# Application Note 119

## Implementing AHB Peripherals in Logic Tiles

Document number: ARM DAI 0119E
Issued: January 2006


ARM

**Application Note 119**
**Implementing AHB Peripherals in Logic Tiles**



### Release information

The following changes have been made to this Application Note.

**Change history**

| Date | Issue | Change |
|---|---|---|
| April 2004 | A | First release – only logic tiles on top of an Integrator motherboard |
| May 2004 | B | Extension to logic tiles on top of Versatile/PB926EJ-S and logic tiles on top of a core module with no Integrator motherboard. Example2 hardware description and programmer's model included in the application note. |
| July 2004 | C | Use AHB-APB bridge from ADK to remove bug in the old one<br>Change to structure of boardfiles folder. Correction of small errors in documentation. |
| September 2004 | D | Clocking scheme for Synchronous and Asynchronous bridges added. Flattening of the design and improved constraints file allows the Versatile design to work up to 37MHz now. File names modified to reflect the standard naming convention and AN119. |
| January 2006 | E | Getting started section added |

### Proprietary notice



### 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 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

# 1 Introduction

## 1.1 Purpose of this application note

The Integrator and Versatile families of ARM development boards enable customers to implement AMBA AHB masters and slaves.

AMBA AHB is a specification intended for an internal bus, in which all masters and slaves are inside an ASIC or FPGA.

In AHB the outputs from all the masters are multiplexed, so that only the master that has been granted the bus can access it. Similarly, the outputs from all the slaves are multiplexed, so that only the slave that is being accessed puts its response on the bus.

This kind of approach is not appropriate in a modular development system, in which different masters and slaves are in different chips or even boards. Implementing AHB in a system like this would require a huge number of signals connecting the chips and boards.

In order to solve this problem, ARM development boards implement a tri-state version of the AHB bus.

This application note explains the tri-state implementation of AHB and shows how to implement AHB masters and slaves in logic tiles, so that they can be connected to the Integrator system bus or the Versatile/PB926EJ-S M1, M2 and S buses.

The application note also includes working example Verilog and VHDL, pre-built FPGA configuration images and test software for the peripherals in the logic tile.

This revision of the application note includes information and example HDL for three possible configurations:

- A typical Integrator system, with an Integrator/AP or CP baseboard, a core module on the HDRA/HDRB stack and an Integrator/IM-LT1 interface module plus a logic tile on the EXPA/EXPB stack
- A typical Versatile system, with a Versatile/PB926EJ-S baseboard and a logic tile
- A motherboard-less Integrator system, consisting of a core module, an Integrator/IM-LT1 and a logic tile stacked together

## 1.2 Getting Started

When using this application note with a baseboard, ensure that the baseboard is programmed and functioning correctly first. Refer to the relevant documentation for further information (for example PB926EJ-S user guide). Once you have done this please follow these steps to program the FPGA image in the Logic Tile with the image provided with this application note.

1. Depending on the system you are using plug the Logic Tile on as follows.

   - Standalone – plug the IM-LT1 on the Core module and the Logic Tile on the IM-LT1.
   - AP – plug the IM-LT1 on the AP Logic module slot and the Logic Tile on the IM-LT1.
   - CP – plug the IM-LT1 on the Core module and the Logic Tile on the IM-LT1.
   - PB926EJ-S – plug the Logic Tile on the Logic Tile slot.

2. Fit the CONFIG jumper link on the PB926EJ-S (J32) or IM-LT1 (J10).
3. Connect RVI or Multi-ICE to the to the JTAG ICE connector (PB926EJ-S J31 or IM-LT1 J9), or a USB cable to the PB926EJ-S USB Debug Port (J30).

4. Check the external supply voltage.

   - Standalone – connect power to the Core module power terminal connector (3V3 2A, 5V 1A).
   - AP – connect an ATX power supply to J3 on the AP board (200W).
   - CP – connect +12V to J4 on the CP board (positive on center pin, +/-10%, 35W).
   - PB926EJ-S – connect +12V to J28 on the PB926EJS board (positive on center pin, +/-10%, 35W).

5. Power-up the boards. The '3V3' and ‘5V’ LEDs should both be lit.
6. If using Multi-ICE, run Multi-ICE Server, press ctrl-L and load the relevant manual configuration file from the \boardfiles\multi-ice directory. Depending on the version of Multi-ICE used it may also be necessary to add new devices to Multi-ICE. Please refer to \boardfiles\irlength_arm.txt for information on how to do this.
7. If using the USB connection, ensure that your PC has correctly identified an ARM® RealView™ ICE Micro Edition device is connected to the USB port. If the Windows operating system requires a USB driver to be installed please refer to \boardfiles\USB_Debug_driver\readme.txt.
8. If using Real View ICE (RVI), you must ensure that the RVI unit is powered and has completed its start-up sequence (check the LEDs on the front panel have stopped flashing).

9. You can now run the relevant ‘progcards’ utility for the connection you have prepared above.

   - progcards_multiice.exe for your Multi ICE connection
   - progcards_rvi.exe for your RealView ICE connection
     When using RVI select the target RVI box you are using.

10. Select the option for the system you are using. The utility will report its progress, it may take several minutes to download. A successful configuration download will be terminated with the message “Programming Successful” .
11. Power down the boards.
12. Set the configuration switches to load Logic Tile FPGA image 0 (S2 on the Logic Tile set to all OFF).
13. Remove the CONFIG jumper link, and power-up the boards. Ensure the 'FPGA_OK’ (Logic Tile D6) LED is lit. On the PB926EJ-S the Character LCD should show the Firmware and Hardware versions indicating that the Boot monitor firmware is running. On the AP or CP systems a terminal can be connected to serial port A (CP serial port 0) to show that the Boot monitor firmware is running, please refer to the baseboard user guide for more information.
14. The system will now be fully configured and ready for use.

## Tri-state AHB implementation in ARM development boards

Figure 1 shows a typical implementation of an AHB system inside an ASIC. This AHB system consists of three masters and three slaves.

The left block diagram shows how the outputs of the masters are multiplexed. The output of the multiplexer is routed to the inputs of the slaves. The multiplexer is controlled by the system arbiter, so only one master accesses the bus at one time.

The right block diagram shows how the outputs of the slaves are multiplexed. The output of the multiplexer is routed to the inputs of the masters. The multiplexer is controlled by the system address decoder, so only one slave is accessed at one time.

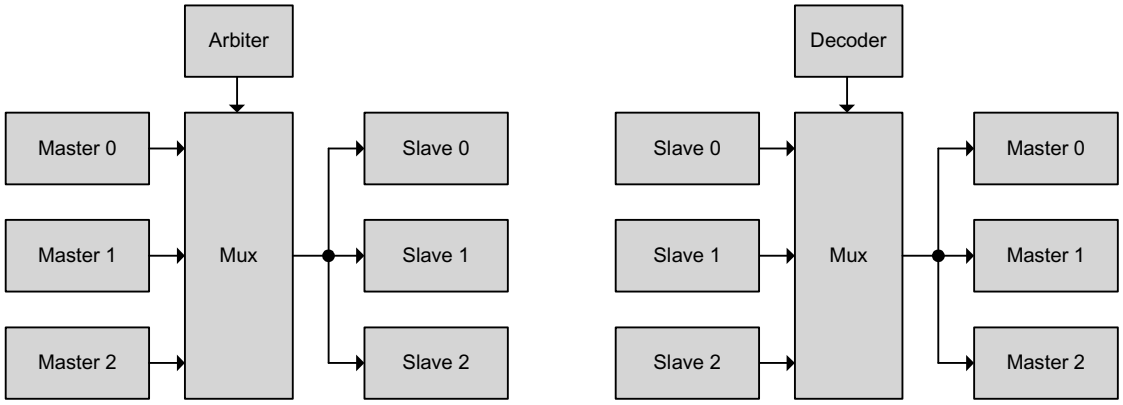


**Figure 1: AHB masters and slaves in an ASIC**

Figure 2 shows how the same system would normally be implemented with development boards. It is important to note that conceptually this implementation in development boards is exactly the same as the implementation in the ASIC.

In this type of system the system bus is routed to all the boards in the system, and is accessed with tri-state buffers. In this example configuration the design is split into three boards.

The left block diagram shows how the outputs from the masters are routed to the system bus.

If there is only one master in a board, the board tri-state multiplexers are enabled when that master is granted the bus.

If there are two or more masters in a board, their outputs are multiplexed. The board tri-state buffers are enabled when one of the masters in the board is granted the bus.

The signals from the system bus drive the inputs of the slaves directly.

It is important to note that there must be a single system bus arbiter which grants the bus to the different masters. This is the only way to guarantee that only one master accesses the bus at a given time.

The way slaves access the system bus is equivalent, as shown in the right block diagram.

The main difference is that there is only one arbiter in the whole system, but there is one address decoder in each one of the boards that implement slaves.

Each address decoder must check if one of the slaves in the board is being accessed. If that is the case, it must enable the tri-state buffers that give access to the system bus. This must be done at the correct phase of the access.

If there are two or more slaves in the same board, the decoder must also control the multiplexing of their outputs.

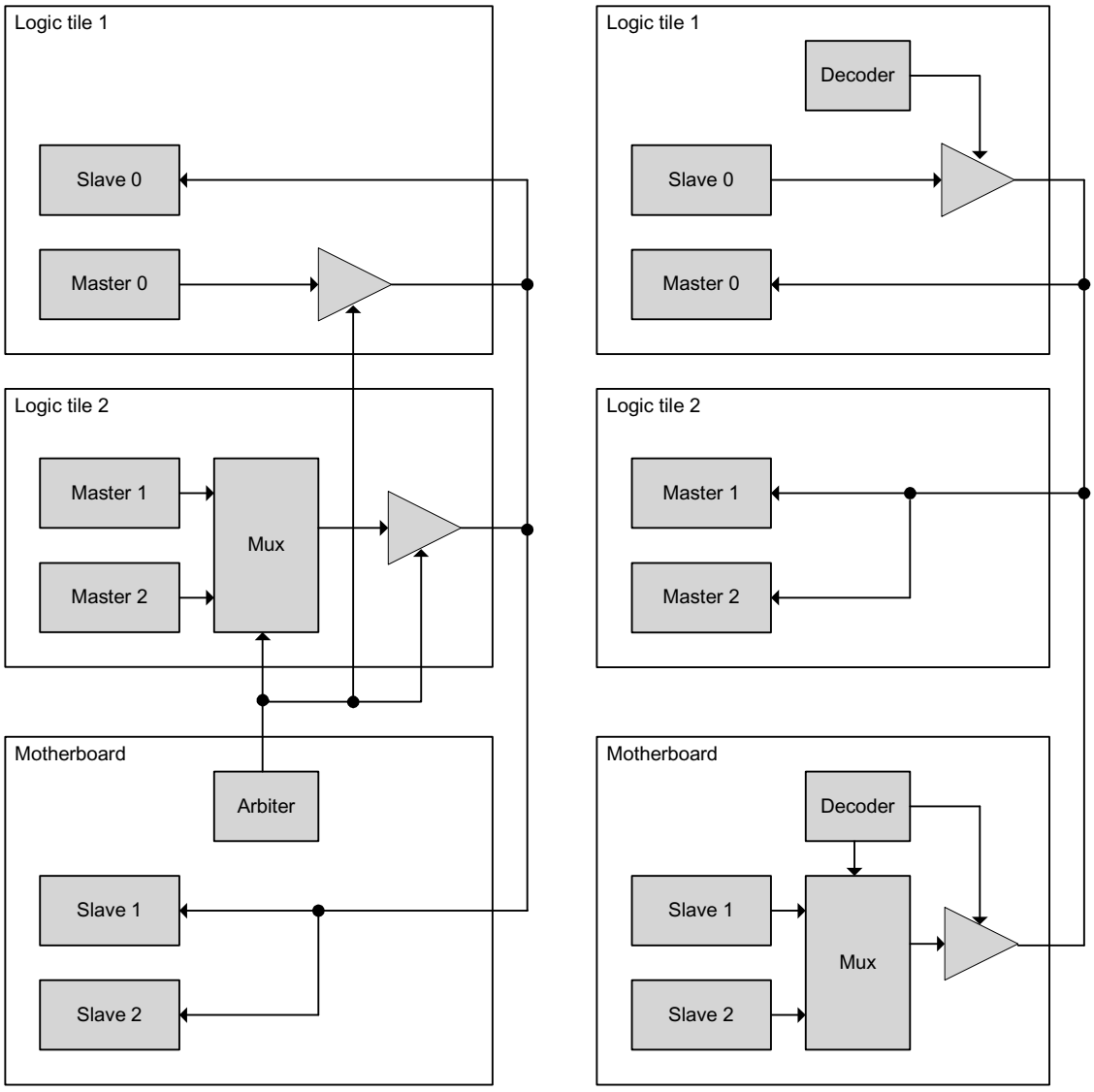


**Figure 2: AHB masters and slaves in development boards**

The example in Figure 2 is only one possible implementation of this system in development boards. There are several alternatives to this configuration, which are explained later in this document.

## 1.3 Accessing the AHB system bus with tri-state buffers

Since AHB is a pipelined bus, the arbitration, address and data phases of each transfer happen in different cycles. The control logic for the tri-state buffers must take this into account, so that the buffers are enabled in the correct clock cycle.

Figure 3 shows the arbitration, address and data phases in an AHB transfer.

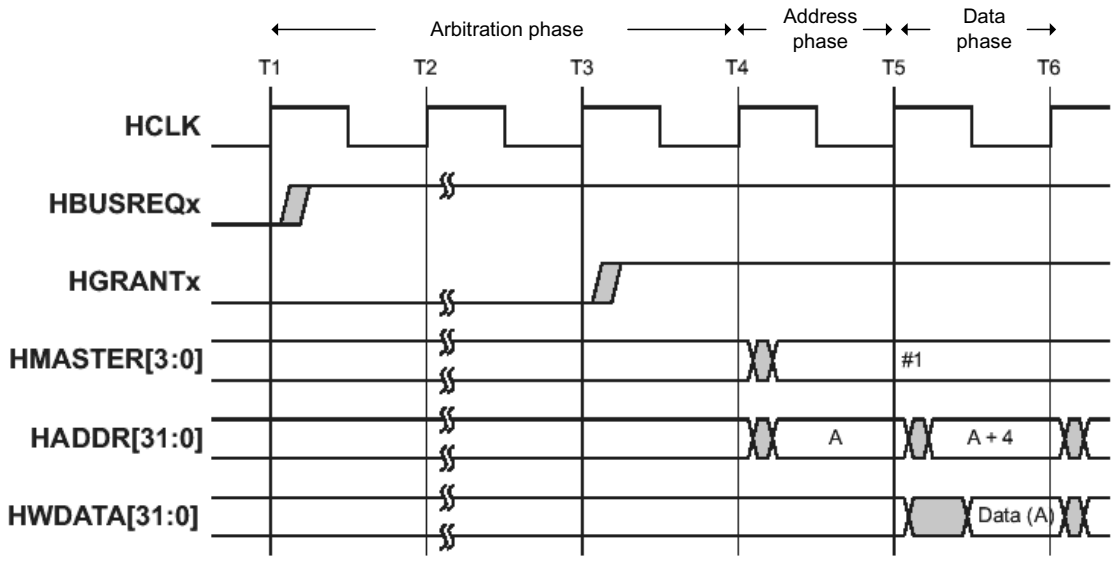


**Figure 3: Phases in an AHB transfer with no wait states**

Figure 4 shows how to connect the signals from the tri-state system bus to the AHB masters in a logic tile.

The master inputs HRESP and HREADY are connected directly from the system bus. HRDATA is connected directly to HDATA in the system bus. HDATA contains the data read from a slave during a read cycle.

The master outputs, HADDR and control signals, are multiplexed and drive the signals of the system bus with tri-state buffers. These buffers are enabled if one of the masters in the logic tile is granted the bus during the address phase of this cycle.

HDATA is driven by HWDATA with a tri-state buffer. This buffer is enabled when one of the masters in the logic tile is granted the bus during the data phase of this cycle, and if the current access is a write access.

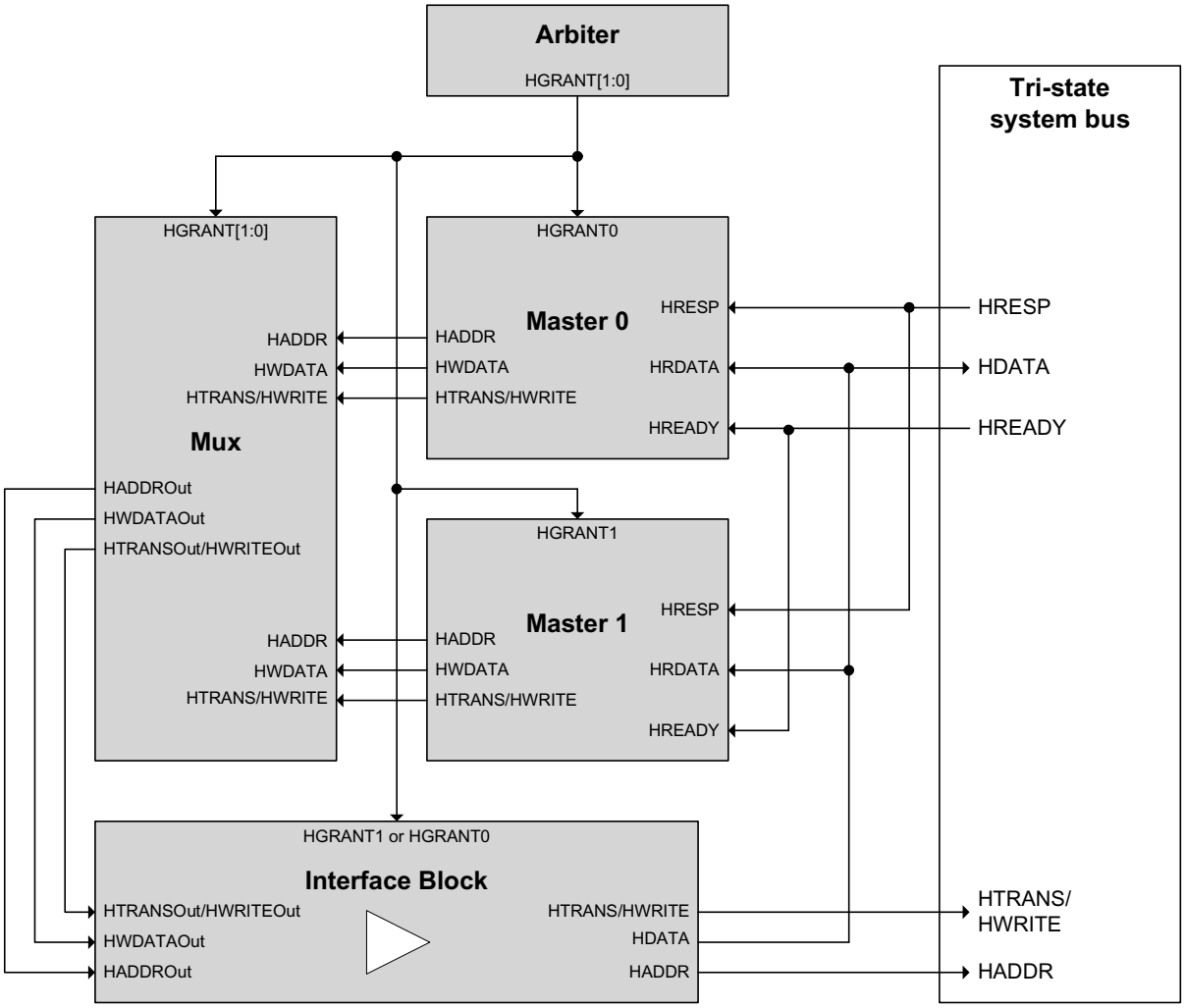


**Figure 4: Connections of AHB masters**

Figure 5 shows how to connect the signals from the tri-state system bus to the AHB slaves in a logic tile.

The address and control signals, which are slave inputs, are connected directly from the system bus.

HREADYIn is connected directly to HREADY from the system bus. HREADYIn indicates if the last transfer in the system bus has finished, so it is needed by all the AHB slaves.

HWDATA is connected directly to HDATA in the system bus. HDATA contains the data written by a master into a slave during a write cycle.

The slave outputs HRESP and HREADY are multiplexed and drive the signals of the system bus with tri-state buffers. These buffers are enabled if one of the slaves in the logic tile is accessed during the address phase of this cycle.

HDATA is driven by HRDATA with a tri-state buffer. This buffer is enabled when one of the slaves in the logic tile is accessed during the data phase of this cycle, and if the current access is a read access.

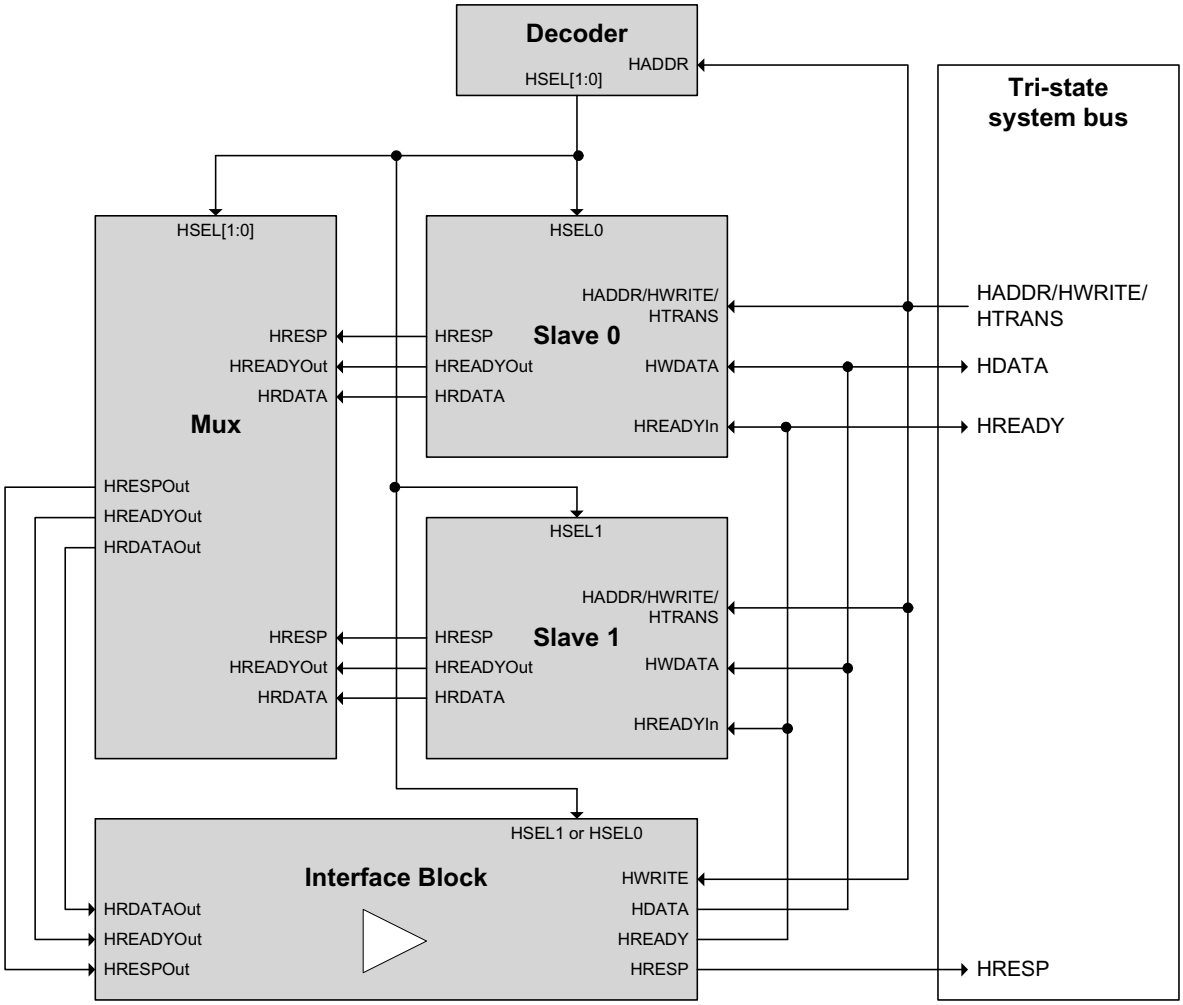


**Figure 5: Connection of AHB slaves**

In systems like Integrator, both masters and slaves are connected to the same system bus, so the HRDATA and HWDATA signals must also be multiplexed. This is shown in Figure 6.

HDATA is driven from HRDATA when HWRITE is low during the data phase of the current cycle. Otherwise HDATA is driven from HWDATA.

The tri-state buffers that gives access to HDATA on the system bus is enabled only if a logic tile master is granted the bus and it is a write cycle, or if a logic tile slave is accessed and it is a read cycle.

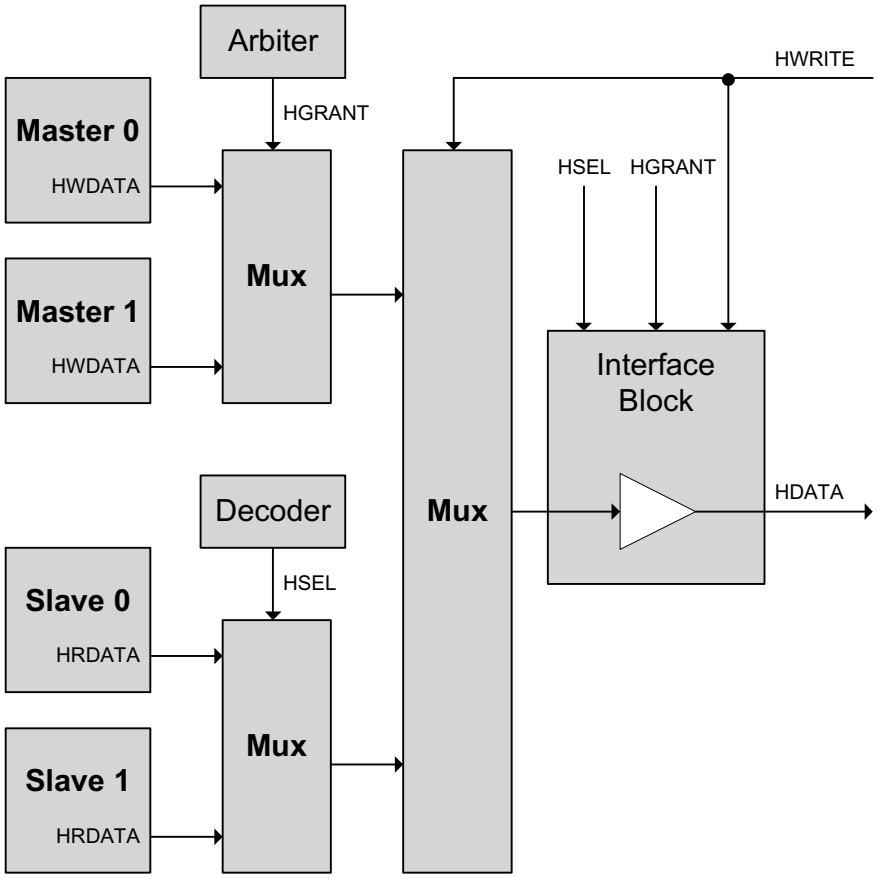


**Figure 6: Multiplexing HRDATA and HWDATA into HDATA**

# 2 Using logic tiles in an Integrator system with a motherboard

## 2.1 The Integrator family of boards

An Integrator system normally consists of:

- One or more Integrator core modules, each one containing an ARM processor and RAM. Part of the core module memory map gives access to the Integrator system bus. This system bus connects all the boards in the system
- An Integrator/AP motherboard, which provides an arbiter and default master for the Integrator system bus, as well as non-volatile memory and peripherals
- One or more Integrator logic modules, which can be configured to implement AMBA bus peripherals or synthesizable cores.

Instead of an Integrator/AP motherboard you can use an Integrator/CP baseboard. The resulting system is almost the same from the logic module or logic tile point of view. The main difference is that a system based on Integrator/CP can only have one master.

Logic tiles have equivalent functionality to logic modules. The main differences are:

- Logic tiles have a bigger Xilinx Virtex-II FPGA , with many more gates and pins
- Logic tiles have higher density stacking connectors, which allow routing more signals between tiles. They also include fold-over and through bus switches to improve routing of signals between tiles
- Logic tiles have a smaller form factor
- Logic tiles have special routing of clocks and JTAG signals

Logic tiles can be connected to an Integrator motherboard with an Integrator/ IM-LT1 interface module.

The Integrator system bus connects all these boards, so that all the masters and slaves in the system are connected to the same bus.

This way, the ARM processors in core modules can access peripherals on the motherboard or in logic modules.

A master in a logic module can access the core module SDRAM, the devices on the motherboard and also peripherals implemented in the logic module itself or in other logic modules.

## 2.2 Buses and clocks

In Integrator there is only one system bus, which is connected to all the boards in the system. The system bus clock is buffered into the signals SYSCLK[3:0]. These signals are routed to the HDRA/HDRB and EXPA/EXPB stacks, so that each core module or logic module get a clock signal.

The Integrator/IM-LT1 interface module routes the system bus from the EXPA/EXPB connectors to the logic tile Z bus. This bus is routed straight up in the stack of logic tiles, so all the logic tiles in the stack have direct access to the system bus.

Therefore the simplest way to implement a design with several logic tiles is the following:

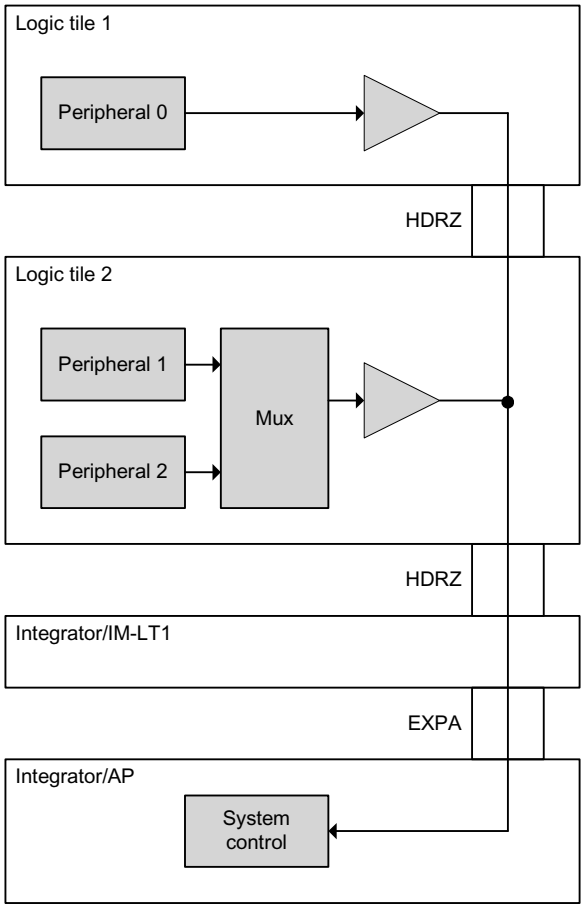


**Figure 7: AHB peripherals in logic tiles in an Integrator system**

Figure 8 shows how the system bus clock is routed in Integrator.

The motherboard buffers the system bus clock into four signals: SYSCLK[3:0]. These signals are routed to the EXPB connector. The Integrator/IM-LT1 only connects SYSCLK1 and SYSCLK2 to the HDRZ connectors. These signals are only connected to the two logic tiles on top of the IM-LT1, as shown in the figure.

CLK_IN_MINUS1 and CLK_IN_MINUS2 are delay-matched signals, so the system bus clock seen by the two logic tiles is the same.

If you need more than two logic tiles in your system we suggest that you use the CLK _GLOBAL signal to clock the system, since it is routed to all the logic tiles in the stack.

The bottom logic tile must route CLK_IN_MINUS1 into CLK _GLOBAL_OUT. It is very important that this path has minimum delay, since it will affect the timing between the logic tiles and the motherboard. The logic in all the logic tiles (including the bottom one) should be clocked with the signal in CLK _GLOBAL_IN.

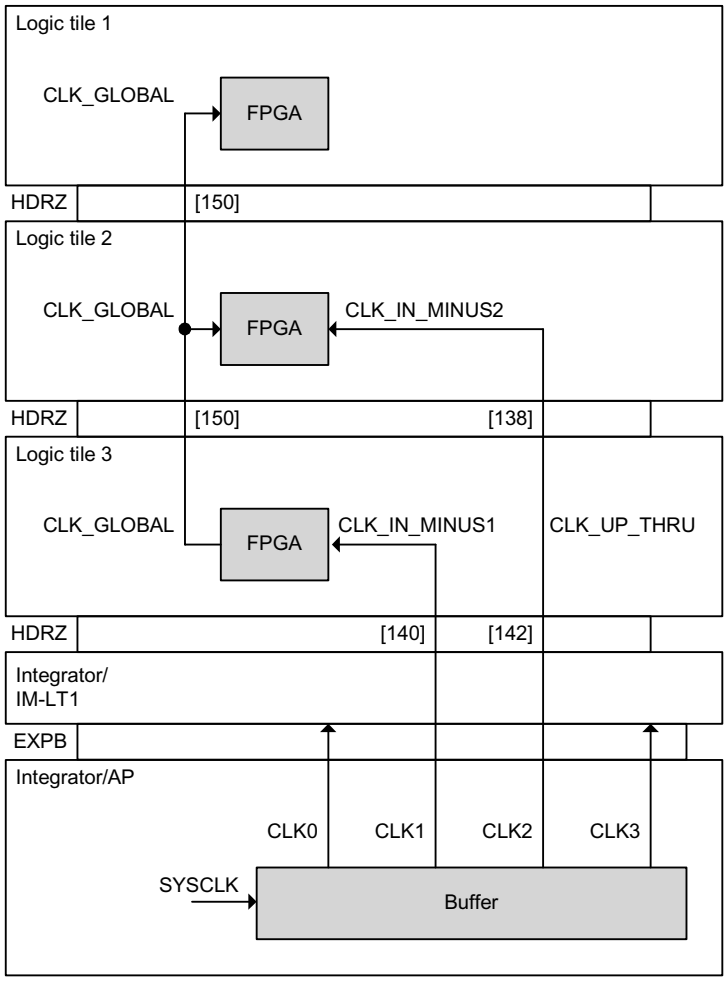


**Figure 8: Routing of the system bus clock in Integrator**

## 2.3 Implementing AHB slaves

In an Integrator system there are ranges of addresses in the memory map reserved for core and logic modules, which are shown in Table 1. The address depends on the stack where the module is located and its position in the stack.

| Platform \ Position in stack | 0 - bottom stack | 1 | 2 | 3 - top stack |
|---|---|---|---|---|
| Integrator/CP | N/A | 0xD0000000-<br>0xDFFFFFFF | 0xE0000000-<br>0xEFFFFFFF | 0xF0000000-<br>0xFFFFFFFF |
| Integrator/AP<br>(HDRA/HDRB) | 0x80000000-<br>0x8FFFFFFF | 0x90000000-<br>0x9FFFFFFF | 0xA0000000-<br>0xAFFFFFFF | 0xB0000000-<br>0xBFFFFFFF |
| Integrator/AP<br>(EXPA/EXPB) | 0xC0000000-<br>0xCFFFFFFF | 0xD0000000-<br>0xDFFFFFFF | 0xE0000000-<br>0xEFFFFFFF | 0xF0000000-<br>0xFFFFFFFF |

**Table 1: Address reserved for a logic module depending on its position in the stack**

If one master tries to access an address mapped to a module which is no present in the stack, the system controller FPGA responds with an error response. The logic module indicates its presence in the stack by driving the local nPRES[0] signal low.

The logic module knows what address it must respond to by the value in the signals ID[3:0]. These signals are also rotated in the stack, so depending on which one is tied low, the logic module knows its location in the stack. This is shown in Table 3.

In an Integrator system with a motherboard, an IM-LT1 interface module and several logic tiles, this may require a different approach.

The IM-LT1 drives nPRES[0] low, so by default 256MB are reserved for the slaves implemented in the stack of logic tiles. This does not depend on the number of logic tiles present in the stack.

If the system requires a greater address range to be mapped to the logic tiles, some of the signals nPRES[3:1] must be driven low.

nPRES[3:1] can be driven by the logic tile at the bottom of the stack, since they are connected to its signals XL[46:44]. These signals are not connected to other logic tiles in the system.

The logic tiles can also use the information in ID[3:0] to respond to a different address range depending on the position of the IM-LT1 in the stack of logic modules. If the position of the IM-LT1 is fixed, then it is simpler to make the logic tiles respond to a fixed address range.

ID[3:0] can be accessed by the logic tile at the bottom of the stack as XL[65:62]. If some slaves are implemented on logic tiles up in the stack, and position-dependant address decoding is being implemented, then ID must be routed through the FPGA to XU[65:62] on the top HDRX header.

## 2.4 Implementing AHB masters

In systems based on Integrator/AP, the system bus arbiter is located in the system controller FPGA on the Integrator/AP motherboard.

This arbiter performs the bus arbitration between the PCI system and up to 5 AHB masters in core modules, logic modules or logic tiles. Each master is connected to the arbiter with a set of HBUSREQ, HGRANT and HLOCK signals.

In a system with core modules and logic modules each module uses its local HBUSREQ0, HGRANT0 and HLOCK0 signals. Depending on the stack where the module is located and its position in the stack, the module gets a different master number in the system bus arbiter.

| Platform \ Position in stack | 0 – bottom of stack | 1 | 2 | 3 – top of stack |
|---|---|---|---|---|
| Integrator/AP (HDRA/HDRB) | Master 0 | Master 1 | Master 2 | Master 3 |
| Integrator/AP (EXPA/EXPB) | Master 4 | Master 3 | Master 2 | Master 1 |

**Table 2: Master number assigned to a module depending on its position in the stack**

The Integrator/IM-LT1 gives the stack of logic tiles access to the arbitration signals HBUSREQ[3:0], HGRANT[3:0] and HLOCK[3:0].

Masters implemented in tiles should normally use the arbiter in the system controller FPGA. In order to avoid conflict with the core modules, the first master should use HBUSREQ0, HGRANT0 and HLOCK0. The second master should use HBUSREQ1, HGRANT1 and HLOCK1, etc.

HBUSREQ[3:0], HGRANT[3:0] and HLOCK[3:0] are connected to the XL and YL buses of the bottom logic tile. If one or more masters are implemented in other logic tiles, then the bottom logic tile must route the necessary signals through the FPGA up the stack.

When there are more than 4 masters in the system it may be necessary to add an extra arbiter in the logic tile, which arbitrates between the masters inside the stack of logic tiles.

It is important to note that the system bus arbiter only needs to be used when the masters in the logic tiles access resources in the system bus: core module SDRAM or memory or peripherals on the baseboard or logic modules.

If the logic tile masters only need to access slaves inside the logic tile, a different approach can be used, as shown in Figure 9.

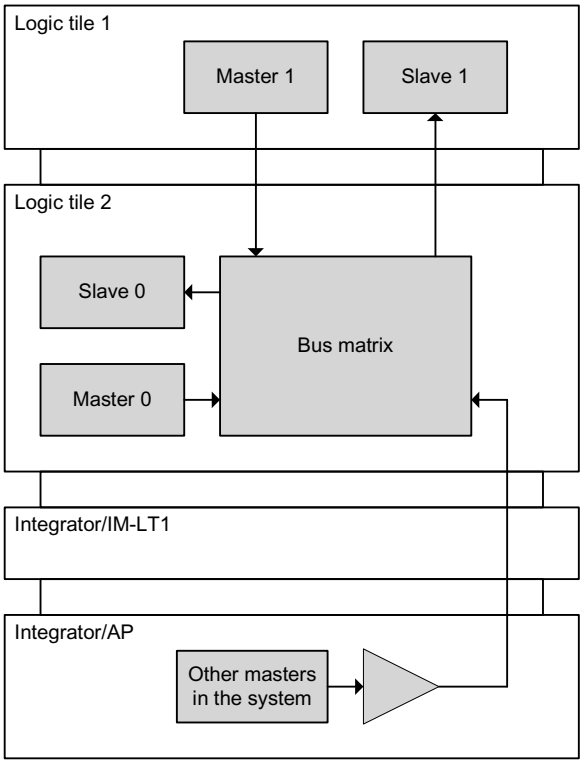


**Figure 9: Bus matrix in a logic tile on top of Integrator**

This type of system can also be used to implement masters in logic tiles with an Integrator/CP baseboard. This is because on Integrator/CP the system bus is AHB-Lite, so only the core module can be a master on that bus.

# 3 Using logic tiles in an Integrator system without a motherboard

## 3.1 Why work without an Integrator motherboard?

An Integrator AP or CP motherboard provides the customer with a working AHB system and several peripherals. This is very convenient for customers developing software and AHB 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 entire 4GB that the ARM core can address
- They need to test their own AHB 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 AHB interface to an ARM testchip.

A possible configuration they can use is a stack of one or more core modules (bottom of the stack) plus an IM-LT1 interface module with one or more logic tiles on top.

This solution is more complex than the standard Integrator system with a motherboard, so a good understanding of the AHB specification and Integrator is required before commencing this work. In some cases some minor modifications of the hardware are also needed.

For general information about stacking core modules and logic modules and/or logic tiles together, see *Application Note 101: Stacking Integrator Modules*.

This section explains the special considerations when implementing AHB masters and slaves in a logic tile on top of a core module without an Integrator motherboard.

## 3.2 AMBA system bus clock

The clock for the system bus is normally provided by the system controller FPGA on the motherboard. In a system without a motherboard one of the logic tiles must provide the system bus clock to the rest of the system.

Due to the design of the Integrator/IM-LT1, logic tiles cannot drive the system clock SYSCLK0-3. This happens because the signals CLK_IN_MINUS1 and CLK_IN_MINUS2 are connected to global clock inputs of the logic tile FPGA, which cannot work as output pins.

In this type of system one of the logic tiles should generate the system bus clock from one of its internal oscillators, for example CLK0. This clock should be routed through the logic tile FPGA to CLK_GLOBAL, so that all the logic tiles on the stack can use it.

The clock routing through the stack of CM + IM-LT1 + LTs is shown in Figure 10.

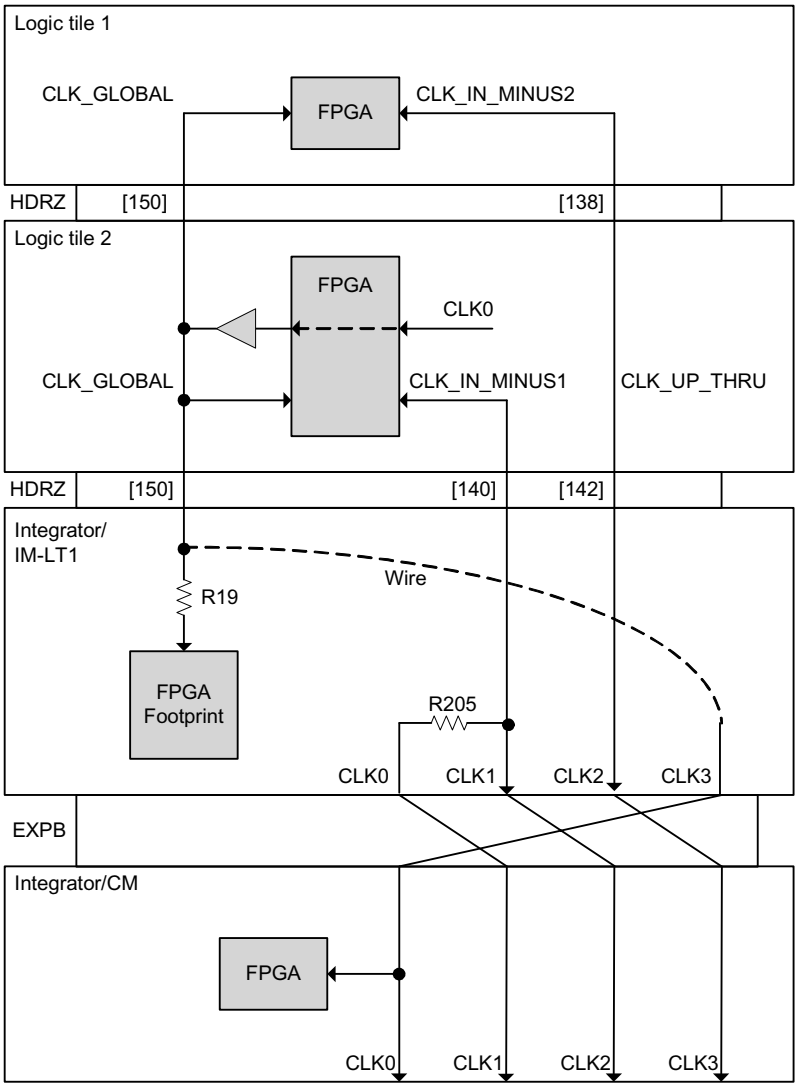


**Figure 10: Integrator system without a motherboard: clock routing**

CLK_GLOBAL must be wired to SYSCLK3 on the IM-LT1 to provide a clock to a core module underneath the IM-LT1. This can be done by soldering a wire between SYSCLK3 (pin 43 of EXPB/HDRB) and the pad of resistor R19 (of IM-LT1) where CLK_GLOBAL is routed.

Figure 11 below shows the IM-LT1 bottom side. The figure shows in black how to wire SYSCLK3 from HDRB to R19.

If there are more core modules underneath, you should solder wires between CLK3 and CLK2, CLK1 or CLK0. This can be done straight at the EXPB connector.

See the Hardware Description Section of the LT-XC2V4000+ user guide (DUI-0186) and the logic tile schematics for more information.

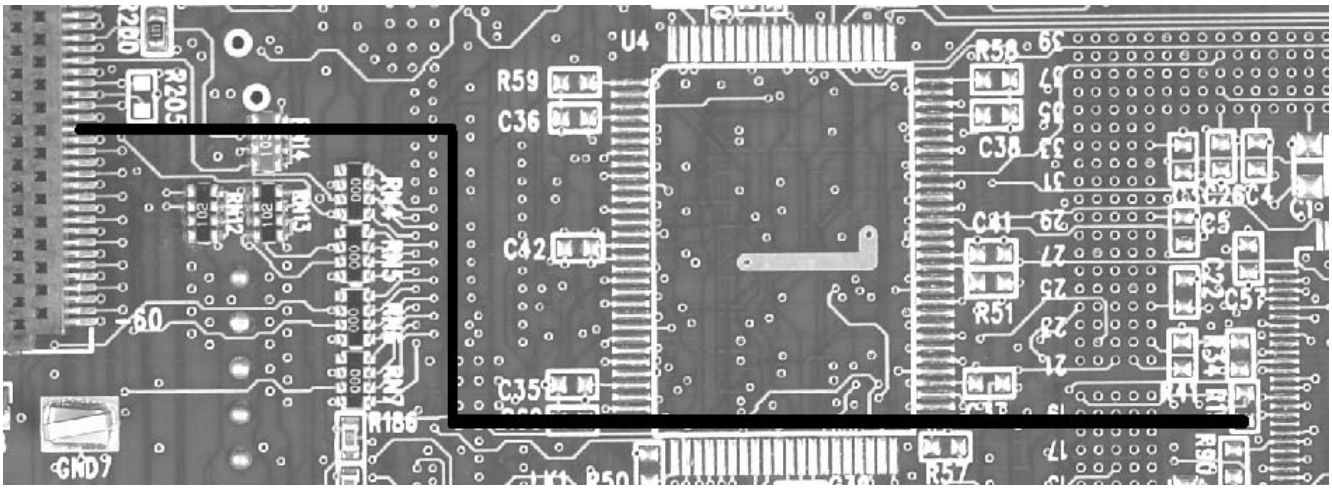


**Figure 11: Wiring the system clock on IM-LT1 - bottom side view**

## 3.3 System reset

nSYSRST is the Integrator system reset. This signal resets all the AHB peripherals in the system as well as the ARM processors. To conform to the AHB specification nSYSRST must be de-asserted synchronously with the system bus clock.

nSYSRST can only be driven by the system reset controller, which is normally implemented on the Integrator motherboard. In a system without a motherboard nSYSRST must be implemented in one of the logic tiles.

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, for example a software reset.

nSRST is driven low automatically by the core module FPGA until GLOBAL_DONE is high, that is until all the FPGAs in the system have loaded their configuration.

nSRST can also be activated via JTAG by an ICE box. JTAG debug tools also monitor nSRST to detect a board reset.

Sources of reset implement in logic tiles should activate nSRST with open collector outputs.

The logic tile can generate nSYSRST from nSRST with a two-stage 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;
----------------------------------------
```

## 3.4 Core module identification

The signals ID[3:0] of the HDRB/EXPB connector (sometimes called HDRID[3:0]) are used by the core modules to know their position in the stack, so they can map the SDRAM alias at the correct address. This information can also be used by the software to know the location of the core module SDRAM alias.

This set of signals is normally driven with the value ‘1110’ by the AP/CP motherboard and is rotated as it passes through the stack.

In an Integrator system without a motherboard the bottom logic tile (the tile straight on top of the IM-LT1) must drive ID[3:0]. The pattern ‘1110’ must be rotated to the right as many times as modules are stacked below the IM-LT1.

| Position of IM-LT1 in stack | Value on ID[3:0] |
|---|---|
| 3 | 1101 |
| 2 | 1011 |
| 1 | 0111 |
| 0 | 1110 |

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

## 3.5 Implementing AHB slaves

In order to conform to the AHB specification, the system bus slaves must respond to all the addresses between 0x0 and 0xFFFFFFFF.

In a typical Integrator system the memory map is pre-defined, so slaves in logic tiles are mapped to a fixed range of addresses.

In an Integrator system without a motherboard, the logic tiles must implement slaves that respond to all the addresses in the memory map except those mapped to the core

module SDRAM alias. Normally the logic tile should contain a default slave that responds with HRESP=ERROR to all the unused addresses of the memory map.

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

**Table 4: Address of SDRAM alias depending on position of core module in the stack**

Another important issue in Integrator systems without a motherboard is the initial value of HREADY after reset.

In an AHB system inside an ASIC, the HREADYOut outputs from the slaves are multiplexed, so HREADY is always driven by one of the slaves. At reset all the slaves must drive their HREADYOut outputs high, so the system is initialized correctly.

In a tri-state implementation of AHB, the slaves drive HREADY with tri-state buffers. The buffers are only enabled when one slave in the board is being accessed. Straight after reset no slave is being accessed properly, so all the tri-state buffers are disabled.

In Integrator HREADY is pulled down, so HREADY is low straight after reset. This prevents the system from booting up, since no master can generate a new transaction on the system bus until HREADY goes high.

The way to solve this problem is to use one of the boards as a “pre-selected slave” after reset. This board drives HREADY and HRESP straight after reset, until the first AHB data cycle is generated on the bus.

In a typical Integrator system, this pre-selected slave is implemented on the AP motherboard. In a system without a motherboard, one of the logic tiles must act as the pre-selected slave.

The example HDL provided with this application note shows how to modify the control logic for the tri-state buffers (generation of `RespEnable`) to make the board work as a pre-selected slave.

## 3.6 AHB masters and arbitration

The AHB system bus always needs to have an arbiter, which drives the grant lines for all the bus masters. This master is normally implemented in the Integrator/AP motherboard.

In an Integrator system without a motherboard the HGRANT lines for the bus masters must be driven by one of the logic tiles.

There are two possible scenarios:

1- The core module is the only master on the system bus

This is the case if there are no AHB masters in the logic tiles.

This also applies to a solution in which the logic tile masters only need to access slaves inside the logic tiles, so they do not access the core module SDRAM alias.

The best solution in this case is to use a bus matrix inside the logic tiles, as shown in figure

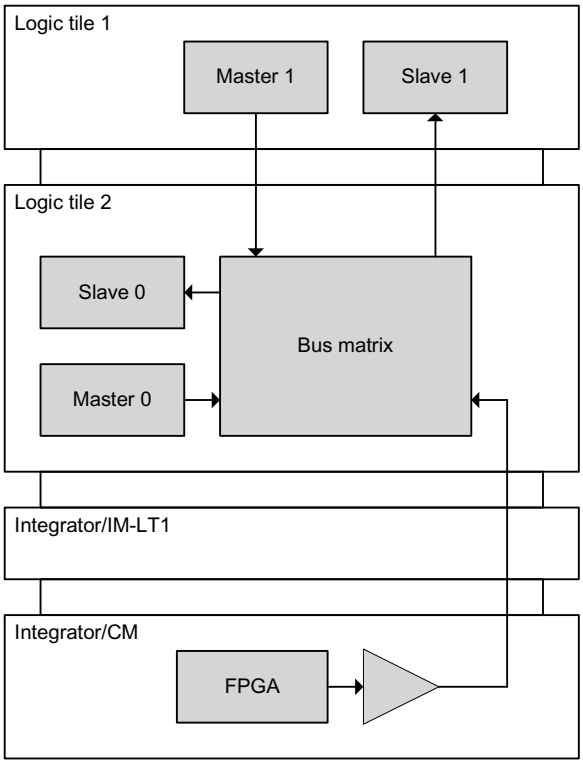


**Figure 12: Bus matrix in a logic tile in an Integrator system without a motherboard**

In this scenario the core module is the only master on the system bus. Therefore the logic tile only needs to drive its corresponding HGRANT signal permanently high.

The core module uses HGRANT0, but due to signal rotation this is connected to HGRANT3 on the IM-LT1.

2- Logic tile masters have access to the system bus

In this case the masters in the logic tiles access the system bus with tri-state buffers. The logic tile must implement a bus arbiter that performs the arbitration between the masters inside the logic tile and the core module.

This configuration is shown in Figure 13.

This arbiter must take into account that the arbitration signals HBUSREQ, HLOCK and HGRANT are rotated through the stack.

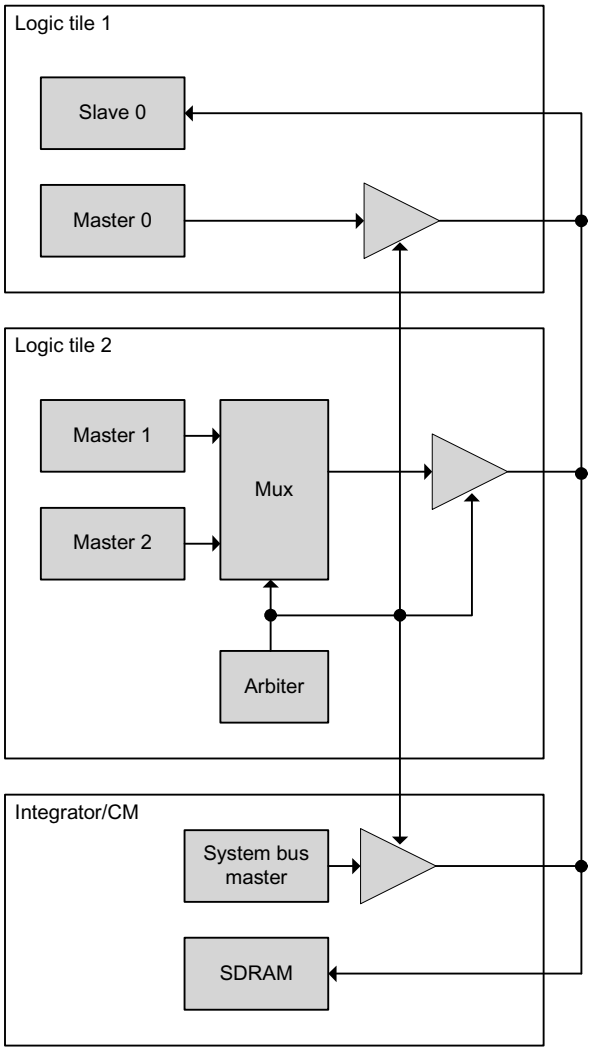


**Figure 13: Arbitration in an Integrator system without a motherboard**

This application note includes example HDL for two Integrator systems without a motherboard. One of them only implements only AHB slaves, as in scenario 1. The other one implements AHB slaves and masters connected to the system bus, as in scenario 2.

An important consideration when implementing an arbiter in logic tiles is how the system exits reset.

During reset (nSYSRST = 0) the arbiter in the logic tile must grant the bus to a master that drives HTRANS = IDLE, so that the system can boot correctly straight after reset.

If no master in the system behaves that way, a “dummy master” must be included in the logic tile. This dummy master drives HTRANS with a fixed IDLE value, so it can be granted the bus at reset.

# 4 Using logic tiles with Versatile/PB926EJ-S

## 4.1 The Versatile family of boards

A Versatile system normally consists of:

- A Versatile baseboard, such as the Versatile/PB926EJ-S. This board contains a development chip with the ARM processor, a bus matrix and a number of peripherals. On the board there is volatile and non-volatile memory, and also an FPGA, which implements more peripherals.
- One or more logic tiles, which can be configured to implement hardwired logic, AHB peripherals or synthesizable cores.

The development chip on the Versatile/PB926EJ-S interfaces external peripherals with three external buses:

- AHB M1 is an external master bus, so the development chip behaves as a bus master. The AHB M1 is connected to the baseboard FPGA and logic tiles, but the baseboard FPGA does not connect any slave to this bus.
- AHB M2 is an external master bus, so the development chip behaves as a bus master. The AHB M2 is connected to the baseboard FPGA and logic tiles. The baseboard FPGA has its slaves connected to this bus.
- AHB S is an external slave bus, so the development chip behaves as a slave of the bus. The AHB S bus is connected to the baseboard FPGA and logic tiles. The baseboard FPGA has the PCI master connected to this bus.

The development chip is based around an internal bus matrix.

- The AHB M1 and M2 bridges inside the development chip are slaves of the bus matrix, mapped at fixed addresses of the memory map.
- The AHB S bridge is a master of the bus matrix, and has its own bus layer.

All masters (including external masters) have access to the M1 and M2 external buses. For details about what slaves can be accessed by each master, please see the Versatile/PB926EJ-S documentation.

## 4.2 Versatile/PB926EJ-S buses: control signals

When connecting a logic tile on top of a Versatile/PB926EJ-S baseboard, some signals must be driven by the logic tile.

The Versatile/PB926EJ-S uses the nTILE_DET signal from the logic tile to detect if logic tiles are present or not. Some control signals for the AHB bridges are not driven by the baseboard when a logic tile is present, so they must be driven by the design in the logic tiles.

These signals are:

- AHB M1 bus:
    - HGRANTM1: Grant signal for the AHB M1 bridge. It can be used to attach more than one master to AHB M1
    - HREADYM1 and HRESPM1: From slaves connected to AHB M1
- AHB M2 bus:
    - HGRANTM2: Grant signal for the AHB M2 bridge. It can be used to attach more than one master to AHB M2
    - HREADYM2 and HRESPM2: From slaves connected to AHB M2. These signals must be enabled only when the master on AHB M2 accesses slaves in the logic tile
- AHB S bus:
    - HSELS: Selection signal from the address decoder, which indicates that the AHB S bridge on the development chip is being accessed. This signal can be used to connect other slaves to the AHB S bus.
    - HMASTLOCKS: Indicates a locked transaction on the AHB S bus.
    - LTHGRANT: Grant signal for the PCI master in the Versatile/PB926EJ-S FPGA.

## 4.3 Versatile/PB926EJ-S buses: standard operation

In order to comply with the AHB specification, the slaves connected to a bus must respond to the whole range of addresses. In a system with a Versatile/PB926EJ-S baseboard and one or more logic tiles this has the following implications:

- The logic tile must respond to the 4GB address range in bus M1
- The logic tile must respond to addresses 0x0 - 0x03FFFFFF and 0x14000000 - 0x1FFFFFFF in bus M2. These addresses are the only ones which are not mapped to any slave in the Versatile/PB926EJ-S FPGA.

Normally a default slave is used to respond to the addresses not mapped to real slaves. The default slave simply drives HREADY = 1 and HRESP = ERROR, so it generates an abort when it is accessed.

The example HDL code provided with this application note implements a standard system in a logic tiles, consisting of the following elements:

- Slaves connected to bus M1 and mapped at address 0xC0000000. Addresses from 0x80000000 to 0xFFFFFFFF are mapped to bus M1 inside the development chip, so these slaves are accessible from all the masters in the system.
- Default slaves respond to accesses to other addresses in bus M1 and to addresses 0x0 - 0x03FFFFFF and 0x14000000 - 0x1FFFFFFF in bus M2.
- Any masters implemented in logic tiles are connected to bus S. These masters can access the resources inside the development chip through the bus matrix, as well as those on buses M1 and M2.

Figure 14 shows a standard system in which the masters and slaves are synthesized in several logic tiles.

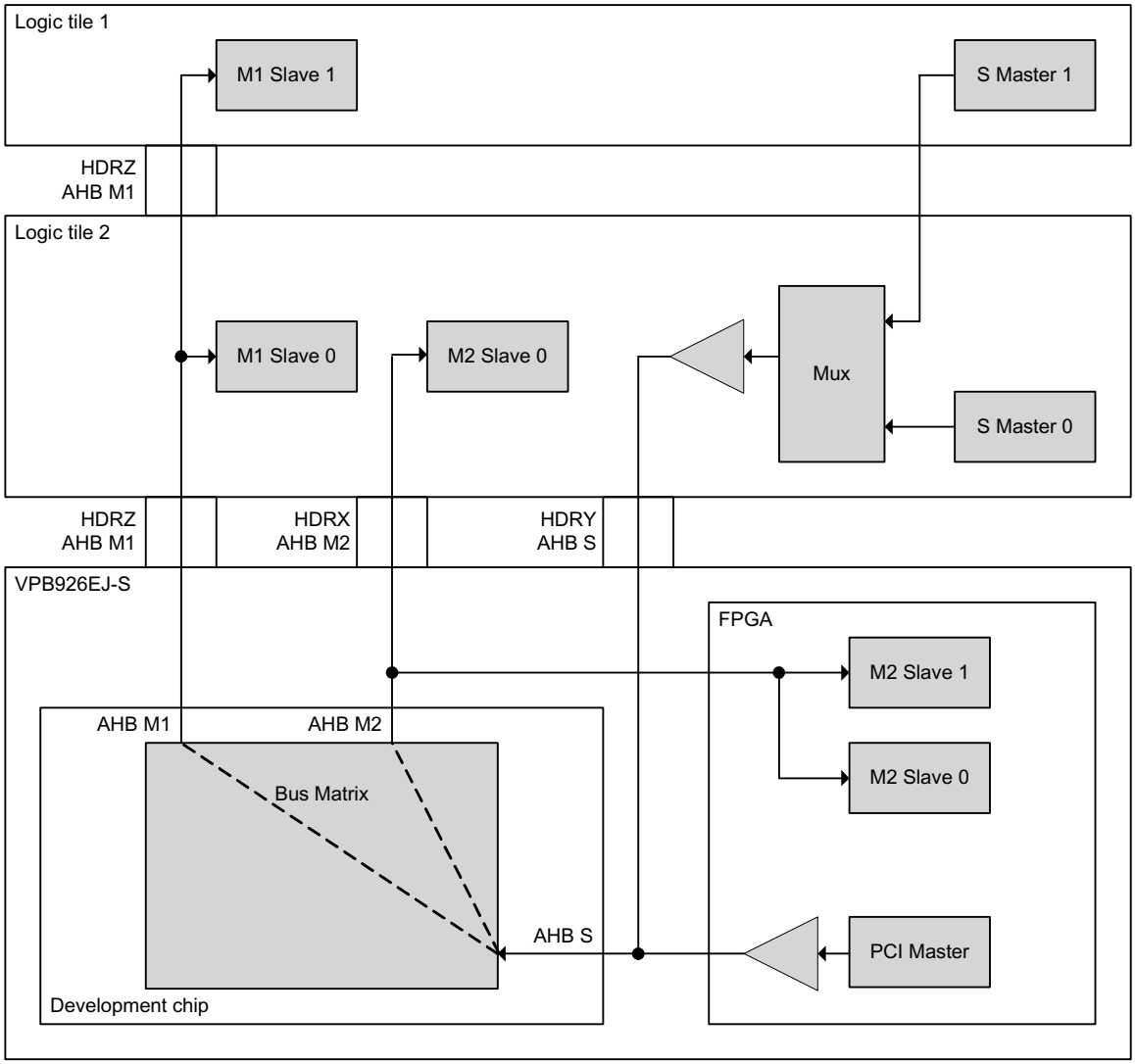


**Figure 14: Standard configuration of Versatile/PB926EJ-S plus logic tiles – flow from masters to slaves**

The figure shows how masters connected to the AHB S bus can access slaves connected to the AHB M1 and AHB M2 buses through the bus matrix.

The figure also shows which signals of the header connectors are used to route the M1, M2 and S buses:

- AHB M1 is routed via HDRZ: This bus is routed straight up the stack, so AHB M1 can be accessed directly by all the logic tiles in the stack.

  Different M1 slaves can be implemented in different logic tiles. They must access the bus with tri-state buffers the same way as it is done in Integrator. The tri-state buffers must only be enabled when the slaves in the logic tile are accessed.

- AHB M2 is routed via HDRX: This bus is only connected to the logic tile at the bottom of the stack. This is normally not a problem since the M2 bus should only be used to access boot memory.

- AHB S is routed via HDRY: This bus is only connected to the logic tile at the bottom of the stack.

  The figure shows how to connect masters implemented in another logic tile to the AHB S bus. Since the S bus is only connected to the bottom tile, that tile needs to route the S bus up the stack using a different set of signals.

  The outputs from the different masters are multiplexed in the bottom tile.

Note that AHB M2 and AHB S are only fully connected to the bottom logic tile if the X and Y bus switches are configured in fold-over mode.

With this configuration the control signals are driven the following way:

HGRANTM1 = 1, HGRANTM2 = 1

READYM1 and HRESPM1 driven by the slaves connected to AHB M1

HREADYM2 and HRESPM2 driven by the default slave connected to AHB M2

HSELS = 1

An arbiter must be provided to arbitrate between the masters in the logic tile and the PCI master on the Versatile/PB926EJ-S FPGA. This arbiter drives LTHGRANT and HMASTLOCKS on the S bus. LTHBUSREQ and LTHLOCK are the bus request and lock signals from the PCI master, and are routed on HDRY.

Sometimes the user may not need to synthesize AHB masters in the logic tile. In this case the masters and arbiter can be removed. The logic tile pins should be driven as follows:

- The signals on the AHB S bus which are outputs of the logic tile should be driven with high impedance.
- LTHGRANT must be tied high, so the PCI master on the baseboard is always granted the bus
- HMASTLOCK must be tied low
- All other control signals (HGRANTM1, HGRANTM2 and HSELS) are tied in the standard way

## 4.4 Versatile/PB926EJ-S buses: booting from AHB expansion bus

The Versatile/PB926EJ-S baseboard has an option to boot from the AHB expansion bus. If this option is selected, the ARM926EJ-S begins to run code from the AHB M2 bus after reset.

If this option is selected, real non-volatile memory should be connected to bus AHB M2. This type of memory is not available on the logic tile itself, so in order to connect the FPGA to a Flash device you may need to design your own Flash board with logic tile connectors or solder a Flash device to an interface tile.

An easier option is to use the logic tile SSRAM to emulate non-volatile memory. In this case the logic tile SSRAM must be connected to bus M2 and mapped to both addresses 0x0 - 0x03FFFFFF and 0x14000000 - 0x1FFFFFFF.

The addresses at 0x14000000 are used as an alias of SSRAM. This is necessary to load the boot code into SSRAM, since the logic tile SSRAM can only be accessed at address 0 when the baseboard boots from AHB expansion bus and before the memory is remapped.

The procedure to boot from AHB expansion bus is the following:

- Boot the board from NOR Flash
- Use a debugger or the boot monitor to load the boot image at address 0x14000000
- Configure the baseboard's DIP switches to boot from AHB expansion bus
- Put the processor in run state and push the DEVCHIP RECONFIG button on the Versatile/PB926EJ-S baseboard
- This causes a reconfiguration of the ARM926EJ-S development chip, so the processor boots from the type of memory selected by the DIP switches

## 4.5 Versatile/PB926EJ-S buses: connecting masters to AHB M1 and M2 buses

In some cases customers may not want to follow the standard configuration, but prefer to connect the logic tile masters to AHB M1 and M2.

Two possible ways to do this are shown in Figure 15:

- AHB M1 uses a multi-layer AHB bus implementation, so the masters in the bus can access different slaves simultaneously
- AHB M2 works as a standard AHB bus, in which all the masters are connected to the same bus

Note that both configurations work and can be used on either M1 or M2.

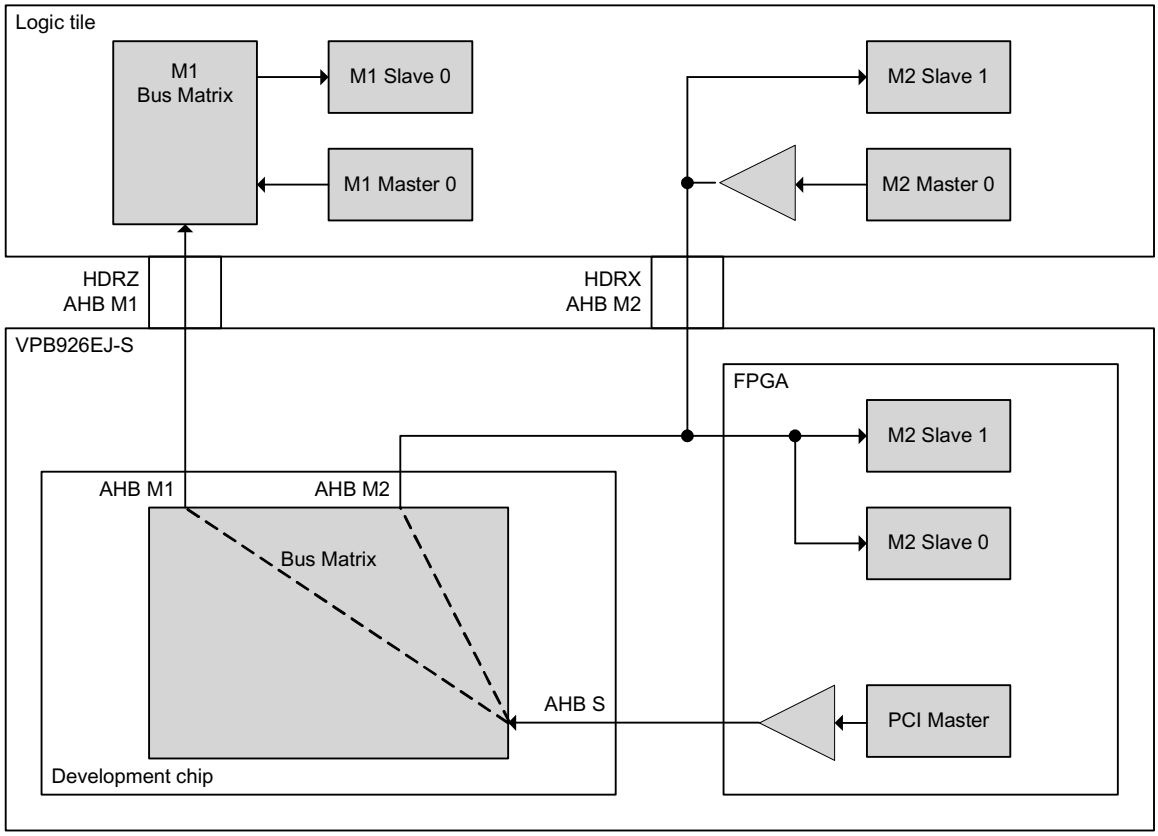


**Figure 15: Connecting logic tile masters to AHB M1 and M2 buses – flow from masters to slaves**

With either of these configurations, the logic tile masters can only access slaves connected to the same bus as the masters. The logic tile masters cannot access the resources inside the development chip or on the other AHB buses.

These configurations represent how Versatile/PB926EJ-S can still be used to emulate an ASIC with a single ARM processor and a standard AHB bus.

In Figure 15, the AHB M1 bus matrix is in charge of performing the arbitration of the masters connected to M1. In this case HGRANTM1, the grant signal for the AHB M1 bridge, can be tied high. The bus matrix drives HREADYM1 low when the AHB M1 bridge has to wait for the internal master to finish a transfer.

The logic tile M2 master accesses the AHB M2 bus with tri-state drivers. In order to avoid bus contention an arbiter must be implemented on the logic tile. This arbiter must drive the HGRANTM2 signal, which enables the outputs of the AHB M2 bridge inside the development chip.

With the configuration shown in the picture, the M2 master in the logic tile can access the slaves inside the baseboard FPGA.

## 4.6 Versatile/PB926EJ-S buses: connecting slaves to the AHB S bus

The ARM926EJ-S development chip provides the HSELS signal so that external slaves can be connected to the AHB S bus. HSELS is routed to the logic tile headers, so that slaves on the S bus can be implemented in logic tiles.

Slaves connected to the AHB S bus can only be accessed by external masters, that is the PCI master or masters synthesized in logic tiles. They cannot be accessed by the masters inside the development chip.

- HSELS must be 1 when an AHB S master is accessing the development chip
- HSELS must be 0 when an AHB S master is accessing the AHB S slave

HSELS must be driven by an address decoder in the logic tile. This lets you map the AHB S slaves “on top” of the standard Versatile/PB926EJ-S memory map.

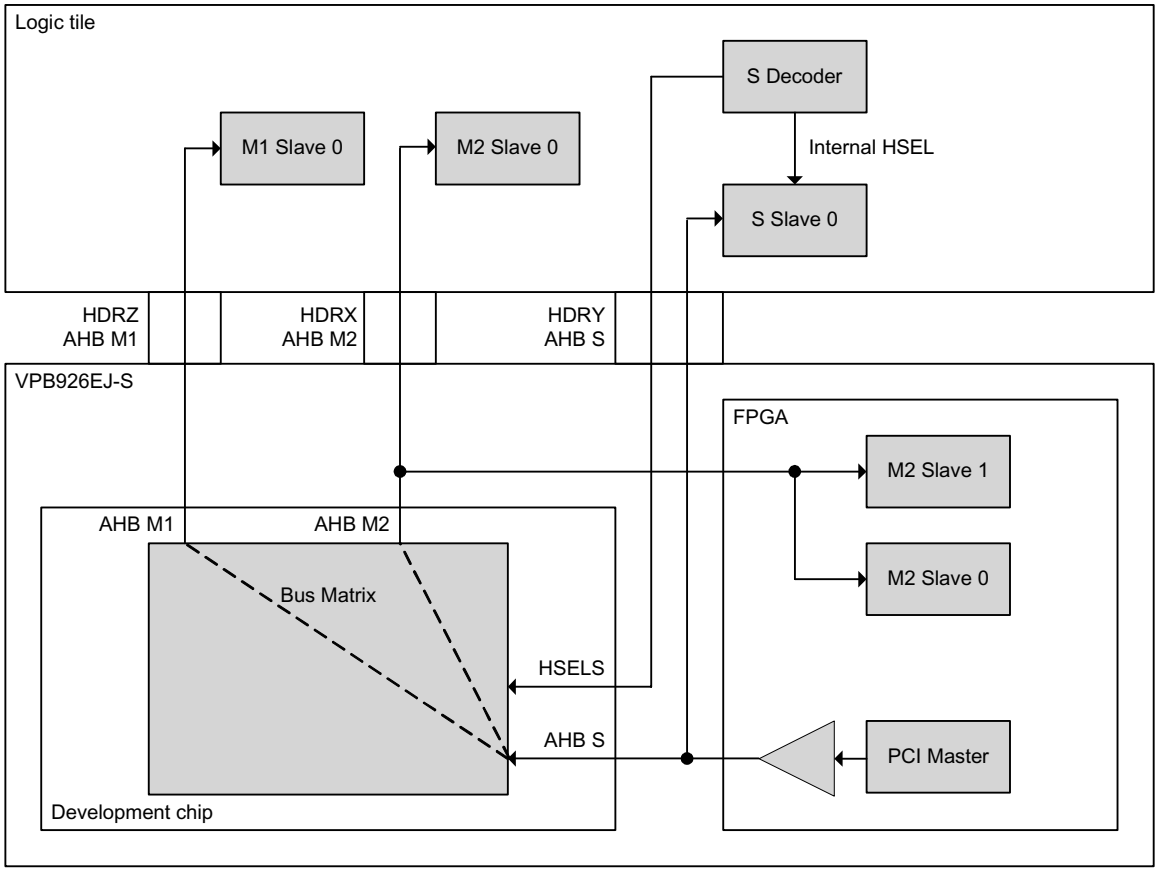


**Figure 16: Connecting logic tile slaves to the S bus – flow from masters to slaves**

Normally HSELS should be tied high and no slaves should be connected to the AHB S bus. Otherwise different masters have different memory maps, which can result in complex problems.

HSELS should only be used in very specific cases, and it requires a very good understanding of AHB and ARM technology.

## 4.7 Versatile/PB926EJ-S clocks

The Versatile/PB926EJ-S clock architecture is complex and highly configurable. The baseboard contains clock multiplexers, so that different clock signals can be used. The clock multiplexers can be configured so that the clocks are driven by either the baseboard or the logic tiles.

The HCLKCTRL bits of the SYS_CFGDATA1 register configure the clock multiplexers. This configuration must match the design in the logic tile, so that the design in the logic tile is clocked with the same signals as the AHB bridges in the development chip. Detailed information about the clock multiplexers can be found in the Versatile/PB926EJ-S user guide.

The baseboard can be configured to generate the clocks for the M1, M2 and S buses. In this case the baseboard FPGA generates the clocks, which are connected to the logic tile on the “F2L” signals (FPGA to Logic tile). These signals are:

- Baseboard’s HCLKM1_F2L connects to logic tile’s CLK_POS_UP_IN
- Baseboard’s HCLKM2_F2L connects to logic tile’s ZL217
- Baseboard’s HCLKS_F2L connects to logic tile’s CLK_NEG_UP_IN

The baseboard’s GLOBALCLK connects to the logic tile’s CLK_GLOBAL. This clock can be driven by either the logic tile or the baseboard. If the global clock is driven from the logic tile, then the baseboard’s driver to GLOBALCLK must be disabled. This can be done by driving the signal F2LSPARE4 high from the logic tile.

Separate clocks for the AHB M1, M2 and S buses can also be driven from the logic tile FPGA’s I/O pins. These signals are named HCLKM1_L2F, HCLKM2_L2F and HCLKS_L2F on the baseboard. “L2F” stands for “Logic tile to FPGA”.

CLK_POS_UP_IN and CLK_NEG_UP_IN are connected to global clock inputs of the bottom logic tile’s FPGA. ZL217 is connected to a normal I/O pin of the FPGA, which makes the AHB M2 clock routing less efficient.

CLK_POS_UP_IN, ZL217 and CLK_NEG_UP_IN, are only connected to the bottom logic tile. A design split into several logic tiles should not be clocked with these signals. Otherwise, the clocks must be routed through the bottom logic tile up the stack, which can cause significant clock skew.

When there is more than one logic tile in the system, the whole design must be clocked with CLK_GLOBAL, since this signal is connected to all the tiles in the system.

All the clocks inside the development chip are derived from GLOBALCLK. Therefore in this case both the maximum speed of the development chip and the design on the logic tiles impose a restriction on the frequency of GLOBALCLK.

When there is only one logic tile on top of the baseboard, it is recommended to use CLK_POS_UP_IN to clock the peripherals connected to AHB M1, ZL217 to clock those connected to AHB M2 and CLK_NEG_UP_IN to clock those connected to AHB S.

This provides greater flexibility, since different buses can be clocked with different clocks:

- AHB M1 can be clocked with signals from the baseboard (OSC0 or OSC1) or from the logic tile (HCLKM1_L2F or CLK_GLOBAL)
- AHB M2 can be clocked with signals from the baseboard (OSC0 or OSC2) or from the logic tile (HCLKM2_L2F or CLK_GLOBAL)
- AHB S can be clocked with signals from the baseboard (OSC0 or OSC3) or from the logic tile (HCLKS_L2F or CLK_GLOBAL)

If the frequency of the AHB M1, M2 and S buses is different from the frequency of GLOBALCLK, then the AHB bridges must be configured in asynchronous mode.

With the default value of HCLKCTRL (0xE0), the baseboard drives all its system bus clocks (GLOBALCLK, HCLKM1_F2L, HCLKM2_F2L and HCLKS_F2L) with the signal OSC0.

Selection of the default bridge mode after power up is set by S1[3]. With S1[3] OFF Synchronous mode is selected and CLK_GLOBAL is provided to drive the logic tile design. With S1[3] ON Asynchronous mode is selected and CLK_POS_UP_IN, ZL217 and CLK_NEG_UP_IN are provided to drive the logic tile design.

When implementing a logic tile design it is important to use the appropriate clock(s) for the mode of operation used. Figure 17 and 18 gives a simplified view of how the clocks are routed on the VPB926EJ-S baseboard to the development chip and the logic tile FPGA. When using Synchronous mode OSCCLK0 drives the development chip and logic tile using GLOBALCLK. Buffers U24 and U3-1 introduce the same delay to the clock path and so GLOBALCLK appears at the development chip and logic tile FPGA simultaneously.

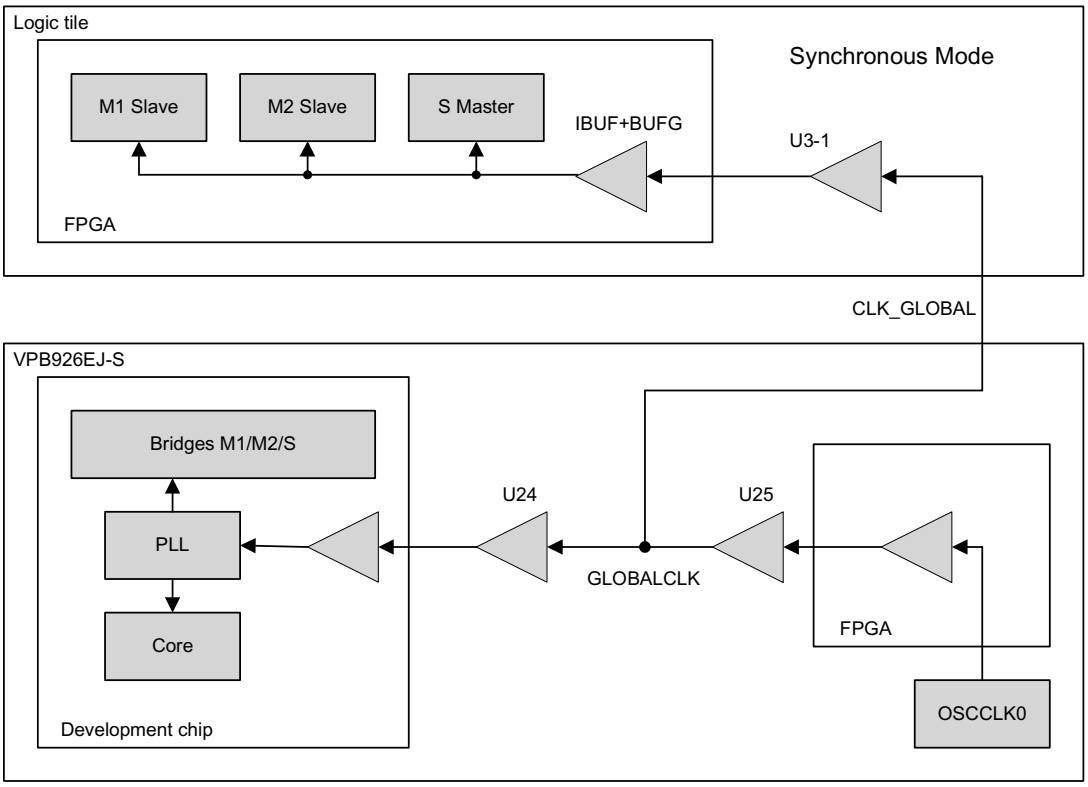


**Figure 17: Synchronous mode**

When using Asynchronous mode OSCCLK0 still drives the Development chip core but the M1, M2 and S bridges are driven asynchronously by CLK_POS_UP_IN, ZL217 and CLK_NEG_UP_IN respectively. There are no buffers on the VPB926EJ-S baseboard introducing delays to the M1, M2 and S clock paths. Similarly there are no buffers on the logic tile for these clocks and so the development chip bridges and logic tile FPGA have matched clock paths.

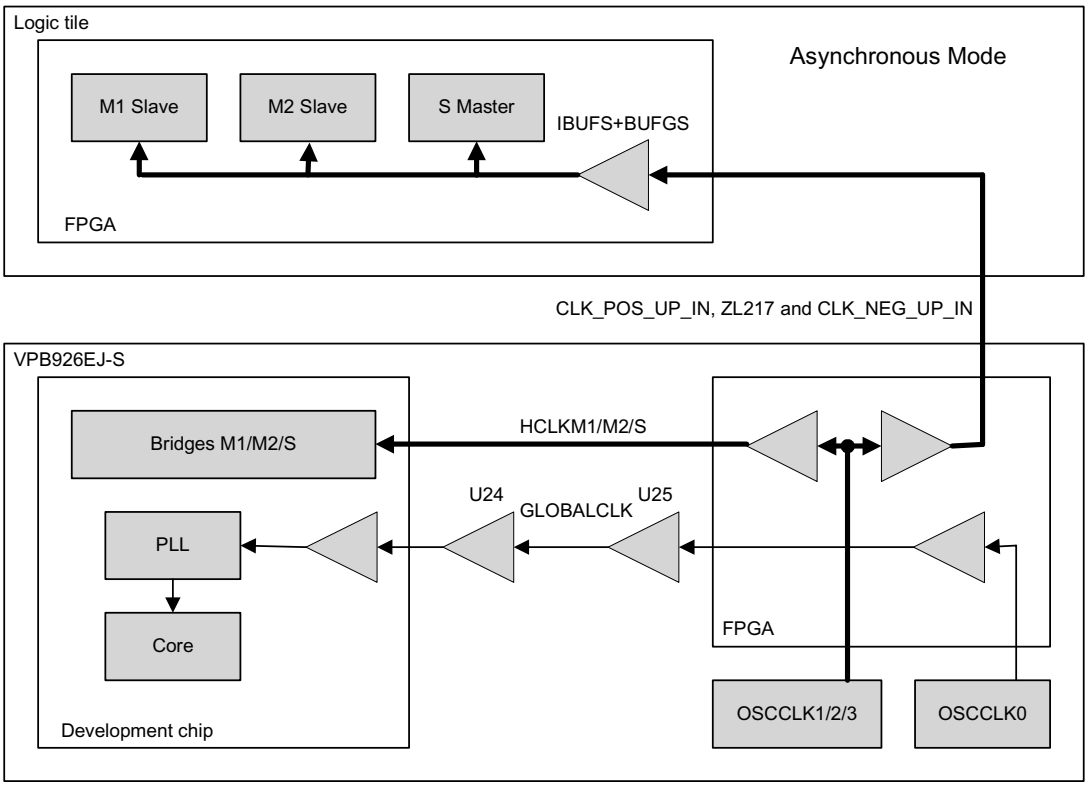


**Figure 18: Asynchronous clocking**

The Example designs provided demonstrate how to implement both Synchronous and Asynchronous bridge modes. The clock buffers must be set appropriately in the Top Level HDL design (AHBTopLevel.v or AHBTopLevel.vhd). Also the constraints file must be set to the appropriate clock selection (an119_veratile_master.ucf).

# 5 Description of the Example HDL

## 5.1 General description

Example Verilog and VHDL code, synthesis scripts and pre-synthesized bit-files are provided with this application note. The example code shows how to implement AHB masters and slaves in a logic tile working in three possible configurations:

- Configuration 1: Integrator/AP or CP baseboard with Integrator/IM-LT1 + logic tile in the logic module stack
- Configuration 2: Integrator/CM + Integrator/IM-LT1 + logic tile stacked together
- Configuration 3: Versatile/PB926EJ-S + logic tile

Note: the example design is loaded into logic tile IMAGE 1 (S2[2] ON, S2[1] OFF).

The example design is based on the Example2 that used to be included on the logic module and logic tile installation disk. This example has been modified to support the configurations above.

The example design implements a simple AHB system. The original example2 only implemented AHB slaves, but AHB masters have also been added, resulting in the following 5 example designs:

- Configuration 1 with AHB slaves
- Configuration 1 with AHB masters and slaves
- Configuration 2 with AHB slaves
- Configuration 2 with AHB masters and slaves
- Configuration 3 with AHB masters and slaves

The example design contains the following AHB slaves:

- Two ZBT SSRAM controllers: give access to the logic tile SSRAM devices from the AHB bus
- An AHB to APB bridge: gives access to the following APB peripherals
    - APB configuration registers
    - APB interrupt controller
- A default slave

In Integrator systems all the AHB slaves are connected to the AHB system bus. The interrupt from the logic tile is routed to the interrupt controller on the baseboard. In the configuration without a motherboard, the interrupt drives directly the nIRQ pin of the core module.

In a Versatile system the AHB slaves are connected to the AHB M1 bus. On the AHB M2 bus there is only a default slave that covers the range of addresses mapped to the logic tile. The logic tile interrupt is routed to the VIC inside the development chip as source number 30.

Two AHB masters have been added to some configurations. The two masters are identical and have very limited functionality, since they are only intended to show how to integrate existing AHB masters in a logic tile.

The example masters generate a transfer when the logic tile push button is pressed. The address of the transfer is programmed in logic tile registers. Which master generates the transfer and the type of transfer is selected by the logic tile switches.

The designs with masters also include an arbiter when necessary, that is in configurations 2 and 3.

The logic tile also includes some miscellaneous logic to control the logic tile hardware:

- A controller for the fold-over and thru switches
- Three ICS307 clock controllers
- Internal routing of the JTAG debug signals: D_TDI to D_TDO and D_TCK to D_RTCK

The example design works at up to **30MHz** system bus clock on an Integrator system and **36MHz** in both Synchronous and Asynchronous bridge modes on a Versatile system.

On a Versatile system the example design can be built for Synchronous or Asynchronous bridge mode. When building the example it is necessary to set the clock constraints in the .ucf constraints file and clock inputs in the AHBTopLevel HDL appropriately. Please refer to the comments in these files for information on setting the appropriate option.

## 5.2 Description of the HDL

Below there is a general description of the source code files provided. All the files are supplied in both Verilog (.v) and VHDL (.vhd). Each file normally contains only one module of the same name, with the exception of APBClocks and APBClockArbiter.

AHBTopLevel: This file is the top level HDL, which instantiates and interconnects the main blocks in the system. It also includes the tri-state interface logic that gives access to the external bus or buses.

AHBDecoder: Generates the HSEL selection signals for the AHB slaves. These signals are generated with combinatorial logic from HADDR. In the example code for configuration 1, the ID signals are used to map the logic tile at different addresses depending on its position in the stack.

In the example code for Versatile there are two decoder blocks, AHBDecoderM1 and AHBDecoderM2, one for each of the external master buses.

AHBMuxS2M: This module multiplexes the HREADY, HRESP, and HRDATA outputs from the AHB slaves. The outputs from the multiplexer are used to drive the tri-state buffers that access the AHB system bus.

AHBZBSRAM: ZBT SSRAM controller that allows word, half-word, and byte access to the logic tile SSRAM from the AHB system bus. Two AHBZBTRAM modules are instantiated in the design, one for each SSRAM device on the board.

AHBDefaultSlave: The HREADY and HRESP signals are driven by the default slave if the logic tile is accessed at an address not covered by any of the peripherals in the design.

AHBAPBSys: The APB components are instantiated in this block. These include the AHB-APB bridge and the APB peripherals.

AHB2APB: This is the bridge that connects the APB peripherals to the internal AHB bus. It also produces the peripheral select signals for each of the APB peripherals.

APBRegs: The APB register peripheral provides memory-mapped registers.

APBIntcon: The APB interrupt controller contains the interrupt controller registers, which can accept up to five external interrupts and four software interrupts. The example code only uses one external interrupt, generated by the push button.

APBClocks: Provides a parallel to serial interface that transfers the APB clock register contents to the ICS307 clock generators.

APBClockArbiter: Implements a simple arbitration scheme to ensure that the three clock controllers are serviced independently.

The example versions that implement masters also include the following files:

AHBExampleMaster: Implements a simple AHB master that generates a transfer when the logic tile push button is pressed.

AHBMuxM2S: Multiplexes the outputs from the masters depending on which one is granted the bus. The outputs from the multiplexer drive the tri-state buffers that give access to the system bus.

AHBArbiter: Arbitrates between the different masters in the system. It implements a round robin scheme with equal priority between the masters. This block is only included in configurations without an Integrator/AP motherboard, since Integrator/AP already has a system arbiter.

Figure 19 shows the structure of the example HDL. The boxes with dotted line are the modules that only appear in some of the configurations.

The HDL provided is commented, especially the top level file that shows the connection to the system bus. Two configurations have special comments that show the specific differences from the original Example2 for Integrator:

- Configuration 1 with AHB masters and slaves: the changes are commented with the label **Example2_Master**.

  This shows how to add masters to an existing design for Integrator

- Configuration 2 with AHB slaves only: the changes are commented with the label **Example2_coremodule**.

  This shows how to modify an existing logic tile design for a standard Integrator system so that it works without a motherboard.

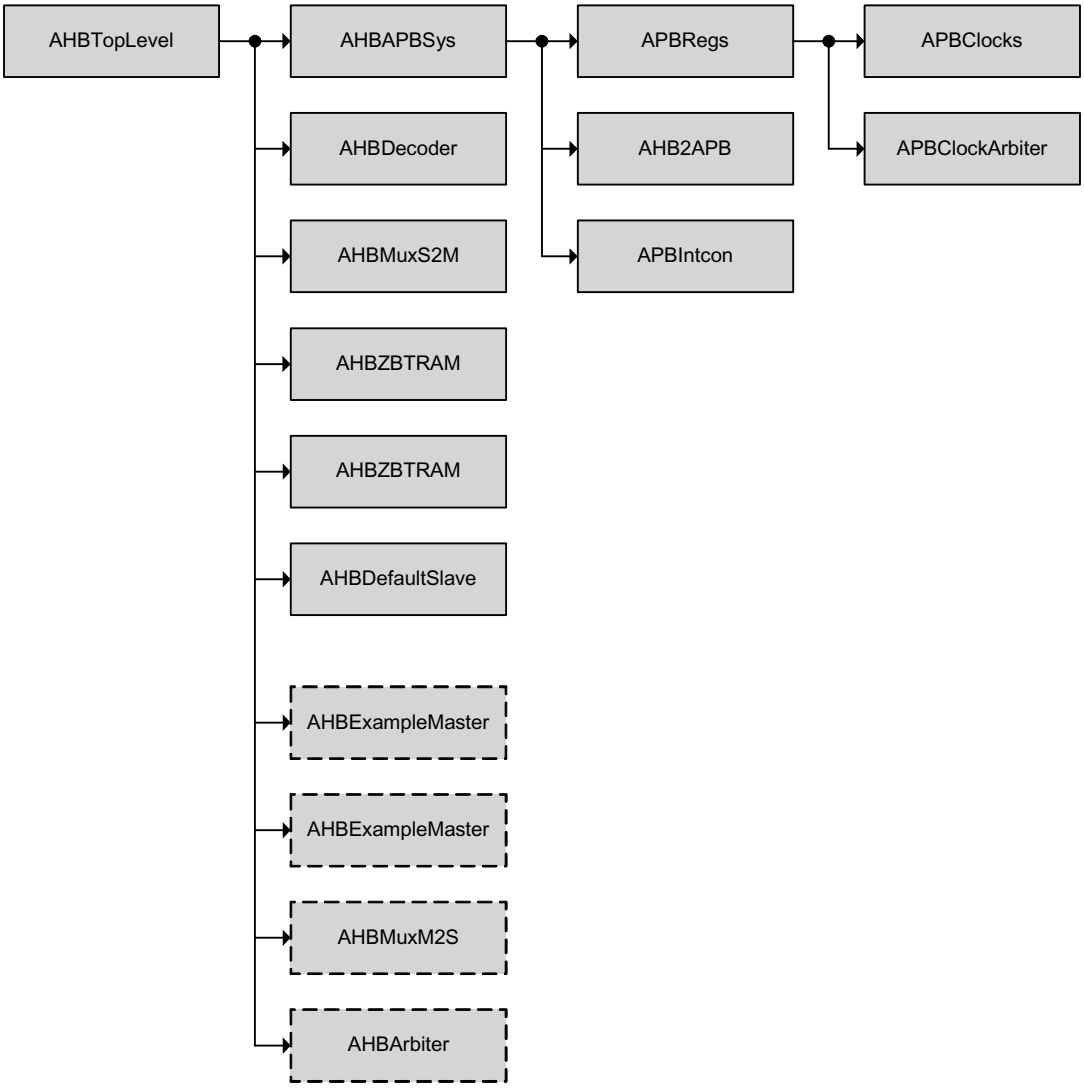


**Figure 19: Structure of the example HDL**

## 5.3 Description of the masters

The example masters generate a single word access in the system bus when the logic tile push button is pressed.

The logic tile switch S1 selects which master generates a transfer and the type of transfer:

- If S1-1 is OFF, master 0 generates the transfer
- If S1-1 is ON, master 1 generates the transfer
- If S1-2 is OFF, the enabled master generates a write transfer
- If S1-2 is ON, the enabled master generates a read transfer

The addresses for the system bus accesses are programmed by writing to two new logic tile registers: HADDR_TRANSFER0 and HADDR_TRANSFER1.

- HADDR_TRANSFER0 contains the transfer address for master 0 and is located at offset address 0x24
- HADDR_TRANSFER1 contains the transfer address for master 1 and is located at offset address 0x28

Masters in configurations 1 and 2

In configurations 1 and 2 (Integrator), the logic tile is stacked on top of an IM-LT1, so the IM-LT1 switches can be used to configure the transfer.

In a read transfer the master reads from the system bus and writes the value read to the IM-LT1 LEDs. The standard logic tile example2 HDL has been modified so that the IM-LT1 LEDs cannot be written by accessing the logic tile registers.

In a write transfer the master reads the value from the IM-LT1 switches and writes it to the system bus.

For example, if you want master 0 to write the data 0x05 to address 0x28000010, you must follow these steps:

- Set S1-1 to OFF, so master 0 generates the transfer
- Set S1-2 to OFF, so the transfer is a write transfer
- Set the IM-LT1 switches to 0x05 (OFF-OFF-OFF-OFF-OFF-ON-OFF-ON)
- Write 0x28000010 to register HADDR_TRANSFER0 at address 0xC0000024
- Push the logic tile push button

Due to the Integrator system architecture, the logic tile master can only access the following memory devices:

- Integrator/AP SSRAM at address 0x28000000
- Integrator/CM SDRAM alias at address 0x8000000 – 0xB0000000 depending on the core module's position in the stack
- Logic tile SSRAM at address 0xC2000000 – 0xF2000000 depending on the value of ID (logic tile's position in the stack)

By default after reset, HADDR_TRANSFER0 and HADDR_TRANSFER1 are programmed to access Integrator/AP SSRAM.

- HADDR_TRANSFER0 = 0x28000000
- HADDR_TRANSFER1 = 0x28000004

Masters in configuration 3

In configuration 3 (Versatile), the logic tile is stacked directly on top of the Versatile/PB926EJ-S baseboard, so the IM-LT1 switches and LEDs are not available.

The example2 for Integrator contains two registers to access the IM-LT1 switches and LEDs. These registers are reused to configure the transfers generated by the masters:

- IM_LT1_LEDS at address offset 0x014 becomes HDATA_WRITE and is a read/write register. The register contains the data that will be written in the next write transfer.
- IM_LT1_SW at address offset 0x20 becomes HDATA_READ and is a read-only register. This register contains the data read in the last read access.

For example, if you want master 0 to write the data 0x05 to address 0x04000010, you must follow these steps:

- Set S1-1 to OFF, so master 0 generates the transfer
- Set S1-2 to OFF, so the transfer is a write transfer
- Write 0x05 to register HDATA_WRITE at address 0xC0000014
- Write 0x04000010 to register HADDR_TRANSFER0 at address 0xC0000024
- Push the logic tile push button

Due to the Versatile system architecture, the logic tile master can access the baseboard devices through the development chip's bus matrix. By default after reset, HADDR_TRANSFER0 and HADDR_TRANSFER1 are programmed to access the baseboard SDRAM.

- HADDR_TRANSFER0 = 0x04000000
- HADDR_TRANSFER1 = 0x04000004

## 5.4 Memory map

All the AHB and APB peripherals instantiated in the design can always be found at the same offset address. The example design responds to 256MB of addresses, for example from address 0xC0000000 to 0xCFFFFFFF. The base address for the logic tile peripherals depends on the configuration chosen and the position of the logic tile in the stack.

The offset for the different peripherals is shown in Table 5:

| **Device** | Registers | Int. Controller | SSRAM 0 | SSRAM 1 | Default Slave |
|---|---|---|---|---|---|
| **Offset** | 0x0 | 0x01000000 | 0x02000000 | 0x02200000 | 0x024000000 |

**Table 5: Offset address for peripherals in example design**

In configuration 1 (Integrator system with a motherboard) the base address depends on the stack where the logic tile is located and the position of the logic tile in the stack. This is shown in Table 1.

The HDRID signals are rotated through the stack by the design in the logic tile, so that a stack of logic tiles has the same memory map as a stack of logic modules.

In configuration 2 (Integrator system without a motherboard) the base address of the peripherals is fixed at 0xC0000000. The logic tile SSRAM is also aliased at address 0, so that the processor in the core module can boot from it.

The default slave responds to all the other addresses in the 4GB range but those mapped to the core module SDRAM alias.

In configuration 3 (Versatile system) all the slave peripherals are connected to AHB M1. The base address of the peripherals is fixed at 0xC0000000. The default slave responds to all the other addresses in the 4GB range.

A default slave is connected to AHB M2, which responds to the addresses assigned to the logic tile.

## 5.5 System registers

Table 6 shows the location of the system registers in the example design. The addresses shown are offsets from the logic tile base address.

| Offset address | Name | Type | Size | Function |
|---|---|---|---|---|
| 0x00000000 | LT_OSC0 | Read/write | 19 | Oscillator 0 divisor register |
| 0x00000004 | LT_OSC1 | Read/write | 19 | Oscillator 1 divisor register |
| 0x00000008 | LT_OSC2 | Read/write | 19 | Oscillator 2 divisor register |
| 0x0000000C | LT_LOCK | Read/write | 17 | Oscillator lock register |
| 0x00000010 | LT_LEDS | Read/write | 4 | User LEDs control register (LT) |
| 0x00000014 | IM_LT1_LEDS / HDATA_ WRITE | Read/write | 8 | User LEDs control register (IM-LT1) / Data to write next transfer |
| 0x00000018 | LT_INT | Read/write | 1 | Push button interrupt register |
| 0x0000001C | LT_SW | Read | 4 | Switches register (logic tile S1) |
| 0x00000020 | IM_LT1_SW / HDATA_READ | Read | 8 | Switches register (IM-LT1 S3) / Data read last transfer |
| 0x00000024 | HADDR_TRANSFER0 | Read/write | 32 | Address next transfer master 0 |
| 0x00000028 | HADDR_TRANSFER1 | Read/write | 32 | Address next transfer master 1 |

**Table 6: Location of system registers**

Oscillator divisor registers

The oscillator registers LT_OSC0, LT_OSC1 and LT-OSC2 (at offset 0x00, 0x04 and 0x08) control the frequency of the clocks generated by the three clock generators on the logic tile.

Before writing to the oscillator registers, you must unlock them by writing the value 0x0000A05F to the LT_LOCK register. After writing the oscillator register, relock it by writing any other value to the LT_LOCK register.

| Bits | Name | Access | Function | Default |
|---|---|---|---|---|
| [18:16] | OD | Read/write | Output divider:<br>000 = divide by 10<br>001 = divide by 2<br>010 = divide by 8<br>011 = divide by 4<br>100 = divide by 5<br>101 = divide by 7<br>110 = divide by 3<br>111 = divide by 6 | 011 |
| [15:7] | VDW | Read/write | Reference divider word. Defines the binary value of the V[7:0] pins of the clock generator | 000001000 |
| [6:0] | RDW | Read/write | VCO divider word. Defines the binary value of the RV[6:0] pins of the clock generator | 0000110 |

**Table 7: LT_OSCx bit pattern**

The reset value of these registers sets the oscillators to 24MHz. More information about setting up the frequency of the logic tile oscillators is available in the LT-XC2V4000+ logic tile.

Oscillator lock register

The lock register LT_LOCK (at offset 0x0C) controls access to the oscillator registers and allows you to lock them and unlock them. This mechanism prevents the oscillator registers from being overwritten accidentally.

| Bits | Name | Access | Function |
|---|---|---|---|
| [16] | LOCKED | Read | This bit indicates if the oscillator registers are locked or unlocked:<br>0 = unlocked<br>1 = locked |
| [15:0] | LOCKVAL | Read/write | Write the value 0xA05F to this field to enable write accesses to the oscillator registers.<br>Write any other value to lock the oscillator registers. |

**Table 8: LT_LOCK bit pattern**

User LEDs control registers

The LT_LED register (at offset 0x10) controls the 4 user LEDs on the logic tile.

The IM_LT1_LED register (at offset 0x14) controls the 8 user LEDs on the IM-LT1 interface module. In configuration 3 (Versatile/PB926EJ-S example HDL), this register has been replaced – see *Registers used by example masters* below.

Writing a 1 to a bit lights the associated LED.

Push button interrupt register

The push button interrupt register LT_INT (at offset 0x18) contains 1 bit. It is a latched indication that the push button has been pressed. The contents of this register are fed to the interrupt controller registers.

| Bits | Name | Access | Function |
|---|---|---|---|
| [0] | LT_INT | Read | If the push button has been pressed, this bit is set. |
| | | Write | Write 0 to this register to clear the latched push button indication. |
| | | | Writing 1 to this register has the same effect as pressing the push button. |

**Table 9: LT_INT bit pattern**

Switch registers

Use the LT_SW register (at offset 0x1C) to read the setting of the 4-way DIP switch on the logic tile.

Use the IM_LT1_SW register (at offset 0x20) to read the setting of the 8-way DIP switch on the IM-LT1 interface module. In configuration 3 (Versatile/PB926EJ-S example HDL), this register has been replaced – see *Registers used by example masters* below.

A value 1 indicates that the associated switch element is CLOSED (ON).

Registers used by example masters

HADDR_TRANSFER0 contains the transfer address for master 0 and is located at offset address 0x24

HADDR_TRANSFER1 contains the transfer address for master 1 and is located at offset address 0x28

In configuration 3 (based on Versatile), the registers IM_LT1_LEDS and IM_LT1_SW are replaced with:

HDATA_WRITE contains the data that will be written in the next write transfers and is located at offset address 0x14.

HDATA_READ contains the data read in the last read transfers and is located at offset address 0x20.

## 5.6 Interrupt controller

The interrupt controller included in the example design generates an interrupt signal from a number of interrupt sources. The output of the interrupt controller is routed in a different way depending on the configuration used:

- Configuration 1: the logic tile interrupt is routed to the interrupt controller on the motherboard
- Configuration 2: the logic tile interrupt is routed directly to the nIRQ input of the core module
- Configuration 3: the logic tile interrupt is routed to the primary interrupt controller inside the ARM926EJ-S development chip as interrupt source 30

The logic tile interrupt controller is designed so that it can accept up to four sources of interrupts and four software interrupts. In the example design only one source of interrupt is used, which is connected to the logic tile push button.

The other three sources or interrupt are unused in the example design, but users can use them to connect interrupt request signals for the peripherals they implement in the logic tile.

The interrupt controller contains registers to enable, disable and monitor the status of the different interrupt sources.

Table 10 shows the interrupt number assigned to each source of interrupt. Each interrupt source is associated with a bit number in the interrupt controller registers.

| Bit | Name | Function |
|---|---|---|
| [7:5] | - | Spare (not used by the example image) |
| [4] | PBINT | Push button interrupt |
| [3:0] | SOFTINT[3:0] | Software interrupt generated by writing to LT_SOFTINT |

**Table 10: Interrupt register bit assignment**

For example, in order to enable the push button interrupt you must set bit 4 of the interrupt enable register.

Table 11 shows the location of the interrupt controller memory mapped registers.

| Offset address | Name | Type | Size | Function |
|---|---|---|---|---|
| 0x01000000 | LT_ISTAT | Read | 8 | Interrupt status |
| 0x01000004 | LT_IRSTAT | Read | 8 | Interrupt raw status |
| 0x01000008 | LT_IENABLE | Read | 8 | Interrupt enable |
| 0x01000008 | LT_IENSET | Write | 8 | Interrupt enable set |
| 0x0100000C | LT_IENCLR | Write | 8 | Interrupt enable clear |
| 0x01000010 | LT_SOFTINT | Read/write | 4 | software interrupt |

**Table 11: Location of interrupt controller registers**

The way that the interrupt enable, clear, status and raw status registers work is illustrated in Figure 20. This figure shows the control logic for one interrupt source, corresponding to one bit of all these registers.

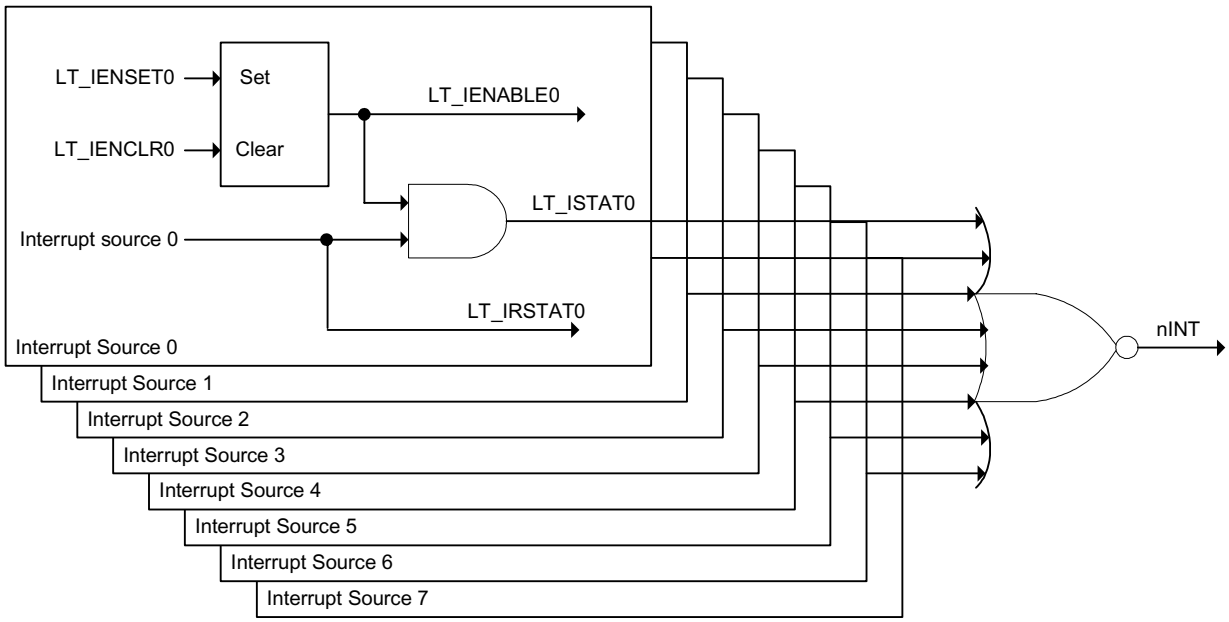


**Figure 20: Interrupt controller internal design**

nINT in the figure is the interrupt request output from the logic tile interrupt controller. This signal is activated when any bit of LT_ISTAT is set.

Interrupt status register

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

Therefore a bit of the status register is 1 when its corresponding interrupt source is active and its corresponding interrupt enable bit has been set.

Interrupt raw status register

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

Interrupt enable, interrupt enable set and interrupt enable clear

Reading from the interrupt enable register LT_IENABLE returns the current state of the interrupt source enable bits.

Writing 1 to a bit of the interrupt enable set register LT_IENSET sets the corresponding bit of LT_IENABLE

Writing 1 to a bit of the interrupt enable clear register LT_IENCLR clears the corresponding bit of LT_IENABLE

Writing 0 to a bit of LT_IENSET or LT_IENCLR leaves the corresponding bit of LT_IENABLE unchanged

LT_IENABLE and LT_IENSET share the same address in the memory map.

Software interrupt register

This register is used to generate interrupts by software.

Writing 1 to any bit position in LT_SOFTINT register sets the corresponding bit in the interrupt controller registers. The LT_SOFTINT register has four bits, corresponding to the four software interrupts.

Writing a 0 to this register clears any software interrupts.

Reading from this register shows the raw status of the software interrupts.

The software interrupts should not be confused with the ARM SWI instruction.

## 5.7 Example software

Example software is provided to verify the example design and the logic tile hardware.

The source files included are logic.c, logic.h and rw_support.s. Logic.c contains the main code, written in C. rw_support.s contains several assembler functions to perform word, half-word and byte accesses to logic tile SSRAM.

A batch file with calls to the compiler, assembler and linker; and a built image are also provided for each of the configurations. The software can be re-built with both ADSv1.2 and RVDSv2.1.

After the FPGA is configured, as indicated by the FPGA_OK LED, you can download and execute the example software on any ARM processor in the system. For example the software can be executed by an Integrator core module or the ARM926EJ-S processor on the Versatile/PB926EJ-S baseboard.

The example code communicates with the user via the debugger's console window. It operates as follows:

1. Reads the baseboard identification register to ensure that the software is executed on the correct system.

2. If the system is an Integrator system with motherboard, checks that the logic tile is present at the bottom of the stack. Depending on the type of motherboard, CP or AP, it accesses the logic tile at a different base address.

3. Sets the logic tile clocks

4. Flashes the LEDs on the logic tile and interface module (if present)

5. Tests the logic tile push button and interrupt controller.

6. Tests the SSRAMs for word, half-word, and byte accesses.

7 Remains in a loop that displays the logic tile and interface module (if present) switch values on the LEDs.