Synthesis of reconfigurable multiplier blocks: part I: fundamentals.

Suleyman Demirsoy[1]
Izzet Kale[1]
Andrew Dempster[2]

[1]Cavendish School of Computer Science, University of Westminster

[2]School of Surveying and Spatial Information Systems, University of New South Wales, Sydney, Australia

Synthesis of Reconfigurable Multiplier Blocks: Part I- Fundamentals

Süleyman Sırrı Demirsoy, Izzet Kale
Applied DSP and VLSI Research Group
University of Westminster
London, W1W 6UW, UK
{demirss, kalei}@wmin.ac.uk

Andrew G. Dempster
School of Surveying and Spatial Information Systems
University of New South Wales
Sydney, Australia
a.dempster@unsw.edu.au

***Abstract*— Reconfigurable Multiplier Blocks (ReMB) offer significant area, delay and possibly power reduction in time-multiplexed implementation of multiple constant multiplications. This paper and its companion paper (subtitled Part II- Algorithm) together present a systematic synthesis method for Single Input Single Output (SISO) and Single Input Multiple Output (SIMO) ReMB designs. This paper presents the necessary foundation and terminology needed for developing a systematic synthesis technique. The companion paper illustrates the synthesis method through examples. The method proposed achieves reduced logic-depth and area over standard multipliers / multiplier blocks.**

I. INTRODUCTION

Primitive Operator Filters [1] and multiplier blocks [2] are especially beneficial for the fully parallel implementation of digital filters and filter banks. They reduce the complexity of the implementation effectively, by exploiting the redundancy of the multiple constant multiplications. Multiplications by coefficients are realized by successive shift and add operations. The intermediate values that are formed during the generation of one coefficient are re-used for other coefficients, and thus reducing the computational redundancy. This topic has been studied extensively in the literature, and many algorithms were developed to design multiplier blocks or - in other words - multiple constant multiplications for different applications. These algorithms can be grouped into two, depending on their approach to the problem:

- Sub-expression sharing method; that works on the Signed Digit (SD) representations of a group of coefficients [3]-[7],
- Numerical (graphical) approach; where a group of coefficient products are generated using common intermediate products [1],[2],[8]-[11].

The savings that can be achieved in implementing fully parallel digital filters as a result of these techniques are impressive both in terms of area, complexity and power reduction [1]-[11].

In recent years, the application of the multiplier blocks to time-multiplexed digital filter designs was studied in [12]-[14]. The coefficient store and the general-purpose multiplier in Fig. 1(a) were replaced by a reconfigurable multiplier block (b), which can generate the required coefficient products with its different configurations. For the example in Fig. 1(b) the ReMB is a Single Input Single Output (SISO) block. A Single Input Multiple Output (SIMO) ReMB can replace the entire fixed multipliers in a bank of filters as shown in Fig 1(c).

It has been shown that the redundancy can be reduced and the resulting specialized multiplier design can be much more efficient in terms of area and computational complexity compared to the general-purpose multiplier with its associated coefficient store [12]-[14]. Guidelines for efficient realization were presented in [12], and an efficient, automated design algorithm based on the graphical approach was developed and reported in [13]. This algorithm was suitable for SIMO systems such as filter banks.

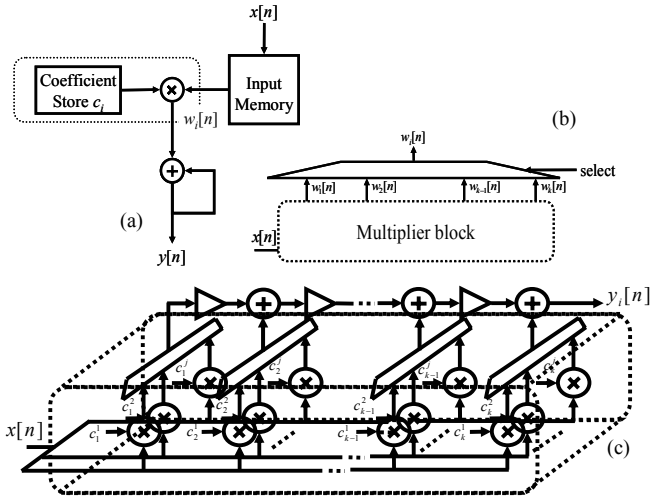

Figure 1 (a) Time-multiplexed Tappled Delay Line (TDL) (direct-form) FIR filter, (b) Conceptual SISO ReMB that would replace the coefficient store and the general purpose multiplier. (c) A SIMO ReMB system can replace the dashed box in a transpose direct form filter bank.

Efficient use of the resources on FPGA structures was studied in [15]. In this study, Turner reported significant savings in the area and delay of some DSP blocks by using the Reduced Coefficient Multiplier (RCM) that uses the configurable resources of a Field Programmable Gate Array (FPGA). His design method [16], which is based on common sub-expression sharing, combines the SD-encoded coefficients on to the Look-Up Tables (LUT) that exist in FPGAs and can be used for SISO and Multiple Input Single Output (MISO) blocks.

In this paper, we will present the fundamentals required and developed in [13] for a systematic synthesis of SISO and SIMO ReMB. Section 2 will focus on the basic structure topology. Section 3 will give the details of the developed foundation for SIMO and low logic-depth, with conclusions in Section 4.

II. Basic Structure Topology

All the examples in the paper are based on the simplest basic structure topology as shown in Fig. 2(a). In general, all ReMB designs are presented as directed-acyclic graphs where each line represents a connection. The (•) represents an adder or a subtractor or an adder/subtractor. One of its inputs is connected to a multiplexer. This basic structure can be configured to operate either on the (A, B) or (A, C) inputs by the help of the select line of the adder resulting in two configuration stages. Some of the possible variants of this topology are (A+B, A+C), (A+B, A-C), (A-B, A-C). These sets of operations are particularly important since they can be efficiently implemented in the Virtex FPGA, with no extra hardware cost for multiplexers [16].

Although the algorithm is structured to employ variants of this basic structure, the idea of how to design ReMB for a given coefficient set is applicable to any basic structure (See [12], [13] for other forms of basic structures and much detailed information on ReMB).

Fig. 2(b) shows two interconnected basic structure. The output produced by the first basic structure is fed to the input of the second one. The total number of outputs that can be produced by this structure is four. When three of the basic structures are interconnected as shown in Fig. 2(c) and (d) the number of coefficients that can be produced at the output (rightmost node) is eight.

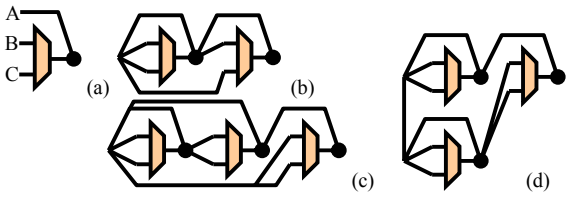

Figure 2 (a) Topology of the simplest basic structure, (b) An example for a cascade of two basic structures. Three basic structures interconnected in the (c) chain form, (d) tree form.

III. Foundation For Synthesis

A. *Efficient Handling of Multiple Outputs*

The multiplier block algorithm RAG-n presented in [2], built the coefficients in a given set one by one in an order generally defined by their costs (minimum number of interconnected adders to generate the coefficient) or magnitudes. The coefficients having the same costs still needed to be built in order, by making use of all the previously generated numbers (both the fundamentals and the coefficients) in the multiplier block. The multiple-output requirement of the multiplier block to be used in the transposed direct form filters (the multiplier block in Fig. 1(b) without the multiplexer) was realized by connecting the generated partial products or coefficients to the corresponding filter taps.

The efficient realization of multiple outputs in a ReMB design has to be different than multiplier blocks. Let us consider a typical time-multiplexed filter bank application as shown in Fig. 1(c), with output nodes $y_1, y_2, \ldots y_k$. Typically each output node of the ReMB has the same set size, i.e. the number of coefficients per output node is the same, which we shall assume to be M. The upper bound of the output set size of a ReMB design grows exponentially as the number of cascaded basic structures increases [13]. We further assume that an output node y_1 is built using several interconnected basic structures shown in Fig. 2(a) and have M different outputs. Any other output node, say y_2, built with the same type of basic structure cascaded to y_1 would typically have the capacity of 2M outputs. Since y_1 and y_2 both have the same output set size, the basic structure of y_2 becomes under-utilized.

One way to make sure that the output nodes are treated independently is to start designing from the output nodes and build the whole design step by step back to the input, as each output node would be a different starting point without any dependence on one another.

B. *Basic Structure Depth*

To avoid under-utilization, all output nodes in the design should have a similar number of interconnected basic structures when traced back to the input. The number of coefficients per output node would put a restriction on both the minimum number of basic structures required and the minimum depth of the ReMB design. For example, it was shown in previous section that, by using the simplest basic structure, a maximum of eight different numbers can be generated at depth 2. In the same way, the maximum number of outputs that can be achieved at a depth of three basic structures is 128 for a possible ReMB design shown in Fig. 3. The basic structures in the diagram are placed in layers to indicate the "basic-structure-depth" of that node.

The maximum number of outputs from a node is 2^{n_i} where i is the basic-structure-depth and n_i can be formulated recursively for ReMB designs comprising the simplest basic structure as follows:

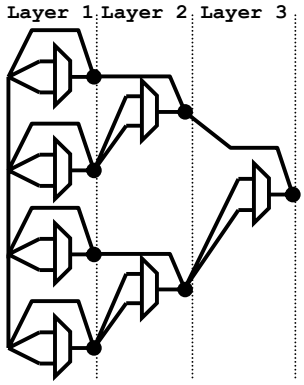

Figure 3 A ReMB design with a basic-structure-depth of three, which can produce 128 different coefficients at the output of layer 3.

$$n_i = 2n_{i-1} + 1 \quad (1)$$

It should be noted that, for a different basic structure, the maximum number of outputs per node would be different.

On the other hand, the individual coefficient costs put a separate restriction to the number of basic-structures interconnected for building a particular output node. For example a cost-3 coefficient needs at least three basic structures to be generated. They can either be in a chain form (Fig. 2c) or in a tree form (Fig. 2d). However it is shown that, the tree form interconnection of n adders/subtractors cannot produce all cost-n numbers in a multiplier block [10]. Therefore, a basic-structure-depth of n would ensure that a cost-n coefficient can be generated.

The basic-structure-depth is important when deciding the layer of an output node. To explain this, consider a node with the fundamental set {39, 45, 41, 47, 61, 11, 27, 57, 119}. All of the fundamentals are cost-2, i.e. each of them requires a cascade of two adders to be generated. However, since there are nine different numbers, the basic-structure-depth of the node would be at least three if the basic structures shown in Fig. 2(a) were to be employed, since the maximum number of outputs at depth-2 is eight. Here, the basic-structure-depth is dictated by the output set size. In a different example, the coefficient set {473, 181, 49} has three different numbers. The coefficient '49' is a cost-2 number whereas 473 and 181 are both cost-3. Again, assuming the simplest basic-structure is used, the output set size only requires a minimum of two interconnected basic structures. This time, the basic-structure-depth is dictated not by the output set size but by the cost of the coefficients, which is three. However, it should be kept in mind that, some cost-3 coefficients can be generated at depth-2. Choosing depth-3 guarantees to cover all the different topologies that generate cost-3 coefficients.

The basic-structure-depth can not always suggest the accurate layer of the output node by checking the coefficient set. For example, the set {9, 15} includes two cost-1 numbers. The coefficient '9' can be realized as (8+1), whereas '15' is generated as (16-1). For an FPGA implementation with a restricted set of basic structures as explained in Section 2, (8+1) and (16-1) cannot be combined on a basic structure. Therefore, the set {9, 15} needs to be designed in layer two.

As a summary, the lower bound to the basic-structure-depth of an output node is the maximum of two values. The first one is the minimum depth that can generate the required output set size. This value depends on the type of the basic structure employed in the design. The second one is the maximum of the adder-costs of the coefficients.

C. Graphs

The realization of any coefficient from a set of fundamentals can be represented on a graph as shown in Fig. 4. For a coefficient x, the 'graph' consists of a set of numbers {a, b, c, d} satisfying the equation:

$$x = ac \pm bd \quad (2)$$

where c and d are in the form of $\pm 2^r$, r being a natural number for integer x.

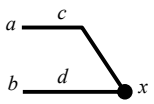

Figure 4 An example graph

Equation (2) results in more than one graph for a coefficient x when $\{a, b, c, d\}$ change in a pre-defined interval. Collecting all such graphs of a coefficient in a table, graph-tables are formed. Graph-tables can be employed in generating efficient ReMB designs.

D. Node-definition

A node-definition is a combination of graphs using a particular basic structure to produce a given coefficient set. Fig. 5 shows a node-definition for the coefficient set {K, L, M} on a basic structure. A, B_1 and B_2 are the inputs of the basic structure. c, d_1, and d_2 are the edge values. $[t_0\ t_1\ t_2]$ are the different configuration states of the resulting ReMB design. $[a_K, a_L, a_M]$, $[b_{1K}, X, b_{1M}]$ and $[X, b_{2L}, X]$ are the fundamental vectors holding the inputs of the basic structure for different configurations. The 'X' (don't care) in $[b_{1K}, X, b_{1M}]$ means the multiplexer does not use B_1 for configuration t_1 but rather uses b_{2L} from B_2 to produce the coefficient L. At configuration t_0, this node generates K as $K = a_K \times c + b_{1K} \times d_1$.

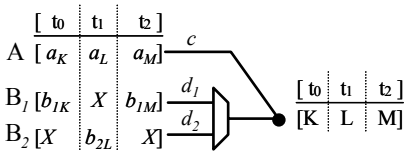

Figure 5 A generalized node-definition includes all the details about the node; edge values, and the fundamentals required to build the coefficient set for a given basic structure.

The graphs of the coefficients should be combined in such a way that, the basic-structure-depth of the resulting fundamental sets at A, B_1 and B_2, should be kept less than the basic-structure-depth of the coefficient set, otherwise the design would not converge back to the graph input. This implies that two parameters have to be decreased while choosing the graphs; the number of different fundamentals (fundamental set size) at an input, and the cost of the fundamentals. Assuming the basic-structure-depth of the coefficient set is three, the fundamentals at the input sets should be at most cost-2, and the fundamental set sizes can be at most eight (i.e. the maximum number of outputs allowed at that particular depth, see (1)).

The node-definitions satisfying the two requirements mentioned above could be found by processing the combinations of graphs that exist in the graph-tables. This method is explained further in the companion paper [17].

E. Algorithm Approach

Fig. 6 shows a typical symbolic SIMO ReMB example that can be generated by the algorithm. There are three output nodes, y_1, y_2, and y_3. As observed from the figure, all output nodes have a basic-structure-depth of three.

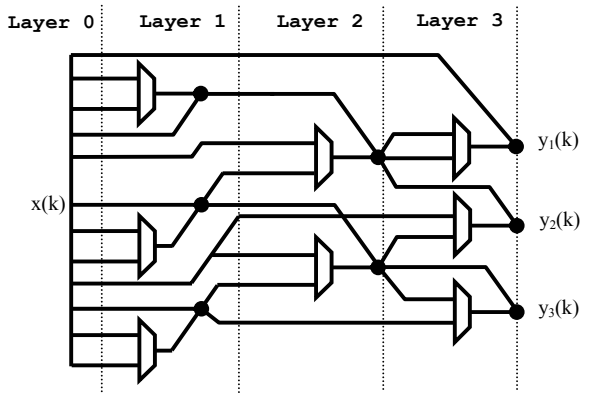

Figure 6 A symbolic diagram for SIMO ReMB

The layers partition the design into smaller units that can systematically be handled by the algorithm. Each layer has output nodes and fundamental sets that feed the basic structures. For an intermediate layer, the fundamental sets are the output nodes generated in the preceding layers. For layer 1, the fundamental set is always the input signal, which is represented as '1'. Starting from the last layer of the design, the algorithm recursively calls itself for each layer until it reaches the input signal. At each call, a number of coefficient sets or output nodes are processed by the algorithm to create node-definitions that generate the required coefficient sets. The fundamental sets that are required by these node-definitions are then designed by recursive calls of the algorithm.

IV. CONCLUSION

As a new design technique, ReMB needs new concepts to be developed for its efficient application. This paper presented new concepts to synthesize SISO and SIMO ReMB circuits. They form the foundation for the algorithm that is presented in the companion paper entitled as "Part II: Algorithm" [17].

The proposed technique divides the whole ReMB design into layers depending on the basic-structure-depth and deals with each layer recursively, starting from the output towards the input.

REFERENCES

[1] Bull D.R. and D.H Horrocks, "Primitive operator digital filters", IEE Proceedings-G, vol. 138, no. 3, pp. 401-412, June 1991.

[2] Dempster A.G. and Macleod M.D., "Use of minimum-adder multiplier-blocks in FIR digital filters", IEEE Trans. CAS-II, vol. 42, no. 9, pp. 569-577, November 1995.

[3] Bernstein R., "Multiplication by integer constants", Software-Practice and Experience, vol. 16, no. 7, pp. 641-652, Academic Press, New York, July 1986.

[4] Hartley R., "Subexpression sharing in filters using canonic signed digit multipliers", IEEE CAS-II, vol. 43, no.10, pp. 677-688, 1996

[5] Pasko R., et al, "A new algorithm for elimination of common sub-expressions", IEEE Trans. CAD ICS, vol. 18, pp.58-68, January 1999.

[6] Martinez-Peiro M., E.I. Boemo and L. Wanhammar, "Design of high-speed multiplierless filters using a nonrecursive signed common subexpression algorithm", IEEE Trans. CAS-II, vol. 49, no. 3, pp. 196-203, March 2002.

[7] Potkonjak M., M.B. Srivastava and A. P. Chandrakasan, "Multiple constant multiplications: Efficient and versatile framework and algorithms for exploring common subexpression elimination", IEEE Trans. on CAD of ICS, vol. 15, no. 2, pp. 151-165, February 1996.

[8] Li D., "Minimum number of adders for implementing a multiplier and its application to the design of multiplierless digital filters", IEEE Trans. CAS-II, vol. 42, no. 7, pp. 453-460, July 1995.

[9] Dempster A. G. and M.D. Macleod, "General algorithms for reduced-adder integer multiplier design", Elec. Letters, vol. 31, no. 21, pp. 1800-1802, October 1995.

[10] Gustafsson O., A. Dempster and L. Wanhammar, "Extended results for minimum-adder constant integer multipliers", IEEE ISCAS'2002, vol. 1, pp. 73-76, May 2002.

[11] Kang H.J. and I.C. Park, "Multiplier-less IIR filter Synthesis algorithms to trade-off the delay and the number of adders", Proceedings of IEEE ISCAS'01, vol. 2, pp. 693-696, Australia 2001.

[12] Demirsoy S. S., A.G. Dempster and I. Kale, "Design Guidelines for Reconfigurable Multiplier Blocks", IEEE ISCAS'03, vol. 4, pp. 293-296, Thailand, May 2003.

[13] Demirsoy S. S., "Complexity Reduction in Digital Filters and Filter Banks", Ph.D. Thesis, University of Westminster, October 2003

[14] Demirsoy S. S., R. Beck, A.G. Dempster and I. Kale, "Reconfigurable implementation of recursive DCT kernels with reduced quantization noise", IEEE ISCAS'2003, vol.4, pp. 289-292, Thailand, May 2003

[15] Turner R. H., T. Courtney and R. Woods, "Implementation of fixed DSP functions using the reduced coefficient multiplier", IEEE Proc. of ICASSP'2001, vol. 2, pp. 881-884, May 2001, USA

[16] Turner R. H., "Functionally diverse programmable logic implementations of digital signal processing algorithms", PhD Thesis, Queen's University of Belfast, August 2002.

[17] Demirsoy S. S., I. Kale, A. G. Dempster, "Synthesis of Reconfigurable Multiplier Blocks: Part II- Details of the Algorithm", to be publised in IEEE ISCAS'05.