International Journal of Communication Networks and Security

Volume 1 | Issue 4 Article 4

April 2012

IMPLEMENTATION OF RSA KEY GENERATION BASED ON RNS USING VERILOG

VISHAK M
Dept. of Electronics and Communication Sri Jayachamarajendra College of Engineering, Mysore, INDIA, Vishak.msp@gmail.com

N. SHANKARAIAH
Dept. of Electronics and Communication Sri Jayachamarajendra College of Engineering, Mysore, INDIA, shankarsjce@gmail.com

Recommended Citation

M, VISHAK and SHANKARAIAH, N. (2012) "IMPLEMENTATION OF RSA KEY GENERATION BASED ON RNS USING VERILOG," *International Journal of Communication Networks and Security*: Vol. 1 : Iss. 4 , Article 4.
DOI: 10.47893/IJCNS.2012.1043
Available at: https://www.interscience.in/ijcns/vol1/iss4/4

IMPLEMENTATION OF RSA KEY GENERATION BASED ON RNS USING VERILOG

VISHAK M[1] & N.SHANKARAIAH[2]

[1,2]Dept. of Electronics and Communication Sri Jayachamarajendra College of Engineering, Mysore, INDIA
E-mail : Vishak.msp@gmail.com & shankarsjce@gmail.com

Abstract - RSA key generation is of great concern for implementation of RSA cryptosystem on embedded system due to its long processing latency. In this paper, a novel architecture is presented to provide high processing speed to RSA key generation for embedded platform with limited processing capacity. In order to exploit more data level parallelism, Residue Number System (RNS) is introduced to accelerate RSA key pair generation, in which these independent elements can be processed simultaneously. A cipher processor based on Transport Triggered Architecture (TTA) is proposed to realize the parallelism at the architecture level. In the meantime, division is avoided in the proposed architecture, which reduces the expense of hardware implementation remarkably. The proposed design is implemented by Verilog HDL and verified in matlab. A rate of 3 pairs per second can be achieved for 1024-bit RSA key generation at the frequency of 100 MHz.

I. INTRODUCTION

Public key cryptography has gained extreme popularity in many applications such as smart cards, digital certificate and so on. One of the most widely used public key algorithms is RSA .However, the process of RSA key generation is very complex and time consuming. Some implementations try to generate the RSA key pairs on a desktop and upload the pair, or only the private key, into a smart card. Take communication security and high-performance processing into consideration, the entire procedure of RSA key generation is preferably performed totally inside cipher chip in order to guarantee the efficiency and the security of the applications. Nevertheless, the limited computational power of embedded systems cannot afford high speed RSA key generation. In this paper, the issue about how to provide high processing speed to RSA key generation for embedded platform with limited processing capacity is discussed. An on-card implementation is presented in which takes up to 6.82 seconds to create a key pair. A scalable hardware architecture is proposed in they have presented the architecture applied for the multiple word Radix-2 Montgomery multiplication algorithm and a processing element (PE) is designed. We propose an efficient solution to accelerate RSA key pair generation from both data and architecture level parallelism, in which RNS and TTA are combined closely. The advantage of RNS Montgomery multiplication algorithm is that large number multiplications can be divided into small elements and each independent element can be processed simultaneously. The advantages of Transport Triggered Architecture (TTA) is that it can be used as an application specific processor, especially as a coprocessor for different DSP applications in SoC. Comparing with the traditional ASIC, TTA is more flexible in application and consumes less silicon area. And comparing with traditional DSP processor, it has more efficiency and better performance. In this paper, function unit (FU)"MMAC" is elaborately designed, which meets the highly parallelism of RNS Montgomery multiplication and reduces redundant data transmission. So an optimized architecture, called TTA-like, is proposed to meet the desire for high-efficient performance. In order to reduce the expense of hardware implementation, division is discarded in the process due to an appropriate selection of algorithms. This improvement makes it possible to fulfill RSA key pair generation with pure hardware design, which is quite different from prior methods that use software programming to do the division. The rest of paper is organized as follows. Section 2 describes the RSA key pair generation and details how to reduce timings related to prime finding algorithm which is the kernel of the whole process. Section 3 presents our architecture for it. The implementation results in this paper are compared finally, in section 4, the conclusions and future work are discussed.

II. DESIGN FLOW AND ALGORITHM ANALYSIS

Common RSA key pair generator generally includes a stage of trial division in the sieve function procedure. We investigate in this section a way of how to avoid this by utilizing invertible numbers which is quite suitable for the hardware implementation. Some algebraic techniques are introduced to speed up modular exponentiation, so the primality test, which is the most time-consuming section practically, can be optimized greatly to achieve a satisfactory implementation; and the private key generation algorithm is improved as well for the sake of limited computation resources in our embedded processor.

Figure 1 gives out a common procedure of the RSA key pair generation.

2.1. Sieve Function

The purpose of the sieve function is to reduce the times of the primality test which is the most time-consuming part of RSA key pair generation. Unlike ordinary sieve functions which use small primes to divide the candidate number, trial divisions is replaced by one-time modular exponentiation as shown in Algorithm 1. According to the architecture presented in this paper, modular exponentiation can be done very fast and no extra hardware consuming is needed. So the processing time of the sieve function can be trivial and circuit area will be saved as well.

2.2. Primality Test

The candidate must be tested for primality in order to be useful for the generation of a RSA key pair. In this section, Miller-Rabin's method is used for our primality test. Since Miller-Rabin test is dominant in the processing time of RSA key pair generation, it is important to improve this part for the sake of high performance. Take notice of $b \leftarrow au \bmod n$ which is the most timing-cost computation in Algorithm 2, we focus on this part and make use of RNS.

2.3. Calculation of Private Key

The private key is the modular inverse of the public key .The particular method chosen for computing modular inverse avoids trial division to make it easier to be implemented in hardware.

2.4. RNS Montgomery Modular Multiplication

Modular multiplication is the kernel operation of RSA key pair generation which is called in quantity and takes up the most time of the whole procedure, it is of great importance to analyze this part and bring out the optimal algorithm for

The hardware we present in this paper.

Algorithm 1 Invertible Number Generation

Input: randomly selected k _ = Π74

i=1 pi; _(_) =

$lcm(p1 - 1; p2 - 1; :::; p74 - 1); pi = 3; 5; 7; :::;$ 379 $L = 8 \times$ _;$M = 3 \times$ pi

Output: q co-prime with the smallest 74 odd primes

if k_(_) mod _ = 1 then

$q = k + L$

else $k = k + 1$, go to 1

end if

if q is even then

$q = q + M$

end if

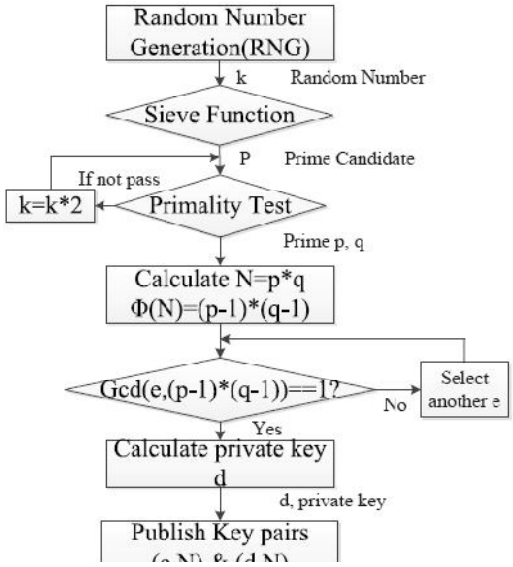

Figure 1 A common flow of RSA key pair generation.

Algorithm 2 Miller-Rabin Test

Input: Odd integer $n \geq 3$

Output: Confirm the primality of n

Find u and k so that $n - 1 = u * 2k$

Let a be randomly chosen from $2; 3; :::; n - 1$

$b \leftarrow a\ u \bmod n$

if $b \in 1; n - 1$ then

print "n is prime"

end if

for $i = 1$ to $k - 1$ do

$b \leftarrow b2 \bmod n$

if $b = n - 1$ then

print "n is prime"

end if

if $b = 1$ then

print "n is composite"

end if

end for

print "n is composite"

Algorithm 3 Improved Stein's method for modular inverse

Input: e(public key),_(n)(_(n) = $(p - 1) \times (q - 1)$,$p; q$ are two primes)

Output: $d = e$□1 mod _(n)

($X1; X2; X3$) = $(1; 0; e)$, ($Y1; Y2; Y3$) = $(0; 1; (n))$

while $X3 = 1$ or $Y3 = 1$ do

if $X3$ is even then

if $X1;X2$ are even then

$(X1;X2;X3) = (X1=2;X2=2;X3=2)$

else

$X1 = X1 + \_(n), X2 = X2 - e$

end if

else if $Y3$ is even then

if $Y1;\ Y2$ are even then

$(Y1;\ Y2;\ Y3) = (Y1=2;\ Y2=2;\ Y3=2)$

else

$Y1 = Y1 + \_(n), Y2 = Y2 - e$

end if

else if $X3 > Y3$ then

$(X1;X2;X3) = (X1 - Y1;X2 - Y2;X3 - Y3)$

else

$(Y1;\ Y2;\ Y3) = (Y1 - X1;\ Y2 - X2;\ Y3 - X3)$

end if

if $X3 = 1$ then

return $X1 = e\text{-}1 \bmod \_(n)$

else

return $Y1 = e\text{-}1 \bmod \_(n)$

Algorithm 4 RNS Montgomery Modular Multiplication

Input: $|X|a[b;\ |Y|a[b;\ (X;\ Y < 2N)$

Output: $[r]a[b;\ (r = XY A\text{-}1) \bmod N, r < 2N$

for i=1 to k do

step1: $zi = (xi \times yi) \bmod ai$

step2: $qi = (zi \times | - N\text{-}1|ai) \bmod ai$

end for

step3: $qj = BT(qi;\ 0)$

for j=1 to k do

step4: $zj = (xj \times yj) \bmod bj$

step5: $wj = (zj + qj \times Nj) \bmod bj$

step6: $rj = (wi \times |A\text{-}1$

$i\ |bj\ )\ (i = j)$

end for

step7: $ri = BT(rj\ ;\ 0x80000000)$

Algorithm 5 Base Extension $qj = BT\ (qi;\ \_)$

Input: $[q]a$

Output: $[q]b;\ qr$

for i=1 to k do

$li = qi \times |A\text{-}1$

$i\ |mi$

end for

$w = (\_ +$

Σk

$i=1\ li) >> 32$

for j=1 to k do

$qj = |$

Σk

$n=1\ |An|ai \times li|mi$

end for

return $[q]b$

III. IMPLEMENTATION FOR RSA KEY PAIR GENERATION

In this section, a TTA-like architecture is presented to illustrate the fast RSA key pair generation in our way. The implementation results are compared with in section 3.

4.1. Architecture Design

Transport Triggered Architecture (TTA) is statically programmed ILP modular architecture which is similar to VLIW architecture. Instead of specifying operation typing and controlling the FUs directly, TTAs specify the required data transports. These transports may trigger operations as side effect implicitly. Figure 2 shows the difference of instruction format between VLIW and TTA. The proposed architecture in this design is shown in Figure 3. It mainly consists of five parts: FUs, RFs, control logic, transport network and on-chip RAM/ROM. Similar to common processors, the control logic composes of instruction fetch, instruction decoder and PC control units. In this design, JMP unit can affect the PC value to realize jump, branch and loop operations.

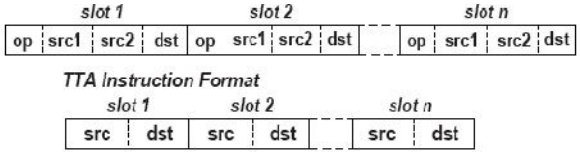

Figure 2.The difference of instruction format between VLIW and TTA

The FUs are the key factors which decide the performance of this processor. According to different applications, various of FUs can be designed and attached to the transport network. To implement RNS Montgomery multiplication algorithm, modular

multiplication and modular multiplication and-accumulation are the key operations in n-bit level, where n is the base size. So, the MMAC units are designed to accelerate the execution speed of these key operations. MMAC units can only do 32-bit× 32-bit modular multiplication, But combing them together with RNS Montgomery multiplication algorithm, long bit modular multiplication can be done very fast. In this work, four MMAC units are designed according to the maximum number of function units used in the processing of executing the operations in Figure 4 before TIME E which refers to step 1 from step 3 in algorithm 4;

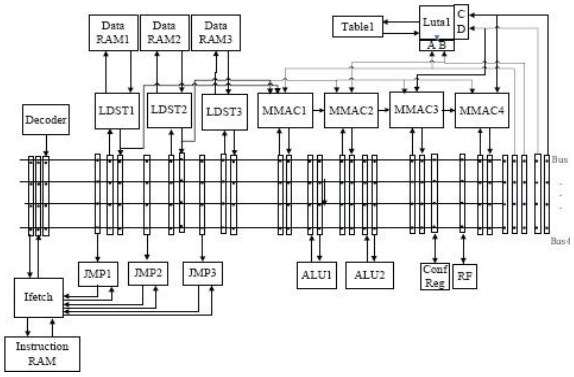

Figure 3. The proposed architecture design

The remaining operations after TIME E also depend on these four MMAC units in order to increase their usability of the architecture and save the valuable hardware resources. Two ALU units are included to implement the modular addition and modular subtraction operations. And some controlling operations are needed such as logical right-shift, arithmetic right-shift and case-select. They are included in ALU units to help finish the extra operation in the process of RSA key pair generation as well. There are three Load-store units (LDST), one Look-up Table unit (LUT). LDSTs are connected with the independent Data RAM. LUT are used to store the pre-computed data. There are direct data paths between LDSTs and MMAC units and also between LUT and the group of MMAC, which are used to reduce redundant data transmission. The transport network is used to transport data from source register to destination register. Four buses are adopted in the transport network to fully exploit the computation capacity of four MMAC units in parallel. The width of each bus is 32-bit which is decided by the selected base size.Figure 5 illustrates the operational process of RSA key pair generation from function unit level. The whole process has three stages:

1) Sieve Function

2) Primality Test

3) Private Key Calculation

According to the algorithm analysis in section 2, these function units are arranged to fulfil the RSA key pair generation: MMAC units do the modular exponentiation; ALU units complete the operation of addition and subtraction To compare the performance of our implementation with other works, the processing time consumed in primality test, prime finding and RSA key pair generation is analyzed. Because of the architecture specified for modular multiplication, the consuming time in primarily test is reduced greatly and numbers of times in prime finding are 36 which is a satisfactory compromise in both circuit area and processing time. As shown in Table 1, the proposed work requires less clock cycles than the other works.

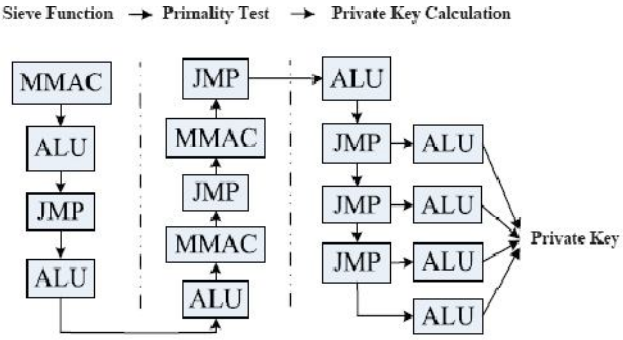

Figure 4. The operation process of RSA key pair generation from functional unit level

	Ref [3]	Ref [4]	This work
Frequency (MHz)	32	10.52	100
Primality test (ms)	84.2	25.29	4.1
prime finding (ms)	3100	/	151
RSA key pair generation (ms)	6820	/	306

Table1. RSA key pair comparison

IV. RESULT

The design was implemented by Verilog HDL and the processing time of 1024-bit RSA key pair generation is 306 ms in average, and the logic area is 131k gates. To compare the performance of our implementation with other works, the processing time consumed in primality test, prime finding and RSA key pair generation is analyzed. Because of the architecture specified for modular multiplication, the consuming time in primarily test is reduced greatly and numbers of times in prime finding are 36 which is a satisfactory compromise in both circuit area and processing time.

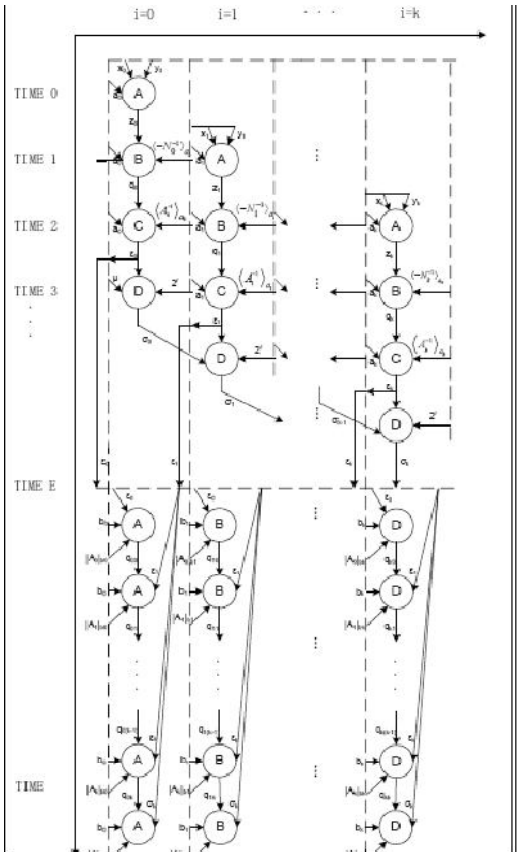

Figure 5. The operation process of MMAC unit

V. CONCLUSION AND FUTURE WORK

This paper presents a novel RSA key pair generation hardware implementation based on TTA, and Montgomery modular multiplication based on RNS is adopted in both sieve function and primality test to improve the performance significantly. FUs suiting the algorithms is designed on TTA,and direct data paths are used to reduce redundant data transmission. Above all, pipeline and parallel technology to improve the computing speed are introduced. At the frequency of 100 MHz, 1024-bit RSA key pair generation needs 306 ms in average, the logic area of the proposed architecture consists of 131k gates. This result shows that our proposed work can achieve high performance and small area for RSA key pair generation.

On-going and future developments include: (1) Preparation for some pre-computed data especially in RNS Montgomery multiplication can be optimized which affect the rate of RSA key pair generation significantly. (2) The conceptof scalable and reconfigurable architecture is introduced, in which not only 1024-bit RSA key pair but also 2048, 4096bit can be implemented in this platform.

REFERENCES

[1] R. L. Rivest, A. Shamir, L. Adleman, ”A Method for Obtaining Digital Signatures and Public-key Cryptosystems,” Communications of the ACM, vol. 21, pp. 120-126, 1978

[2] Chenghuai Lu, Andre L. M. dos Santos, Francisco R. Pimentel, ”Implementation of Fast RSA Key Generation on Smart Cards,” Proceedings of the ACM Symposium on Applied Computing, pp. 214-220, 2002

[3] Bahadori M, Mali M. R, Sarbishei O, Atarodi M, Sharifkhani M, ”A Novel Approach for Secure and Fast Generation of RSA Public and Private Keys on SmartCard,” 2010 8th IEEE International NEWCAS Conference (NEWCAS 2010),pp. 265-268, 2010

[4] Cheung Ray C. C, Brown Ashley, Luk Wayne, Cheung Peter Y K, ”A Scalable Hardware Architecture for Prime Number Validation,” Proceedings - 2004 IEEE International Conference on Field-Programmable Technology, FPT ’04, pp. 177-184, 2004

[5] Laurent Imbert, Jean-Claude Bajard, ”A Full RNS Implementation of RSA,” IEEE Transactions on Computers, vol. 53,2004, pp. 769-774.

[6] Wei Guo, Jizeng Wei, Yongbin Yao et al, ”Design of a configurable and extensible Tcore processor based on Transport Triggered Architecture,” World Congress on Computer Science and Information Engineering, pp. 536-540, 2009

[7] Hanae Nozaki, Masahiko Motoyama, Atsushi Shimbo et al,”Implementation of RSA Algorithm Based on RNS Montgomery Multiplication,” Cryptographic Hardware and Embedded Systems(CHES), pp. 364-376, 2001

[8] Kenneth R, Castleman, ”Digital image processing[M],” NewJersey:Prentice-Hall, 1996

[9] Joye M, Paillier P, Vaudenay S, ”Efficient generation of prime numbers,” Cryptographic Hardware and Embedded Systems-CHES 2000. Second International Workshop. Proceedings(Lecture Notes in Computer Science Vol. 1965), pp. 340-54,2000

[10] Maurer U. M, ”Fast generation of prime numbers and secure public-key cryptographic parameters,” Journal of Cryptology,pp. 123-55, 1995