Enhancing FPGA Based Encryption on the Cray XD-1*

Joseph Fernando, Dennis Dalessandro, Ananth Devulapalli, Ashok Krishnamurthy Ohio Supercomputer Center - Springfield {fernando, dennis, ananth, ashok}@osc.edu

Abstract

The Cray XD1 is a high performance distributed Linux cluster that, as one of its advanced features, includes integrated FPGAs. This platform gives us the opportunity to integrate into user level applications, the ability to interface with an FPGA. We have utilized the FGPAs on the Cray XD1 in order to carry out AES Encryption. Previous attempts have yielded sub-optimal performance results. In this work we build upon past experiences by implementing a new mechanism to move data in and out of the FPGA, as well a streamlined AES Encryption algorithm. In making use of a new mechanism to move data between host processor and FPGA, not only will our AES Encryption benefit, but all FPGA cores built for the Cray XD1 could take advantage of this method and see vast increases in performance.

Introduction

AES Encryption [1], is a task which is computationally expensive, and often times not the main focus of an application. Therefore, it is desirable to offload these computations from the CPU, thus enabling the CPU to handle other tasks. In our previous work [2] we have successfully interfaced with the FPGA on the Cray XD1 in order to carry out AES Encryption. Due to constraints on the mechanism used to exchange data with the FPGA, the performance was sub-optimal. We now aim to improve upon this by utilizing a high performance I/O method. The preliminary results of which, we show in the sections to follow.

The Cray XD1 [3] is a distributed Linux cluster which incorporates many novel features. The Cray XD1 at the Ohio Supercomputer Center (OSC) is comprised of 3 chassis, for a total of 36 processors and 6 FPGAs. Each chassis contains 6 symmetric multi-processors (SMP), each SMP is comprised of 2 Opteron processors, thus 12 processors per chassis. The 6 SMPs in each chassis are connected with each other via a high performance interconnect based on InfiniBand, known as the Rapid Array [4]. Each chassis is connected to the other chassis via the same high performance interconnect. Currently one of the chassis is equipped with 6 Xilinx Virtex II Pro FPGAs [5]. Each of the 6 FPGAs is connected to 1 SMP via the hyper transport of one of the Opterons. The high performance Rapid Array makes it efficient for multiple processors to share data, and allows us to use multiple FPGAs in parallel.

In our previous work [2], we have interfaced with the FPGA by writing 16 bytes of data to an FPGA register. The host application (running on the Opteron) was then able to read the encrypted result immediately from another FPGA register. The overhead of actually calling the func-tion to read/write takes more time than for the FPGA to encrypt the 16 bytes of data. We experienced a great slowdown from this reading and writing 16 bytes of data at a time. In fact the overhead was so great, that 1 CPU was able outperform 1 FPGA. We have found that we can vastly improve our results and actually out perform a CPU by utilizing the FPGA Transfer Region (FTR) of memory, described in the following section. Taking advantage of the multiple FPGAs, we have implemented a resource manager which distributes data to a number of FPGAs for encryption, and collects the final results. This parallel use of FP-GAs allowed us to achieve better performance than a CPU for large amounts of data. However, by utilizing the FTR mechanism we enable the FPGA to outperform the CPU in all cases.

*Support for this project was provided to the Ohio Supercomputer Center through the Department of Energy ASC program.

AES Core Design

The FTR memory, also known as ftrmem, is an area of main memory reserved by the FPGA device, which used for communication between the FPGA and the host processor (Opteron). Currently, only 2MB of the ftrmem is available per application. This is a limiting factor, but something that will probably change in the future as the XD1 is still considered a beta system, particularly where the FP-GAs are concerned. This 2MB of ftrmem is partitioned into 1MB chunks for loading the raw data by the host (CPU) and storing the processed data by the FPGA.

We have implemented the Electronic Code Book (ECB)[1] method of the AES algorithm. Our implementation was based on an open-source AES encryption core [6]. As it turns out, encryption in general, and AES in particular, are very FPGA friendly. In the ECB method, the AES algorithm operates on 128-bits at a time, which involves 10 iterations resulting in a 10-cycle latency.

In our previous work [2], we utilized a push-model, in which the host processor takes the initiative to push the data to the FPGA's registers, and to then collect the results from a destination register on the FPGA . Due to high overhead of the function calls to read and write the FPGA registers the total throughput of FPGA based encryption was less than that of a host-only (no FPGA) implementation. This has forced us to look at alternative ways of communication between the FPGA and the host processor.

We have designed a general-purpose and efficient I/O subsystem (IOSS) module, which is generic enough to be used in any FPGA core implemented on the XD1. The use of the IOSS will be in pull-model based applications, where the FPGA needs to pull data from the ftrmem and process it. The function of IOSS is to enable FPGA to efficiently read and write the ftrmem. The IOSS makes use of the burst communication mode of the hyper transport. The main challenges involved in design are, handling out-of-order hyper transport packets, implementing efficient read and write modules, and synchronization between the read and write modules. Resource-wise, the IOSS is very efficient and occupies less than 10% of the available FPGA slices. More details on design issues of the IOSS are available in [2].

Preliminary Results

Frequency	199 MHz
Peak throughput	796 MB/sec
Sustainable throughput	459 MB/sec

Table 1. Performance metrics of synthesized I/O subsystem.

Table 1, shows the performance metrics of the IOSS. We are able to synthesize the IOSS at 199 MHz, which is close to the peak (200MHz) clock rate available on the Cray XD1. The peak throughput, shown in Table 1, was measured by timing the transfer of 1MB of data. A single transfer involves reading data from the ftrmem source, and writing the processed data to the ftrmem destination. The sustainable throughput, also shown in Table 1, has been measured by timing the transfer of datasets larger than 1MB. Sustainable throughput involves two additional copies, one from the original data source to the ftrmem source and another from the ftrmem destination to the data's final destination. The overhead of the memory copies reduces performance by almost 40%. Future versions of the XD1 system software should alleviate some of this overhead by allowing larger regions of ftrmem to be allocated and used. When there are no copies, the peak throughput achieved is 796 MB/sec which is almost 50% of the one-way theoretical peak bandwidth 1600 MB/sec (200MHz * 64-bit bus). Clearly this is the best case scenario. There are two underlying reasons why sustainable throughput is lower. Since reads and writes are happening concurrently, the host-based memory controller will arbitrate the requests, and as a result the requests will never happen simultaneously in a cycle. The other reason, is that the FPGA read and write requests as well as FGPA output data share the same bus, thus reducing available bandwidth.

	Projected Freq.	Projected Peak Throughput	CPU Throughput [7]
Encryption	160 MHz	640 MB/sec	13.34 MB/sec
Decryption	150 MHZ	600 MB/sec	11.73 MB/sec

Table 2 Projected performance of AES encryption engine

Tabe2, shows the projected throughput of our pipelined AES encryption and decryption cores which will utilize our IOSS for communication. The projected frequencies are actually the frequencies of corresponding non-pipelined cores. Note, that since the IOSS is running at 199 MHz, the bottleneck is in the cores. The projected peak throughput is well below that of AES-ASICs[7], but in an HPC environment, like the Cray XD1, the reconfigurability of FPGAs far outweighs the loss in performance. As mentioned previously, our design is limited by the lack of features in the XD1 system software. Future revisions should translate into better performing applications.

Conclusion

We have shown that utilizing a more effective mechanism to move data between host processor and FPGA can have a great impact on the FPGAs performance for a given task. In particular with our implementation of AES Encryption, we are able to achieve much higher performance by utilizing the FPGA Transfer Region of memory. The I/O subsystem that we have developed serves as a means for any FPGA core to utilize the FPGA Transfer Region of memory. Future improvements to the Cray XD1 and its system software will further improve performance and we look forward to taking advantage of these improvements as they become available.

References

[1] J. Daemen, V. Rijmen, AES Proposal: Rijndael, 1999.

[2] J. Fernando, D. Dalessandro, A. Devulapalli, K. Wohlever, Accelerated FPGA Based Encryption, In Proceedings of *The Cray User Group 2005,* 2005.

[3] Cray Inc., The Cray XD-1 Supercomputer, 2005, http://www.cray.com/products/xd1/index.html.

[4] Cray Inc, Cray XD1 Data Sheet, 2005, http://www.cray.com/downloads/Cray_XD1_Datasheet.pdf.

[5] Xilinx Inc, Virtex-II Pro and Virtex-II Pro X Platform FPGA: Complete Data Sheet, 2005, http://direct.xilinx.com/bvdocs/publications/ds083.pdf.

[6] Opencores.org, Open Cores, 2005, http://www.opencores.org.

[7] VIA Technologies Inc., VIA PadLock Hardware Security Suite, 2005, http://www.via.com.tw/en/initiatives/padlock/hardware.jsp.