

Column Compression Pipelined Multipliers

Luca Breveglieri Luigi Dadda Vincenzo Piuri

Dipartimento di Elettronica e Informazione
Politecnico di Milano
Piazza Leonardo Da Vinci n. 32
I-20133 Milano - Italy

Abstract

The paper presents a study on the introduction of pipelining in parallel VLSI multipliers, built according to the column compression (CC) design techniques. A number of CC multiplier schemes have been proposed in the literature, aimed at reducing the number of stages of adders necessary to compute a multiplication. More recently CC multiplier schemes aimed at optimising the required silicon area, the regularity and the locality of the interconnections among the adders, have been proposed [5]. The paper affords the introduction of pipelining in these last structures and compares the obtained results with existing structures, in terms of required number of components and operation frequency.

1. Introduction

Parallel/parallel to parallel multipliers for integers can be implemented by means of a number of different architectures, which aim at optimising assigned figures of merit. Since the structure of a multiplier consists mainly of (full and half) adders and flip-flops, the main figures of merit are the number of full adders and flip-flops required to implement the multiplier, and the throughput.

There exist several approaches to the optimisation of the architecture of parallel/parallel to parallel multipliers, among which interesting ones are provided by the column compression (CC) multiplication algorithms [1, 2, 4, 5]. The derived design techniques tend to reduce the depth of the summation tree, the nodes of which are adders, necessary for adding up the partial products to compute the final result of the multiplication. In order to do this, the summation of the partial products is carried out in stages, each one of which reduces the number of partial products to add. Carry propagation in each stage is eliminated by means of various carry save schemes [1, 2, 4, 5]; the whole chain of stages reduces the array of partial products to a pair of numbers. At the end of the chain of stages a final fast adder (frequently a carry look-ahead adder) computes the last addition and outputs the result of the multiplication. The schemes of RA, Dadda and Wallace CC multipliers, respectively presented in [1, 2, 4], optimise, in various ways, the stages of half and full adders necessary for computing the compression of the columns of partial products of the partial product array. It is known that the number of stages is lower bounded by the relationship $\lceil \log_{1.5} n \rceil$, if only counters of type (3, 2) (full adders) and possibly of type (2, 2) (half adders) are used, and all CC multiplier schemes respect this lower bound, but there exist several ways to design each stage, requiring different numbers of half and full adders, flip-flops and producing distinct interconnection schemes. References [2] and [4] contain two well-known (the Dadda and Wallace multipliers, respectively) different design methodologies for the stages of CC multipliers; reference [1] (the Reduced Area multiplier) presents improvements with respect to [2] and [4], and gives a

comprehensive discussion and comparison of all such design techniques. In all such cases the main goal is to minimise the size of the stages.

In [6] a more refined reasoning is introduced, based on the observation that the sum output of a full (half) adder has a different time delay with respect to the carry output, as sums and carries are computed by different boolean functions. Schemes are presented, that aim at minimising the overall summation time of the array of the partial products, taking into account the different propagation delays of sum bits and carry bits.

In [5] a different design technique (in the sequel called the Windsor multiplier) aiming at a different optimisation criterion has been presented. The "area efficiency" of the multiplier is considered, a figure of merit defined as the percentage of used silicon area of the circuit with respect to the total area of the rectangular bounding box including the circuit. This figure of merit is significant because implementation technologies often require circuits to have a rectangular shape. This design technique aims at building a summation array of half and full adders, that adds up the matrix of the partial products according to a CC algorithm, but optimising the area efficiency and placing half and full adders in such a way as to have only interconnections to the nearest neighbours [5]. In [5] only a set of rules is presented, that admits several solutions; automated extensive check has proven that the multiplier shown in figure 1 is the optimal solution (for $n = 8$ bits) with respect to the area efficiency and also the minimization of non-local connections. The obtained structure sensibly improves these two figures of merit with respect to the structures proposed in [1, 2, 4].

Very often pipelining is easily implementable in multiplier structures, as they are highly regular: the interconnections among the adders are iterative and the partial products are injected into the summation array in a regular way. Thus, for instance, CC multipliers as those proposed in [1, 2, 4] can be pipelined by storing the multiplicand factor in parallel and by supplying the multiplier factor one bit per clock cycle, therefore increasing the clock frequency and hence the throughput (but at the price of introducing latency and of enlarging the size of the device). Instead, the partial products are injected into the summation array proposed in [5] in a highly irregular way, as this design technique privileges the maximisation of the area efficiency and of the locality of the interconnections. This structure is therefore not well-suited, as it is, to the introduction of pipelining.

The problem of introducing or optimising the pipelining of parallel/parallel to parallel multipliers can be split into two subproblems: pipelining the partial product generator and the summation array. In general, these two subproblems are dependent. In fact, the propagation delay of the partial product generator (which contributes, together with the propagation delays of the stages of the summation array, to determine the minimum clock period of the whole multiplier) is influenced by the distribution of the injection points of the partial products into the summation array; on the other side, different summation arrays have different distributions of injection points of the partial products. Therefore, solving the two design problems independently only allows to achieve sub-optimal solutions; a global approach is required for achieving optimal solutions.

In this paper we afford only the problem of introducing pipelining into the summation array of the multiplier. This is a simplification of the real design problem of a multiplier. However, there exist practical situations where these simplifying assumptions are well motivated. For instance, the convolution operation in DSP applications requires the computation of a continuous flow of multiplications, each one having one fixed, or very unfrequently modifiable, factor (such fixed factors are the coefficients of the kernel of the convolution). Since in DSP applications data are fed continuously and possibly at a very high rate, the introduction of pipelining in the multipliers forming the convolver is a reasonable design

choice. But if one factor of a multiplication is fixed, it does not give rise to propagation delays, therefore simplifying the design of the partial product generator. In this case it is advisable to concentrate the attention on the hard part of the problem, which is designing an optimised pipelined summation array.

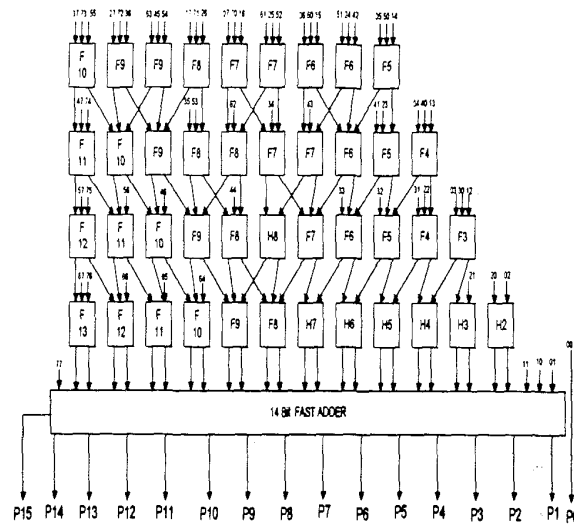
In this paper we present a technique for modifying the structures of the summation array of parallel CC Windsor multipliers, described in [5], for improving their fitness to pipelining. The goal is to obtain a structure that, while preserving some of the advantages of [5], exhibits a better fitness to pipelining, like the structures discussed in [1, 2, 4]; in fact, such structures present a more regular distribution of the inputs feeding the partial products to the summation array, which allows to pipeline the array more easily. In section 2 design rules for reshaping the structure introduced in [5] in order to improve its fitness to pipelining are discussed, by working out some significant examples. In section 3 the structures obtained in section 2 are evaluated, with respect to the number of components, half adders, full adders and flip-flops, and with respect to the throughput; these performances are compared to those of the original Windsor CC parallel multipliers of [5] and to the classical Braun parallel multiplier (the diagonal summation array of full adders that directly implements the addition of partial products, without column compression [3]). It is proven that the new design rules sensibly improve the fitness to pipelining of the Windsor CC parallel multiplier of [5] and preserve its notable locality properties, although the number of components is still larger than that of the pipelined Braun parallel multiplier. This gap gets narrower by reducing the degree of pipelining. The conclusion lists some directions for future research.

2. Pipelining of column compression multipliers

We give a set of rules for the introduction of pipelining in column compression Windsor parallel multipliers [5]. First we consider the simplest form of pipelining for multipliers: the multiplicand is supplied and stored in parallel form and its bits are all simultaneously available, whereas the multiplier is supplied serially one bit at a time, in LSB form; at clock $i \geq 0$ n bits, of weight from 2^0 to 2^{n-1} , each one belonging to a different multiplier factor, are supplied to the device. One new row of the partial product matrix is available every clock period, for every multiplier factor. The multiplier must then be partitioned into *stages*, each one of which computes a new step of a different multiplication. The stages must be separated by flip-flops to decouple them in time.

Figure 1 shows that the distribution of the partial products (PP) in the 8×8 bits CC Windsor summation array is not suited to this form of pipelining. For instance, in the top stage of the multiplier the PP's 37 and 73 are added up by the same full adder, but there is a time gap of 4 clock periods between the generation of these two PP's if the multiplier factor is supplied one bit per clock period.

To consider this fact, which is the main obstacle to pipelining, a PP $p_{j,i} = a_i \times b_j$, resulting from the product of the bit i of the multiplicand a and the bit j of the multiplier b , is said to have a higher *priority* than the PP $p_{h,k}$ if $j < h$. The *weight* of the PP $p_{j,i}$ is $i + j$; therefore the PP's $p_{j,i}$ and $p_{h,k}$ belong to the same column, i.e. have the same weight, iff $i + j = k + h$. In this case the PP $p_{j,i}$ must be added by an earlier stage than the PP $p_{h,k}$, otherwise it would be necessary to store the PP $p_{j,i}$ for $h - j$ clock periods to synchronise it with $p_{h,k}$. In general, the best solution is to add a PP as soon as it becomes available, for this avoids the introduction of delay flip-flops, but this is not always possible. Therefore we propose to apply to the Windsor CC multiplier the following reshaping rules, aimed at increasing its fitness to pipelining.

Figure 1: The Windsor CC parallel multiplier, with $n = 8$ bits operands.

Swapping partial products: high and low priority PP's belonging to the same column are swapped, so that high priority PP's are added up and hence consumed earlier, not requiring long term storage. Figure 2 shows an example, where the (low priority) PP 40 is swapped with the higher priority PP 22.

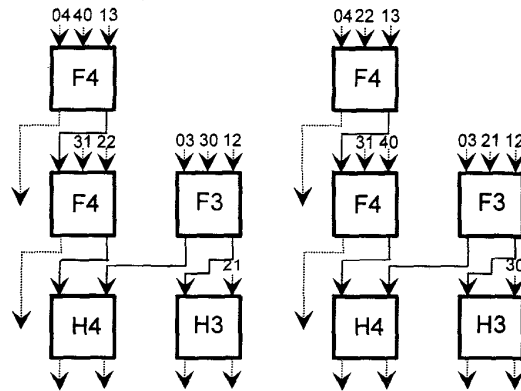


Figure 2: Multiplier reshaping: swapping partial products.

Swapping full and half adders: full and half adders belonging to the same column are swapped, in order to place a new available PP input in some stage of the summation array.

This may be useful, for instance, when the rule (1) is no longer applicable for lack of PP inputs in the array. Figure 3 shows an example, where in order to move downwards the (low priority) PP 71 it has been necessary to swap the leftmost top full adder with the rightmost bottom half adder.

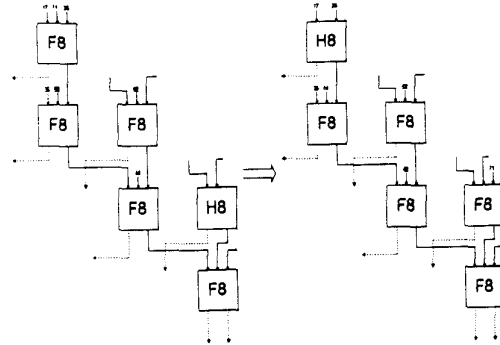


Figure 3: Multiplier reshaping: swapping full and half adders.

Re-routing of the carries: the insertion point in column $c + 1$ of a carry outgoing from column c is moved inside column $c + 1$, in order to have a new available PP input in some stage of the summation array. This may be useful whenever rules (1) and (2) are no longer applicable for lack of inputs in the array. Figure 4 shows an example, where two carries are re-routed with the aim of later moving downwards low priority PP's and to anticipate the additions of the full adders F6 and F5.

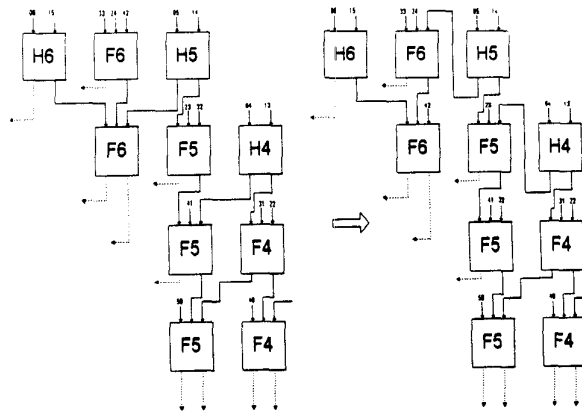


Figure 4: Multiplier reshaping: re-routing of the carries.

Prepending full adders to half adders: full adders are moved, whenever possible, towards the earliest stages, because a full adder in an early stage allows to save flip-flops in any subsequent stage. Figure 5 shows an example.

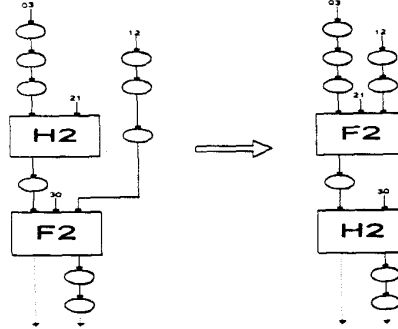


Figure 5: Multiplier reshaping: prepending full adders to half adders.

Reducing the size of the final fast adder: the length of the final fast adder is reduced, by computing some final addition directly in the summation array. The additions of the PP's that are injected directly or crossing a small number of stages into the final fast adder are anticipated. Figure 6 shows an example.

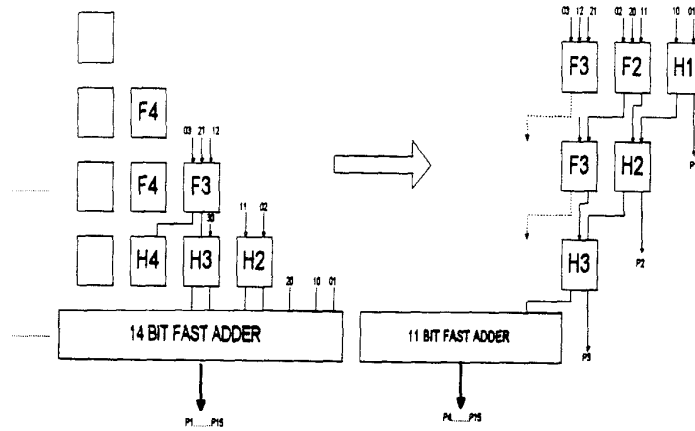


Figure 6: Multiplier reshaping: reducing the size of the final fast adder.

All the above proposed rules preserve the locality of the interconnections inside the array, but rule (3), which, if locality is to be preserved, should be applied with care, avoiding all non-local re-routing of the carries.

Rules (2), (3) and (4) must be applied with some care, in order to avoid to introduce flip-flops because sum and/or carry bits must be delayed to wait for a PP to become available. In practice, a useful precondition for rules (2) and (3) is to check whether the preceding sum and carry bits are already available when a PP is to be added. In particular, when re-routing carries, rule (3), it is advisable to check whether the adder that generates the carry bit receives higher priority PP's than the adder that receives the carry bit, and so on descending the chain; this rule of thumb always avoids the introduction of flip-flops. Finally, note that rule (4) may be contradictory to rule (2); for eliminating it one should remember that the best policy is to add a PP as soon as it is generated, as this avoids the introduction of delay flip-flops.

Figure 8 shows a 8×8 bits Windsor CC parallel multiplier modified by applying by hand the above reshaping rules (1, 2, 3, 4, 5)¹. Table 1 shows the dimensions, measured in terms of components, of 4, 6 and 8 bits Braun² and Windsor CC (modified) multipliers with a final fast adder of 10 bits.

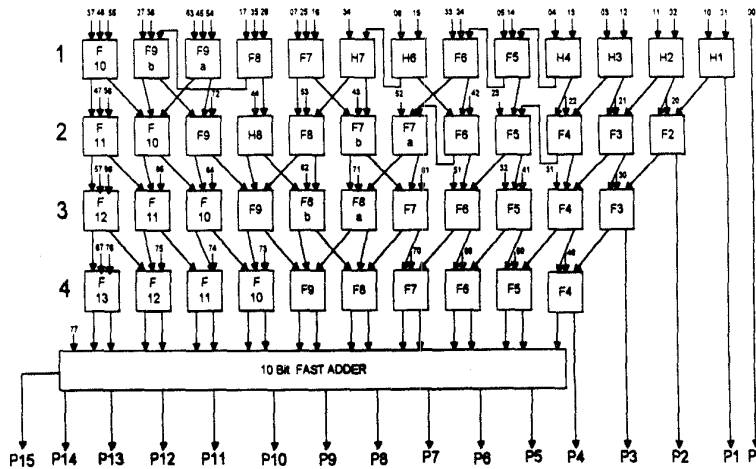


Figure 7: Modified Windsor CC parallel multiplier, with $n = 8$ bits operands.

	Braun ²			Windsor CC			modified Windsor CC (10 bits final fast adder)		
	4 bit	6 bit	8 bit	4 bit	6 bit	8 bit	4 bit	6 bit	8 bit
adders	8	24	48	6	20	42	8	22	46

Table 1: Dimensions of CC parallel multipliers.

¹ The structure contains a single non-local connection, involving adders F8 and F9b in the top row.

² The number of full adders of the Braun multiplier is here computed by means of the well-known formula $n(n-2)$, since the first stage of full adders can add up three rows [3].

As it is easy to see, the application of the above reshaping rules slightly worsens the size of the Windsor multiplier; however, it still remains smaller than the traditional Braun multiplier. Of course, CC multipliers have a far smaller number of stages than the traditional Braun multiplier. As said before, the applied reshaping rules are intended to improve the fitness to pipelining. Table 2 shows the introduction of pipelining in a multiplier, in the case of the modified Windsor CC multiplier with a 10 bit fast adder, shown in figure 7, where the multiplicator factor is supplied one bit per clock period. Each stage contains only the adders that can be used during the same clock period. The adders are denoted *Frow/column* (or *Hrow/column* if they are half adders): for instance, half adder H1 in row 1 (H1/1) can be used simultaneously to half adder H2 in row 1 (H1/2), etc. (see figure 7).

stage	adders	n. full adders	n. half adders	n. flip-flops	total
I	none	0	0	8	8
II	H1/1, H1/2, H1/3, H1/4, H1/6	0	5	16	21
III	F2/2, F2/3, F2/4, F1/5, F1/7	5	0	19	24
IV	F3/3, F3/4, F2/5, F1/6, H1/7, F1/8	5	1	22	28
V	F4/4, F3/5, F2/6, F2/7b, H2/8, F1/9b	5	1	25	31
VI	F4/5, F3/6, F2/7a, F2/8, F1/10	5	0	28	33
VII	F4/6, F3/7, F3/8b, F1/9a, F2/11	5	0	31	36
VIII	F4/7, F3/8a, F2/9, F2/10, F3/12	5	0	34	39
IX	F4/8, F3/9, F3/10, F3/11, F4/13	5	0	29	34
X	F4/9, F4/10, F4/11, F4/12	4	0	25	29
total		39	7	237	283

Figure 8: Pipelining a $n = 8$ bits modified Windsor CC parallel multiplier: the multiplicand is stored in parallel, the multiplier is supplied one bit per clock cycle.

It is however possible to refer to various degrees of pipelining, by partitioning the multiplicator factor into groups of parallel bits; for instance, if $n = 8$, then the multiplicator factor could be partitioned into two groups of four bits, four groups of two bits or one group of eight bits³. Each group of parallel bits allows to generate a number of partial products. The same technique shown in figure 8 can be used to pipeline the multiplier, provided any two cascaded adders are decoupled with a flip-flop and that the PP's supplied to the latter one are delayed of one clock period. In practice, instead of having single rows of PP's, one needs to consider groups of rows of PP's, but all the above discussion still applies. Figure 9 shows the pipelining of a $n = 8$ bits modified Windsor CC multiplier (see figure 7), with a multiplicator factor supplied completely in parallel.

³ Other partitionings can be obtained by allowing one of the groups of bits to be smaller than the others; this is not much difficulty for the multiplication, as the missing bits can be sign extended.

stage	adders	n. full adders	n. half adders	n. flip-flops	total
I	H1/1, F1/2, F2/4, F1/5, F1/6a, F1/6b, F1/7a, F1/7b, F1/8, F1/9b, F1/10	11	1	53	65
II	H2/2, F3/3, F3/4, F2/5, F2/6, F2/7a, F2/7b, F2/8a, F2/8b, F2/9, F2/10, F2/11	11	1	42	54
III	F4/3, H4/4, F3/5, F3/6, F3/7, H3/8, F3/8, F3/9, F3/10, F3/11, F3/12	9	2	33	44
IV	H4/5, H4/6, H4/7, F4/8, F4/9, F4/10, F4/11, F4/12, F4/13	6	3	27	36
total		37	7	155	199

Figure 9: Pipelining a $n = 8$ bits modified Windsor CC parallel multiplier: the multiplicand is stored in parallel, the multiplier is supplied eight bits per clock cycle.

Note that, although the multiplier factor is supplied in parallel, the multiplier does not reduce itself to a purely combinational summation array. In fact, cascaded adders are separated into stages with flip-flops and the input PP's are consequently delayed. However, the number of stages is lower than that for fully pipelined multiplier factors (see for instance figure 8).

3. Evaluation of cost and performance

A number of pipelined multiplier structures have been designed and compared against one another, for some commonly used number of bits ($n = 4, 6, 8$). The figures of merit are: the number of components, i.e. (full and half) adders and flip-flops, and the area efficiency, defined as the ratio between the area of the circuit and the area of its rectangular bounding box. To obtain the area of the circuit, some layout design is necessary: all adders belonging to the same stage are arranged into a single line, together with the associated flip-flops; the areas of the adders and of the flip-flops are assumed to be equivalent. Table 2 shows the results for various types of multipliers, when the multiplier factor is fully pipelined.

	Braun ⁴		
	4 bit	6 bit	8 bit
adders	12	30	56
flip-flops	31	76	141
total	43	106	197
area eff.	82.69%	84.1%	84.9%

⁴ The number of full adders of the Braun multiplier is here computed by means of the well-known formula $n(n-1)$ (the first stage adds up only two rows), since this structure is more easily pipelined than a matrix of $n(n-2)$ full adders [3].

	Windsor CC		
	4 bit	6 bit	8 bit
adders	6	20	42
flip-flops	51	172	388
total	57	192	430
area eff.	67%	61.1%	61.1%

	modified Windsor CC (10 bits final fast adder)		
	4 bit	6 bit	8 bit
adders	8	22	46
flip-flops	33	111	237
total	41	133	283
area eff.	73.2%	73.07%	72.6%

Table 2: Comparison of $n = 8$ bits fully pipelined parallel multipliers.

The area efficiency of the modified Windsor CC multiplier shows an improvement with respect to that of the original Windsor CC multiplier, although it is still slightly lower than that of the Braun multiplier; this is due to the high regularity of the Braun structure. The number of components for the modified Windsor CC multiplier is sensibly lower than that for the original Windsor CC multiplier, but it is still far higher than that of the Braun multiplier, again for the latter one is highly regular; however, the reshaping rules above discussed improve notably the pipelinability of the Windsor CC summation array. Moreover, the modified Windsor CC multiplier retains the locality properties of the original Windsor multiplier (all interconnections are routed only to the nearest neighbours).

Table 3 shows instead the results for the modified Windsor CC multiplier, with different degrees of pipelining of the multiplicator factor; the figures of merit shown in the fourth column of the table refer to the pipelining shown in figure 9, those in the first column are taken from table 2.

	8 groups of 1 bit (fully pipelined) (see tab. 2)	4 groups of 2 bits (four parts)	2 groups of 4 bits (two parts)	1 group of 8 bits (fully parallel) (see fig. 9)
adders	46	44	44	44
flip-flops	237	212	190	155
total	283	256	234	199
area eff.	72.6%	74.6%	74.3%	76.5%

Table 3: Comparison of $n = 8$ bits modified Windsor CC parallel multipliers with partial pipelining: the multiplicator is supplied in groups of bits.

The area efficiency shows a slight improvement with respect to the case of a fully pipelined multiplicator factor (first column of the table or table 2), but the number of components

decreases sensibly when the degree of pipelining diminishes. Note that when the multiplier factor becomes completely parallel, the number of components of the modified Windsor CC multiplier is comparable to that of the Braun multiplier (as when a CC multiplier is pipelined and the multiplier factor is supplied in parallel, the number of stages must follow the formula $\lceil \log_{1.5} n \rceil$). The number of stages of the modified Windsor CC multiplier in this case is however lower (see figure 9) than that of the Braun multiplier, which is n ; this implies that the latency of the former one is lower, too. Moreover, the modified Windsor CC multiplier has only local interconnections among its components. Finally, note that in the modified Windsor CC multiplier the length of the final fast adder is highly optimised, as some of the final additions are anticipated in the summation array itself.

4. Conclusion

The issues of the introduction of pipelining in CC parallel multipliers have been discussed, with particular regard to the highly optimised multiplier architectures presented in [5]. In general, pipelining such multipliers requires structural modifications, which however preserve the locality of interconnections and do not significantly increase the number of adders, at least when the degree of pipelining is suitably chosen. Pipelining of the modified multipliers proves to be viable in some cases, yielding multipliers of cost comparable to that of more traditional structures.

Among the possible directions for future research there is the implementation of pipelined CC parallel multipliers and their evaluation, and the comparison of pipelined CC parallel multipliers with RA, Dadda and Wallace CC parallel multipliers [1, 2, 4]; this comparison is not an immediate task, as the optimisation criteria fulfilled by these architectures are different.

References

- [1] A. C. Bickerstaff, M. Schulte, E. E. Swartzlander, Reduced Area Multipliers, Proceedings of ASAP '93, L. Dadda, B. Wah (eds.), IEEE Society Press, pp. 478-489, Venice, Italy, 1993
- [2] L. Dadda, Some Schemes for parallel Multipliers, in *Alta Frequenza*, vol. 45, pp. 574-580, 1976
- [3] K. Hwang, *Computer Arithmetic*, John Wiley & Son, 1974
- [4] C. S. Wallace, A Suggestion for a fast Multiplier, in *IEEE Transactions on electronic Computers*, vol. EC-13, pp. 14-17, 1964
- [5] Zhongde Wang, G. A. Jullien, C. Miller, A new Design Technique for Column Compression Multipliers, University of Windsor, Ontario, Canada, N9B 3P4
- [6] M. S. Paterson, N. Pippinger, U. Zwick, Faster Circuits and shorter Formulae for Multiple Addition, Multiplication and symmetric Boolean Functions, CH2925-6/90/0000/0642\$01.00 1990 IEEE, pp. 642-650, 1990