Integrator Performance FAQ v1.1

Table of contents

1. Introduction

- ***Scope of this document***

The information contained in this document applies to systems based on Integrator core modules, Integrator/AP and/or Integrator/CP.

It doesn't apply to Integrator Logic Tiles used as an emulation platform. It may not apply either to new generation development boards that use the generic name "Integrator".

- ***Purpose of this document***

This document intends to answer the frequently asked question "How fast is Integrator?"

Integrator is a complex system with several buses, clock domains, memory devices and peripherals, so this question does not have a simple answer. The speed of the system depends on the following factors:

- The clock and bus architecture of the system
- The frequency of the core and the memory bus clocks
- The number of wait states inserted by the memory devices or memory controllers

This document contains detailed information about these three factors in Integrator. This information can be used to:

- Program the Integrator clocks with appropriate frequencies
- Configure your system and select in which area of memory you place your code and data to achieve optimum performance
- Benchmark your code: if you know the speed of memory accesses in Integrator you can calculate approximately how fast your code will run in your final product.

This document is divided in the following sections:

- Integrator Clocks and Bridges: This section explains the different clock domains in Integrator systems and the maximum frequency at which you can clock the core and the memory system
- The Memory Access Timing Spreadsheet: This spreadsheet contains the number of bus cycles needed to access each area of memory of Integrator. The section describes the spreadsheet and how the accesses were measured
- The Integrator Memory System: This section analyzes the information in the Memory Access Timing Spreadsheet and explains the differences between areas of memory on different boards
- Optimizing the Performance of Integrator: This section gives a few hints to improve the overall speed of your design based on the analysis done in the previous sections. These hints can also apply in many cases to the customers' own hardware

- ***Using Integrator as a benchmarking platform?***

At this point it is important to note that Integrator was designed to be a very configurable and expandable product in which you can test your hardware and software: it was not intended to be a benchmarking platform.

This means that in general you will find that Integrator is slower than the hardware of your final product. This happens because:

- Due to its modular design there are several bridges in the system, which introduce long delays when accessing certain areas of the memory map
- Most of the logic in Integrator boards is implemented in FPGAs, which limit the maximum frequency of the bus clock
- Core modules have test chips from different manufacturers and silicon processes. In many cases the maximum speed of these chips is slower than the maximum speed you will get from your real silicon

Your system engineers should ensure that the design of your board or ASIC achieves the required performance. This performance will normally be better than that of a generic system like Integrator.

- ***Supporting files***

The zip file *Integrator_Performance_FAQ_v1_1_Supporting_Files.zip*, which can be downloaded from www.arm.com separately, includes the following supporting files:

- *Access_Generation* and *Range_Generation* folders: Include the software that has been used to generate memory accesses from the test chip. The memory access time for these accesses was measured and the information was summarized in the Memory Access Timing Spreadsheet
- *HDL* folder: Includes modified FPGA images for several core modules. These images take some signals from the test chip pins to the logic analyser connectors, so memory accesses can be measured with a logic analyser
- *Waveforms* folder: Includes waveforms measured with a logic analyser for all accesses to the Integrator memory system

There is more information about the supporting files in section 3. Memory Access Timing Spreadsheet.

2. Integrator Clocks and Bridges

- ***Integrator Clock Domains and System Bus Bridges***

The clock and bus architecture in Integrator is different for different core modules and platforms. The platform board chosen affects the clock architecture in the following way:

Integrator/AP

Systems based around an AP board comprise the AP motherboard and a number of core modules and logic modules or tiles. These systems have typically 3 clock domains:

- Core clock - clocks the ARM core inside the test chip. Memory accesses to the caches or TCMs in the core are done at core clock speed. The core clock is generated by the core module
- Local bus clock - clocks the AMBA bus on the core module and the ARM core bus interface logic. Accesses to resources on the core module are done at local bus clock speed. Amongst these resources are the core module SSRAM and SDRAM DIMM when accessed at its default location (e.g. at address 0x00200000). The local bus clock is generated by the core module
- System bus clock - clocks the system AMBA bus, which interconnects all the core modules and logic modules together with the resources on the AP motherboard. Accesses to the core module SDRAM alias, to any logic module or tile and resources on the AP motherboard are done through the system bus. The system bus clock is generated an by oscillator on the AP motherboard. This oscillator is programmed via configuration registers in the system controller FPGA.

Accesses to the system bus go through an asynchronous bridge between the local bus and the system bus domains. This bridge normally adds significant delay, due to the synchronisation between the two clocks and the complexity of the bridge itself. The speed of accesses to the system bus depends on the frequency of the local bus and the system bus clock.

Integrator/CP

Systems based around a CP board comprise the CP board, one core module and a number of logic modules or tiles. This system has 2 clock domains:

- Core clock: As per the AP, it clocks the ARM core inside the test chip. This core is generated on the core module.
- System bus clock: It clocks the system AMBA bus, which connects the ARM core with the memory devices and peripherals on the core module, the CP baseboard and the logic modules. The system bus clock is also generated by the core module.

Since the CP system only has one system bus clock there is no need for an asynchronous bridge between the core module and the CP baseboard. Therefore accesses to the baseboard resources are faster in a CP system than in an AP system.

Core modules also have different internal clock architectures. The differences are shown below:

CM7TDMI

This core module has only one programmable clock oscillator, since the core and local bus activity are timed by the same clock signal (MCLK). This clock is generated by an oscillator on the core module, which is programmed via configuration registers in the core module FPGA.

CM720T, CM740T, CM920T, CM940T

These modules have two programmable clock oscillators - one each for core and local bus clocks.

ARM7x0T and ARM9x0T cores support three clocking schemes: Fastbus, Synchronous and Asynchronous modes. These define the relationship between the core and local bus clock signals. In Fastbus mode, the core clock signal is ignored, and the core and local bus are timed by the bus clock signal (see the core TRM for full details).

The Integrator core modules for these cores only support Asynchronous and Fastbus modes because the core clock and local bus clock are not synchronous.

The core and local bus clocks are generated by oscillators on the core module. These oscillators are programmed via configuration registers in the core module FPGA.

CM922T-XA10

The CM922T-XA10 or Excalibur core module is significantly different from the other core modules. The Excalibur PLD contains a stripe with the ARM core, two AHB buses, SSRAM, DPSSRAM and an SDRAM controller.

The ARM core, SSRAM, DPSSRAM and SDRAM controller are all connected to the stripe bus AHB1. The core and the AHB1 bus are clocked with the same clock, so the core should always be configured in Fastbus mode.

The internal logic of the SDRAM controller is clocked with a different clock (see below).

The second AHB bus, AHB2, runs at half the speed of AHB1. AHB2 is also connected to the SSRAM, DPSSRAM and the SDRAM controller. There is a synchronous AHB-AHB bridge between buses AHB1 and AHB2.

The core module registers and other logic implemented in the Excalibur PLD (outside of the stripe) are clocked by the system bus clock. The stripe implements an asynchronous bridge between the internal AHB2 bus and the system bus.

The ARM922T core clock frequency can be changed from its default value by writing to the appropriate PLL control registers within the Excalibur device. There are two PLLs in the device; one which supplies the core and local AHB clocks, and another for the SDRAM controller clock.

Please refer to the *Excalibur Devices Hardware Reference Manual* for full details. This document can be downloaded from Altera's website: http://www.altera.com.

CM946ES, CM966ES, CM926EJS

These modules have one programmable clock source, which is generated by a programmable oscillator on the core module. This clock is fed to the ARM test chip and is used to derive both the core and local bus clock signals. The core clock is by default the same frequency as the test chip input clock frequency, but can be multiplied up by a PLL in some test chips.

The ratio between the clock source, the core clock and the local bus clock is programmed via the core module registers in the FPGA.

The local bus clock signal is generated by the ARM test chip, from the core clock. It is synchronous to the core clock, and the division ratio is selectable between 1 and 8.

Note that some changes to clocking scheme configuration registers will not take effect until a soft reset has been performed.

CM10200E

The clock architecture of this module is similar to the CM9x6ES. It also has one programmable clock source, which is fed to the ARM test chip, but in this module three clocks are derived from it: core clock, internal bus clock and local bus clock.

The ratio between the clock source, the core clock, the internal bus clock and the local bus clock is programmed via accesses to ARM coprocessor 15 and the core module registers in the FPGA.

The internal bus clock is used by the SDRAM controller inside the test chip, so accesses to private SDRAM at address 0x30000000 are made at the internal bus clock speed.

- ***What clock speed do the boards operate at?***

Integrator/AP system

Typically, the core frequency generator can provide a clock signal at up to 160MHz. Some test chips, namely the ARM946ES, ARM966ES, ARM926EJS and ARM10200E, can multiply this frequency with an internal PLL to generate the core clock internally.

The fact that you can generate a very fast core clock does not mean that the test chip can run at that frequency. The manufacturing process and particular implementation of each test chip imposes a limit on its actual maximum core clock frequency.

Similarly, core module memory bus speeds are limited by the test chip and memory controller design. This is typically 30-40MHz. In an AP system, the core module communicates with other modules and the system controller FPGA at the system bus frequency, which is typically 33MHz.

All interfaces are implemented in programmable logic and the bus speed achievable is dependant partly on the FPGA/PLD configuration and partly on the system bus interface chosen, which can be programmed to be either ASB or AHB.

To ensure correct operation the core clock cannot be slower than the local bus clock. This effectively sets the lower operational limit for the core clock.

The processor test chips used on ARM core modules are produced by a number of different silicon vendors and are not speed graded. Currently, core modules are not sold by test chip type, so we cannot specify maximum operating frequencies for a particular core module.

An exception is the CM922T-XA10 core module. Excalibur core modules are built with production grade Excalibur chips, rather than test chips, as are used on the other Integrator core module types. Thus, it is possible to be more specific about the maximum clock speeds attainable with this module. The core will run at a maximum of 200MHz, but the maximum core clock frequency that can be set on the core module is 198MHz (33MHz x 6). This is also the reset default.

In summary, across different ARM cores, test chip types and bus interfaces, the maximum frequency of operation will vary.

The figures in the table below are typical operating frequencies that should be obtainable, taking into account the slowest test chips that we have found. For the newest core modules we include the maximum core clock frequency for the slowest and fastest test chips.

Depending on the exact boards used and the nature of the code that you are running, you may be able to achieve higher frequencies than the values in the table. In some instances slightly lower frequencies may be seen. These figures are not guaranteed maximums, and should be used for guidance only. These figures are based on silicon that we have seen operating at room temperature.

CM + AP	Core	Local Bus	System Bus	
			ASB	AHB
CM7TDMI	N/A	45MHz	25MHz	33MHz
CM720T	80MHz	40MHz	25MHz	33MHz
CM740T	60MHz	25MHz	25MHz	33MHz
CM940T	120MHz	30MHz	25MHz	33MHz
CM920T	120 - 160MHz	35MHz	25MHz	33MHz
CM922T-XA10	198MHz	198MHz	N/A	33MHz
CM946E-S	65 - 160MHz	30MHz	N/A	33MHz
CM966E-S	133 - 185MHz	40MHz	N/A	33MHz
CM926EJ-S	120 - 200MHz	40MHz	N/A	33MHz
CM10200E	260MHz	70MHz Priv SDRAM 40MHz Local Bus	N/A	33MHz

Table 1: Integrator/AP maximum frequencies

Note that newer core modules only support AHB system bus. The CM922T-XA10 local bus is inside the Excalibur stripe and is clocked as the same frequency as the core. For the CM10200E we give maximum frequencies for the internal bus clock for the SDRAM controller inside the test chip, and for the core module local bus.

Integrator/CP system

As per the AP system, the CP's programmable clock oscillators can be set to values outside the operating limits of the ARM test chip and system logic.

Unlike the AP system, the CP also has some minimum frequency limits for correct operation. This is because Delay Locked Loops (DLLs) are used within the system logic to produce the system bus clock. These components cannot achieve lock below a certain clock frequency.

As with standard core modules, to ensure correct operation the core clock cannot be slower than the local bus clock. This effectively sets the lower operational limit for the core clock. In the case of the Excalibur core module, the core clock is generated with an internal PLL, which can only provide a clock with a frequency higher than 22MHz.

This information is summarized in the following table:

CM + CP	Core Max	Bus Max	Core Min	Bus Min
CP920T	120 - 160MHz	30MHz	25MHz	25MHz
CP922T-XA10	198MHz	198MHz	22MHz	22MHz
CP946E-S	65 - 160MHz	40MHz	25MHz	25MHz
CP966E-S	133 - 185MHz	40MHz	25MHz	25MHz
CP926EJ-S	120 - 200MHz	40MHz	25MHz	25MHz

Table 2: Integrator/CP maximum and minimum frequencies

- ***Further guidelines on configuring core module clocks***

When a core module is powered on, the programmable clock generators are set to 'safe' frequencies by bias resistors on the board. Once the FPGA has configured successfully, it starts driving its outputs, and this sets the default clock frequencies. The hardware default frequency for each module is documented in the module User Guide.

These are safe operating frequencies for the module type, taking into account the variety of test chips which could be fitted to a board, so may be considerably slower than the maximum achievable frequencies. There has been one exception to this rule, where a particular batch of ARM946E-S test chips fitted to core modules have only been found to run reliably at core frequencies of less than about 65MHz, if the TCMs are enabled.

It may therefore be necessary to program the oscillator control registers with different values if higher clock frequencies are required by your application, or lower frequencies are required to make the test chip function reliably.

If the motherboard DIP switches are set to run the boot monitor then jump to a previously selected image in application flash (S1[1] = ON and S1[4] = OFF), then core and bus clock frequencies will be changed to the settings stored in the SIB (System Information Block). The SIB resides in the application flash area on an Integrator motherboard, and is automatically created from the hardware defaults, the first time the boot monitor is run. The user can subsequently change the SIB frequencies by using the boot monitor program.

The Integrator core modules CM7x0T, CM9x0T and CM922T-XA10 only support Asynchronous and Fastbus modes. Fastbus mode is the reset default, but the boot monitor will always select Asynchronous mode if allowed to run (S1[1] = ON).

- ***System Bus Bridge Considerations***

Integrator/CP system

The core module FPGA images for Integrator/CP implement a synchronous bridge between the local bus and the system bus. This bridge is unidirectional, so the core can access the resources in the system bus, but the core module resources cannot be accessed from the system bus.

The synchronous system bus bridge is simple and only introduces a 2-cycle delay per access. In order to keep its simplicity it does not implement a write buffer (a FIFO), so write accesses through the system bus are not completed in a single cycle.

Integrator/AP system

The core module FPGA images for Integrator/AP implement an asynchronous bridge between the local bus and the system bus.

Some core modules have a 16-entry FIFO for accesses between the local bus and the system bus. The purpose of these buffers is to allow the majority of writes through the asynchronous bridge to complete in just one or two bus cycles. When the FIFO is full, the bridge generates a RETRY response in the system bus, so it is not stalled.

Other core modules only implement a 1-bit deep FIFO, so only the first write through the bridge is speeded up. Subsequent writes through the bridge need to wait until the previous access completes. For these core modules write accesses to the system bus take 7 to 10 cycles with a local to system bus clock ratio of 1:1.

Since this system bus bridge doesn't implement a deep FIFO, it doesn't generate any RETRY responses. During the time that it takes for the write access to complete, the system bus bridge drives HREADY low in the system bus.

The system bus bridge in different core modules is depicted in the following two diagrams:

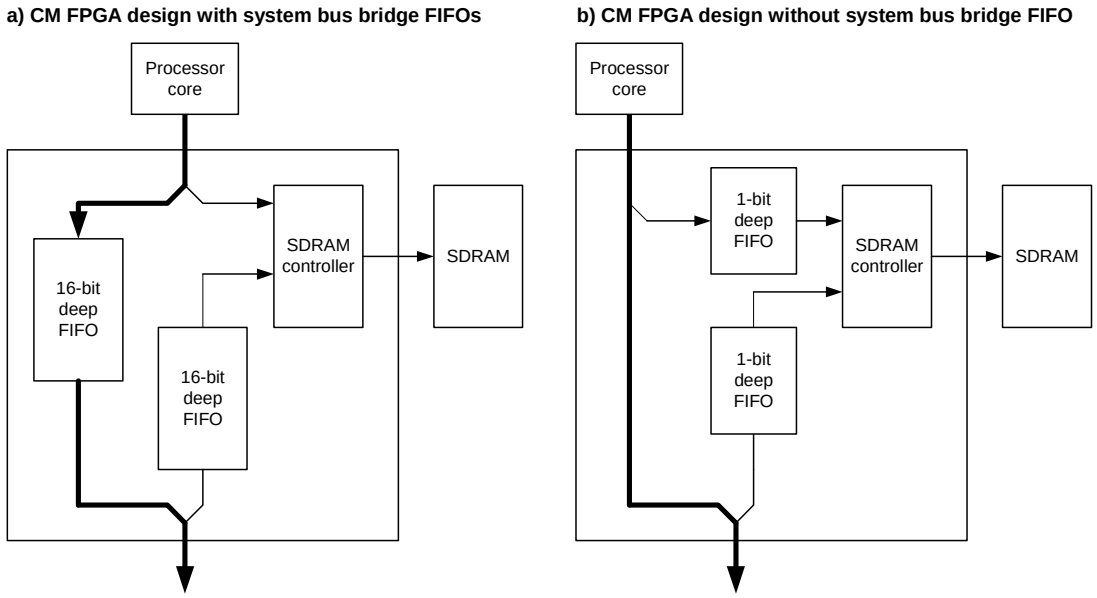

Figure 1: The core module asynchronous bridge (AP images only)

Modules which have the FIFO are:	Modules which do not have the FIFO are:
CM7TDMI	CM946E-S
CM720T	CM966E-S
CM740T	CM926EJ-S
CM940T (HBI-0047)	CM10200
CM920T (HBI-0047)	CM10200E
CM920T-ETM (HBI-0070)	

Table 3: Core modules with and without a FIFO in the asynchronous bridge

The current information about the system bus bridge in the user guides for the CM9x6E-S, CM926EJ-S and CM10200E is incorrect. This will be fixed in future revisions of the user guide.

There are currently no plans to add the system bus FIFO to core modules that do not incorporate it.

3. Memory Access Timing Spreadsheet

- ***How many cycles does it take to access memory on my Integrator board?***

The number of wait states inserted by the memory system has an important effect on the performance of a system. This section gives information about the number of bus cycles needed to access each area of memory for all Integrator core modules and platforms.

The memory access time has been measured by monitoring the address and control buses of the ARM test chip with a logic analyser. The steps taken to generate each access are:

- Configure the core clock, local bus clock (if applicable) and system bus clock with the boot monitor. Unless otherwise stated the core runs at 80MHz and the buses at 20MHz. For AP images, the system bus is configured as AHB

- Generate the appropriate memory access. This has been done with the application *Access_Generation*. We have measured memory accesses for cache line fills, ARM write buffer drain, and word, half-word and byte reads and writes

- Measure the memory access with a logic analyser and count the number of wait cycles inserted by the local memory bus

This information is summarized in the memory access timing spreadsheet on the following page (an A3 page).

Table 4: Memory Access Timing Spreadsheet

	CM7TDMI	CM740T	CM720T	CM940T	CM920T	CM920-ETM	CM922T-XA10	CM946ES	CM966ES	CM926EJS	CM10200	CM10200E		CP920T	CP946ES	CP966ES	CP926EJS	CP922T-XA10
CM SSRAM R/D											1-1	1-1						
CM SSRAM W/D											1-1	1-1						
CM SSRAM R/W	2-1[1]	2-1[1]	2-1[1]	2-1[1]	2-1[1]	1-1		1-1	1-1	1-1	1-1	1-1		1-1	1-1	1-1	1-1	
CM SSRAM W/W	2-1[1]	2-1[1]	2-1[1]	2-1[1]	2-1[1]	1-1		1-1	1-1	1-1	1-1	1-1		1-1	1-1	1-1	1-1	
CM SSRAM R/H	2-2	2-2	2-2	2-2	2-2	1-1		1-1	1-1	1-1	1-1	1-1		1-1	1-1	1-1	1-1	
CM SSRAM W/H	2-2	2-2	2-2	2-2	2-2	1-1		1-1	1-1	1-1	1-1	1-1		1-1	1-1	1-1	1-1	
CM SSRAM R/B	2-2	2-2	2-2	2-2	2-2	1-1		1-1	1-1	1-1	1-1	1-1		1-1	1-1	1-1	1-1	
CM SSRAM W/B	2-2	2-2	2-2	2-2	2-2	1-1		1-1	1-1	1-1	1-1	1-1		1-1	1-1	1-1	1-1	
CM SDRAM R/D											12-12	11-11						
CM SDRAM W/D											1-1	1-1						
CM SDRAM R/W	9-1[2]	9-1[2]	9-1[2]	9-1[2]	9-1[2]	9-1[2]		11-12	11-12	11-12	11-11	11-11		8-3	6-1	6-1	6-1	
CM SDRAM W/W	3-1[3]	2-1[3]	2-1[3]	2-1[3]	2-1[3]	2-1[3]		1-1-12[5]	1-1-13[5]	1-1-13[5]	1-1	1-1		9-3	7-1	7-1	7-1	
CM SDRAM R/H	10-10	9-9	11-9	11-9	11-9	9-8[4]		11-11	11-11	11-11	11-11	11-11		8-3	6-1	6-1	6-1	
CM SDRAM W/H	3-6	3-8	2-7	2-7	2-7	2-6[4]		1-1-12[5]	1-1-11[5]	1-1-12[5]	1-1	1-13[5]		9-3	7-1	7-1	7-1	
CM SDRAM R/B	10-10	9-9	9-9	9-9	9-9	9-12[4]		11-11	11-11	11-11	11-11	11-11		8-3	6-1	6-1	6-1	
CM SDRAM W/B	3-6	3-8	2-7	2-7	2-7	2-6[4]		1-1-12[5]	1-1-11[5]	1-1-13[5]	1-1	1-9[5]		9-3	7-1	7-1	7-1	
CM SDRAM ALIAS R/D											40-40	41-41						
CM SDRAM ALIAS W/D											15-15	20-20						
CM SDRAM ALIAS R/W	29-29	29-29	25-25	25-25	25-25	26-26		25-25	24-24	24-24	26-26	25-25						
CM SDRAM ALIAS W/W	2-1[6]	2-1[6]	2-1[6]	2-1[6]	2-1[6]	2-1[6]		13-13	13-13	13-13	13-13	13-13						
CM SDRAM ALIAS R/H	27-27	27-27	25-25	25-25	25-25	24-24		25-25	24-24	25-25	20-20	21-21						
CM SDRAM ALIAS W/H	3-3	2-2	2-2	2-2	2-2	2-2		14-14	15-15	13-13	10-10	13-13						
CM SDRAM ALIAS R/B	26-26	25-25	25-25	25-25	25-25	25-25		24-24	23-23	23-23	20-20	20-20						
CM SDRAM ALIAS W/B	3-3	2-2	2-2	2-2	2-2	2-2		14-14	13-13	13-13	10-10	13-13						
CM REGISTERS R/D											6-6	6-6						
CM REGISTERS W/D											1-1	1-2						
CM REGISTERS R/W	5-4	3-3	3-3	3-3	3-3	3-3	5-5	4-4	4-4	4-4	4-4	4-4		6-6	4-4	4-4	4-4	4-4
CM REGISTERS W/W	5-4	3-3	3-3	3-3	3-3	4-4	1-2	1-2	1-2	1-2	1-1	1-1		3-3	1-2	1-2	1-2	1-2
MB REGISTERS (AP) R/D											14-14	15-15						
MB REGISTERS (AP) W/D											7-7	9-9						
MB REGISTERS (AP) R/W	15-15	15-15	16-16	16-16	16-16	15-15	7-6	10-10	10-10	10-10	11-11	10-10						
MB REGISTERS (AP) W/W	3-1[6]	2-1[6]	2-1[6]	2-1[6]	2-1[6]	2-1[6]	4-4	8-8	7-7	7-7	8-8	7-7						
MB REGISTERS (CP) R/D																		
MB REGISTERS (CP) W/D																		
MB REGISTERS (CP) R/W														5-5	3-2	3-2	4-4	3-2
MB REGISTERS (CP) W/W														5-5	3-3	3-3	1-2	3-3
MB FLASH R/D											16-16	16-16						
MB FLASH R/W	16-16	17-17	16-16	16-16	16-16	16-16	8-7	11-11	11-11	11-11	11-11	11-11		10-10	8-7	8-7	8-7	8-8
MB FLASH R/H	16-16	16-16	16-16	16-16	16-16	16-16	7-7[7]	11-11	11-11	11-11	8-8	7-7		10-10	8-8	8-8	8-8	8-8[7]
MB FLASH R/B	16-16	16-16	16-16	16-16	16-16	16-16	7-7[7]	11-11	11-11	11-11	8-8	7-7		10-10	8-8	8-8	8-8	8-8[7]
MB SSRAM R/D											12-12	13-13						
MB SSRAM W/D											8-8	12-12						
MB SSRAM R/W	14-14	15-15	14-14	14-14	14-14	14-14	6-5	9-9	9-9	9-9	9-9	9-9						
MB SSRAM W/W	3-1[6]	2-1[6]	2-1[6]	2-1[6]	2-1[6]	2-1[6]	6-6	9-9	10-10	9-9	9-9	9-9						
MB SSRAM R/H	14-14	14-14	15-15	15-15	14-14	14-14	6-6[7]	10-10	9-9	10-10	6-6	6-6						
MB SSRAM W/H	3-3	2-2	2-2	2-2	2-2	2-2	6-6[7]	9-9	9-9	10-10	6-6	9-9						
MB SSRAM R/B	14-14	14-14	14-14	14-14	14-14	14-14	6-6[7]	9-9	9-9	9-9	6-6	6-6						
MB SSRAM W/B	3-3	2-2	2-2	2-2	2-2	2-2	6-6[7]	9-9	9-9	9-9	6-6	9-9						
MB PCI R/D											25-25	24-24						
MB PCI W/D											7-7	8-8						
MB PCI R/W	21-21	21-21	21-21	21-21	20-20	22-22	14-14	15-15	15-15	16-16	16-16	15-15						
MB PCI W/W	3-1[6]	2-1[6]	2-1[6]	2-1[6]	2-1[6]	2-1[6]	3-3	7-7	7-7	7-7	7-7	8-8						
MB PCI R/H	21-21	20-20	21-21	21-21	21-21	22-22	13-13[7]	16-16	16-16	16-16	15-15	15-15						
MB PCI W/H	3-3	2-2	2-2	2-2	2-2	2-2	4-4[7]	7-7	8-8	7-7	7-7	8-8						
MB PCI R/B	21-21	22-22	21-21	20-20	22-22	21-21	13-13[7]	16-16	15-15	16-16	16-16	16-16						
MB PCI W/B	3-3	2-2	2-2	2-2	2-2	2-2	4-4[7]	7-7	8-8	7-7	7-7	8-8						
LM SSRAM R/D											8-8	8-8						
LM SSRAM W/D											8-8	8-8						
LM SSRAM R/W	12-12	12-12	12-12	12-12	12-13	12-12	4-3	7-7	7-7	7-7	6-6	7-7						
LM SSRAM W/W	3-1[6]	2-1[6]	2-1[6]	2-1[6]	2-1[6]	2-1[6]	4-4	7-7	7-7	7-7	6-6	7-7						
LM SSRAM R/H	12-12	12-12	12-12	12-12	13-13	11-11	4-4[7]	7-7	7-7	7-7	8-8	7-7						
LM SSRAM W/H	3-3	2-2	2-2	2-2	2-2	2-2	4-3[7]	7-7	7-7	7-7	7-7	7-7						
LM SSRAM R/B	12-12	12-12	12-12	12-12	12-12	11-11	4-4[7]	7-7	7-7	7-7	7-7	7-7						
LM SSRAM W/B	3-3	2-2	2-2	2-2	2-2	2-2	4-3[7]	7-7	7-7	7-7	7-7	7-7						
CM PRIV SDRAM							***6 cycles/word[9]***					***2.4 cycles/word[8]***						***6 cycles/word[9]***
CM STRIPE SSRAM							***2 cycles/word[10]***											***2 cycles/word[10]***
CM STRIPE DPSSRAM							***2 cycles/word[10]***											***2 cycles/word[10]***

- ***Understanding the information in the memory access timing spreadsheet***

The spreadsheet summarizes the number of cycles needed to access each area of the memory map. This information is given for different types of access and different Integrator boards.

TYPES OF CELLS

- White cells with normal font contain information about areas of memory that can be accessed by the core and can be observed with a logic analyser.

Each of these cells contains two numbers:

1 - Number of bus cycles needed to get the first word / half-word / byte in a burst (non-sequential access)

2 - Number of bus cycles needed to get any subsequent words / half-words / bytes in a burst (sequential access)

The number of access cycles is calculated as the number of wait cycles inserted by the memory system plus one. The wait cycles are indicated by the nWAIT, BWAIT or HREADY input of the test chip depending on the core module.

All accesses measured correspond to one non-sequential access followed by seven sequential accesses, so if the cell contains the values 6-1, we should expect the access to take 6-1-1-1-1-1-1-1 cycles for the complete burst.

The memory access time figure does not include idle / busy cycles generated by the core.

Waveforms measured with a logic analyser for each type of core module and each type of access are included in the *Waveforms* folder. The signals shown on the waveforms are core signals (not system bus signals), so they show the access timing from the core's point of view. The waveforms also show information about the idle / busy cycles generated by the core between memory accesses.

Memory accesses were generated with the software included in the *Access_Generation* folder.

- White cells with italic font indicate areas of memory that can be accessed by the core but cannot be observed with a logic analyser. The value given in these cells is the average number of bus cycles that it takes to access a word in memory.

These accesses were measured by running the software in the *Range_Generation* folder.

- Gray cells indicate an area of memory that is not accessible by the core

COLUMNS

Each column of the spreadsheet corresponds to a different development system set-up.

The first twelve columns are for core modules mounted on an Integrator/AP motherboard (e.g. CM946ES).

There are five additional columns for core modules mounted on top of a CP baseboard (e.g. CP946ES).

The core module CM920T corresponds to the old version of the PCB (HBI-0047), whereas the core module CM920T-ETM corresponds to the new version of the PCB (HBI-0070)

The CM10200 is a non-released core module. It has an ARM10200T (rev0) test chip and was superseded by the CM10200E before being released. Since there are partners that have got one of these pre-released core modules, we have included performance figures as well.

ROWS

Each row of the spreadsheet corresponds to a different location in memory and/or a different type of access.

The possible types of access are:

- R/D: Read double word

  This only applies to the CM10200 and CM10200E. The access is generated with an LDM instruction of 8 registers with caches enabled, which generates a burst of four double-word read accesses.

- W/D: Write double word

  This only applies to the CM10200 and CM10200E. The access is generated with an STM instruction of 8 registers with caches enabled and configured in write-through mode, which generates a burst of four double-word write accesses.

- R/W: Read word

  For the CM10200 and CM10200E this cannot be generated with an LDM instruction, since it generates double-word accesses. We generate this access with eight LDR instructions. These eight instructions are not sequential since there are instruction fetches between data accesses.

  For other core modules, the access is generated with an LDM instruction of 8 registers with caches enabled, which generates a burst of eight word read accesses.

- W/W: Write word

  For the CM10200 and CM10200E this cannot be generated with an STM instruction, since it generates double-word accesses. We generate this access with eight STR instructions. These eight instructions are not sequential since there are instruction fetches between data accesses.

  For other core modules, the access is generated with an STM instruction of 8 registers with caches configured in write-through mode, which generates a burst of eight word write accesses.

  The waveforms for the CM946ES show instruction fetches between the 8 write accesses corresponding to the STM instructions. This happens because we have used an ARM946ES rev0, which has this bug in the silicon.

- R/H: Read halfword

  This access is generated with eight LDRH instructions with caches disabled.

- W/H: Write halfword

  This access is generated with eight STRH instructions with caches configured in write-through mode.

- R/B: Read byte

This access is generated with eight LDRB instructions with caches disabled.

- W/B: Write byte

  This access is generated with eight STRB instructions with caches configured in write-through mode.

The areas of memory accessed are:

- CM SSRAM: core module SSRAM at address 0x00002000
- CM SDRAM: core module SDRAM DIMM at its default location, address 0x00200000
- CM SDRAM ALIAS: core module SDRAM DIMM at its alias location at address 0x80000000
- CM REGISTERS: core module registers at address 0x10000000
- MB REGISTERS (AP): AP motherboard registers at address 0x11000000
- MB REGISTERS (CP): CP motherboard registers at address 0xCB000000
- MB FLASH: motherboard flash at address 0x24000000
- MB SSRAM: motherboard SSRAM at address 0x28000000 (AP only)
- MB PCI: motherboard PCI controller at address 0x62000000 (AP only). This access time does not include the access time of a PCI card and the bridge AMBA-PCI
- LM SSRAM: logic module SSRAM at address 0xC2000000. Logic module 'Example2' loaded into LM-XCV2000

  The CM10200 cannot access memory on the logic module at address 0xC2000000 because this area of memory is mapped to the test chip SDRAM controller configuration area. We have measured accesses to logic module SSRAM at address 0xD2000000 by simply stacking two logic modules together.

NOTES

Some of the cells in the spreadsheet have a note number in superscript. This is where an access pattern is more complex than a sequence of one non-sequential access and several identical sequential accesses. There is an explanation for these notes in the relevant section of this document.

- ***Access_Generation software***

Access_Generation is the software used to generate memory accesses, so we can measure the memory access timings with a logic analyser.

The only source file is *Access_Generation.s*, which contains code that:

- Configures the MMU / MPU of cached cores with flat mapping and write-through strategy
- Enables the caches of cached cores if the assembly switch CACHE_ENABLED is used
- Reads an enumerated command code from address 0 of memory and an address from address 4 and generates the appropriate memory access starting at that address

You can use *Access_Generation* to benchmark Integrator or your own hardware. You can also reuse the MMU / MPU configuration code in your application.

BUILDING THE CODE

A batch file *Access_Generation_Build.bat* is provided, so the user can generate images for different cores modules with caches enabled or not enabled in a simple way.

Access_Generation_Build.bat contains several calls to the ARM assembler, each one with different assembly options. You just need to remove the word “rem” for the assembler call you want to use.

```
rem armasm -PD "CACHE_ENABLED SETL {FALSE}" -keep -PD "CM7TDMI SETL {TRUE}" -g Access_Generation.s
rem armasm -PD "CACHE_ENABLED SETL {FALSE}" -keep -PD "CM720T SETL {TRUE}" -g Access_Generation.s
rem armasm -PD "CACHE_ENABLED SETL {TRUE}" -keep -PD "CM720T SETL {TRUE}" -g Access_Generation.s
rem armasm -PD "CACHE_ENABLED SETL {FALSE}" -keep -PD "CM740T SETL {TRUE}" -g Access_Generation.s
rem armasm -PD "CACHE_ENABLED SETL {TRUE}" -keep -PD "CM740T SETL {TRUE}" -g Access_Generation.s
rem armasm -PD "CACHE_ENABLED SETL {FALSE}" -keep -PD "CM920T SETL {TRUE}" -g Access_Generation.s
rem armasm -PD "CACHE_ENABLED SETL {TRUE}" -keep -PD "CM920T SETL {TRUE}" -g Access_Generation.s
rem armasm -PD "CACHE_ENABLED SETL {FALSE}" -keep -PD "CM922T SETL {TRUE}" -g Access_Generation.s
rem armasm -PD "CACHE_ENABLED SETL {TRUE}" -keep -PD "CM922T SETL {TRUE}" -g Access_Generation.s
rem armasm -PD "CACHE_ENABLED SETL {FALSE}" -keep -PD "CM940T SETL {TRUE}" -g Access_Generation.s
rem armasm -PD "CACHE_ENABLED SETL {TRUE}" -keep -PD "CM940T SETL {TRUE}" -g Access_Generation.s
rem armasm -PD "CACHE_ENABLED SETL {FALSE}" -keep -PD "CM926EJS SETL {TRUE}" -g Access_Generation.s
rem armasm -PD "CACHE_ENABLED SETL {TRUE}" -keep -PD "CM926EJS SETL {TRUE}" -g Access_Generation.s
rem armasm -PD "CACHE_ENABLED SETL {FALSE}" -keep -PD "CM946ES SETL {TRUE}" -g Access_Generation.s
rem armasm -PD "CACHE_ENABLED SETL {TRUE}" -keep -PD "CM946ES SETL {TRUE}" -g Access_Generation.s
rem armasm -PD "CACHE_ENABLED SETL {FALSE}" -keep -PD "CM966ES SETL {TRUE}" -g Access_Generation.s
rem armasm -PD "CACHE_ENABLED SETL {TRUE}" -keep -PD "CM966ES SETL {TRUE}" -g Access_Generation.s
rem armasm -PD "CACHE_ENABLED SETL {FALSE}" -keep -PD "CM1020T SETL {TRUE}" -g Access_Generation.s
rem armasm -PD "CACHE_ENABLED SETL {TRUE}" -keep -PD "CM1020T SETL {TRUE}" -g Access_Generation.s
rem armasm -PD "CACHE_ENABLED SETL {FALSE}" -keep -PD "CM1020E SETL {TRUE}" -g Access_Generation.s
rem armasm -PD "CACHE_ENABLED SETL {TRUE}" -keep -PD "CM1020E SETL {TRUE}" -g Access_Generation.s

armlink Access_Generation.o -info totals -o Access_Generation.axf
```

The image is linked by default at address 0x8000 and the translation table is placed at address 0x4000, so the code runs from the Integrator core module SSRAM.

TRIGGERING THE LOGIC ANALYSER

Some core modules have logic analyser connectors that include the address bus signals. These core modules are:

CM7TDMI
CM920T-ETM (HBI-0070)
CM946E-S
CM966E-S
CM926EJ-S
CM10200
CM10200E

When measuring accesses with these core modules you should trigger the logic analyser at the address marked in the source file comments. Use 'Interleave disassembly' in AXD to reveal the addresses of the instructions in the source window.

Other core modules do not bring the address bus signals out to a logic analyser pod. These core modules are:

CM720T
CM740T
CM9x0T (HBI-0047)

CM922T-XA10
CP922T-XA10

With these core modules you need to re-program the core module's FPGA configuration flash with a special configuration provided in the *HDL* folder, so a trigger signal is taken to the logic analyser connectors.

The special FPGA image for the CM922T-XA10 and CP922T-XA10 takes the whole address bus to the logic analyser connector, so you can trigger the logic analyser at the address marked in the source file comments.

The other core modules do not have enough pins in the logic analyser connector, so a special trigger signal is generated when the code accesses any area of memory where the program itself is not loaded. This always corresponds to one of the accesses we want to benchmark.

More information about the special FPGA builds, including the logic analyser pod pin-outs can be found in the section "Special FPGA images".

RUNNING THE CODE

- Remember to set the core, local and system bus clocks to the desired speeds before running the tests if you wish to match the results in the spreadsheet and logic analyser traces. This is easily done using the boot monitor.

- Select the required test action and address by storing the appropriate values into addresses 0x0 and 0x4. These two addresses should be manually modified with the debugger after loading the image, but prior to running the test code, since the program reads these locations to determine which action is to be performed.

Address 0x0 should be set to select the required access type, as per the following table:

0 = LOAD MULTIPLE	generates an LDM of 8 registers
1 = STORE MULTIPLE	generates an STM of 8 registers
2 = LOAD WORD	generates 8 LDR instructions
3 = STORE WORD	generates 8 STR instructions
4 = LOAD HALFWORD	generates 8 LDRH instructions
5 = STORE HALFWORD	generates 8 STRH instructions
6 = LOAD BYTE	generates 8 LDRB instructions
7 = STORE BYTE	generates 8 STRB instructions

Address 0x4 should be set to contain the address at which the desired memory accesses will be performed, e.g. core module SSRAM at 0x2000 or core module SDRAM Alias at 0x80000000.

- Remember to disable vector catch in AXD, so the contents of memory at addresses 0 and 4 are not replaced with software breakpoints (the debugger's memory window hides these from you, showing instead the values that you expect to be there).

- If you are running a series of tests with different access types - some of which require caches enabled and some disabled, the core should be reset in between tests. This is because building the code with the '"CACHE_ENABLED = TRUE" option includes code to enable the caches, but "CACHE_ENABLED = FALSE" does not include code to disable the caches.

- Some memory areas such as the SDRAM DIMM in Integrator/CP contain a mini cache for recent accesses. Accesses to data contained in this mini cache are done in a single cycle, so if you want to measure the "real" access time of the memory device, you may also need to reset the board between accesses.

- ***Range_Generation software***

Range_Generation is the software used to measure access time for areas of memory where you cannot use a logic analyser, e.g. SSRAM memory inside the Excalibur PLD in the CM922T-XA10.

The only source file is *Range_Generation.s*, which contains code that:

- Configures the MMU / MPU of cached cores with flat mapping and write-through strategy
- Enables the caches of cached cores and sets the clocks with fixed frequencies
- Copies a sequence of SWP – NOP – NOP – ..... – NOP – SWP – B instructions to the address pointed to by the value stored at memory location 0. Memory location 4 must contain the size in bytes of the block of memory filled with NOPs. This number must be a multiple of 16 bytes
- Branches to the memory location pointed to by the value stored at memory location 0, so the core begins the execution of the sequence of NOP instructions. After executing these instructions it stays in an infinite loop

The code executed by the core between the SWP instructions is completely linear, so we ensure that every instruction is fetched from memory. By measuring the time between the SWP instructions we can calculate how long it takes to do a cache line fill and execute 8 NOPs. The time needed to execute the NOP instructions should be much smaller than the time needed to do the cache line fill as long as the core clock is much faster than the bus clock.

BUILDING THE CODE

A batch file *Range_Generation_Build.bat* is provided, so the user can generate images for different cores in a simple way.

In *Range_Generation_Build.bat* there are four possible calls to the ARM assembler.

```
rem armasm -PD "CACHE_ENABLED SETL {FALSE}" -keep -PD "CM922T SETL {TRUE}" -g Range_Generation.s
rem armasm -PD "CACHE_ENABLED SETL {TRUE}" -keep -PD "CM922T SETL {TRUE}" -g Range_Generation.s
rem armasm -PD "CACHE_ENABLED SETL {FALSE}" -keep -PD "CM1020E SETL {TRUE}" -g Range_Generation.s
rem armasm -PD "CACHE_ENABLED SETL {TRUE}" -keep -PD "CM1020E SETL {TRUE}" -g Range_Generation.s

armlink Range_Generation.o -info totals -o Range_Generation.axf
```

They select generation of code for different processors (CM922T and CM10200E) with caches enabled or not enabled. Remove the word "rem" for the assembler call you want to use. The timing figures included in this document were measured with caches enabled.

The image is linked by default at address 0x8000 and the translation table is placed at address 0x4000, so the initial code runs from the Integrator core module SSRAM.

TRIGGERING THE LOGIC ANALYSER

You should trigger the logic analyser when HLOCK goes high, which only happens during the SWP instructions. The time between the two HLOCK pulses is the time taken to execute the block of NOP instructions.

Alternatively you can trigger the analyser when the core writes to the address of memory swapped (address 0x10000000).

RUNNING THE CODE

The code itself sets up the clocks, so you do not need to program them yourself.

- Write the address of the memory area that you want to benchmark into memory address 0. Write in memory address 4 the size (in bytes) of the block of NOPs and run the code.

- If you use this code, remember to remove vector catch in AXD, so the contents of memory at addresses 0 and 4 are not replaced with software breakpoints.

Note that disabling the cache requires a reset.

• *Versions of the boards used*

The tests explained in this document were done with the following versions of the Integrator boards and FPGA images. Please note that different versions of the FPGAs for the same core module can have different performance.

Board	PCB code	FPGA build	Silicon rev	FPGA fitted
CM7TDMI	HBI 0056B	RvB build11	Silicon ID 0	XC4062XLA
CM740T	HBI 0058A	RvB build11	Core Rev 0	XC4036XL
CM720T	HBI 0050B	RvB build11	Core Rev 0	XC4036XL
CM940T	HBI 0047C	RvB build11	Core Rev 0	XC4036XL
CM920T	HBI 0047B	RvB build11	Core Rev 0	XC4036XL
CM920T-ETM	HBI 0070C	RvB build15	Core Rev 1	XCV600
CM922T-XA10	HBI 0100B	RvB build1		
CM946ES	HBI 0066D	RvB build27	Core Rev 0	XCV600
CM966ES	HBI 0066B	RvB build27	Core Rev 0	XCV600
CM926EJS	HBI 0087B	RvB build18	Core Rev 1	XCV600
CM10200	HBI 0067C	RvB build12	Core Rev 1	XCV600
CM10200E	HBI 0098A	RvB build15	Core Rev F	XCV600
CP920T-ETM	HBI 0070C	RvD build18	Core Rev 1	XCV600
CP922T-XA10	HBI 0100B	RvD build1		
CP946ES	HBI 0066D	RvD build18	Core Rev 0	XCV600
CP966ES	HBI 0066B	RvD build18	Core Rev 0	XCV600
CP926EJS	HBI 0087B	RvD build06	Core Rev 1	XCV600
AP	HBI 0048D	RvB build27		
CP	HBI 0086B	Eth build13	Disp build 3	

Table 5: Versions of the boards used for the tests

• *Special FPGA images*

Some core modules do not bring the address bus signals out to a logic analyser pod. These core modules are:

CM720T
CM740T
CM9x0T (HBI-0047)
CM922T-XA10
CP922T-XA10

With these core modules you need to reprogram the core module's FPGA configuration flash with the correct special configuration provided in the *HDL* folder, so a trigger signal is taken to the logic analyser connectors.

You need the Progcards utility and a Multi-ICE unit to perform the reprogramming procedure. You also need to write or modify an existing Progcards board file to carry out the programming procedure.

Note that the special FPGA images were designed to work with an Integrator/AP motherboard in AHB mode only. They cannot be used on top of an Integrator/CP baseboard. Since this code is used to measure accesses to resources controlled by logic in the test chip or PLD stripe, the performance must be the same for both AP and CP platforms.

This is an explanation of the modifications to the FPGA images of the core modules previously described. Refer to the core module circuit schematics for translation of net names to logic analyser pod pins.

CM922T-XA10 and CP922T-XA10

The .mcs file for the modified CM922T-XA10 image is *Special_Cmxa10fpga*.hex.
The .mcs file for the modified CP922T-XA10 image is *Special_Cp922txafpga.hex*.

The signals taken to the logic analyser connectors are the Control, Address and Data signals of the AHB system bus inside the FPGA.

LA_SPARE (1:0) = HTRANS (1:0)
LA_SPARE (4:2) = HBURST (2:0)
LA_SPARE (15:5) = 0
LA_SPARE (16) = HWRITE
LA_SPARE (18:17) = HPROT (1:0)
LA_SPARE (20:19) = HSIZE (1:0)
LA_SPARE (21) = 0
LA_SPARE (22) = HREADY
LA_SPARE (24:23) = HRESP (1:0)
LA_SPARE (26:25) = HPROT (3:2)
LA_SPARE (27) = HLOCK
LA_SPARE (31:28) = 0
LA_SPARE (63:32) = HADDR (31:0)
LA_SPARE (95:64) = HRDATA (31:0)

CM940T and CM920T (HBI-0047)

There is only one image for use with CM920T or CM940T, since these are both built on the same PCB. The .mcs file is *Special_cm9x0t_47b_fpga_ahbboth.mcs*.

The signals taken to the logic analyser connectors come from test chip pins.

```
SPAREIO0 = BSIZE_M (0);
SPAREIO1 = BSIZE_M (1);
SPAREIO2 = BWRITE_M;
SPAREIO3 = 0 when
   (BA_M (31:12) = "00000000000000000000") or
   (BA_M (31:12) = "00000000000000000100") or
   (BA_M (31:12) = "00000000000000000101") or
   (BA_M (31:12) = "00000000000000000110") or
   (BA_M (31:12) = "00000000000000000111") or
   (BA_M (31:12) = "00000000000000001000")
   else 1;
```

SPAREIO3 is the trigger signal. It is 1 when the core accesses memory which is not:
- At address 0x8xxx -> location of the image
- At address 0x4xxx -> location of the MMU translation table

- At address 0x0xxx -> location of the command to execute

That is, SPAREIO3 is 1 only when the core is doing the access that we want to measure.

CM720T or CM740T

There are separate images for each, since these are two different PCBs. The LA Pod pin-out is the same in both cases.

The .mcs file for the modified CM740T FPGA image is *Special_cm740t_58a_fpga_ahbboth.mcs*.
The .mcs file for the modified CM720T FPGA image is *Special_cm720t_50b_fpga_ahbboth.mcs*.

The signals taken to the logic analyser connector come from the test chip pins.

```
SPAREIO7 = BSIZE_M (0);
SPAREIO8 = BSIZE_M (1);
SPAREIO9 = 0 when
   (BA_M (31:12) = "00000000000000000000") or
   (BA_M (31:12) = "00000000000000000100") or
   (BA_M (31:12) = "00000000000000000101") or
   (BA_M (31:12) = "00000000000000000110") or
   (BA_M (31:12) = "00000000000000000111") or
   (BA_M (31:12) = "00000000000000001000")
   else 1;
```

SPAREIO9 is the trigger signal. It is 1 when the core accesses memory which is not:
- At address 0x8xxx -> location of the image
- At address 0x4xxx -> location of the MMU translation table
- At address 0x0xxx -> location of the command to execute

That is, SPAREIO9 is 1 only when the core is doing the access that we want to measure.

- ***Logic analyser waveforms***

The *Waveforms* folder contains waveforms captured with a logic analyser. There is one waveform for each cell in the memory access timing spreadsheet.

Each subfolder corresponds to one column of the spreadsheet, i.e. a specific core module and platform. Inside the subfolder there is a tiff file for each type of memory access. The name of the tiff file includes the type of board, area of memory accessed and type of access. For example:

CM922T-XA_MBSSRAM_RW.TIF – A CM922T-XA10 core module on an AP motherboard, performing a read word access to motherboard SSRAM (address 0x28000000)

CP920TETM_CMSDRAM_WH.TIF – A CM920T core module on a CP baseboard, performing a write half-word access to the CM SDRAM DIMM (address 0x00200000)

4. Integrator Memory System

In this section we give extra information about accesses to different areas of memory. This completes the information in the memory access timing spreadsheet and explains the references to notes in the spreadsheet.

- ***Access to Core Module SSRAM***

The core module SSRAM at address 0 is the fastest Integrator memory and is usually accessed in one or two cycles.

Older core modules (i.e. CM7TDMI, CM720T, CM740T, CM940T and CM920T) have old flow-through SSRAM devices fitted. The memory interface of these devices does not perfectly match the ASB bus or the ARM7TDMI bus interface, so wait cycles are inserted by the memory system.

Spreadsheet note 1

The timing of SSRAM word accesses for CM7TDMI, CM720T, CM740T, CM920T and CM940T is 2-1-1-1-2-1-1-1 cycles.

The size of the cache line for these cores (except the ARM7TDMI) is 4 words, so at the time the core modules were designed there was no big advantage in having an SSRAM access time of 2-1-1-1-1-1-1-1 cycles.

New core modules, however, use ZBT SSRAM, which can be accessed with no wait cycles from the test chip ASB or AHB bus.

The Excalibur core module does not have SSRAM fitted on the core module, but has SSRAM inside the Excalibur stripe. The access to this memory is analysed in a different section.

- ***Access to Core Module SDRAM DIMM at its default location***

The core module SDRAM DIMM can be accessed at low addresses in the memory map, just above the core module SSRAM (e.g. at address 0x00200000).

When the core module is on top of an AP motherboard the SDRAM can also be accessed at its alias location (address 0x80000000 if the core module is at the bottom of the stack). If the core module is stacked on top of a CP motherboard there is no SDRAM alias facility.

Accesses to the SDRAM at its default location go directly to the SDRAM controller, so in general are quite fast. Accesses to the SDRAM at its alias location go through the core module asynchronous bridge to the system bus; then they come back again through the asynchronous bridge to the second port of the SDRAM controller. Obviously accesses to the SDRAM at its alias location are considerably slower.

In this section we analyse accesses to the SDRAM at its default location. Some general considerations are:

- The SDRAM access figures given in the spreadsheet are the best you can get. Periodically the SDRAM controller generates refresh cycles. If there is a refresh cycle during an access to SDRAM, this access will be extended by about 10 cycles

  The DIMM SDRAM controller of Integrator/AP images has a refresh rate of 24MHz (REFCLK) divided by 256, that is 10.67us between refreshes

The SDRAM DIMM controller in the Integrator/CP images has a refresh rate of 14.7MHz (UARTCLK) divided by 192, that is 13.06us between refreshes

- All the SDRAM accesses shown in the *Waveforms* directory correspond to bursts that begin on an 8-word boundary. In this optimum case the first access in the burst is non-sequential and the rest are sequential.

  There is a worse case in which the burst does not begin on an 8-word boundary (4-word boundary for older cores), so in the middle of the burst the boundary is crossed. In this case both the first access and the access that crosses the boundary are non-sequential, and therefore slower.

  Most cached cores fill cache lines with INCR4/INCR8 burst accesses aligned to the appropriate boundaries. For these cores, running code from SDRAM with caches enabled gives the optimum access timing.

  The newest cached cores such as ARM926EJS and ARM10200E fill a cache line with WRAP4 or WRAP8 bursts. For these cores SDRAM is accessed with sub-optimum timing even with caches enabled. If the user is interested in having the best possible performance when running code from SDRAM they should use or design an improved SDRAM controller that uses the WRAP burst information to eliminate the second non-sequential access to SDRAM.

The Excalibur core module does not include an SDRAM DIMM controller, but uses the SDRAM controller inside the Excalibur stripe. The access to this memory is analysed in a different section.

Integrator/AP with CM7TDMI, CM720T, CM740T, CM920T, CM940T and CM920T-ETM

These core module images implement a simple SDRAM controller. This SDRAM controller begins a 4 word burst access to SDRAM for every non-sequential access issued by the test chip. If the core generates a 4-word burst read access (e.g. a cache line fill), all the data read from the SDRAM are sent to the core. If the core generates two non-sequential accesses in a row, some of the data read from the SDRAM are discarded.

For this type of SDRAM controller, the access time of these core modules is close to the best you can get in your final hardware.

The SDRAM DIMM is accessed in bursts of 32-bit reads/writes, so word bursts from the test chip match the timing of the SDRAM well and are quite fast. When the test chip generates half-word or byte accesses, each access is seen as a non-sequential access by the SDRAM controller, so each access generates a 4-word burst to SDRAM.

A general recommendation is to always have caches enabled when running code from SDRAM, so word bursts are generated by the core. In the case of non-cached cores such as the ARM7TDMI you should only run ARM code (not Thumb) from SDRAM, so the core does not generate half-word accesses for instruction fetches.

Spreadsheet note 2

The timing of SDRAM word read accesses for CM7TDMI, CM720T, CM740T, CM920T, CM940T and CM920-ETM is 9-1-1-1-9-1-1-1 cycles; that is 9 cycles access time for the first word of a cache line fill and 1 cycle for each of the rest.

Spreadsheet note 3

The timing of SDRAM word write accesses of CM7TDMI and CM740T is 3-1-1-1-5-1-1-1 cycles.

The timing of SDRAM word write accesses of CM720T is 2-1-1-1-13-1-1-1 cycles.

The timing of SDRAM word write accesses of the CM940T is 2-1-1-1-4-1-1-1 cycles.

The timing of SDRAM word write accesses of CM920T and CM920T-ETM is 2-1-1-1-5-1-1-1 cycles.

When writing to SDRAM, the SDRAM controller puts the data in the SDRAM bus at the same time as the control and data signals. However, when reading from SDRAM, the memory puts the data in the data bus several cycles (SDRAM access latency) after the test chip issues the control and address signals. This explains why write accesses are faster than read accesses.

The timing differences between core modules are due to the different SDRAM controllers in the core module FPGA.

Spreadsheet note 4

The timing of SDRAM half-word read accesses of CM920T-ETM is 9-8-8-8-8-15-8-8 cycles.

The timing of SDRAM half-word write accesses of CM920T-ETM is 2-6-6-6-6-6-6-6 cycles.

The timing of SDRAM byte read accesses of CM920T-ETM is 9-12-9-9-9-9-9-9 cycles.

The timing of SDRAM byte write accesses of CM920T-ETM is 2-6-14-2-6-6-6-6 cycles.

Integrator/AP with CM946ES, CM966ES, CM926EJS, CM10200, CM10200E

These core modules have test chips with an AHB interface. When the design of the FPGA images for these core modules was done, we reused the SDRAM controller from the existing core modules. The adaptation of this SDRAM controller to the test chip's AHB interface was not perfect, so these core modules have worse access time to SDRAM than the older core modules.

Since our current design efforts are focused on the CP platform and beyond, there are no plans to improve this access time.

In general every read access from SDRAM generates a read burst from the memory device, so that word, half-word and byte accesses take the same number of cycles for non-sequential and sequential accesses.

Write accesses are improved with a write FIFO inside the SDRAM controller, which can store up to four words when carrying out an STM or cache write back. The first words are written into the FIFO in a single cycle, but when the FIFO is full, the core needs to wait for an average of 10-14 cycles. This effect can be seen in the following timings.

Spreadsheet note 5

CM946ES

The timing of SDRAM word write accesses is 1-1-1-1-1-1-1-12
The timing of SDRAM half-word write accesses is 1-1-1-1-1-1-3-5
The timing of SDRAM byte write accesses is 1-1-1-1-1-8-12-12

CM966ES

The timing of SDRAM word write accesses is 1-1-1-1-1-1-10-13-13

The timing of SDRAM half-word write accesses is 1-1-1-1-1-1-1-11-11
The timing of SDRAM byte write accesses is 1-1-1-1-1-1-1-11-11

CM926EJS
The timing of SDRAM word write accesses is 1-1-1-1-1-1-13-13-13
The timing of SDRAM half-word write accesses is 1-1-1-1-1-1-10-2-12
The timing of SDRAM byte write accesses is 1-1-1-1-1-1-1-13-13

CM10200E
The timing of SDRAM half-word write accesses is 1-1-1-1-1-13-12-12
The timing of SDRAM byte write accesses is 1-1-1-1-1-1-9-11

Integrator/CP with CP946ES, CP966ES, CP926EJS

The SDRAM controller in the CP images is an optimized controller with an internal mini-cache that stores the last 8 words read or written. Any access to data stored in the mini-cache is done in a single bus cycle. This also allows sequential sub-word write accesses to be done in a single cycle.

The SDRAM can also be read from the CLCD port, which has an 8 double-word mini-cache.

The CP920T has worse access time to SDRAM than the CP9x6ES because it has an ASB-AHB synchronous bridge between the core and the AHB bus inside the core module FPGA.

- ***Accesses to the AP motherboard, CP baseboard or logic module resources***

Accesses to the AP motherboard, CP baseboard and logic module resources go through the system bus. There is a big difference in how these accesses are handled by AP and CP images.

Integrator/AP

When the core module is on top of an AP motherboard, accesses outside the core module must pass through the asynchronous bridge between the core module local bus and the system bus.

The speed of accesses to the system bus depends on both the core module local bus clock and the system bus clock frequencies. The tests we have done are always at 20MHz local bus clock and 20MHz system bus clock. If you take into account the fact that normally the local bus clock is faster than the system bus clock, accesses to the system bus resources will be slower (in terms of local bus clock cycles) than the data shown in the spreadsheet.

The delay introduced by the asynchronous bridge is variable by +/-1 cycle, depending on the phases of the local bus clock and the system bus clock, so the figures given in the spreadsheet may vary slightly.

Read accesses through the asynchronous bridge are always slow. Depending on the core module, write accesses can be fast or slow.

- The CM7TDMI, CM720T, CM740T, CM920T, CM940T and CM920T-ETM implement a 16-entry FIFO write buffer in the asynchronous bridge, so while this buffer is not full, write accesses can be finished in a single cycle.

  Spreadsheet note 6

  The timing of word write accesses through the asynchronous bridge for the CM7TDMI is 3-1-1-1-2-1-1-1 cycles.

The timing of sub-word write accesses through the asynchronous bridge for the CM7TDMI is 3-3-3-3-3-3-3-3 cycles.

The timing of word write accesses through the asynchronous bridge for the CM720T, CM740T, CM920T, CM940T and CM920-ETM is 2-1-1-1-2-1-1-1 cycles.

The timing of sub-word write accesses through the asynchronous bridge for the CM720T, CM740T, CM920T, CM940T and CM920-ETM is 2-2-2-2-2-2-2-2 cycles.

- The CM946ES, CM966ES, CM926EJS, CM10200 and CM10200E have a simpler asynchronous bridge without a FIFO write buffer. In general reads through the asynchronous bridge are 5 cycles faster than older core modules, but writes through the asynchronous bridge take from 7 to 10 cycles instead of finishing in a single cycle.

The speed at which different areas of the system bus memory map are accessed depends on the bridges and memory controllers in the motherboard and the logic module. In general, the fastest memory is the logic module SSRAM, followed by the motherboard SSRAM, motherboard Flash and finally the PCI system.

Accesses to the SDRAM alias at address 0x80000000 also go through the asynchronous bridge, so read accesses are very slow and write accesses are fast or slow depending on the core module.

Integrator/CP

When a core module is stacked on top of a CP motherboard the core module local bus and the system bus are clocked by the same signal, which eliminates the need for an asynchronous system bus bridge.

In general, accesses to the resources outside the core module are much faster in Integrator/CP than in Integrator/AP and depend on the access time of the memory device itself instead of the delay added by the system bus bridge.

Integrator/CP images do not map an alias of the SDRAM at address 0x80000000, so this test is not possible and hence is missing from the spreadsheet. The CP baseboard does not have any SSRAM or a PCI system either. The logic module can be accessed, but in that case you cannot connect a logic analyser to the core module pods, so that access information is not included in the spreadsheet.

CM922T-XA10 and CP922T-XA10

The CM922T-XA10 or CP922-XA is a special case. The stripe in the Excalibur device implements an asynchronous bridge between the ARM stripe and the logic in the PLD. The logic in the PLD belongs to the system bus and is clocked with the system bus clock, so responds very fast to accesses from the stripe.

Since the ARM core is embedded in the stripe we cannot measure the signals coming from the core; only the signals coming from the stripe. We can calculate the access time to Integrator resources as the number of wait states inserted by the memory system plus one. Between accesses the asynchronous bridge generates a number of idle cycles: some of them correspond to core idle cycles and some to delay introduced by the asynchronous bridge.

Spreadsheet note 7

During word accesses the asynchronous bridge does not insert extra idle cycles.

During sub-word read accesses the asynchronous bridge inserts 14 idle cycles per access when the core clock is 198MHz and the bus clock is 50MHz. It inserts 10 idle cycles per access when the core clock is 198MHz and the bus clock is 14MHz.

During sub-word write accesses the asynchronous bridge inserts 6 idle cycles per access when the core clock is 198MHz and the bus clock is 50MHz.

Please refer to the *Excalibur Devices Hardware Reference Manual* for full details. This document can be downloaded from Altera's website: http://www.altera.com.

- ***Accesses to internal resources***

In order to measure the access time for memory blocks and memory controllers inside the test chip or the Excalibur stripe you cannot use a logic analyser, so another method must be used.

We have measured these accesses in two different ways:

1. Using a Multi-Trace unit and enabling cycle accurate tracing. This counts the number of core cycles needed to execute each instruction

2. Filling an area of memory with a block NOP instructions and measuring the time needed to execute them. This is generated with the *Range_Generation* software.

CM10200E Private SDRAM

The CM10200E has 64MB of private SDRAM fitted to the board and mapped at address 0x30000000 (not to be confused with the SDRAM DIMM). This SDRAM is accessed by an SDRAM controller implemented in the ARM10200E test chip. This SDRAM controller is clocked with the "internal clock", whose frequency is a multiple of the local bus clock.

Spreadsheet note 8

The private SDRAM access speed was measured with the clocks configured at 280MHz (core), 40MHz (internal bus) and 20MHz (local bus).

Range_Generation was used to fill a block of memory of 0x800 bytes from address 0x30000000 with NOP instructions. The time to execute these instructions is 30.52us.

Since the internal bus clock is much slower than the core clock we can assume that the time needed to execute each instruction is approximately the time needed to fetch it from memory. Since the internal bus clock is 40MHz it takes 1220 cycles to fetch 512 instructions, that is 2.4 cycles per instruction.

Since the cache line fills generate double-word accesses, we can assume that the private SDRAM has an access time of 5 cycles for a double-word access.

CM922T-XA10 and CP922T-XA10 Private SDRAM

The Excalibur board has 128 MB of Private SDRAM fitted to the board and mapped at address 0x04000000. This SDRAM is accessed by the SDRAM controller inside the Excalibur stripe.

Spreadsheet note 9

The AHB1 bus between the core and the SDRAM controller runs at 198MHz and the SDRAM controller is clocked at 123MHz. This is the default for the board after power-up reset.

Range_Generation was used to fill a block of memory of 0x400 bytes from address 0x04000000 with NOP instructions. The time to execute these instructions is 8.068us.

Since the AHB1 bus clock is 198MHz, it takes 1597 bus cycles to execute 256 instructions, that is, on average it takes 6 cycles to execute each instruction.

In order to double-check these results, we measured the execution trace with Multi-Trace and cycle accurate tracing. The trace recorded is the following:

Cycle	Address	Instruction				Timestamp (ns)
215	0x04000004	0xE2800000	add	r0,r0,#0	ARM	1090.0
216	0x04000008	0xE2800000	add	r0,r0,#0	ARM	1110.0
217	0x0400000c	0xE2800000	add	r0,r0,#0	ARM	1110.0
218	0x04000010	0xE2800000	add	r0,r0,#0	ARM	1110.0
219	0x04000014	0xE2800000	add	r0,r0,#0	ARM	1110.0
220	0x04000018	0xE2800000	add	r0,r0,#0	ARM	1130.0
260	0x0400001c	0xE2800000	add	r0,r0,#0	ARM	1340.0
261	0x04000020	0xE2800000	add	r0,r0,#0	ARM	1340.0
262	0x04000024	0xE2800000	add	r0,r0,#0	ARM	1340.0
263	0x04000028	0xE2800000	add	r0,r0,#0	ARM	1340.0
264	0x0400002c	0xE2800000	add	r0,r0,#0	ARM	1360.0
265	0x04000030	0xE2800000	add	r0,r0,#0	ARM	1360.0
266	0x04000034	0xE2800000	add	r0,r0,#0	ARM	1360.0
267	0x04000038	0xE2800000	add	r0,r0,#0	ARM	1360.0
308	0x0400003c	0xE2800000	add	r0,r0,#0	ARM	1590.0
309	0x04000040	0xE2800000	add	r0,r0,#0	ARM	1590.0

The execution timing is 40-1-1-1-1-1-1-1 - 40-1-1-1-1-1-1-1 cycles. The 40 cycles for the non-sequential cycles include 4 idle cycles inserted by the core before beginning the cache line fill, 4 inserted by the AHB wrapper and 32 wait state cycles inserted by the memory system.

After this each word in the burst of eight is fetched in a single cycle.

CM922T-XA10 and CP922T-XA10 Stripe SSRAM and DPSSRAM

The Excalibur stripe has 256KB of internal SSRAM and 128KB of internal Dual Port SSRAM, accessed at addresses 0x08000000 and 0x08100000 respectively.

Spreadsheet note 10

The SSRAM and DPSSRAM are connected to the AHB1 bus, which runs at 198MHz.

Range_Generation was used to fill a block of memory of 0x1000 bytes from address 0x08000000 and 0x08100000 with NOP instructions. The time to execute these instructions is 10.71us in both cases.

Since the AHB1 bus clock is 198MHz, it takes 2121 bus cycles to execute 1024 instructions, that is in average it takes 2 cycles to execute each instruction.

In order to double-check these results, we measured the execution trace with Multi-Trace and cycle accurate tracing. The trace recorded is the following:

Cycle	Address	Instruction				Timestamp (ns)
139	0x08000008	0xE2800000	add	r0,r0,#0	ARM	700.0
140	0x0800000c	0xE2800000	add	r0,r0,#0	ARM	720.0
141	0x08000010	0xE2800000	add	r0,r0,#0	ARM	720.0
142	0x08000014	0xE2800000	add	r0,r0,#0	ARM	720.0
143	0x08000018	0xE2800000	add	r0,r0,#0	ARM	720.0
151	0x0800001c	0xE2800000	add	r0,r0,#0	ARM	760.0
152	0x08000020	0xE2800000	add	r0,r0,#0	ARM	790.0

153	0x08000024	0xE2800000	add	r0,r0,#0	ARM	790.0
154	0x08000028	0xE2800000	add	r0,r0,#0	ARM	790.0
155	0x0800002c	0xE2800000	add	r0,r0,#0	ARM	790.0
156	0x08000030	0xE2800000	add	r0,r0,#0	ARM	810.0
157	0x08000034	0xE2800000	add	r0,r0,#0	ARM	810.0
158	0x08000038	0xE2800000	add	r0,r0,#0	ARM	810.0
166	0x0800003c	0xE2800000	add	r0,r0,#0	ARM	850.0
167	0x08000040	0xE2800000	add	r0,r0,#0	ARM	850.0
168	0x08000044	0xE2800000	add	r0,r0,#0	ARM	880.0

The execution timing is 8-1-1-1-1-1-1-1 - 8-1-1-1-1-1-1-1 cycles. The 8 cycles for the non-sequential cycles include 4 cycles inserted by the core before beginning the cache line fill and 4 inserted by the AHB wrapper. The memory device itself is accessed in a single cycle.

5. Optimizing the Performance of Integrator

- ***General hints***

  The following are general hints that help improve the performance of Integrator:

  - Program the clocks to the desired speeds, the defaults are low
  - Set up the core with the appropriate clock mode (if applicable)
  - The core TCMs are usually accessed in a single clock cycle. If you core has TCMs, use them
  - If using a cached core ensure the cache is correctly configured and enabled. In general all areas that can contain your code or data should be cached (e.g. SSRAM, DRAM and Flash). I/O addresses must not be cached
  - Code and data accessed very often (e.g. the stack) or which require fast access (e.g. FIQ handler) should be placed in the fastest memory areas
  - Integrator/CP is faster than Integrator/AP on aggregate

- ***Comparison between areas of memory and boards***

  This section summarizes the information of the memory access timing spreadsheet.

  - The core module SSRAM is the fastest memory device in Integrator. It is accessed in 1 or 2 bus cycles
  - Read access to the SDRAM DIMM is very slow for CM946ES, CM966ES, CM926EJS, CM10200 and CM10200E. If you need fast execution for a big program that does not fit in the SSRAM, you should either:
    - Use an Integrator/CP instead: this applies to CP946ES, CP966ES and CP926EJS; there is no CP10200E at present
    - Use the core module private SDRAM: this applies to CM10200E
  - Word access to the SDRAM DIMM is fast for CM7TDMI, CM740T, CM720T, CM940T, CM920T and CM920T-ETM, but sub-word access is very slow. You should always:
    - Enable caches when running code from SDRAM: This only applies to cached cores
    - Execute only ARM code (not Thumb) from SDRAM and avoid declaring half-word and byte variables in SDRAM if you are using a CM7TDMI
  - The motherboard Flash is slower than the core module SDRAM. If your code needs to boot from Flash, it should copy itself to SDRAM and continue running from SDRAM
  - The logic module SSRAM is accessed faster than the motherboard SSRAM (AP only)
  - Write accesses through the asynchronous bridge are very fast in CM7TDMI, CM740T, CM720T, CM940T, CM920T and CM920T-ETM because these have a write buffer in the asynchronous bridge. However, they are very slow in CM946ES, CM966ES, CM926EJS, CM10200 and CM10200E because this write buffer is not present in the asynchronous bridge on these modules

- ***At what frequency should I set up the clocks to achieve maximum performance?***
- ***What clock mode should I choose for my core module?***

Since our boards are not speed graded, the only way to achieve the maximum possible frequency is by experimentation. We recommend customers to begin to work with their boards configured at a safe frequency and then increment the frequency slowly until errors begin to be generated. The frequency should then be reduced a bit to achieve the maximum operating frequency.

Integrator/AP

In Integrator/AP the core module local bus and the system bus are connected by an asynchronous bridge, so their clock frequencies do not depend on each other. You should set up the maximum possible frequency for both the local bus and the system bus.

Integrator/CP

In Integrator/CP the same signal clocks the local bus and the system bus. You should set up this clock at the maximum possible frequency.

CM7TDMI

This core module has one clock for both the local bus and the core. You should set up this clock at the maximum frequency allowed by the test chip and the memory system.

CM720T, CM740T, CM920T, CM940T

These modules can be configured in either asynchronous or Fastbus mode. In general the maximum core speed of the test chips fitted on the core modules is considerably higher than the maximum local bus speed, so the best performance is achieved by setting both cores at their maximum values and configuring the processor in asynchronous mode.

CM922T-XA10

In the CM922T-XA10 the core and bus are clocked by the same signal (AHB1 clock), so the core should always be configured in Fastbus mode. The Boot Monitor configures the core by default in asynchronous mode, so this should be changed by the user's software.

CM946ES, CM966ES, CM926EJS

These modules only operate in synchronous mode, so the core clock frequency must be a multiple of the local bus clock frequency. This imposes an important restriction for the values at which you can program the clocks. In general you can only configure one clock at its maximum frequency at the expense of having the other clock running below its potential maximum frequency.

The optimum clock set up depends on your application. Memory intensive applications should have a fast local bus clock frequency whereas applications running mainly from cache should have a fast core clock frequency.

The only way to know which configuration is best is by benchmarking your application with different clock settings.

CM10200E

The clock architecture of the CM10200E is similar to the CM9x6ES. The main difference is that the CM10200E has an extra clock domain for the test chip SDRAM controller.

Normally applications would be run from private SDRAM, so the user must find a compromise between the core clock and internal clock frequencies.

- ***How can I maximize performance across the system bus bridge?***

When implementing a peripheral in a logic module, there are two alternatives:

1. Implement the peripheral as a slave. Whenever the slave needs to be read from or written to, it generates an interrupt to the ARM core. The logic module SSRAM can be used as a data buffer between the ARM core and the peripheral

2. Implement a master in the logic module that performs DMA operations between the peripheral and Integrator memory

The best solution depends on the Integrator platform and core module used.

Integrator/CP

Integrator/CP implements an AHB-Lite system bus, so there can only be a system bus master. For Integrator/CP only the first alternative can be implemented.

Integrator/AP with CM7TDMI, CM720T, CM740T, CM920T, CM940T and CM920T-ETM

These core modules have a write buffer in the asynchronous bridge, so normally write accesses to the system bus in both directions are performed in one or two cycles, whereas read accesses take several cycles. In order to achieve the best possible performance the second alternative (DMA) must be used, so there are only write transfers through the asynchronous bridge.

For transfers from the ARM core to the peripheral the logic module SSRAM should be used as a temporary buffer. The ARM core writes data through the asynchronous bridge to the logic module SSRAM in a single local bus cycle. The master in the logic module should be able to read from the logic module SSRAM via the system bus in two or three system bus cycles.

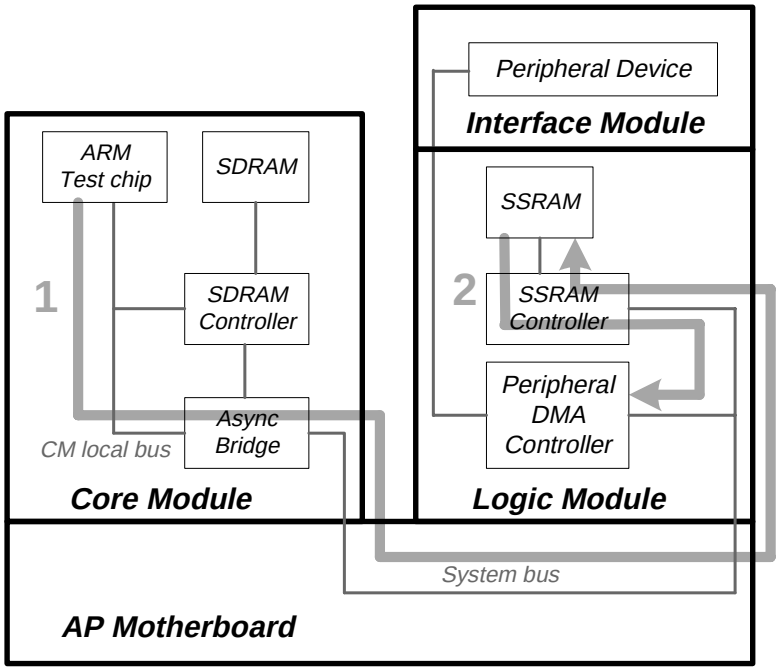

Figure 2: Transfers from the ARM core to the peripheral

For transfers from the peripheral to the ARM core the core module SDRAM should be used as a temporary buffer. The peripheral writes to the core module SDRAM at its alias location in a single cycle using the write buffer in the asynchronous bridge. The ARM core should read from SDRAM at its default location (e.g. address 0x00200000), which is done with a timing of 9-1-1-1 cycles for a burst of 4 words.

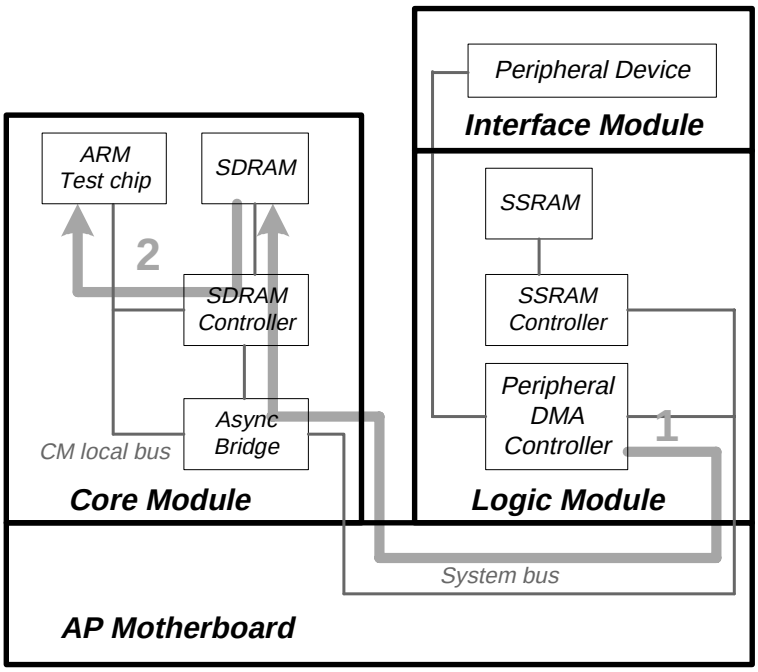

Figure 3: Transfers from the peripheral to the ARM core

Integrator/AP with CM946ES, CM966ES, CM926EJS, CM10200, CM10200E

These core modules do not have a write buffer in the asynchronous bridge, so read and write accesses to the system bus take a similar number of cycles.

In this case the second alternative does not offer any advantage and is more difficult to implement, so you should choose the first option (interrupt).