Pilchard – A Reconfigurable Computing Platform with Memory Slot Interface

P.H.W. Leong, M.P. Leong, O.Y.H. Cheung, T. Tung, C.M. Kwok, M.Y. Wong, K.H. Lee
{phwl,mpleong,yhcheung,ttung,cmkwok,mywong,khlee}@cse.cuhk.edu.hk
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, NT Hong Kong

Abstract

A reconfigurable computing development environment called Pilchard, employing a field programmable gate array (FPGA) which plugs into a standard personal computer's (PC) 133 MHz synchronous dynamic RAM Dual In-line Memory Modules (DIMMs) slot is presented. Compared with a traditional PCI interfaced reconfigurable computing board, the DIMM interface offers higher bandwidth, a simpler interface and lower latency. A comparison of the transfer rate of the Pilchard board compared with a standard PCI32 reconfigurable computing board is presented as well as an implementation of the data encryption standard (DES). Together, the board and interface generator provide an easy to use, low cost and high performance platform for reconfigurable computing.

1 Introduction

Reconfigurable computing (RC) exploits the reprogrammable nature of field programmable gate array (FPGA) devices to perform computing. The hardware design implemented on the FPGA can usually operate with a level of parallelism much higher than that achievable in software. In many cases, the FPGA acts as a coprocessor for a host personal computer (PC) which provides data to the FPGA.

RC systems can be implemented with shorter development times and lower cost than an equivalent custom VLSI chip, yet have proved to be the fastest or most economical way to solve certain problems in DNA sequence matching, signal processing, emulation and cryptography [1].

Unfortunately, current bus technology has not kept pace with improvements in FPGA and microprocessor technology. Although FPGA systems can operate at clock frequencies over 100 MHz and microprocessors operate above 1 GHz, standard PCI bus technology lags behind. The speed of a coprocessor system is often limited not by the speed of the FPGA circuit, but by the interconnecting bus between the processor and the FPGA board. An example of such a system is our 62 MB/s implementation of the IDEA cipher which could only achieve 4.9 MB/s without DMA through a CardBus interface [2]. Other implementations such as a 1.2 GB/s implementation of DES [3] would also certainly be bottlenecked by the PCI bus.

Although server class machines employ the higher speed, higher bandwidth 64-bit, 66 MHz PCI64 bus, the majority of personal computers still use the original 32-bit, 33 MHz PCI32 bus which has a maximum transfer rate of 132 MB/s. Many manufacturers of reconfigurable computing hardware have not yet updated their designs to support PCI64. In the future, machine will use the PCI-X specification which is a 64-bit bus operating at 133 MHz and hence has a maximum throughput of 1064 MB/s.

In any balanced PC or workstation, the memory bandwidth is higher and of a lower latency than that of the peripheral bus. This is because memory accesses are made much more frequently than input/output (I/O) requests. As an example, the standard dual inline memory modules (DIMM) used even in low-end PCs operate at either 100 MHz or 133 MHz with 64-bit data, providing a maximum bandwidth of 1064 MB/s.

To address the PC/FPGA bandwidth issue, a DIMM based reconfigurable computing platform called "Pilchard" (not an acronym but named after the Western Australian Pilchard, *Sardinops sagax neopilchardus*, a small, cheap and abundant baitfish which is an important part of the food chain). Pilchard was developed with the following design goals

- achieve higher bandwidth and lower latency than traditional PCI based RC systems

- few components and low cost, facilitating its use in educational and research applications
- operate on low-end PC motherboards
- support Xilinx Virtex and Virtex–E FPGAs
- allow connection of memory and/or peripherals via a daughter card
- able to take advantage of Pentium write combining features for improved throughput
- use a simple register based interface for simple interfacing
- uses the Linux operating system
- user mode programs can access the board via a simple yet efficient `mmap()` based scheme.

In this paper, the Pilchard system's design is presented, its performance is compared with that of a PCI32 reconfigurable computing card and an implementation of the data encryption standard (DES) on the Pilchard board is described. The Pilchard card is similar to the commercial Nuron AcB [4], but was developed independently. The Nuron product appears to have 64 MB of onboard SDRAM and a DIMM bus download capability.

The rest of the paper is organized as follows, in Section 2, the hardware design is described. Section 3 describes issues dealing with the Linux Operating System interface to the Pilchard system. Experiments conducted using the Pilchard prototype including a performance comparison with a PCI32 card and DES implementation are presented in Section 4. Plans for future work are discussed in Section 5 and finally, conclusions are drawn in Section 6.

2 Pilchard Hardware Design

A block diagram of the Pilchard board is shown in Figure 1. The main components are the FPGA, FPGA bitstream download and debug interface, configuration PROM interface and an expansion header which is used for connection to a logic analyzer or interfacing to other peripheral or memory devices. The logic for the DIMM memory interface and clock generation is implemented in the FPGA.

The board can be populated with any Virtex or Virtex–E device in a PQ240 or HQ240 package. This currently ranges from the XCV150 to XCV1000E devices.

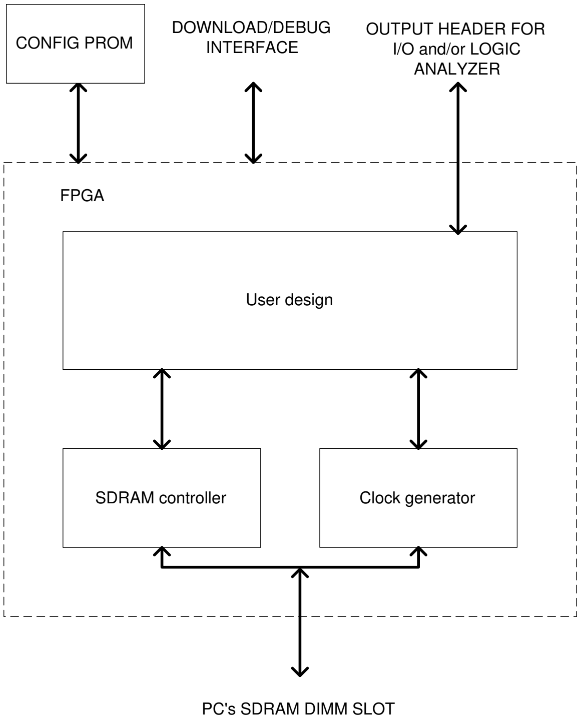


Figure 1: Block diagram of the Pilchard board.

A photograph of the populated Pilchard board is shown in Figure 2. As can be seen from the picture, a minimal number of components are used and the only expensive component is the FPGA.

2.1 DIMM Interface

The Pilchard board was designed to be compatible with the 168 pin 3.3 Volt, 133 MHz, 72-bit, registered synchronous DRAM in-line memory modules (SDRAM DIMMs) PC133 standard [5, 6]. Since the pinouts are the same as the 66/100 MHz DIMM, the PC100 standard can also be supported. The PC133 standard supports 64 MB, 128 MB, 256 MB, 512 MB and 1 GB capacities which offers ample space for memory mapped I/O.

2.1.1 Printed Circuit Board

The nominal dimensions of a standard DIMM card are 133.37 × 38.12 × 1.27 mm. However, a Virtex PQ240 device is 32 × 32 mm in size so it was not possible to use a standard sized DIMM card. The Pilchard board is 133.37 × 70.39 × 1.27 mm, roughly double the height of a standard DIMM card.

The printed circuit board is a 6 layer impedance controlled FR4 board, designed in-house and manu-

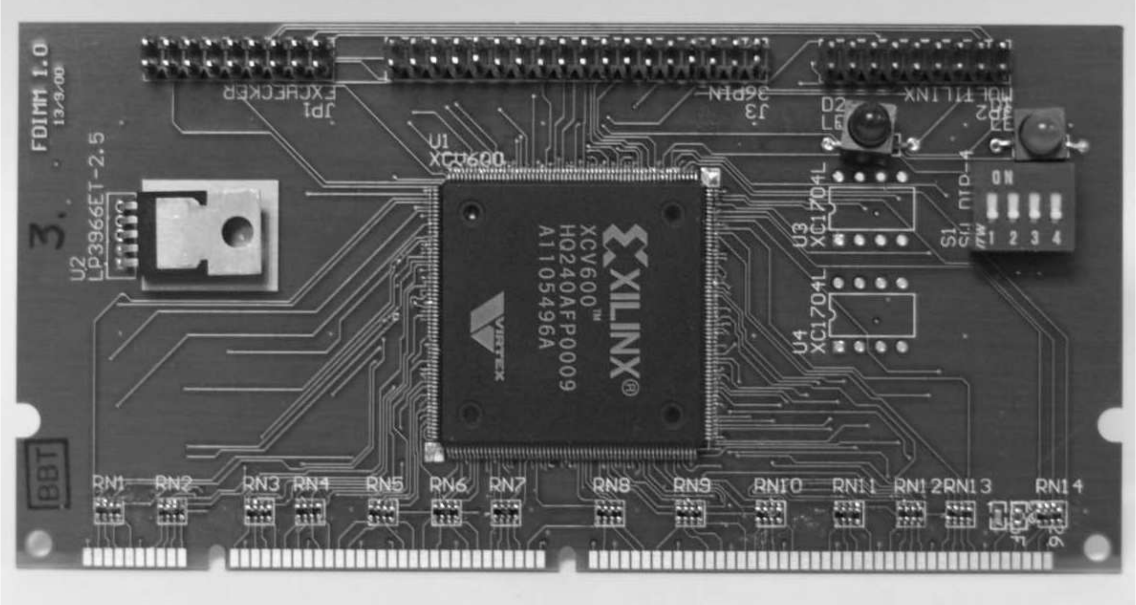


Figure 2: Photograph of the Pilchard board.

factured by a third party. The small number of layers is a direct consequence of our decision to use FPGAs in the PQ240 package. An added advantage of using PQ240 devices is that hand soldering of the board is possible.

An expansion header connected to 25 general purpose I/O pins of the FPGA is supplied. This header was designed to connect either to a logic analyzer (for debugging purposes), or to enable daughter boards with memory or other peripheral chips to be connected to the Pilchard board.

2.1.2 Serial Presence Detect

The PC133 DIMM standard includes a mandatory serial presence detect (SPD) interface [7] which allows a DIMM card to describe its configuration to the PC. The PC's BIOS interrogates each DIMM slot's SPD to determine the presence or absence of a card, its timing parameters and its size. It then performs a memory test on all available memory before proceeding with the rest of the boot process.

The memory test described above provides an obstacle for a non-memory card, particularly if the card does not have sufficient memory on board to completely mimic a normal DIMM. A possible solution would be to modify the BIOS so that it does not perform a memory test, however, this is very difficult to do when the source code to the BIOS is not available. It is also motherboard and BIOS dependent, hence different machines would need to be patched differently.

Our solution to this problem is to allow the machine to boot normally. The Pilchard board does not have SPD so the BIOS identifies the slot as an empty slot. Once the operating system has been booted, the PC's chipset registers are modified via a device driver (see Section 3) to enable operation of the DIMM slot. Although this technique is also dependent on the PC's chipset, it can be easily done for any chipset which has adequate documentation of its memory controller registers (see Section 2.4 for motherboard details).

2.1.3 SDRAM Controller

We have developed an interface generator that produces simple yet efficient interfaces between user designs and the device driver. The generator reads a simple configuration file and emits VHDL code that implements the interface.

Commands which can be sent from the motherboard to the DIMM are COMMAND INHIBIT (NOP), NO OPERATION (NOP), ACTIVE (Select bank and active row), READ (Select bank and column, start READ burst), WRITE (Select bank and column, start WRITE burst), BURST TERMINATE, PRECHARGE (deactivate row in bank), AUTO

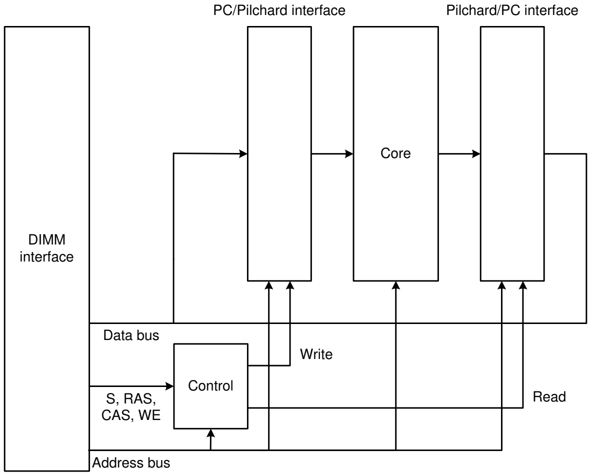


Figure 3: Architecture of PC–FPGA interface.

REFRESH, LOAD MODE REGISTER, Write Enable/Output Enable and Write Inhibit/Output High-Z [8]. In the context of a memory mapped device, most of these commands can be ignored. Simplifications were made to the SDRAM controller to reduce its complexity without reducing functionality.

The interface to a user's core design consists of two parts, namely the PC-to-core interface and the core-to-PC interface. The architecture of the interface is illustrated in Figure 3.

A SDRAM multiplexes its address inputs to save pins and hence addresses are decomposed into banks, rows and columns. Although a typical SDRAM has megabytes of address space, memory mapped peripherals normally use few registers. Hence in the present design, the interface interprets only the 8-bit column addresses. Row and bank select commands are thus ignored by the SDRAM controller. The controller could be changed to handle row and bank selection commands and latch their values to form a full address should the need arise.

The interface uses four control signals from the DIMM interface, namely S (select), RAS (row address strobe), CAS (column address strobe) and WE (write enable) to generate the appropriate board read and write signals. Both the interfaces can be independently configured as registers, a BlockRAM or a direct connection. The register configuration supports up to $2^8 = 256$ 64-bit registers. The BlockRAM configuration uses a 256 × 32-bit dual-port BlockRAM. With dual-port BlockRAMs the internal core can operate at a different clock rate than the 133 MHz DIMM interface. The direct connection configuration is a simple bypass between the input and the output of the interface. The advantage of the register configuration is that the core has simultaneous accesses to all memory location. The advantage of the BlockRAM configuration is its reduced area overhead. The direct connection configuration requries minimal area (for example, only 2 Virtex slices are required if both interfaces are configured as direct connection), but it requires the core to decode the address bus by itself.

A minimal 64-bit Pilchard interface requires only 2 Virtex slices to generate the board's READ and WRITE signals. Latching of the address and data buses are performed in the input-output blocks (IOBs). Registers and BlockRAMs used for interfacing purposes require additional resources. In contrast, the Xilinx LogiCORE PCI64 interface uses 300–350 slices [12].

2.2 Clock Generation

The only clock input to the FPGA is that supplied by the SDRAM interface. This 133 MHz clock is deskewed using a high frequency delay locked loop (CLKDLLHF) within the Virtex chip [9]. It can also be divided down inside the FPGA and multiplied by another DLL to generate different frequencies.

2.3 Downloading and Debugging

The Pilchard system currently requires connection to another machine via a Xilinx Xchecker or Multilynx cable [9] for bitstream download since downloading via the DIMM bus was not incorporated in the prototype and we do not have a Linux version of the download software for use from the same machine. Future versions will use a second programmable logic device to enable downloading via the DIMM bus, and we will also develop a method to download the bitstream from Linux. Readback and single stepping via the Multilynx cable is also supported.

Optional configuration PROMs are also supported so that once a design has been completed, it can be placed in PROMs and automatically downloaded upon powerup. It is currently not possible to download a bitstream to the Pilchard system via the DIMM bus interface.

2.4 PC Motherboard

For all the results described in this paper, an ASUS CUSL2-C motherboard using the Intel 815EP chipset was used. Important features of this motherboard are

- it supports 133 MHz SDRAM and 100 MHz SDRAM
- the Intel 815EP chipset is well documented
- it has 3 DIMM slots and hence supports configurations with one DIMM memory card and either one or two Pilchard boards
- it supports Intel Pentium III Coppermine and Intel Celeron processors
- it is a low cost desktop type motherboard.

Note that for Pilchard designs which cannot meet a 133 MHz interface timing constraint, it is possible to use the Pilchard board at 100 MHz via dip switch settings on the motherboard. Unfortunately, this also limits the speed of memory accesses to 100 MHz.

3 Operating System Interface

A simple Linux device driver was developed which allows user mode programs to access the Pilchard hardware. Although this driver was tested only with Linux kernel 2.2.17, ports to other operating systems and Linux versions should be trivial.

During initialization, the device driver is responsible for programming the PC chipset's memory controller registers to enable the Pilchard's DIMM slot. This fools the motherboard into thinking that the slot is populated with a DIMM memory card and access cycles directed to this portion of the memory space will generate appropriate signals in the DIMM slot.

A user interface using the UNIX `mmap()` system call has been developed. User programs can access the Pilchard board's registers, by performing a `mmap()` call which maps virtual addresses in the user space to the bus address of the Pilchard board. Following this process, the user can manipulate the registers of the Pilchard board directly without incurring the overhead of a system call.

3.1 Memory Cache Control

Central processing unit (CPU) caching of reads and writes to Pilchard registers could lead to incorrect results. The Intel Pentium Pro, Pentium II and Pentium III has a Memory Type Range Register (MTRR), accessible from Linux, which allows different memory regions to be of different types [10]. MTRRs can be easily manipulated under Linux via the /proc/mtrr interface [11].

The "Uncacheable" memory type guarantees that all reads and writes will appear on the system bus in the same order as the program. Furthermore, no speculative memory accesses, page-table talks or prefetches of speculated branch targets will occur [10]. Although the most conservative, it also leads to the lowest performance.

The "Write Combining" (WC) memory type allows 32-bit writes to be delayed and later merged together in write-combining buffers. It is typically used to improve the performance of frame buffers for graphics. Upon reaching a serializing event such as a read from an uncacheable location, the write-combining buffer is flushed in an efficient manner. For example, when a WC buffer becomes full, the processor will evict the buffer to system memory in a single burst transaction of 64-bit writes [10]. As will be seen in the results section, careful use of the WC memory type can lead to greatly improved performance. The performance achieved in these different modes are discussed later in Section 4. WC was not used on our board for reads since this mode allows speculative reads which could interfere with memory mapped read cycles that have side effects. This was achieved by using different address regions for reads and writes with their MTRRs set to uncacheable and WC respectively.

4 Results

The performance of the Pilchard board using an XCV300–6 FPGA was compared with a PCI32 board also using an XCV300–6 device. The PCI board used the Xilinx REAL 64/66 PCI LogiCORE V3.0 [12] for its PCI controller. All the experiments presented in this section were measured on the same machine, an Asus CUSL2 motherboard (Intel 815EP chipset) with 800 MHz Pentium III processor and 32-bit PCI slots (the 64-bit PCI card was used in a PCI32 slot in backwards compatible mode). All tests were conducted with the DIMM slot operating at 133 MHz.

Unfortunately, since our Linux driver for the PCI card did not support DMA, its performance in this mode could not be tested. DMA would certainly offer better performance for large blocks, however, there is a large overhead associated with setting up DMA transactions.

All testing for PCI and Pilchard was performed via a Linux loadable kernel module device driver which ensures that the best performance was achieved. Memory transfers were performed using the `memcpy()` kernel function. For all of the measurements below, the Linux kernel function `do_gettimeofday()` was used to perform timing and the results reflect all associated software overheads.

4.1 Throughput Measurements

A simple design was used to measure the I/O performance of the Pilchard board. In this design, read and write cycles cause the lower and upper 32 bits of a 64 bit register on the Pilchard card to be incremented. Upon completion of the benchmark, the number of reads and writes are read back to the PC to verify that all the data were transferred.

4.1.1 Write Benchmark

The write benchmark was conducted by performing $2^{20} = 1048576$ 32-bit writes to blocks of consecutive memory locations on the respective cards. This test was conducted with MTRRs set to WC and uncacheable on the Pilchard board, and uncacheable for PCI.

Measurements of throughput for different 32-bit block sizes are presented in the top half of Table 1 and are plotted in Figure 6. The PCI32 interface is always slower than the Pilchard interface with uncacheable MTRR, particularly for small block sizes. Write combining on Pilchard gives a further three to fourfold performance gain over uncacheable since it is able to combine software 32-bit cycles and write them using 64-bit transfers.

Write performance can be further improved by using the Pentium MMX "movq" instruction to perform 64-bit write transactions. Using assembly language to access the movq instruction, the results in the lower half of Table 1 were obtained.

An Agilent Technologies 16700A logic analyzer was used to capture the waveforms for uncacheable and write combining writes to the Pilchard board (Figures 4 and 5). In the uncacheable case shown in Figure 4, all writes must occur immediately and no write bursts will occur. The highest performance is achieved in the write-combining case of Figure 5 where start of a burst transfer can be identified by the "write" signal being high. In this particular example, it can be seen that 16× 64-bit writes are performed in approximately 300 ns which equates to 426 MB/s, consistent with the measured results in Table 1. As can be seen in Figures 4 and 5, even in a tight loop, DIMM board transfers do not occur on every cycle and this is the main cause of lost efficiency in our system. Hopefully, this will improve with newer processors and chipsets.

The measured WC performance is six times that of the measured PCI bus transfer rate. The maximum bandwidth of a DIMM interface is 1064 MB/s and our best measured performance using WC and including software overheads was 400 MB/s (more than three times the maximum transfer rate of PCI32).

4.1.2 Read Benchmark

Similar to the write benchmark, the time taken to perform $2^{20} = 1048576$ reads of uncacheable memory locations in differently sized blocks of consecutive locations was measured. The read performances for different block sizes are shown in Table 1 and are plotted in Figure 7.

Read cycles (4× 64-bit) can be seen in Figure 4 when the "read" signal is high, and the throughput can be seen to be approximately 64 MB/s which is consistent with a measured value of 52 MB/s in Table 1. We have not been able to produce burst 64-bit read transactions. The read performance is approximately seven times higher than that of PCI.

4.1.3 Read/Write Benchmark

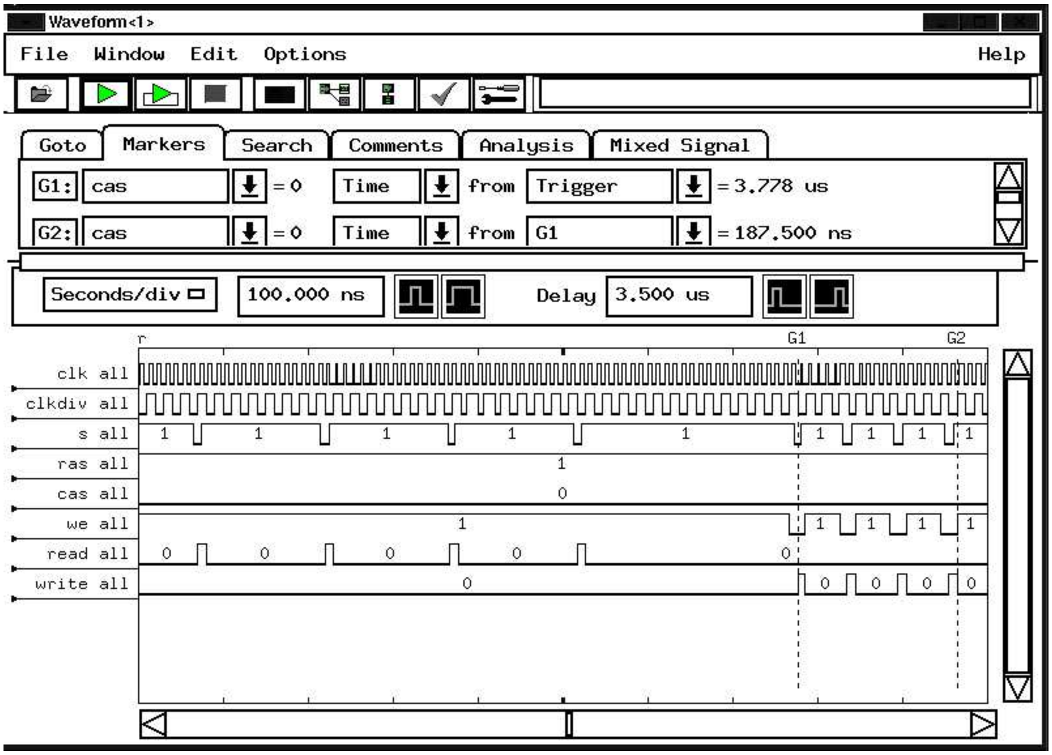
The read/write benchmark involves alternating writes and reads of data. Two separate memory regions were used for this test, reads being made on an uncacheable region and writes to a WC or UC region. Results are shown in Table 1 and the corresponding plots are shown in Figure 8. This mode is even faster than pure read cycles since writes are faster than reads, thus improving the overall transfer rate.

As for read and write cycles, using the "movq" instruction to achieve 64-bit transfers significantly improves the performance and the results are presented in Table 1. The 64-bit Pilchard transfer rates were approximately six to ten times faster than PCI.

4.2 DES Core

A straightforward implementation of a fully parallel, pipelined data encryption standard (DES) [13] in electronic codebook (ECB) mode but with fixed key was used to verify the correctness and test the reliability of the Pilchard system. The DES core is capable of operating at 66 MHz on 64-bit data and thus has a maximum bandwidth of 528 MB/s. More optimized DES implementations with 1.2 GB/s throughput have been reported [3] however, in this application, a faster core is unnecessary since the Pilchard system cannot perform data transfers at that rate. The implementation used a total of 1895 Virtex slices.

As expected, the kernel mode performance was the same as that for the Pil/UC-RW case of Table 1.



Waveform<1>

G1: cas = 0 Time from Trigger = 3.778 us

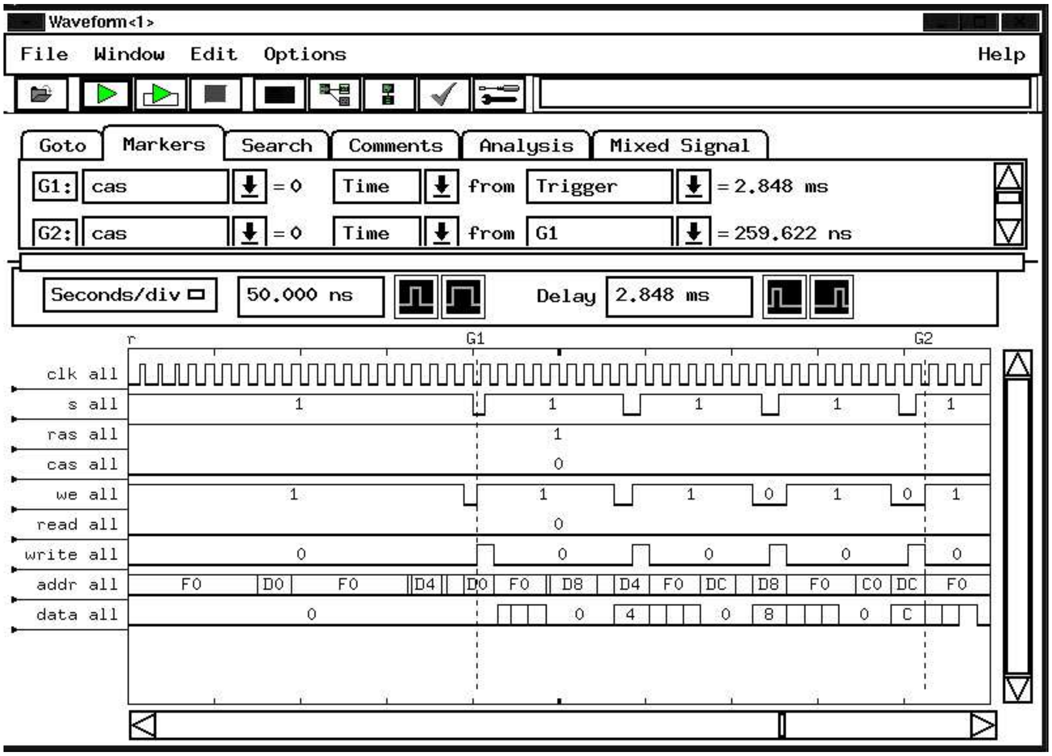
G2: cas = 0 Time from G1 = 187.500 ns

Seconds/div 100.000 ns Delay 3.500 us

Figure 4: Logic analyzer trace showing 64-bit Pilchard read and write cycles to uncacheable memory regions using the "movq" instruction. "clk" is a 133 MHz clock; "s", "ras", "cas" and "we" are the DIMM interface signals; and "read" and "write" are the Pilchard decoded read and write signals. The trace shows 4 64-bit reads followed by 4 64-bit writes.

32–BIT MEMCPY TRANSFERS								
Block size	Transfer rate (MB/sec)							
(words)	Pil/UC	Pil/WC	Pil/RD	Pil/UC-RW	Pil/WC-RW	PCI/WR	PCI/RD	PCI/RW
1	72.80	69.83	25.29	29.47	24.34	25.47	6.37	9.62
2	78.22	150.02	28.93	35.93	36.18	42.45	6.53	10.07
4	81.51	297.53	31.67	42.43	46.11	61.74	6.61	10.84
8	81.47	298.28	32.65	42.42	46.11	62.68	6.66	11.71
16	81.46	297.93	32.98	42.41	46.11	63.17	6.67	11.90
32	81.49	298.08	33.28	42.42	46.11	63.42	6.67	11.99
64	81.49	297.80	32.76	42.42	46.11	63.17	6.65	12.01
64–BIT MOVQ TRANSFERS								
Block size	Transfer rate (MB/sec)							
(any)	132.88	409.64	52.80	74.49	120.78			

Table 1: Read and write performance of Pilchard and a comparison with the PCI interface. (Pil/UC: Pilchard write performance (with MTRR set to uncacheable), Pil/WC: Pilchard write performance (with MTRR set to WC), Pil/RD: Pilchard read performance, Pil/UC-RW: Pilchard read/write performance (with MTRR set to uncacheable), Pil/WC-RW: Pilchard read/write performance (with MTRR set to WC), PCI/WR: PCI interface write performance, PCI/RD: PCI interface read performance, PCI/RW: PCI interface read/write performance.)



Waveform<1>

G1: cas = 0 Time from Trigger = 2.848 ms

G2: cas = 0 Time from G1 = 259.622 ns

Seconds/div 50.000 ns Delay 2.848 ms

Figure 5: Logic analyzer trace showing bursting behaviour for 64-bit write cycles to consecutive addresses in a write combining memory region. “clk” is a 133 MHz clock; “s”, “ras”, “cas” and “we” are the DIMM interface signals; “read” and “write” are the Pilchard decoded read and write signals; and “addr” and “data” show the Pilchard address and data buses. The trace shows 16 64-bit writes which have been merged in the write combining buffer into 4 burst transactions.

Figure 6: Write performance of Pilchard and PCI interface for different block sizes.

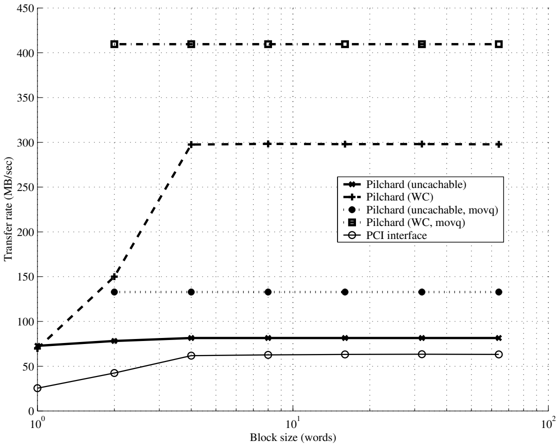


Figure 7: Read performance of Pilchard and PCI interface for different block sizes.

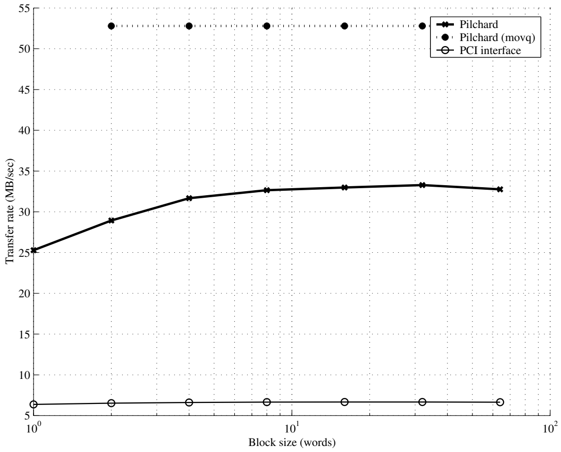
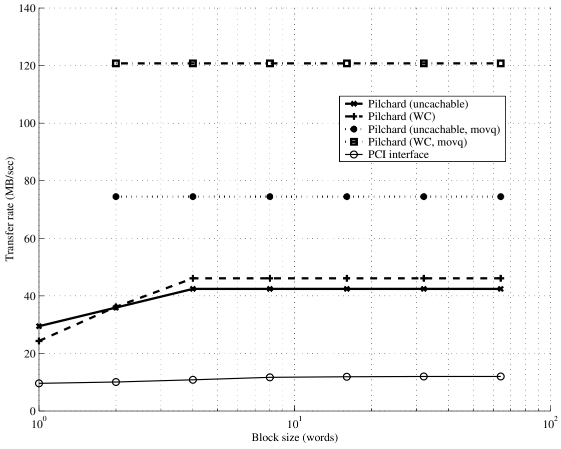


Figure 8: Read/write performance of Pilchard and PCI interface for different block sizes.



However, the encryption rate is half the transfer rate since one encryption involves two transfers (sending the plaintext data and receiving the encrypted data). Hence the resulting kernel mode encryption rate was over 35 MB/s. A user mode version using the `mmap()` interface described in Section 3 has the same performance.

5 Future Work

The availability of a low cost, high speed RC platform opens many opportunities for computer engineering research and education. In this section, some of our future plans are discussed.

The current Pilchard board is limited by the choice of the PQ240 package to XCV1000E devices and 158 I/O pins. Future versions of the Pilchard board may use the BG560 packaged Virtex FPGAs with 404 I/O pins. This has the advantages of more pins being available for the expansion header as well as enabling the use of devices up to the XCV3200E. A ball grid array packaged FPGA will probably increase the number of layers required in the printed circuit board.

Manufacturers such as IBM, VIA, Iwill, Asus and Acer have recently started shipping Pentium III, Athlon and Pentium IV motherboards which support Double Data Rate (DDR) SDRAM. This new technology achieves double the data rate of SDRAM by supporting data transfer on both rising and falling edges of a 133 MHz clock [14] and has a maximum transfer rate of 2128 MB/s. We are planning to use the DDR SDRAM interface in future versions of the Pilchard board.

The Pilchard system was originally developed for applications in FPGA based cryptographic hardware. In such applications, processor/FPGA bandwidth is important since for encryption or decryption, data must be first written to the RC board, processed and read back from the RC board. Since the speed of write transactions is signficantly faster than that of read transactions, the Pilchard system would be particularly effective for computing cryptographic hash functions which do not require large amounts of data to be read back from the RC board.

We feel the best way to develop larger RC systems is not by making cards with large arrays of FPGAs, but by interconnected Pilchard equipped PCs. This has the advantage that the PC's CPU as well as the Pilchard card can be used for processing. The availability of low cost PCs and so-called Beowulf software [15] to support massively parallel clusters of Linux machines provides the software infrastructure to develop powerful applications.

To explore opportunities in this direction, we aim to use the Pilchard system described earlier as a platform for testing parallel applications. Apart from straightforward applications which utilize Pilchard cards as accelerators in Beowulf clusters, we also hope to experiment with using the Pilchard cards as an alternative to high speed interconnect networks (such as Myrinet) for Beowulf style cluster computing. Using the SelectLink communications channels, very high bandwidths (200 Mb/s/pin) can be achieved [16]. We expect that using the Pilchard card as parallel point-to-point computer interconnect, it is possible to achieve higher bandwidth and lower latency than current Myrinet and Gigabit Ethernet technologies. Using this idea, we hope to develop low cost, high speed systolic arrays.

We aim to equip networked Linux PC's with Pilchard cards, in the Department's undergraduate digital systems laboratory. The availability of a low cost, high speed, high capacity RC platform will enable our Department to outfit an undergraduate digital systems laboratory with Virtex FPGAs. Students will be able to use this platform for courses in digital systems, computer architecture, device drivers and reconfigurable computing. Although we considered using commercial RC platforms, we found that the cost prohibitively high. We also hope to collaborate with other Universities to introduce Pilchard cards into their teaching programs.

6 Conclusion

The PC/FPGA interface is a major bottleneck for current reconfigurable computing systems. A solution to this problem was developed in the form of the Pilchard system which demonstrates the feasibility of utilizing the DIMM slots of a standard PC as a bus for attaching a reconfigurable computing card. This was shown to offer greatly improved bandwidth and latency over the ubiquitous PCI bus. The Pilchard system is simpler, uses less resources than conventional systems and may enable the development of reconfigurable systems with lower cost and signficantly improved performance.

References

[1] A. DeHon, "The density advantage of configurable computing," *IEEE Computer*, pp. 41–R49, April 2000.

[2] M. P. Leong, O. Y. H. Cheung, K. H. Tsoi, and P. H. W. Leong, "Bit-serial implementation of the international data encryption algorithm IDEA," in *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pp. 122–131, 2000.

[3] C. Patterson, "High Performance DES Encryption in Virtex FPGAs using JBits," in *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pp. 113–121, 2000.

[4] Nuron. http://www.nuron.com/pdf/nuronacb.pdf.

[5] Intel Corp., *PC SDRAM Registered DIMM Design Support Document Revision 1.2*, 1998.

[6] IBM and Reliance Computer Corp., *PC133 SDRAM Registered DIMM Revision 1.1*, August 1999.

[7] JEDEC, *Configurations for Solid State Memories*, Release 7r8r9.

[8] *Micron MT48LC16M8A2 Synchronous DRAM Datasheet*, November 1999.

[9] Xilinx Inc., *Xilinx Databook*, 2000.

[10] *IA-32 Intel Architecture Software Developer's Manual, Volume 3: System Programming Guide*, 2000.

[11] *Linux v2.2 documentation* . http://www.linuxhq.com/kernel/v2.2/doc/mtrr.-txt.html.

[12] Xilinx Inc., *PCI64 Virtex Interface V3.0 Datasheet*, 2000.

[13] B. Schneier, *Applied Cryptography.* Wiley, 2nd ed., 1996.

[14] Xilinx Inc., *Applications Note XAPP200: Synthesizable 1.6Gbytes/s DDR SDRAM controller*, 2000.

[15] Scyld Computing Corp., *The Beowulf Project*, 2000. http://www.beowulf.org.

[16] Xilinx Inc., *Applications Note XAPP234: Virtex SelectLink Communications Channel*, 2000.