Proceedings of the 8th IEEE Mediterranean Conference
on Control and Automation (MED 2000)
Rio, Patras, GREECE 17-19 July, 2000

AN EFFICIENT SERIAL PIPELINED IMPLEMENTATION OF $ax+by$

KIAMAL. Z. PEKMESTZI†, PARASKEVAS KALIVAS†, NIKOS MOSHOPOULOS†, IOANNIS SIFNAIOS†

† National Technical University of Athens, Department of Electrical and Computer Engineering, Heroon Polytehniou 9, Athens, Greece, pekmes@microlab.ntua.gr.

Abstract. An efficient implementation of the operation $ax+by$, on which the construction of second order digital filters and complex numbers multipliers is based, is presented. The quantities x and y are 2's complement numbers in serial form. The numbers a and b are constant coefficients in Canonical Signed Digit form (CSD) while the result is obtained in serial 2's complement form. The proposed scheme operates in pipeline mode with 100% hardware efficiency, namely, no sign extension words between successive data words are required. The implementation is based on merging of two serial multipliers, which yields significant hardware reduction.

Key Words. Serial multipliers, Systolic multipliers, Digital filters

1. INTRODUCTION

The computation of the expression $ax+by$, where a and b are constant numbers, is essential for many applications such as digital filters of constant coefficients and complex multipliers of constant coefficients. An obvious implementation of this expression contains two serial/parallel multipliers. The selection of the appropriate multiplier affects the efficiency of the resulting circuit. The serial/parallel multiplier [5], is the most prevalent architecture for implementing serial multiplications, but has the serious disadvantage that additional idle cycles are needed for every multiplication to be completed. It requires the interleaving of zero words between successive data words. Thus, the operational efficiency of the circuit drops to 50%. The systolic serial multiplier presented by R. F. Lyon in [4] operates with 100% efficiency but truncates the result and has increased latency and hardware complexity. Another serial scheme presented in [1] eliminates this latency at the expense of significantly increased circuit complexity producing the full result with 50% efficiency. An alternative architecture presented in [3], produces the full product with no latency, operates with 100% efficiency but requires extra hardware and is not systolic. According to this architecture, the least and most significant parts of the result are generated in two overlapped stages.

In this paper, we merge two serial/parallel multipliers and present a new design for the implementation of $ax+by$, where x, y are two's complement numbers, and a , b are constant numbers in CSD form. This design is based on the serial/parallel pipeline architecture but it operates with 100% efficiency. Finally, we exploit the special features of CSD representation and convert the above scheme in systolic form.

Specifically, in Section 2 we present an algorithm for the multiplication of two's complement with CSD numbers. A detailed design of the circuit is given in Section 3. Finally, a comparison is given between the proposed scheme and a circuit consisted of two separate serial multipliers for two's complement numbers.

2. DESCRIPTION OF THE ALGORITHM

Let us consider the multiplication of a two's complement number X

$$X = -x_{n-1} \cdot 2^{n-1} + \sum_{j=0}^{n-2} x_j \cdot 2^j \qquad (1)$$

with a constant coefficient a in CSD form, which is

$$a = \sum_{i=0}^{m-1} a_i \cdot 2^i, \; a_i = 0, \pm 1 \cdot \tag{2}$$

The multiplication can be expressed as

$$P = X \cdot a = \sum_{i=0}^{m-1} P_i \cdot 2^i \tag{3}$$

where

$$P_i = X \cdot a_i = -x_{n-1} \cdot a_i \cdot 2^{n-1} + \sum_{j=0}^{n-2} x_j \cdot a_i \cdot 2^j \tag{4}$$

The term P_i is the partial product corresponding to the a_i coefficient digit. At the bit level, the partial products can be replaced by the next relation

$$x_j a_i = -s_i + (x_j \oplus s_i) \cdot \boldsymbol{a}_i \tag{5}$$

and $$-x_{n-1} a_i = -\bar{s}_i + (\bar{x}_{n-1} \oplus s_i) \cdot \boldsymbol{a}_i \tag{6}$$

where $\boldsymbol{a}_i$ and s_i represent the absolute value and the sign of a_i digit respectively. Specifically, the value of s_i is assumed to be 1 for $a_i < 0$ and 0 otherwise. The symbol $\oplus$ represents the XOR operator. The results of the inversion and the XOR operator are used as arithmetic quantities in the arithmetic expressions. In (5) and (6) the quantity $x_j \oplus s_i$ implies the inversion of x_j, if a_i is negative and the factor $\boldsymbol{a}_i$ expresses the partial product elimination, if a_i is zero. Using these equations the partial product can be rewritten as follows:

$$P_i = -\bar{s}_i \cdot 2^{n-1} - \sum_{j=0}^{n-2} s_i \cdot 2^j + P_i^* \tag{7}$$

where

$$P_i^* = (\bar{x}_{n-1} \oplus s_i) \cdot \boldsymbol{a}_i \cdot 2^{n-1} + \sum_{j=0}^{n-2} (x_j \oplus s_i) \cdot \boldsymbol{a}_i \cdot 2^j \cdot \tag{8}$$

It holds that

$-\bar{s}_{n-1} \cdot 2^{n-1} - \sum_{j=0}^{n-2} s_j \cdot 2^j = -\boldsymbol{a}_i \cdot 2^{n-1} + s_i$, since $\bar{s}_i + s_i = \boldsymbol{a}_i$ and thus P_i can be written as

$$P_i = -\boldsymbol{a}_i \cdot 2^{n-1} + s_i + P_i^* \cdot \tag{9}$$

This relation implies that the partial products can be manipulated as positive quantities by inverting the bits of X, when a_i is negative, except for the MSB that is inverted when a_i is positive, and by adding the quantity $-\boldsymbol{a}_i \cdot 2^{n-1} + s_i$. By applying (9) in (3) we obtain

$$P = -\sum_{i=0}^{m-1} \boldsymbol{a}_i \cdot 2^{n+i-1} + S + \sum_{i=0}^{m-1} P_i^* \cdot 2^i \tag{10}$$

where $S = \sum_{i=0}^{m-1} s_i \cdot 2^i$

By replacing the first term of (10) with its equivalent in two's complement form, we obtain

$$P = -2^{n+m-1} + \sum_{i=0}^{m-1} \bar{\boldsymbol{a}}_i \cdot 2^{n+i-1} + 2^{n-1} + S + \sum_{i=0}^{m-1} P_i^* \cdot 2^i \tag{11}$$

which is the product in two's complement form. This equation implies that for the implementation of the above two's complement multiplication, the number $\bar{x}_{n-1} x_{n-2} \cdots x_1 x_0$ properly weighted, must be added inverted or not whether a_i is positive or negative respectively. Additionally, the correction term

$$C = -2^{n+m-1} + \sum_{i=0}^{m-1} \bar{\boldsymbol{a}}_i \cdot 2^{n-1+i} + 2^{n-1} + S \tag{12}$$

that depends only on a must be included.

We apply (11) for the computation of $ax + by$ and we obtain the following result

$$ax + by = C_{ab} + \sum_{i=0}^{m-1} [P_i^*(x,a) + P_i^*(x,b)] \cdot 2^i \tag{13}$$

where $P_i^*(x,a)$, $P_i^*(x,b)$ represent P_i^* for ax, bx respectively, and

$$C_{ab} = -2^{n+m} + \sum_{i=0}^{m-1} (\bar{\boldsymbol{a}}_i + \bar{\boldsymbol{b}}_i) \cdot 2^{n-1+i} + 2^n + S_{ab} \tag{14}$$

where $S_{ab} = \sum_{i=0}^{m-1} (s_{a_i} + s_{b_i}) \cdot 2^i$ and s_{a_i}, s_{b_i} correspond to a_i and b_i coefficient bits respectively.

By applying the relation $2^{n-1} = 2^{n+m-1} + \sum_{i=0}^{m-1} -2^{n-1+i}$ in (14) we obtain

$$C_{ab} = -2^{n+m-1} + \sum_{i=0}^{m-1} (\bar{\boldsymbol{a}}_i + \bar{\boldsymbol{b}}_i - 1) \cdot 2^{n-1+i} + 2^{n-1} + S_{ab} \tag{15}$$

According to this equation, this correction term is almost the same as previously. In the empty positions, where both coefficient bits are zero, the corresponding digit of the correction term is 1. In the positions, where only one of the coefficient bits is zero, the correction term digit is 0. In the positions, where both coefficient bits are non-zero, the correction term digit is $\bar{1}$. In the latter case, the next position to the

left is always empty and consequently, the corresponding correction term digit is 1. These consecutive digits $1\bar{1}$ can be replaced by 01.

3. CIRCUIT IMPLEMENTATION

First, a single serial-parallel multiplier, which operates with 100% efficiency, is presented. Next, two such multipliers are merged in order the expression $ax+by$ to be implemented. A multiplier for the coefficient $01010\bar{1}$ is shown in Fig. 1. The multiplier coefficient enters the circuit serially through the X input while the multiplicand enters in parallel form through the lines a_i. The least significant part of the result is obtained from the upper output line P_L. The most significant part of the result is obtained from the lower output line P_U. Empty multiplier cells, namely cells without Full-Adder, correspond to zero coefficient bits. For the 100% operational efficiency of the circuit, a double shift register is used. When the computation of the least significant part of the result is completed, the sum and carry outputs of each multiplier cell are loaded into a double shift register. Thus, the double register contains the most significant part of the product in carry-save form, and feeds a serial adder, which produce the final result in 2's complement form. A control signal *R* activates the switches for loading when the LSB of a new *X* enters the circuit.

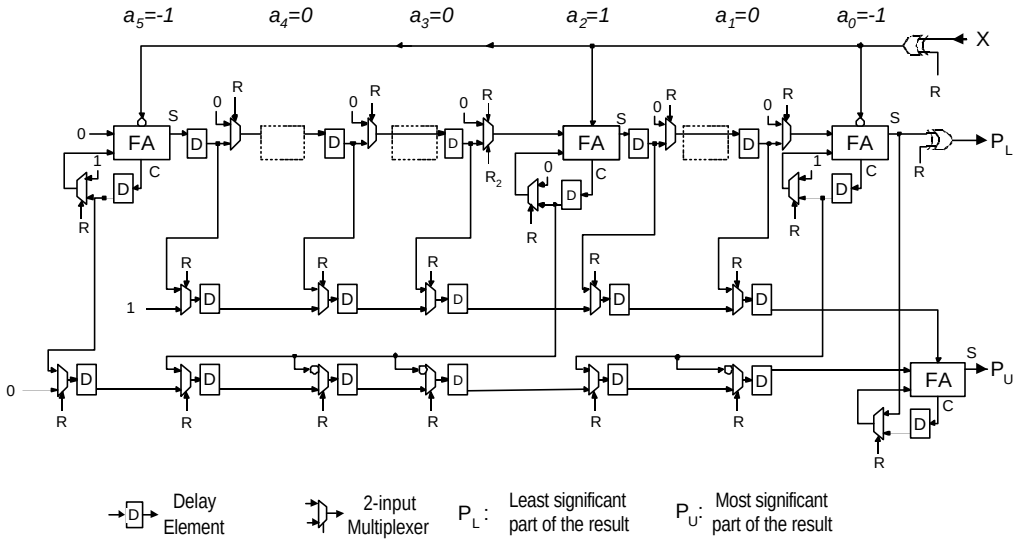

Fig.1. A serial/parallel multiplier for the coefficient $\bar{1}00101\bar{1}$ operating with 100% efficiency

In the above scheme, the correction term given by (12) is incorporated. The part *S* is added by inverting the *X* input and initializing the carry input of each Full-Adder that corresponds to a negative coefficient bit with '1'. The terms that remain are added when the most significant part of the product is loaded in carry-save form into the shift registers. This is achieved, by exploiting the delay elements of the carry shift register that are not used for carry loading. Specifically, for every sequence of *k* empty cells, the corresponding delay elements of the carry shift register are not used for loading except the right most, where the carry of the next right non-empty cell is loaded. For the addition of the corresponding sequence of *k* bits of the correction term, which are all '1', the following expression is applied

$$\sum_{i=0}^{k-1} 2^i + c = c \cdot 2^k + \sum_{i=0}^{k-1} \overline{c} \cdot 2^i \qquad (16)$$

where *c* is the carry loaded at the rightmost position. According to the above equation, the carry *c* must be extended leftwards and inverted. The term $c \cdot 2^k$ is added at the next left position of the sequence, which is unused. The term 2^{n-1} is added by inverting the last bit of P_L and using its non-inverted value as the initial value for the carry of the serial adder that produces the P_U. The term -2^{n+m-1} is added by entering serially a '1' into the left end of the carry shift register.

Two multipliers like the one described above are merged for the implementation of $ax+by$. This is shown in Fig.2 for the coefficients $a=01010\bar{1}$ and $b=\bar{1}00\bar{1}01$. As shown in the figure, the result of the merging is a single multiplier as the one described above with different cells that correspond to two non-zero coefficient digits of the same order. These cells include two Full-Adders and consequently the combinational delay of the circuit is doubled. These cells are referred as "double cells" for the rest of the explanation.

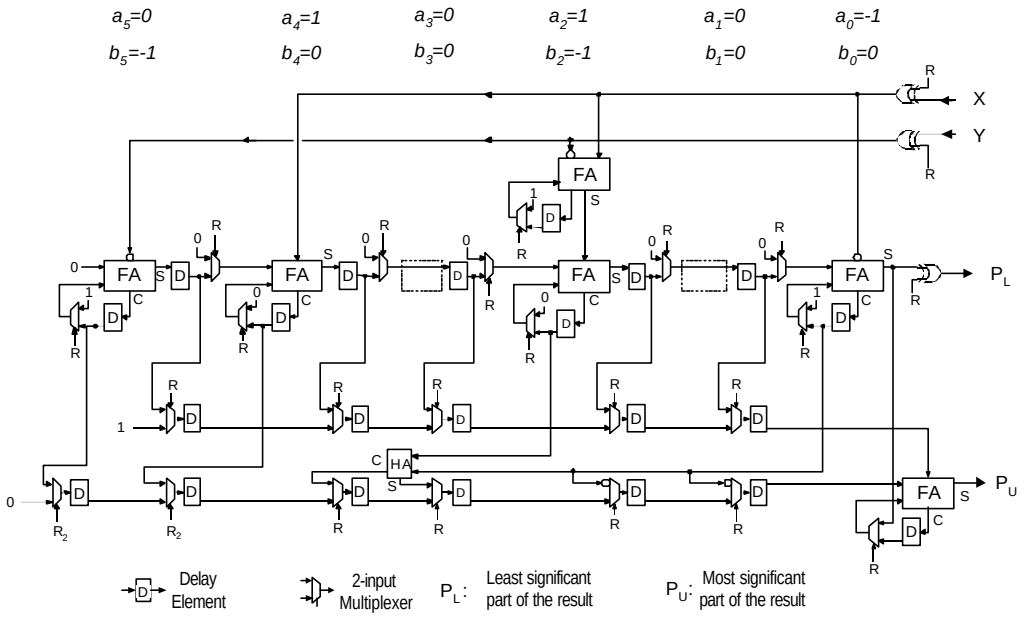

Fig. 2. The implementation of $ax+by$ for $a = 010101\bar{1}$ and $b = \bar{1}00\bar{1}01$.

A delay rearrangement [2,3,6] based on a simple graph property is applied for the internal pipelining of the double cells. According to this property, if we consider all lines that are intersected by a cut across a graph, we can remove one delay element from all lines that have the same direction and insert a delay element into the remaining lines with the opposite direction. The resulting circuit is shown in Fig. 3. The cut for the double cell is also shown in this figure. The result of the rearrangement is a delay element to be removed from the sum line and the shift registers, and a new one to be inserted into lines *X*, *Y*, the control signal *R* and between the sum line that connects the two Full Adders. The loading of the sum and carry lasts two consecutive cycles, in order the timing gap caused by the removed delay elements to be covered. Additionally, the insertion of a delay element into the sum line that connects the two Full-Adders in the double cells has as consequence the need for an extra clock cycle in order the carry of the upper Full-Adder to be loaded into the shift register. Thus, a zero bit must be inserted between the data words that enter through *X* and *Y* lines.

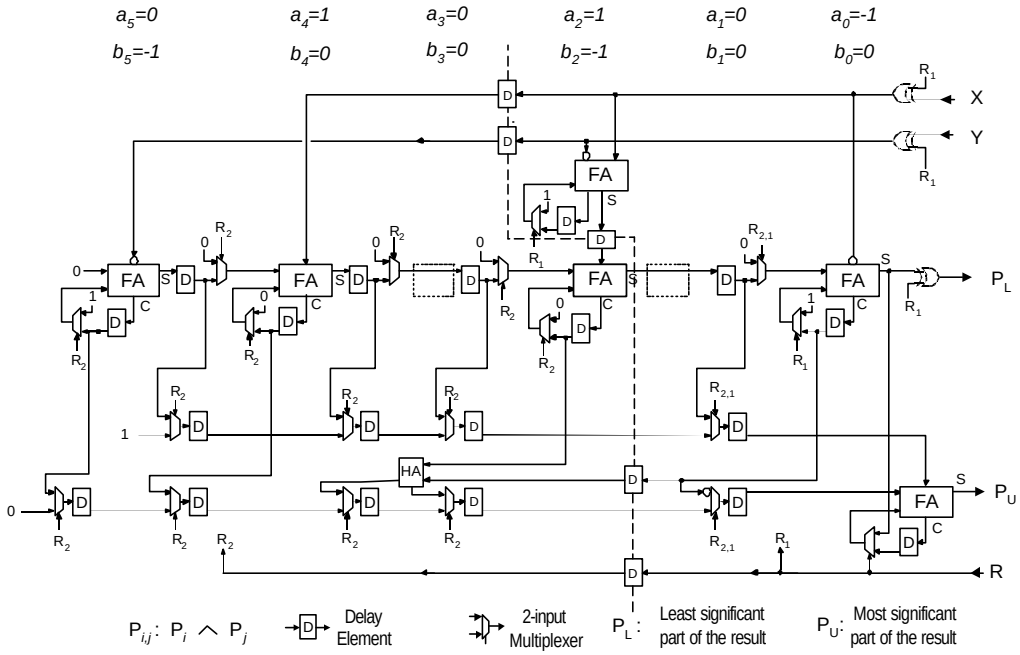

Fig. 3. The circuit of Fig.2 with the delay rearrangement.

The correction term as given by (15) is basically the same as that for the single multiplier. Therefore, they are incorporated in the circuit in the same way. The only difference is in the double cells, where the corresponding correction term digit is 1, while the corresponding digit of the next left empty cell is 0. Thus, the carry bit that comes from the adjacent right empty cell must be added with the carry of the double cell.

In Table 1, the proposed scheme is compared from the aspect of hardware complexity against other schemes where two separate multipliers for constant coefficients are used. For the computation of the hardware complexity, we have assumed the average case, where the number of the non-zero bits in a CSD number is $\frac{1}{3}m$, where m is the length of the number. Consequently, the empty cells in the merged scheme are $\frac{4}{9}m$. Half of them are adjacent to $\frac{1}{9}m$ double cells. Therefore a Half-Adder must be inserted as shown in Fig. 3.

Another consequence of the previously described transformation is the decrease in the broadcasting of lines *X*, *Y* and *R*. Further reduction of this broadcasting can be achieved by applying the previously mentioned graph property for the empty cells. Thus, the circuit is transformed into a systolic form.

Table 1. Comparison of the proposed scheme with schemes where separate multipliers are used

Multiplication scheme	Hardware complexity per bit (in transistors)	Efficiency
Two serial/parallel multipliers for constant coefficients in 2's complement form	$FA+3DE+3SW=64$	50%
Two serial/parallel multipliers of Fig. 1	$\frac{2}{3}FA+\frac{20}{3}DE+\frac{20}{3}SW=109$	100%
Proposed merged scheme	$\frac{2}{3}FA+\frac{1}{9}HA+\frac{24}{9}SW+\frac{35}{9}DE+NAND=68$	100%

FA: Full-Adder (24 transistors), HA: Half Adder (10 transistors), DE: Dynamic Delay Element (8 transistors), SW: Switch (6 transistors) [7]

According to this table, the CSD representation significantly reduces the hardware complexity. Thus, the second scheme of the table has almost the same hardware complexity in spite of the double shift register overhead. Furthermore, the merging technique yields a significant hardware reduction, because only one double shift register and one sum line are included in the circuit and therefore the number of delay elements is significantly reduced. Another important advantage of the proposed scheme is that every three cells, on average, a delay element is inserted in *X*, *Y* and *R* lines. Thus, the broadcasting of these lines is limited to three cells.

4. CONCLUSION

In this paper, two serial-parallel multipliers for constant coefficients, which operate with 100% efficiency, are merged for the implementation of $ax+by$. This technique combined with the representation of the coefficients in CSD form, reduces significantly the hardware complexity. Moreover, the resulting circuit operates in pipeline mode with 100% efficiency. The above architectures have been developed in the context of a project that concerns the hardware implementation of digital filters. All circuits presented in this paper are extensively verified through simulation.

5. REFERENCES

[1] Ait-Boudoud D., Ibrahim M. K., Hayes-Gill B. R. Novel Pipelined Serial/Parallel Multiplier, IEE Electronics Letters, vol. 26, 1990, pp. 582-583.

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

[3] Caraiscos C., Pekmestzi K. Z. Low-latency bit-parallel systolic VLSI implementation of FIR digital filters, IEEE Trans. Circuits Syst.- II: Analog and Digital Signal Proc., vol. 43, 1996, no. 7, pp. 529-534.

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

[5] Ienne P., Viredaz M. A. Bit-Serial Multipliers and Squarers, IEEE Trans. On Comput., vol. 43, no 12, 1994, pp. 1445-1450.

[6] Kung, S. Y. On supercomputing with systolic/ wavefront arrays, Proc. IEEE, vol. 72, 1984, no. 7, pp. 867-884.

[7] Weste N., Eshraghian K. Principles of CMOS VLSI Design, Reading, MA: Addison-Wesley, 1994.