Accelerated FPGA Based AES Encryption

Joseph A. Fernando, Dennis Dalessandro, Ananth Devulapalli, Kevin Wohlever

{fernando, dennis, ananth, kevin}@osc.edu

Ohio Supercomputer Center (OSC)

Springfield, Ohio 45502

Agenda

- About OSC
- Introduction
- Motivation and Rationale
- Objectives /Scope
- Cray XD1 at OSC
- FPGA Resource Manager
  - Distributed AES encryption
  - Register based interface for encryption
  - Performance metrics
- Other possible I/O methods
- FPGA Transfer Region (FTR) memory based I/O
- Preliminary Results
- Conclusions
  - Future work
- Questions and Answers

About OSC

- OSC is a State of Ohio entity
  - Administered by Ohio State University
- Main purpose is to provide Supercomputing facilities to state and private academic/research institution of Ohio
- Maintains a statewide high speed network
- Presently over 80 colleges and campuses are interconnected through a 10 Gbit/s network
- Schools in the future
- Conducts research related High Performance Computing in various fields
- Main office located in Columbus Ohio with another office in Springfield Ohio
  - Totals about 100 employees

Introduction

- Nodes that do not have FPGAs cannot get services at present
  - Resource Manager (RM)
    - To provide FPGA services to other nodes/remote nodes
    - to schedule
- AES encryption algorithm is a FPGA friendly algorithm
  - Mainly bit manipulation using Finite Field Arithmetic
    - Addition and subtraction done by exclusive OR of the two inputs
  - Electronic Code Book mode of the AES algorithm can be easily parallelized
    - 128 bit blocks required for 128 bit key
    - 10 rounds for 128 bit key

Motivation and Rationale

- Leverage the capabilities of FPGAs to speedup
  - Exploit Parallelism
  - Replicate processing elements
- AES and other bit manipulation algorithms cannot be efficiently mapped to general purpose CPUs
  - Primitive logic operations cannot be parallelized on CPUs
  - Cannot replicate processing units
  - atomic data type is byte in CPUs
- Encryption could be computed with low latency and high throughput by using FPGAs
  - Map every round to a different clock cycle or different processing unit

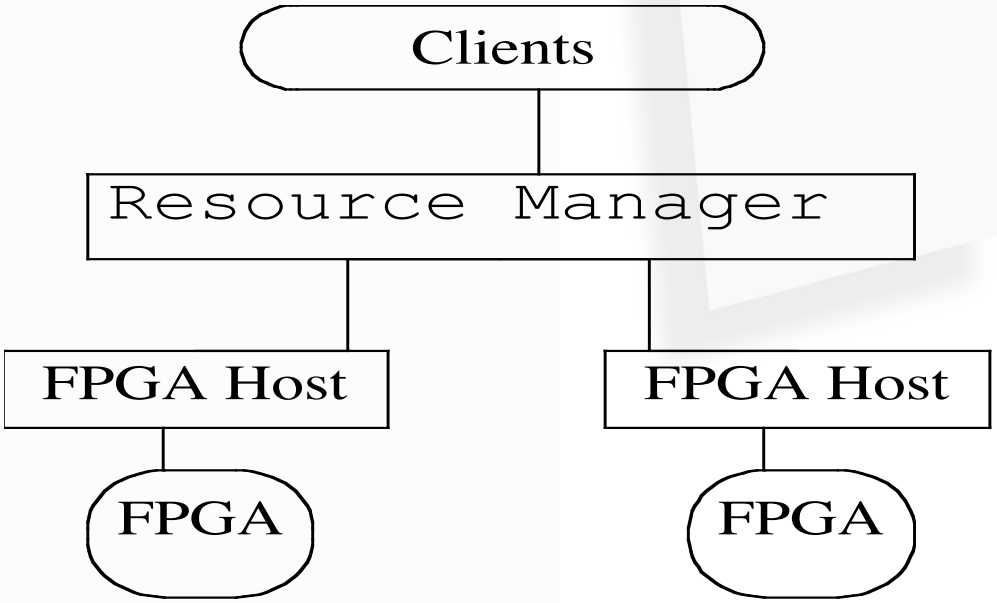
Objective and Scope

- Develop a RM to provide services to nodes that do not have FPGAs
  - Use RM for scheduling distributed resources
  - XD1 nodes as other remote nodes
- RM Specs
  - Use multiple FPGA on the XD1
  - Encryption and decryption units
    - Use the ECB mode of the AES algorithm for higher throughput
    - 128 bit key only

Cray XD1 at OSC

- 3 chassis with 18 Symmetric Multi Processors (SMP) working at 2.2 GHz
- Each SMP has 2 Opterons
- Each SMP connected to a Rapid Array Processor for communication and a FPGA card
- Only 6 FPGAs in a single chassis
- Use Xilinx 2VP50 FPGA devices

Resource Manager



- RM resides on a node that does not have a FPGA
- Use Message Passing Interface (MPI)
- XD1 nodes that do not have FPGAs and remote nodes can request FPGA services
- Schedules all the FPGAs
- Distributes encryption function to up to six FPGAs

Resource Manager (Cont.)

- Distributed AES Encryption
  - Distribute input data at to all the FPGAs
  - Use 16 byte aligned buffers to distribute data
    - AES uses 128-bit blocks
  - Collect the outputs and merge them in the same order
  - Encryption and decryption done the same way
  - Key is kept same to make this process easy

Resource Manager (Cont.)

- Register based AES encryption
  - Load keys, and use another registers to indicate the keys
  - Use registers to input 128-bit (2x64 bit) block of data
  - Output is received in another 128-bit (2x64 bit) register
  - Use 10 clock cycles to encrypt 128 bit buffer
  - Host writes to input buffer and reads from output buffer
  - Do not have to poll as the latency of the function call is adequate for synchronization
  - Continue writing data to buffer until the end of file

Resource Manager (Cont.)

- Performance metrics
  - Encryption using OpenSSL code on Opteron can achieve a throughput of 4 MB/s
    - Use a RAM based method (Different than method used on FPGA)
  - On a single FPGA using register based Interface 3 MB/s
  - Near linear increase of throughput with number of FPGAs
  - Encryption: clock speed 160 MHz 3010 (12 %) slices
  - Decryption clock speed 150 MHz 3508 (14%) slices
  - Bottleneck is in the I/O system

Alternative I/O capabilities

- QDR RAM based I/O
  - Uses the push model
    - Host processor dumps data in QDR memory
  - Synchronization done by stopping data transfers
  - Faster than register based I/O
    - No function overhead
    - Burst mode usage, depends on the implementation
  - Effective bandwidth depends on the processor load
- BRAM based I/O
  - Same as above other than using FPGA based Block RAM

FPGA Transfer Region (FTR) memory based I/O

- OSC has implemented a preliminary version of a I/O interface using the FTR memory
- Objective is to use this for general purpose I/O (e.g. AES encryption)
- Use the Pull model
  - FPGA read and writes data from/to host Opteron memory
- Use burst mode to read and write
- Concurrent read and writes
- Performance nearly host processor load independent
  - Minimal degradation of effective bandwidth under load

FTR memory (Cont.)

- Implemented a ftrmem_copy program
  - Copies a given buffer of a length to a destination (same host)
- Uses semaphores to synchronize read and write
- Use a dual port BRAM as a scratchpad
- Preliminary results are encouraging
  - Achieved a 800 MB/sec transfer rate (each way) for copy program for no load processor
  - Achieved a 785 MB/sec transfer rate under fully loaded conditions (or 2% degradation under load)
  - Could achieve 1.6 GB/s if limited to one way communication
- Further tests required to test robustness

Conclusion

- When Algorithms require streaming of data using FTR memory based I/O could provide a higher I/O bandwidth and make the I/O transfer host processor load independent while ensuring data integrity.

Conclusion (Cont.)

- Future work
  - Test and make this I/O subsystem robust
  - Integrate with a pipelined AES encryption and decryption engines (partially done)
  - Transfer lossless compressed data (??)
  - Develop for other scenarios
  - Integrate other algorithms that require high-speed streaming data
  - Just stay tuned

Questions & Answers