# C-One core development

Technical Ref. Manual for the C-One Reconfigurable Computer

Peter Wendrich
pwsoft@syntiac.com

June 2, 2009

## Contents

**1 Introduction 2**

**2 Overview C-One PCB 2**
2.1 ACEX 1K30 FPGA . . . 2
2.2 ACEX 1K100 FPGA . . . 2
2.3 Extender Board . . . 2
2.4 FPGA resources . . . 2

**3 FPGA to FPGA communication 3**

**4 Designing FPGA cores 3**
4.1 1k100 . . . 3
4.2 Extender Board . . . 3
4.3 Triple FPGA designs . . . 4
4.4 Steps to reconfigure the 1k30 . . . 4

**5 CPLD 3032 5**

**6 CPU-Slot 5**
6.1 Signals and pins . . . 5
6.2 128 KByte SRAM . . . 6
6.3 WDC 65816 CPU . . . 6

**7 CPLD 7064 6**
7.1 Signals and pins . . . 6
7.2 cselect . . . 6
7.3 Joysticks . . . 7
7.3.1 Example code for reading the joysticks . . . 8
7.3.2 POT X/Y . . . 8
7.4 IEC serial bus . . . 9
7.5 Printer port . . . 9

**8 VGA video 9**
8.1 Generating a picture . . . 9
8.2 VGA from the Extender board . . . 9
8.3 VGA from the 1k30 . . . 10
8.4 VGA from the 1k100 . . . 10

**9 PS/2 Keyboard and Mouse 10**
9.1 Using a PS/2 keyboard . . . 10
9.2 Using a PS/2 mouse . . . 10

**10 Accessing the SID chips 11**

**11 Cartridge port** **11**
11.1 BA mod . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11

**12 Pins and Signals** **11**

## 1 Introduction

The C-One is called the reconfigurable computer because it has to FPGAs that reconfigure each other. This runtime reconfigurability is a great feature of the platform, but it can make it a challenge to develop for. Recently Individual Computers has developed a extender board that plugs into the C-One PCB, bringing the FPGA count on three. The extra power this extender board with the Cyclone-III FPGA brings extends the C-One beyond its original goal of 8 bit home computer emulation. There is now enough FPGA logic available to emulate also many 16 and 32 bit machines. This manual tries to give enough technical information and examples to tame this beast. The C-One was always plagued by lack of documentation. Lets hope this issue is now solved.

Thanks go out to Jens Schoenfeld and Tobias Gubener for their input and support.

## 2 Overview C-One PCB

### 2.1 ACEX 1K30 FPGA

This FPGA is connected to the mass storage devices, has a SIMM module as memory and can generate video (or use the same port to communicate with the extender board). It doesn't have much logic (FPGA fabric) so most designs will be running on either the 1K100 FPGA or extender board.

### 2.2 ACEX 1K100 FPGA

This FPGA is designed to run the 8 bit emulations (although many are being ported to the extender board). It has direct access to the C64 compatible cartridge slot, 128 KByte of static memory and a 65816 processor. The 65816 is a 16 bit successor to the 6502 microprocessor which was used as CPU in many home-computers. Many of the emulator cores however use a soft processor inside the FPGA.

### 2.3 Extender Board

This is an add-on board that plugs into two rows of header pins close to the VGA connector. The board adds a powerful Cyclone III FPGA to the C-one. The FPGA connects between the 1k30 and the VGA output and also gets a few global signals from the geek-port (that was not used for anything else yet). The board has its own crystal clocks and power regulators. There are about 5 times as many logic resources and 10 times the memory bits inside the Cyclone III compared to the 1k100. This large increase of available logic blocks allows 16 and 32 bits computers to be emulated. The first core that uses this extra power is a port of the Amiga emulator "minimig". All new C-One's are sold with this add-on board pre-installed.

### 2.4 FPGA resources

| FPGA | id | I/O Pins | LUTs/LEs | Memory bits | 9x9 Mult. | PLLs |
|---|---|---|---|---|---|---|
| Acex 1K30 | EP1K30QC208-3 | 147 | 1728 | 24576 | 0 | 0 |
| Acex 1K100 | EP1K100QC208-3 | 147 | 4992 | 49152 | 0 | 0 |
| Cyclone III | EP3C25E144C8 | 83 | 24624 | 608256 | 132 | 4 |

# 3 FPGA to FPGA communication

There are two busses used for direct FPGA to FPGA communication. Between the 1k100 and 1k30 there are 8 pins available called **gb[7..0]**. The support routines for these pins are called the gbridge and consists of two pieces of AHDL source code. One runs on the 1k100 and the other on the 1k30. When loading custom designs in both FPGAs then the gb pins can be redefined as needed.

The 1k30 and extender communicate with the signals on the 1k30 that are also used for video. Initially the extender board is set into bypass mode so the the 1k30 (or 1k100 via the gbridge) can generate the video signal. After a design is loaded these 16 pins labeled **vid[15..0]** can be redefined for other purposes. Some of the pins are input-only on the extender however.

The databus lines **mdb[7..0]** are available to all 3 FPGAs, because these signals are connected to many devices and also the cartridge port they can't support very fast changing signals. The maximum supported clock frequency will be somewhere from 8 to 10 Mhz, faster signals will degrade too much and result in unreliable operation. When accessing the SID chip these signals need to be held stable for about 500ns even slowing them down more.

The lowest address lines are shared between 1k100 and 1k30. These could also be used for FPGA to FPGA communication, however just as the mdb signals they are quite slow.

# 4 Designing FPGA cores

## 4.1 1k100

For small to medium size designs the 1k100 is the most logical FPGA to program for. It has direct connection to most peripherals on the board like keyboard, joysticks and cartridge port. For video it depends on the 1k30 and a bit of logic called the GBridge (see chapter 8.4). The 1k100 is the only FPGA that can drive the cpu-slot. It has a small red colored PCB mounted by default that contains a 65816 CPU and 128 KByte of SRAM. The CPU and SRAM share the address and data lines which restricts the possible memory layouts when the CPU is used. The CPU port was only designed for CPUs not memory and this was the best that could be done with the available signals. The 1k100 is also connected to the DIMM slot for huge amount of memory. Unfortunetely the DIMM slot is directly connected to the 101 Mhz sysclk and doesn't give the FPGA the possibility to generate a clock itself. This creates a challenge for meeting the correct timing specifications. Therefore most cores use the 128 KByte SRAM as it is much easier to use.

The 1k100 is also connected to the PCI slot and the C64 compatible cartridge connector. With the BA line modification (see chapter 11.1) all signals of the cartridge connector can be controlled from the 1k100. At this moment there are no cores existing using the PCI connector. However the PCI connector can also be seen as 32 dedicated 5 volt tolerant I/O lines. So it probably will be reused in the future for additional modifications, fixes and enhancements to the C-one PCB.

## 4.2 Extender Board

The extender board can handle large designs. It has fast SDRAM memory and direct access to the VGA port. It therefore can generate higher video resolutions with more colors compared to the 1k100. Because the extender board has no direct connections to any of the I/O most often it needs custom designs in both the 1k100 and 1k30 too. The functions is the 1k100 and 1k30 can be simple for most cores, just forwarding the the keyboard, mouse and joystick information. This might however not always be sufficient. When SID chip(s), IEC bus, CF card or cartridge port is needed the logic is more complex and a protocol must be developed to communicate everything through the 16 available lines.

## 4.3 Triple FPGA designs

With the NewBoot BIOS the 1k30 FPGA is used to reconfigure the 1k100 and/or extender board. The standard image in 1k30 can forward VGA data from 1k100 to the video port. The extender can either forward the video from the 1k30 or generate it's own. This is sufficient for many designs. It is however possible (and in some cases necessary) to build a triple fpga design where all 3 fpga's contain custom logic. This creates a bit of a chicken and egg problem. There is always one FPGA that needs to have some logic included in its design to reconfigure the last FPGA.

The 1k30 running NewBoot will reconfigure the extender board first. It then loads a new core into the 1k100. The 1k30 loads an image file with a core for itself into the first half of the 128 KByte SRAM (Located on the little red PCB with the 65816 cpu). Inside the 1k100 core there must be a small bit of logic (about 200 LEs in size) that reconfigures the 1k30 from the SRAM image and then switches off. When the SRAM is filled the 1k30 will signal the 1k100 so it can start reconfiguring the 1k30. Now all 3 fpgas should have be loaded with custom cores.

The loading of SRAM and the reconfiguration of the 1k30 can be divided into two different cores as the 1k100 can be reconfigured a second time while data stays in SRAM. If this method is used, less logic needs to be present in the second image (only the core reconfiguration logic) and you don't have to worry about storing data comming from 1k30 running NewBoot into the SRAM. If this method is used the "support.rbf" file only contains a memory loader and then "target.rbf" is loaded into the 1k100 as final core which contains the reconfiguration logic itself. This standard "support.rbf" memory loader can be copied from one of the archives (it is used for the VC20 and FPGA-64 cores for example). It can load a maximum of 128 Kbyte stored on the CF card as "support.bin". This a binary file that is loaded into the SRAM starting from memory address zero.

Keep in mind when desiging triple FPGA designs that not all the FPGA's are loaded at the same time. The extender is configured first, then the 1k100 is reloaded (either once or twice) and finally the 1k30. So there must be some synchronisation mechanism implemented that checks if all 3 FPGAs are present and ready before starting. Using the same technique it is also possible to make dual FPGA designs. In that case only the 1k100 and 1k30 are used and the extender board is left in bypass mode (or can even be absent). As before the 1k100 core contains logic that reconfigures the 1k30 from SRAM.

Summary of the loading steps of a triple FPGA design:

- 1k30 reconfigure extender board from the file "extender.rbf".
- 1k30 reconfigures 1k100 from the file "support.rbf".
- 1k30 loads the file "support.bin" into first half of the SRAM using the core in the 1k100 (which must support this!).
- 1k30 loads the file "target.rbf" if it exists and reconfigures the 1k100 with it.
- 1k100 can now reconfigure 1k30 from SRAM (by using logic loaded into the 1k100, which is either "support.rbf" or "target.rbf").

When writing your own "support.rbf" core it is important to wait for the 1k30 completing its memory transfer before starting the reconfiguration. Most of the pins of the 1k100 should be kept in tri-state (input). The 1k30 signals when it is ready by a transition on the pin **gb[6]** from low to high. When the 1k30 is being reconfigured all its pins are tri-stated automatically so the 1k100 can use all signals during this time.

## 4.4 Steps to reconfigure the 1k30

Before staring the reconfiguration the 1k30 core image must be present in SRAM. If NewBoot provides the file the logic should wait for a low to high transition on the **gb[6]** pin. Otherwise the loading of the 1k30 core might be interrupted and it will not work reliably.

Perform the following steps in the 1k100 core to reconfigure the 1k30. The 1k30 image should already be present in the SRAM:

- Clear flash address to zero the **flashA$_{16}$ / conf-done** line. (A$_3$=0, A$_2$=0 and cselect=0110$_b$)
- make pin nConfig on 1k30 low
- wait until 1k30 is cleared
- make pin nConfig on 1k30 high
- wait until 1k30 is ready to be reconfigured
- load data into 1k30 from SRAM (59215 bytes)
  - Read byte (on mdb[7..0] lines) from SRAM (cs = 0, wr = 1 and address is supplied)
  - Store byte into 1k30 (cs = 1, wr = 0, mdb[7..0] = byte). The 1k30 takes over the byte when pin **wr** makes a low to high transition.
- Make line **flashA$_{16}$ / conf-done** high (repeat 16 times increment-flash-address and shift-left)

## 5 CPLD 3032

This CPLD is located on the instant-on board. Most of its pins are connected to the flash-rom. During powerup it configures the C-One from flash. It also plays a minor role in reconfiguration of one FPGA from another. The flashA$_{16}$ line of the CPLD is connected to both the flash and the **config-done** pins of the 1k30 FPGA. To select the CPLD set the **cselect[3..0]** lines to 0110$_b$. The address lines **A$_2$** and **A$_3$** select the function to be performed. The CPLD performs the action when the select line is deasserted (cselect = 1111$_b$). The CPLD is clocked by the slowish 2 Mhz clock so each access must be atleast 1 uSec long. Keep the **A$_2$** and **A$_3$** stable for another 500 nSec after deasserting the cselect line.

| action | A$_3$ | A$_2$ | cselect | comment |
|---|---|---|---|---|
| Clear flash address | 0 | 0 | 0110$_b$ | |
| Shift address 1 bit left | 0 | 1 | 0110$_b$ | |
| Increment flash address | 1 | 0 | 0110$_b$ | |
| FlashRom access | 1 | 1 | 0110$_b$ | A$_{1..0}$ are flashA$_{18..17}$ for 512Kbyte ROMs |

With the increment and shift left commands it is possible to set any address into the CPLD. It will however have to be done bit for bit. Linear reading from the flash ROM however is quite efficient. One clock tick to read a byte and one other clock tick to advance the address with 1. It is not possible to read back the current address stored in the CPLD. If knowledge about this address is necessary, it needs to be duplicated inside the FPGA core.

## 6 CPU-Slot

The cpu-slot was designed to host various microprocessors depending on the core(s) that would be run on the C-One. In practise the cpu-slot always contains a small red PCB that has a 128 KByte SRAM and a 65816 CPU made by WDC.

### 6.1 Signals and pins

Because the SRAM and CPU share a limited number of pins on the CPU-Slot (designed to host a CPU only) not all signals of the 65816 are connected. The lines of the CPU left open are:

- MX - ”Mode select”
- VPA - ”Valid Program Address”
- VDA - ”Valid Data Address”

The MX line gives additional information about the opcode that is currently being executed, which gives additional possibilities for memory management. In detail, this line shows the Accumulator (M) and Index (X) elect flags (bits 5 and 4 of the Processor Status register). The C-One does not require this information, so the pin is left floating in the design.

VPA and VDA are also status outputs of the CPU. They indicate if an access to memory is a program access, data access, opcode fetch or a "dummy access", where the address bus may contain an invalid address. Not having these lines available is not a big loss. In theory, they would give the system designer a method to implement a harvard architecture, where code and data memory spaces are separated. For the majority of the (potential) C-One cores this is not an issue as most 8 bit machines from the past used the von-Neumann architecture where code and data is all in the same address space.

What was formerly the VDA line is now the Chip-Select line of the S-Ram, and it's controlled by the 1k100 FPGA (pin 199). The former VPA line is now the higher-order address line of the S-Ram chip ($A_{16}$), and it's also controlled by the 1k100 FPGA (pin 202). The MX line is not available on the CPU-Slot at all. The Output-Enable of the SRAM is tied to ground. The access to the SRAM can be controlled with the Chip-Select.

## 6.2 128 KByte SRAM

The SRAM is connected to the data-lines (mdb[7..0]) and 17 address-lines (a[16..0]), cs and wr. To read from the RAM the address must be set and **wr** must be high, then make cs low and the data appears on the data-lines. To write to the RAM set the address and data-lines and **wr** must be low, then make cs low for 10 nSecs or longer.

## 6.3 WDC 65816 CPU

TODO

# 7 CPLD 7064

This CPLD is on the board to extend the number available I/O lines. It is used for reading the joysticks, accessing the printer port and controlling the IEC setial bus. It also plays a small role in FPGA reconfiguration.

Most of the control signals needed for the CPLD are avaible on both the 1k30 and 1k100 FPGA. However the 1k100 is always involved as this FPGA is the only one capable of generating the **pclk** signal (pin 168). This is the clock that triggers the CPLD. It takes action on a low to high transition of **pclk**.

## 7.1 Signals and pins

| Signal name | purpose |
|---|---|
| pclk | Clock transition from 0 to 1 clocks data into or out of the CPLD |
| cselect[3..0] | Selects which I/O block or device is addressed. |
| wr | Global write also used for the CPLD. 0=write, 1=read |
| mdb[7..0] | 8 data lines to read or write data |
| ioshift | Serial stream with joystick information. The bits are shifted out on pclk |

## 7.2 cselect

The 4 cselect signals give access to a number of I/O devices on the board or cartridge port. If cselect[3] is 1 it is processed by the 7064 CPLD for things such as IEC and printer port. Because the bits of cselect that are connected to the mux are in the order 3,0,1,2 the table is a little weird.

| cslect | action | comments |
|---|---|---|
| $0000_b$ | ROML | select ROM on cartridge (low ROM) |
| $0001_b$ | IO2 | select IO area on cartridge port for address range $DF00_h$ to $DFFF_h$ |
| $0010_b$ | clk1cs | pull the chip-select line on the clock port |
| $0011_b$ | SID 1 | Read or write to first SID chip |
| $0100_b$ | ROMH | select ROM on cartridge (high ROM) |
| $0101_b$ | IO1 | select IO area on cartridge port for address range $DE00_h$ to $DEFF_h$ |
| $0110_b$ | clk2cs | select 3032 CPLD (on instant-boot PCB) |
| $0111_b$ | SID 2 | Read or write to second SID chip |
| $1000_b$ | 7064 status | irq and reconfiguration register |
| $1001_b$ | 7064 lpdata | Data for the printer port |
| $1010_b$ | 7064 lpdir | Data direction for the printer (0=input, 1=output) |
| $1011_b$ | 7064 IEC port | C64 compatible serial disk-drive and printer port |
| $1100_b$ | 7064 PCI | PCI control pgnt0=1 pgnt1=1 |
| $1101_b$ | 7064 PCI | PCI control pgnt0=0 |
| $1110_b$ | 7064 PCI | PCI control pgnt1=0 |
| $1111_b$ | nop | nothing selected (default) |

## 7.3 Joysticks

The joystick ports are input only, this can not be changed. The signals **joya[6..0]** are connected to J18 (the first joystick), which is the port close to the IEC connector. The signals **joyb[6..0]** are connected to J17 (the second joystick), which is the port close to the VGA connector. The first joystick port (J18) connects to the pot-X/Y inputs on the first SID (but the X and Y are swapped compared to a C64 machine). The second joystick port (J17) connects to the pot-X/Y inputs on the second SID (again the X and Y are swapped compared to a C64 machine). In the C64/C128 an analog muxtiplexer is present on the potX and potY lines to switch between the two joystick ports, both signal pairs go to the same SID. On the C-One however both SIDs need to be present if the pot inputs are used on the second port as each SID chip only receives input from one joystick port.

Take note that the C1 schematic is very confusing on this topic as many joystick signals have the wrong names and labels. For example signal potx_sid2 is connected to the pot-Y input of the first SID!

The joysticks data is read-out serially (a strange design decision as the 7064 CPLD has a 8 bit data bus). On every 0 to 1 transition on **pclk** a new bit is shifted out on the **ioshift** pin. The sequence is 16 bits long and then repeats. A sync bit is provided to identify the start of the sequence. The following table show the 16 bits that are shifted out in order on the **ioshift** pin.

| bit | signal name | connector pin | comment |
|---|---|---|---|
| 0 | '0' | – | Synchronisation bit always 0 |
| 1 | lpt-busy | 11 (J16) | busy pin of the printer port |
| 2 | joya[0] | 5 (J18) | also connected to potX SID 1 |
| 3 | joya[1] | 9 (J18) | also connected to potY SID 1 |
| 4 | joya[2] | 4 (J18) | stick 1 RIGHT |
| 5 | joya[3] | 3 (J18) | stick 1 LEFT |
| 6 | joya[4] | 2 (J18) | stick 1 DOWN |
| 7 | joya[5] | 6 (J18) | stick 1 fire button |
| 8 | joya[6] | 1 (J18) | stick 1 UP |
| 9 | joyb[0] | 1 (J17) | stick 2 UP |
| 10 | joyb[1] | 6 (J17) | stick 2 fire button |
| 11 | joyb[2] | 2 (J17) | stick 2 DOWN |
| 12 | joyb[3] | 3 (J17) | stick 2 LEFT |
| 13 | joyb[4] | 4 (J17) | stick 2 RIGHT |
| 14 | joyb[5] | 9 (J17) | also connected to potY SID 2 |
| 15 | joyb[6] | 5 (J17) | also connected to potX SID 2 |

### 7.3.1 Example code for reading the joysticks

The following code can be used on the 1k100 FPGA to read the joysticks.

```
architecture ... of ... is
...
    -- Shift register to deserialize joystick information
    signal joyshift : unsigned(15 downto 0);

    -- Contains current joystick info (updated every 16 clock ticks)
    signal joystick : unsigned(15 downto 0);
...
begin
...
    -- Feed clock to CPLD
    pclk <= clk;

    -- Deserialize joystick information
    process(clk)
    begin
        if rising_edge(clk) then
            joyshift <= ioShift & joyshift(15 downto 1);
            if joyshift(0) = '0' then
                joyshift(14 downto 0) <= (others => '1');
                joystick <= joyshift;
            end if;
        end if;
    end process;
...
end architecture;
```

### 7.3.2 POT X/Y

Following table shows the POT X/Y connections between the joystick connectors and the SID chips. Note that the X and Y inputs are swapped (this is true for both SIDs).

| Source | SID connection |
|---|---|
| POT AX port 1 (J18 pin 9) | SID 1 potY (pin 23) |
| POT AY port 1 (J18 pin 5) | SID 1 potX (pin 24) |
| POT BX port 2 (J17 pin 9) | SID 2 potY (pin 23) |
| POT BY port 2 (J17 pin 5) | SID 2 potX (pin 24) |

### 7.4 IEC serial bus

TODO

### 7.5 Printer port

TODO

# 8 VGA video

## 8.1 Generating a picture

To generate a picture there are three things that need to be created.

- Logic that creates pixels that will be displayed
- A horizontal-sync generator
- A vertical-sync generator

The VGA interface was designed for CRT screens that work with coils and raster beams. The screen is scanned from left to right and from top to bottom all the while hitting phosphor making it glow. It takes some time for the beam to reset from the right side back to the left and from the bottom of the screen to the top. Therefore there are more horizontal pixels and lines as you can see in visible pixels.

For example a screen with a visible resolution of 800 by 600 is actually 1040 pixels wide by 666 lines high. In these extra invisible pixels the beam is sweeped back to its start position. It is very important that during this time the color send to the monitor is black (all bits zero). One reason is that not all monitors suppress the beam completely and you would get horizontal or vertical lines across the screen. The other reason is that during this time the actual color value is taken as reference level for black. Electronic circuits always have problems reaching the limits of the power supply. So exactly 0 volt is difficult to create. By measuring the actual voltages for black during the sweep time the monitor can adjust its color amplifiers to match this level so black is black (and not gray or dark green).

## 8.2 VGA from the Extender board

Generating VGA video from the cyclone III on the extender board is the most direct way. This FPGA has direct connections to all VGA pins. Take note that the **v-sync** and **h-sync** signals are shared between the 1k30 FPGA and the extender board. Only one of the two must output on these pins at one time. When NewBoot loads a core into the extender board it automatically releases **v-sync** and **h-sync** on the 1k30 and put these in tri-state.

The extender board has two crystals that output exact multiples of NTSC and PAL color burst frequencies. It is therefore possible to generate a composite video signal by using only one of the VGA color channels and proper logic inside the FPGA.

## 8.3 VGA from the 1k30

During startup the extender board is loaded with a default core that routes the color information from the 1k30 to the VGA. This way the NewBoot menu that runs on the 1k30 can display its output both with and without an extender board present. If a core is loaded on the 1k100 FPGA, the extender board stays in this feed-tru mode. This way old cores can still run.

Because of this feed-tru mode a 1k30 core can access the VGA port directly.

## 8.4 VGA from the 1k100

The 1k100 FPGA doesn't have access to any of the video signals. Anything it wants to display needs to go through the 1k30 FPGA. The eight pin **gb[7..0]** can be used for this purpose. If only the 1k100 is programmed the standard gbridge logic can be instantiated. It accepts video in either 16 bit color resolution with a 25 Mhz pixel-clock (GBmode=0) or in 12 bit color resolution with a 33 Mhz pixel-clock (GBmode=1). The 25 Mhz pixel-clock is perfect for a 640x480 standard VGA picure. The 33 Mhz is fast enough for a 50Hz or 60Hz 800x600 screen. The gbridge outputs a clock enable (that can also be used as clock source) at the pixel-clock rate. Video data is taken over on the falling edge.

Many emulator cores use the pixel-clock as main clock source. The system clock of 101 Mhz is often too fast for the slower ACEX 1K series FPGAs.

# 9 PS/2 Keyboard and Mouse

There are 4 signals related to PS/2. A clock and data line for keyboard and a clock and data line for the mouse. All signals are both input and output and are available on both FPGAs on the C-One PCB (but on different pin numbers). The extender board does not have direct access to these pins and must gain access to them via the 1k30 FPGA. Take note that in the schematic the clock and data lines are swapped (both for keyboard and for the mouse). See next table for the correct pin layout.

| signal name | Pin number on 1K30 | Pin number on 1K100 |
|---|---|---|
| keyb-clk | 203 | 70 |
| keyb-dat | 202 | 69 |
| mouse-clk | 200 | 68 |
| mouse-dat | 199 | 67 |

The PS/2 is an open-collector (or open-drain) bus with pull-ups. This allows both the host (computer) and the device (mouse or keyboard) to send and receive data. To simulate a open-collector with a FPGA the following code can be used. The ps2_clk and ps2_dat signals represent the I/O pins and the ps2_clk_out and ps2_dat_out are the signals that the core generates. Only zeros drive the pin low and ones let it float (and turn in into an input).

```
ps2_clk <= '0' when ps2_clk_out = '0' else 'Z';
ps2_dat <= '0' when ps2_dat_out = '0' else 'Z';
```

## 9.1 Using a PS/2 keyboard

TODO

## 9.2 Using a PS/2 mouse

TODO

## 10 Accessing the SID chips

The two SID chips are accessable from both the 1k100 and 1k30, but only the 1k30 can generate the necessary 1 Mhz clock. So if the 1k100 is using the SID chips it must transfer the clock pulse to the 1k30 (most likely using one of the gb pins) or the 1k30 must recreate this clock itself.

The SID chips are connected to address line $\mathbf{A}_{4..0}$, data lines $\mathbf{mdb}_{7..0}$ and the **WR** for selecting read (WR=1) or write (WR=0). During access the 1Mhz clock must be high having close to 50 percent duty cycle. So each access takes 500 ns. The chip-selects of the chips are controlled with cselect.

| cslect | action | comments |
|---|---|---|
| $0011_b$ | SID 1 | Read or write to first SID chip |
| $0111_b$ | SID 2 | Read or write to second SID chip |
| $1111_b$ | nop | nothing selected (default) |

## 11 Cartridge port

### 11.1 BA mod

Due to a design issue the BA pin of the cartridge connector is not available on a FPGA pin. The “BA Mod” a small PCB modification solves this problem. Resistor R72 needs to be removed and a wire connected from the BA pin to the (previously unused) PCI connector. BA must be connected to data bit 1 on the PCI.

All new boards will be shipped with the patch already applied.

## 12 Pins and Signals

Clocks

| Name | 1K30 | 1K100 | Ext. | Cart. | comments |
|---|---|---|---|---|---|
| sysclk | 79 | 79 | – | – | 101 Mhz system clock |
| pclk | – | 168 | – | – | Clock for CPLD 7064 |
| cpuclk | 183 | 163 | – | E | Clock for the cartridge port and 65816 |
| dot-clock | 80 | 16 | – | 6 | C64 pixel clock, input only on 1k30 |
| 2mhz | 184 | – | – | – | Fixed 2Mhz clock derived from 101 Mhz |
| 1mhz | 150 | – | – | – | Clock for the SID chips |

Control signals

| Name | 1K30 | 1K100 | Ext. | Cart. | comments |
|---|---|---|---|---|---|
| cselect[0] | 29 | 169 | – | – | |
| cselect[1] | 30 | 170 | – | – | |
| cselect[2] | 31 | 172 | – | – | |
| cselect[3] | 27 | 173 | – | – | |
| nreset | 182 | 78 | – | C | |
| wr | 206 | 206 | – | 5 | Write enable (0=write, 1=read) |
| cs | – | 199 | – | – | Chip select of the 128K SRAM |

Cartridge control signals

| Name | 1K30 | 1K100 | Ext. | Cart. | comments |
|---|---|---|---|---|---|
| exrom | – | 166 | – | 9 | |
| game | – | 183 | – | 8 | |
| dma | – | 80 | – | 13 | |

| | | | | | |
|---|---|---|---|---|---|
| ba | – | 149 | – | 12 | Requires the BA-Mod (see chapter 11.1) |

## Data and Address lines

| Name | 1K30 | 1K100 | Ext. | Cart. | comments |
|---|---|---|---|---|---|
| mdb[0] | 160 | 160 | 25 | 21 | |
| mdb[1] | 157 | 157 | 24 | 20 | |
| mdb[2] | 158 | 158 | 23 | 19 | |
| mdb[3] | 159 | 159 | 22 | 18 | |
| mdb[4] | 161 | 161 | 13 | 17 | |
| mdb[5] | 162 | 162 | 127 | 16 | |
| mdb[6] | 164 | 164 | 128 | 15 | |
| mdb[7] | 166 | 166 | 129 | 14 | |
| a[0] | 163 | 198 | – | Y | |
| a[1] | 167 | 197 | – | X | |
| a[2] | 168 | 196 | – | W | |
| a[3] | 169 | 195 | – | V | |
| a[4] | 25 | 193 | – | U | |
| a[5] | – | 192 | – | T | |
| a[6] | – | 191 | – | S | |
| a[7] | – | 190 | – | R | |
| a[8] | – | 189 | – | P | |
| a[9] | – | 187 | – | N | |
| a[10] | – | 186 | – | M | |
| a[11] | – | 180 | – | L | |
| a[12] | – | 179 | – | K | |
| a[13] | – | 177 | – | J | |
| a[14] | – | 176 | – | H | |
| a[15] | – | 175 | – | F | |
| a[16] | – | 202 | – | – | |

## I/O

| Name | 1K30 | 1K100 | Ext. | Cart. | comments |
|---|---|---|---|---|---|
| keyb-clk | 203 | 70 | – | – | |
| keyb-dat | 202 | 69 | – | – | |
| mouse-clk | 200 | 68 | – | – | |
| mouse-dat | 199 | 67 | – | – | |
| ioshift | – | 174 | – | – | serial joystick data from CPLD |

## GBridge

| Name | 1K30 | 1K100 | Ext. | Cart. | comments |
|---|---|---|---|---|---|
| gb[0] | 36 | 87 | – | – | |
| gb[1] | 37 | 86 | – | – | |
| gb[2] | 38 | 85 | – | – | |
| gb[3] | 39 | 83 | – | – | |
| gb[4] | 40 | 75 | – | – | |
| gb[5] | 41 | 74 | – | – | |
| gb[6] | 44 | 73 | – | – | |
| gb[7] | 45 | 71 | – | – | |

## Video (extender)

| Name | 1K30 | 1K100 | Ext. | Cart. | comments |
|---|---|---|---|---|---|
| v-sync | 197 | – | 39 | – | On extender it is nVSync (negative logic) |
| h-sync | 198 | – | 33 | – | On extender it is nHSync (negative logic) |
| vid[0] | 193 | – | 91 | – | red[0] in bypass (input only on extender) |
| vid[1] | 192 | – | 90 | – | red[1] in bypass (input only on extender) |
| vid[2] | 191 | – | 89 | – | red[2] in bypass (input only on extender) |

| | | | | | |
|---|---|---|---|---|---|
| vid[3] | 190 | – | 88 | – | red[3] in bypass (input only on extender) |
| vid[4] | 189 | – | 79 | – | red[4] in bypass |
| vid[5] | 187 | – | 76 | – | grn[0] in bypass |
| vid[6] | 186 | – | 71 | – | grn[1] in bypass |
| vid[7] | 180 | – | 68 | – | grn[2] in bypass |
| vid[8] | 179 | – | 66 | – | grn[3] in bypass |
| vid[9] | 177 | – | 60 | – | grn[4] in bypass |
| vid[10] | 176 | – | 58 | – | grn[5] in bypass |
| vid[11] | 175 | – | 54 | – | blu[0] in bypass (input only on extender) |
| vid[12] | 174 | – | 53 | – | blu[1] in bypass (input only on extender) |
| vid[13] | 173 | – | 52 | – | blu[2] in bypass (input only on extender) |
| vid[14] | 172 | – | 46 | – | blu[3] in bypass |
| vid[15] | 170 | – | 43 | – | blu[4] in bypass |
| red[0] | – | – | 86 | – | |
| red[1] | – | – | 85 | – | |
| red[2] | – | – | 83 | – | |
| red[3] | – | – | 80 | – | |
| red[4] | – | – | 77 | – | |
| grn[0] | – | – | 72 | – | |
| grn[1] | – | – | 69 | – | |
| grn[2] | – | – | 67 | – | |
| grn[3] | – | – | 65 | – | |
| grn[4] | – | – | 64 | – | |
| grn[5] | – | – | 59 | – | |
| blu[0] | – | – | 49 | – | |
| blu[1] | – | – | 51 | – | |
| blu[2] | – | – | 50 | – | |
| blu[3] | – | – | 44 | – | |
| blu[4] | – | – | 42 | – | |

## Various

| Name | 1K30 | 1K100 | Ext. | Cart. | comments |
|---|---|---|---|---|---|
| power-down | 28 | – | – | – | Pull low to turn power off |