Amber Open Source Project

Amber 2 Core Specification

March 2015

Table of Contents

1 Introduction

The Amber processor core is an ARM-compatible 32-bit RISC processor. The Amber core is fully compatible with the ARM® v2a instruction set architecture (ISA) and is therefore supported by the GNU toolset. This older version of the ARM instruction set is supported because it is not covered by patents so can be implemented without a license from ARM. The Amber project provides a complete embedded system incorporating the Amber core and a number of peripherals, including UARTs, timers and an Ethernet MAC.

There are two versions of the core provided in the Amber project. The Amber 23 has a 3-stage pipeline, a unified instruction & data cache, a Wishbone interface, and is capable of 0.8 DMIPS per MHz. The Amber 25 has a 5-stage pipeline, seperate data and instruction caches, a Wishbone interface, and is capable of 1.0 DMIPS per Mhz. Both cores implement exactly the same ISA and are 100% software compatible.

The Amber 23 core is a very small 32-bit core that provides good performance. Register based instructions execute in a single cycle, except for instructions involving multiplication. Load and store instructions require three cycles. The core's pipeline is stalled either when a cache miss occurs, or when the core performs a wishbone access.

The Amber 25 core is a little larger and provides 15% to 20% better performance that the 23 core. Register based instructions execute in a single cycle, except for instructions involving multiplication. Load and store instructions also execute in a single cycle unless there is a register conflict with a following instruction. The core's pipeline is stalled when a cache miss occurs in either cache, when an instruction conflict is detected, or when the core performs a wishbone access.

Both cores has been verified by booting a 2.4 Linux kernel. Versions of the Linux kernel from the 2.4 branch and earlier contain configurations for the supported ISA. The 2.6 version of Linux does not explicitly support the ARM v2a ISA so requires more modifications to run. Also note that the cores do not contain a memory management unit (MMU) so they can only run the non-virtual memory variant of Linux.

The cores were developed in Verilog 2001, and are optimized for FPGA synthesis. For example there is no reset logic, all registers are reset as part of FPGA initialization. The complete system has been tested extensively on the Xilinx SP605 Spartan-6 FPGA board. The full Amber system with the A23 core uses 32% of the Spartan-6 XC6SLX45T-3 FPGA Look Up Tables (LUTs), with the core itself occupying less than 20% of the device using the default configuration, and running at 40MHz. It has also been synthesized to a Virtex-6 device at 80MHz, but not yet tested on a real Virtex-6 device. The maximum frequency is limited by the execution stage of the pipline which includes a 32-bit barrel shifter, 32-bit ALU and address incrementing logic.

For a description of the ISA, see "Archimedes Operating System - A Dabhand Guide, Copyright Dabs Press 1991", or "Acorn RISC Machine Family Data Manual, VLSI

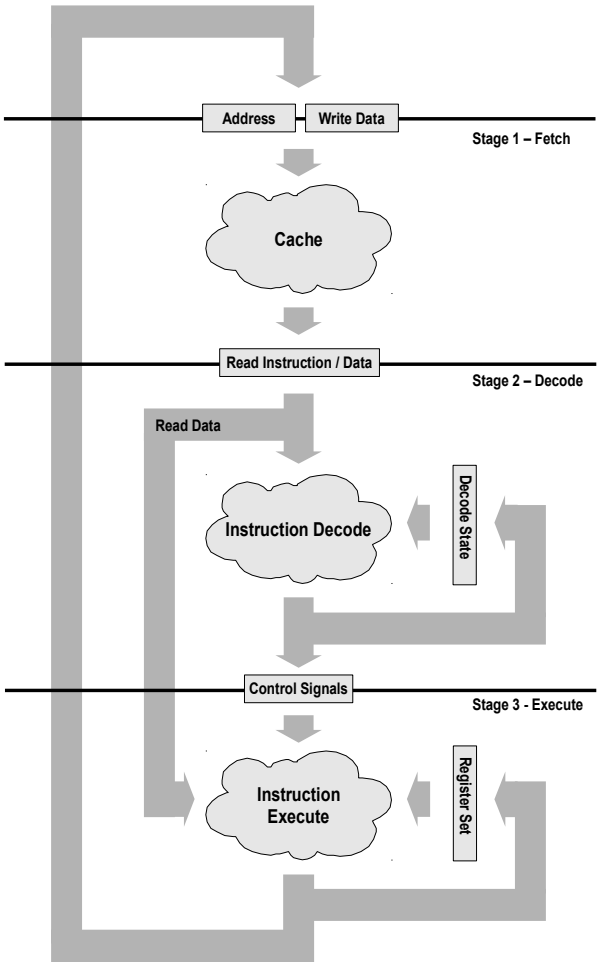
Technology Inc., 1990".

1.1 Amber 23 Features

- 3-stage pipeline.
- 32-bit Wishbone system bus.
- Unified instruction and data cache, with write through and a read-miss replacement policy. The cache can have 2, 3, 4 or 8 ways and each way is 4kB.
- Multiply and multiply-accumulate operations with 32-bit inputs and 32-bit output in 34 clock cycles using the Booth algorithm. This is a small and slow multiplier implementation.
- Little endian only, i.e. Byte 0 is stored in bits 7:0 and byte 3 in bits 31:24.

The following diagram shows the data flow through the 3-stage core.

Figure 1 - *Amber 23 Core pipeline stages*



1.2 Amber 25 Features

- 5-stage pipeline.
- 32-bit Wishbone system bus.
- Seperate instruction and data caches. Each cache can be either 2,3,4 or 8 ways and each way is 4kB. Both caches use a read replacement policy and the data

cache operates as write through. The instruction cache is read only.

- Multiply and multiply-accumulate operations with 32-bit inputs and 32-bit output in 34 clock cycles using the Booth algorithm. This is a small and slow multiplier implementation.
- Little endian only, i.e. Byte 0 is stored in bits 7:0 and byte 3 in bits 31:24.

The following diagram shows the data flow through the 5-stage core.

Figure 2 - *Amber 25 Core pipeline stages*