ULTRA HIGH SPEED SHA-256 HASHING CRYPTOGRAPHIC MODULE FOR IPSEC HARDWARE/SOFTWARE CODESIGN

Harris Michail[1, 2, 3], George Athanasiou[1], Angeliki Kritikakou[1],Costas Goutis[1]
[1]*VLSI Laboratory, Departement of Electrical & Computer Engineering, University of Patras, Rio Campus, Patras, Greece*
[2]*Departement of Computer Engineering and Informatics, University of Patras, Rio Campus, Patras, Greece*
[3] *Departement of Mechanical Engineering, Technological and Educational Institute of Patras, Patras, Greece*

Andreas Gregoriades, Vicky Papadopoulou
Departement of Computer Science and Engineering, European University of Cyprus, Nicosia, Cyprus

Keywords: Hash-Functions, Hardware design, VLSI, High-Throughput, IPSec, SHA-256

Abstract: Nowadays, more than ever, security is considered to be critical issue for all electronic transactions. This is the reason why security services like those described in IPSec are mandatory to IPV6 which will be adopted as the new IP standard the next years. Moreover the need for security services in every data packet that is transmitted via IPv6, illustrates the need for designing security products able to achieve higher throughput rates for the incorporated security schemes. In this paper such a design is presented which manages to increase throughput of SHA-256 hash function enabling efficient software/hardware co-design.

1 INTRODUCTION

Security is now considered as a must-have service for almost all kind of e-applications. This is the reason why in IPv6, which is bound to be adopted worldwide, IPSec (SP800-77, 2005) is a mandatory protocol. IPSec (Internet Protocol Security) is a protocol suite for securing Internet Protocol (IP) communications by authenticating and encrypting each IP packet of a data stream. IPSec can be used to protect data flows between a pair of hosts (e.g. computer users or servers), between a pair of security gateways (e.g. routers or firewalls), or between a security gateway and a host.

In IPSec and in other applications like keyed-hash message authentication codes (HMACs) (FIPS 198-1, 2007) and the 802.16 standard for Local and Metropolitan Area Networks incorporate authenticating services, an authenticating module that includes a hash function is nested in the implementation of the application.

However, in these specific applications there is an urgent need to increase their throughput, especially of the corresponding server of these applications and this is why, as time goes by, many leading companies improve their implementations of hash functions. Although software encryption is becoming more prevalent today, hardware is the embodiment of choice for military and many commercial applications (Schneier, 1996). The security scheme of these throughput-demanding applications like HMAC in IPSEC and SSL\TLS incorporate encryption and authenticating modules.

The latter mentioned facts were strong motivation to propose a hardware design and implementation applicable to SHA-256 (FIPS 180-2, 2002) hash function which will dominate in the near future.

The above mentioned SHA-256 module is able to be efficiently embedded in a design and mapping of IPSec components in a reconfigurable platform. This way, a platform aiming to boost performance of IPSec with low cost will be created, in which only the critical kernels/components of IPSec will be mapped for execution on the (expensive) reconfigurable logic. This way the proposed design is ideal for IPSec software/hardware co-design aiming to achieve high throughput data rates.

Michail H., Athanasiou G., Kritikakou A., Goutis C., Gregoriades A. and Papadopoulou V. (2010).
ULTRA HIGH SPEED SHA-256 HASHING CRYPTOGRAPHIC MODULE FOR IPSEC HARDWARE/SOFTWARE CODESIGN.
In *Proceedings of the International Conference on Security and Cryptography*, pages 309-313
DOI: 10.5220/0002991403090313

2 PROPOSED DESIGN

The generic architecture of a hash function is shown in Fig. 1. Due to the blocks' logic variation from round to round numerous implementations (Diez, 2002), (Sklavos, 2005 - Lee, 2006), are based on four pipeline stages of single operation blocks. Also from a heuristic survey (Sklavos, 2005) to hash functions it is clear enough that the best compromise is to apply four pipeline stages so as to quadruple throughput and keep the hash core small as well. This selection was made in the presented optimization process as it is shown in Fig.1.

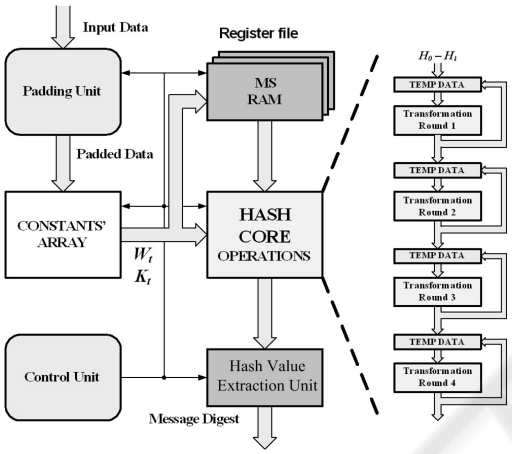

Figure 1: SHA-256 hash core architecture with 4 pipeline stages.

Exploring the generic architecture of Fig. 1 it is easily extracted that the critical path is located between the pipeline stages. The other units, MS RAM and the array of constants, do not contribute due to their nature (memory and hardwired logic respectively), while control unit is a block containing very small counters which also don't contribute to the overall maximum delay. Thus, optimization of the critical path should be solely focused on the operation block.

The operation block of SHA-256 is shown in Fig.2. The critical path (darker line) is located on the computation of a_t and e_t values that requires four addition stages and a multiplexer for feeding back the output data.

At the first step of our optimization process, a number of operations are partially unrolled. That number is determined by a separate analysis on SHA-256 hash function.

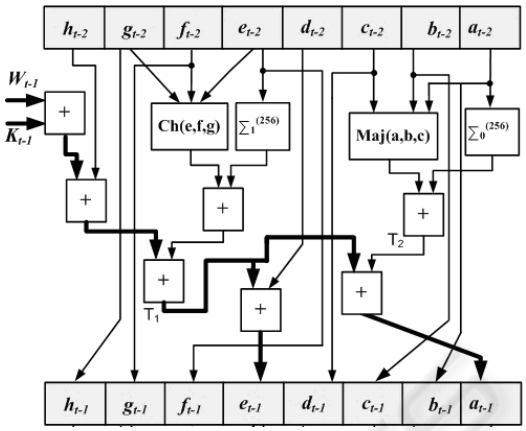

Figure 2: SHA-256 operation block.

This analysis compares variations of partially unrolled operations, their corresponding throughput, the required area and then calculating the proper ratio (cost function). In Fig. 3, the results of a cost function analysis for SHA-256 algorithm, performed in Virtex-II FPGA family, are illustrated. As it is shown, selecting to partially unroll two operations results in the best achieved Throughput/Area ratio (ratio > 2).

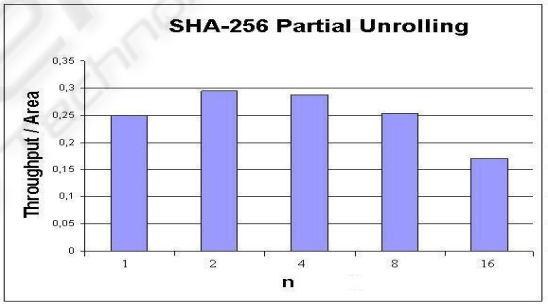

Figure 3: Effect of unrolling the operation blocks of SHA-256.

In Fig. 4, the consecutive SHA-256 operation blocks of Fig. 2, have been modified so as to exploit parallel calculations. The gray marked areas on Fig. 4 indicate the parts of the proposed SHA-256 operation block that operate in parallel.

It is noticed that two single addition levels have been introduced to the critical path that now consists of six addition stages needed for the computation of a_t and e_t values. Although, this reduces the maximum operation frequency, the throughput is increased significantly since the message digest is now computed in only 32 clock cycles (instead of 64). The area requirements are increased since more adders have been used in order to achieve the partial unrolling.

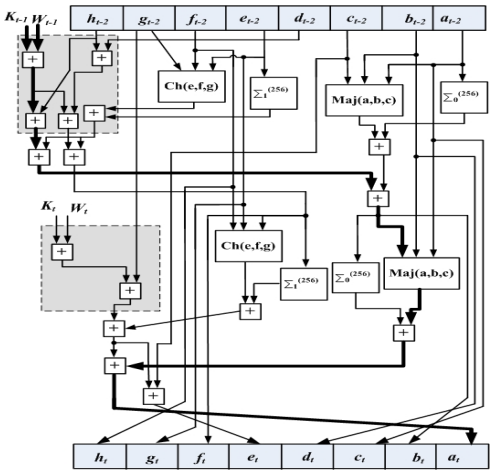

Figure 4: Two unrolled SHA-256 operation blocks.

The next step of the optimization process for our design has to do with the spatial pre-computation technique. Taking into consideration the fact that some outputs are derived directly from some inputs values respectively we can assume that it is possible during one operation to pre-calculate some intermediate values that will be used in the next operation. These pre-calculations are related only with those output values that derive directly from the latter mentioned input values. This pre-computation technique is applied on the partially unrolled operation block in Fig. 4 and the new modified operation block is shown in Fig. 5.

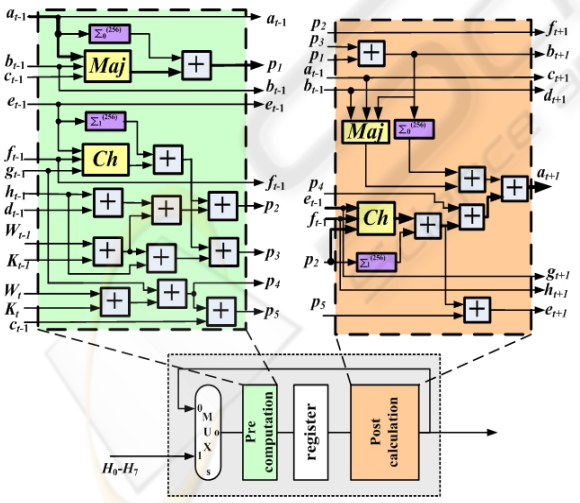

Figure 5: Partially unrolled operation block with pre-computed values.

Observing Fig. 5 it is noticed that the critical path is now located on the computation of the peripheral value p_1 that is introduced in Fig. 5. The critical path has been reduced from six addition stages and a multiplexer to four addition stages, two non-linear functions (noted as Maj and Ch in Fig. 5) and a multiplexer. Comparing to the conventional implementation of the single operation block shown in Fig. 2, theoretically throughput has been increased by 80%-90%.

This has been achieved by pre-calculating some intermediate values and moving the pipeline registers to an appropriate intermediate point to store them. The new operation block now consists of two units, the "Pre-Computation" unit which is responsible for the pre-computation of the values that are needed in the next operation and the "Post-Computation" unit which is responsible for the final computations of each operation.

The third step of our optimization process is a technique for applying the system-level pre-computation, so as to achieve data pre-fetching. It was noticed that all W_t values can be computed and be available for adequate time before they are really needed in each operation t since they are computed through some XOR bitwise operations. Also the values of the constants K_t are known a priori. These two facts give us the potential of pre-computing the sum $W_t + K_t$ outside of the operation block. The sum is then saved into a register that feeds the operation block and thus the externally (regarding the operational block) pre-computed sum $W_t + K_t$ is available at the beginning of each tth operation. So at the operational block, from now on it will be assumed that this sum available at the beginning of each operation and its computational time is excluded from the critical path. The new operational block is illustrated in Fig. 6.

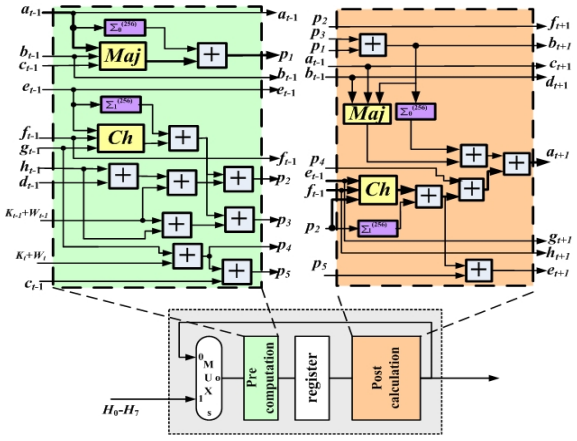

Figure 6: Partially unrolled operation block with pre-computed values for SHA-256 with pre-fetching of W+K values.

Inspecting Fig. 6, we observe that the critical path is located on the computation of the peripheral value p_1, and consists of four addition stages and two non-linear functions. However we notice that at the beginning of this path there is the value p_4 that is pending to be added to a sum that at the same time is being calculated.

So for this case, a CSA can be used in order to add the three values in advance compared to the necessary time in case we used two adders as in Fig.6. The Carry Save Adder is applied on the "Post-Computation" unit as it is depicted in Fig.7 where we have also used a Carry Save Adder in the "Pre-Computation" unit. This way the critical path inside the operation block has been reduced to one Addition stage, two Non-linear functions and two Carry Save Adders that are required in order to compute the value p_1.

The final proposed operation block for SHA-256 is illustrated in Fig. 7. It processes two operations in a single clock cycle, and the critical path is shorter than that of the conventional implementation, resulting in an increase of through-put of more than 110% (theoretical). The introduced area penalty is 3 adders, 4 Carry Save Adders, two 32-bit registers and 2 non-linear functions. The introduced area penalty is about 35% for the whole SHA-256 core compared to the conventional pipelined implementation. This corresponds to an area penalty of about 9% for the whole security scheme. This area penalty is worth paying for about 110% increase of throughput.

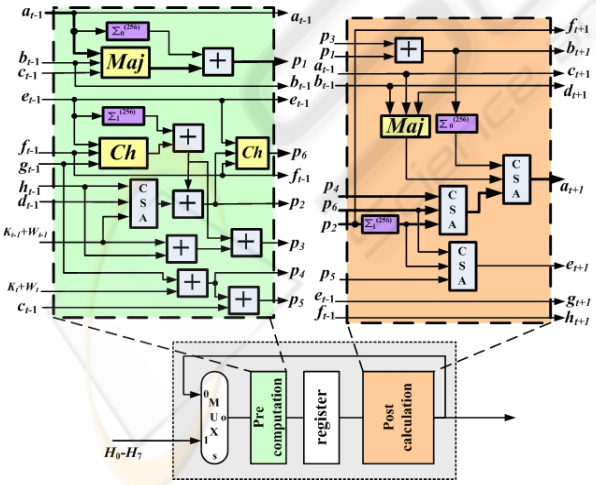

Figure 7: Proposed SHA-256 operation block.

3 RESULTS AND COMPARISONS

In order to evaluate the proposed optimized design, SHA-256 hash function was captured in VHDL and was fully simulated and verified. The XILINX FPGA technologies were selected as the targeted technologies, synthesizing the designs for the Virtex FPGA family. To exhibit the benefits of applying our optimization process, SHA-256 hash function was implemented following the steps of this optimization process and is compared with other existing implementations proposed either by academia or industry.

Table 1: Performance Characteristics and comparisons.

SHA-256				
Implementation	Op. Freq. (MHz)	Throughput (Mbps)		
		Post-synthesis	Post Place & Route	Area CLBs
(Ting et al, 2002) [a]	88.0	87	-	1261
(Sklavos et al, 2007) [a]	83	326	-	1060
(Chaves et al, 2006) [a]	82	646	-	653
(Glabb et al, 2008) [a]	77	308	-	1480
(Zeghid et al, 2007) [a]	53	848	-	2530
Proposed [a]	**35.1**	**2210**	**2077**	**1534**
(Chaves et al, 2006) [a]	150	1184	-	797
(McEvoy et al, 2005) [b]	133	1009	-	1373
(Zeghida et al, 2007) [a]	81	1296	-	1938
Proposed [c]	**36.4**	**2330**	**2190**	**1655**
(CAST Inc) [d] *(Commercial IP)*	96	-	756	945
(Helion Inc) [d] *(Commercial IP)*	-	-	1900	1614 (LUTs)
(Cadence Inc) d *(Commercial IP)*	133	-	971	asic

[a] *Virtex FPGA family* [c] *Virtex-E FPGA family*
[d] *Virtex 4 FPGA family*

The results from the latter implementations are shown in Table 1, for a variety of FPGA families. There are reported both post-synthesis and post-place & route results. The reported operating

frequencies for the proposed implementations are related to the corresponding post-synthesis results

As it can be easily seen, the increase observed for SHA-256, is about 110% gain in throughput and 30% area penalty compared to a non-optimized implementation with four pipeline stages (implemented in the same technology).

This way the improvement that arises from our optimization process is confirmed and evaluated fairly, verifying the theoretical analysis in the previous section. Furthermore, comparing the implementations of other researchers to our implementations, it can be observed that all of them fall short in throughput, in a range that varies from 0.75 – 26.4 times less than the proposed implementation.

4 CONCLUSIONS

In this paper a design for SHA-256 was proposed, which achieves high throughput with a small area penalty, thus enabling efficient software/hardware co-design. The optimization process used is generic and can be exploited to a wide range of existing hash functions that are currently used or will be deployed in the future and call for high throughputs.

This optimization process led to a design with significant increase of throughput (about 110% for SHA-256), compared to corresponding conventional designs and implementations, with a small area penalty. The results derived from their implementation in FPGA technologies confirm the theoretical results of the proposed design and implementation.

ACKNOWLEDGEMENTS

This work was supported by action "Young Researchers from Abroad" which is funded by the Cypriot state-Research Promotion Foundation (RPF/IPE).

REFERENCES

Chaves, R. and Kuzmanov, G.K. and Sousa, L. A. and Vassiliadis, S. (2006) *"Improving SHA-2 Hardware Implementations"*, Workshop on Cryptographic Hardware and Embedded Systems (CHES 2006), pp. 298-310.

Cadence, *"Hashing Algorithm Generator SHA-256: Technical Data Sheet"*, Web page available at http://www.cadence.com/datasheets/SHA256_Datasheet.pdf.

CAST Inc., Web page, available at http://www.cast-inc.com/cores.

FIPS 180-2, (2002) *"Secure Hash Standard"*, FIPS Publication 180-1, NIST, US Dept of Commerce.

FIPS 198-1, *"The Keyed-Hash Message Authentication Code (HMAC)"*, FIPS Publication 180-1, NIST, US Dept of Commerce, 2007.

Glabb, R. And Imbertb, L. and Julliena, G. and Tisserandb, A. and Charvillon, N.V. (2007) *"Multi-mode operator for SHA-2 hash functions"*, Journal of Systems Architecture, Elsevier Publishing, vol. 53, is. 2-3B, pp. 127–138.

Helion Technology Ltd, Data Security Products, Web page, available at http://www.heliontech.com/auth.htm.

Lien, R. And Grembowski, T. And Gaj, K. (2004) *"A 1 Gbit/s Partially Unrolled Architecture of Hash Functions SHA-1 and SHA-512"*, in LNCS, vol. 2964, pp. 324-338, Springer.

McEvoy, R.P. and Crowe, F.M. and Murphy, C.C. and William, P. (2006) *"Optimisation of the SHA-2 Family of Hash Functions on FPGAs"*, Emerging VLSI Technologies and Architectures (ISVLSI'06), pp.317-322.

Schneier, B. (1996). *"Applied Cryptography – Protocols, Algorithms and Source Code in C"* , Second Edition, John Wiley and Sons.

Selimis, G. and Sklavos, N. and Koufopavlou, O. (2003) *"VLSI Implementation of the Keyed-Hash Message Authentication Code for the Wireless Application Protocol"*, in ICECS'03, pp.24-27.

Sklavos, N. and Koufopavlou, O. (2005) *"Implementation of the SHA-2 Hash Family Standard Using FPGAs"*, Journal of Supercomputing, Kluwer Academic Publishers, vol. 31, pp. 227-248.

SP800-77, *"Guide to IPSec VPN's"*, NIST, US Dept of Commerce, 2005.

Ting, K. K. and Yuen, S. C. L. and Lee, K.-H. and Leong, P. H. W. (2002)*"An FPGA based SHA-256 processor"*, Lecture Notes in Computer Science (LNCS), vol. 2438, pp. 577–585. Springer.

Zeghid, M. and Bouallegue, B. and Bagagne, A. Machhoot, M. and Tourki, R. (2007) "*A Reconfigurable Implementation of the new Hash Algorithm"*, Availability, Reliavility and Security, (ARES 2007), pp.281-285.

Zeghid, M. and Bouallegue, B. and Machhoot, M. and Bagagne, A. and Tourki, R. (2008) *"Architectural Design Features of a Programmable Hgh Throughput Reconfigurable SHA-256 Processor"*, Journal of Information Assurance and Security, pp.147-158.