Area efficient and A High Bit Rate Serial-Serial Multiplier with On-the-Fly Accumulation by Asynchronous Counters

K.Jyothirmai, R.Vinay Kumar
Kakinada institute of Engineering and Technology

Abstract— A novel approach of designing serial hybrid multiplier is proposed for applications with high data sampling rate (4 GHz). The conventional way of partial product formation is revamped. Our proposed technique effectively forms the entire partial product matrix in just sampling cycles for an \( n \times n \) multiplication instead of at least \( 2n \) cycles in the conventional serial multipliers. It achieves a high bit sampling rate by replacing conventional full adders and 5:3 counters with asynchronous 1’s counters so that the critical path is limited to only an AND gate and a D flip-flop (DFF). The use of 1’s counter to column compress the partial products preliminarily reduce the height of the partial product matrix, resulting in a significant complexity reduction of the resultant adder tree. The proposed hybrid column compressed multiplier consists of a serial-serial data accumulation unit and a parallel carry save adder (CSA) array that occupies approximately 35% and 58% less silicon area than the full CSA array multiplier with operands of word length 32*32 and 64*64, respectively. The post-layout simulation results based on 90-nm seven metal single poly MOS process technology shows that our 64*64 multiplier dissipates 39% less average power at a sampling rate of 4 GHz, and has only 11% additional delay penalty to complete a multiplication compared to the conventional fully parallel CSA array multiplier.

Index Terms Binary multiplication, on-chip serial link bus architecture, on-the-fly accumulation, parallel multipliers, serial-serial and serial parallel multiplier.

I. INTRODUCTION

MULTIPLIERS are the fundamental and essential building blocks of VLSI systems. The design and implementation approaches of multipliers contribute substantially to the area, speed and power consumption of computational intensive VLSI systems. Often, the delay of multipliers dominates the critical path of these systems and due to issues concerning reliability and portability, power consumption is a critical criterion for applications that demand low-power as its primary metric. While low power and high speed multiplier circuits are highly demanded, it is not always possible to achieve both criteria simultaneously. Therefore, a good multiplier design requires some tradeoff between speed and power consumption. As the device size shrinks, there are other side effects and it becomes increasingly difficult to achieve a good tradeoff by device scaling or sizing the transistors [1].

A more effective driver to optimize the area, delay and power consumption of arithmetic computations in battery powered VLSI circuits is to explore alternative architectural concepts for the design of digital multipliers. Typically, hardware implementation of a multiplication operation consists of three stages, specifically the generation of partial products (PPs), the reduction of PPs and the final carry-propagation addition [2]. The partial products can be generated either in parallel or serially, depending on the target application and the availability of input data. The partial products are generally reduced by carry-save adders (CSAs) using an array or a tree structure.

Carry propagation addition is inevitable when the number of partial products is reduced to two rows. This final adder can be a simple ripple carry adder (RCA) for low power or a carry look-ahead adder (CLA) for high speed [2]. As the height of PP tree increases linearly with the word length of the multiplier, it aggravates the area, delay and power dissipation of the two subsequent stages. Therefore, it is highly desirable to reduce the number of partial products before the CSA stage. This can be achieved by Modified Booth algorithm to reduce the height of the PP matrix [3]. Another approach is to use high order column compressors instead of full adders (FAs) to increase the PP reduction ratio of the CSA stage [4], [5]. The drawback is that Modified Booth encoder adds both area and delay overheads to the simple partial product generation process, and higher
order compressors are slower and consume more power than the full adders. Hence a hybrid combination of both techniques is often considered. To reduce the wiring cost, it has been a common practice to transmit data over a communication channel through a high speed serial link [6]. For some integrated circuits constrained by I/O pins, the designers often try to reduce the number of I/O pads by serializing I/O data because I/O pads occupy large silicon area and consume high power [7].

Therefore, effort has been made to design high speed serial interface [6], [8]–[12] in order to facilitate on-chip buffering and parallel processing. Parallel multipliers are popular for their high speed operation but long word length multiplications are often constrained by the hardware cost and power consumption of the applications. In public key cryptography like RSA encryption and decryption, integer multiplications of 1024 bits are typical [13]. In Elliptic Curve Cryptography (ECC), key lengths of 112 bits and 109 bits are commonly used for prime field and binary field multiplication, respectively. Therefore, low cost serial multipliers are widely adopted in hardware cryptography [14], [15].

Serial multipliers also find applications in system-on-chip (SoC) design. As technology scales, more intellectual property cores and logic blocks will be integrated in an SoC, resulting in larger interconnect area and higher power dissipation [17]. The increase in integration density of the on-chip modules causes the buses connecting these modules to become highly congested. To overcome this problem, new techniques have been evolved recently to have on-chip data transfer in a high speed serial link instead of conventional bus [16]–[18]. Fig. 1(a) and (b) depict the conventional on-chip bus and alternative on-chip serial-link bus structures, respectively.

In Fig. 1(b), the serializer at the source module converts the parallel outputs to a bit stream that can be transferred in a simple routing network and at the destination module they are converted back to parallel data by the deserializer. The on-chip serial-link is capable of transmitting data at Gb/s so that a chunk of parallel data is available when the destination module finishes the previous computation. Under the new on-chip communication paradigm for digital signal processing, it is desirable to have a low complexity data processing unit as the destination module that is able to perform partial computation on the incoming data stream at high speed while the data is being buffered. Fig. 2 illustrates a potential use of a serial-serial multiplier as a destination module in a SoC with serial-link bus architecture. The low complexity precomputation unit forms part of the serial-serial multiplier and could perform partial computation on the high speed serial bit stream. The unit doubles as a buffer and eliminates the deserializer.

As the data has been partially processed and buffered, the completion of the multiplication can be done at a lower speed with a less complex parallel multiplier. The challenge in such a scheme lies in reducing the critical path delay of the pre computation unit to that of the deserializer, which usually has bit rate in the order of several Gb/s. We introduce this new scheme for the design of serial-serial multiplier suitable for SoCs with on-chip serial-link bus architecture [16]. The proposed scheme could also be used as an alternative to embedded multipliers in the future field programmable gate array (FPGA), where configurable logic blocks (CLBs), embedded multipliers and memory blocks are integrated with serializers/deserializers to facilitate on-chip serial data transfer in order to reduce interconnect complexity [19].

The rest of this paper is organized as follows. Section II revisits some of the existing serial multiplier architectures in the literature. In Section III, a serial accumulator developed based on the new design paradigm is proposed to deal with very high-speed data sampling rate of above 4 GHz. The accumulator employs asynchronous counters1 to perform bit accumulation at each bit position of the PP matrix, resulting in low critical path delay and small area, especially for operands with long wordlength. Asynchronous counter has a low hardware complexity but the outputs are not synchronized with the clock which leads to a timing delay before all output bits of the counter have settled to their final states. The correct output of the counter is read after a timing delay to be analyzed from the timing diagram in Section VI-B.

The data dependent counters change states only when the input bit is “1,” which leads to low switching power dissipation. The height of the PP matrix after buffering by the asynchronous counters is reduced logarithmically to before it is further reduced by the CSA tree.

The details of the newly proposed serial-serial multiplier are described in Section IV. The method is extended to signed multiplication of 2’s complement numbers in Section V. Section VI presents the application-specific integrated circuit (ASIC) implementation results and their comparisons with other design on various performance measures. The
energy efficiency of the proposed serial-serial multiplier design is demonstrated also by the post layout simulation results of the designs implemented with STM 90-nm CMOS process. Section VII concludes this paper.

![Diagram](image1)

**Fig. 1.** On-chip communication among parallel functional modules in SoC. (a) Conventional bus structure. (b) Serial-link bus structure [16].

![Diagram](image2)

**Fig. 2.** Suitability of serial-serial multiplier for upcoming on-chip serial-link bus architectures in complex SoC.

II. REVIEW OF SERIAL MULTIPLIERS

Serial multipliers are popular for their low area and power [20]–[35], and are more suitable for bit-serial signal processing applications with I/O constraints and on-chip serial-link bus architectures. They are broadly classified into two categories, namely serial and serial-parallel multiplier. In a serial multiplier both the operands are loaded in a serial fashion, reducing the data input pads to two [20]–[32]. On the other hand, a serial-parallel multiplier loads one operand in a bit-serial fashion and the other is always available for parallel operation [27], [33]–[35]. Lyon [21] proposed a bit-serial input output multiplier in 1976 which features high throughput at the expense of truncated output. A full precision bit serial multiplier was introduced by Strader et al. for unsigned numbers [22].

The rudimentary cell consists of a 5:3 counter and some DFFs. Later, Gnanasekaran [28] extended the work in [22] and developed the first bit-serial multiplier that directly handles the negative weight of the most significant bit (MSB) in 2's complement representation. This method needs only cells for an –

bit multiplication but it introduces an XOR gate in the critical path, which ends up with a more complicated overall design. Lenne et al. [23] designed a bit-serial-serial multiplier that is modular in structure and can operate on both signed and unsigned numbers. The 1-bit slice of a typical serial-serial multiplier, called a bit-cell (BC), is excerpted from [23] and shown in Fig. 3.

Such cells are interconnected to produce the output in a bit-serial manner for an serial-serial multiplier. The operand bits are loaded serially in each cycle and added with the far carry, local carry, and the partial sum in the 5:3 counter. The cascaded cells form a shift register chain with of one cell connected to the of the next cell. The addition of the symmetric partial product bits (i.e., and ) is serially enabled by pulling first bit (FB) signal low after the first cycle. The partial sum is shifted out serially through another shift register chain formed by the cascaded to connection. The registers are cleared by activating last bit (LB) before the start of a new multiplication. Aggoun et al. [30] proposed a new architecture for serial-serial multiplication with 50% reduction in hardware without degrading the speed. It is observed that all the reported multipliers have a common computational unit known as the 5:3 counter in the critical path. In addition, there are also DFFs and AND gates in the critical path, which lower the operation speed and limit the input bit rate.

Many attempts have been made to reduce either the hardware cost or latency [22], [23] but there is no improvement on the critical path. To overcome this problem, Pekmestzi et al. [26] designed a systolic serial multiplier with a critical path comprising an FA, a 2:1 MUX, a DFF and an AND gate. Nibouche et al. [31] proposed two serial-serial architectures, Structure I and Structure II, which can handle a sampling frequency close to that of the serial-parallel multiplier. The critical path of Structure I consists of an FA, a DFF, and an AND gate but it has a latency of cycles for an multiplication and requires cycles to complete one multiplication.

To reduce the number of computational cycles from to in an serial multiplier, several serial-parallel multipliers have been developed over the years [27], [33]–[35]. Most of them are based on a carry save add shift (CSAS) structure. Fig. 4 shows the unsigned and 2’s complement serial-parallel multiplier based on the CSAS structure. It can be observed that the critical path consists of an FA, a DFF, and an AND gate for the unsigned multiplier in Fig. 4(a) and an extra XOR gate for the 2’s complement multiplier in Fig. 4(b). Gnanasekaran [34] proposed a fast CSAS
multiplier capable of producing -bit output in clock cycles at the expense of an extra RCA.

Fig. 3. Bit serial multiplier and its basic bit-cell

A low complexity 2’s complement serial-parallel multiplier was proposed by Sunder et al. [35]. It used the Baugh–Wooley algorithm to avoid the sign extension problem [2]. Saleh et al. [27] designed many serial parallel systolic and non-systolic multipliers with low complexity based on the Booth’s multi-bit recoding. Although the sampling frequency has been improved in the serial-parallel multipliers and the total number of computational cycles is halved, one of the operands needs to be loaded in parallel. Recently, there has not been any new development in serial serial multiplier design due to the maturity of conventional architectures.

Parallel multipliers are more popular as the size is less critical due to technology scaling. However, with the emerging development of the on-chip serial-link bus architectures [16], [17], [19], serial-serial multipliers could find their potential roles in the new generation of SoCs and FPGAs. In the following sections, an approach to the design of serial multiplier that is capable of processing input data at Gb/s without input buffering and with reduced total number of computational cycles is proposed.

III. PROPOSED SERIAL ACCUMULATOR

Accumulation is an integral part of serial multiplier design. A typical accumulator is simply an adder that successively adds the current input with the value stored in its internal register. Generally, the adder can be a simple RCA but the speed of accumulation is limited by the carry propagation chain. The accumulation can be speed up by using a CSA with two registers to store the intermediate sum and carry vectors, but a more complex fast vector merged adder is needed to add the final outputs of these registers. In either case, the basic functional unit is an FA cell. A new approach to serial accumulation of data by using asynchronous counters is suggested here which essentially count the number of 1’s in respective input sequences (columns). A rudimentary version of our proposed serial accumulator was first introduced in [36].

The inner-summation of (3), represents the sum of all the 1’s presented in the th column and it can easily be accomplished by a simple serial 1’s counter of width . Thus, an -bit accumulator can be realized with such dedicated counters to concurrently accumulate the 1’s in each bit position. The dependency graph (DG) of such a scheme with and is shown in Fig. 5. The nodes in the DG perform , which is equivalent to an increment of if . An example illustrating the serial accumulation of the inputs, “10110001,” “00000001,” “11001101,” “11100011,” “01110001,” “11001101,” and “10001101” in the DG is shown in Fig. 6. The accumulated output after each cycle is indicated by the integer in each node and the final accumulated output in each column can be represented by a 3-bit binary number. At the end of the th iteration, each counter produces an -bit binary weighted output. By arranging all counter output bits of the same positional weight in the same column with the least significant bit (LSB) of each counter output aligned at the th column, the result of the accumulation can be obtained by column.
compressing the counter outputs with a CSA tree structure of height. Note that the height of the CSA tree would be a binary exponent of should the summation be carried out without the 1’s counters. The architecture of the accumulator corresponding to Fig. 5 is shown in Fig. 7. The bits of the input operands are serially fed into their corresponding counters from column 0 (right-most in Fig. 5) to column 7. These counters execute independently and concurrently. In each cycle of accumulation, a new operand is loaded and the counters corresponding to the columns that have a 1 input are incremented. The counters can be clocked at high frequency and all the operands will be accumulated at the end of the th clock. The final outputs of the counters need to be further reduced to only two rows of partial products by a CSA tree, such as a Wallace’s or Dadda’s tree [2].

![Fig. 5. Dependency graph of the proposed accumulator for 7 8-bit operands.](image)

Fig. 5. Dependency graph of the proposed accumulator for 7 8-bit operands.

A carry propagate adder is then used to obtain the final sum. In Fig. 7, the counters C are used to count the number of 1’s in a column. Each of them is a simple DFF-based ripple counter. The clock is provided to the first DFF and all the other DFFs are triggered by the preceding DFF outputs. A typical 3-bit 1’s counter is shown in Fig. 8. The clock input is synchronized with the input data rate and thus the operands can be accumulated with a high frequency defined by the setup time and propagation delay of a DFF. Moreover, the counters change states only when the input is “1,” which leads to lowswitching power. This simple and efficient bit accumulation technique is used to design the proposed serial-serial multiplier.

![Fig. 6. Example for Fig. 5.](image)

Fig. 6. Example for Fig. 5.

![Fig. 7. Architecture of an accumulator](image)

![Fig. 8. Hardware architecture of a 3-bit 1’s counter.](image)

IV. PROPOSED SERIAL-SERIAL MULTIPLIER

This section proposes a new technique of generating the individual row of partial products by considering two serial inputs, one starting from the LSB and the other from MSB. Using this feeding sequence and the proposed counter-based accumulation technique presented in Section III, it takes only cycles to complete the entire partial product generation and accumulation process for an multiplication. The theoretical underpinning of this design is elaborated as follows. The product of two -bit unsigned binary numbers and can be expressed as

![clock input diagram](image)
\[ P = X \cdot Y = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} x_i y_j 2^{i+j} \]  
(4)

By reversing the sequence of index of (4)

\[ P = \sum_{i=0}^{n-1} \sum_{j=1}^{n-1} x_i y_j 2^{i+j} \]  
(5)

By decomposing and rearranging (5)

\[ P = \sum_{r=0}^{n-1} PP_r \]  
(6)

\[
PP_r = PP_r^L + PP_r^C + PP_r^R
\]

\[
PP_r^L = \sum_{k=0}^{r-1} x_{n-k-1} y_j 2^{n+k-1}, \quad r = 1, 2, \ldots, n-1
\]

\[
PP_r^C = x_{n-r-1} y_j 2^{n-1}, \quad r = 0, 1, \ldots, n-1
\]

\[
PP_r^R = \sum_{k=2}^{r-1} x_{n-r-1} y_j 2^{n-k-1}, \quad r = 1, 2, \ldots, n-1
\]

The PPs in Fig. 9(b) are generated in such an unconventional way in order to facilitate their accumulation on-the-fly by the proposed counter based accumulation technique. A bank of counters is deployed, one in each column, to accumulate the bits arriving in the respective column. The DG shown in Fig. 10 illustrates the complete operation of the PP generation and accumulation for a general multiplication. Each node (circle) in the DG represents a binary counter and an ancillary AND gate to generate a partial product bit. All nodes are identical in functionality and have three inputs and three outputs, except that the data are propagated in different directions as depicted in Fig. 10. It can be seen from Fig. 10 that the nodes (L) on the left of the middle column have the identical properties of shifting to the immediate left node in , computing the PP bit and updating the counter output. Similarly, the nodes (R) on the right of the middle column exhibit identical property but shift instead.

In Fig. 10, the middle column has a maximum height of 8 and a 4-bit counter is sufficient to account for the maximum column sum. The column height decreases gradually to either side and the counter width also decreases correspondingly. The operation of the multiplier can be explained with the aid of the structure in Fig. 11.

A PP bit corresponding to the middle column of the PP is produced by the center AND gate when a new pair of input bits ( and ) is latched by the two DFFs (top middle) in each clock cycle. In the next cycle, are shifted to the left and right, respectively, to produce the partial product bits with another pair of input bits and by the array of AND gates. The AND gates are gated by to ensure that the outputs driving the counters are free of glitches. Each counter changes state at the rising edge of the clock line only if a “1” is produced by its driving AND gate. After cycles, the counters hold the sums of all the 1’s in the respective columns and their outputs are latched to the second stage for summation. The latched outputs are wired to the correct FAs and HAs (half adders) according to the positional weights of the output bits produced by the counters. From Fig. 11, it is observed that the column height has been reduced from 8 to 4 and the final product, , can be obtained with two stages of CSA tree and a final RCA. Similarly, for 16 16, 32 32 and 64 64 multipliers the column heights are reduced logarithmically from 16, 32, and 64 to 5, 6, and 7, respectively. This drastic reduction in column height leads to a much simpler CSA tree, and hence reducing the overall hardware complexity and power consumption.

The latching register between the counter and the adder stages not only makes it possible to pipeline the serial data accumulation and the CSA tree reduction, but also prevents the spurious transitions from propagating into the adder tree. Two clocks, namely and , are employed to synchronize the data flow between the two stages. The counter stage is driven by to process the inputs at high speed as the critical path is defined by the delay of only an AND gate and a DFF. The counters and shift registers are reset to “0” in every cycles to allow a new set of operands to be loaded. is derived from to drive the latching register. The output sequence of a 16 16 multiplication is depicted in Fig. 12 without the complication of the actual timing. The detailed performance and timing analysis will be discussed in Section VI. The –bit product of an multiplication is produced in parallel by the final RCA and can be interfaced with other logic circuitry either in parallel or in serial depending on the requirement.
VI. PERFORMANCE COMPARISON AND IMPLEMENTATION RESULTS

In this section, the performances of the proposed unsigned and 2's complement multiplier architectures are evaluated in terms of area, speed, power, and other implementation factors. As the circuits being compared contain different types of logic gates and logic modules, the simplistic unit gate model that assumes all cells used are primitive two-input logic gate is inadequate. It is better to preserve the basic modules as described in the architectures as long as they correspond to the commonly available cells in a typical standard cell library. The area and delay of the proposed multiplier are derived and expressed in terms of the area and delay of the basic logic modules that can be found in a typical standard cell library for different operator sizes. These theoretical estimates are then calibrated by the basic logic modules from STM 90-nm CMOS standard cell library and used to benchmark the proposed multiplier design against other serial-serial multipliers in Section VI-A. The proposed and another parallel CSA array multipliers are also physically synthesized and laid out using the same cell library. The results are presented and discussed in Section VI-B.

A. Comparison With Serial-Serial Multipliers

In our proposed technique, PP bits are formed and accumulated by the 1’s counters in just cycles to reduce the height of the CSA tree logarithmically. The serial operands are input at a high frequency as the critical path of the input stage consists of only an AND gate and a DFF. The critical path for the 2’s complement multiplier remains the same, which is in contrary to most CSAS 2’s complement multipliers where an extra XOR gate delay is required in the critical path. For the proposed structures

\[ T_{\text{Clock}1} = T_{\text{DFF}} + T_{\text{AND3}} \]

\[ T_{\text{Count}} = n \times T_{\text{Clock}1} \]

\[ T_{\text{Clock}2} = (\lceil \log_2 n \rceil - 1) \times T_{\text{HA}} + T_{\text{CPA}} \]

The area complexity of the proposed unsigned and signed multiplier is derived from its basic logic modules. The 1st stage consists of DFFs and AND gates only. The height of the partial product column is symmetrical around the center column and decrements from to 1 towards both sides. Thus, the width of the counter at columns away from the center column is, and the total number of DFFs required for the counters in a multiplier is given by

\[ T(n) = h + 2 \sum_{i=1}^{h-1} i \times 2^{i-1} \]

The area complexity of the second stage can be determined from the number of FAs and HAs required for the CSA tree and the CPA. The CPA is assumed to be implemented by the simplest and lower power dissipation RCA. It should be noted the pattern of the dot matrix in this stage is different from that of the conventional multiplier (see Fig. 11). Owing to the preliminary counter-based reduction, the height of the matrix is reduced logarithmically. In the following derivations, the number of HAs and FAs is derived from the number of stages, \( h \), that is required to reduce the height to 2. The number of HAs, \( s \), required for the proposed multiplier depends on only is constant. The number of HAs in the first stage (for the stage below the latching register) is, and for each of the subsequent stages. One HA is also required in the final RCA. Therefore, can be expressed as

\[ \Pi A(n) = \sum_{i=1}^{s} (2^{s-i+1} - 1) + 1 + 1 = 2^0 + 2^1 + \cdots + 2^s - s + 1 = 2^s - 1 - s \]

To comprehensively evaluate the area-time complexity of our proposed design against some other serial-serial multipliers, the area and delay of the logic cells are normalized with a basic inverter from the standard cell library. This benchmarking method has been widely adopted in the literature as a fair alternative practice when it is impractical to implement all competitor circuits for different dimensions [27], [35]. The normalized areas and delays of various logic cells are tabulated in Table I using the STM 90-nm standard cell library datasheet [38], where the delay and area of an inverter are 7.2 ps and 1.56 m, respectively. As an inverter has a normalized delay of unity, a two-input NAND gate with a normalized delay of 1.57 in Table I implies that its actual delay is 11.3 ps. The data for the logic modules which are not available in the library (e.g., 5:3 Counter) is interpolated from their constituent logic gates. Table II compares the critical path delay, cycles required and computation time of the proposed design with the existing serial-serial multipliers. In the “Area Expression” column of Table II, the critical path delay is expressed in terms of the wordlength and the normalized delays from Table I. Among all, the proposed counter-based multiplier exhibits the
lowest critical path delay of 21.41. This is achieved by eliminating the complex 5:3 counter and replacing it by a simple 1’s counter. In addition, all the PP bits can be accumulated in just cycles in the proposed method while the others need cycles. The logarithmically reduced partial product bits are summed in a single cycle of which can be pipelined with the accumulation stage. The period of is approximated times as that of . Thus, an initial latency of cycles of is used to produce the first output and thereafter one output is generated in every cycle. All other serial-serial multipliers are not easily pipelinable due to their tight integration of the PP formation, summation and carry propagation. The critical path delay remains the same for the proposed 2’s complement multiplier while the others have an additional XOR gate or MUX in the critical path [23], [28], [35]. Fig. 14 shows the total computation time (normalized) of the existing and proposed unsigned serial-serial multipliers for 8, 16, 32, 64, 128 in a bar chart. It is evident from Fig. 14 that the proposed multiplier is much faster than all the other existing serial-serial multipliers, making it suitable for on-the-fly multiplication for high data rate applications, especially those with on-chip serial-link bus architectures. To the best of our knowledge this is the first serial-serial multiplier that can operate at input data sampling rate that is even higher than the serial-parallel multipliers [27], [33]–[35].

![Normalized Computation Time](image)

**Fig. 14. Normalized computation time for unsigned serial-serial multipliers**

The proposed multiplier has an area penalty over other methods due to its semi-serial and semi-parallel structure. It needs a small adder tree of height in the second stage which is not required in the other methods. It also needs more flip flops to accumulate and latch the PP bits. The area complexities of different serial-serial multipliers are compared in Table III, where the “Normalized Area” column lists the area in terms of the wordlength and the normalized values from Table I. The approximation of is used to simplify the normalized area expression of our design in the last column of Table III. The cost of performing the accumulation as fast as the input data buffering required by an I/O limited design by using mixed execution modes in the two stages is the hardware overhead. The tradeoff between area and delay is better illustrated by the area-delay product (ADP) for 8, 16, 32, 64, 128 in Fig. 15. The ADP is shown in logarithmic scale due to the large variation in ADP for different . The area-time tradeoff of our proposed multiplier is acceptable based on the moderate ADP from Fig. 15. In comparison with the parallel multiplier, the proposed multiplier has better ADP, which is discussed in the next subsection.

**B. ASIC Implementation and Post-Layout Simulation Result**

The designs of the proposed 16 16, 32 32, and 64 64 multipliers were coded in structural VHDL, synthesized and technology mapped to the STM 90 nm standard cell library by Synopsys Design Compiler to obtain the gate-level netlists and then placed and routed using Cadence SoC Encounter. The layouts were generated using Cadence Virtuoso and they were DRC and LVS verified. The post layout simulation results were obtained by extracting the RC from the routed designs using Synopsys StarRC-XT. A transistor level simulation was performed on the extracted netlist. The post-layout simulation results of the proposed multiplier are compared with those of the conventional parallel CSA array multiplier to ascertain its merits. For a fair comparison, I/O pads were not included in the layout and the conventional 16 16, 32 32, and 64 64 parallel CSA array multipliers were implemented with on-chip serial data input buffer.

An important implementation issue associated with our counter-based serial-serial multiplier is that due to the asynchronous nature of the serial 1’s counter, the counter output is not available immediately after clock cycles as the last input has to propagate from the LSB to the MSB of the counter. A setup and hold violation can occur if the counter output bits are read at the edge of before they are stabilized. In this case the latching of counter outputs to the second adder stage must be delayed by in the worst case for the longest counter of width at the center column of Fig. 11. This is addressed by refraining to trigger the
counter for the last, which provides an additional guard band of for clock uncertainty. The serial data inputs and are kept at “0” during these cycles to maintain the counter outputs. A min-max timing analysis is performed on the design after placement and routing to ensure that the worst case delay has been considered to avoid any timing violations. The timing waveforms of a 16 16 multiplication captured from the post-layout simulation with Synopsys NanoSim are shown at the bottom of Fig. 16, where and are the inputs (all 1’s for the worse case) to the 5-bit counter at the center column (for , it is a 4-bit counter as in Fig. 11), “Counter” is the output of the 5-bit counter, “Latch” is the output of the latching register, and is set at 4 GHz.

The timing waveforms in the shaded window is zoomed in and generalized for multiplication as shown at the top of Fig. 16. From the standard cell library, it is known that the setup time, clock-Q delay and hold time of a DFF are 70, 40, and 20 ps, respectively. It can be seen that there are almost two cycles of 40 ps and one cycle of 20 ps before and after the active edge of . Thus, the setup and hold time violations are avoided. It can be seen from Fig. 16 that the counter is able to count from 0 to 16 (“10” in HEX) with the outputs correctly latched to the second pipeline adder tree on . The post-layout area, delay and ADP of the proposed and conventional array multipliers for wordlengths of 16 16, 32 32 and 64 64 are tabulated in Table IV.

The delay of the CSA array multipliers consists of the time required to buffer a complete set of operands and the time required for the PP reduction and final CPA. For the proposed counter-based multipliers, the delay consists of the accumulation delay , the delay of CSA tree and final CPA , and the additional delay required for bit-propagation of the counters . From the post layout simulation results of Table IV, our proposed multiplier achieves a delay comparable to parallel multiplier but the area is significantly smaller, especially for larger operands due to its simpler adder network. From Table IV, the ADPs of the proposed 32 32 and 64 64 multipliers are approximately 28% and 52% lower than the corresponding CSA array counterparts.

If a series of multiplications is to be performed consecutively, the proposed multipliers can be easily pipelined into two stages. Although and are defined by (14) and (16), respectively, and of (15) must be equalized in pipeline mode for proper synchronization of the two stages and accordingly and are then determined.
that both the proposed and the CSA array multiplier reported in Table IV are not pipelined at the adder tree. For a fully pipelined CSA multiplier, the clocking frequency could be higher, and so is its throughput, but the area and routing complexity would be significantly increased as each stage of FAs requires a set of DFFs. The average power consumption was estimated by Synopsys Nanosim with random serial inputs at 4 GHz and a supply voltage of 1 V. Monte Carlo statistical model [39] is adopted to obtain the mean power dissipation of each design with 99.9% confidence level that the error is bound below 0.25% at convergence. The Monte Carlo simulation of the mean power for the proposed 64 x 64 multiplier is shown in Fig. 17.

It shows that the cumulative average power recorded after every 0.25 s (Sampling Time) converges rapidly within the 0.25% error bound of the final estimate. The average power consumptions of the proposed and the CSA multipliers were listed in Table V.

Fig. 17. Power convergence result

<table>
<thead>
<tr>
<th>Table VI</th>
<th>Power (mW) AND Energy (µJ) Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>WordLength</td>
<td>CSA Array</td>
</tr>
<tr>
<td>Average Power</td>
<td>Energy</td>
</tr>
<tr>
<td>16 x 16</td>
<td>6.84</td>
</tr>
<tr>
<td>32 x 32</td>
<td>15.75</td>
</tr>
<tr>
<td>64 x 64</td>
<td>41.28</td>
</tr>
</tbody>
</table>

The energy per operation in Table V is obtained by the product of the estimated mean power and the computation time of one multiplication operation. In the case of our proposed counter-based multiplier, the additional cycles required for compensating the bit propagation of the counters from the LSB to the MSB were included.

VII. CONCLUSION

In this paper, a new method for computing serial multiplication is introduced by using low complexity asynchronous counters. By exploiting the relationship among the bits of a partial product matrix, it is possible to generate all the rows serially in just cycles for a multiplication. Employing counters to count the number of 1’s in each column allows the partial product bits to be generated on-the-fly and partially accumulated in place with a critical path delay of only an AND gate and a DFF. The counter-based accumulation reduces the PP height logarithmically and makes it possible to achieve an effective reduction rate of using an FA-based CSA tree. The post-layout simulation results show that the serial input can be sampled at a rate as high as 4.54 GHz when the multiplier is mapped to an ASIC with the STM 90-nm CMOS process. The sampling rate can be increased to 8 GHz if high speed cells from the same library are used. The proposed counter-based multiplier outperforms many serial-serial and serial parallel multipliers in speed but its hybrid architecture does carry an area overhead. Overall, the ADP is comparable to other serial-serial multipliers. Comparing with the 32 x 32 and 64 x 64 parallel CSA array multipliers, the proposed multiplier has comparable speed but is 35% and 58% more area efficient, respectively. In addition, our 64 x 64 multiplier consumes about 31% less energy per operation than its parallel counterpart. Last but not least, the counter-based approach has clear advantage of low I/O requirement and hence is most suitable for complex SoCs, advanced FPGAs and high-speed bit serial applications.

REFERENCES


