Bit-Level Pipelinable General and Fixed Coefficient Digit-Serial/Parallel Multipliers Based on Shift-Accumulation

Oscar Gustafsson and Lars Wanhammar

Department of Electrical Engineering
Linköping University, SE-581 83 Linköping, SWEDEN
E-mail: {oscarg, larsw}@isy.liu.se

ABSTRACT

In this work we introduce a novel approach to digit-serial/parallel multiplication. This general class of multipliers is based on shift-accumulation which also makes the approach suitable for implementation of shift-accumulators in distributed arithmetic. As a variable in the design process, the maximal number of cascaded full-adders can be selected. Thus, it is possible to as a special case obtain a bit-level pipelined multiplier. Both general and fixed coefficient multiplication is considered. The hardware complexity is low compared with other approaches.

1. INTRODUCTION

Digit-serial processing techniques has received considerable attention during the last decade [1][2]. In digit-serial processing a number of bits of the input word, a digit, is processed in parallel. If the digit-size, i.e., the number of bits processed concurrently, is one the digit-serial system reduces to a bit-serial system, while for a digit-size equal to the word length the system reduces to a bit-parallel system. The motivation for digit-serial processing is to find an optimum trade-off between the low area of bit-serial processing and the high processing power of bit-parallel processing. As low power consumption also has been a key interest in recent years, this is also a figure of merit to optimize for.

Traditionally, digit-serial multipliers has been obtained either via unfolding of a bit-serial multiplier [3] or via folding of a bit-parallel multiplier [4]. The problem with these approaches is that the obtained circuits have not been pipelinable at the bit-level. The reason for this is shown in Fig. 1 where a digit-serial adder obtained via unfolding is shown. The recursive loop prohibits the insertion of pipelining to reduce the critical path to less than d full-adders. Solutions to this problem has been proposed in a number of papers [5]-[7]. These solutions and our proposed solution are based on a redundant intermediate representation and a pipelined digit-serial adder at the output.

In this work we will present a novel method for obtaining digit-serial/parallel multipliers with arbitrary short critical path. The method is based on shift-accumulation and is thus also suitable for designing shift-accumulators used in distributed arithmetic. Both general and fixed coefficient multipliers will be considered.

2. PROPOSED STRUCTURES

Starting from the bit-serial multiplier in Fig. 2 we will derive the principles of a digit-serial multiplier and introduce our proposed multiplier structure. We will start with unsigned multiplication and extend to signed multiplication. Each clock cycle the multiplier in Fig. 2 inputs a bit from the input word and the partial products is formed in the AND-gates. These partial products is accumulated in the adders and shifted at the next clock cycle. Considering a dot representation of the partial products of the complete multiplication each clock period corresponds to a row in the partial product matrix [8].

The multiplier can be unfolded d times to obtain a digit-serial implementation with digit-size d. Each clock period d bits of the input word is applied to the input and d rows of the partial product matrix is added to the previous product stored in the registers. The different partial products in the operation is shown in Fig. 3. The top rectangle corresponds the partial products forming the intermediate sum in a carry-save representation. The parallelogram corresponds to the partial products of the current input digit.

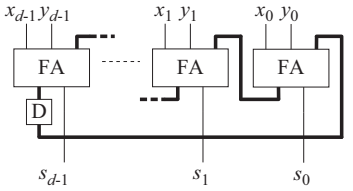

Fig. 1. Digit-serial adder obtained from unfolding.

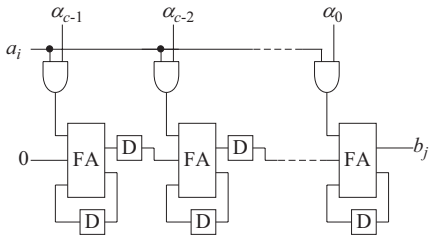

Fig. 2. Bit-serial/parallel multiplier.

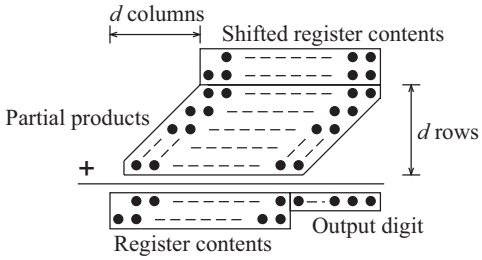

Fig. 3. Partial products in the unfolded digit-serial/parallel multiplier.

The main idea in this work is that instead of reducing the partial products to a maximum of two values as shown in Fig. 3, we allow an arbitrary number of registers. Thus, the registers be used both for pipelining and storing the intermediate values in the shift-accumulation process. By limiting the height of the reduction tree in the accumulator to h levels, the critical path will contain at most h full-adders. The partial products in this generalized shift-accumulation is shown in Fig. 4. The corresponding implementation is shown in Fig. 5. Note that the level of full-adders h is equal to the pipeline level. Therefore, selecting $h = 1$ leads to a bit-level pipelined implementation.

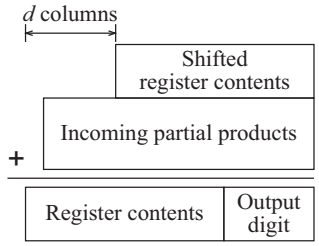

Fig. 4. Partial products in a general digit-size *d* shift-accumulation.

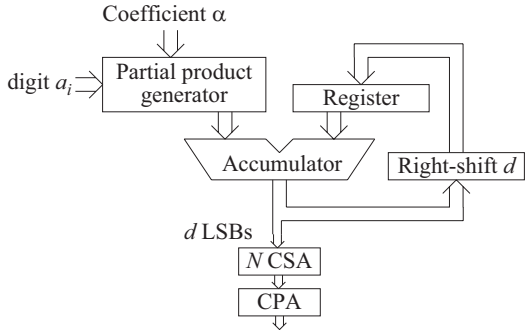

Fig. 5. Proposed multiplier structure.

The output digit of the shift-accumulator is no longer a single binary digit, but a digit represented in a redundant form. A number of digit-serial carry-save adders (CSA) is required to reduce the number of partial products of each significance level to a maximum of two. These two partial products can then be added using a digit-serial carry propagation adder (CPA) to form a single output word. A digit-serial CSA is shown in Fig. 6 and a digit-serial CPA is shown in Fig. 7 (a). It is possible to use the digit-serial adder in Fig. 1 as CPA, but as that adder can not be pipelined it is in most cases not a good choice. Instead it is possible to utilize a carry-lookahead adder, which can be pipelined to a maximal critical path of one AND-gate and one OR-gate as shown in Fig. 7 (b) [5]. Note that it is not necessary to have full-adders at all positions in the CSA, but only those significance levels with extra bits.

A modified structure is shown in Fig. 8. Here, one or more extra reduction trees of height *h* are inserted between the partial product generation and the accumulator. The reduction trees have registers between them to keep the critical path to at most *h* full-adders. This reduces the number of partial products to add in the accumulator, and, thus, reduces the complexity of the accumulator. However, as the pre-accumulator stages also requires both registers and full-adders the best overall structure must be evaluated for each case. As will be shown the number of total number of full-adders is constant for a given number of input partial products. These full-adders will then be distributed between the accumulator, CSA stages, and possible pre-accumulator. The stages require a different number of registers depending on the design variables and structure chosen.

To handle two's-complement data the most significant adder in Fig. 2 can be replaced by a subtractor as a W_c-bit two's complement number *A* can be written as

$$A = -\alpha_{W_c-1}2^{W_c-1} + \sum_{i=0}^{W_c-2}\alpha_i 2^i \qquad (1)$$

where α_i are the coefficient bits. A bit-serial subtractor can be implemented by inverting the partial product and adding a one to the LSB. The extra one can be inserted by setting the corresponding register to one at initialization. For a digit-serial multiplier it is possible to, in a similar way, invert all partial products corresponding to the sign-bit and set a register at the corresponding significance

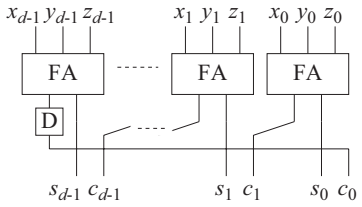

Fig. 6. Digit-serial carry-save adder with digit-size *d*.

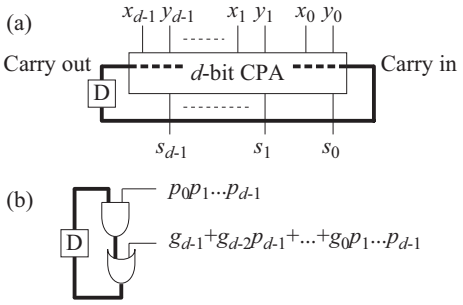

Fig. 7. (a) Digit-serial carry propagation adder and (b) minimal critical path using a carry-lookahead adder. Both with digit-size *d*.

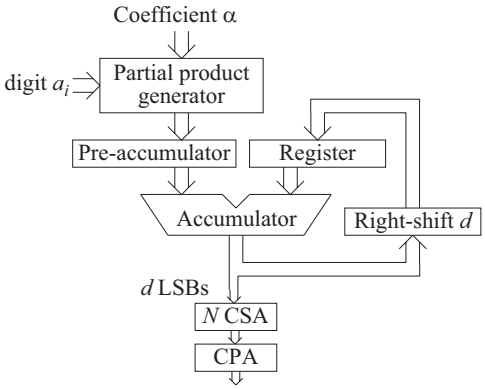

Fig. 8. Proposed multiplier structure with pre-accumulation of the partial products.

level to one. The input can be sign-extended to obtain all output digits.

Note that the coefficients are not assumed to be a multiple of the digit-size. This makes the multipliers suitable for e.g. digital filters where the coefficient wordlength and data wordlength are independent. However, to obtain output digits that are aligned to the input digits, it may be necessary to add an alignment stage consisting of at most *d*–1 registers.

2.1. Fixed Coefficient Multiplier

A multiplier with a fixed coefficient can be obtained using the following methodology. First, all AND-gates required for the partial product generation can be removed as the coefficient input is known. When the coefficient bit is one the AND-gate simplifies to a wire and when the coefficient bit is zero the AND-gate is removed. Removing an AND-gate also leads to that the corresponding partial product is always zero and can be removed from the partial product matrix.

To reduce the number of partial products, canonic signed digit (CSD) representation can be used [8]. The average number of non-zero bits is $W_c/3$ for CSD representation compared with $W_c/2$ for two's-complement. The CSD representation utilizes –1, 0, and 1 as values for the coefficient bit. However, –1-bits can be handled just

like the sign-bit for twos'-complement representation, i.e., inverting the partial product and adding a one.

The obtained partial products after simplification can be handled exactly like the partial products obtained from the general coefficient approach in the design method to follow.

3. DESIGN METHOD

The main design step is to decide how many registers are required for the requested degree of pipelining (or tree height h). To do this we propose the following algorithm

I. Let $\boldsymbol{A}$ be a vector containing the number of partial products for each significance level. Let $\boldsymbol{B}$ be a vector containing the number of registers for each significance level. Initialize all elements in $\boldsymbol{B}$ to zero.

II. Form $\boldsymbol{C} = \boldsymbol{A} + \boldsymbol{B}$, where $\boldsymbol{C}$ is the vector containing all product to be added.

III. Reduce the vector $\boldsymbol{C}$ using h levels full-adders to form the vector $\boldsymbol{D}$. $\boldsymbol{D}$ contains the number of partial products for each significance level after the accumulator. Three bits in $\boldsymbol{C}$ can be reduced to one sum bit in the same position and a carry bit in the next significance position.

IV. The d least significant positions of $\boldsymbol{D}$ now contains the output digit. Split $\boldsymbol{D}$ into the output digit $\boldsymbol{E}$ and new number of required registers, $\boldsymbol{B}$', where $\boldsymbol{B}$' is zero-extended to the same length as $\boldsymbol{A}$.

V. If $\boldsymbol{B}$' is equal to $\boldsymbol{B}$ the design process is finished and $\boldsymbol{B}$ is the required number of registers. If not set $\boldsymbol{B} = \boldsymbol{B}$' and go to II.

This algorithm converges fast and gives correct results for all examples tested. However, for digit-size one, i.e. bit-serial and tree-height one the algorithm produces a sub-optimal solution. This can be avoided by having a tree height of two, which does not affect the solution as no full-adders will be allocated to the second level.

When pre-accumulation is used, the partial products from thew partial product generation is first reduced using the given number of full-adder levels, h, and stages. The output of the pre-accumulator forms $\boldsymbol{A}$.

If the number of partial products for any significance level in the output digit is larger than two, CSAs must be included. These can be designed by inserting the required number of CSAs with pipeline stages after every h stages. Note that not all full-adders must be used in a CSA stage if the number of partial products for a given significance level is two or less. These can be removed and the products fed through to the next stage.

4. COMPLEXITY RESULTS

In the proposed multiplier the required number of full-adders is equivalent for all different structures and design choices. Assuming a digit-size of d, the number of partial products after the partial product generation is $P = W_c d$ for a general coefficient multiplier and $P = W_{nz} d$ for a fixed coefficient multiplier, where W_c is the coefficient wordlength and W_{nz} is the number of non-zero bits in the coefficient, respectively. To generate the partial products $W_c d$ and-gates are required for the general coefficient multiplier, while these can be removed for the fixed-coefficient multiplier. After the CSA stage there are $2d$ partial products. As each full-adder reduces three partial products to two, the number of full-adders is $N_{FA} = P - 2d$. These will be distributed between the different stages depending on the design variables.

The required number of registers for a 32-bit coefficient multiplier without pre-accumulation stage is shown in Fig. 9 for varying digit-size and pipeline level, h. It is clear that the required number of registers increases as the digit-size increases and pipeline level decreases. Note that the registers includes what would be seen as pipeline registers in other proposed multipliers, thus, it is natural for the number of registers to increase as the pipelining increases. Further, for a large enough tree-height, only the sum and carry bit are stored for each significance level. This corresponds to the traditional approach, but with a Wallace tree type of partial product reduction.

The required number of CSA stages is also an interesting factor as this increases the latency as pipeline registers are inserted after every h stages. The number of CSA stages for a multiplier with the same characteristics as above is shown in Fig. 10. Note that the lines for $h = 4$ and $h = 5$ are identical. It can be seen that if h is large, no CSA stages must be used as the reduction tree has reduced the partial products in the output digit to at most two bits.

Table 1 shows metrics of a general 16-bit coefficient multiplier with digit-size four using different architectures and maximal critical path lengths. For $h = 1$ the multiplier without pre-accumulation yields the implementation with the lowest number of registers. However, for h =2 the pre-accumulation multiplier yields the lowest complexity. It is clear from the figure that the number of full-adders are constant, but they are distributed between the different stages depending on the design variables. These different implementations should also be evaluated for power consumption. Again, note that the registers includes the pipeline registers.

A fixed coefficient multiplier with coefficient 1839 and digit-size four was also designed and the metrics of the different implementations are shown in Table 2. No pre-accumulation could be used as the number of partial products for any significance level was smaller than three.

Table 2: Complexity of a fixed multiplier with digit-size four and coefficient 1839 (excluding CPA stage). No pre-accumulation is used.

Pipeline level, h	Registers			Full-adders			CSA Stages
	Acc.	CSA stage*	Total	Acc.	CSA stage	Total	
1	21	20	41	9	3	12	1
2	19	8	27	12	-	12	0

* Including input and output registers of CSA stage.

The overall area complexity of the proposed multipliers is lower than the multipliers proposed in [5] and [7], and about the same as most of the multipliers presented in [6].

Table 1: Complexity of a general 16-bit multiplier with digit-size four (excluding partial product generation and CPA stage).

Structure	Pipeline level, h	Registers				Full-adders				CSA Stages
		Pre-acc.	Acc.	CSA stage*	Total	Pre-acc.	Acc.	CSA stage	Total	
No pre-acc.	1	-	75	64	139	-	40	16	56	3
No pre-acc.	2	-	51	19	70	-	54	2	56	1
No pre-acc.	3	-	29	8	37	-	56	-	56	0
1 level pre-acc.	1	49	62	55	166	15	30	11	56	3
1 level pre-acc.	2	25	29	8	62	29	27	-	56	0
2 level pre-acc.	1	84	47	35	166	29	21	6	56	2

* Including input and output registers of CSA stage.

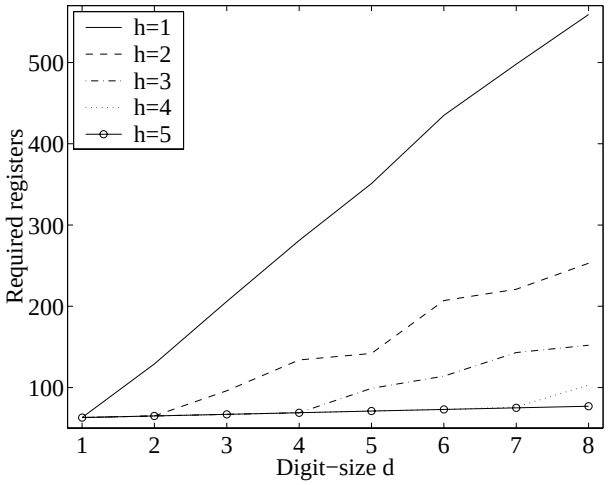

Fig. 9. Required registers for a 32-bit coefficient multiplier without pre-accumulation with various digit-sizes and pipeline levels.

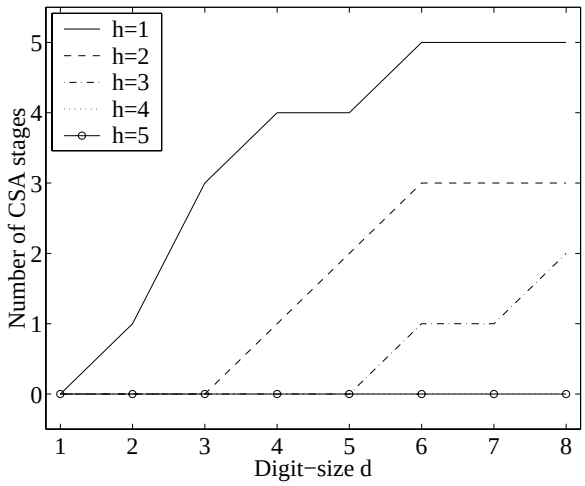

Fig. 10. Required number of CSA stages after the shift-accumulator for a 32-bit coefficient multiplier.

5. EXAMPLE

In this example a bit-level pipelined fixed multiplier with digit-size $d = 3$ and coefficient $-25 = -10100\text{–}1_{CSD}$ will be designed. The partial products for the multiplication is shown in Fig. 11 (a). Each diagonal row corresponds to a digit times a non-zero bit of the coefficient. If the bit is minus one, the digit is subtracted, and, thus, the bits are inverted and an LSB is added by setting a proper register. Using the algorithm in Section 3 gives that

$$\boldsymbol{A} = \begin{bmatrix} 1 & 1 & 2 & 1 & 1 & 1 & 1 \end{bmatrix}^T$$

When $\boldsymbol{B}$ is initialized to all zeros we first obtain $\boldsymbol{C} = \boldsymbol{A}$ and $\boldsymbol{D} = \boldsymbol{A}$ as no full-adders can be utilized. This gives

$$\boldsymbol{B}' = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 2 & 1 & 1 \end{bmatrix}^T$$

As $\boldsymbol{B} \neq \boldsymbol{B}'$ we set $\boldsymbol{B} = \boldsymbol{B}'$ and start again. Now

$$\boldsymbol{C} = \begin{bmatrix} 1 & 1 & 2 & 2 & 2 & 3 & 2 & 2 \end{bmatrix}^T$$

$$\boldsymbol{D} = \begin{bmatrix} 1 & 1 & 2 & 2 & 3 & 2 & 2 & 2 \end{bmatrix}^T$$

$$\boldsymbol{B}' = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 2 & 2 & 3 \end{bmatrix}^T$$

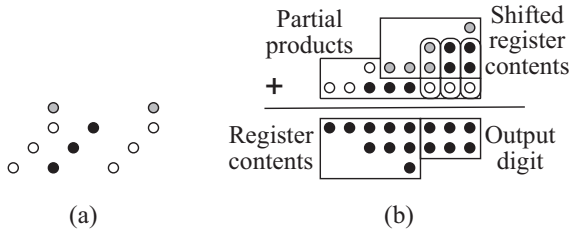

Fig. 11. Multiplication with $-25 = -10100\text{–}1_{CSD}$ and digit-size 3. (a) generated partial products and (b) partial products in shift-accumulator. A white dot denotes inverted partial product and a gray dot denotes a register to be initialized to one.

In the next iteration

$$\boldsymbol{C} = \begin{bmatrix} 1 & 1 & 2 & 2 & 2 & 3 & 3 & 4 \end{bmatrix}^T$$

$$\boldsymbol{D} = \begin{bmatrix} 1 & 1 & 2 & 2 & 3 & 2 & 2 & 2 \end{bmatrix}^T$$

$$\boldsymbol{B} = \boldsymbol{B}' = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 2 & 2 & 3 \end{bmatrix}^T$$

The algorithm stops as $\boldsymbol{B} = \boldsymbol{B}'$ and a multiplier with partial products and registers as in Fig. 11 (b) is obtained. The output digit $\boldsymbol{E}$ is [2 2 2] so no CSA stage is required. As there are two subtractions two registers must be set to one at initialization as marked by the gray dots in Fig. 11 (a). With no register available for the most significant one, it may be necessary to add an extra partial product. However, is this case the one can be distributed to registers of lower significance as shown in Fig. 11 (b). The multiplier requires three adders in the accumulator and the partial products added is also shown in Fig. 11 (b).

6. CONCLUSIONS

In this work we have introduced a novel approach to digit-serial/parallel multiplication. This general class of multipliers is based on shift-accumulation which also makes the approach suitable for implementation of shift-accumulators in distributed arithmetic. As a variable in the design process, the maximal number of cascaded full-adders can be selected. Thus, it is possible as a special case to obtain a bit-level pipelined multiplier. Both general and fixed coefficient multiplication were considered and the multipliers are easy to simplify for a fixed coefficient. The hardware complexity is low compared with other approaches. The coefficients are not divided into digits. This is suitable for e.g., digital filters where the coefficient wordlength and data wordlength are independent.

7. REFERENCES

[1] S. G. Smith and P. B. Denyer, *Serial Data Computation*, Kluwer, 1988.

[2] R. I. Hartley and K. K. Parhi, *Digit-Serial Computation*, Kluwer, 1995.

[3] K. K. Parhi, "A systematic approach for design of digit-serial signal processing architectures," *IEEE Trans. Circuits Syst.*, vol. 38, pp. 358–375, Apr. 1991.

[4] K. K. Parhi, C.-Y. Wang, and A. P. Brown, "Synthesis of control circuits in folded pipelined DSP architectures," *IEEE J. Solid-State Circuits*, vol. 27, pp. 29–43, Jan. 1992.

[5] A. S. Ashur, M. K. Ibrahim, and A. Aggoun, "Systolic digit-serial multiplier," *IEE Proc. Circuits, Devices, Syst.*, vol. 142, no. 1, pp.14–20, Feb. 1996.

[6] Y.-N. Chang, J. H. Satyanarayana, and K. K. Parhi, "Systematic design of high-speed and low-power digit-serial multipliers, "*IEEE Trans. Circuits Syst.–II*, vol. 45, no. 12, pp. 1585–1596, Dec. 1998.

[7] O. Nibouche, A. Bouridane, M. Nibouche, and D. Crookes, "A new pipelined digit-serial multiplier," *Proc. IEEE Int. Symp. Circuits Syst.*, Geneva, Switzerland, May 28–31, 2000, vol. 1, pp. 12–15.

[8] L. Wanhammar, *DSP Integrated Circuits*, Academic Press, 1999.