aoOCS Specification

Author: Aleksander Osman

alfik@poczta.fm

Rev. 1.0

December 20, 2010

This page has been intentionally left blank.

Revision History

Rev.	Date	Author	Description
1.0	20.12.2010	Aleksander Osman	First Publish

Contents

1. Introduction

The OpenCores aoOCS SoC is a Wishbone compatible implementation of most of the Amiga Original Chip Set (OCS) and computer functionality.

aoOCS is not related in any way with Minimig - it is a new and independent Amiga OCS implementation.

Features

- The aoOCS SoC contains the following Amiga/OCS components:
  - blitter
  - copper
  - system control (interrupts)
  - video: bitplains, sprites, collision detection
  - audio: 4 channels, low-pass filter
  - user input: PS/2 mouse, PS/2 keyboard and joystick (keyboard arrow keys)
  - floppy: read and write ADF files directly from a SD card. Only the internal floppy drive is implemented
  - 8520 CIA
  - ao68000 OpenCores IP core is used as the aoOCS processor
- All of the above components are WISHBONE revision B.3 compatible
- The aoOCS contains the following additional components:
  - SD card controller written in HDL with DMA. Supports SDHC cards only.
  - 10/100 Mbit Ethernet controller written in HDL to send the current VGA frames (frame grabber)
  - HDL drivers for SSRAM, PS/2 keyboard, PS/2 mouse, audio codec, VGA DAC

- aoOCS uses only one external memory: a SSRAM with 36-bit words and pipelined access. A video buffer with about 250KB is located SSRAM. Another 256KB are used by the ROM. All the rest memory can be used as Chip RAM.
- The On-Screen-Display is implemented in HDL as a finite state machine. No additional controller/processor with firmware required to handle the SoC.
- The following options are available on the On-Screen-Display:
    - select ROM file to load (only Amiga Kickstart v1.2 was tested)
    - enable or disable Joystick (keyboard arrow keys)
    - enable or disable floppy write protection
    - insert a floppy - select one from a list
    - eject an inserted floppy
    - reset the system
- The On-Screen-Display is independent of the running Amiga software. It is enabled and disabled by the Home key and controled by the keyboard arrow keys and the right CTRL key.
- Only PAL timings are implemented.
- The video output is VGA compatible: 640x480 at 70 Hz. A rather simple method is used to extend the 256 PAL horizontal lines to 480 VGA lines: all lines are doubled except for every 8th one.
- The system uses generally a single clock: 30 MHz. There are two more clocks: 12 MHz, 25 MHz generated to interface with external hardware (Audio codec, Ethernet controller). A single altpll is used to generate all three clocks from one 50 MHz external clock. More information about clocks is available at Clocks.
- A VGA frame grabber is implemented that sends captured frames by 100 Mbit Ethernet in IP/UDP packets.
- The system uses about 26.400 LE on Altera Cyclone II and about 267.000 bits of on-chip RAM.
- The blitter functionality was tested against the E-UAE Amiga software emulator.
- Tested only on a Terasic DE2-70 board (www.terasic.com.tw).
- 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 interface,
- 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.

Similar projects

Other Open-Source Amiga implementations include:

- Minimig (http://code.google.com/p/minimig/) - FPGA-based re-implementation of the original Amiga 500 hardware. Runs on the Minimig PCB and also on Terasic DE1,2 boards.

Limitations

- No filesystem support on the SD card. Data is read from fixed positions. The contents of the SD card is generated by the `aoOCS_tool` described at Operation.
- No video external synchronize, lace mode, lightpen, genlock audio enable, color composite (BPLCON0)
- All bitplain data is fetched at once in a burst memory read at the begining of each line. No changes to the bitplain data done after the beginning of a line are visible.
- Currently aoOCS requires an 36-bit word SSRAM to store the video buffer. This way 3 pixels 12-bits each can be stored in one word.
- Serial port not implemented.
- Parallel port not implemented.
- Low-pass filter disable/enable by CIA-A port A bit 1 not implemented.
- Proportional controller and light pen not implemented.
- Some rarely used OCS registers are not implemented: strobe video sync, write beam position, coprocessor instruction fetch identify. For a complete list of not implemented registers look at Registers.
- Only some of the Amiga software was tested and works on the aoOCS. A list of aoOCS software compatability is located at Operation.

TODO

- Fix some of the above limitations.
- Optimize the design.
- Run WISHBONE verification models.
- More documentation of Verilog sources.
- Describe testing and changes done in E-UAE sources.

- Prepare scripts for VATS: run_sim -r -> regresion test.
- Port the aoOCS SoC to a Xilinx FPGA.

Status

- Amiga Workbench v1.2 runs with some minor graphic problems: bottom of screen not displayed correctly.
- Prince of Persia runs perfectly.
- Wings of Fury runs correctly. Some sound glitches in intro.
- Lotus 2 runs correctly. Some sound problems in intro.
- Warzone runs poor. Some major graphic problems.
- More information about aoOCS software compatability is available at Operation.

Requirements

- Altera Quartus II synthesis tool (http://www.altera.com) is required to synthesise the `aoOCS` System-on-Chip.
- Java SDK (http://java.sun.com) is required to compile the `aoOCS_tool` (The tool is described in Operation).
- A FPGA board. Currently only the Terasic DE2-70 board was tested.
- Icarus Verilog simulator (http://www.icarus.com/eda/verilog/) is required to compile the and run some tests.
- Access to Altera Quartus II directory (directory eda/sim_lib/) is required to compile and run some tests.
- GCC (http://gcc.gnu.org) is required to compile some testes based on E-UAE sources.

2. Architecture

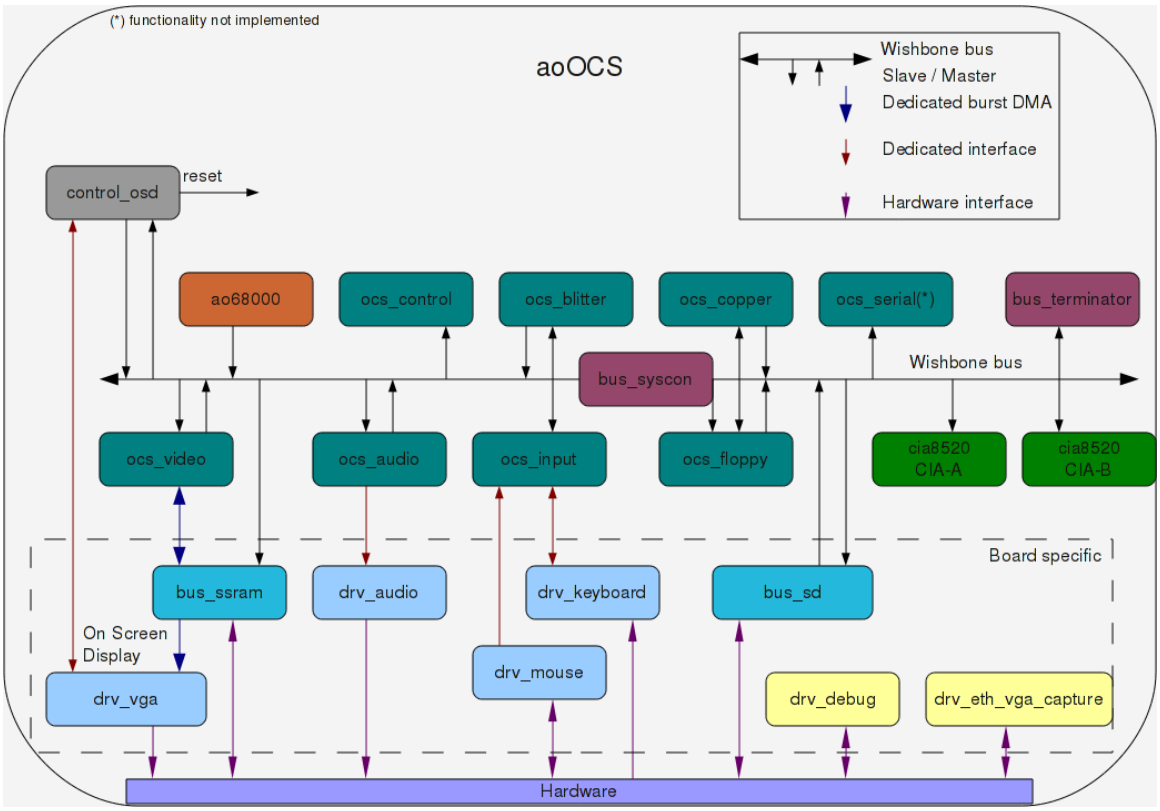


Figure 1: aoOCS structure.

control_osd

On-Screen-Display and overall system management.

ao68000

ao68000 top level module.

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

ocs_control

OCS system control implementation with WISHBONE slave interface.

List of system control registers:

```
Implemented:
     [DDFSTOP      094  W   A     Display bitplane data fetch stop
                                  (horiz. position)                          write not implemented here]
    DMACON       096  W   ADP     DMA control write (clear or set)           write not implemented here

    DMACONR     *002  R   AP      DMA control (and blitter status) read
    VPOSR       *004  R   A( E )  Read vert most signif. bit (and frame flop)
    VHPOSR      *006  R   A       Read vert and horiz. position of beam

    ADKCON       09E  W   P       Audio, disk, UART control

    ADKCONR     *010  R   P       Audio, disk control register read
     [POT0DAT   *012  R   P( E )  Pot counter pair 0 data (vert,horiz)        read implemented here]

    INTENAR     *01C  R   P       Interrupt enable bits read
    INTREQR     *01E  R   P       Interrupt request bits read

     [CLXCON     098  W   D       Collision control                          write not implemented here]
    INTENA       09A  W   P       Interrupt enable bits (clear or set bits)  write not implemented here
    INTREQ       09C  W   P       Interrupt request bits (clear or set bits)

Not implemented:
    REFPTR    & *028  W   A       Refresh pointer
    VPOSW       *02A  W   A       Write vert most signif. bit (and frame flop)
    VHPOSW      *02C  W   A       Write vert and horiz position of beam

    STREQU    & *038  S   D       Strobe for horiz sync with VB and EQU
    STRVBL    & *03A  S   D       Strobe for horiz sync with VB (vert. blank)
    STRHOR    & *03C  S   DP      Strobe for horiz sync
    STRLONG   & *03E  S   D( E )  Strobe for identification of long horiz. line.

    RESERVED     1110X
    RESERVED     1111X
    NO-OP(NULL)  1FE
```

ocs_blitter

OCS blitter implementation with WISHBONE master and slave interface.

List of blitter registers:

```
Implemented:
   BLTCON0     ~040  W   A       Blitter control register 0
   BLTCON1     ~042  W   A( E )  Blitter control register 1
   BLTAFWM     ~044  W   A       Blitter first word mask for source A
   BLTALWM     ~046  W   A       Blitter last word mask for source A
   BLTCPTH   + ~048  W   A       Blitter pointer to source C (high 3 bits)
   BLTCPTL   + ~04A  W   A       Blitter pointer to source C (low 15 bits)
   BLTBPTH   + ~04C  W   A       Blitter pointer to source B (high 3 bits)
   BLTBPTL   + ~04E  W   A       Blitter pointer to source B (low 15 bits)
   BLTAPTH   + ~050  W   A( E )  Blitter pointer to source A (high 3 bits)
   BLTAPTL   + ~052  W   A       Blitter pointer to source A (low 15 bits)
   BLTDPTH   + ~054  W   A       Blitter pointer to destination D (high 3 bits)
   BLTDPTL   + ~056  W   A       Blitter pointer to destination D (low 15 bits)
   BLTSIZE     ~058  W   A       Blitter start and size (window width,height)
   BLTCMOD     ~060  W   A       Blitter modulo for source C
   BLTBMOD     ~062  W   A       Blitter modulo for source B
   BLTAMOD     ~064  W   A       Blitter modulo for source A
   BLTDMOD     ~066  W   A       Blitter modulo for destination D
   BLTCDAT   % ~070  W   A       Blitter source C data register
   BLTBDAT   % ~072  W   A       Blitter source B data register
   BLTADAT   % ~074  W   A       Blitter source A data register
Not implemented:
   BLTDDAT   & *000  ER  A       Blitter destination early read (dummy address)
```

ocs_copper

OCS copper implementation with WISHBONE master and slave interface.

List of copper registers:

```
Implemented:
    COPCON      *02E  W   A( E )  Coprocessor control register (CDANG)
    COP1LCH   +  080  W   A( E )  Coprocessor first location register (high 3 bits, high 5 bits if ECS)
    COP1LCL   +  082  W   A       Coprocessor first location register (low 15 bits)
    COP2LCH   +  084  W   A( E )  Coprocessor second location register (high 3 bits, high 5 bits if ECS)
    COP2LCL   +  086  W   A       Coprocessor second location register (low 15 bits)
    COPJMP1      088  S   A       Coprocessor restart at first location
    COPJMP2      08A  S   A       Coprocessor restart at second location
Not implemented:
    COPINS       08C  W   A       Coprocessor instruction fetch identify
```

Note:

- `COPINS` is not implemented.

ocs_serial

OCS serial port implementation with WISHBONE slave interface. [functionality not implemented].

List of serial registers:

```
Not implemented:
    SERDATR     *018  R   P       Serial port data and status read             read implemented here
     [DSKBYTR   *01A  R   P       Disk data byte and status read               read implemented here]

    SERDAT      *030  W   P       Serial port data and stop bits write
    SERPER      *032  W   P       Serial port period and control
```

bus_terminator

Terminator for not handled WISHBONE bus cycles.

bus_syscon

WISHBONE priority and round-robin SYSCON.

ocs_video

OCS video implementation with WISHBONE master and slave interface.

List of video registers:

```
Implemented:
    DIWSTRT      08E  W   A       Display window start (upper left vert-horiz position)
    DIWSTOP      090  W   A       Display window stop (lower right vert.-horiz. position)
    DDFSTRT      092  W   A       Display bitplane data fetch start (horiz. position)
    DDFSTOP      094  W   A       Display bitplane data fetch stop             write implemented here
     [DMACON     096  W   ADP     DMA control write (clear or set)             write implemented here]

     [JOY1DAT   *00C  R   D       Joystick-mouse 1 data (vert,horiz)           read not implemented here]
    CLXDAT      *00E  R   D       Collision data register (read and clear)     read not implemented here
    CLXCON       098  W   D       Collision control                            write implemented here
     [INTENA     09A  W   P       Interrupt enable bits (clear or set bits)    write implemented here]

    BPLCON0      100  W   AD( E ) Bitplane control register (misc. control bits)
    BPLCON1      102  W   D       Bitplane control reg. (scroll value PF1, PF2)
    BPLCON2      104  W   D( E )  Bitplane control reg. (priority control)

    BPL1MOD      108  W   A       Bitplane modulo (odd planes)
```

```
BPL2MOD      10A  W   A       Bitplane modulo (even planes)

BPL1PTH   +  0E0  W   A       Bitplane 1 pointer (high 3 bits)
BPL1PTL   +  0E2  W   A       Bitplane 1 pointer (low 15 bits)
BPL2PTH   +  0E4  W   A       Bitplane 2 pointer (high 3 bits)
BPL2PTL   +  0E6  W   A       Bitplane 2 pointer (low 15 bits)
BPL3PTH   +  0E8  W   A       Bitplane 3 pointer (high 3 bits)
BPL3PTL   +  0EA  W   A       Bitplane 3 pointer (low 15 bits)
BPL4PTH   +  0EC  W   A       Bitplane 4 pointer (high 3 bits)
BPL4PTL   +  0EE  W   A       Bitplane 4 pointer (low 15 bits)
BPL5PTH   +  0F0  W   A       Bitplane 5 pointer (high 3 bits)
BPL5PTL   +  0F2  W   A       Bitplane 5 pointer (low 15 bits)
BPL6PTH   +  0F4  W   A       Bitplane 6 pointer (high 3 bits)
BPL6PTL   +  0F6  W   A       Bitplane 6 pointer (low 15 bits)

BPL1DAT   &  110  W   D       Bitplane 1 data (parallel-to-serial convert)
BPL2DAT   &  112  W   D       Bitplane 2 data (parallel-to-serial convert)
BPL3DAT   &  114  W   D       Bitplane 3 data (parallel-to-serial convert)
BPL4DAT   &  116  W   D       Bitplane 4 data (parallel-to-serial convert)
BPL5DAT   &  118  W   D       Bitplane 5 data (parallel-to-serial convert)
BPL6DAT   &  11A  W   D       Bitplane 6 data (parallel-to-serial convert)

SPR0PTH   +  120  W   A       Sprite 0 pointer (high 3 bits)
SPR0PTL   +  122  W   A       Sprite 0 pointer (low 15 bits)
SPR0POS   %  140  W   AD      Sprite 0 vert-horiz start position data
SPR0CTL   %  142  W   AD( E ) Sprite 0 vert stop position and control data
SPR0DATA  %  144  W   D       Sprite 0 image data register A
SPR0DATB  %  146  W   D       Sprite 0 image data register B
SPR1PTH   +  124  W   A       Sprite 1 pointer (high 3 bits)
SPR1PTL   +  126  W   A       Sprite 1 pointer (low 15 bits)
SPR1POS   %  148  W   AD      Sprite 1 vert-horiz start position  data
SPR1CTL   %  14A  W   AD      Sprite 1 vert stop position and control data
SPR1DATA  %  14C  W   D       Sprite 1 image data register A
SPR1DATB  %  14E  W   D       Sprite 1 image data register B
SPR2PTH   +  128  W   A       Sprite 2 pointer (high 3 bits)
SPR2PTL   +  12A  W   A       Sprite 2 pointer (low 15 bits)
SPR2POS   %  150  W   AD      Sprite 2 vert-horiz start position data
SPR2CTL   %  152  W   AD      Sprite 2 vert stop position and control data
SPR2DATA  %  154  W   D       Sprite 2 image data register A
SPR2DATB  %  156  W   D       Sprite 2 image data register B
SPR3PTH   +  12C  W   A       Sprite 3 pointer (high 3 bits)
SPR3PTL   +  12E  W   A       Sprite 3 pointer (low 15 bits)
SPR3POS   %  158  W   AD      Sprite 3 vert-horiz start position data
SPR3CTL   %  15A  W   AD      Sprite 3 vert stop position and control data
SPR3DATA  %  15C  W   D       Sprite 3 image data register A
SPR3DATB  %  15E  W   D       Sprite 3 image data register B
SPR4PTH   +  130  W   A       Sprite 4 pointer (high 3 bits)
SPR4PTL   +  132  W   A       Sprite 4 pointer (low 15 bits)
SPR4POS   %  160  W   AD      Sprite 4 vert-horiz start position data
SPR4CTL   %  162  W   AD      Sprite 4 vert stop position and control data
SPR4DATA  %  164  W   D       Sprite 4 image data register A
SPR4DATB  %  166  W   D       Sprite 4 image data register B
SPR5PTH   +  134  W   A       Sprite 5 pointer (high 3 bits)
SPR5PTL   +  136  W   A       Sprite 5 pointer (low 15 bits)
SPR5POS   %  168  W   AD      Sprite 5 vert-horiz start position data
SPR5CTL   %  16A  W   AD      Sprite 5 vert stop position and control data
SPR5DATA  %  16C  W   D       Sprite 5 image data register A
SPR5DATB  %  16E  W   D       Sprite 5 image data register B
SPR6PTH   +  138  W   A       Sprite 6 pointer (high 3 bits)
SPR6PTL   +  13A  W   A       Sprite 6 pointer (low 15 bits)
SPR6POS   %  170  W   AD      Sprite 6 vert-horiz start position data
SPR6CTL   %  172  W   AD      Sprite 6 vert stop position and control data
SPR6DATA  %  174  W   D       Sprite 6 image data register A
SPR6DATB  %  176  W   D       Sprite 6 image data register B
SPR7PTH   +  13C  W   A       Sprite 7 pointer (high 3 bits)
SPR7PTL   +  13E  W   A       Sprite 7 pointer (low 15 bits)
SPR7POS   %  178  W   AD      Sprite 7 vert-horiz start position data
SPR7CTL   %  17A  W   AD      Sprite 7 vert stop position and control data
SPR7DATA  %  17C  W   D       Sprite 7 image data register A
SPR7DATB  %  17E  W   D       Sprite 7 image data register B

COLOR00      180  W   D       Color table 00
COLOR01      182  W   D       Color table 01
COLOR02      184  W   D       Color table 02
COLOR03      186  W   D       Color table 03
COLOR04      188  W   D       Color table 04
```

```
    COLOR05      18A  W   D       Color table 05
    COLOR06      18C  W   D       Color table 06
    COLOR07      18E  W   D       Color table 07
    COLOR08      190  W   D       Color table 08
    COLOR09      192  W   D       Color table 09
    COLOR10      194  W   D       Color table 10
    COLOR11      196  W   D       Color table 11
    COLOR12      198  W   D       Color table 12
    COLOR13      19A  W   D       Color table 13
    COLOR14      19C  W   D       Color table 14
    COLOR15      19E  W   D       Color table 15
    COLOR16      1A0  W   D       Color table 16
    COLOR17      1A2  W   D       Color table 17
    COLOR18      1A4  W   D       Color table 18
    COLOR19      1A6  W   D       Color table 19
    COLOR20      1A8  W   D       Color table 20
    COLOR21      1AA  W   D       Color table 21
    COLOR22      1AC  W   D       Color table 22
    COLOR23      1AE  W   D       Color table 23
    COLOR24      1B0  W   D       Color table 24
    COLOR25      1B2  W   D       Color table 25
    COLOR26      1B4  W   D       Color table 26
    COLOR27      1B6  W   D       Color table 27
    COLOR28      1B8  W   D       Color table 28
    COLOR29      1BA  W   D       Color table 29
    COLOR30      1BC  W   D       Color table 30
    COLOR31      1BE  W   D       Color table 31
```

ocs_audio

OCS audio implementation with WISHBONE master and slave interface.

List of audio registers:

```
Implemented:
    AUD0LCH   +  0A0  W   A( E )  Audio channel 0 location (high 3 bits, 5 if ECS)
    AUD0LCL   +  0A2  W   A       Audio channel 0 location (low 15 bits) (horiz. position)
    AUD0LEN      0A4  W   P       Audio channel 0 length
    AUD0PER      0A6  W   P( E )  Audio channel 0 period
    AUD0VOL      0A8  W   P       Audio channel 0 volume
    AUD0DAT   &  0AA  W   P       Audio channel 0 data

    AUD1LCH   +  0B0  W   A       Audio channel 1 location (high 3 bits)
    AUD1LCL   +  0B2  W   A       Audio channel 1 location (low 15 bits)
    AUD1LEN      0B4  W   P       Audio channel 1 length
    AUD1PER      0B6  W   P       Audio channel 1 period
    AUD1VOL      0B8  W   P       Audio channel 1 volume
    AUD1DAT   &  0BA  W   P       Audio channel 1 data

    AUD2LCH   +  0C0  W   A       Audio channel 2 location (high 3 bits)
    AUD2LCL   +  0C2  W   A       Audio channel 2 location (low 15 bits)
    AUD2LEN      0C4  W   P       Audio channel 2 length
    AUD2PER      0C6  W   P       Audio channel 2 period
    AUD2VOL      0C8  W   P       Audio channel 2 volume
    AUD2DAT   &  0CA  W   P       Audio channel 2 data

    AUD3LCH   +  0D0  W   A       Audio channel 3 location (high 3 bits)
    AUD3LCL   +  0D2  W   A       Audio channel 3 location (low 15 bits)
    AUD3LEN      0D4  W   P       Audio channel 3 length
    AUD3PER      0D6  W   P       Audio channel 3 period
    AUD3VOL      0D8  W   P       Audio channel 3 volume
    AUD3DAT   &  0DA  W   P       Audio channel 3 data
```

ocs_input

OCS user input implementation with WISHBONE slave interface.

List of user input registers:

```
Implemented:
     [DSKDATR & *008  ER  P       Disk data early read (dummy address)         not implemented]
```

```
    JOY0DAT     *00A  R   D       Joystick-mouse 0 data (vert,horiz)            read implemented here

    JOY1DAT     *00C  R   D       Joystick-mouse 1 data (vert,horiz)            read implemented here
     [CLXDAT    *00E  R   D       Collision data register (read and clear)      read implemented here]

    JOYTEST     *036  W   D       Write to all four joystick-mouse counters at once

Not implemented:
     [ADKCONR   *010  R   P       Audio, disk control register read             read not implemented here]
    POT0DAT     *012  R   P( E )  Pot counter pair 0 data (vert,horiz)          read not implemented here

    POT1DAT     *014  R   P( E )  Pot counter pair 1 data (vert,horiz)
    POTGOR      *016  R   P       Pot port data read (formerly POTINP)
    POTGO       *034  W   P       Pot port data write and start
```

ocs_floppy

OCS floppy implementation with WISHBONE master and slave interface.

List of floppy registers:

```
Implemented:
     [SERDATR   *018  R   P       Serial port data and status read              read not implemented here]
    DSKBYTR     *01A  R   P       Disk data byte and status read                read not implemented here

    DSKPTH    + *020  W   A( E )  Disk pointer (high 3 bits, 5 bits if ECS)
    DSKPTL    + *022  W   A       Disk pointer (low 15 bits)
    DSKLEN      *024  W   P       Disk length
    DSKDAT    & *026  W   P       Disk DMA data write

        [not used 07C]
    DSKSYNC     ~07E  W   P       Disk sync pattern register for disk read

Not implemented:
    DSKDATR   & *008  ER  P       Disk data early read (dummy address)          not implemented
     [JOY0DAT   *00A  R   D       Joystick-mouse 0 data (vert,horiz)            read not implemented here]
```

cia8520

Commodore 8520 Complex Interface Adapter implementation.

drv_vga

ADV7123 Video DAC driver for VGA output.

bus_ssram

IS61LPS51236A pipelined SSRAM driver with WISHBONE slave interface.

drv_audio

WM8731 audio codec driver for stereo audio output.

drv_keyboard

PS/2 keyboard driver.

drv_mouse

PS/2 mouse driver.

drv_debug

Switches and hex leds driver for debug purposes.

drv_eth_vga_capture

DM9000A 10/100 Mbit Ethernet driver for a VGA frame grabber.

3. Operation

SD card

The aoOCS SoC requires a SD card containing a list of available ROMs and floppy images together with images themselves. The aoOCS does not support any filesystem on the card. A binary image file must be prepared and written on the card starting at sector 0. The aoOCS_tool is used to prepare the image in the following way:

- a title screen PNG image must be available. A default image is available at ./sw/aoOCS_tool/title.png
- a directory with ROM files must be created
- a directory with floppy images (ADF files) must be created
- the following `make` command must be run at the base directory of the project:

```
    make sd_disk AO_INTRO_IMAGE=<path to title screen PNG image> AO_ROMS=<path to ROMs directory>
AO_FLOPPIES=<path to floppy images directory>
```

- The SD disk image is generated and saved to ./tmp/sd_disk.img. That image must be written directly on a SD disk bypassing and most probably destroying the filesystem on the disk. The easiest way to write the image is to run the `dd` command as a super-user on a Linux system:

```
dd if=<path to image file> of=<path to SD device>
```

aoOCS_tool

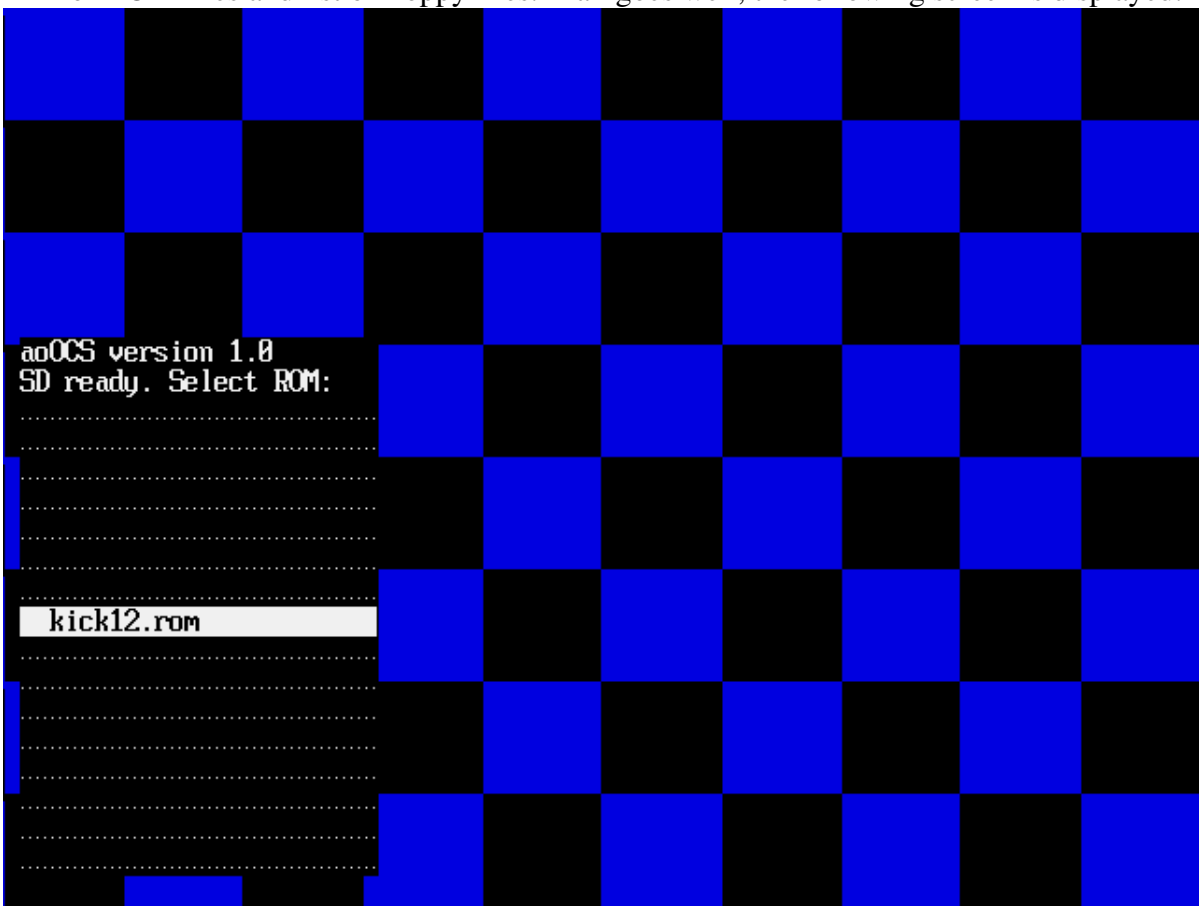
The aoOCS_tool is used to:

- Create the contents of the SD card. A image containing: a title screen, ROMs and floppy disk ADF files is created. This image has to be written to the SD card starting from sector 0.
- Extract vga frames from the drv_eth_vga_capture module as PNG images.
- Generate `./rtl/control_osd.mif` memory initialization file with On-Screen-Display text strings.
- Extract the specification contents from Doxygen HTML output, to generate the specification ODT file.

The source code for the tool is located at: `./sw/aoOCS_tool/`.

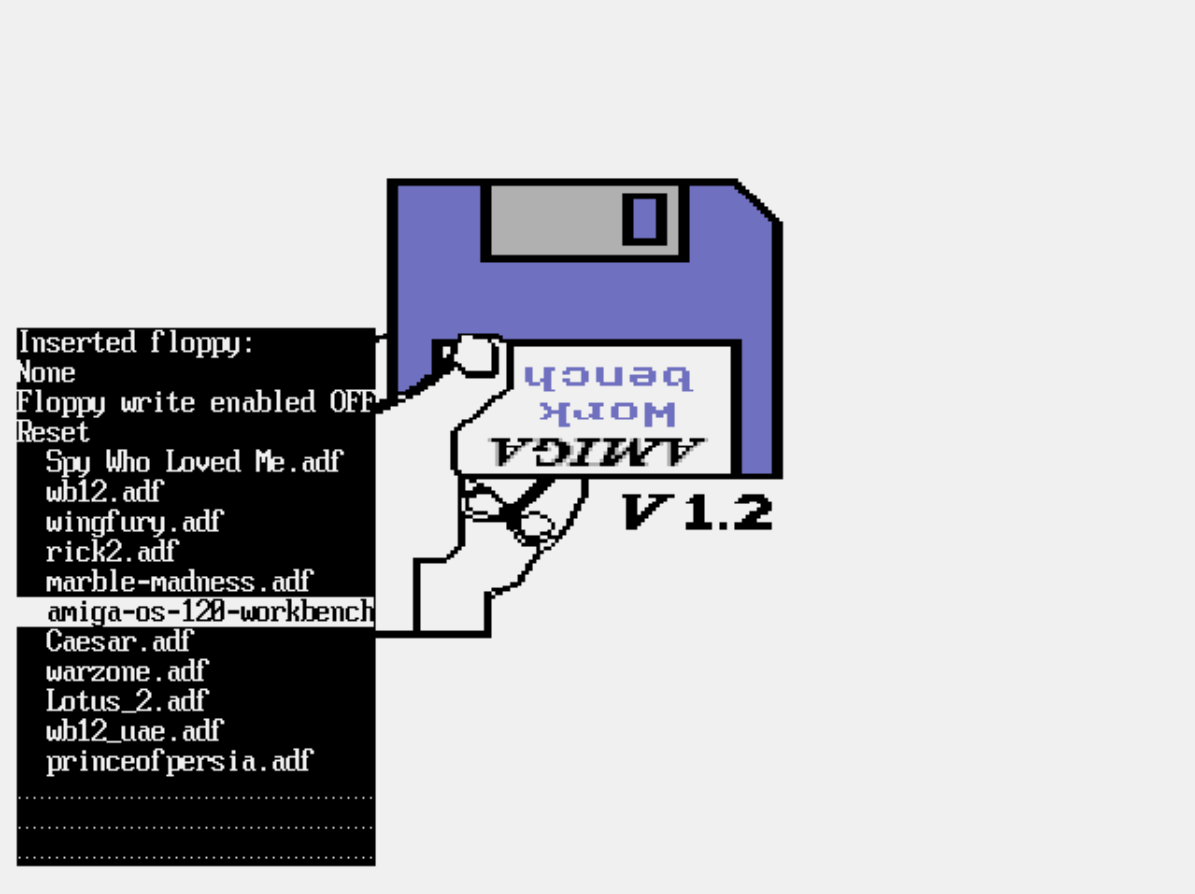
On-Screen-Display description

- After powerup or reset, the SoC tries to initialize the SD card and read the title screen, list of ROM files and list of floppy files. If all goes well, the following screen is displayed:



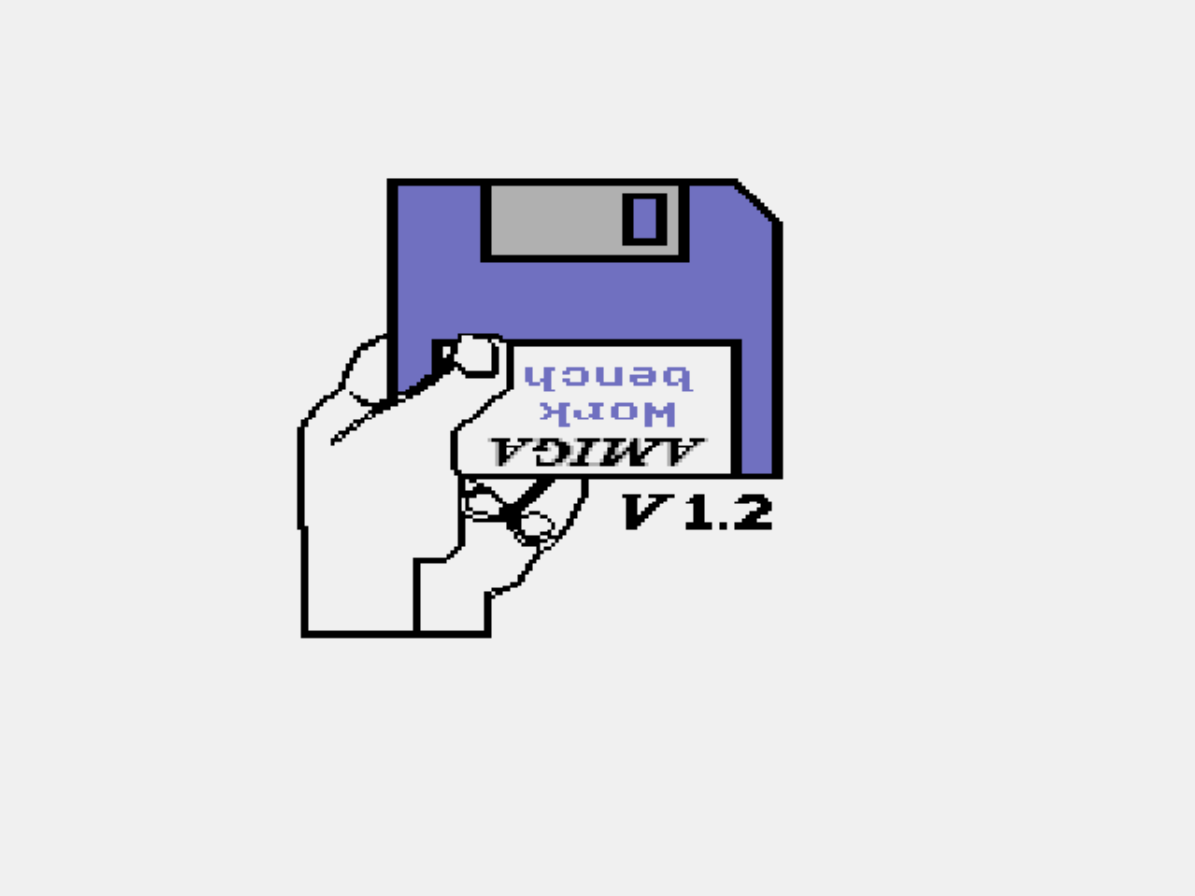
aoOCS title screen with ROM selection menu

- The On-Screen-Display is controlled by the keyboard arrow Up and Down keys. To select an item use the right Control key:



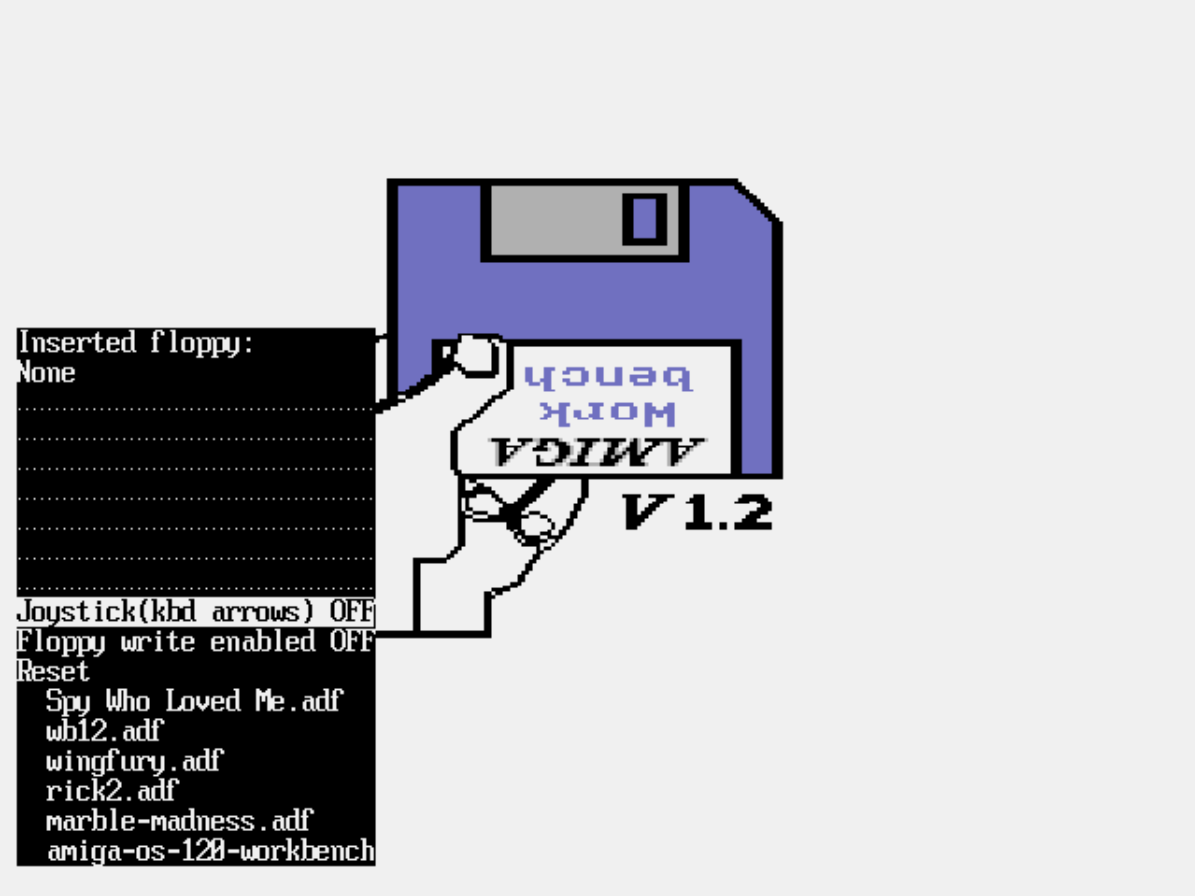
aoOCS floppy selection menu with Amiga Workbench 1.2 floppy highlighted

- After selecting the ROM file, the menu disappears and the Amiga boots from the ROM:



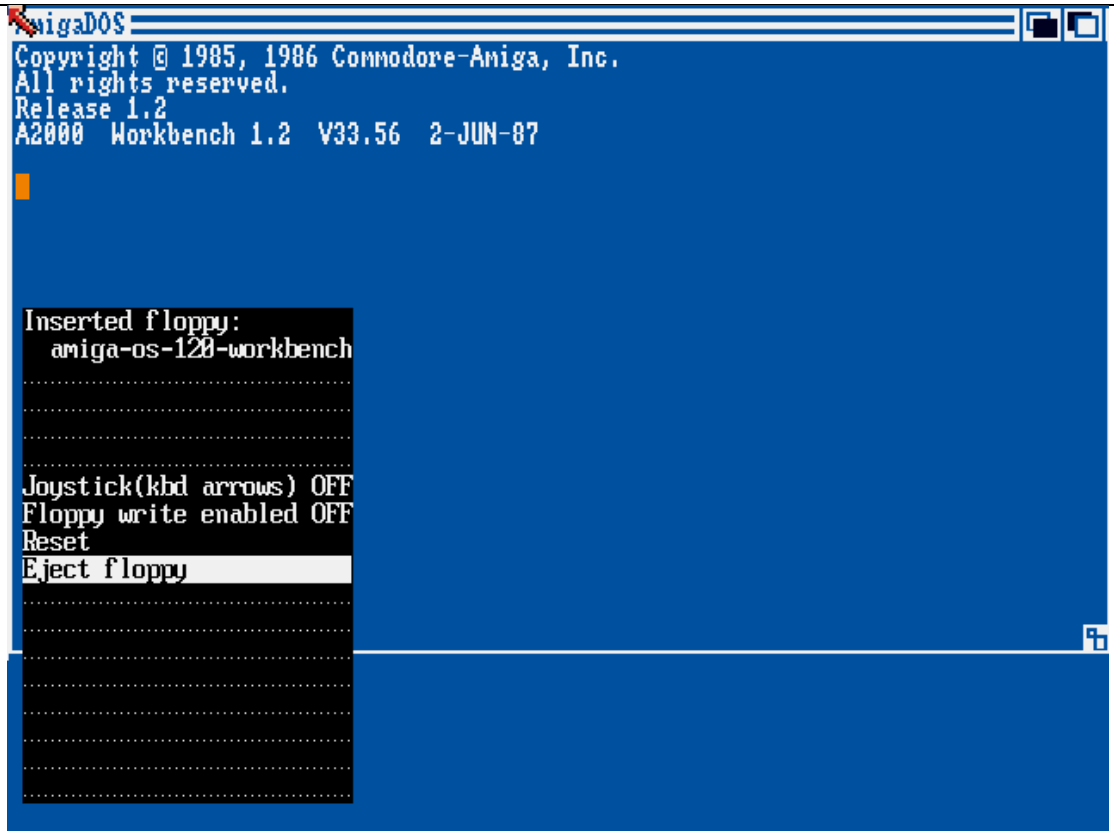
Amiga Kickstart v1.2 bootstrap screen

- To select a floppy disk to insert into the internal disk drive, the On-Screen-Display is used. To display the menu press the Home key. The Home key is also used to hide the menu. The menu when no floppy is inserted looks like this:



aoOCS floppy selection menu

- The following options are available:
  - `Joystick (kbd arrows)`: enable or disable the joystick on Amiga port 1. The joystick is controlled by the keyboard arrow keys and the right Control key. When enabled, the arrow keys are unavailable to the Amiga keyboard - the key strokes are redirected to the joystick.
  - `Floppy write enabled`: enable or disable floppy writes. The floppy changes are made directly on the SD disk.
  - `Reset`: reset the aoOCS SoC
  - Below is a list of available floppy disks to insert. After selecting a floppy the display changes to the following:



aoOCS menu after floppy selection

- In this menu it is possible to eject the floppy disk. After ejecting the floppy the previous menu is dispayed.

Software compatability list

The state of software can be:

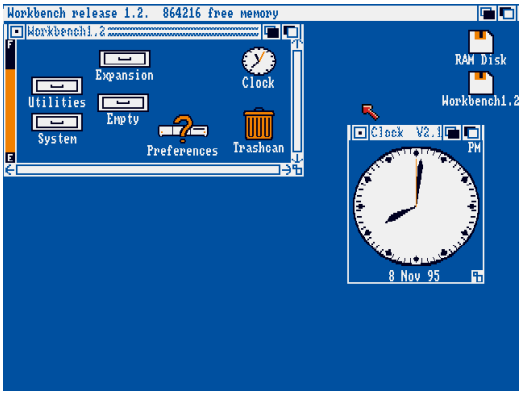
PERFECT	no visible and no audible distortions
GOOD	some minor distortions
FAIR	software starts but has major distortions
FAILED	software does not start

- Amiga Workbench version 1.2

aoOCS version: 1.0

State: *GOOD*

Description: Some minor graphic problems: bottom of screen not displayed correctly. Most probably problem with Copper and vertical beam position.



Amiga Workbench v1.2 screen

- Prince of Persia

aoOCS version: 1.0

State: *PERFECT*

Description: No problems.

Prince of Persia

- Wings of Fury

aoOCS version: 1.0

State: *GOOD*

Description: Some sound gliches in introduction. The game itself works perfect.

Wing of Fury

- Lotus 2

aoOCS version: 1.0

State: *GOOD*

Description: Some minor sound problems in introduction - most probably some bug in the low-pass filter or channel modulation. The VGA frame was captured in the middle of screen update so there are some distortions. In real-time in looks OK.

Lotus 2

- Warzone

aoOCS version: 1.0

State: *FAIR*

Description: Major graphic problems - as seen on captured VGA frame.

Warzone

4. Registers

List of not implemented OCS registers:

ocs_control

```
 STREQU    & *038  S   D       Strobe for horiz sync with VB and EQU
STRVBL    & *03A  S   D       Strobe for horiz sync with VB (vert. blank)
STRHOR    & *03C  S   DP      Strobe for horiz sync
STRLONG   & *03E  S   D( E )  Strobe for identification of long horiz. line.
VHPOSW      *02C  W   A       Write vert and horiz position of beam
```

ocs_input

```
 POT0DAT     *012  R   P( E )  Pot counter pair 0 data (vert,horiz)
POT1DAT     *014  R   P( E )  Pot counter pair 1 data (vert,horiz)
POTGOR      *016  R   P       Pot port data read (formerly POTINP)
POTGO       *034  W   P       Pot port data write and start
```

ocs_copper

```
 COPINS       08C  W   A       Coprocessor instruction fetch identify
```

ocs_serial

```
 SERDATR     *018  R   P       Serial port data and status read
SERDAT      *030  W   P       Serial port data and stop bits write
SERPER      *032  W   P       Serial port period and control
```

ocs_floppy

```
  DSKDATR   & *008  ER  P       Disk data early read (dummy address)
```

ocs_blitter

```
BLTDDAT   & *000  ER  A       Blitter destination early read (dummy address)
```

5. Clocks

Name	Source	Rates (MHz)			Remarks	Description
		Max	Min	Resolution		
clk_50	Input Port	50	50	-	-	External input clock. Used only as input to altpll.
clk_30	altpll output	30	30	-	-	Main system clock.
clk_12	altpll output	12	12	-	-	Used only in drv_audio to clock the WM8731 audio codec hardware.
clk_25	altpll output	25	25	-	-	Used only in drv_eth_vga_capture to clock the DM9000A Ethernet hardware.

Table 1: List of clocks.

6.

IO Ports

aoOCS top-level IO Ports

```
Inputs
    Clock and reset
        clk_50
        reset_ext_n
    DM9000A Ethernet hardware interface
        enet_irq
    Switches and hex leds hardware interface from drv_debug
        debug_sw1_pc
        debug_sw2_adr
        debug_sw3_halt
Inouts
    IS61LPS51236A pipelined SSRAM hardware interface
        ssram_data          [35:0]
    SD bus 1-bit hardware interface
        sd_cmd_io
        sd_dat_io
    PS/2 keyboard hardware interface
        ps2_kbclk
        ps2_kbdat
    PS/2 mouse hardware interface
        ps2_mouseclk
        ps2_mousedat
    WM8731 audio codec hardware interface
        ac_sdat
    DM9000A Ethernet hardware interface
        enet_data           [15:0]
Outputs
    IS61LPS51236A pipelined SSRAM hardware interface
        ssram_address       [18:0]
        ssram_oe_n
        ssram_writeen_n
```

```
        ssram_byteen_n      [3:0]
        ssram_clk
        ssram_globalw_n
        ssram_advance_n
        ssram_adsp_n
        ssram_adsc_n
        ssram_ce1_n
        ssram_ce2
        ssram_ce3_n
    SD bus 1-bit hardware interface
        sd_clk_o
    ADV7123 Video DAC hardware interface
        vga_r   [9:0]
        vga_g   [9:0]
        vga_b   [9:0]
        vga_blank_n
        vga_sync_n
        vga_clock
        vga_hsync
        vga_vsync
    WM8731 audio codec hardware interface
        ac_sclk
        ac_xclk
        ac_bclk
        ac_dat
        ac_lr
    DM9000A Ethernet hardware interface
        enet_clk_25
        enet_reset_n
        enet_cs_n
        enet_ior_n
        enet_iow_n
        enet_cmd
    Switches and hex leds hardware interface from drv_debug
        hex0                [7:0]
        hex1                [7:0]
        hex2                [7:0]
        hex3                [7:0]
        hex4                [7:0]
        hex5                [7:0]
        hex6                [7:0]
        hex7                [7:0]
    Leds hardware interface for debug purposes
        debug_sd            [7:0]
        debug_68k_state     [7:0]
        debug_floppy        [7:0]
```

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. *AmigaÂ® Hardware Reference Manual.*
   Revised edition (March 15, 1990)
   Addison Wesley Longman Publishing Co.

3. *E-UAE Amiga software emulator sources.*
   Source archive: `e-uae-0.8.29-WIP4.tar.bz2`
   Available from: http://http://www.rcdrummond.net/uae/

4. *aoOCS Doxygen(Design) Documentation.*

5. *ao68000 Doxygen(Design) Documentation.*
   Available from: http://www.opencores.org