Long Number Bit-Serial Squarers

E. Chaniotakis, P. Kalivas and K. Z. Pekmestzi
are with the National Technical University of Athens, 157 73 Zographou, Athens, Greece.
E-mail: lchaniot, paraskevas, pekmes@microlab.ntua.gr

Abstract

New bit serial squarers for long numbers in LSB first form, are presented in this paper. The first presented scheme is a 50% operational efficient squarer than has the half number of cells compared to the traditional squarers. The second scheme is a 100% operational efficient squarer. In this scheme, the number of the cells remain unchanged compared to other proposed schemes but the number of the required registers is reduced significantly. Both schemes are presented in non-systolic and systolic form and are compared against other squarers presented in the bibliography from the aspect of hardware complexity.

1. Introduction

Bit-serial arithmetic is often used in applications like cryptography, where long numbers computations are required, to reduce the wiring down to a reasonable level. Squaring is a common computation in cryptography algorithms. The widespread use of cryptography today combined with the demand for low power devices justifies design efforts for more hardware efficient implementation of this operation.

A bit-serial squarer for unsigned numbers has been presented in [1]. Another scheme presented in [2] has the same hardware complexity as that in [1] but has the additional advantage of immediate response. A disadvantage of the above schemes is the lack of systolicity. Systolicity is a very important property of bit-serial circuits operating on long numbers because the same data input is broadcasted to many cells. Thus, more hardware is required for driving the broadcasted lines resulting in increased combinational delay. Another squarer scheme based on a new squaring algorithm is presented in [3]. It is of immediate response and its hardware complexity is comparable to the squarer presented in [2] but the application of a technique that merges neighboring cells [4][5] yields systolic architectures.

In this paper, a new squarer is proposed based on the scheme presented in [3]. A technique, presented in [6] for unsigned serial-serial multiplier, is applied in order to reduce by half the number of cells. This technique uses the same cells to compute both the low and high order part of the result. The most significant part of the input word is stored into an external shift register and when the least significant part of the result has been computed the circuit is fed with these stored data to compute the most significant part. The proposed squarer is also presented in systolic form.

A significant drawback of the schemes in [1] and [2] is that they operates with 50% efficiency, namely, a number of zero bits equal to the operand length must be inserted between successive input words to obtain the full product. Twofold efficiency and throughput can be achieved by combining two squarers to operate in parallel. A more hardware efficient technique is presented in [3] and [8]. According to it, at the middle of the multiplication cycle the most significant part of the result is downloaded in carry-save form into a double shift register. The multiplier is free to proceed to the next multiplication while the most significant part is shifted through a bit-serial adder and is obtained in binary form. In [3], the above technique is applied to a squarer to achieve 100% operational efficiency.

The organization of the papers is the following: In Section 2, the squaring algorithm and the squarer of [3] are presented. The proposed 50% efficient scheme is presented in the same section while the proposed 100% efficient scheme is presented in Section 3.

2. The proposed 50% operational efficient squarer

Consider a positive integer X:

$$X = \sum_{i=0}^{w-1} x_i 2^i \qquad (1)$$

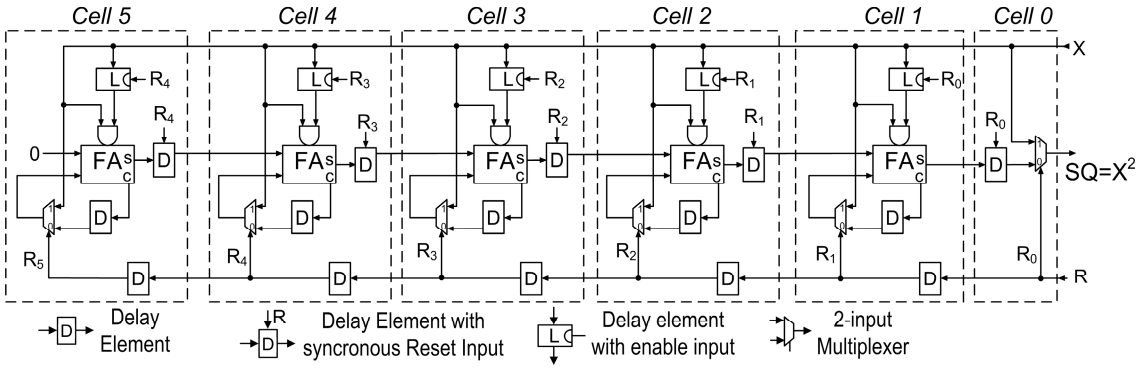

Figure 1. The circuit implementation of the squaring algorithm for 6-bit data word length.

According to the squaring algorithm presented in [3] the square is given by the following equation:

$$X^2 = \sum_{i=0}^{w-1} \{x_i X_i 2 + x_i 2^i\} 2^i \qquad (2)$$

The quantity X_i in the above equation is the number obtained by truncating the $w-i$ most significant bits of X, namely,

$$X_i = \sum_{j=0}^{i-1} x_j 2^j \quad \text{and} \quad X_0 = 0 \qquad (3)$$

A circuit implementation of the above algorithm for 6-bit data word length is shown in Figure 1.

The number of cells is equal to the data word bit-length. The quantity X_i is stored progressively into the row of delay elements with enable input R_i. At the i-th

Table 1. Correspondence between the partial results and the squarer cells.

Cell / Clock	7	6	5	4	3	2	1	0	Result
0								x_0	x_0^2
1							$x_1x_0 + x_1$	0	x_1^2
2						$x_2x_1 + x_2$	x_2x_0	↘	x_2^2
3					$x_3x_2 + x_3$	x_3x_1	x_3x_0	↘	x_3^2
4				$x_4x_3 + x_4$	x_4x_2	x_4x_1	x_4x_0	↘	x_4^2
5			$x_5x_4 + x_5$	x_5x_3	x_5x_2	x_5x_1	x_5x_0	↘	x_5^2
6		$x_6x_5 + x_6$	x_6x_4	x_6x_3	x_6x_2	x_6x_1	x_6x_0	↘	x_6^2
7	$x_7x_6 + x_7$	x_7x_5	x_7x_4	x_7x_3	x_7x_2	x_7x_1	x_7x_0	↘	x_7^2
8		↘	↘	↘	↘	↘	↘	↘	x_8^2
9			↘	↘	↘	↘	↘	↘	x_9^2
10				↘	↘	↘	↘	↘	x_{10}^2
11					↘	↘	↘	↘	x_{11}^2
12						↘	↘	↘	x_{12}^2
13							↘	↘	x_{13}^2
14								↘	x_{14}^2
15									x_{15}^2

Table 2. Correspondence between the partial results and the squarer cells when the half number of cells is used.

Cell / Clock	4	3	2	1	0	Result
0			$\textcircled{x_1}$		x_0	x_0^2
1		$\textcircled{x_2}$		$x_1x_0 + x_1$	0	x_1^2
2	$\textcircled{x_3}$		$x_2x_1 + x_2$	x_2x_0	↘	x_2^2
3		$x_3x_2 + x_3$	x_3x_1	x_3x_0	↘	x_3^2
4	$x_4x_3 + \textcircled{x_4}$	x_4x_2	x_4x_1	x_4x_0	↘	x_4^2
5	x_5x_3	x_5x_2	x_5x_1	x_5x_0	↘	x_5^2
6	x_6x_3	x_6x_2	x_6x_1	x_6x_0	↘	x_6^2
7	x_7x_3	x_7x_2	x_7x_1	x_7x_0	↘	x_7^2
8		↘	$\textcircled{x_5}$ ↘	↘	↘	x_8^2
9		$\textcircled{x_6}$	↘	$x_5x_4 + x_5$	↘	x_9^2
10	$\textcircled{x_7}$		$x_6x_5 + x_6$	x_6x_4	↘	x_{10}^2
11		$x_7x_6 + x_7$	x_7x_5	x_7x_4	↘	x_{11}^2
12			↘	↘	↘	x_{12}^2
13				↘	↘	x_{13}^2
14					↘	x_{14}^2
15						x_{15}^2

clock cycle the quantity x_iX_i2 is added along with the term x_i2^i in the cell-i. The term x_i2^i is added through the carry input of the Full-Adder included in the cell. The right most cell is reduced to a 2-input multiplexer since it adds only bit x_0. The control signal R is a traveling one that enters the circuit at the first clock cycle of each square operation.

Table 1 shows the correspondence between the partial results and the cells of the circuit in Figure 1 that adds these results. In this table we can see that the cells are underutilized because a zero word is inserted be-

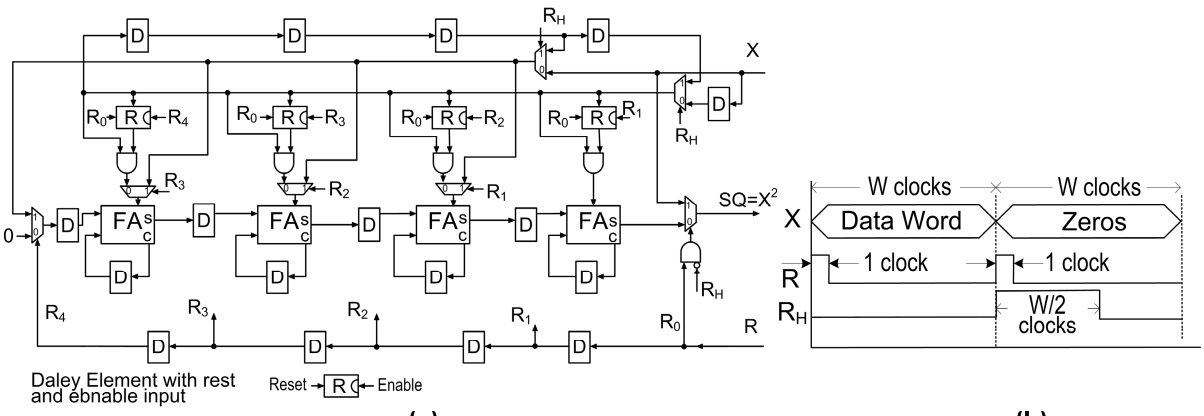

Figure 2. (a) The proposed 50% operational efficient squarer for 8-bit data word length in non-systolic form. (b) The timing of the control signals.

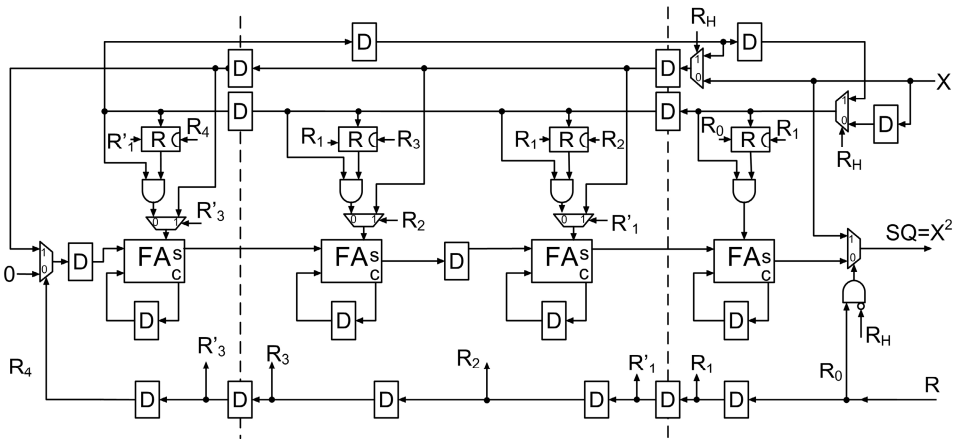

Figure 3. Applying the merging technique to the circuit in Figure 2.

tween successive data words to obtain the most significant part of the result. Table 2 shows that the half number of the cells can be used for computing both the low and high order part of the square by rearranging the partial results. According to this table, only the most significant bits $x_{w-1}\cdots x_{w/2-1}$ of the operand are involved in the computation of the high order part of the result. Thus, these bits must be stored in order to be reused for the computation of the high order part.

The proposed scheme that implements the above idea is shown in Figure 2a. It has only $\frac{w}{2}+1$ cells and the rightmost of them is reduced to a 2-input multiplexer since it adds only the term x_0. During a computational cycle the circuit performs two independent computations. During the first half of this cycle all the bits of X enter the circuit and the low order part of the result is computed. After that, the most significant bits of X are stored into a shift register and reenter the circuit during the computation of the high order part. The delay elements with the enable input must be reset at the beginning and the middle of the computation cycle. The timing of the control signals is shown in Figure 2b. The control signal R is a traveling one that enters the circuit at the beginning and the middle of the computation cycle. The signal R_H allows the reentrance of the most significant part of the input data word.

In the circuit of Figure 1 the term $x_i 2^i$ is added through the carry input of the Full-Adder in each cell. In the circuit of Figure 2 this is not feasible because the carry from the computation of low order part of the result must be added during the computation of the high order part. The solution to this problem adopted here is the term $x_i 2^i$ to be added in the neighboring cell the left a clock cycle earlier. For example, according to Table 2, the term $x_5 2^5$ must be added in cell 1 at clock cycle 9. In the circuit of Figure 2, it is added in cell 2 at clock cycle 8. At clock cycle 8 the cell-2 has not yet start participating in the computation of the high order part and therefore the term $x_5 2^5$ can be added. In Table 2 terms $x_i 2^i$ are in circles to show in which cell and at which clock cycle each of them is added. In the leftmost cell, cell 4 in Fig. 2, where there is no neighboring cell at the left the term $x_4 2^4$ is added through the free Full-Adder input. The above solution requires that both x_i and x_{i+1} are available in every cell.

The proposed circuit can be transformed into systolic form by applying the merging technique. According to this technique two cells are merged into one cell by drawing a vertical retiming cut between them and rearranging the two Full-Adders. Figure 3 shows the first step of merging where the retiming cuts (shown with dashed lines) result in removing delay (shown with dashed boxes) and inserting new ones into the lines of x_i, x_{i+1} and R. Figure 4 shows the final circuit where each merged cell is internally pipelined to avoid increasing the combinational delay.

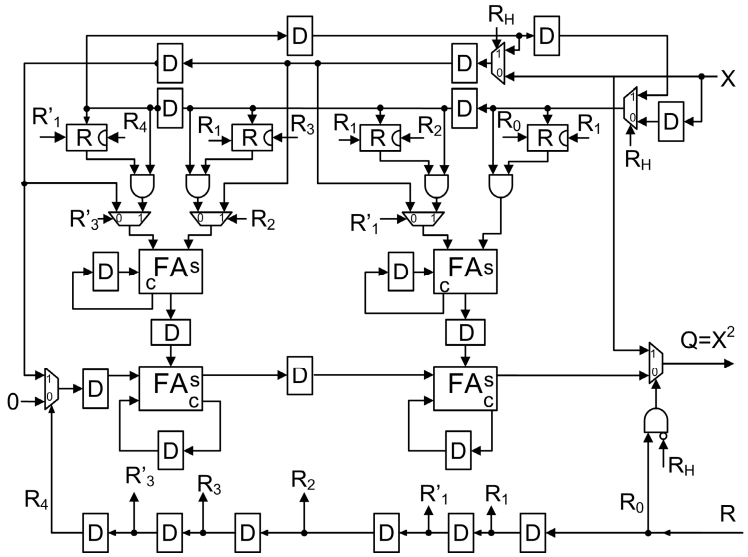

Figure 4. The proposed 50% operational efficient squarer in systolic form.

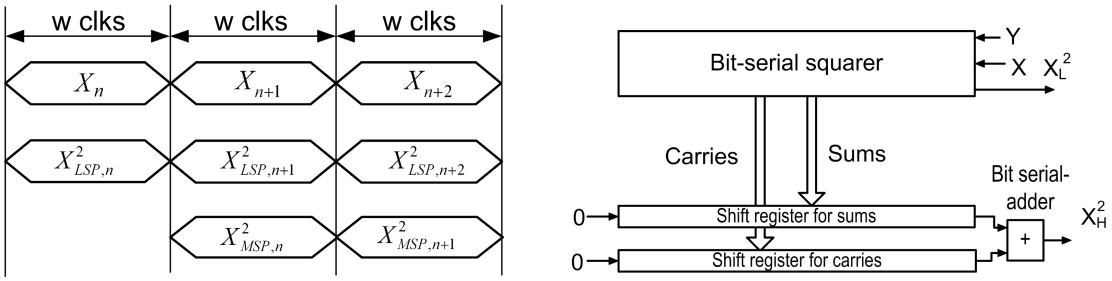

Figure 5. (a) The timing diagram of a 100% operational efficient squarer (b) The block diagram of the 100% operational efficient square presented in [3].

3. The proposed 100% operational efficient squarer

The timing diagram in Figure 5a clarifies the operation of a 100% efficient squarer. The least and most significant parts $X^2_{LSB,n}$ and $X^2_{MSB,n}$ of the n-th square computation are obtained from different outputs. The computations of $X^2_{LSP,n}$ and $X^2_{MSP,n}$ are overlapped and consequently no zero bits are required between successive input words. The squarer presented in this paper is based on a 100% operational efficient squarer presented in [3] which operation is clarified by the block diagram in Figure 5b

The high and low order parts of the result are received from the outputs X^2_H and X^2_L respectively. During the first w clock cycles of the operation the least significant part of $X^2_{LSP,n}$ is obtained in binary form from the output X^2_L. At the end of the w-th clock cycle sum and carry outputs of all the Full-Adders that constitute high order part of the result are downloaded into two shift registers and the squarer is free to start the computation of $X^2_{LSP,n+1}$. After downloading, the shift registers contain the most significant part $X^2_{MSP,n}$ in carry-save form. During the next w cycles, $X^2_{LSP,n+1}$ is received from the output X^2_L of the squarer and $X^2_{MSP,n}$ shifts through the bit-serial adder shown in the figure and is being obtained from the output X^2_H.

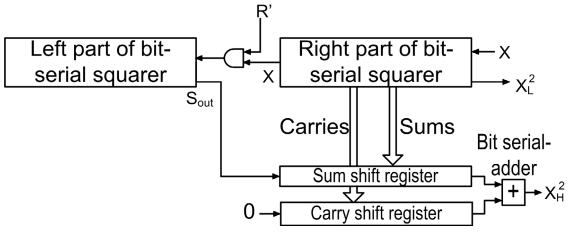

Figure 6. The block diagram of the proposed 100% efficient squarer.

Table 2 shows that when a new squaring starts the cells from $\frac{w}{2}$ to $w-1$ are involved in the computation of the square after $\frac{w}{2}$ cycles. These "idle" cycles can be exploited to empty the left half part of the circuit by inserting $\frac{w}{2}$ zero bits. Consequently the left half part of the circuit does not require the two shift registers.

The block diagram in Figure 6 shows the structure of the proposed 100% operational efficient squarer. The squarer consists of two parts. At the first w clock cycles of the multiplication the quantity $X^2_{LSP,n}$ is obtained from the output X^2_L. At w-th cycle the carry and sums of the Full-Adder in the right part are downloaded into the two shift registers. During the next $\frac{w}{2}$ clock cycles the multiplier operates as following:

The right part outputs the $\frac{w}{2}$ lower order bits of $X^2_{LSP,n+1}$ at X^2_L. The left part produces the $\frac{w}{2}$ higher order bits of $X^2_{MSP,n}$ and pass them to the sum shift register. The content of both shift registers is shifted through the bit-serial adder producing the $\frac{w}{2}$ lower order bits of $X^2_{MSP,n}$ at X^2_H. During those $\frac{w}{2}$ clock cycles the control signal R' isolates the left part from the new values of x that enter the squarer. During the last $\frac{w}{2}$ clock cycles the $\frac{w}{2}$ most significant bits of $X^2_{LSP,n+1}$ and the $X^2_{MSP,n}$ and are produced. A circuit implementation of the above scheme is shown in Figure 7.

The control signal Q is activated at the w-th clock cycle of the squaring operation and downloads the carries and sums of the right stage and sets the JK flip-flop which in turn activates the signal R'. The signal R' is deactivated after $\frac{w}{2}$ clocks.

Applying the merging technique as in the proposed squarer in Figure 4 we obtained the proposed 100% operational efficient squarer in systolic form shown in Figure 8. The retiming inserts delay elements into the line of the control signal R that is used for downloading the high order part of the result. Thus, Q is a traveling one that enters the squarer at the first clock of each

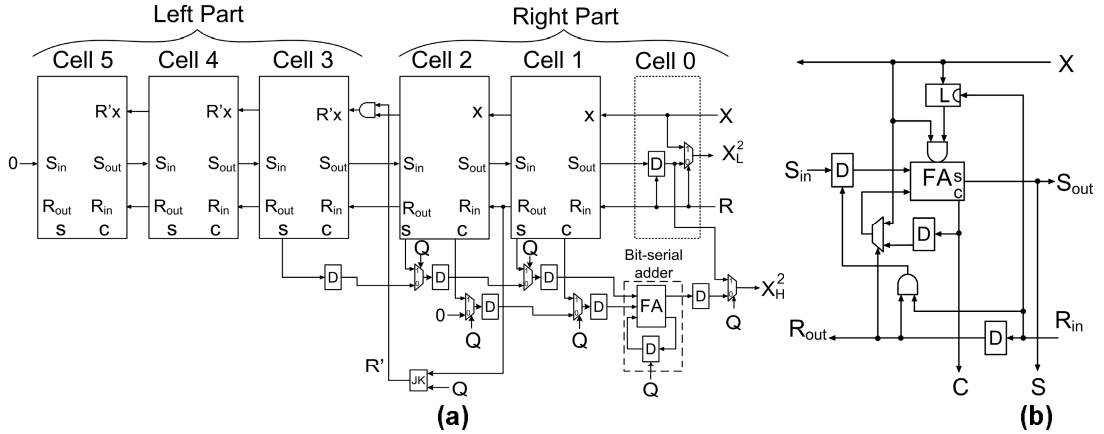

Figure 7. (a) The circuit implementation of the proposed 100% efficient squarer for 6-bit data word length. (b) The circuit of the cell.

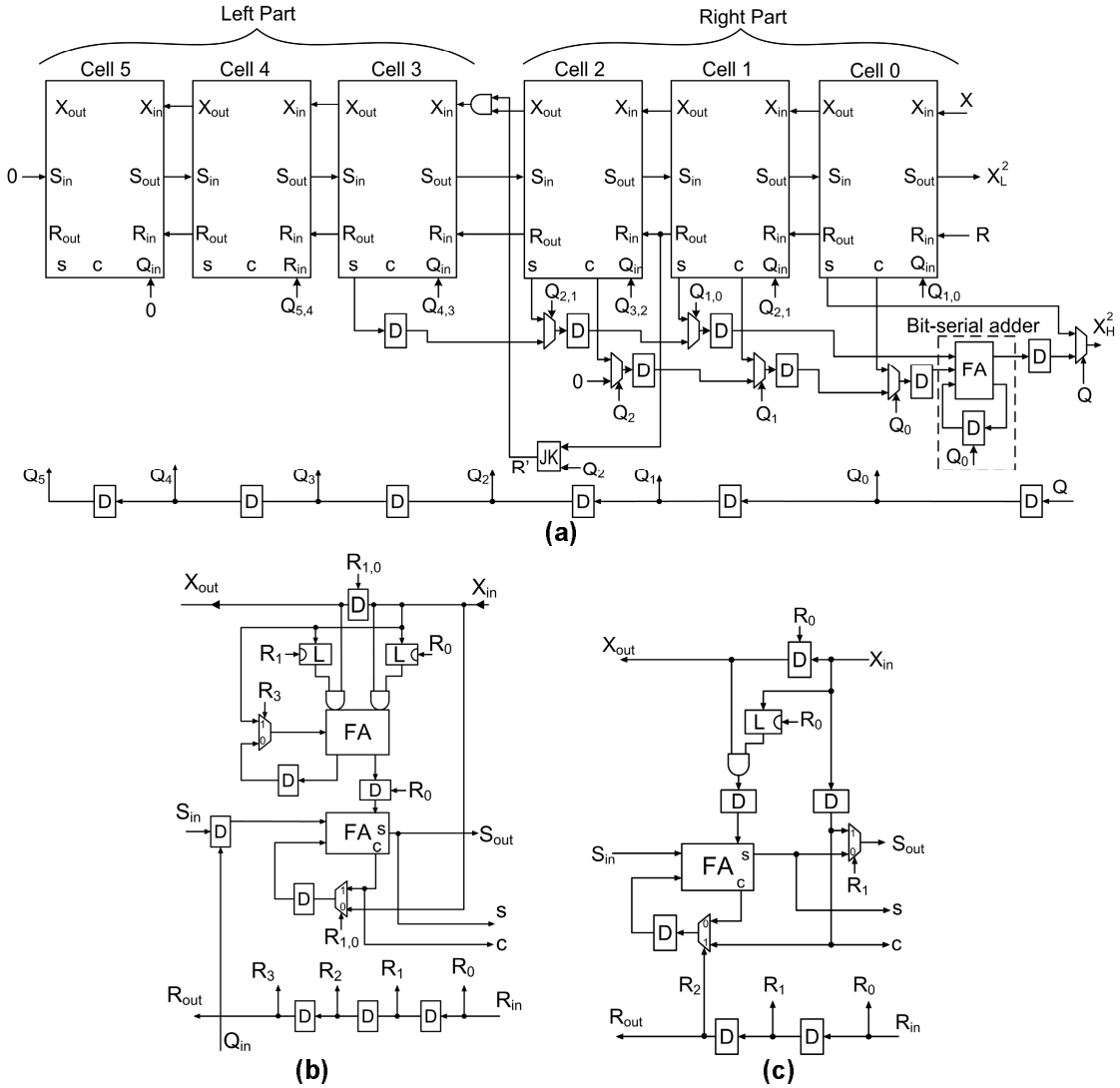

Figure 8. (a) The circuit implementation of the proposed 100% efficient squarer in systolic form for 12-bit data word length (b) The circuit of the cell (c) The circuit of the rightmost cell (cell 0).

operation and the sums and carries in each cell are downloaded progressively.

In the merged cell, we have to download two sums and two carries from the two Full- Adders. Due to the fact that the cell is internally pipelined two additional clock cycles are required for downloading. These cycles are created by inserting two zero bits between successive input data words. The detailed description of the control signals is given in [3] taking into account that the right part of the circuit is the same as the circuit presented in [3].

4. Comparison of the proposed squarers with other schemes

In order to evaluate the efficiency of the proposed schemes we compare them with the squarer presented in [2] and [3]. The results are presented in Table 3. All the schemes compared in this table have the same combinational delay. Therefore, only the hardware complexity is included in the table.

As far the 50% operational efficiency is concerned, Table 3 reveals that the proposed systolic and non-systolic squarers requires about 45% and 35% less

Table 3. Comparison between the proposed scheme and the squarers presented in [2] and [3].

Multiplier	Total hardware Complexity for input data word length equal to w	Hardware complexity in transistors for w=16
Squarer presented in [2].	$4(w-1)D^*+(w-1)FA+(w-1)S+wAND_2$	1234
Squarer presented in [3] (Figure1).	$wD^*+2(w-1)D+(w-1)DE+(w-1)FA+wS+(w-1)AND_2$	1128
Proposed 50% efficient non-systolic squarer (Figure 2a).	$\frac{w}{2}D^*+\frac{3w}{2}D+\frac{w}{2}DE+\frac{w}{2}FA+\left(\frac{w}{2}+3\right)S+(w+1)AND_2$	710
Proposed 50% efficient systolic squarer (Figure 4).	$\frac{w}{2}D^*+\left(\frac{10}{4}w+1\right)D+\frac{w}{2}DE+\frac{w}{2}FA+\left(\frac{w}{2}+2\right)S+(w+1)AND_2$	838
100% efficient non-systolic squarer presented in [3]	$(w-1)D^*+(4w-3)D+(w-1)DE+wFA+(3w-1)S+2(w-1)AND_2$	1634
Proposed 100% efficient non-systolic squarer (Figure 7).	$(w+1)D^*+(3w+1)D+(w-1)DE+wFA+(2w-1)S+(2w-1)AND_2+JK$	1484
100% operational efficient systolic squarer presented in [3]	$\left(\frac{3w}{2}+1\right)D^*+(4w-1)D+(w-1)DE+wFA+(2w-1)S+(2w-3)AND_2$	1670
Proposed 100% efficient systolic square (Fig. 8).	$\left(\frac{3w}{2}-1\right)D^*+\left(\frac{7w}{2}-4\right)D+(w-1)DE+2wFA+\frac{3w}{2}S+(2w-3)AND_2+JK$	1536

D:1-bit register (8 transistors) D:1-bit register with synchronous reset input (12 transistors), DE:1-bit register with enable input (12 transistors), S:2 input switch (6 transistors) FA: Full-Adder (16 transistors) AND_2: 2-input AND gate (4 transistors) (6 transistors) JK:JK flip-flop (14 transistors) [9]*

hardware than the corresponding squarers in [2] and [3] respectively. As far the 100% operational efficiency is concerned the proposed systolic and non-systolic scheme requires 10% less hardware than the corresponding schemes in [3].

5. Conclusion

New bit-serial squarers for unsigned numbers are presented in this paper. For their implementation two techniques are used which lead to hardware reduction. According to the first of them only the half number of squarer cells is required because they are reused for the computation of the low and the high order part of the result. This technique leads to very hardware efficient circuits.

The second technique is applied to 100% operational efficient squares and reduces by half the length of the double shift register that is used for downloading the most significant part of the result. This technique can lead to significant hardware reduction in case of long number squarers.

6. References

[1] L. Dadda "Squares for Binary Numbers in Serial Form" *7th IEEE Symposium on Computer Arithmetic Proceedings*, June 1985, pp. 173-180

[2] Paolo Ienne and Marc A. Viredaz "Bit-Serial Multipliers and Squarers" *IEEE Transactions on Computers*, v.43 n.12, December 1994, pp.1445-1450.

[3] K.Z. Pekmestzi, P. Kalivas and N. Moshopoulos, "Long unsigned number systolic serial multipliers and squarers", *IEEE Circuits and Systems II:*, vol. 48, no 3, Mar. 2001, pp. 316 -321.

[4] A.S. Ashur, M.K Ibrahim, and A. Aggoun, "Systolic digit-serial multiplier," *IEE Proc.- Circuits Devices & Systems*, vol. 143, Feb 1996, pp. 14-20.

[5] C. Caraiscos, K. Z. Pekmestzi, "A class of systolic bit-serial multipliers," *International Journal of Electronics*, vol. 76, 1994, no 3, pp. 463-468.

[6] A. Aggoun, A. Ashur and M.K. Ibrahim,"Area-time efficient serial-serial multipliers", *ISCAS 2000-IEEE International Symposium on Circuits and Systems*, May 28-31,2000, Geneva, Switzerland. pp. 585-588.

[7] G. Even, "Two's complement pipeline multipliers," *Integration, the VLSI journal*, no 22, 1997, pp. 23-38.

[8] K.Z. Pekmestzi and P. Kalivas, "Constant Number Serial Pipeline Multipliers" *The Journal of VLSI Signal Processing-Systems for Signal, Image, and Video Technology, Kluwer Academic Publishers*, no. 26, November 2000, pp. 361-368.

[9] N. Waste, K. Eshraghian, *Principles of CMOS VLSI design*, Reading, MA: Addison- Wesley, 1994.