Application Note 138

Using Core Tiles Stand-Alone (with IM-LT3)

Document number: ARM DAI 0138B
Issued: March 2006

ARM

Application Note 138
Using Core Tiles Stand Alone

Release information

The following changes have been made to this Application Note.

Change history

Date	Issue	Change
January 04, 2005	A.01	First release
March 27, 2006	B	Current version for web release (FPGA Build 4)

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

This application note describes an example design implemented in an Integrator/IM-LT3 to demonstrate the CT as a simple stand-alone system. The design is for an IM-LT3 which accompanies the Core Tile. This system is intended as a starting point for hardware developers to build a simple AHB system around a Core Tile.

1.2 CT and Integrator/IM-LT3 overview

The CT provides a platform for a Test Chip to be connected into a Logic Tile system. The IM-LT3 provides an FPGA in which to position all the custom logic that is required around a core to make a useful system as well as acting as a baseboard. A baseboard of some description is required to provide power and JTAG connectivity.

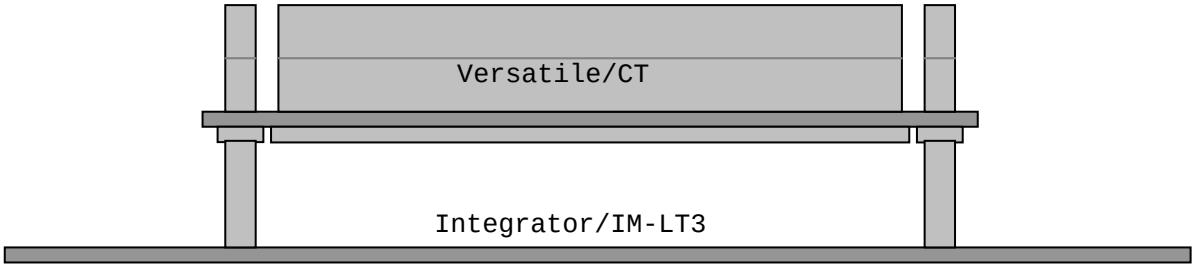


Figure 1-1 Stand-alone hardware stack

2 Getting Started

1. Fit the Core Tile on top of the IM-LT3 and press firmly to ensure good connections.
2. Fit the CONFIG jumper link to IM-LT3 J14.
3. Connect Multi-ICE to the IM-LT1 JTAG connector J13.
4. Check the external supply voltages are +5V and +3.3V, then power down and connect them to the power terminals J9.
5. Power-up the board. The '3V3' LED on should be lit.
6. Run Multi-ICE Server, and press ctrl-A in order to autodetect your boards.
7. Run progcards.exe in directory \AN138\boardfiles and select the option for your Core Tile type.
8. Power down the system.
9. Set the configuration switches to load FPGA image 0. (S4 on the Logic Tile set to all OFF).
10. Remove the CONFIG jumper link and power-cycle the boards.
11. In Multi-ICE Server select File -> Auto-Configure and the auto-detected TAP configuration should appear. Then run your debugger (for instance RVD).

3 Architecture

3.1 System Architecture

Figure 3-1 shows the contents of this example, for use stand alone. The example design is based around an AHB bus. The test chip on the CT is the only master on this bus, and uses it to access the AHB slaves in the design. This design doesn't give access to any other boards.

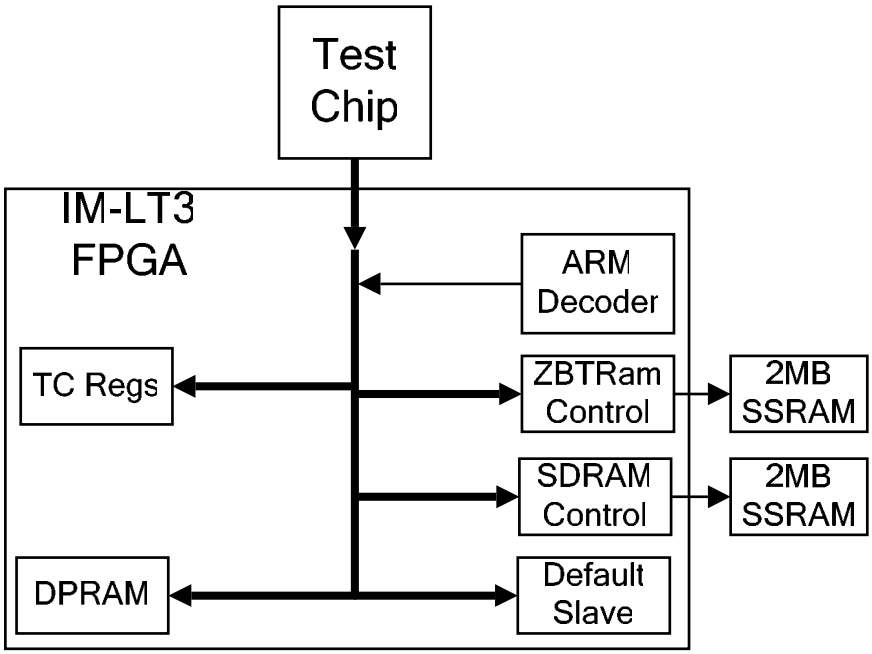


Figure 3-1 Versatile/IT1 example design block diagram

Bus names:

TC: The Test Chip bus is the exposed AHB system from the Test Chip fitted to the Core Tile. The Test Chip is the only master on this bus.

Note that the direction of the arrows indicates the direction of control, ie it points from the Master to the Slave. An AHB bus consists of signals in both directions.

The function of each of these blocks is as follows:

ARM Decoder: This selects which slave the Test Chip (TC bus) is currently accessing.

Default: This slave generates an error response to all otherwise unused addresses so as to avoid undefined bus responses.

DPRAM: This is to provide a pool of memory that both the local Test Chip and external masters can access without impeding each others operation.

GTCRegs: This block contains registers that are of relevance to control of the Test Chip operation such as clock frequencies, power control and measurement, identification etc.

Muxes: These allow multiple slaves to be connected into a single AHB bus.

ZBTRam: This is to provide each Test Chip with 2MB of relatively fast, low latency, private memory.

SDRAM: This block allows the AHB bus to make use of the SDRAM DIMM fitted on the board.

3.2 Clock architecture

The clock architecture is designed to make use of both the Test Chip and IM-LT3 clock architectures.

The User Guides for all the boards used in this configuration explain the clock options they support.

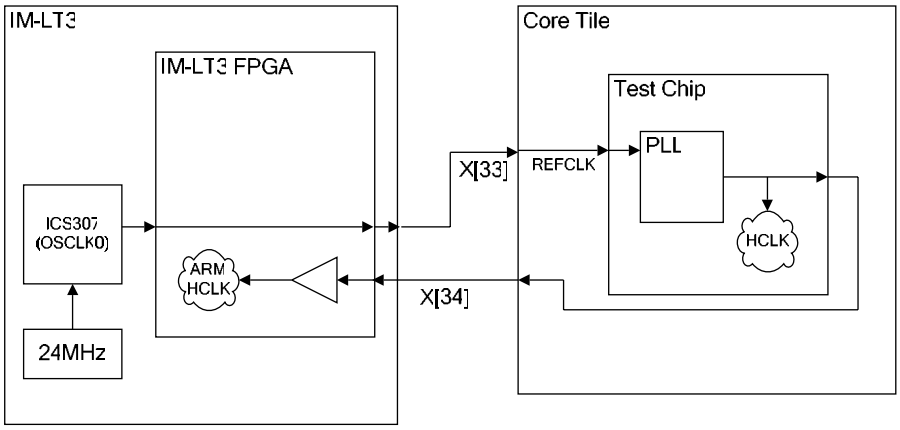


Figure 3-2 Clock architecture – HCLK

This clock architecture has been chosen for this system for a number of reasons.

1. The Core Tile can use either XL[33] or XU[33] as its reference clock. Since the Core Tile will always be connected on top of the Integrator/IM-LT3, its clock multiplexers are configured so that REFCLK is sourced from XL[33]. Therefore the logic tile must drive a reference clock on XU[33].
2. The test chip then generates the HCLK for the ARM system based on its PLL configuration settings. REFCLK should not be used to clock any part of the design, since it may note be synchronous with HCLK.
3. The PLL is set into NORMAL mode (PLLBYPASS = 1'b0). This is so that the CPU on the Core Tiles can be used at its maximum frequency.
4. X[34] is chosen as the exported HCLK from the Test Chip, so that Test Chips with different VDDIO settings can be used. All signals on HDRX are able to support different voltage ranges. To understand how to set up boards for non 3.3V signaling read the appropriate User Guides.

3.2.1 Core Tile Clock configuration

In order to facilitate the above clock architecture the Core Tile Serial Stream must be correctly configured.

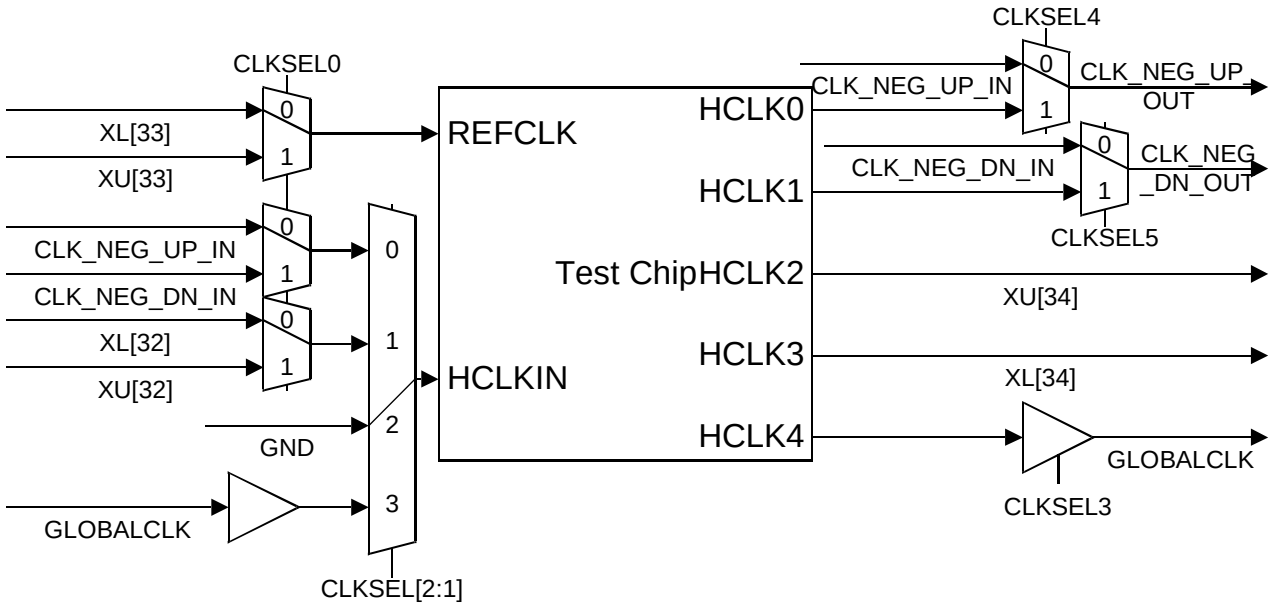


Figure 3-3 Core Tile clock routing

This configuration is chosen to match the overall clock architecture scheme. All the AHB Core Tiles designed for use in this system export their HCLK rather than synchronizing to an input HCLK (HCLKIN). Therefore HCLKIN is configured to always be connected to ground.

The CLK_NEG_UP signal, since it is not used for HCLKIN or HCLK, is configured to be passed through the Core Tile. This makes it available for use in other boards above the Core Tile.

Global Clock is not driven from the Core Tile directly, because there would be a delay between the Global Clock signal on Tile Site 1 and Tile Site 2.

3.2.2 Test Chip Clock Configuration

By default the test chip PLL is enabled (PLLBYPASS = 1'b0).

The settings for the PLL are dependent upon the specific Test Chip being used. Refer to the registers verilog file to see what defaults are being applied.

3.3 Reset Architecture

The baseboard has primary responsibility for nSYSPOR. Refer to the IM-LT3 User Guide for how this works.

The IM-LT3 and Core Tile resets are then generated from this signal (nSYSPOR).

All Power On resets are directly driven from nSYSPOR. This signal is completely asynchronous to any system clocks.

The other resets (including HRESETn) are then generated after a few milliseconds. This is to allow all PLLs and clocks to have settled prior to releasing reset. This delay is done with a counter in the FPGA based on the 24MHz reference clock.

Either nSYSPOR or nSRST being actively driven will cause all the other resets to be driven.

3.4 Interrupt Architecture

There is a simple interrupt controller embedded into the registers block at 0x10400040. For information on how to operate this block refer to section 4.

Some test chips also include their own Vectored Interrupt Controller inside the Test Chip. The following Test Chips presently have their own VIC:-

ARM926EJ-S UMC 0.18um LF712

ARM926EJ-S TSMC 0.18um LF711

This VIC allows the Test Chip to handle interrupts from embedded peripherals much faster (eg the DMA Controller). In order to allow interrupts through from outside the Test Chip, source 5 (nIRQ) and 4 (nFIQ) must be passed through this interrupt controller. To do this, do the following:-

Set 0x3FFE000C = 0x00000010. This Sets Source 4 to be routed to nFIQ rather than nIRQ.

Set 0x3FFE0014 = 0xFFFFFFFF. This clears any currently raised interrupts prior to enabling the interrupts.

Set 0x3FFE0010 = 0x00000030. This allows source 4 and source 5 to generate an interrupt to the processor.

4 Software Interface

This section describes the Programmer's view of the example design.

4.1 Memory Map

Peripheral	CPU Address
DPRAM	0x00000000 – 0x0000FFFF
SDRAM	0x00010000 – 0x0FFFFFFF
Default Slave	0x10000000 – 0x103FFFFF
GTC Regs	0x10400000 – 0x104FFFFF
Default Slave	0x10500000 – 0x107FFFFF
ZBT RAM	0x10800000 – 0x108FFFFF
Default Slave	0x10900000 – 0x3FFBFFFF
Test Chip Internals	0x3FFC0000 – 0x3FFFFFFF
Default Slave	0x40000000 – 0xFFFFFFFF

Figure 4-1 CT Stand-Alone example design memory map

4.2 GTC Regs

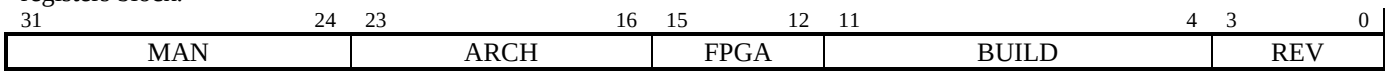
This block is based upon the core module register block that existed for Core Modules. Its function is to control all locally available resources such as Test Chip control signals, clock generation and routing, power control, DAC readings, interrupts etc. Most of these registers are similar to their Core Module (CM_) counterparts.

Register	**Address**	**read/write**	**reset by**
-- generic registers			
CT_BASE	0x10400000		
CT_ID	CT_BASE	r	static
CT_PROC	CT_BASE+0x4	r	static
CT_OSC	CT_BASE+0x8	r/w	POR
CT_CTRL	CT_BASE+0xC	r/w	RESET
CT_STAT	CT_BASE+0x10	r	RESET
CT_LOCK	CT_BASE+0x14	r/w	RESET
CT_LMBUSCNT	CT_BASE+0x18	r	RESET
CT_AUXOSC	CT_BASE+0x1C	r/w	POR
CT_INIT	CT_BASE+0x24	r/w	POR
CT_REFCNT	CT_BASE+0x28	r	RESET
-- flags registers			
CT_FLAGS	CT_BASE_0x30	r	RESET
CT_FLAGSS	CT_BASE_0x30	w	RESET
CT_FLAGSC	CT_BASE_0x34	w	RESET
CT_NVFLAGS	CT_BASE_0x38	r	POR
CT_NVFLAGSS	CT_BASE_0x38	w	POR
CT_NVFLAGSC	CT_BASE_0x3C	w	POR
-- interrupt controller			
CT_IRQ_STATUS	CT_BASE+0x40	r	RESET
CT_IRQ_RAW	CT_BASE+0x44	r	RESET
CT_IRQ_ENABLE	CT_BASE+0x48	r	RESET
CT_IRQ_ENABLES	CT_BASE+0x48	w	RESET
CT_IRQ_ENABLEC	CT_BASE+0x4c	w	RESET
CT_SOFT_INT	CT_BASE+0x50	r	RESET
CT_SOFT_INTS	CT_BASE+0x50	w	RESET
CT_SOFT_INTC	CT_BASE+0x54	w	RESET
CT_FIQ_STATUS	CT_BASE+0x60	r	RESET
CT_FIQ_RAW	CT_BASE+0x64	r	RESET
CT_FIQ_ENABLE	CT_BASE+0x68	r	RESET
CT_FIQ_ENABLES	CT_BASE+0x68	w	RESET
CT_FIQ_ENABLEC	CT_BASE+0x6C	w	RESET
-- cross triggering			
CT_DBGXTRIG	CT_BASE+0x70	r/w	RESET
-- voltage and power management			
CT_VOLTAGE0	CT_BASE+0x80	r/w	POR
CT_VOLTAGE1	CT_BASE+0x84	r/w	POR
CT_VOLTAGE2	CT_BASE+0x88	r/w	POR
CT_VOLTAGE3	CT_BASE+0x8C	r	POR

```
-- CT PLD Control
CT_PLD_CTRL         CT_BASE+0x94        r/w          POR
-- voltage and power management (cont.)
CT_VOLTAGE4         CT_BASE+0xA0        r            POR
CT_VOLTAGE5         CT_BASE+0xA4        r            POR
CT_VOLTAGE6         CT_BASE+0xA8        r            POR
CT_VOLTAGE7         CT_BASE+0xAC        r            POR/RESET
-- Memory Expansion SPD memory
CT_SPDBASE          CT_BASE+0x100       r            POR
```

4.2.1 CT_ID

This register is to allow the user to determine the type of board that this is. It contains information about the build of the fpga, and the class of system that the FPGA image is designed for. It is a read-only register, with values coded into the RTL registers block.




```
CT_ID          Identification
3:0            Revision
11:4           Build value (version) for internal development 0x00-0x99, binary coded decimal
15:12          FPGA type
23:16          Architecture
31:24          Manufacturer
```

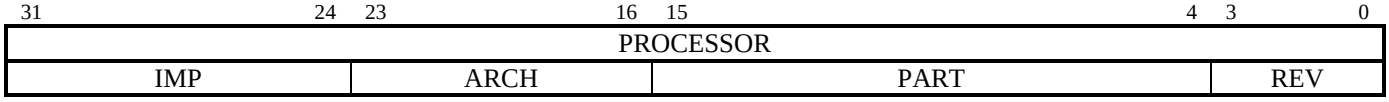
All FPGAs directly associated with Core Tiles should contain identification registers. The allocation of bit fields is described in Table 4-1.

Bits	Name	Description	Allowable values
31:24	MAN	manufacturer	0x41 = ARM, other values may be used for special customer/partner applications. In general use the same encoding as for processor cores.
23:16	ARCH	architecture	Core module encoding as follows: bit [1:0] – processor bus type, 00 = ASB, 01 = 7TDMI, 10 = AHB bit [3:2] – system bus type, 00 = ASB, 10 = AHB bit 4 – SDRAM data width, 0 = 32-bit, 1 = 64-bit bit 5– SDRAM burst length, 0 = 4 b00011010 = AHB processor bus, AHB system bus, 4x64-bit burst SDRAM controller
15:12	FPGA	FPGA type	0x8 = Xilinx Virtex II XC2V6000
11:4	BUILD	build number	Build value or version for ARM identification, binary coded decimal e.g. 0x40 = version 40
3:0	REV	revision number	Release revision 0x0 = Revision A (ASB system) 0x1 = Revision B (AHB system) 0x2 = Revision C (Bluetooth AHB system) 0x3 = Revision D (AHB-lite system, i.e. Integrator/CP) 0x4 = Revision E (926 DevChip, Multi-layer AHB)

Table 4-1: ID Registers

4.2.2 CT_PROC

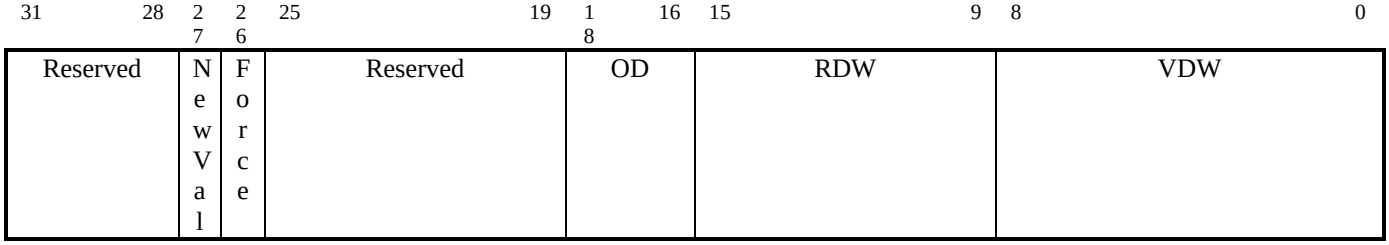
This register exists to allow the user to determine the type of Test Chip currently fitted to this board. It is usually set to zero to indicate that the Test Chip can be directly interrogated in its CP15 registers (CP15 r0).



CT_PROC **Processor Identification**
31:0 If value = 0x00000000, read CP15r0 for details
or
3:0 Revision
15:4 Part
23:16 Architecture
31:24 Implementor

4.2.3 CT_OSC

This register allows the user to set a range of different Test Chip Reference Clock (REFCLK) frequencies. This is then used by the Test Chip PLL to generate the CPU clock and any other synchronous clocks.
REFCLK = 48 * (VDW + 8) / (RDW + 2) * OD.
Note that the value entered into this register will only be applied on either a reset of the core, or if the Force Immediate Update bit is set (bit 26). A 1 in bit 27 indicates that the value of CM_OSC has been changed, but that change has not yet been applied to the clock.
In order to understand the range of permissible values for each of these variables please refer to the ICS307 datasheet.

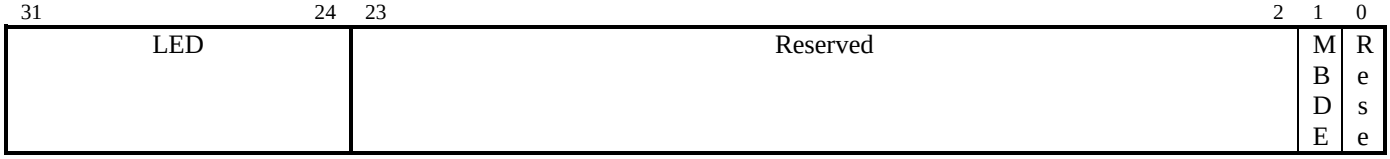


CT_OSC **Oscillator Divisors**
Freq = 48(VDW + 8)/OD(RDW+2)
8:0 CPU clock divider VDW
15:9 CPU clock divider RDW
18:16 CPU clock output divider (OD)
000 divide by 10
001 divide by 2
010 divide by 8
011 divide by 4
100 divide by 5
101 divide by 7
110 divide by 9
111 divide by 6 (default)
25:19 Unused (reserved)
26 Force immediate update
27 New Value pending
31:28 Unused (reserved)

Note: *This is the input clock to the processor and maybe the reference for the PLL, or the core clock in bypass mode. External memory clock is derived from within the processor device. A second oscillator is available but is not programmed via this register. See CT_AUXOSC for details.*

4.2.4 CT_CTRL

This register is to allow control over extra hardware features. In the case of this system it is used to control the four LEDs fitted to a Logic Tile.



		T	rved

CT_CTRL **Control**
0 Unused (reserved)
1 Motherboard detect, 0 = motherboard, 1 = stand alone (read)
23:2 Unused (reserved)
31:24 User LEDs 7:0

4.2.5 CT_STAT

This register shows the Status of the system at present. It shows the current status of the SSRAM configuration, the User Switches fitted to the Logic Tile, and the position of this board in a stack.

31 28	27 24	23 16	15 8	7 0
SW	Reserved	SSRAMSIZE	SILICON ID	POS

CT_STAT **Status**
7:0 Master Tile number in stack
0x00 = Core Tile 0
0x01 = Core Tile 1
0x02 = Core Tile 2
0x03 = Core Tile 3
15:8 Silicon ID (board specific field)
23:16 SSRAM memory size
Appending 0x0000 gives memory size:
0x20 => SSRAM=2MB (0x200000)
27:24 Unused (reserved)
31:28 User switches 3:0

4.2.6 CT_LOCK

This register is to prevent accidental writes to important system registers. It locks all of the following registers: CT_OSC, CT_VOLTAGE0, CT_VOLTAGE1, CT_VOLTAGE2, CT_VOLTAGE3, CT_PLD, CT_REFRESH, CT_AUXOSC, CT_INIT. Before these registers can be written to, the key 0xA05F must first have been written to the CT_LOCK register. In order to relock these register simply write any value other than 0xA05F to this register.

31 17	16	15 0
Reserved	LOCKED	LOCKVAL

CT_LOCK **Lock**
15:0 Lock value, write 0xA05F to unlock, any other value locks
16 Lock bit, read 1 = locked, 0 = unlocked

4.2.7 CT_LMBUSCNT

This register allows the user to determine the number of HCLK cycles that have transpired between two accesses to this register. It can be used for basic profiling, or time control on a Core Tile System.

31 0
LMBUSCNT

CT_LMBUSCNT **Local Memory Bus Cycle Counter**
31:0 32-bit count value, resets at 0 and counts up

4.2.8 CT_AUXOSC

This register allows the user to set a range of different auxiliary clock frequencies. This clock is unused in the default example design, but could be added by the user in future designs. Also the PLL control for the Test Chip can be set in this register when the Test Chip makes use of the ARM_PLL* signals. To determine whether your specific Test Chip does, refer to the Reference Manual accompanying your Test Chip.

REFCLK = 48 * (VDW + 8) / (RDW + 2) * OD.
In order to understand the range of permissible values for each of these variables please refer to the ICS307 datasheet.

31 28	27 24	23 22	21 20	19	18 16	15 9	8 0
PLLOUTDIV	PLLREFDIV	Reserved	PLLCTRL	Reserved	OD	RDW	VDW

CT_AUXOSC **Auxiliary Oscillator Divisors**

```
                Freq = 48(VDW + 8)/OD(RDW+2)
                defaults give Freq = 48x263/(6x65) = 32.37MHz
8:0             Aux clock divider VDW
                011111111   255 (default)
15:9            Aux clock divider RDW
                0111111     63 (default)
18:16           Aux clock output divider (OD)
                000    divide by 10
                001    divide by 2
                010    divide by 8
                011    divide by 4
                100    divide by 5
                101    divide by 7
                110    divide by 9
                111    divide by 6 (default)
19              Unused (reserved)
21:20           PLLCTRL[1:0]
23:22           Unused (reserved)
27:24           PLLREFDIV[3:0]
31:28           PLLOUTDIV[3:0]
```

Note: *This clock is not presently used for anything in this system. This infrastructure is added only to allow an additional clock domain item to be easily added. A second oscillator is available but is not programmed via this register. See CT_OSC for details.*

4.2.9 CT_INIT

This register is for initialization of the Test Chip. It controls the configuration of the Test Chip at reset. In order for these values to take effect a soft reset must be applied after changes have been implemented.

31 30	29 24	23 17	16	15 8	7	6 4	3	2	1	0
Reserved	USERIN	Reserved	INITRAM	PLLFBDIV	Reserved	HCLKDIV	Reserved	VINITHI	PLLBYPASS	

CT_INIT **Test Chip Initialisation**

```
0               PLLBYPASS, 0 = off, 1=on (default), takes effect only after reset
1               PLLBYPASS (value read from pin after reset) – read only
2               VINITHI, 0 = vectors at 0, 1 = vectors at 0xffff0000
3               Reserved
6:4             HCLK divider
7               Reserved
15:8            PLLFBDIV[7:0]
16              Internal RAM enable, 0 = disabled (default), 1 = enabled
23:17           Reserved
29:24           USERIN[5:0]
31:30           Reserved
```

4.2.10 CT_REFCNT

This register allows the user to determine the number of 24MHz clock cycles that have transpired between two accesses to this register. It can be used for basic profiling, or time control on a Core Tile System.

31	0
REFCNT	

CT_REFCNT **Reference Clock (24MHz) Counter**
31:0 32-bit count value, resets at 0 and counts up

4.2.11 CT_FLAGS

These flags can be used by software. Note that the NVFlags will not be cleared on a reset, so can be used for retaining data during a reset, such as if for example software is trying to change some startup settings.

CT_FLAGS **Flags**
31:0 1 = flag set, 0 = flag clear

CT_FLAGSS **Flags Set**
31:0 1 = set flag, 0 = no change

CT_FLAGSC **Flags Clear**
31:0 1 = clear flag, 0 = no change

CT_NVFLAGS **Non-volatile Flags**
31:0 1 = flag set, 0 = flag clear

CT_NVFLAGSS **Non-volatile Flags Set**
31:0 1 = set flag, 0 = no change

CT_NVFLAGSC **Non-volatile Flags Clear**
31:0 1 = clear flag, 0 = no change

4.2.12 CT_IRQ

The core module implements a 6-bit IRQ interrupt controller and a 6-bit FIQ interrupt controller. Refer to the table of bit assignments below for details.

These registers can be used to determine which interrupts are currently active, and set whether they cause a normal or fast interrupt type. They can also be used to mask off certain sorts of interrupt, such as when the user wishes to only poll for certain interrupt types rather than take a direct interrupt handler.

31	3	2	1	0
Reserved		COMMTX	COMMRX	SOFTINT

IRQ and FIQ bit assignment for status, raw status and enable registers

0 SOFTINT Local Soft interrupt (see note about generating soft interrupts)
1 COMMRX Communications channel receive interrupt source
2 COMMTX Communications channel transmit interrupt source

CT_IRQ_STATUS **IRQ Status**
5:0 Interrupt source is enabled and asserted

CT_IRQ_RAW **Raw IRQ Status**
5:0 Interrupt source is asserted

CT_IRQ_ENABLE **IRQ Enable Register**
5:0 1 = Interrupt IRQ is enabled, 0 = disabled

CT_IRQ_ENABLES **IRQ Enable Set**
5:0 1 = Interrupt IRQ Enable set, 0 = no change

CT_IRQ_ENABLEC	**IRQ Enable Clear**
5:0	1 = Interrupt IRQ Enable clear, 0 = no change
CT_SOFT_INT	**Local Soft Interrupt Register**
0	1 = Local Soft Interrupt is set, 0 = Local Soft Interrupt is clear
CT_SOFT_INTS	**Local Soft Interrupt Set**
0	1 = Local Soft Interrupt set, 0 = no change
CT_SOFT_INTC	**Local Soft Interrupt Clear**
0	1 = Local Soft Interrupt clear, 0 = no change
CT_FIQ_STATUS	**FIQ Status**
5:0	Interrupt source is enabled and asserted
CT_FIQ_RAW	**Raw FIQ Status (same as raw IRQ status)**
5:0	Interrupt source is asserted
CT_FIQ_ENABLE	**FIQ Enable**
5:0	1 = Interrupt FIQ is enabled, 0 = disabled
CT_FIQ_ENABLES	**FIQ Enable Set**
5:0	1 = Interrupt FIQ Enable set, 0 = no change
CT_FIQ_ENABLEC	**FIQ Enable Clear**
5:0	1 = Interrupt FIQ Enable clear, 0 = no change

4.2.13 CT_DBGXTRIG

This register is not functional in the Stand-alone system, as it is intended as a single master system only.

31	0
Reserved	

4.2.14 CT_VOLTAGE

This register is to allow use of the DACs and ADCs fitted to the Core Tile.

31 20	19 8	7 0
ADC	ADC	DAC

Note: *This functionality allows voltage control and read back over a number of DAC and ADC channels. Depending on the Core Tile build not all three Voltage domains may be present, and the scaling of the voltage domains varies between builds. It is recommended that when scaling the voltage, the algorithm to set the new voltage works by incrementing the DAC and using to ADC to determine the effect this is having. See the Core Tile User Guide for a full explanation of this circuitry, and how to make use of these values.*

CT_VOLTAGE_CTL0 **Voltage Control 0**
7:0 Core voltage A DAC value
19:8 VDDCORE1 voltage ADC value (12 bits)
31:20 VDDCORE1 DIFF

CT_VOLTAGE_CTL1 **Voltage Control 1**
7:0 Core voltage B DAC value
19:8 VDDCORE2 voltage ADC value (12 bits)
31:20 VDDCORE2 DIFF

CT_VOLTAGE_CTL2 **Voltage Control 2**
7:0 Core voltage C DAC value
19:8 VDDCORE3 voltage ADC value (12 bits)
31:20 VDDCORE3 DIFF

CT_VOLTAGE_CTL3 **Voltage Control 3**
0 Core voltage A enable (nSHDN) (RESET)
1 Core voltage B enable (nSHDN) (RESET)
2 Core voltage C enable (nSHDN) (RESET)
7:3 Reserved
19:8 VDDCORE4 voltage ADC value (12 bits)
31:20 VDDCORE4 DIFF

CT_VOLTAGE_CTL4 **Voltage Control 4**
7:0 Reserved
19:8 VDDCORE5 voltage ADC value (12 bits)
31:20 VDDCORE5 DIFF

CT_VOLTAGE_CTL5 **Voltage Control 5**
7:0 Reserved
19:8 VDDCORE6 voltage ADC value (12 bits)
31:20 VDDCORE6 DIFF

CT_VOLTAGE_CTL6 **Voltage Control 6**
7:0 Reserved
19:8 VDDPLL1 voltage ADC value (12 bits)
31:20 VDDPLL2 voltage ADC value (12 bits)

CT_VOLTAGE_CTL7 **Voltage Control 7**
7:0 Reserved
19:8 VDDIO voltage ADC value (12 bits)
31:20 Test Point voltage ADC value (12 bits)

4.2.15 CT_PLD_CTRL

This register allows the user to communicate with the configurable options on the Core Tile PLD. These values are communicated across the Serial Interface to the Core Tile PLD.

31 28	27 25	24	23 13	12	11 10	9 4	3 0
PLD ID	Reserved	SDRAM	Reserved	PGOOD	Reserved	CLKSEL	ZCTL

CT_PLD_CTRL **Core Tile PLD Control/Status**

```
0                   ZCTL0: Set high to break Z[31:0] bus
1                   ZCTL1: Set high to break Z[63:32] bus
2                   ZCTL2: Set high to break Z[95:64] bus
3                   ZCTL3: Set high to break Z[127:96] bus
4                   CLKSEL0: Set low for tile below/high for tile above
6:5                 CLKSEL[2:1]: HCLK gets CLK_NEG, X_HCLK, GND, CLK_GLOBAL
7                   CLKSEL3: Set low to drive CLK_GLOBAL
8                   CLKSEL4: NEG_UP_OUT assigned HCLK,NEG_UP_IN
9                   CLKSEL5: NEG_DN_OUT assigned HCLK,NEG_DN_IN
12                  PGOOD. Active high signal indicating that power supply on Core Tile
                    board is working OK.
24                  SDRAM mode. 1=MemExp. 0=CP Legacy
31:28               PLD_ID. Determines the current build/type of the PLD image.
```

4.2.16 CT_SPDBASE

This is a copy of the contents of the SPD or other EEPROM associated with the SDRAM fitted to this system.

CT_SPDBASE	**SPD Base**
Byte 2	Memory type
Byte 3	Number of Row addresses
Byte 4	Number of Column addresses
Byte 5	Number of Banks
Byte 18	CAS latencies supported
Byte 31	Module bank density (MB divided by 4)
Byte 63	Checksum
Byte 64-71	Manufacturer
Byte 73-90	Module Part Number

Some Tiles may have a memory expansion card fitted. These Memory Expansion Cards are fitted with a serial presence detect (SPD) EPROM, containing information about the manufacturer and memory type.

On power up, the module FPGA reads this SPD EPROM and stores the data in 256 bytes of RAM which can be randomly accessed at addresses above 0x10400100. A bit is set in the CT_SDRAM register to indicate when the EPROM has been read and information stored in RAM. If there is no DIMM fitted then this process reads invalid data, but the bit will still be set indicating completion. To check for valid SPD data it is necessary to checksum the memory. The algorithm is:

1. add up all bytes 0 to 62
2. logical AND the result with 0xff
3. compare the result with byte 63
4. if the two values match then the SPD data is valid

4.3 ZBT RAM

The ZBT RAM slave block is 2 MB of zero wait memory that is private to the local CPU. The memory is wrapped across the entire address range 0x10800000 – 0x10FFFFFF.

4.4 DPRAM

The DPRAM slave block contains 64kB of block ram. It is full dual port memory so transactions can simultaneously take place in both the local CPU clock domain and an additional external master clock domain (2nd port unused in this design).

The DPRAM appears at address 0x0 for the local CPUs, which is where the boot code for each processor is located. Refer to the section on booting for more information on the proposed boot mechanism.

4.5 SDRAM

The SDRAM appears above the DPRAM at 0x00010000. The SDRAM is configured from the SPD at power up, so does not require any additional software configuration.

The SDRAM is aliased across the range 0x00010000 – 0x0FFFFFFF in blocks dependent upon the size of the DIMM fitted. DIMM sizes between 16MB and 256MB are supported by this controller.

4.6 Test Chip Internal Bus

Some test chips have some embedded peripherals and/or memory. The area 0x3FFC0000 – 0x3FFFFFFF is reserved for this purpose.

In the case of the ARM926EJ-S (LF711 and LF712) processors there are the following blocks:-

0x3FFC0000 – 0x3FFCFFFF Embedded GPIO(PL061 PrimeCell)/EPHEM IO controller.

0x3FFD0000 – 0x3FFDFFFF Embedded DMA controller (PL081 PrimeCell)

0x3FFE0000 – 0x3FFEFFFF Embedded VIC (PL190 PrimeCell)

0x3FFF0000 – 0x3FFFFFFF On-chip RAM.

The other Test Chips do not make use of this area at present.

For information on how to use these blocks refer to the appropriate PrimeCell manual.

5 Boot Operation

Each Core Tile System has pre-initialized memory at address 0x0. This is inside the DPRAM as part of the FPGA build, as all Block RAM inside the FPGA can be initialized to a non-zero value. The boot code is thus embedded into the FPGA image. This memory has the following program in it:-

```
0x0    0xEA000006    B 0x20              ; Jumps past Exception Handlers
0x4    0xEAFFFFFE    B 0x4               ; Dummy Undefined instruction handler
0x8    0xEAFFFFFE    B 0x8               ; Dummy Software Interrupt handler
0xC    0xEAFFFFFE    B 0xC               ; Dummy Prefetch Abort handler
0x10   0xEAFFFFFE    B 0x10              ; Dummy Data Abort handler
0x14   0xEAFFFFFE    B 0x14              ; Reserved
0x18   0xEAFFFFFE    B 0x18              ; Dummy IRQ handler
0x1C   0xEAFFFFFE    B 0x1C              ; Dummy FIQ handler
0x20   0xE3A00000    MOV r0,#0           ; Set counter to 0
0x24   0xE3A01B40    MOV R1,#0x40, 22    ; Set maximum counter value to 0x10000
0x28   0xE59F2018    LDR R2,0x00000048   ; Set pointer to LED control register 0x1040000C
0x2C   0xE5923000    LDR R3,[R2,#0]      ; Load current contents of LED control register
0x30   0xE2800001    ADD R0,R0,#1        ; Increment counter
0x34   0xE1500001    CMP R0,R1           ; Check if counter is at maximum value
0x38   0x02833540    ADDEQ R3,R3,#0x40, 10 ; If counter at max, add one to LED part register
0x3C   0x05823000    STREQ R3,[R2,#0]    ; If counter at max, write new LED output
0x40   0x03A00000    MOVEQ R0,#0         ; If counter at max, reset counter
0x44   0XE59FF000    LDR PC,0x4C         ; Return to start of loop.
0x48   0x1040000C    DCD 0x1040000C      ; Data value of CT_CTRL register
0x4C   0x00000030    DCD 0x00000030      ; Address value of LDR PC command
0x50   0x00000000    DCD 0x00000000      ; Test Pattern 0
0x54   0x55555555    DCD 0x55555555      ; Test Pattern 1
0x58   0xAAAAAAAA    DCD 0xAAAAAAAA      ; Test Pattern 2
0x5C   0xFFFFFFFF    DCD 0xFFFFFFFF      ; Test Pattern 3
```

This code flashes the LEDs on the corresponding Logic Tile system. This shows the operator that the CPU has been successfully brought out of reset, and has appropriate clocks applied.

The Test Patterns at the end are so the user can read these values if a fault on the data bus is suspected.

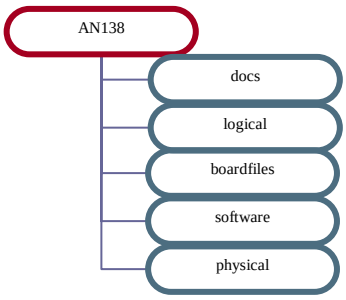
Note that in this code, the exception handlers (Undef, SWI, PF, Abort, D Abort, IRQ, FIQ) will merely trap the exception, by causing the process to sit in a closed loop. This will mean that on connection with a debugger it will be evident that an exception was hit, but in order to make proper use of exception handling these routines must be replaced.

In order to execute more code a debugger is required to load code into the volatile memory regions.

6 RTL

All of the RTL for this design is provided as verilog. 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 five effective sub areas. 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.
- Software : ARM code to run on the AN125 system
- Physical : Synthesis and P&R scripts and builds for target boards.

6.2 Logical

The logical directory contains all the verilog required to build the system. The function of each block is shown earlier in Section 3.1.

Each Primecell or other large IP block has its own directory (eg Ahb2AhbSync).

The top level for the system including a CTxxx and an Integrator/IM-LT3 is in IMLT3_CTxxx.

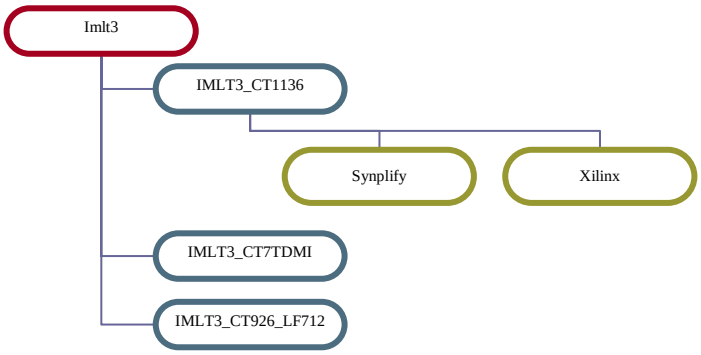
The required registers, and their default settings are in the CTxxxRegs directories. Differences in hardware such as Test Chips means that often different Register settings are required for a system.

The remaining RTL to complete the system is in the CT_xxx directories. There is a CT_Generic directory containing RTL applicable to the majority of builds, and specific CT_xxx directories for each different PCB in the Core Tile Range.

6.3 Targets

Each potential target has a directory (eg imlt3). Within these directories are all the different builds which exist for this target. In turn each of these has directories for each tool used in the build process.

Synplify Pro and Xilinx ISE are the tools used for building the design for Xilinx FPGA implementation.



7 Example Software

7.1 Voltage Control

The ‘control’ example is intended to be executed directly by the Core Tile. It demonstrates the use of the DAC and ADC parts fitted to the Core Tile. It acts as an easy way to confirm that the system is powered correctly with the correct voltages, and allows the user to implement changes to the programmable supplies.

7.1.1 Source Code

This simple example consists of a header file with a definition of the registers to be used, and a C code file with functions to be executed by the Core Tiles. This code cannot be run on the Development Chip, as the Development Chip does not have registers to read and set the voltage supplies.

7.1.2 Functionality

The main() function provides the user with a menu to select the function to be run.

Read ID parses the CT_ID and CT_STAT registers to determine the type of Core Tile Fitted. This is important as different Core Tiles have different Voltage requirements and specifications, so the user can use this information to determine the expected behaviour of this board.

The Read Voltages function goes through each ADC reading and presents the results to the user in a useful format (converted to Voltage or Current measurements).

The set voltage function allows the user to change the VDDCORE power supplies. Since every different board has different sensitivities to these numbers it is the responsibility of the user to ensure they know what they are doing prior to changing these values (see Core Tile User Guide for more information).

8 Clock frequency settings

8.1 Default frequencies

The default frequencies are set to generate a stable system with the system bus clock as close to the maximum operating frequency as possible, and the CPU set to a safe working frequency that is a multiple of the bus frequency.

For example the following systems have these default frequencies:-

Test Chip	REFCLK	CPUCLK	HCLKI	HCLKE
ARM926EJ-S UMC 0.18um LF712	40 MHz	100 MHz	50 MHz	25 MHz
ARM1136JZF-S TSMC 0.13um	40 MHz	240 MHz	120 MHz	30 MHz
ARM7TDMI EPSON T0464FA70	20 MHz	20 MHz	-	20 MHz

HCLKI is the bus frequency for any on-chip peripherals.

HCLKE is the external bus frequency used in the Logic Tile.

For benchmarking purposes it is simpler to scale your results from these frequencies, rather than attempt changing frequency as there are a number of factors that must be carefully considered prior to changing frequency.

8.2 Changing the startup frequencies

In circumstances where the user needs to change these startup clock frequencies then they should rebuild the FPGA image with their preferred startup values. This method will then mean that the user does not have to worry about running code at startup to deal with changing the clocks after power up.

In order to do this the following method is recommended:-

1. Determine the required frequencies that are desired for CPUCLK, HCLKI and HCLKE. HCLKE must be between 25 and 35 MHz with the standard clock architecture provided (required to meet the timing constraints and DCM requirements of the FPGA design). Refer to the release notes for the maximum CPUCLK settings for your CPU. The HCLKE frequency must be an integer factor of the CPUCLK frequency.
2. Calculate all the PLL settings that need to be applied to the Test Chip to generate the chosen CPUCLK.

   $F_{CPUCLK} = (F_{REFCLK} / NR) \times (NF / OD)$

   NR = Reference Divider = (PLLREFDIV[3:0] + 1)

   NF = Feedback Divider = (PLLFBDIV[7:0] + 1)

   OD = Output Divider = (PLLOUTDIV[2:0] + 1)

   $137kHz < (F_{REFCLK} / NR) < 275 MHz$

   $500kHz < ((F_{REFCLK} / NR) \times NF) < 275 MHz$
3. Change the parameters declared within the Regs file for your Test Chip. These are:-

   CM_PLLREFDIV_VAL

   CM_PLLFBDIV_VAL

   CM_PLLOUTDIV_VAL
4. If a different value of F_{REFCLK} is required to get the exact CPU frequency desired then the CM_OSC values need to be changed accordingly. There is a calculator on the internet at www.icst.com. Otherwise refer to section 4.3.3 for how to calculate all the required fields. Change the parameters declared within the Regs file for your Test Chip. These are:-

CM_CPU_VDW_VAL

CM_CPU_RDW_VAL

CM_CPU_OD_VAL

5. Calculate the CPUCLK:HCLKI settings.

   For ARM9 based systems this can be either 1:1 or 2:1. To set 1:1 set CM_USERIN_VAL[5] to 1'b0, and for 2:1 set CM_USERIN_VAL[5] to 1'b1. Always set CM_USERIN_VAL[3] to 1'b1 to enable HCLKI.

   For ARM11 based systems this can be any integer value from 1:1 to 64:1. Set the ratio value -1 to CM_HCLKI_VAL (ie tie to 6'b000000 for 1:1 or 6'b000001 for 2:1).

6. Calculate the HCLKI:HCLKE settings.

   For ARM9 based systems this can be any integer value from 1:1 to 16:1.

   Ratio = {USERIN[1:0], HCLKDIV[2:0]} + 1

   For ARM11 based systems this can be any integer value from 1:1 to 64:1. Set the ratio value -1 to CM_HCLKE_VAL (ie tie to 6'b000000 for 1:1 or 6'b000001 for 2:1). This ratio is relative to the CPUCLK rather than HCLKI, but HCLKE must be equal to or slower than HCLKI.

   Change the following parameters following these calculations:-

   CM_USERIN_VAL[1:0]

   CM_HCLKDIV_VAL

   CM_HCLKE_VAL

7. Remember to change the CM_BUILD and CM_MANUFACTURER parameters to your own unique identification numbers so that you can tell this FPGA build apart from other builds. Then rebuild the FPGA with your new Regs file, and program into the Flash.

8.3 System bus frequency testing

It is sometimes desirable to temporarily change the frequency rather than doing a permanent FPGA build. For example if the user wants to test a frequency prior to building a new FPGA image.

In most cases the user will only care about the system bus frequency. In order to accommodate this CPU frequencies have been chosen that can be moved a little faster or slower. Therefore this change can be made to a live system by slowly changing the REFCLK frequency only.

In order to do this, do the following:-

1. Unlock the register bank by writing 0xA05F to 0x10400014.
2. Read register 0x10400008.
3. If decreasing frequency subtract 1 from the value in the last 9 bits. If increasing frequency add 1 to the value in the last 9 bits. Set bit 26 to 1'b1 (force update bit).
4. Write this value back to 0x10400008.
5. Calculate your new REFCLK setting (see section 4.3.3). This can then be used to calculate your new CPUCLK, HCLKI, and HCLKE frequencies (they all scale linearly with REFCLK).
6. Repeat steps 2-5 until you reach the desired operating range. Allow time for the clocks to settle to their new frequency between each loop.

8.4 Full frequency testing

If the user wishes to change the CPU frequency by a large amount, or wishes to change the ratios of the different frequencies then it is necessary to take some additional steps in order to allow the PLL/DCMs in the system to correctly lock to their new frequencies.

Since the ARM11 class Test Chips only read in the HCLKI and HCLKE values as part of a Power On Reset, it is not possible to change these values through software.

In order to do this, follow all the calculations in 8.2. However rather than changing the listed parameters above, do the following:-

1. Unlock the registers bank by writing 0xA05F to 0x10400008.
2. Read the value in CM_AUXOSC register (0x1040001C).
3. Modify bits [31:28] to the new PLLOUTDIV value.
4. Modify bits [27:24] to the new PLLREFDIV value.
5. Write this value back to CM_AUXOSC.
6. Read the value in CM_INIT register (0x10400024).
7. Modify bits [29:24] to the new USERIN value.
8. Modify bits [15:8] to the new PLLFBDIV value.
9. Modify bits [6:4] to the new HCLKDIV value.
10. Write this value back to CM_INIT.
11. Read the value in CM_OSC (0x10400008).
12. Modify bits [18:16] to the new OD value.
13. Modify bits [15:9] to the new RDW value.
14. Modify bits [8:0] to the new VDW value.
15. Write this value back to CM_OSC.
16. Request a reset by pulling nSRST low (the ICE tool can be asked to do this). This will now reset the system, with the Core Tile restarting in its new configuration. If connected with a debugger, it is probable that a breakpoint will have been left at 0x0, and so it will be necessary to use JTAG to instruct the processor to proceed with execution of its boot code.

9 Variations

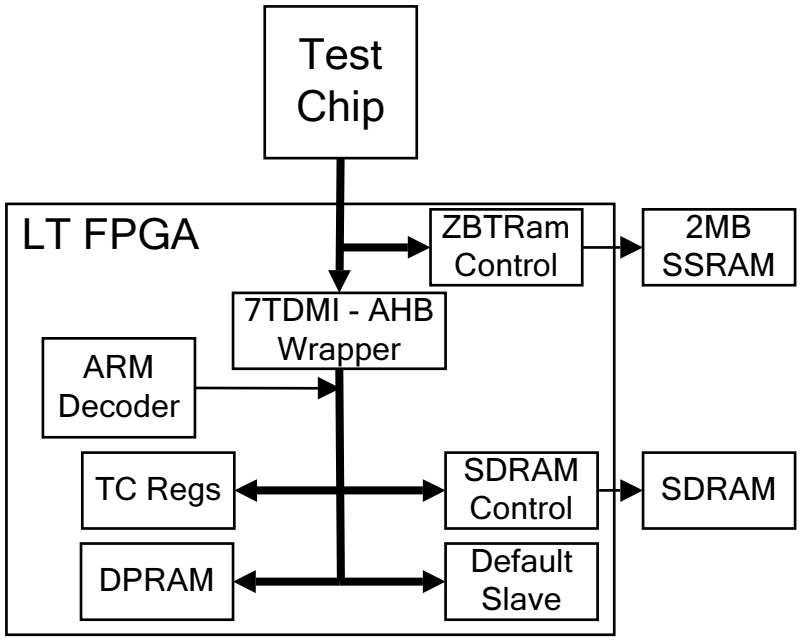
9.1 Overview

This Application Note is primarily aimed at using an AHB Test Chip. When used in conjunction with a different bus type, there are a number of differences to the design which the user should be aware of.

9.2 CT-7TDMI

9.2.1 Architecture

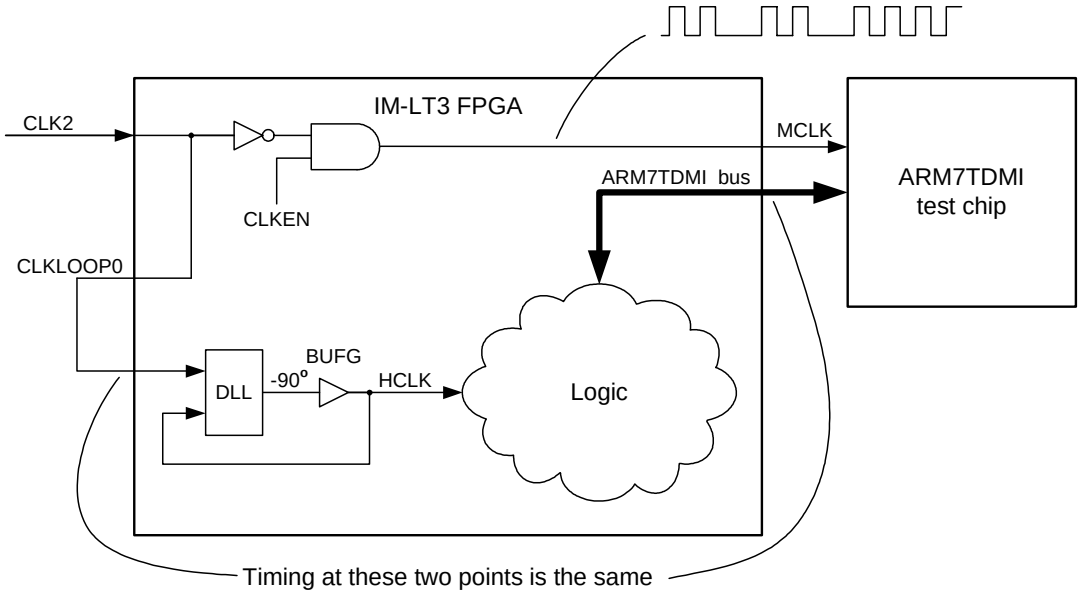
In order to support the fact that a different bus type is attempting to interact with an otherwise AHB system, a wrapper has been inserted between the Test Chip and the remaining system. The SSRAM control must also be moved into the 7TDMI domain, as the SSRAM data bus is shared with the Test Chip data bus on an IM-LT3 system.



The clock architecture is slightly modified to allow for the fact that the ARM7 Test Chip is not inside the same chip as the wrapper (as was intended). Instead of gating the local copy of the AHB clock to use as the 7TDMI clock, instead an early version is gated to compensate for the delay from the FPGA to the Test Chip.

9.2.2 Clock structure

The ARM7TDMI test chip requires a bus synchronous clock, rather than providing one. The ARM7TDMI is also designed to operate with a gated clock to maximize power efficiency. For this reason the ARM7TDMI uses a different clock scheme. A diagram showing the ARM7TDMI clock scheme is shown below.



9.2.3 Software Interface

The memory map for the 7TDMI system is the same as for a native AHB Test Chip. The only difference will be in the timing of the 7TDMI system, due to the insertion of the wrapper on the Test Chip bus.

9.2.4 GTC Regs

There are a few minor changes to the contents of some of the Registers. These changes are partially to reflect that the system is different, and partially due to the fact that the 7TDMI Test Chips are fitted to a different PCB which has some subtlety different configuration options.

- CT_ID : The architecture field will now report the processor bus as being a 7TDMI.
- CT_PROC : Since the 7TDMI does not feature a co-processor with identification information, this register will contain information on the type of processor fitted.
- CT_STAT : Since this is a different architecture chip, the Silicon ID values will overlap with the native AHB chips.
- CT_AUXOSC : There is no PLL fitted to 7TDMI class test chips. The PLL control values will have no effect.
- CT_INIT : As with CT_AUXOSC the PLL control values will have no effect. The USERIN values also have no effect.
- CT_VOLTAGE : The 7TDMI class test chip has fewer voltage domains than the native AHB test chips. For this reason CT_VOLTAGE_CTL4,5,6,7 are not valid. The other CT_VOLTAGE_CTL registers have the following effect:

```
CT_VOLTAGE_CTL0      Voltage Control 0
7:0                  Core voltage A DAC value
19:8                 VDDCOREA voltage ADC value (12 bits)
31:20                VDDCOREA DIFF

CT_VOLTAGE_CTL1      Voltage Control 1
7:0                  Core voltage B DAC value
19:8                 VDDCOREB voltage ADC value (12 bits)
31:20                VDDCOREB DIFF

CT_VOLTAGE_CTL2      Voltage Control 2
7:0                  Reserved
19:8                 PISMO 1.8V supply voltage ADC value (12 bits). *Not scaled by 2!
31:20                VDDIO voltage ADC value (12 bits)

CT_VOLTAGE_CTL3      Voltage Control 3
```

```
7:0                 Reserved
19:8                TP2 voltage ADC value (12 bits)
31:20               TP1 voltage ADC value (12 bits)
```

- CT_PLD_CTRL : The 7TDMI system does not sense whether all the core power nets are good (PGOOD). There are also only 5 CLKSEL controls rather than 6.

```
4                   CLKSEL0: Set low for tile below/high for tile above
6:5                 CLKSEL[2:1]: HCLK gets CLK_NEG, X_MCLK, GND, CLK_GLOBAL
7                   CLKSEL3: NEG_UP_OUT assigned HCLK,NEG_UP_IN
8                   CLKSEL4: NEG_DN_OUT assigned HCLK,NEG_DN_IN
9                   CLKSEL5: Route X_MCLK and X_ECLK across Core Tile
```

9.2.5 Example Software

The example software has been coded to support both native AHB test chips and 7TDMI class test chips. By reading the processor bus type where applicable the software can operate across different systems. Since all the examples are compiled for the lowest common code architecture (v4 is the lowest expected with this hardware), the code will run on all test chips.