

AN EFFICIENT IMPLEMENTATION OF THE DIGITAL SIGNATURE ALGORITHM

P. Kitsos, N. Sklavos and O. Koufopavlou

VLSI Design Laboratory
Electrical and Computer Engineering Department
University of Patras. Patras, GREECE
E-mail: pkitsos@ee.upatras.gr

ABSTRACT

Digital Signature schemes are commonly used as primitives in cryptographic protocols that provide other services including entity authentication, authenticated key transport, and authenticated key agreement. A VLSI implementation of the digital signature scheme is proposed in this paper, for efficient usage in any cryptographic protocol. This architecture is based on Secure Hash Function and the 512-bit RSA cryptographic algorithm. The whole design was captured by using VHDL language and a FPGA device was used for the hardware implementation of the architecture. A method to reduce the switching activity of the overall design is introduced. The proposed VLSI implementation of the Digital Signature scheme achieves a data throughput up to 32 Kbit/sec.

1. INTRODUCTION

Emerging applications like electronic commerce and secure communications over open networks have made clear the fundamental role of public key cryptosystem as unique security solutions. On the other hand, these solutions clearly expose the fact, that the protection of private keys is a security bottleneck in these sensitive applications. This problem is further worsened in the cases where a single and unchanged private key must be kept secret for very long time (such is the case of certification authority keys, and e-cash keys).

A Digital Signature is a checksum which depends on the time period during which it was produced. It depends on all the bits of a transmitted message, and also on a secret key, but which can be checked without knowledge of the secret key. A major difference between handwritten and digital signatures is that a digital signature cannot be a constant; it must be a function of the document that it signs. If this were not the case then a signature, could be attached to any document. Furthermore, a signature must be a function of the entire

document; changing even a single bit should produce a different signature.

A digital signature algorithm authenticates the integrity of the signed data and the identity of the signatory. A digital signature algorithm may also be used in proving to a third party that data was actually signed by the generator of the signature. Is intended for use in electronic mail, electronic data interchange, software distribution, and other applications that require data integrity assurance and data origin authentication [1]. The wireless protocols, like HiperLAN/2 [2], and WAP [3], have specified security layers and the digital signature algorithm have been applied for the authentication purposes.

In this paper an implementation of the Digital Signature Standard is proposed where a Secure Hash Function [4], and a 512-bit RSA algorithm [5] is used.

The remainder of this paper is organized as follows: in section 2 a brief description of the Digital Signature Scheme is given. In the next section the proposed architecture is presented and analysed in details. The VLSI implementation results are described in section 4 and finally some conclusions are given in the last section.

2. DIGITAL SIGNATURE ALGORITHM

A digital signature is represented in a computer as a string of binary digits. A digital signature is computed using a set of parameters and authenticates the integrity of the signed data and the identity of the signatory. An algorithm provides the capability to generate and verify signature. Signature generation makes use of a private key to generate a digital signature. Signature verification makes use of a public key, which corresponds to, but is not the same as, the private key. Each user possesses a private and public key pair. Public keys are assumed to be known to the public in general. Private keys are never shared. Anyone can verify the signature of a user by employing that user public key. Only the possessor of the user private key can perform signature generation.

A hash function is used in the signature generation process to obtain a condensed version of data, called a message digest (Figure 1). The message digest is then input to the digital signature algorithm to generate the digital signature. The digital signature is sent to the intended verifier along with the message. The verifier of the message and signature verifies the signature by using the sender's public key.

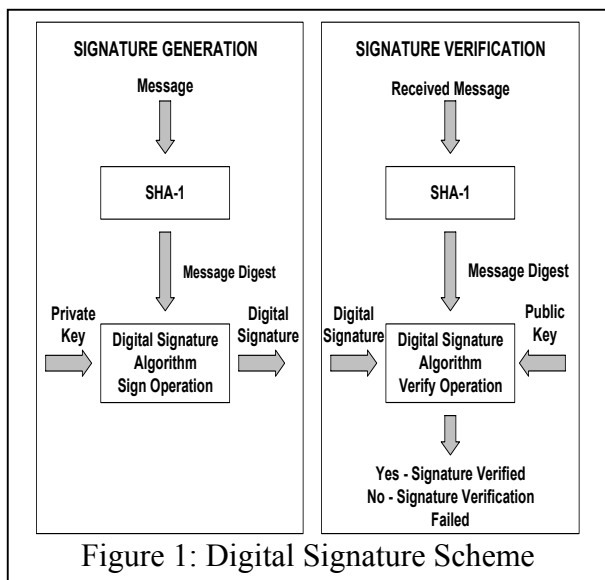


Figure 1: Digital Signature Scheme

The same hash function must also be used in the verification process. The hash function is specified in a separate standard, the Secure Hash Standard, FIPS 180-1 [4], [5]. FIPS approved digital signature algorithms must be implemented with the Secure Hash Standard. Similar procedures may be used to generate and verify signatures for stored as well as transmitted data.

The Digital Signature Standard (DSS) uses three algorithms for digital signature generation and verification [1]. The Digital Signature Algorithm (DSA), the RSA digital signature algorithm as defined in ANSI X9.31 and Elliptic Curve digital signature algorithm (ECDSA) as defined in ANSI X9.62.

3. PROPOSED ARCHITECTURE

The proposed architecture for the implementation of the digital signature scheme is shown in Figure 2. Basically it consists of two main parts, the Secure Hash Function SHA-1, and a 512-bit RSA algorithm [5]. For the RSA implementation the methodology of Blakley [6] for computing the modular exponentiation is used. This is chosen because it can achieve an appreciable decrease of covered area, and sometimes can increase the time-performance comparing with others methodologies [7].

Between of these two parts there is an intermediate (pipeline) register, which temporary hold the result of the SHA-1. The control unit generates all the appropriate control signals in order to activate all the data movement. The "Latch" signal is used in order to define

the operation start of the digital signature algorithm execution.

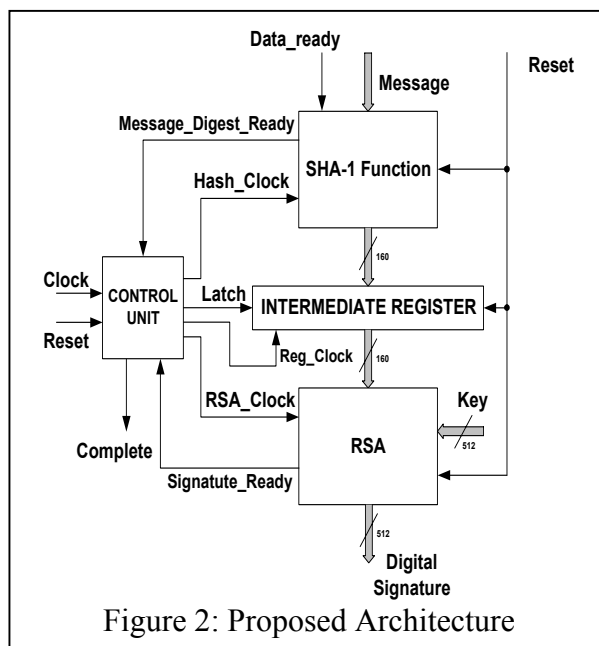


Figure 2: Proposed Architecture

3.1 SHA-1 Hash Function Architecture

The architecture for the implementation of the SHA-1 hash function is shown in details in Fig. 3. The algorithm basic round, as the specifications of the standard define, has 80 steps. In order to implement in hardware this operation a feedback loop in the basic round output is added.

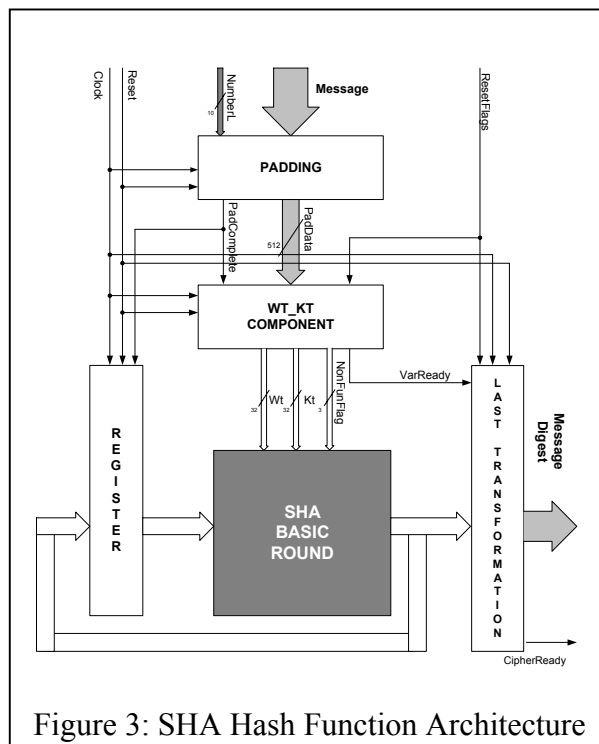


Figure 3: SHA Hash Function Architecture

Each message, after it has been appended by padding bits (PADDING component), is processed in blocks of 512 bits in length. The processing relies on four additive

constants, bit-rotations (circular left shift) and additions modulo 2^{32} . Three different primitive logical functions are used, each one for a corresponding step of the processing. Each logical function performs a set of bit-wise logical operations. It takes three 32-bit words as input and produces a 32-bit word as output (WT_KT component).

An intermediate data register (REGISTER) is used for temporary storage of the data after every transformation round execution. The last transformation unit transforms the data in final form. The output of this unit is the final message digest.

3.2 RSA Algorithm Architecture

For the RSA public-key cryptosystem implementation, the methodology of Blakley [6] is used. Blakley introduced an algorithm in order to calculate the equation, $P = A*B \text{ mod } M$. This algorithm is based on the repeats of one basic process until the desirable calculation of product P. The vector wide of A, B, M, that is 512-bit in our case, defines the equal number of the algorithms repeats procedure. Fig. 4 shows the implemented RSA algorithm architecture.

The RSA_CONFIG is used in order to store the system configuration information. It consists of two registers that are used for the storage of the intermediate plaintext and key. The control unit ensures the correct information movements between the units.

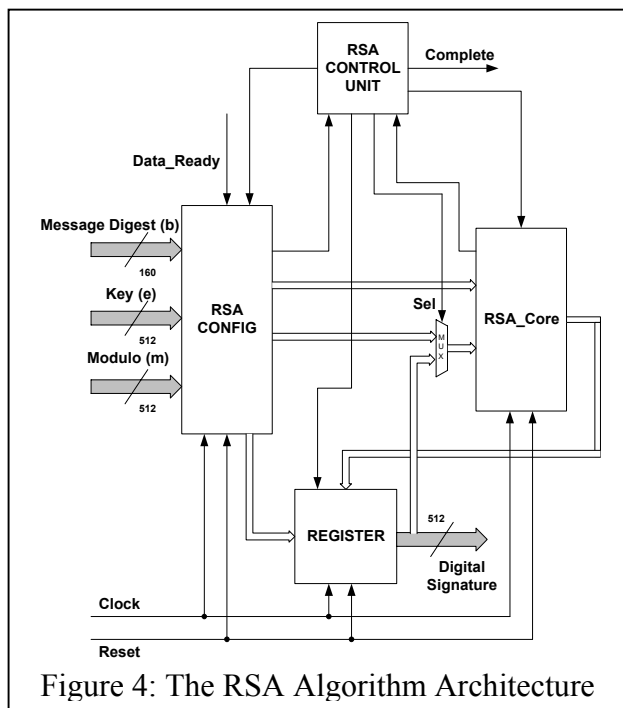


Figure 4: The RSA Algorithm Architecture

The REGISTER unit holds temporarily the results of the RSA_Core unit. In the begin it used for the initial storage of the modulo (M) value. The RSA_Core unit performs both calculations ($X^2 \text{ mod } m$), and ($X*Y \text{ mod } m$) as the left to right method defines [8]. The detailed structure of this unit is illustrated in Fig. 5.

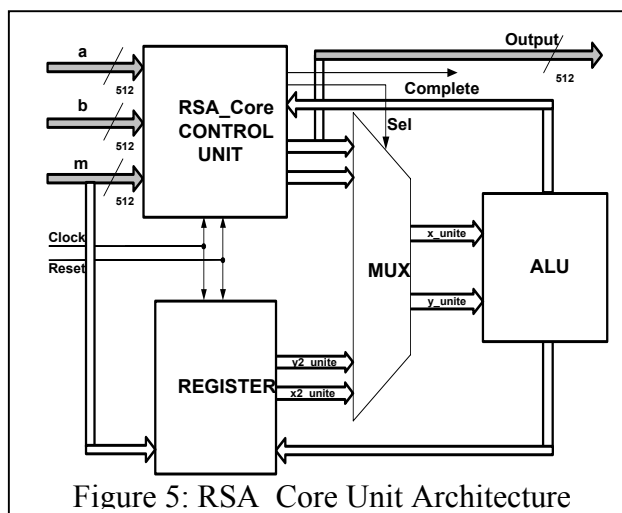


Figure 5: RSA Core Unit Architecture

The RSA_Core unit consists of four components: the control unit, a 512-bit register (intermediate register), the ALU unit and the multiplexer. The register is used to store the partial products of the Blakley algorithm execution. The multiplexer choose between the two input pairs depending of the required in ALU operation (addition or subtraction). The ALU unit performs both addition and subtraction functions.

3.3 Control Unit

The control unit is responsible for the proper operation of the whole system. It generates all the control and clock signals. Gating clock design methodology was implemented in order to reduce the total power dissipation. Only few latches were added for the glitches avoidance [9].

4. VLSI IMPLEMENTATION RESULTS

The proposed architecture was captured by using VHDL. All the system components were described with structural architecture. The whole design was synthesized, placed and routed using XILINX FPGA devices [10]. After routing the system was simulated again, for the verification of the correct functionality. The SHA-1 implementation was validated using the know-answer test vector that provided by [4]. Since conformance tests for the Digital Signature Standard using the RSA algorithm, are not available yet, test vectors were constructed, and applied to assure that the design was working as intended.

For comparison purposes the Secure Hash function (SHA-1) and RSA architectures have been implemented in FPGA devices. Synthesis results for these two implementations are shown in Tables I and II.

Table I: SHA-1 FPGA Implementation

FPGA DEVICE	Xilinx v300pq240	
AREA ALLOCATION	Used / Available	Utilization
I/Os	150 / 166	90 %
Fun. Generators	5112 / 6144	85 %
CLB Slices	2506 / 3072	85 %
Dffs or Latches	1505 / 6144	25 %
F (MHz)	47	

Table II: RSA FPGA Implementation

FPGA DEVICE	Xilinx v400bg432	
AREA ALLOCATION	Used / Available	Utilization
I/Os	300 / 316	95 %
Fun. Gener.	5528 / 9600	57.6 %
CLB Slices	3250 / 4800	67.7 %
Dffs or Latches	4186 / 9600	43.6 %
F (MHz)	40	

Comparison between the proposed RSA implementation and other previous reported implementations are presented in Table III. Area resource comparisons are not given because they are not provide by the other implementations.

Table III: RSA Implementations Comparison

RSA Architecture	F (MHz)	Data rate (Kb/s)
RSA in [11]	45.6	140-460
RSA in [12]	100	100
RSA in [13]	50	24.3
Proposed	40	35

Synthesis results for the total design implementation are shown in the table IV. The system operates with two different clocks one for each part (SHA-1 and RSA).

Table IV: Proposed Design Implementation

FPGA DEVICE	Xilinx v600bg432	
AREA ALLOCATION	Used / Available	Utilization
I/Os	200 / 316	63.3 %
Fun. Generators.	10740 / 12288	87.4 %
CLB Slices	5856 / 69120	84.7 %
Dffs or Latches	5746 / 12288	46.8 %

5. CONCLUSIONS

A VLSI implementation of the digital signature scheme is presented in this paper. The proposed design provides high-speed performance and minimized covered area. In addition due to the use of the clock-gating technique the

implementation consumes low power dissipation. For better time-performance operation the proposed system was designed by using two different clocks. It achieves a data throughput up to 32 Kbit/sec for 512-bit RSA implementation. It is a flexible solution for any cryptographic system and security layers of wireless protocol, such as HiperLAN/2 [2], and WAP [3].

REFERENCES

- [1] National Institute of Standards and Technology (NIST), Digital Signature Standard, FIPS PUB 186-2, <http://csrc.nist.gov/publications/fips/fips186-2/fips186-2.pdf>
- [2] Martin Johnsson, "HiperLAN/2 – The Broadband Radio Transmission Technology Operating in the 5 GHz Frequency Band", HiperLAN/2 Global Forum, 1999, Version 1.0 white-paper, on line available at <http://www.hiperlan2.com/technology.asp>
- [3] WAP Forum: "Wireless Application Protocol Architecture Specification" and "WAP White Paper", www.wapforum.org.
- [4] National Institute of Standards and Technology (NIST), Secure Hash Standard, FIPS PUB 180-1, www.itl.nist.gov/fipspubs/fip180-1.htm
- [5] Bruce Schneier, *Applied Cryptography – Protocols, Algorithms and Source Code in C*, Second Edition, John Wiley and Sons, New York, 1996.
- [6] G. R. Blakley, "A Computer Algorithm for Calculating the Product AB modulo M", *IEEE Transactions on Computers*, Vol. C-32, May 1983.
- [7] Kazuo Sakiyama, Sungha Kim, "An FPGA implementation and Performance Evaluation of Modular Multiplication Operation for RSA Cryptography algorithm", on line available at <http://www.cs.ucla.edu/~milos/PROJ02/KSreport.pdf>
- [8] Cetin Kaya Koc, "RSA Hardware Implementation", RSA Laboratories, RSA Data Security, Inc., on line available <http://security.ece.orst.edu/koc/ece575/rsalabs/tr-801.pdf>
- [9] Luca Benini and Giovanni De Micheli, *Dynamic Power Management. Design Techniques and CAD Tools*, Kluwer Academic Publishers, USA, 1998.
- [10] Xilinx, San Jose, California, USA, Virtex, 2.5 V Field Programmable Gate Arrays, 2001, www.xilinx.com
- [11] T. Blum and C. Paar, "High-Radix Montgomery Modular Exponentiation on Reconfigurable Hardware", *IEEE Transactions on Computers*, vol. 50, no. 7, 2001.
- [12] Chih-Yuang Su, Shih-Arn Hwang, Po-Song Chen, and Cheng-Wen Wu, "An Improved Montgomery's Algorithm for High-Speed RSA Public-Key Cryptosystem", *IEEE Transaction on Very Large Scale Integration (VLSI) Systems*, Vol. 7, No. 2, June 1999.
- [13] P. S. Chen, S. A. Hwang, and C. W. Wu, "A systolic RSA public key cryptosystem," in *Proceedings of International Symposium of Circuit and System (ISCAS'96) 1996*, vol. 4, pp. 408-411.