Required Reading

Xilinx, Inc.
Virtex-5 FPGA Family

Virtex-5 FPGA User Guide
Chapter 5: Configurable Logic Blocks (CLBs)

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>40 nm</td>
<td>Spartan 6</td>
<td>Virtex 6</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></td>
<td>Stratix</td>
</tr>
<tr>
<td>90 nm</td>
<td>Cyclone II</td>
<td></td>
<td>Stratix II</td>
</tr>
<tr>
<td>65 nm</td>
<td>Cyclone III</td>
<td>Arria I</td>
<td>Stratix III</td>
</tr>
<tr>
<td>40 nm</td>
<td>Cyclone IV</td>
<td>Arria II</td>
<td>Stratix IV</td>
</tr>
</tbody>
</table>

High-Performance Xilinx FPGAs
Virtex 5
Arrangement of Slices within the CLB

Row and Column Relationship between CLBs and Slices

Major Differences between Xilinx Families

<table>
<thead>
<tr>
<th>Look-Up Tables</th>
<th>Spartan 3</th>
<th>Virtex 4</th>
<th>Virtex 5, Virtex 6, Spartan 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-input</td>
<td>6-input</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number of CLB slices per CLB</th>
<th>4</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of LUTs per CLB slice</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

Distributed RAM Configurations

<table>
<thead>
<tr>
<th>RAM</th>
<th>Number of LUTs</th>
</tr>
</thead>
<tbody>
<tr>
<td>32 x 1s</td>
<td>1</td>
</tr>
<tr>
<td>32 x 1D</td>
<td>2</td>
</tr>
<tr>
<td>32 x 2D</td>
<td>4</td>
</tr>
<tr>
<td>32 x 64D D1</td>
<td>4</td>
</tr>
<tr>
<td>64 x 1D</td>
<td>2</td>
</tr>
<tr>
<td>64 x 1D D1</td>
<td>4</td>
</tr>
<tr>
<td>64 x 3D D1</td>
<td>4</td>
</tr>
<tr>
<td>128 x 10</td>
<td>2</td>
</tr>
<tr>
<td>128 x 1D</td>
<td>4</td>
</tr>
<tr>
<td>256 x 10</td>
<td>4</td>
</tr>
</tbody>
</table>

Notes:
1. S = single-port configuration; D = dual-port configuration; Q = quad-port configuration; SDP = single dual-port configuration.
2. RAM28DM is the associated primitive for this configuration.
3. RAM28MM is the associated primitive for this configuration.
### ROM Configurations

<table>
<thead>
<tr>
<th>ROM</th>
<th>Number of LUTs</th>
</tr>
</thead>
<tbody>
<tr>
<td>64 x 1</td>
<td>1</td>
</tr>
<tr>
<td>128 x 1</td>
<td>2</td>
</tr>
<tr>
<td>256 x 1</td>
<td>1</td>
</tr>
</tbody>
</table>

### 32-bit Shift Register, SRL

![32-bit Shift Register, SRL](image)

### 32-bit Shift Register

![32-bit Shift Register](image)

### Dual 16-bit Shift Register

![Dual 16-bit Shift Register](image)
Major Differences between Xilinx Families

<table>
<thead>
<tr>
<th></th>
<th>Spartan 3</th>
<th>Virtex 5, Virtex 6, Spartan 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Single-Port Memory Size per LUT</td>
<td>16 x 1</td>
<td>64 x 1</td>
</tr>
<tr>
<td>Maximum Shift Register Size per LUT</td>
<td>16 bits</td>
<td>32 bits</td>
</tr>
<tr>
<td>Number of adder stages per CLB slice</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

AltECE	
8
–
FPGA	
and	
ASIC	
Design	
with	
VHDL

Low-cost Altera FPGAs

Alterna Cyclone III
Logic Element (LE) – Normal Mode
Altera Cyclone III Logic Element (LE) – Arithmetic Mode

High-Performance Altera FPGAs

Stratix III Logic Array Blocks (LABs)

High-Level Block Diagram of the Stratix III ALM

Altera Stratix III Adaptive Logic Modules (ALM) – Normal Mode

4 × 2 Crossbar Switch Example
Register Packing

Template for Seven-Input Functions Supported in Extended LUT Mode

Altera Stratix III, Stratix IV
Adaptive Logic Modules (ALM) – Arithmetic Mode

Performing Operation
\[ R = (X < Y) ? Y : X \]

Three Operand Addition Utilizing Shared Arithmetic Mode

LUT-Register Mode
ATHENa – Automated Tool for Hardware EvaluatioN

Supported in part by the National Institute of Standards & Technology (NIST)

ATHENa Team

Venkata “Vinny” MS CpE student
Ekawat “Ice” PhD CpE student
Marcin PhD ECE student
John PhD ECE student
Rajesh PhD ECE student
Michal PhD exchange student from Slovakia

Why Athena?

“The Greek goddess Athena was frequently called upon to settle disputes between the gods or various mortals. Athena Goddess of Wisdom was known for her superb logic and intellect. Her decisions were usually well-considered, highly ethical, and seldom motivated by self-interest.”

from “Athena, Greek Goddess of Wisdom and Craftsmanship”

Basic Dataflow of ATHENa

User

Database query

ATHENa Server

Ranking of designs

HDL + scripts + configuration files

Result Summary + Database Entries

Download scripts and configuration files

HDL + FPGA Tools

Designer

Interfaces

Testbenches

configuration files

constraint files

testbench

synthesizable source files

result summary (user-friendly)
database entries (machine-friendly)
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
  - Virtex 5
  - Virtex 6
  - Virtex 7
- 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

Generation of Results Facilitated by ATHENA
- batch mode of FPGA tools
- ease of extraction and tabulation of results
  - Text Reports, Excel, CSV (Comma-Separated Values)
- optimized choice of tool options
  - GMU_optimization_1 strategy

Relative Improvement of Results from Using ATHENA
Virtex 5, 256-bit Variants of Hash Functions
Ratios of results obtained using ATHENA suggested options vs. default options of FPGA tools

Other (Somewhat) Similar Tools
- ExploreAhead (part of PlanAhead)
- Design Space Explorer (DSE)
- Boldport Flow
- EDAX10 Cloud Platform

Distinguishing Features of ATHENA
- Support for multiple tools from multiple vendors
- Optimization strategies aimed at the best possible performance rather than design closure
- Extraction and presentation of results
- Seamless integration with the ATHENA database of results
How To Start Working With ATHENA?

One-Time Tasks

1. Download and unzip ATHENA:
   http://cryptography.gmu.edu/athena/
2. Read the Tutorial!
3. Install the Required Tools
   (see Tutorial - Part 1 – Tools Installation)
4. Run ATHENA_setup

How To Start Working With ATHENA?

Repetitive Tasks

1. Prepare or modify your source files & source_list.txt
2. Modify design.config.txt + possibly other configuration files
3. Run ATHENA

---

**design.config.txt**

Your Design

```
# directory containing synthesizable source files for the project
SOURCE_DIR = examples/sha256_rs

# A file list containing list of files in the order suitable for synthesis and implementation
# low level modules first, top level entity last
SOURCE_LIST_FILE = source_list.txt

# project name
# it will be used in the names of result directories
PROJECT_NAME = SHA256

# name of top level entity
TOP_LEVEL_ENTITY = sha256

# name of top level architecture
TOP_LEVEL_ARCH = rs_arch

# name of clock net
CLOCK_NET = clk
```

---

**design.config.txt**

Timing Formulas

```
# formula for latency
LATENCY = TCLK*65

# formula for throughput
THROUGHPUT = 512/(TCLK*65)
```

---

**design.config.txt**

Application & Optimization Target

```
# OPTIMIZATION_TARGET = speed | area | balanced
OPTIMIZATION_TARGET = speed

# OPTIONS = default | user
OPTIONS = default

# APPLICATION = single_run | exhaustive_search | placement_search | frequency_search
# GRAY_Optimization_1 | GRAY_Xilinx_optimization_1
APPLICATION = single_run

# TRIM_MODE = off | op | delete
TRIM_MODE = op
```

---

**design.config.txt**

FPGA Families

```
# commenting the next line removes all families of Xilinx
FPGA_VENDOR = xilinx

# commenting the next line removes a given family
FPGA_FAMILY = spartan3

# FPGA_DEVICES = list of devices> | best_match | all

# SYM_CONSTRAINT_FILE = default
# IMP_CONSTRAINT_FILE = default
# REQ_SYN_FREQ = 120
# REQ_IMP_FREQ = 110
# MAX.Slice_Utilization = 0.8
# MAX.BRAM_Utilization = 0.8
# MAX.MUL_Utilization = 1
# MAX.Pin_Utilization = 0.9
END_FAMILY
```

---

END_VENDOR
### design.config.txt

#### FPGA Families

```
# commenting the next line removes all families of Altera
FPGA_VENDOR = altera

# commenting the next line removes a given family
FPGA_FAMILY = spartan6

# FPGA_DEVICES = list of devices | best_match | all
FPGA_DEVICES = best_match

SYN_CONSTRAINT_FILE = default

CHIP_CONSTRAINT_FILE = default

REQ_IMP_FREQ = 120
MAX_SLOW_UTILIZATION = 0.8
MAX_MOST_UTILIZATION = 0.8
MAX_DSP_UTILIZATION = 0
MAX_MUX_UTILIZATION = 0

END FAMILY

END VENDOR
```

### Library Files

**device_lib/xilinx_device_lib.txt**

<table>
<thead>
<tr>
<th>FAMILY</th>
<th>DEVICE</th>
<th>RUN</th>
<th>LUTs</th>
<th>%</th>
<th>SLICES</th>
<th>%</th>
<th>BRAMs</th>
<th>%</th>
<th>MULTs</th>
<th>%</th>
<th>DSPs</th>
<th>%</th>
<th>IO</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>xc6slx9csg324-3*</td>
<td>1</td>
<td>44</td>
<td>1</td>
<td>21</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>56</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>GENERIC</td>
<td>DEVICE</td>
<td>RUN</td>
<td>LUTs</td>
<td>%</td>
<td>SLICES</td>
<td>%</td>
<td>BRAMs</td>
<td>%</td>
<td>MULTs</td>
<td>%</td>
<td>DSPs</td>
<td>%</td>
<td>IO</td>
<td>%</td>
</tr>
</tbody>
</table>

### Result Files

**report_resource_utilization.txt**

<table>
<thead>
<tr>
<th>DEVICE</th>
<th>RESOURCE</th>
<th>BUS</th>
<th>SLICE</th>
<th>SRC</th>
<th>LUT</th>
<th>SRC</th>
<th>RAM</th>
<th>SRC</th>
<th>MULT</th>
<th>SRC</th>
<th>DSP</th>
<th>SRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>xilinx:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>virtex6:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Result Files

**report_synth_cost.txt**

<table>
<thead>
<tr>
<th>DEVICE</th>
<th>RESOURCE</th>
<th>BUS</th>
<th>SLICE</th>
<th>SRC</th>
<th>LUT</th>
<th>SRC</th>
<th>RAM</th>
<th>SRC</th>
<th>MULT</th>
<th>SRC</th>
<th>DSP</th>
<th>SRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>xilinx:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>virtex6:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**device_lib/altera_device_lib.txt**

### Library Files

- Files created during ATHENa setup
- Characterize FPGA families and libraries available in the version of Xilinx and Altera tools installed on your computer

- Currently supported tool versions:
  - Xilinx WebPACK 9.1, 9.2, 10.1, 11.5, 12.1, 12.2, 12.3
  - Altera Quartus II Web Edition 8.1, 8.2, 9.0, 9.1, 10.0
  - Altera Quartus II Subscription Edition 9.1, 10.0

- In case a library for a given version not available yet, use a library from the closest available version
### design.config.txt

**Global Generics**

```
GLOBAL_GENERICS_BEGIN
# n is currently set to the default value i.e n=16
# for other values of n, modify the formulas for Latency and Throughput accordingly
n = 16

# Memory type: 0 = MEM_DISTRIBUTED, 1 = MEM_EMBEDDED
mem_type = 0, 1

# Adder type: 0 = ADD_SCCA_BASED (Simple Carry Chain Adder, "+" in VHDL), 1 = ADD_DSP_BASED
adder = 0, 1

# Multiplier type: 0 = MAX_LOGIC_BASED (multiplier based on configurable logic), 1 = MAX_DSP_BASED
multiplier = 0, 1

# Allowed combinations of adder and multiplier types:
# (adder_type, multiplier_type) = (0, 0), (0, 1), (1, 0), (1, 1)

GLOBAL_GENERICS_END
```

### design.config.txt

**FPGA Family Specific Generics**

```
FPGA_FAMILY = Cyclone II

GENERICS_BEGIN
# FPGA vendor: 0 = XILINX, 1 = ALTERA
vendor = 0

# Memory block size: 0 = M128, 1 = M256, 2 = M512, 3 = M1K, 4 = M4K, 5 = M16K, 6 = M64K
mem_block_size = 1

GENERICS_END
```

---

### design.config.txt

**Functional Simulation (1)**

```
# directory containing source files of the testbench
VERIFICATION_DIR = <examples/sha256_rs/tb>

# A file containing a list of testbench files in the order suitable for compilation;
# line modules first, top level entity last.
# Test vector files should be located in the same directory and listed
# in the same file, unless fixed path is used. Please refer to tutorial for more detail.
VERIFICATION_LIST_FILE = <tb_srcs.txt>

# name of testbench's top level entity
TB_TOP_LEVEL_ENTITY = <sha_tb>

# name of testbench's top level architecture
TB_TOP_LEVEL_ARCH = <design>
```

---

### design.config.txt

**Functional Simulation (2)**

```
# MAX_TIME_FUNCTIONAL_VERIFICATION = <Time Sort>
# max_time = 10000

# If blank, simulation will run until it finishes
# = no-changes in signals, i.e., clock is stopped and no more inputs coming in.
MAX_TIME_FUNCTIONAL_VERIFICATION = <off>

# Perform only verification (synthesis and implementation parameters are ignored)
# VERIFICATION_ONLY = <on | off>
VERIFICATION_ONLY = <off>
```

---

### design.config.txt

**Result Files**

```
Result Files

report_execution_time.txt

design.circuit

design.config.txt

design.vhd

design.vhd

design.vhd

design.vhd

design.vhd

report_executable.txt

design.config.txt
```

---

### design.config.txt

**GLOBAL GENERICS**

```
GLOBAL_GENERICS_BEGIN

| GENERIC | DEVICE            | RUN | Synthesis Time | Implementation Time | Elapsed Time |
|---------|-------------------|-----|----------------+---------------------+--------------|
| xilinx  | virtex6           | 1   | 0d 0h:0m:39s   | 0d 0h:1m:50s        | 0d 0h:2m:29s |

| GENERIC | DEVICE            | RUN | Synthesis Time | Implementation Time | Elapsed Time |
|---------|-------------------|-----|----------------+---------------------+--------------|
| xilinx  | spartan6          |     |                |                     |              |

| GENERIC | DEVICE            | RUN | Synthesis Time | Implementation Time | Elapsed Time |
|---------|-------------------|-----|----------------+---------------------+--------------|
| xilinx  | spartan3          |     |                |                     |              |

GLOBAL_GENERICS_END
```
ATHENA Database – Result View

- Algorithm parameters
- Design parameters
  - Optimization target
  - Architecture type
  - Datapath width
  - I/O bus widths
  - Availability of source code
- Platform
  - Vendor, Family, Device
- Timing
  - Maximum clock frequency
  - Maximum throughput
- Resource utilization
  - Logic blocks (Slices/LEs/ALUTs)
  - Multipliers/DSP units
- Tools
  - Names & versions
  - Detailed options
- Credits
  - Designers & contact information

ATHENA Database – Compare Feature

Matching fields in grey
Non-matching fields in red and blue

Currently in the Database

Hash Functions in FPGAs

GMU Results for

- 20 hash functions
  ( 14 Round 2 SHA-3 + 5 Round 3 SHA-3 + SHA-2 )
  x 2 variants (256-bit output & 512-bit output)
  x 11 FPGA families = 440 combinations

(440-not_fitting) = 423 optimized results

Coming soon!

- GMU results for Hash Functions in FPGAs
  - Folded & unrolled architectures
  - Pipelined architectures
  - Lightweight architectures
  - Architectures based on embedded resources
- Other Groups’ results for Hash Functions in FPGAs
- Other Groups’ results for Hash Functions in ASICs
- Modular Arithmetic (basis of public key cryptography) in FPGAs & ASICs
Possible Future Customizations

The same basic database can be customized and adapted for other domains, such as
• Digital Signal Processing
• Bioinformatics
• Communications
• Scientific Computing, etc.

ATHENa Website

http://cryptography.gmu.edu/athena/

• Download of ATHENa Tool
• Links to related tools

SHA-3 Competition in FPGAs & ASICs

• Specifications of candidates
• Interface proposals
• RTL source codes
• Testbenches
• ATHENa database of results
• Related papers & presentations

GMU Source Codes and Block Diagrams

• First batch of GMU Source Codes for all Round 3 SHA-3 Candidates & SHA-2 made available at the ATHENa website at: http://cryptography.gmu.edu/athena

• Included in this release:
  • Basic architectures
  • Folded architectures
  • Unrolled architectures
  • Each code supports two variants: with 256-bit and 512-bit output.
  • Each source code accompanied by comprehensive hierarchical block diagrams

ATHENa Result Replication Files

• Scripts and configuration files sufficient to easily reproduce all results (without repeating optimizations)
• Automatically created by ATHENa for all results generated using ATHENa
• Stored in the ATHENa Database

In the same spirit of Reproducible Research as:

• J. Claerbout (Stanford University)


Benchmarking Goals Facilitated by ATHENa

Comparing multiple:
1. cryptographic algorithms
2. hardware architectures or implementations of the same cryptographic algorithm
3. hardware platforms from the point of view of their suitability for the implementation of a given algorithm, (e.g., choice of an FPGA device or FPGA board)
4. tools and languages in terms of quality of results they generate (e.g. Verilog vs. VHDL, Synplicity Synplify Premier vs. Xilinx XST, ISE v. 13.1 vs. ISE v. 12.3)