Watermarking Multiple Constant Multiplications Solutions

Jennifer L. Wong*, Ji-Qing Ya, Miodrag Potkonjak*
*University of California, Los Angeles, CA 90095

Abstract— **Multiplications and multipliers dominate area, power, and speed requirements of modern designs. A standard technique to reduce these requirements is to use combinations of shifts and additions to execute multiplications. The approach is particularly effective when multiple constant multiplications are necessary. We introduce a simple, yet effective multiple constant multiplication (MCM) algorithm based on the iterative probabilistic greedy principle.**

Reusability and time to market constraints and therefore rapid implementations of integrated circuits placed design intellectual property at risk for piracy. We introduce a new intellectual property protection (IPP) technique for the protection of IP during the process of implementing constant multiplications using shifts and additions. Specifically, we introduce eight hard and soft watermarking techniques applied during the pre-processing phase of the MCM process. We demonstrate the technique on a number of constant multiplication benchmarks. The technique consistently achieves high statistical proof of authorship while introducing acceptable overhead in terms of required shifters and adders.

I. Introduction

Multiplications are pervasive in many types of computations and they often dominate the implementation in terms of required implementation area, timing requirements, and power budget. A great majority of implementation in many classes of applications are ones where a variable is multiplied by a constant. The main reason is that many applications rely heavily on linear models. The standard approach for implementation of constant multiplication is to use a combination of shifts and additions. In many situations even straight-forward realizations of constant multiplication using shift and adds results in significant reduction in area, time and power resources. Numerous techniques for mapping of a multiplication with a constant to a series of shift and additions were proposed [9], [8], [7], [11]. Another common situation is that it is often required to multiply a single variable with a number of different constants. For example, many linear transform such as FFT and DCT are computations that are consisting solely of multiple constant multiplications. Therefore, in the last ten years efficient implementation of multiple constant multiplications (MCM) have received significant amounts of attention [1], [3], [2]. Special techniques for target fast transforms and fast convolutions [1], [3], FIR and IIR filters [2], and matrix multipliers [5] have been proposed.

At the same time, recently a multi-billion dollar market for integrated circuit intellectual property has emerged. The viability of this market has a mandatory requisite of intellectual property protection (IPP) techniques. A number of different techniques have been proposed ranging from simple design tagging to hardware metering [10]. Probably the most attractive and certainly the most studied among them are watermarking-based techniques for IPP. The main idea is to add a number of constraints to the design at any level of the specification in such a way that the functionality and timing requirements of the design are not altered, its implementation cost is minimally impacted, and that it simultaneously one can establish the authorship of the design with very high probability. The proof is established by demonstrating a connection between unique knowledge available to the author of design (signature) and the structure of the implementation. For example, watermarking techniques at the system level, behavioral level, logic synthesis, physical design level have been demonstrated [6], [4]. An important observation is that embedding the authorship signature at a higher level of abstraction protects the design not only at that level but also, and more importantly, at all lower levels of abstraction.

In this paper we have two goals. The first is to propose a new approach for efficient realization of MCM. The technique provides a convenient trade-off between runtime and effectiveness. Our second and main goal is to propose the first technique for watermarking the realization of MCMs. Specifically, we do not propose only a single technique but several classes of techniques that are mutually orthogonal and can bs easily combined.

II. Multiple Constant Multiplications

In this section we first introduce the formulation of multiple constant multiplication (MCM) problem. We present both the standard formulation and one where the goal is to minimize the number of additions for a given number of shifts. Next, we provide a small example in order to not only clarify the formulation, but to also illustrate the main trade-offs and establish insights required for the development of the algorithm for the restricted version of MCM. We conclude the section by presenting a new MCM algorithm and by analyzing its advantages and limitations.

The input to MCM problem is a sequence of numbers that indicate constants with which a particular variable should be multiplied. Before the multiplication is accomplished each constant is translated into its binary representation. Constants could be either integers or decimal numbers. Note that decimal numbers can be translated into corresponding integers using the normalization process where each number is shifted to the left by a specific number of binary positions. Numerous

0-7803-8622-1/04/$20.00 ©2004 IEEE

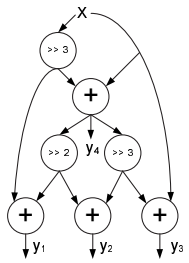

Fig. 1. Illustration of multiple constant multiplications.

	X_7 128	X_6 64	X_5 32	X_4 16	X_3 8	X_2 4	X_1 2	X_0 1
a (179)	1	0	1	1	0	0	1	1
b (74)	0	1	0	0	1	0	1	0
c (58)	0	0	1	1	1	0	1	0
d (48)	0	0	1	1	0	0	0	0
e (178)	1	0	1	1	0	0	1	0
f (182)	1	0	1	1	0	1	1	0
g (8)	0	0	0	0	1	0	0	0

$$
\begin{array}{lrcl}
\text{A} = & X_4 + X_5 & = & X_5 + X_4 \\
\text{B} = & X_1 + X_3 & = & X_3 + X_1 \\
\text{C} = & X_1 + X_7 & = & X_7 + X_1 \\
\text{D} = & B + X_6 & = & X_6 + X_3 + X_1 \\
\text{E} = & A + C & = & X_7 + X_5 + X_4 + X_1 \\
\text{F} = & E + X_0 & = & X_7 + X_5 + X_4 + X_1 + X_0 \\
\text{G} = & E + X_2 & = & X_7 + X_5 + X_4 + X_2 + X_1 \\
\text{H} = & A + B & = & X_5 + X_4 + X_3 + X_1
\end{array}
$$

Fig. 2. Illustration of restricted multiple constant multiplication approach.

approaches for efficient implementation of MCM have been and can be envisioned. For example, one can interleave shifting with addition in order to achieve a very small number of shifts and additions.

For example, suppose that we have to produce the following products: $y_4 = 9 * x$, $y_3 = 73 * x$, $y_2 = 108 * x$, $y_1 = 44 * x$. The straightforward approach that does not share common subexpressions requires ten additions and eight shifts. However, as shown in Figure 1, actually four additions and three shifts are sufficient for the realization these four constant multiplications. We restrict our attention to a specific instance of the MCM approach where we first establish all shifted versions of the variable that is used in multiplications. After that our goal is to minimize the number of used additions by exploring common subexpressions required for the realization of different multiplications. Obviously, the unrestricted version of MCM can result in a smaller number of both additions and/or shifts. However, the restricted version of the problem has several attractive features. First, it is very amenable for optimization intensive treatment due to its conceptual simplicity. Also if the number of different multiplications with the same variable is high, it is easy to see that the restricted version is actually the optimal one.

We will use the following example to illustrate the potential of the restricted version of the MCM problem. Lets assume that our goal is to obtain seven different products of a variable with the following constants 179, 74, 54, 48, 178, 182, and 8. This instance of the MCM problem is shown in Figure 2, after each constant is translated into its binary representation. Our first step is to shift the variable in the products by 1, 2,

```
1. while(!termination criteria) {
2.   Merge necessary and eliminate completed;
3.   while(all constants not completed) {
4.     Calculate benefit and damage for each product pair;
5.     Probabilistically select pair to merge;
6.     Create new bit holder;
7.     Update binary representation of all constants;
8.     Merge necessary and eliminate completed; }}
```

Fig. 3. Probabilistic greedy approach to the MCM problem.

3, 4, 5, 6, and 7 positions. For example, this can be efficiently accomplished on a shifter that shifts only by one position by iteratively repeating the process seven times. Therefore the cost and time expense for shifting is rather low. Before we begin the MCM process, two observation can be made.

The first is that for variable g we do not need any additions and therefore can be excluded from further consideration. Secondly, for all constants that have two one's in their representation (constant d) we must directly create the shifted versions of the variable. Therefore any algorithm for the restricted version of MCM should start by taking advantage of these two observations. At the bottom of Figure 2 we show a system of eight additions that are sufficient to form all the required products. If we count the number of additions required for independent creation of all required multiplications by counting the number of ones in the representation of each constant reduced by one, we see that the total number of required number of additions in the non-optimized approach is 17. Therefore, the system of additions at the bottom of Figure 2 reduces the addition requirement by a factor more than 2.

We now show an approach that in a heuristic and probabilistic way aims to automatically produce the sequence of additions that maximally reduces the required number of additions for a given instance of the MCM problem. The approach is summarized using the pseudo-code in Figure 3.

Lines 2 and 8 refer to two steps that are in all situations are necessary and optimal. The *merge necessary* step is one where a particular number with which a variable is multiplied has only two ones in its binary representation. It is necessary because we must create that number and the only way to create it is to add this sifted version of the variable. Once a multiplicand contains a single one in its binary representation the corresponding product is created and therefore there is no need for any additional attention to the product. Line 4 introduces the objective function which guides our process of selecting common subexpressions. It is based on two simple observations. The first is that the benefit for creating a subexpression of a particular form is proportional to the number of appearances of that subexpression minus one. This is because this is the number of operations which will be saved if we reuse that subexpression in all places where it is present. For example, in Figure 2 sub-expression $X_4 + X_5$ is present in five different constants and therefore will reduce the required number of additions by four in the best case scenario. The damage is proportional to the number of common subexpression where either X_4 or X_5 are components, because we can not simultaneous reuse for the creation of these products

	B	A	X_7	X_6	X_5	X_4	X_3	X_2	X_1	X_0
a	0	1	1	0	0	0	0	0	1	1
b	1	0	0	1	0	0	0	0	0	0
c	1	1	0	0	0	0	0	0	0	0
d	0	1	0	0	0	0	0	0	0	0
e	0	1	1	0	0	0	0	0	1	0
f	0	1	1	0	0	0	0	1	1	0
g	0	0	0	0	0	0	1	0	0	0

Fig. 4. Selection process for creating common subexpressions.

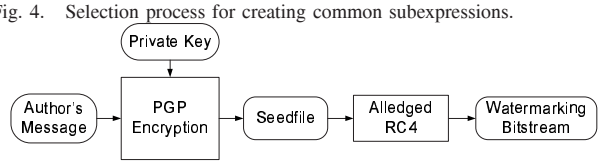

Fig. 5. Author signature conversion to bitstream.

both sub-expressions. Therefore, we try to select at each step of our procedure the sub-expression that has high benefit and low damage. The benefit and damage for each sum pair is calculated using weighted benefit and weighted damage where the weight factor for benefit is set to ten times higher weight than for damage.

At step five, we select a specific pair of sub-expressions to create a new sub-expression by probabilistically selecting a pair. During this process higher quality pairs are probabilistically favored more. We experimented with several weight factors to assign probabilities and settled for one where the probability is proportional to the square of the overall quality. Line 6 is the most important line related to the creation of the data structures which is used for the execution of the algorithm. We create a new binary position for each sub-expression which we created. In step seven we update the binary representations for all constants after a particular sub-expressions is created. Essentially, we replace on all positions where components of the sub-expression were present (indicated by binary ones) to binary value 0 and creation of the new column that indicates that sub-expression. The process terminates when the table has exactly a single one for all representations of all constants. The effectiveness of the basic mechanism of the procedure can be significantly enhanced if it is placed within iterative restart framework. The termination criteria for the iterative framework can be defined in a number of ways. In our experimentation we used one where the restarts are terminated if no improvement is found after a 1,000 iterations.

III. Watermarking

In this Section, we present the new watermarking approach for protecting MCM efficient implementations. We first discuss the overall generic watermarking approach and identify specific issues related to watermarking MCM instances. After that, we introduce eight ways to watermark MCM instances. Finally, we present and analyze in detail one of them, the mandatory merging technique.

When watermarking IC designs it is difficult to find structures of the design that can be interpreted as the signature of the designer. Additionally, this interpretation must be identified after a standard mapping from the design to a particular message is established.

The MCM watermarking is conducted in the following way. Figure 5 shows how the signature string can be created. As the first step, we encode a message, which corresponds to the designer's signature, into an infinite streams of zero and ones. The stream is cryptographically sound and from statistical and pattern points of view has all the properties of a stream produced by a random number generator. The author message and private key on one side and the stream on other side form a one-way function: it is easy to convert from the message and the key to the stream and it is impossible (at least in the cryptographical sense) to convert from the stream back to the message and the key.

Another enabling step is unique ordering of all possible MCM common subexpressions. For example, that can be accomplished by sorting all constants and sorting all common-subexpressions in each of constants starting from the smallest bit. The watermarking stream is, next, translated in a set of constraints that ensure that a particular subset of common subexpression is formed or not allowed to form. This step is accomplished by transforming the specification of the MCM problem. The final two steps of the generic watermarking approach is the actual execution of an MCM algorithm and the calculation of the strength of the signature and watermarking overhead in terms of required additional additions with respect to the solution to the non-watermarked MCM problem.

One can envision apply two types of MCM watermarking: soft and hard. In hard watermarking, all imposed constraints must be satisfied. In soft watermarking, there is the option to satisfy only a specified percentage of watermarking constraints. The idea is to explore the trade-off imposed by each constraint in terms of strength of signature and negative impact on deign metrics of interest such as area, speed, and power and select only constraints that result in favorable results. We have developed the following eight hard watermarking techniques: *Merge, No-Merge, Pairs for Merge/No-Merge, Add Binary Positions, Add Constant Multiplications, Constant Pairs Must have Common Merge Pairs, Group Merging*, and *Merge/No-Merge Lists*.

The first MCM watermarking technique embeds the signature by specifying the subexpression pairs which must be formed. Subexpression pairs are selected by the watermarking signature in two phases. All constants and intermediate shifts are sorted in increasing order. First a constant is selected. After that from the possible subexpressions for that constant a pair of subexpressions is selected. The watermarking signature determines the constant value to select a pair from by considering the first k_1 bits of the stream, where $k_1 = \lceil log_2 n \rceil$ and n is the number of current number of constants to be addressed, i.e. constants that are not completed or necessary. Next, one of the possible subexpression pairs for this constant is selected by the signature using k_2 bits, where k_2 is the minimum number of bits needed to select a subexpression pair.

The approach is illustrated using a short authorship stream that is embedded into the constant multiplication example in Figure 2. According to the watermarking technique the entire stream shown in 6 is embedded by specifying in which constant a particular subexpression pair must be created. There are seven constants. However, only five of the constants

Signature: 110011101111001

Constants to consider = {a,b,c,e,f}
$k_1 = \lceil log_2(5) \rceil$ = 3
k_1 bits: $110 \rightarrow 6\%5 = 1 \rightarrow b$
Possible pairs for b =$\{X_1 + X_2, X_1 + X_6, X_2 + X_6\}$
$k_2 = \lceil log_2(3) \rceil$ = 2
k_2 bits: $01 \rightarrow 1\%3 = 1 \rightarrow X_1 + X_3$
Merge of $X_1 + X_3 \rightarrow X_A$

Signature: 1101111001

Constants to consider = {a,c,e,f}
$k_1 = \lceil log_2(4) \rceil$ = 2
k_1 bits: $11 \rightarrow 3\%4 = 3 \rightarrow f$
Possible pairs for f =$\{X_1 + X_2, X_1 + X_4,$
$X_1 + X_5, X_1 + X_7, X_2 + X_4, X_2 + X_5, X_2 + X_7,$
$X_4 + X_5, X_4 + X_7, X_5 + X_7\}$
$k_2 = \lceil log_2(10) \rceil$ = 4
k_2 bits: $0111 = 7\%10 = 7 \rightarrow X_4 + X_5$
Merge of $X_4 + X_5 \rightarrow X_B$

Signature: 1001

Constants to consider = {a,c,e,f}
$k_1 = \lceil log_2(4) \rceil$ = 2
k_1 bits: $10 \rightarrow 2\%4 = 2 \rightarrow e$
Possible pairs for f =$\{X_1 + X_7, X_1 + X_B, X_7 + X_B\}$
$k_2 = \lceil log_2(3) \rceil$ = 3
k_2 bits: $01 = 1\%3 = 1 \rightarrow X_1 + X_B$
Merge of $X_1 + X_B \rightarrow X_C$

Fig. 6. Illustration of the Merge Watermarking Technique.

Instance	Bits	n	Orig	MIN MCM	% Imprv.
steam	14	5	101	64	36.63
ber	19	8	142	103	27.46
HL	14	8	195	124	36.41
hvs	13	8	224	152	32.14
7	14	8	234	156	33.33
DCT8	20	8	566	360	36.40
DA	20	8	589	377	35.99

TABLE I
PERFORMANCE OF MCM ALGORITHM ON MCM BENCHMARKS.

have multiple subexpression pairs (constant d and g can be eliminated trivially). We begin be determining the minimal number of bits, k_1 needed to select a constant. There are five different constants, therefore $k_1 = \lceil log_2 5 \rceil = 3$. Using the first three bits of the watermarking stream we select constant b for the selection of a subexpression. For the constant b, we enumerate all possible subexpression and select one dictated by k_2 bits of the signature stream. We see that in this step there are three possible subexpressions for constant b. Therefore two bits are used to select $X_1 + X_6$ to create an subexpression. As in the MCM algorithm, we create a new bit representation (X_A)for the merge and simplify the problem. After the merge, product with constant b can completed using a necessary merge, and therefore is no longer considered in the WM process. While there is bits of the signature still to be embedded and subexpression pairs to select from, we repeat the watermarking process. The final two step of the merge watermarking technique are shown in 6. Once the entire watermark is embedded, the remaining instance is solved by a MCM algorithm.

The second MCM watermarking approach determines the set of subexpressions pairs which will not be formed. This approach can be applied in two ways. The first is to convert the entire stream into the selection of subexpression pairs in the same way as in merging-based MCM watermarking. However, these subexpression pairs are designated as no-merge pairs and therefore as the MCM approach selects pairs of subexpressions to merge, it must verify it is not a designated no-merge pairs. Note that this technique is not transparent, due to the fact that the MCM algorithm must be altered. An alternative transparent approach which can be used in conjunction with any MCM algorithm can be applied by selecting subexpression pairs as described in the merge technique. One of the columns of the selected subexpression pair is then chosen to be eliminated from the binary specifications of all constants that contain this pair. Therefore, the selected common subexpression will never be created by any MCM algorithm.

There are two potential drawbacks to this approach. First, not only is the selected subexpression eliminated, but possibly a number of other subexpression pairs can be eliminated due to the fact that the eliminated column can not be merged with other columns unless other constants have the subexpression pair. For example, consider only constants b and c from Figure 2. If the subexpression $X_1 + X_3$ was selected for no-merge and X_1 was to be eliminated, then the binary representations for b and c would become 01001000 and 00111000 respectively. In this case, because no other constants contain a bit for X_1 the following possible subexpression have also been eliminated $\{X_1 + X_6, X_1 + X_4, X_1 + X_5\}$. However, if we were considering constants b, c, and f only $X_1 + X_6$ would be eliminated due to the fact that the other subexpression can be created in constant f.

The second drawback is that for each no-merge pair that is eliminated a post-processing step must be performed in order to correct for the elimination of the merge by introducing the eliminated additions, resulting in addition operations that is no shared. The penalty from these drawbacks can be minimized in several ways. For example, we can only allow the selection of subexpression pairs which appear in less than k constants.

An extension of the no-merge/merge watermarking approach is to designate a set of no-merge/merge pairs which is conditional. In this case for each signature defined pair of subexpression pairs only one of the two cases can be applied, either the pair must be merged or the pair may not be merged. In this case, a longer signature stream can be embedded, but the strength of proof declines due to the absence of absolute constraints in the MCM.

Two other simple watermarking techniques which can be applied to the MCM procedure are to introduce extra conditions or constants to the problem. One way is to extend the original binary representation to include additional bits. The bit values are then determined by the signature stream. While the MCM algorithm will address these bits, these bits will not be actually introduced in the shift and add implementation. Essentially, the bits are introduced in order to constrain the MCM approach to address the problem differently. Another approach is to add additional constant multiplications which were not necessary. In this case, the MCM approach is influenced by the presence of additional subexpression. The values of the constants to be added can be determined directly from the signature streams.

The remaining three techniques focus on watermarking

Instance	MIN MCM	10n Total +	10n Inc. %	10n SofP	20n Total +	20n Inc. %	20n SofP	50n Total +	50n Inc. %	50n SofP	75n Total +	75n Inc. %	75n SofP
steam	64	84	23.8	10^{-24}	90	28.8	10^{-39}	122	47.5	10^{-85}	141	54.6	10^{-119}
ber	103	127	18.9	10^{-30}	135	23.7	10^{-57}	174	40.8	10^{-133}	218	52.7	10^{-199}
HL	124	259	52.1	10^{-37}	265	53.2	10^{-66}	309	59.8	10^{-154}	348	64.3	10^{-222}
hvs	152	180	15.5	10^{-39}	201	24.3	10^{-66}	240	36.6	10^{-143}	265	42.6	10^{-203}
7	156	187	16.5	10^{-39}	199	21.6	10^{-66}	245	36.3	10^{-145}	284	45.0	10^{-207}
DCT8	360	406	11.3	10^{-36}	412	12.6	10^{-72}	459	21.5	10^{-179}	487	26.0	10^{-269}
DA	377	429	12.1	10^{-37}	419	10.0	10^{-73}	474	20.4	10^{-181}	495	23.8	10^{-271}

TABLE II
EXPERIMENTAL RESULTS FOR THE MERGE WATERMARKING METHOD.

through the use of combinations of common subexpressions. The MCM problem can be constrained by specifying that two constants must share a given number of merge elements. Alternatively, group constraints can be placed on the merging of common subexpressions. Lastly, the signature stream can be converted into lists of possible merge or no-merge pairs.

Any of these techniques can be converted into soft watermarking techniques by simply specifying that only a set percentage of the watermarking constraints must be ensured. For example, in the first technique instead of forcing each subexpression pair be merged, a soft technique would specify that 75% of the specified merge pairs must be used. The advantage of soft watermarking techniques is that a longer signature can be translated into constraints.

IV. Experimental Results

In this section, we present the experimental analysis of the MCM algorithm and the merge watermarking technique. In order to perform our analysis we used a subset of MCM benchmarks which are shown in Table I. The benchmarks were selected from a number of real-life applications. Details about each benchmark are shown including the number of bits needed for representation of the constant values, the number of sets of constants in each benchmark, and the number of additions necessary if zero optimization is performed when translating each constant from multiplications to shifts and adds. In the final two columns of the table, we present the amount of optimization the proposed MCM algorithm was able to achieve in terms of the minimum number of additions in the solution found by the MCM algorithm after 100 restarts and the percentage improvement. For the examples, the maximum reduction in terms of number of additions was 36.6% and the average reduction 34.1%.

In order to evaluate the effectiveness of the merge MCM watermarking technique for the MCM problem, we embedded signatures of length 10, 20, 50, 75, and 100 times the number of constant sets in each benchmark. Once the watermark is embedded we apply the MCM algorithm with 100 restarts to find the final solution. The results are shown in Table II. For each watermark length we present the total number of additions in the final solution, the overhead of the watermarking approach over the MCM solution without watermarking, and the strength of proof for each watermarked solution (the likelihood of the solution being selected at random). Note, that extremely high strength of proof is achieved with the embedding of short watermarks. However, it is achieved at the expense of at least 10% overhead in terms of extra additions. Therefore, most likely most practical schemes are ones that apply very short watermarks.

V. Conclusion

We introduced a new iterative probabilistic greedy algorithm for the MCM problem. The algorithm is very fast and achieved a $\sim$30% reduction in the number of additions necessary for implementation of MCM instances. Furthermore, we have developed eight hard and soft watermarking techniques for IPP at the MCM level. Strong strength of proof can be achieved even for very short watermarks.

References

[1] J. Guo, R. Ju, and J. Chen. An efficient 2-d DCT/IDCT core design using cyclic convolution and adder-based realization. *CirSysVideo*, 14(4):416–428, April 2004.

[2] H. Johansson and L. Wanhammar. Wave digital filter structures for high-speed narrow-band and wideband filtering. *IEEE Trans. Circuits Syst. II*, 46(6):726–741, June 1999.

[3] N. Julien, S. Gailhard, and E. Martin. Low power synthesis methodology with data format optimization applied on a dwt. *J. VLSI Signal Process. Syst.*, 35(2):195–211, 2003.

[4] A. Kahng et al. Constraint-based watermarking techniques for design ip protection. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(10):1236–1252, 2001.

[5] M. D. Macleod and A. G. Dempster. Common subexpression elimination algorithm for low-cost multiplierless implementation of matrix multipliers. *Electronics Letters*, 40(11):651–652, May 2004.

[6] A. Oliveira. Techniques for the creation of digital watermarks in sequential circuit designs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(9):1101–1117, 2001.

[7] R. Pasko, P. Schaumont, V. Derudder, S. Vernalde, and D. Durackova. A new algorithm for elimination of common subexpressions. *IEEE Trans. Computer-Aided Design*, 18:58–68, 1999.

[8] J. O. Penhollow. *Study of arithmetic recording with applicaitons in multiplication and division*. PhD thesis, University of Illinois, Urbana, September 1962.

[9] M. Potkonjak, M. Srivastava, and A. Chandrakasan. Multiple constant multiplications: efficient and versatile framework and algorithms for exploring common subexpression elimination. *IEEE Trans. on CAD*, 15(2), Feb 1996.

[10] G. Qu and M. Potkonjak. *Intellectual Property Protection in VLSI Design Theory and Practice*. Kluwer Publishing, 2003.

[11] J. E. Robertson. Theory of computer arithmetic employed in the design of the computer at the university of illinois. Technical Report 319, University of Illinois, Urbana, June 1960.