POWER AND AREA OPTIMIZATION FOR MULTIPLE RESTRICTED MULTIPLICATION

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

Department of Electrical & Electronic Engineering,
Imperial College London,
Exhibition Road, London SW7 2BT, England
email: nalin.sidahao@imperial.ac.uk, george.constantinides@ieee.org, p.cheung@imperial.ac.uk

ABSTRACT

This paper presents a design and optimization technique for the Multiple Restricted Multiplication problem [1]. This refers to a situation where a single variable is multiplied by several coefficients which, while not constant, are drawn from a finite set of constants that change with time. The approach exploits dedicated registers in FPGA architecture for further time-step based optimization over previous approaches [1, 2]. It is also combined with an effective technique, based on high-level power modelling, for power optimization. The problem is formulated into an integer linear program for finding solutions to the minimum-costs. The new approach results up to 22% area saving compared to the optimal non-register approach in [1], and 80% of all results also show 21%-48% power savings.

1. INTRODUCTION AND BACKGROUND

In the past, there has been some work on a multiplication problem where, for each multiplier, one multiplicand consists of a set of coefficients which change with time. Such an operation could be considered as time-multiplexed multiplication [1, 2, 3, 4]. A type of this problem referred to as Multiple Restricted Multiplication (MRM) was proposed in [1]. MRM 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 commonly appears in folded implementation of FIR filter [5], and polynomial evaluation [6, 7]. Fig. 1(a) illustrates a general structure that is a straightforward approach using ROMs and multipliers to implement this multiplication. The structure in Fig. 1(a) contains several multipliers with a common input x, and sets of constant multiplicands labelled as $\{c_{11}, c_{12}, \ldots, c_{1T}\}$, $\{c_{21}, c_{22}, \ldots, c_{2T}\}$, $\ldots$, $\{c_{C1}, c_{C2}, \ldots, c_{CT}\}$, where C and T are the number of the sets or outputs, and the number of the coefficients in each set, respectively. 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.

The work on this paper was supported by a Royal Society grant (2004/R1).

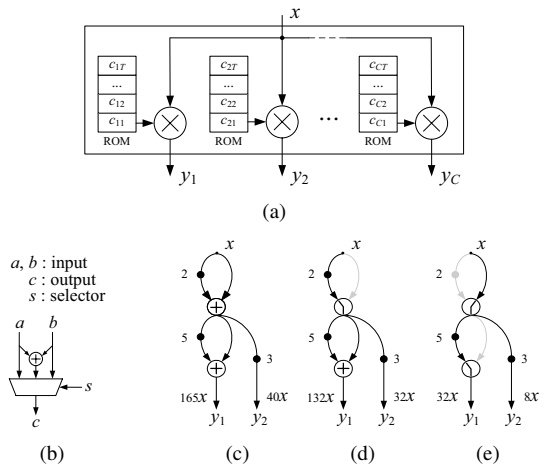

Fig. 1. (a) A straightforward approach using ROMs and multipliers for MRM. (b) An adder/multiplexer node, and configurations to create the coefficient sets $\{165, 132, 32\}$ and $\{40, 32, 8\}$ in (c) 1^{st} time step, (d) 2^{nd} time step, and (e) 3^{rd} time step. (Black dots represent left-shift operations)

In [1], it was shown that the MRM problem can be addressed through extending the basic unit of operation from an addition, used in Multiple Constant Multiplication (MCM) [8], to an adder/multiplexer combination shown in Fig. 1(b). As an example, assume we have a single variable input x multiplied by two sets of coefficients $\{165, 132, 32\}$ and $\{40, 32, 8\}$. An optimized implementation of this MRM problem can be seen in Figs. 1(c)–(e), and is described below.

Fig. 1(c) is recognizable as a standard MCM solution, containing two addition nodes, and generating the two values $165x$ and $40x$. Figs. 1(d) and (e) illustrate how the same Data Flow Graph (DFG) structure can be used to obtain the remaining coefficients by selecting the behaviour of the nodes from the possibilities shown in Fig. 1(b).

Recent work on time-multiplexed multiplication that relate to MRM has appeared in the literature [2, 4]. The approaches essentially work by constructing such multiplication as a combination of adder/subtractor/multiplexer node. The most important feature is that for Xilinx-based implementations, implementing these nodes requires the same resources that would be used in an addition. This provides an efficient design in terms of area complexity reduction. In [4], it was investigated that using adder/subtractor-multiplexer combination can be configured up to 63 different cells. These are applied in design methodology for implementing multipliers with a limited range of coefficients. The work by [2] has begun to address this problem using the type of computational node demonstrated in [4].

Due to demanding increases in low-power electronic applications, power consumption has become a critical issue. As a result, in this paper, we present a design and optimization technique for the MRM problem by extending previous approaches to exploit the dedicated registers in a Virtex slice, and introducing a technique to optimize power consumption.

For a given set of coefficients, it is possible to have more than one feasible solution with the same area. Although in area terms there are equivalent, the solutions are distinct in the way that the functions of each node are performed as well as the graph topologies. Distinct changing from one function to another causes different signal transitions or switching activity. From the power consumption point of view, it is advantageous to minimize the switching. The proposed approach introduces a high-level power macro-model which is a function of the number of adder/multiplexer nodes, and of switchings at output nodes caused by the changing of node function. From empirical evidence, these parameters show considerable impact on power optimization. The parameters are used to form a cost function for guiding the optimization process when it is formulated into an Integer Linear Program (ILP) [9].

This paper therefore makes the following novel contributions: 1. To our knowledge, the first algorithm dealing with time-multiplexed multiplication that exploits dedicated registers for optimized FPGA-based implementation. 2. The first quantification of the power savings possible on time-multiplexed multiplication. 3. Introduction of power macro-modelling for the MRM problem. 4. Extended formulation of the minimum-area MRM and power cost as ILP formulation. The rest of this paper is organized as follows: Section 2 demonstrates an efficient implementation of adder/multiplexer exploiting dedicated register on Xilinx Virtex family of FPGAs for MRM solution. Section 3 presents representing general DFGs for MRM problem and then formulating into ILP is given in Section 4. An approach for power optimization is described in Section 5. Section 6 collects results and concludes the paper.

2. AN EFFICIENT IMPLEMENTATION OF ADDER/MULTIPLEXER WITH DEDICATED REGISTER

The main functional block of Virtex series FPGAs [10] is Configurable Logic Block (CLB). It consists of slices each containing two Logic Cells (LCs). The simplified LC structure is illustrated in Fig. 2(a). Fig. 2(b) demonstrates one possible configuration used by the approach presented in this paper. Two functions perform passing an input a through the multiplexer, and 1-bit adding between an input b and a previous output q stored in register. They are controlled by a selector sel. For a B-bit structure, B cells are required to construct vertically upward allowing using fast carry chain.

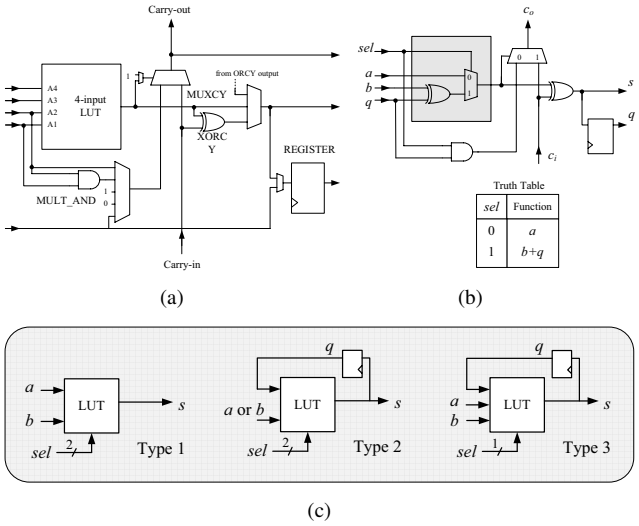

Fig. 2. (a) Simplified LC structure of Virtex series architecture. (b) An example of configuration that exploits dedicated register. (c) Three types of configuration.

2.1. Node Functions of Adder/multiplexer with Register for MRM Problem

It can be seen that the 4-input LUT can be configured to be various functions. If we allow the dedicated register output (the q variable in Fig. 2(b)) to be an input to those functions, so we will have more configurations possible than there in previous work [1]. This increases the possibility for more resource sharing leading to minimization of the hardware resources.

Since the LUT has 4 inputs, we distinguish two conditions: either 2 variables and 2 selectors, allowing switching between up to 4 functions, or 3 variables and 1 selector, allowing switching between up to 2 functions. These are shown as Fig. 2(c).

Our approach focuses on 7 common functions: a, b, $a+b, a+q, b+q, 2q$ and q. The possible configurations are classified as three types (Type 1, 2 and 3) as shown in

Table 1. Listing of all possible configurations by type

	Type 1	Type 2	Type 3
Configuration	$a, b, a+b$	$a, a+q, 2q, q$	$a, b+q$
		$b, b+q, 2q, q$	$b, a+q$
			$a+b, a+q$
			$a+b, b+q$
			$a+b, 2q$
			$a+b, q$
			$a+q, b+q$

Fig. 2(c), and listed in columns by type in Table 1. For example, there is only one configuration of Type 1, which allows the selector to switch between the operations a, b, and $a+b$. For type 2, there are two possible configurations, with the options shown in Table 1.

3. REPRESENTING GENERAL DFGS FOR MRM PROBLEM

The computational model of MRM developed from [1] is illustrated in Fig. 3(a). It shows a general DFG for problem model of three computational nodes (N) corresponding to adder/multiplexer with register. Higher node graph structures can be developed in a similar fashion. A square box represents each input and output node, and a path with a big black dot to perform shift operation. The multiplexers labelled model multiplexers will not be realized in the final circuit; they provide a model for the ILP problem thus allowing all graphs 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.

This structure can perform various operations depending on path selection of all nodes of adder/multiplexer with register, and model multiplexers. The number of outputs, which we shall denote C, corresponds to the number of sets of time-varying coefficient(s). 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, as illustrated in Table 1.

Fig. 3(b) shows a portion of the general structure and the detail of a computational node. To perform the function that corresponds to register, an output of the i^{th} node of model in previous time step $t-1$ ($x_{i,t-1}$) will be an input for the i^{th} node of the model in t. All notations provided here are explained in detail to discuss in ILP in Section 4 and 5.

4. TRANSFORMATION INTO ILP FORMULATION

An instance of the problem is encoded as a $T \times C$ matrix, where T is the number of rows corresponding to the number

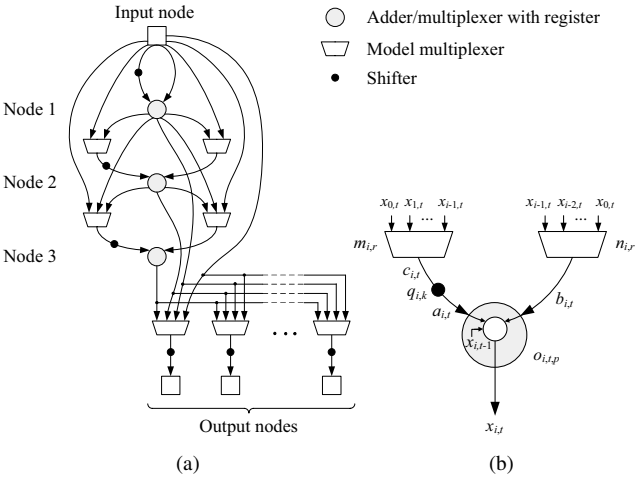

Fig. 3. (a) A general DFG of a 3-node structure. (b) A portion of the structure: detail of a computational node corresponding to the adder/multiplexer with register.

of time steps and C is number of columns representing outputs. As the number of computational nodes corresponds to the area, finding the minimum-node solution is the area optimal solution for the $T \times C$ problem. In this section, a set of ILP is presented, which is feasible iff the problem can be solved using a fixed number, N, of nodes.

There are three main components: model multiplexer, shifter and adder/multiplexer with register, as shown in the proposed model in Fig. 3. The model consists of integer and binary variables. In (1), the binary decision variables $m_{i,r}$ represent the selection of input $x_{r,t}$ to the model multiplexer, for node i at time step t, $r \in \{0, 1, \ldots, i-1\}$. In (2), variables $q_{i,k}$ represent the degree of shifting: $q_{i,k} = 1$ means that $a_{i,t}$ is the value $c_{i,t}$ that is shifted left by k bits, where $k \in \{0, 1, \ldots, B-1\}$ and B is the number of bits used. Finally, in (3), the binary variables $o_{i,t,p}$ represent which of the seven operations listed in Table 1 is to be performed at node i during time step t, where $p \in \{0, 1, \ldots, 6\}$.

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 with register function

$$x_{i,t} = \begin{cases} a_{i,t} & \text{if } o_{i,t,0} = 1 \text{ (option 0)} \\ b_{i,t} & \text{if } o_{i,t,1} = 1 \text{ (option 1)} \\ a_{i,t} + b_{i,t} & \text{if } o_{i,t,2} = 1 \text{ (option 2)} \\ a_{i,t} + x_{i,t-1} & \text{if } o_{i,t,3} = 1 \text{ (option 3)} \\ b_{i,t} + x_{i,t-1} & \text{if } o_{i,t,4} = 1 \text{ (option 4)} \\ 2x_{i,t-1} & \text{if } o_{i,t,5} = 1 \text{ (option 5)} \\ x_{i,t-1} & \text{if } o_{i,t,6} = 1 \text{ (option 6)} \end{cases} \tag{3}$$

4. One source node for each input

$$\forall i,\ \sum_{r=0}^{i-1} m_{i,r} = 1 \tag{4}$$

$$\forall i,\ \sum_{r=0}^{i-1} n_{i,r} = 1 \tag{5}$$

5. One shift for each node

$$\forall i,\ \sum_{k=0}^{B-1} q_{i,k} = 1 \tag{6}$$

6. One operation per time step

$$\forall i, \forall t,\ \sum_{p=0}^{6} o_{i,t,p} = 1 \tag{7}$$

7. Table 1 type constraints (described in text)

$$\forall i, \forall t,\ \sum_{p=0}^{6} f_{i,p} - \sum_{p=0}^{6} o_{i,t,p} \geq 0 \tag{8}$$

$$\begin{aligned} \forall i,\ va_i - f_{i,0} &\geq 0 \\ va_i - f_{i,2} &\geq 0 \\ va_i - f_{i,3} &\geq 0 \end{aligned} \tag{9}$$

$$\begin{aligned} \forall i,\ vb_i - f_{i,1} &\geq 0 \\ vb_i - f_{i,2} &\geq 0 \\ vb_i - f_{i,4} &\geq 0 \end{aligned} \tag{10}$$

$$\begin{aligned} \forall i,\ vq_i - f_{i,3} &\geq 0 \\ vq_i - f_{i,4} &\geq 0 \\ vq_i - f_{i,5} &\geq 0 \\ vq_i - f_{i,6} &\geq 0 \end{aligned} \tag{11}$$

$$\forall i, \sum_{p=0}^{6} f_{i,p} + 2va_i + 2vb_i + 2vq_i \leq 8 \tag{12}$$

Constraint (4) and (5) state that multiplexer must only select one input, (6) states that shifter must only shift by one k, (7) states that only one operation can be performed at any one time step. However, the total number of operations of each computational node must conform to the configuration shown in Table 1. This requires (8)–(12). Binary variables $f_{i,p}$ represent the functions that node i performs: $f_{i,p} = 1 \Leftrightarrow$ node i, *at some time* performs operation p. Binary variables va_i, vb_i and vq_i represent whether the node operation requires input $a_{i,t}$, $b_{i,t}$ or $x_{i,t-1}$ respectively. The variables va_i, vb_i and vq_i will be then represented in setting the condition of the number of operations of the node in (12), which ensures that each node is only of one of the possible Type (1, 2, or 3), shown in Table 1.

Clearly (1)–(3) are not yet in linear form. However, we have previously demonstrated how these constraints can be effectively linearized [1].

5. POWER OPTIMIZATION

In this section, a technique for power optimization is presented. A power model is proposed and used as a guideline for constructing constraints for ILP to optimize power consumption.

5.1. Power Dissipation in FPGAs and Macro-modelling

The total power in a CMOS circuit design can be divided into static and dynamic power. Static power dissipation depends on the physical properties of the devices, mainly occurring as a result of leakage currents and is therefore outside the designers' control for FPGA designs. Dynamic power dissipation is mainly due to charging and discharging of parasitic capacitance which can be defined by (13), where C_n is the load capacitance, f_n is the average toggle rate or switching activity of the net n, and V is the supply voltage [11]. As we can see, the average switching activity is a key parameter that can be kept under control by circuit design techniques. Therefore, our focus in this paper is on reducing the power dissipated of MRM design in FPGA due to this quantity. A power macro-model is proposed which can be employed as a predictor to approximate the power.

$$P_{avg} = \frac{1}{2} \sum_{n} C_n f_n V^2 \tag{13}$$

Occasionally a set of coefficients can be computed by several different MRM graph topologies. As an example, Figs. 4(a) and 4(b) depict two possible solutions of 2×1 problem with matrix $[12, 7]^T$.

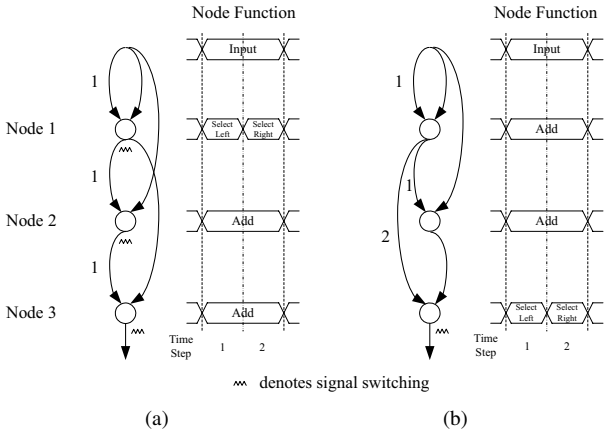

Fig. 4. A MRM structure for $\{12, 7\}$ having function changing on (a) node 1 and (b) node 3.

Although both Figs. 4(a) and (b) provide the same area (same number of nodes), the signal transitions or switching activity are different due to the distinct function changing of the nodes. In Fig. 4(a), changing the function at node 1 produces a signal switching at its output which then propagates to 2 and 3 to generate the further signal switchings. In 4(b), switching happens primarily at the output of node 3. Thus, it would be expected that structure in Fig. 4(b) is superior to Fig. 4(a) in terms of power consumption.

Our approach introduces a power macro-model that takes into account the effect of the number of the computational nodes (N) and the signal switchings (S). In Fig. 4(a), $N = 3$

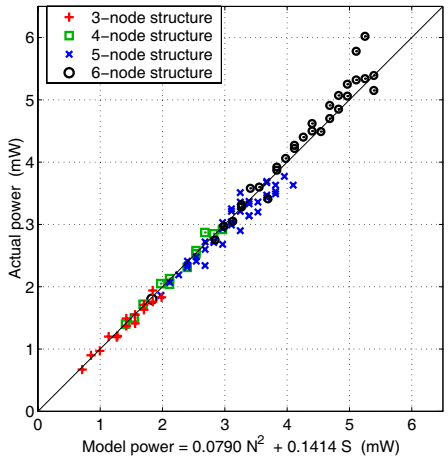

Fig. 5. Scatter plot between power from Xilinx XPower simulation and the model (15).

and $S = 3$, and in (b) $N = 3$ and $S = 1$. Thus, the power model is given by

$$\text{Power} = f(N, S) \tag{14}$$

In this work, we focus on the sum of the logic power and signal power. We propose (15) as a suitable functional form of the power model. The term N^2 is used to model the fact that with a large area, proportional to N, the capacitance on the routes also increases proportional to N. To investigate the factors k_1 and k_2 in (15), many random structures were generated and synthesized for Xilinx Virtex-II XC2V1000-4 device [10]. The route models were analyzed using value-change-dump files by XPower tool [10]. A least-squares fitting approach was then used.

$$\text{Power} = k_1.N^2 + k_2.S \tag{15}$$

Fig. 5 illustrates the scatter plot of the actual power against model power of (15) with $k_1 = 0.079$ and $k_2 = 0.1414$. It is shown that Fig. 5 fits with worst-case error 0.14% (average 0.04%).

5.2. ILP Modelling for Signal Switching Reduction

It is shown that a reduction in number of the nodes and the signal switchings can lead to a reduction in the power dissipation. In this subsection, the signal switching based power consumption will be incorporated into the ILP formulation previously presented in Section 4.

A binary variable $F_{i,w}$ represents whether there is a function switching from time step w to $w+1$ of i^{th} node obtained from (16) where $p \in \{0, 1, \ldots, 6\}$ and $w \in \{1, 2, \ldots, T-1\}$. The variable F in (17) represents the total number of the function switchings, and is used as an objective function in ILP.

However, in structures in Fig. 4(a) and (b), it is shown that although have the same value function switchings, they are different in terms of propagation caused by the signal switchings; therefore extra constraints are required to model this propagation.

1. Function switching
$$F_{i,w} \geq o_{i,w,p} - o_{i,w+1,p} \tag{16}$$
2. Total number of the function switchings
$$F = \sum_{i=1}^{N} \sum_{w=1}^{T-1} F_{i,w} \tag{17}$$
3. Constraints for function switchings
$$F_{i,w} \geq F_{h,w} + L_{i,h,w} - 1 \tag{18}$$
$$F_{i,w} \geq F_{h,w} + R_{i,h,w} - 1 \tag{19}$$
$$F_{i,w} \geq F_{h,w} + AL_{i,h,w} - 1 \tag{20}$$
$$F_{i,w} \geq F_{h,w} + AR_{i,h,w} - 1 \tag{21}$$
4. Node functions of left input connection
$$L_{i,h,w} \geq o_{i,w,0} + m_{i,h} - 1 \tag{22}$$
$$L_{i,h,w} \geq o_{i,w,3} + m_{i,h} - 1 \tag{23}$$
5. Node functions of right input connection
$$R_{i,h,w} \geq o_{i,w,1} + n_{i,h} - 1 \tag{24}$$
$$R_{i,h,w} \geq o_{i,w,4} + n_{i,h} - 1 \tag{25}$$
6. Node functions of both input connection
$$AL_{i,h,w} \geq o_{i,w,2} + m_{i,h} - 1 \tag{26}$$
$$AR_{i,h,w} \geq o_{i,w,2} + n_{i,h} - 1 \tag{27}$$

where all these constraints are valid for $\forall i > 1$.

The additional constraints (18)–(21) are based on the fact that when, at time step w, there is a solution from (22)–(27) that, at least, one input of node i connects to the output of node j, then

$$F_{i,w} \geq F_{j,w} \tag{28}$$

Referring to Fig. 3(b), in (22)–(23), the binary variables $L_{i,h,w}$ represent the i^{th} node functions (option 0 and 3 of (3)) that require a left input value which is obtained from a previous h^{th} node, where $h \in \{1, 2, \ldots, i-1\}$. In (24)–(25), the variables $R_{i,h,w}$ represent the i^{th} node functions (option 1 and 4 of (3)) that require a right input value which is obtained from the previous h^{th} node. The variables of $AL_{i,h,w}$ and $AR_{i,h,w}$ in (26)–(27) represent the i^{th} node function (option 2 of (3)) that requires both left and right inputs, and each is connected to output of the h^{th} node(s).

When the value of $L_{i,h,w}$, $R_{i,h,w}$, $AL_{i,h,w}$ or $AR_{i,h,w}$ is "1", it makes (18), (19), (20) and (21) respectively, to satisfy the required condition (28).

6. RESULTS AND CONCLUSION

The synthesis results of average area and power targeting Xilinx Virtex-II XC2V1000-4 device [10] are illustrated in Table 2 and Table 3 respectively. All approaches were tested using 10 sets of 4-bit coefficients generated randomly. All ILP models were solved using MOSEK optimization software [12].

Table 2. Average area (in LUTs) of non-register and register approach, and % area saving.

problem $T \times C$	non-register	register	% area saving
2x1	16.3	15.7	3.68
3x1	17.1	16.4	4.09
2x2	25.2	24.3	3.57
3x2	26.8	25.6	4.47
2x3	38.3	30.5	20.36
3x3	44.7	34.7	22.37

Table 3. % results representing power reduction, and average % power reduction for these results.

problem $T \times C$	% results representing power reduction	average % power reduction
2x1	70	39.62
2x2	70	38.12
2x3	80	48.30
3x1	80	40.74
3x2	100	32.07
3x3	80	21.55
Average	80	36.73

Table 2 shows the average area comparison of the proposed approach using all types of configuration in Table 1, to the optimal non-register approach [1] by using Type 1 configuration. Area savings of up to 22% (at 3×3 problem) have been achieved over the non-register approach. This is because exploiting the register allows sub-expression generated from the previous cycles to be re-used in later cycles, providing more resource sharing, and leading to less hardware requirements.

When combined with the power optimization technique, there is a significant impact on power consumption (Table 3). 80% of all results show a reduction in power, of which the energy saved ranges from 21% to 48%.

To our knowledge, there is no existing algorithm dealing with time-multiplexed multiplication that exploits dedicated registers for optimized FPGA-based implementation. This paper has presented such a design and optimization technique for the MRM problem. A power macro-modelling approach is used to estimate power, which is a function of the number of computational nodes and signal switchings. It has shown a significant effect on power reduction due to reduced signal activity level. Since one of the main sources of logic activity in DSP systems is from multiplier circuits, this is likely to lead to a significant reduction in the overall signal activity level. In the future we intend to investigate the impact of other parameters for power estimation, such as clock distribution.

7. REFERENCES

[1] N. Sidahao, G. A. Constantinides, and P. Y. Cheung, "Multiple restricted multiplication," in *Proc. 14th Int. Conf. Field Programmable Logic and Application (FPL'04)*, Aug./Sept. 2004, pp. 374–383.

[2] S. S. Demirsoy, A. G. Dempster, and I. Kale, "Design guidelines for reconfigurable multiplier blocks," in *Proc. IEEE Int. Symp. on Circ. and Syst. (ISCAS'03)*, vol. 4, May 2003, pp. IV–293–IV–296.

[3] P. Tummeltshammer, J. C. Hoe, and M. Puschel, "Multiple constant multiplication by time-multiplexed mapping of addition chains," in *Proc. 41st Conf. Design Automation (DAC'04)*, June 2004, pp. 826–829.

[4] R. H. Turner and R. F. Woods, "Highly efficient, limited range multipliers for LUT-based FPGA architectures," *IEEE Trans. VLSI Syst.*, vol. 12, no. 10, pp. 1113–1118, Oct. 2004.

[5] R. H. Turner, R. Woods, and T. Courtney, "Multiplier-less realization of a poly-phase filter using LUT-based FPGAs." in *Proc. 12th Int. Conf. Field Programmable Logic and Application (FPL'02)*, Sept. 2002, pp. 192–201.

[6] J. Villalba, G. Bandera, M. A. Gonzalez, J. Hormigo, and E. L. Zapata, "Polynomial evaluation on multimedia processors," in *Proc. IEEE Int. Conf. Application-Specific Systems, Architectures, and Processors (ASAP'02)*, July 2002, pp. 265–274.

[7] 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*, Sept. 1991, pp. 13–24.

[8] 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. Comput. Aided Des. Integrated Circ. Syst.*, vol. 15, no. 2, pp. 151–165, Feb. 1996.

[9] R. S. Garfinkel and G. L. Nemhauser, *Integer Programming*. Wiley & Sons, 1972.

[10] Xilinx, Inc., "Xilinx Documentation and Literature," http://www.xilinx.com/support/library.htm (accessed 24 May 2005).

[11] J. H. Anderson and F. N. Najm, "Switching activity analysis and pre-layout activity prediction for FPGAs," in *Proc. ACM/IEEE Int. Workshop on System-Level Interconnect Prediction*, Apr. 2003, pp. 15–21.

[12] MOSEK ApS, "MOSEK ApS optimization software," http://www.mosek.com (accessed 24 May 2005).