Option Space Exploration Using Distributed Computing for Efficient Benchmarking of FPGA Cryptographic Modules

**Motivation and Background**

- **ATHENA** is an open-source benchmarking environment aimed at:
  - Automated generation of optimized results for multiple hardware platforms.
  - Support for multiple tools from various vendors.
  - Optimization strategies aimed at the best possible performance.
  - Extraction and presentation of results.
  - Seamless integration with the **ATHENA** database of results.

**Limitations of the previous version of **ATHENA**:**
- Previous heuristic algorithms used required significant amount of run time.
- Unable to utilize parallelism across computing nodes.
- Not easy to maintain.

**Proposed Environment and Improvement**

- **Parallel Execution on Multiple Computers**
- **Utilize idle resources**
- **Increase throughput of benchmarking tasks**
- **Decrease benchmarking time**
- **Optimization Space Exploration**
  - **Search more options**
  - **Decrease search time**
  - **Increase optimization end performance**
- **Usability**
  - **GUI**
  - **Monitoring and control**
  - **Benchmark configuration**

**Optimization Algorithms**

- Utilize algorithms inspired by previous research on the programming language compilers:
  - Least Effort - LE
  - Most Effort - ME
  - Batch Elimination - BE
  - Iterative Elimination - IE
  - Orthogonal Arrays - OA
- Optimize FPGA-specific algorithms introduced in previous version of **ATHENA**:
  - Frequency Search - FS
  - Placement Search - PS

**Least Effort & Most Effort**

- **Least Effort** - minimum execution time, worst results
  - Easy or naive optimization
  - Used as a baseline
  - Minimum amount of work needed to optimize
  - Almost never optimal
- **Most Effort** - maximum execution time, best results
  - Also known as Exhaustive Search
  - Guarantee optimal result
  - Least time-efficient
  - Inapplicable for more than a handful of options
  - Number of runs needed: 2^n, where n is the number of options

**Frequency Search & Placement Search**

- There are two largest driving factors in performance for cryptographic cores in Xilinx FPGAs:
  1. The desired input frequency we wish to achieve - Frequency Search (FS)
  2. A seed value for the tools to begin the placing process - Placement Search (PS)
- **Frequency Search** (FS) attempts to determine the input frequency that yields the highest performance from the design.
- **Placement Search** (PS) is a very basic search that does an exhaustive search of a subset of possible placement values then refines the search and performs a second exhaustive search on a more granular set of placement options.

**Distinguishing features**

- **Least Effort** - minimum execution time, worst results
  - Easy or naive optimization
  - Used as a baseline
  - Minimum amount of work needed to optimize
  - Almost never optimal
- **Most Effort** - maximum execution time, best results
  - Also known as Exhaustive Search
  - Guarantee optimal result
  - Least time-efficient
  - Inapplicable for more than a handful of options
  - Number of runs needed: 2^n, where n is the number of options

**Proposed Differing Search Algorithms**

- **Batch Elimination**
  - **Opt. 1** - LE
  - **Opt. 2** - PS
  - **Opt. 3** - PS
  - **Opt. 4** - PS
  - **Number of runs**: n + 2
  - **Number of run levels**: n

**Iterative Elimination**

- **Iterative Elimination** takes into account the interaction of optimization options into consideration.
- **Increases algorithm time complexity**
  - **Opt. 1** - LE
  - **Opt. 2** - PS
  - **Opt. 3** - PS
  - **Opt. 4** - PS
  - **Number of runs**: n

**Orthogonal Arrays**


**Experiments**

- **Codes**: 2 SHA-3 candidate algorithms: BLAKE and JH
- **FPGA families**: Spartan 3 and Virtex 6
- **Version of tools**: Xilinx ISE v13.1
- **Hosts**: Two eight core Linux workstations – total of 16 execute nodes
- **Optimization Target**: Throughput/Area Ratio

**Experiment 1**

- **Optimization algorithm chaining**
  - Determineability of Batch Elimination, Iterative Elimination and Orthogonal Array to optimize results
  - Use expanded 9 option set and optimization algorithms chaining
  - Determine whether further improvement can be achieved if more options and algorithms chaining are used.

**Result**

**Experiment 1 Results**

<table>
<thead>
<tr>
<th>Spartan 3</th>
<th>Above Least Effort (%)</th>
<th>Below Most Effort (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>JH</td>
<td>5.3 18.0</td>
<td>15.5 -9.8 -0.7</td>
</tr>
<tr>
<td>BLAKE</td>
<td>7.9 33.0</td>
<td>3.0 -18.9</td>
</tr>
<tr>
<td>Stein</td>
<td>3.3 5.9</td>
<td>-1.9 -12.8 -10.6</td>
</tr>
<tr>
<td>Keccek</td>
<td>1.3 10.8</td>
<td>8.5 -10.9</td>
</tr>
<tr>
<td>Average</td>
<td>3.8 16.4</td>
<td>4.7 -13.1 -2.8</td>
</tr>
<tr>
<td>Median</td>
<td>4.3 13.4</td>
<td>3.2 -11.8 -0.3</td>
</tr>
</tbody>
</table>

**Experiment 2 Results**

<table>
<thead>
<tr>
<th>Virtex 6</th>
<th>Above Least Effort (%)</th>
<th>Below Most Effort (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>JH</td>
<td>8.6 33.5</td>
<td>13.5 -4.3 0</td>
</tr>
<tr>
<td>BLAKE</td>
<td>26.4 36.4</td>
<td>26.5 -7.3 0</td>
</tr>
<tr>
<td>Stein</td>
<td>2.6 9.4</td>
<td>7.2 -11.0</td>
</tr>
<tr>
<td>Keccek</td>
<td>2.6 11.1</td>
<td>-3.7 -8.5 -5.1</td>
</tr>
<tr>
<td>Average</td>
<td>7.5 15.1</td>
<td>10.9 -7.8 -1.3</td>
</tr>
<tr>
<td>Median</td>
<td>7.1 11.4</td>
<td>10.9 -7.8 -4.9</td>
</tr>
</tbody>
</table>

**Results**

**Conclusion**

- Distributed architecture and parallelization increase throughput of benchmarking tasks.
- Parallelization extended beyond core count of a single machine
- More efficient use of resources
- Greater tool flexibility
- More heuristic search options
- Increases number of effectively searched options
- Iterative Elimination is a viable alternative to Most Effort optimization with larger option sets.
- Optimization algorithm chaining yields results that outperform previous version of **ATHENA** and Xilinx PlanAhead.

---

**Note**: The table and diagrams provide a visual representation of the experimental results and comparisons across different platforms and optimization methods. The tables include columns for various options and algorithms, with metrics such as improvement percentage (RIP) and average improvements across different implementations.

---

**References**


---

**Acknowledgements**

- ECE Department, George Mason University, Fairfax, VA 22030, U.S.A.
- Crypto Group, George Mason University, Fairfax, VA 22030, U.S.A.