8 Amber Project

The Amber project is a complete processor system implemented on an FPGA development board. The purpose of the project is to provide an evironment that gives an example usage of the Amber 2 core, and supports a set of tests that verify the correct functionality of the code. This is especially important if modificatiosn to the core are made.

8.1 Amber Port List

The following table gives the port list for the Amber 2x core. The Amber 23 and Amber 25 cores have identical port lists.

Table 17 Amber 2x Core Port List

Name	Width	Direction	Description
i_clk	1	in	Clock input. The core only has a single clock. The Wishbone interface also works on this clock.
i_irq	1	in	Interrupt request, active high. Causes the core to switch to IRQ mode and jump to the IRQ address vector when asserted. The switch does not occur until the end of the current instruction. For example if the core is executing a stm instruction it could take 40 or 50 cycles to complete this instruction. Once the instruction has completed the core will jump to the IRQ vector and execute the instruction at that location.
i_firq	1	in	Fast Interrupt request, active high. Causes the core to switch to FIRQ mode and jump to the FIRQ address vector when asserted. Again the core makes the switch after the current instruction has completed.
i_system_rdy	1	in	Connected to the stall signal that stalls the decode and execute stages of the core. The system uses this signal to freeze the core until the DDR3 main memory initialization has completed.
Wishbone Interface			
o_wb_adr	32	out	Byte address. Note that the core only generates 26-bit instruction addresses but can generate full 32-bit data addresses.
o_wb_sel	4	out	Byte enable for writes. Bit 0 corresponds to byte 0 which is bits [7:0] on the data buses.
o_wb_we	1	out	Write enable, active high.
i_wb_dat	32	in	Read data. Active when i_wb_ack is asserted in a read cycle.
o_wb_dat	32	out	Write data. Active when o_wb_stb is high.
o_wb_cyc	1	out	Holds bus ownership during multi-cycle accesses.
o_wb_stb	1	out	Per-cycle strobe.
i_wb_ack	1	in	Used to terminate read and write accesses.
i_wb_err	1	in	Used to indicate an error on an access. Currently not used within the Amber 2 core.

8.2 Amber 23 Verilog Files

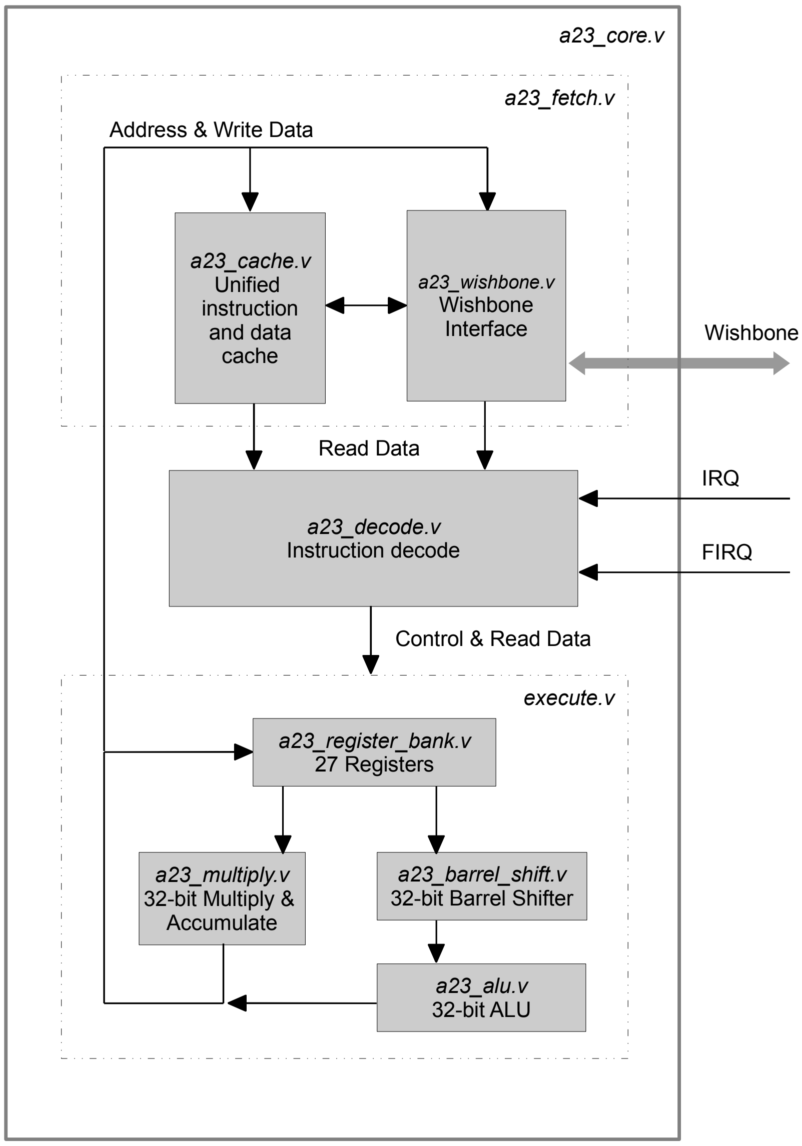
The following table describes each Verilog source file in the Amber 2 core. These files are located in $AMBER_BASE/hw/vlog/amber.

Table 18 Amber 23 Core Source Files

Name	Description
a23_config_definesv	Defines used to configure the amber core. The number of ways in the cache is configurable. Also contains a set of debug switches which enable debug messages to be printed during simulation.
a23_localparams.v	Local parameters used in various amber source files.
a23_wishbone.v	The Wishbone interface connecting the Execute stage and Cache to the rest of the system. Instantiated in Fetch.
a23_alu.v	The arithmetic logic unit. Includes a 32-bit 2's compliment adder/subtractor as well as logical functions such as AND and XOR.
a23_functions.v	Common Verilog functions.
a23_core.v	Top-level Amber module.
a23_barrel_shifter.v	32-bit barrel shifter instantiated in Execute.
a23_cache.v	Synthesizable cache. Instantiated in Fetch. Cache misses cause the core to stall. The cache then issues a quad-word read on the wishbone bus, starting with the word that missed, and wrapping at the quad-word boundary.
a23_coprocessor.v	Co-processor 15 registers and control signals. Instantiated in Amber.
a23_decode.v	The instruction decode pipeline stage. Instantiated in Amber.
a23_decompile.v	The decompiler. This is a non-synthesizable debug module. It creates the amber.dis file which lists every instruction executed by the core.
a23_execute.v	The execute pipeline stage. Instantiated in Amber. It contains the alu, multiply, and register_bank sub-modules.
a23_fetch.v	The Fetch stage. This contains the Cache and Wishbone interface modules. It is instantiated in Amber.
a23_multiply.v	32-bit 2's compliment multiply and multiply-accumulate unit. Uses the Booth algorithm and takes 34 cycles to complete a signed multiply-accumulate operation but is quite small in logic area.
a23_register_bank.v	Contains all 27 registers r0 to r15 for each mode of operation. Registers are implemented as real flipflops in the FPGA. This allows multiple read and write access to the bank simultaneously.

The following diagram shows the Verilog module structure within the Amber 2 core.

Figure 5 - *Amber 23 Core Verilog Structure*


a23_core.v
a23_fetch.v
Address & Write Data
a23_cache.v Unified instruction and data cache
a23_wishbone.v Wishbone Interface
Wishbone I/F
Read Data
a23_decode.v Instruction decode
IRQ
FIRQ
Control & Read Data
execute.v
a23_register_bank.v 27 Registers
a23_multiply.v 32-bit Multiply & Accumulate
a23_barrel_shift.v 32-bit Barrel Shifter
a23_alu.v 32-bit ALU