Application Of Novel Techniques In RIPEMD-160 Hash Function Aiming At High-Throughput

H. Michail,V. Thanasoulis, D. Schinianakis, G. Panagiotakopoulos and C. Goutis
Electrical and Computer Engineering Department
University of Patras
Gr-26500 Patra, GREECE
E-mail: michail@ece.upatras.gr, vthanasouli@upnet.gr, dimmugr@yahoo.gr, gpanagiotak@upnet.gr, goutis.ece.upatras.gr

KEYWORDS
Security, hash functions, RIPEMD-160, hardware implementation, high-throughput.

ABSTRACT

Hash functions, form a special family of cryptographic algorithms that address the requirements for security, confidentiality and validity for several applications in technology. Many applications like PKI, IPSec, DSA, MAC's need the requirements mentioned before. All the previous applications incorporate hash functions and address, as time passes, to more and more users-clients and thus the increase of their throughput is necessary. In this paper we propose an implementation that increases throughput and operating frequency significantly and at the same time keeps the area small enough for the hash function RIPEMD-160. The deployed technique involves the application of spatial and temporal pre-computation to the conventional operation block. The proposed implementation leads to an implementation that achieves 35% higher throughput.

INTRODUCTION

Nowadays many applications like the Public Key Infrastructure (PKI) (Entrust Technologies 1999), IPSec (National Institute of Standards and Technology 2005), and the 802.16 (Johnston and Walker 2004) standard for Local and Metropolitan Area Networks incorporate authenticating services. All these applications pre-suppose that an authenticating module, that includes a hash function, is nested in the implementation of the application. Moreover, digital signature algorithms like DSA (National Institute of Standards and Technology 1994) used for authenticating services like electronic mail, electronic funds transfer, electronic data interchange, software distribution, data storage etc are based on using critical cryptographic primitives like block ciphers and hash functions . Hashes are used also in identifying files on peer-to-peer filesharing networks, for example in an ed2k link. Furthermore, hashing cores are also essential for security in networks and mobile services, as in SSL. They are also the main modules that exist in the HMAC implementations that produce Message Authentication Codes (MAC's) (National Institute of Standards and Technology 1995).

Taking into consideration the rapid evolution of the communication standards that include message authenticity, integrity verification and non-repudiation of the sender's identity, it is obvious that hash functions are widely used in most popular cryptographic schemes. All the aforementioned applications ,which incorporate hash functions, are being used more and more as time goes by. So, it is necessary to increase their throughput, so as to enable the cryptographic system to satisfy immediately all requests from all users-clients. In some of these cryptographic schemes the throughput of the incorporated hash functions determines the throughput of the whole security scheme.

The latter mentioned facts were a strong motivation to propose a novel technique for increasing throughput of hash functions. In this work we propose an optimizes implementation for RIPEMD-160. The proposed implementation introduces a negligible area penalty, increasing the throughput and keeping the area small enough as required by most portable communication devices.

PROPOSED IMPLEMENTATION

In Fig. 1, the general architecture of RIPEMD-160 core with pipelined structure is illustrated. It has to be mentioned that no many research work has been conducted concerning the RIPEMD algorithm since the vast majority of both academia nad industry is focused on proposing optimizations for the SHA hash family which is the most widely adopted.

From (Dobbertin, et al 1996) it can be seen that RIPEMD-160 uses two parallel processes of five rounds, with sixteen operations for each round (5 x 16 operations for the process). This lead us to the logical assumption to use five pipeline stages for each process and a single operation block for each round among with the rest necessary parts. This way not only do we achieve to increase throughput drastically but also keep the hash core small enough.

However this what most researchers do and not much effort has been made in optimization of the operational block. In our paper we propose a methodology that intends to optimize the implementation both by applying pipeline stages but also by optimizing the internal operational block so as to achieve an even shorter critical path. This way it will be achieved to

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

obtain an implementation with much higher throughput and with negligible and small area penalty which is the main objective of our work.

The critical path of the illustrated architecture is located between the pipeline stages. Thus, the optimization of the critical path is focused on the operation block. This way the increase of operating frequency can be achieved resulting to an implementation with a higher throughput. The throughput of a hash function implementation is given by the following equation:

$$Throughput = \frac{\#bits \cdot f_{operation}}{\#operations} \qquad (1)$$

where #bits is equal to the number of bits processed by the hash function, #operations corresponds to the required clock cycles between successive messages to generate each Message Digest and $f_{operation}$ indicates the maximum operating frequency of the circuit.

A message block, as provided by the padding unit, is at most 512 bits, therefore the two terms that can be manipulated in Eq.(1) is either #operations or the circuit's operating frequency, $f_{operation}$. Manipulation of the #operations is translated to the introduction of more than five pipeline stages. This is possible but it might result in area violation since extra circuitry must be inserted.

Thus, the targeted design approach should focus on increasing the operating frequency, $f_{operation}$, without introducing any significant area penalty.

Optimizing block's operating frequency

The applied technique consists of the following 2 sub-techniques:

- Spatial Pre-computation of additions contributing to the critical path

- Temporal Pre-computation of some values that are needed in following operations

Unfolding the expressions of a_t, b_t, c_t, d_t, e_t as they are described in [8], it is observed that b_{t-1}, c_{t-1} , d_{t-1} , e_{t-1} values are assigned directly to outputs c_t, d_t, e_t, a_t respectively. In Eq. (2) the expressions of a_t, b_t , c_t , d_t , e_t are defined.

$$\begin{aligned} &e_t = d_{t-1} \\ &d_t = ROL_{10}(c_{t-1}) \\ &c_t = b_{t-1} \\ &b_t = e_{t-1} + ROL_s[f_t(b_{t-1}, c_{t-1}, d_{t-1}) + a_{t-1} + X_i + K_j] \\ &a_t = e_{t-1} \end{aligned} \qquad (2)$$

where ROLx(y) represents cyclic shift (rotation) of word y to the left by x bits and ft(z, w, v) denotes the non-linear function which depends on the round being in process.

From Eq.(2), it is derived that the maximum delay is observed on the calculation of the b_t, value from a_{t-1} and value b_{t-1}, value. Obviously the critical path consists of three addition stages as it can be seen observing Fig. 2 and a multiplexer via which the values pass each time to/and feed the operation block.

A notice that one can make observing the Eq. (2) is that some outputs derive directly from some inputs values respectively. So it can be assumed that is possible during one operation to pre-calculate some intermediate values that will be used in the next operation so as to achieve concurrent calculations.

Therefore, while the main calculations are in progress, at the same time some values that are needed in the next operation can also be in progress of calculation. Furthermore, moving the pipeline stage to an appropriate intermediate point to store these intermediate calculated values, the critical path is divided resulting in a decrease of the maximum delay without paying any worth-mentioning area penalty. This way higher operating frequency is achieved and consequently higher throughput

This technique introduces the spatial pre-computation and it is used in order to reduce the critical path. From the Eq. (2) we can observe that the outputs c_t, d_t, e_t, a_t derive directly from the values b_{t-1}, c_{t-1}, d_{t-1}, e_{t-1}, respectively, and it is possible to pre-calculate some intermediate values.

Thus, Eq. (2) is transformed to generate the intermediate values a^*_{t-1}, b^*_{t-1}, c^*_{t-1}, d^*_{t-1}, e^*_{t-1} and g_{t-1} as described in Table 1.

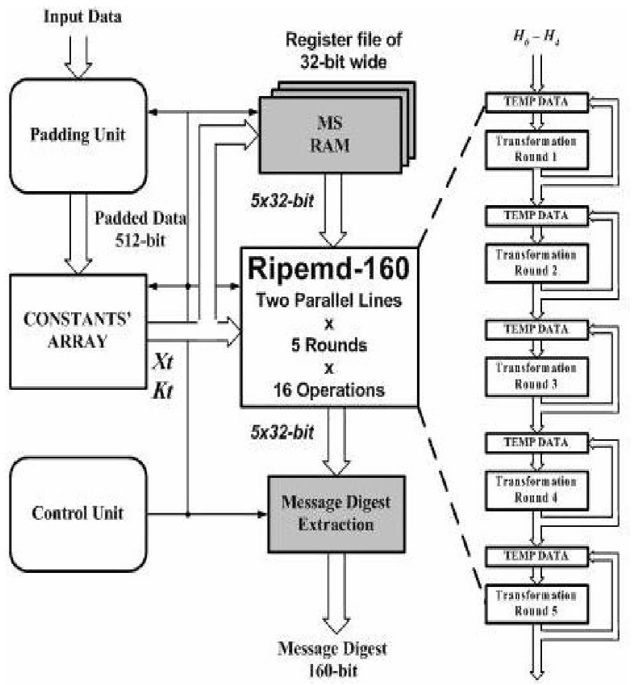


Figure 1: RIPEMD - 160 Architecture Core with Five Pipeline Stages Including a Single Operation Block

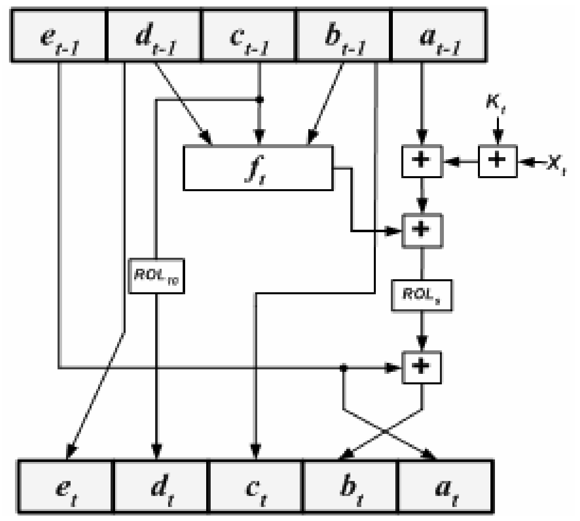


Figure 2: A Single RIPEMD - 160 Operation Block

Table 1: Expressions for Spatial Technique

Pre-Computation	Final Calculation
$e^*_{t-1} = e_{t-1}$	$e_t = d^*_{t-1}$
$d^*_{t-1} = d_{t-1}$	$d_t = ROL_{10}(c^*_{t-1})$
$c^*_{t-1} = c_{t-1}$	$c_t = b^*_{t-1}$
$b^*_{t-1} = b_{t-1}$	$b_t = e^*_{t-1} + ROL_s(g_{t-1} + a^*_{t-1})$
$a^*_{t-1} = a_{t-1}$	$a_t = e^*_{t-1}$
$g_{t-1} = f_t(b_{t-1}, c_{t-1}, d_{t-1}) + X_t + K_t$	

In Fig.3 the pre-computation technique applied in RIPEMD-160 hash function is illustrated. Each 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 "Final-Calculation" unit which is responsible for the final computations of each operation.

Notice that in Fig.3 output b_t enters the multiplexer and feeds a no-load wire b_{t-1} which stores its value to the register as b^*_{t-1}. Also notice at the "Pre-Computation" unit that the inputs a_{t-1}, c_{t-1}, d_{t-1}, e_{t-1}, which is equal with the values a^*_{t-1}, c^*_{t-1}, d^*_{t-1}, e^*_{t-1} respectively, are fed through the multiplexer from the intermediate register outputs e^*_{t-1}, b^*_{t-1}, c^*_{t-1}, d^*_{t-1} respectively.

The introduced area penalty is small, only a single register for each "round", that stores the intermediate value g_{t-1}.

Moreover, power dissipation is kept low and almost the same to that of the initial implementation as illustrated in Fig.2.In order to reduce the critical path by one addition level, we will continue with the application of the second technique, which introduces a temporal pre-computation of the values. From the "Final-Calculation" stage of Fig.3, one can observe that in every operation, from the current value of d_{t-1}, derives directly the value of e_t (at the next operation). Also, from the current value of e_t, derives directly the value of a_{t+1}. Consequently, the value of a, is the same as the value of was two operations earlier. So it is valid to write the following equation:

$$a_{t+1} = e_t = d_{t-1} \qquad (3)$$

Thus, we perform the temporal pre-computation of the sum $(X_{t+2} + K_{t+2}) + a_{t+1}$ two operations before it is used, by calculating the sum $(X_{t+2} + K_{t+2}) + d_{t-1}$ at the "Final-Calculation" unit, when the operation t is being executed. Then this sum at the "Pre-Computation" stage of the next operation (t+1) saved into the register h and represent the sum $(X_{t+2} + K_{t+2}) + e_t$.

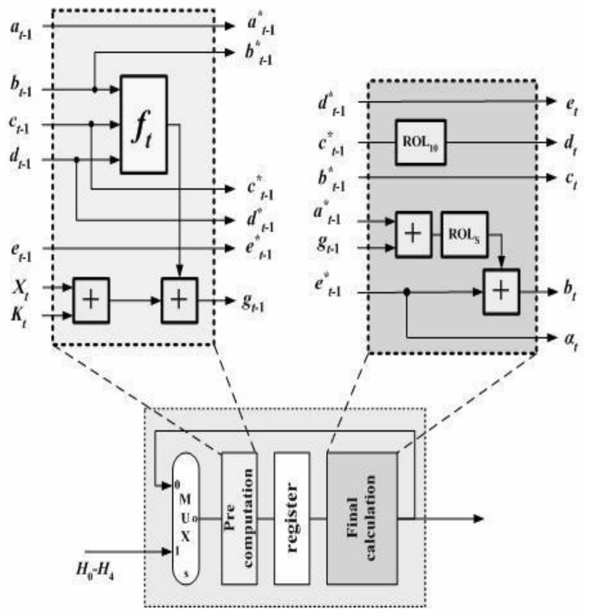


Figure 3: The Modified Operation Block of RIPEMD-160 Hashing Algorithm

At the "Final-Calculation" unit of the same operation, the value of W derives directly from the value of h. The computed sum now of the value W represents the sum $(X_{t+2} + K_{t+2}) + a_{t+1}$. Finally at the "Pre-Computation" unit on the next operation (which is the operation t+2) the sum Z= W + f_t is calculated.

The computed sum now represents the value $(X_{t+2} + K_{t+2}) + f_t + a_{t+1}$. This sum is part of the computations needed for the calculation of b_{t+2} value. What remains for the computation of the value b_{t+2} is the rotation (Rol_s) of the value Z and then its addition with the value e^*_{t+1}, as is performed in the "Final-Calculation" in Fig.4.

Observing Fig.4 it can be realized that the critical path is not located any more in the computation of the b_t value but in the computation of the value of Z. This means that the critical path in Fig. 4 has been reduced from three addition stages, a Non Linear Function f_t and

a multiplexer in Fig.3 to two addition stages, a Non Linear Function f_t and a multiplexer.
Hence, the critical path is shortened by one adder level, which contributes approximately 30% to the overall maximum delay. Moreover, it has to be noticed that an initialization of the values of W and h is needed as illustrated in Fig.4. At the first operation of every round the current values of X_t and K_t contribute for the computation of the value b_{t+2}. Thus, before the first operation begins, the value of W must be equal to the sum $(X_1 + K_1) + a_0$, which will be used at the "Final-Calculation" of the first operation for the calculation of the value b_t . Also the value of h must be equal to the sum $(X_2 + K_2) + e_0$, which will be used for the calculation of the value b_{t+1} at the "Final-Calculation stage of the second operation. Therefore, another one modification that introduces two adders is needed. However, this change does not have any effect on the critical path.

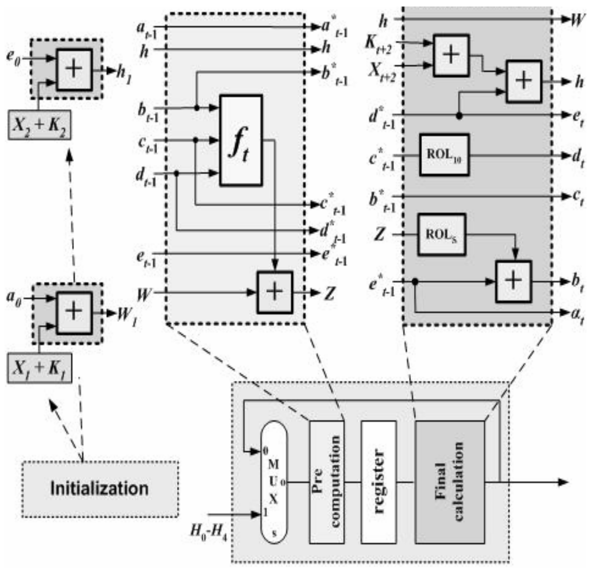


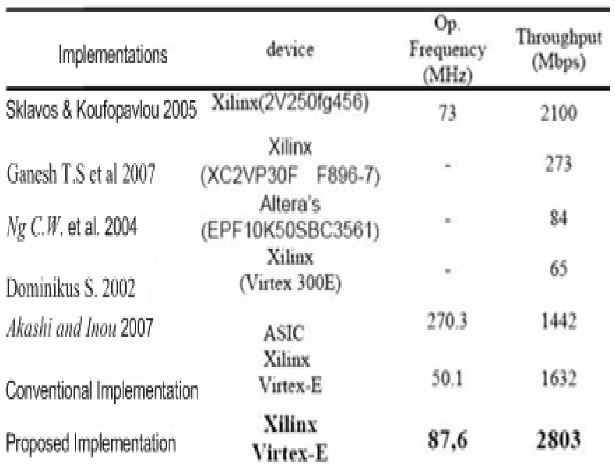
Figure 4: The Proposed Operation Block of RIPEMD-160 algorithm

The introduced area penalty is only two 32-bit registers, which are used for storing the intermediate results of the values W and h that are required. This area penalty sure enough is worth paying for an increase of throughput at about 36%.

EXPERIMENTAL RESULTS

The proposed hashing cores that were presented as examples were captured in VHDL and were fully simulated and verified using the Model Technology's ModelSim Simulator. The designs were fully verified using a large set of test vectors, apart from the test example proposed by the standards. The synthesis tool used to port VHDL to the targeted technologies was Synplicity's Synplify Pro Synthesis Tool. Simulation of the designs was also performed after synthesis, exploiting the back annotated information that was extracted from the synthesis tool. Further evaluation of the designs was performed using the prototype board for the Xilinx Virtex device family.
Probing of the FPGA's pins was done using a logic analyzer. No scaling frequency technique was followed, selecting one master clock for the system, which was driven in the FPGA from an onboard oscillator. The behavior of the implementation was verified exploiting the large capacity of the FPGA device. The achieved operating frequency is equal to 87,6 MHz.
Furthermore, as far as it concerns the introduced area overhead for the RIPEMD-160 hash core, the proposed implementation presents an increase of approximately 8%. From the experimental results, it was proved that RIPEMD-160 proposed implementation was about 36% faster than the conventional implementation. From the above results, it derives that the proposed implementation is a worth-making optimization for the hashing core since the required area for the whole security scheme is much more than that needed for the RIPEMD-160 hashing core.

Table 2: Performance Characteristics of RIPE-MD 160 Hash Function Implementations and Comparisons



Implementations	device	Op. Frequency (MHz)	Throughput (Mbps)
Sklavos & Koufopavlou 2005	Xilinx(2V250fg456)	73	2100
Ganesh T.S et al 2007	Xilinx (XC2VP30F F896-7)	-	273
Ng C.W. et al. 2004	Altera's (EPF10K50SBC3561)	-	84
Dominikus S. 2002	Xilinx (Virtex 300E)	-	65
Akashi and Inou 2007	ASIC	270.3	1442
Conventional Implementation	Xilinx Virtex-E	50.1	1632
Proposed Implementation	**Xilinx Virtex-E**	**87,6**	**2803**

It has to be added that the above comparisons concern hardware implementations mainly in FPGA boards. However due to the limited number of published work concerning RIPEMD all implementation results have been included regardless of the utilized FPGA family.
For this reason the evaluation board in each case is mentioned so as to be easy to adapt the results and make a fair comparison.
Beyond that however, the comparison results just confirm the efficiency of the proposed technique and this is accomplished with the comparison to the conventional implementation that has also been evaluated from our research time in the same FPGA board. From this comparison it can be inferred that the proposed implementation achieves its objective of higher throughput of about 35% with only 8% area

penalty for evaluation in the same FPGA board (used in our research laboratory).

CONCLUSIONS

A novel hardware implementation of RIPEMD-160 hash function was presented in this paper. Two techniques were evaluated so as to increase throughput and thus make it suitable for the corresponding server of data intensive applications. The proposed implementation has a throughput of about 2.8 Gbps, about 35% higher from the next better performing implementation. The experimental results showed that a small area penalty was introduced for a remarkable increase of throughput.

Therefore, the proposed implementation increases the throughput and frequency significantly and keeps at the same time the area small. This makes the proposed implementation suitable for server side cryptographic schemes as well as and for every new wireless and mobile communication application that urges for high-performance and small-sized solutions.

This methodology can be applied to all other hash functions such as MD-5, SHA-1, SHA-256, SHA-384, SHA-512 in order to increase their throughput.

ACKNOWLEDGEMENT

This work was supported by the project PENED 2003 No 03ED507, which is funded in 75% by the European Union- European Social fund and in 25% by the Greek state-Greek Secretariat for Research and Technology.

REFERENCES

Akashi S. and T, Inoue. 2007. “ASIC-hardware-focused comparison for hash functions MD5, RIPEMD-160, and SHS” Intergration, the Vlsi Journal 40 , 3-10.

Dobbertin H., Bosselaers A. and Preneel B. 1996. “ RIPEMD-160: A Strengthened Version of RIPEMD”.

Dominikus S. 2002. “A hardware implementation of MD4-family hash algorithms,” Proc. 9th Int. Conf. on Electronics, Circuits and Systems, vol. 3, pp. 1143-1146.

Entrust Technologies. 1999. “RFC 2510 - Internet X.509 PKI - Certificate Management Protocols”,

Ganesh T.S, Frederick M.T, Sudarshan T.S.B. and.Somani A.K. 2007. "Hashchip: A shared-resource multi-hash function processor architecture on FPGA" Intergration, the Vlsi Journal 40 11-19.

Johnston D, and Walker J. 2004. “Overview of IEEE802.16 Security” , IEEE Security and Privacy.

National Instititute of Standards and Technlogy. 1994. “FIPS 186, (DSS), Digital Signature Standard”

National Instititute of Standards and Technlogy. 1995. “FIPS 198, The Keyed-Hash Message Authentication Code (HMAC)”

National Instititute of Standards and Technlogy. 2005. SP800-77 , “Guide to IPSec VPN’s”.

Ng C.W., Ng T.S. and Yip K.W. , 2004. “A unified architecture of MD5 and RIPEMD-160 Hash algorithms”, IEEE International Symposium on Circuits and Systems.

Sklavos N. and Koufopavlou O. .2005. “On the hardware implementation of RIPEMD processor: Networking high speed hashing, up to 2 Gbps”, Computers and Electrical Engineering Journal 31 361–379.

AUTHOR BIOGRAPHIES

HARRIS MICHAIL (S’04) received a Diploma in Electrical & Computer Engineering from the University of Patras, Greece and since then he has been working towards his PhD degree, in the domain of computer security, hardware design and reconfigurable architectures.

VASSILIS THANASOULIS is an under-graduate student in the Department of Electrical .Eng, University of Patras, Greece. He is currently working on his thesis that lies in the domain of security.

DIMITRIS SCHINIANAKIS received a Diploma in Electrical & Computer Engineering from the University of Patras, Greece and since then he has been working towards his PhD degree, in the domain of of computer security, hardware design

GEORGE PANAGIOTAKOPOULOS received a Diploma in Electrical & Computer Engineering from the University of Patras, Greece and since then he has been working towards his PhD degree, in the domain of embedded computers.

COSTAS GOUTIS (S’70-M’78) received B.Sc in Physics, Diploma in Electronic Engineering, M.Sc from the University of Heriott-Watt and Ph.D from the University of Southampton. He is currently a Professor with the ECE Department, University of Patras.