ICCAD 2005 - International Conference on Computing Aided Design, San Jose, California, Nov. 6-10, 2005

An Exact Algorithm for the Maximal Sharing of Partial Terms in Multiple Constant Multiplications

Paulo Flores
IST/INESC-ID, Lisboa, Portugal
pff@inesc-id.pt

José Monteiro
IST/INESC-ID, Lisboa, Portugal
jcm@inesc-id.pt

Eduardo Costa
UCPel, Pelotas, Brazil
ecosta@ucpel.tche.br

Abstract— **In this paper we propose an exact algorithm that maximizes the sharing of partial terms in Multiple Constant Multiplication (MCM) operations. We model this problem as a Boolean network that covers all possible partial terms which may be used to generate the set of coefficients in the MCM instance. The PIs to this network are shifted versions of the MCM input. An AND gate represents an adder or a subtracter, *i.e.*, an AND gate generates a new partial term. All partial terms that have the same numerical value are ORed together. There is a single output which is an AND over all the coefficients in the MCM. We cast this problem into a 0-1 Integer Linear Programming (ILP) problem by requiring that the output is asserted while minimizing the total number of AND gates that evaluate to one. A SAT-based solver is used to obtain the exact solution. We argue that for many real problems the size of the problem is within the capabilities of current SAT solvers. We present results using binary, CSD and MSD representations. Two main conclusions can be drawn from the results. One is that, in many cases, existing heuristics perform well, computing the best solution, or one close to it. The other is that the flexibility of the MSD representation does not have a significant impact in the solution obtained.**

I. Introduction

Several computationally intensive operations, such as, Finite Impulse Response (FIR) filters and Fast Fourier Transforms (FFT), involve a sequence of Multiply-Accumulate (MAC) operations with constant coefficients. These operations are typical in Digital Signal Processing (DSP) applications. Hardwired dedicated architectures are the best option for maximum performance and minimum power consumption.

Constant coefficients allow for a great simplification of the multipliers, which can be reduced to shift-adders [1]. In these multipliers, a bit set to 1 in position m of the coefficient implies that the input shifted left by m positions is to be added to the partial sum. Shifts are free in terms of hardware, hence the hardware required for a multiplication with a constant with n bits set to 1 is simply $n-1$ adders.

In many MAC operations, the same input is to be multiplied by a set of coefficients, an operation known as Multiple Constant Multiplications (MCM). An example of this is the transposed form architecture of a FIR filter. In this situation, significant reductions in hardware, and consequently power, can be obtained by sharing the partial products of the input. We propose an algorithm that optimally solves this maximal sharing problem. Although this problem has been proven to be NP-complete [2], we show that for many practical instances we can find an optimum solution.

This problem has been the subject of extensive research in recent years. Two key strategies have had a large impact in the optimization of MCMs. One is to consider not only adders, but also subtracters to combine partial terms, thus increasing the opportunity for the sharing of common subexpressions. The second is the usage of the Canonical Sign Digit (CSD) representation for the coefficients. This representation minimizes the number of non-zero digits, hence the maximal subexpression sharing search starts from a minimal level of complexity.

In a recent paper, Park et al. [3] propose the usage of a Minimal Signed Digit (MSD) representation for the coefficients. The MSD representation is obtained from the CSD representation by relaxing the requirement that there cannot be two consecutive non-zero digits. Under the MSD representation, a given numerical value can have multiple representations. However, in all of them, the number of non-zero digits is the same as the CSD representation. The algorithm proposed in [3] exploits the redundancy of the MSD representation by choosing the MSD instance that leads to a maximal sharing in the implementation efficient FIR filters.

To the best of our knowledge, all previous solutions to this problem have been heuristic, providing no indication as to how far from the optimum their solution is. We propose an exact algorithm that is feasible for many real situations. We model this problem as a Boolean network that covers all possible partial terms which may be used to generate the set of coefficients in the MCM instance. The inputs to this network are shifted versions of the value that serves as input to the MCM operation. Each adder and subtracter used to generate a given partial term is represented as an AND gate. All partial terms that represent the same numerical value are ORed together. There is a single output which is an AND over all the coefficients in the MCM. We cast this problem into a 0-1 Integer Linear Programming (ILP) problem by requiring: that the output is asserted, meaning that all coefficients are covered by the set of partial terms found; while minimizing the total number of AND gates that evaluate to one, *i.e.*, the number of adders/subtracters. A SAT-based solver is used to obtain the exact solution.

We have applied this algorithm to coefficients represented in binary, CSD and MSD representations. Note that the redundancy of the MSD representation can be readily incorporated in our model, where the equivalent MSD representations are simply new inputs to the OR gate that generates a given coefficient.

Two main conclusions can be drawn from the results. One is that, in many cases, existing heuristics perform well, computing the best solution, or one close to it. The other is that the flexibility of the MSD representation does not have a significant impact in the solution obtained.

II. Related Work

A large amount of work has addressed the use of efficient implementations of multiplier-less MCMs. The techniques include the use of different number representation schemes, the use of different architectures and implementation styles, and coefficient optimization techniques, *e.g.*, [4].

Synthesis algorithms have been proposed that are based on the Canonical Signed Digit (CSD) representation [5]. CSD is a signed digit system with the digit set $\{1, 0, \bar{1}\}$, where $\bar{1}$ denotes -1. The CSD representation is unique and uses two main properties: (1) the number of non-zero digits is minimal, (2) two non-zero digits are not adjacent. Hardware requirements are reduced because the numerical values are represented with a maximal number of zero digits.

In [3], the Minimum Signed Digit (MSD) representation is proposed for the coefficients. The MSD representation is obtained by removing the second property of the CSD representation. Thus, a constant can have several MSD representations, but all with a maximum number of zero bits. For example, the value 6 is represented using 4 bits in CSD as $10\bar{1}0$, but both $10\bar{1}0$ and 0110 are valid representations in MSD. In the algorithm described in [3], *Cset* represents the coefficient set to be synthesized and contains all MSD representations for all coefficients. The first representation the matches a combination of subexpressions is used.

All these methods use heuristic algorithms to minimize the total number of adders/subtracters. In [6] an exact algorithm is presented that finds the best representation, but for a single coefficient. The solution we propose is based on solving a 0-1 Integer Linear Programming (ILP) over a Boolean network, asserting the output while minimizing the number of ones in a set of nodes. Generic SAT-solvers [7] can be adapted to iteratively solve this optimization problem. However, recent solvers, targeted specifically for Pseudo Boolean Optimization (PBO) problems, have been proved to be significantly more efficient. In this work, we are using an efficient SAT-based solver which incorporates several advanced optimization techniques and has been applied to several classes of problems [8].

III. Proposed Algorithm

In this section we describe the proposed algorithm for the maximal sharing of partial terms. First, we present our optimization model for the binary representation, and its generalization for CSD and MSD representations. Then, we describe the algorithm implemented to generate the set of constraints and optimization function to be solved by a SAT-based 0-1 ILP solver.

A. Model

As mentioned before, we model the maximal sharing of partial terms by a Boolean network with only AND and OR gates. Each AND gate represents an adder or subtracter which produces some partial term value. Each OR combines all partial terms that yield the same value. Any signal in this network represents one value in a selected representation: binary, CSD or MSD.

When considering only the binary representation of the coefficients, the Boolean network consists of all possible combinations of partial terms (sums) that can be used to obtain the multiplication of a value with a given set of constant coefficients. The Boolean network that has all possible partial terms has the following characteristics:

- the primary inputs (PIs) of the network are the input value (the value we are applying the MCM operation on) or shifted versions of the input value.
- there is an AND gate to represent a simple adder for each partial term that could be used to generate the coefficients. Hence, each AND gate would have two inputs. However, we add a third input that is left as a free variable. An AND gate evaluating to a 1 (meaning that the free variable is set to 1) indicates that a particular partial term is present in the solution of the MCM problem. Since shifts are free, equivalent classes can be created from shifted versions of partial terms, thus reducing the total number of AND gates.
- there is an OR gate to assemble all the different combinations of partial terms that yield a given value.
- a single output is generated by an AND gate that combines all the coefficients in the MCM problem (outputs of the associated OR gates), hence the output of this AND gate will only evaluate to 1 when all the coefficients are covered.

Given this model the optimization SAT-based solver has to search for an input combination that sets the output to a 1 while minimizing the cost function defined as the number of AND gates which evaluate to a 1.

As an illustrative example, consider a single 4-bit coefficient, 15 (in binary, 1111). The value can be obtained as 8+7 ($1\boxed{111}$), 11+4 ($\overset{\frown}{\boxed{1}1\boxed{11}}$), 13+2 ($\overset{\frown}{\boxed{11}1\boxed{1}}$) or 14+1 ($\boxed{111}1$), by adding a single bit to a partial sum, or as 9+6 ($\overset{\frown}{\boxed{1}11\boxed{1}}$), 12+3 ($11\boxed{11}$) or 10+5 ($\overset{\frown}{\boxed{1}1\boxed{1}}1$), by adding two partial sums. In turn, 7+8, for instance, requires that 7 be obtained either as 6+1 ($0\boxed{11}1$), 5+2 ($0\overset{\frown}{\boxed{1}1\boxed{1}}$) or 4+3 ($01\boxed{11}$). The same observation applies to all the other partial sums. The complete Boolean network for this example is presented in Figure 1.

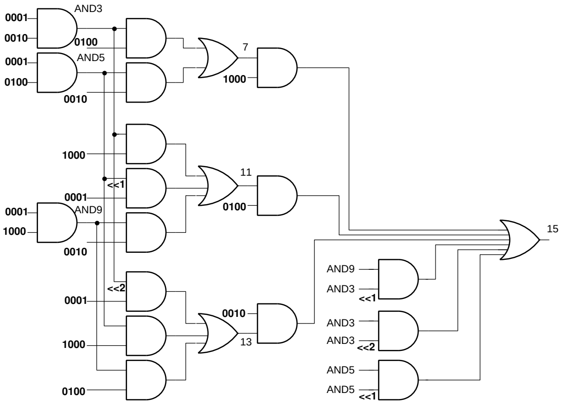

Fig. 1. Boolean network representing the coverage of coefficient 15.

In general, a coefficient with value v can be obtained from $\lceil \frac{v}{2} \rceil$ partial sums. However, we can create equivalent classes from cases that differ only on a shift, thus reducing significantly the total number of cases. From the example above, 14+1 and 7+8 are equivalent because 14 and 7 are partial sums that differ only on a shift and the same for 1 and 8. Similarly for 6+1 and 3+4.

The model using CSD or MSD representations generates a similar Boolean network. However, the values considered to generated partial terms for a given value are only the correspondent CSD or MSD representations. In these models an AND can represent either an adder or a subtracter. This is a result of the signed digit system where some partial terms (covers of a value) are implemented as subtractions.

Consider, for example, a single 3-bit wide coefficient with the value 3. The CSD representation of the coefficient is $10\overline{1}$ ($\overline{1}$ stands for -1). Therefore this value can be obtained with single subtracter as 4-1 ($10\overline{1}$). For the MSD model, the value 3 can be represented both by 011 and $10\overline{1}$, which can be obtained with an adder as 2+1 or with a subtracter as 4-1.

B. Implementation

The implemented algorithm to generated the above optimization model can be used for any type of coefficient representation: binary, CSD or MSD. However, using the MSD representation results in a more elaborated algorithm, because several representation may exist for the same value. We will describe first the MSD implementation of the algorithm and then we summarize the changes for binary or CSD representations.

In a preprocessing phase, all coefficients are converted to positive and then made odd by successive divisions by 2, *i.e.*, we shift all coefficients to the right so that zero bits on the right are eliminated. Each new resulting coefficient is added to the set of coefficients to synthesize, the *Iset*. This set represents the minimum number of coefficients necessary to synthesize for the MCM implementation.

For each element i in the *Iset* all MSD representations are determined using $\lceil \log_2(i) \rceil + 1$ bits and inserted in the *Cset*. Therefore, *Cset* begins with all the MSD coefficients representations as in [3]. However, during our algorithm execution *Cset* will be augmented with MSD representations of partial terms.

Then we enter in the main algorithm loop where an element c, removed from *Cset* and representing a number i, is processed to determine its covers: 1) compute all non-symmetric partial term pairs that covers the element c; 2) converted to positive and made odd each element of the cover pair; 3) add each cover pair to the corresponding set of covers of the element being processed, $Aset_i$; 4) add the MSD representations of each cover to the *Cset* if the representation has not been processed yet and it is not in the set. Covers with only one non-zero digit are skipped.

This loop is repeat until there are no more elements in *Cset*. The pair of elements in each $Aset_i$ represents all possible alternatives of partial terms for a value i based on its MSD representations.

The mapping of the Boolean network into a 0-1 ILP optimization model is obtained by representing each gate in a Conjunctive Normal

Form (CNF clause) [7]. Each clause is then converted into a 0-1 ILP constraint.

The final 0-1 ILP optimization model is generated in three steps: 1) for each pair element in *Aset*$_i$ generate the corresponding AND gate. Generate an OR gate for the value i with the outputs of all the ANDs resulting from *Aset*$_i$; 2) identify all the OR outputs that represent a coefficient (values belonging to *Iset*) and force their outputs to be 1. This is equivalent to have one AND gate that combines all the coefficients, but reduces the size of the model; 3) generate the function to be minimized. This function is a linear combination of all the AND gates outputs. However, we note that minimizing the number of AND gates set to 1 is equivalent to minimizing the number of OR gates set to 1. In order to minimize the number of optimization variables we add an extra 2-input AND gate at the output of each OR. The other input of this gate has a free optimization variable that is set to 1 by the solver if this partial term is needed to generate any coefficient. Therefore, the final optimization function is a sum of the outputs of these extra AND gates.

Note that this algorithm can be easily adapted to obtain the 0-1 ILP optimization model using different coefficient representations. When the MSD representation of a coefficient or partial term is determined, one needs only to compute a binary or CSD representation instead. Moreover, mixed representations (*i.e.*, binary and CSD, or binary and MSD) can also be computed and added to *Cset*.

C. Cost Analysis

Given a coefficient with n bits all set to 1, than the Boolean network will generate all partial terms with $\leq n$ bits set to 1. Thus, any other coefficient in the problem is simply obtained by adding its value, already an output of some OR gate in the network, to the final AND whose output we require to be one. Hence, for n-bit coefficients, the complexity of the problem is bounded above by the case of a single coefficient will all the n bits set to 1.

For a given value with n bits set to 1, the total number of gates in the Boolean network is given by:

$$\begin{aligned} G_{\text{or}}(n) &= \sum_{i=3}^{n} \frac{1}{(i-1)!} \prod_{k=1}^{i-1} (n-k) \\ G_{\text{and}}(n) &= n-1+\sum_{i=3}^{n} \frac{2^{i-1}-1}{(i-1)!} \prod_{k=1}^{i-1} (n-k) \end{aligned} \tag{1}$$

As we cast this into 0-1 ILP problem to be handled by a SAT-solver, the relevant complexity parameters are: number of variables, number of clauses and number of optimization variables. As we discussed in the previous section, the number of optimization variables is simply the number of OR gates, $\#_{opt\_vars}(n) = G_{\text{or}}(n)$. The total number of variables is given by the total number of gates in the circuit, plus the primary inputs, *i.e.*, $\#_{vars}(n) = n + G_{\text{or}}(n) + G_{\text{and}}(n)$. Finally, the number of clauses can be computed by noting that, for each logic gate, the number of clauses is given by the number of gate inputs plus one. All AND gates in our network have two inputs, hence each contributes with 3 clauses. Although the number of inputs to the OR gates varies, we note that for a given level the total number of inputs to all the OR gates at that level is the number of AND gates. The total number of clauses is thus:

$$\begin{aligned} \#_{clauses} &= \overbrace{3\,G_{\text{and}}(n)}^{\text{due to ANDs}} + \overbrace{(G_{\text{and}}(n) + G_{\text{or}}(n))}^{\text{due to ORs}} \\ &= 4\,G_{\text{and}}(n) + G_{\text{or}}(n) \end{aligned}$$

These values correspond to the original Boolean network. However, due to the optimization made in the previous section that allows for the minimization of OR gates evaluating to 1 instead of AND gates, an extra 2-input AND gate is placed at the output of each OR gate. This means that we have $3G_{\text{or}}(n)$ extra clauses and $2G_{\text{or}}(n)$ extra variables.

Table I gives the size of the Boolean network in terms of the number of AND and OR gates, and the size of the SAT problem in terms of the number of clauses, number of variables and number of optimization variables for a single coefficient with different values of n bits, all set to 1. Although we can observe the exponential growth in complexity, there are two points we should stress. One is that this is for the case of a coefficient with all bits set to 1, which the worst case in terms of the number of partial terms. In many cases, the all-1s coefficient does not appear, as observed is Section IV. Second, the size of the SAT problem for $n = 12$ is already within reach of current SAT-solvers and, thus, can be solved exactly. In practice, many problems do not require coefficients larger than 12 bits.

A similar analysis can be made for the CSD and MSD representations. We note that, since the complexity is related to the total number of bits set to one in any given coefficient and at least half of the bits in both the CSD and MSD representations are zero, the complexity of these representations is significantly lower than that of the binary.

IV. Results

We present results obtained with the exact algorithm applied to the optimization of FIR filters and compare them with the heuristic approach of [3]. We first present results for some filter instances, where the coefficients were computed with MATLAB using the Remez algorithm. The filters' specifications are presented in Table II, together with the respective problem size: column *filter* is just an index for each example; *pass* and *stop* are normalized frequencies that define the passband and stopband, respectively; *#tap* is the number of coefficients; *width* is the bit-width of the coefficients; *#coef* indicates the total number of different coefficients; *#nzbit* gives the maximum number of non-zero bits over all the coefficients. The next three sets of three columns indicate the size of the 0-1 ILP problem for the binary, CSD and MSD representations, in terms of the number of variables, clauses and optimization variables. We can observe that the problem size varies widely, even for filters of similar specifications. Moreover, the correlation between the size of the problem and the number of coefficients and maximum number of non-zero bits is not so direct. The reason for this is that the way the Boolean network is constructed depends heavily on the relation between the coefficients in terms of the sub-expressions that they have in common. In any case, we can observe that the binary representation is the most complex because the coefficients have a larger number of non-zero digits. Although the MSD representation requires the same number of zero digits per coefficient as CSD, we need to represent a larger number of coefficient patterns due to the redundancy in MSD. Note that the complexity of the 0-1 ILP problem to be passed to the SAT solver (we used [8]) is much lower than the worst-case scenario derived in the previous section.

Table III presents the results obtained for the selected benchmarks using the three representations under consideration: binary, CSD and MSD. Column *adders* gives the minimum number of adders/subtracters required to implement the filter, column *steps* gives the maximum depth in terms of adder-steps for all coefficients and *CPU* is the CPU time in seconds used to compute this exact solution on a PC with dual Pentium Xeon at 2.4GHz, with 4GB of main memory, running Linux. We note that most of the problem instances

TABLE I

Size of the Boolean network and SAT problem.

n	OR	AND	$\#clauses$	$\#vars$	$\#opt\_vars$
8	120	2,059	8,716	2,427	120
10	502	19,171	78,692	20,687	502
12	2,036	175,099	708,540	181,219	2,036
14	8,178	1,586,131	6,377,236	1,610,679	8,178
16	32,752	14,316,139	57,395,564	14,414,411	32,752

TABLE II

CHARACTERISTICS OF THE FIR FILTERS AND PROBLEM SIZE.

Filter	Filter Specification				Filter Params		Binary			CSD			MSD		
	pass	stop	#tap	width	#coef	#nzbit	vars	clauses	optv	vars	clauses	optv	vars	clauses	optv
1	0.20	0.25	120	8	10	5	284	576	28	195	352	23	474	924	36
2	0.10	0.25	100	10	10	5	1449	3470	100	513	982	54	976	2014	71
3	0.15	0.25	40	12	14	8	4955	14116	220	651	1288	66	1860	4112	112
4	0.20	0.25	80	12	28	10	1639	4008	109	1022	2088	97	2722	6220	143
5	0.24	0.25	120	12	34	8	4846	13540	228	866	1756	83	1729	3732	116
6	0.15	0.25	60	14	20	8	8017	23460	322	1351	2880	120	5488	13788	222
7	0.15	0.20	60	14	29	9	15039	46034	510	1460	3046	133	2733	6026	175
8	0.10	0.15	60	14	28	9	16851	51616	566	1796	3840	157	4595	10752	243

TABLE III

SUMMARY OF RESULTS FOR THE DIFFERENT COEFFICIENT REPRESENTATIONS.

Filter	Exact Binary			Exact CSD			Exact MSD				Heuristic MSD [3]	
	adders	steps	CPU	adders	steps	CPU	adders	steps	CPU		adders	steps
1	10	3	0.05	10	3	0.04	10	3	0.17		10	3
2	18	4	2.24	17	3	0.11	17	2	0.96		18	4
3	16	4	85.42	16	3	0.20	16	3	4.85		18	4
4	29	4	8.59	29	3	0.44	29	3	12.73		29	4
5	35	4	169.93	34	3	0.38	34	3	2.25		34	3
6	*25*	*4*	*3607.80*	23	3	0.95	22	3	191.52		22	4
7	–	–	–	35	3	19.32	34	3	22.00		35	3
8	*36*	*4*	*3600.50*	35	3	10.99	*35*	*3*	*3602.00*		37	4

are solved in a very small period of CPU time. There are four cases for which our algorithm did not finish within an hour of CPU time. For three of these, we obtained a non-optimal solution, represented in italic in this table. Only for filter 7 under the binary representation we were not able to obtain any solution.

We can observe that the solutions obtained with MSD do improve on the results of CSD, but only in a few cases. Namely, for filters 6 and 7, the number of required adders or subtracters is reduced by one, and for filter 2 the depth is reduced by one level. The other observation is that the solution obtained for the binary representation has similar number of adders/subtracters for small examples, although the depth may increase. CSD and MSD perform slightly better for larger examples.

The last two columns of Table III give the results obtained with the heuristic algorithm of [3] that uses the MSD representation. We can observe that in almost half of the filters the heuristic algorithm obtains the optimum number of adder/subtracters. However, there are cases where it is two units off, namely filters 3 and 8, and one unit off for filters 2 and 7. The difference is more noticeable in the number of adder-steps where our exact solution almost always has less levels, with a difference of two, which may represent 50% decrease in latency.

Figure 2 gives a plot of the average number of adders/subtracters obtained with our exact method and the heuristic of [3] versus the number of coefficients. We used coefficients with 10 bits and, for each number of coefficients, we run 30 instances with randomly generated coefficients. We observe that the average solution obtained with heuristic method only has at most 3 adders more than the exact and this distance remains almost constant with the number of coefficients. Hence, the relative quality of the heuristic solution increases with the number of coefficients.

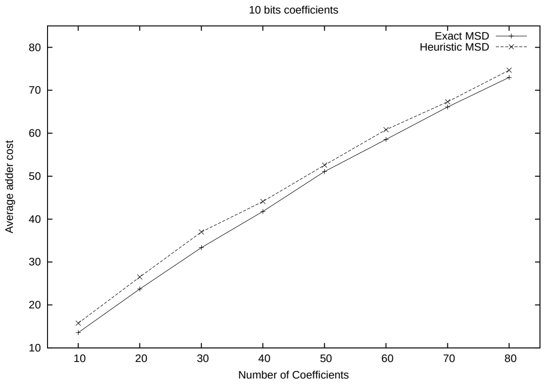

Fig. 2. Comparison of exact and heuristic results for random instances.

V. CONCLUSIONS

We have described a new algorithm that computes the exact minimum number of adder/subtracter modules in the implementation of MCM structures by maximizing the sharing of common subexpressions. The algorithm can handle binary, CSD and MSD representations for the coefficients. We presented results for digital filter synthesis where we demonstrate that the exact algorithm can be applied to real size examples. We compare with previously proposed heuristic algorithms and showed that, though these algorithms cannot guarantee an optimum solution, they perform reasonably well.

As future developments of this work, we are currently working on two different avenues of research. One is to find tecnhiques that enable the simplification of the Boolean network that we feed to the SAT-solver. The other is to develop and explore more general representations for the coefficients.

ACKNOWLEDGMENTS

This research was supported in part by the portuguese FCT under program POCTI.

REFERENCES

[1] H. Nguyen and A. Chatterjee. Number-Splitting with Shift-and-Add Decomposition for Power and Hardware Optimization in Linear DSP Synthesis. *IEEE Trans. on VLSI*, 8(4):419–424, August 2000.

[2] P. Cappello and K. Steiglitz. Some Complexity Issues in Digital Signal Processing. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 32(5):1037–1041, October 1984.

[3] I-C. Park and H-J. Kang. Digital Filter Synthesis Based on Minimal Signed Digit Representation. In *DAC*, pages 468–473, 2001.

[4] A. Nannarelli, M. Re, and G. Cardarilli. Tradeoffs between Residue Number System and Traditional FIR Filters. In *ISCAS*, May 2001.

[5] R. Pasko, P. Schaumont, V. Derudder, S. Vernalde, and D. Durackova. A New Algorithm for Elimination of Common Subexpressions. *TCAD*, 18:58–68, January 1999.

[6] A. Dempster and M. Macleod. Using All Signed-Digit Representations To Design Single Integer Multipliers Using Subexpression Elimination. In *ISCAS*, pages 24–26, May 2004.

[7] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an Efficient SAT Solver. In *DAC*, 2001.

[8] V. Manquinho and J. Marques-Silva. Effective Lower Bounding Techniques for Pseudo-Boolean Optimization. In *DATE*, March 2005.