Multiple Restricted Multiplication

Nalin Sidahao, George A. Constantinides, and Peter Y.K. Cheung

Department of Electrical & Electronic Engineering, Imperial College, London, UK,
{nalin.sidahao, g.constantinides, p.cheung}@imperial.ac.uk

Abstract. This paper focuses on a class of problem relating to the multiplication of a single number by several coefficients that, while not constant, are drawn from a finite set of constants that change with time. To minimize the number of operations, we present the formulation as a form of common sub-expression elimination. The proposed scheme avoids the implementation of full multiplication. In addition, an efficient implemenation is presented targeting the Xilinx Virtex / Virtex-II family of FPGAs. We also introduce a novel use of Integer Linear Programming for finding solutions to the minimum-cost of such a multiplication problem. Our formulation results in area savings even for modest problem sizes.

1 Introduction

For Digital Signal Processing (DSP) or arithmetic intensive applications, multiplication is considered as an expensive operation. This is because, typically, the main part of the area consumed in their implementation comes from multipliers. For a constant coefficient multiplication, instead of using a full multiplier, a general method to efficiently reduce the hardware usage is to use a series of binary shifts and adders. A shift operation may have almost negligible cost since it is hard-wired. Therefore, the total hardware cost is approximately corresponding to the area of adders required.

Reducing the number of adders in constant multiplication is an optimization problem. The key point of most existing research is the minimization of this quantity, which is an NP-hard problem [1].

By contrast, our approach introduces a form of the common sub-expression (CSE) elimination problem, which we refer to as multiple restricted multiplication (MRM). This refers to a situation where a single variable is multiplied by several coefficients which, while not constant, are drawn from a relatively small set of values. Such a situation arises commonly in synthesis due to resource sharing, for example in a folded implementation of a FIR filter [2] or a polynomial evaluation using Estrin's method [3,4]. Recent FPGA architectures have included dedicated multiplier blocks. By exploiting our technique, these blocks are freed to be used for true general multiplications.

Existing approaches to CSE are unable to take advantage of such a situation, resulting in the use of expensive general multipliers, as shown in Fig. 1. Fig. 1(a) shows a Data Flow Graph (DFG) with input x tied together, input sets of

J. Becker, M. Platzner, S. Vernalde (Eds.): FPL 2004, LNCS 3203, pp. 374–383, 2004.

constant multiplicands labelled as $\{c_{11}, c_{12}, \ldots, c_{1T}\}$ and $\{c_{21}, c_{22}, \ldots, c_{2T}\}$. The first subscript here refers to the spatial index and the second to the time index, i.e. c_{it} is the value of multiplicand i at time t. A standard technique using ROMs and multipliers is depicted in Fig. 1(b) and our approach performing equivalently is shown as a "black block" in Fig. 1(c).

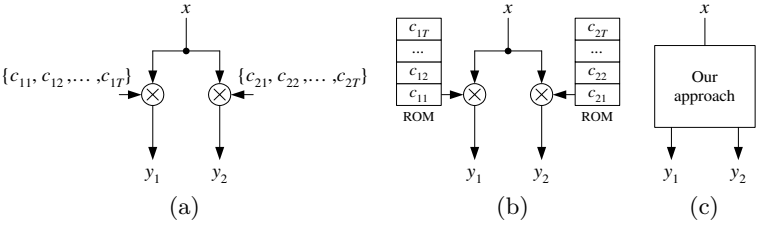


Fig. 1. (a) A DFG with only multiplier nodes, one input x, and other two inputs of multipliers. (b) The standard implementation with ROMs and Multipliers. (c) A black box interpretation of our approach.

In this paper, it is shown that the MRM problem can be addressed through extending the basic unit of operation from an addition, used in multiple constant multiplication (MCM) [7,8], to a novel adder-multiplexer combination. It is further demonstrated that for Xilinx-based implementations, the Xilinx Virtex / Virtex-II slice architecture [9] can be used to implement such a basic computational unit with no area overhead compared to the equivalent adder used in MCM.

A similar technique was previously presented by R.H. Turner, *et al.* [5] for implementing multipliers with a limited range of coefficients, which we extend by making use of the dedicated AND gate presented in the slice. The key is to exploit the set of primitive components: the supplementary logic gates, next to each LUT, and the dedicated carry-chain logic. Full utilization of these allows the implementation of an adder and/or a substractor along with a multiplexer in a novel configuration. This can be applied to constant multiplication using sub-expression sharing to achieve efficient FPGA implementation. A recent work by S.S. Demirsoy, *et al.* [6] has begun to address this problem using the type of computational node demostrated in [5].

Since MCM is NP-hard [1] and is a special case of MRM, it follows that MRM is NP-hard. Thus in order to find the area-optimal implementation of a given MRM block, a novel formulation of the optimization problem as a class of Integer Linear Program (ILP) is proposed. This approach allows us to leverage the recent advances in the field of ILP solution.

This paper therefore has the following novel contributions: 1. the introduction of the MRM problem, and its solution using a novel extension of adder-multiplexer cells. 2. the formulation of the minimum-area MRM as ILP formulation. 3. an efficient use of the Xilinx Virtex / Virtex-II slice structure to implement MRM.

This paper has the following structure. Section 2 describes the background of MCM. Section 3 describes the proposed architectural solution to the MRM problem, and Section 4 demonstrates that this solution can be efficiently implemented in the Xilinx Virtex family of FPGAs. Section 5 formulates the optimization problem in ILP, Section 6 collects and discusses results from MRM, and Section 7 concludes the paper.

2 Background

2.1 Multiple Constant Multiplication

MCM, a special case of the MRM problem address in this paper, has been used in many research fields, especially in DSP applications. A common use of MCM is within the design of fully unfolded digital filters [2]. The main idea of this technique involves removing the redundancy inherent in the re-computation of common sub-expressions. Applying this approach provides a significant reduction of area necessary to implement multiple constant multiplications. Even within a single multiplication, common sub-expressions exist. As an example, consider integer multiplication with constant coefficient 10100101. Let us denote left-shift as $<<$. Instead of performing $(x << 7) + (x << 5) + (x << 2) + x$ where x is an input variable, we can perform $(y << 5) + y$ where $y = (x << 2) + x$. Hardware is then saved due to the elimination of the 101 $(x << 2) + x$ sub-expression. Sharing such sub-expressions across several coefficients results in significant savings.

2.2 Representing Multiplicands with Data Flow Graphs

DFGs are the basis of a computational model used extensively in DSP. A DFG is a directed graph, with nodes in one to one correspondence with operations and edges in one to one correspondence with data flow. We shall consider edge-weighted DFGs, where the edge weight corresponds to a shift operation. For example, the CSE case considered above may be represented as a DFG in Fig. 2.

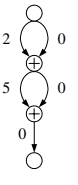


Fig. 2. DFG representation of coefficient 10100101

The topmost and bottommost nodes are the initial node (input node) and terminal node (output node), respectively. In general, a DFG may have more than one terminal node, corresponding to the different constant coefficients.

Each intermediate node (an adder in standard MCM), has two input edges and at least one output edge. Each DFG represents a way of sequencing addition operations such that the required coefficients are produced. We may then ask the optimization question: What is the minimum number of nodes required to compute a given set of constant coefficients? This is the problem addressed by existing work on MCM [7,8], which we extend here to the case of MRM.

3 MRM with Adder/Multiplexer Nodes

We now extend the DFG model from using adder nodes to using adder/multiplexer nodes. Each node's internal structure now consists of not only an adder, but also a 4-1 multiplexer, as shown in Fig. 3.

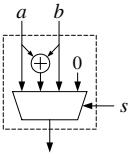


Fig. 3. An adder/multiplexer node

Using such circuit provides operations that perform adding $(a + b)$, passing one of two input values (a, b) through the multiplexer, and generating a zero to output value. Each operation is selected by a 2-bit selector s. Applying this technique provides the flexibility to move from MCM to MRM. Each node may perform a different operation at each different time frame. For instance, when applied to Fig. 2, the multiplicand can be 10100101 (lower node and upper node adding), 10000100 (lower node adding, upper node passing through the left-hand input) or some other coefficients; depending on what the selectors are. The DFG structure, and the shift quantities, remain constant over all time steps. This means that the structure can be directly mapped into a circuit, and the shifts remain cost-free.

4 An Efficient Xilinx Virtex / Virtex-II Implementation

This section describes an efficient adder/multiplexer implementation in a Xilinx Virtex / Virtex-II device. The Virtex series utilizes a Configurable Logic Block (CLB) architecture. Each CLB consists of two slices, each containing two logic cells (LCs). A diagram shown in Fig. 4 is a simplified Virtex-II architecture; more information can be found in [9]. An LC has several logic components including one 4-input look-up table (LUT), some MUXes and some dedicated logic, including one MUXCY, one XORCY, and one MULT_AND gate.

The adder/multiplexer node is designed using a similar idea to the way a ripple-carry adder is implemented using the Virtex carry chain. A full adder/mul-

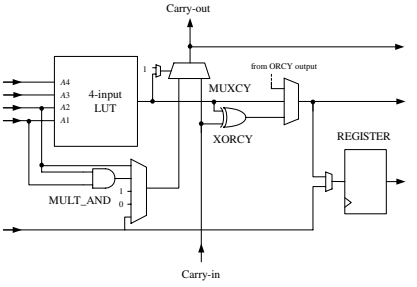
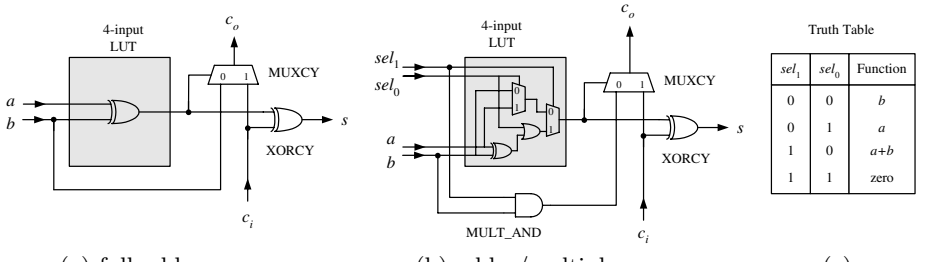


Fig. 4. Simplified Virtex-II structure (top half of a slice)

tiplexer can be efficiently implemented using the same logic hardware resources as a common adder. The key is leveraging MULT_AND gate, an additional 2-input dedicated AND gate (typically used to implement an efficient 1-bit multiplier), and the carry-chain logic.

Fig. 5(a) illustrates a simple 1-bit full-adder which performs addition two inputs a, b and carry-in c_i, and results the summation s and carry-out c_o. We can extend this structure to perform a bit-slice of the entire function shown in Fig. 3. The four operations are controlled by a 2-bit selector (sel_1, sel_0) absorbed into the 4-input LUT. A 1-bit adder/multiplexer can fit in one LC as depicted in Fig. 5(b).



(a) full adder (b) adder/multiplexer (c)

Fig. 5. 1-bit adder and adder/multiplexer implementation in an LC

In order to obtain the operation of generating logic "0" at all bit outputs of an adder/multiplexer node, an extra AND gate is required. Two selector signals, sel_1 and sel_0, are the inputs of this gate. When both signals are "1", we make use of a "1" on its output to force on the carry-in of LC that computes the LSB of adder/multiplexer, and thus also on an input of the XOR gate (XORCY) in the carry-chain logic. Meanwhile, a logic "1" is obtained on LUT output connected to the other input of each XORCY gate. This yields "0" at the output s. The logic of LUT output also selects carry-in value pass through multiplexer (MUXCY) for forcing the carry-in of the next upper cell to operate in a similar manner.

B one-bit adder/multiplexers are implemented vertically providing a B-bit adder/multiplexer to perform fast carry-chain addition and multiplexing. The carry-in signal is applied to at the bottom LC of the structure, and is cascaded upwards by the Virtex architecture. This allows the design to have minimum propagation delay.

The proposed structure is an efficient implementation on the Virtex device by fully utilizing the resources of the cell. This provides more functionality with minimal extra cost required. Although an additional AND gate is required, this is a very small logic overhead compared to the size of B-bit add/multiplexer.

5 Transformation into ILP Formulation

A given set of MRMs may or may not be implementable using a fixed number N of computational nodes. In this section, we propose an ILP model, the solution to which corresponds to a time-ordered sequence of multiplexer select lines to implement the required behaviour, if one exists.

5.1 Representing General DFGs

Fig. 6(a) depicts a general structure for describing all computations containing three adder/multiplexer nodes (higher node structures can be developed in a similar fashion). For clarity, we use a square box to represent each input and output node, and a path with a big black dot to perform shift operation. The multiplexers in Fig. 6(a), labelled "model multiplexers", will not be realized in the final circuit; they provide a model for ILP problem thus allowing all DFGs of N nodes to be modelled. Once implemented, these multiplexers are replaced by wires, as only one value of the select lines is active, for all time. The proposed model therefore contains three main components: shifter, adder/multiplexer and model multiplexer. This structure can perform various operations depending on path selection of all adder/multiplexers and model multiplexers. The number of outputs, which we shall denote C, corresponds to the number of sets of time-varying coefficient(s).

5.2 Encoding the Problem

An instance of the problem is encoded as a $T \times C$ matrix, where T is the number of rows corresponding to the number of time steps and C is number of columns representing outputs. This implies that a $1 \times C$ matrix corresponds to the standard MCM problem. For example, Fig. 1 is a $T \times 2$ problem. Since the MRM problem includes a time element, the first step of our algorithm is to unroll over time. This is accomplished by repeating the general graph. For T-time steps, we require overall T repetitions; all signals that control each corresponding shifter and model multiplexer are tied together. This ensures that shifting and routing for all graphs (all T) are the same. The only select line allowed to change with the time is the select line internal to each adder/multiplexer node, which can be changed to achieved the desired output values.

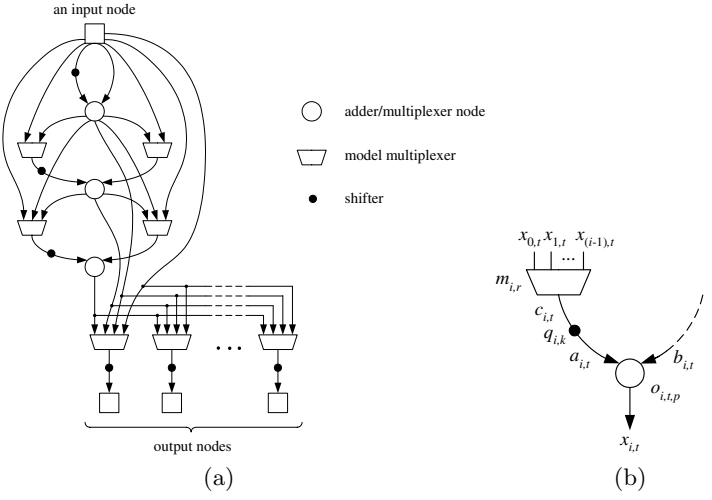


Fig. 6. (a) A general DFG of three adder/multiplexer nodes. (b) A portion of general structure.

5.3 ILP Modelling

ILP models are able to provide a formal method to describe and solve the MRM problem. Our approach is to model the N-node problem as an ILP for fixed N, and then to iterate ILP solver to find the lowest value of N resulting in a feasible solution. For minimizing the $T \times C$ matrix problem, suppose that there are total N add/multiplexer nodes operating in a B-bit number system.

A portion of the general structure is depicted in Fig. 6(b) and its notations, described below, will be required for understanding the ILP model.

Both integer and binary variables are used within the model:

- The integer variables $a_{i,t}$ and $b_{i,t}$ are the inputs of i^{th} adder/multiplexer node at step t and its output is represented by variable $x_{i,t}$. The model multiplexer has i inputs, corresponding to the previous node outputs, and its own output is represented by variable $c_{i,t}$.
- The binary decision variables $o_{i,t,p}$ represent which of the four operations to be performed at adder/multiplexer node i during time step t, where $p \in \{0, 1, 2, 3\}$. Variables $m_{i,r}$ represent the selection of input $x_{r,t}$ to the model multiplexer, where $r \in \{0, 1, \ldots, i-1\}$. Finally, variables $q_{i,k}$ represent the degree of shifting : $q_{i,k} = 1$ means that this input of node i should be shifted left by k bits, where $k \in \{0, 1, \ldots, B-1\}$.

We therefore have the following constraints, which are not yet in linear form:
1. Model multiplexer function

$$c_{i,t} = x_{r,t} \text{ if } m_{i,r} = 1. \tag{1}$$

2. Shifter function

$$a_{i,t} = 2^k c_{i,t} \text{ if } q_{i,k} = 1. \tag{2}$$

3. Adder/multiplexer function

$$x_i = \begin{cases} b_{i,t} & \text{if } o_{i,t,0} = 1 \quad \text{(operation 0)} \\ a_{i,t} & \text{if } o_{i,t,1} = 1 \quad \text{(operation 1)} \\ a_{i,t} + b_{i,t} & \text{if } o_{i,t,2} = 1 \quad \text{(operation 2)} \\ 0 & \text{if } o_{i,t,3} = 1 \quad \text{(operation 3).} \end{cases} \tag{3}$$

A problem now arises since all above constraints are nonlinear problems, due to the "if" selectors. However, these constraints can be reformulated as linear constraints in the following way. For example, in the model multiplexer,

$$c_{i,t} = x_{r,t} \text{ if } m_{i,r} = 1 \quad \Rightarrow \quad c_{i,t} - x_{r,t} = 0 \text{ if } m_{i,r} = 1 \tag{4}$$

which is equivalent to

$$c_{i,t} - x_{r,t} \leq \alpha(1 - m_{i,r}) \tag{5}$$

and

$$c_{i,t} - x_{r,t} \geq \beta(1 - m_{i,r}). \tag{6}$$

where α and β are known finite lower and upper bound on the left-hand side of (5) and (6), respectively. For the unsigned binary number system, $\alpha = 2^B - 1$ and $\beta = -2^B + 1$ are sufficient. We can see that $m = 1$ reduces (5) and (6) to (4). When $m = 0$, $c_{i,t}$ and $x_{r,t}$ can be any values (0 to $2^B - 1$) and still satisfy (5) and (6). Extending this approach to other constraints results the reduction of (1)– (3) to linear constraints problem to be an ILP.

There are a number of additional equality constraints that need to be added:

$$\text{For all nodes } i, \qquad \sum_{r=0}^{i-1} m_{i,r} = 1 \tag{7}$$

$$\text{For all nodes } i, \qquad \sum_{r=0}^{B-1} q_{i,k} = 1 \tag{8}$$

$$\text{For all nodes } i \text{ and time steps } t, \qquad \sum_{p=0}^{3} o_{i,t,p} = 1 \tag{9}$$

where constraint (7) states that multiplexer has to select only one input, (8) states that shifter must be shifted by only one k, (9) states that only one operation has to be performed at any one time step.

The minimum area solution for a $T \times C$ problem can be obtained by proceeding as follows:

1. Set $N = 1$.
2. Determine whether a feasible solution exists.
3. If it does, terminate the process, otherwise increase N and repeat from step 2.

6 Results

We compare our approach to two methods: the ROM and general multiplier approach shown in Fig. 1(b), and the unfolded use of MCM. The latter approach consists of using MCM to create the optimum implementation of all TC coefficients (for $T \times C$ problem), and then using T-to-1 MUXes to select the output at each time step.

Note that both comparative approaches can be considered as special cases of the general MRM structure used. $N \times M$ general multiplier can be considered as a cascade of M N-bit adder/multiplexer nodes, where the shift is always unity and the select line, controlled by the equivalent bit in the multiplicand, selects between options 1 and 2 in (3). The MCM-MUX comparative approach is also a special case, where MCM is performed by adder/multiplexer nodes always fixed at option 2 of (3), and the multiplexing is performed by adder/multiplexer nodes which can choose between options 0 and 1 of (3).

These approaches were tested using sets of 4-bit coefficients generated randomly. All ILP models are solved using the MOSEK optimization software [11]. Table 1 shows the synthesis results of average area and delay targeting Xilinx Virtex-II XC2V1000-4 device [9].

As with the MCM problem, it is expected that the area improvement grows with problem size [8]. However, even for the small benchmarks, our approach compared to MCM-MUX provides less area for the larger instances shown in Table 1. Compared to ROM and multiplier, the crossover point occurs at the 3×3 problem which results in a 24% improvement. It is likely that for large problems, even greater saving will be possible.

Table 1. Average area and propagation delay. The upper figure is the area (in slices), the lower figure is the delay (nanoseconds)

		ROM & Multiplier			our approach			MCM-MUX		
		Number of outputs C			Number of outputs C			Number of outputs C		
		1	2	3	1	2	3	1	2	3
Number of time steps T	1	3.3, 12.27	4.7, 11.66	6.4, 11.71	5.9, 16.65	6.5, 16.51	8.9, 17.71	5.9, 16.59	6.5, 15.92	8.9, 17.32
	2	6.6, 15.22	11.4, 17.18	19.4, 16.91	10.2, 18.26	12.6, 19.14	21.1, 19.99	11.2, 18.15	15.7, 18.17	23.6, 19.15
	3	9.9, 17.05	18.7, 16.82	29.5, 18.04	10.8, 18.89	18.7, 20.47	22.3, 20.70	14.0, 18.53	23.0, 18.24	30.5, 18.02

Since we do not explicitly target delay, the maximum average delay of our approach is 51% longer than that of ROM and general multiplier and 22% of MCM-MUX approach. However, it would be straight-forward to incorperate DFG path length based delay into the ILP objective function, if this were a problem.

7 Conclusion

To our knowledge, there is no existing algorithm to directly deal with the MRM problem. We introduce a new approach to optimize this multiplication problem by formulation into ILP one and employ an efficient ILP software to find the solution. Although such technique have limitations when the problem becomes very large, the results obtained give us important measures of optimality for future developments of a heuristic approach. We also present how to take advantage of all of the hardware present in the Virtex / Virtex-II family to ensure optimal area results. Our further work aims to develop such heuristic approaches, and to exploit dedicated registers for further time-step based optimization.

References

1. K. Matsuura and A. Nagoya. Formulation of the Addition-Shift-Sequence Problem and Its Complexity. In *Proc. 8^{th} Int. Symp. on Algorithms and Computation (ISAAC)*, pages 42–51, Singapore, December 1997.
2. K.K. Parhi. *VLSI Digital Signal Processing Systems : Design and Implementation.* Wiley-Interscience, 1998.
3. J. Villalba, G. Bandera, M. A. Gonzalez, J. Hormigo, and E. L. Zapata. Polynomial evaluation on multimedia processors. In *Proc. Int. Conf. Application Specific Systems, Architectures and Processors (ASAP)*, San Jose, California, 2002.
4. G. Corbaz, J. Duprat, B. Hochet and J.-M. Muller. Implementation of a VLSI polynomial evaluator for real-time applications. In *Proc. Int. Conf. Application Specific Array Processors*, pages 13–24, September 1991.
5. R.H. Turner, T. Courtney, R. Woods. Implementation of fixed DSP functions using the reduced coefficient multiplier. In *Proc. Int. Conf. Acoustics, Speech, and Signal Processing*, pages 881–884, May 2001.
6. S.S. Demirsoy, A.G. Dempster and I. Kale. Design guidelines for reconfigurable multiplier blocks In *Proc. Int. Symp. on Circuits and Systems (ISCAS)*, volume 4, pages IV-293–IV-296, May 2003.
7. M. Potkonjak, M.B. Srivastava and A.P. Chandrakasan. Multiple constant multiplications: efficient and versatile framework and algorithms for exploring common subexpression elimination *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, 15(2):151–165, 1996
8. R. Pasko, P. Schaumont, V. Derudder and D. Durackova. Optimization method for broadband modem FIR filter design using common subexpression elimination In *Proc. 10^{th} Int. Symp. on System Synthesis (ISSS)*, pages 100–106, September 1997.
9. Xilinx Documentation and Literature, available from http://www.xilinx.com/literature/index.htm; accessed 15 March 2004.
10. R.S. Garfinkel and G.L. Nemhauser. *Integer Programming.* Wiley and Sons, 1972.
11. MOSEK ApS optimization software, available from http://www.mosek.com; accessed 15 March 2004.