CUBE: A 512-FPGA CLUSTER

Oskar Mencer, Kuen Hung Tsoi, Stephen Craimer, Timothy Todman and Wayne Luk

Dept. of Computing, Imperial College London
{o.mencer,khtsoi,s.craimer,tjt97,wl}@doc.ic.ac.uk

Ming Yee Wong and Philip Heng Wai Leong

Dept. of Computer Science and Engineering
The Chinese University of Hong Kong
{mywong,phwl}@cse.cuhk.edu.hk

ABSTRACT

Cube, a massively-parallel FPGA-based platform is presented. The machine is made from boards each containing 64 FPGA devices and eight boards can be connected in a cube structure for a total of 512 FPGA devices. With high bandwidth systolic inter-FPGA communication and a flexible programming scheme, the result is a low power, high density and scalable supercomputing machine suitable for various large scale parallel applications. A RC4 key search engine was built as an demonstration application. In a fully implemented Cube, the engine can perform a full search on the 40-bit key space within 3 minutes, this being 359 times faster than a multi-threaded software implementation running on a 2.5GHz Intel Quad-Core Xeon processor.

1 Introduction

Reconfigurable gate array technology has been used in many areas for both research and industrial applications; examples include cryptographic systems, architectural exploration, multimedia processing, physical or financial simulation and system emulation. The major advantage of reconfigurable platforms over general purpose processors and ASICs is the balance between circuit level specialization and programming flexibility.

The available resources in field programmable gate array (FPGA) devices increase each year due to Moore's Law with the addition of embedded RAM, DSP block and processor core, but the demand for more programmable resources is even higher as more sophisticated systems are being implemented. A common solution is to use multiple FPGA devices for a single design. In such an environment, design partitioning, data communication and logic configuration become increasingly complicated with the number of devices employed.

Although research on computing systems with large numbers of parallel ICs or large numbers of processing elements on a single IC has been well studied, studies with large numbers of reconfigurable devices have not been fully explored. Practices applied to systems with small numbers of devices are not applicable to systems with hundreds of FPGAs. In particular, issues concerning the clock distribution scheme, data communication paths, configuration requirements and the increasing cost of system debugging requires new ideas and techniques on both hardware construction and development flow.

The lack of a cost effective massive FPGA cluster framework has become an obstacle for researchers exploring the properties and applications on this class of system. In this paper, we describe a massively-parallel reconfigurable platform designed for both advancing research in the field and solving real-world applications. The major contributions of this work include:

- A novel massively-parallel FPGA architecture, called the Cube, is proposed. The architecture balances scalability, flexibility and fault tolerance, providing a low cost, high density and versatile research platform for large scale parallel reconfigurable clusters.
- A Single Configuration Multiple Data (SCMD) programming paradigm is used in the Cube for efficient FPGA configuration. Using SCMD, all 512 FPGAs can be programmed to the same bitstream in parallel within a few seconds which is suitable for constructing large scale systolic processor array.
- A prototype of the Cube platform was built and tested. The hardware consists of multiple boards each hosting 64 Xilinx FPGA devices. The complete system is a 512-node FPGA systolic array with 3.2 Tbps inter-FPGA bandwidth. A standard I/O interface and simulation framework are also provided.
- A key search engine was implemented in the Cube to demonstrate the computing power of the system. With 49152 independent key search cores, it can fully search the 40-bit key space of the RC4 encryption algorithm in 3 minutes.

Section 2 reviews previous work on massively-parallel processing (MPP) platforms. Section 3 details the architecture and design details of Cube platform. Section 4 presents a fully functional 64-FPGA module, the critical component of the Cube platform. Section 5 describes and evaluates an RC4 key search engine for the Cube. Finally, Section 6 presents conclusions and describes future directions of the Cube project.

2 Related Work

The basic idea of MPP is to partition the problem into subtasks and distribute them to different nodes called processing elements (PEs). Total processing time is reduced as computations in the PEs are in parallel. This section reviews some contemporary MPP systems.

In 1994, the first prototype of the GRAPE-4 [1] system for computing the N-body problem in astrophysics was presented. In 1995, the measured peak performance of a completed GRAPE-4 system was reported as 1.08 Tflops [2]. The system had 40 modules, each carrying 48 Hermite AcceleRator Pipeline (HARP) processors. The HARP was a dedicated ASIC for gravitational force computations running at 15MHz. All modules in GRAPE-4 were connected to a central host station through a shared bus. In 2002, the GRAPE-6 system with 1728 to 2048 processors achieved 64 Tflops [3]. Each processor in GRAPE-6 had 4 independent force pipelines. The processors were connected in a hierarchical network including switch boards and Gigi-bit Ethernet. In 2005, an SIMD architecture, Network on Chip (NoC) and other approaches were proposed for the new GRAPE-DR system [4] which targeted Pflops performance. The current GRAPE hardware designs are specialized for gravitational force computations and do not support more general applications.

The Berkeley Emulation Engine 2 (BEE2) system was developed for event-driven network simulation in 2004 [5]. In BEE2, five Xilinx Virtex-II Pro 70 FPGAs were hosted on a single Print Circuit Board (PCB). A star topology was used to connect the four computational FPGAs in a 64-bit ring and a control FPGA as the center of the star network. All connections between FPGAs and on-board memories ran at 200MHz. Computationally intensive tasks ran on the outer ring while the control FPGA ran a Linux OS and managed off-board I/Os. The asymmetry between the control FPGA and computation FPGA complicated the programming model.

In 2006, COPACOBANA, a low cost cryptanalysis system using large numbers of FPGAs was described [6]. In the system, 6 Xilinx Spartan-3-1000 FPGAs were grouped in a DIMM module. All modules were connected by a shared 64-bit data bus on a DIMM backplane. In a 2007 implementation [7] the system running at 136MHz can search a full 56-bit DES key space in 12.8 days. New versions of the hardware described in 2008 used more powerful Xilinx Virtex-4 FPGAs. This system is scalable in physical form but not logically. Users can add more DIMM modules as needed to expand the system but are limited by the global shared bus. Unlike cryptanalysis, most applications require communication between PEs, where the shared bus architecture becomes a bottleneck.

In 2007, Nvidia released their C compiler suite, CUDA, for Graphic Processing Units (GPU) [8]. Users can use the standard C language to utilize the massively-parallel threading feature in GPU for general purposes computation. In a GPU chip, simple PEs executing linear threads communicate to each other through shared memory. GPUs are increasingly attractive to both academia and industry due to ease of programming and high-performance floating point units. The scalability of GPUs is largely limited by their dependency on a host computer system; data communication overhead between the host and GPU through a PCIe interface makes it difficult to integrate large numbers of GPU chips with low latency over a dedicated high speed network.

3 System Architecture

In this section, the details of the Cube architecture are presented. Fig. 1 shows the block diagram of a complete Cube platform. Each FPGA is considered as an individual PE. All PEs are connected in a systolic chain with identical interfaces between them. There are no storage components in the system except for the PEs' internal memories. Also, there are no global wires for data communication and clock distribution. All FPGAs can be configured with the same design concurrently. Each PE accepts data from the previous one, processes them and passes them to the next PE. There are several advantages to this approach.

- **Scalability:** A centralized shared bus/memory architecture is not suitable for scaling up to massive amount of elements due to resource conflicts. The cost of full point-to-point topologies such as cross-bars increases exponentially with the number of PEs and thus become prohibitively expensive in systems with hundreds of PEs. In the Cube platform, a linear systolic interface is used which has cost which is linear with the number of PEs.
- **High Throughput:** Synchronizing high frequency clocks between 64 FPGA devices on a single board or across multiple boards is difficult. In the Cube system, short tracks between neighboring FPGA devices for clock and data distribution can easily achieve over 100MHz clock rates for inter-PE communication. Also, minimizing the overhead of handshaking and traffic switching results in low latency and deterministic communication channels.
- **Rapid Development:** Design partitioning and workload distribution in a large scale FPGA cluster are eased by a unified interface and by each PE playing a symmetric role in the system. All FPGAs can be programmed to the same configuration making for constant time configuration, rather than having time proportional to the number of PEs.
- **Low cost:** On-board tracks in the Cube are less expensive than the high speed switches and backplanes employed previous systems. Also, the regular layout in the Cube avoids the expensive and time consuming

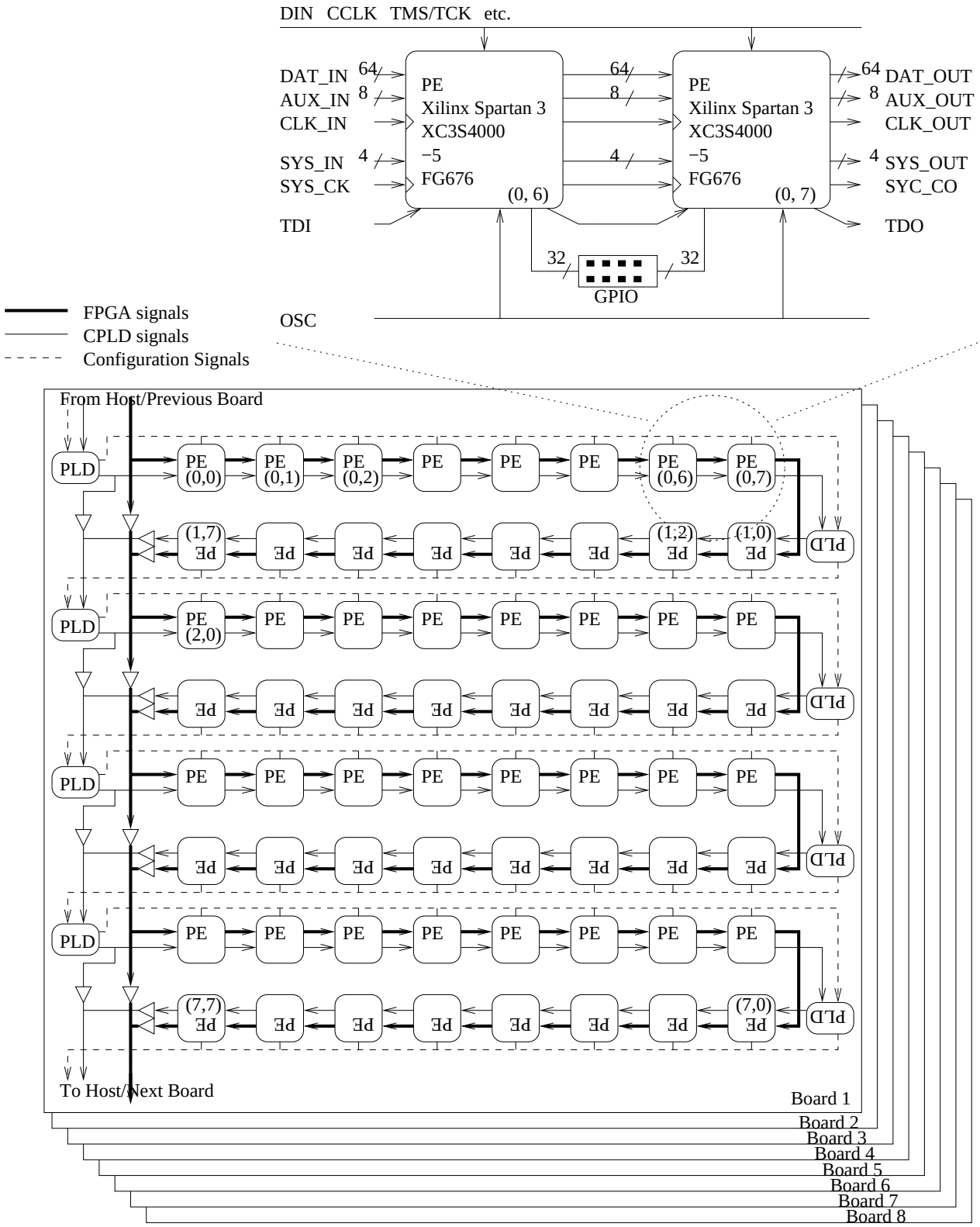

Fig. 1. Cube architecture.

processes of testing and verifying the signal integrity of the boards.

3.1 The FPGA Systolic Array

Each module in the Cube platform hosts 64 Xilinx FPGAs (XC3S4000-5-FG676) arranged in an 8 by 8 matrix as shown in Fig. 1. Each FPGA has a unique ID indicating the logical X-Y location of the device. Eight FPGAs are grouped together in a row and have independent configuration inputs and power supplies. The complete system consists of 8 connected boards in a cabinet forming an $8 \times 8 \times 8$ cluster of 512 FPGAs, and thus named Cube.

There are two systolic buses in the system: the `PE` bus is the major data communication channel between PEs and the `SYS` bus is used for system control. Each PE has a 64-bit data bus I/O (`DAT`), an 8-bit auxiliary bus I/O (`AUX`) and a dedicated clock line I/O (`CLK`), connecting the previous PE to the next PE. The `SYS` bus, with 1 dedicated clock line and 4-bit data I/Os, goes through all PEs and CPLDs. All these buses connect adjacent PEs only. These short point-to-point parallel buses significantly simplify the programming model and higher inter-PE bandwidth is achieved. The requirements on PCB layout and FPGA I/O interface are also relaxed for this topology compared to gigahertz serial communications in other designs. On the other hand, the systolic chain enables multiple boards to be cascaded for better scalability without reducing the I/O clock rate. The PE bus was designed to work at 100MHz and thus providing 6.4Gbps data bandwidth between PEs with additional control and handshaking signals on the `AUX` bus.

All these buses are freely available for user designs. The buses are also routed from/to external headers for communication between host and board or between multiple boards. In most applications, `CLK_IN` is driven by previous PE or external source from headers. The clock is then replicated for use internally and forwarded to the next PE through a delay locked loop digital clock manager (DCM) in the FPGA. In the design, the `DAT` and the `AUX` buses can easily match the wire length of the clock line for improved I/O thoughput.

The internal logic of the PE can only be used after the input clock source is stable. There is a delay between DCM reset and when the clock output is usable. The long distribution lines of the global reset and the long cascaded chain of 64 DCMs make it impossible to synchronize the DCM locking sequence of a 64-FPGA module concurrently. To solve this problem, the `LOCKED` output of the current DCM is used to reset the following DCM. This proceeds in a sequential fashion as described in [9]. Global synchronization of clock signals is feedforward in nature and skew is dependent on the performance of the DLLs. An additional 25MHz oscillator is provided in each row for increased flexibility. This clock source is broadcasted to the row and shared by both FPGAs and the row associated CPLD.

3.2 Configuration of FPGAs

In the Cube platform, different FPGA configuration schemes are provided under SCMD for minimum configuration time and maximum flexibility. Considering the number of FPGAs and the size of the board in our design, commodity programming equipment cannot provide sufficient driving power to configure all devices concurrently. Thus a CPLD (Xilinx CoolRunner-II XC2C256-VQ100) is installed in each row to control and drive the configuration signals. Both JTAG and Slave Serial (SS) programming modes are supported by selecting the `M1` input to the FPGA through the CPLD. This can be controlled by on board DIP switches or external host through the `SYS` bus. The CPLDs are programmed by a separated JTAG chain.

Slave serial (SS) mode provides the fastest way to configure all FPGAs in parallel. The SS configuration signals are sampled and buffered by internal Schmitt triggers in the CPLDs and thus all associated FPGAs receive clean and synchronized signals. As shown in Fig. 2, there are three output links from each CPLD for SS configuration. Two of these links broadcast the signals to FPGAs in the odd and even position of the row, while the third link sends the SS signal to the CPLD of the next row. This provides extra flexibility for enabling user to program different configurations in odd and even FPGAs. It is also possible to program different configurations to different rows of FPGAs.

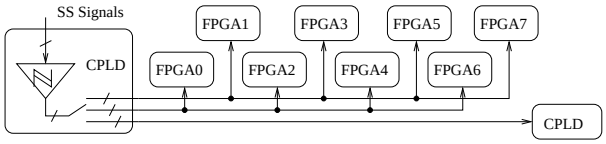

Fig. 2. Slave Serial Configuration Mode in Cube.

JTAG mode allows users to program individual FPGAs and read back internal values. Using JTAG in conjunction with SS mode enables users to configure most FPGAs in parallel rapidly and change the contexts of some FPGAs later. For example, it may be necessary to change the head and tail of the systolic chain in certain applications.

3.3 External Interface

The first and the last FPGAs in the systolic chain are connected to external headers on the 64-FPGA board. Both the `PE` and the `SYS` buses are available. For both input or output, there are 78 signal lines grouped into three standard IDE headers which can be connected to external devices or another module in the Cube through standard IDE ribbon cables.

There are also three pairs of programming headers for CPLD JTAG, FPGA JTAG and FPGA Slave Serial configuration. By bridging the output headers of the current module to the input headers of the next module, a single set of programming cables can be used to configure 512 FPGAs

concurrently.

Each pair of FPGAs has an extra 32-bit (EXT) bus. Using this channel, pairs of FPGAs can be grouped to form a tightly coupled PE for larger designs which cannot fit in a single FPGA. This bus can also be used as general purpose IOs (GPIOs) shared by the pair for debugging or external connections as all 32 signals are connected to high density headers.

To provide a simple way to monitor the internal status of FPGAs without extra equipment, each FPGA drives four LEDs of different colors (red, green, yellow and orange). In addition, each CPLD is connected to an 8-bit DIP switch and eight LEDs for controlling and debugging in the field.

3.4 Fault Tolerance

Installed electronic devices can be damaged by high Electrostatic Discharge (ESD), high temperature and/or physical stress. As all PEs are connected in a systolic chain, any malfunctioning FPGA in the chain will break the data path and render the system unusable for most applications. Replacing the soldered FPGA device is expensive due to the high pin count Ball Grid Array (BGA) packages employed and close component placement. The cost of large BGA sockets is also too high and occupies too much real estate for production.

To address this issue, a row-skipping scheme is implemented which allows pairs of rows to be bypassed. By setting a pair of tri-state switches as the small triangles shown in Fig. 1, users can configure the outputs of an odd row to be either the outputs from the last PE in an odd row or the inputs to the first PE in a grouped even row. When inputs are fed through directly to the outputs, the pair of rows are skipped in the design without affecting the rest of the board. Besides the PE and SYS buses, all programming signals can also skipped through these switches. Users can now work around damaged FPGAs in arbitrary position without trashing the whole board with 64 FPGAs. It also enables users to reduce power consumption when not all FPGAs are required for the application.

4 Cube Prototype

The major component of the Cube platform is the 64-FPGA module. In the prototype Cube system, two of these modules were built for experiments and evaluation. Fig. 3 shows a photograph of a fully populated 64-FPGA module which was programmed for system connectivity testing.

4.1 The 64-FPGA Module

The PCB hosting 64 Xilinx Spartan-3 FPGAs is $320mm \times 480mm$ in size. An 8-layer FR-4 PCB, four layers were used for power/ground plans and another four for signal routing. Following the standard BGA escape pattern in [10], $5mil$ and $6mil$ tracks are used for signal routing.

On the right hand side of the module, two 20-pin ATX power sockets are installed as the main power inputs. For each row, a PTH05030W DC/DC switching power module converts the +5.0V input to the 1.2V VCCINT supply. For each ATX power socket, two LDO voltage converters are used to provide the 2.5V VCCAUX and the 1.8V VCCCPLD from the +3.3V input. The +3.3V input source is used directly as the VCCIO33 supply. Negative and +12V inputs from the ATX Power Supply Unit (PSU) are not used. This scheme can provide sufficient power for 90% of the LUTs/FFs, 100% of the BRAMs and 100% of the DSPs of all FPGAs to switch at 200MHz, and all I/Os to switch at 100MHz. The switching probability was set to 33% except that of LUT/FFs was set to 12%.

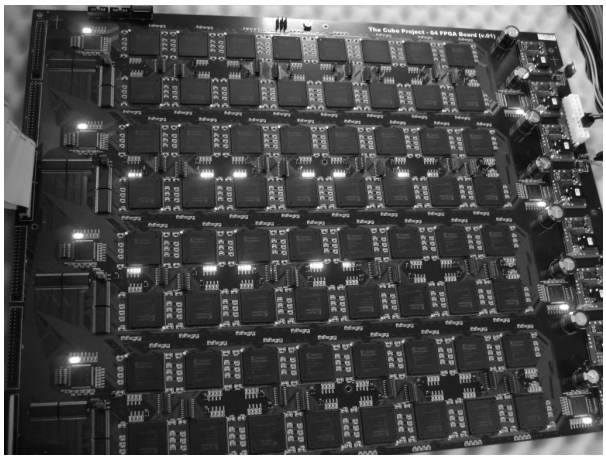

Fig. 3. A 64-FPGA Module in Action.

The 64-FPGA modules are kept in a 12U 19 $inch^3$ rack cabinet. Although only passive cooling heat sinks are installed in the modules, fans are mounted on the cabinet for system level active cooling. All FPGAs on all the modules are tested to be operational and can be configured in both JTAG and SS mode successfully. In SS mode, all 64 FPGAs in a module can be configured in less than four seconds. We also tested that the PE bus can run at 100MHz with an external clock source. In the initial setup, two Xilinx V2P XUP boards [11] were used for I/O interfacing. The XUP boards provide various interfaces for external devices including SATA, Ethernet, Flash and MGT. In practice, any device with 40-pin IDE header in LVCMOS 3.3V I/O standard can be interfaced to the Cube system.

4.2 Development Environment

Programming using both VHDL and Verilog is supported by the Cube platform via a set of standard templates. The templates include pin assignment in a UCF file, top level entity (module) interface, clock/reset systems and user logic interfaces. Our designs are synthesized and implemented using the Xilinx ISE 10.1i tool chain. Configurations are downloaded to the FPGAs through Xilinx iMPACT tools. The RTL package also includes a parameterized simulation template for simulating multiple PEs as in the real hardware.

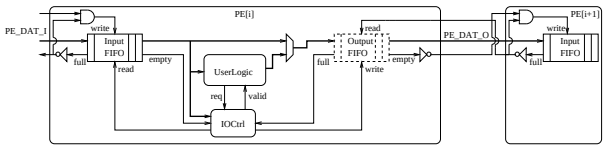

Fig. 4. Default Interface Template.

In the provided template, the `PE` bus is interfaced to internal FIFOs as shown in Fig. 4. Users can change the contents of `UserLogic` and `IOCtrl` blocks to realize specific applications while leaving the interface intact. The (`full`, `empty`) and (`read`, `write`) pairs between PEs automatically transmit data at maximum bandwidth. Internally, the `IOCtrl` block, provided by user, controls the data flow in top level. If `UserLogic` requests data from previous PE, the `IOCtrl` will assert the valid flag when a data addressing current PE appears at the output port of the input FIFO. If `UserLogic` has data to send out, it will direct the data through the output MUX. If there is no I/O request from `UserLogic`, it will forward data from the input FIFO to the output FIFO. This communication system facilitates packet streaming and insertion between PEs where user logics need to handle simple FIFO interface only. In addition, using asynchronous FIFOs can provide a clean separation of clock domains so that user logic can work at a clock rate higher than the I/O interface.

5 Application: Key Search Engine

The RC4 encryption algorithm was developed by RSA Labs in 1987. The algorithm is dated and proved to be insecure due to lack of nonce information and correlation between output stream and key. But the long history and ease of implementation make it a widely used encryption algorithm in both software and hardware systems. Applications include WEP in 802.11, Secure Sockets Layer (SSL), Kerberos authentication and Microsoft Office.

We constructed a RC4 key search engine based on the design in [12] to demonstrate the idea of MPP on the Cube platform. The encryption algorithm is presented below where $S[]$ is a state array of a Random Number Generator (RNG). The RNG output stream is then used to scramble messages using the XOR operation.

/* initialization phase */
for i = 0 to 255 **do**
  $S[i] \leftarrow i$
end for
/* setup phase */
$j \leftarrow 0$
for i = 0 to 255 **do**
  $j \leftarrow (j + S[i] + key[i \ mod \ key\_length]) \ mod \ 256$
  swap $S[i], S[j]$
end for
/* RNG phase */
$j \leftarrow 0$
while has input **do**

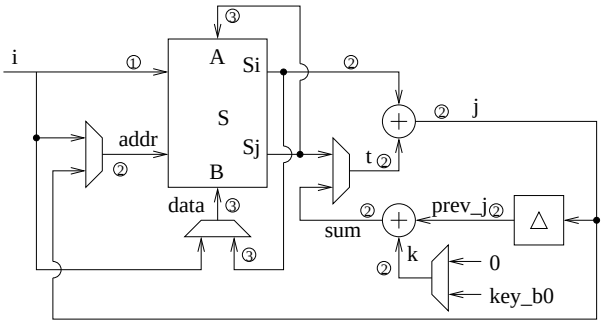

Fig. 5. RC4 Key Search Unit.

  $i \leftarrow (i + 1) \ mod \ 256$
  $j \leftarrow (j + S[i]) \ mod \ 256$
  swap $S[i], S[j]$
  $RNG_{out} \leftarrow S[(S[i] + S[j]) \ mod \ 256]$
end while

In our implementation, a dual port BlockRAM (BRAM) is used to store two copies of the state array. The data path of a single key search core is shown in Fig.5. Due to data dependencies, the tasks in the setup loop and the RNG loop must be performed sequentially in hardware. Thus at least three clock cycles are required for a single iteration in these loops. The numbers in circles along the signal lines in Fig. 5 indicate the activities of the circuit in different clock cycles. In the 1^{st} cycle, $S[i]$ is read and j computed. key_b0 is the lowest byte from the 40-bit shifting key array. In the 2^{nd} cycle, $S[j]$ is read. The swap of $S[i]$ and $S[j]$ takes place in the 3^{rd} cycle. In 1^{st} cycle of the setup phase, i is used to initialize the other half of the BRAM through port B as the initialization phase for next key. In the RNG phase, $S[(S[i] + S[j]) \ mod \ 256]$ is read at the 1^{st} cycle of next iteration through port B. Then the resulting RNG byte is compared against a pre-stored reference. If all RNG bytes match the reference bytes, the current key under test is the encryption key and the checking process stops. Else, the MSB of BRAM address is toggled in both ports and the checking precess restarts with a new key using the other half of BRAM as state array which is initialized in the previous checking process.

Since there are 96 BRAMs in a XC3S4000 device, we implemented 96 key search cores in a single FPGA. The core is self- contained and includes the data path of Fig. 5, the associated control logic and a 40-bit local key. All cores within the same PE work synchronously. The MSBs of the local key are fixed to the ID of the PE while the LSBs are initialized from 0 to 95 according to the position of the cores. After a key is tested, the local key is increased by 96 for next test until the encryption key is found.

Each PE has a global control unit with the same FSM as that in the key search core except that no RNG stream is generated and checked. This global control unit collects the `found` signal from all the 96 cores on chip. Based on the ID

Table 1. Performance Comparisons.

	64-FPGA	Cube	Xeon	Cluster
Keys/Second	753M	6024M	2.1M	753M
Search time (s)	1460	183	262K	1460
No. of PE	64	512	4	1463
Frequency (Hz)	100M	100M	2.5G	2.5G
Power (W)	104	832	400	71.8K
Size (U)	1	9	1	180

of the PE, the position of the core which found the key and the local key of the global control unit, the 40-bit encryption key can be reconstructed and reported to the external host through the 64-bit `DAT` bus. In this implementation, a single key search core occupied 169 LUTs and 91 FFs. The complete design, operating at 100MHz, consumes 14202 LUTs (25%), 8968 FFs (16%) and 96 BRAMs (100%) in the Xilinx FPGA device. The critical path is from the local key shifting register output, key_b0, through three multiplexers and two adders to the $addr$ input of the BlockRAM.

It takes $(256 + n) \times 3$ clock cycles to check a key where n is the length of referencing text. For $n = 16$, a core requires $8.16\mu s$ to check a single key. There are 96 cores on a single FPGA and thus $96 \times 64 = 6144$ cores on board. One 64-FPGA module can search full 2^{40} key space in 1460 seconds. The total power of the key search engine running on a single 64-FPGA module was measured to be 104W. On average, an optimized multi-threaded C version of the RC4 key search engine compiled using GCC v4.1 with `-O3` option can check a key in $0.477\mu s$ on a 2.5GHz Intel Quad-Core Xeon processor. Thus 145 hours for the full key space. The experimental results show that a single module has similar computing power as a cluster of 359 high-end CPU with significantly reduced space and power consumption. With 512 FPGAs in the Cube, this task can be completed in around 3 minutes. The performance measurements and comparisons are shown in Table 1. Here we assumed the cluster has 180 1U server boards each hosting two Quad-Core Xeon CPU. The search time of searching a full 2^{40} key was used in the table.

6 Conclusion

We have presented the design, implementation and performance evaluation of the Cube platform as a massively-parallel reconfigurable cluster with 512 FPGA devices. The RC4 key search engine demonstrated the high computation density of the Cube over standard technology such as PC clusters. Compared to previous efforts in reconfigurable MPP architectures, the Cube platform has advantages in scalability, flexibility, fault tolerance, accessibility and cost-performance ratio. The fixed systolic connection of the Cube makes it less flexible than existing FPGA clusters and limited the class of suitable application. This is the trade-off for easy interfacing and low system cost. Currently, multiple applications are being developed on the Cube platform including particle collision simulation, constrain solving and cellular automata. In the future, we will extend our studies on the behaviors of both scheduled, deterministic communication and random event driven communication on this architecture. We will also study automated design partitioning tools and work load distribution for the Cube.

7 References

[1] J. Makino, M. Taiji, T. Ebisuzaki, and D. Sugimoto, "GRAPE 4: a one-Tflops special-purpose computer for astrophysical N-body problem," *Supercomputing Proceedings*, pp. 429–438, Nov. 1994.

[2] M. Junichiro, T. Makoto, E. Toshikazu, and S. Daiichiro, "Grape-4: A massively parallel special-purpose computer for collisional n-body simulations." *Astrophysical Journal*, vol. 4, pp. 432–446, 1997.

[3] J. Makino, T. Fukushige, M. Koga, and K. Namura, "GRAPE-6: Massively-parallel special-purpose computer for astrophysical particle simulations," *Astronomical Society of Japan*, vol. 55, pp. 1163–1187, Dec. 2003.

[4] J. Makino, K. Hiraki, and M. Inaba, "GRAPE-DR: 2-pflops massively-parallel computer with 512-core, 512-gflops processor chips for scientific computing," in *Proceedings of the 2007 ACM/IEEE conference on Supercomputing SC'07*. New York, NY, USA: ACM, 2007, pp. 1–11.

[5] C. Chang, J. Wawrzynek, and R. Brodersen, "BEE2: a high-end reconfigurable computing system," *Design and Test of Computers, IEEE*, vol. 22, no. 2, pp. 114–125, March-April 2005.

[6] E. Kumar, C. Paar, J. Pelzl, G. Pfeiffer, and M. Schimmler, "Breaking ciphers with COPACOBANA - a cost-optimized parallel code breaker," in *Workshop on Cryptographic Hardware and Embedded Systems*, Oct. 2006, pp. 101–118.

[7] T. Guneysu, T. Kasper, M. Novotny, C. Paar, and A. Rupp, "Cryptanalysis with COPACOBANA," *IEEE Trans. Comput.*, vol. 57, no. 11, pp. 1498–1513, 2008.

[8] I. Buck, "GPU computing: Programming a massively parallel processor," in *CGO '07: Proceedings of the International Symposium on Code Generation and Optimization*. Washington, DC, USA: IEEE Computer Society, 2007, p. 17.

[9] *Using the Virtex Delay-Locked Loop*, Xilinx, Inc., 2006, version 2.8.

[10] *Four- and Six-Layer, High-Speed PCB Design for the Spartan-3E FT256 BGA Package*, Xilinx, Inc., 2006, version 1.0.

[11] *Xilinx University Program Virtex-II Pro Development System*, Xilinx, Inc., 2008, version 1.1.

[12] K. H. Tsoi, K. H. Lee, and P. H. W. Leong, "A massively parallel RC4 key search engine," in *Proc. IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*. Washington, DC, USA: IEEE Computer Society, 2002, pp. 13–21.