International Journal of Natural and Engineering Sciences 5 (1): 53-58, 2011
ISSN: 1307-1149, E-ISSN: 2146-0086, www.nobel.gen.tr

A New High-Speed Multiplier Using Modified Partial Product Reduction Algorithm

P. ASADEE*
Islamic Azad University, Varamin branch, IRAN

***Corresponding Author**
e-mail: p_asadi@iauvaramin.ac.ir

Received : August 08, 2010
Accepted : December 15, 2010

Abstract

In This paper a new high-speed multiplier using modified partial product reduction algorithm with Wallace method is proposed. Three important modifications are done for multiplier in this study. In partial product generation step of multiplication a new Booth algorithm is proposed which decreases number of partial products quarterly. In partial product reduction method a modified Wallace algorithm is presented that sums partial products very fast and is more regular than previous works. In final addition step a novel final adder are presented that sums two final operands efficiently. Simulations are done using HSPICE in 80 nm CMOS technology. Presented multiplier decreases number of transistors more than 14 percent, power consumption reduction is 16 percent and area reduction is 8 percent in compare with previous works.

Keywords: adder, Booth, CMOS, VLSI, Wallace

INTRODUCTION

Multiplier is one of the important operators in arithmetic processors and DSP applications. Therefore there are growing requests for high-performance multipliers in different structures of new computer architectures, like scientific computation, multimedia systems and so on. Much research has been done until now, analyzing different high-speed multiplier structures. Because its speed often decides how quickly the processors can operate. The increasing expansion of electronic systems in recent years has done new problems to the integrated circuit engineers. When designers use more and more capabilities with lot of real-time calculation, while performing at the high speeds, there are needs major changes at all stages of IC implementation. Improved calculating power has capable us to expand complicated structures and perform high-performance procedures. It also consequences in a need to grow processing component even more to do these functions more powerfully. As a result, studies have always been done to achieve more high-speed calculation hardware. This requests the improvement of high-speed arithmetic operators, such as adders, dividers, multipliers, etc.

The fast changes of mobile electronic equipments with bounded power and layout has developed a variety of low-power consumption and high-speed gate design requests and has major effect in IC structures [1]. Mobile systems and laptops are two types of mobile electronic systems that are used much more than before. Newly, the power of integrated circuits has obtained much consideration because of the propagation of high-speed, mobile, low-power electronic systems [2]. Multiplier is an important basic operator in arithmetic processors. Because, multiplication are used in most processor algorithms; therefore, high-performance multiplier is much requested. Multiplication delay has an important role in deciding the instruction period of arithmetic algorithms. With a substantial growing need for greater calculating power on portable systems, design parameters have moved from decreasing delay time and layout size to minimizing power consumption while still obtaining the high efficiency. The three important parameters for IC implementation are power consumption, layout and delay. There are many presented logic methods for low-power consumption and high speed and each method has its importance in expression of more speed and less power. Pass-transistor technology is proposed as one of the ways that can impact on circuit efficiency [2].

In section two, a new modified Booth algorithm is presented which decreases number of partial products efficiently. In section three a novel full adder is proposed which has better electronic parameters than previous works. In section four a four to two counter is designed. In section five, overall multiplier is shown. In section six a carry lookahead adder with modified structure is proposed. Simulations are done in section seven.

Modified Booth Algorithm

In this section a new modified Booth algorithm is presented. The modified Booth's encoding based on a high-radix, is the most used way for designing high-speed multipliers with various encoding [3]. It has a digit set {0,±1,±2} to decrease the amount of the partial products to $\bar{n} = [\frac{n+3}{2}]$. The high-radix Booth method is based on a parallel encoding to decrease this number to $\bar{n} = [\frac{n+2}{3}]$. All digit encoding {0,±1,±2,±3} are performed by simple moving and inverted operations, except production of the multiple 2X (referred to as complex multiple), which is calculated by an summing and moving operation, 4X=2X+2X; -2X can be produced by invert 2X. Some arithmetic processors have used the Booth-4 method to decrease the adder tree of partial products and to raise the speed of the multiplier. Other designs [4] are used Booth-6 multiplication to decrease the layout, power and delay [3]. The structure of modified Booth encoding is shown in Fig. 1.

It uses a collection of carry skip adders with no carry chain and one two's complement number must be summed. However, an implementation trade-off must be performed. High-radix method based on two's complement logic to remove the 4X calculating is proposed in [5], while limitations such as speed or layout are not synthesized. In other Booth structures, as explained in the implementation of arithmetic processors, the 4X is pre-calculated the crucial path, concluding in a high-speed and low power consumption multiplier. We explain N bit numbers A and B by expressions $a_{N-1}a_{N-2}..a_0$ and $b_{N-1}b_{N-2}..b_0$ correspondingly. The produce of the two numbers can be expressed as

$$A \times B = \left(-a_{n-1}2^{n-1} + \sum_{i=0}^{n-2} a_i 2^i\right) \times \left(-b_{n-1}2^{n-1} + \sum_{j=0}^{n-2} b_j 2^j\right) \tag{1}$$

In a basic parallel multiplication process of two Q bit numbers, all the Q partial products are produced concurrently and the summation operation between them is performed by an array of Q*(Q+1) counters. The multiplication speed is stated by the latency time related with the counters. For the mentioned multiplier, the latency is 2N+1 counter delay since the critical carry propagate that spreads in the array is through the left and top borders of the adder tree [6]. The conventional Booth algorithm permits the multiplication algorithm to hop over any adjacent sequence of zeros and ones in the operator, before use a partial product for every bit. Hopping over a sequence of zeros is simple, but in hopping over a sequence of ones, the following principle is performed; a sequence of ones can be calculated by take away the weight of the leftmost one from the operand.

At last, NEG produces a 0 or a 1 relating upon whether the MUX produces a positive or a negative bit. Subtraction can be done using two's complement summation. This includes sum one to the negative of the multiplicand at the least significant bit for ADD and 2ADD procedures. The additional one is produced by the algorithm. Fig. 1 presents the method for two's complement Booth encoding multiplication. Each MUX is eight bits; six bits are for the multiplicand and two additional bits at the MSB are for the sign of the multiplicand and to perform the right-move procedures add and sub functions are used. Booth's encoding can in addition be used to perform negative binary illustrations by using a redundant number system [4]. Redundant number systems permit both positive and negative bits within the number. The main thought behind Booth's encoding is to remove long sequence of one's by modifying the bit illustration.

Sign expansion is presented in Fig. 1 by two values PY and RY where Y shows the row in the partial product table. The sign expansion is handled by Y which shows whether the generated Booth value is sign extended. Each Y expression is then summed to the section in the last location for suitable two's complement encoding. It is important to recognize that sign expansion can solve out the bits in the partial product table when the multiplicand is positive and the multiplier chooses a positive value. As a result, a EX-OR gate between the leftmost bit of the multiplicand and the high significant bit for the partial product generated bits in the multiplier produces the one to remove the encoded ones accurately as showed Fig. 1. In addition, because the LSB of the input partial products is the encoded bit, modified Booth multipliers decreases the amount of partial products to n/3.

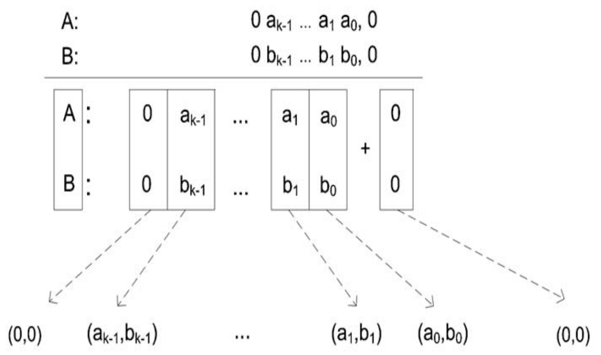


Fig 1. Modified Booth encoding

Proposed Full Adder

In this section a new full adder is proposed. The full adder is the basic gate in many arithmetic operators, such as multipliers. Because these algorithms have major affect the general speed performance in VLSI designs [2], their efficiency modification is critical in high-performance systems, and special applications usually need a trade-off between power and delay [1]. Additionally, as arithmetic operators have considerable effect to the whole power consumption, their power decrease becomes an important aim to design low-power chips used in mobile electronic

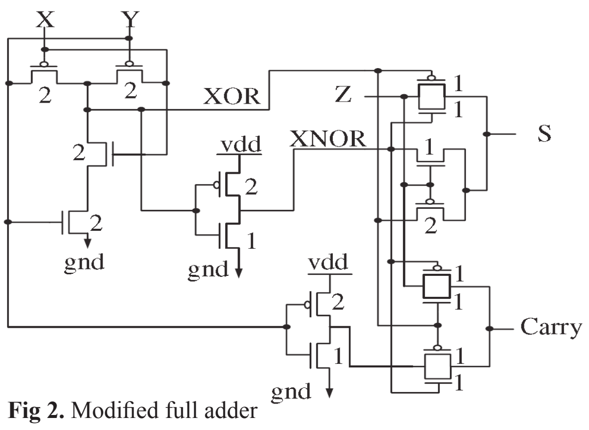


Fig 2. Modified full adder

devices. Fig. 2 shows efficient proposed full adder using transmission gates.

Recently many algorithms both with and without driving capacity have been implemented to design full adders. In full adders with dynamic capability, inputs and outputs are buffered because the way between them contains the gate input of one or more components, as the conventional CMOS full adder. In full adders without pass-transistors, inputs and outputs are not buffered, as in design with domino-CMOS and transmission gates which do not use inverters between the input and output elements. Considering the full adder designs with pass-transistors, the domino CMOS are used in blocks of Fig. 3 which has confirmed to be the most high-speed but has high energy consumption.

Fig. 2 presents a full adder module as a carry save summation that is usually used in IC implementation. It has three inputs and two outputs. The three inputs have an equivalent weight and the output 2 bits are a carry signal and a save signal, accordingly. This is the binary illustration of input addition. Carry chain bit (p_i) and carry produce bit (g_i) can be described as $X_i + Y_i$ and $X_i Y_i$ accordingly. So, when a sequence of adders is used concurrently, three bits (x_i , y_i and c_i) can be decreased to two digits (S_i , C_{i+1}) which is a carry digit and a sum digit.

Easiest way of power decrease is to apply three-input NAND gate base on the sum and carry expression that is a sequence for the carry skip adder. In addition, the amount of power decrease is important because it calculates the delay of the whole power consumption decrease stages. Also, the amount of power decrease decides the needed number of fundamental modules of full adders in the whole power decrease stage while implementing the low power adder with multi stage power decrease way.

High efficiency implementation with high-speed adders is one of the most conventional applications performed in arithmetic processors and image processing systems. Due to its high delay and the needed large power consumption, we used a new synthesis and verification way, multi stage power decrease method, for this component implementation. This is one of the critical methods that estimate system performance.

Some preceding known full adder designs are presented [2]. The performance of these preceding implementations and the presented full adder module has been synthesized. Component simulation was done with HSPICE using 80 nm CMOS technology for supply voltages form 1.2 to 2.4 V. Input bits were used through buffers (W_p=90 nm, W_n=85nm) and the output signals were derived by output of three domino CMOS inverter gates. Calculated interconnect wiring were used to implementing circuit outputs for the simulations. The power, latency and power-latency product are obtained by simulation for power supply from 1.2 to 2.4 V for three different implementations. The presented 14-transistor adder cell is the most power efficient among all these full adder modules [3]. It has the least power consumption and chain delay. While the 10-transistor full adder does not performed reliably at a low power supply (V_{dd}=1~1.5 V), the presented 14-transistor full adder performs appropriately at a low power supply.

Proposed Compressor

A new four to two compressor is proposed. The presented four to two counter called static CMOS four to two counter is described in the following. It is base in dynamic CMOS logic [6]. In this section the XNOR logic is performed. It has no through path to ground therefore decreases the power. Here the load saved at component is used for the control logics. The mixture of not having a shortest path to ground and re-use of the load capacitance to the control logic does the energy improving full adder effective and this can be designed only using 18 transistors.

A conventional way as applied in [5] has been performed. Here four input expressions C_1,C_2,C_3 and C_4 obtained from inputs I_1,I_2,I_3 and I_4 as described before. Each expression is tested 4 times using different technologies. All input bits have an increase node and a fall node of 400 ps. So, for each logic 15 HSPICE synthesize are done and in each circuit the rise time latency and fall time latency for sum are wrote down and the maximum latency and maximum energy consumption is shown for the circuits in the same method as is performed in [2]. Every time the area is measured and the simulation calculations are computed with number of transistors and power latency product. With computing of area the latency, power and power latency product are mapped.

Multiplier Design

The multiplier runs the R=P*Q+L expression, where P is the m-bit multiplier, Q is the n-bit multiplicand, and C is the y-bit computed number which is saved once in the register and then returned (y>m+n). Fig. 3 shows partial product reduction tree presented in this study. Conventionally, the multiplier is made of the reduction tree and the final adder. When a high-speed operator is needed, a CMOS buffer is substituted between two steps. In this architecture, the latency of the entire system is affected by the latency of the multiplier.

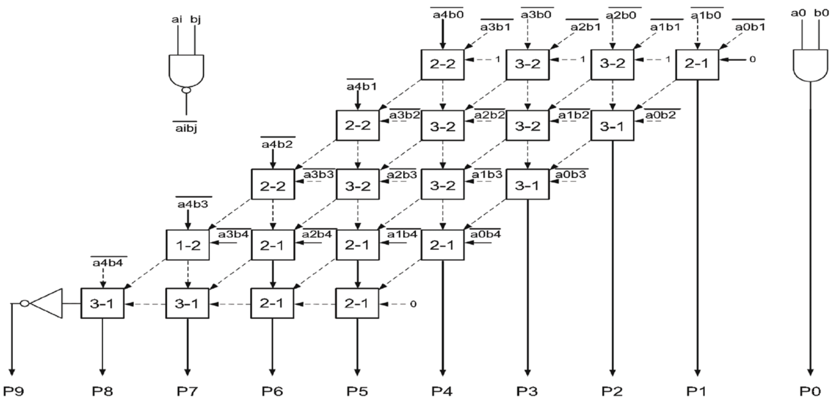


Fig 3. Proposed partial product reduction tree

It has been analyzed a problem in which multiplicand Q, which is used by the modified Booth encoding, is sign-extended to 2m bits (m is a number of not less than one). The 2m bits resulted by the sign extension are a string of 0 when the weight is negative and are a string of 1 when positive. As a result, the partial products of the sign-extension slice are all 0, independent of whether operand Q is positive or negative. By using the above specifications, it can be determined that the output of the multiplier is sign-extended to (q+r+1) bits by summing 0s to all bits after the sign bit of the top stage partial product. Multiplication formula is modified in Eq. (2)

$$P = \sum_{k=0}^{(n/2)-1} \mathbf{x}_{2k}\mathbf{y}_{2k}2^{4k} + \sum_{k=1}^{(n/2)-1} \{\mathbf{x}_{2k}Y_{2k} + \mathbf{y}_{2k}X_{2k}\}2^{2k} \quad (2)$$

The presented EXOR-NAND multipliers use five skip full adders, and the three new four to two counters, to allow the use of the 6-transisotor EXOR gates in producing the partial product bits, as a replacement for the 5-transistor AND and OR gates. The aim is the decrease of both the amount of transistors and the power consumption of the high-speed multipliers.

Previous partial product bits production, of two's complement tree multipliers, has $(n+1)^2+3$ AND gates and 2(n+1) NAND gates [4]. The presented tree multiplier needs (n^2+3) NAND gates and only two AND gate to produce the partial product bits. The concurrent use if NAND gates, in place of AND gates, concludes in decreasing the amount of transistors by [3(n+1)-3] and results a shorter time latency and a lower power consumption are needed. The carry skip adder implementation was made of the presented kind four adders [2]. This is in reason of that two computed rows are complemented and type two adders use all of this four signals complemented while the non-complemented Q output (Fig. 2) results the correct final product. This is like to the use of kind two skip adders as carry chain adder in the conventional array multiplier.

When implementing a multiplier, a regular layout is very important. The layout regularity will not be significantly changed by using four various adders since a multiplier with a large operands includes mostly of three to two counters. In addition, the dynamic adder modules use the equal number of transistors and have same amount of inputs and outputs. As a result, they all have layouts of

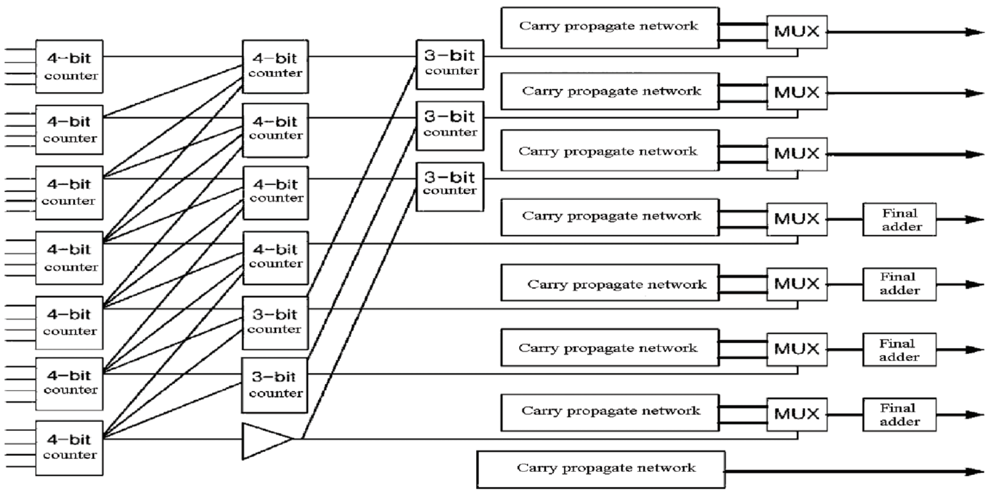


Fig 4. Presented multiplier architecture

about the same size. One important factor affecting layout is the structure of input and outputs of each module to make the way among near modules very simple. Fig. 4 shows presented multiplier structure. The second step of the multiplier is a partial product reduction algorithm, using counters. Four to two counters are analyzed in this study because they use two steps of gates and their parallel structure is like to two chained adders. Some counter circuits in low power circuit implementations use several simpler gates. However only three components are applied in this design, the gate paralleled structure needs four buffers. Since, counter circuit design without chaining gates is also used.

For a counter implementation that does not use chain logic circuits, the gate paralleled structure of a partial product reduction tree is used by summing a buffer at every counter output and by using of a number of latency optimized buffers. Wallace tree bit part decreasing the amount of partial products from 16 to 3, is presented. It uses 10 full adders and has 8 parallel steps. It also includes a total of 18 buffers, 7 of them are the latency optimized buffers. Since, a Wallace tree with the gate parallel structure of the counter presented includes 55 buffers and performs in 12 parallel steps.

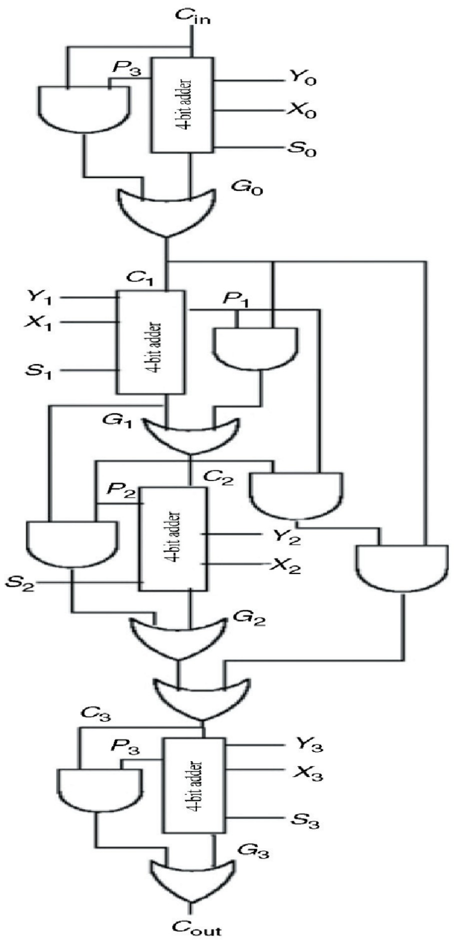


Fig 5. Proposed carry lookahead adder structure

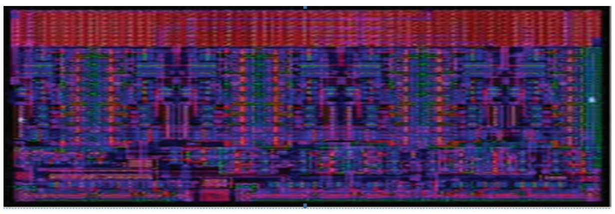
Carry Lookahead Adder

The simplest adder structure is the carry-chain, which is shown in Fig. 5. Modules M_0 and M_1 are counters. Each counter calculates a sum and carry that produces twoaddends and a carry-in as follows:

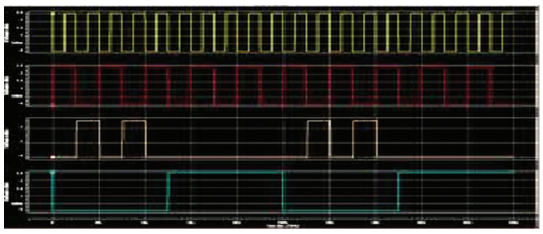
$$\begin{aligned} \overline{H}_{i+1} &= \overline{x}_{i+1}\left(\overline{x}_i + \overline{H}_{i-1}\right), \\ K_{i+1} &= x_{i+1}(x_i + K_{i-1}), \\ S_i &= x_i \oplus \left(\overline{H}_{i-1} K_{i-1}\right), \\ S_{i+1} &= \overline{x}_{i+1} \oplus \left(\overline{x}_i \overline{H}_{i-1} + x_i K_{i-1}\right). \end{aligned} \tag{3}$$

Simulations are done using HSPICE in 80 nm CMOS technology. The presented partial product algorithm uses a new high-radix method for partial product generation step. Presented encoding method has benefits like new carry-skip algorithm proposed in this study, which improves prefix operation in the new encoding. An appropriate benefit when compared to other partial product generation implementations is that our algorithm does not need any additional hardware for changing produced partial products to high-radix structure. This structure also removes the need for a generating word like the one used in previous algorithms [5].

The comparison of different partial product generation modules for a 32*32 and 48*48-bit multiplier is synthesized. Our partial product generation method obtains the highest amount of regularity among other multipliers. The results about the amount of partial products for 64 bit multipliers was obtained from the previous articles [4], however the computation on the amount of partial products for the 64 bit multiplier was perform in this study. It may be important that in the case of 54 bit multipliers all the previous multiplier algorithms increased the optimum number of partial products (24) for a 3-step partial product generator. However, for the implementation of 54 bit multipliers using previous designs, an additional adder step will have to be included in the partial product reduction, concluding in a expected increase in multiplier latency. Presented multiplier layout and its simulations are shown in Fig. 6. Table I shows comparison between 32*32-bit multipliers.



(a)



(b)

Fig 6. Presented multiplier a) layout b) simulation results

CONCLUSIONS

In this paper a new high speed multiplier is presented. A multiplier algorithm consists of three Parts. These three parts are partial product generation, partial product reduction and final addition. In partial product generation step, a new Booth encoding are presented that reduces number of partial products efficiently. In partial product reduction step a novel Wallace tree is proposed which is much more regular than previous designs. In final addition step of algorithm a carry lookahead adder are presented that sums two final operands in a fast way. Simulations are done using HSPICE in 80 nm CMOS technology. Presented multiplier decreases number of transistors more than 14 percent, power consumption reduction is 16 percent and area reduction is 8 percent in compare with previous works [2,6].

REFERENCES

[1] O.T.C. Chen, S. Wang and Y.W. Wu, "Minimization of switching activities of partial products for designing low-power multipliers", IEEE Transactions on VLSI systems, vol. 11, issue 3, pp. 418-433, 2003.

[2] K.S. Chong, B.H. Gwee and J.S. Chang, "Low energy 16-bit Booth leapfrog array multiplier using dynamic adders", IET circuits, devices & systems, vol. 1, issue 2, pp. 170-174, 2007.

[3] L.D. Van and C.C. Yang, "Generalized low-error area-efficient fixed-width multipliers", IEEE Transactions on circuits and systems I, vol. 52, issue 8, pp. 1608-1619, 2005.

[4] Z. Huang and M.D. Ercegovac, "High-performance low-power left-to-right array multiplier design", IEEE transactions on computers, vol. 54, issue 3, pp. 272-283, 2005.

[5] J.H. Tu and L.D. Van, "Power-efficient pipelined reconfigurable fixed-width Baugh-Wooley multipliers", IEEE transactions on computers, vol. 58, issue 10, pp. 1346-1355.

[6] L. Sousa and R. Chaves, "A universal architecture for designing efficient modulo 2^n+1 multipliers", IEEE transactions on circuits and systems I, vol. 52, issue 6, pp. 1166-1178, 2005.