A High Bit Rate Serial-Serial Multiplier

B. Ramadevi[1] & J. Karthik[2]

[1]*Dept. Of ECE, Kakatiya institute of technology and science, WGL*
[2]*KITS,KU,Warangal*
E-mail : ramadevikitsw@gmail.com[1], karthik.j205@gmail.com[2]

Abstract **- A design of serial-serial hybrid multiplier is proposed for applications with high data rate. Here the proposed technique effectively forms the entire partial product rows in just n cycles where as conventional serial-serial multipliers take 2n cycles to form all partial products. The conventional way of partial product formation is rearranged here. Here the proposed architecture achieves high data rate by replacing full adders with asynchronous 1's counter so that critical path is limited to only DFF and an AND gate. The use of asynchronous counter reduces the height of the partial product rows from n to $[\log_2 n]+1$, resulting in reduction of complex adder tree. The proposed multiplier consists of serial-serial data accumulation unit followed by a dadda multiplier which reduces the average power dissipation. It has a small delay penalty to complete a multiplication when compared to a conventional parallel array multiplier.**

Keywords - Binary multiplication, parallel multipliers, serial multipliers, on-chip serial link bus architecture, dadda multiplier.

I. INTRODUCTION

Multipliers are the fundamental and essential building blocks of VLSI systems. They are the complex arithmetic operation, which is reflected in its relatively high signal propagation delay, high power dissipation and large area requirement. Often, the delay of multiplication dominates the critical path of the system. So multipliers with low power and high speed are required. There are good multiplier designs require tradeoff between power consumption and speed.

The multiplication can be performed in two ways. They are parallel multipliers and serial multipliers. Parallel multipliers are multiplier which has parallel operands i.e., the entire operand bits is available at a time. Whereas, in serial multipliers the two input operands are given serially i.e., here only single bit is received at the multiplier at a time from each operand.

Typically, multiplier in hardware is implemented in three stages. They are partial product generation (PPs), reduction of partial product terms and the final carry propagation adder. Generally partial products are generated using AND gates. The partial product rows are reduced to two rows using different reduction techniques. Such as carry save adder [3], Wallace [4] and dadda[5]. Now the reduced two partial product rows are given as input to the final carry propagate adders such as rca, carry look ahead adder, sklansky adder and koggestone adder. As the height of the partial product increases linearly with the word length of the multiplier, it affects the subsequent two stages a lot. Therefore it is highly important to reduce the number of partial product rows before performing the reduction stage of the multiplier. This can be achieved using modified booth algorithm [6] where the partial product rows are reduced before the reduction stage. But the drawback of using modified booth algorithm is due to its encoding technique which adds both area and delay over head to the simple partial product generation process. If we go through some other reduction techniques like higher order column compression techniques instead of full adders they are slower and consume more power than full adders.

And also some times to reduce the wiring cost, it is common practice to transmit the data through a high speed serial link [7], [8].

In some ICs the designers try to reduce the number of IO pads in order to reduce the power consumption and silicon area. Therefore efforts are made to design high speed serial interface in order to facilitate on-chip buffering and parallel processing.

Parallel multipliers are popular for their high speed operation but if the word lengths go higher they are often constrained by the hardware cost and power consumption of the applications. In general applications

like dsp, image processing, encryption and decryption techniques the word lengths are of hundreds of bits. Therefore, low-cost serial multipliers are widely used.

Serial multipliers are also found in applications like system-on-chip (SoC) design. In present technology more intellectual property cores are integrated on the Soc, which results in large interconnect area and high power consumption. And also due to more interconnections the routing between one module to another module will be very difficult and also congested. To overcome this problem we are evolved recently to on-chip bus and alternative on-chip serial-link bus structure [9].

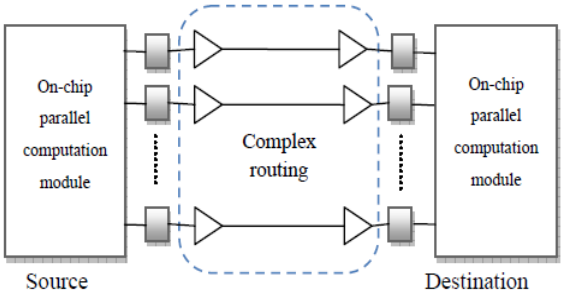

Fig. 1 : Conventional bus structure

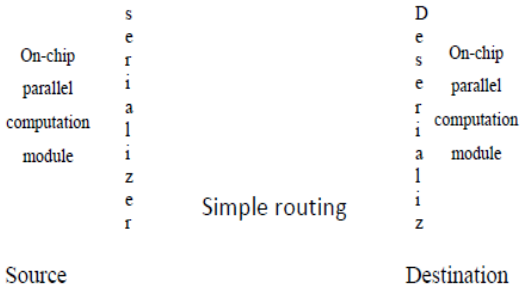

Fig. 2 : Serial-link bus structure

In the above figures (1) and (2) shows the difference between a conventional parallel on-chip bus and an on-chip serial bus, respectively. In figure (2) serializer is used to convert the parallel data stream to a serial bit stream in order to pass through simply routed serial link. At the destination end the deserializer is used to convert the data stream from a serial link again back to parallel data.

So to make computations with normal speed we have to transfer the data through serial link at high speed in order that it should not wait for any data. We can do this by performing some partial computations on the incoming data stream at high speed while data is being buffered.

The rest of the paper is organized as follows section II revises the existing serial-serial multipliers, section III deals with the proposed serial-serial multiplier, section IV gives the implementation results and section V will concludes the paper followed by reference.

II. Existing Serial Multipliers

Generally serial multipliers are of two types they are serial-serial multipliers and serial-parallel multipliers. In serial-serial multipliers both the operands are loaded in bit serial fashion, in order to reduce data input pads two. Where as in serial-parallel multiplier loads one operand in a bit serial fashion and the other in always available in parallel fashion. The existing architectures in serial-serial multipliers have so many disadvantages. The architecture proposed by Manas [1]-[2] use a csa architecture in which the area occupied by the reduction block is very high because of large number of full adders used. It also has large delay due to propagation through adders.

III. Proposed Serila Multiplier

Here in this section we will propose a technique for the generation of individual rows of partial products by considering two serial input operands. In which one operand is starting from LSB and another starts from MSB. These generated partial product rows are given as input to the counter based accumulator which will be explained in this section later. This multiplier will generate all partial product rows and their accumulation will be done in only n cycles for an n×n multiplication.

A. Partial product generation

The product of two n-bit unsigned binary numbers X and Y can be expressed as shown below

$$P = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} xi.yj.2^{i+j} \qquad (1)$$

Where xi and yj are the ith and jth bits of X and Y, respectively.

For the further solution regarding this go through A high bit rate serial-serial multiplier with on-the-fly accumulation by asynchronous counters [1].

The proposed partial product rows schemes will be generated as shown below figure

$$x_7y_0$$
$$x_7y_1 \quad x_6y_1 \quad x_6y_0$$
$$x_7y_2 \quad x_6y_2 \quad x_5y_2 \quad x_5y_1 \quad x_5y_0$$
$$x_7y_3 \quad x_6y_3 \quad x_5y_3 \quad x_4y_3 \quad x_4y_2 \quad x_4y_1 \quad x_4y_0$$
$$x_7y_4 \quad x_6y_4 \quad x_5y_4 \quad x_4y_4 \quad x_3y_4 \quad x_3y_3 \quad x_3y_2 \quad x_3y_1 \quad x_3y_0$$
$$\bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet$$

Fig. 3: Proposed partial product generation

The above figure (3) is shown the generation of partial products for the first five input bits the same will be continued up to n bits.

The following figure will illustrate the generation of partial product terms

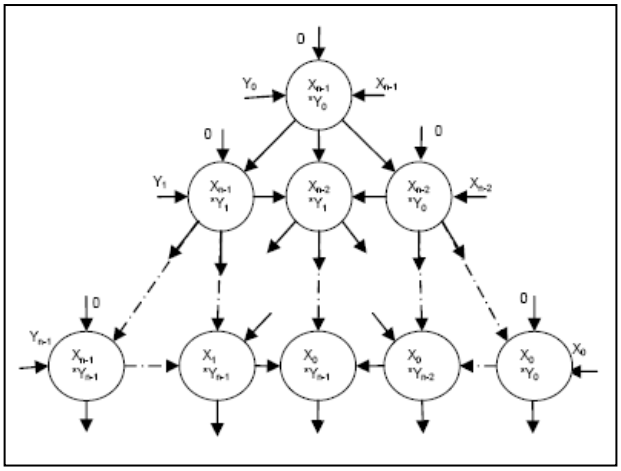

Fig. 4 : Dependency graph for n×n serial-serial multiplication

Here bank of counters are arranged at each column to accumulate the bits arrived at each column. This will be done at counter block shown in figure (7). The figure 4 illustrates the complete operation and generation of partial product for a general n×n multiplication. In the figure 4 each circle (node) represents a binary counter and an ancillary AND gate to generate partial product bit. The below will cover the blocks present in the proposed architecture.

B. Counter block/Accumulation unit

Accumulator is a typical unit in datapath circuits which will add the current input to the value stored in its internal registers. We have so many adders like RCA but the speed will be limited by carry propagation chain. There are so many disadvantages by using normal adders, in adding column wise partial product bits in serial multipliers. They are limited by delays through them so here we are using a counter in order to accumulate the column wise summation value.

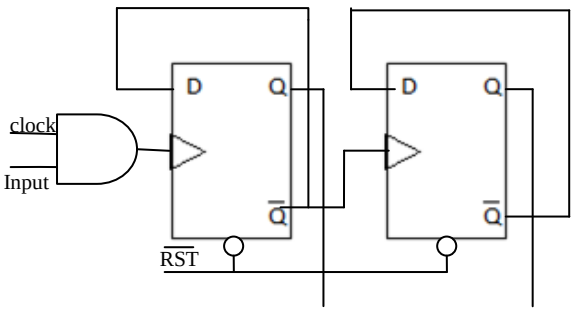

Fig . 5: 2-bit counter architecture

By using these counters we can also reduce the height of the partial product terms which will make easy in further reducing them. The outputs of the counters are arranged in an order such that the LSB bit is placed on the same column position where as the next MSB is positioned to next column toward right. After arranging all the counter outputs regarding their position compress them to final two rows using dadda algorithm.

C. Dadda algorithm

For the reduction of the partial product matrix, dadda [5] proposes a sequence of matrix heights that are determined by working back from the final two-row matrix. In order to implement the minimum number of reduction stages, the height of each intermediate matrix is limited to the least integer that is no more than 1.5 times the height of its successor. Dadda methods dose the minimum reduction necessary at each level to perform the reduction in the same number of levels required by a Wallace multiplier.

The reduction process for a dada multiplier is developed using the following recursive algorithm

1. Let d1=2 and $d_j+1 = [1.5*d_j]$, where d_j is the matrix height for the jth stage from the end. Find the smallest j such that at least one column of the original partial product matrix has more than d_j bits.
2. In the j^{th} stage from the end, employ (3, 2) and (2, 2) counter to obtain a reduced matrix with no more than d_j bits in any column.
3. Let j = j-1 and repeat step 2 until a matrix with only two rows is generated.

Another advantage to utilizing Dadda multipliers is that it utilizes the minimum number of (3, 2) counters (full adders). Therefore, the number of intermediate stages is set in terms of lower bounds: 2→3→4→6→9→. . . .

For Dadda multipliers there are N^2 bits in the original partial product matrix and 4.N-3 bits in the final two row matrix. Since each (3, 2) counter takes three inputs and produces two outputs, the number of bits in the matrix is reduced by one with each applied (3, 2) counter.

The total number of (3,2) counters = $N^2 – 4.N+3$.

The length of the carry propagation adder = 2.N – 2.

The number of (2, 2) counters =N-1.

The dot diagram for a 8 by 8 Dadda multiplier is shown in figure1. Dot diagrams are a useful tool for depicting the placement of (3, 2) and (2, 2) counter in parallel multipliers. Each partial product bit is represented by a dot. The output of each (3, 2) and

(2, 2) counter are represented as two dots connected by a plain diagonal line. The outputs of each (2, 2) counter are represented as two dots connected by a crossed diagonal line.

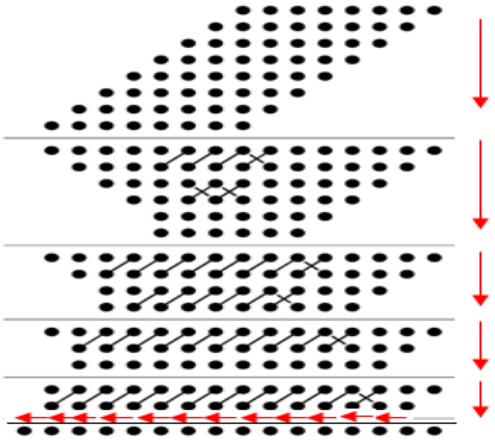

Fig. 6 : Dot diagram for 8 by 8 Dadda Multiplier

The 8 by 8 multiplier takes 4 reduction stages, with matrix height 6, 4, 3 and 2. The reduction uses 35 (3, 2) counters, 7 (2, 2) counters, reduction uses 35 (3, 2) counters, 7 (2, 2) counters, and a 14-bit carry propagate adder. The total delay for the generation of the final product is the sum of one AND gate delay, one (3, 2) counter delay for each of the four reduction stages, and the delay through the final 14-bit carry propagate adder arrive later, which effectively reduces the worst case delay of carry propagate adder.

Now the final output of a dadda multiplier is two rows of partial product rows which is given as input to the final carry adder like RCA. The output of the final carry propagate adder is the required product of two unsigned serial operand.

The final architecture of the proposed serial-serial multiplier is as shown in the below figure.

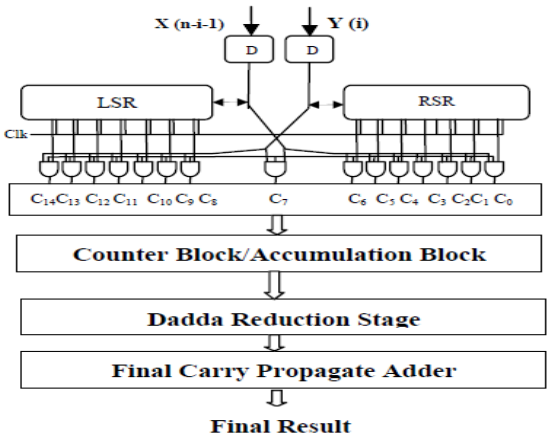

Fig. 7: Proposed 8×8 serial-serial multiplier architecture

IV. Implementation Results

The whole multiplier is implemented using verilog in Xilinx ISE 13.4 and it is targeting Xilinx Virtex-5 xc5vlx20t-2ff323.A testbench is used to generate the stimulus and applies it to the implemented serial-serial multiplier. The serial-serial multiplier code was also checked using Xilinx [10]. The design was synthesized using Xilinx synthesis XST tool [10] . Post synthesis and place and route simulations were made to ensure the design functionality after synthesis and place and route. Table IV shows the resources utilized by it while serial-serial multiplier implemented.

TABLE I : RESULTS OBTAINED

	Proposed serial-serial multiplier
Slice LUT'S Utilized	90
MAX FREQUENCY	870 MHZ

V. Conclusion

In this paper, a different approach of computing serial-serial multiplier is proposed by using asynchronous counters. By using this multiplier we can easily generate all partial rows in just n cycles for n×n multiplication. By using the counters here we are reducing the partial product height logarithmically and make it possible to achieve an effective reduction rate of $\log_2 n$ which will further reduces the area occupied by it. Here by using dadda algorithm in reduction stage we are still able to reduce the area occupied by the full adders and also speed is increased. The counter based approach has advantage of low I/O requirement and hence is most suitable for complex Soc and advance FPGAs.

References

[1] Manas Ranjan Meher, Ching Chuen Jong, and Chip-Hong Chang,A High Bit Rate Serial-Serial Multiplier With On-the-Fly Accumulation by Asynchronous, Digital Object Identifier 10.1109/TVLSI.2010.2060374.

[2] M. R. Meher, C. C. Jong, and C. H. Chang, "High-speed and lowpowerserial accumulator for serial/parallel multiplier," in Proc. IEEE Asia-Pacific Conf. Circuits Syst. (APCCAS), Macau, China, 2008, pp. 176–179.

[3] R.Menon "High performance 5:2 compressor architecture " IEE proc-Circuits Devices Syst., Vol. 153, no.5, pp. 447-452, oct. 2006.

[4] Ron S. Waters, Earl E.Swartzlander "A Reduced Complexity Wallace multiplier reduction" in IEEE transactions on Computers, Aug.

[5] Whytney J. Townsend, Earl E. Swartz, "A Comparison of Dadda and Wallace multiplier delays". Computer Engineering Research Center, The University of Texas.

[6] A. D. Booth, "A signed binary multiplication technique," Quarterly J. Mechan. Appl. Math., vol. 4, no. 2, pp. 236–240, Aug. 1951.

[7] M. Ghoneima, Y. Ismail, M. Khellah, J. Tschanz, and V. De, "Seriallink bus: A low-power on-chip bus architecture," IEEE Trans. Circuit.Syst. I, Reg. Papers, vol. 56, no. 9, pp. 2020–2032, Sep. 2009.

[8] R. Dobkin, M. Moyal, A. Kolodny, and R. Ginosar, "Asynchronous current mode serial communication," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 18, no. 7, pp. 1107–1117, Jul. 2010.

[9] M.Burzio and P,.pellegriono,"seriailizer-paralleliing circuit for high speed digital signals",U.S Patent,Aug.4,1998.

[10] "Xilinx13.4, Synthesis and Simulation Design Guide", UG626 (v13.4) January 19, 2012.

◈◈◈