 4-4
4.3 Loading new FPGA configurations ... 4-6
4.4 Reprogramming the PLD ... 4-8

Chapter 5 FPGA Configuration Examples
5.1 About the FPGA configuration examples ... 5-2
5.2 Example 2 programmer's reference ... 5-5

Appendix A Signal Descriptions
A.1 Logic module connector EXPA ... A-2
A.2 Logic module connector EXPB ... A-5
A.3 Connector EXPM ... A-7
A.4 Diagnostic connectors ... A-9

Appendix B FPGA Pin Allocation
B.1 Clock signals ... B-2
B.2 LEDs and switches ... B-4
B.3 Diagnostics ... B-5
B.4 SSRAM ... B-8
B.5 Prototyping grid ... B-11
B.6 GPIO (or F BUS) ... B-12
B.7 External Bus Interface (EBI) ... B-13
B.8 ASB/AHB signals ... B-16
B.9 Miscellaneous input/output signals ... B-18
B.10 Modem card signals ... B-20

Appendix C Mechanical Specification
C.1 Mechanical details ... C-2

Preface

This preface introduces the ARM Integrator/LM-XCV400+ logic module and its reference documentation. It contains the following sections:

- *About this document* on page vi
- *Further reading* on page viii
- *Feedback* on page ix.

About this document

This document describes how to set up and use the ARM Integrator/LM-XCV400+ logic module.

Intended audience

This document has been written for experienced hardware and software developers as an aid to developing ARM-based products using the ARM Integrator/LM-XCV400+ as a standalone development system or as part of an Integrator development system.

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 facilities.

Chapter 4 *Designer's Reference*

Read this chapter for a description of how FPGA configurations are selected and how to download new FPGA configurations.

Chapter 5 *FPGA Configuration 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 example as part of an Integrator development system.

Appendix A *Signal Descriptions*

Refer to this appendix for connector pinouts.

Appendix B *FPGA Pin Allocation*

Refer to this appendix for a listing by function of the FPGA input/output pins.

Appendix C *Mechanical Specification*

Refer to this appendix for mechanical details of the logic module.

Typographical conventions

The following typographical conventions are used in this document:

bold Highlights ARM processor signal names within text, and interface elements such as menu names. May also be used for emphasis in descriptive lists where appropriate.

italic Highlights special terminology, cross-references and citations.

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

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

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

`typewriter 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)
- *ARM Software Development Toolkit User Guide* (ARM DUI 0040)
- *ARM Software Development Toolkit Reference Guide* (ARM DUI 0041)
- *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)
- *ADS CodeWarrior IDE Guid*e (ARM DUI 0065).

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.

The following publications provide information and guidelines for developing products for Microsoft Windows CE:

- *Standard Development Board for Microsoft® Windows® CE*, 1998, Microsoft Corporation
- *HARP Enclosure Requirements for Microsoft® Windows® CE*, 1998, Microsoft Corporation.

Further information on these topics is available from the Microsoft web site.

Feedback

ARM Limited welcomes feedback both on the ARM Integrator/LM-XCV400+ logic module 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-XCV400+

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-XCV400+ logic module. It contains the following sections:

- *About the ARM Integrator/LM-XCV400+ logic module* on page 1-2
- *Architecture overview* on page 1-4
- *Links, indicators, and switches* on page 1-5.

1.1 About the ARM Integrator/LM-XCV400+ logic module

The Integrator/LM-XCV400+ logic module is designed as a platform for developing ASB, AHB, and APB peripherals for use with ARM cores. You can use it in three ways:

- as a standalone system, with a bench power supply
- 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.

Figure 1-1 on page 1-3 shows the layout of the logic module.

The logic module is fitted with a Xilinx Virtex FPGA. The basic functionality of the logic module is defined by a configuration which is loaded into the FPGA at power-up. Two FPGA configuration examples are preloaded into flash to get you started. You can download your own configurations using the Xilinx XChecker tool or Multi-ICE.

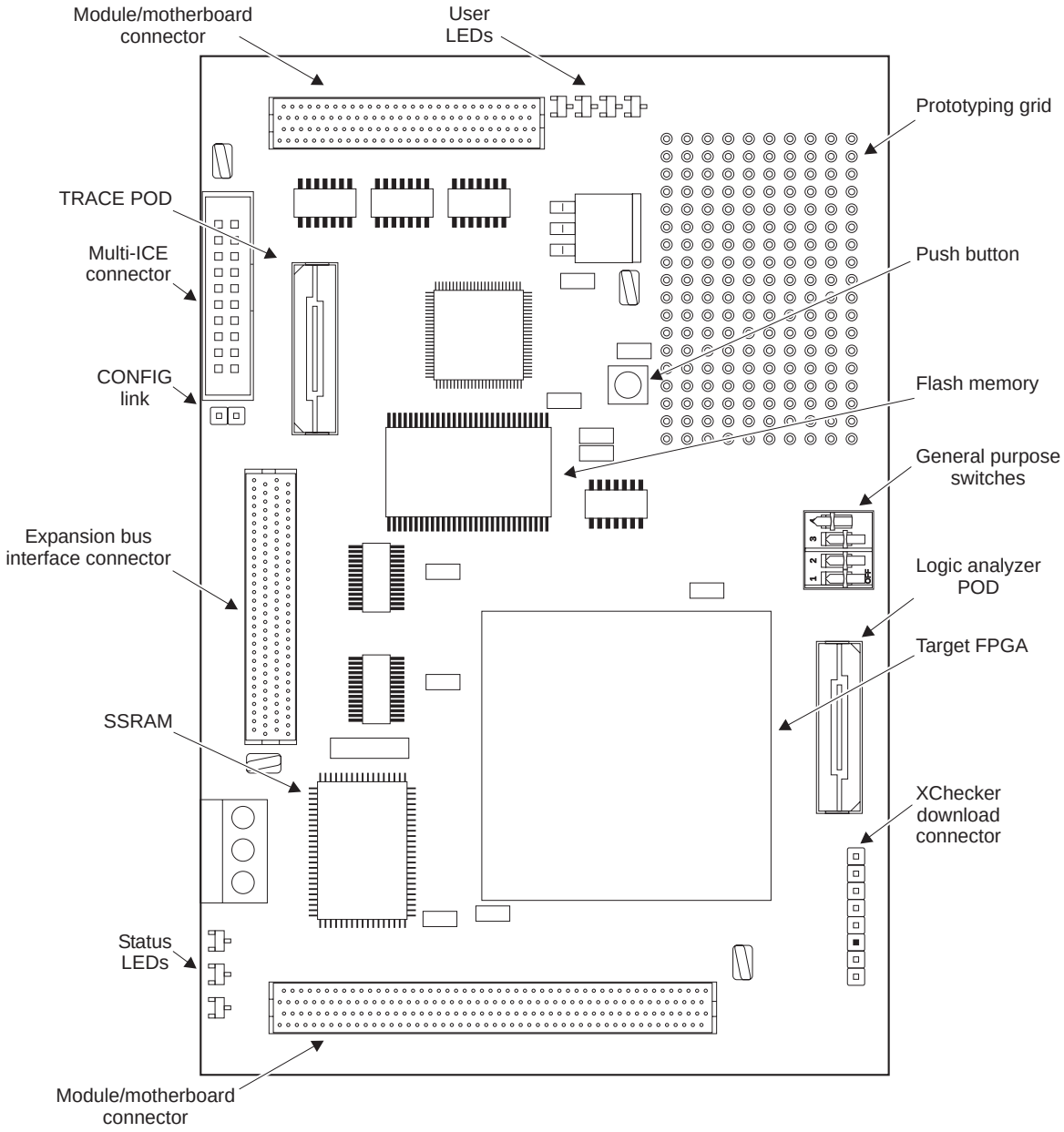


Figure 1-1 Integrator/LM-XCV400+ layout

1.2 Architecture overview

Figure 1-2 shows the architecture of the logic module.

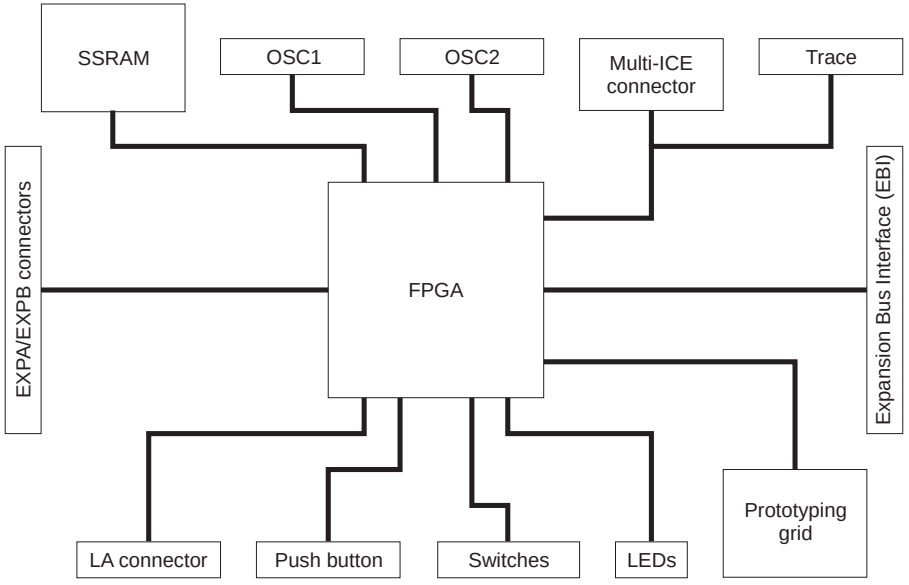


Figure 1-2 System architecture

The logic module comprises the following:

- Xilinx Virtex XCV400, 600, 800, or 1000 FPGA
- configuration PLD and flash memory for storing FPGA configurations
- 256KB SSRAM
- clock generators and reset sources
- switches
- LEDs
- prototyping grid
- JTAG, Trace, and logic analyzer connectors
- system bus connectors to a motherboard and/or core modules and/or logic modules.

These components are discussed in detail in Chapter 3 *Hardware Description*.

1.3 Links, indicators, and switches

The logic module provides:

- one link
- seven surface-mounted LEDs
- one push button switch
- a 4-way DIL switch pack.

You can also add your own links and switches 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 illustrated in Figure 1-3.

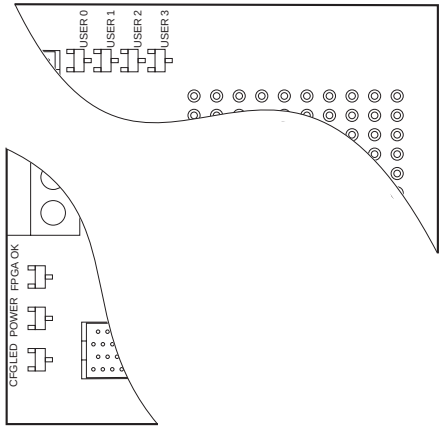


Figure 1-3 LEDs

The logic module provides seven LEDs, as summarized in Table 1-1.

Table 1-1 LED summary

Name	Color	Function
CFGLED	Orange	Configuration mode. This LED is lit when the CONFIG link is fitted.
POWER	Green	Power supply OK. This LED is lit when 3.3V power is supplied to the board.
FPGA_OK	Green	FPGA is configured. This LED is lit when power is supplied and the FPGA has loaded its configuration.
USER[3:0]	Green	User LEDs. These are general purpose LEDs connected to FPGA pins. Drive LOW to light.

1.3.3 Switches

The DIL switch and push button provide general purpose inputs to the FPGA.

The PLD uses two of the switches within the DIL pack to select the FPGA configuration when the logic module is powered up without a motherboard present.

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 DIL switches* on page 2-5
- *Using Multi-ICE or JTAG equipment* on page 2-6
- *Differences between core and logic modules* on page 2-7
- *Care of modules* on page 2-8.

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. 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.

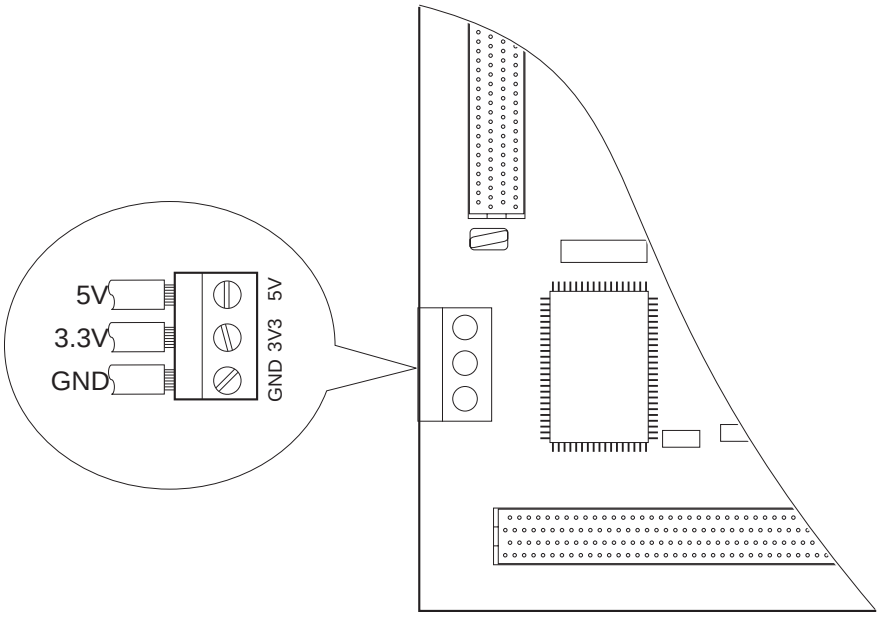


Figure 2-1 Power supply screw terminals

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*
- *Mounting on an Integrator/SP* on page 2-4.

2.2.1 Mounting the logic module on an Integrator/AP

You can mount modules on the Integrator/AP in the following positions:

- logic modules using the connectors EXPA and EXPB
- core modules using the connectors HDRA and HDRB.

Note

Logic modules can be mounted on the HDRA and HDRB connectors. However, the interrupt signals on the HDRB and EXPB are routed differently (see *Differences between core and logic modules* on page 2-7).

Precautions

To prevent damage to the Integrator/AP and modules:

- Power down before fitting or removing core modules or logic modules.
- Do not exceed four core modules or logic modules in one stack.
- Do not exceed a combined total of five core modules or logic modules on the Integrator/AP.

Fitting procedure

Fit a logic module as follows:

1. Place the Integrator/AP on a firm level surface.
2. Align connectors EXPA and EXPB on core 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.

Figure 2-2 on page 2-4 shows four core modules and one logic module attached to an Integrator/AP.

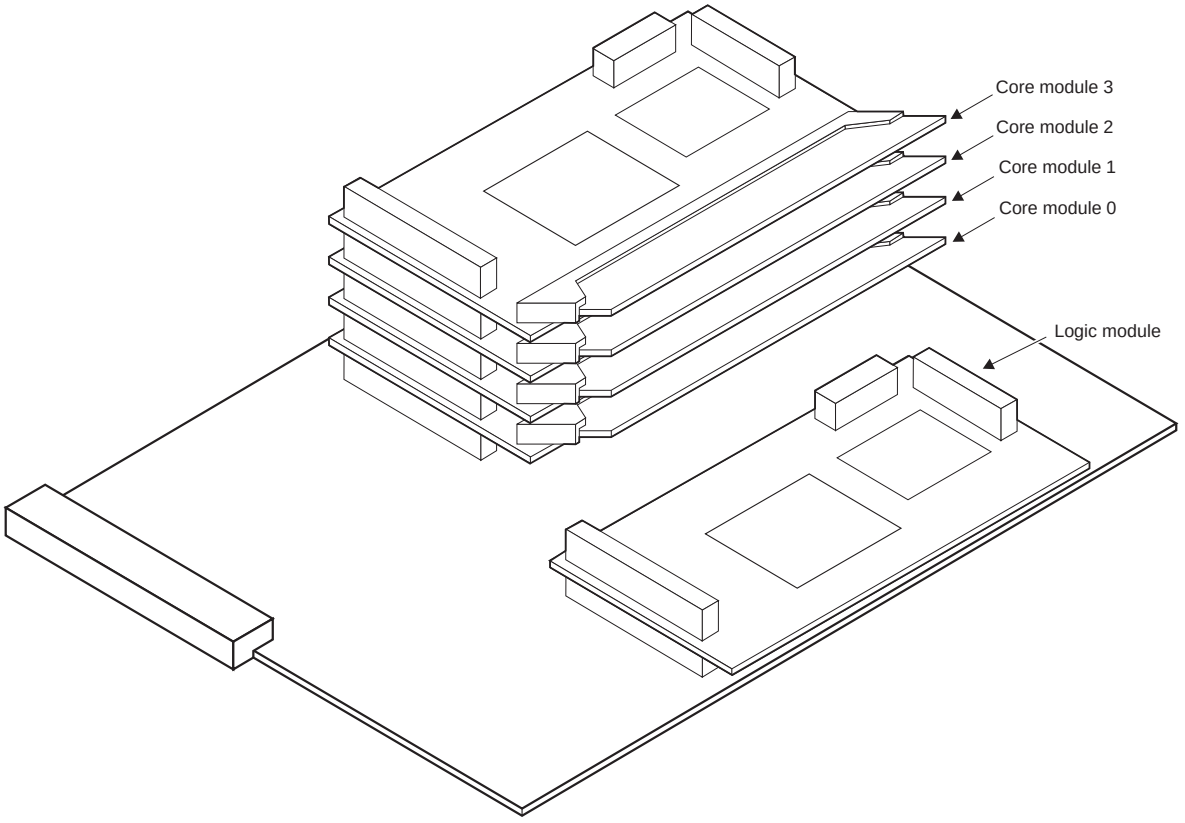


Figure 2-2 Assembled Integrator/AP development system

2.2.2 Mounting on an Integrator/SP

The core modules and logic modules mount in a single stack using the HDRA/EXPA and HDRB/EXPB connectors. You must follow the same precautions and limit the number of modules to four.

Note

Logic modules and core modules use the interrupt signals on the HDRB and EXPB differently (see *Differences between core and logic modules* on page 2-7).

2.3 Setting the DIL switches

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

- On a standalone module, the 4-pole DIL switch (S1), illustrated in Figure 2-3, is used by the preloaded PLD configuration to select the FPGA configuration.

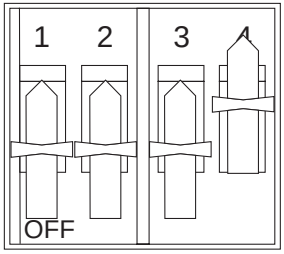


Figure 2-3 DIP switches

- If the logic module is mounted on a motherboard, the signals **CFGSEL [1:0]** from the motherboard are used to select the FPGA configuration. This selects an AHB or ASB example depending on the motherboard configuration.

For a full description of FPGA configuration image selection, see *Configuring the FPGA from flash* on page 4-4.

2.4 Using Multi-ICE or JTAG equipment

JTAG equipment, such as Multi-ICE, is connected to the 20-way box header, as shown in Figure 2-4. 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 *JTAG support* on page 3-12 for a description of the JTAG system.

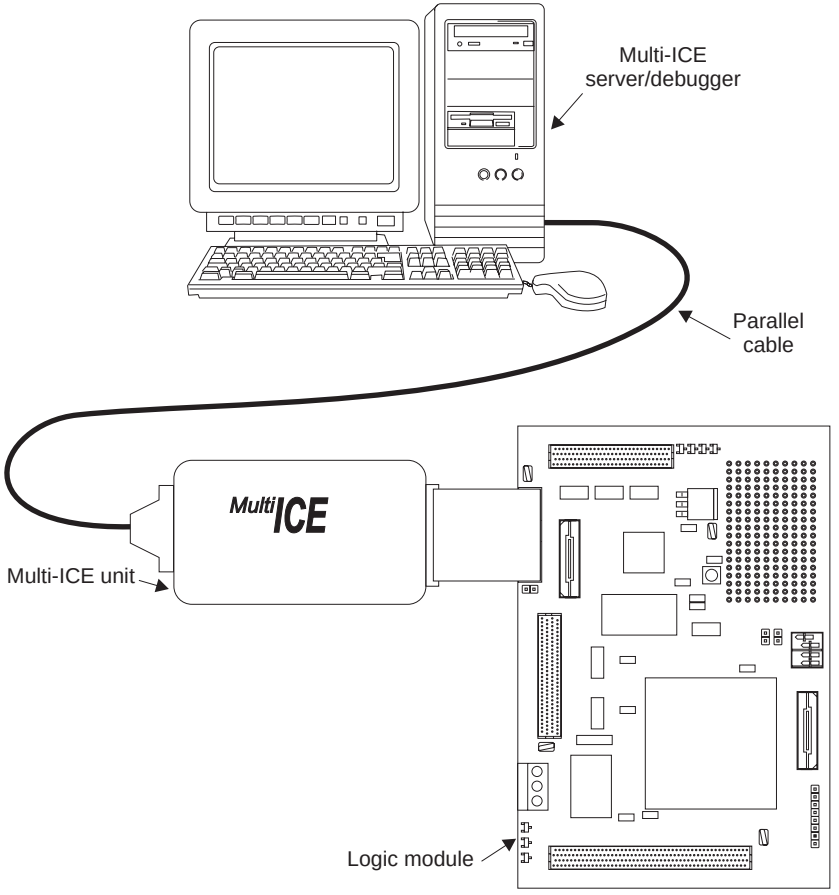


Figure 2-4 Connecting Multi-ICE

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.

2.5 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 2-1. All signals on these pins are 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 2-1 Interrupt Pins on HDRB and EXPB

HDRB	Label	Description	EXPB	Label	Description
nFIQ0	H16	Fast interrupt to module 0	-	E16	Not used
nFIQ1	H17	Fast interrupt to module 1	-	E17	Not used
nFIQ2	H18	Fast interrupt to module 2	-	E18	Not used
nFIQ3	H19	Fast interrupt to module 3	-	E19	Not used
nIRQ0	H20	Interrupt to module 0	**IRQSRC0**	E20	Interrupt source from module 0 to interrupt controller
nIRQ1	H21	Interrupt to module 1	**IRQSRC1**	E21	Interrupt source from module 1 to interrupt controller
nIRQ2	H22	Interrupt to module 2	**IRQSRC2**	E22	Interrupt source from module 2 to interrupt controller
nIRQ3	H23	Interrupt to module 3	**IRQSRC3**	E23	Interrupt source from module 3 to interrupt controller

You can use the logic module to implement a synthesized processor, such as an ARM 7TDMI-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 (otherwise know as the F BUS) lines are routed to the EXPB connector, but not the HDRB connector. The GPIO signals are therefore not available on the Integrator/SP.

2.6 Care of modules

When you remove a core or logic module from a motherboard, or separate modules, take care not to damage the EXPA/EXPB/EXPM connectors. Do not apply a twisting force to the ends of the connectors on removal. Ideally, loosen each side first before pulling the module directly upwards at both ends.

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. It is worth giving the module a quick visual inspection to ensure that connector holes are clear before mounting on another board.

Observe ESD precautions when handling any Integrator PCB.

Chapter 3
Hardware Description

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

- *Logic module FPGA* on page 3-2
- *Bus interfaces* on page 3-5
- *Clock control* on page 3-7
- *Reset control* on page 3-10
- *JTAG support* on page 3-12
- *Memory* on page 3-17
- *LEDs and switches* on page 3-18
- *Prototyping grid* on page 3-19
- *Debug connectors* on page 3-20.

3.1 Logic module FPGA

The logic module is fitted with Xilinx Virtex XCV400, 600, 800, or 1000 FPGA. The assignment of its input/output banks, JTAG implementation, and how the device is configured are described in the following sections:

- *FPGA bank assignment*
- *JTAG and the FPGA* on page 3-3
- *FPGA configuration* on page 3-3.

3.1.1 FPGA bank assignment

The FPGA input/output pins are organized into eight banks. If the logic module is configured for an Integrator motherboard or core module, the function of seven banks are defined as shown in Table 3-1. One bank is assigned to the prototyping grid (see *Prototyping grid* on page 3-19).

Table 3-1 FPGA input/output bank assignment

Bank	Function
0	This bank is connected to the memory expansion connector (EXPM). The supplied configurations set this bank up as a memory interface. If expansion memory is not required, it can be assigned to another function.
1	This bank provides addresses for the on-board SSRAM, LEDs, and memory access control on memory expansion connector (EXPM).
2	These banks provide access to the system bus, which provides the main interface with other Integrator boards. These connect to EXPA.
3	
4	This bank provides a Trace port, divider control for **CLK2** clock generator, and JTAG.
5	Prototyping grid.
6	On-board SSRAM data bus, divider control for **CLK1** clock generator, and reset control signals.
7	GPIO signals to and from the EXPA and EXPB connectors.

Note

For a complete listing of the FPGA connections, see Appendix B *FPGA Pin Allocation*.

3.1.2 JTAG and the FPGA

The FPGA provides 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 virtual TAP controller which you can synthesize into the FPGA configuration. You can use the virtual TAP controller to access a processor synthesized on 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 3-12).

The Example 2 configurations supplied with the logic module include a virtual TAP controller (see *FPGA Configuration Examples* on page 5-1).

3.1.3 FPGA configuration

At power-up the FPGA loads configuration data to its internal configuration memory. Figure 3-1 shows the architecture of the FPGA configuration system.

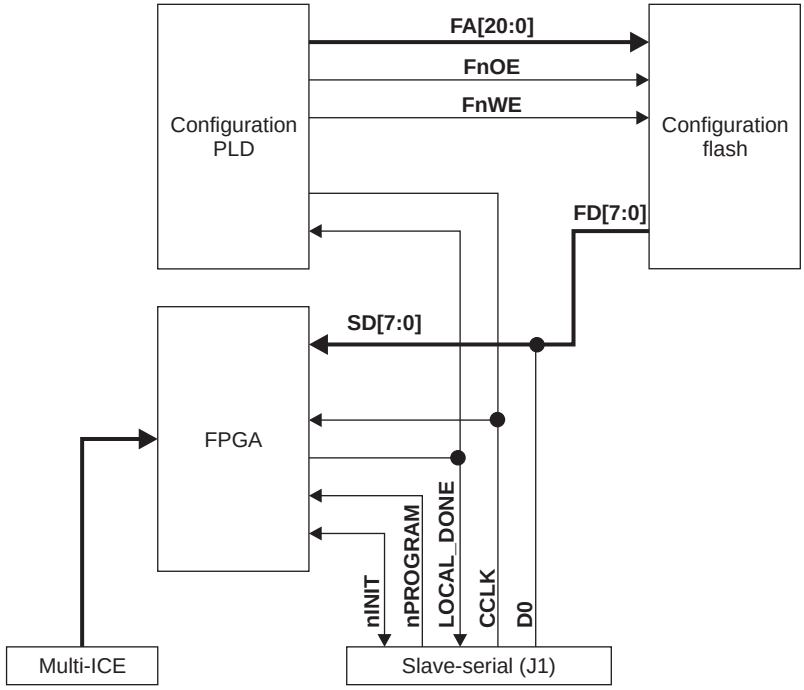


Figure 3-1 FPGA configuration architecture

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 3-2.

Table 3-2 FPGA programming modes

M2	M1	M0	Mode selected
0	1	1	Slave serial mode
0	1	0	Select MAP mode

Note

The signals **FD[7:0]** are connected with **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 **FnOE** and **FnWE** HIGH to inhibit reads and writes to the flash.

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 *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 DIL switched or by **CFGSEL[1:0]** from the motherboard (see *Configuring the FPGA from flash* on page 4-4).

Slave serial mode

You can use slave serial mode to configure the FPGA with the XChecker tool. This mode is selected by fitting the CONFIG link.

Boundary scan programming

You can use the Multi-ICE JTAG port to download configurations when the CONFIG link is fitted (see *JTAG support* on page 3-12).

3.2 Bus interfaces

When used with an Integrator motherboard or core module, the logic modules require a system bus interface. You can also implement an expansion bus interface.

3.2.1 System bus interface

The system bus interface connects the logic module with other Integrator modules. This can be implemented using AHB or ASB interface. Example configurations are supplied with the logic module to get you started and to enable you to develop with various bus configurations (see Chapter 5 *FPGA Configuration Examples*).

In a conventional AMBA system, a single central decoder is used to provide a select signal (**DSELx** 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 EXPA/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]**, **nPRES[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. Table 3-3 shows addresses for modules in either stack position.

Table 3-3 Logic module base address decodes

ID[3:0]	Module ID	Base Address		Size
		EXPA/EXPB	HDRA/HDRB	
1101	3 (top)	0xF0000000	0xB0000000	256MB
1011	2	0xE0000000	0xA0000000	256MB
0111	1	0xD0000000	0x90000000	256MB
1110	0 (bottom)	0xC0000000	0x80000000	256MB

3.2.2 Expansion bus interface

The *Expansion Bus Interface* (EBI) supports the addition of memory or peripheral expansion to the EBI connector EXPM. It provides a 32-bit data bus and a 26-bit address bus, and control signals. You can use the EBI as an expansion of the EBI on the motherboard or for local expansion for the logic module.

The *Integrator/AP User Guide* and *Integrator/SP User Guide* provide details about the expansion bus interface.

3.3 Clock control

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

- *Clock architecture*
- *Programming the on-board clock generators* on page 3-8.

3.3.1 Clock architecture

Figure 3-2 shows the architecture of the clocking system.

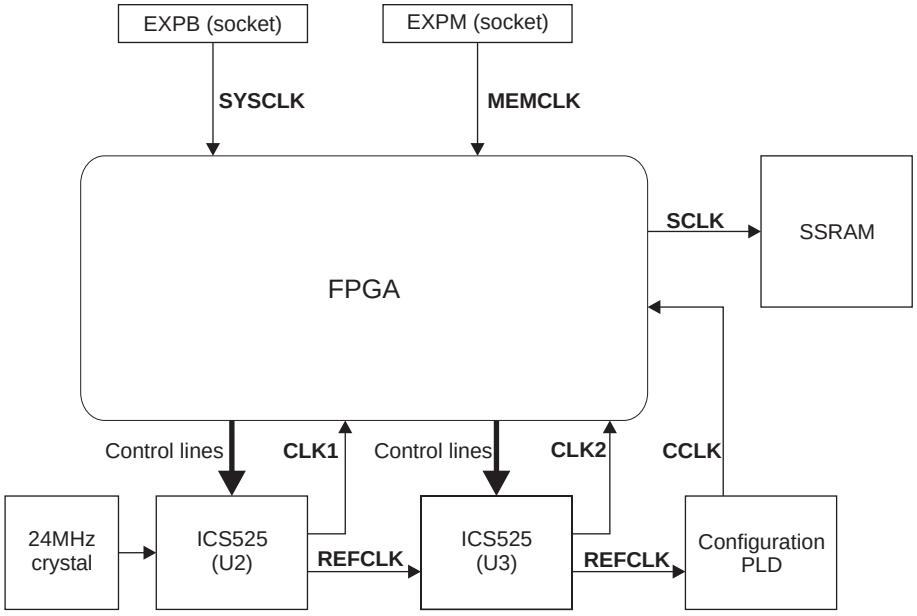


Figure 3-2 Clock architecture

The ICS525s are two programmable clock sources that are supplied with a reference clock signal by a 24MHz crystal oscillator. The output frequency from each ICS525 is configured by setting signal levels on its *divider* input pins. These are connected to the FPGA. Pull-down resistors on the divider inputs ensure that the oscillator outputs default to 4.8MHz when 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 APB register peripheral* on page 5-7).

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

Table 3-4 provides a summary of the clock signals on the logic module.

Table 3-4 Logic module clock signals

FPGA pin name	Clock name	Clock source
GCK0	**SYSCLK**	Motherboard system clock.
GCK1	**CLK1**	On-board clock generator (programmable).
GCK2	**CLK2**	On-board clock generator (programmable).
GCK3	**MEMCLK**	Motherboard memory expansion clock.
CCLK	**CCLK**	Configuration clock supplied by the PLD to the FPGA to synchronize FPGA configuration. It is also used by the XChecker tool in slave serial mode. It is unused at other times.
-	**SCLK**	The example FPGA configuration uses an FPGA input/output pin to supply a clock signal to the SSRAM.

Note

The **MEMCLK** signal is supplied by the Integrator/AP motherboard through corresponding EXPM connectors on both the logic module and motherboard. The Integrator/SP does not provide a similar connector and, therefore, does not supply **MEMCLK** to the logic module.

3.3.2 Programming the on-board clock generators

The two clock generators are independently programmable and can produce frequencies in the range 1 MHz to 160 MHz. The output frequency is calculated using the following formula:

$$\text{Frequency} = 48\text{MHz} \cdot \frac{(\text{CLK\_CTRL}[8:0] + 8)}{(\text{CLK\_CTRL}[15:9] + 2) \cdot S}$$

where S can be assigned any value shown in Table 3-5 on page 3-9.

Table 3-5 Values for clock divisor S

Divisor S	CLK_CTRL[18:16]
2	001
4	011
5	100
6	111
7	101
8	010
9	110
10	000

The following operating range limits must be observed:

$$10\text{MHz} < 48\text{MHz} \cdot \frac{(\text{CLK\_CTRL}[8:0] + 8)}{(\text{CLK\_CTRL}[15:9] + 2)}$$

$$\text{CLK\_CTRL}[15:9] < 118$$

Note

You can calculate values for the clock control signals using the ICS525 calculator on the Microclock website.

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 3-3 shows the architecture of the reset system.

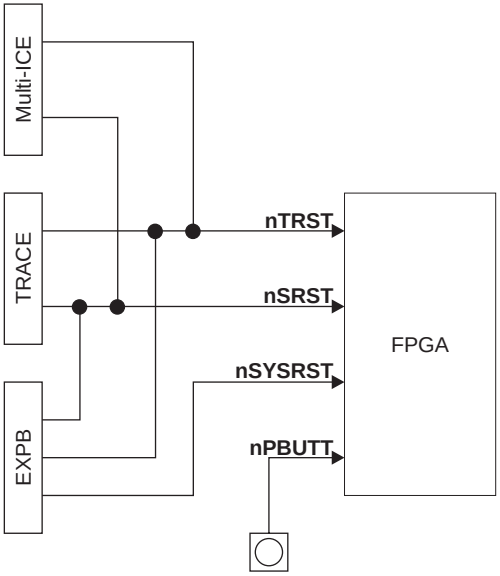


Figure 3-3 Reset architecture

3.4.1 JTAG test reset

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 on the access to a TAP controller synthesized in the FPGA configuration. There are three possible sources of the **nTRST** signal:

- Multi-ICE connector
- Trace (embedded trace module) connector
- motherboard or core module on the EXPB connectors

Note

- The **nTRST** signal is also used to control FPGA **nPROGRAM** signal when the CONFIG link is fitted.
- On the Integrator/AP, the expansion connector (EXPB) version of **nTRST** is completely isolated from the core module (HDRB) version of **nTRST**.

3.4.2 System reset

The 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) version of **nSRST** is completely separate from the core module (HDRB) version of **nSRST** (see the *Integrator/AP User Guide* for more details).

3.4.3 Motherboard reset

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).

3.4.4 Push button

The push button is general purpose but can be used as a reset button. It provides an active LOW input to an input/output pin on the FPGA. This is separate to any of the other reset signals.

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 JTAG signals **TMS**, **TCK**, and **TDI** route directly down to the motherboard.

The routing of the JTAG signals depends on two 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.

The CONFIG link allows you to select between two JTAG modes:

- configuration mode, which is used for in-system reprogramming of the FPGA or PLD
- user mode, which is used for designs that synthesize a virtual TAP controller.

Note

The Integrator/AP provides logic module and a 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 3-4 shows the JTAG routing on the logic module in configuration mode.

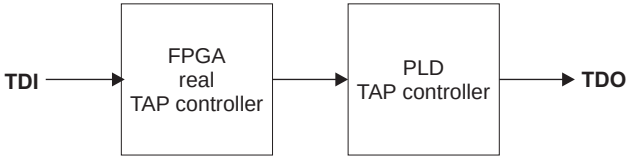


Figure 3-4 Configuration mode JTAG routing

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, it might be necessary to reduce the JTAG **TCK** speed to 1MHz to ensure reliable operation (see the *Multi-ICE User Guide*).

3.5.2 User mode

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

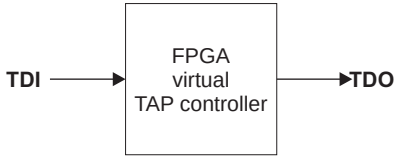


Figure 3-5 User and configuration mode JTAG routing

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 that have been designated for implementing a *virtual* TAP controller. This facility is provided for FPGA designs that require a TAP controller, for example, designs that include a synthesized processor.

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.

In this 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 3-15).

3.5.3 Using JTAG with a multi-module Integrator system

Figure 3-6 on page 3-14 shows the JTAG data routing for a logic module, core module, and 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, **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.

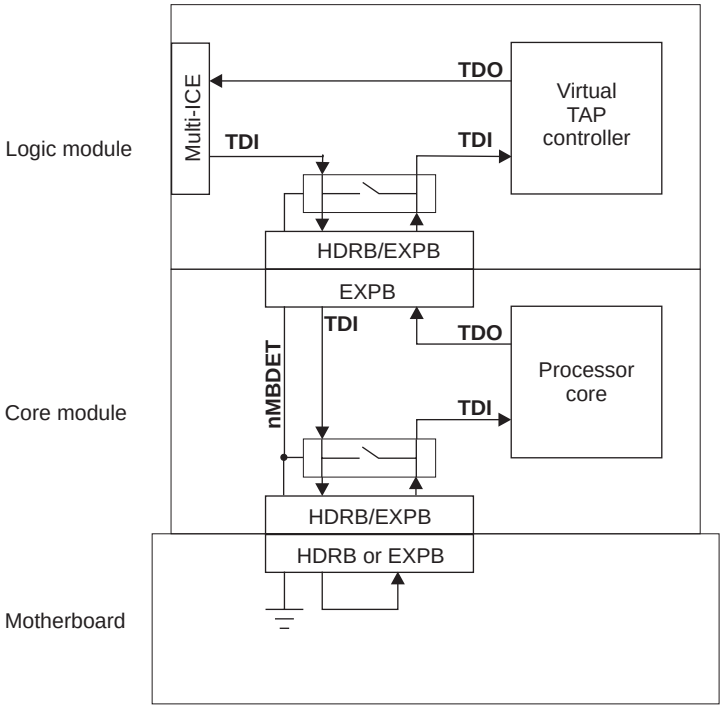


Figure 3-6 JTAG data path (user mode)

Note

- The JTAG port does not 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.
- If the logic module is mounted in a stack that includes a core module or another logic module that implements a TAP controller, the FPGA design must route **TDI** to **TDO** and **TCK** to **RCLK** for the JTAG system to work correctly.

3.5.4 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.

3.5.5 JTAG signal descriptions

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

Table 3-6 JTAG and related signals

Signal	Description	Function
TDI	Test Data In (from JTAG tool)	This signal goes down the stack of modules to the motherboard and then back up the stack, labelled **TDO**.
TDO	Test Data Out (to JTAG tool)	This signal is the return path of the data input signal **TDI**. The module connectors HDRB/EXPB have two pins, labelled **TDI** and **TDO**. **TDI** refers to data flowing down the stack and **TDO** to data flowing up the stack. The JTAG components are connected in the return path so that the length of track driven by the last component in the chain is kept as short as possible.
TCK	Test Clock (from JTAG tool)	This signal synchronizes all JTAG transactions. **TCK** connects to all JTAG components in the **TDI**-**TDO** chain. It makes use of series termination resistors on stubs to reduce reflections and maintain good signal integrity. **TCK** flows down the stack of modules and connects to each JTAG component, but if there is a device in the scan chain that synchronizes **TCK** to some other clock, then all down-stream devices are connected to the **RTCK** signal on that component (see **RTCK** below).
TMS	Test Mode Select (from JTAG tool)	This signal controls transitions in the tap controller state machine. **TMS** connects to all JTAG components in the scan chain as the signal flows down the module stack.

Table 3-6 JTAG and related signals (continued)

Signal	Description	Function
RTCK	Return **TCK** (to JTAG tool)	Some devices sample **TCK** (for example, a synthesizable core with only one clock), and this delays the time at which a component actually captures data. **RTCK** is used to return the sampled clock to the JTAG equipment, so that the **TCK** is not advanced until the synchronizing device has captured the data. In adaptive clocking mode, Multi-ICE must detect an edge on **RTCK** before changing **TCK**. In a multiple-device JTAG chain, the **RTCK** output from a component connects to the **TCK** input of the down-stream device. The **RTCK** signal on the module connectors HDRB/EXPB returns **TCK** to the JTAG equipment. If there are no synchronizing components in the **TDI**-**TDO** chain then, it is not necessary to use the **RTCK** signal and it is connected to ground on the motherboard.
nRTCKEN	Return **TCK** enable (from module to motherboard)	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 if it is not in configuration mode.
nCFGEN	Configuration enable (from jumper on module at the top of the stack)	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.
GLOBAL_DONE	All FPGAs are configured	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/ouptut pad during synthesis.

Note

The signal naming on the logic module differs slightly from the rest of the Integrator syste. 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 SSRAM and flash memory.

3.6.1 SSRAM

A 64K x 32-bit SSRAM (Micron part number MT58LC128K32B4) 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 ASB/AHB 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

There are four general purpose green LEDs. These are lit by driving the associated LED output pin LOW.

The Example 2 FPGA configuration supplied with the logic module provides the register LM_LEDs to control the LEDs (see *Example 2 APB register peripheral* on page 5-7).

The location of the LEDs is illustrated in Figure 1-3 on page 1-5.

A 4-way DIL switch on the module is provided for user-defined operation. However when the logic module is powered up in user mode, the lower two switches are used to select the flash image (see *Configuring the FPGA from flash* on page 4-4).

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

3.8 Prototyping grid

The prototyping grid consists of an 18 x10 grid of 0.1 inch pitch plated-through holes. The holes are labelled A to R, and 0 to 9.

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 grounds are provided around an area of open circuit holes.

You can use the prototyping grid to, for example:

- wire to off-board circuitry
- mount connectors
- mount small integrated circuits.

All of the signals from FPGA bank 5 (see *FPGA bank assignment* on page 3-2) are routed to the prototyping grid so that you can use the Virtex SelectIO feature of the FPGA (see *Virtex Application Note XAPP133*). The SelectIO feature provides support for a range of interfacing standards.

You can set the input voltage thresholds using the VREF_XXX inputs on prototyping holes E[0:7]. These signals can be assigned as input/output signals when they are not required as input reference signals (see *Virtex Application Note XAPP133*).

Larger FPGA devices use more of the input/output pins as voltage reference inputs. For example, the signal VREF_6_IO is an optional voltage reference input on XCV600 devices and above, VREF_8_IO on XCV800 and so on (see *Virtex Application Note XAPP133*).

The output voltage for bank 5 is set by the signal VCCO_5. There are three options for providing the voltage for this signal:

- 3.3V supplied from the logic module (default position)
- 2.5V supplied from the logic module
- user-supplied voltage from prototyping hole E8 (link LK1 must be removed for this option).

These options are selected using soldered link LK1.

Full details of the signals available on the prototyping grid are provided in Appendix B *FPGA Pin Allocation*.

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
Designer's Reference

This chapter describes how to download your own FPGA configurations and how configuration images in flash are selected. It contains the following sections:

- *Tool flow* on page 4-2
- *Configuring the FPGA from flash* on page 4-4
- *Loading new FPGA configurations* on page 4-6
- *Reprogramming the PLD* on page 4-8.

4.1 Tool flow

Preparing FPGA configuration files entails two steps:

1. Synthesis.
2. Place and route.

Figure 4-1 illustrates the basic tool flow process.

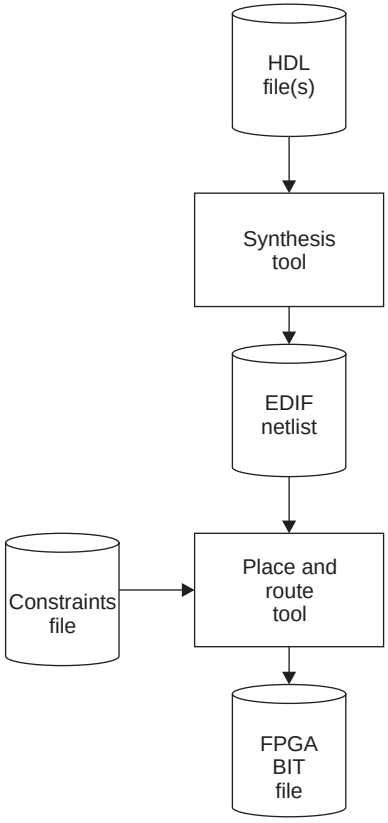


Figure 4-1 Basic tool flow

4.1.1 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 the logic module, the target technology is Xilinx Virtex. There are several synthesis tools available for both PC and UNIX, which 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, which is place and route.

Note

To ensure that the Integrator system operates correctly, a pull-up on the **GLOBAL_DONE** signal is required on all designs. Specify this during the synthesis stage.

4.1.2 Place and route

Place and route for the LM-XCV400+ is performed by Xilinx-specific software, to produce a `.bit` file which is used to program the FPGA. The Virtex targeted `.edf` is aimed at a particular device, which takes into account the device size, package type, and speed grade.

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 file `pinout.ucf` 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 an route process.

4.2 Configuring the FPGA from flash

The flash memory has space to store up to four configurations for XCV400 or XCV600 FPGA types, or up to two configurations for XCV800 or XCV1000 sizes. The configuration image is selected when the logic module is powered up. Image selection is controlled by the configuration PLD.

The logic module is supplied with two PLD programs, `pld_a.svf` and `pld_b.svf` (the PLD is preprogrammed with `pld_a.svf`). These provide two options for the way the flash image is selected. The PLD configurations are:

`pld_a.svf` When the logic module is used standalone, SW1-1 and SW1-2 are set to select the FPGA configuration.

If the logic module is mounted on a motherboard, **CFGSEL[1:0]** (from the motherboard) are used to select the FPGA configuration.

`pld_b.svf` SW1-1 and SW1-2 are always used to select the configuration.

Table 4-1 and Table 4-2 show the FPGA image selection options.

Table 4-1 Configuration image selection for XVC400 and XVC600 FPGAs

Flash image	Image base address	CFGSEL[1:0]	SW1-1	SW1-2	SW1-3	SW1-4
0	`0x000000`	00	ON	ON	X	X
1	`0x080000`	01	ON	OFF	X	X
2	`0x100000`	10	OFF	ON	X	X
3	`0x180000`	11	OFF	OFF	X	X

Table 4-2 Configuration image selection for XVC800 and XVC1000 FPGAs

Flash image	Image base address	CFGSEL[1:0]	SW1-1	SW1-2	SW1-3	SW1-4
0	`0x000000`	X0	ON	X	X	X
1	`0x100000`	X1	OFF	X	X	X

Note

- The switch labels used in the Table 4-1 and Table 4-2 on page 4-4 refer to the markings on the board. These are named **SW[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.
- The PLD detects the presence of the XCV800/1000 by the omission of R25.

See *Example 2 programmer's reference* on page 5-5 for a description of the example images stored in flash when the logic module is shipped.

4.3 Loading new 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 configure mode. This is enabled by fitting the CONFIG link (J13). The CFGLED is lit as an indication that configure mode is selected.

For a description of the configure mode, see *JTAG support* on page 3-12.

4.3.1 Reconfiguring the FPGA directly with JTAG

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 PC, for example a 400MHz machine.

Using Xilinx XChecker cable

Use the DOWNLOAD connector (J1) with the Xilinx XChecker cable. Refer to Xilinx documentation for the use and operation of this tool.

Note

A 3.3V voltage adapter should be used with Virtex 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 online document file `progsuite.pdf` on the supplied CD.

To load a new configuration into the FPGA:

1. Produce a `<filename>.bit` file with a **CCLK** start up clock 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, `LMXCV1000.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 cannot autodetect logic modules. You must manually configure the Multi-ICE server. Refer to the *Multi-ICE User Guide* for further information.

4.3.2 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 with JTAG* on page 4-6. 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.

4.4 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. An invalid or incorrect design can cause the JTAG signal routing circuitry to fail and prevent the device being reprogrammed.

Program the PLD as follows:

1. Put the logic module into configuration mode by fitting the CONFIG link (J13) and power-up.
2. Start the Multi-ICE server and the load the configuration file for your logic module. For example:
   `<CD drive>:\LM-XCV400\programming\LMXCV1000.cfg`
3. Start a command prompt and move to the directory
   `<CD drive>:\LM-XCV400\programming\`
4. Run the `progcards` utility.
5. Choose the required PLD image.

Chapter 5
FPGA Configuration Examples

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

- *About the FPGA configuration examples* on page 5-2
- *Example 2 programmer's reference* on page 5-5.

5.1 About the FPGA configuration examples

The logic module is supplied with two example FPGA configurations. These are supplied to allow you to gain experience with synthesis, design, and place and route for the logic module. The examples are written in VHDL.

5.1.1 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 **SW[1:0]**, and places a binary count on the logic analyzer connector.

One HDL source file is supplied and bit files for all supported sizes of Virtex FPGAs are included.

5.1.2 Example 2

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

- ASB motherboard and ASB peripherals (preloaded flash image at `0x000000`)
- ASB motherboard and AHB peripherals
- AHB motherboard and ASB peripherals
- AHB motherboard and AHB peripherals (preloaded flash image at `0x100000`).

Note

The ASB-ASB and AHB-AHB versions of Example 2 are preloaded into flash (see *Configuring the FPGA from flash* on page 4-4).

The four 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 signals from the logic module are routed to the interrupt controller on the Integrator/AP. You can also use the logic module with the Integrator/SP but interrupt requests from the logic module are not supported.

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

- an SSRAM controller
- an ASB/AHB to APB bridge
- an APB register peripheral
- an APB interrupt controller.

Figure 5-1 shows the examples without a system bus bridge.

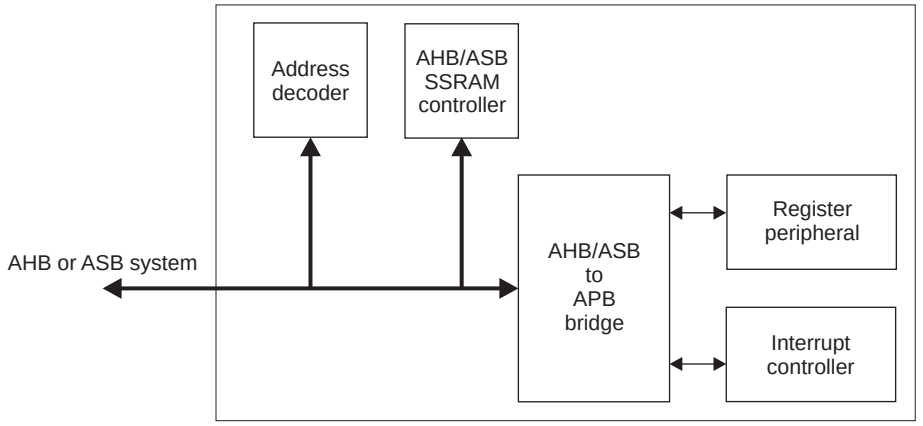


Figure 5-1 Example without bridge (ASBASB and AHBAHB)

Figure 5-2 shows the examples with a system bus bridge.

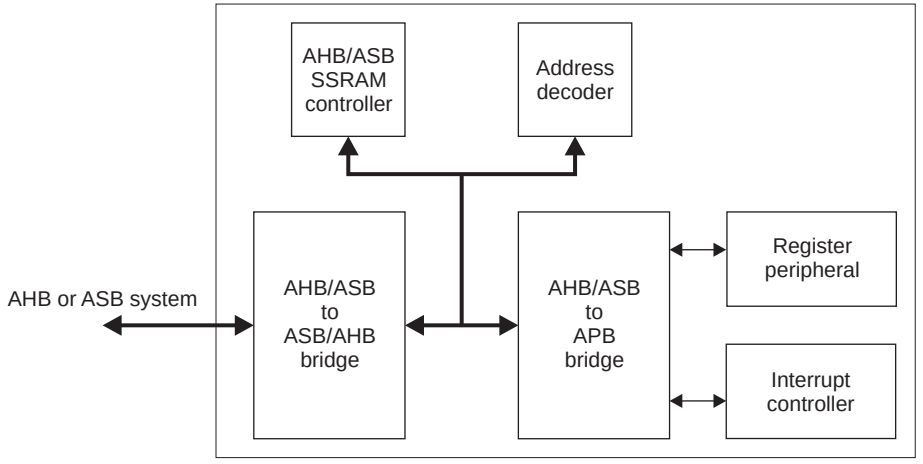


Figure 5-2 Example with bridge (ASBAHB and AHBASB)

Table 5-1 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 5-1 HDL file descriptions

File	Description
ASBASBTop.vhd ASBAHBTop.vhd AHBASBTop.vhd AHBASBTop.vhd	These four 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.
ASB2AHB.vhd AHB2ASB.vhd	These are the bridges required to connect ASB or AHB peripherals to a non-native system.
ASBASBDecoder.vhd AHBASBDecoder.vhd AHBDecoder.vhd	The decoder blocks provide 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. There are two different decoders supplied for the ASP peripheral examples. This is because the selection of the default slave and ASB response signals are different when the AHB2ASB bridge is present (see VHDL code for details). The Integrator family of boards uses a distributed address decoding system (see *Example 2 memory map* on page 5-6).
AHBMuxS2M.vhd	This is the AHB multiplexor that connects the read data buses from all of the slaves to the AHB master(s).
ASBSSRAM.vhd AHBSSRAM.vhd	High-speed peripherals require SSRAM controller blocks that support word, halfword, and byte operations to the 256KB of SSRAM on the logic module.
ASB2APB.vhd AHB2APB.vhd	These are the bridge blocks required to connect APB peripherals to the high-speed AMBA buses. They produce the PSELs for each of the APB peripherals.
ASBAPBSys.vhd AHBAPBSys.vhd	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.
APBRegs.vhd	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 four user LEDs • read the four switch inputs. It also latches the pressing of the push button to generate an expansion interrupt.
APBIntcon.vhd	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 AxBAPBSys block.) Four software interrupts are implemented.

5.2 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 all four combinations of Example 2. The software uses the *ARM Firmware Suite* (AFS) to set up the environment and to run the example code.

There are separate project files for all processors currently supported by the Integrator platform for both the Software Development Toolkit v2.50 and the ARM Developer Suite v1.0.

Before compiling the Example 2 project, you must also build the µHAL library for the logic module.

5.2.1 Software description

There are three source files included in Example 2 that link with the µHAL libraries associated with the processor you are using:

`logic.c` The main C code.

`logic.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 operation of the code is as follows:

1. Determines DRAM size on the core module and set up the system controller.
2. Checks that the logic module is present in the AP expansion position.
3. Reports module information.
4. Sets system and logic module clock frequencies.
5. Sets up system interrupt controller and logic module interrupt controller.
6. Tests SSRAM for word, halfword, and byte accesses.
7. Enters and remains in LED flashing loop.
8. Responds to the pressing of the logic module push button by generating an IRQ and displaying the 4-way DIL switch value.

5.2.2 Example 2 memory map

Example 2 sets up the memory map for the logic module. Figure 5-3 shows the locations to which logic modules are assigned within the overall Integrator system memory map.

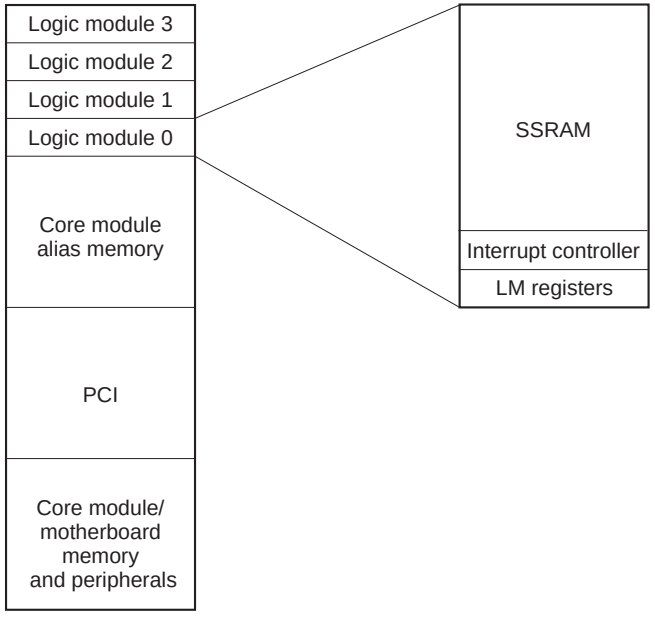


Figure 5-3 Integrator memory map

The Integrator system implements a distributed address decoding scheme in which each core or logic module is responsible for decoding its own area of memory.

Note

It is important when implementing a logic module design, to ensure that the expansion logic implements a decoder and responds to all memory accesses in the appropriate memory region (see *System bus interface* on page 3-5).

5.2.3 Example 2 APB register peripheral

Table 5-2 shows the mapping of the logic module registers. The addresses shown are offsets from the base addresses shown in Figure 5-3 on page 5-6.

Table 5-2 Logic module registers

Offset address	Name	Type	Size	Function
0x0000000	LM_ID	Read/write	24	Logic module ID register
0x0000004	LM_OSC1	Read/write	19	Oscillator divisor register 1
0x0000008	LM_OSC2	Read/write	19	Oscillator divisor register 2
0x000000C	LM_LOCK	Read/write	17	Oscillator lock register
0x0000014	LM_LEDS	Read/write	4	User LEDs control register
0x0000018	LM_INT	Read/write	1	Push button interrupt register
0x000001C	LM_SW	Read	4	Switches register

Logic module ID register

The ID register identifies the board manufacturer, board type, and revision.

Oscillator divisor registers

The oscillator registers control the frequency of the clocks generated by the two clock generators (see *Clock control* on page 3-7).

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 5-3 describes the oscillator register bits.

Table 5-3 LM_OSCx registers

Bits	Name	Access	Function	Default
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 default values set the oscillators to 1MHz.

For information about setting the clock frequency, see *Clock control* on page 3-7.

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 5-4 describes the lock register bits.

Table 5-4 LM_LOCK register

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

User LEDs control register

The LEDs register is used to control the four user LEDs (see *LEDs summary* on page 1-5). 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 5-5 describes the operation of this register.

Table 5-5 LM_INT register

Bits	Name	Access	Function
0	LM_INT	Read	This bit when SET is a latched indication that the push button has been pressed.
		Write	Write 0 to this register to CLEAR the latched indication. Writing 1 to this register has the same effect as pressing the push button.

Switches register

This register is used to read the setting of the 4-way DIL switch. A 0 indicates that the associated switch element is closed (ON).

5.2.4 Example 2 interrupt controller

The interrupt control registers are listed in Table 5-6.

Table 5-6 Interrupt controller registers

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

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 5-4 on page 5-11 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.

Note

This function is not supported if the logic module is mounted in the HDRA/HDRB position on an Integrator/AP or on an Integrator/SP (see *Differences between core and logic modules* on page 2-7).

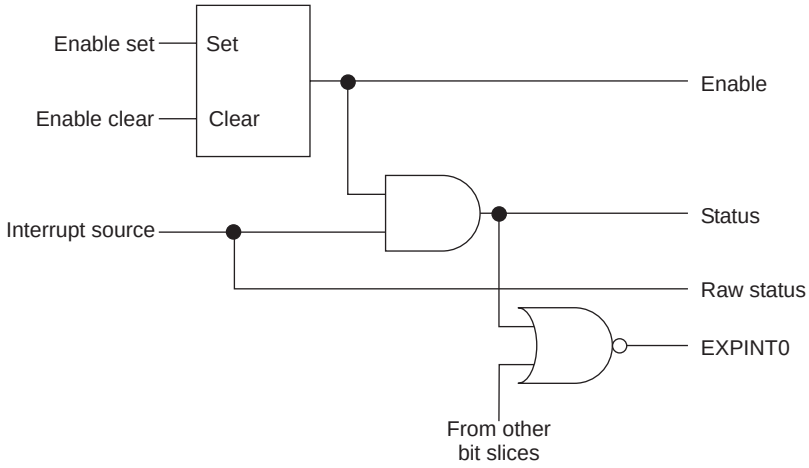


Figure 5-4 Interrupt control

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 5-7.

Table 5-7 Interrupt register bit assignment

Bit	Name	Function
7:5	-	Spare
4	PBINT	Push button interrupt
3:0	SOFTINT[3:0]	Interrupt generated by writing to the LM_SOFTINT location

Appendix A
Signal Descriptions

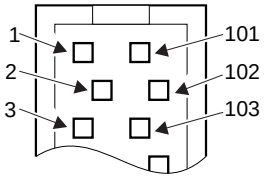
This appendix describes the Integrator/AP interface connectors and signal connections. It contains the following sections:

- *Logic module connector EXPA* on page A-2
- *Logic module connector EXPB* on page A-5
- *Connector EXPM* on page A-7
- *Diagnostic connectors* on page A-9.

A.1 Logic module connector EXPA

Figure A-1 shows the pin numbers of the EXPA connectors. It shows the location of the power and ground pins.

Pin numbers for 200-way plug, viewed from above board



Samtec TOLC series

1	A0		GND		101
2		GND		D0	102
3	A1		D1		103
4		A2		D2	104
5	A3		GND		105
6		GND		D3	106
7	A4		D4		107
8		A5		D5	108
9	A6		GND		109
10		GND		D6	110
11	A7		D7		111
12		A8		D8	112
13	A9		GND		113
14		GND		D9	114
15	A10		D10		115
16		A11		D11	116
17	A12		GND		117
18		GND		D12	118
19	A13		D13		119
20		A14		D14	120
21	A15		GND		121
22		GND		D15	122
23	A16		D16		123
24		A17		D17	124
25	A18		GND		125
26		GND		D18	126
27	A19		D19		127
28		A20		D20	128
29	A21		GND		129
30		GND		D21	130
31	A22		D22		131
32		A23		D23	132
33	A24		GND		133
34		GND		D24	134
35	A25		D25		135
36		A26		D26	136
37	A27		GND		137
38		GND		D27	138
39	A28		D28		139
40		A29		D29	140
41	A30		GND		141
42		GND		D30	142
43	A31		D31		143
44		B0		C0	144
45	B1		GND		145
46		GND		C1	146
47	B2		C2		147
48		B3		C3	148
49	B4		GND		149
50		GND		C4	150
51	B5		C5		151
52		B6		C6	152
53	B7		GND		153
54		GND		C7	154
55	B8		C8		155
56		B9		C9	156
57	B10		GND		157
58		GND		C10	158
59	B11		C11		159
60		B12		C12	160
61	B13		GND		161
62		GND		C13	162
63	B14		C14		163
64		B15		C15	164
65	B16		GND		165
66		GND		C16	166
67	B17		C17		167
68		B18		C18	168
69	B19		GND		169
70		GND		C19	170
71	B20		C20		171
72		B21		C21	172
73	B22		GND		173
74		GND		C22	174
75	B23		C23		175
76		B24		C24	176
77	B25		GND		177
78		GND		C25	178
79	B26		C26		179
80		B27		C27	180
81	B28		GND		181
82		GND		C28	182
83	B29		C29		183
84		B30		C30	184
85	B31		GND		185
86		GND		C31	186
87	5V		3V3		187
88		3V3		12V	188
89	5V		3V3		189
90		3V3		12V	190
91	5V		3V3		191
92		3V3		12V	192
93	5V		3V3		193
94		3V3		12V	194
95	5V		3V3		195
96		3V3		12V	196
97	5V		3V3		197
98		3V3		12V	198
99	5V		3V3		199
100		3V3		12V	200

Figure A-1 HDRA/EXPA pin numbering

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.

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

Pin label	Name (ASB)	Description
A[31:0]	**ADDR[31:0]**	System address bus
B[31:0]	Not used	-
C[31:0]	**CONT[31:0]**	System control bus
D[31:0]	**DATA[31:0]**	System data bus

Table A-2 shows **CONT[31:0]** signal descriptions for an ASB bus implementation.

Table A-2 C bus detail for ASB

Position	Signal name	Description
CONT[31:16]	Not used	Connected to FPGA input/output and reserved
CONT15	**BLOK**	Locked transaction
CONT14	**BLAST**	Last response
CONT13	**BERROR**	Error response
CONT12	**BWAIT**	Wait response
CONT11	**BWRITE**	Write transaction
CONT10	Not used	Connected to FPGA input/output and reserved
CONT[9:8]	**BPROT[1:0]**	Transaction protection type
CONT7	Not used	Connected to FPGA input/output and reserved
CONT[6:5]	**BURST[1:0]**	Transaction burst size
CONT4	Not used	Connected to FPGA input/output and reserved
CONT[3:2]	**BSIZE[1:0]**	Transaction width
CONT[1:0]	**BTRAN[1:0]**	Transaction type

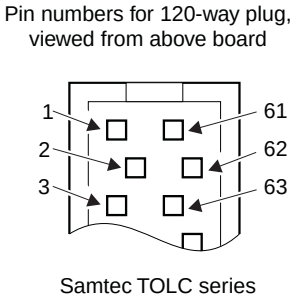
Table A-3 shows **CONT[31:0]** signal descriptions for an AHB bus implementation.

Table A-3 C bus detail for AHB

Position	Signal name	Description
CONT[31:16]	**HGRANT[7:6]**	Connected to FPGA input/output and reserved
CONT[29:28]	**HBUSREQ[7:6]**	Connected to FPGA input/output and reserved
CONT[27:26]	**HLOCK[7:6]**	Connected to FPGA input/output and reserved
CONT[25:24]	Note used	Connected to FPGA input/output and reserved
CONT[23:16]	**HSPLIT[7:0]**	Split transaction
CONT15	**HMASTLOCK**	Locked transaction
CONT[14:13]	**HRESP[1:0]**	Transfer response
CONT12	**HREADY**	Ready response
CONT11	**HWRITE**	Write transaction
CONT[10:8]	**BPROT[2:0]**	Transaction protection type
CONT[7:5]	**HBURST[2:0]**	Transaction burst size
CONT4	**HPROT3**	Transaction protection type
CONT[3:2]	**HSIZE[1:0]**	Transaction width
CONT[1:0]	**HTRANS[1:0]**	Transaction type

A.2 Logic module connector EXPB

Figure A-2 shows the pin numbers of the connector EXPB.



1	H0		GND				61
2		GND		GPIO0			62
3	H1		GPIO1				63
4		H2		GPIO2			64
5	H3		GND				65
6		GND		GPIO3			66
7	H4		GPIO4				67
8		H5		GPIO5			68
9	H6		GND				69
10		GND		GPIO6			70
11	H7		GPIO7				71
12		H8		GPIO8			72
13	H9		GND				73
14		GND		GPIO9			74
15	H10		GPIO10				75
16		H9		GPIO11			76
17	H12		GND				77
18		GND		GPIO12			78
19	H13		GPIO13				79
20		H14		GPIO14			80
21	H15		GND				81
22		GND		GPIO15			82
23	H16		GPIO16				83
24		H17		GPIO17			84
25	H18		GND				85
26		GND		GPIO18			86
27	H19		GPIO19				87
28		H20		GPIO20			88
29	H21		GND				89
30		GND		GPIO21			90
31	H22		GPIO22				91
32		H21		GPIO23			92
33	H24		GND				93
34		GND		GPIO24			94
35	H25		GPIO25				95
36		H26		GPIO26			96
37	H27		GND				97
38		GND		GPIO27			98
39	H28		GPIO28				99
40		H29		GPIO29			100
41	H30		GND				101
42		GND		GPIO30			102
43	H31		GPIO31				103
44		J0		J8			104
45	J1		GND				105
46		GND		J9			106
47	J2		J10				107
48		J3		J11			108
49	J4		GND				109
50		GND		J12			110
51	J5		J13				111
52		J6		J14			112
53	J7		J16				113
54		GND		J15			114
55	5V		-12V				115
56		3V3		12V			116
57	5V		-12V				117
58		3V3		12V			118
59	5V		-12V				119
60		3V3		12V			120

Figure A-2 EXPB pin numbering

Table A-4 describes the signals on the pins labeled H[31:0] and J[15:0]. The pins marked GPIO[31:0] are connected to the GPIO port of the FPGA.

Table A-4 EXPB signal description

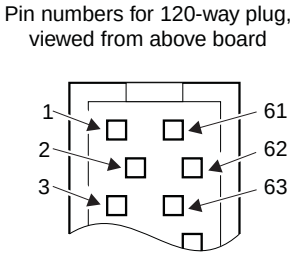
Pin label	Name	Description
H[31:28]	**SYSCLK[7:4]**	System clock to each logic module (**BLCK** in AMBA ASB).
H[27:24]	**nEPPRES[3:0]**	Expansion module present.
H[23:20]	**nIRQSRC[3:0]**	Interrupt source from expansion module 3, 2, 1, and 0 respectively.
H[19:16]	-	Not connected.
H[15:12]	**ID[3:0]**	Logic module position indicator.

Table A-4 EXPB signal description (continued)

Pin label	Name	Description
H[11:8]	**SLOCK[3:0]**	System bus lock from processors 3, 2, 1, and 0 respectively (not used in AMBA ASB).
H[7:4]	**SGNT[3:0]**	System bus grant.
H[3:0]	**SREQ[3:0]**	System bus request.
J16	**nRTCKEN**	**RTCK** AND gate enable.
J[15:14]	**CFGSEL[1:0]**	FPGA configuration select.
J13	**nCFGEN**	Sets the stack and motherboard into configuration mode.
J12	**nSRST**	Multi-ICE reset (open collector).
J11	**FPGADONE**	Indicates when FPGA configuration is complete (open collector). Named **LOCAL_DONE** on logic module.
J10	**RTCK**	JTAG return **TCK**. Used for Multi-ICE adpative clocking.
J9	**nSYSRST**	Buffered system reset.
J8	**nTRST**	JTAG reset.
J7	**TDO**	JTAG test data out.
J6	**TDI**	JTAG test data in.
J5	**TMS**	JTAG test mode select.
J4	**TCK**	JTAG test clock.
J[3:1]	**MASTER[2:0]**	Master ID. Binary encoding of the master currently performing a transfer on the bus. Corresponds to BUSREQ and BUSGNT line numbers.
J0	**nMBDET**	Motherboard detect pin.

A.3 Connector EXPM

Figure A-3 shows the pin numbers of the connector EXPM.



1	MA0		GND				61
2		GND		MD0			62
3	MA1		MD1				63
4		MA2		MD2			64
5	MA3		GND				65
6		GND		MD3			66
7	MA4		MD4				67
8		MA5		MD5			68
9	MA6		GND				69
10		GND		MD6			70
11	MA7		MD7				71
12		MA8		MD8			72
13	MA9		GND				73
14		GND		MD9			74
15	MA10		MD10				75
16		MAH9		MD11			76
17	MA12		GND				77
18		GND		MD12			78
19	MA13		MD13				79
20		MA14		MD14			80
21	MA15		GND				81
22		GND		MD15			82
23	MA16		MD16				83
24		MA17		MD17			84
25	MA18		GND				85
26		GND		MD18			86
27	MA19		MD19				87
28		MA20		MD20			88
29	MA21		GND				89
30		GND		MD21			90
31	MA22		MD22				91
32		MA23		MD23			92
33	MA24		GND				93
34		GND		MD24			94
35	MA25		MD25				95
36		FLWP		MD26			96
37	FLVPP		GND				97
38		GND		MD27			98
39	nBANK4		MD28				99
40		nBANK5		MD29			100
41	nBANK6		GND				101
42		GND		MD30			102
43	nBANK7		MD31				103
44		nMCS0		nMWR0			104
45	nMCS1		GND				105
46		GND		nMWR1			106
47	nMCS2		nMWR2				107
48		nMCS3		nMWR3			108
49	EBIEN		GND				109
50		GND		nMOE			110
51	MCS0EN		MRDY				111
52		nXCS0		nSYSRST			112
53	MEMCLK		GND				113
54		GND		N/C			114
55	5V		3V3				115
56		3V3		12V			116
57	5V		3V3				117
58		3V3		12V			118
59	5V		3V3				119
60		3V3		12V			120

Figure A-3 EXPM pin numbering

Table A-5 describes the signals on the EXPM pins.

Table A-5 EXPM signal description

Name	Description
MA[25:0]	Memory address bus.
nFLWP	Flash write protect.
nFLVPP	Flash Vpp enable.
nBANK[7:4]	Memory bank selects for **nMCS1**.
nMCS[3:0]	Memory chip select.
EBIEN	EBI enable. Drive LOW to tristate **MA[25:0]**, **nMCSN[3:0]**, **nMWR[3:0]**, and **nMOE**.

Table A-5 EXPM signal description (continued)

Name	Description
MCS0EN	Motherboard SC0 enable/disable.
nXCS0	CS0 to module. Enabled when MCS0EN is LOW.
MEMCLK	Memory clock.
MD[31:0]	Memory data bus.
nMWR[3:0]	Memory write strobe (active LOW).
nMOE	Memory output enable (active LOW).
MRDY	Memory ready. Drive LOW to wait processor, pull up to 3.3V for normal operation.
nSYSRST2	Buffered system reset.

A.4 Diagnostic connectors

This section provides details about the logic analyzer and Trace connectors. Figure A-4 shows the pin numbers of this type of connector.

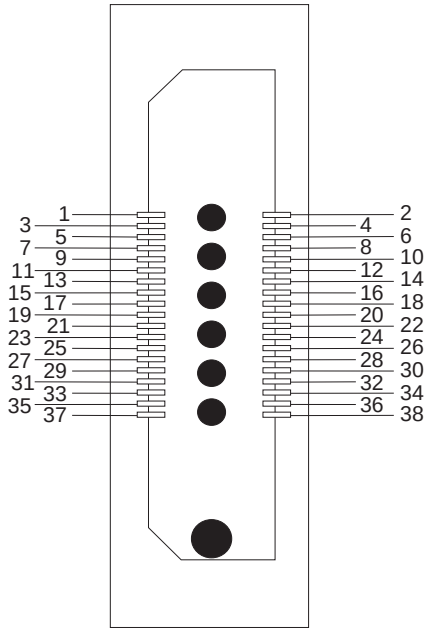


Figure A-4 Diagnostic connector pin locations

A.4.1 Logic analyzer connector

Table A-6 shows the pinout of the logic analyzer connector J7.

Table A-6 J7 Signals descriptions

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

A.4.2 Trace

Table A-7 shows the pinout of the Trace connector.

Table A-7 Trace signals

Pin	Signal	Pin	Signal
1	-	2	-
3	-	4	-
5	**DBGACK**	6	**TRCCLK**
7	**DBGRQ**	8	**TRCPKT11**
9	**nSRST**	10	**TRCPKT10**
11	**TDO**	12	**TRCPKT9**
13	**RTCK**	14	**TRCPKT8**
15	**TCK**	16	**TRCPKT7**
17	**TMS**	18	**TRCPKT6**
19	**TDI**	20	**TRCPKT5**
21	**nTRST**	22	**TRCPKT4**
23	**3V3**	24	**TRCPKT3**
25	**EXTTRIG**	26	**TRCPKT2**
27	**3V3**	28	**TRCPKT1**
29	**3V3**	30	**TRCPKT0**
31	**TRCPKT15**	32	**TRCSYNC**
33	**TRCPKT14**	34	**PIPESTAT2**
35	**TRCPKT13**	36	**PIPESTAT1**
37	**TRCPKT12**	38	**PIPESTAT0**

A.4.3 Multi-ICE (JTAG)

Figure A-5 shows the pinout of the Multi-ICE connector J12. For a detailed description of the JTAG signals, see *JTAG signal descriptions* on page 3-15.

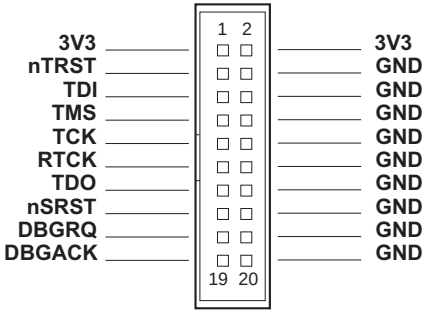


Figure A-5 Muti-ICE connctor pinout

Appendix B
FPGA Pin Allocation

This appendix describes the Integrator/LM-XCV400+ interface connectors and signal connections. It contains the following sections:

- *Clock signals* on page B-2
- *LEDs and switches* on page B-4
- *Diagnostics* on page B-5
- *SSRAM* on page B-8
- *Prototyping grid* on page B-11
- *GPIO (or F BUS)* on page B-12
- *External Bus Interface (EBI)* on page B-13
- *ASB/AHB signals* on page B-16
- *Miscellaneous input/output signals* on page B-18
- *Modem card signals* on page B-20.

Note

A full pin constraints file is provided on the CD supplied with the logic module as a starting point for your design:
`\\LM-XCV400\place-and-route\pinout.ucf`.

B.1 Clock signals

This section summarizes the signal connections related to clock signals.

B.1.1 Clock control signals

Table B-1 lists the clock control signals.

Table B-1 Clock control signals

Signal	Pin (CLK1)	Pin (CLK2)
CLK_CTRL0	AC29	AM9
CLK_CTRL1	AD30	AN9
CLK_CTRL2	AD29	AJ10
CLK_CTRL3	AE33	AK10
CLK_CTRL4	AE32	AM10
CLK_CTRL5	AE30	AJ11
CLK_CTRL6	AF32	AL11
CLK_CTRL7	AF31	AN11
CLK_CTRL8	AF30	AJ12
CLK_CTRL9	AF29	AK12
CLK_CTRL10	AG33	AL12
CLK_CTRL11	AG30	AM12
CLK_CTRL12	AG29	AL13
CLK_CTRL13	AH33	AM13
CLK_CTRL14	AH32	AN13
CLK_CTRL15	AH31	AJ14
CLK_CTRL16	AH29	AK14
CLK_CTRL17	AJ32	AJ15
CLK_CTRL18	AJ31	AK15

B.1.2 Clock inputs

Table B-2 lists the clock input signals.

Table B-2 Clock inputs

Signal	Pin
SYSCLK	AL17
CLK1	AJ17
CLK2	D17
MEMCLK	A17

B.2 LEDs and switches

This section summarizes the signal connections related to LED control and switch sensing.

B.2.1 LEDs

Table B-3 lists the LED control signals.

Table B-3 LED control

LED	Pin	Notes
LED 0	B17	-
LED 1	C17	-
LED 2	E17	-
LED 3	D4	Also indicates FPGA BUSY during flash configuration.

B.2.2 Switches

Table B-4 lists the switch input signals.

Table B-4 Switch sensing

Switch	Pin	Notes
SW0	D15	Also used for FPGA image selection.
SW1	B16	Also used for FPGA image selection if XCV400/600 fitted.
SW2	C16	-
SW3	E16	-

B.3 Diagnostics

This section summarizes the logic analyzer and trace signal connections.

B.3.1 Logic analyzer

Table B-5 lists the logic analyzer signal connections. These pins are shared with prototyping grid.

Table B-5 Logic analyzer signals

	Channel A		Channel B	
Signal	**FPGA pin**	**Connector pin**	**FPGA pin**	**Connector pin**
LA_[B:A]0	AM31	37	AL25	38
LA_[B:A]1	AN31	35	AJ24	36
LA_[B:A]2	AL30	33	AK24	34
LA_[B:A]3	AM30	31	AM24	32
LA_[B:A]4	AM29	29	AJ23	30
LA_[B:A]5	AN29	27	AK23	28
LA_[B:A]6	AL28	25	AL23	26
LA_[B:A]7	AN28	23	AM23	24
LA_[B:A]8	AJ27	21	AJ22	22
LA_[B:A]9	AK27	19	AK22	20
LA_[B:A]10	AM27	17	AM22	18
LA_[B:A]11	AJ26	15	AJ21	16
LA_[B:A]12	AK26	13	AK21	14
LA_[B:A]13	AL26	11	AL21	12
LA_[B:A]14	AN26	9	AN21	10
LA_[B:A]15	AK25	7	AJ20	8
LA_[B:A]CLK	AM20	5	AJ19	6

B.3.2 Trace (ETM) connector

Table B-6 lists the trace signal connections.

Table B-6 Trace signals

Signal	Trace pin	FPGA/JTAG pin
N/C	1,2,3,4	Not applicable
DBGACK	5	AN17
TRCCLK	6	AJ9
DBGRQ	7	AM17
TRCPKT 11	8	AL6
nSRST	9	V30
TRCPKT 10	10	AK6
TDO	11	Test Data Out
TRCPKT 9	12	AJ6
RTCK	13	Return **TCK**
TRCPKT 8	14	AM5
TCK	15	Test Clock
TRCPKT 7	16	AL5
TMS	17	Test Mode Select
TRCPKT 6	18	AK5
TDI	19	Test Data In
TRCPKT 5	20	AL4
nTRST	21	Test Reset
TRCPKT 4	22	AN3
3V3	23	Not applicable
TRCPKT 3	24	AK13
EXTTRIG	25	AM8
TRCPKT 2	26	AL9

Table B-6 Trace signals (continued)

Signal	Trace pin	FPGA/JTAG pin
3V3	27	Not applicable
TRCPKT 1	28	AM14
3V3	29	Not applicable
TRCPKT 0	30	AM4
TRCPKT 15	31	AK7
TRCSYNC	32	AK9
TRCPKT 14	33	AJ7
PIPESTAT 2	34	AL8
TRCPKT 13	35	AN6
PIPESTAT 1	36	AJ8
TRCPKT 12	37	AM6
PIPESTAT 0	38	AK6

B.4 SSRAM

This section summarizes the connections related to the SSRAM.

Table B-7 lists the SSRAM address signals.

Table B-7 SSRAM address pins

Signal	Pin	Notes
SA2	A3	**SA[18:2]** are shared with the flash memory.
SA3	A4	
SA4	B4	
SA5	A5	
SA6	B5	
SA7	C5	
SA8	C6	
SA9	B7	
SA10	C7	
SA11	A8	
SA12	B8	
SA13	C8	
SA14	D8	
SA15	E8	
SA16	A9	
SA17	C9	
SA18	D9	
SA19	B10	B10 and C10 are reserved for future use.
SA20	C10	

Table B-8 lists the SSRAM data signals.

Table B-8 SSRAM data signals

Signal	Pin	Notes	Signal	Pin	Notes
SD0	E4	**SD[7:0]** are shared with flash memory.	**SD16**	AB30	-
SD1	K3		**SD17**	AB29	-
SD2	L4		**SD18**	AC33	-
SD3	P3		**SD19**	AC31	-
SD4	W4		**SD20**	AC30	-
SD5	AB5		**SD21**	AJ30	-
SD6	AC4		**SD22**	AK31	-
SD7	AJ4		**SD23**	AL33	-
SD8	W29	-	**SD24**	V29	-
SD9	Y30	-	**SD25**	Y32	-
SD10	Y29	-	**SD26**	AD31	-
SD11	AA33	-	**SD27**	AE29	-
SD12	AA32	-	**SD28**	AK32	-
SD13	AA31	-	**SD29**	AE31	-
SD14	AA29	-	**SD30**	AA30	-
SD15	AB31	-	**SD31**	AH30	-

Table B-9 lists the SSRAM clock and control signals.

Table B-9 SSRAM clock and control signals

Signal	Pin	Notes
SCLK	A13	-
SnCE	D12	-
SnOE	C12	-
SnWBYTE0	E13	-

Table B-9 SSRAM clock and control signals

Signal	Pin	Notes
SnWBYTE1	C14	-
SnWBYTE2	D14	-
SnWBYTE3	E14	-
SnADV	B11	-
SnADSC	A11	-
SnADSP	E10	-
SnFT	E11	Must be set LOW. This signal is reserved for alternative SSRAM device.
SMODE	C11	-

B.5 Prototyping grid

Table B-10 lists the grid of *Plated Through Holes* (PLT). It lists the FPGA pin or other signal to which each PLT connects. Holes H[9:1] and R[9:1] are connected to GND, holes [H..R]0 are 3.3V, and all other holes are N/C.

Table B-10 Prototyping grid

	A	B	C	D	E	F	G
0	AM31	AM27	AJ23	AN21	AL29	**nPOR**	**LOCAL_DONE**
1	AN31	AJ26	AK23	AJ20	AL24	**nPBUTT**	**BUSY_DOUT**
2	AL30	AK26	AL23	AM20	AL20	**nSRST**	**DXN**
3	AM30	AL26	AM23	AJ19	AJ25	**nSYSRST**	**DXP**
4	AM29	AN26	AJ22	AK19	AJ18	**SW0**	**5V**
5	AN29	AK25	AK22	AL19	AM26	**SW1**	**5V**
6	AL28	AL25	AM22	AN19	AN23	**SW2**	**5V**
7	AN28	AJ24	AJ21	AK18	AK28	**SW3**	**5V**
8	AJ27	AK24	AK21	AL18	**VCCO_5**	**nPROGRAM**	**5V**
9	AK27	AM24	AL21	AM18	**GLOBAL_DONE**	**nINIT**	**5V**

Note

- VCCO_5 is the output voltage for bank 5 (see *Prototyping grid* on page 3-19).
- You can use DXN and DXP to determine the core temperature of the FPGA (see the Xilinx data sheet).

B.6 GPIO (or F BUS)

Table B-11 lists the connections to the GPIO signals on the EXPB connector (see *Logic module connector EXPB* on page A-5).

Table B-11 GPIO signal connections

Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin
GPIO0	G32	**GPIO8**	K29	**GPIO16**	M31	**GPIO24**	R30
GPIO1	H29	**GPIO9**	K30	**GPIO17**	M32	**GPIO25**	R31
GPIO2	H31	**GPIO10**	L29	**GPIO18**	N30	**GPIO26**	R33
GPIO3	H33	**GPIO11**	L30	**GPIO19**	N31	**GPIO27**	T29
GPIO4	J29	**GPIO12**	L31	**GPIO20**	P29	**GPIO28**	T30
GPIO5	J30	**GPIO13**	L32	**GPIO21**	P30	**GPIO29**	T32
GPIO6	J31	**GPIO14**	M29	**GPIO22**	P32	**GPIO30**	U31
GPIO7	J33	**GPIO15**	M30	**GPIO23**	R29	**GPIO31**	U32

B.7 External Bus Interface (EBI)

This section summarizes the signals connected to the EXPM connector (see *Connector EXPM* on page A-7).

Table B-12 lists the address bus signals.

Table B-12 EBI address bus signals

Signal	Pin	Notes	Signal	Pin	Notes
MA0	C18	These signals are also used by the motherboard during configuration. If a design uses these signals, they must be tristated until **GLOBAL_DONE** goes HIGH.	**MA13**	D23	These signals are also used by the motherboard during configuration. If a design uses these signals, they must be tristated until **GLOBAL_DONE** goes HIGH.
MA1	E19		**MA14**	C23	
MA2	D19		**MA15**	A23	
MA3	C19		**MA16**	D24	
MA4	E20		**MA17**	B24	
MA5	C20		**MA18**	E25	
MA6	B20		**MA19**	D25	-
MA7	D21		**MA20**	C25	-
MA8	C21		**MA21**	B25	-
MA9	B21		**MA22**	A25	-
MA10	E22		**MA23**	E26	-
MA11	D22		**MA24**	C26	-
MA12	B22		**MA25**	B26	-

Table B-13 lists the data bus signals.

Table B-13 EBI data bus signals

Signal	Pin	Notes	Signal	Pin	Notes
MD0	D27	These signals are also used by the motherboard during configuration. If a design uses these signals, they must be tristated until **GLOBAL_DONE** goes HIGH.	**MD16**	E23	-
MD1	C27		**MD17**	E27	-
MD2	A27		**MD18**	E24	-
MD3	E28		**MD19**	E21	-
MD4	D28		**MD20**	D29	-
MD5	C28		**MD21**	C33	-
MD6	A28		**MD22**	D32	-
MD7	C29		**MD23**	E30	-
MD8	B29	-	**MD24**	E32	-
MD9	D30	-	**MD25**	E33	-
MD10	C30	-	**MD26**	F29	-
MD11	B30	-	**MD27**	F30	-
MD12	A31	-	**MD28**	F31	-
MD13	A19	-	**MD29**	F33	-
MD14	D20	-	**MD30**	G29	-
MD15	D26	-	**MD31**	G30	-

Table B-13 lists the EBI control signals

Table B-14 EBI control signals

Signal	Pin	Notes
MSC0EN	D13	-
nXCS0	E18	-
nMOE	D18	Also used by the motherboard during configuration. If a design uses this signal, it must be tristated until **GLOBAL_DONE** goes HIGH.
MRDY	E7	-
nMWR0	D11	Also used by the motherboard during configuration. If a design uses this signal, it must be tristated until **GLOBAL_DONE** goes HIGH.
nMWR1	D16	-
nMWR2	E15	-
nMWR3	D10	-
nMCS2	A6	-
nMCS3	D7	-

Note

MCLK is listed in Table B-2 on page B-3.

B.8 ASB/AHB signals

This section summarizes signals related to ASB and AHB signals.

Table B-15 lists the address bus signals.

Table B-15 ASB/AHB address bus signals

Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin
ADDR0	M4	**ADDR8**	R4	**ADDR16**	W5	**ADDR24**	U2
ADDR1	M5	**ADDR9**	R5	**ADDR17**	AD3	**ADDR25**	U1
ADDR2	N2	**ADDR10**	T2	**ADDR18**	AE5	**ADDR26**	V5
ADDR3	N3	**ADDR11**	T3	**ADDR19**	AK2	**ADDR27**	V3
ADDR4	N4	**ADDR12**	T4	**ADDR20**	AF1	**ADDR28**	V2
ADDR5	P2	**ADDR13**	T5	**ADDR21**	AA4	**ADDR29**	W3
ADDR6	P5	**ADDR14**	U3	**ADDR22**	AH4	**ADDR30**	W1
ADDR7	R3	**ADDR15**	V4	**ADDR23**	U4	**ADDR31**	Y5

Table B-16 lists the data bus signals.

Table B-16 ASB/AHB data bus signals

Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin
DATA0	G5	**DATA8**	C1	**DATA16**	F5	**DATA24**	J3
DATA1	H4	**DATA9**	D2	**DATA17**	G1	**DATA25**	J4
DATA2	L5	**DATA10**	D3	**DATA18**	G3	**DATA26**	J5
DATA3	P4	**DATA11**	E2	**DATA19**	G4	**DATA27**	K2
DATA4	R1	**DATA12**	E3	**DATA20**	H2	**DATA28**	K4
DATA5	K5	**DATA13**	F1	**DATA21**	H3	**DATA29**	L1
DATA6	N5	**DATA14**	F3	**DATA22**	H5	**DATA30**	L3
DATA7	B3	**DATA15**	F4	**DATA23**	J2	**DATA31**	M2

Table 5-8 ASB/AHB bus control signals

Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin
CONT0	Y4	**CONT8**	AC3	**CONT16**	AF4	**CONT24**	AJ3
CONT1	Y3	**CONT9**	AC1	**CONT17**	AF3	**CONT25**	AJ2
CONT2	AA5	**CONT10**	AD5	**CONT18**	AF2	**CONT26**	AJ1
CONT3	AA3	**CONT11**	AD4	**CONT19**	AG5	**CONT27**	AK3
CONT4	AA1	**CONT12**	AE4	**CONT20**	AG4	**CONT28**	AL1
CONT5	AB4	**CONT13**	AE3	**CONT21**	AG2	**CONT29**	AL7
CONT6	AB3	**CONT14**	AE1	**CONT22**	AH3	**CONT30**	AL10
CONT7	AC5	**CONT15**	AF5	**CONT23**	AH1	**CONT31**	AL16

B.9 Miscellaneous input/output signals

Table B-17 lists miscellaneous input/output signals.

Table B-17 Miscellaneous input/output signals

Signal	Pin	Notes
nIRQ / **nEXPINT**	K31	**nIRQ** input (in CM position), **nEXPINT** output in expansion position.
nFIQ	P31	**nFIQ** input (only when LM is in CM position).
DBGRQ	AM17	From Multi-ICE or Trace connector. Intended for use when the logic module implements a processor design in the core module position on an Integrator/AP or /SP.
DBGACK	AN17	To Multi-ICE or Trace connector. Intended for use when the logic module implements a processor design in the core module position on an Integrator/AP or /SP.
FPGA_V_TCK	AK16	Virtual JTAG port.
FPGA_V_TMS	AJ16	Virtual JTAG port.
FPGA_TDI	AL15	Virtual JTAG port.
nTRST	W30	Virtual JTAG port.
FPGA_V_TDO	AN15	Virtual JTAG port.
FPGA_V_RTCK	AM16	Virtual JTAG port.
ID0	T31	LM position indication in stack.
ID1	H32	LM position indication in stack.
ID2	L33	LM position indication in stack.
ID3	D31	LM position indication in stack.
SGNT	G31	System bus grant.
SREQ	U33	System bus request.
SLOCK	E31	System bus lock.
nPBUTT	W31	Push button.
nMBDET	V31	Motherboard present indication.

Table B-17 Miscellaneous input/output signals

Signal	Pin	Notes
nSYSRST	W33	System reset from system controller.
nSRST	V30	Reset from Multi-ICE or Trace tool.
GLOBAL_DONE	C13	Indicates that all FPGAs in the system have configured.

B.10 Modem card signals

The MDC connector is not fitted as standard. Table B-18 lists the FPGA pin allocation. This is detailed here in case you fit a connector (AMP part number 3-179397-0). The modem pins are shared with the prototyping grid.

Table B-18 Modem card signals

Signal	MDC pin	FPGA pin	Notes
MONO_OUT	1	N/A	Analog signal connected to jumper J10/J15 (not fitted)
AUD_PWRDWN	2	AJ25	-
GND	3	N/A	-
MONO_PHONE	4	N/A	Analog signal connected to jumper J10/J15 (not fitted)
AUXA_RIGHT	5	N/A	Analog signal connected to jumper J10/J15 (not fitted)
RESVD0	6	AJ18	-
AUXA_LEFT	7	N/A	Analog signal connected to jumper J10/J15 (not fitted)
GND	8	N/A	-
GND	9	N/A	-
5V	10	N/A	-
GND	11	N/A	-
RESVD1	12	AM26	-
GND	13	N/A	-
RESVD2	14	AN23	-
GND	15	N/A	-
PRIMARY_DN	16	AK28	-
3V3	17	N/A	-
5V	18	N/A	-
GND	19	N/A	-

Table B-18 Modem card signals

Signal	MDC pin	FPGA pin	Notes
GND	20	N/A	-
3V3	21	N/A	-
AC_SYNC	22	AM18	-
AC_SDOUT	23	AL29	-
SDATA_INB	24	AL18	-
AC_nRESET	25	AL24	-
SDATA_INA	26	AK18	-
GND	27	N/A	-
GND	28	N/A	-
AC_MSTRCLK	29	AL20	-
AC_BITCLK	30	AN19	-

Appendix C
Mechanical Specification

This appendix contains the specifications for the ARM Integrator/LM-XCV400+ logic module. It contains the following section:

- *Mechanical details* on page C-2.

C.1 Mechanical details

The logic module is designed to be stackable on a number of different motherboards. Its size allows it to be mounted onto a CompactPCI motherboard while allowing the motherboard to be installed in a card cage. Figure C-1 shows the mechanical outline of the logic module.

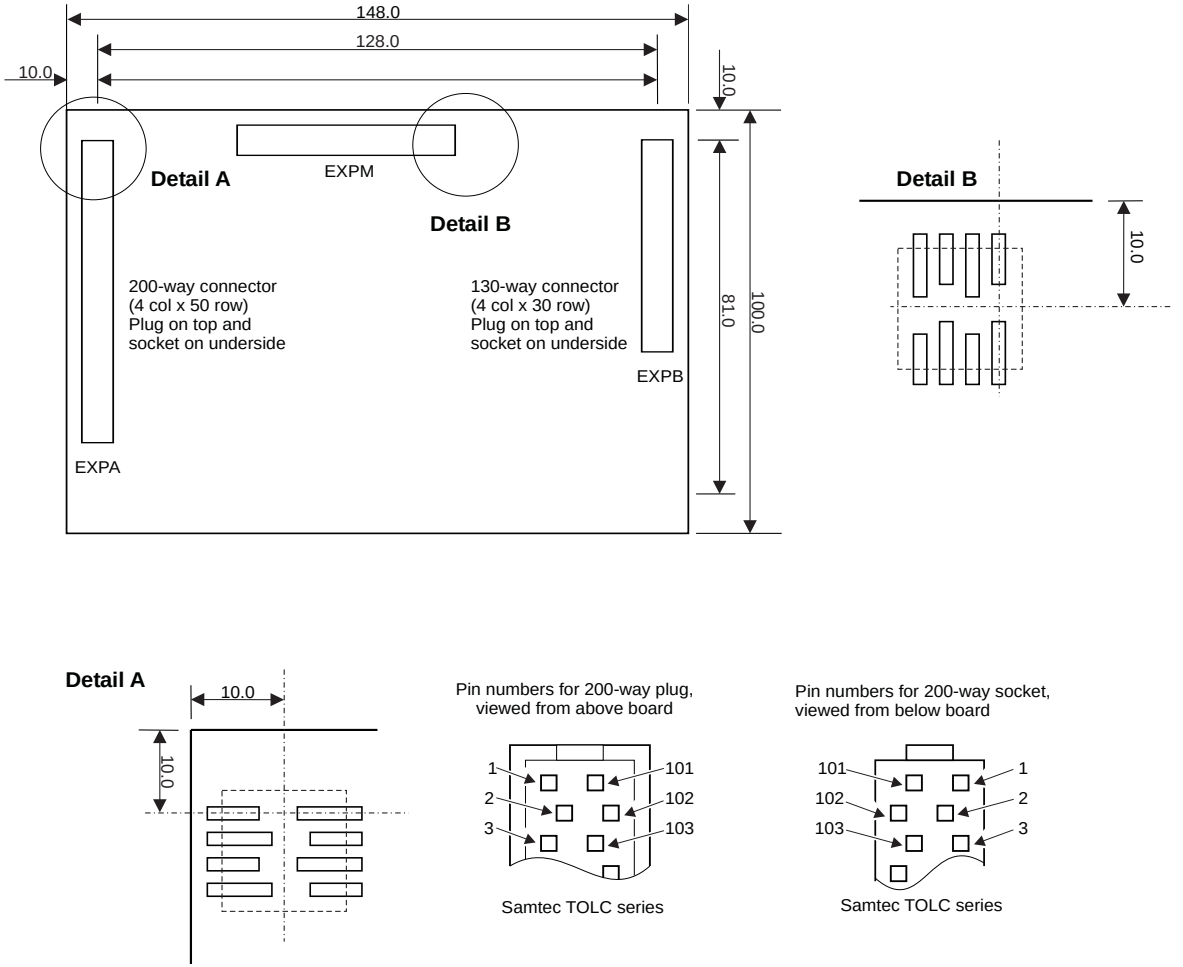


Figure C-1 Board outline

Index

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

A

About this book
  feedback ix
  typographical conventions vii
Architecture
  clock 3-7
  FPGA configuration 3-3
  reset 3-10
  system 1-4

B

Boundary scan programming 3-4
Bus interfaces 3-5

C

Care of modules 2-8
Clock control 3-7
Clock control signals B-2
Clock inputs B-3
Clock signals 3-8
CONFIG link 1-5
Configuration
  FPGA 3-3
Configuration Example 1 5-2
Configuration mode 3-12
Configuration modes, FPGA 3-4
Connecting Multi-ICE 2-6
Connectors
  EXPA 2-3, A-2
  EXPB 2-3, A-5
  EXPM A-7
  HDRA 2-3
  HDRB 2-3
  logic analyzer A-10
  Multi-ICE A-12
  Trace A-11
Control
  clocks 3-8
  interrupts 5-11
  reset 3-10
Core and logic module differences 2-7

D

Diagnostic connectors A-9
DIL switches, setting 2-5

E

Electronic Data Interchange Format 4-3
Enable register, interrupt 5-11, 5-12
Example memory map 5-6
EXPA connector 2-3, A-2
Expansion bus interface 3-6
EXPB connector 2-3, A-5
EXPM signal description A-7

F

Fitting modules to motherboard 2-3
Flash memory 3-17
FPGA configuration 2-5, 3-3
FPGA configuration examples 5-2
FPGA input/output pins 3-2
FPGA pin allocation B-1
FPGA place and route 4-2
FPGA synthesis 4-2
FPGA, description 3-2

G

GPIO signals A-5

H

HDL files 4-2
HDRA connectors 2-3
HDRB connectors 2-3

I

ICS525 3-7
ID register 5-7
Indicators 1-5
Input/output, FPGA 3-2
Integrator memory map 5-6
Integrator/LM-XCV400+ layout 1-3
Interfaces, bus 3-5
Inter-module connectors A-2
Interrupt control 5-11
Interrupt pins 2-7
Interrupt register bit assignment 5-12
Interupt status 5-11

J

JTAG and the FPGA 3-3
JTAG data path 3-14
JTAG signal descriptions 3-15
JTAG signal routing 3-12, 3-13
JTAG support 3-12
JTAG test reset 3-10

L

LEDs B-4
LEDs summary 1-5
Links 1-5
Logic analyzer connectors A-10
Logic analyzer signals B-5
Logic module base address decodes 3-5
Logic module registers 5-7
Logic modules, attaching 2-3
Logic modules, maximum number 2-3

M

Maximum number of modules 2-3
Memory map, example 5-6
Module fitting 2-3
Modules, combined maximium 2-3
Motherboard configuration signals 2-5
Motherboard reset 3-11
Multi-ICE (JTAG) connector A-12
Multi-ICE adaptive clocking 3-15
Multi-ICE connector 3-10
Multi-ICE, connecting 2-6

O

Oscillator divisor registers 5-7
Oscillator lock register 5-9

P

Physical layout 1-3
Pinouts
  EXPA A-2
  EXPB A-5
  EXPM A-7
  logic analyzer connector A-10
  Muilti-ICE connector A-12
  Trace A-11
Place and route 4-2
Preventing damage 2-3
Product feedback ix
Programmable clock sources 3-7
Programming, clocks 3-8
Prototyping grid 1-4
Push button 3-11
Push button interrupt register 5-9

R

Raw status register, interrupt 5-11
Registers
  LM_ID 5-7
  LM_IENCLR 5-10
  LM_IENSET 5-10
  LM_INT 5-7
  LM_IRSTAT 5-10
  LM_ISTAT 5-10
  LM_LEDS 5-7
  LM_LOCK 5-7
  LM_OSC1 5-7
  LM_OSC2 5-7
  LM_SOFTINT 5-10
  LM_SW 5-7
Related publications viii
Reset architecture 3-10
Reset control 3-10

S

Select MAP mode 3-4
Setting the DIL switches 2-5
Signals
  clocks 3-8
  JTAG 3-15
Slave serial mode 3-4
SSRAM 3-17
SSRAM, FPGA pins B-8
Status register, interrupt 5-11
Switches 1-5, 1-6, B-4
Switches register 5-10
Synthesis, FPGA 4-2
System address bus A-3
System architecture 1-4
System bus interface 3-5
System control bus A-3
System reset 3-11

T

Tool flow 4-2
Trace 3-10
Trace (ETM) connector B-6
Trace connector A-11
Trace port 3-2
Typographical conventions vii

U

User LEDs control register 5-9