ECE 545
Digital System Design with VHDL

Course web page:
ECE web page → Courses → Course web pages → ECE 545
http://ece.gmu.edu/coursewebpages/ECE/ECE545/F10/

Kris Gaj

Research and teaching interests:
• reconfigurable computing
• computer arithmetic
• cryptography
• network security

Contact:
The Engineering Building, room 3225
kgaj@gmu.edu

Office hours: Monday, 7:30-8:30 PM,
Wednesday, 6:00-7:00 PM,
and by appointment

ECE 545
Part of:
MS in Computer Engineering
MS in Electrical Engineering

Strongly suggested for two concentration areas:
Digital Systems Design
Microprocessor and Embedded Systems
Elective course in the remaining concentration areas

Elective

Grading Scheme

• Homework - 10%
• Project - 40%
• Midterm Exam - 20%
• Final Exam - 30%

DIGITAL SYSTEMS DESIGN

Concentration advisors: Kris Gaj, Jens-Peter Kaps, Ken Hintz

1. ECE 545 Digital System Design with VHDL
   – K. Gaj, project, FPGA design with VHDL,
     Aldec/Mentor Graphics, Xilinx/Altera

2. ECE 645 Computer Arithmetic
   – K. Gaj, project, FPGA design with VHDL or Verilog,
     Aldec/Mentor Graphics, Xilinx/Altera

3. ECE 681 VLSI Design for ASICs
   – N. Klimavicz, project/lab, back-end ASIC design with
     Synopsys tools

4. ECE 586 Digital Integrated Circuits
   – D. Ioannou, R. Mulpuri

5. ECE 682 VLSI Test Concepts
   – T. Storey

Courses

Design level
Digital System Design with VHDL
Computer Arithmetic
VLSI Design
VLSI Test Concepts

algorithmic
register-transfer
gate
transistor
layout
devices

Physical Design
Digital Integrated Circuits

ECE 545
ECE 645
ECE 586
ECE 681
ECE 682

Semiconductor Device Fundamentals
MOS Device Electronics
**Midterm exam 1**
- 2 hours 30 minutes
- in class
- design-oriented
- open-books, open-notes
- practice exams will be available on the web

  **Tentative date:**
  Monday, November 1st

**Final exam**
- 2 hours 45 minutes
- in class
- design-oriented
- open-books, open-notes
- practice exams will be available on the web

  **Date:**
  Monday, December 20, 7:30-10:15pm

**Project**
- individual
- semester-long
- related to the research project conducted by Cryptographic Engineering Research Group (CERG) at GMU
- supporting NIST (National Institute of Standards and Technology) in the evaluation of candidates for a new cryptographic standard

**Background**

**Hash Function**

It is computationally infeasible to find such \( m \) and \( m' \) that \( h(m)=h(m') \)

fixed length

\[ h(m) \]

hash function

arbitrary length

\[ m \]

message

hash value
Main Application: Digital Signature

**Signature**

**HANDWRITTEN**

A6E3891F2939E38C745B

2528936CA345BEF5349

245CBA65344BE349EA47

**DIGITAL**

Main Goals:

- unique identification
- proof of agreement to the contents of the document

Typical Digital Signature Scheme

**Alice**

- Message
- Signature

**Bob**

- Message
- Signature

**Hash function**

Hash value

**Public key cipher**

Yes

No

Alice’s private key

Alice’s public key

Handwritten and Digital Signatures

**Common Features**

<table>
<thead>
<tr>
<th>Handwritten signature</th>
<th>Digital signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Unique</td>
<td>6. Associated physically with the document</td>
</tr>
<tr>
<td>2. Impossible to be forged</td>
<td>7. Almost identical for all documents</td>
</tr>
<tr>
<td>3. Impossible to be denied by the author</td>
<td>8. Usually at the last page</td>
</tr>
<tr>
<td>4. Easy to verify by an independent judge</td>
<td>9. Covers the entire document</td>
</tr>
<tr>
<td>5. Easy to generate</td>
<td></td>
</tr>
</tbody>
</table>

Handwritten and Digital Signatures

**Differences**

<table>
<thead>
<tr>
<th>Handwritten signature</th>
<th>Digital signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>6. Associated physically with the document</td>
<td>6. Can be stored and transmitted independently of the document</td>
</tr>
<tr>
<td>7. Almost identical for all documents</td>
<td>7. Function of the document</td>
</tr>
<tr>
<td>8. Usually at the last page</td>
<td>8. Covers the entire document</td>
</tr>
</tbody>
</table>

Hash function algorithms

<table>
<thead>
<tr>
<th>Customized (dedicated)</th>
<th>Based on block ciphers</th>
<th>Based on modular arithmetic</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD2, MD4</td>
<td>MDC-2, MDC-4</td>
<td>MASH-1, 1988-1996</td>
</tr>
<tr>
<td>IBF, Bracht, Meyer, Schilling, 1988</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MD5</td>
<td>SHA-0, SHA-1</td>
<td>RIPEMD-160, 1992</td>
</tr>
<tr>
<td>NIST, 1992</td>
<td>NIST, 1995</td>
<td>European RACE Integrity Primitives Evaluation Project, 1992</td>
</tr>
<tr>
<td>SHA-256, SHA-384, SHA-512</td>
<td></td>
<td>NSA, 2000</td>
</tr>
</tbody>
</table>

Attacks against dedicated hash functions known by 2004

<table>
<thead>
<tr>
<th>MD2, MD4</th>
<th>Broken, H. Dobbertin, 1995</th>
</tr>
</thead>
<tbody>
<tr>
<td>(one hour on PC, 20 free bytes at the start of the message)</td>
<td></td>
</tr>
</tbody>
</table>

| MD5 | Weakness discovered, 1995 NSA, France |
|     | Reduced round version broken, Dobbertin 1995 |
|     | SHA-1, 1995 NSA, 1998 France |
|     | RIPEMD-160, 10 hours on PC |

<table>
<thead>
<tr>
<th>SHA-256, SHA-384, SHA-512</th>
<th>Partially broken</th>
</tr>
</thead>
</table>
Cryptographic Standards

So how the cryptographic standards have been created so far?

NSA-developed Cryptographic Standards

<table>
<thead>
<tr>
<th>Year</th>
<th>Block Ciphers</th>
<th>Hash Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1977</td>
<td>DES – Data Encryption Standard</td>
<td>SHA-0</td>
</tr>
<tr>
<td>1999</td>
<td>Triple DES</td>
<td>SHA-1 – Secure Hash Algorithm</td>
</tr>
<tr>
<td>2005</td>
<td></td>
<td>SHA-2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Cryptographic Standard Contests

<table>
<thead>
<tr>
<th>Year</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>IX.1997</td>
<td>AES</td>
</tr>
<tr>
<td>X.2000</td>
<td>15 block ciphers → 1 winner</td>
</tr>
<tr>
<td>2000</td>
<td>34 stream ciphers → 4 SW+4 HW winners</td>
</tr>
<tr>
<td>X.2002</td>
<td>51 hash functions → 1 winner</td>
</tr>
<tr>
<td>X.2004</td>
<td>SHA-3</td>
</tr>
<tr>
<td>V.2008</td>
<td>eSTREAM</td>
</tr>
<tr>
<td>X.2012</td>
<td>SHA-3</td>
</tr>
</tbody>
</table>

2^n operations

Schneier, 2005

In hardware:
Machine similar to the one used to break DES:
Cost = $50,000-$70,000
Time: 18 days
or
Cost = $0.9-$1.26M
Time: 24 hours

In software:
Computer network similar to distributed.net
used to break DES (~331,252 computers):
Cost = ~ $0
Time: 7 months

NSA
National Security Agency
(also known as “No Such Agency”
or “Never Say Anything”)

Created in 1952 by president Truman
Goals:
• designing strong ciphers (to protect U.S. communications)
• breaking ciphers (to listen to non-U.S. communications)

Budget and number of employees kept secret
Largest employer of mathematicians in the world
Larger purchaser of computer hardware

What was discovered in 2004-2005?

(manually, without using a computer)

SHA-256, SHA-384, SHA-512

Crypto 2004
(1 hr on a PC)
SHA-3 Contest - NIST Evaluation Criteria

Software or hardware?

SOFTWARE

- Security
- Software Efficiency
  - ASICs
  - FPGAs
- Flexibility
- Simplicity
- Licensing

HARDWARE

- Security of data during transmission
- Hardware Efficiency
  - speed
  - random key generation
  - access control to keys
  - resistance to side-channel attacks
  - tamper resistance
- Flexibility
  - (new cryptoalgorithms, protection against new attacks)

Primary efficiency indicators

**Software**

- Speed
- Memory

**Hardware**

- Speed
- Area
- Power consumption

Efficiency parameters

**Latency**

- Time to encrypt/decrypt a single block of data

**Throughput = Speed**

- Encryption/decryption
  - Number of bits encrypted/decrypted in a unit of time

Throughput = Block_size · Number_of_blocks_processed_simultaneously / Latency


- **June 1998**
  - 15 Candidates
    - from USA, Canada, Belgium, France, Germany, Norway, UK, Israel, Korea, Japan, Australia, Costa Rica
- **Round 1**
  - Security
  - Hardware efficiency
  - Flexibility
- **August 1999**
  - 5 final candidates
    - Mars, RC6, Rijndael, Serpent, Twofish
- **Round 2**
  - Security
  - Hardware efficiency
- **October 2000**
  - 1 winner: Rijndael
    - Belgium

Speed of the final AES candidates in Xilinx FPGAs

**Speed [Mbit/s]**

- Serpent
- Rijndael
- Twofish
- RC6
- Mars

K. Gaj, P. Chodowiec, AES3, April, 2000
Survey filled by 167 participants of the Third AES Conference, April 2000

Results of the NSA group

Efficiency in software: NIST-specified platform

NIST Report: Security

GMU Team Goals

• Fair and comprehensive methodology for evaluation of hardware performance in FPGAs

• High-speed fully autonomous implementations of all 14 SHA-3 candidates & SHA-2

256-bit & 512-bit variants optimized for the maximum throughput to area ratio

• Open-source benchmarking tool supporting optimization of tool options and efficient generation of results for multiple FPGA families
Primary Designers of GMU Codes
Ekawat HomSirirakomol
a.k.a “Ice”
Marcin Rogawski

Developed optimized VHDL implementations of 14 Round 2 SHA-3 candidates + SHA-2 in two variants each (256 & 512-bit output), for some functions using several alternative architectures

Methodology

Comprehensive Evaluation

- two major vendors: Altera and Xilinx (~90% of the market)
- multiple high-performance and low-cost families

<table>
<thead>
<tr>
<th>Technology</th>
<th>Altera</th>
<th>Xilinx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low-cost</td>
<td>High-performance</td>
<td>Low-cost</td>
</tr>
<tr>
<td>90 nm</td>
<td>Cyclone II</td>
<td>Stratix II</td>
</tr>
<tr>
<td>65 nm</td>
<td>Cyclone III</td>
<td>Stratix III</td>
</tr>
</tbody>
</table>

Uniform Evaluation

- Language: VHDL
- Tools: FPGA vendor tools
- Interface
- Performance Metrics
- Design Methodology
- Benchmarking

Why Interface Matters?

- Pin limit
  Total number of i/o ports ≤ Total number of an FPGA i/o pins
- Support for the maximum throughput
  Time to load the next message block ≤ Time to process previous block

Interface: Two possible solutions

Length of the message communicated at the beginning
+ easy to implement passive source circuit
− area overhead for the counter of message bits

Dedicated end of message port
− more intelligent source circuit required
+ no need for internal message bit counter
SHA Core: Interface & Typical Configuration

- SHA core is an active component; surrounding FIFOs are passive and widely available.
- Input interface is separate from an output interface.
- Processing a current block, reading the next block, and storing a result for the previous message can be all done in parallel.

Performance Metrics

Primary
1. Throughput (single long message)
2. Area
3. Throughput / Area

Secondary
1. Hash Time for Short Messages (up to 1000 bits)

Choice of Optimization Target

Primary Optimization Target: Throughput to Area Ratio
Features:
- practical: good balance between speed and cost
- very reliable guide through the entire design process, facilitating the choice of
  - high-level architecture
  - implementation of basic components
  - choice of tool options
- leads to high-speed, close-to-maximum-throughput designs

Performance Metrics - Area

Resource Utilization_{Spantek} = (#CLB slices, #BRAMs, #MULs)

Resource Utilization_{Cyclone III} = (#LE, #memory_bits, #MULs).

We force these vectors to look as follows through the synthesis and implementation options:

<table>
<thead>
<tr>
<th>Resource Utilization_{Spantek}</th>
<th>Resource Utilization_{Cyclone III}</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 ; 0 ; 0</td>
<td>0 ; 0 ; 0</td>
</tr>
</tbody>
</table>

Area

Our Design Flow

Specification
Datapath Block diagram
Controller ASM Chart
Library of Basic Components

Interface
Controller Template

Formulas for Throughput & Hash time
Max. Clock Freq., Resource Utilization

Throughput, Area, Throughput/Area, Hash Time for Short Messages
Basic Operations of 14 SHA-3 Candidates

<table>
<thead>
<tr>
<th>Function</th>
<th>NTT</th>
<th>Linear code</th>
<th>GF MUL</th>
<th>MUL</th>
<th>mADD</th>
<th>AES/MIB</th>
<th>ADX/MIB</th>
<th>Boolean</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLAKE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>XORB</td>
<td>XORB</td>
<td></td>
</tr>
<tr>
<td>BMI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>XORB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Civilian</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIG5</td>
<td>AES</td>
<td>(20, 40)</td>
<td>(60, 60)</td>
<td></td>
<td></td>
<td>XORB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gost128</td>
<td>AES</td>
<td>(40, 80)</td>
<td>(80, 80)</td>
<td></td>
<td></td>
<td>XORB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hamsi</td>
<td>AES</td>
<td>(40, 80)</td>
<td>(80, 80)</td>
<td></td>
<td></td>
<td>XORB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JH</td>
<td>AES</td>
<td>(40, 80)</td>
<td>(80, 80)</td>
<td></td>
<td></td>
<td>XORB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Keccak</td>
<td>AES</td>
<td>(40, 80)</td>
<td>(80, 80)</td>
<td></td>
<td></td>
<td>XORB</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NTT – Number Theoretic Transform, GF MUL – Galois Field multiplication, MUL – integer multiplication, mADDn – multioperand addition with n operands

ATHENa – Automated Tool for Hardware Evaluation

http://cryptography.gmu.edu/athena

Benchmarking open-source tool, written in Perl, aimed at an AUTOMATED generation of OPTIMIZED results for MULTIPLE FPGA platforms

Under development at George Mason University.

Basic Dataflow of ATHENa

ATHENa Major Features (1)
- synthesis, implementation, and timing analysis in batch mode
- support for devices and tools of multiple FPGA vendors:
  - Xilinx
  - Altera
- generation of results for multiple families of FPGAs of a given vendor
- automated choice of a best-matching device within a given family

ATHENa Major Features (2)
- automated verification of designs through simulation in batch mode
- support for multi-core processing
- automated extraction and tabulation of results
- several optimization strategies aimed at finding
  - optimum options of tools
  - best target clock frequency
  - best starting point of placement

User

Database query

ATHENa Server

Ranking of designs

User

Database query

ATHENa Server

Download scripts and configuration files

Database Entries

Interfaces + Testbenches

HDL + FPGA Tools

Result Summary + Database Entries

HDL + scripts + configuration files

Configurable source files

Result summary (user-friendly)

Configuration files

Testbench

Database entries (machine-friendly)

ATHENa Major Features (2)
- automated verification of designs through simulation in batch mode
- support for multi-core processing
- automated extraction and tabulation of results
- several optimization strategies aimed at finding
  - optimum options of tools
  - best target clock frequency
  - best starting point of placement
Generation of Results Facilitated by ATHENa

- batch mode of FPGA tools
- ease of extraction and tabulation of results
  - Excel, CSV (available), LaTeX (coming soon)
- optimized choice of tool options

Relative Improvement of Results from Using ATHENa

Virtex 5, 256-bit Variants of Hash Functions

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Area</th>
<th>Thr</th>
<th>Thr/Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>Groestl</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shavite-3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Luffa</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Keccak</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hamsi</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Echo</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Skein</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fugue</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHA-2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMW</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CubeHash</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLAKE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIMD</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Throughput [Mbit/s]

Virtex 5, 512-bit variants of algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECHO</td>
<td></td>
</tr>
<tr>
<td>Keccak</td>
<td></td>
</tr>
<tr>
<td>Groestl</td>
<td></td>
</tr>
<tr>
<td>Luffa</td>
<td></td>
</tr>
<tr>
<td>SHA-2</td>
<td></td>
</tr>
<tr>
<td>BMW</td>
<td></td>
</tr>
<tr>
<td>CubeHash</td>
<td></td>
</tr>
<tr>
<td>BLAKE</td>
<td></td>
</tr>
<tr>
<td>SIMD</td>
<td></td>
</tr>
<tr>
<td>Hamsi</td>
<td></td>
</tr>
<tr>
<td>Fugue</td>
<td></td>
</tr>
</tbody>
</table>

Results
### Normalization & Compressing of Results

- **Absolute result**
  
e.g., throughput in Mbits/s, area in CLB slices

- **Normalized result**
  
\[
\text{normalized result} = \frac{\text{result for SHA-3 candidate}}{\text{result for SHA-2}}
\]

- **Overall normalized result**
  
Geometric mean of normalized results for all investigated FPGA families

### Normalized Throughput & Overall Normalized Throughput

<table>
<thead>
<tr>
<th>Candidate</th>
<th>Spartan-6</th>
<th>Virtex-4</th>
<th>Virtex-5</th>
<th>Cyclone IV</th>
<th>Cyclone III</th>
<th>Stratix III</th>
<th>Stratix II</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keccak</td>
<td>4.07</td>
<td>4.07</td>
<td>4.07</td>
<td>3.94</td>
<td>3.94</td>
<td>3.94</td>
<td>3.94</td>
<td>3.94</td>
</tr>
<tr>
<td>Salsa20</td>
<td>5.01</td>
<td>6.01</td>
<td>6.01</td>
<td>5.00</td>
<td>5.00</td>
<td>5.00</td>
<td>5.00</td>
<td>5.00</td>
</tr>
<tr>
<td>RC6</td>
<td>3.89</td>
<td>3.89</td>
<td>3.89</td>
<td>3.89</td>
<td>3.89</td>
<td>3.89</td>
<td>3.89</td>
<td>3.89</td>
</tr>
</tbody>
</table>

### Overall Normalized Throughput: 256-bit variants of algorithms

Normalized to SHA-256, Averaged over 7 FPGA families

### Area [CLB slices]

**Virtex 5, 256-bit variants of algorithms**

<table>
<thead>
<tr>
<th>Candidate</th>
<th>Spartan-6</th>
<th>Virtex-4</th>
<th>Virtex-5</th>
<th>Cyclone IV</th>
<th>Cyclone III</th>
<th>Stratix III</th>
<th>Stratix II</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>CubeHash</td>
<td>1.94</td>
<td>1.94</td>
<td>1.94</td>
<td>1.94</td>
<td>1.94</td>
<td>1.94</td>
<td>1.94</td>
<td>1.94</td>
</tr>
<tr>
<td>SHA-3</td>
<td>1.93</td>
<td>1.93</td>
<td>1.93</td>
<td>1.93</td>
<td>1.93</td>
<td>1.93</td>
<td>1.93</td>
<td>1.93</td>
</tr>
</tbody>
</table>

### Area [CLB slices]

**Virtex 5, 512-bit variants of algorithms**

<table>
<thead>
<tr>
<th>Candidate</th>
<th>Spartan-6</th>
<th>Virtex-4</th>
<th>Virtex-5</th>
<th>Cyclone IV</th>
<th>Cyclone III</th>
<th>Stratix III</th>
<th>Stratix II</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>CubeHash</td>
<td>1.94</td>
<td>1.94</td>
<td>1.94</td>
<td>1.94</td>
<td>1.94</td>
<td>1.94</td>
<td>1.94</td>
<td>1.94</td>
</tr>
</tbody>
</table>
Overall Normalized Area: 256-bit variants of algorithms
Normalized to SHA-256, Averaged over 7 FPGA families

Overall Normalized Area: 512-bit variants of algorithms
Normalized to SHA-512, Averaged over 7 FPGA families

Overall Normalized Throughput/Area: 256-bit variants
Normalized to SHA-256, Averaged over 7 FPGA families

Overall Normalized Throughput/Area: 512-bit variants
Normalized to SHA-512, Averaged over 7 FPGA families

Throughput vs. Area Normalized to Results for SHA-256 and Averaged over 7 FPGA Families – 256-bit variants

Throughput vs. Area Normalized to Results for SHA-512 and Averaged over 7 FPGA Families – 512-bit variants
Execution Time for Short Messages up to 1000 bits
Virtex 5, 256-bit variants of algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Thr/Area</th>
<th>Thr</th>
<th>Area</th>
<th>Short msg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLAKE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMW</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CubeHash</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ECHO</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fugue</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Groestl</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hamsi</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JH</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Keccak</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Luffa</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shabal</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHAvite-3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIMD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Skein</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Execution Time for Short Messages up to 1000 bits
Virtex 5, 512-bit variants of algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Thr/Area</th>
<th>Thr</th>
<th>Area</th>
<th>Short msg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLAKE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMW</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CubeHash</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ECHO</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fugue</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Groestl</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hamsi</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JH</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Keccak</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Luffa</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shabal</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHAvite-3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIMD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Skein</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Summary of Results

- Throughput/Area & Throughput most crucial for high-speed implementations
- Area cannot be easily traded for Throughput

Best performers so far:
1-2. Keccak & Luffa
3. Groestl

Worst performers so far:
14. SIMD
13. ECHO
12. BMW

More About our Designs & Tools

- Cryptology e-Print Archive - 2010/445 (100+ pages)
  - Detailed hierarchical block diagrams
  - Corresponding formulas for execution time and throughput
- FPL 2010 paper
  - ATHENA features
  - Case studies
- ATHENA web site
  - Most recent results
  - Comparisons with results from other groups
  - Optimum options of tools

Comparison with Other Groups
Comparison with Best Results Reported by Other Groups
Virtex 5, 256-bit variants of algorithms

<table>
<thead>
<tr>
<th>OTHER GROUPS</th>
<th>GMU</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Area</td>
</tr>
<tr>
<td>BLAKE</td>
<td>1660</td>
</tr>
<tr>
<td>CubeHash</td>
<td>590</td>
</tr>
<tr>
<td>ECHO</td>
<td>9333</td>
</tr>
<tr>
<td>Grunet</td>
<td>1722</td>
</tr>
<tr>
<td>Hashi</td>
<td>718</td>
</tr>
<tr>
<td>Keccak</td>
<td>1412</td>
</tr>
<tr>
<td>Luffa</td>
<td>1048</td>
</tr>
<tr>
<td>Shabal</td>
<td>153</td>
</tr>
<tr>
<td>Skein (Estimated)</td>
<td>1632</td>
</tr>
</tbody>
</table>

Best Overall Reported Results as of Aug. 6, 2010
Virtex 5, 256-bit variants of algorithms

<table>
<thead>
<tr>
<th>BEST REPORTED RESULTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>BLAKE</td>
</tr>
<tr>
<td>BMW</td>
</tr>
<tr>
<td>CubeHash</td>
</tr>
<tr>
<td>ECHO</td>
</tr>
<tr>
<td>Fugue</td>
</tr>
<tr>
<td>Grueti</td>
</tr>
<tr>
<td>Hashi</td>
</tr>
<tr>
<td>IH</td>
</tr>
<tr>
<td>JH</td>
</tr>
<tr>
<td>Luffa</td>
</tr>
<tr>
<td>Shabal</td>
</tr>
<tr>
<td>SHAvite-3</td>
</tr>
<tr>
<td>SIMD</td>
</tr>
<tr>
<td>Skein</td>
</tr>
</tbody>
</table>

Throughput vs. Area: Best reported results
Virtex 5, 256-bit variants of algorithms

Analysis of Alternative Architectures - Unrolled

Analysis of Alternative Architectures - Folded
Preliminary results for CubeHash, Groestl, Keccak & Luffa in Virtex 5

Your Project
• 14 SHA-3 candidates left in the contest

• Given:
  ✓ specification of the function
  ✓ reference implementation in C
  ✓ interface
  ✓ testbench and test vectors
  ✓ GMU implementation of the basic version including
    ✓ block diagrams
    ✓ ASM charts
    ✓ short description
    ✓ formulas for execution time & throughput
    ✓ source codes
    ✓ results for Xilinx and Altera FPGAs

Your Project
Develop:
✓ Block diagram
✓ ASM chart
✓ Formulas for execution time & throughput
✓ Synthesizable code in VHDL
✓ Results for multiple families of FPGAs from Xilinx and Altera

for at least one architecture from each of the following three classes of architectures:
– Unrolled architecture
– Folded architecture
– Architecture based on the use of embedded FPGA resources (BRAMs, multipliers, DSP units, etc.)
[256 bit only, 512-bit only, or both]

What is an FPGA?

RAM Blocks and Multipliers in Xilinx FPGAs

Using Embedded FPGA Resources

Resource Utilization (unroll3) = (#CLBslices, #BRAMs, #MULs)

<table>
<thead>
<tr>
<th>Design</th>
<th>CLBslices</th>
<th>BRAMs</th>
<th>MULs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic</td>
<td>1536</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Your</td>
<td>768</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

Resource Utilization (unroll2) = (#LE, #memory bits, #MULs)

<table>
<thead>
<tr>
<th>Design</th>
<th>LE</th>
<th>Memory Bits</th>
<th>MULs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic</td>
<td>3010</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Your</td>
<td>1505</td>
<td>32 kbit</td>
<td>4</td>
</tr>
</tbody>
</table>
Block RAM

- Most efficient memory implementation
  - Dedicated blocks of memory
- Ideal for most memory requirements
  - 4 to 104 memory blocks
  - 18 kbits = 18,432 bits per block (16 k without parity bits)
  - Use multiple blocks for larger memories
- Builds both single and true dual-port RAMs
- Synchronous write and read (different from distributed RAM)

Block RAM can have various configurations (port aspect ratios)

Dual-Port Bus Flexibility

- Port A In: 16-bit Width
- Port A Out: 16-bit Width
- Port B In: 1k-bit Depth
- Port B Out: 18-bit Width

Embedded Multipliers in Spartan 3

- 18x18 bit signed multipliers with optional input/output registers

Multiplier-Accumulator - MAC

Xilinx XtremeDSP

- Starting with Virtex 4 family, Xilinx introduced DSP48 block for high-speed DSP on FPGAs
- Essentially a multiply-accumulate core with many other features
- Now also Spartan-3A and Virtex 5 have DSP blocks
DSP48 Slice: Virtex 4

Simplified Form of DSP48

Xilinx FPGA Devices

<table>
<thead>
<tr>
<th>Technology</th>
<th>Low-cost</th>
<th>High-performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>120/150 nm</td>
<td>Virtex 2, 2 Pro</td>
<td></td>
</tr>
<tr>
<td>90 nm</td>
<td>Spartan 3</td>
<td>Virtex 4</td>
</tr>
<tr>
<td>65 nm</td>
<td>Spartan 6</td>
<td>Virtex 5</td>
</tr>
<tr>
<td>45 nm</td>
<td>Spartan 6</td>
<td>Virtex 6</td>
</tr>
<tr>
<td>40 nm</td>
<td>Virgin 6</td>
<td></td>
</tr>
</tbody>
</table>

Altera FPGA Devices

<table>
<thead>
<tr>
<th>Technology</th>
<th>Low-cost</th>
<th>Mid-range</th>
<th>High-performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>130 nm</td>
<td>Cyclone</td>
<td>Stratix</td>
<td></td>
</tr>
<tr>
<td>90 nm</td>
<td>Cyclone II</td>
<td>Stratix II</td>
<td></td>
</tr>
<tr>
<td>65 nm</td>
<td>Cyclone III</td>
<td>Arria I</td>
<td></td>
</tr>
<tr>
<td>40 nm</td>
<td>Cyclone IV</td>
<td>Arria II</td>
<td></td>
</tr>
</tbody>
</table>

All Projects - Organization

- Projects divided into phases
- Deliverables for each phase submitted through Blackboard at selected checkpoints and evaluated by the instructor and/or TA
- Feedback provided to students on a best effort basis
- Final report and codes submitted using Blackboard at the end of the semester

Honor Code Rules

- All students are expected to write and debug their codes individually
- Students are encouraged to help and support each other in all problems related to the operation of the CAD tools, basic understanding of the problem.
Course Objectives

• At the end of this course you should be able to:
  • Code in VHDL for synthesis
  • Decompose a digital system into a controller (FSM) and datapath, and code accordingly
  • Write VHDL testbenches
  • Synthesize and implement digital systems on FPGAs
  • Effectively code digital systems for cryptography, signal processing, and microprocessor applications
  • This knowledge will come about through homework, exams, and an extensive project
  • The project in particular will help you know VHDL and the FPGA design flow from beginning to end

Additional Skills Learned in the Project

• Reading & understanding specification of a complex algorithm
• Design of new hardware architectures based on existing architectures (datapath & controller)
• Reading, understanding, and modifying existing VHDL code
• Using embedded resources of modern FPGAs
• Characterizing performance of your codes for multiple FPGA families

Project Task 1

• Read the following chapters from the GMU technical report published at http://eprint.iacr.org/2010/445
  • Chapter 1 Introduction & Motivation
  • Chapter 2 Methodology
  • Chapter 3 Comprehensive Designs of SHA-3 Candidates 3.1, 3.2 + subsection concerning your algorithm
  • Chapter 4 Design Summary and Results
• Download and get familiar with the package of a hash function assigned to you http://csrc.nist.gov/groups/ST/hash/sha-3/Round2/submissions_md2.html
• Read carefully the specification of your algorithm

Project Task 1 – cont.

In one week:
Meeting with the instructor devoted to fully understanding the GMU report, specification, block diagrams, interface, and timing formulas.

In two weeks:
Draft block diagrams of the
  • selected unrolled architecture
  • selected folded architecture.
Corresponding timing formulas for execution time & throughput.