Freescale Semiconductor
Application Note

Document Number: AN3057
Rev. 0, 01/2006

Solving Sequential Problems in Parallel

An SIMD Solution to RSA Cryptography

by *Bo Lin*
Digital Systems Division, NCSG
East Kilbride, Scotland

1 Overview

This paper presents an efficient method to implement the widely used RSA public-key cryptography with AltiVec™, a Freescale's SIMD extension to PowerPC® architecture, and summarizes the advantages of dual-core processors. Performance benchmark of an implementation of the e600 processor core, the MPC7447 will be shown, and compared to benchmark results of other processors based on the e600 core. The paper analyses the implementation's power efficiency in "performance per watt", which is a popular criterion in embedded applications and compares power efficiency on various e600-based processors, including Freescale's new dual-core processor MPC8641D. The result shows the dual-core processor is the most power efficient among all of them.

Contents

2 Introduction

SIMD (Single Instruction Multiple Data) technology significantly increases performance with minimum overhead on both silicon area and power consumption and it has been adopted by many embedded processors such as PowerPC® [Reference 3][9], ARM[1], SuperH[11], TI, Intel x86, etc. Although the technology was originally designed to accelerate multimedia applications[3], such as image and graphics, efficient SIMD solutions are now more often used to solve conventional sequential problems in many embedded applications, especially in networking, telecommunications, and automotive applications. For example, AltiVec™, a vector SIMD extension to Freescale's high-performance PowerPC® processors using the e600 core, executes the EEMBC's (the Embedded Microprocessor Consortium) TeleMark suite (see www.eembc.org) more than 12 times faster than the scalar core without using AltiVec™. It is very attractive to leverage SIMD technology in high-performance, low power consumption embedded applications.

This paper presents an AltiVec™ powered SIMD solution for the widely-used Internet security algorithm RSA. RSA is regarded as a highly sequential application since it is heavily involved with big-number modulo exponentiation. In addition, RSA is often used to benchmark various processors. [6][12][2]

The SIMD solution presented in this paper achieves the performance of signing or decrypting a 1024-bit RSA data block within 3.72 ms on a 1 GHz MPC7447 platform. The code is written in C and compiled with gcc in a Linux environment. The performance achieved by AltiVec™ is in the range of many high-performance hardware solutions, even if no in-line assembly or unrolled loop optimizations were used.

In the rest of the paper, Section 3, "RSA Public-key Cryptography", provides a brief overview of RSA public-key cryptography and gives some numerical examples to illustrate this algorithm and a fast decryption method based on the Chinese Remainder Theorem. Section 4, "Applying AltiVec™ to Modulo Exponentiation", summarizes AltiVec™ technology with emphasis on two specific vector instructions, vector multiply sum (vec_msum) and vector permute (vec_perm), which play a key role on this application. In addition, a 128-bit × 128-bit ◊ 256-bit "large multiplier" is sketched to demonstrate how to transform typical sequential operations into SIMD ones. Section 5, "Dual-core Processor and its Performance", scales the AltiVec™ enabled RSA performance to other Freescale's processors including a dual-core processor, MPC8641D, and presents a performance comparison between all the processors on the criterion of "performance per watt" or "operations per second per watt". Section 6, "Conclusion", concludes the application and discusses further directions of applying SIMD technology to sequential problems.

3 RSA Public-key Cryptography

RSA public-key cryptography [10] is the most widely used algorithm in Internet security [4]. Most web browsers implement RSA algorithm to support Secure Socket Layer (SSL) which enables most of today's e-commerce.

In conventional cryptography, a sender and a recipient in secure communication share the same key. The sender uses the key to encrypt a piece of information (plaintext) into an illegible form (ciphertext) and transmit the ciphertext to the recipient over a public network. Only the recipient, who shares the same key, is able to read the message by decrypting the ciphertext.

In public-key cryptography, everyone possesses a pair of keys, a public-key and a private-key. A person publishes their public-key in public sites for anyone else to send encrypted message to them. The private key is kept confidential. Only the person is able to decrypt the encrypted message using their own private-key which is paired with the published public-key.

In RSA public-key cryptography, a person chooses two big prime numbers. They publish the product of the two prime numbers as their public-key and keep the two prime numbers as their private-key. It is widely accepted that the most efficient way to break RSA is to factor the published product of the two big prime numbers. However, to factor the product of two big prime numbers, say, a prime number consisting of 512 bits (i.e., the product is around 1024 bits which is about 310 decimals) is computationally infeasible.

3.1 RSA Details

Denote p and q the two chosen big prime numbers and N their product pq (i.e. $N = pq$). Let M be the plaintext to be encrypted and C the ciphertext. A public parameter e is chosen to be part of public-key. The following is a setup for RSA cryptography:

Public-key: (e, N)

Private-key: (d, N) where $d = e^{-1} \bmod (p-1)(q-1)$ which needs p and q to work out

Plaintext: M

Ciphertext: C

Encryption:

$C = M^e \bmod N$

Decryption:

$M = C^d \bmod N$

It is widely regarded that factoring N is the most efficient way to work out d [10]. Therefore, a person is able to publish (e, N) as their public key and keep (d, N) as private-key.

The following numerical example demonstrates the above procedure:

Example 1:

A person chooses $p = 11$, $q = 17$, $e = 3$, works out

$N = pq = 11\times17 = 187$ and

$d = e^{-1} \bmod (11-1)(17-1) = e^{-1} \bmod 160 = 107$ by using the extended Euclid algorithm,

publishes their public-key, (3, 187), while keeps their private-key, (107, 187), secret.

To encrypt a plaintext $M = 116$ to the person, we calculate

$C = 116^3 \bmod 187 = 129$

and send $C = 129$ to them.

The person can recover the plaintext with their private-key (107, 187) by calculating

$M = 129^{107} \bmod 187 = 116$.

Example 1 verifies the RSA public-key cryptography procedure.

3.2 RSA Decryption with the Chinese Remainder Theorem (CRT)

The RSA execution time is closely related to modulo exponentiation, which is proportional to a cubic function of the size of modulus. In other words, it has a $O(n^3)$ time complexity where n is the bit number of the modulus and the "$O(\,)$" stands for "Order of". Hence, a half-sized modulo exponentiation only takes 1/8 execution time as that on the full-sized ones.

In practice, a small number is always chosen to be the parameter e since it is published and no known weakness has been found for this kind of choice. As a result, RSA encryption is executed very efficiently. On the other hand, because the secret parameter d derived from e and $(p–1)(q–1)$ is usually around the same size of N, RSA decryption is very time-consuming.

However, since knowing d and N is equivalent to knowing p and q, the following procedure based on the Chinese Remainder Theorem (CRT) accelerates RSA decryption:

$M = ((M_q - M_p)A \bmod q)p + M_p$ with the where

$M_p = C_p^{dp} \bmod p,$ $\qquad M_q = C_q^{dq} \bmod q$

with $\quad d_p = d \bmod (p - 1),$ $\qquad d_q = d \bmod (q - 1)$

and $\quad A = p^{-1} \bmod q$

The above procedure involves two half-sized modulo exponentiations, and, as a result, it is 4 times as fast as the RSA decryption with full-sized modulo exponentiation asymptotically. The following numerical example demonstrates the RSA decryption with CRT:

Example 2:

Given $p = 11$, $q = 17$, $C = 116$, $d = 107$, to find M?

Pre-calculate $A = p^{-1} \bmod q = 11^{-1} \bmod 17 = 14$ as part of private-key.

$Mp = 116^{107 \bmod (11-1)} \bmod 11 = 6^7 \bmod 11 = 8.$

$Mq = 116^{107 \bmod (17-1)} \bmod 17 = 14^{11} \bmod 17 = 10.$

$M = [(10 - 8) \cdot 14 \bmod 17] \cdot 11 + 8 = 129.$

The above Example 2 is consistent with the result in Example 1.

4 Applying AltiVec™ to Modulo Exponentiation

4.1 AltiVec™ Technology

AltiVec™ technology is Freescale's high-performance SIMD expansion to the PowerPC® RISC processor architecture. It extends the PowerPC® architecture through the addition of 128-bit vector execution unit, which operates concurrently with PowerPC®'s scalar integer and floating units. It is highly paralleled, allowing for simultaneous execution of up to 16 operations in a single clock cycle.

AltiVec™ offer support for:

- 16-way parallelism for 8-bit signed and unsigned integers,
- 8-way parallelism for 16-bit signed and unsigned integers,
- 4-way parallelism for 32-bit signed and unsigned integers and IEEE floating-point numbers.

Figure 1 illustrates a four-operand, eight-element intra-element operation which does a vector operation as **t = vec_op(a, b, c)** where **op** stands for any defined AltiVec operation such as addition, multiplication, and so on.

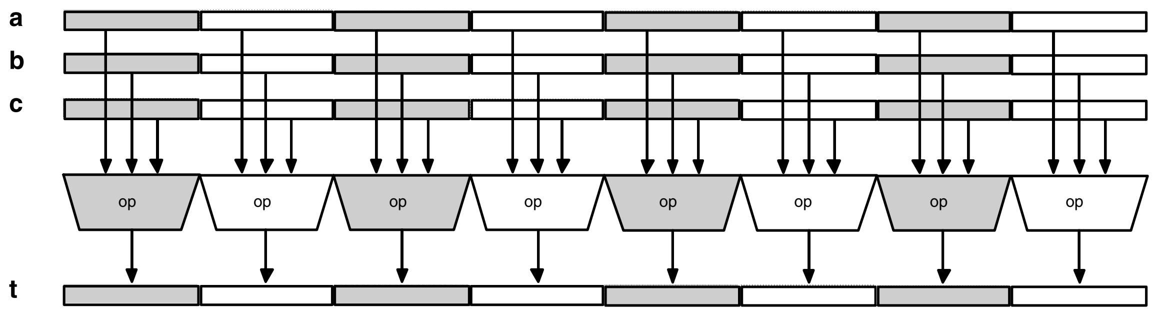

Figure 1. AltiVec™ Four-operand, Multi-element Intra-element Operation

In addition to the intra-element operation, AltiVec™ has a vector permute unit (VPU) which can be regarded having the capability to collect and sort arbitrary 16 bytes from two vectors (32 bytes). This can be shown in Figure 2 as an example where vector **d** “collects” 16 values in **a**||**b** (the concatenation of vector **a** and vector **b**) by specifying 16 indexes in vector **c**.

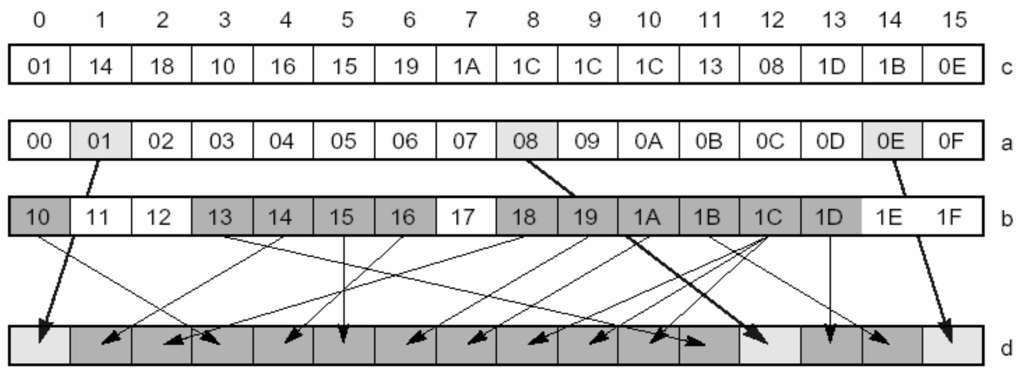
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
01 14 18 10 16 15 19 1A 1C 1C 1C 13 08 1D 1B 0E
c
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
a
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
b
d

Figure 2. Permute Unit

Figure 2 illustrates an AltiVec™ instruction **d = vec_perm(a, b, c)** where byte elements in each vector register are indexed by 0, 1, …, 15. However, we imagine that byte elements in **a**||**b** are indexed by 0, 1, …, 31. The content of byte 0 of **c** is 0x01 which means to move the content of byte 1 of **a**||**b** to byte 0 of **d**. The content of byte 1 of **c** is 0x14 which means to move content of byte 20(0x14) of **a**||**b** to byte 1 of **d**. The same procedure applies to all the other byte elements of **c** and we obtain the result **d** as

```
01 14 18 1016 15 19 1A 1C 1C 1C 13 08 1D 0E
```

By carefully balancing circuit complexity and performance, AltiVec™ provides a set of multiplication instructions. Two of them, vec_mule and vec_msum, are particularly beneficial to big-number arithmetic.

***vC* = vec_mule(*vA*, *vB*)** is described in Figure 3 which is a 4-way 16-bit×16-bit ◊ 32-bit multiplication.

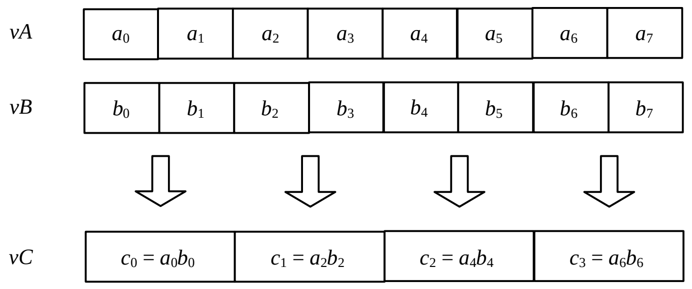
vA
a_0 a_1 a_2 a_3 a_4 a_5 a_6 a_7
vB
b_0 b_1 b_2 b_3 b_4 b_5 b_6 b_7
vC
$c_0 = a_0b_0$
$c_1 = a_2b_2$
$c_2 = a_4b_4$
$c_3 = a_6b_6$

Figure 3. ***vC*** **= vec_mule(*vA*, *vB*)**

vD **= vec_msum(*vA*, *vB*, *vC*)** is illustrated in Figure 4 where a_i and b_i are of 16 bits while c_i and d_i are of 32 bits. The instruction does 8 MAC-like (Multiply-Accumulation) in one cycle.

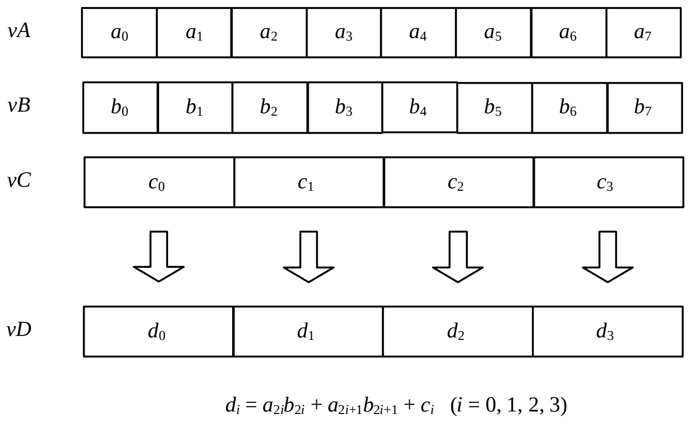
vA
a_0 a_1 a_2 a_3 a_4 a_5 a_6 a_7
vB
b_0 b_1 b_2 b_3 b_4 b_5 b_6 b_7
vC
c_0 c_1 c_2 c_3
vD
d_0 d_1 d_2 d_3
$d_i = a_{2i}b_{2i} + a_{2i+1}b_{2i+1} + c_i \quad (i = 0, 1, 2, 3)$

Figure 4. ***vD*** **= vec_msum(*vA*, *vB*, *vC*)**

4.2 SIMD RSA1024 and its Performance on e600

The key to the SIMD solution to RSA is to implement an efficient 128-bit × 128-bit ◊ 256-bit multiplier with AltiVec™. One approach can be sketched in Figure 5.

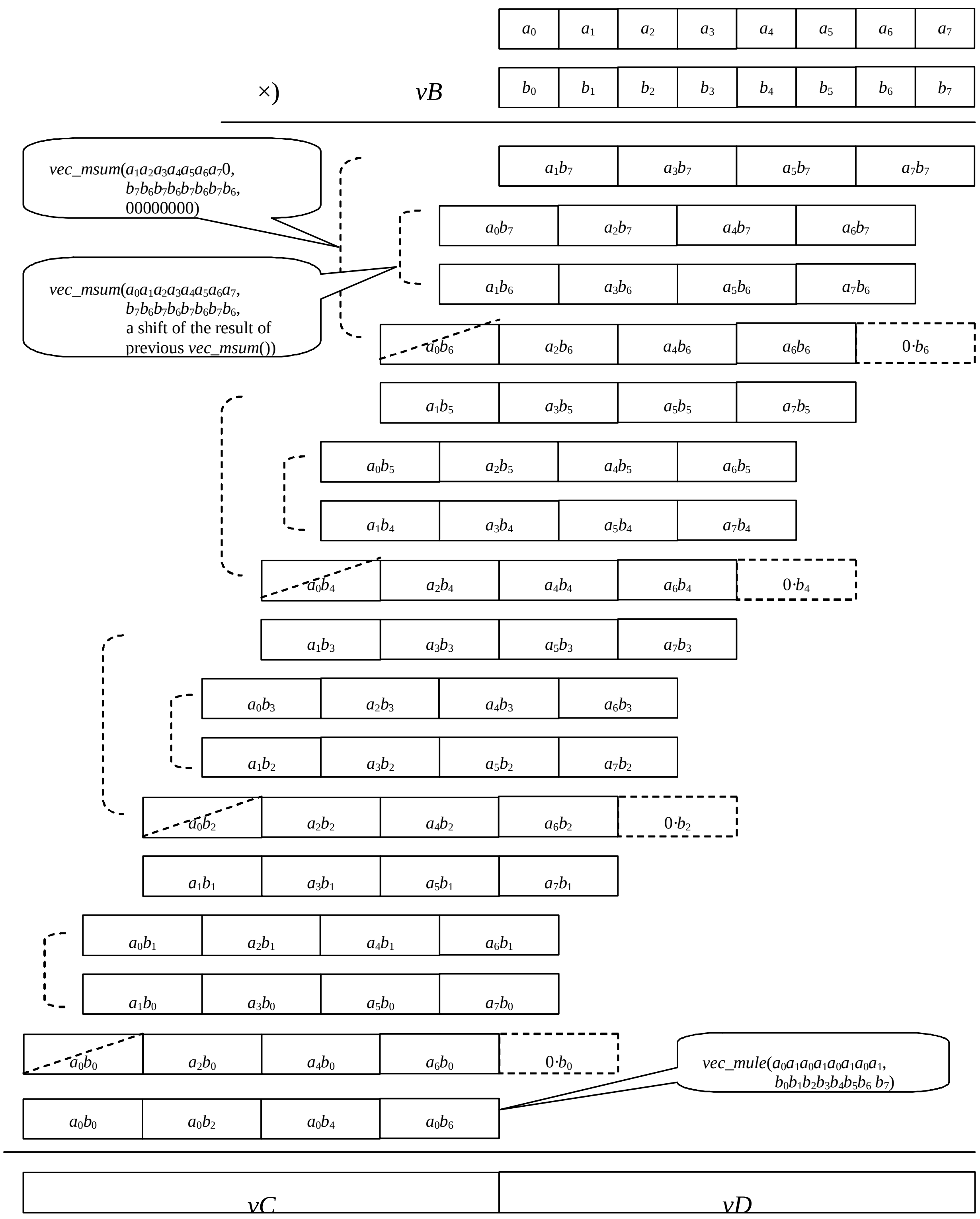
×) vB
vec_msum(a1a2a3a4a5a6a70, b7b6b7b6b7b6b7b6, 00000000)
vec_msum(a0a1a2a3a4a5a6a7, b7b6b7b6b7b6b7b6, a shift of the result of previous vec_msum())
vec_mule(a0a1a0a1a0a1a0a1, b0b1b2b3b4b5b6 b7)
vC
vD

Figure 5. AltiVec™ Implementation of 128-bit × 128-bit ◊ 256-bit (vC||vD = vA x vB)

In this approach, 16-bit elements in *vA* and *vB* are re-arranged into a set of temporary vector registers by AltiVec™'s Vector Permute Unit (VPU) as working vectors for the vec_msum instruction as shown in the callouts in Figure 5. In addition, to keep the involved vectors aligned with vec_msum(), the four products, a_0b_0, a_0b_2, a_0b_4, and a_0b_6, are generated by a single vec_mule() instruction. The sum of all the results from the instructions of vec_msum() and vec_mule() generates the product of $vC||vD = vA \times vB$. Note that many details are not presented here such as the generation of the mentioned temporary vectors, the treatment of overflows, and so on.

Although this approach needs multiple instructions to realize a 128-bit × 128-bit ◊ 256-bit multiplier, it is extremely efficient for several reasons. First, all AltiVec™ instructions are pipelined in three different AltiVec™ units (VPU, VIU1, and VIU2) with the execution rate of one instruction per cycle at each AltiVec™ unit. Second, an e600 core is able to issue two AltiVec™ instructions per cycle, meaning that as AltiVec™ can complete up to two instructions in one cycle. Third, AltiVec™ is able to operate on 128-bit operands, which saves a lot of CPU cycles for load and store.

The 128-bit × 128-bit ◊ 256-bit multiplier efficiently executes big-number modulo exponentiation with the popular Montgomery arithmetic [8]. As a result, a 1024-bit RSA decryption with the Chinese Remainder Theorem achieves less than 3.72 ms to complete on a 1GHz e600-based platform (MPC7447) which is equivalent to 268.81 signatures per second. The code is compiled with gcc 3.3 and benchmarked on Freescale's Sandpoint evaluation platform running under Yellow Dog Linux. The code is written in C with AltiVec™ intrinsics. Neither in-line assembly nor unrolled loops is employed to optimize the code.

The above performance is benchmarked on Freescale's MPC7447, which executes AltiVec™ executions in the program order (in-order execution). Newer products, such as MPC7448, MPC8641, and dual-core MPC8641D, give even better performance because they support a new out-of-order AltiVec™ execution feature, which allows an AltiVec instruction can be issued to an execution unit even if its predecessor is blocked in waiting for resolution of operands.

Besides the out-of-order execution feature, a dual-core processor doubles the performance of RSA decryption/signing in two ways. If operation rate is concerned, a dual-core processor simply doubles the operation rate with each RSA decryption/signing having the same latency. For example, a 1GHz dual-core processor can do 537 RSA decryptions in one second with 3.72ms latency for each operation. However, if absolute latency is concerned for each operation, the two half-sized modulo exponentiations can be assigned to the two cores on the processor to achieve a 1.86ms latency. Both methods provide the same throughput.

Figure 6 extrapolates this RSA1024 signing performance to higher frequencies for the same type of single-core and dual core (1.8 x 2) devices under in-order AltiVec™ execution.

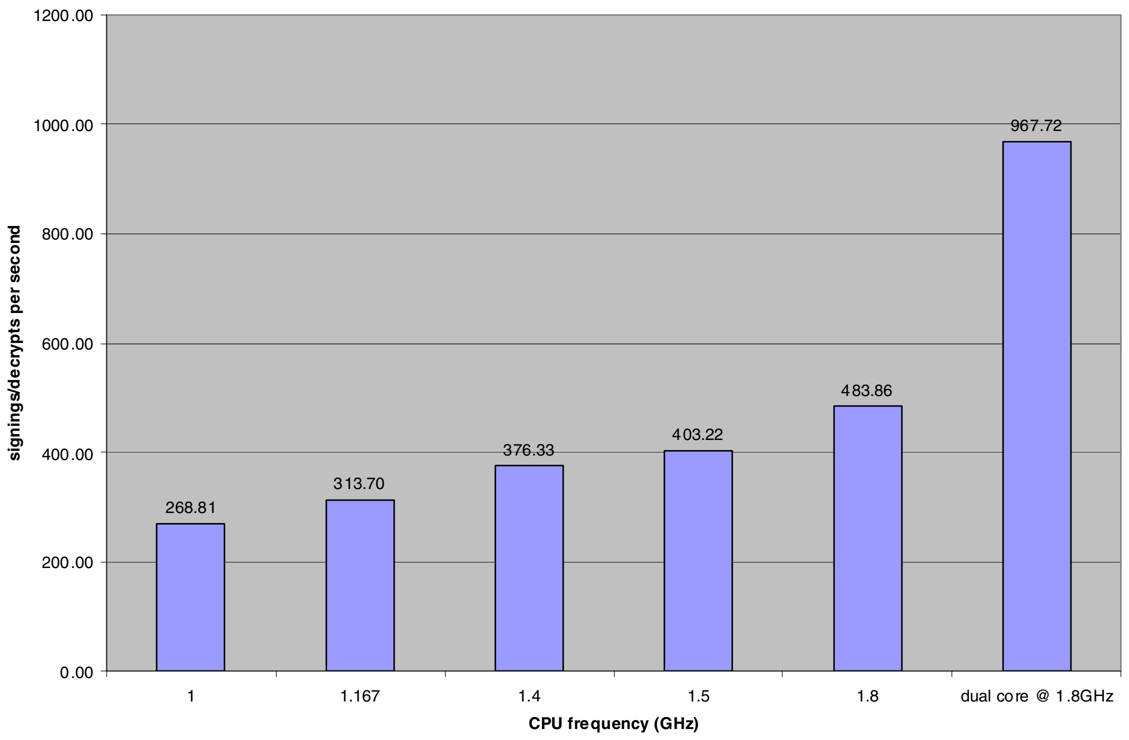

Figure 6. RSA1024 Decrypting/Signing Performance on Single Core and In-order AltiVec Instructions

In addition to increasing performance, dual-core technology is more efficient within a given power consumption budget. Section 4 discusses power efficiency of various Freescale PowerPC® processors and analyses their performance per watt in detail.

5 Dual-core Processor and its Performance

5.1 Advantages of Dual-core Processors

Most embedded applications are both power sensitive and interrupt-response critical for the reasons of reliability, real time, battery life, and so on. It becomes more and more difficult to increase the performance within a given power budget by increasing the clock frequency of the CPU core because increasing frequency demands faster switching transistors that have higher operating voltages and hence higher power dissipation. In addition, it is more challenging for system design because higher frequency leads to smaller timing budgets for peripheral hardware circuits. Furthermore, higher frequency needs deeper pipelines to sustain, which increases the complexity of today's already complicated microprocessors and causes significant stalls following pipeline flushes which happen after the processor takes an unpredicted branch in code or an interrupt, which can seriously impact performance. Usually, doubling the CPU core frequency to increase the performance increases the power dissipation by a factor of four.

One way to overcome the above difficulties is to provide dual-core devices which offer the approximately the same performance as a single-core device running at a doubled frequency. Dual-core devices provide rich CPU cycles in a given time interval which benefit many applications. For example, in a communication data plane application, each core can deal with data traffic one direction. It is also possible to have two cores in a communication system, one for the control plane and the other for the data plane. Dual-core devices are very suitable for low-power consumption but CPU cycle hungry applications.

Advantages of dual-core processors can be summarized as:

1. high-performance with low power dissipation
2. low latency to interrupts
3. easy of system design
4. efficient dual-OS support on one chip

Reference [5] provides a detailed discussion of advantages of dual-core technology.

5.2 Performance Per Watt on Dual-core Processors

There are two very important performance criteria in embedded applications. One is absolute performance and the other is performance per watt. Absolute performance provides information on the number of operations per second while performance per watt indicates the number of operations per second per watt which can be further elaborated as “operation per joule”.

Embedded system engineers need to apply both criteria to their candidate processors for their applications. They usually use absolute performance to gauge whether a processor is capable to complete a task in a given time interval and use performance per watt to see if the processor can be fit into a board within given power budget.

The following Table 1 compares several Freescale low-power PowerPC® processors:

Table 1. Power Dissipation of Freescale High-performance PowerPC® Products

MPC7447	MPC7447A	MPC7448	MPC8641	MPC8641D
e600 core with AltiVec™	e600 core with AltiVec™	e600 core with AltiVec™ (out-of-order instructions)	e600 core with AltiVec™ (out-of-order instructions)	Two e600 cores which each support their own AltiVec™ (out-of-order instructions)
L2 cache: 512KB	L2 cache: 512KB	L2 cache: 1MB+ECC (Error Correcting Code)	L2 cache: 1MB+ECC	L2 cache: 1MB+ECC on each core
8.3W@1000MHz	9.2W@1167MHz	11W@1400MHz	10.9W@1500MHz 6.1W@1000MHz	18.6W@1500MHz 10.1W@1000MHz

Figure 7 compares the power efficiency of some of Freescale's current PowerPC® processors. The power efficiency in Figure 7 is the ratio of the RSA1024 decryption/signing performance from Table 6 to the power consumption of each processor (see Table 1). Note that the power for 8641 and 8641D are estimates based on engineering models and are subject to change.

Figure 7 clearly shows that dual-core processor such as the MPC8641D are more power efficient than single-core devices. The power efficiency numbers in Figure 7 can also be used to compare PowerPC® processors with AltiVec™ to other processors that don’t have AltiVec™ capability. According to [7], the power efficient Low Voltage Intel Itanium™ 2 processor is 31 RSA decrypts per second per watt (1 GHz with 1.5M L3 cache), while the PowerPC® processor MPC8641 (single core, 1 GHz) achieves 44 decrypts per second per watt and the dual-core (1 GHz) achieves 53 decrypts per second per watt. This comparison underlines the power-efficient architecture of Freescale's PowerPC® processors, making this processor family an ideal choice for embedded system designs.

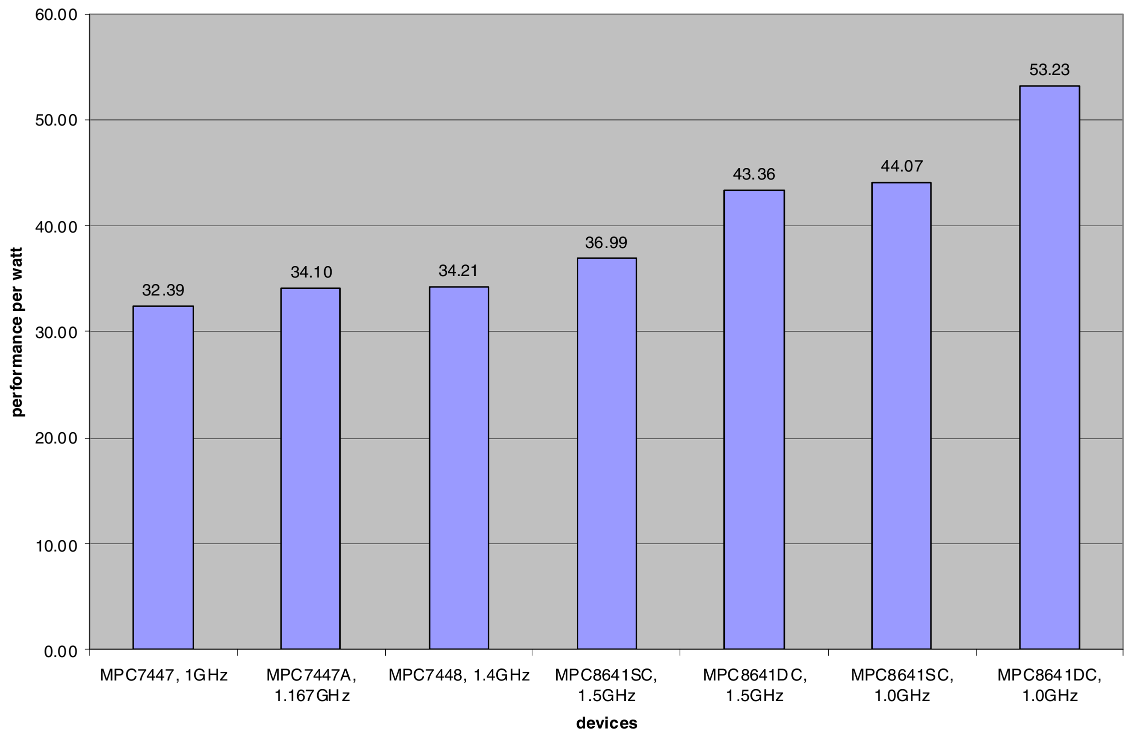

Figure 7. Performance Per Watt Comparison

6 Conclusion

This paper presents an efficient SIMD solution to very popular RSA public-key cryptography. Although the solution is based on AltiVec™, similar ideas can be applied to other SIMD architectures. In addition, power efficiency is examined on Freescale's e600-based processors with emphasis on dual-core devices. It is concluded that dual-core devices are more energy efficient than single-core devices.

In addition to RSA public-key cryptography, SIMD solutions have been found for many other sequential problems such as convolutional encoder, cyclic redundancy check (CRC), Reed-Solomon code, and so on. Many materials, application notes, and white papers on Freescale PowerPC® processors and AltiVec™ are available at http://www.freescale.com/powerpc. SIMD technology is pervasive in embedded applications.

7 References

1. ARM, http://www.arm.com/products/CPUs/arch-simd.html
2. Dai, W., http://www.eskimo.com/~weidai/
3. Diefendorff, K., P. K. Dubey, R. Hochsprung, H. Scales, “AltiVec Extension to PowerPC Accelerates Media Processing”, IEEE Micro, March-April, 2000.
4. Dierks, T. and C. Allen, “The TLS Protocol Version 1.0”, RFC 2246, January 1999.
5. Hamilton, D, “Two heads are better than one”, IEE Electronics Systems and Software, October/November 2004.
6. Intel white paper, “Intel Itanium Processor – High Performance On Security Algorithms (RSA decryption Kernel)”, available at http://www.intel.com
7. Intel white paper, “Broadening the Reach of the Intel Itanium 2 Processor Family”, available at http://www.intel.com
8. Montgomery, P.L., “Modular Multiplication without Trial Division,” Mathematics of Computation, Vol. 44, No. 170, Apr. 1985, pp. 519-521.
9. PowerPC, http://www.freescale.com/powerpc
10. Rivest, R. L., A. Shamir, L. Adleman, “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems,” Communications of ACM, Vol.21, No. 2, Feb. 1978, pp. 120-126.
11. SuperH, http://www.superh.com/products/sh5.htm
12. Welker, M. W., “AMD Processor Performance Evaluation Guide”, September 2003, available at http://www.amd.com

THIS PAGE INTENTIONALLY LEFT BLANK

THIS PAGE INTENTIONALLY LEFT BLANK

THIS PAGE INTENTIONALLY LEFT BLANK

How to Reach Us:

Home Page:
www.freescale.com

email:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
1-800-521-6274
480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku
Tokyo 153-0064, Japan
0120 191014
+81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate,
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor
Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447
303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor
@hibbertgroup.com

Document Number: AN3057
Rev. 0
01/2006

freescale™
semiconductor