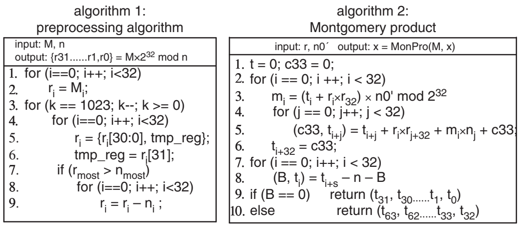
Area-efficient and ultra-low-power architecture of RSA processor for RFID

D.M. Wang, Y.Y. Ding, J. Zhang, J.G. Hu and H.Z. Tan

Presented is an area-efficient and ultra-low-power hardware architecture of a 1024-bit RSA processor using a modified Montgomery algorithm. Since RSA for RFID often offers authentication and data encryption, small area, low power and high speed are its final goal. Proposed is the following progress: 1. to improve the Montgomery algorithm including preprocessing and Montgomery multiplication; 2. to design an architecture by pipelining two parallel multiply-add units using two-port register files; 3. to provide low power design methods. The result is the lowest power architecture of an RSA processor. The design has been fabricated using SMIC 0.13 μm CMOS technology and the test results show that the proposal design is most suitable for the low power systems.

Introduction: RSA is an important public key cryptography and is widely used in cryptosystems. As an excellent digital signature and encryption algorithm, RSA offers a secure mechanism for network communication, key management etc. These applications never worry about power consumption. Yet in passive devices like RFID tag chips, mobile phones and portable devices, designers may find that RSA is hard to integrate due to its low power requirement and slow runtime. Recent research is mainly concerned about speed and throughput but less consideration is given to power and area which are very important for RFID and low power systems. Reference [1] introduced an architecture and implemented on a Xilinx FPGA; though its throughput is really high, it does not consider power consumption. There are several FPGA implementations aiming at speed such as those in [2]. A few other researchers used ASIC to realise high speed and throughput, for example, [3] designed a reconfigurable RSA with high performance as well as large area and power; and [4]'s systematic design also aims at high throughput.



algorithm 1:
preprocessing algorithm

input: M, n
output: {r31......r1,r0} = M×2^{32} mod n

```
for (i==0; i++; i<32)
      $r_i = M_i$;
for (k == 1023; k--; k >= 0)
      for (i==0; i++; i<32)
            $r_i$ = {$r_i$[30:0], tmp_reg};
            tmp_reg = $r_i$[31];
      if ($r_{most} > n_{most}$)
            for (i==0; i++; i<32)
                  $r_i = r_i - n_i$ ;
```

algorithm 2:
Montgomery product

input: r, n0´   output: x = MonPro(M, x)

```
t = 0; c33 = 0;
for (i == 0; i ++; i < 32)
      $m_i = (t_i + r_i \times r_{32}) \times n0' \bmod 2^{32}$
      for (j == 0; j++; j < 32)
            $(c33, t_{i+j}) = t_{i+j} + r_i \times r_{j+32} + m_i \times n_j + c33$;
      $t_{i+32}$ = c33;
for (i == 0; i++; i < 32)
      $(B, t_i) = t_{i+s} - n - B$
if (B == 0)     return ($t_{31}, t_{30} \ldots t_1, t_0$)
else            return ($t_{63}, t_{62} \ldots t_{33}, t_{32}$)
```

Fig. 1 *Low power algorithm improvement*

Some researchers realise the importance of power consumption. Reference [5] claimed a design that is compact and has high-performance and can be used in mobile applications with power dissipation 61.5 mW at 40 MHz. Reference [6]'s 32.5 mW power consumption at 200 MHz is even lower. However, the passive RFID tag chip acquires power supply from RF signals; in order to realise at least 10 cm read/write distance, the average power consumption must be less than 1 mA [7], and the peak power must be below 50 mW. At the same time, the total transaction time of RFID is less than 300 ms. Therefore, the power and clock cycles spent on RSA must be low enough that the other logic can work on RFID.

$$\begin{aligned} M &= M \times 2^{32} \bmod n \\ x &= 1 \times 2^{32} \bmod n \end{aligned} \tag{1}$$

Algorithm improvement: Using the Montgomery algorithm to finish the RSA operation includes two steps: first we should build two new n-residues M and x which we call preprocessing as shown in (1), then the binary exponentiation operation will be executed by performing Montgomery product. Preprocessing is time consuming and needs almost 1024 times of 1024-bit subtraction. Note that $M \times 2^{32}$ is a 2048-bit operand, and subtraction of such a large number is a hard problem. Based on radix $W = 2^w$, the k-bit operand can be broken into s blocks, each block accounts for w bit, hence a w-bit addition unit will be enough. A new low-power and high-speed preprocessing method is illustrated in algorithm 1 of Fig. 1, where $w = 32$. This algorithm is based on Blakey's method and makes the following improvement (here we only discuss the calculation of $M \times 2^{32} \bmod n$): r_i, n_i and M_i are 32-bit binary numbers, tmp_reg is a one bit binary number and is used for the temporary shifted result, $r$most and $n$most are the biggest 33-bit operand of r and n. First, M is loaded into r step by step; secondly, a shift and remainder calculation operation is performed for the 1024 times, in the inner step, the 1024-bit shift unit is divided into 32-bit units, thus reducing power consumption. For the sake of the rise of the operation time, pre-calculation is executed. Once $n$most is greater than $r$most, there is no need to subtract, while subtraction is needed, when the step by step method is also used. The final result is stored in r's lower 1024-bit register. The same operation is performed to acquire the result of the other preprocessing, but the result is stored in r's higher 1024-bit register.

Algorithm 2 of Fig. 1 shows the high radix Montgomery product method; since finishing RSA encryption or decryption needs over 1024 times Montgomery product, speed optimisation becomes more important. The key operation is the computation of the formula:

$$(c33, t_{i+j}) = t_{i+j} + r_i \times r_j + 32 + m_i \times n_j + c33 \tag{2}$$

Some researchers break it into two steps, hence these operations need at least two clock cycles. Consider that the RFID's working frequency is low (about 13.56 MHz), there would be enough time to finish two 32-bit × 32-bit parallel multipliers; thus the calculation of the above polynomial only needs one clock cycle. All temporary data is stored in the t register, and the carry is saved in the $c33$ register, which would reduce area to a certain degree. Finally if the computation result is greater than n, step by step subtraction is also needed. Therefore, we get twice the speed.

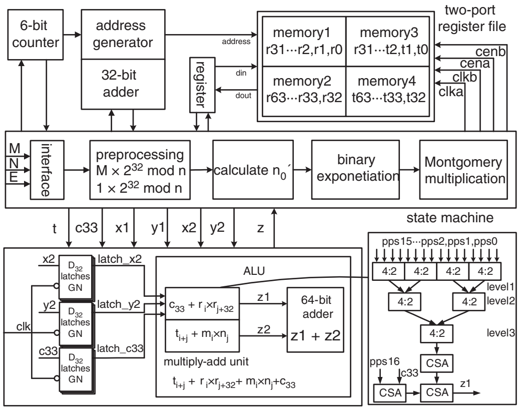


Fig. 2 *Hardware architecture of RSA processor*

Architecture: We designed our hardware architecture as shown in Fig. 2. The key control logic is a state machine which decides the state transition and generates all of the control signals for the other datapath logic. Data is input from the interface, and begins preprocessing using a low power step by step method; during the calculation, only the shift unit and 32-bit adder are needed. The results of $M \times 2^{32} \bmod n$ and $1 \times 2^{32} \bmod n$ are stored in memory1 and memory2. Then comes the calculation of $n_0' = -n_0^{-1} \bmod 2^w$; we can use the special Euclidean algorithm, it is not as complicated as n'. The left to right binary exponentiation method is performed and the Montgomery multiplication is the key operation; most of the Montgomery operations are to fetch data from memory, load data into the multiply-add unit and write the result back to memory. The output of the 6-bit counter is connected to the address generator and will generate read or write address signals for all of the memories. The two-port register file memory is used to save area, by the control of the state machine, memories are read in a clock's falling edge and written in a clock's rising edge. To read data from memory, load data into the ALU and write data back to memory in one clock; we should pipeline the design and add registers at the input or output of the memories. Memory1 and memory2 are used to store r (r_{63}, $r_{62} \ldots r_2$, r_1, r_0), memory3 and memory4 are used to store t ($t_{63}, t_{62} \ldots t_2, t_1, t_0$). Another key component is the ALU unit, it consists of latches, two three-term parallel multiply-add units and one 64-bit adder. Here we discuss the delay optimisation of three-term multiply-add unit. Since multiply-add must be finished in a half clock cycle, a 4-2 compressor is used to decrease the critical path. There are a total of 17 partial products (pps16…pps2, pps1, pps0), therefore, four 4-2 compressors are used to compress 16 partial products at the first level while the last partial product (pps16) is added with $c33$ and the output is loaded into the last CSA, then two compressors are needed at level 2, and at level 3 one compressor is needed. At last two 64-bit CSAs are used to get the final result.

Low power improvement: We found that once the operand is loaded into the multiply-add unit, two parallel units work at the same time and the parallel method needs large one-bit full adders, thus switching power rises instantly. The solution is to build latch structures to control the input of two parallel multiply-add units, only one unit needs latches. The output of the latches follows by the input while the clock is at low level, and holds the value at high level. Therefore, at a clock's high level, one multiplier works (without latches) and the other is isolated by latches, and vice versa. In this way we can save 50% power consumption while the performance stays the same.

The other large power module is memory. To save temporary data, at least 4096-bit space is needed, registers are not suitable to store large data which means large area, and single-port memory could not finish read and write speed during one clock, therefore a two-port register file will be the best candidate. Because of large switching power, memory needs to be split into pieces. One 64 × 32 memory is split into two 32 × 32 memories and only one of the two memories works, for example, memory1 works while memory2 stops. Instead of one big memory, splitting into four memories can save over 50% power consumption.

Performance and cost analysis: Since frequency is fixed, the best way to measure speed is the total clock cycles. To get a fair comparison, we select papers that give power test results because of its importance in RFID. We calculate the product of area-power-cycles (APC), the smallest result would be the best design, as shown in the Table 1. Reference [4] has the fastest speed, yet uses large area, though lack of power results, we found that the power is larger than ours since the 64-bit multiplier is used. Reference [8] is fast enough but the area and power are large. References [9] and [6] use less area and power but the APC is still high. Thus, in terms of APC product, our design is the best choice for low power systems.

Table 1: Test results and comparison

Ref.	Tech. (μm)	Freq. (MHz)	Cell area (mm^2)	Power (mW)	Cycles (M)	APC
[4]	0.09	421.94	0.76	–	0.283	–
[8]	0.18	460	5.76	830	0.785	3753
[9]	0.18	60	0.32	70	30	672
[6]	0.18	200	0.61	32.5	1.92	38
Ours	0.13	13.56	0.27	15	2	8.1

There are huge numbers of RSA patents, yet it is difficult to compare them with ours because those patents never offer test results. However, when we compare their algorithms and low power methods, we conclude that our design is the most suitable for low power systems.

Conclusion: An area-efficient and low power architecture is presented for RFID. Benefiting from algorithm improvement and architecture design as well as low power optimisation, the proposed scheme not only reduces power but also saves area. Based on SMIC 0.13 µm CMOS technology, the area of 0.27 mm^2 and the power of 15mW make the processor especially suitable for passive RFID and other low power systems. From the APC product, we can see that our test result has better performance than existing designs.

Acknowledgment: This work was supported in part by NSFC-Guangdong Key project, grant (U0935002).

21 May 2012
doi: 10.1049/el.2012.1767

D.M. Wang, Y.Y. Ding, J. Zhang, J.G. Hu and H.Z. Tan (*School of Information Science and Technology, Sun Yat-sen University, Guangzhou, People's Republic of China*)

E-mail: is04wdm@mail2.sysu.edu.cn

References

1 Sutter, G.D.: 'Modular multiplication and exponentiation architectures for fast RSA cryptosystem based on digit serial computation', *IEEE Trans. Ind. Electron.*, 2011, **58**, (7), pp. 3101–3109

2 Iput Heri, K.: 'Very fast pipelined RSA architecture based on Montgomery's algorithm'. 2009 Int. Conf. on Electrical Engineering and Informatics, Selangor, Malaysia, 2009, pp. 491–495

3 Chen, J.H.: 'A high-performance unified-field reconfigurable cryptographic processor', *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 2010, **18**, (8), pp. 1145–1158

4 Miyamoto, A.: 'Systematic design of RSA processors based on high-radix Montgomery multipliers', *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 2011, **19**, (7), pp. 1136–1146

5 Hisakado, T.: '61.5 mW 2048-bit RSA cryptographic co-processor LSI based on N bit-wised modular multiplier'. 2006 Int. Symp. on VLSI Design, Automation and Test, Hsinchu, Taiwan, 2006, pp. 1–4

6 Zheng, X.: 'Design and implementation of an ultra low power RSA coprocessor'. 4th Int. Conf. on Wireless Communications, Networking and Mobile Computing, Dalian, China, 2008, pp. 1–5

7 Finkenzeller, K.: 'RFID handbook: Fundamentals and applications in contactless smart cards and identification' (John Wiley & Sons, 2003, 2nd edn), p. 44

8 Yeh, C.: 'An 830 mW, 586 kbps 1024-bit RSA chip design'. Proc. Design, Automation and Test in Europe, Munich, Germany, 2006, Vol. 2

9 Lu, R.: 'A low-cost cryptographic processor for security embedded system'. ASPDAC 2008: Design Automation Conf., Seoul, Republic of Korea, 2008, pp. 113–114