RipeMD-160 Implementation Optimized in Terms of Throughput

A.P. Kakarountas, H.E. Michail, and C.E. Goutis
University of Patras, Greece
kakarountas@ieee.org, michail@ece.upatras.gr, goutis@ece.upatras.gr

ABSTRACT

Hash functions, which are a special family of cryptographic algorithms, satisfy the requirements of our days for security, confidentiality and validity for several services in technology. Many applications incorporate hash functions and address, as time passes, to more and more users-clients and thus the increase of their throughput is a primal necessity. In this paper we propose an implementation that increases the throughput of RIPEMD-160 hash function. This technique involves the application of spatial and temporal pre-computation. Comparing to conventional pipelined implementation of RIPEMD-160 hash function, the proposed technique leads to an implementation with more than 35% higher throughput.

Key-words: Message Authentication Code, CMAC, VLSI Implementation

1. Introduction

Nowadays many applications incorporate authenticating services. These applications pre-suppose that an authenticating module that includes a hash function is nested in the implementation of the application. Digital signature algorithms, used for authenticating services like electronic mail, electronic funds transfer etc, Peer-to-Peer file-sharing networks, security in networks and mobile services, and many other applications are based on using a critical cryptographic primitive like hash functions.

All the mentioned before applications which incorporate hash functions are addressing to more and more users-clients and thus it is a primal necessity the increase of their throughput particularly in order to the cryptographic system satisfy immediately all requests for service from all users-clients.

The latter mentioned facts were strong motivation to propose a novel technique for increasing throughput of hash functions. Efforts for optimization have been paid for both SHA-1 and SHA-256 [1], [2] which are the most widely deployed hash functions. In this work we propose an optimization for RIPEMD-160. The latter is an algorithm very strong in cryptanalysis which is not very widely used in antithesis to SHA family and MD-5 which are currently the most widely used hash functions. For some applications this can be considered as an advantage since not many people will try to break RIPEMD-160 hash function. The proposed implementation introduces a negligible area penalty, increasing the throughput and keeping the area small enough as required by most portable communication devices.

2. Proposed Implementation

In Fig.1, the general architecture for RIPEMD-160 core with pipelined structure is illustrated, where there are five pipeline stages and a single operation block for each round among with the rest necessary parts. The critical path of the illustrated architecture is located between the pipeline stages.

Thus, the optimization of the critical path is solely focused on the operation block in order to reduce the delay and thus increase the operating frequency. The targeted design approach focuses on increasing the operating frequency, $f_{operation}$, without introducing a significant area penalty.

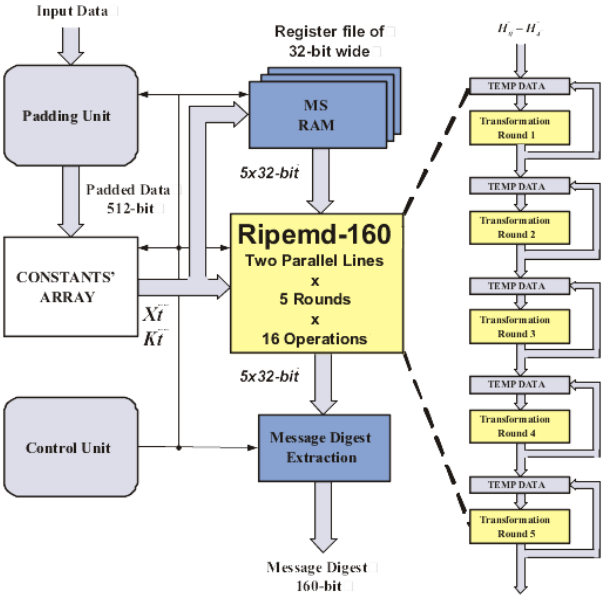

Figure 1: Ripemd-160 core architecture with 5 pipeline stages including a single operation block

3. Optimizing block's operating frequency

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

1) Spatial Pre-computation of additions contributing to the critical path.

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

Examining the expressions described in [3] and represented in Fig. 2, it is observed that some input values are assigned directly to some output values respectively. From Fig. 2, it is derived that the maximum delay is observed on the calculation of the b_t, value. Obviously the critical path consists of three addition stages and a multiplexer that feeds back the operation block as it can be seen studying Fig.2.

A notice that one can make observing the Fig. 2 is that some outputs are derived directly from some inputs values respectively. So we can assume that it is possible during one operation to pre-calculate some intermediate values that will be used in the next operation. Therefore, while the main calculations are in progress, at the same time some values that are going to be needed in the next operation can also be in progress of calculation. Furthermore, moving the pipeline stage to an appropriate intermediate point we can 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.

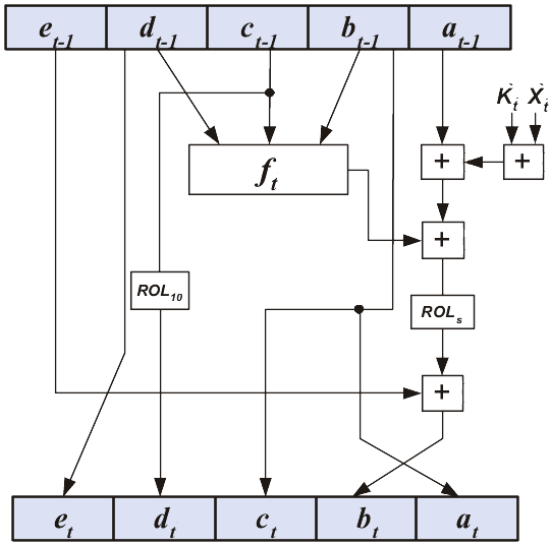

Figure 2: A single RipeMD- 160 operation block

Thus, the RIPEMD-160 equations representing Fig.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 illustrated in Fig.3. 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}. 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:

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

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 $(X_{t+2} + K_{t+2}) + e_t$. 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 $(X_{t+2} + K_{t+2}) + a_{t+1} + f_t$. 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 (Rols) of the value Z and then the addition in this result of the value e_{t+2}, as is performed in the "Final-Calculation" in Fig.4.
Observing Fig.4 we see 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 ft and a multiplexer in Fig.3, to two addition stages, a Non Linear Function f_t and multiplexer. Thus, the critical path is shortened by one adder level, which contributes approximately 30% to the overall maximum delay. Moreover, we must notice that an initialization of the values of W and h is needed as it is illustrated in Fig.4. 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 35%.

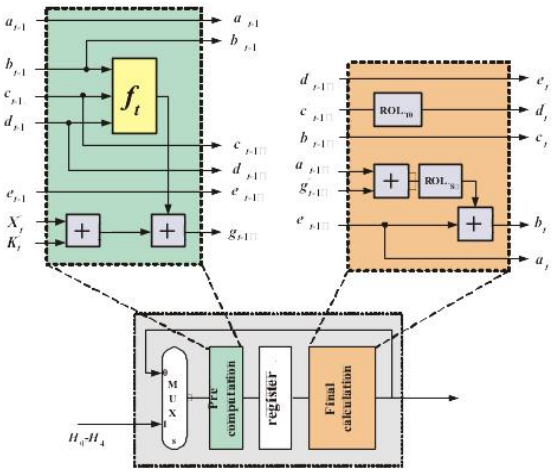

Figure 3: The parallelism block of SHA-256

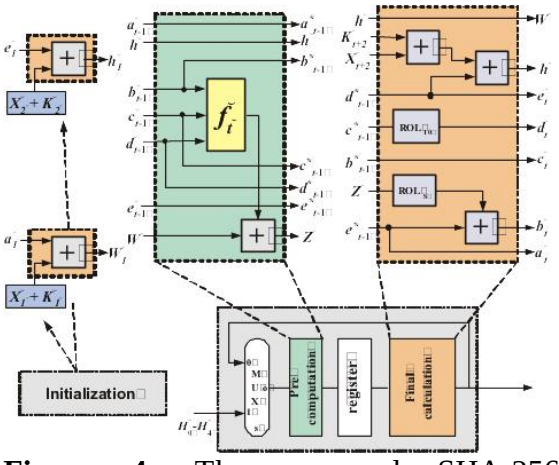

Figure 4: The proposed SHA-256 operation block

4. Experimental Results-Conclusions

The proposed hashing core that was presented was captured in VHDL and was fully simulated and verified with the ModelSim Simulator. The achieved

operating frequency is equal to 87,6 MHz. From the experimental results, it was proved that Ripemd-160 proposed implementation was about 35% faster than the conventional implementation.

5. Acknowledgment

We thank European Social Fund (ESF), Operational Program for Educational and Vocational Training II (EPEAEK II) and particularly the program PYTHAGORAS, for funding the above work. This work was also co-funded from National Funds and from the European Commission - European Social Fund (ESF), Program for Supporting Research Manpower (PENED 2003).

References:

[1] I.I. Yiakoumis, M. Papadonikolakis H.E. Michail, A.P. Kakarountas, C.E. Goutis, "Maximizing the Hash Function of Authentication Codes," IEEE Potentials, March-May 2006 pp. 2-6, 2006.

[2] H.E. Michail, A.P. Kakarountas, G. Selimis, C.E. Goutis, "Optimizing SHA-1 Hash Function for High Throughput with a Partial Unrolling Study", in Proc. of 2005 IEEE International Workshop on Power And Timing Modeling, Optimization and Simulation (PATMOS'05), pp. 591-600, Leuven, Belgium, 20-23 September, 2005.

[3] H.Dobbertin, A.Bosselaers, B.Preneel, RIPEMD-160: A Strengthened Version of RIPEMD, 18 April 1996.

[4] N. Sklavos, O. Koufopavlou, "Implementation of the SHA-2 Hash Family Standard Using FPGAs," Journal of Supercomputing, Kluwer Academic Publishers, Vol. 31, No 3, Issue: X, pp. 227-248, 2005.

[5] N. Sklavos, O. Koufopavlou, "On the Hardware Implementations of the SHA-2 (256, 384, 512) Hash Functions," proceedings of IEEE International Symposium on Circuits & Systems (ISCAS'03), Vol. V, pp. 153-156, Thailand, May 25-28, 2003.