Throughput and Efficiency Analysis of Unrolled Hardware Architectures for the SHA-512 Hash Algorithm

Ignacio Algredo-Badillo*, Miguel Morales-Sandoval‡, Claudia Feregrino-Uribe†, René Cumplido†
*Computer Engineering, University of Istmo, UNISTMO
Tehuantepec, Oaxaca, Mexico 70760
Email: $algredobadillo@sandunga.unistmo.edu.mx$
‡Universidad Politécnica de Victoria
Cd. Victoria, Tamaulipas, Mexico 87138
Email: $mmoraless@upv.edu.mx$
†Instituto Nacional de Astrofísica, Óptica y Electrónica, INAOE
Tonantzintla, Puebla, Mexico 72840
Email: $\{cferegrino, rcumplido\}@inaoep.mx$

***Abstract*—In order to design efficient hardware implementations of cryptographic algorithms for a particular application, it is often required to explore several architectures in order to select the one that offers the appropriate trade-off between throughput and hardware resources. A natural choice for performing a design space exploration are the Field Programmable Gate Arrays (FPGAs) for being reconfigurable, flexible and physically secure devices. In this paper we explore several architectures for implementing the SHA-512 algorithm based on the loop unrolling technique and analyze their area-performance trade-offs. The analysis consists on unrolling at different levels the main loop which is the most costly part in the SHA-512 algorithm. The resulting hardware architectures are implemented and analyzed in order to identify the critical path and make decisions on the architectural design. The obtained results provide a practical guide to understand the effect of introducing different levels (1, 2, 4, 5, 8) of unrolling in terms of throughput and hardware resources. The hardware architecture 4x that partially unrolls four iterations of the main loop of the SHA-512 algorithm reports the best performance compared against related works, while the 1x architecture exhibits the best efficiency.**

I. Introduction

Hash functions are a kind of cryptographic algorithms mainly used in digital signature algorithms to provide the integrity and authentication security services [1], [2], [3]. There are several hash algorithms currently used in security protocols for protecting digital communications, some of them are MD5, SHA-0, SHA-1 and SHA-2 [4]. Compared with other hash algorithms such as Haval [5] or Tiger [6], the family of algorithms named SHA-2 are currently accepted as a standard and recommended for practical uses [7]. In the present, there is a competition to select the next standard of hash algorithms named SHA-3 [8]. Efficient implementation of hash function algorithms has been an active research topic, being hardware implementation better preferred [5], [6], [9], [10]. In hardware, the implementation of hash functions allows to explore several techniques in order to meet specific requirements such as throughput, performance, area and power consumption. Some of these implementation techniques include loop unrolling, paralelization, segmentation, and iteration. The critical part in the SHA-2 algorithms is a main loop that executes 80 iterations to process a data blok and computes its corresponding hash value. The high data dependency in this main loop makes difficult to fully apply the parallelization and segmentation techniques to design hardware architectures for the SHA-2 family algorithms. In this work we explore the use of the partial unrolling technique for designing and implementing hardware architectures for the SHA-512 hash algorithm, the most secure algorithm in the SHA-2 family. The results and conclusions obtained on this work can be extended to other SHA-2 algorithms and to other hash functions with similar structure of the inner loop. Loop unrolling is a technique that takes advantage of parallelism in a loop at data level. Loop unrolling by a factor N executes N consecutive iterations in the loop at a time. In this work, unrolling is applied at different factors N to execute more than one iteration at a time in just one clock cycle. This implies a replication considering the data dependency. As more hardware resources are reused, interconnected and feedbacked, more input data can be processed in less time, resulting in higher throughput but, at the same time, the critical path also increases which results in a reduction of the maximum operating frequency and as a consequence a reduction of the overall performance. In order to find the best area/performance trade off we analyze in this work several hardware architectures for the SHA-512 algorithm unrolling the main round at different levels. The SHA-512 algorithm defines 80 iterations, each involving several arithmetic and logic operations. We design hardware architectures named 1x, 2x, 3x, 4x, 5x and 8x with loop unrolling factors of $N = 1$, 2, 4, 5, and 8 respectively. For example, the hardware architecture 1x, with $N = 1$, computes all the instructions inside the main loop of the SHA-512

978-0-7695-4767-1/12 $26.00 © 2012 IEEE
DOI 10.1109/ISVLSI.2012.63

TABLE I
CHARACTERISTICS OF SHA-2 FAMILY ALGORITHMS.

Algorithm	SHA-1	SHA-256	SHA-384	SHA-512
Message size	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block size	512	512	1024	1024
Word size	32	32	64	64
Message digest size	160	256	384	512
Security	80	128	192	256

algorithm in just one clock cycle. The entire loop is executed in 80 clock cycles. The hardware architecture 2x with $N = 2$ computes two iterations of the main loop in just one clock cycle, reducing the computation time of the entire main to 40 clock cycles. The area/performance trade-offs presented in this work allows to select the most appropriate implementation of the SHA-512, compromising area, throughput, efficiency, and power consumption. The hardware architecture 4x that partially unrolls four iterations from the main loop reports the best performance compared against related works, while the 1x architecture exhibits the best efficiency. The rest of this document is organized as follows: section 2 describes the SHA-512 algorithm. Section 3 explains the design and development of the proposed hardware designs. Section 4 discusses the implementation results in reconfigurable logic and provides a comparison against related works. Finally, section 5 concludes this work.

II. SHA-512 ALGORITHM

Algorithms in the SHA-2 family share a common functional structure. The main differences among them are in the size of the input data, the size of the block processed internally, the word-size, and length of the digest computed by the hash algorithm. The specific characteristics of SHA-2 algorithms are shown in table I [7].

The execution of a hash function algorithm comprises two stages: preprocessing and computing. The first stage guarantees that the length n of the input message be multiple of the given block size (see table I), allowing the input message to be divided into S exact blocks B_i ($i = 1, 2, ..., S$). In the case of the SHA-512 algorithm, B_i is of size 1024 bits. In addition, in this stage a initial value is assigned to the set of registers A, B, ..., H, that is, a initial hash value is given. In the second stage, each block B_i is processed iteratively through 80 rounds by a set of functions and mathematic operations. At each round t, a constant K_t and a word W_t are generated and used in functions that perform arithmetic and logic operations to process the incoming block message B_i. The second stage applies 80 iterations over B_i and the result is a set of values A', B', ..., H'. Using these values, an accumulative sum is performed over the set of register A, B, ..., H to finally compute an intermediate hash value of the given B_i block. The same process is applied to the next block B_{i+1}, where the intermediate hash value computed for the previous block becomes the initial hash value for computing the hash value for the current one. After all B_i blocks are processed the final hash value of the incoming message is obtained, see Figure 1.

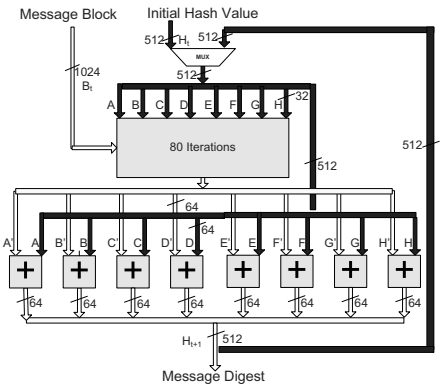

Fig. 1. Functional description for the SHA-512 algorithm.

For t from 0 to 79:
- Compute T_1 by using E, F, G, H, K_t y W_t
- Compute T_2 by using A, B y C
- Compute new H, G, F
- Compute new E = D + T_1
- Compute new D, C, B
- Compute new A = T_1 + T_2

Compute the *i-th* message hash $H^{(i)}$

Fig. 2. Pseudocode of the internal loop in the SHA-512 algorithm to compute an intermediate hash value.

The SHA-512 algorithm exhibits a high data dependency which challenges its hardware implementation and demands high spatial and temporal computational resources. Dependency is generated when lines 3-4 of algorithm shown in figure 2 are computed because most of the variables (registers) are updated based on values computed in previous iterations.

III. PROPOSED HARDWARE ARCHITECTURES

The design methodology for the proposed hardware architectures for SHA-512 algorithm is based on three phases:

1) Analyze the best way to implement the main round of the SHA-512 algorithm. In this round is located the high data dependency and the critical path could be originated in this part. This analysis is used to make decisions to reduce the critical path by reorganizing the data flow in the main round.
2) After analyzing the implementation of the SHA-512 main round, the entire loop of the SHA-512 algorithm was implemented, taking into account the decisions made in the previous step. In this phase, a 1x unrolled fully iterative architecture that computes the hash value in 80 clock cycles was designed.
3) Finally, the main loop is unrolled leading to different hardware architectures named Nx, being N the factor

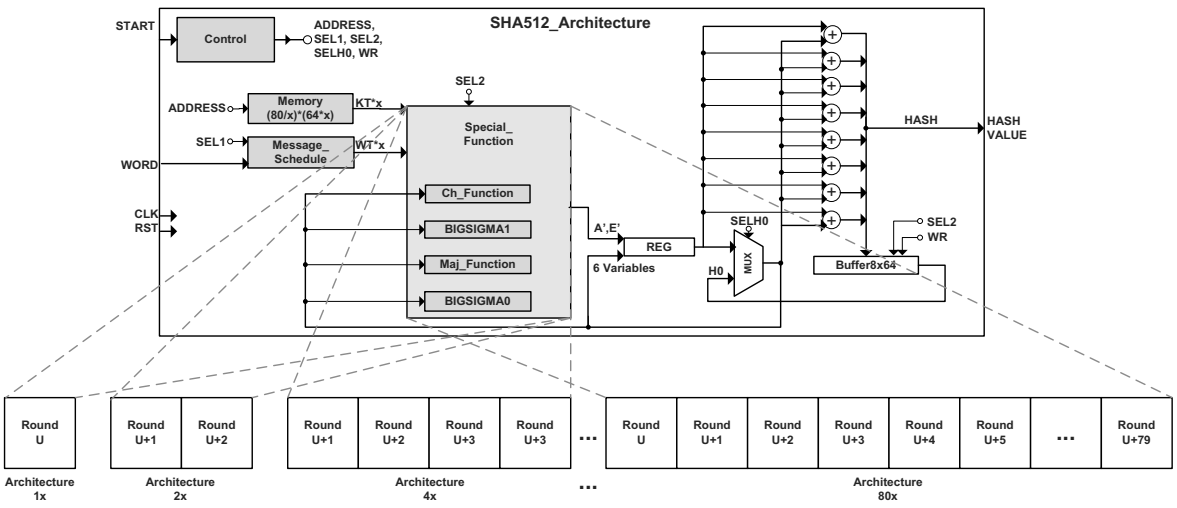

Fig. 3. Block diagram for the SHA-512 algorithm.

of unrolling. The number of combinatorial stages is incremented to compute more than one iteration at a time, taking into account the generation of the proper W_t and K_t values for a given round. What is not unrolled is the logic for feedback, which is required when the state buffer representing the intermediate hash value is updated, that is, when more than one data block B_i is being processed.

Figure 3 shows the block diagram of the proposed unrolled architectures. The general process for each unrolled architecture is:

1) store in a memory named Buffer8×64 the initial hash value (initial values for the 8 registers).
2) generate the K_t and W_t at each round
3) compute the special functions Ch, Maj, $\sum_0$, and $\sum_1$ using the initial hash value.
4) update the 8 registers represented by the buffer Buffer8×64.
5) finally, after the 80 required rounds, 8 adders are used to compute the final hash value.

As the number of functional unrolled units increases inside the hardware architecture for the SHA-512 algorithm, the unique blocks that change their size and the size of their buses are the ones located to the left of the unrolled block named "*Special_Function*". For example, in the architecture 1x (the main round is implementing using only one unrolling) one W_t and one K_t value is generate at a time, both being 64-bit long. A and E are computed and the rest of variables are updated from a redirection of the variables computed in the previous stage, see Figure 3.

On the contrary, for the 2x architecture, the internal blocks are duplicated and the size of the input buses changes while the output buses remain unchanged. In this case, two words W_t, W_{t+1} and two constants K_t, K_{t+1} are needed. Also, the double of adders and special functions are required. A block

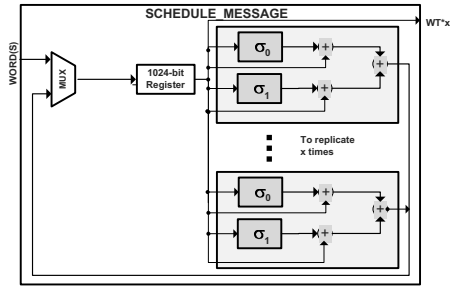

Fig. 4. Block diagram for words generation K_t in the SHA-512 algorithm.

memory of (80/2)(64*2) = 40 locations per 128-bits is needed. The rest of the blocks remain the same, i.e. the computation of A and E, the adders for computing the final hash and hence, the buses in the right side of figure 3. This same idea is applied for the architectures 4x, 5x, and 8x.

The number of processing cycles is reduced when the "*Special_Function*" module is unrolled. Other important module affecting the processing time of the SHA-512 algorithm is the "*ScheduleMessage*", that is shown in figure 4. The SHA-512 algorithm establishes that the first 16 words W_t in the algorithm are taken directly from the input block and the next 64 words are generated from the first 16 words by applying a series of special functions and arithmetic operations. In all cases, only the last previous 16 words are used to compute the next 16 words. Because of this, a memory of size 16×64 bits = 1024 bits is needed. We use a linear memory of size 1024 bits to store all the 16 W_t words.

The "*ScheduleMessage*" module generates N words W_t according to the unrolling factor N being used and the current iteration t. As we commented before, a total of 80 iterations are performed but N of them are computed in a single clock cycle. The control unit is implemented as a finite state machine

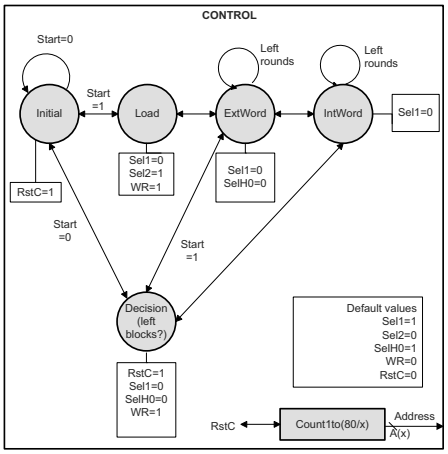

Fig. 5. State diagram for the control unit of the SHA-512 algorithm.

whose main responsibility is to orchestrate the data flow. Specifically, this module: i) multiplexes the 16 input words W_t and the 64 words computed (using signal $Sel1$), ii) selects the initial value for the variables A, B, ..., H (using signals $SElH0$ and $Sel2$), iii) updates the intermediate hash value when the number of blocks B_i is greater than 1 (using signals $Sel2$ and WR), and iv) generates the memory addresses to the memory that stores the 64 constant K_t, using a 7-bit counter controlled by the reset signal $RstC$ and the master clock. The diagram of the control unit is shown in figure 5.

IV. Results

The architecture has been modeled in the Very High Speed Integrated Circuits Hardware Description Language (VHDL) using the Active-HDL 8.2 tools and has been synthesized, placed and routed with the ISE Foundation tools from Xilinx and targeted to a Virtex-4 FPGA device. ISE tools allow to calculate hardware resources utilized and maximum clock frequency, useful parameters for measuring the Throughput (Eq. 1) [11], Performance in relation to the clock frequency (Eq. 2) and Efficiency which can be described as the ratio of Performance against hardware resource requirements (Eq. 3).

$$\text{Throughput} = \frac{m \times \text{BS} \times F}{\text{Trpc} + (m \times \text{Tb})} \quad (1)$$

$$\text{Efficiency} = \frac{\text{Throughput}}{\text{Clock Frequency}} \quad (2)$$

$$\text{Efficiency} = \frac{\text{Throughput}}{\text{Number of hardware resources}} \quad (3)$$

Table II shows an increase in hardware resources usage, including LUTs and Slices, due to the internal components of each replicated round and to the round unrolled N times. In this way, the architecture requires a higher amount of resources both setting of the functional modules, and for buses routing, this produces larger paths as the interconnection of modules increases. Then, as frequency depends inversely to the critical path (largest combinational path) frequency decreases. In the first classification, we find works with an iterative process and with an unrolling of 1x, 2x and 5x, among others, see Table III. When comparing this first set of related work, it can be seen that the range of throughputs reported is large, from some tens of megabits per second (Mbps) to over 1 Gbps. In general, loop unrolling implementations proposed in this work increase the throughput with low frequencies and higher cost of hardware resources, but combining these results, it is shown that it is possibel to obtain inprovement in efficiency. Particularly the 8x implementation of this work reaches maximum throughput of 1.616 Gbps. Comparing the 2x unrolling architecture that slightly overcomes the throughput reported in [12], the proposed solution requires a lower frequency at the cost of nearly half the hardware resources. Considering the required hardware resources and comparing the 1x implementation against the compact architecture reported in [13], the later architecture requires 9% more resources, although it reports 4.5% of the throughput and a frequency over 8 times higher.

TABLE II
Implementation results of the SHA-512 algorithm using unrolling factors $N = 1, 2, 4, 5, 8$.

Version	Results	Performance
1x	12.344 ns	1.024 Gbps
	81.011 MHz	12.64 bps/Hz
81cc	4277 LUTs (2%)	0.239 Mbps/LUT
	2667 Slices (2%)	0.384 Mbps/Slice
2x	17.937 ns	1.392 Gbps
	55.751 MHz	24.97 bps/Hz
41cc	6582 LUTs (4%)	0.211 Mbps/LUT
	4422 Slices (6%)	0.314 Mbps/Slice
4x	30.169 ns	1.616Gbps
	33.147 MHz	48.76 bps/Hz
21cc	7408 LUTs (4%)	0.154 Mbps/LUT
	5164 Slices (5%)	0.313 Mbps/Slice
5x	39.607 ns	1.520 Gbps
	25.248 MHz	60.23 bps/Hz
17cc	10474 LUTs (5%)	0.145 Mbps/LUT
	7054 Slices (7%)	0.215 Mbps/Slice
8x	70.033 ns	1.329 Gbps
	14.279 MHz	93.09 bps/Hz
11cc	11255 LUTs (6%)	0.118 Mbps/LUT
	6988 Slices (7%)	0.190 Mbps/Slice

From the implementation results shown in table II we analyze how area and throughput affect the performance and efficiency of the proposed hardware architectures.

As it can be observed in figure 6, there is a limit in the performance as the factor unrolling increases. According to these results, being the 4x architecture the best performer further research is motivated to explore optimizations of unrolled architectures for N near to 4. On the contrary, the efficiency is reduced as the unrolling factor N increases, see figure 7. This is due to two main reasons: 1) all the instructions inside the critical loop of SHA-512 are computed in a single

TABLE III
A COMPARISON OF HARDWARE ARCHITECTURES FOR THE SHA-512 ALGORITHM.

Work	Hw resources	Impl. results
[13]	271 MHz 251 Slices	46 Mbps
[14] 1x	74 MHz 2384 CLBs 81 ciclos de reloj	467 Mbps
[15] 1x	69 MHz 2545 Slices 82 ciclos de reloj	442 Mbps 0.164 Mbps/Slice
[12] 1x	106.65 MHz 2073 CLBs	1365 Mbps
[16] 5x	72 MHz 3517 Slices	1034 Mbps
[17] 1x	80.21 MHz 3213 Slices 80 ciclos de reloj	1026.68 Mbps
[18] 2x	53 MHz 2385 Slices 42 ciclos de reloj	1292 Mbps
Proposed 1x	81.01 MHz 2667 Slices 81 ciclos de reloj	1024 Mbps 0.384 Mbps/Slice
Proposed 2x	55.75 MHz 4422 Slices 41 ciclos de reloj	1392 Gbps 0.314 Mbps/Slice
Proposed 4x	33.147 MHz 5164 Slices 21 ciclos de reloj	1616 Mbps 0.154 Mbps/Slice

clock cycle, that is, the replicated logic when applying the unrolling is only combinatorial logic which increases the critical path, 2) additionally to the more area resources for the unrolling, it is necessary more area resources for routing and interconnecting all the involved buses. The results shown in figures 6 and 7 allow establishing area/performance trade offs when deciding what level of parallelism could be applied in the implementation of SHA-512 algorithm based on specific requirements of performance and efficiency. As the hardware architectures unrolls more iterations of the critical loop in the SHA-512 algorithm they gets less inefficient, with low clock frequencies which in some cases is a requirement for an application still achieving an acceptable performance. In this example, the low clock frequency could lead to hardware solutions with low power consumption.

V. CONCLUSION

Design techniques such as loop unrolling may improve architecture performance; however, there is a limit in the unrolling level where it can contribute to this improvement, besides, it does not exist proportionality in improving. Furthermore, loop unrolling allows high performance architectures

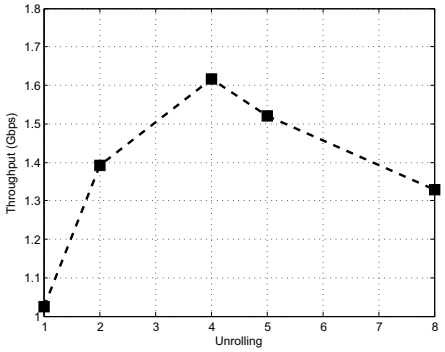

Fig. 6. Performance of SHA-512 algorithm implemented at different unrolling factors N.

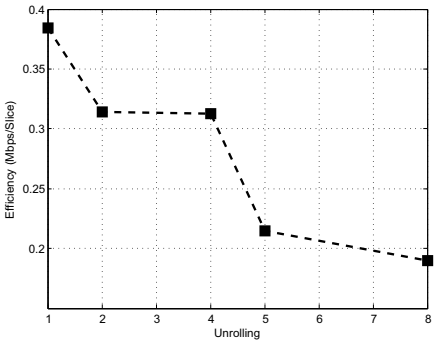

Fig. 7. Efficiency of SHA-512 algorithm implemented at different unrolling factors N.

with high data dependency as is the case of the cryptographic algorithm SHA-512. In this way, performance is limited when the architecture is designed to process a considerable amount of data because the tools for place and routing require more hardware elements as the unrolling level increases, creating so large paths. Also, the limit in performance is due to high data dependency, feedback and the necessity of processing an entire block to be able to start processing the next standardized block for the SHA-512 algorithm. Throughput relates indirectly with the critical path generated when placing and routing the hardware elements, which establishes the maximum operation frequency. New improvements must be done to shorten the critical path mainly in the high dependency, without neglecting combinational paths from signals coming from control units. Unrolled implementations proposed in this work exhibit a high throughput at resources expenses but with a low clock frequency that benefits in lower power consumption. These implementations show that performance can not improve proportionally to the level of unrolling; it may even diminish from 5x unrolling. However, hardware resources and critical path increase proportionally to the level of unrolling. As the unrolling level increases two issues are observed: i) there is a point where performance start diminishing while hardware resources and efficiency per area diminishes, and ii) efficiency

per frequency increases since there is a bigger processing capacity per clock cycle, independently of the device. Finally, a power consumption analysis would be interesting to know how it changes as the hardware resources increases and the clock frequency diminishes. Besides, it would be important to work the segmentation technique to perform new analysis.

REFERENCES

[1] American Bankers Association, "ANSI X9.62-1998: Public key cryptography for the financial services industry: The elliptic curve digital signature algorithm (ECDSA)," 1998.

[2] W. Stallings, *Cryptography and Network Security*. NJ: Prentice Hall, 2003.

[3] N. Ferguson and B. Schneier, *Practical Cryptography*. Indianapolis, Indiana: Wiley, 2003.

[4] N. Sklavos and X. Zhang, *Wireless Security and Cryptography: Specifications and Implementations*, 1st ed. Boca Raton, FL, USA: CRC Press, Inc., 2007.

[5] N. Sklavos and C. Efstathiou, "On the FPGA implementation of HAVAL hash functions," in *Proceedings of The IEEE Region 8 EUROCON 2007, International Conference on "Computer as a Tool"*. Serbia & Montenegro: IEEE, 2005, pp. 21–24.

[6] A. Satoh and N. Sklavos, "Compact and high-speed hardware architectures for hash function Tiger," in *ISCAS'09*, 2009, pp. 1401–1404.

[7] *Secure hash standard*. Washington: National Institute of Standards and Technology, 2002, URL: `http://csrc.nist.gov/publications/fips/`. Note: Federal Information Processing Standard 180-2.

[8] X. Guo, S. Huang, L. Nazhandali, and P. Schaumont, "Fair and Comprehensive Performance Evaluation of 14 Second Round SHA-3 ASIC Implementations," in *The Second SHA-3 Candidate Conference*, August 2010.

[9] T. Wollinger, J. Guajardo, and C. Paar, "Security on FPGAs: State-of-the-art implementations and attacks," *ACM Trans. Embed. Comput. Syst.*, vol. 3, no. 3, pp. 534–574, Aug. 2004. [Online]. Available: http://doi.acm.org/10.1145/1015047.1015052

[10] J. Pelzl, " Cryptanalysis with a cost-optimized FPGA cluster ," in *UCLA IPAM Workshop IV Special Purpose Hardware for Cryptography: Attacks and Applications*, December 2006.

[11] E. Khan, M. W. El-Kharashi, F. Gebali, and M. Abd-El-Barr, in *3rd International Conference on Information and Communications Technology*, 2005, pp. 861–874.

[12] P. Zalewski, M. Lukowiak, and S. Radziszowski, "Case study on FPGA performance of parallel hash functions," *Przeglad Elektrotechniczny/Electrical Review*, pp. 151–155, 2010.

[13] H. Technology, "Efficient Tiny Hash Core Family for Xilinx FPGA, Datasheet, Helion Technology Limited," 2010.

[14] N. Sklavos and O. G. Koufopavlou, "Implementation of the SHA-2 hash family standard using FPGAs." *The Journal of Supercomputing*, vol. 31, no. 3, pp. 227–248, 2005. [Online]. Available: http://dblp.uni-trier.de/db/journals/tjs/tjs31.html#SklavosK05

[15] R. Glabb, L. Imbert, G. Jullien, A. Tisserand, and N. Veyrat-Charvillon, "Multi-mode operator for SHA-2 hash functions," *J. Syst. Archit.*, vol. 53, no. 2-3, pp. 127–138, Feb. 2007. [Online]. Available: http://dx.doi.org/10.1016/j.sysarc.2006.09.006

[16] R. Lien, T. Grembowski, and K. Gaj, "A 1 Gbit/s partially unrolled architecture of hash functions SHA-1 and SHA-512." in *CT-RSA*, ser. Lecture Notes in Computer Science, T. Okamoto, Ed., vol. 2964. Springer, 2004, pp. 324–338. [Online]. Available: http://dblp.uni-trier.de/db/conf/ctrsa/ctrsa2004.html#LienGG04

[17] A. P. Kakarountas and H. Michail, "Implementation of a cryptographic co-processor," in *Proceedings of the 6th WSEAS international conference on Information security and privacy*, ser. ISP'07. Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS), 2007, pp. 160–165. [Online]. Available: http://dl.acm.org/citation.cfm?id=1981242.1981270

[18] M. Zeghid, B. Bouallegue, M. Machhout, A. baganne, and R. Tourki, "Architectural design features of a programmable high throughput reconfigurable SHA-2 processor," vol. 2, 2008, pp. 147–158.