# Area Efficient and Reduced Pin Count Multipliers

**Omar Nibouche** *o.nibouche@tu.edu.sa*
*College of Computers and Information Technology*
*Taif University*
*POB 888 Taif 21964, KSA*

**Abstract**

Fully serial multipliers can play an important role in the implementation of DSP algorithms in resource-limited chips such as FPGAs; offering area efficient architectures with a reduced pin count and moderate throughput rates. In this paper two structures that implement the fully serial multiplication operation are presented. One significant aspect of the new designs is that they are systolic and require near communication links only. They are superior in speed and area usage to similar architectures in the literature. The paper also present a new fully serial multiplier optimized for area-time$^2$ efficiency with better performance than available architectures in the open literature.

**Keywords:** Educed Pin Count, Serial Multiplication, Area-Time$^2$.

## 1. INTRODUCTION

There is a crucial advantage offered by bit-serial processors over their parallel counterpart, which lies in the very efficient use of chip area. They are particularly suitable for applications that require slow to moderate speeds and in batch mode applications. By contrast, bit-parallel processors are useful for fast speed systems, but at the expense of larger a area usage and thus they are more expensive [1-2].

Traditional bit serial multiplier structures suffer from an inefficient generation of partial products, which leads to hardware overuse and slow speed systems. In this paper, two structures for bit serial multiplication are presented. The first structure, called *structure I*, is the first fully serial multiplier reported in the literature with comparable performance - in terms of speed- to existing serial-parallel multipliers. The second structure, termed *structure II*, requires an extra multiplexer in the clock path; thus making it slower, but has the merit of reducing the latency of the multiplier.

The remainder of the paper is organised as follows: in section 2, the previous work in the literature is reviewed, while section 3 describes the new structures for fully serial multiplication. Section 4 is concerned with an optimisation of the multiplier of *Structure I* in terms of area-time$^2$ efficiency. A comparison of performance is shown in section 5 and conclusions are given in section 6.

## 2. BIT SERIAL MULTIPLIERS: A REVIEW

One of the early bit serial multipliers was proposed by [3]. It generates the partial products in a recursive fashion. Consider the multiplication of two n-bit positive numbers *A* and *B* as follows:

$$A = \sum_{i=0}^{i=n-1} a_i 2^i \tag{1}$$

and

$$B = \sum_{i=0}^{i=n-1} b_i 2^i \tag{2}$$

Let $P_i$ represents the partial product computed after the i[th] bit is fed [3]. $P_i$ is given by:

$$P_i = P_{i-1} + 2^i (a_i B_{i-1} + b_i A_{i-1}) + 2^{2i} a_i b_i \quad (3)$$

where $A_i$ and $B_i$ represent the value of the operands $A$ and $B$, respectively, and by considering only bits from the Least Significant Bit (LSB) to the i[th] bit, that is,

$$A_i = A \bmod 2^{i+1} \quad (4)$$

and

$$B_i = B \bmod 2^{i+1} \quad (5)$$

with the initial values $P_{-1} = A_{-1} = B_{-1} = 0$.

The generation of the partial product, using equation (3), and their assignment to the multiplier cells is shown in Table 1 below:

| Cycle | P | Cell 1 | Cell 2 | Cell 3 | Cell 4 |
|---|---|---|---|---|---|
| 1 | $P_0$ | $a_0b_0$ | | | |
| 2 | $P_1$ | $a_0b_1+a_1b_0$ | $a_1b_1$ | | |
| 3 | $P_2$ | $a_0b_2+a_2b_0$ | $a_1b_2+a_2b_1$ | $a_2b_2$ | |
| 4 | $P_3$ | $a_0b_3+a_3b_3$ | $a_1b_3+a_3b_1$ | $a_2b_3+a_3b_2$ | $a_3b_3$ |
| 5 | $P_4$ | ↙ | ↙ | ↙ | ↙ |
| 6 | $P_5$ | ↙ | ↙ | ↙ | |
| 7 | $P_6$ | ↙ | ↙ | | |
| 8 | $P_7$ | ↙ | | | |

**TABLE 1:** Multiplication Scheme of [3].

From the above table, each cell generates two new bit-products every cycle. To cope with this constraint, The Basic Cell (BC) of the multiplier proposed by [3] is built around a 5 to 3 counter. The counter is capable of accumulating five inputs of the same weight to a sum-bit ($S_{out}$) of the same weight as the inputs, i.e.$2^0$, 1[st] carry-bit ($C^1_{out}$) of a weight of $2^1$ and 2[nd] carry-bit ($C^2_{out}$) of a weight of $2^2$. In particular, the sum-bit is calculated through a tree of EXOR gate to reduce the propagation delay within the cell. The BC of the multiplier is shown in Figure 1. The multiplier uses $n$ identical cells to perform the multiplication of two $n$-bit numbers in $2n$ cycles, as it can be seen in Figure 2.

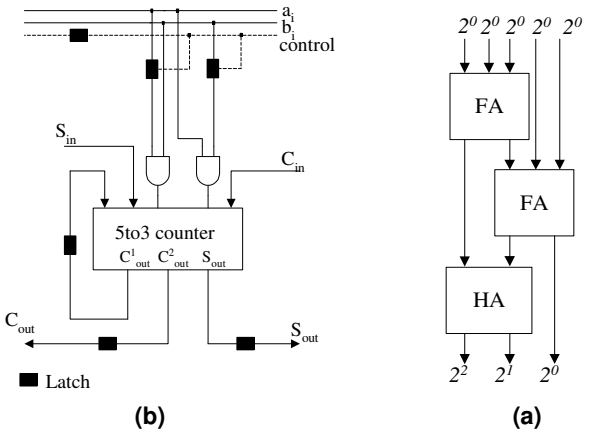


**FIGURE 1 (a):** 5 to 3 Counter Made of Two FAs and One HA **(b).** The BC of [3].

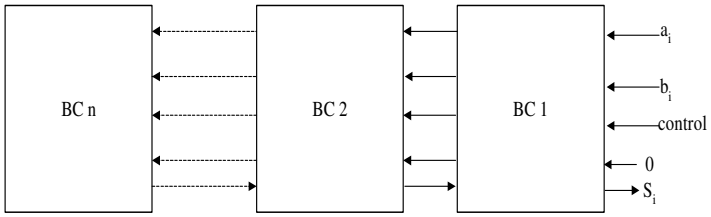


**FIGURE 2:** The n-bit Multiplier by [3].

In [4], modifications were carried out on the multiplication scheme of Table 1 to make a more efficient use of the hardware. In fact, the multiplier proposed by [4] uses only half the number of cells required by [3]. To achieve this, the partial products generated by the last *n/2* cells in [3] were reallocated to the first *n/2* cells and rescheduled at cycle *n+1*. In this way, a full utilisation of cells *1* to *n/2* can be achieved, as it can be shown in Table 2.

| Cycle | P | Cell 1 | Cell 2 |
|---|---|---|---|
| 1 | $P_0$ | $a_0b_0$ | |
| 2 | $P_1$ | $a_0b_1+a_1b_0$ | $a_1b_1$ |
| 3 | $P_2$ | $a_0b_2+a_2b_0$ | $a_1b_2+a_2b_1$ |
| 4 | $P_3$ | $a_0b_3+a_3b_3$ | $a_1b_3+a_3b_1$ |
| 5 | $P_4$ | $a_2b_2$ | $a_3b_3$ |
| 6 | $P_5$ | $a_2b_3+a_3b_2$ | ↙ |
| 7 | $P_6$ | ↙ | |
| 8 | $P_7$ | ↙ | |

**TABLE 2:** Multiplication Scheme by [4].

The multiplier was modified using little extra hardware: two *n/2* shift registers are used to store the *n/2* most significant bits of the data words *A* and *B* and *n/2* multiplexers. The BC of the modified multiplier d is shown in Figure 3. The multiplier structure as proposed by [4] is shown in Figure 4. This architecture has reduced the number of 5 to 3 counters by 50%, as well as reducing the number of latches by 33%. The clock path equals the delay of a multiplexer, a gated counter and a latch.

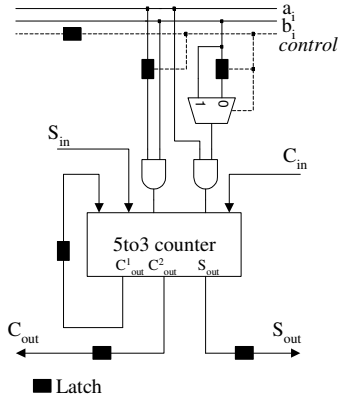


**FIGURE 3:** The BC of [4].

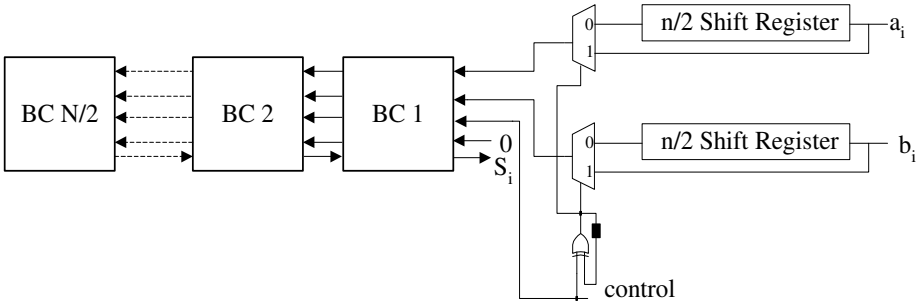


**FIGURE 4:** The n-bit Multiplier by [4].

## 3. THE NEW FULLY SERIAL MULTIPLIERS

Although about 50% of the area used in [3] has been saved by [4], the throughput rate has not been increased. On the contrary, it has decreased as a multiplexer was added to the structure making the clock period of the multiplier in [4] equivalent to the delay of a multiplexer, an AND gate, a 5 to 3 counter and a latch. To remedy this problem, two new structures are proposed.

### 3.1. *Structure I*

In order to reduce the clock path as described above, the multiplication algorithm has been modified. It generates the bit-products associated with cells *2* to *n* at the $(n+1)^{th}$ cycle. The scheduling of the tasks of the first cell is kept unchanged, but the latency of the multiplier is increased to *n* cycles. The multiplication scheme is shown in Table 3 for 4-bit operands. The multiplication operation can be divided into two parts, which can easily be done by rewriting the product of the two numbers, *A* and *B*, in the following way:

$$A * B = a_0 B + \sum_{i=1}^{i=n-1} \sum_{j=0}^{j=n-1} a_i b_j 2^{i+j} \quad (6)$$

To keep working under the constraints of sending one bit of each operand at a time, the term $a_0B$ in equation (6) is generated by the first cell of the multiplier during the first $n$ cycles. At this stage, the other cells only propagate the partial products already generated by the first cell. At the $(n+1)^{th}$ cycle, all the operand bits have been fed and the term $\sum_{i=1}^{i=n-1}\sum_{j=0}^{j=n-1} a_i b_j 2^{i+j}$ can be generated during the last $n$ cycles. The clock period is equivalent to the delay of a FA, an AND gate and a latch, as shown in Figure 5. Therefore, *Structure I* achieves similar speed performances when compared with serial-parallel multipliers.

### 3.2. *Structure II*

The 5 to 3 counters have been widely used in the literature [3-6]. Basically, such a counter reduces 5 bits of the same weight to three bits: a result-bit of the same weight as the inputs (a weight of $2^0$), a carry-bit, which has a weight of $2^1$, and a far carry-bit that has a weight of $2^2$. It is clear that while the sum of the inputs is up to 5, the sum of the outputs is up to 6, and as such, two representations of the outputs are excluded. Therefore, it is clearly more appropriate to reduce the 5 inputs to 3 outputs: a result-bit of the same weight as the inputs, and 2 carry-bits of twice the weight of the result-bit. For this purpose, a new cell has been developed by using two FAs as shown in Figure 6. The first FA is used to accumulate two bit products with a carry feedback. The second FA is used to generate a result-bit from the result of the first FA, the result-bit from the adjacent cell and a carry feedback. The two carry-bits generated by the new cell are fed back and accumulated with the bit-products of the next cycle. The sum-bit of the first FA is registered, and thus making the clock period equivalent to the delay of a multiplexer, an AND gate, a FA and a latch. The multiplier structure implements directly the algorithm shown in Table 2. The multiplier requires only $n/2$ cells for the multiplication of two $n$-bit numbers. It is also modular and needs near communication links only.

**TABLE 3:** *Structure I* Multiplication Scheme For 4-bit Operands.

| Cycle | P | Cell 1 | Cell 2 | Cell 3 | Cell 4 |
|---|---|---|---|---|---|
| 1 | | $a_0b_0$ | | | |
| 2 | | $a_0b_1$ | ↘ | | |
| 3 | | $a_0b_2$ | ↘ | ↘ | |
| 4 | $P_0$ | $a_0b_3$ | ↘ | ↘ | ↘ |
| 5 | $P_1$ | | $a_3b_0$ | $a_2b_0$ | $a_1b_0$ |
| 6 | $P_2$ | | $a_3b_1$ | $a_2b_1$ | $a_1b_1$ |
| 7 | $P_3$ | | $a_3b_2$ | $a_2b_2$ | $a_1b_2$ |
| 8 | $P_4$ | | $a_3b_3$ | $a_2b_3$ | $a_1b_3$ |
| 9 | $P_5$ | | ↘ | ↘ | ↘ |
| 10 | $P_6$ | | | ↘ | ↘ |
| 11 | $P_7$ | | | | ↘ |

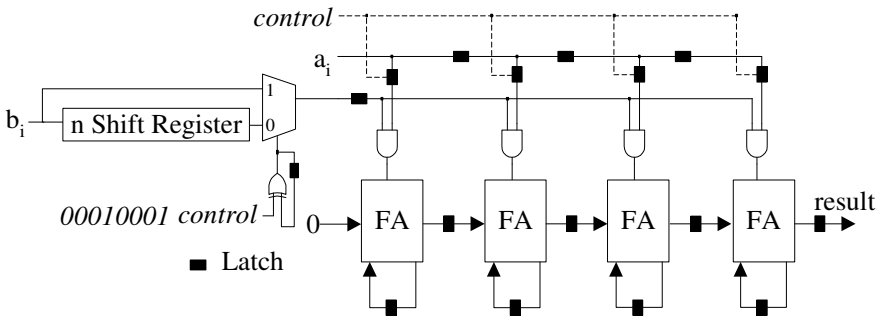


**FIGURE 5:** *Structure I* bit-bit Serial Multiplier.

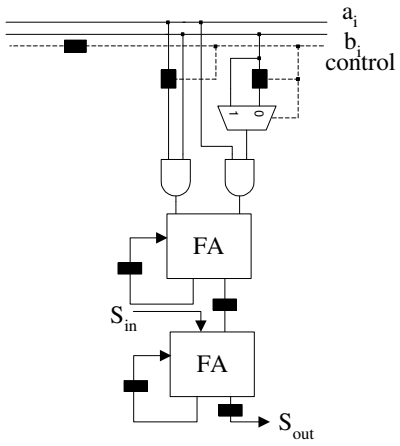


**FIGURE 6:** The Basic Cell of *Structure II* Fully Serial Multiplier

## 4. AREA-TIME$^2$ EFFICIENT BIT-BIT SERIAL MULTIPLIER

In this section, a new multiplier structure, which is capable of multiplexing two multiplication operations into *Structure I* is proposed. This has the merit of doubling the throughput rate at the expense of extra hardware consisting of *2n* multiplexers and *n* latches. By optimizing the multiplier for *area-time*$^2$ efficiency, the problem of lost cycles is circumvented. The lost cycles are the cycles needed for carry propagation once the generation of the partial-products is finished.

The best structure in the literature that can multiplex two multiplication operations into the same multiplier was described in [7]. The algorithm presented in [7] is an improvement made on the multiplication scheme of Table 2. Starting from this multiplication scheme, it reassigns and reschedules the partial products generated by the $(n/4+1)^{th}$ cell and above starting at $(n+n/4)^{th}$ cycle to the cells from *1* to *n/4* at the $(n+n/2)^{th}$ cycle, respectively. This has the effect of freeing *n/4* most significant cells at the $(n+n/4)^{th}$ cycle and thereafter. The multiplication scheme adopted by [7] to achieve this operation is shown in Table 5.

| Cycle | P | Cell 1 | Cell 2 |
|---|---|---|---|
| 1 | $P_0$ | $a_0b_0$ | |
| 2 | $P_1$ | $a_0b_1+a_1b_0$ | $a_1b_1$ |
| 3 | $P_2$ | $a_0b_2+a_2b_0$ | $a_1b_2+a_2b_1$ |
| 4 | $P_3$ | $a_0b_3+a_3b_0$ | $a_{13}+a_3b_1$ |
| 5 | $P_4$ | $a_2b_2$ | ↙ |
| 6 | $P_5$ | $a_2b_3+a_3b_2$ | |
| 7 | $P_6$ | $a_3b_3$ | ↙ |
| 8 | $P_7$ | ↙ | |

**TABLE 5:** Multiplication Scheme of [7].

Although the work presented in [7] can be applied to the multiplier of *Structure II*, it is easier to optimize the multiplier of *Structure I* for area-time$^2$ efficiency. One can clearly observe that the result from the first cell is not accumulated with the bit products of the other cells until the $(n+1)^{th}$ cycle. Therefore, instead of feeding the result of the first cell to its neighbouring cell, it is delayed by *n* cycles using *n* latches before being accumulated with the rest of the partial products, as can be seen in Figure 7. In this way, the first cell is used for *n* cycles to generate the bit-products of the first multiplication operation, then is reinitialised during one cycle before being used to generate the bit-products of the second multiplication operation for a duration of another *n* cycles and so on. The remaining cells operate almost in the same fashion. The key point is that they generate and accumulate the bit products of the first pair of operands only when the first cell has finished producing its bit-products. This operation lasts for *n* cycles before the propagation of the partial results is switched to the multiplexers, which allows the cells to generate and accumulate the bit-product of the second pair of data. Consequently, two multiplication operations can be multiplexed into this multiplier every *2n* cycles. The proposed architecture is depicted in Figure 6.

## 5. COMPARISON OF PERFORMANCE

A performance comparison of the new proposed architectures with similar structures available in the literature [3,4] in terms of area usage and the speed of the multipliers is presented in Table 6. In terms of speed, the multiplier of *Structure I* has a clock period equivalent to the delay of a FA, an AND gate and a latch, and as such operates at faster speeds. Furthermore, the multiplier of *Structure II* has a clock period of a multiplexer, an AND gate, a FA and a latch which makes it faster than the multiplier described in [3]. In terms of area usage, the improvements introduced in [4] on the multiplier of [3] have resulted in saving half the total number of cells. In terms of FPGA area usage and in the case of n-bit operands, *Structure I* is mapped into *5n/2* slices of a virtex-4 FPGA and *Structure II* uses *2n* slices. The multiplier given in [3] is mapped into *5n* slices while the multiplier described in [4] requires *2n* slices. These results clearly show the advantages of the new structures in terms of both speed and area usage.

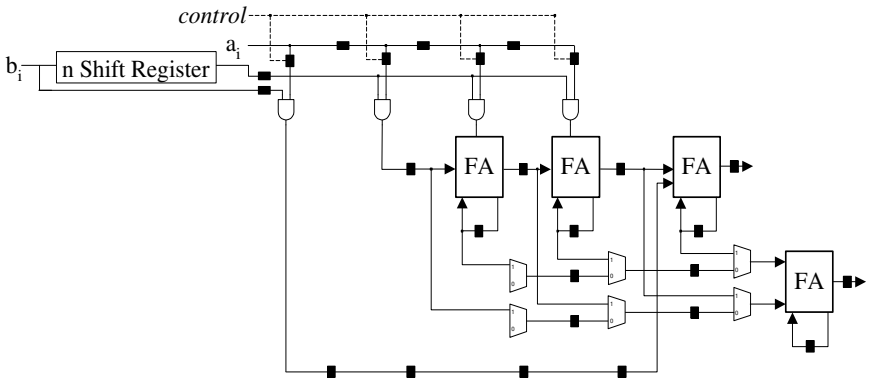


**FIGURE 7:** The New Area-Time$^2$ Efficient Multiplier.

| | Multiplier in [3] | Multiplier in [4] | *Structure I* | *Structure II* |
|---|---|---|---|---|
| Basic Cell | counter + 2 AND gates+ multiplexer + 6 latches | Counter + 2 AND gates + 6 latches + multiplexer | FA + AND gate + 4 latches | 2 Fas + 2 AND gates + multiplexer + 6 latches |
| n-bit multiplier area usage | n BCs | n/2 BCs + n + latches | n BCs + n latches | n/2 BCs + n latches |
| Longest path | AND gate + counter + multiplexer + latch | AND gate + counter + multiplexer + latch | AND gate + FA + latch | AND gate + FA + multiplexer + latch |
| Latency | 1 cycle | 1 cycle | n cycles | 2 cycles |
| n-bit multiplier area usage in FPGA Virxtex-4 | 5n slices | 2n slices | 5n/2 slices | 2n slices |

**TABLE 6:** Performance Comparison.

| | Multiplier in [7] | New multiplier |
|---|---|---|
| Basic Cell | counter + 2 AND gates+ multiplexer + 6 latches | FA + AND gate + 4 latches |
| n-bit multiplier area usage | 3n/4 BCs + 2n latches ≈63n gates (100%) | n BCs + 4n latches + 2n multiplexers ≈66n gates (104%) |
| Longest path | AND gate + counter + multiplexer + latch | AND gate + FA + latch |

**TABLE 8:** Performance Comparison of Area-Time$^2$ Efficient Structures.

Table 8 shows a comparison of performance in terms of hardware usage and speed between the new area-time$^2$ efficient multiplier and the multiplier described in [7]. An estimation of the area usage for both structures made on the number of gates is also shown. It is assumed that the area of the 5 to 3 counter is equal to that of two FAs and a Half Adder, as shown in Figure 1a, which has the same behavior as the 5 to 3 counter. The area usages of both structures are almost similar, but the BCs and the longest path have not been changed. It is worth pointing out the reason behind the choice of the multiplication scheme of Table 4. One may comment that a *"parallel to serial converter"* added to a bit serial-parallel multiplier transforms it to a fully serial multiplier with identical features to those of *Structure I* multiplier. Had this approach been adopted, once the multiplier is optimised for area-time$^2$ efficiency, an extra *n* multiplexers would

have been added to the multiplier. These multiplexers are to be added in the path of the data making its clock path equal to the delay of a multiplexer, a gated FA and a latch; making it slower than the multiplier derived from *Structure I.*

## 6. CONCLUSIONS

In this paper, new structures for reduced pin count multiplication architectures have been presented. These multipliers are systolic and scalable, thus suitable for VLSI implementation. They are both modular and need near communication links only. *Structure I* is the first bit-bit serial multiplier with speed performances similar to existing serial-parallel multipliers. In *Structure II*, the basic cell has been modified to a more appropriate 5 to 3 counter, thus increasing the throughput rate of the multiplier. *Structure I* has been optimised for Area-Time$^2$ efficiency, which has resulted in doubling the throughput rate.

## 7. REFERENCES

[1] K.K. Parhi, "VLSI Digital Signal Processing Systems: Design and Implementation", A Wiley-Interscience Publication, 1999.

[2] A. Aggoun, A. Farwan, M.K. Ibrahim and A.S. Ashur, “Radix-2$^n$ Serial-Serial Multipliers”, IEE proc. Circuits, Devices and Systems, vol. 151, issue 6, pp. 503-509, Dec. 2004.

[3] P. Ienne, and M. Viredaz, “A bit-serial multipliers and squarers”, IEEE Trans. Computer, 1994, 43, (12), pp.1445-1450.

[4] A. Aggoun, A.S. Ashur, and M.K. Ibrahim, “Area-Time efficient serial-serial multipliers”, IEEE International Symp. On Circuits and Systems (ISCAS), pp.V-585-588, GENEVA, May 2000.

[5] N. Strader and V. Rhyne, “A canonical bit-sequential multiplier”, IEEE Trans. Computer, 1982, vol. 31, pp. 691-626.

[6] J. Scanlon and W. Fuchs, “'High-performance bit-serial multiplication”, in Proc. IEEE ICCD'86, Rye Brook, NY, Oct. 1986.

[7] A.S. Ashur, “New Efficient Multiplication Structure and their Applications”. Ph.D. thesis, Dept. of Electrical and Electronic Eng., the University of Nottingham, 1996.