

28-Jan-04 (1)

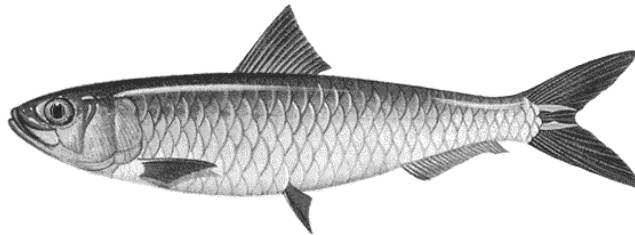
*CEG5010: Pilchard - A
Reconfigurable Computing Platform
with Memory Slot Interface*

“See it big, and keep it simple.”
- Wilfred Peterson

28-Jan-04 (2)

Pilchard

- WA Pilchard (*Sardinops sagax neopilchardus*)
 - Member of the herring family also called sardines
- Small, cheap, abundant baitfish which is an important part of the foodchain



European sardine (*Sardina pilchardus*)

28-Jan-04 (3)

Overview

- Introduction
- Pilchard's design
- Transfer rate experiments
- DES application
- Future work
- Conclusions

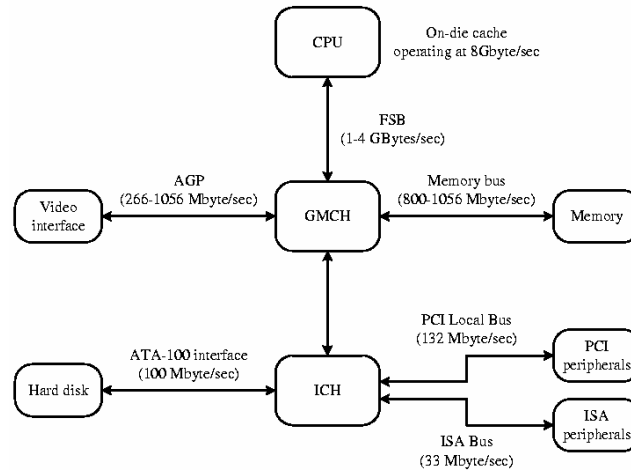
28-Jan-04 (4)

Introduction

- Most reconfigurable computing systems are interfaced to computer via PCI bus
- PCI bus has not kept up with uP technology ($\approx 1\text{GHz}$)
 - PCI32: 32-bit 33 MHz, 133 MB/s (desktop)
 - PCI64: 64-bit 66 MHz, 528 MB/s (server)
 - PCI-X: 64-bit 133 MHz, 1056 MB/s (future servers)
- Bottleneck for reconfigurable computing systems
 - e.g. Patterson DES implementation: 1.2 GB/s

28-Jan-04 (5)

PC Bus Bandwidths (Intel 815E Chipset)



28-Jan-04 (6)

Design Goals

- Hardware
 - DIMM Memory slot interface for high bandwidth, low latency (**64-bit, 133MHz bus**)
 - Low cost
 - Operate on low-end desktop PCs (low cost system)
 - Support any PQ240 Virtex or Virtex-E FPGA
- Interface
 - Use open source software (Linux)
 - Simple registers (high performance without DMA)
 - User-mode (does not require a complicated Kernel driver)

28-Jan-04 (7)

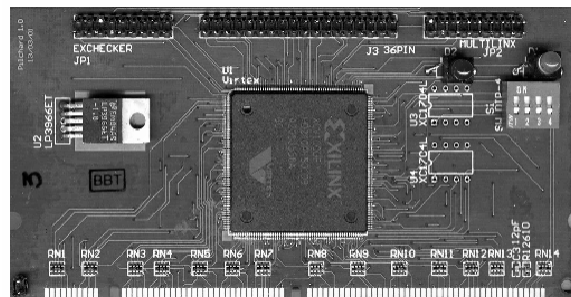
Disadvantages of DIMM slot approach

- Difficult to make work for different PC chipsets
 - We only support one particular motherboard although others which use the Intel 815E chipset should work as well
- Cannot be bus master

28-Jan-04 (8)

Pilchard Board

- Card for Dual inline memory module (DIMM) slot
- 6 layer impedance controlled FR4 board



28-Jan-04 (9)

PCB Design

- All I/O is LVTTTL
- Registered DMM specs: min/max rise time 1.0/2.5 V/ns
 - Tr is 10-90% of Vdd, approx 1ns
- Length of rising edge ($l=tr/D$)
 - $D=140-180\text{ps/inch}$ (inner trace), 180ps/inch (outer) for an FR4 PCB
 - Signal travels 5.5 inches in a nanosecond (l)
 - txline if trace $> l/6 = 0.9''$All traces longer than 0.9" must be terminated!

28-Jan-04 (10)

Power Distribution

- Vcco is 3.3V
- Vccint is 1.8V (generated using a 3A LP3966ET-1.8 linear regulator from 3.3V)

28-Jan-04 (11)

Bypassing

- Vcco
 - 24 Vcco PWR/GND pairs each has a 0.1uF X7R directly underneath
 - plus a 47uF AVX TPS series tantalum (see XAPP158)
- Vccint
 - 12 Vccint each has a 0.1uF X7R
 - 6 47uF AVX TPS series tantalum + 1 470uF tantalum for power supply
- PCB is a very low impedance capacitor

28-Jan-04 (12)

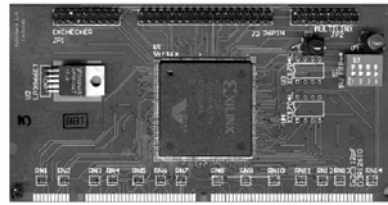
GND/GND connections

- Recall $l=5.6''$
- Rule of thumb: we want a GND/GND via every $l/6$ (i.e. every inch)

28-Jan-04 (13)

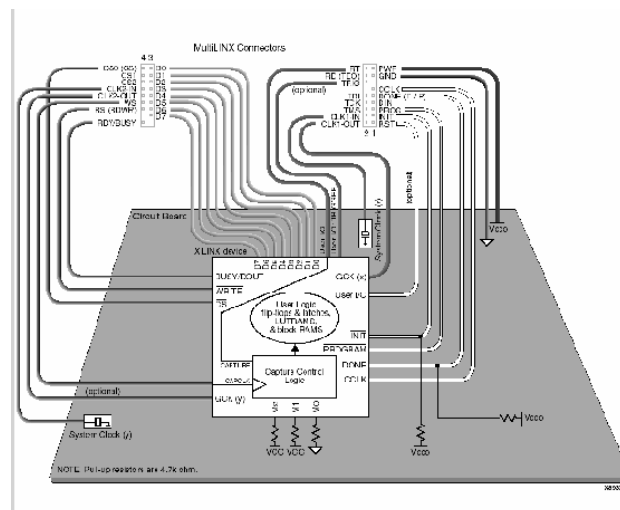
Downloading and Debugging

- Interfaces
 - Xchecker: Download only
 - Multilyn: Download, Readback, Single stepping
- 25 pin Standard size header for connection to a logic analyzer, cable or daughter board
- Optional configuration PROMs
 - Turnkey designs



28-Jan-04 (14)

Multilyn circuit



28-Jan-04 (15)

PC Motherboard

- ASUS CUSL2 motherboard
 - Intel 815EP chipset well documented
 - Supports 133MHz and 100MHz DIMMs
 - Selectable via jumpers so we can use the slower 100MHz clock if necessary
 - 3 DIMM slots
- Low cost (low end desktop class machines)
 - Complete machine with 800MHz PIII processor, disk, memory etc < US\$800 (not including FPGA card)

28-Jan-04 (16)

Boot Process and DIMM Identification

PC Bootup sequence

1. BIOS: Serial presence detect
 - Normal DIMM card must tell BIOS about its characteristics (size, speed etc) using an onboard serial PROM
2. BIOS: Memory test

Problem: Pilchard board is not memory

- How do we bypass the memory test?

28-Jan-04 (17)

Pilchard Bootup Sequence

- PC boots as if nothing in DIMM slot
 - Gets around memory test from BIOS
- Motherboard chipset registers programmed after booting to indicate presence of a memory card
 - Necessary in order to make chipset hardware map memory accesses to the DIMM slot
 - Pilchard address space memory mapped

28-Jan-04 (18)

Operating System Interface Device Driver

- Linux 2.4 loadable Kernel module
- Pilchard board can interface to software via
 - User developed device driver specific to the design
 - Standard driver which allows user mode access to the Pilchard board via a mmap(2) interface
 - Can access the Pilchard board's registers directly without incurring overhead of a system call
- Pentium does not have 64-bit registers so how can we do 64-bit transfers to the Pilchard card?
 - MMX "movq" instruction

28-Jan-04 (19)

Pentium MTRRs

- On Pentium, different memory regions can be of different types
 - Uncacheable - all reads and writes appear on bus in same order as the program
 - Write combining - writes combined in buffer and perhaps merged later and transferred more efficiently as 64-bit cycles and/or bursts (speculative reads, out of order writes, caching of writes)
- WC offers much higher performance

28-Jan-04 (20)

User program - main

```
int main (int argc, char *argv[])
{
    int fd;
    static char *memp = NULL;

    fd = open(DEVICE, O_RDWR);
    if ((memp = (char *)
        mmap(NULL, MTRRZ, PROT_READ, MAP_PRIVATE, fd, 0))
        == MAP_FAILED)
    {
        perror(DEVICE);
        return 1;
    }
    cochlea(memp);
    munmap(memp, MTRRZ);
    close(fd);
    return 0;
}
```

28-Jan-04 (21)

User program – accessing board

```
typedef struct
{
    int w[2];
} int64;

void write64(int64 twrite, char *addr);
void read64(int64 *data, char *addr);
```

- Note addr is a char *
- Note that 64 bit writes may be broken into 2x 32 bit writes (must qualify with dmask)

28-Jan-04 (22)

Interfacing a User Designed FPGA Core to Pilchard

- Interfacing a user designed core to an FPGA board often difficult
- Pilchard uses a memory mapped register based interface
 - Minimal Pilchard interface is 2 slices since most of the registers etc are in IOBs (c.f. LogiCORE PCI64 300 slices)
 - Registers manipulated via a memory mapped region in the user's program
 - 25 signals go out to the I/O header (can use for debugging via a logic analyzer)

28-Jan-04 (23)

Pilchard interface - pcore

```
entity pcore is
port (
  clk: in std_logic;
  clkdiv: in std_logic;
  reset: in std_logic;
  read: in std_logic;
  write: in std_logic;
  readp: in std_logic;
  writep: in std_logic;
  addr: in std_logic_vector(13 downto 0);
  din: in std_logic_vector(63 downto 0);
  dout: out std_logic_vector(63 downto 0);
  dmask: in std_logic_vector(63 downto 0);
  extin: in std_logic_vector(25 downto 0);
  extout: out std_logic_vector(25 downto 0);
  extctrl: out std_logic_vector(25 downto 0) );
end pcore;
```

28-Jan-04 (24)

Pilchard interface – using pcore

```
process(clk, reset)
begin
  if rising_edge(clk) then
    if read0 = '1' then
      -- read in cochlea data
      dout(63 downto 0) <= muxdata;
      addrcount <= addrcount + 1;
    elsif read1 = '1' then
      -- read in counter
      dout(3 downto 0) <= addrcount;
    elsif write2 = '1' then
      -- clear counter
      addrcount <= (others => '0');
    end if;
  end if;
end process;
```

28-Jan-04 (25)

Pilchard interface – generation of read and write

```
read0_p <= '1' when readp = '1' and addr(0) = '1' else '0';
write0_p <= '1' when writep = '1' and addr(0) = '1' else '0';
```

```
U_read0: FDC port map (
  C => CLK,
  CLR => RESET,
  D => read0_p,
  Q => read0 );
```

```
U_write0: FDC port map (
  C => CLK,
  CLR => RESET,
  D => write0_p,
  Q => write0 );
```

28-Jan-04 (26)

Pilchard – generation of logic analyzer outputs

```
-- logic analyzer outputs
extout(7 downto 0) <= sample(7 downto 0);
extout(15 downto 8) <= channel_0(7 downto 0);
extout(16) <= clk;
extout(17) <= clkdiv;
extout(18) <= reset;
extout(19) <= sample_ctrl(0);
extout(20) <= channel_clear(0);
extout(25 downto 21) <= mux16_a(4 downto 0);
extctrl <= (others => '0');
```

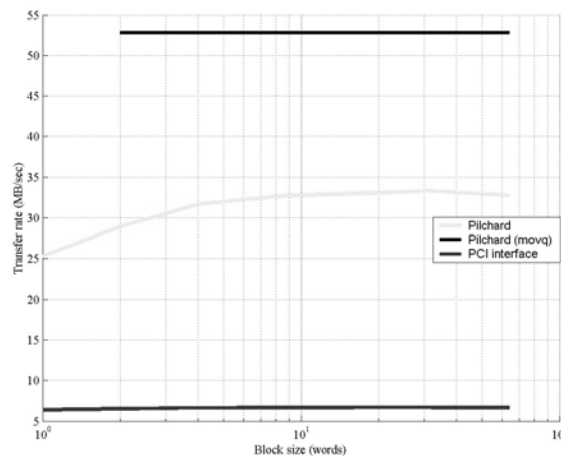
28-Jan-04 (27)

Transfer speed experiments

- All experiments used an ASUS Intel 815EP based computer with 800MHz PIII processor
- Speed of block transfers measured on Pilchard compared with a PCI board using the Xilinx REAL 64/66 PCI LogiCORE V3.0
 - Measurements including all software overheads
 - Could not try 64-bit PCI transactions since motherboard did not support PCI64 (remember: currently only server class machines support PCI64)
 - Could not use DMA since no driver supported it
 - Above 2 features could significantly improve PCI performance

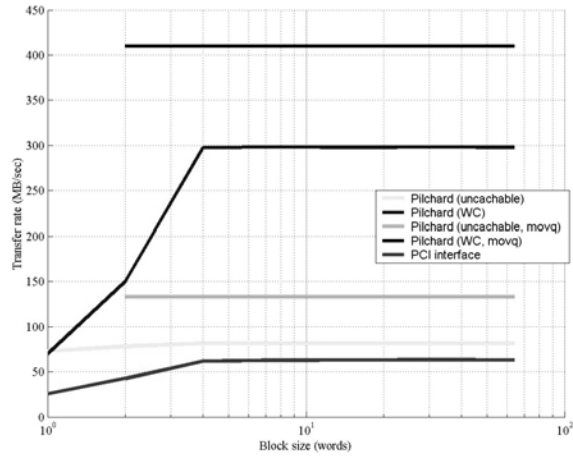
28-Jan-04 (28)

Read performance



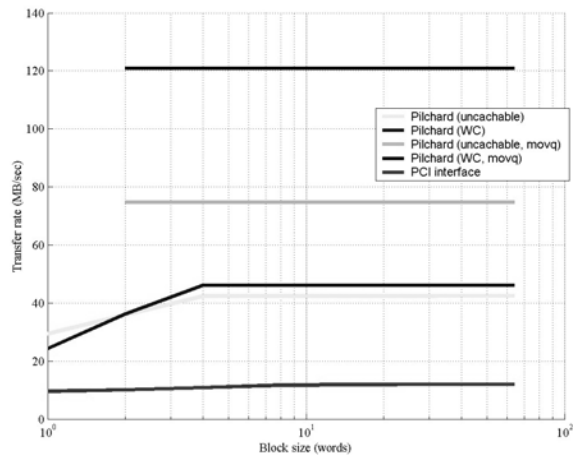
28-Jan-04 (29)

Write performance



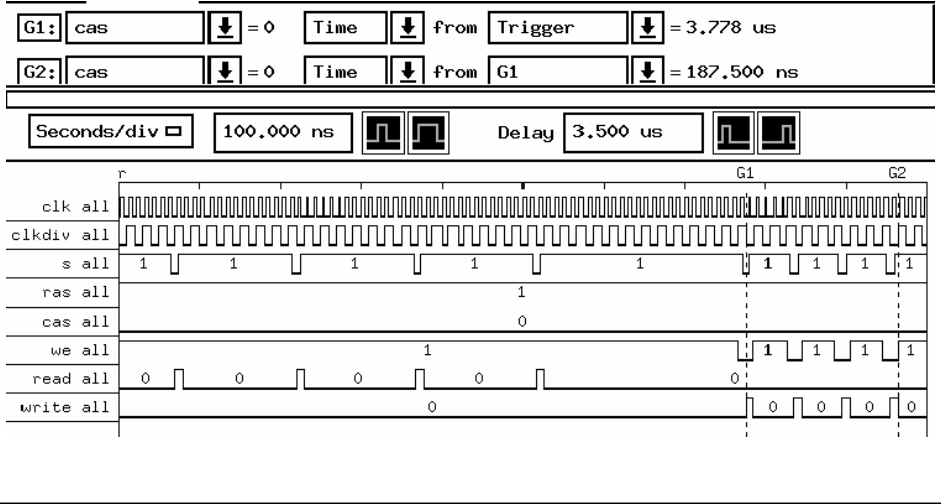
28-Jan-04 (30)

Read/Write Performance



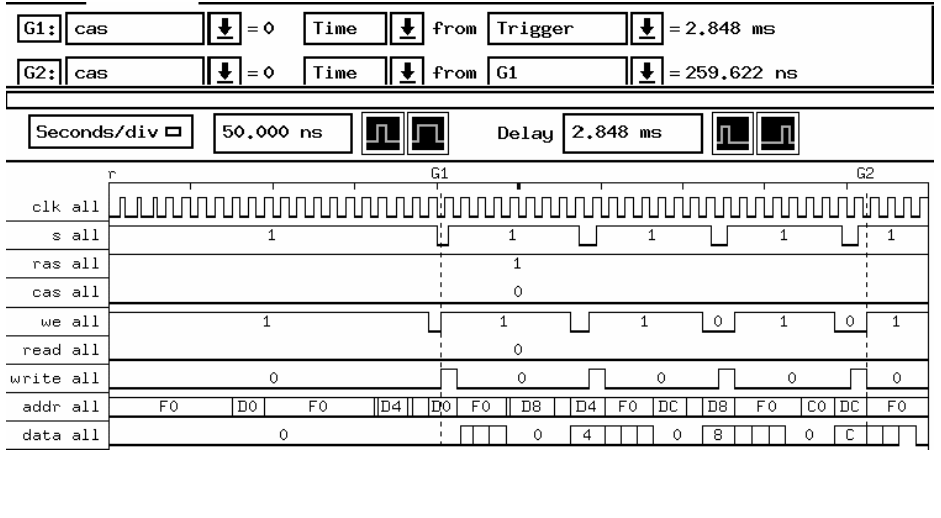
28-Jan-04 (31)

Logic analyzer traces uncacheable



28-Jan-04 (32)

Logic analyzer traces Write Combining



28-Jan-04 (33)

DES Core

- DES core (ECB mode) interfaced to Pilchard board
 - 64-bit 66MHz (528 MB/s design)
 - Average Pilchard transfer rate does not exceed DES core rate so PC can read/write at full rate
 - Interface
 - PC writes plaintext to a 64-bit BlockRAM on Pilchard at 133MHz
 - DES core reads from the BlockRAM at 66MHz, encrypts and writes ciphertext to another BlockRAM
 - PC reads ciphertext from BlockRAM
 - Uncacheable MTRR
 - Measured encryption rate with sw overheads: 35MB/s

28-Jan-04 (34)

Future Work

- Virtex II support using BGA packages
- Double data rate (DDR) DIMMs
 - Uses both clock edges to double throughput
- Applications
 - Cryptography, DSP, Beowulf clusters of Pilchard equipped PCs, interconnect, systolic arrays
- Education
 - Make Pilchard available to reconfigurable computing community for teaching and research
 - Suitable since it is low cost, easy to use, high performance

Conclusions

- On any computer system, memory interface should be faster than I/O interface
- Pilchard system demonstrates the feasibility of DIMM slot based FPGA board
 - Offers simpler interface and better performance than PCI32
- 35MB/s DES system demonstrated as a sample application