Application Note 152

Using a CT11MPCore with the RealView™ Emulation Baseboard

Document number: ARM DAI 0152E

Issued: June 2008

Application Note 152
Using a CT11MPCore Tile with EB

Release information

The following changes have been made to this Application Note.

Change history

Date	Issue	Change
June 29, 2005	A	First release
January 18, 2006	B	Getting started section added
July 14, 2006	C	Extra detail on clocking strategy, serial data stream. Fixed table and figure numbering. Change to reflect synchronous interface to tile site 1 Correct Interrupts in normal mode
October 19, 2006	D	Reduced MPCore Core Tile information, referenced CT11MPCore UG to avoid duplication. Asynchronous interface to TS1 (CT11MPCore), synchronous interface to PL340 (DMC), timing changed to 25MHz for TS1 (CT11MPCore). Updated to improve interrupt description.
June 22, 2008	E	Synchronous interface to TS1 (CT11MPCore), synchronous interface to PL340 (DMC). Added additional information on CT11MPCore Clarified differences between C6 and C7 builds Corrected frequency calculations in table 3-3, expanded reset section Updated diagrams to reflect latest build numbers

Proprietary notice

Confidentiality status

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

References

Core Tile for ARM11 MPCore User Guide DUI0318B

Table of Contents

1 Introduction

1.1 Purpose of this application note

This application note covers the operation of the Emulation Baseboard (EB) with a CT11MPCore. It examines the contents of the baseboard FPGA, the system interconnect, the clock structure, and specifics of the programmer's model directly relevant to Core Tile operation.

After reading this Application Note the user should be in a position to make changes to the provided baseboard FPGA design, add their own AHB or AXI based peripherals to it, or debug and analyze the operation of the provided images.

1.2 EB and CT11MPCore Tile overview

This application note is designed for a CT11MPCore Tile fitted to Tile Site 1 on the EB as shown in Figure 1-1. This application note only works with CT11MPCore. The FPGA image for the baseboard is provided as part of the EB CD installation.

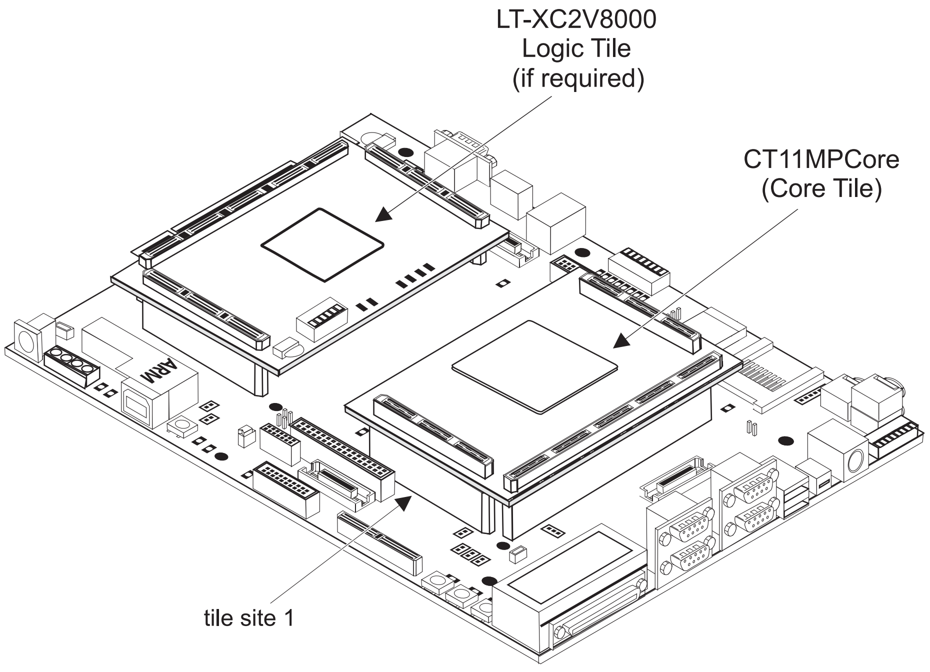


Figure 1-1 Core Tile, Logic Tile and EB system

1.3 Optional Logic Tile

One or more logic tiles can be fitted to Tile Site 2. These Logic Tiles can contain both additional masters and slaves. See application note AN151 for an example and information about how to do this.

2 Getting started

Before you can use this application note, you will need to program the Emulation Baseboard with the required FPGA image to enable the CT11MPCore to function correctly. Follow these steps to program the FPGA image.

1. Plug the CT11MPCore tile onto TILE SITE 1 of the Emulation Baseboard.
2. Slide the CONFIG switch (S1) to the ON position.
3. Connect RVI or Multi-ICE to the Emulation Baseboard JTAG ICE connector (J18), or a USB cable to the USB Debug Port (J16).
4. Check the external supply voltage is +12V (positive on center pin, +/-10%, 35W), and connect it to the power connector (J28).
5. Power-up the boards. The '3V3 OK' LED and '5V OK' on the Emulation Baseboard 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 either PB926EJS or EB \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_usb.exe for your USB 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 CT11MPCore. 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 off the boards.
12. Set the configuration switches to load FPGA image 0. (S10 on the Emulation Baseboard set to all OFF).
13. Slide the CONFIG switch to the OFF position, and power on the boards. Ensure GLOBAL_DONE (D35) and the 'POWER' (D1) LEDs are lit. The Character LCD should show the Firmware and Hardware versions indicating that the Boot monitor firmware is running.
14. The system will now be fully configured and ready for use.

3 Architecture

This application note implements an AXI (AMBA 3.0) based system on the EB FPGA. The EB image exposes one master port and three slave ports (all 64 bit muxed AXI). There are two slave ports routed to tile site1 and one slave and one master port routed to tile site 2.

Note that the direction of the arrows indicates the direction of control: it points from the Master to the Slave. An AXI bus contains signals going in both directions.

The PCI core RTL is not included in any build as it is third party IP, it is an optional component. The DMA controller is not included in the same build as PCI due to the capacity of the baseboard FPGA.

3.1 System overview

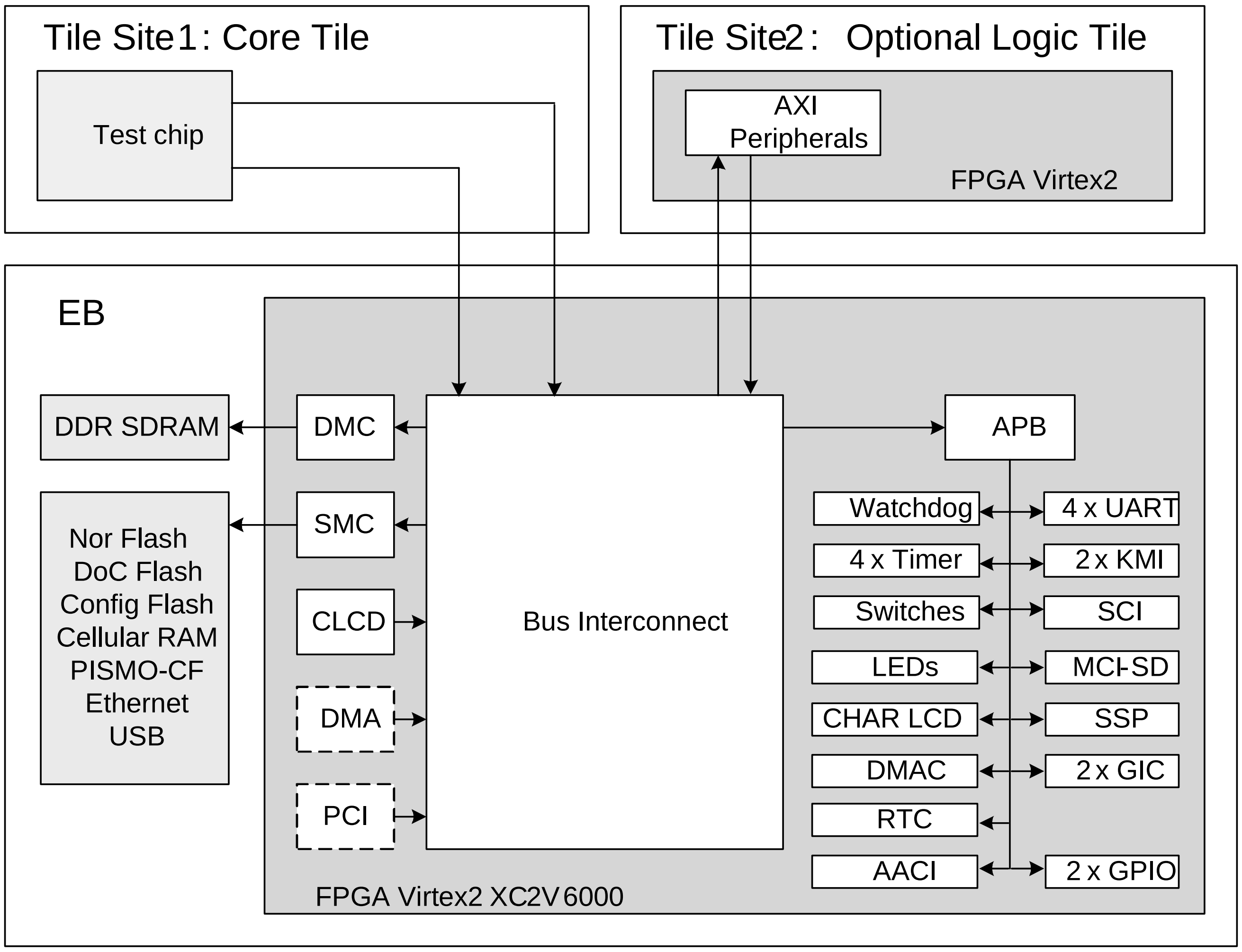


Figure 3-1 System interconnect

3.2 System architecture

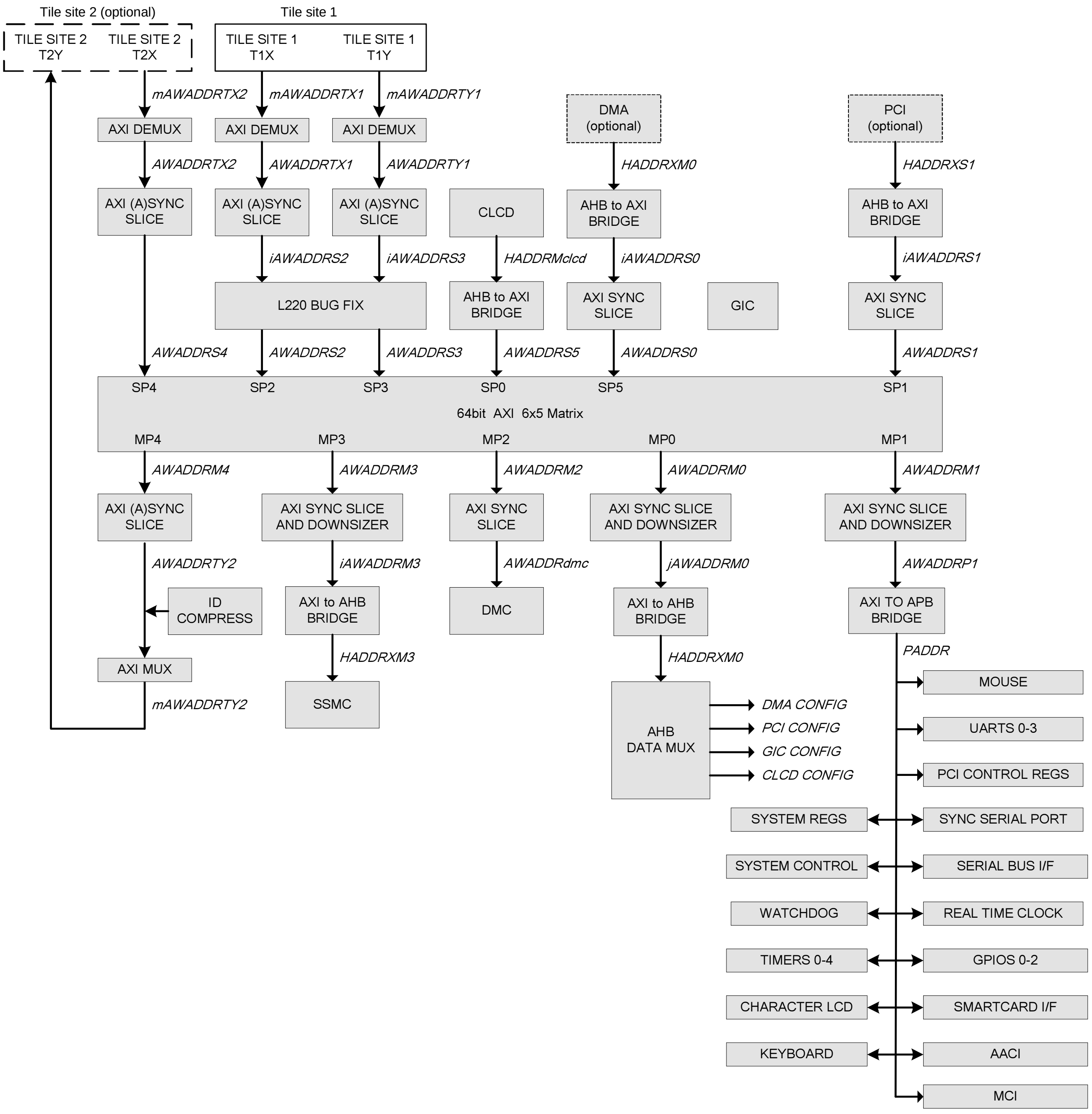


Figure 3-2 AN152 example design block diagram

3.2.1 EB Module functionality

The function of each of these blocks is as follows:

AXI SYNC SLICE

This is a synchronous register slice that is added to the incoming master ports S0, S1 and S5 on the bus matrix to ensure that data is available for a complete cycle prior to AXI matrix address decoding. It also helps to increase the operating frequency.

AXI ASYNC SLICE

This is an asynchronous bridge that is added to the incoming master port from the tile sites to ensure that data is available for a complete cycle prior to AXI matrix address decoding and allows that site to operate in a different clock domain. It increases the operating frequency and introduces one clock cycle of latency.

AXI DEMUX

This block demultiplexes the multiplexes signals from the master ports on tile site 1 and 2. The muxing scheme is described in a section 4.3.

ID COMPRESS

This block merges the ID fields on the AXI bus to reduce the width of each ID channel. The ID is compressed using a simple algorithm to save pins.

AXI MUX

This block multiplexes the signals from the master port Mp4. The muxing scheme is described in a section 4.3.

64-bit AXI 6x5 Bus Matrix

This provides the bulk of the interconnect structure. It allows any of the 6 slave ports to connect to any of the 5 master ports without blocking the other masters. It also contains the decoder mapping to determine the address map, and a scheme to determine priority of competing masters to a single slave.

SSMC

This is a Synchronous Static Memory Controller. ARM PrimeCell PL093 is used in this design. For more information please refer to the PrimeCell documentation.

AHBtoAXI BRIDGE

This is a bridge component to change from an AHB bus to an AXI bus. Where nn is the input width and mm is the output width in decimal.

DMC

This is a Dynamic Memory Controller. ARM PrimeCell PL340 is used in this design. For more information please refer to the PrimeCell documentation. The default frequency for the DMC is 30MHz (OSCCLK1 divided by 4)

Since the Dynamic Memory Controller connects to high speed double data rate clocked devices it is necessary to make use of special pad and signal types to and from the Dynamic Memory. This block instantiated within the DMC

AXItoAPB BRIDGE

This is a bridge component to change from an AXI bus to an APB3 bus. This component contains decoding scheme for the bus, allowing 25 APB peripherals to be connected.

DMA

This is a direct memory access component. This design allows for the PrimeCell PL081 controllers to be added. It also allows for no DMA. For more information on the DMA blocks refer to the PrimeCell documentation.

PCI

The ARM provided image includes the option of including a Xilinx PCI component (version 3.0). This is a 32-bit wide PCI component, and operates at 33MHz. Refer to the EB Users Guide for further information.

CLCD

This is a color liquid crystal display controller. ARM PrimeCell PL111 is used in this design. For more information please refer to the PrimeCell documentation.

SYSTEM REGS

This contains a set of APB registers for hardware control of the EB. For a complete list of the functionality of these registers refer to the EB user guide.

SYSTEM CONTROL

This contains a generic set of system control registers. ARM ADK component SP810 is used in this design.

SERIAL BUS iI/F

This is a controller for the serial bus to the PISMO and Time of Year clock.

AACI

This is an advanced audio codec interface. ARM PrimeCell PL041 is used in this design. The FIFO is increased in size from the standard to better suit the FPGA operating environment (lower bus frequency). For more information please refer to the PrimeCell documentation. This block instantiates Xilinx block rams.

MCI

This is the multimedia card interface. ARM PrimeCell PL180 is used in this design. For more information please refer to the PrimeCell documentation.

MOUSE

These are keyboard and mouse interfaces. ARM PrimeCell PL050 is used in this design. For more information please refer to the PrimeCell documentation.

CHARACTER LCD

This is a character LCD display interface. It allows the system to communicate with the character LCD display fitted to the baseboard.

UART0-3

These are universal asynchronous receiver-transmitter interfaces (RS-232 serial). ARM PrimeCell PL010 is used in this design. For more information please refer to the PrimeCell documentation.

SSP

This is a synchronous serial port. ARM PrimeCell PL022 is used in this design. For more information please refer to the PrimeCell documentation. This block instantiates Xilinx block rams.

SCI

This is the smart card interface. ARM PrimeCell PL131 is used in this design. For more information please refer to the PrimeCell documentation.

WATCHDOG

This is a watchdog controller. It allows for the generation of an interrupt or reset after a defined time, to prevent against system lockup/failure. ARM ADK component SD805 is used in this design.

TIMERS 0-3

These are timer modules. ARM ADK component SP804 is used in this design.

GPIO 0-2

These are general purpose input/output modules. ARM PrimeCell PL061 is used in this design. For more information please refer to the PrimeCell documentation.

REAL TIME CLOCK

This is a real time clock module. Real time refers to total time from an event, and not actual real world time (measured using the Time Of Year Clock). ARM PrimeCell PL031 is used in this design. For more information please refer to the PrimeCell documentation.

PCI CONTROL REGS

This is the PCI control block. This contains a set of registers to allow configuration of the PCI system (if used). See the programmer's reference section for more information on the functionality of these registers.

3.3 Clock architecture

The clock architecture is carefully designed to minimize the skew (difference) in the clock edge position between different components across the system. The User Guides for all the boards used in this configuration explain the clock options. AXI clock (ACLK) and AHB clock (HCLK) inputs are both connected to OSCCLK3 on build C8.

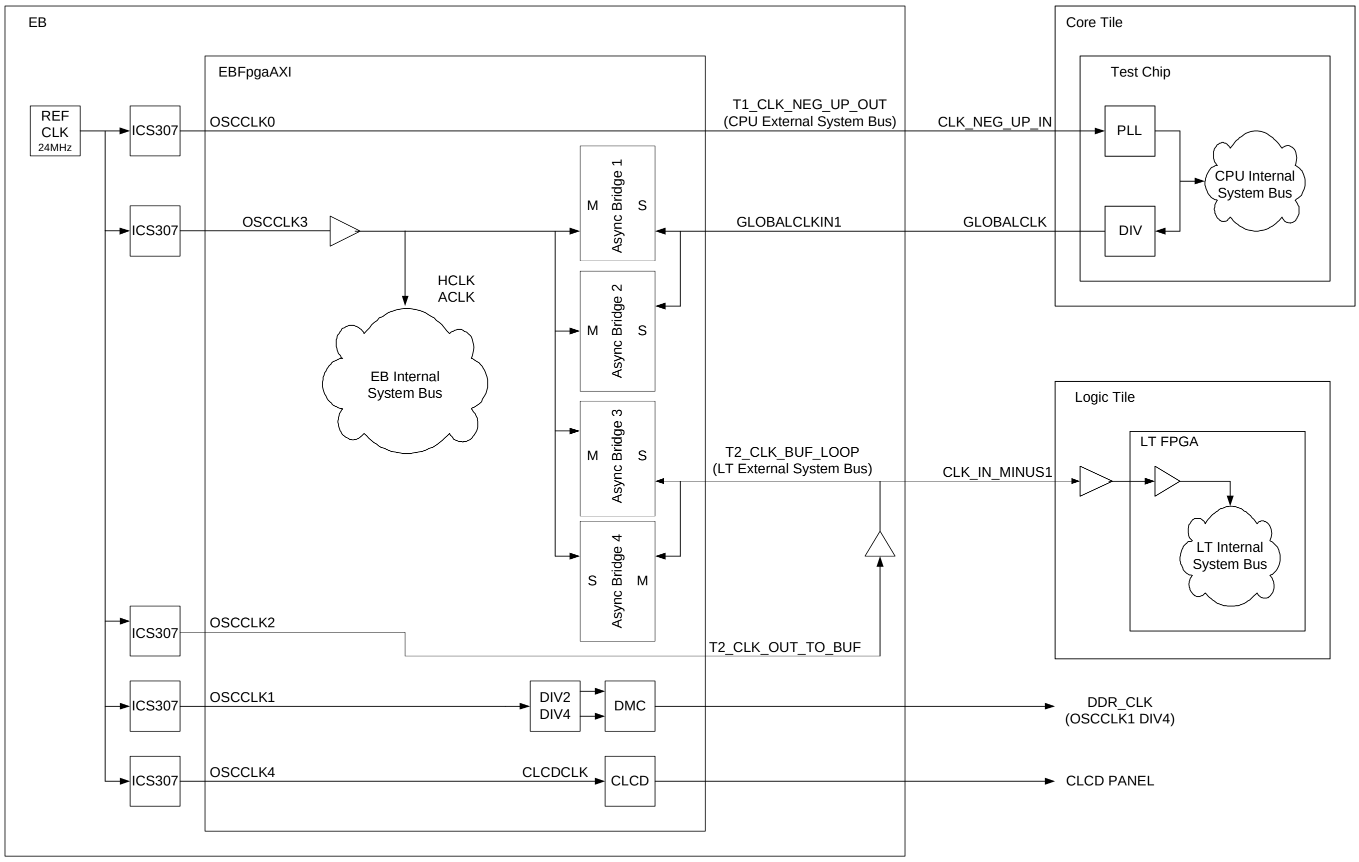


Figure 3-3 Clock architecture build C8

There are 6 clock domains in the build C8.

3.3.1 EB Internal System Bus

The EB Internal System Bus is clocked by HCLK. On build C8, HCLK is generated from OSCCLK3. HCLK is used to drive both AHB and AXI peripherals inside the FPGA.

3.3.2 CPU External System Bus

On build C8 the CPU External System Bus is clocked directly by OSCCLK0. The Core Tile multiplies this frequency by a value (default 7) to drive the CPU Internal System Bus. GLOBALCLKIN1 is generated from the Core Tile by dividing down the CPU Internal System Bus frequency by a value (default 8). This is used to clock data out of the Test Chip and drive the asynchronous bridge.

3.3.3 CPU Internal System Bus

The CPU Internal System Bus clock domain includes the ARM11 MPCore CPUs, the L220 Cache Controller and on-chip peripherals. The default behavior is for the PLL in the test chip to multiply the Core Tile input clock by 7 to drive this clock domain.

3.3.4 LT System Bus

The LT External System Bus is clocked by OSCCLK2.

The LT Internal System Bus in this diagram is the same frequency as the LT External System Bus. (LT Internal System Bus = LT External).

3.3.5 DDR_CLK

DDR_CLK is one quarter the frequency of OSCCLK1. OSCCLK1 is the source for all the DMC clocks, it is divided by 2 and 4 as required by the DMC..

3.3.6 CLCDCLK

CLCDCLK is directly connected to OSCCLK4.

3.3.7 Default operating frequencies

Table 3-1 Default operating frequencies

Clock	Use	Default Frequency build C8
OSCCLK0	CPU REFCLK	30MHz
OSCCLK1	DDR_CLK (OSCCLK1 divided by 4)	120MHz (100MHz for PCI build)
OSCCLK2	Tile site 2 (LT External System Bus)	24MHz
OSCCLK3	HCLK	30MHz
OSCCLK4	CLCD (CLCDCLK)	25MHz

3.3.8 EB Clock Configuration Switches

Boot switch SW8.6 is to select the bus clock and core frequency to make it easy to set the system clocks for most customers.

If the design is built with PCI, or enables the L220 cache controller in the Test chip, SW8.6 should be OFF (UP position).

Setting SW8.6 ON (DOWN position), will run the system faster, please note the design may not work correctly at the higher speed setting as it is outside the timing constraints of the design.

SW8.6 – sets the default operating frequency. **Table 3-2 System Bus clock selection on build C8**

SW8.6	Build C8
OFF	CPU External system bus (T1_CLK_NEG_UP_OUT) is 30MHz CPU Internal system bus is 210MHz (CLK_NEG_UP_IN X 7)
ON	CPU External system bus (T1_CLK_NEG_UP_OUT) is 33MHz CPU Internal system bus is 264MHz (CLK_NEG_UP_IN X 8)

SW8.7 – sets L2MASTNUM, this sets the number of masters coming from the L220 in the Test chip.

Table 3-3 L2MASTNUM selection

SW8.7	Use
OFF	1 Master is used from the L220, connects to T1X
ON	2 Masters are used from the L220, connects to T1X and T1Y

SW8.8 – sets L2BYPASS, this bypasses the L220 in the Test chip when in the ON position

Table 3-4 L2BYPASS selection

SW8.8	Use
OFF	L220 is not bypassed
ON	L220 is bypassed

These switches are only sampled at the initial power up, it is still possible to set any clock frequency in software using the EB SYS_OSCRESETx registers and the EB SYS_PLD_INIT and SYS_PLD_CTRLx registers. More information on these registers can be found in section 4.7.

3.3.9 PL340 Dynamic memory clocking

The PL340 Dynamic Memory Controller requires three input clocks to drive the Dynamic Memory Interface. These are mclk, mclk2x, and fbclk_in. For complete descriptions of the functionality of these clocks refer to the PL340 PrimeCell documentation (ARM DUI 0267). These clocks are synchronous to the HCLK/ACLK which allows for a lower latency on transfers (not requiring an asynchronous FIFO).

3.4 CT11MPCore reset structure

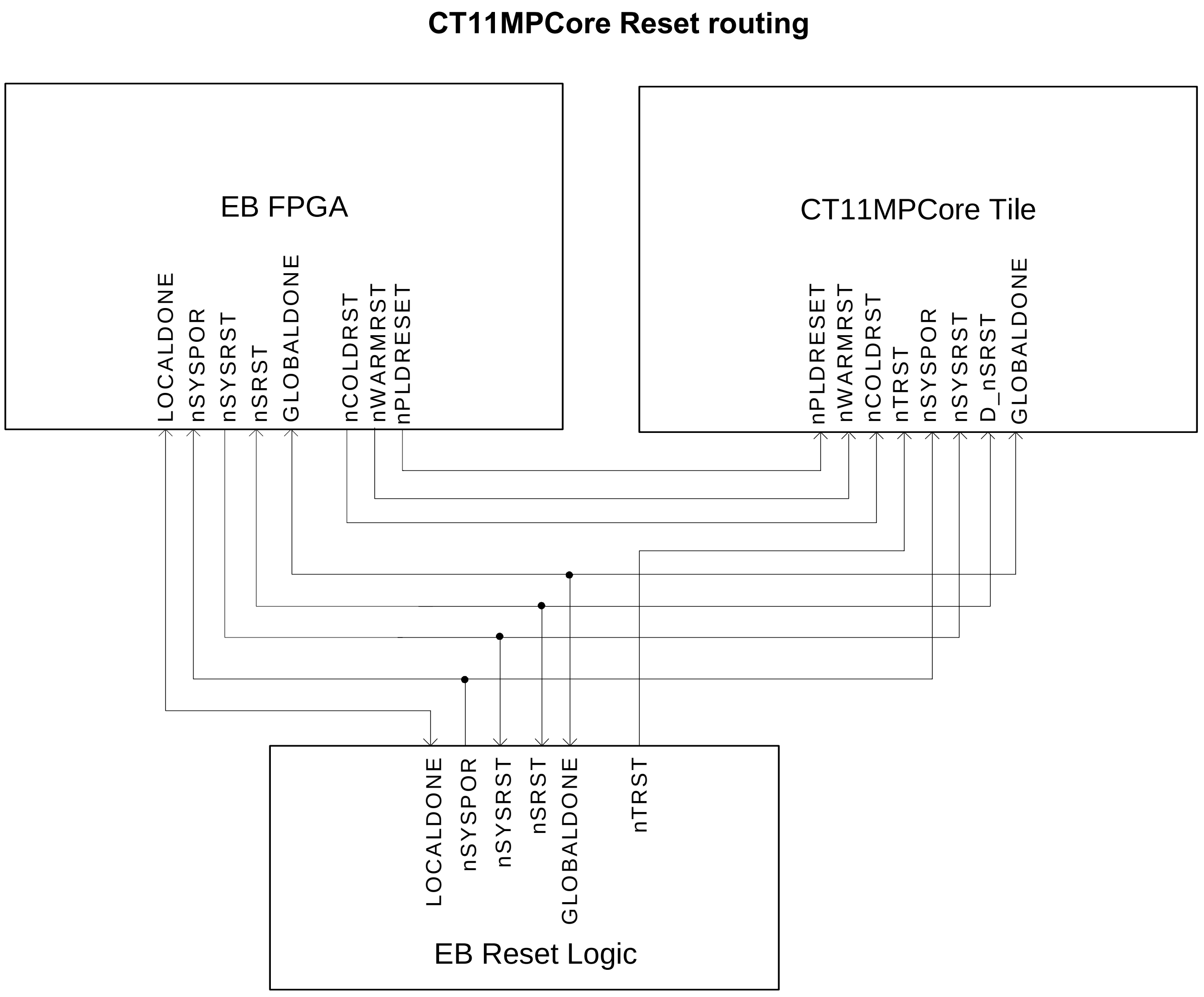


Figure 3-4 CT11MPCore reset routing

Table 3-5 Reset and related signals

Name	**Width (bits)**	**Direction to/from Reset Logic**	**Note**
nBOARDPOR	1	n/a	Board power on reset
LOCALDONE	1	Output	Goes high when the EB FPGA has configured
GLOBALDONE	1	Input open drain	Goes high when all FPGAs have configured and the serial PLD interfaces have configured the TestChip PLL and Tile Mux/Switch
nSYSPOR	1	Output	Goes high approximately 7µs after GLOBAL_DONE is high
nSYSRST	1	Output	Goes high approximately 20µs after nSYSPOR goes high or

			goes low after nSRST goes low and returns high approximately 20us after nSRST goes high
nPLDRESET	1	Output	Resets and starts the PLD serial data transfer
nSRST	1	Input open drain	System reset input from the debugger on the JTAG connector.Generates an nSYSRST request
nTRST	1	Input open drain	JTAG Test Access Port (TAP) reset input from the debugger on the JTAG connector
nCOLDRESET	1	Output	From nSYSRST, drives nPORESET, nWDRESET, nSYSPORESET
nWARMRESET	1	Output	From nSYSRST, drives nCPURESET

Note: a lower case letter ‘n’ before the signal signifies an active low signal

3.4.1 Reset sequence

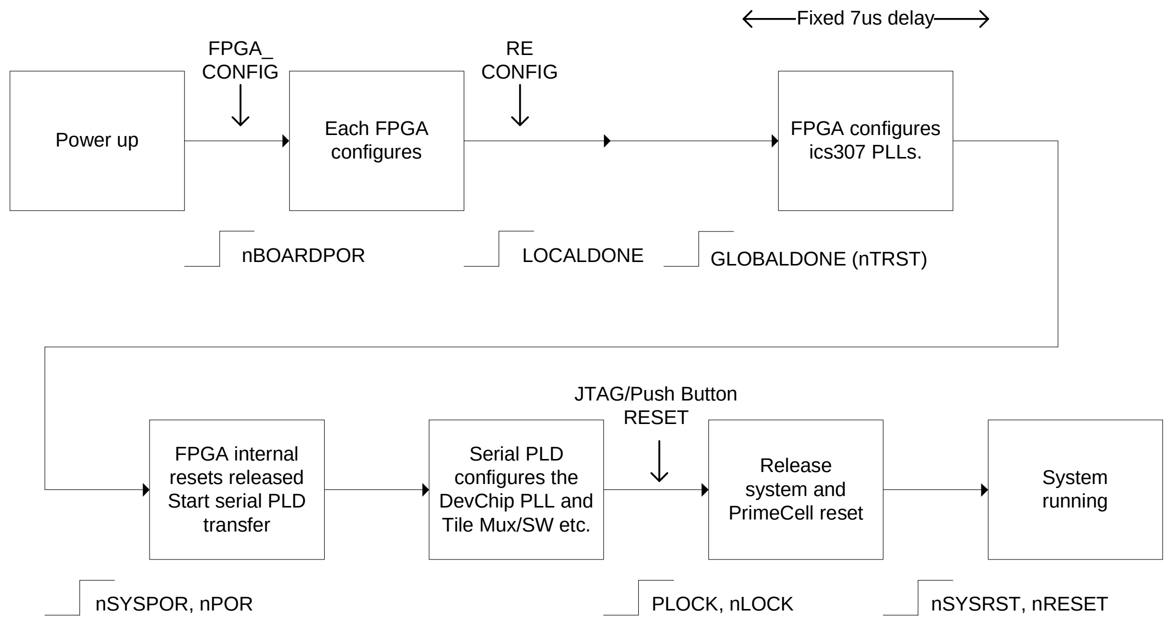


Figure 3-5 Reset sequence

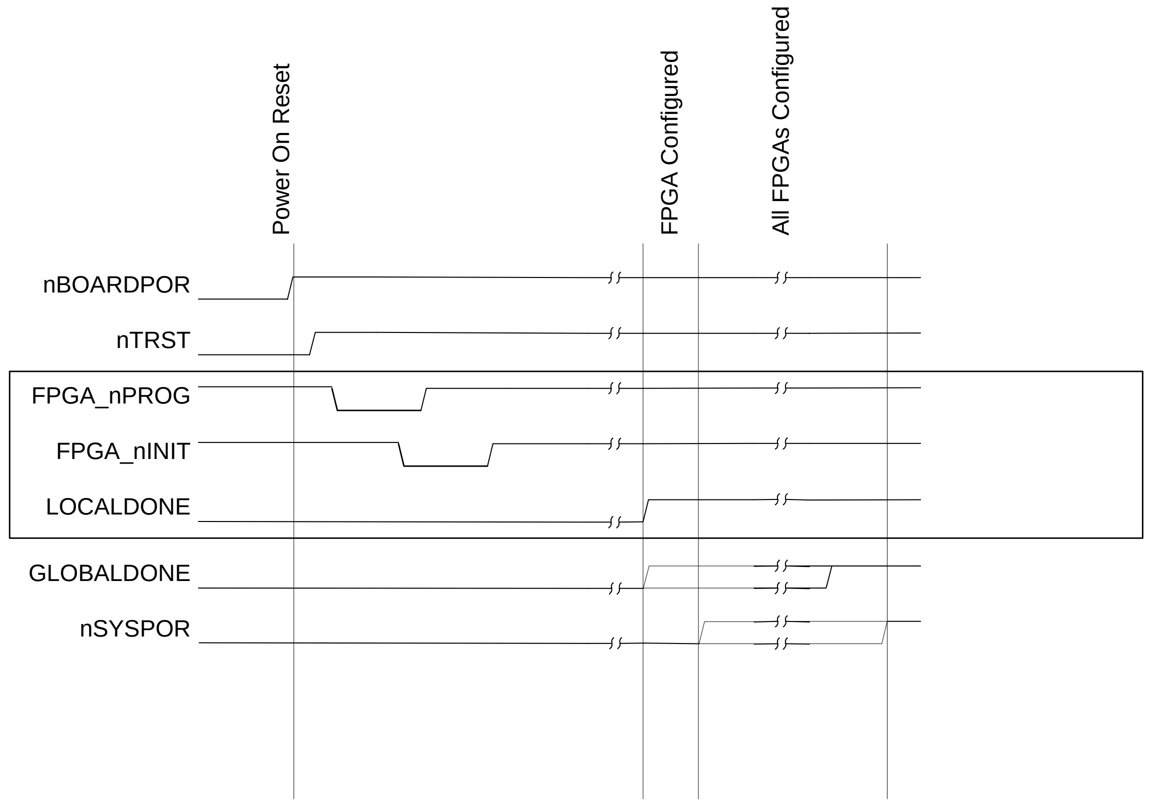


Figure 3-6 FPGA initialisation sequence timing diagram

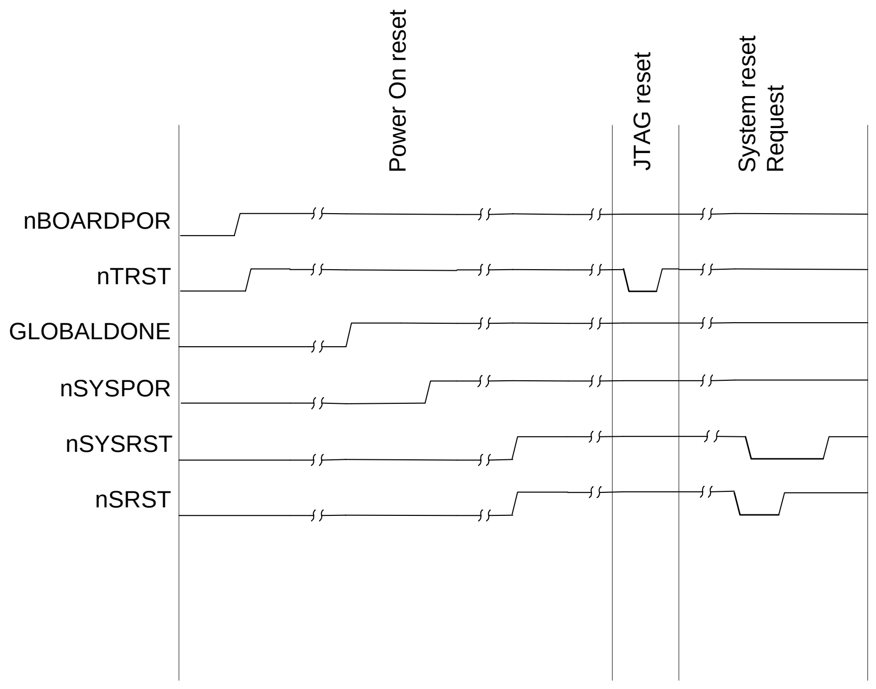


Figure 3-7 Reset sequence timing diagram

Note there is an important difference between nTRST and nSRST. nTRST resets the JTAG Test Access Port (TAP) of the ARM core. The TAP holds hardware breakpoints, and should be reset by the debugger when it connects to the ARM core. nSRST resets the system, not including the TAP. This means you can use the debugger to reset the system without losing any breakpoints or other configuration stored in the TAP.

3.5 Single Core Boot

SW8.5 – allows the user to boot the MPCore with only CPU0, this is useful to recover if the Flash has been erased, by only allowing CPURESET[0] to be brought out of reset, the remaining CPUs are held in reset. This allows the user to connect to the system with a debugger and reprogram the flash.

Table 3-6 Single Core Boot

SW8.5	Use
OFF	CPURESET[3:0] all can be brought out of reset (normal operation)
ON	Only CPURESET[0] only CPU0 can be brought out of reset .

3.6 EB Interrupt routing

Two interrupt modes are implemented in this application note. The figure below details the structure of the interrupt logic in the design.

’Legacy mode’ provides CPU 0 with a single nIRQ interrupt, as used by all single-core ARM processors. This is the default mode of operation.

’Normal mode’ provides 16 external interrupt signals to the MPCore’s Interrupt Distributor. The Interrupt Distributor then assigns the IRQ to one of the CPUs. This is only used on multi-core processors.

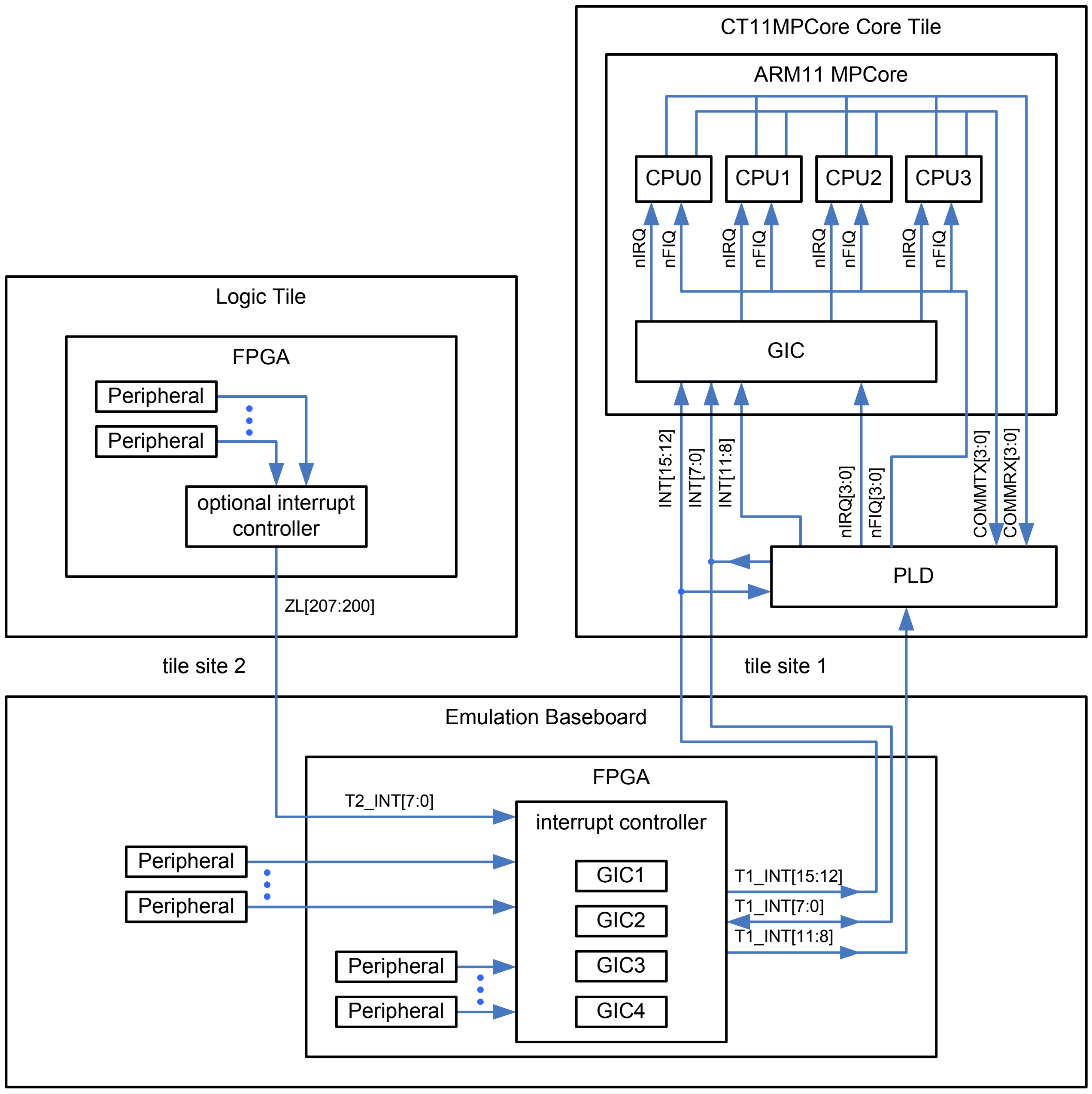


Figure 3-8 Interrupt routing

Control of the interrupt mode is selected by the Core Tile PLD (serial bits INTMODE[2:0]), as shown in the table below.

Table 3-7- EB interrupt mode control bits

INTMODE	Mode	Comments
b000	Legacy mode no FIQ	Used by boot monitor
b001	Normal mode with DCC and no FIQ	Not supported
b010	Normal mode without DCC and no FIQ	Recommended for an SMP system
b011	Reserved for future use	Do not use
b100	Legacy mode with FIQ	Can be useful for a single CPU
b101	Normal mode with DCC and FIQ	Not supported
b110	Normal mode without DCC and with FIQ	Not supported
b111	Reserved for future use	Do not use

SMP = symmetric multiprocessing

For example software refer to the boot monitor source file sys_interrupts_eb.c which is provided on the Versatile CD shipped with the EB.
Legacy mode is the same interrupt mode as used by all non MultiCore processors and is the default mode of operation.
Normal mode could also be called MultiCore mode and is used only on MultiCore processors.

3.6.1 Legacy mode interrupt routing

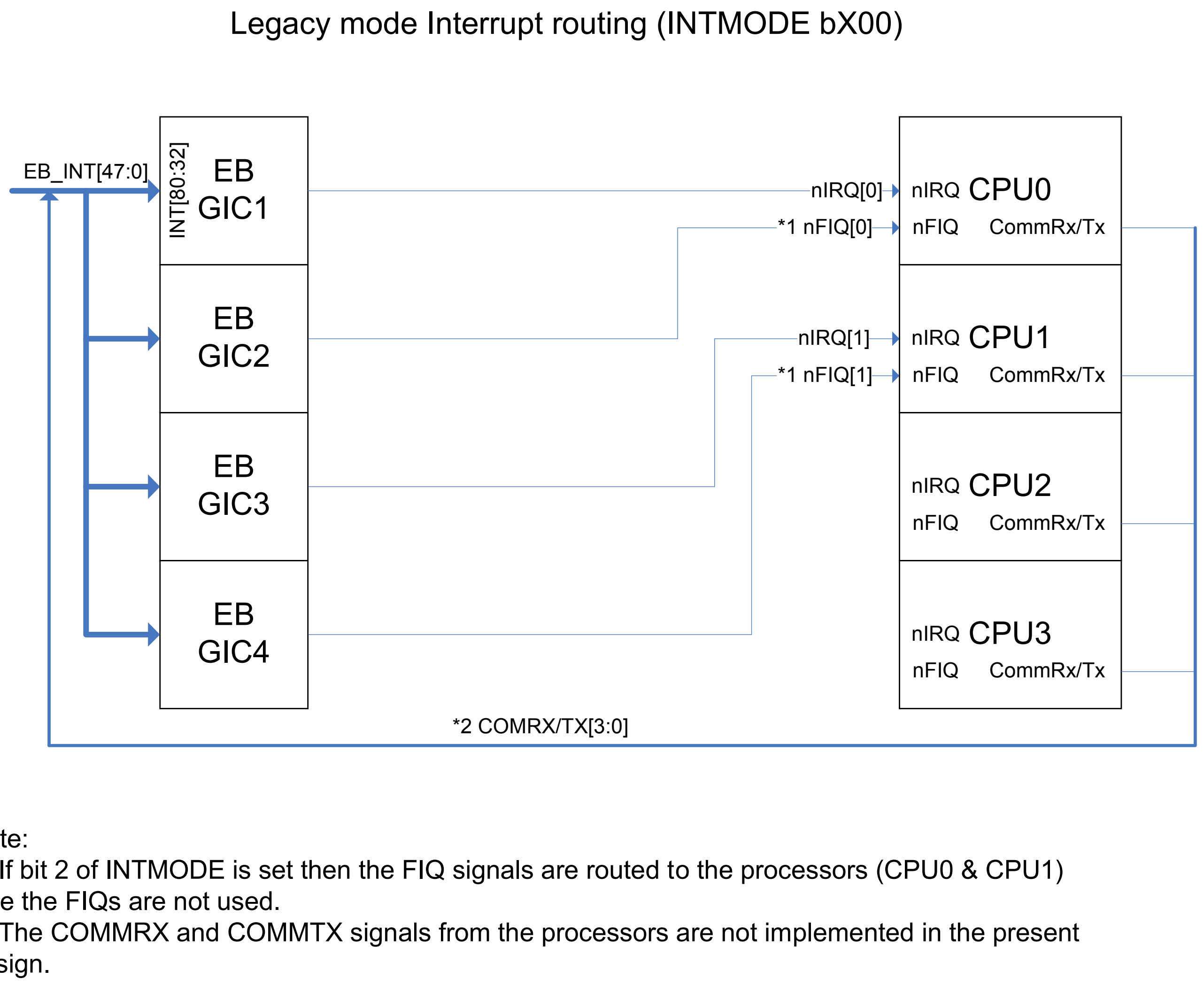

Legacy mode Interrupt routing (INTMODE bX00)
EB_INT[47:0]
INT[80:32]
EB GIC1
EB GIC2
EB GIC3
EB GIC4
nIRQ[0]
*1 nFIQ[0]
nIRQ[1]
*1 nFIQ[1]
nIRQ CPU0 nFIQ CommRx/Tx
nIRQ CPU1 nFIQ CommRx/Tx
nIRQ CPU2 nFIQ CommRx/Tx
nIRQ CPU3 nFIQ CommRx/Tx
*2 COMRX/TX[3:0]
Note:
*1 If bit 2 of INTMODE is set then the FIQ signals are routed to the processors (CPU0 & CPU1) else the FIQs are not used.
*2 The COMMRX and COMMTX signals from the processors are not implemented in the present design.

Figure 3-9 Legacy mode (INTMODE bX00)

Table 3-8 Legacy interrupt table

Reference Name	Source Signal Name	Direction	Destination Signal Name	Tile Signal Name
INT[0]	MPCore COMMRX[0]	MPCore to EB	EB Not Used	Z200
INT[1]	MPCore COMMRX[1]	MPCore to EB	EB Not Used	Z201
INT[2]	MPCore COMMRX[2]	MPCore to EB	EB Not Used	Z202
INT[3]	MPCore COMMRX[3]	MPCore to EB	EB Not Used	Z203
INT[4]	MPCore COMMTX[0]	MPCore to EB	EB Not Used	Z204
INT[5]	MPCore COMMTX[1]	MPCore to EB	EB Not Used	Z205
INT[6]	MPCore COMMTX[2]	MPCore to EB	EB Not Used	Z206
INT[7]	MPCore COMMTX[3]	MPCore to EB	EB Not Used	Z207
INT[8]	EB GIC1	EB to MPCore	MPCore CPU0 nIRQ[0]	Z208
INT[9]	EB GIC3	EB to MPCore	MPCore CPU1 nIRQ[1]	Z209
INT[10]	Not Used	EB to MPCore	MPCore Not Used	Z210
INT[11]	Not Used	EB to MPCore	MPCore Not Used	Z211
INT[12] *1	EB GIC2	EB to MPCore	MPCore CPU0 nFIQ[0]	Z212
INT[13] *1	EB GIC4	EB to MPCore	MPCore CPU1 nFIQ[1]	Z213
INT[14]	Not Used	EB to MPCore	MPCore Not Used	Z214
INT[15]	Not Used	EB to MPCore	MPCore Not Used	Z215

Note: *1 If bit 2 of INTMODE is set (INTMODE b100) then the nFIQ signals are routed to the processors (CPU0 & CPU1) else (INTMODE b000) the FIQs are not used.
Reference name is the name used in the diagram to should signal path from EB to CT11MPCore in Figure 3-8

3.6.2 Normal mode without DCC interrupt routing

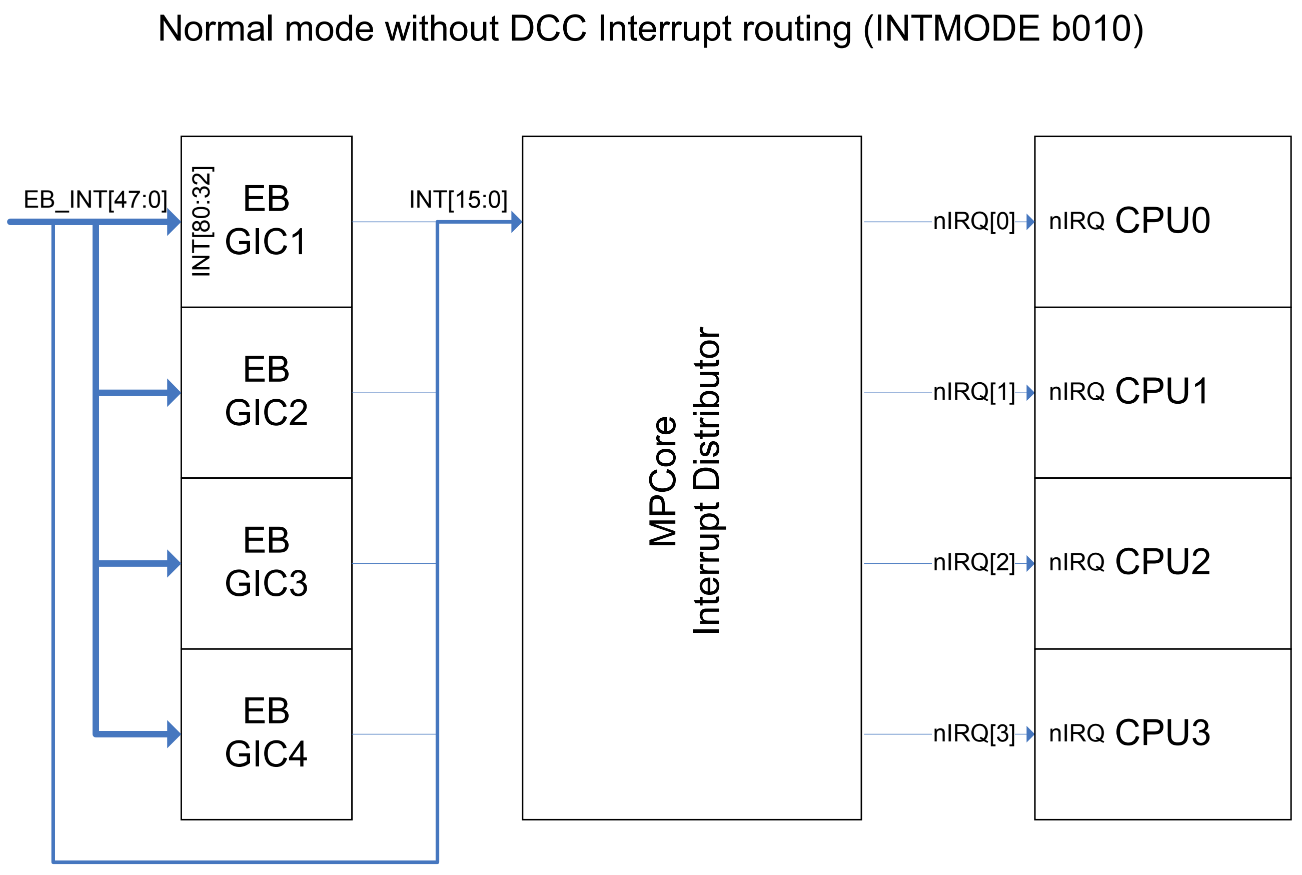


Figure 3-10 Normal mode without DCC (INTMODE b010)

Table 3-9 Normal mode without DCC (INTMODE b010)

Reference Name	Source Signal Name	Direction	Destination Signal Name	Tile Signal Name
INT[0]	AACIINTR	EB to MPCore	MPCore GIC INT[0]	Z200
INT[1]	TIMERINT01	EB to MPCore	MPCore GIC INT[1]	Z201
INT[2]	TIMERINT23	EB to MPCore	MPCore GIC INT[2]	Z202
INT[3]	USBnINT	EB to MPCore	MPCore GIC INT[3]	Z203
INT[4]	UARTINT0	EB to MPCore	MPCore GIC INT[4]	Z204
INT[5]	UARTINT1	EB to MPCore	MPCore GIC INT[5]	Z205
INT[6]	RTCINT	EB to MPCore	MPCore GIC INT[6]	Z206
INT[7]	KMIINT0	EB to MPCore	MPCore GIC INT[7]	Z207
INT[8]	KMIINT1	EB to MPCore	MPCore GIC INT[8]	Z208
INT[9]	ETHINTR	EB to MPCore	MPCore GIC INT[9]	Z209
INT[10]	GIC1	EB to MPCore	MPCore GIC INT[10]	Z210
INT[11]	GIC2	EB to MPCore	MPCore GIC INT[11]	Z211
INT[12]	GIC3	EB to MPCore	MPCore GIC INT[12]	Z212
INT[13]	GIC4	EB to MPCore	MPCore GIC INT[13]	Z213
INT[14]	MCIINTR0	EB to MPCore	MPCore GIC INT[14]	Z214
INT[15]	MCIINTR1	EB to MPCore	MPCore GIC INT[15]	Z215

Reference name is the name used in the diagram to should signal path from EB to CT11MPCore in Figure 3-8

4 Hardware Description

4.1 EB Top Level (EBFpga.v)

The top level of the design is of particular importance. This level defines the mapping from the HDRX, HDRY and HDRZ busses from the tile site into their functional allocations.

4.2 EB AXI Subsystem (EBFpgaCT11MPCore.v)

This level connects all the components together and ties off static pins. This includes all the major blocks as shown in Figure 2.2

4.3 EB AXI Multiplexing Scheme

By using a 2:1 multiplexer and latch scheme as shown below it is possible to reduce the pin count for the AXI buses into a realistic size for implementation on the Tile XL and YL headers.

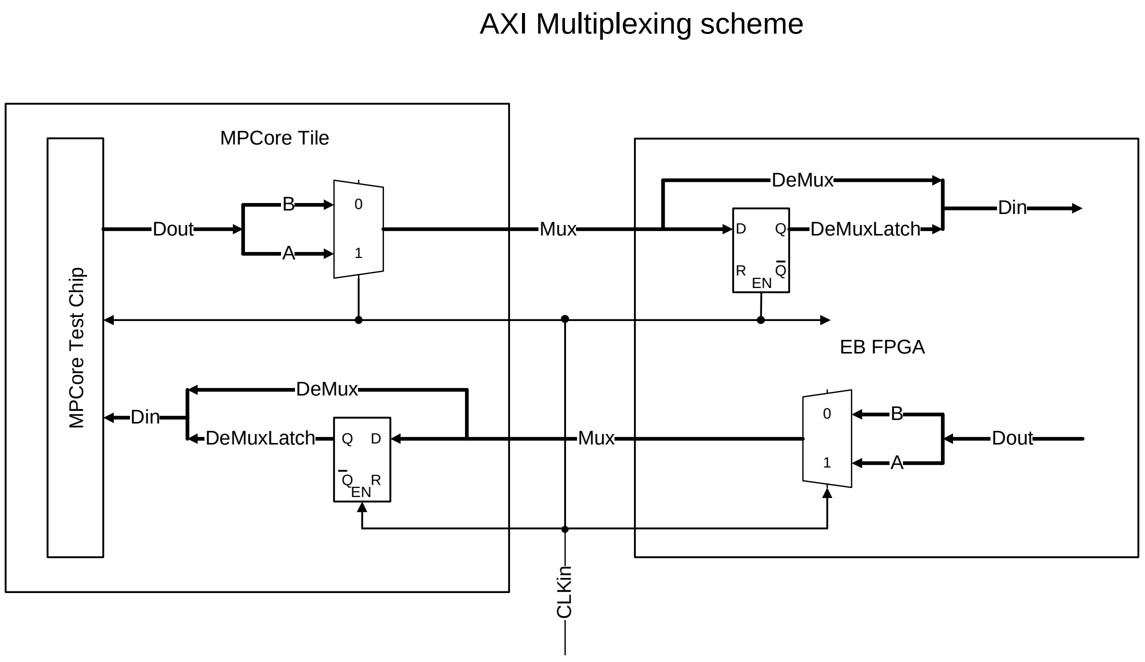
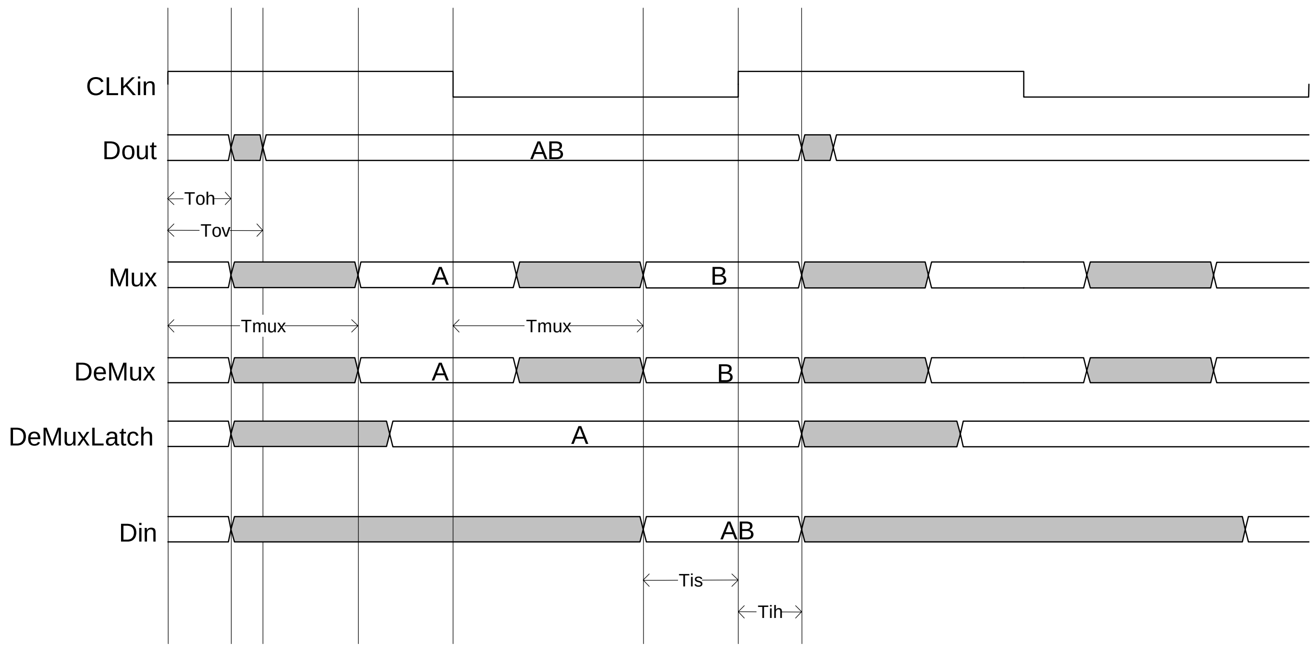


Figure 4-1 AXI multiplexing scheme

The output data is multiplexed on the level of CLKin (which in this design is ACLK), to generate the multiplexed bus (Mux). The demultiplexing is performed by latching the data (A) generated on the high level of CLKin when CLKin goes low (DeMuxLatch). The data (B), generated on the low side of CLKin is passed straight through (DeMux). This design assumes data is always generated and captured on the rising edge of CLKin.

Most of the Valid and Ready signals on AXI are not be multiplexed in this way to reduce latency, the exceptions are noted in Table 4-3 Header HDRX and HDRY AXI pin allocationTable 4-3 Header HDRX and HDRY AXI pin allocation.

The design supplies four multiplexed AXI buses (T1X,T1Y,T2X & T2Y). The following timing diagram shows the data flow through the design with expected delays from standard components.



Toh	min	= 0ns	(output hold)
Tov	max	= 2ns	(output valid)
Tis	max	= 2ns	(Input setup)
Tih	max	= 0ns	(input hold)
Tmux	max	= 6ns	(multiplexer and board delay)

The CLKin is the clock driven into the Logic Tile from the board . All I/O timing must be with respect to this clock .

Figure 4-2 AXI timing requirements for CT11MPCore

4.4 CT11MPCore and AXI (AMBA 3) pin allocation differences

A number of signals from the CT11MPCore test chip differ from the generic tile header pin allocation as shown below, refer to CT11MPCore user guide for more information on this. Refer to the MPCore TRM for more information.

Table 4-1 CT11MPCore and AXI pin allocation differences

Signal	CT11MPCore	Generic AXI on X/Y	Notes
AWUSER[4:0]	Required	Not available	Not defined by AMBA 3 (for L220)
ARUSER[4:0]	Required	Not available	Not defined by AMBA 3 (for L220)

The following signals are multiplexed and connected to the Z headers along with the interrupt and serial PLD control signals described later on in this document,

Table 4-2 Z bus signals

Z Bus	HDRZ pin	CT11MPCore signal	Function
200	112	INT0	CT11MPCore interrupts (inputs)
201	110	INT1	CT11MPCore interrupts (inputs)
202	108	INT2	CT11MPCore interrupts (inputs)
203	106	INT3	CT11MPCore interrupts (inputs)
204	104	INT4	CT11MPCore interrupts (inputs)
205	102	INT5	CT11MPCore interrupts (inputs)
206	100	INT6	CT11MPCore interrupts (inputs)
207	98	INT7	CT11MPCore interrupts (inputs)
208	96	INT8	CT11MPCore interrupts (inputs)
209	94	INT9	CT11MPCore interrupts (inputs)
210	92	INT10	CT11MPCore interrupts (inputs)
211	90	INT11	CT11MPCore interrupts (inputs)
212	88	INT12	CT11MPCore interrupts (inputs)
213	86	INT13	CT11MPCore interrupts (inputs)
214	84	INT14	CT11MPCore interrupts (inputs)
215	82	INT15	CT11MPCore interrupts (inputs)
216	80	----	Reserved
217	78	AWUSER0/ARUSER0	From AXI Port 0 (Y header)
218	76	AWUSER1/ARUSER1	
219	74	AWUSER2/ARUSER2	
220	72	AWUSER3/ARUSER3	
221	70	AWUSER4/ARUSER4	
222	68	AWUSER0/ARUSER0	From AXI Port 1 (X header)
223	66	AWUSER1/ARUSER1	
224	64	AWUSER2/ARUSER2	
225	62	AWUSER3/ARUSER3	
226	60	AWUSER4/ARUSER4	
227	58	PLDD1	Serial PLD control (PLD I/P)
228	56	PLDD0	(PLD I/P)
229	54	PLDRESETn	
230	52	PLDCLK	

4.5 Header HDRX and HDRY AXI pin allocation

The four AXI buses (T1X, T2X, T1Y, T2Y) connect to the HDRX and HDRY Tile headers. The pin connections are the same for HDRX and HDRY.

Table 4-3 Header HDRX and HDRY AXI pin allocation

X/Y Bus	HDRX pin	HDRY pin	signal	X/Y Bus	HDRX pin	HDRY pin	signal
0	180	179	WDATA0/32	72	36	35	BID1/3
1	178	177	WDATA1/33	73	34	33	BID4/BID5
2	176	175	WDATA2/34	74	32	31	BRESP0/1
3	174	173	WDATA3/35	75	30	29	BVALID
4	172	171	WDATA4/36	76	28	27	BREADY
5	170	169	WDATA5/37	77	26	25	ARADDR0/16
6	168	167	WDATA6/38	78	24	23	ARADDR1/17
7	166	165	WDATA7/39	79	22	21	ARADDR2/18
8	164	163	WDATA8/40	80	20	19	ARADDR3/19
9	162	161	WDATA9/41	81	18	17	ARADDR4/20
10	160	159	WDATA10/42	82	16	15	ARADDR5/21
11	158	157	WDATA11/43	83	14	13	ARADDR6/22
12	156	155	WDATA12/44	84	12	11	ARADDR7/23
13	154	153	WDATA13/45	85	10	9	ARADDR8/24
14	152	151	WDATA14/46	86	8	7	ARADDR9/25
15	150	149	WDATA15/47	87	6	5	ARADDR10/26
16	148	147	WDATA16/48	88	4	3	ARADDR11/27
17	146	145	WDATA17/49	89	2	1	ARADDR12/28
18	144	143	WDATA18/50	90	1	2	ARADDR13/29
19	142	141	WDATA19/51	91	3	4	ARADDR14/30
20	140	139	WDATA20/52	92	5	6	ARADDR15/31
21	138	137	WDATA21/53	93	7	8	ARID0/2
22	136	135	WDATA22/54	94	9	10	ARID1/3
23	134	133	WDATA23/55	95	11	12	ARLEN0/2
24	132	131	WDATA24/56	96	13	14	ARLEN1/3
25	130	129	WDATA25/57	97	15	16	ARSIZE0/1
26	128	127	WDATA26/58	98	17	18	ARID4/ARPROT2
27	126	125	WDATA27/59	99	19	20	ARPROT0/1
28	124	123	WDATA28/60	100	21	22	ARBURST0/1
29	122	121	WDATA29/61	101	23	24	ARLOCK0/1
30	120	119	WDATA30/62	102	25	26	ARCACHE0/2
31	118	117	WDATA31/63	103	27	28	ARCACHE1/3
32	116	115	WID0/2	104	29	30	ARVALID/ARID5
33	114	113	WID1/3	105	31	32	ARREADY
34	112	111	WSTRB0/4	106	33	34	RDATA0/32
35	110	109	WSTRB1/5	107	35	36	RDATA1/33
36	108	107	WSTRB2/6	108	37	38	RDATA2/34
37	106	105	WSTRB3/7	109	39	40	RDATA3/35
38	104	103	WLAST/WID4	110	41	42	RDATA4/36
39	102	101	WVALID/WID5	111	43	44	RDATA5/37
40	100	99	WREADY	112	45	46	RDATA6/38
41	98	97	AWADDR0/16	113	47	48	RDATA7/39

42	96	95	AWADDR1/17	114	49	50	RDATA8/40
43	94	93	AWADDR2/18	115	51	52	RDATA9/41
44	92	91	AWADDR3/19	116	53	54	RDATA10/42
45	90	89	AWADDR4/20	117	55	56	RDATA11/43
46	88	87	AWADDR5/21	118	57	58	RDATA12/44
47	86	85	AWADDR6/22	119	59	60	RDATA13/45
48	84	83	AWADDR7/23	120	61	62	RDATA14/46
49	82	81	AWADDR8/24	121	63	64	RDATA15/47
50	80	79	AWADDR9/25	122	65	66	RDATA16/48
51	78	77	AWADDR10/26	123	67	68	RDATA17/49
52	76	75	AWADDR11/27	124	69	70	RDATA18/50
53	74	73	AWADDR12/28	125	71	72	RDATA19/51
54	72	71	AWADDR13/29	126	73	74	RDATA20/52
55	70	69	AWADDR14/30	127	75	76	RDATA21/53
56	68	67	AWADDR15/31	128	77	78	RDATA22/54
57	66	65	AWID0/2	129	79	80	RDATA23/55
58	64	63	AWID1/3	130	81	82	RDATA24/56
59	62	61	AWLEN0/2	131	83	84	RDATA25/57
60	60	59	AWLEN1/3	132	85	86	RDATA26/58
61	58	57	AWSIZE0/1	133	87	88	RDATA27/59
62	56	55	AWID4/AWPROT2	134	89	90	RDATA28/60
63	54	53	ARM_nRESET	135	91	92	RDATA29/61
64	52	51	AWPROT0/1	136	93	94	RDATA30/62
65	50	49	AWBURST0/1	137	95	96	RDATA31/63
66	48	47	AWLOCK0/1	138	97	98	RID0/2
67	46	45	AWCACHE0/2	139	99	100	RID1/3
68	44	43	AWCACHE1/3	140	101	102	RRESP0/1
69	42	41	AWVALID/AWID5	141	103	104	RLAST/RID4
70	40	39	AWREADY	142	105	106	RVALID/RID5
71	38	37	BID0/2	143	107	108	RREADY

4.6 CT11MPCore configuration PLD serial interface

Serial control of the configuration PLD is achieved through a three wire interface. Two reset signals nBOARDPOR and nPLDRESET are also used to ensure the PLD logic is in the defined stated before the data stream is loaded.

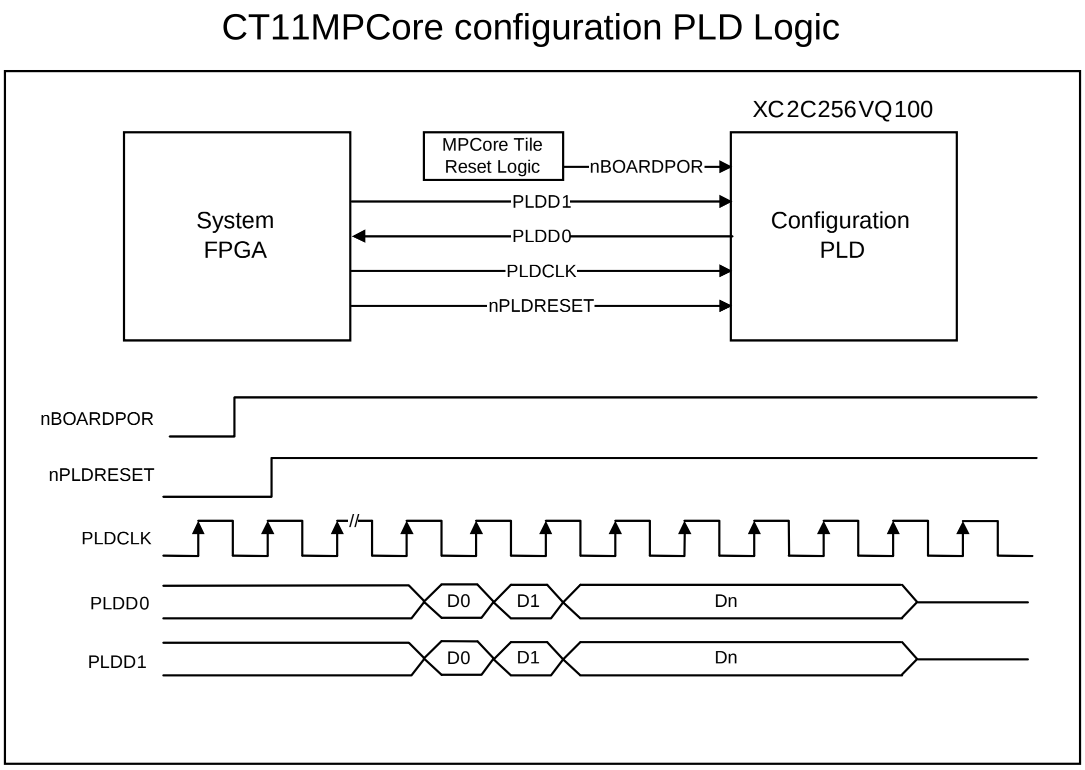


Figure 4-3 CT11MPCore configuration PLD

All data bits are clocked into the PLD on the rising edge of PLDCLK. The nPLDRESET enable defines when data is shifted into and out of the PLD starting with D0. The serial stream is constantly updating ensuring both FPGA and PLD registers are identical.

4.7 Serial write data register

The system FPGA contains a 60-bit shift register, the contents of which are continuously sent to the data register in the configuration PLD. The write data register is mapped on to the external pins of the configuration PLD with the following signals. RD_DIV[0] is the first piece of data transmitted, the others follow in the order shown.

Serial Bits	**Field name**	**Reset value**	**Definition**	**User access via**
4	RD_DIV[3:0]	b0110	CPU Internal Bus clock divider (default is ÷ 7)	SYS_PLD_CTRL1
16	RD_CTRL[31:16]	0x1191	MPCore PLL Clock control	SYS_PLD_CTRL1
1	PLLUPDATE	b0	Start PLL update flag (Unused in this design)	No access
1	L2BYPASS	b0	MPCore L2 bypassing, 0 = L2 enabled, 1 = L2 bypass	SYS_PLD_CTRL1
1	L2MASTNUM	b0	MPCore L2 ports, 0 = M0 enabled, 1 = M0 and M1 enabled	No access - Defined as a constant in RTL

1	MASTNUM	b1	MPCore, 0 = 1 master port (AXI port M1), 1 = 2 master ports (AXI ports M0 and M1)	SYS_PLD_CTRL1
1	AXInOE0	b0	AXI Mux output enable. (1 = Disable to isolate Y headers from AXI bus)	No access - Defined as a constant in RTL
1	HDRZEN	b0	Header Z optional signal mux enable (1 = optional HDRZ signals enabled)	SYS_PLD_CTRL1
4	VINITHI[3:0]	b0000	MPCore high-vecs mode	SYS_PLD_CTRL1
2	CFGEND[1:0]	b00	MPCore endianess configuration	SYS_PLD_CTRL1
4	nCPURESET[3:0]	b1111	Individual CPU reset	SYS_PLD_CTRL1
12	DBGMUX[11:0]	0x000	Debug matrix select	SYS_PLD_CTRL1
3	INTMODE[2:0]	b000	Interrupt mode. x00:Legacy, x01:Reserved x10:Normal no DCC, x11:Reserved 1xx:FIQ[3:0] enable.	SYS_PLD_CTRL1
8	DACDAT[7:0]	0x00	DAC data for CHA/B	SYS_VOLTAGEx
1	DACSEL	b0	DAC channel select (0 = CHA/VDDCORE, 1 = CHB/AVDD)	No access - Toggled automatically
60	**Total**			

Table 4-4 Serial write data register

These configuration values are loaded with default values during a power-on reset condition. Some of the settings can be subsequently changed by writing to the relevant EB registers and then performing a reset, to have the changes take effect. This is described in section 4.9. Some of the settings cannot be changed without modifying and re-building the FPGA image. Please see section 0 for register descriptions.

4.8 Serial read data register

The system FPGA continuously reads a 43 bit value from the data register in the configuration PLD. The read data register is mapped from the external pins of the configuration PLD with the following signals. PLOCK is the first piece of data to be transferred; the others follow in the sequence shown.

Serial Bits	Pin name	Definition	User access via
1	PLOCK	MPCore PLL lock indicator	No access – input to reset controller
1	ISP0nLOCK	ispClock5520 PLL lock indicator 0	No access – input to reset controller
1	ISP1nLOCK	ispClock5520 PLL lock indicator 1	No access – input to reset controller

1	PLLUPDATED	MPCore PLL update complete	No access – input to reset controller
4	RESETREQ[3:0]	Individual watchdog reset request	SYS_PLD_CTRL2
4	SMPnAMP[3:0]	Indicates AMP or SMP mode for each MP11 CPU	SYS_PLD_CTRL2
4	STANDBYWFI[3:0]	Indicates whether a CPU is in WFI state	SYS_PLD_CTRL2
4	COMMTX[3:0]	Comms channels receive	SYS_PLD_CTRL2
4	COMMRX[3:0]	Comms channels transmit	SYS_PLD_CTRL2
12	ADCDAT[11:0]	Power sensing 12 bit ADC value (selected by ADCSEL[2:0])	SYS_VOLTAGEx
3	ADCSEL[2:0]	ADC channel currently being converted (ADCDAT value is from channel ADCSEL – 1)	No Access – Used by FPGA to update SYS_VOLTAGEx
4	PLDVER[3:0]	PLD build version	SYS_PLD_CTRL2
43	**Total**		

Table 4-5 Serial Data Read register

Some of these data values can be read from memory mapped registers within the EB FPGA. Others cannot be accessed by the user.

4.9 CT11MPCore PLL configuration

REFCLK for the ARM11 MPCore test chip is derived from the baseboard (T1_CLK_NEG_UP_OUT). On power up the clock generator (OSCCLK0) defaults to 24MHz. Transferring the 20bit clock setting value to the CT11MPCore test chip requires several transfers to set both the clock divide and PLL values. The data transfers and control of the CONFIGINIT pin is the responsibility of the configuration PLD. PLLUPDATED, PLL lock, PLOCK and nLOCK are continually monitored by the returned serial PLD data and waited on before the FPGA internal System and PrimeCell resets are released (see Section 3.4).

Following GLOBALDONE of the EB FPGA being asserted, the configuration PLD transfers a default value to the CT11MPCore PLL registers via RDATA1[31:0]. This sets the default core frequency to 7 times the Core Tile AXI bus frequency giving 210MHz for the CPU internal system bus and 26MHz CPU external system bus frequency (dividing by 8) on build C8.

The SYS_PLD_INIT register can be used to change the default PLL control and CPU External System Bus divider values. (Note that SYS_PLD_INIT[19:16] is hard coded to b0001which means M divider is always b0, PBSTBY and STBY are always b0 and PLL enable is b1). This update will only take place after the EB nPB reset switch is pressed to ensure that all system PLLs have locked. The sequence for changing the CT11MPCore PLL should be:

1. Unlock the Lock register by writing 0xA05F to (base + 0x020).
2. Write the new PLL control and divide value to SYS_PLD_INIT register (base + 0x07C).
3. Press the nPB reset switch on the EB.

Table 4-6 CT11MPCore PLL configuration

Pin name	Default value	Definition
RD[31:16]	0x2261	MPCore PLL control: [31:28] PB divider [27:24] PA divider [23:20] N divider [19] M divider [18] PBSTBY [17] STBY [16] PLL enable
RD[15:4]	0x000	Not used
RD[3:0]	b0111	MPCore External System Bus clock divider

For a more detailed description of the operation of the PLL and bus clock divider, please refer to the Test Chip Hardware Description section of the CT11MPCore Tile User Guide. In summary, the PLL output frequency (CPU Internal System Bus clock) is given by the following equation:

$$f = \frac{REFCLK \times PAval \times Nval}{Mval \times PBval}$$, where:

PAval = PA[3:0] + 1

PBval = PB[3:0] + 1

Mval = M + 1

Nval = N[3:0] + 1

For example, with the default settings:

$$f_{CPU} = \frac{30MHz \times 3 \times 7}{1 \times 3} = 210MHz$$

4.10 MPCore power measurement

The MPCore test chip can be put into low power state (WFI) when not executing code. Linux for example makes use of this feature to significantly reduce the total power consumption. A feature of the test chip requires TCK to be low while in WFI state to reduce power. By default when the JTAG connector is removed TCK is pulled high so an external link must be placed on the JTAG connector when making power measurements without the JTAG connected.

Link pins 9 and 10 on J18 to pull TCK low, also link 16 and 18 for JTAG detect:

```
  ---
 |+ +|
 |+ +|
 |+ +|
 |+ +|
 |O O| <- link pins 9 and 10 with 2 pin jumper
 |+ +|
 |+ +|
 |+ +|
 |+ O| <- link pins 18 and 20 with 2 pin jumper
 |+ O|
  ---
```

```
MPCore Test chip power measurement
----------------------------------
MPCore Test chip power measurement
```

4.11 MPCore Test Chip power measurement

The CT11MPCore tile allows power measurement of the 1.2V core supply (Vddcore) using software through a 12bit ADC. An example program for reading the core power is provided in the software directory (MPCpower.axf). Alternatively a series resistor on the tile allows direct measurement of the core power using a digital multi meter - DVM. Connect the DVM to J9, the core power can be calculated from:

Power = (V measured / 0.025R) * 1.2V

Typically a single core running MPCpower.axf will take 355mW (7.4mV on DVM). The software power reading has an initial offset error of +/-20mW max with a gain error of 0.5% max.

4.12 CT11MPCore status LEDs

The four status LEDs D2,3,4,5 indicate the status of the PLL lock signals and serial interface as follows. The "~" means that the signal has been inverted.

Table 4-7 Status LEDs

LED	D2	D3	D4	D5
Signal	Serial link locked	~ISP1_nLock	~ISP0_nLock	MPCore PLOCK
Status after reset	1Hz flashing	ON	ON	ON

4.13 EB CT11MPCore specific registers

The EB register base is 0x10000000. All registers must be unlocked by writing 0xA05F to base+0x20 first.

The SYS_PLD_INIT register at base+0x7C loads the CT11MPCore PLL register ONLY after the nPBRESET button (S2 on the EB) is pressed.

SYS_PLD_INIT	PB[3:0]:PA[3:0]:N[3:0]:M[3:0]:PBSTBY:STBY:PLLEN:000000000000:CLKOUTDIV[3:0]
Direction	`WWWW:WWWW:WWWW:RRRR:R:R:R:RRRRRRRRR:WWWW`
Default	`0010:0010:0110:0001:0:0:0:000000000:0111`

The SYS_VOLTAGE0 register at base+0xA0 sets the VDDCORE voltage and reads the VDDCORE voltage.

SYS_VOLTAGE0	ADC_DATA[11:0]:DAC_DATA[7:0]
Direction	`RRRRRRRRRRRR:WWWWWWWW`
Default	`000000000000:10000000`

The SYS_VOLTAGE1 register at base+0xA4 sets the AVDD voltage and reads the AVDD voltage.

SYS_VOLTAGE1	ADC_DATB[11:0]:DAC_DATB[7:0]
Direction	`RRRRRRRRRRRR:WWWWWWWW`
Default	`000000000000:10000000`

The SYS_VOLTAGE2 register at base+0xA8 reads the VDDCORE current.

SYS_VOLTAGE2	ADC_DATC[11:0]:DAC_DATC[7:0] (not used)
Direction	`RRRRRRRRRRRR:WWWWWWWW`
Default	`000000000000:00000000`

The SYS_VOLTAGE3 register at base+0xAC reads the AVDD current.

SYS_VOLTAGE3	ADC_DATD[11:0]:DAC_DATD[7:0] (not used)
Direction	`RRRRRRRRRRRR:WWWWWWWW`
Default	`000000000000:00000000`

The SYS_PLD_CTRL1 register at base+0x74 connects to the following serial registers.

SYS_PLD_CTRL1	00000:MASTNUM:L2BYPASS:INTMODE[2:0]:DGBMUX[11:0]:nCPURESET[3:0]:CFGEND[1:0]:VINITHI[3:0]
Direction	`RRRRR:W:W:WWW:WWWWWWWWWWWW:WWWW:WW:WWWW`
Default	`00000:1:0:000:000000000000:1111:00:0000`

The SYS_PLD_CTRL2 register at base+0x78 connects to the following serial registers.

SYS_PLD_CTRL2	00000000:COMMRX[3:0]:COMMTX[3:0]:STANDBYWFI[3:0]:SMPnAMP[3:0]:RESETREQ[3:0]:PLDVER[3:0]

Direction	RRRRRRRR:RRRR:RRRR:RRRR:RRRR:RRRR:RRRR
Default	00000000:XXXX:XXXX:XXXX:XXXX:XXXX:XXXX

4.14 Register Changes for build C8

The SYS_RESETCTL register at base+0x40, adds a feature where unlocking the register, writing b1 to SWRESET will cause a software reset (this is the same as if the user pressed the PBRESET button)

SYS_RESETCTL	SWRESET:000
Direction	W:RRR
Default	0000

5 Programmer's Model

5.1 CT11MPCore boot up operation overview

This section is intended as a summary of the boot operation, please refer to the MPCore TRM for more information on the use of CP15 registers and exact operation.

All four cores in CT11MPCore reset at the same time, each core then determines its CPU ID via CP15 registers. The CPU with ID zero begins to execute code, while the remaining cores enter a loop, waiting for a software interrupt (WFI).

To generate a software interrupt to a core, the associated GIC for that core needed to be enabled and a software interrupt enabled. Once a core leaves the WFI loop, it reads the address in the SysFlags register in the System register block on EB (0x1000030), and starts to execute code from this address. A CPU can determine which CPU it is by simply reading the CPU ID via the CP15 registers.

5.2 EB Memory Map

The CT11MPCore on EB example design provides a software interface with the majority of features required for a system. Refer to the EB user guide for more information

Table 5-1 Memory map

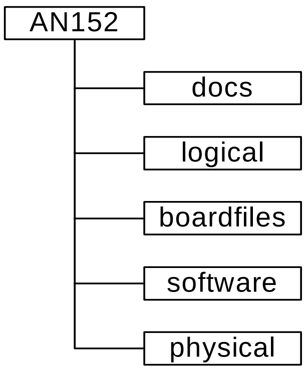
	Memory range		**Bus type**	**Memory region size**
Peripheral	**Lower limit**	**Upper limit**		
Dynamic Memory	0x00000000	0x0FFFFFFF	AXI	256M
System Registers	0x10000000	0x10000FFF	APB	4K
System Controller (SP810)	0x10001000	0x10001FFF	APB	4K
I2C control	0x10002000	0x10002FFF	APB	4K
Reserved	0x10003000	0x10003FFF	APB	4K
AACI	0x10004000	0x10004FFF	APB	4K
MCI0	0x10005000	0x10005FFF	APB	4K
KMI0	0x10006000	0x10006FFF	APB	4K
KMI1	0x10007000	0x10007FFF	APB	4K
Character LCD	0x10008000	0x10008FFF	APB	4K
UART0	0x10009000	0x10009FFF	APB	4K
UART1	0x1000A000	0x1000AFFF	APB	4K
UART2	0x1000B000	0x1000BFFF	APB	4K
UART3	0x1000C000	0x1000CFFF	APB	4K
SSP0	0x1000D000	0x1000DFFF	APB	4K
SCI0	0x1000E000	0x1000EFFF	APB	4K
Reserved	*0x1000F000*	*0x1000FFFF*	*APB*	*4K*
Watchdog	0x10010000	0x10010FFF	APB	4K
Timer 0&1	0x10011000	0x10011FFF	APB	4K
Timer 2&3	0x10012000	0x10012FFF	APB	4K
GPIO 0	0x10013000	0x10013FFF	APB	4K
GPIO 1	0x10014000	0x10014FFF	APB	4K
GPIO 2 (Misc onboard I/O)	0x10015000	0x10015FFF	APB	4K
Reserved	*0x10016000*	*0x10016FFF*	*APB*	*4K*
RTC	0x10017000	0x10017FFF	APB	4K
DMC configuration	0x10018000	0x10018FFF	APB	4K
PCI configuration	0x10019000	0x10019FFF	AHB	4K
Reserved for future use	*0x1001A000*	*0x1001FFFF*	*APB*	*28K (4K * 6)*

CLCD configuration	0x10020000	0x1002FFFF	AHB	64K
DMAC configuration	0x10030000	0x1003FFFF	AHB	64K
GIC1 (nIRQ IC) Tile Site 1	0x10040000	0x1004FFFF	AHB	64K
GIC2 (nFIQ IC) Tile Site 1	0x10050000	0x1005FFFF	AHB	64K
GIC3 (nIRQ IC) Tile Site 2	0x10060000	0x1006FFFF	AHB	64K
GIC4 (nFIQ IC) Tile Site 2	0x10070000	0x1007FFFF	AHB	64K
SMC configuration	0x10080000	0x1008FFFF	AHB	64K
Reserved for future use	*0x10090000*	*0x100EFFFF*	*AHB*	*448K (64K * 7)*
DAP ROM table	0x100F0000	0x100FFFFF	APB	64K
Reserved	*0x10100000*	*0x17FFFFFF*	*N/A*	*112M*
Reserved	*0x18000000*	*0x1FFFFFFF*	*AHB*	*128M*
Reserved	*0x20000000*	*0x3FFFFFFF*	*N/A*	*512M*
Static Memory Controller	0x40000000	0x5FFFFFFF	*AHB/AXI*	512M
PCI interface	0x60000000	0x6FFFFFFF	*AHB/AXI*	256M
Dynamic Memory (alias)	0x70000000	0x7FFFFFFF	AHB/AXI	256M
Logic Tile Site 2	0x80000000	0xFFFFFFFF	AHB/AXI	2G

6 RTL

All of the APB RTL for this design is provided as verilog except for MMCI. AXI components are supplied as netlists. Example files are provided to allow building the system with Synplicity, Synplify Pro and Xilinx ISE tools.

6.1 Directory structure



The application note has directories. These are:

- docs : Related documents including this document.
- logical : All the verilog RTL required for the design.
- boardfiles : The files required to program the design into ARM development boards.
- physical : Synthesis and place and route (P&R) scripts and builds for target board.
- software : ARM code to run on the AN152 system

6.2 logical

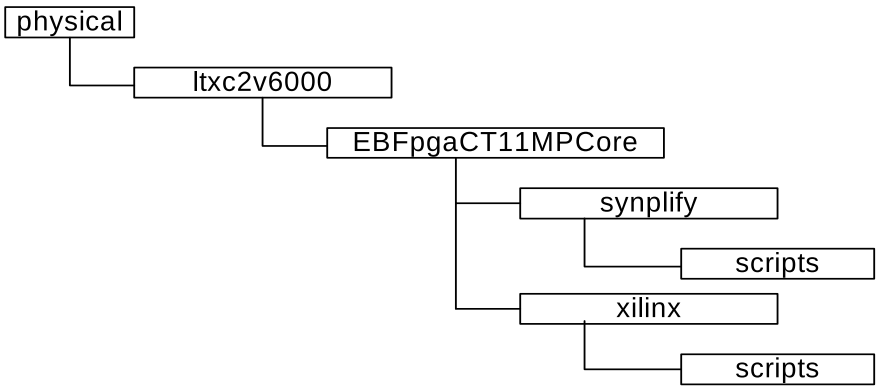
The logical directory contains all the verilog required to build the system. The physical directory contains pre-synthesised components. The function of each block is shown earlier in EB Module functionality

Each PrimeCell or other large IP block has its own directory (for example AXIToAHB).

The top level for this system is in EBFpga.

6.3 physical

The physical directory contains the scripts for the tools used in the build process.



6.4 Building the App Note using Microsoft Windows or Unix

To rebuild the FPGA image for the EB with CT11MPCore you need to change into the EBFpgaCT11MPCore directory and run the make.scr (for UNIX) or make.bat (for windows) script. This will synthesis and Place & Route the design, linking in the required .NGO files at P&R. Read the readme.txt file in the directory for further build options.

make.bat	(default Synthesis and Place & Route)
make.bat all	(Synthesis and Place & Route)
make.bat synth	(Synthesis only)
make.bat par	(build -> map -> par -> bitgen)
make.bat all EBFpgaCTxxx dma	(build with DMA)

To build the EBFpgaCT11MPCore image with DMA

```
cd EBFpgaCT11MPCore
make.bat all EBFpgaCT11MPCore dma
```

Note:
You need to ensure that the Synplify and Xilinx tools executable directories are in the path environment variable for DOS and UNIX.
The Synplify tools do not automatically add the path and the user is required to enter it manually. The Xilinx tools give you the
option at installation time.

6.5 Board file selection

To use the pre-built bit file, use one of the following board file, for axample.

an152_eb_140cd_xc2v6000_ct11mpcore_dma_le_build8_mux_xc2c128_build2_cfg_xc2c128_build3.brd

To use a customer version, use one of the following board files.

an152_eb_0140cd_xc2v6000_CT11MPCore_dma_customer_rebuild.brd