ao68000 Specification

Author: Aleksander Osman

alfik@poczta.fm

Rev. 1.1

December 11, 2010

This page has been intentionally left blank.

Revision History

Rev.	Date	Author	Description
1.0	28.03.2010	Aleksander Osman	First Draft
1.1	11.12.2010	Aleksander Osman	DBcc opcode microcode fix. Wishbone SEL signal fix. Project directory structure simplification.

Contents

1. Introduction

The OpenCores ao68000 IP Core is a Motorola MC68000 compatible processor.

Features

- CISC processor with microcode,
- WISHBONE revision B.3 compatible MASTER interface,
- Not cycle exact with the MC68000, some instructions take more cycles to complete, some less,
- Uses about 7500 LE on Altera Cyclone II and about 45000 bits of RAM for microcode,
- Tested against the WinUAE M68000 software emulator. Every 16-bit instruction was tested with random register contents and RAM contents (Processor verification). The result of execution was compared,
- Runs Linux kernel version 2.6.33.1 up to init process lookup (System-on-Chip example with ao68000 running Linux),
- Contains a simple prefetch which is capable of holding up to 5 16-bit instruction words,
- Documentation generated by Doxygen (www.doxygen.org) with doxverilog patch (http://developer.berlios.de/projects/doxverilog/). The specification is automatically extracted from the Doxygen HTML output.

WISHBONE compatibility

- Version: WISHBONE specification Revision B.3,
- General description: 32-bit WISHBONE Master interface,
- WISHBONE signals described in IO Ports,

- Supported cycles: Master Read/Write, Master Block Read/Write, Master Read-Modify-Write for TAS instruction, Register Feedback Bus Cycles as described in chapter 4 of the WISHBONE specification,
- Use of ERR_I: on memory access bus error, on interrupt acknowledge: spurious interrupt,
- Use of RTY_I: on memory access repeat access, on interrupt acknowledge: generate auto-vector,
- WISHBONE data port size: 32-bit,
- Data port granularity: 8-bits,
- Data port maximum operand size: 32-bits,
- Data transfer ordering: BIG ENDIAN,
- Data transfer sequencing: UNDEFINED,
- Constraints on `CLK_I` signal: described in Clocks, maximum frequency: about 70 MHz.

Use

The ao68000 can be used as an processor in a System-on-Chip booting Linux kernel up to `init` program lookup (System-on-Chip example with ao68000 running Linux).

Similar projects

Other free soft-core implementations of M68000 microprocessor include:

- OpenCores TG68 (http://www.opencores.org/project,tg68) - runs Amiga software, used as part of the Minimig Core,
- Suska Atari VHDL WF_68K00_IP Core (http://www.experiment-s.de/en) - runs Atari software,
- OpenCores K68 (http://www.opencores.org/project,k68) - no user and supervisor modes distinction, executes most instructions, but not all.
- OpenCores ae68 (http://www.opencores.org/project,ae68) - no files uploaded as of 27.03.2010.

Limitations

- Microcode not optimized: some instructions take more cycles to execute than the original MC68000,
- TRACE not tested,
- The core is large compared to other implementations.

TODO

- Optimize the microcode and count the exact cycle count for every instruction,
- Test TRACE,
- Run WISHBONE verification models,
- More documentation of the ao68000 module: signal description, operation, FSM in `bus_control`,
- Describe changes done in WinUAE sources (copy from ao.c),
- Describe microcode words and subprocedures,
- Document the `soc_for_linux` modules,
- Prepare scripts for VATS: run_sim -r -> regresion test,
- Use memories from OpenCore common.

Status

- Tested with WinUAE software MC68000 emulator,
- Booted Linux kernel up to `init` process lookup.

Requirements

- Icarus Verilog simulator (http://www.icarus.com/eda/verilog/) is required to compile the `tb_ao68000` testbench/wrapper,
- Access to Altera Quartus II instalation directory (directory eda/sim_lib/) is required to compile the `tb_ao68000` testbench/wrapper,
- GCC (http://gcc.gnu.org) is required to compile the WinUAE MC68000 software emulator,
- Java runtime (http://java.sun.com) is required to run the `ao68000_tool` (ao68000_tool documentation),
- Java SDK (http://java.sun.com) is required to compile the `ao68000_tool` (ao68000_tool documentation),
- Altera Quartus II synthesis tool (http://www.altera.com) is required to synthesise the `soc_for_linux` System-on-Chip (System-on-Chip example with ao68000 running Linux).

Glossary

- **ao68000** - the ao68000 IP Core processor,

- **MC68000** - the original Motorola MC68000 processor.

2. Architecture

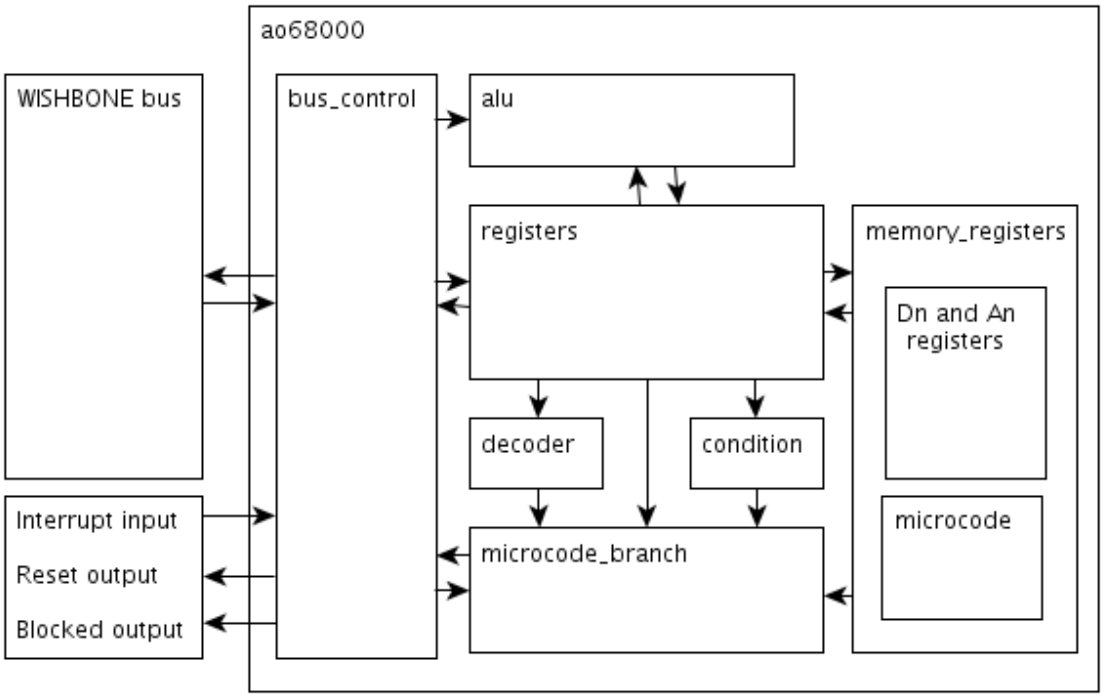

Figure 1: Simplified block diagram of ao68000 top module.

ao68000

ao68000 top level module.

This module contains only instantiations of sub-modules and wire declarations.

bus_control

Initiate WISHBONE MASTER bus cycles.

The bus_control module is the only module that has contact with signals from outside of the IP core. It is responsible for initiating WISHBONE MASTER bus cycles. The cycles can be divided into:

- memory read cycles (supervisor data, supervisor program, user data, user program)
- memory write cycles (supervisor data, user data),
- interrupt acknowledge.

Every cycle is supplemented with the following tags:

- standard WISHBONE cycle tags: SGL_O, BLK_O, RMW_O,
- register feedback WISHBONE address tags: CTI_O and BTE_O,
- ao68000 specific cycle tag: fc_o which is equivalent to MC68000 function codes.

The bus_control module is also responsible for registering interrupt inputs and initiating the interrupt acknowledge cycle in response to a microcode request. Microcode requests a interrupt acknowledge at the end of instruction processing, when the interrupt privilege level is higher than the current interrupt privilege mask, as specified in the MC68000 User's Manual.

Finally, bus_control controls also two ao68000 specific core outputs:

- blocked output, high when that the processor is blocked after encountering a double bus error. The only way to leave this block state is by reseting the ao68000 by the asynchronous reset input signal.
- reset output, high when processing the RESET instruction. Can be used to reset external devices.

registers

Microcode controlled registers.

Most of the ao68000 IP core registers are located in this module. At every clock cycle the microcode controls what to save into these registers. Some of the more important registers include:

- operand1, operand2 registers are inputs to the ALU,
- address, size, do_read_flag, do_write_flag, do_interrupt_flag registers tell the bus_control module what kind of bus cycle to perform,
- pc register stores the current program counter,
- ir register stores the current instruction word,
- ea_mod, ea_type registers store the currently selected addressing mode.

memory_registers

Contains the microcode ROM and D0-D7, A0-A7 registers.

The memory_registers module contains:

- data and address registers (D0-D7, A0-A7) implemented as an on-chip RAM.
- the microcode implemented as an on-chip ROM.

Currently this module contains *altsyncram* instantiations from Altera Megafunction/LPM library.

decoder

Decode instruction and addressing mode.

The decoder is an instruction and addressing mode decoder. For instructions it takes as input the ir register from the registers module. The output of the decoder, in this case, is a microcode address of the first microcode word that performs the instruction.

In case of addressing mode decoding, the output is the address of the first microcode word that performs the operand loading or saving. This address is obtained from the currently selected addressing mode saved in the ea_mod and ea_type registers in the registers module.

condition

Condition tests.

The condition module implements the condition tests of the MC68000. Its inputs are the condition codes and the currently selected test. The output is binary: the test is true or false. The output of the condition module is an input to the microcode_branch module, that decides which microcode word to execute next.

alu

Arithmetic and Logic Unit.

The alu module is responsible for performing all of the arithmetic and logic operations of the ao68000 processor. It operates on two 32-bit registers: operand1 and operand2 from the registers module. The output is saved into a result 32-bit register. This register is located in the alu module.

The alu module also contains the status register (SR) with the condition code register. The microcode decides what operation the alu performs.

microcode_branch

Select the next microcode word to execute.

The microcode_branch module is responsible for selecting the next microcode word to execute. This decision is based on the value of the current microcode word, the value of the interrupt privilege level, the state of the current bus cycle and other internal signals.

The microcode_branch module implements a simple stack for the microcode addresses. This makes it possible to call subroutines inside the microcode.

3. Operation

The ao68000 IP Core is designed to operate in a similar way as the original MC68000. The most import differences are:

- the core IO ports are compatible with the WISHBONE specification,
- the execution of instructions in the ao68000 core is not cycle-exact with the original MC68000 and usually takes a few cycles longer.

Setting up the core

The ao68000 IP Core has an WISHBONE MASTER interface. All standard memory access bus cycles conform to the WISHBONE specification. These cycles include:

- instruction fetch,
- data read,
- data write.

The cycles are either Single, Block or Read-Modify-Write (for the TAS instruction). When waiting to finish a bus cycle the ao68000 reacts on the following input signals:

- ACK_I: the cycle is completed successfully,
- RTY_I: the cycle is immediately repeated, the processor does not continue its operation before the current bus cycle is finished. In case of the Read-Modify-Write cycle - only the current bus cycle is repeated: either the read or write.
- ERR_I: the cycle is terminated and a bus error is processed. In case of double bus error the processor enters the blocked state.

There is also a special bus cycle: the interrupt acknowledge cycle. This cycle is a reaction on receiving a external interrupt from the ipl_i inputs. The processor only samples the ipl_i lines after processing an instruction, so the interrupt lines have to be asserted for

some time before the core reacts. The interrupt acknowledge cycle is performed in the following way:

- ADR_O is set to { 27'b111_1111_1111_1111_1111_1111_1111, 3 bits indicating the interrupt priority level for this cycle },
- SEL_O is set to 4'b1111,
- fc_o is set to 3'b111 to indicate a CPU Cycle as in the original MC68000.

The ao68000 reacts on the following signals when waiting to finish a interrupt acknowledge bus cycle:

- ACK_I: the cycle is completed successfully and the interrupt vector is read from DAT_I[7:0],
- RTY_I: the cycle is completed successfully and the processor generates a auto-vector internally,
- ERR_I: the cycle is terminated and the processor starts processing a spurious interrupt exception.

Every bus cycle is supplemented with output tags:

- WISHBONE standard tags: SGL_O, BLK_O, RMW_O, CTI_O, BTE_O,
- ao68000 custom tag: fc_o that operates like the Function Code of the original MC68000.

The ao68000 core has two additional outputs that are used to indicate the state of the processor:

- reset_o is a external device reset signal. It is asserted when processing the RESET instruction. It is asserted for 124 bus cycles. After that the processor returns to normal instruction processing.
- blocked_o is an output that indicates that the processor is blocked after a double bus error. When this output line is asserted the processor is blocked and does not process any instructions. The only way to continue processing instructions is to reset the core.

Resetting the core

The ao68000 core is reset with a asynchronous reset_n input. After deasserting the signal, the core starts its standard startup sequence, which is similar to the one performed by the original MC68000:

- the value of the SSP register is read from address 0,
- the value of the PC is read from address 1.

An identical sequence is performed when powering up the core for the first time.

Processor modes

The ao68000 core has two modes of operation - exactly like the original MC68000:

- Supervisor mode
- User mode.

Performing a privileged instruction when running in user mode results in a privilege exception, just like in MC68000.

Processor states

The ao68000 core can be in one of the following states:

- instruction processing, which includes group 2 exception processing,
- group 0 and group 1 exception processing,
- external device reset state when processing the RESET instruction,
- blocked state after a double bus error.

4. Registers

The ao68000 IP Core is a WISHBONE Master and does not contain any registers available for reading or writing from outside of the core.

5. Clocks

Name	Source	Rates (MHz)			Remarks	Description
		Max	Min	Resolution		
CLK_I	Input Port	70	-	-	-	System clock.

Table 1: List of clocks.

6.

IO Ports

WISHBONE IO Ports

Port	Width	Direction	Description
CLK_I	1	Input	WISHBONE Clock Input
reset_n	1	Input	Asynchronous Reset Input
CYC_O	1	Output	WISHBONE Master Cycle Output
ADR_O	30	Output	WISHBONE Master Address Output
DAT_O	32	Output	WISHBONE Master Data Output
DAT_I	32	Input	WISHBONE Master Data Input
SEL_O	4	Output	WISHBONE Master Byte Select
STB_O	1	Output	WISHBONE Master Strobe Output
WE_O	1	Output	WISHBONE Master Write Enable Output
ACK_I	1	Input	WISHBONE Master Acknowledge Input: • on normal cycle: acknowledge, • on interrupt acknowledge cycle: external vector provided on DAT_I[7:0].
ERR_I	1	Input	WISHBONE Master Error Input • on normal cycle: bus error, • on interrupt acknowledge cycle: spurious interrupt.

RTY_I	1	Input	WISHBONE Master Retry Input • on normal cycle: retry bus cycle, • on interrupt acknowledge: use auto-vector.
SGL_O	1	Output	WISHBONE Cycle Tag, TAG_TYPE: TGC_O, Single Bus Cycle.
BLK_O	1	Output	WISHBONE Cycle Tag, TAG_TYPE: TGC_O, Block Bus Cycle.
RMW_O	1	Output	WISHBONE Cycle Tag, TAG_TYPE: TGC_O, Read-Modify-Write Cycle.
CTI_O	3	Output	WISHBONE Address Tag, TAG_TYPE: TGA_O, Cycle Type Identifier, Incrementing Bus Cycle or End-of-Burst Cycle.
BTE_O	2	Output	WISHBONE Address Tag, TAG_TYPE: TGA_O, Burst Type Extension, always Linear Burst.
fc_o	3	Output	Custom TAG_TYPE: TGC_O, Cycle Tag, Processor Function Code: • 1 - user data, • 2 - user program, • 5 - supervisor data : all exception vector entries except reset, • 6 - supervisor program : exception vector for reset, • 7 - cpu space: interrupt acknowledge.

Table 1: List of WISHBONE IO ports.

Other IO Ports

Port	Width	Direction	Description
ipl_i	3	Input	Interrupt Priority Level Interrupt acknowledge cycle: • ACK_I: interrupt vector on DAT_I[7:0], • ERR_I: spurious interrupt, • RTY_I: auto-vector.
reset_o	1	Output	External device reset. Output high when processing the RESET instruction.
blocked_o	1	Output	Processor blocked indicator. The processor is blocked after

a double bus error.

Table 2: List of Other IO ports.

7.

References

1. *Specification for the: WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores.*
   Revision: B.3.
   Released: September 7, 2002.
   Available from: http://www.opencores.org.

2. *M68000 8-/16-/32-Bit Microprocessors Userâ€™s Manual.*
   Ninth Edition.
   Freescale Semiconductor, Inc.
   Available from: http://www.freescale.com.

3. *MOTOROLA M68000 FAMILY Programmerâ€™s Reference Manual (Includes CPU32 Instructions).*
   MOTOROLA INC., 1992. M68000PM/AD REV.1.
   Available form: http://www.freescale.com.

4. *ao68000 Doxygen(Design) Documentation.*