International Journal of Natural and Engineering Sciences 4 (1): 73-79, 2010
ISSN: 1307-1149, E-ISSN: 2146-0086, www.nobel.gen.tr

A New High-Speed Booth Multiplier Using Modified Components

P. ASSADY

Islamic Azad University, Varamin branch, Iran

Corresponding author
E-mail: p_asadi@iauvaramin.ac.ir, assady2@gmail.com

Received: November 04, 2009
Accepted: Januraary 29, 2010

Abstract

A new multiplier has been presented. Multiplication is a basic and important building block in all arithmetic logic units. Due to the large delay of multipliers, different methods have been designed to increase speed. In this study, a novel tree multiplier structure is presented that has regularity of array multipliers and the efficiency of tree multipliers and is capable to implement in large structures. In partial product generation step a new recoding technique is proposed. This algorithm efficiently decreases number of partial products. In partial product reduction step, a modified Dadda structure is presented. This method sums partial products efficiently and is regular. In final addition step a high-speed propagate adder is designed which adds two final operands. Simulations are done using HSPICE and C codes. Proposed multiplier implementation decreases number of transistor count about 8.5 percent, delay reduction is 10 percent and power dissipation is decreased 11 percent in compare with other multiplication algorithms.

Keywords: Booth, CMOS, counter, Dadda, multiplier

INTRODUCTION

In computer systems, high speed multipliers are basic component [1]. As computer systems become more complicated, high-speed and more accurate multipliers have been more essential. The structure of multipliers is different based on different systems because its applications are different. All arithmetic logic units have multipliers in their structure. However, the multiplication is complicated and difficult in VLSI implementation [2,3]. Multiplication is done into two stages fundamentally. One is the production of the partial result, and another is summation of produced partial results to be added to the final addition. Best way to increasing the performance of multiplier is to decrease the number of partial results while decreasing its production time. Because a decreased structure has delay problem related with the production of the partial result, the delay time in a processor without multiplier is compared with the delay time of a multiplier and then an architecture is chosen [4,5]. Multipliers use adders in their structure. Designing high-speed adders have important role in multiplier performance. Adders are basic units that are chained in Dadda structure.
Real time system for speech, video, and other such applications is expected due to the increasing use of mobile systems, e.g., cellular phone, and laptop computers. Most new used IC implementations for the multimedia system are RISC architectures and DSP structures. Higher-bit processor has been broadly used in computer architecture, however could not gratify new application areas like real-time graphic and/or web requests. In addition, traditional 16-bit or 32-bit DSP has been used for signal processing merely, which isn't appropriate for image processing and cannot implement in real-time due to large amounts of data needed for image and multimedia process. So, high speed multiplier for the multimedia DSP structure or RISC system must be used. The multiplier should be implemented to present both fast multiplication and less hardware. High Radix Booth's algorithm for the multiplier is usually used and XOR circuits, adder, Booth encoder, MUX and fast final adder are used for high speed and low power circuit in this project [5,6].

MODIFIED BOOTH ENCODER

In this section a new Booth encoder is presented. Initially, we briefly review the previous modified Booth algorithm that has been conventionally used [2]. It is designed on the encoding the two's complement multiplier to facilitate decrease the number of partial results to be summed [3]. This results to high-speed multiplier and with a reduction of hardware layout. The modified Booth algorithms using high radix is made of the slicing of the multiplier into partly cover sets of 3-bits, as three-bit encoding, and each set is encoded to produce the correct partial result. The multiplier, Y, in the two's complements form can be expressed as:

$$Y = \sum_{i=0}^{\frac{n}{2}-1} (Y_{2i-1} + Y_{2i} - 2Y_{2i+1})2^{2i}$$

It can be rewrite using three bit structure and produced signals are:

$$zero_i = (\bar{x}_{2i+1}.\bar{x}_{2i}.\bar{x}_{2i-1})$$

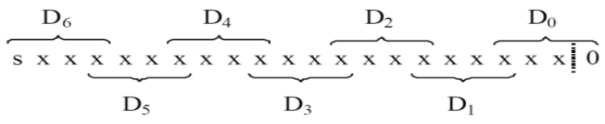
$$two_i = (x_{2i} + x_{2i-1}).\overline{zero}$$

$$neg_i = (x_{2i}.x_{2i-1}).x_{2i+1}$$

Calculating Y, using the modified Booth algorithm, produces five digits. It can be calculated using above equation, the expressions in brackets have values -2, -1, 0, 1 and 2. Each digit in the multiplier does a definite operation on the multiplicand, X, as illustrated in Fig. 1. The multiplication of two numbers produces only 4 partial results. The partial result in this case is presented on three groups. For the addition of a correct partial result, the signs are expanded, as shown in Fig. 1. So, a new modified Booth algorithm is presented that reduces number of partial products efficiently.

MULTIPLIER STRUCTRE

In this section, a new 4-2 counter is presented. The multiplication operation for two binary numbers, using 2's-complement representation and high-radix Booth algorithm is done. High radix encoding of the multiplier means a decreasing in the number of digits



Then, the partial results multiplexer must select one out of choices relating on the value of the following digit. In this step, the partial result is two bits wider than the multiplicand, resulting 24-bit partial results. The most appropriate structure for the reduction of 8 partial results uses only 4-2 counters [5] (instead of the conventional 3-2 counters) which is presented in Fig. 2. The eight to two counter uses 3-2 and 4-2 counters. It has 6 logic gates delay. The final addition must be done in a fast method, so they are designed with carry and sum propagate adders. Proposed multiplier design has important effect in final results.

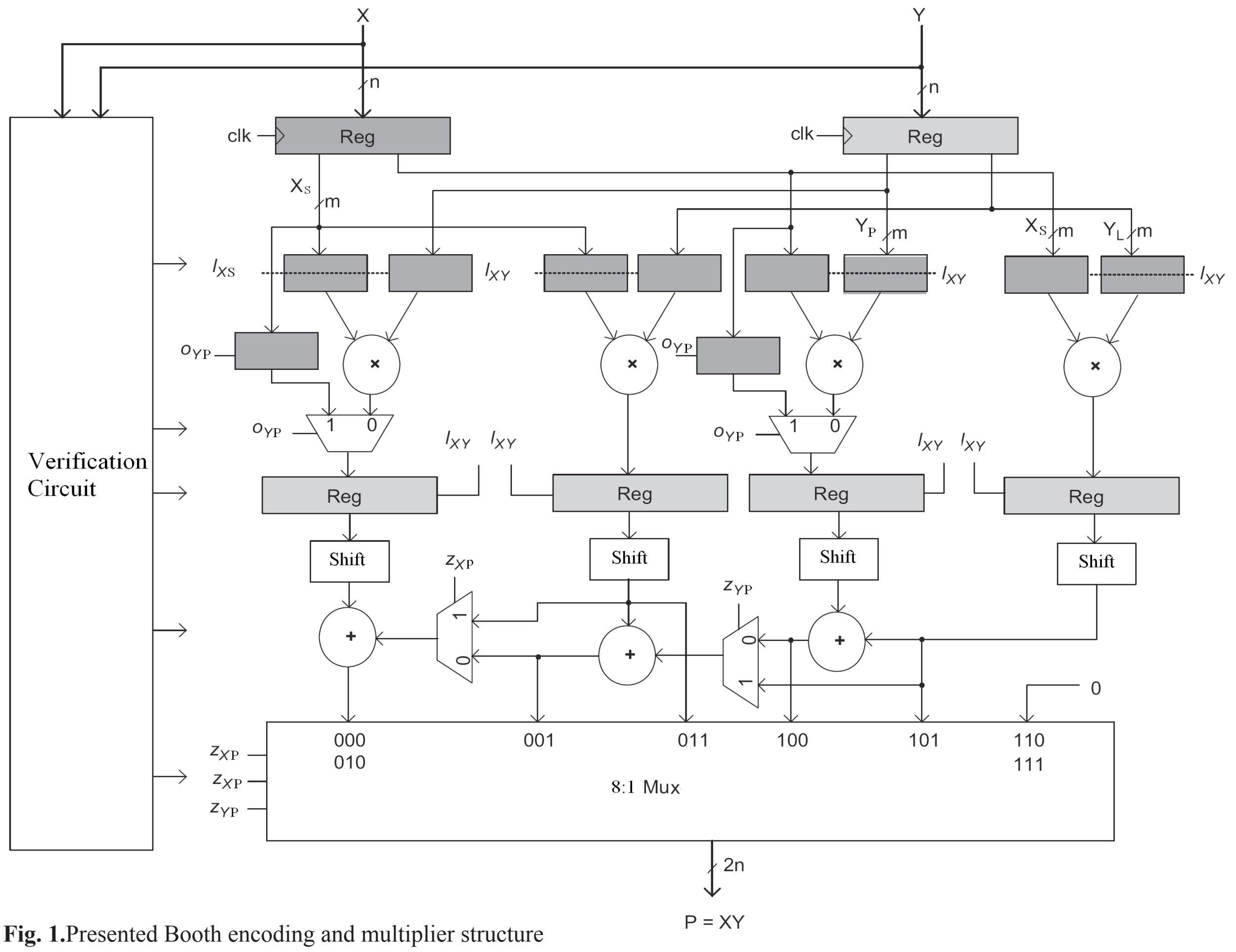


Fig. 1. Presented Booth encoding and multiplier structure

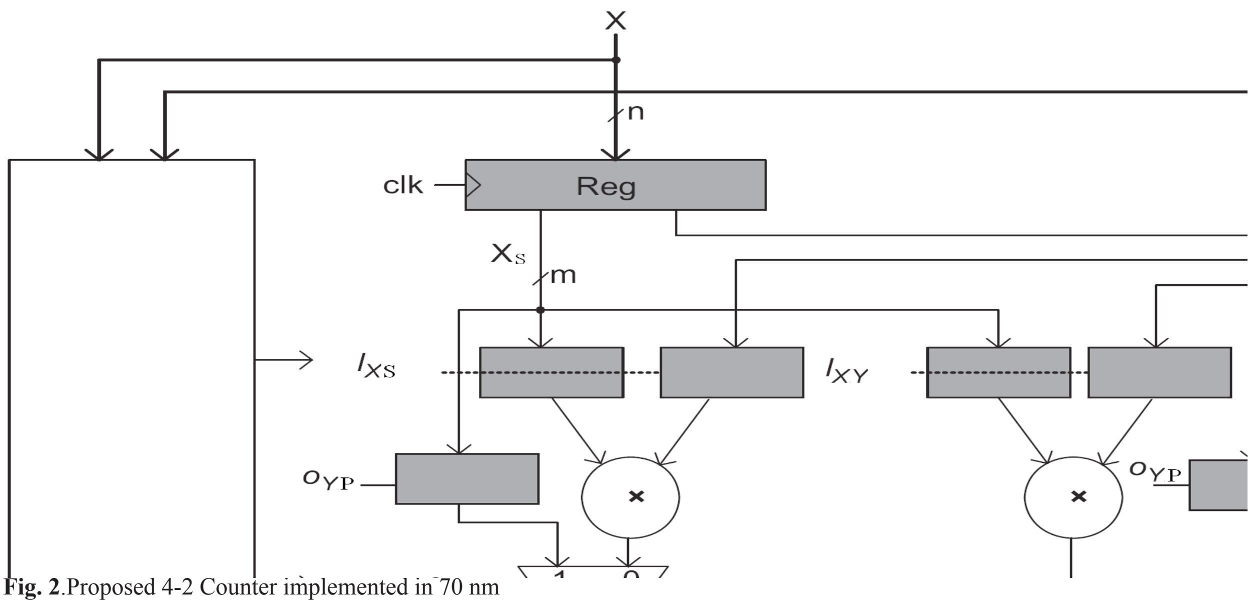


Fig. 2.Proposed 4-2 Counter implemented in 70 nm

PARTIAL PRODUCT REDUCTION

In this section, a modified Dadda tree is presented. Dadda structure makes possible to have an architecture, which does the addition step in parallel, thus resulting in high speed. Adders include a delay proportional to the logarithm of the operand size n, which is smaller than the array structures [4]. On the other hand, the problem of Dadda tree is its complicated and irregular structure in compare with array architectures. In Fig. 3, an example of multiplication using Dadda tree partial result reduction algorithm is presented. It is important that parallelizing two adder operations result in a smaller partial result addition after only one stage [3]. A three view of the 8 input 4-2 tree has been presented. In generating 4-2 multiplier, each tree part forms a single bit slice of the array. The regularity of 4-2 trees over 3-2 trees, therefore, results to more performance. One cause is that regular structures have expected capacity. Proposed multiplier layout is showed in Fig. 4.

The first idea behind the selection of the structure is the comparison between area, speed and power. An ideal design should use fewer layouts but still perform high-speed operation. From the argument of the multiplier structure, the smallest design of the multiplier is array multiplier, although, it is also the slowest structure. The second idea behind the selection of the structure is the regularity, which means the implementation can be partitioned into several equal components. By connecting these components, a new circuit can be easily made. The implementation time is considerably decreased in this method. Large requests are performed of regular architectures to modification of the implementation process. Regularity results a modification in design by generating specific implementations in a number of locations, thereby decreasing the number of different implementations that require to be designed. Array multipliers with Booth encoding are more regular than other structures. A parallel array multiplier has many advantages. It has the benefits of conventional array multipliers such as regular design and minimum layout. By entering one or two parallel steps into the multiplier, the speed can be increased. The speed problem of the array multiplier is solved in this method. Logic implementation is vital in the structure of the multiplier. First, to warranty the multiplier to operate at the high speed, the delay of the crucial path must be computed and the needed time of entering a parallel step should be calculated. Second, to decrease the layout of the multiplier, different structures of adders are developed. Logic synthesis is required to experiment speed and efficiency of the adders. The structure of the adder has to be designed first. Then the number of the parallel steps can be determined by the performance of the adder. The dimension of the multiplier should be as short width as possible if all the needs can be obtained. Final result are calculated using

$$P = \left\{ x_{n-1}y_{m-1}2^{m+n-2} + \sum_{i=0}^{n-2}\sum_{j=0}^{m-2} x_i y_i 2^{i+j} \right\}$$

As discussed before, the usual method in all of the partial result summation algorithms is to decrease the m partial results to two final operands. A carry hybrid is then designed to sum these two operands. One of the first summation designs was the Wallace algorithm, where counters are used to decrease 4 bits of each column to 2 bits. For 16*16 multiplications using 2*2 multiplier productions, height is 9 and five steps of counters needed. Dadda used the expression “counter”. Counter is a logic network with n outputs and m inputs. The n outputs show a binary amount of one’s entered at the inputs. The adder in the previous Wallace tree presented in [4] is a three to two counter.

The Wallace algorithm can be merged with shift and add methods to implement high performance designs. A fundamental need for such design is an appropriate flip flop

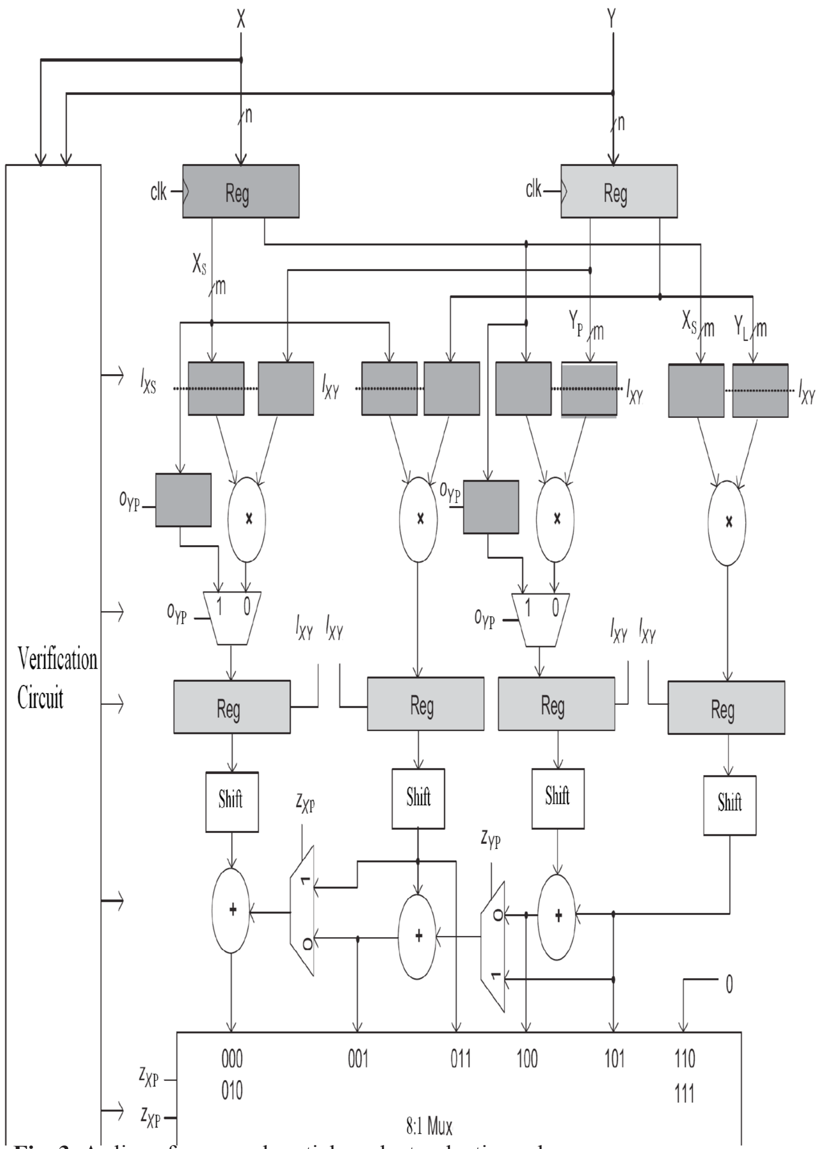


Fig. 3..A slice of proposed partial product reduction scheme

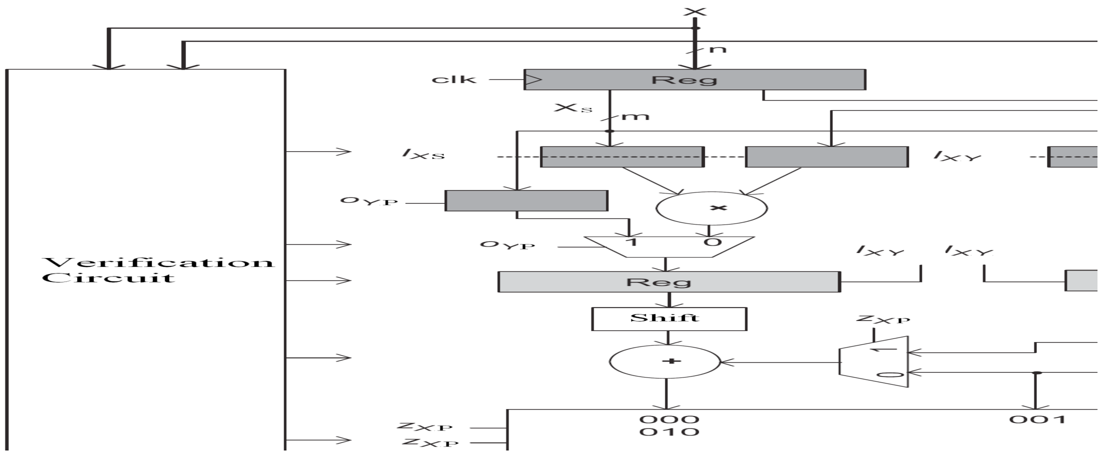


Fig. 4..Proposed multiplier layout

to save middle products. In this sentence "appropriate" means that it does not use extra delay to the calculation. The proposed flip flop, perform this by connect a path from the output of a circuit to an input. Therefore, path delay is not raised. In a given design, existing "and-or" logic is substituted by the flip flop which also does the "and-or" procedure. Use of this flip-flop can considerably decrease design overheads if suitable care is taken in implementation. Assume the multiplication of an m-bit multiplier on an n-bit multiplicand; that is, we decrease m partial results, each of n bits, to two final results, and then chain the carries to obtain final result.

In the implementation of a summation tree, two things are necessary to be solved. First, which kind of counter structure is to be used and second, how these designs should be used to obtain less delay. A counter structure adds the number of ones on its inputs and sums that number at the outputs. The simplest counter is full-adder module, but more complex and efficient counters have been proposed [6]. To achieve the overall short delay, the final adder has to be with awareness designed to the delay computed of the main outputs from the partial result summation tree. Designed algorithms suitable for the final adder are described by [5]. In this study, it is limited the choice of addition structures to half adders, full adders and high speed counters. [4] and [6] have both computed the minimum number of counter circuits, in a summation tree, using different counters. This lower bound in fact matches to the hardware performance in e.g., Dadda, Wallace, and array multiplier methods. The counters can be designed in many methods as triangular assignment, or rectangular assignment. The distance between the modules in the triangular form is taken to be a minimum layout. A triangular form is analyzed, which makes the most appropriate performance. A triangular form is suitable for some multiplier methods, but this will be insignificant for the conclusion of this paper. A large area wasted will remain regardless of form of structures.

The two important tree multipliers are those proposed by Wallace and Dadda. Wallace proofed that the delay for an M*M multiplier can be decreased to log M, making more speed than the array multiplier. In Wallace's algorithm, a high performance adder (M adders without carry propagate) is designed to add three rows into a two row product with only one conventional full adder delay. The function is done continuously to produce two operands of partial results from M operands of partial results for an M*M multiplier. These two rows are then merged using a fast carry chain adder. Dadda developed and expanded Wallace's conclusions by nothing that a counter can be thought of as a circuit which adds the number of bits in the input, and then outputs that number in binary structure. Using such an adder, Dadda assumed that, at each step, only minimum number of summation should be performed so as to decrease the partial results by a factor of 0.5. In the Wallace algorithm, the partial results are decreased very fast. In comparison, Dadda's algorithm does the minimum summation needed at each step to do the summation in the same amount of steps as needed by the Wallace algorithm concluding in an implementation with less counters.

CARRY RIPPLE ADDER

In this section, a new final adder is proposed. A critical part of multiplier is the final addition adder. Because a random number of bits of the middle result could be transferred into the lower half, this adder must be capable to hold changeable bit-width operands. In addition, some of the least significant bits of the middle result have previously been decided; therefore, no addition should be done at these bit locations. A bit, from lower stage is used to decide which slice of the shifted middle result is going to be determined. This bit is consequent from the output of the priority bit, therefore other bits, which relate to the omitted bits of the multiplier; generate a 1, while other bits are 0. The resulting bits have as many one's as the number of bits that are transmitted and, thus, as the number of bits that would be summed. It is a carry-chain adder, where a carry-in can be used at any bit location and one of operands can be include the carry slice of the middle result neg or zero. Table I shows comparison between different 54*54 bit multipliers. A part of multiplier (8*8-bit) is simulated using HSPISE which is shown in Fig. 5. Proposed final adder improves final addition step of multiplier.

The carry-in of module $M_i M_i$ is calculated by the carry-out of $M_{i-1} M_{i-1}$. The carry-chain adder is layout efficient and appropriate to implement, but is slow since the whole latency is related to the amount of bits of the operands. Carry lookahead adders have more speed than carry-chain architectures, because the carry propagate is sliced at the block edges. The worst case latency of the carry lookahead adder is depended to the maximum module length $(\max(P_0, \ldots, P_{m-1}))(\max(P_0, \ldots, P_{m-1}))$ and to the amount m of modules.

The optimal module slice of lookahead adders is quite complex. If the counter is created by many small modules then the latency through the carry-lookahead networks may considerably reduce speed the calculation. However, if few large modules are used then the component latency limits the whole efficiency. Several designs have been presented [6], to facilitate the best trade-off between these structures, presenting that a non homogeneous module size can be used to decrease the latency. An extra latency optimization with regard to the structure of Fig. 2 can be obtained by using a two-step

Table 1. Comparison between 54×54 bit multipliers

Multipliers	Present study	[3]	[6]
Technology (nm)	70	70	70
Transistor counts	24213	26135	28531
Multiplication time (ns)	3.4	4.4	4.2
Chip Area (mm^2)	0.55	0.67	0.72
Power Diss. (mW)	0.54	0.63	0.67

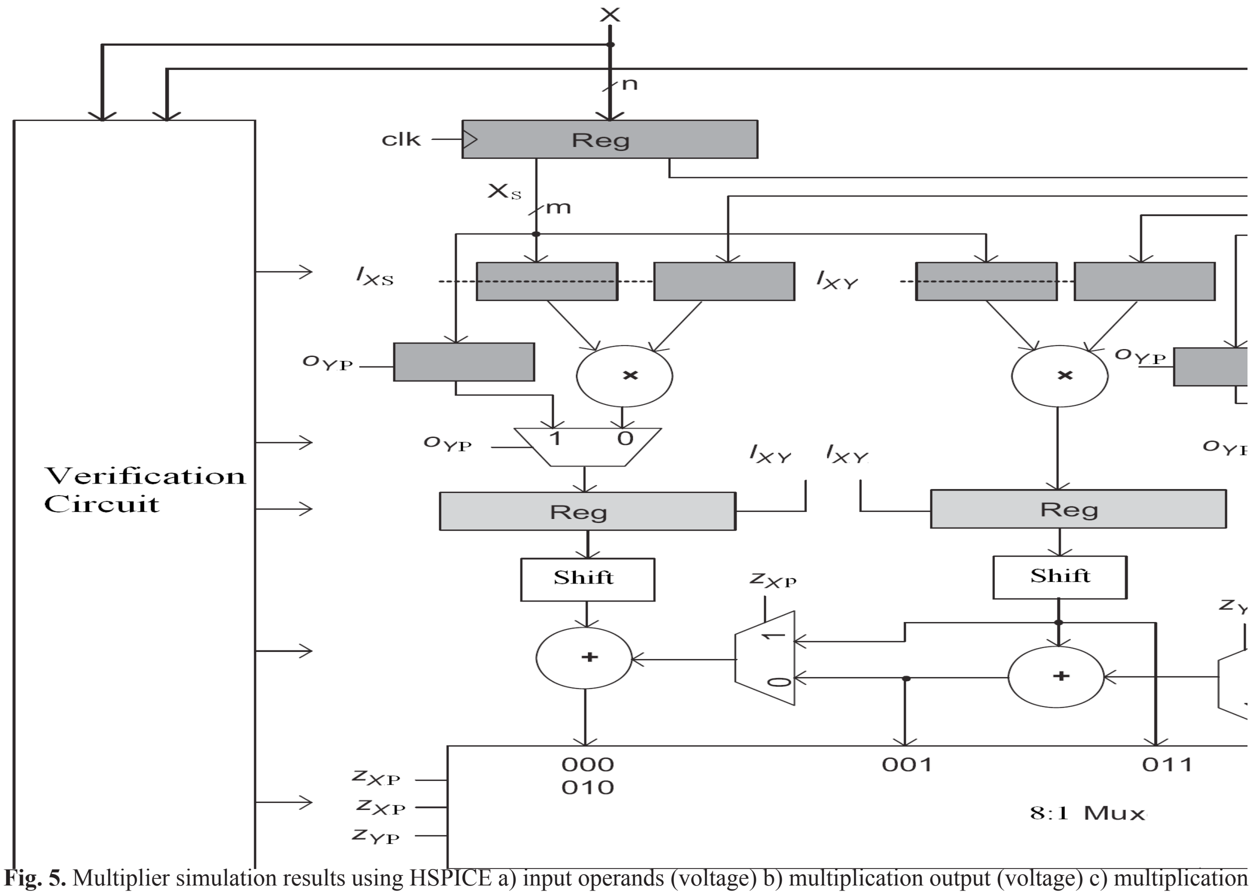


Fig. 5. Multiplier simulation results using HSPICE a) input operands (voltage) b) multiplication output (voltage) c) multiplication output (current)

carry lookahead adder. In this case the counter is sliced in several stages. Each slice is themselves combined by modules of bits (see Fig. 1, which presents the way of a slice combined by two modules). The modules have the equal architecture of Fig. 3.

The presented counter uses domino CMOS encoded carry bits produced by two adders. Therefore the ith carry C_iC_i is showed by two bits. We describe the rise time T_rT_r as the delay from the time when the output bit become available to the last inverted bit. (i.e. T_rT_r=rise time for signal which is generated should be positive ($T_r T_r$>0)). This issue is called the rise time signal latency. In latency-sensitive components, such an issue is analyzed. However, because delay-sensitive structures suppose that latencies of components and layouts are limited, the rise time (T_rT_r>0) must be synthesized. Rise time logic results latency and layout overhead. The larger this component is, the higher the layout overhead and the larger the rise time T_rT_r will be. High-performance multiplier circuitry should be high-speed, simple with small layout overhead, and small $T_r T_r$. A latency-sensitive multiplier structure has large layout overhead and rise time even for small counters. The counter presented here uses a much more high-performance logic circuit which provides a limited latency model. While latency-sensitive circuit analyses calculation completion of all blocks carry nodes, we try to only analyze the completion of calculation of the blocks middle carry structures $C_i C_i$and $C_{i+1} C_{i+1}$ and use that to single completion of calculation for the full three-bit counter block. With $C_3 C_3$ used to verify output of carry calculation of the three-bit counter block, a problem may occur when $C_3 C_3$ has either a carry propagate or a carry free condition while $q_2 = q_3 = 0 q_2 = q_3 = 0$. In this form $C_3 C_3$ may be calculated faster than other signals in the same counter block, with the slowest signal being C_5 C_5 which may be use two transmission-gates latency after $C_3 C_3$.

CONCLUSIONS

Proposed algorithm has done three modifications in conventional multiplier architectures. Multiplier design has three important steps, which include partial product generation step, partial product reduction step and final addition step. In the first step of algorithm, a new modified Booth structure is presented. This algorithm halves number of partial results which results smaller tree in the second step of multiplier. In partial result reduction step, a new algorithm to sum partial results is presented. This algorithm sums partial results very fast and it has regular structure. In the final addition step, a new changeable length adder is proposed which sums two final operands in a new way which, increases speed. Simulations are done using HSPICE and C codes. Presented 54*54-bit multiplier delay is 3.4 ns, transistor count is 24213 and power dissipation is 0.54 mWatt in 70 nm CMOS technology. Proposed multiplier implementation decreases number of transistor count about 8.5 percent. Delay reduction is 10 percent. Power dissipation is decreased 11 percent in compare with other algorithms [3,6].

REFERENCES

[1] W.C. Yeh and C.W. Jen, "High-speed Booth encoded parallel multiplier design," *IEEE Transactions on Computers*, vol. 49, no. 7, pp. 692-701, July 2000.

[2] F. Elguibaly, "A fast parallel multiplier-accumulator using the modified Booth algorithm," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 47, no. 9, pp. 902-908, Sep 2000.

[3] H. Lee, "A power-aware scalable pipelined Booth multiplier," IEEE International SOC Conference, pp. 123-126, Sep 2004.

[4] C. Efstathiou, H.T. Vergos and D. Nikolos, "Modified Booth modulo 2/sup n/-1 multipliers," *IEEE Transactions on Computers*, vol. 53, no. 3, pp. 370-374, Mar 2004.

[5] 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 very large scale integration (VLSI) systems*, vol. 11, no. 3, pp. 418-433, June 2003.

[6] N. Itoh, Y. Naemura, H. Makino, Y. Nakase, T. Yoshihara and Y. Horiba, "A 600-MHz 54*54-bit multiplier with rectangular-styled Wallace tree," *IEEE Journal of Solid-State Circuits*, vol. 36, no. 2, pp. 249-257, Feb 2001.