CUDA COMPATIBLE GPU AS AN EFFICIENT HARDWARE ACCELERATOR FOR AES CRYPTOGRAPHY

Svetlin A. Manavski

Presented by:
Gareth Ferneyhough
CS 791V
UNR, Fall 2011
Outline

● Cryptography and AES Overview
● Previous GPU implementation of AES
  ○ OpenGL Pipeline
● CUDA Implementation
  ○ Advantages
  ○ Method
● Results
● Conclusion
AES - Advanced Encryption Standard

• AES is a block cipher algorithm
• Symmetric-key: encryption and decryption use same main key (cipher key).
• Federal Government encryption standard since 2002
• Block size: 128 bits
• Key size: 128, 192, or 256 bits
AES - Advanced Encryption Standard

- Encryption performed on block (state) size of 128 bits
  - 4x4 matrix of bytes
- Entire message is split into several of these blocks; each block encrypted separately
  - Final block is padded, if necessary
- The main key (128, 192, or 156 bits) is expanded into several sub-keys (round keys)
  - 4x4 matrix of bytes

| \( a_{0,0} \) | \( a_{0,1} \) | \( a_{0,2} \) | \( a_{0,3} \) |
| \( a_{1,0} \) | \( a_{1,1} \) | \( a_{1,2} \) | \( a_{1,3} \) |
| \( a_{2,0} \) | \( a_{2,1} \) | \( a_{2,2} \) | \( a_{2,3} \) |
| \( a_{3,0} \) | \( a_{3,1} \) | \( a_{3,2} \) | \( a_{3,3} \) |

= 128 bits
AES - Advanced Encryption Standard

Steps:

1. Key expansion - several sub-keys (called round keys) derived from main key
2. Initial Round
   1. Add round key
3. Rounds (9 total)
   1. Substitute bytes
   2. Shift rows
   3. Mix columns
   4. Add round key
4. Final round
   1. All 3 round steps except mix columns
AES - Advanced Encryption Standard

1. Byte Sub
2. Shift Row
3. Mix Columns
4. Add Round Key

Repeat $n$ Times
AES - Advanced Encryption Standard

- each byte in state is replaced with corresponding entry in a look-up table

- each row is shifted left $n$ times, where $n$ is the row's index

- each column is multiplied by a known matrix

- state is XORed with the $ith$ round key
AES - Advanced Encryption Standard

Encryption process

AddRoundKey

Initial round

Cipher Key

1-SubBytes
2-ShiftRows
3-MixColumns
4-AddRoundKey

Round key 7

Round key 10

SubBytes
ShiftRows
AddRoundKey

Round 7
9 rounds

Final round
AES - Advanced Encryption Standard

Optimization:

On 32 bit or larger platforms, substitute bytes, shift rows, and mix columns can be combined into a series of table look-ups, speeding up the execution of the cipher

- Requires four 256-entry, 32-bit tables
  - 4096 bytes of memory (1KB each)

- Each round can now be done with 16 table lookups, 12 32-bit XORs, and four 32-bit XORs for the add round key step
Previous GPU implementation of AES

- Hardware solutions exist for AES
  - ASIC, FPGAs
- Previous researchers were forced to use fixed OpenGL graphics pipeline
  - Three types of processors
    - Rasterizer
    - Vertex
    - Fragment
      - Capable of *gather*, but not *scatter*
      - Most frequently used
        - More numerous
        - Closer to end of pipeline
Previous GPU implementation of AES

Disadvantages of OpenGL implementation:

- Only one AES round per kernel call
  - CPU responsible for getting outputs and setting inputs and calling each round

- Lack of bitwise logical operations in programmable shaders
  - XOR was implemented with a 256x256 look-up table

- Result: Slow
Previous GPU implementation of AES

Disadvantages of OpenGL implementation:

- Only one AES round per kernel call
  - CPU responsible for getting outputs and setting inputs and calling each round

- Lack of bitwise logical operations in programmable shaders
  - XOR was implemented with a 256x256 look-up table

- Result: Slow
  - How slow?
    - 40 times slower than CPU!
Previous GPU implementation of AES

Disadvantages of OpenGL implementation:

- Only one AES round per kernel call
  - CPU responsible for getting outputs and setting inputs and calling each round

- Lack of bitwise logical operations in programmable shaders
  - XOR was implemented with a 256x256 look-up table

- Result: Slow
  - How slow?
    - 40 times slower than CPU!
    - :(
CUDA Implementation

● CUDA to the rescue!
  ○ Programmers no longer constrained by the fixed graphics pipeline
  ○ 32-bit native XOR
  ○ Allowed general access to memory
    ■ Scatter and gather
CUDA Implementation

- CUDA to the rescue!
  - Programmers no longer constrained by the fixed graphics pipeline
  - 32-bit native XOR
  - Allowed general access to memory
    - Scatter and gather

Rocket central competition

Gather Ye Rosebuds While Ye May (Waterhouse)
CUDA Implementation

- Take advantage of AES 32-bit optimization

\[ e_j = T_0[a_{0,j}] \oplus T_1[a_{1,j+1}] \oplus T_2[a_{2,j+2}] \oplus T_3[a_{3,j+3}] \oplus k_j \]  

[1]

- 4x4 round input matrix
- one column of output
- Look-up table
- XOR
- one column of stage key

- 4 look-ups and 4 XORs per column per round
- So, a single round takes four iterations of equation
CUDA Implementation

Steps:
● input data and expanded keys stored in GPU global memory

● pre-computed look-up tables stored in specific constant memory of GPU

● input data divided into chunks of 1024 bytes and encrypted and decrypted in parallel
  ○ one CUDA block of threads is responsible for one chunk of input
    ■ one block = 256 GPU threads
    ■ threads in same block share expanded key, input data
CUDA Implementation

Steps (cont.):

● each block contains two 1KB arrays
  ○ input and output for each AES round
  ○ arrays are swapped after each round, allowing for complete encryption of the input chunk without exiting kernel

● finally, the result is saved to GPU global memory and transferred back to CPU
  ○ once launched, entire processes requires no intervention from the CPU
Results

- GPU faster than CPU for every input-size (including transfer times)
- Peak throughput rate on GPU = 8.28 Gbit/s
  - with input size of 8MB
  - 19.60 times faster than CPU

Performance for AES 256 [1]
Results

Performance for AES 256 [1]
Conclusion

• CUDA allows for significant speedup of AES encryption/decryption

• Future work:
  ○ GPU implementation of other symmetric algorithms
  ○ hashing, public key algorithms

• Questions?
References


[5] Dr. Gunes' slides from CS 450