

# Application Note **101**

## Stacking Integrator Modules

Document number: ARM DAI 0101C

Issued: August 2006

Copyright ARM Limited 2003

# ARM

## Application Note 101 Stacking Integrator Modules

Copyright © 2003 ARM Limited. All rights reserved.

### Release information

The following changes have been made to this Application Note.

#### Change history

Date	Issue	Change
August 2002	A	First release
October 2003	B	Added how to stack core modules with logic tiles Added notes on default values of system bus signals Bundled with example code for CM + IM-LT1 + LT configuration
August 2006	C	Added information to the clock modification section for IM-LT1, including the newer lead free IM-LT1

### Proprietary notice

ARM, the ARM Powered logo, Thumb and StrongARM are registered trademarks of ARM Limited.

The ARM logo, AMBA, Angel, ARMulator, EmbeddedICE, ModelGen, Multi-ICE, ARM7TDMI, ARM9TDMI, TDMI and STRONG are trademarks of ARM Limited.

All other products, or services, mentioned herein may be trademarks of their respective owners.

### Confidentiality status

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

### Feedback on this Application Note

If you have any comments on this Application Note, please send email to [errata@arm.com](mailto: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.

### ARM web address

<http://www.arm.com>

# Table of Contents

<b>1</b>	<b>Introduction .....</b>	<b>4</b>
1.1	The Integrator family .....	4
1.2	Logic Tiles and IM-LT1 Interface Modules.....	4
1.3	Working without an Integrator motherboard .....	4
1.4	Stacking core and Logic Modules or tiles without a motherboard .....	5
1.5	What does the application note cover? .....	6
<b>2</b>	<b>Issues stacking Core Modules and Logic Modules or tiles together .....</b>	<b>7</b>
2.1	Mechanical and electrical constraints .....	7
2.2	Configuring Altera Logic Modules .....	8
2.3	Interrupts .....	9
2.4	B and F buses .....	10
2.5	Virtual TAP controller .....	10
<b>3</b>	<b>Motherboard detection and loop-back of JTAG signals .....</b>	<b>12</b>
3.1	Signal description and routing .....	12
3.2	Stacking mode PCB links.....	13
3.3	Order of Integrator modules in a stack.....	14
<b>4</b>	<b>Signals normally driven by the motherboard.....</b>	<b>16</b>
4.1	AMBA system bus clock .....	16
4.2	System reset .....	17
4.3	FPGA image selection .....	18
4.4	Core Module identification .....	19
<b>5</b>	<b>System bus design considerations.....</b>	<b>20</b>
5.1	Arbiter.....	20
5.2	Default Slave.....	20
5.3	Dummy master.....	21
<b>6</b>	<b>Example HDL code and bit-files .....</b>	<b>22</b>
6.1	Hardware configuration .....	22
6.2	Description of the Example3 HDL.....	22
<b>7</b>	<b>References .....</b>	<b>24</b>

# 1 Introduction

## 1.1 The Integrator family

The Integrator family consists of a series of modules and platform boards that support the development of applications and hardware for ARM processor-based products.

An Integrator system usually consists of an AP motherboard with two separate stacks of Integrator modules: one of Core Modules (CM) and one of Logic Modules (LM)

- The AP motherboard provides the AMBA bus infrastructure for the system as well as non-volatile memory, peripherals and PCI bus interface.
- The Core Modules contain ARM cores, which behave as masters of the AMBA bus, and RAM
- The Logic Modules can be configured to implement hardwired logic, AMBA bus peripherals or synthesizable cores. Logic Modules have the same form factor as Core Modules and are based on Xilinx Virtex or Altera APEX FPGAs.

Another possibility is to use a CP baseboard with a Core Module and one or more Logic Modules on top.

## 1.2 Logic Tiles and IM-LT1 Interface Modules

In 2002 ARM launched a new range of development boards called Logic Tiles, which extend the functionality of Integrator Logic Modules.

Logic Tiles are highly configurable boards based on Xilinx Virtex-II FPGAs. Logic Tiles are smaller in size than Logic Modules and their stacking connectors are not compatible with Integrator.

You can use Logic Tiles with Integrator in order to implement complex AMBA peripherals. Normally the Logic Tiles should be in the Logic Module stack. The Integrator IM-LT1 board provides mechanical and electrical interface between Integrator modules and Logic Tiles.

Logic Tiles are also used to implement Soft Testchip Models (STM) of ARM synthesizable cores. In this case the Logic Tile or tiles synthesize the core and an Integrator IM-LT2 board provides the interface with Integrator. The STM should be connected on top of the Core Module stack.

## 1.3 Working without an Integrator motherboard

An Integrator AP/CP motherboard provides the customer with a working AMBA system and several peripherals. This is very convenient for customers developing software and AMBA slaves and can be used as a starting point to test whole system-on-chip designs.

However, some other customers need to work without an Integrator motherboard for several reasons, for example:

- They need a different memory map or access to the 4GB that the ARM core can address
- They need to test their own AMBA management blocks: arbiter, decoder, default slave...
- They need access to more memory or special peripherals

These customers normally want to use an Integrator Core Module because it provides an AMBA interface to an ARM testchip, but they need a solution different from an Integrator motherboard. The alternatives they have are:

- Design a custom motherboard
- Stack the Core Module together with a Logic Module or tile without a motherboard

In either case the solution is complex and a good understanding of the AMBA specification and Integrator systems is required before commencing this work. In some cases some minor modifications of the hardware are also needed.

## 1.4 Stacking core and Logic Modules or tiles without a motherboard

The Integrator core and Logic Modules were first designed to work either standalone or stacked on top of a motherboard.

Core Modules and Logic Modules can be stacked together without a motherboard provided that a minimum set of the functions of the motherboard's system FPGA are implemented in one of the Logic Modules.

The level of difficulty in stacking modules without a motherboard depends on the type of module. Newer modules include a 'stacking mode' zero-ohm PCB link that can be moved to enable motherboard-less stacking. Older modules were not designed with this feature in mind, and have no such link. Thus, with some combinations of boards, extra wiring must be added to enable this stacking method.

The following table shows what modules have the stacking mode link:

Core Module Name	PCB Number ('x' is revision)	'Stacking mode' link
Integrator/CM9x0T (*)	HBI-0047x	No
Integrator/CM720T	HBI-0050x	No
Integrator/CM7TDMI	HBI-0056x	No
Integrator/CM740T	HBI-0058x	No
Integrator/CM9x6E-S	HBI-0066x (rev C and later)	PCB ref LK8
Integrator/CM10200	HBI-0067x (rev C and later)	PCB ref LK11
Integrator/CM9x0T (*)	HBI-0070x (rev C and later)	PCB ref LK11
Integrator/CM926EJ-S	HBI-0087x	PCB ref LK1
Integrator/CM10200E	HBI-0098x	PCB ref LK1

Logic Module Name	PCB Number ('x' is revision)	'Stacking mode' link
Integrator/LM-XCV400+	HBI-0059x	No
Integrator/LM-XCV600E+	HBI-0072x	PCB ref LK3
Integrator/LM-EP20K600E+	HBI-0073x	PCB ref LK3
Integrator/LM-XCV3200E+	HBI-0079x	PCB ref LK4

**Table 1: Stacking link on Core Modules and Logic Modules**

**Note** (\*) The CM920T/CM940T Core Modules are sold with two possible versions of the PCB

You can also stack a Core Module with a Logic Tile or stack of tiles. In this case an IM-LT1 is needed to provide the interface between the two.

## 1.5 What does the application note cover?

This application note covers the issues that need to be considered when designing your own motherboard or stacking Integrator modules without an AP motherboard. It includes the following:

- Existing constraints when stacking core and Logic Modules or tiles together
- PCB settings and/or modifications needed
- Functionality to implement in a Logic Module, Logic Tile or custom motherboard to have a working system bus
- Example3 HDL code and bit-files provided to stack a Core Module with an IM-LT1 and a Logic Tile

## 2 Issues stacking Core Modules and Logic Modules or tiles together

### 2.1 Mechanical and electrical constraints

#### Mechanical interference:

When stacking modules, the first constraint is mechanical: Core Modules must not be stacked on top of Logic Modules. The two modules fit together but there is mechanical interference from some components, so the boards are bent and strain is placed on the solder joints. This configuration should not be used.

Therefore, in stacks containing Core Modules and Logic Modules, the module at the bottom of the stack is always a Core Module.

For detailed information about mechanical constraints when stacking Integrator modules, see the FAQ entry: “Some Integrator modules will not fit on top of others”

Core Modules and Logic Tiles have different form factor and stacking connectors. An interface board is required to stack the Logic Tile on top of the Core Module. The recommended interface board is an Integrator IM-LT1.

#### Signal rotation:

In the Core Module and Logic Module stacking headers, some of the signals are not routed straight up the stack but are rotated through the stack.

The information carried by these signals is used by the design in the Core Module and Logic Module to know its position in the stack. It also enables the use of a common hardware and firmware design for the Core Modules and Logic Modules regardless their position in the stack.

Some of the signals affected by signal rotation are HBUSREQ, HGRANT, ID and PPRES/EPRES.

The way these signals are routed makes it impossible to have more than 4 Core Modules and Logic Modules stacked together.

However, in Logic Tiles these signals are either:

- not connected
- routed straight up the stack
- connected to the first tile FPGA only, so they are routed by the design inside the FPGA

In any case, the mechanical restriction on the maximum number of tiles stacked together disappears. However, having many Logic Tiles stacked together increment the capacity on the system bus and JTAG signals, so the maximum speed of this signals is reduced.

In summary, you can stack:

- Up to 4 Core Modules and Logic Modules stacked together. For example:
  - 1 CM + 3 LM
  - 3 CM + 1 LM
- Up to 3 Core Modules and Logic Modules plus an IM-LT1 and several Logic Tiles:
  - 1 CM + 2 LM + IM-LT1 + LT + LT + LT...
  - 3 CM + IM-LT1 + LT + LT + LT...

## 2.2 Configuring Altera Logic Modules

This section only applies to Altera Logic Modules (LM-EP20K600E+)

When stacking some modules together the JTAG signals are routed through a longer path and their quality deteriorates. This affects the FPGAs in Altera Logic Modules, so when there are two or more modules in a stack Multi-ICE cannot configure them correctly.

A possible solution is to remove the Logic Module from the stack each time that a new configuration is loaded.

In order to configure Altera Logic Modules when they are in a stack, they need to be modified depending on the total number of boards stacked (see diagram below or LM-EP20K600E+ rev. A circuit schematics):

- One Logic Module: no changes are needed
- Two Logic Modules: substitute the 33-ohm series resistors on the TCK signal (R82, R83, R84) with 100-ohm resistors (see **Error! Reference source not found.**)

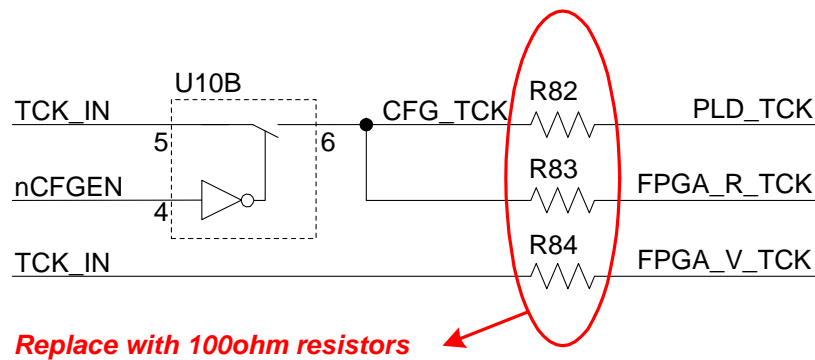


Figure 1: Board changes for two Altera Logic Modules

- More than two Logic Modules: the quality of the TCK signal needs to be improved by placing a buffer on each branch of the TCK tree (see Figure 2). The steps to do this are:
  1. Lift U10A pin2 and place a buffer between the pad and the pin
  2. Lift U10B pin5 and place a buffer between the pad and the pin
  3. Remove the series resistor between TCK\_IN and FPGA\_V\_TCK (R84). If the virtual controller is used, substitute the resistor with another buffer

The buffers can be implemented with spare non-inverting gates on the board (e.g. the OR gates U15B-D) or with additional gate



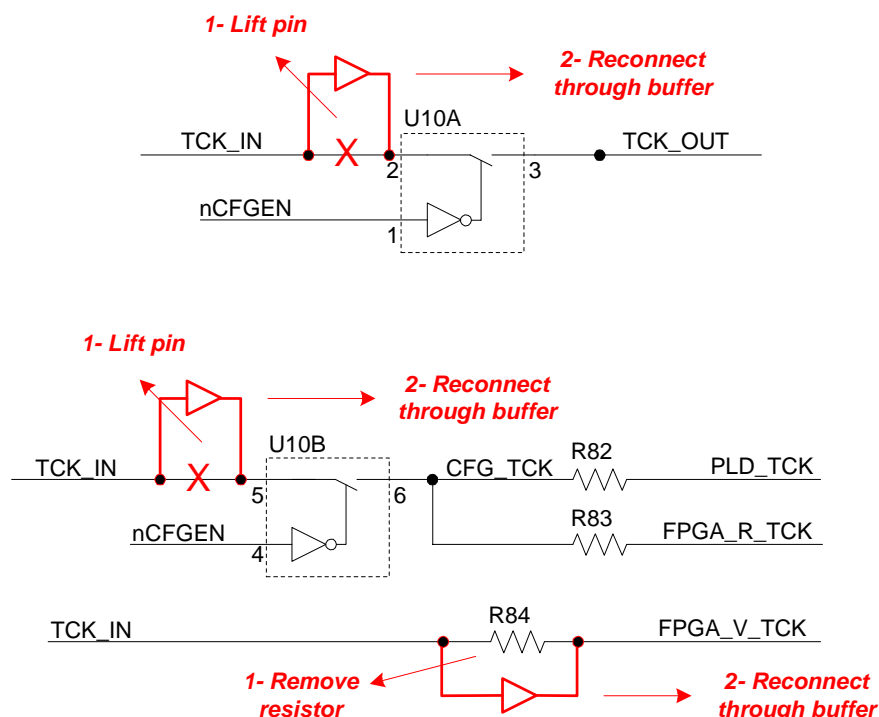


Figure 2: Board changes for more than two Altera Logic Modules

## 2.3 Interrupts

Logic Modules and tiles usually implement peripherals that generate interrupts. Core Modules receive and handle those interrupts. The text below is written for Logic Modules but also applies for a stack of Logic Tiles on top of an IM-LT1.

When a motherboard is present in the system, the system FPGA implements an interrupt controller that monitors the nIRQSRC and nFIQSRC (interrupt source) signals from the Logic Modules and drives the nIRQ and nFIQ (interrupt request) signals to the Core Modules. The following considerations are explained for nIRQ but also apply to nFIQ.

The Core Module FPGA activates the nIRQ input of the ARM core when either:

- nIRQ0 (HDRB socket pin 28) is low and nMBDET is low (motherboard present)
- An interrupt is generated inside the Core Module, i.e. interrupt status register

When Core Modules and Logic Modules are stacked together nIRQ[3:0] of the Logic Modules are connected to nIRQSRC[3:0] of the Core Modules. These signals are rotated in every module, so when a Logic Module activates one of the nIRQSRC pins, one of the Core Modules will be interrupted depending on the positions of the two modules in the stack.

Old Logic Modules (i.e. LM-XCV400+) can only drive nIRQSRC[0], but new Logic Modules and Logic Tiles can drive all nIRQSRC[3:0] so they can select which Core Module they interrupt.

The Core Module's FPGA has internal pull-ups on its nIRQ and nFIQ inputs, so when no Logic Module is driving these signals they are inactive. More than one Logic Module can generate an interrupt for a common Core Module by driving nIRQSRC with open-collector output, that is using wired-AND logic.

For more information, please see the Integrator FAQ entry "How are Interrupts routed in an Integrator system?"

## 2.4 B and F buses

B[31:0] and F[31:0] are a set of signals on the header connectors that are routed straight through the stack. These buses are used to carry system specific signals between modules.

The F bus is connected to the FPGA in Logic Modules and tiles. The B bus is also connected to spare pins of the FPGA in Logic Tiles and new Logic Modules (all but LM-XCV400+)

The use of these buses is defined in the following cases.

- If there is an IM-PD1 or an IM-AD1 board in the stack the B and F buses are used to connect these boards to a Logic Module in the stack.
- When stacking modules on a motherboard the F bus of the Logic Module stack carries the GPIO signals between the FPGA and the Logic Modules.

When stacking several Logic Modules together the F and B buses are available for the user to pass signals between them. However, this may cause problems in future releases of the Integrator modules, when these buses may have new functionality assigned.

On old Core Modules (CM7xx), the B and F buses are left floating. On new Core Modules (CM9xx, CM10xx) those signals are pulled down inside the FPGA. Therefore the user cannot rely on pull-ups in the Logic Modules to assign a default value to those signals.

As a rule of thumb, these signals should always be driven with “strong” signal levels. If the signals are used as a bidirectional bus, the default value of the bus should be fixed with pull-downs, so its value is always defined regardless of the Core Modules present in the stack.

## 2.5 Virtual TAP controller

When the CONFIG link is fitted on any of the modules of the stack, the JTAG chain is routed through the (real) TAP controllers of all the programmable devices of the system. This configuration mode is used to load new images to these devices.

When working in user mode (not configuration mode) the CONFIG link is not fitted and the JTAG chain is routed:

- In Core Modules: only through the ARM core, to allow software debugging
- In Logic Modules and tiles: through a virtual TAP controller implemented inside the FPGA

The virtual TAP controller is necessary to provide JTAG debugging functionality for a core synthesized in the Logic Module's FPGA. The routing of the JTAG chain enables the user to include these synthesizable cores in the chain.

In the Logic Module schematics, the signals routed through the real TAP controller of the FPGA begin with 'FPGA\_R\_', while the signals routed through the virtual TAP controller of the FPGA begin with 'FPGA\_V\_'.

In the Logic Tile schematics the signals routed through the real TAP controller of the FPGA begin with 'C\_' while the signals routed through the virtual TAP controller of the FPGA begin with 'D\_'.

When stacking Core Modules and Logic Modules or tiles together every programmable FPGA in the stack must either:

- Implement a TAP controller
- Provide a through path inside the FPGA from the virtual TDI to the virtual TDO and from the virtual TCK to the virtual RTCK

If this is not done the JTAG signal path will be open, and the ICE will not be able to communicate with the ARM cores on the Core Modules.

The signal routing through the Logic Module's FPGA is implemented in the Example 2 supplied with the Logic Modules and tiles, and Example 3 supplied with this Application Note.

## 3 Motherboard detection and loop-back of JTAG signals

### 3.1 Signal description and routing

The nMBDET signal means “motherboard detected” and is connected through the HDRB/EXPB headers to all the modules in the stack. nMBDET is pulled up on all the modules and grounded in the motherboard, therefore it is active low only when a motherboard is present.

The nMBDET signal is used by:

- The Core Module system bridge inside the FPGA. The bridge gives access to the system bus depending on the value of nMBDET and the address of memory accessed. When the ARM core tries to access the system bus (addresses over 0x11000000) and the motherboard is not present, the bridge aborts the access. If the motherboard is present it passes the accesses to the header connectors onto the system bus.
- The Integrator modules' on-board JTAG switches. These switches loop back the JTAG signals on the module when no motherboard is present and pass them to the module below when the motherboard is present.

The JTAG signals that need to be looped back at the bottom of the stack are:

- TDI: HDRB/EXPB socket pin 52
- TDO: HDRB/EXPB socket pin 53
- TCK: HDRB/EXPB socket pin 49
- RTCK: HDRB/EXPB socket pin 107

The JTAG programmer (e.g. Multi-ICE) drives TDI and TCK and monitors TDO and RTCK.

TDI is routed straight down the stack to the motherboard, where it is connected (looped back) to TDO. From there the signal goes up through all the cores and programmable devices of the stack.

Similarly, TCK is routed through all the synthesizable cores down the stack to the motherboard, where it is connected to RTCK. From there it goes straight up the stack.

In order to keep the JTAG TDI-TDO and TCK-RTCK paths, the module at the bottom of the stack has to loop back these signals when no motherboard is present.

This configuration works when either:

- There is a motherboard which drives nMBDET low and does the JTAG loop back. This is done by the Integrator motherboards and must also be done if the customer designs their own motherboard
- The Core Module or Logic Module is standalone or there is a standalone stack of Logic Tiles on top of an IM-LT1. In these cases nMBDET is high, so the JTAG signals are looped back by the module at the bottom of the stack (Core Module, Logic Module or IM-LT1)

However, this does not work when there is a stack of several modules without a motherboard. In that case, the board at the bottom of the stack must loop back the JTAG signals and drive nMBDET low, allowing the access to AMBA bus peripherals implemented in the Logic Modules

In order to do this the module at the bottom of the stack needs to be modified. This modification is easier when using modules that have a 'stacking mode' PCB link (see table in section 0.

## 3.2 Stacking mode PCB links

Stacking mode links on Core Modules have three pads (A, C, B) and links on Logic Modules have four pads (A, B, C, D)

Core Module links: default position A-C behaves like modules without stacking mode link

Pad A: When high, enables the JTAG loop-back switches. It is pulled up

Pad C: Connected to nMBDET and pulled up

Pad B: Connected to ground

Logic Module links: default position B-C behaves like modules without stacking mode link

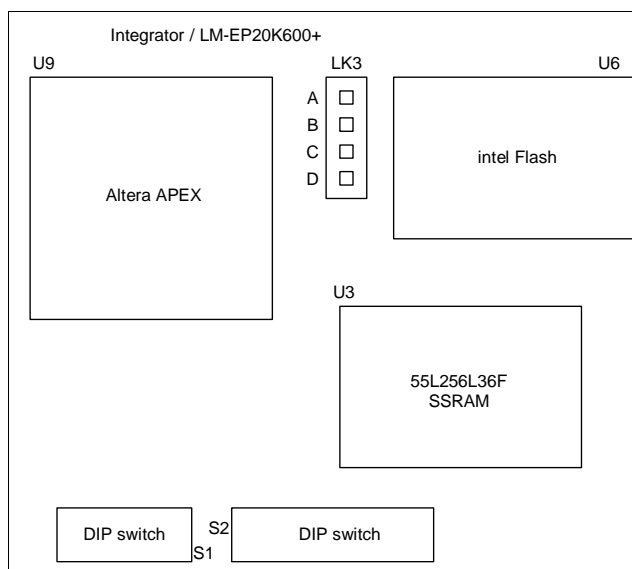
Pad A: Connected to ground

Pad B: Connected to nMBDET and pulled up

Pad C: When high, enables the JTAG loop-back switches. It is pulled up

Pad D: Connected to ground

On the LM-EP20K600E+, pads A, B, C, D of the stacking mode link LK3 are not labeled on the silk screen, but it is easy to determine which pad is which. Looking at the top side of the board, from top to bottom (the PCB being held with the DIP switches along the lower edge), the pads should be labeled A, B, C, D (see Figure 3)



**Figure 3: Stacking link in Altera Logic Module**

### 3.3 Order of Integrator modules in a stack

The circuit schematics and the manuals of the core and Logic Modules describe how to use the stacking mode link depending on the position of the board in the stack.

In this section there is a summary of the settings for each combination of boards in the stack. A stack of Logic Tiles with an IM-LT1 is equivalent to a Logic Module with no stacking link:

1) Stack of Core Modules and/or Logic Modules with a motherboard

Core Modules have default link A-C

Logic Modules have default link B-C

The motherboard drives nMBDET low and loops back the JTAG signals. The JTAG loop-back switches are inactive on all the modules. The cores are allowed to access the AMBA system bus, which is implemented in the motherboard.

2) Standalone Core Module or Logic Module

Core Module has default link A-C

Logic Module has default link B-C

Since there is no motherboard, nMBDET is high. This activates the JTAG loop-back switches on the module, closing the JTAG path. Accesses to the AMBA system bus are aborted.

3) Stack of Logic Modules without a motherboard: at least one module has link

Place the Logic Module which has link at the bottom of the stack. Link in position A-B

The other Logic Modules have default link B-C

The Logic Module at the bottom at the stack behaves like the motherboard, driving nMBDET low and looping back the JTAG signals. The JTAG loop-back switches are inactive on the other modules

4) Stack of Core Modules and Logic Modules without a motherboard: at least one Core Module has link

Place the Core Module with a link at the bottom of the stack. Link in position B-C

The other Core Modules have default link A-C

The Logic Modules are on top of the stack (mechanical constraint) and have default link B-C

The Core Module at the bottom at the stack behaves like the motherboard, driving nMBDET low and looping back the JTAG signals. The JTAG loop-back switches are inactive on the other modules

At least one Logic Module must be in the stack in order to provide AMBA system bus arbitration

5) Stack of Logic Modules without a motherboard: no module has link

The module at the bottom of the stack needs to be modified in order to act like the motherboard, driving nMBDET low for the other modules and looping back the JTAG signals

See case 6 below for the possible solutions

6) Stack of Core Modules and Logic Modules without a motherboard: no Core Module has link

The Core Module at the bottom of the stack needs to be modified in order to act like the motherboard, driving nMBDET low for the other modules and looping back the JTAG signals

The Logic Modules are on top of the stack (mechanical constraint) and have default link B-C

There are two possible solutions for this problem:

- a) Make a custom 'dumb' motherboard that connects nMBDET to ground and loops TDI to TDO and TCK to RTCK. This is the preferred solution because the modules are not modified, so they are available for use with an Integrator AP motherboard without any modification
- b) Solder wires to the signals coming from / going to the HDRB connector of the Core Module: connect nMBDET to ground, TDI to TDO and TCK to RTCK. This modification needs to be undone if the Core Module is stacked on top of a motherboard, else damage to the boards may occur.

## 4 Signals normally driven by the motherboard

Several signals on the HDRB/EXPB connectors are normally driven by the motherboard. Some of these signals are needed to have a working system, so when stacking modules without a motherboard a Logic Module must drive them. If the customer designs a custom motherboard, this board must drive these signals.

### 4.1 AMBA system bus clock

The clock for the AMBA system bus (HCLK) is normally provided by the system FPGA on the motherboard and is taken to all the modules of the stack via the signals SYCLK[3:0]. See schematics of core and Logic Module for more information

The system clock is used by the system bus bridge in the Core Modules and by the AMBA peripherals in the Logic Modules and tiles.

#### Bus clock provided by a Logic Module:

When no motherboard is present, one of the Logic Modules should provide the system clock. The CLK2 buffer on that Logic Module should be enabled by driving low the signal PWRDWN\_CLK2 from the FPGA. A suitable clock frequency should also be set for CLK2.

See section 3.3 of the Logic Module user guide (DUI-0146) to know how to program and enable the clocks.

#### Bus clock provided by a Logic Tile:

Logic Tiles do not have a clock buffer like the one in Logic Modules. Therefore, they cannot drive the four clock signals SYCLK[3:0] at the same time.

When stacking a single Core Module with a Logic Tile without a motherboard, the tile must supply the system bus clock to the Core Module. This signal is SYCLK0 on the Core Module, but due to the signal rotation it is connected to SYCLK3 in the IM-LT1.

If you have a newer, lead free IM-LT1 (board number HBI-0106D), then you will need to fit resistor R206 to complete the clock net. A 33R resistor is present in the sending end of CLK\_GLOBAL on the Logic Tile, therefore R206 should be a zero ohm 0603 size link.

The older IM-LT1 (board number HBI-0106C) does not connect this signal to the tiles, so it needs to be soldered with a wire to the appropriate clock signal of the Logic Tile.

You can do this by configuring the Logic Tile to drive CLK\_GLOBAL with the system bus clock. Then the signal SYCLK3 (pin 43 of EXPB/HDRB) on the IM-LT1 can be wired to the pad of resistor R19 (of IM-LT1) where CLK\_GLOBAL is routed.

Figure 4 below shows the IM-LT1 bottom side. The figure shows in red how to wire SYCLK3 from HDRB to R19.

Although as previously mentioned, there is a sending end resistor, some customers have reported that it is sometimes necessary to fit an additional 33R in series with this wire modification to ensure reliable operation.

See section 3.4 of the Logic Tile user guide (DUI-0186) and the Logic Tile schematics for more information.



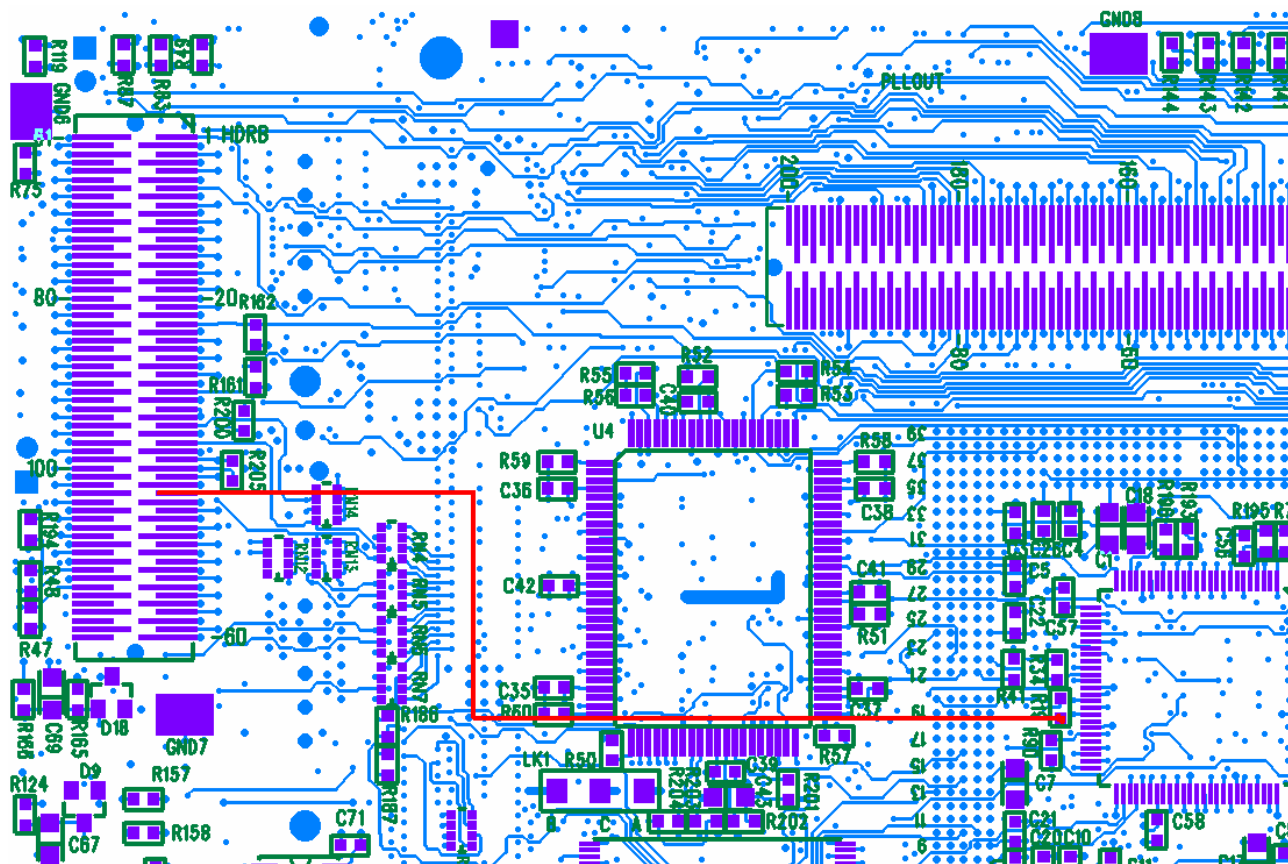


Figure 4: Wiring the system clock in IM-LT1 - bottom side view

## 4.2 System reset

The ARM cores on the Core Modules are reset by the signal ARM\_nRESET, which is provided by the Core Module FPGA. This signal has two sources depending on the value of nMBDET (motherboard present)

- nSRST (pin 110 on HDRB) is used when no motherboard is present
- nSYSRST (the AMBA bus system reset HRESETn, pin 106 on HDRB) is used when there is a motherboard

nSRST is a wired-AND signal and any of the modules in the system can activate it when they detect a cause for a reset, i.e. a software reset. This signal can also be activated via JTAG. JTAG debug tools also monitor nSRST to detect a board reset.

nSYSRST is normally driven by the Integrator motherboard. Since nSYSRST is not connected to the JTAG interface, it cannot be detected by a debugger.

The system FPGA on the motherboard activates nSYSRST when either:

- nSRST is low: this propagates the reset signal to all the elements in the system
- GLOBAL\_DONE is low: this keeps the system reset until all the FPGAs have loaded their configuration. This is important for the correct initialization of the AMBA bus, since the masters should begin to access the bus only after all the slaves (in the Logic Modules) are ready to be accessed

If the customer designs their own motherboard, nMBDET is tied low, so the motherboard must provide a valid nSYSRST. nSYSRST needs to be driven low for at least 2 SYSCLK cycles to reset all the elements on the system bus.

When stacking several modules without a motherboard, a Logic Module or tile must provide nSYSRST.

nSYSRST (HRESETn) is the only active low signal in the AMBA AHB specification, and is the primary reset for all bus elements. The reset may be asserted asynchronously, but is de-asserted synchronously after the rising edge of HCLK.

Normally, this can be achieved with a two-state synchronizer. See example VHDL code below:

```
-----
signal nSYSRST    : std_logic; -- reset controller output
signal nSYSRST1   : std_logic; -- reset controller interm stage

p_Reset : process (nSRST, HCLK)
begin
    if (nSRST = '0') then
        nSYSRST    <= '0';
        nSYSRST1   <= '0';
    elsif (HCLK'event and HCLK = '1') then
        nSYSRST1   <= '1';
        nSYSRST    <= nSYSRST1;
    end if;
end process p_Reset;
-----
```

### 4.3 FPGA image selection

FPGA configuration data for the Core Module, Logic Module and Logic Tile are held in flash memory on each board. These data is loaded into the FPGAs at power-on. Core Modules usually have one FPGA image to work with AHB, one to work with ASB and one to work with AHB-Lite (should only be used to work with Integrator CP, not with a custom motherboard).

There are two pins on the header connectors, namely CFGSEL[1:0], which select one of these two images from flash. The Integrator motherboard drives these signals, but bias resistors are also provided on every module to select a value for them in case no motherboard is present.

The encoding for CFGSEL[1:0] is:

- 00 - ASB
- 01 - Reserved
- 10 - AHB
- 11 - AHB-Lite

CFGSEL[1:0] are sampled at power-up by the bit-streamer PLD, which decides which image is loaded into the FPGA. Therefore these signals must be driven high or low statically. For instance, it is incorrect to drive them with outputs from an FPGA on a custom motherboard.

It is advisable to move the existing bias resistors on CFGSEL[1:0] on all the modules to achieve the desired result. If the signals are wired to a supply rail, this modification needs to be undone before stacking the module on top of a motherboard. Failure to do this could result in damage of the boards.

See the Integrator board schematics for the reference ID of the bias resistors on each board.

## 4.4 Core Module identification

The signals ID[3:0] of the HDRB/EXPB connector (sometimes called HDRID[3:0]) are used by the cores modules to know their position in the stack, so they can respond to the correct SDRAM alias.

This set of signals is driven with the value '1110' by the AP/CP motherboard and is rotated as it passes through the stack, so depending on the position of the '0', each module knows its position in the stack.

The correct bit pattern should be driven by the customer's custom motherboard to tell the Core Modules their position in the stack. This is necessary even if the SDRAM alias is not accessed.

If the system is a stack of Core Module and Logic Modules or tiles without a motherboard, a Logic Module or tile should drive these signals with the pattern '1110' rotated to the right as many times as modules are stacked below it.

More information is available in the "Hardware Description" section of the Integrator/CM documentation.

Position of LM in stack	Value on ID[3:0]
3	1101
2	1011
1	0111
0	1110

**Table 2: How to drive ID[3:0] depending on the position of the LM in the stack**

## 5 System bus design considerations

The AMBA system bus arbitration and management are normally implemented in the system FPGA on the motherboard.

If the customer works with a custom motherboard, it must provide this functionality. In a stack of modules without a motherboard, one Logic Module or tile must behave like the motherboard.

In order to connect a JTAG emulator to the ARM core in a Core Module, the following signals of the system bus need to have a valid state:

- HGRANT for the Core Module must not be permanently 0
- HREADY must not be permanently 0
- HRESP must be either OK or ERROR

If you implement your system in FPGAs it is advisable to begin by driving these signals with fixed values (HGRANT = 1, HREADY = 1, HRESP = OK). Once you can connect to the core with a debugger, you can modify the FPGA logic to implement your real system.

The AMBA management logic must include the blocks below:

### 5.1 Arbiter

The AMBA bus always needs to have an arbiter, which drives the grant lines for all the bus masters.

If there is only one master in the system (i.e. a Core Module) then the Core Module's grant line can be tied high permanently.

The arbiter must take into account that all the AMBA bus control signals are rotated through the stack. This applies to the signals HBUSREQ, HGRANT and HLOCK.

### 5.2 Default Slave

Accesses to every single address on the system bus must invoke a valid AMBA response, even if there is no memory or peripheral using that address. This is entirely necessary, as accesses to unaccounted-for addresses will cause the bus to lock up, and the core to stall permanently.

The only exception is for the Core Module SDRAM alias regions of Core Modules present in the stack. Those accesses will be handled by the logic in the accessed Core Module. There are four regions:

Module number	SDRAM alias address
CM0	0x80000000 – 0x8FFFFFFF
CM1	0x90000000 – 0x9FFFFFFF
CM2	0xA0000000 – 0xAFFFFFFF
CM3	0xB0000000 – 0xBFFFFFFF

**Table 3: Location of SDRAM alias for each Core Module**

In short, the Logic Modules or tiles must provide HREADY and HRESP responses to all addresses except existing Core Module alias regions.

The default slave must ensure that HREADY is driven with '1' during reset, so that devices can begin to be accessed straight after reset. A piece of code like the following can be used:

```

-----
signal nSYSRST    : std_logic; -- system reset

signal HCLK       : std_logic; -- system clock

signal HSEL_LM    : std_logic; -- Output of combinational address
decoder. It is 1 whenever HADDR is not in CM SDRAM alias range

signal RespEnable: std_logic; -- Enables tri-state buffer to drive
HRESP, HREADY

signal ReadEnable: std_logic; -- Enables tri-state buffer to drive HDATA

p_ReadEnSeq : process (HCLK, nSYSRST)
begin
    if (nSYSRST = '0') then
        RespEnable <='1';
        ReadEnable <='0';
    elsif (HCLK'event and HCLK = '1') then
        if (HREADY = '1') then
            if (HSEL_LM = '1') then
                RespEnable <='1';
                ReadEnable <= not (HWRITE);
            else
                RespEnable <= '0';
                ReadEnable <= '0';
            end if;
        end if;
    end if;
end process p_ReadEnSeq;
-----

```

### 5.3 Dummy master

At power up all the FPGAs in the system load their configuration in parallel. When the last FPGA finishes loading its configuration the system comes out of reset and the ARM cores begin to execute code.

The AMBA bus specification requires that there is always a master driving the bus control signals. This is necessary as floating bus lines might result in no slave response, hence no new master could be granted, and the system would hang.

Therefore, when the arbiter FPGA finishes loading its configuration it must grant the bus to a master. The problem is that at that time all the AMBA masters are in reset.

A solution is to implement a 'dummy master' of the system AMBA bus in the Logic Module that performs the AMBA bus arbitration. At reset the bus is granted to this dummy master, so it holds the bus in IDLE state until the bus masters come out of reset. This way masters are only assigned the bus when they request it, else it is assigned to the dummy master.

## 6 Example HDL code and bit-files

Together with this document we include example HDL code, synthesis and place&route scripts and bit-files for an LT-XC2V6000 Logic Tile. With this configuration the Logic Tile can be stacked together with an IM-LT1 interface module and a Core Module without a motherboard.

The HDL code is provided in both VHDL and Verilog. It can be used as a starting point for more complex systems based on Logic Modules, Logic Tiles or a custom motherboard.

The example provided is called Example3 and it is based on the Example2 provided in the Logic Tile installation CD. The HDL code has the minimum number of modifications needed to use the Logic Tile with a Core Module without a motherboard.

### 6.1 Hardware configuration

The Example3 bit-files can be programmed on an LT-XC2V6000 Logic Tile with a Multi-ICE unit and the utility Progcards. Please see section 4 of the Logic Tile user guide (DUI-0186) for information about this procedure.

The Logic Tile must be stacked on top of an IM-LT1 interface module. The IM-LT1 is stacked directly on top of a Core Module.

The IM-LT1 needs to be modified so that the Logic Tile can drive the Core Module's system clock. This is explained in section 4.1 of this document.

The Core Module stacking link needs to be moved so that the JTAG signals are looped back at the bottom of the stack. See sections 3.2 and 3.3 of this document for more information.

By default, the CFGSEL[1:0] bias resistors select AHB system bus configuration in all boards. However, you should check that the correct resistors are fitted. For more information, see section 4.3 of this document.

### 6.2 Description of the Example3 HDL

Example3 is based on the Example2 provided in the Logic Tile installation CD. Example2 is described in detail in sections 5.3 and 5.4 of the Logic Tile user guide (DUI-0186).

The following modifications have been done on the Example2 code:

- Drive the system clock with a signal from a Logic Tile oscillator.

CLK0 is used to generate the system clock. This signal is buffered to CLK\_GLOBAL\_OUT, which is wired to SYSCLK3.

CLK\_GLOBAL\_IN is used as the system bus clock in the design inside the Logic Tile.

The frequency of the system bus clock can be changed by programming the Logic Tile configuration registers.

- Drive high the Core Module's HGRANT

- Generate nSYSRST from nSRST synchronized with the system clock  
This is done with a 2-stage synchronizer. The Core Module FPGA ensures that nSRST is driven low until the test chip's PLLs have locked.
- Drive HDRID with the right pattern: "0111"
- Drive HREADY = '1' during reset, so the system bus can be accessed straight after reset
- Change the address decoder, so that the Logic Tile's SSRAM is aliased at address 0. The resulting memory map is:
  - The Logic Tile registers are at address 0xC0000000
  - The Logic Tile interrupt controller is at address 0xC1000000
  - The Logic Tile SSRAM is at address 0xC2000000 and 0x00000000
  - Abort accesses to any other areas of the memory map
- Drive SYS\_nIRQ3 and SYS\_nFIQ3 of the IM-LT1 to take into account the signal rotation. Drive it with open-collector buffers, so that multiple interrupts can be AND-wired

You can find the modifications in the HDL code by searching the string **\*\*Example3\*\***.

## 7 References

Additional information can be found in the following documents. They can be downloaded from the ARM website – [www.arm.com](http://www.arm.com).

- AMBA bus specification rev2
- Schematics, netlists and bill of materials of the Integrator boards
- Integrator/CM (Core Modules), Integrator/LM (Logic Modules), Integrator/AP (AP motherboard) and Versatile/LT (Logic Tiles) user guides
- Integrator Frequently Asked Questions (Integrator FAQ)