

Wrapper Latency in Excalibur Devices

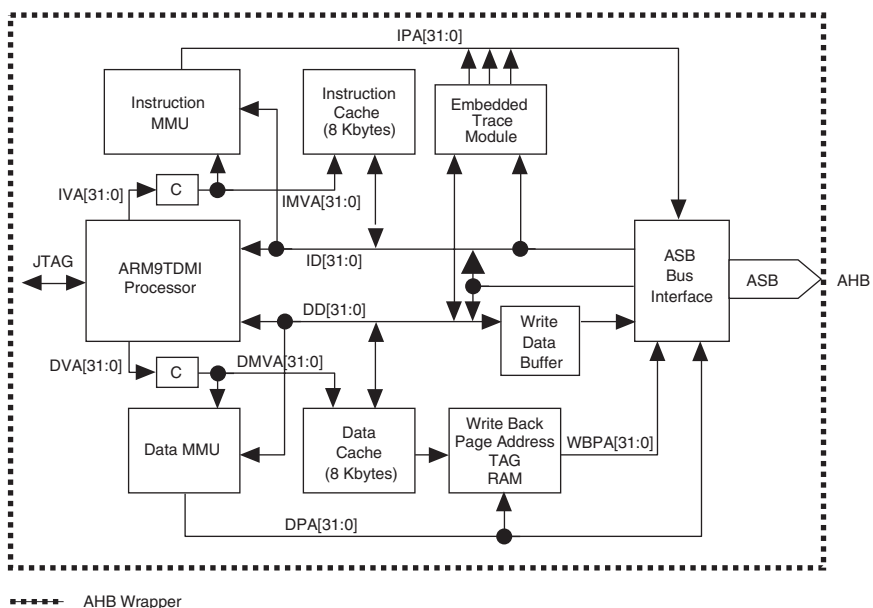
Introduction

This document provides details about the AHB wrapper that is used on the embedded stripe of Excalibur™ devices. It focuses specifically on the area of the latency introduced due to the instantiation of the AHB wrapper synthesized around the processor core.

Overview of the AHB Wrapper

The embedded stripe of an Excalibur device uses the ARM922TDMI™ as its processor core. The ARM922TDMI is a Harvard architecture processor core, implemented using a five-stage pipeline consisting of fetch, decode, execute, memory and writeback stages. The ARM922TDMI has separate 8-K instruction and data cache and memory management units to provide translation and access permission checks for instruction and data addresses. The internal organization of the ARM922TDMI is shown in Figure 1.

Figure 1. Internal Organization of the ARM922TDMI



The ARM922TDMI interfaces to the rest of the system over an advanced system bus (ASB), which is translated to AHB by a wrapper that interfaces with the Excalibur device AHB1 bus. See the *AMBA Specification* for details of AHB and ASB.

Latency of the Wrapper

The addition of the AHB wrapper adds an additional 3 or 4 clock cycle latency for access to resources outside of the processor core. Of this number, 2.5 or 3.5 clocks are needed for the transaction to propagate out of the wrapper, and 0.5 clocks are needed for the transaction response to propagate back through the wrapper.

The processor is clocked on an inverted version of the AHB1 clock, which accounts for the half-clock cycles in the latency figures.

SRAM Access Example

The following section demonstrates an access to SRAM. The example shows how the latency is distributed for a sequence of accesses to SRAM on AHB1.

Figure 2 shows a portion of the disassembly code for a test instruction sequence.

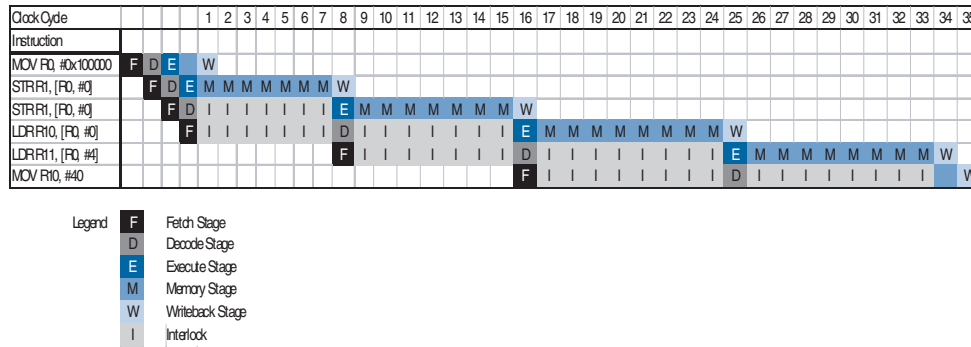
Figure 2. Disassembly of Code for a Test Instruction Sequence

Address	Instruction
0x0000005c:	MOV r0, #0x100000
0x00000060:	STR r1, [r0, #0]
0x00000064:	STR r2, [r0, #4]
0x00000068:	LDR r10, [r0, #0]
0x0000006c:	LDR r11, [r0, #4]
0x00000070:	MOV r10, #0x40

The test isolates the memory accesses, so it is locked into cache. The processor pipeline is not stalled until the first store instruction at address 0x60.

As mentioned above, the ARM922TDMI has a five-stage pipeline. Figure 3 on page 2 shows how the instructions flow through the pipeline.

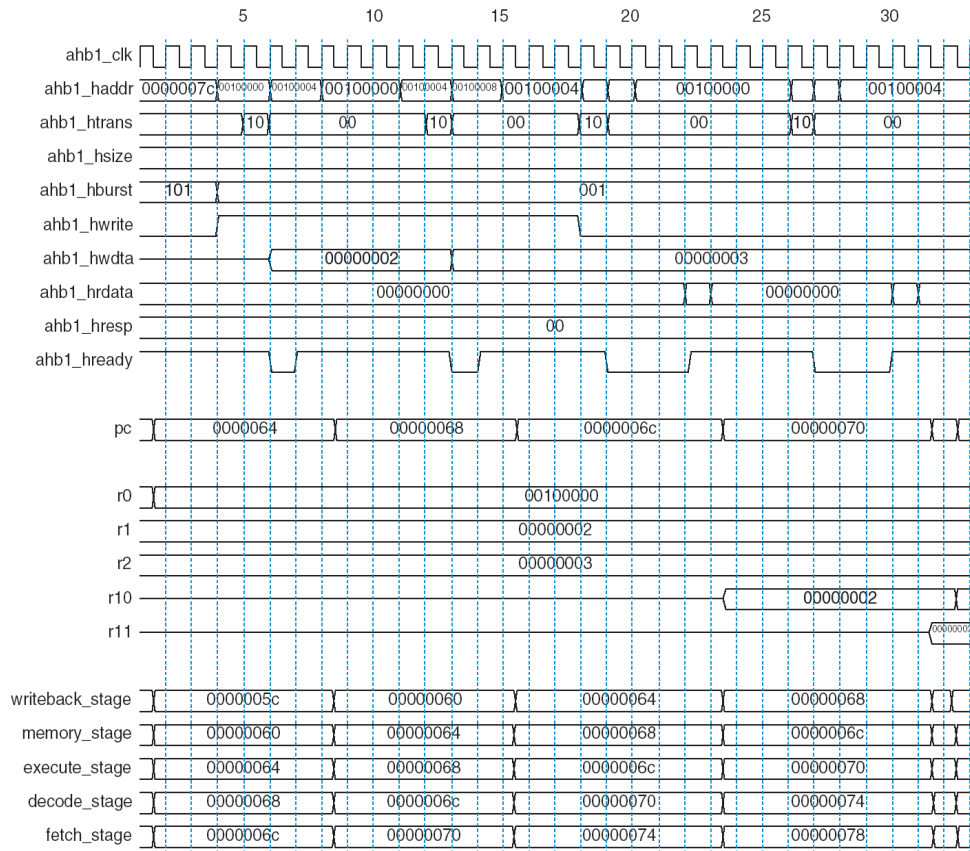
Figure 3. Instruction Flow through the ARM922TDMI Pipeline



In clock cycle 1, the processor enters the memory stage for the first store instruction. The access to memory extends for 7 clocks and the register is updated during clock cycle 8. In clock cycle 17, the memory stage for the first load instruction starts and extends to clock cycle 24, lasting a total of 8 clock cycles.

Figure 4 on page 3 shows a timing diagram of the instruction sequence.

Figure 4. Timing Diagram of the Instruction Sequence



The timing diagram shows AHB1 and some of the general purpose registers. It also shows the address of the instructions in the stages of the processor pipeline.

Clock cycles 1 to 8 demonstrate that the latency for access to SRAM is distributed as follows:

- Store instruction at address 0x60 (write to SRAM)
 - 3.5 clocks from the instruction entering the memory stage until the memory access appears on AHB1
 - 1 clock for the address phase of the transaction on AHB1
 - 2 clocks for the data phase of the transaction on AHB1 (one wait state inserted)
 - -0.5 clock for the processor to receive the transaction and progress the pipeline

Therefore the latency due to the wrapper is 4 clocks.

- Load instruction at address 0x68 (read from SRAM)
 - 2.5 clocks from the instruction entering the memory stage until the memory access appears on AHB1
 - 1 clock for the address phase of the transaction on AHB1
 - 4 clocks for the data phase of the transaction on AHB1 (three wait states inserted)
 - 0.5 clock for the processor to receive the transaction and progress the pipeline

Therefore the latency due to the wrapper is 3 clocks.

Minimizing the Impact of Wrapper Latency on System Performance

The example demonstrates that the wrapper adds three or four clocks to accesses to resources outside the processor core. To minimize the impact on system performance, the cache must be enabled. Enabling the cache increases performance by reducing the memory stage to one clock once data is in the cache. In addition, enabling caches forces the processor to use eight-beat bursts rather than single transactions to access resources outside the core. Table 1 summarizes the number of clocks to access SRAM both with and without cache enabled.

Table 1. Comparison Between Cached and Non-Cached SRAM Accesses

	AHB Burst Transaction	Latency/Beat	Clocks/Word	Total Clocks
Without Cache	Single	8-8-8-8-8-8-8-8	8	64
With Cache	INCR8	8-1-1-1-1-1-1-1	1.875	15

An eight-beat burst incurs the same amount of latency as a single transaction for the first beat. However, remaining beats do not incur any latency due to the wrapper, which substantially reduces the number of processor stalls during the memory stage and increases the overall system performance.

Conclusion

The ARM922TDMI processor core used in Excalibur devices uses an ASB to communicate with peripherals. The processor ASB is translated to AHB by a wrapper to work on AHB1 in the embedded stripe of Excalibur devices. The wrapper adds three clock cycles of latency for reads and four clock cycles of latency for writes to resources outside the processor core. To minimize the effect the latency on system performance, the cache must be enabled.



EXCALIBUR™



ALTERA®

101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
<http://www.altera.com>

Copyright © 2002 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.