Partial Product Reduction based on Look-Up Tables

H. Mora Mora, J. Mora Pascual, J.L. Sánchez Romero, F. Pujol López
Department of Computer Science Technology and Computation, University of Alicante, Spain
{hmora, jeronimo, sanchez, fpujol}@dtic.ua.es

Abstract

In this paper a new technique for partial product reduction based on the use of look-up tables for efficient processing is presented. We describe how to construct counter devices with pre-calculated data and their subsequent integration into the whole operation. The development of reduction trees organizations for this kind of devices uses the inherent integration benefits of computer memories and offers an alternative implementation to classic operation methods. Therefore, in our experiments we compare our implementation model with CMOS technology model in homogeneous terms.

1. Introduction

The multiplication operation is needed in most of the processor arithmetic units and is usually a part of the set of primitive functions these units provide [1], [2]. Since this operation is commonly utilized in many current applications [3], some efforts should be considered to improve the multiplication algorithms in order to reduce their processing complexity.

Many practical multiplication techniques employ a three-stage calculation method [1], [4], [5]: partial product generation, partial product reduction and the final addition. The partial product reduction stage consists of transforming all the generated products into only two. This stage takes a significant amount of time in relation to the total delay of the operation [1], [5], [13]. As a consequence, improving this part of the operation will have a great effect on the performance of the complete operator. In this paper, we study the suitability of using tables with pre-calculated data (LUT —Look-Up Table) for the partial product reduction stage in order to improve its latency.

This work is structured as follows: in section 2, we show a survey of the most usual methods for partial product reduction. Then, the proposed LUT-based method for reducing partial products is described in section 3. In section 4, the method is evaluated in terms of the time delay and the required area. In section 5, the method is compared with other reduction techniques. Finally, we conclude with some advantages of using our method and some future research.

2. A survey of partial product reduction methods

The usual partial product simplification method consists of using structures based on combinational logic until only two summands are considered. The final addition, in the subsequent stage, produces the final result of the operation. These schemes often make use of some counter and compressor components organized into different topological configurations. There is a huge list of works that deal with improving the performance of these organizations [1], [4], [7], [8], [9], [10], [11].

The counters and compressors are based on Carry Save Adders (CSA). Each CSA receives three bits and provide a sum and a carry bit as outputs. The resulting carry bit does not propagate to the next most significant position but is a part of an output carry vector [12]. Counters are combinational devices that have the function of counting the number of their inputs with the logical value ‘1’. They are usually denoted as a pair (p, q), which indicates the number of inputs (p) and outputs (q) of the circuit, where $q \geq \lg_2(p+1)$ [4]. The most commonly used counter is the (3, 2), which accepts a 3-bit vector as input and produces a 2-bit output. This counter is a basic component for the construction of the most advanced counters and compressors [5].

The highest-grade counters increase the reduction ratios of their inputs and create more compact counting structures by maintaining an asymptotic time cost with the number of summands to be reduced. [12]. We must remark (7, 3), (15, 4) and (27, 5) counters because they have been used and studied in other related works [10], [13], [14]. To reduce the set of multiplication partial products, the counters are usually organized into vector structures [15] or tree structures [8], [9]. The

design of their organization and some improvements related to their performance has been reported in several research reports, such as [4], [6], [16]. The number of counter levels for these trees is proportional to $lg_{p/q}$ N, where N is the number of partial products to be reduced.

The Three-Dimensional Method (TDM) is one of the most noticeable algorithms for partial product reduction [7], [17] because of its good results. This technique takes into account the fact that not all the inputs to the reduction tree counters contribute in the same way to the delay. This method distributes the inputs and outputs into a three-dimensional reduction structure which favors the minimum delay in the critical path.

Compressors are also combinational circuits that count the number of logical '1' values on its input lines [18]; in this case, an independent horizontal path is established for the propagation of the carry in each device, which reduces the critical path of the counting and allows the design of more simple and regular structures than counters. The compressor notation is defined as a pair [p, q], where (p) is the number of input lines and (q) the output lines [5]. The most frequent compressors are of the [4:2] type. A compressor tree [p:q] reduces the partial products by $lg_{p/q}N$, where N is the number of levels [5].

To sum up, the great number of research works dealing with this topic show how important the partial product reduction stage is for an efficient computing of the whole multiplication. This fact justifies the efforts dedicated to the search for new proposals that offer better results in further developments.

3. Partial product reduction by means of stored logic

Recent advances in electronic technology and increasing capacities have resulted in the conception of new operator designs in the construction of arithmetic units that improve performance both in quantitative and qualitative terms.

Our proposal is based on the stored logic paradigm so that it is used in a complementary way to the combinational design. The technique essentially consists of storing the results of any combination of operands for the function desired in an attached memory. When a result of the operation is required, the cell that contains its value is selected, without combinational circuitry or additional processing.

Nowadays, in the VLSI operator design, there are not many designs of hybrid combinational-stored logic operators [26]. LUTs are normally used to generate initial values or seeds; then, the execution of the algorithms is performed by combinational circuits [19], [21]. However, if memories are only used for this task, their potential performance would be wasted in the calculation of the required operators. That is the reason why we propose the use of LUTs as a central element in partial product reduction, taking advantage of the most recent advances in technology and development of memory devices [22], [23]. To do this, the design stage starts with the creation of a central LUT which will work as a generalized counter element; let us call it *LUT-Counter*. Its notation is carried out by means of a pair (p_t, q), in which p indicates the columns to be counted, t the number of bits per column and q the output lines of the device. So, the LUT-Counter module has p·t input lines and q output lines. The number of output lines in relation to the input lines is expressed as:

$$t \cdot (2^p - 1) = 2^q - 1 \qquad (1)$$

The inputs consist of p columns of t bits corresponding to the partial products and the output is related to their adjusted weights. For any set of partial products, the same type of reduction has to be performed, either simultaneously or consecutively, for all the summands until the required degree of reduction is reached. To achieve this goal, several design methods are proposed, considering the performance and size area of the operator: systematic reuse for successive reductions, high performance equipment using multiport memories that allow several simultaneous accesses, design of the LUT devices for concurrent access, or any combination of these proposals. For example, figure 1 shows a general reduction design with several LUT-Counter devices.

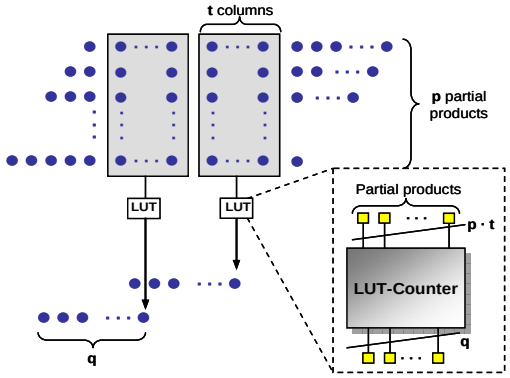

Figure 1: Partial product reduction using LUTs

The compression degree achieved when counting several columns simultaneously for rectangular sizes is defined as:

$$t:(q/p) \quad (2)$$

The design based on stored logic for counter elements discussed allows similar organizations to adapt the reduction stage to the characteristics of an operand. Moreover, the construction of hybrid structures with some counters based on LUTs and others based on combinational logic, provides flexibility in the design of the multiplier. To summarize, we propose the use of LUTs instead of gates, since current memory capacity enables the design of the reduction stage with a greater compression ratio than traditional combinational circuits. This will lead to the reduction of the partial products in fewer stages, as we will show next.

4. Analysis of the performance of the LUT-counter

As mentioned before, the use of LUTs for the practical processing of some stages in usual arithmetic operators takes many of the advantages of the increased integration density in VLSI implementations compared with other combinational methods of computation. Their use reduces the costs of hardware development, offers flexibility and favors the scalability of such kind of systems.

The LUT memory devices can be reused and provide a high degree of parallelism: the use of multiport memories with several access channels enable parallel results to be obtained on the same storage chip. Other benefits come from the inherent characteristics of computer memories: the rewriting of their contents allows their reconfiguration and application in the calculus of other operators, whereas the incorporation of some error detection techniques or any other correction element within the data facilitates a more robust processing [7], [19], [20]. Nevertheless, this implementation cannot be generalized for all operators, since there are no advantages for all cases. As a consequence, they must be analyzed to appreciate the potential benefits.

Let us consider next a comparison between the occupied area and the temporal delay. Other characteristics, such as energy consumption, are not discussed in this paper, although they should be considered for a final real chip implementation.

4.1 Area estimations

It is clear that the main factor that will affect the final chip area for the partial product reduction stage is the size of the LUTs. From a general point of view, a LUT-Counter with p and t input lines and q output lines needs a memory size (expressed in bits) obtained as:

$$\text{Area} = 2^{p \cdot t} \cdot q \text{ bits} \quad (3)$$

As we can see in equation (3), the growth of the space is directly related to the degree of the reduction and it increases exponentially with regard to the increase in the number of input lines. That is why the desired values for p, t and q are generally limited and affect their desired value. However, the size of the table can be reduced by including a pre-processing stage before the access to the LUT and increasing, this way, their application scope. The idea consists of reducing the possible combinations of the inputs. That is, the pre-processing can be summarized as follows:

1. Dividing the initial sequence of p bits to be counted, w, into three parts: two of them, w_l and w_r, of size $t = \lfloor p/2 \rfloor$ and the third one, w_m, of size 1. Fragment w_l corresponds to the leftmost p bits, whereas w_r corresponds to the rightmost t bits from the initial sequence of p bits, respectively; finally, w_m is the bit located in the middle of the sequence. Therefore, sequence w can now be expressed as follows:

$$\begin{aligned} w &= w_{l,t-1}w_{l,t-2}\ldots w_{l,0}\, w_m\, w_{r,t-1}w_{r,t-2}\ldots w_{r,0} \\ &\equiv w_l\, w_m\, w_r \end{aligned} \quad (4)$$

2. Constructing a new sequence of bits, v, resulting from applying the following logical operations:

$$v_l \equiv w_l \vee w_r;\ v_r \equiv w_l \wedge w_r;\ v \equiv v_l\, w_m\, v_r \quad (5)$$

The sequence v has the same number of ones as the original sequence w, so its sum will give the same result. However, there are a fewer number of valid combinations of v after this pre-processing stage with regard to w: w has 2^{p-1} different configurations, whereas the valid configurations for v are:

$$\text{combinations} = 2\sum_{i=0}^{k} 2^{u_i} \quad (6)$$

where $k = 2^p-1$ and u_i is the number of existing logical ones in the binary representation of number i.

The effect of this pre-processing on the LUT structure is evident, since the amount of data contained in the LUT is reduced considerably. The memory only

stores the cells corresponding to the valid configurations extracted from Equation (6), in spite of having the same addressing lines, as any other permutation is not allowed. For instance, figure 2 shows how we can group a set of inputs in the pre-processing stage of a LUT-Counter 7:3 (p=7, q=3, t=1).

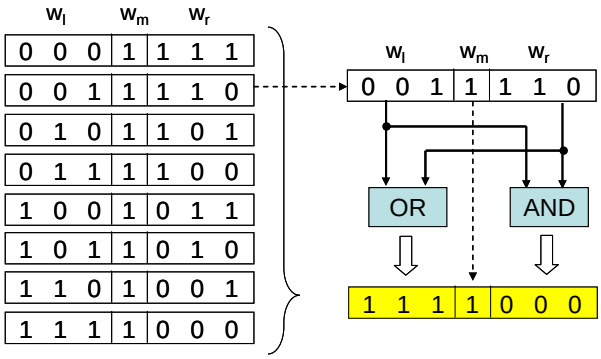

Figure 2: Original sequences of bits corresponding to the same LUT input

Let us consider now a comparison of our method with other reduction techniques. Let us express the required area in terms of complex gates by means of a relation between its size, expressed in bits, and the addressing lines of the memory. Let τ_a be the area of a complex gate. According to the model* presented in [19] and [21], an area of $40\tau_a$/Kbit is assumed for tables of up to 6 inputs, $35\tau_a$/Kbit for tables of 7-11 inputs, $30\tau_a$/Kbit for 12-13 inputs, $25\tau_a$/Kbit for tables of 14-15 inputs and $20\tau_a$/Kbit upwards. Table 1 shows the LUT-Counter sizes after the pre-processing and the space saving produced by this pre-processing.

Table 1: Area cost of the LUT-Counter for different reduction degrees

p	q	LUT-Counter size	Complex gates	Space saving (%)
3	2	12 bits	2 τ_a	87,5
7	3	162 bits	6 τ_a	57,8
15	4	17,08 Kbit	425 Kτ_a	86,7
31	5	17 Mbytes	> 426 Mτ_a	98,3

More complex hardware stages could be considered in order to reduce the table size even more, but the consequences in terms of delay would also have to be considered to evaluate the suitability of their use. This issue is discussed in the following section.

* Implementation using a family of standard gates from the AMS 0.35 μm CMOS library

4.2 Delay estimations

We can identify the delay of the counter circuit based on LUTs with the time required to access the table and capture the result; therefore, it is constant and independent of the operand values. This delay depends on both the memory architecture and its internal structure and consists of the access time to the table and the response time [24], [25].

The use of stored logic in this stage has a great influence on the temporal delay of the stage. For instance, the use of multiport memories with several access routes allows the reduction of multiple columns of data within the same table, that is, the LUTs can be replicated in order to answer several parallel requests; the location of the LUT as a memory embedded in the arithmetic unit and near to the rest of the logic of the operation reduces its access time [22], [23].

To estimate the delay, we have performed a benchmark using, firstly, the same estimation model than in the previous section and, on the other hand, the results of our own experiments† [7], [20]. Let τ_t be the delay for a complex gate. For the whole memory, we assume a delay of $3\tau_t$ for tables with up to 7 inputs, $3.5\tau_t$ for tables with 8 inputs, $4\tau_t$ for tables with 9 inputs, $4.5\tau_t$ for tables with 10-11 inputs, $5\tau_t$ for tables with 12-13 inputs and $6\tau_t$ for tables with 14-15 inputs. The pre-processing stage and the resulting memory space are also considered. The architecture of the LUT in the simulations is based on the design proposed in [24] and [25]. The LUT-Counter has been integrated into the multiplication operator together with the rest of the logic. In relation to these estimations, table 2 shows the delays in terms of complex gates for a LUT with different reduction degrees.

Table 2: Temporal cost of the LUT-counter for different degrees of reduction

t	p	q	Degree of reduction	Complex gates (τ_t)
2	5	4	(5, 2)	5
1	3	2	(3, 2)	3.5
1	7	3	(7, 3)	3.5
1	15	4	(15, 4)	6.5

As it can be observed in table 2, the delay, in terms of complex gates, depends on the number of LUT inputs. However, in our simulations, we found that the reduction in the table size due to the pre-processing reduces the delay and improves the global performance. This can be explained by the fact that the addressing decoders do not implement invalid combinations after the pre-processing phase.

† Simulation of the model made with VHDL using ModelSim 6.0.

5. Comparison with other reduction techniques

To validate our proposal, it is necessary to compare it with different well-known reduction algorithms, which must be independent on the final technology used in its practical implementation. The data taken for the delay have been compiled from relevant publications in this field. Therefore, to call the attention to the importance of both technology and implementation, in our experiments we have also calculated the results of the reduction stages based on other generally accepted configurations. Finally, a practical example of a complete multiplication is performed to check the effects of applying different reduction organizations on this operator.

5.1. Comparison in terms of complex gates

According to [5] and [7], a reduction stage (3:2) has a delay of 2 complex gates. Other counters with a greater reduction ratio require more complex gates per stage. For example, the (7,3) counter requires 4 τ_t [4] and the (27,5) counter requires 12 τ_t [10]. Compressors provide more compact structures, such that the [4:2] model has a delay of 3 complex gates, the [6:2] compressor a delay of 5 τ_t and the [9:2] a delay of 7 τ_t [4]. Table 3 shows these data and their relation to the reduction performed; then, it compares them with the designs based on stored logic. Other structures that use algorithmic reduction schemes, such as the Three-Dimensional Method (TDM) [4], [17], have delays that are essentially related to the number of the generated partial products; that is why they are not included in this comparison and are considered within the framework of the complete operation, instead.

Table 3: Comparison of characteristics among reduction structures

	counter/compressor [4]						LUT-Counter		
	(3,2)	[4:2]	(7,3)	[6:2]	[9:2]	(27,5)	(5,2)	(7,3)	(15,4)
Reduction ratio (C_r)	1.5	2.0	2.33	3	4.5	5.4	2.5	2.33	3.75
Complex gates of critical path	2	3	4	5	7	12	4	3	5.5
Reduction ratio per complex gate	0.75	0.66	0.58	0.6	0.64	0.45	0.63	0.78	0.68

As it can be extracted from the table above, LUT-Counters achieve reduction ratios per complex gate comparable to other commonly used structures. Let us remark that, in the case of the (7,3) LUT-Counter, the reduction ratio per complex gate is even higher than the one for the (3,2) counter, despite the fact that the latter has a much higher compression ratio.

As a conclusion, the implementation of the structures mentioned in the previous section empirically verifies our assumptions. The simulation takes into account the average delays of the connections, the fan-out of the gates and the characteristics of the reconfigurable device. These estimations may vary according to the manufacturing technology and we think there would be a good behavior in relation to their delays after their integration into the operator and their subsequent implementation on an ASIC. For this reason the simulations of the model, whatever technology is used, have special relevance.

5.2. Multiplication examples comparison

To complete our experimentation, some tests considering the partial product reduction in multiplications of a usual number of bits were completed. In particular, 24 and 53 bits were used for the factors, as they are the length of the mantissas usually employed in floating-point operations. Due to their complexity, neither TDM structures nor optimized trees [16] have been implemented, so that their performance features have been taken from the works by Oklobdzija in [7] and [17].

Table 4 summarizes the results of the delays and the stages required for the previous configurations. This table shows, in brackets, the number of stages necessary for each partial product reduction structure. It can be seen that for both factor lengths, the proposed method offers delays that are greater than those ones based on Wallace trees and compressors [4:2], and that they are comparable to the methods organized into three-dimensional structures (TDM). In addition, all the advantages inherent to the nature of the memories that were mentioned before are now obtained and, moreover, we achieve a segmented operation since our approach uses a chain of successive multiplications. Other schemes, such as the three-dimensional reduction structure (TDM), do not easily allow segmented operation strategies.

Table 4: Delay (τ_t) and reduction stages

Reduction scheme	24x24 bit	53x53 bit
Wallace tree (3,2) counters	14 (7 stages)	18 (9 stages)
[4:2] tree	12 (4 stages)	15 (5 stages)
optimized trees [16]	12	15
TDM [7]	10	13
Combination of LUT counters	10 (3 stages)	13 (4 stages)

Other advantages are obtained because all the signals from the reduction structure based on stored

logic reach the last stage of the multiplier (i.e., the final addition) at the same time. There are few changes in the response delay of a look-up table compared with the delay of a combinational circuit using CSA elements, where there are generally multiple different paths.

6. Conclusions and future work

This work describes a new proposal of partial product reduction based on look-up tables. The method offers several advantages over classic combinational logic, due to its flexibility, reuse, fault tolerance and memory parallelism. The proposal is suitable for embedded systems design using FPGA where LUTS can be utilized to implement a soft processor as well as the stored logic for multiplication. We have made a rigorous analysis and some comparisons with other well-known reduction techniques in terms of temporal cost; as a consequence, we have shown that our method is a valid alternative to traditional circuits. Our experimentation gives fine results, which are comparable with those ones obtained from the best existing methods. In terms of the required area, the proposed method is not, generally speaking, comparable to others based on combinational logic. Nevertheless, the inclusion of the pre-processing stages before the table access to save size on-chip is feasible in a near future.

Our research is currently focused on improving the integration cost of look-up tables inside the processor and on extending this technique to other operators in the arithmetic unit. In future works, we expect to obtain good quality results that allow an improvement in the performance of current microprocessors.

7. References

[1] Bewick, G. W.: Fast multiplication: Algorithms and implementations. PhD Thesis, Dept. of Electrical Engineering, Stanford University, 1994.

[2] S.F. Oberman. Design Issues in High Performance Floating Point Arithmetic Units. TR CSL-TR-96-711. Computer System Lab., Stanford University. 1996.

[3] SPEC Benchmark Suite Release 2/92.

[4] Oklobdzija, V. G., Villeger, D.: “Improving Multiplier Design by Using Improved Column Compression Tree and Optimized Final Adder in CMOS Technology”. "*IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 3, pp. 292-301, 1995.

[5] Flynn, M. J.; Oberman, S. F.: Advanced Computer Arithmetic Design. John Wiley & Sons, 2001.

[6] M.D. Ercegovac and T. Lang, Digital Arithmetic, Morgan Kaufmann Publishers, Elsevier Science, 2004.

[7] Oklobdzija, V. G., Villeger, D., Liu, S. S.:” A Method for Speed Optimized Partial Product Reduction and Generation of Fast Parallel Multipliers Using an Algorithmic Approach”. *IEEE Transactions on Computers*, vol. 45, no.3, pp. 294-306, 1996.

[8] Wallace, C. S.: “A Suggestion for Fast Multipliers”. *IEEE Transactions on Computers*, vol. 13, no. 2, 1964.

[9] Dadda, L.: “Some Schemes for Parallel Multipliers”. *Alta frequenza*, vol. 34, pp. 349-356, 1965.

[10] Yeung, A. K. et al: A self-timed multiplier with optimized final adder, Univ. California, Berkeley, Final Rep., CS 2921, 1989.

[11] N. Takagi, H. Yasuura, S. Yajima. “High-Speed VLSI Multiplication Algorithm with a Redundant Binary Addition Tree”, *IEEE Transaction on Computers*, Vol C-34, no. 9. , 1985.

[12] Zimmermann, H.: Binary Adder Architectures for Cell-based VLSI and their Synthesis. PhD Thesis, Swiss Federal Institute of Technology, 1997.

[13] R. Lin et al; “A Novel Approach for CMOS Parallel Counter Design”, *Euromicro Conference, vol 1*, pp. 1112-1119, 1999.

[14] R. McIlhenny and M.D. Ercegovac, “On Using 1-out-of-n Codes for (p,q) Counter Implementations”. *Asilomar Conference on Signals, Systems, and Computers*, 1996.

[15] S.D. Peraris, “A 40 ns 17-bit Array Multiplier”, *IEEE Transaction on Computers*, vol. 20 pp 442-447, 1971.

[16] J. Fadavi-Ardekani, “M x N Booth Encoded Multiplier Generator Using Optimized Wallace Trees”, *IEEE Transactions on VLSI Systems*, vol. 1 no. 2, 1993.

[17] Stelling, P. F., Martel, C. U., Oklobdzija, V. G., Ravi, R.: “Optimal Circuits for Parallel Multipliers”. *IEEE Trans. on Computers*, vol. 47 (3), pp. 273-285, 1998.

[18] A. Weinberger, “4:2 Carry-Save Adder Module”, *IBM Technical Disclosure Bull*, vol. 23, 1981.

[19] M.D. Ercegovac et al, “Reciprocation, Square Root, Inverse Square Root, and Some Elementary Functions Using Small Multipliers”, *Transactions on Computers*. vol. 49 (7) pp. 628-637, 2000.

[20] J.M. García Chamizo et al, “Time-Precision Flexible Adder”, *10th IEEE ICECS* (2003) 994-997

[21] J.A. Piñeiro, J. Díaz Bruguera, “High-Speed Double-Precision Computation. of Reciprocal, Square Root, and Inverse Square Root”, *IEEE Transactions on Computers*. vol. 51 (12),1377-1388, 2002.

[22] R. Rajsuman. “Design and Test of Large Embedded Memories: An Overview”. *IEEE Design and Test of Computers*, vol. 18, no. 3, pp. 16-27, 2001.

[23] ISSCC Roundtable, “Embedded Memories for the Future”, *IEEE Design and Test of Computers*, vol. 20, n. 4, pp. 66-81, 2003.

[24] S.J.E. Wilton, N.P. Jouppi. “An Enhanced Access and Cycle Time Model for On-Chip Caches”. Digital Western Research Laboratory. 1994.

[25] T. Wada, S. Rajan, S.A. Przybylski, “An Analytical Access Time Model for On-Chip Cache Memories”. *IEEE Journal of Solid-State Circuits*, vol. 27 (8),1992.

[26] E.G. Walters, J. Schelessman, M.J. Schulte, “Combined Unsigned and Two’s Complement Hybrid Squarers”, 35th *Asilomar Conference on Signals, Systems, and Computers*, California, pp. 861-866, 2001