Groestl Tweaks and their Effect on FPGA Results

Marcin Rogawski and Kris Gaj
George Mason University
{kgaj, mrogawsk}@gmu.edu

Abstract. In January 2011, Groestl team published tweaks to their specification of Groestl In this paper, we investigate the influence of these tweaks on the Groestl performance in hardware. The results indicate that the performance penalty in terms of the throughput to area ratio depends strongly on the architecture used. This penalty is smaller in case of architecture in which permutations P and Q are implemented using two independent units.

1. Introduction

In December 2010, NIST announced the list of five Round 3 SHA-3 candidates, including BLAKE, Groestl, JH, Keccak, and Skein. In January 2011, Groestl team published tweaks to their specification of Groestl [GKM11a, GKM11]. An algorithm described by the original Groestl specification [GKM08] has been renamed to Groestl-0, and the tweaked version of Groestl, described by the revised specification [GKM11], is from this point-on called Groestl. The proposed tweaks are aimed primarily at the increase in the algorithm resistance to cryptanalysis [GKM11a]. This increased resistance in security, typically comes together with some limited penalty in terms of performance in hardware. In this short report, we evaluate this penalty in terms of throughput, area, and throughput to area ratio for two modern high-performance FPGA families: Virtex 5 from Xilinx, and Stratix III from Altera.

2. Previous work

Groestl-0 has been implemented by several groups in FPGAs and ASICs [SHZ11]. In this report, we focus on implementations targeting FPGAs and optimized for high speed rather than low area.

High-speed implementations of Groestl typically use two major architectures. In the first architecture, reported first in [GKM08], permutations P and Q are implemented using two independent units, working in parallel. We call this architecture *parallel architecture*. In the second architecture, introduced in [TFK10], the same unit is used to implement both P and Q. This unit is composed of two pipeline stages that allow interleaving computations belonging to permutations P and Q. We call this architecture *quasi-pipeline architecture*, as it is based on the similar principles as the quasi-pipelined architectures of SHA-1 and SHA-2 reported in [DMO04, MD05]. The details of the quasi-pipelined architecture of Groestl are described in [TFK10, Section 9] and [HRG10, Section 3.8].

In Tables 1 and 2, we summarize results of high-speed implementations of Groestl-0 obtained by different groups for Virtex 5 FPGAs. At this point, the most efficient implementation of Groestl-0-256 is a parallel architecture implementation from the Groestl team [GKM08], and the most efficient implementation of Groestl-0-512 is the quasi-pipelined architecture implementation from Homsirikamol et al. [HRG10].

3. Methodology

In order to compare hardware implementations of Groestl-0 and Groestl, we have first estimated the amount of logic resources required to implement modified operations of Groestl, namely *AddRoundConstant* and *ShiftBytesWide*, in ASICs and Virtex 5 FPGAs. For the FPGA implementations, we have assumed that these operations are performed using separate CLB slices (i.e., CLB slices not used for any other operation of Groestl), and we have used the property of Virtex 5 FPGAs that each CLB slice contains four 6-input look-up tables (LUTs). We have also assumed that inversion does not require any CLB slices, as it can be very easily combined with the following operation.

All logic resources required to implement modified operations of Groestl in ASICs and FPGAs are summarized in Tables 3 and 4. Table 3 refers to the quasi-pipelined architecture, and Table 4 to the parallel architecture. These tables reveal that we should expect significantly smaller area penalty in parallel architecture compared to the quasi-pipelined architecture.

Table 1. Results of Implementations for Groestl-0-256, using Xilinx Virtex 5 FPGAs.

Source	Architecture	Impl. Details	Block Memory [#BRAMs]	Clk. Freq. [MHz]	Throughput [Mbit/s]	Area [CLB slices]	Thr/Area [(Mbit/s)/ CLB_slices]
The Groestl team [GKM08]	parallel	N/A	N/A	200.7	10276	1722	5.97
Homsirikamol et al. [HRG10]	quasi-pipelined	64-bit interface	0	323.4	7885	1597	4.94
Matsuo et al. [MKS10]	parallel	S-boxes in distributed memory	0	154.0	7885	2616	3.01
Baldwin et al. [BHH10]	parallel	ideal interface, no padding unit	0	101.3	5187	2391	2.17
Kobayashi et al. [KIM10]	parallel	S-boxes decomposed into logic	0	101.0	5171	4057	1.27
Guo et al. [GHN10]	parallel	S-box decomposed into logic	0	80.2	4106	3308	1.24
Baldwin et al. [BHH10]	parallel	32-bit interface, no padding unit	0	101.3	3242	2391	1.36
Baldwin et al. [BHH10]	parallel	32-bit interface, padding unit	0	78.1	2498	2579	0.97

Table 2. Results of Implementations for Groestl-0-512, using Xilinx Virtex 5 FPGAs.

Source	Architecture	Impl. Details	Block Memory [#BRAMs]	Clk. Freq. [MHz]	Throughput [Mbit/s]	Area [CLB slices]	Thr/Area [(Mbit/s)/ CLB_slices]
Homsirikamol et al. [HRG10]	quasi-pipelined	64-bit interface	0	292.1	10314	3138	3.29
The Groestl team [GKM08]	parallel	N/A	N/A	210.5	15395	5419	2.84
Baldwin et al. [BHH10]	parallel	ideal interface, no padding unit	0	123.4	9025	4845	1.86
Baldwin et al. [BHH10]	parallel	32-bit interface, no padding unit	0	123.4	3948	4845	0.81
Baldwin et al. [BHH10]	parallel	32-bit interface, padding unit	0	113.1	3619	4525	0.80

Table 3. Use of logic resources for the modified operations of Groestl in ASICs and Virtex 5 FPGAs in case of the *quasi-pipelined architecture*. Notation: NOT – inverter, XOR2 – two input XOR gate, MUX2 – a two-to-one multiplexer, CLB – configurable logic block.

Operation	Groestl-0	Groestl	Area penalty
256-bit variant			
AddRoundConstant	**ASICs:** 16xXOR2, 8xNOT, 16xMUX2	**ASICs:** 128xXOR2, 456xNOT, 512xMUX2	**ASICs:** 108xXOR2, 448xNOT, 496 MUX2
	FPGAs: 4 CLB slices	**FPGAs:** 128 CLB slices	**FPGAs:** 124 CLB slices
ShiftBytes	**ASICs:** routing resources	**ASICs:** 512xMUX2	**ASICs:** 512 MUX2
	FPGAs: routing resources	**FPGAs:** 128 CLB slices	**FPGAs:** 128 CLB slices
512-bit variant			
AddRoundConstant	**ASICs:** 16xXOR2, 8xNOT, 16xMUX2	**ASICs:** 256xXOR2, 904xNOT, 1024xMUX2	**ASICs:** 240xXOR2, 896xNOT, 1008 MUX2
	FPGAs: 4 CLB slices	**FPGAs:** 256 CLB slices	**FPGAs:** 252 CLB slices
ShiftBytesWide	**ASICs:** routing resources	**ASICs:** 1024xMUX2	**ASICs:** 1024 MUX2
	FPGAs: routing resources	**FPGAs:** 256 CLB slices	**FPGAs:** 256 CLB slices

Table 4. Use of logic resources for the modified operations of Groestl in ASICs and Virtex 5 FPGAs in case of the *parallel architecture*. Notation: NOT – inverter, XOR2 – two input XOR gate, MUX2 – a two-to-one multiplexer, CLB – configurable logic block.

Operation	Groestl-0	Groestl	Area penalty
256-bit variant			
AddRoundConstant	**ASICs:** 16xXOR2, 8xNOT	**ASICs:** 128xXOR2, 456xNOT	**ASICs:** 108xXOR2, 448xNOT
	FPGAs: 4 CLB slices	**FPGAs:** 32 CLB slices	**FPGAs:** 28 CLB slices
ShiftBytes	**ASICs:** routing resources	**ASICs:** routing resources	**ASICs:** none
	FPGAs: routing resources	**FPGAs:** routing resources	**FPGAs:** none
512-bit variant			
AddRoundConstant	**ASICs:** 16xXOR2, 8xNOT	**ASICs:** 256xXOR2, 904xNOT	**ASICs:** 240xXOR2, 896xNOT
	FPGAs: 4 CLB slices	**FPGAs:** 64 CLB slices	**FPGAs:** 60 CLB slices
ShiftBytesWide	**ASICs:** routing resources	**ASICs:** routing resources	**ASICs:** routing resources
	FPGAs: routing resources	**FPGAs:** routing resources	**FPGAs:** routing resources

We have followed our first order estimations with the actual implementations of the tweaked Groestl. As a starting point, we have used two GMU implementations of Groestl-0, one based on the quasi-pipelined architecture and the second based on the parallel architecture. The former of these two implementations was reported in [HRG10].

As our target platform, we have chosen two very popular FPGA families: Virtex 5 from Xilinx, and Stratix III from Altera. Both of these families are based on the same 65 nm semiconductor technology. As our tools, we have used *Xilinx ISE 12.3* and *Altera Quartus II 10.1*. All architectures have been first modeled in VHDL-93, then synthesized, placed and routed using tools of the respective vendor. Maximum clock frequencies have been determined using static timing analysis tools provided as a part of the respective software packages (*quartus_sta* for Altera and *trace* for Xilinx). The tool options were selected in such a way, that no embedded resources, such as block memories or DSP units, were used during implementation. This choice was made in order to enable the comparison of all implementations in terms of area and throughput to area ratio. The Automated Tool for Hardware Evaluation (ATHENa) [ATH11, GKA10] has been used to optimize tool options and facilitate collection of results. This tool was run in the mode called GMU_optimization_1.

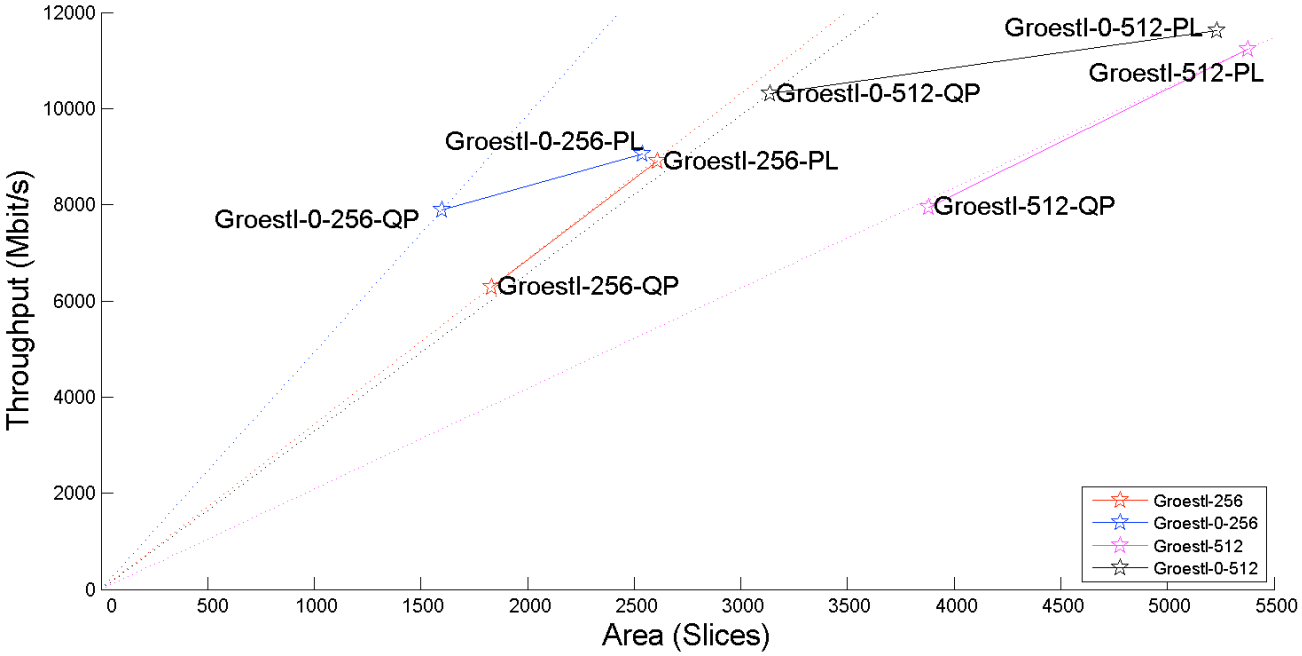

Fig. 1. Throughput vs. Area diagram for Groestl-0 and Groestl in Xilinx Virtex-5 FPGAs. Notation: PL - parallel architecture, QP - quasi-pipelined architecture.

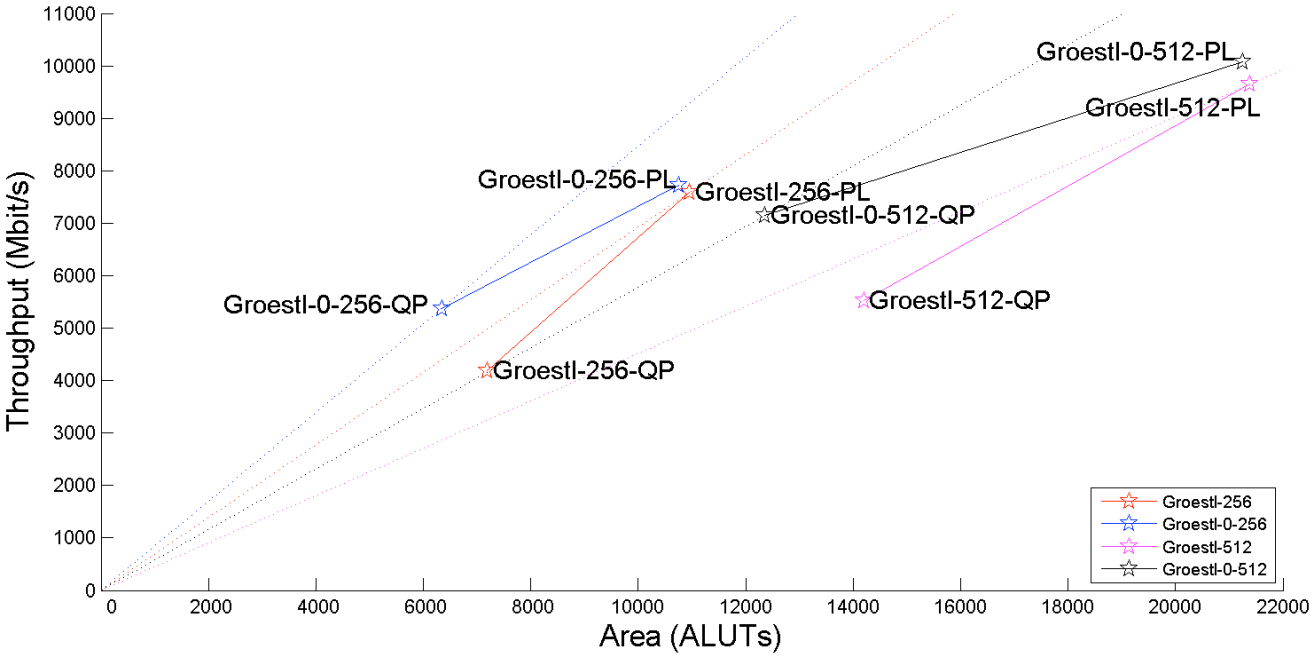

Fig. 2. Throughput vs. Area diagram for Groestl-0 and Groestl in Altera Stratix III FPGAs. Notation: PL - parallel architecture, QP - quasi-pipelined architecture.

Table 5. Implementation results for the *256-bit* variant of Groestl-0 and Groestl in case of the *quasi-pipelined architecture*.

	Groestl-0	Groestl	Percentage difference [%]
Xilinx Virtex 5			
Area (CLB slices)	1597	1831	14.7
Frequency (MHz)	323.4	258.1	-20.2
Throughput (Mbit/s)	7885	6294	-20.2
Throughput/Area ((Mbit/s)/CLB slices)	4.94	3.44	**-30.4**
Altera Stratix III			
Area (ALUTs)	6350	7189	11.7
Frequency (MHz)	220.7	171.7	-22.2
Throughput (Mbit/s)	5380	4187	-22.2
Throughput/Area ((Mbit/s)/ALUTs)	0.85	0.58	**-31.8**

Table 6. Implementation results for the *512-bit* variant of Groestl-0 and Groestl in case of the *quasi-pipelined architecture*.

	Groestl-0	Groestl	Percentage difference [%]
Xilinx Virtex 5			
Area (CLB slices)	3138	3880	23.6
Frequency (MHz)	292.1	225.2	-22.9
Throughput (Mbit/s)	10314	7953	-22.9
Throughput/Area ((Mbit/s)/CLB slices)	3.29	2.05	**-37.7**
Altera Stratix III			
Area (ALUTs)	12355	14194	14.9
Frequency (MHz)	202.3	156.4	-22.7
Throughput (Mbit/s)	7142	5523	-22.7
Throughput/Area ((Mbit/s)/ALUTs)	0.58	0.39	**-29.3**

Table 7. Implementation results for the *256-bit* variant of Groestl-0 and Groestl in case of the *parallel architecture*.

	Groestl-0	Groestl	Percentage difference [%]
Xilinx Virtex 5			
Area (CLB slices)	2539	2610	2.8
Frequency (MHz)	176.8	173.8	-1.7
Throughput (Mbit/s)	9054	8900	-1.7
Throughput/Area ((Mbit/s)/CLB slices)	3.57	3.41	**-4.5**
Altera Stratix III			
Area (ALUTs)	10749	10952	1.9
Frequency (MHz)	151.0	148.2	-1.9
Throughput (Mbit/s)	7730	7585	-1.9
Throughput/Area ((Mbit/s)/ALUTs)	0.72	0.69	**-4.2**

Table 8. Implementation results for the *512-bit* variant of Groestl-0 and Groestl in case of the *parallel architecture*.

	Groestl-0	Groestl	Percentage difference [%]
Xilinx Virtex 5			
Area (CLB slices)	5233	5379	2.7
Frequency (MHz)	159.0	153.6	-3.4
Throughput (Mbit/s)	11628	11237	-3.4
Throughput/Area ((Mbit/s)/CLB slices)	2.22	2.09	**-5.9**
Altera Stratix III			
Area (ALUTs)	21249	21379	0.6
Frequency (MHz)	137.8	131.9	-4.3
Throughput (Mbit/s)	10079	9648	-4.3
Throughput/Area ((Mbit/s)/ALUTs)	0.47	0.45	**-4.3**

4. Results

Our results for the quasi-pipelined architecture and the parallel architecture are summarized in Tables 5-8, and Figs. 1 and 2. These results clearly indicate that the differences between Groestl-0 and Groestl implementations are much greater for the quasi-pipelined architecture, in terms of both area and throughput. The performance penalty of the Groestl tweaks for the parallel architecture appears to be very small.

In terms of the choice of the architecture, for Groestl-0, the qusi-pipelined architecture gives consistently much better results in terms of the throughput to area ratio for both hash function variants, and both investigated FPGA families. For Groestl, the parallel architecture becomes very comparable to the quasi-pipelined architecture for Virtex 5, and clearly better for Stratix III.

5. Conclusions

We have performed the first order analysis of the influence of the Round 3 tweaks in Groestl on the performance of this algorithm in FPGAs. Both Groestl-0 and the revised Groestl have been fully implemented in VHDL using two alternative architectures: quasi-pipelined and parallel.

The results indicate that the performance penalty in terms of the throughput to area ratio depends strongly on the architecture used. In case of the quasi-pipelined architecture, we have observed from 29% to 38% decrease in the throughput to area ratio for Altera and Xilinx FPGAs. For the parallel architecture, we expected much smaller penalty. The obtained results are very consistent and the overall throughput to area ratio decreased by 4-6%.

Interestingly, for Groestl-0, the quasi-pipelined architecture consistently outperformed the parallel architecture in terms of the throughput to area ratio for all investigated FPGA families and output sizes. After the tweak was applied, the parallel architecture became slightly better or at least comparable in terms of the same performance measure, and may become the architecture of choice for future high-speed implementations.

References:

[ATH11] ATHENa web site, available at http://cryptography.gmu.edu/athena/

[BHH10] B. Baldwin, N. Hanley, M. Hamilton, L. Lu, A. Byrne, M. O'Neill, and W.P. Marnane, "FPGA Implementations of the Round Two SHA-3 Candidates," Second SHA-3 Candidate Conference, 2010, available online at http://csrc.nist.gov/groups/ST/hash/sha-3/Round2/Aug2010/documents/papers/BALDWIN_FPGA_SHA3.pdf

[DMO04] L. Dadda, M. Macchetti, and J. Owen, "The design of a high speed ASIC unit for the hash function SHA-256 (384, 512)," in Proc. Design, Automation and Test in Europe Conference 2004 (DATE'04), pp. 70–75.

[GHN10] X. Guo, S. Huang, L. Nazhandali, and P. Schaumont. "On The Impact of Target Technology in SHA-3 Hardware Benchmark Rankings," Cryptology ePrint Archive: Report 2010/536, available online at http://eprint.iacr.org/2010/536.pdf

[GKA10] K. Gaj, J.-P. Kaps, V. Amirineni, M. Rogawski, E. Homsirikamol, and B.Y. Brewster, ATHENa – Automated Tool for Hardware EvaluatioN: Toward Fair and Comprehensive Benchmarking of Cryptographic Hardware Using FPGAs, 20th International Conference on Field Programmable Logic and Applications - FPL 2010, IEEE, 2010, available on line at http://cryptography.gmu.edu/athena/papers/GMU_FPL_2010_ATHENa.pdf

[GKM08] P. Gauravaram, L.R. Knudsen, K. Matusiewicz, F. Mendel, Ch. Rechberger, M. Schläffer, S.S. Thomsen, "Groestl – SHA3 candidate", original specification, October 31, 2008, available on line at http://www.groestl.info/Groestl-0.pdf

[GKM11] P. Gauravaram, L.R. Knudsen, K. Matusiewicz, F. Mendel, Ch. Rechberger, M. Schläffer, S.S. Thomsen, "Groestl – SHA3 candidate", January 16, 2011, available at http://csrc.nist.gov/groups/ST/hash/sha-3/Round3/submissions_rnd3.html

[GKM11a] P. Gauravaram, L.R. Knudsen, K. Matusiewicz, F. Mendel, Ch. Rechberger, M. Schläffer, S.S. Thomsen, "Tweaks on Grostl", January 16, 2011, available online at http://www.groestl.info/Round3Mods.pdf

[HRG10] E. Homsirikamol, M. Rogawski, and K. Gaj, "Comparing Hardware Performance of Fourteen Round Two SHA-3 Candidates Using FPGAs," Cryptology ePrint Archive: Report 2010/445, available on line at http://eprint.iacr.org/2010/445.pdf

[KIM10] K. Kobayashi, J. Ikegami, S. Matsuo, K. Sakiyama, and K. Ohta, "Evaluation of Hardware Performance for the SHA-3 Candidates Using SASEBO-GII," Cryptology ePrint Archive: Report 2010/010, available online at http://eprint.iacr.org/2010/010.pdf

[MD05] M. Macchetti and L. Dadda, "Quasi-pipelined hash circuits," in Proc. IEEE Symposium on Computer Arithmetic, 2005, pp. 222–229.

[MKS10] S. Matsuo, M. Knezevic, P. Schaumont, I. Verbauwhede, A. Satoh, K. Sakiyama, and K. Ota, "How Can We Conduct "Fair and Consistent" Hardware Evaluation for SHA-3 Candidate?" Second SHA-3 Candidate Conference, 2010, available online at http://csrc.nist.gov/groups/ST/hash/sha-3/Round2/Aug2010/documents/papers/MATSUO_SHA-3_Criteria_Hardware_revised.pdf

[TFK10] S. Tillich and M. Feldhofer and M. Kirschbaum and T. Plos and J.-M. Schmidt and A. Szekely, "High-Speed Hardware Implementations of BLAKE, Blue Midnight Wish, CubeHash, ECHO, Fugue, Grostl, Hamsi, JH, Keccak, Luffa, Shabal, SHAvite-3, SIMD, and Skein," Cryptology ePrint Archive: Report 2009/510, available online at http://eprint.iacr.org/2009/510.pdf

[SHZ11] SHA-3 Zoo: Hardware Implementations, available at http://ehash.iaik.tugraz.at/wiki/SHA-3_Hardware_Implementations