HAL
open science

Radix-2r Arithmetic for Multiplication by a Constant.

Abdelkrim K. Oudjida, Nicolas Chaillet

To cite this version:

Abdelkrim K. Oudjida, Nicolas Chaillet. Radix-2r Arithmetic for Multiplication by a Constant.. IEEE Transactions on Circuits and Systems II: Express Briefs, 2014, 61, pp.349-353. 10.1109/TCSII.2014.2312799 . hal-01002468

HAL Id: hal-01002468

https://hal.science/hal-01002468v1

Submitted on 6 Jun 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Radix-2^r Arithmetic for Multiplication by a Constant

Abdelkrim K. Oudjida, and Nicolas Chaillet, *Member, IEEE*

***Abstract*—In this paper, radix-2^r arithmetic is explored to minimize the number of additions in the multiplication by a constant. We provide the formal proof that for an *N*-bit constant, the maximum number of additions using radix-2^r is lower than Dimitrov's estimated upper-bound (2.*N*/log(*N*)) using double base number system (DBNS). In comparison to canonical signed digit (CSD) and DBNS, the new radix-2^r recoding requires an average of 23.12% and 3.07% less additions for 64-bit constant, respectively.**

***Index Terms*—Double Base Number System (DBNS), High-Speed and Low-Power Design, Linear-Time-Invariant (LTI) Systems, Multiplierless Single/Mutiple Constant Multiplication (SCM/MCM), Radix-2^r Arithmetic.**

I. Background and Motivation

Many applications in DSP and control, such as LTI filters/controllers, involve the computation of a large number of multiplications of one variable by a set of constants. To be efficiently handled the implementation must be multiplierless, that is, using exclusively additions, subtractions, and left-shifts. This problem is called single/multiple constant multiplication (SCM/MCM). Its computational complexity still seems to be unknown. But because the solution space to explore is so huge, one has to use heuristics. Due to the importance of this issue, a large number of heuristics have been proposed. They are classified in four categories:

- Digit-recoding algorithms such as the canonical signed digit (CSD) representation [1], Booth recoding [2], and Dimitrov's DBNS recoding [3];
- Common subexpression elimination (CSE) using pattern matching performed after an initial digit-recoding. Typical examples are Hartley [4], Lefèvre [5], and Boullis [6];
- Directed acyclic graph (DAG) based algorithms. This category includes Bernstein [7], MAG [8], H(*k*) [9], and Hcub [10];
- Mixed algorithms combining CSE and DAG such as the recent optimal algorithm BIGE [11].

Surveys and detailed comparative studies showing pros and cons of various algorithms are given in [10][11].

Despite the large number of proposed heuristics, to our knowledge, only three heuristics are accompanied with their respective addition-cost complexity [11][12]. This issue is very important as it informs on the heuristic capabilities and limitations with regard to the constant bit-size (*N*). For low values of *N* (*N*≤32), H(*k*) [9] and Hcub [10] are, up to date, considered as the best heuristics for SCM and MCM, respectively. As long as their respective addition complexities are unknown, there is no guarantee that they will preserve their leading positions for high values of *N*.

A.K. Oudjida is with "Centre de Développement des Technologies Avancées", CDTA, Cité du 20 août 1956, Baba-Hassen, Algiers, Algeria.
N. Chaillet is with FEMTO-ST Institute, UFC/CNRS/ENSMM/UTBM, 32 avenue de l'Observatoire, 25044 Besançon, Cedex, France.

It was shown in [13] that the number of additions for an *N*-bit constant in CSD is bounded by (*N*+1)/2–1 and tends asymptotically to an average value of (*N*/3)–8/9, which yields 33% saving over the naive add-and-shift approach. Pinch [14] was the first to prove that the multiplication by a constant is sublinear: $O(N/(\log(N))^{\alpha})$ with α<1, where log is the natural logarithm (Napierian). Based on the DBNS arithmetic [15], Dimitrov [3] showed that the condition α<1 in Pinch's complexity is not necessary, decreasing therefore the upper limit to $O(N/\log(N))$. Even more, in 2011, Dimitrov [16] estimated the hidden constant in the big-*O* notation as being less than 2. Since then, 2.*N*/log(*N*) is considered as the *lowest* analytic upper-bound estimated so far. On the other hand, according to [5], Ross Donelly was the first to determine in 2000 via an exhaustive search that 699829 is the smallest value (20 bits) that can not be obtained with 5 adders or less. Thong [11] did better with the exact BIGE algorithm as he conjectured (no proof) that 7 additions are enough up to 32 bits. Though BIGE guarantees optimality via an exhaustive search, it requires an exponential runtime and storage with respect to *N* [11]. Nevertheless, with BIGE we can observe how much any heuristic is far from optimality up to 32 bits.

The main purpose of this work is the minimization of the total number of additions. Based on the radix-2^r arithmetic [17] [18], a new digit recoding is proposed with an upper limit equal to $\lceil (N+1)/r + 2^{r-2} - 2 \rceil$, where, $r = 2 \cdot \mathrm{W}\left(\sqrt{(N+1)\cdot\log(2)}\right)/\log(2)$, W is the Lambert function, and $\lceil\ \rceil$ is the ceiling function (e.g. $\lceil 5.29 \rceil = 6$). This upper-bound is lower than 2.*N*/log(*N*) for any value of *N*. The method described in this paper is actually a variant of Pinch's method: instead of splitting the binary representation into blocks of fixed weight, it is split into blocks of fixed lengths (*r*).

The paper is organized as follows. Section I outlines the need of addition-cost complexity for large constant bit-widths. Section II introduces the radix-2^r recoding for multiplication by an *N*-bit constant, while Section III determines its upper-bound in number of additions and compares the results to existing heuristics. Section IV presents an illustrative example. Finally, Section V gives some concluding remarks and suggestions for future work.

II. Radix-2^r for Multiplication by an *N*-bit Constant

A non-negative *N*-bit constant *C* is expressed in radix-2^r as:

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

where $c_{-1} = c_N = 0$ and $r \in \mathrm{N}^*$. For simplicity purposes and without loss of generality, we assume that *r* is a divider of *N*+1. In eq. (1), the two's complement representation of the

constant C is split into $(N+1)/r$ two's complement slices (Q_j), each of r bit length because it goes from 2^0 to 2^{r-1}. However, Q_j needs an additional bit (c_{rj-1}) equal to the most significant bit of the previous digit (Q_{j-1}), which could be seen as some form of carry due to the use of signed digits; it comes from the following formula: $-2^{r-1}c_{rj+r-1}\times 2^{rj}+c_{rj+r-1}\times 2^{r(j+1)}=c_{rj+r-1}\times 2^{rj+r-1}$. This formula expresses the transformation of the conventional radix-2^r representation to the signed-digit radix-2^r one.

A digit-set $DS(2^r)$ corresponds to eq. (1), such as

$$Q_j \in DS(2^r)=\{-2^{r-1},-2^{r-1}+1,\ldots,-1,0,1,\ldots,2^{r-1}-1,2^{r-1}\}.$$

Thus, the product becomes: $$C\times X=\sum_{j=0}^{(N+1)/r-1} X\times Q_j\times 2^{rj}. \quad (2)$$

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 $m_j\in OM(2^r)\cup\{0\}$, where $OM(2^r)=\{1,3,5,\ldots,2^{r-1}-1\}$. $OM(2^r)$ is the set of odd positive digits in radix-2^r recoding, with $|OM(2^r)|=2^{r-2}$. To $|Q_j|=0$ corresponds $m_j=0$. Finally, the product can be expressed as follows: $$C\times X=\sum_{j=0}^{(N+1)/r-1}(-1)^{c_{rj+r-1}}\times(m_j\times X)\times 2^{rj+k_j}. \quad (3)$$

Unlike the multiplication by a variable ($Y\times X$) where the entire set of partial-products ($m_j\times X$) must be precomputed, only a subset is needed in the multiplication by a constant ($C\times X$). In fact, the number of partial-products is equal to the number of different values m_j induced by the encoding process of the $(N+1)/r$ slices (terms Q_j). Therefore, the generation of partial products (PP) consists first, if $m_j\neq 0$, in computing the PP $m_j\times X$ *if it has not been precomputed before*. It is then submitted to a hardwired left-shift of $rj+k_j$ positions, and finally, conditionally negated $(-1)^{c_{rj+r-1}}$ depending on the sign bit c_{rj+r-1} of Q_j. An illustrative example is given in Section IV.

III. Maximum Number of Additions for an N-bit Constant

On the one hand, there are $(N+1)/r$ iterations in eq. (3). Each iteration generates one PP. Thus, the maximal number of PP is $(N+1)/r$, which requires a maximum of $N_{pp}=(N+1)/r-1$ additions. On the other hand, a maximum of $2^{r-2}-1$ non-trivial PP $\{3\times X,\ 5\times X,\ 7\times X,\ \ldots,\ (2^{r-1}-1)\times X\}$ can be invoked during the PP generation process. They are built using the binary method, from the least significant bit to the most significant bit. That is, the m_j elements 3, 5, 7, ..., $2^{r-1}-1$ are built one after the other, each time by using a single addition between an element that has already been built and a power of two. This process is summarized by the following recurrence relation: $m_j=2^p+d$, where $p\leq r-2$ because $m_j\leq 2^{r-1}-1$, and $0<d<2^p$.

Theorem 1. *In radix-2^r, the precomputation of the entire set of non-trivial PP $\{3\times X,\ 5\times X,\ 7\times X,\ldots,(2^{r-1}-1)\times X\}$ yields an adder-cost and an adder-depth of $2^{r-2}-1$ and $r-2$, respectively.*

Proof. Since each new non-trivial digit requires only one addition (recurrence relation), the adder-cost is the number of non-trivial digits: $N_{om}=|OM(2^r)|-1=2^{r-2}-1$.

As the binary method is used, the adder-depth is deduced from the maximum number of non-zero bits in the binary representation of a digit: $(r-1)-1=r-2$. Since there are $(N+1)/r$ PP, the maximum adder-depth (Ath) in cascaded adders is:

$$\mathrm{Ath}(r)=\left\lceil\frac{N+1}{r}-1+r-2\right\rceil=\left\lceil\frac{N+1}{r}+r-3\right\rceil.$$

We illustrate the construction process of non-trivial PP with the following radix-2^6 example:

$$\begin{aligned}OM(2^6)&=\{1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31\}\\&=\{1\}\cup\{2^1+1=3\}\cup\{2^2+1=5,2^2+3=7\}\cup\{2^3+1=9,2^3+3=11,\\&\quad 2^3+5=13,2^3+7=15\}\cup\{2^4+1=17,2^4+3=19,2^4+5=21,\\&\quad 2^4+7=23,2^4+9=25,2^4+11=27,2^4+13=29,2^4+15=31\}.\end{aligned}$$

Thus, the PP ($m_j\times X$) corresponding to $OM(2^6)$ are subsequently calculated in the following order (6–2=4 steps):

$\{3\times X\}$; $\{5\times X, 7\times X\}$; $\{9\times X, 11\times X, 13\times X, 15\times X\}$;
$\{17\times X, 19\times X, 21\times X, 23\times X, 25\times X, 27\times X, 29\times X, 31\times X\}$.

Fig.1 provides all necessary details for hardware implementation. It now becomes clear that eq. (3) involves only *additions*, *subtractions*, and *left-shifts*. Note that right-shifts are not allowed since r, j, and k_j are non-negative integers.

Consequently, the total number of additions required by radix-2^r is equal to:

$$\mathrm{Upb}(r)=N_{pp}+N_{om}=\left\lceil\frac{N+1}{r}+2^{r-2}-2\right\rceil.$$

Upb(r) is minimal for $r=2\cdot \mathrm{W}\left(\sqrt{(N+1)\cdot\log(2)}\right)/\log(2)$, where W is the Lambert Function. The minimum is obtained for one of the two enclosing integers of r (since the upper limit is a convex function of r), and both must be tested. Table I gives the values of r that lead to the minimum number of additions for N ranging from 8 to 8192. It also gives the corresponding adder-depths. Fig.2 depicts the upper-bounds in number of additions for CSD, DBNS, and RADIX-2^r.

TABLE I
UPPER-BOUND (UPB), ADDER-DEPTH (ATH), AND r VALUES FOR A NON-NEGATIVE N-BIT CONSTANT USING RADIX-2^r

N	8	16	32	64	128	256	512	1024	2048	4096	8192
r	3	3	4	5	5	6	6	7	8	8	9
Upb(r)	3	6	11	19	32	57	100	177	319	575	1037
Ath(r)	3	6	10	15	28	46	89	151	262	518	917

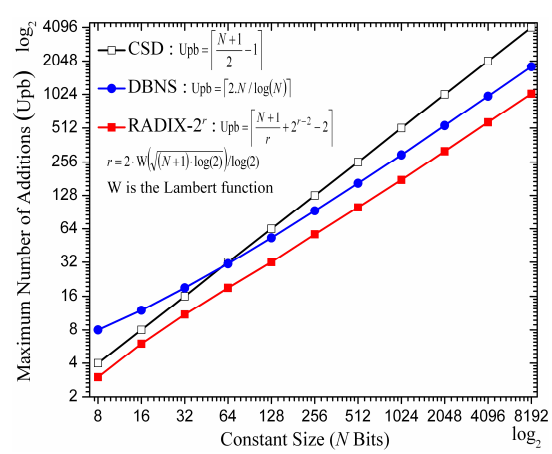


Fig. 2. Upb comparison for an N-bit constant.

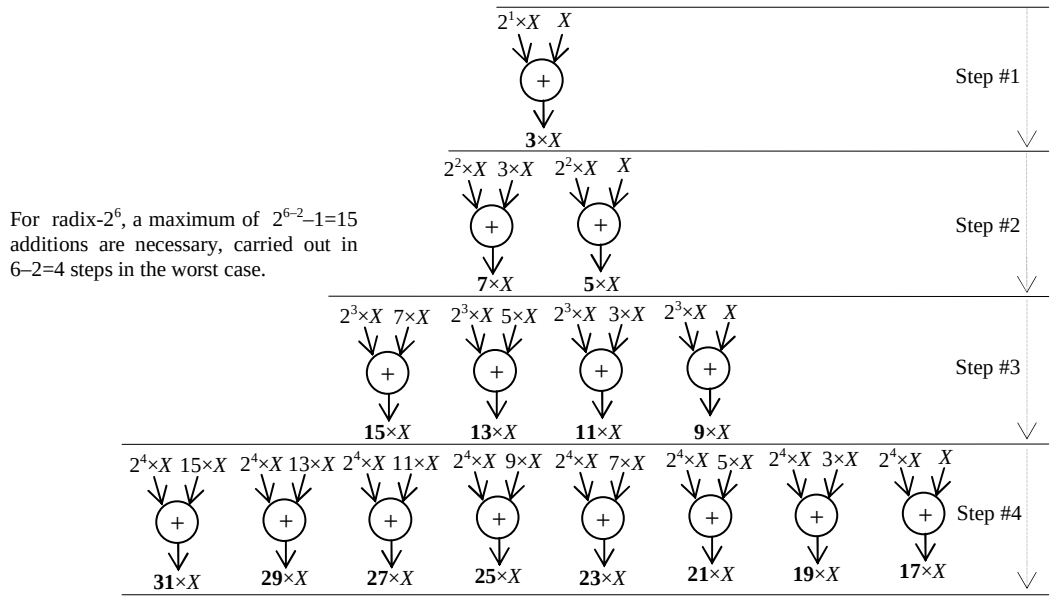


Fig. 1. Sequential order of computation of the entire set of partial-products needed by radix-2^6.

As for the average number of additions (Avg), it has been *exhaustively* calculated for values of C varying from 0 to 2^N–1, for N=8, 16, 24, and 32. But for N=64, we have calculated Avg using 10^5, 10^6, 10^9 and 10^{10} *uniformly distributed random values* of C. While the difference between the four obtained results is insignificant ($<10^{-3}$), the value Avg oscillates around 15.7165 additions. Results are reported in Table II. For N=64, RADIX-2^r uses 23.12 % less additions than CSD. This gain seems to grow linearly for low values of N.

TABLE II
RADIX-2^r VERSUS CSD: AVERAGE NUMBER OF ADDITIONS (Avg) AND UPPER-BOUND (Upb)

Constant Bit-width N	CSD		RADIX-2^r		Saving (Avg,%)
	Avg	Upb	Avg	Upb	
8	1.7882	4	1.8645	3	-4.2668^+
16	4.4445	8	4.5127	6	-1.5344^+
24	7.1111	12	6.7994	9	4.3832
32	9.7777	16	8.9627	11	8.3352
64	20.4444	32	15.7165*	19	23.1256

*: Obtained from 10^{10} uniformly distributed random values of C.
+: RADIX-2^r average is higher than CSD's.
CSD Avg = (N/3)–8/9 and CSD Upb=$\lceil (N+1)/2-1 \rceil$.

Regarding DBNS, Dimitrov [3] calculated Avg and Upb from 10^5 uniformly distributed *random* constants, for 32 and 64 bits only (Table III). Note that DBNS Upb will be higher if the worst cases are not attained by the pattern of 10^5 constants.

We have also compared RADIX-2^r to some non-recoding heuristics (CSE and DAG) based on programs and numeric data kindly provided by Lefèvre and Voronenko. While Fig. 3 shows lower values of Avg for non-recoding heuristics as expected due to a larger exploration of the solution space, Table IV exhibits rather a *higher* value of Upb for Bernstein's heuristic. *Significant conclusion*: a lower Avg does not guarantee a lower Upb.

TABLE III
RADIX-2^r VERSUS DBNS : AVERAGE NUMBER OF ADDITIONS (Avg) AND UPPER-BOUND (Upb)

Constant Bit-width N	DBNS [3]		RADIX-2^r		Saving (Avg,%)
	Avg	Upb	Avg	Upb	
32	$\approx 9.05^+$*	13*	8.9627	11	0.9646
64	16.2151*	21*	15.7165	19	3.0749

+: Taken from Fig.1 in [3]; *: Obtained from 10^5 uniformly distributed random values of C.

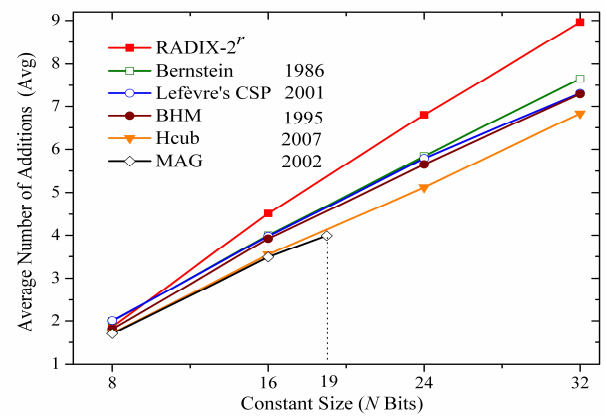


Fig. 3. Avg comparison for an N-bit constant.

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. Table V summarizes this information for a 32-bit constant. Note that starting from q=7, higher values are given by RADIX-2^r compared to CSD.

IV. ILLUSTRATIVE EXAMPLE

The product 10599×X is first calculated in CSD, DBNS, and RADIX-2^r. Let us note that $(10599)_{10}=(10100101100111)_2$.
$P_{CSD}=(X\times 2^{13})+(X\times 2^{11})+(X\times 2^{9})-(X\times 2^{7})-(X\times 2^{5})+(X\times 2^{3})-X$.
$P_{DBNS}=((X_1\times 2^1)+X_1)+(X\times 2^{13})+(X\times 2^3)-X$,
with $X_1=((X_0\times 2^1)+X_0)+(X\times 2^5)$ and $X_0=(X\times 2^8)$ [3].

In order to express the product in P_{RADIX}, a two's complement representation of $(10599)_{10}$ is necessary, which is $(010100101100111)_2$. Thus, in two's complement notation, the constant size becomes N+1 (14+1=15 for 10599).

TABLE IV
RADIX-2^r VERSUS NON-RECODING ALGORITHMS: RUNTIME COMPLEXITY AND NUMBER OF ADDITIONS OF SOME SPECIAL CASES

Algorithm	$(84AB5)_H$ N=20$^+$	$(64AB55)_H$ N=23$^+$	$(5959595B)_H$ N=31$^+$	Runtime [10]
BIGE [11]	<u>4</u>	<u>5</u>	<u>6</u>	$O(2^N)$
Bernstein [7]	8^G	7	8	$O(2^N)$ [5]
Hcub* [10]	<u>4</u>	6	–	$O(N^6)$
BHM* [19]	5	7	–	$O(N^4)$
Lefèvre's CSP [5]	<u>4</u>	6	9	$O(N^3)$
RADIX-2^r	5	7	10	$O(N)$

N: Constant bit-size; +: In RADIX-2^r, a zero bit is added in the MSB position to ensure a non-negative value of the constant in the recoding.
G: Greater than RADIX-2^r Upb; RADIX-2^r Upb=7, 8, and 10, for N=20, 23, and 31, respectively; *: Values are delivered by Spiral web version [20], limited to 26 bits; <u>X</u>: Optimal number of additions.

The BIGE optimal solutions for the indicated values are obtained as follows:
$(84AB5)_H$: 15 = (2^4)–1 ; 3825 = (15×2^8)–15 ; 19125 = (3825×2^2)+3825; 543413 = (2^{19})+19125.
$(64AB55)_H$: 255 = (2^8)–1; 65281 = (255×2^8)+1; 1109777 = (65281×2^4)+65281; 5548885 = (1109777×2^2)+1109777; 6597461 = (2^{20})+5548885.
$(5959595B)_H$: 257= (2^8)+1; 16843009= (257×2^{16})+257; 16843011= (2)+16843009; 50529027 = (16843009×2)+16843009; 421075227 = (50529027×2^3) +16843011; 1499027803 = (16843009×2^6)+421075227.

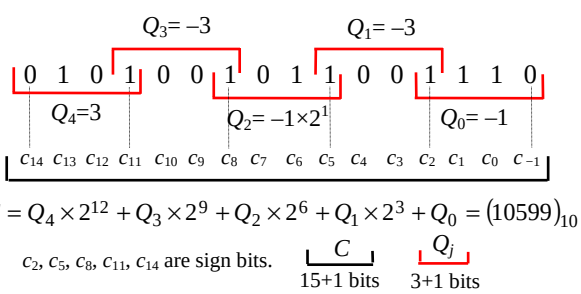
To N=14 corresponds r=3 (see Upb formula). For C=10599, eq. (1) and (3) become respectively:

$$C=\sum_{j=0}^{4} Q_j\times 2^{3j} \quad \text{, and} \quad P_{RADIX}=\sum_{j=0}^{4}(-1)^{c_{3j+2}}\times\left(m_j\times X\right)\times 2^{3j+k_j}\ .$$

Fig.4 depicts the five terms Q_j. To determine the unknown values c_{3j+2}, m_j, and k_j, the radix-2^3 look-up table (Table VI) is indexed by the terms Q_j. Referring to Table VI, the triplets (c_{3j+2}, m_j, k_j) corresponding to Q_0, Q_1, Q_2, Q_3, and Q_4 are (1,1,0), (1,3,0), (1,1,1), (1,3,0), and (0,3,0), respectively. The recoding of C=10599 involves the precomputation of the PP $3\times X$. Consequently, we can write:

$$P_{RADIX}=(3\times X)\times2^{12}-(3\times X)\times2^9-(1\times X)\times2^7-(3\times X)\times2^3-(1\times X)$$
$$=(X_0\times2^{12})-(X_0\times2^9)-(X\times2^7)-(X_0\times2^3)-X,$$

with $X_0=(X\times2)+X$.



$$C=Q_4\times2^{12}+Q_3\times2^9+Q_2\times2^6+Q_1\times2^3+Q_0=(10599)_{10}$$

c_2, c_5, c_8, c_{11}, c_{14} are sign bits. $\underbrace{C}_{15+1\text{ bits}}$ $\underbrace{Q_j}_{3+1\text{ bits}}$

Fig. 4. Partitioning of $(10599)_{10}$ in radix-2^3.

It has to be noted that for C=10599, P_{CSD} and P_{DBNS} require both 6 additions, while P_{RADIX} requires 5. The naive shift-and-add approach would have required 7 additions. We assume that addition and subtraction have the same area/speed cost, and that shift is costless since it can be realized without any gates, i.e. just by using hard wiring.

Simplifications in eq. (3) are possible in case two consecutive terms Q_j and Q_{j+1} with opposite signs exhibit pairs (m_j , k_j) of the form (1, r–1) and (1, 0), respectively. This is illustrated by the two following possibilities:

TABLE V
RADIX-2^r VERSUS CSD, LEFEVRE'S CSP, AND EXHAUSTIVE SEARCH: SMALLEST VALUES UP TO A 32-BIT CONSTANT

Number of Additions (q)	CSD	RADIX-2^r	Lefèvre's CSP* [5]	Exhaustive search [5]
1	3	3	3	3
2	11	11	11	11
3	43	43	43	43
4	171	139	213	683
5	683	651	1703	14709
6	2731	2699	13623	699829
7	10923	33419	174903	171398453$^+$
8	43691	526491	1420471	–
9	174763	8422027	13479381	–
10	699051	134744219	–	–
11	2796203	2155905675	–	–
12	11184811	–	–	–
13	44739243	–	–	–
14	178956971	–	–	–
15	715827883	–	–	–

*: Lefèvre calculated the values for q up to 9. This means that the common subpattern algorithm (CSP) exhibits an Upb ≥ 9 among all 32-bit constants.
+: This is the sole value which has not been confirmed by Lefèvre's exhaustive algorithm. It has been found only by Donelly [5], using left-shifts exclusively. If "right-shifts" are allowed, the value is strictly higher since the BIGE solution using right-shifts gives 6 additions, as follows: 5 = (2^2)+1; 639 = (5×2^7)–1; 317 = $(639–5)\times2^{-1}$; 5194045 = (317×2^{14})+317; 171393341 = (317×2^{19})+5194045; 171398453 = (639×2^3)+171393341.
Thong [11] *conjectured* that 7 additions are enough up to 32 bits, *allowing* right-shifts (exhaustive BIGE algorithm). It has been *proved* via RADIX-2^r *heuristic* that 11 additions are sufficient up to 32 bits, using left-shifts only.

$$\cdots+X\times2^{r(j+1)}-X\times2^{rj+(r-1)}\pm\cdots=\cdots+X\times2^{rj+(r-1)}\pm\cdots$$
$$\cdots-X\times2^{r(j+1)}+X\times2^{rj+(r-1)}\pm\cdots=\cdots-X\times2^{rj+(r-1)}\pm\cdots$$

Another interesting idea is to include *redundancy* in the terms Q_j of eq. (1). These two tricks will decrease the average number of additions in RADIX-2^r (Table II, III, and Fig. 3).

In addition to higher compression capabilities of RADIX-2^r compared to CSD and DBNS, its runtime complexity is *linearly* proportional to N as shown by eq. (1). Moreover the required memory space is *very small* (for a 8192-bit constant corresponds a look-up table of 2^{9+1}=1024 entries). These two features make RADIX-2^r very useful for huge constants.

TABLE VI
RADIX-2^3 LOOK-UP TABLE

Q_j				m_j	k_j
c_{3j+2}	c_{3j+1}	c_{3j}	c_{3j-1}		
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	1	0
0	0	1	1	1	1
0	1	0	0	1	1
0	1	0	1	3	0
0	1	1	0	3	0
0	1	1	1	1	2
1	0	0	0	1	2
1	0	0	1	3	0
1	0	1	0	3	0
1	0	1	1	1	1
1	1	0	0	1	1
1	1	0	1	1	0
1	1	1	0	1	0
1	1	1	1	0	0

Note that for radix-2^3,
$k_j\in\{0,1,2\}$ and $m_j\in\{0,1,3\}$.

Since the introduction of H(k) [9] in 2004, CSE heuristics have outperformed DAGs at SCM [11]. This was achieved by applying CSE to each possible signed-digit (SD) form of the constant. Likewise, the search space of CSE can be expanded considering RADIX-2^r recoding instead of SD representation. For such a goal (SCM/MCM), Lefèvre's CSP heuristic [5] stands as the best CSE candidate for its lower computational complexity $O(N^3)$ in comparison to its CSE counterparts [10].

Many conversion techniques from unsigned or two's complement number to its CSD form are proposed to reduce the hardware complexity and increase the speed of variable multipliers [21]. Based on RADIX-2^r, we proposed several conversion techniques and determined the most efficient one. For more details on our extensive work on RADIX-2^r multiplication problem, reader is referred to [22] [23] [24].

V. Conclusion and Future Work

Based on radix-2^r arithmetic, we have developed a new linear-time recoding (RADIX-2^r) accompanied with its upper-bound complexity. The latter is the lowest upper-bound known so far for the multiplication by a constant. While the bound is for a minimal set of operations (additions, subtractions, and left-shifts), it remains valid if any other operation (such as right-shifts) is allowed.

Not only RADIX-2^r achieves better compression ratio than DBNS and CSD, which yields more speed and less area and power consumption, but also stands as a practical alternative to non-recoding heuristics for large constant bit-widths. Further improvements of RADIX-2^r are possible using redundancy in the recoding.

Our current work deals with exact analytic expressions of the average number of additions as well as the minimal adder-depth of RADIX-2^r, which are still to be determined.

Acknowledgment

This work is supported by "Centre de Développement des Technologies Avancées", CDTA, Algiers, Algeria, under project contract number: 21/CRSOC/DMN/CDTA/2011. The project progresses under a close cooperation with FEMTO-ST institute, Besançon, France.

Authors wish to thank M.L. Berrandjia and F. Ykhlef for their technical help, and B. Djezzar for his careful review of this manuscript.

References

[1] 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.

[2] 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.

[3] V.S. Dimitrov, L. Imbert, and A. Zakaluzny, "Multiplication by a Constant is Sublinear," Proceedings of the 18th IEEE Symposium on Computer Arithmetic (ARITH), pp. 261-268, June 2007.

[4] 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.

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

[6] 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.

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

[8] 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.

[9] 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.

[10] 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.

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

[12] O. Gustafsson, "Lower Bounds for Constant Multiplication Problems," IEEE Trans. on Circuits and Systems II: Express Brief, vol. 54, No. 11, pp. 974-978, November 2007.

[13] R. W. Reitwiesner, "Binary Arithmetic," Advances in Computers, New York: Academic, vol. I , pp. 231-308, 1966.

[14] R. G. E. Pinch, "Asymptotic Upper Bound for Multiplier Design," Electronics Letters, vol. 32, N° 5, pp. 420-421, February 1996.

[15] V.S. Dimitrov, G.A. Jullien, and W.C. Miller, "Theory and Applications of the Double-Base Number System," IEEE Trans. on Computers (TC), vol. 48, No. 10, pp. 1098-1106, October 1999.

[16] V.S. Dimitrov, K.U. Järvinen, and J. adikari, "Area Efficient Multipliers Based on Multiple-Radix Representations," IEEE Trans. on Computers (TC), vol. 60, N° 2, pp 189-201, February 2011.

[17] S. Homayoon and A. Gupta, "A Generalized Multibit Recoding of Two's Complement Binary Numbers and its Proof with Application in Multiplier Implementation," IEEE Trans. on Computers (TC), vol. 39, N° 8, August 1990.

[18] P.M. Seidel, L. D. McFearin, and D.W. Matula, "Secondary Radix Recodings for Higher Radix Multipliers," IEEE Trans. on Computers (TC), vol. 54, N°2, February 2005.

[19] A.G. Dempster and M.D. Macleod, "Use of Minimum Adder Multiplier Blocks in FIR Digital Filters," IEEE Trans. on Circuits and Systems-II: Analog and Digital Signal Processing 42, 9, pp. 569-567, 1995.

[20] Available at: http://spiral.ece.cmu.edu/mcm/gen.html

[21] R. Guo and L.S. DeBruner, "A Novel Fast Canonical-Signed-Digit Conversion for Multiplication," Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1637-1640, Prague, Czech Republic, May 22-27 2011.

[22] A.K. Oudjida, N. Chaillet, M.L. Berrandjia, and A. Liacha, "A New High Radix-2^r ($r \geq 8$) Multibit Recoding Algorithm for Large Operand Size ($N \geq 32$) Multipliers," Journal of Low Power Electronics (JOLPE), vol. 9, N° 1, pp. 50-62, ISSN: 1546-1998/2013/9/50/62, American Scientific Publishers (ASP), April 2013.

[23] A.K. Oudjida, N. Chaillet, A. Liacha, and M.L. Berrandjia, "A New Recursive Multibit Recoding Algorithm for High-Speed and Low-Power Multiplier, " Journal of Low Power Electronics (JOLPE), vol. 8, N° 5, pp. 579-594, ISSN: 1546-1998/2012/8/579/594, American Scientific Publishers (ASP), December 2012.

[24] A.K. Oudjida, N. Chaillet, A. Liacha, and M.L. Berrandjia "New High-Speed and Low-Power Radix-2^r Multiplication Algorithms," Proceedings of the 11th edition of IEEE-FTFC Low-Voltage Low-Power Conference, ISSN: 978-1-4673-0821-2/12, Paris, June 2012.