Parallel cube testing on GPUs

Sudarshan Rao

June 10, 2010
1. Introduction and Background
   - Background
   - Cube Testing
   - CUDA
   - Primitives

2. Framework

3. Experiments
   - Description of experiments
   - Results
   - Timing

4. Conclusions

5. Future Work
Outline

1. Introduction and Background
   - Background
   - Cube Testing
   - CUDA
   - Primitives

2. Framework

3. Experiments
   - Description of experiments
   - Results
   - Timing

4. Conclusions

5. Future Work
Cryptographic primitives

- Algorithms used to construct security systems
- Crypto primitives used everywhere
- Security is essential
- Hash functions, block ciphers, stream ciphers etc
Hash functions

- Convert variable length message to fixed length message digest
- Used in digital signatures, message authentication codes etc
- Necessary security properties - Preimage resistance, Collision resistance, Second preimage resistance
- Brute force attacks - birthday paradox
- e.g., MD5, SHA family etc
Block cipher

- Encrypt fixed blocks of data
- Used to encrypt certain fixed sized data blocks, construction of stream ciphers etc
- Components of a block cipher
  - Plaintext
  - Key
  - Ciphertext
- e.g., DES, AES, Twofish, etc
Cube attack

- Cube attack - Itai Dinur and Adi Shamir
- Successful against low degree based primitives
- Treats primitive under attack as a black box
- Attacks on Trivium reported
Terminology

- In GF(2)
  - $X + Y = X \text{ xor } Y$
  - $X \ast Y = X \text{ and } Y$
- $p(x_1, x_2, \ldots x_n)$: Polynomial
- $p(x_1, x_2, \ldots x_n) = t_I \cdot p_{S(I)} + q(x_1, x_2, \ldots x_n)$
- $I \subseteq \{1, 2, \ldots n\}$: Index set
- $p_{S(I)}$: Superpoly
- $q$: Remainder
- $t_I = x_i x_{i+1} \ldots x_j$ where $i, (i + 1) \ldots j \in I$
- $x_i, x_{i+1} \ldots x_j$ are known as the cube variables
Evaluation of a superpoly

- $p = x_1x_2(x_3 + x_4) + x_1x_3$
- $x_1, x_2$ are cube variables

Consider

\[
\sum_{x_1x_2=00} p
\]

\[
= 0 \cdot 0(x_3 + x_4) + 0 \cdot x_3 + 0 \cdot 1(x_3 + x_4) + 0 \cdot x_3 \\
+ 1 \cdot 0(x_3 + x_4) + 1 \cdot x_3 + 1 \cdot 1(x_3 + x_4) + 1 \cdot x_3
\]
Evaluation of the superpoly

- \[ p(x_1, x_2, \ldots x_n) = t_I \cdot p_S(I) + q(x_1, x_2, \ldots x_n) \]
- \( q \) misses at least one \( x_i, \ i \in I \)
- \( q \) is added even number of times
- \( p_S(I) \) is added only once
\[ \sum_{i} t_i \cdot p_S(i) + q(x_1, x_2, \ldots, x_n) = p_S(i) \]
Find the value of the superpoly

Choose a set of cube variables say $c_1, c_2, \ldots c_n$
Choose a set of superpoly variables say $s_1, s_2, \ldots s_m$
Choose a random assignment for $s_1, s_2, \ldots s_m$

for $c_1, c_2, \ldots c_n = 000 \ldots 00$ to $111 \ldots 11$ do
  $Q = Q \oplus p(c_1, c_2, \ldots c_n, s_1, s_2, \ldots s_m)$
end for
Cube Testing

- $Q$ should be a random polynomial
- Can perform a variety of tests on $Q$
- Cube testing
  - Test for balance of $Q$
  - Test for linear variables in $Q$
  - Test for neutral variables in $Q$
  - Test for low degree $Q$
CUDA

- NVIDIA’s SDK for programming their GPUs
- C for CUDA enables developers to write C like programs
- Functions called **kernels** get executed on the GPU
- Kernels get executed in parallel on the GPU
CUDA contd...

Figure: Cuda program execution[3]
CUDA concepts

- Thread hierarchy
  - Thread blocks, grids
- Memory hierarchy
  - Global memory, shared memory, registers
AES

- Block cipher standardized by NIST in 2000
- Block sizes of 128 bits, 192 bits or 256 bits
- Not based on popular Feistel network

Figure: AES Round function[1]

In our tests we use AES-128
Threefish

- Tweakable block cipher
- Component in Skein, a NIST SHA-3 contest candidate
- Block sizes of 256 bits, 512 bits and 1024 bits
- Many simpler rounds more effective than few complicated rounds
- We use Threefish—256 in our tests
Threefish Mix and Round functions

Figure: Threefish Mix and Round function[2]
Keccak

- Keccak - candidate hash algorithm in the SHA-3 contest
- Based on sponge construction
- Uses a permutation as part of construction
- Keccak-$f[1600]$ permutation is studied
Keccak permutation

- Keccak-f[1600] - 3-dimensional array
- \( R = \iota \circ \chi \circ \pi \circ \rho \circ \theta \)
- \( \chi \) is a non-linear mapping
- \( \theta, \pi, \rho \) - operations that permute the state
- \( \iota \) - Mixing a round constant
Outline

1 Introduction and Background
   - Background
   - Cube Testing
   - CUDA
   - Primitives

2 Framework

3 Experiments
   - Description of experiments
   - Results
   - Timing

4 Conclusions

5 Future Work
Design of the framework

- CUDA and Java
  - CUDA - Data collection
  - Statistical analysis in Java
- Majority of computation offloaded to GPU
Data collection

Data collection performed by CUDA program

Choose a random subset of the plaintext bits as the cube variables say $c_1, c_2, \ldots c_n$
Choose a random subset of the plaintext bits as the superpoly variables say $s_1, s_2, \ldots s_m$

{Outer parallel loop - splitting among thread blocks}

\begin{verbatim}
for i = 1 to N do
    Choose a random assignment for $s_1, s_2, \ldots s_m$
    {Inner parallel loop - splitting among threads } 
    for $c_1, c_2, \ldots c_n$ = 000...00 to 111...11 do
        $Q_i = Q_i \oplus F_i(c_1, c_2, \ldots c_n, s_1, s_2, \ldots s_m)$
    end for
end for
\end{verbatim}

Write the values of $Q_i$ to a output file
786432274
203b3a06433a16480d4077af23830b01
43 102 86 81 10 17
51 72 107 41 45 12 71 31 95 117
16
0 FAC660A226D84441536B6DBE1F4DE419
1 15BD983E24D135969C5F891007805132
2 E6327AEC447FBEA5CFE0D97F0A7A7AD9
3 426A1ABBE71F6181FA9551967BCAB1CD
4 E907E333D4C476ADB0076DF299FE9C20
5 B4DAEB1D515767B9F5C5DA99CC33DE17
6 FB6AE7838E383226EB55B9C41E4FD227
7 0DE3FC648462065F200CAABCAC6792A5
Statistical Analysis

- Output files analysed by Java program
- Study data with different significance levels, number of samples
- Statistical functions - Parallel Java Library[4]
- Plots - Cube Test Library[5]
1 Introduction and Background
   - Background
   - Cube Testing
   - CUDA
   - Primitives

2 Framework

3 Experiments
   - Description of experiments
   - Results
   - Timing

4 Conclusions

5 Future Work
Balance Test of 1 superpoly

- Let $Q$ be a superpoly

- Hypothesis
  - $Q$ is a random polynomial
  - The value of $Q$ is 0/1 with equal probability

- Let $N$ be number of random assignments to superpoly variables

- $\chi^2$ test
  - Expected number of 0s = Expected number of 1s = $N/2$
  - $n_0 = \text{Observed number of 0s}$
  - $n_1 = \text{Observed number of 1s}$
  - $\chi^2 = \frac{(n_0 - N/2)^2}{N/2} + \frac{(n_1 - N/2)^2}{N/2}$
  - Calculate $p$-value (for $\chi^2$ distribution with 1 degree of freedom)
  - Test fails if $p$-value less than significance level
Balance Test of all superpolys

- Hypothesis (significance level of $P$)
  - A superpoly will pass the balance test with a probability of $(1 - P)$

- Let $N$ be the number of superpolys being tested

- $\chi^2$ Test
  - $N_p = \text{Expected number of passes} = (1 - P) \cdot N$
  - $N_f = \text{Expected number of failures} = P \cdot N$
  - $n_0 = \text{Observed number of passed tests}$
  - $n_1 = \text{Observed number of failed tests}$
  - $\chi^2 = \frac{(n_0 - N_p)^2}{N_p} + \frac{(n_1 - N_f)^2}{N_f}$
  - Calculate $p$-value (for $\chi^2$ distribution with 1 degree of freedom)
  - Test fails if $p$-value less than significance level
Output/Output independence Test

- Let $Q_i$ and $Q_j$ be two superpolys
- **Hypothesis**
  - The value of $Q_i$ is independent of the value of $Q_j$
- Let $N$ be number of random assignments to superpoly variables
- $\chi^2$ Test
  - Expected number of (0,0) values for $(Q_i, Q_j) = N/4$ (same for (0,1), (1,0), (1,1))
  - Let $n_0, n_1, n_2$ and $n_3$ be the observed counts of (0,0),(0,1), (1,0) and (1,1) values for $(Q_i, Q_j)$
  - $\chi^2 = \frac{(n_0-N/4)^2}{N/4} + \frac{(n_1-N/4)^2}{N/4} + \frac{(n_2-N/4)^2}{N/4} + \frac{(n_3-N/4)^2}{N/4}$
  - Calculate $p$-value (for $\chi^2$ distribution with 3 degrees of freedom)
  - Test fails if $p$-value less than significance level
AES-128 Balance Test

**Figure:** AES-128 Balance Test

![AES Balance Test](image)
Figure: AES-128 Balance Test
AES-128 Output/Output Independence Test

Figure: AES-128 Independence Test
AES-128 Output/Output Independence Test

Figure: AES-128 Independence Test
Figure: Threefish-256 Balance Test
Threefish-256 Balance Test

Figure: Threefish-256 Balance Test
Figure: Threefish-256 Independence Test
Threefish-256 Output/Output Independence Test

Figure: Threefish-256 Independence Test
Figure: Keccak-\(f[1600]\) Balance Test
Figure: Keccak-$f[1600]$ Independence Test
Figure: Speedup (1 thread per block)
Speedup plots

**Figure:** Speedup (32 thread per block)
**Speedup plots**

**Figure:** Speedup (64 thread per block)
Figure: Speedup (20 thread blocks)
Outline

1 Introduction and Background
   - Background
   - Cube Testing
   - CUDA
   - Primitives

2 Framework

3 Experiments
   - Description of experiments
   - Results
   - Timing

4 Conclusions

5 Future Work
Conclusions

- GPUs are excellent platforms for executing massively parallel programs
- Non randomness was not detected in the balance test on all three primitives
- Output/Output independence test shows non-randomness in all three primitives
Outline

1 Introduction and Background
   - Background
   - Cube Testing
   - CUDA
   - Primitives

2 Framework

3 Experiments
   - Description of experiments
   - Results
   - Timing

4 Conclusions

5 Future Work
Future Work

- Perform similar studies on the remaining SHA-3 candidates
- Running different cube tests on primitives, such as tests for linear, neutral variables
- More performance testing on the framework
- OpenCL, object-oriented approach
References


3. NVIDIA. *NVIDIA CUDA programming guide*.

   http://www.cs.rit.edu/~ark/pj.shtml

   http://www.cs.rit.edu/~ark/parallelcrypto/cubetest01/
Questions?