A Variable Radix-2^r Algorithm for Single Constant Multiplication

Ahmed Liacha[1,2,3], Abdelkrim K. Oudjida[1], Farid Ferguene[2], José Monteiro[3], and Paulo Flores[3]
Centre de Développement des Technologies Avancées[1] (CDTA), System Architecture and Multimedia Division,
Cité du 20 août 1956, Baba-Hassen, Algiers, Algeria. liacha@cdta.dz , a_oudjida@cdta.dz
Université des Sciences et de la Technologie Houari Boumediene[2] (USTHB), LRPE Laboratory,
BP 32, El Alia 16111, Algiers, Algeria. fferguene@usthb.dz
INESC-ID/IST[3], Universidade de Lisboa, Rua Alves Redol 9,1000-029 Lisbon, Portugal. jcm@inesc-id.pt , pff@inesc-id.pt

***Abstract*— In previous work, a fully predictable sub-linear runtime heuristic for the multiplication by a constant based on radix-2^r arithmetic using a fixed radix was developed, called RADIX-2^r. In this paper, we introduce a new constant multiplication algorithm based also on Radix-2^r arithmetic but considering a variable radix. The new version is named RADIX-2^r-VAR. Using a variable radix allows to optimize the average number of additions in the constant multiplication since a larger search space is explored. The new RADIX-2^r-VAR recoding requires an average of 4.4% and 2.2% less additions than RADIX-2^r for 24 and 32 bits, respectively. The RADIX-2^r-VAR algorithm is combined with RADIX-2^r for more improvements in the average number of additions. An overall saving of 6.7% and 5.5% is obtained over RADIX-2^r for 24 and 32 bits, respectively.**

Keywords*— *High-Speed and Low-Power Design, Linear-Time-Invariant (LTI) Systems, Multiplierless Single/Multiple Constant Multiplication (SCM/MCM). Radix-2^r Arithmetic.

I. Background and Motivation

Multiplication of data input by a constant is a fundamental operation in many DSP and control systems, such as fast Fourier transform (FFT), discrete cosine transforms (DCT), FIR/IIR filters, and digital controllers. To be efficiently designed, instead of a full-block generic multiplier, an implementation based on addition/subtraction and shift operations should be used. This problem is called single constant multiplication (SCM). In this paper, we assume that addition and subtraction have the same hardware cost, and that the shift is costless since it can be realized without any gates, i.e., just by hardwiring. The single constant multiplication (SCM) problem is defined as finding the minimum number of addition/subtraction operations to implement the constant multiplication. The computational complexity of SCM is conjectured to be NP-hard [1]. Hence the optimal algorithms are impractical, only heuristics can respond in a reasonable amount of time. Due to the importance of problem, many efficient heuristics have been proposed for the multiplierless design of the constant multiplier block, targeting the minimization of the number of additions (hardware design complexity). They are classified in four main categories:

- Digit-recoding algorithms such as minimal signed digit (MSD) recoding [2], the canonical signed digit (CSD) representation [3], Booth recoding [4], and RADIX-2^r recoding [5] [6][7].
- Common subexpression elimination (CSE) using pattern matching performed after an initial digit-recoding. Typical examples are Hartley [8], Lefèvre [9], Boullis [10], and Aksoy [11][12].
- Directed acyclic graph (DAG) based algorithms. This category includes Bernstein [13], MAG [14], H(k) [15], Hcub [16], and DFSearch [17].
- Mixed algorithms combining CSE and DAG such as the optimal algorithm BIGE [1].

In previous work, a new heuristic (RADIX-2^r) for constant multiplication based on radix-2^r arithmetic was developed. It has the major advantage of being fully predictable in the maximum number of additions (upper-bound), in the average number of additions, and in the number of cascaded adders (adder-depth) forming the critical path. In RADIX-2^r the two's complement representation of each constant is split into slices of the same bit-length (r+1). Since all slices have equal bit-size, the decomposition covers only a part of the research space. Therefore better solutions could be missed.

The main purpose of this work is to further reduce the average number of additions (*Avg*) by developing a new digit recoding heuristic (RADIX-2^r-VAR) based on the radix-2^r arithmetic. This algorithm preserves the same advantage as RADIX-2^r in terms of predictability in upper bound and adder depth. In RADIX-2^r-VAR the two's complement recoding of the constant is split into slices of variable bit-length, allowing to obtain a better solution (minimal number of adders) since a larger search space is explored, while keeping the execution time within acceptable limits.

This paper is organized as follows. Section II gives an overview on the RADIX-2^r SCM algorithms. Section III presents the new RADIX-2^r-VAR SCM algorithm. Results are discussed in Section IV. Finally, Section V provides some concluding remarks and suggestions for future work.

II. Radix-2r SCM Algorithm

In radix-2^r, a non-negative N-bit constant C is expressed as follows:

$$C = \sum_{j=0}^{(N+1)/r-1} \left(c_{rj-1} + 2^0 c_{rj} + 2^1 c_{rj+1} + 2^2 c_{rj+2} + \cdots + 2^{r-2} c_{rj+r-2} - 2^{r-1} c_{rj+r-1} \right) \times 2^{rj}$$
$$= \sum_{j=0}^{(N+1)/r-1} Q_j \times 2^{rj} , \quad (1)$$

where $c_{-1} = c_N = 0$ and $r \in \mathrm{N}^*$. In (1) the two's complement representation (TCR) of C is divided into $\lceil (N+1)/r \rceil$ slices (Q_j), each one has r+1 bit length. Each of two adjacent slices has one overlapping bit. A digit-set $DS(2^r)$ corresponds to (1), such as $Q_j \in DS(2^r) = \{-2^{r-1}, -2^{r-1}+1, \ldots, -1, 0, 1, \ldots, 2^{r-1}-1, 2^{r-1}\}$.

The sign of the Q_j term is given by the c_{rj+r-1} bit, and $|Q_j| = 2^{k_j} \times m_j$, with $k_j \in \{0,1,2,\ldots,r-1\}$ and $m_j \in OM(2^r) \cup \{0,1\}$, where $OM(2^r) = \{3,5,7,\ldots,2^{r-1}-1\}$. $OM(2^r)$ is the set of odd positive digits in RADIX-2^r recoding, with $|OM(2^r)| = 2^{r-2} - 1$. Finally, the constant C can be expressed as follows

$$C = \sum_{j=0}^{(N+1)/r-1} (-1)^{c_{rj+r-1}} \times m_j \times 2^{rj+k_j}. \quad (2)$$

There exists a number of metrics for single (SCM) constant multiplication, but the most commonly used are:

- Upper-bound (*Upb*): For each *N*-bit constant C_i, let A_i be the number of additions for the implementation of $C_i \times X$. Thus, *Upb* = max (A_i).
- Average (*Avg*): The average number of additions for *N* bit constant is $Avg = \sum_{i=1}^{m} A_i / m$, where m is the total number of C_i constants, i.e., $m = 2^N - 1$.
- Adder-Depth (*Ath*): Let D_i be the number of cascaded adders along any path i from the input to any of the outputs in the logic circuit of the constant multiplication. *Ath* is equal to max (D_i).

The average (Avg), the maximum adder-depth in cascaded additions (Ath), and the maximum number of additions required by RADIX-2^r are reported in Table I. The main feature of RADIX-2^r is that it is fully predictable and it has a sublinear runtime complexity $O(N/r)$ with respect to the constant size N [7].

III. RADIX-2^r-VAR SCM ALGORITHM

In radix-2^r, a non-negative *N*-bit constant C is generally expressed by (1). But C can be expressed differently as:

$$\begin{aligned} C &= \sum_{j=0}^{1} \left(c_{rj-1} + 2^0 c_{rj} + 2^1 c_{rj+1} + 2^2 c_{rj+2} + \cdots + 2^{r-2} c_{rj+s-2} - 2^{r-1} c_{rj+r-1}\right) \times 2^{rj} \\ &= \left(c_{-1} + 2^0 c_0 + 2^1 c_1 + 2^2 c_2 + \ldots + 2^{r-2} c_{r-2} - 2^{r-1} c_{r-1}\right) + \\ &\quad \left(c_{r-1} + 2^0 c_r + 2^1 c_{r+1} + 2^2 c_{r+2} + \ldots + 2^{N-(r+1)} c_{N-1} - 2^{N-r} c_N\right) \times 2^r \\ &= Q_0 + Q_1 \times 2^r, \end{aligned} \quad (3)$$

where $c_{-1} = c_N = 0$ and $r \in \mathrm{N}^*$. The TCR of the constant C is split into two of two's complement slices (Q_0, Q_1). The two slices are adjacent and have one overlapping bit. In this algorithm, the bit length of each slice can vary from 2 up to *N* bits. The sum of the bit-length of the two slices is equal to *N*+2.

A digit-set $DS(2^r)$ corresponds to (3), such as $Q_0 \in DS(2^r) = \{-2^r, -2^r+1, \ldots, -1, 0, 1, \ldots, 2^{r-1}-1, 2^{r-1}\}$ and $Q_1 \in DS(2^{N-(r-1)})$, where r is variable, and $r \in \{1,2,\ldots,N\}$.

TABLE I. RADIX-2^r SCM FOR A NONNEGATIVE *N*-BIT CONSTANT

Metrics	Equations
Adder cost	$Upb(r) = \left\lceil \frac{N+1}{r} \right\rceil + 2^{r-2} - 2$
Adder depth	$Ath(r) = \left\lceil \frac{N+1}{r} \right\rceil + r - 3$
Average	$-1 + Avg_{pp} + Avg_{om} \le Avg(r) \le -2 + Avg_{pp} + 2^{r-2}$ with $Avg_{pp} = (1-2^{-r}) \times \left\lceil \frac{N+1}{r} \right\rceil$, $Avg_{om} = \sum_{j=0}^{\left\lceil \frac{N+1}{r} \right\rceil - 1} \left\{ \sum_{k=1}^{2^{r-2}-1} P(m_{jk}) \times [1 - P(m_{jk})]^j \right\}$, $P(m_{jk}) = \frac{\log_2 \left\lceil \frac{2^{r-1}}{2 \times k + 1} \right\rceil}{2^{r-1}}$,
$r = 2 \cdot \mathrm{W}(\sqrt{(N+1) \cdot \log(2)})/\log(2)$	

W: Lambert function; $\lceil\ \rceil$: Ceiling function.

The sign of Q_0 and Q_1 terms is given by the c_{r-1} and c_N bits, respectively. The two terms Q_0 and Q_1 can be written as: $|Q_0| = 2^{k_0} \times m_0$ and $|Q_1| = 2^{k_1} \times m_1$, with $k_0 \in \{0,1,2,\ldots,r-1\}$, $k_1 \in \{0,1,2,\ldots,N-r\}$, $m_0 \in OM(2^r) \cup \{0,1\}$, $m_1 \in OM(2^{N-(r-1)}) \cup \{0,1\}$, where $OM(2^r) = \{3,5,7,\ldots,2^{r-1}-1\}$. $OM(2^r)$ is the set of odd positive digits in RADIX-2^r-VAR recoding, with $|OM(2^r)| = 2^{r-2} - 1$. Finally, the constant C can be expressed as

$$\begin{aligned} C &= (-1)^{c_{r-1}} \times m_0 \times 2^{k_0} + (-1)^{c_N} \times m_1 \times 2^{r+k_1} \\ &= (-1)^{c_{r-1}} \times m_0 \times 2^{k_0} + m_1 \times 2^{r+k_1}. \end{aligned} \quad (4)$$

The main objective is to decrease *Avg* without increasing *Upb* and *Ath* as in RADIX-2^r. As shown in the pseudo-code given in Fig. 1, the RADIX-2^r-VAR algorithm is divided into two parts. The first part of the algorithm consists in constructing (off-line) a table containing the recoding with an optimized adder cost for all partial products that can be produced during the different partitioning of the constant. This table is built only one time and saved. In this part, we apply the *A*-operation [16] to optimize the recoding of the terms Q_0 and Q_1, so that they can be expressed as $|Q| = u \times 2^l + (-1)^e \times v \times 2^h$, where l, $h \ge 0$ are integers denoting left shifts, and $e \in \{0,1\}$ represents the sign that indicates if an addition or a subtraction operation is to be performed. Using the new expression of Q, the equation (4) can be rewritten in more details as $C = (-1)^{c_{r-1}} \times \left[u_0 \times 2^{l_0} + (-1)^{e_0} \times v_0 \times 2^{h_0}\right] + \left[u_1 \times 2^{l_1} + (-1)^{e_1} \times v_1 \times 2^{h_1}\right] \times 2^r$, (5) where, $u, v \in \{0,1,3,5,\ldots,2^{(r/2)-1} - 1\}$, $l \in \{0,1,2,\ldots,r-1\}$, $e \in \{0,1\}$, and $h \in \{0,1,2,\ldots,(r/2)-1\}$.

For the same value of Q, the quintuplet (*u, l, e, v, h*) is not unique. Therefore several valid combinations may exist. For instance $|Q| = 25$ can be expressed as $25 = 3 \times 2^3 + 1 \times 2^0$, or $25 = 5 \times 2^2 + 5 \times 2^0$, or $25 = 7 \times 2^2 - 3 \times 2^0$.

For each odd $|Q|$ varying from 1 to $2^{r-1}-1$, we exhaustively explore all (*u, l, e, v, h*) possibilities and select the least adder consumer combination according to the following priority ordering: $(u,v)=(u,0)$; $(u,v)=(1,1)$; $(u, \pm 1 \times 2^0)$; $(u,v)=(1,3)$ or $(3,1)$; $(u,v)=(3,3)$; $(u,v)=(1,5)$ or $(5,1)$; $(u,v)=(5,5)$; $(u,v)=(1,7)$ or $(7,1)$; $(u,v)=(7,7)$; $(u,v)=(3,5)$ or $(5,3)$; $(u,v)=(3,7)$ or $(7,3)$; $(u,v)=(5,7)$ or $(7,5)$.

RADIX-2^r-VAR(C)
N: bit length of *C*
bc: best cost
brec: best recoding
n: number of transitions
OM_Table: optimal recoding of odd multiples
begin
1: TCR(*C*) // two's complement representation of *C*
2: Determine vector *V* containing 0 to 1 and 1 to 0 transitions
3: **for** *i* = 2 to *n* do
J = *V*(*i*).
$Q_0 = (C_j...0)$ // calculate Q_0 value and make it odd and positive.
$Q_1 = (C_N...C_j)$ // calculate Q_1 value and make it odd and positive.
Rec // check Q_0 and Q_1 recoding in *OM_Table*
Cost // calculate the cost
if *i* = 2 begin
brec = *rec*
bc = *cost*
elseif *cost* < *bc* begin
brec = *rec*
bc = *cost*
end // *for*
return *brec* and *bc*

Fig. 1. The RADIX-2^r-VAR algorithm.

This ordering reduces the number of additions in the whole recoding of the constant C by maximizing the occurrences of the digit "1", then of "3", etc, and minimize the occurrences of those requiring a high number of adders [6].

The second part consists in finding the recoding (*brec*) that ensures the minimum cost (*bc*). We split the constant into two TCS (4), calculate the values of Q_0 and Q_1, and then check the recoding of Q_0 and Q_1 in the table of partial product recoding. We start with two slices of bit-length of 2 and *N*+1 for the slice 1 and 2, respectively. The process repeats in the same way, increasing the bit length of the first slice by one and decreasing the second slice by one until we reach a bit length of *N*+1 and 2 for the slice 1 and 2, respectively. Fig. 2(a) depicts all possible partitionings of the constant C= $(455)_{10}$.

In radix-2^r representation, the slices (10), (110), (1110), and (11…10) are the same and equal to -1. Note that the "0" was added and is not part of the original number. Likewise, the slices (01) and (0…01) are the same and equal to 1. We apply these two equivalences to reduce the number of operations in our algorithm by eliminating the groups which gives the same value. Thus, instead of varying the partitioning of the constant by incrementing the bit-size of the slices 1 by one, the partitioning is realized whenever there is a transition from 1 to 0 or from 0 to 1. Fig. 2(b) depicts all possible partitionings defined by the transition from 1 to 0 and from 0 to 1 of the constant C= (455)10.

A. Illustrative Example

To illustrate RADIX-2^r-VAR algorithm, let us consider *C*= $(571113)_{10}$, which in TCR is represented as $(01000101101101101001)_2$. Thus, the constant bit-size is *N*+1 (20+1=21 bits for 571113).

We scan all possible bit-length combinations of the two slices Q_1 and Q_2. For each value of Q_1 and Q_2 we check in the table of recoding, calculate the adder cost, and finally take the minimum. Depending on the choice of grouping, several solutions of recoding for the constant *C* are obtained, for example *C* can be written as:

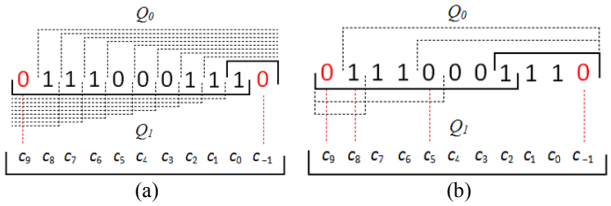

Fig. 2. Partitioning of $(455)_{10}$ in RADIX-2^r-VAR. (a) all cases, (b) with transition from 1-0 and 0-1.

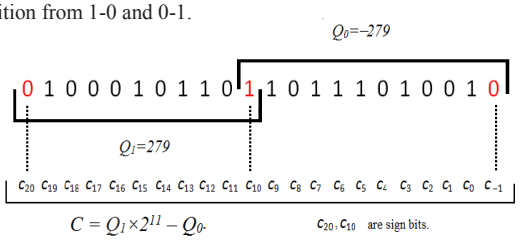

Fig. 3. Partitioning of $(571113)_{10}$ in RADIX-2^r-VAR.

571113=71389<<3+1, with 71389=279<<8–35, 279=9<<5–9, 35=1<<5+3, 9=1<<3+1, and 3=1<<1+1. Which gives an adder cost of 6.
571113=35695<<4–7, with 35695=35<<10–145, 145=9<<4+1, 35=1<<5+3, 9=1<<3+1, 7=1<<3–1, and 3=1<<1+1. Which gives an adder cost of 7.
571113=17847<<5 +9, with 17847=35<<9 –73, 73=9<<3+1, 35=1<<5+3, 9=1<<3+1, and 3=1<<1+1. Which gives an adder cost of 6.
571113=2231<<8–23, with 2231=35<<6–9, 35=1<<5+3, 23=3<<3–1, 9=1<<3+1, and 3=1<<1+1. Which gives an adder cost of 6.

The recoding that ensures a minimum adder cost (Fig. 3) is $C=279\times2^{11}-279=571113$, with $279=9\times2^5-9$ and $9=1\times2^3+1$. This recoding involves the pre-computation of two Partial Products (PPs) only $\{279\times X, 9\times X\}$. It has to be noted that the adder cost and the adder depth for *C*=571113 are 3 and 3, respectively. Note that RADIX-2^r and CSD recodings of C=571113 requires both 7 additions, where RADIX-2^r-VAR needs only 3. A saving of 4 additions is achieved. Compared to RADIX-2^r algorithm that has a sublinear runtime complexity $O(N/r)$, RADIX-2^r-VAR algorithm improves the number of additions at the cost of small increase in execution time $O(N)$. For more improvement in average number of additions, the RADIX-2^r-VAR algorithm is combined to RADIX-2^r algorithm. Here for each constant we execute the two algorithms RADIX-2^r and RADIX-2^r-VAR and choose the minimum adder cost.

IV. EXPERIMENTAL RESULTS

The average number of additions (*Avg*) of both algorithms (RADIX-2^r-VAR and combined RADIX-2^r-VAR with RADIX-2^r) has been calculated exhaustively for each *C* varying from 0 to $2^{N-1}-1$, with *N*=8, 16, and 24. For *N*=32, we have calculated *Avg* using constant sets with 10^4, 10^5, 10^6, and 10^7 uniformly distributed random values of *C*. The *Avg* value oscillates around 7.985 additions for the pure RADIX-2^r-VAR and 7.716 additions for the combined RADIX-2^r-VAR with RADIX-2^r. Note that the difference between the average obtained results for different sizes of the constant sets is insignificant ($<10^{-3}$).

The results of comparison between the RADIX-2^r-VAR algorithm and the CSD representation are reported in Table II. For N=24 and N=32, we achieved a savings of 20.862% and 21.026% additions over CSD, respectively. Table III shows a comparison between RADIX-2^r-VAR and RADIX-2^r algorithms. For N equals 16, 24, and 32 bits, RADIX-2^r-VAR uses 3.268%, 4.444%, and 2.204% less additions than RADIX-2^r, respectively. Results of the combined RADIX-2^r-VAR to RADIX-2^r are reported in Table IV. We have obtained better savings: 4.866%, 6.699%, and 5.499% for N=16, N=24, and N=32, respectively.

The results show that RADIX-2^r-VAR algorithm has higher capability to find better solutions when compared to CSD and RADIX-2^r algorithms. The combination of RADIX-2^r with RADIX-2^r-VAR helps to further improve the adder cost.

Another performance indicator of the recoding is the smallest value that requires q additions, for q varying from 1 to the upper-bound of the recoding. The results are reported in Table V for a 32-bit constant (24-bit constant only for the RADIX-2^r-VAR). Note that starting from q=4, higher values are obtained by RADIX-2^r-VAR when compared to CSD.

V. CONCLUSION AND FUTURE WORK

Based on radix-2^r arithmetic, we have developed a new algorithm (RADIX-2^r-VAR) for single constant multiplication with a variable radix. The latter improved the average number of additions over RADIX-2^r algorithm and CSD recoding. Further improvements are achieved combining RADIX-2^r-VAR and RADIX-2^r. The average number of additions has been improved at the cost of a small increase in execution time.

Our on-going work deals with the application of RADIX-2^r-VAR algorithm to the multiple constant multiplication (MCM) problem.

TABLE II. RADIX-2^r-VAR VERSUS CSD: AVG

N (bits)	CSD	RADIX-2^r-VAR	Saving %
8	2.111	2.001	5.210
16	4.777	3.995	16.370
24	7.444	5.891	20.862
32	10.111	7.985	21.026

TABLE V. RADIX-2^r-VAR VERSUS RADIX-2^r , CSD AND EXHAUSTIVE SEARCH: SMALLEST VALUES UP TO A 32-BIT CONSTANT

Number of additions (q)	CSD	RADIX-2^r	RADIX-2^r-VAR	Exhaustive search [9]
1	3	3	3	3
2	11	11	11	11
3	43	43	43	43
4	171	339	343	683
5	683	2387	2347	14709
6	2731	18605	14635	699829
7	10923	148825	158153	171398453
8	43691	1186451	898219	–
9	174763	9521325	6994263	–
10	699051	143739053	NA	–
11	2796203	2291222701	NA	–
12	11184811	–	–	–
13	44739243	–	–	–
14	178956971	–	–	–
15	715827883	–	–	–
16	2863311531	-	-	-

NA: Not available

TABLE III. RADIX-2^r-VAR VERSUS RADIX-2^r: *Avg*

N (bits)	RADIX-2^r	RADIX-2^r-VAR	Saving %
8	2.001	2.001	0.000
16	4.130	3.995	3.268
24	6.165	5.891	4.444
32	8.165	7.985	2.204

TABLE IV. COMBINING RADIX-2^r WITH RADIX-2^r-VAR VERSUS RADIX-2^r: *Avg*

N (bits)	RADIX-2^r	COMBIN. RADIX-2^r & RADIX-2^r-VAR	Saving %
8	2.001	2.001	0.000
16	4.130	3.929	4.866
24	6.165	5.752	6.699
32	8.165	7.716	5.499

REFERENCES

[1] J. Thong and N. Nicolici, "An optimal and practical approach to single constant multiplication," IEEE Trans. on Computer-Aided Design of Integrated Circ. and Systems, vol. 30, no. 9, pp. 1373-1386, Sep. 2011.

[2] I-C. park and H-J. Kang, "Digital filter synthesis based on minimal signed digit representation, " in Proceeding of DAC, 2001, pp, 468-473

[3] A. Avizienis, "Signed-digit number representation for fast parallel arithmetic," IRE Trans. on Electronic Computers, vol. EC-10, No. 3, pp. 389–400, September 1961.

[4] Y.E. Kim et *al*., "Efficient Design of Modified Booth Multipliers for Predetermined Coefficients," Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), pp. 2717-2720, Island of Kos, Greece, May 2006.

[5] A.K. Oudjida and N. Chaillet, "Radix-2r Arithmetic for Multiplication by a Constant," IEEE Trans. on Circuits and Systems II: Express Brief, 61, (5), pp. 349-353, May 2014.

[6] A.K. Oudjida, N. Chaillet, and M.L. Berrandjia, "Radix-2r Arithmetic for Multiplication by a Constant: Further Results and Improvements," IEEE Trans. on Circ. and Systems II, 62, (4), pp. 372-376, April 2015.

[7] A.K. Oudjida, A. Liacha, M. Bakiri, and N. Chaillet, "Multiple Constant Multiplication Algorithm for High Speed and Low Power Design," IEEE Trans. on Circuits and Systems II: Express Brief, vol. 63, no 2, pp. 176-180, February 2016.

[8] R.I. Hartley, "Subexpression Sharing in Filters Using Canonic Signed Digit Multipliers," IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing, vol. 43, No. 10, pp. 677-688, October 1996.

[9] V. Lefèvre, "Multiplication by an Integer Constant," INRIA Research Report, No. 4192, Lyon, France, May 2001.

[10] N. Boullis and A. Tisserand, "Some Optimizations of Hardware Multiplication by Constant Matrices," IEEE Trans. on Computers (TC), vol. 54, No. 10, pp. 1271-1282, October 2005.

[11] Levent Aksoy et *al*., "Exact and approximate algorithms for the optimization of area and delay in multiple constant multiplications," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), 27(6):1013-1026, June 2008.

[12] L. Aksoy et *al*.,,"Optimization algorithms for the multiplierless realization of linear transforms". ACM Trans. on Design Automation of Electronic Systems (TODAES), 17(1):3:1 - 3:27, January 2012.

[13] R.L. Bernstein, "Multiplication by Integer Constant," Software– Practice and Experience 16, 7, pp. 641-652, 1986.

[14] O. Gustafsson, A.G. Dempster, and L. Wanhammar, "Extended Results for Minimum-Adder Constant Integer Multipliers," Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), vol. 1, pp. I-73 I-76, Scottsdale Arizona, USA, May 2002.

[15] A. Dempster and M. Macleod, "Using Signed-Digit Representations to Design Single Integer Multipliers Using Subexpression Elimination," Proceedings of the IEEE International Symp.m on Circuits and Systems (ISCAS), vol. 3, pp. III-165-168, Vancouver, Canada, May 2004.

[16] Y. Voronenko and M. Püschel, "Multiplierless Multiple Constant Multiplication," ACM Trans. on Algorithms (TALG), vol. 3, No. 2, article 11, pp. 1-38, May 2007.

[17] L. Akso et *al*., "Search algorithms for the multiple constant multiplications problem: Exact and approximate". Microprocessors and Microsystems: Embedded Hardware Design (MICPRO), 34(5):151-162, August 2010. Special issue on selected papers from NORCHIP, 2008.