Multiple Constant Multiplication Algorithm for High Speed and Low Power Design

Abdelkrim K. Oudjida, Ahmed Liacha, Mohammed Bakiri, and Nicolas Chaillet

***Abstract*—In this paper, radix-2^r arithmetic is applied to the multiple constant multiplication (MCM) problem. Given a number *M* of nonnegative constants with a bit-length *N*, we determine the analytic formulas for the maximum number of additions, the average number of additions, and the maximum number of cascaded additions forming the critical path. We get the first proved bounds known so far for MCM. In addition of being fully-predictable with respect to the problem size (*M*, *N*), the RADIX-2^r MCM heuristic exhibits a sublinear runtime-complexity $O(M\times N/r)$, where *r* is a function of (*M*, *N*). For high-complexity problems, it is most likely the only one that is even feasible to run. Another merit is that it has the shortest adder-depth in comparison to the best published MCM algorithms.**

***Index Terms*— 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

MCM is an arithmetic operation that multiplies a set of fixed-point constants $\{C_0, C_1, C_2, \cdots, C_{M-1}\}$ with the same fixed-point variable *X*. This operation dominates the complexity of many numeric systems such as FIR/IIR filters, DSP transforms (DCT, DFT, Walsh, …), LTI controllers, crypto-systems, etc. To be efficiently implemented, i.e., rapid, compact, and low-power, MCM must avoid costly multipliers. The hardware alternative will be multiplierless, i.e., using only additions, subtractions, and left-shifts. We assume that addition and subtraction have the same area/speed cost, and that the shift is costless since it can be realized without any gates, i.e., just by using hardwiring. Therefore, the MCM problem is defined as the process of finding the minimum number of addition/subtraction operations. The computational complexity of MCM is conjectured to be NP-hard [1]. But because the solution-space to explore is so huge, optimal solutions require excessive runtime and become impractical even for MCM operations of a medium complexity [1][2]. Only MCM heuristics can react in a reasonable amount of time, producing however, suboptimal solutions.

Using the radix-2^r arithmetic, we developed in a previous work [3][4] a fully-predictable heuristic for SCM, denoted RADIX-2^r SCM. We obtained the lowest analytic bounds known so far for SCM in adder-cost (*Upb*), average (*Avg*), and adder-depth (*Ath*). Compared to the standard Canonical-Signed-Digit (CSD) representation [5] in the case of a serial implementation (adders connected in series), for an *N*-bit constant a saving of 50% is attained at *N*=1134, *N*=128, and *N*=64, in *Avg*, *Upb*, and *Ath*, respectively. The savings keep increasing as *N* is getting larger.

A.K. Oudjida (a_oudjida@cdta.dz), A. Liacha, and M. Bakiri are with "Centre de Développement des Technologies Avancées", CDTA, Cité du 20 août 1956, Baba-Hassen, Algiers, 16303, Algeria.
N. Chaillet (nicolas.chaillet@femto-st.fr) and M. Bakiri are with FEMTO-ST Institute, UFC/CNRS/ENSMM/UTBM, 32 avenue de l'Observatoire, 25044 Besançon, Cedex, France.

In addition, RADIX-2^r SCM shows a sublinear runtime complexity with respect to *N*, and the memory space required is very small; for *N*=8192 corresponds a look-up table of 1024 entries only. These two features makes RADIX-2^r SCM very useful for huge constants, given that the lowest runtime complexity of non-digit-recoding algorithms (Bernstein [6], Lefèvre [7], BHM [8], Hcub [9], and MAG [10]) is $O(N^3)$ [3]. A summary of the main features of RADIX-2^r SCM is given in Table I.

The main idea of RADIX-2^r SCM is that the base number system (2^r) is properly adapted to the bit-length (*N*) of the constant to achieve, either an *optimal* adder-cost (r_1) [3] or a *lower* adder-depth (r_2) [4]. To decide which expression of *r* (r_1 or r_2) to choose depends actually on the design requirements. If area is targeted, r_1 is used. But in case speed or power is a concern, r_2 is suitable. Note that intermediate values of *r* ($r_1<r<r_2$) lead to a tradeoff between area and speed/power.

The main purpose of this work is to first apply the radix-2^r arithmetic to the MCM problem and derive the analytic expressions for *Upb*, *Ath*, and *Avg* in the same way we did for SCM [3][4]. As a second step, we look at an actual circuit implementation through the application of RADIX-2^r MCM to the design optimization of a benchmark FIR filter.

This paper is organized as follows. Section I gives an overview on RADIX-2^r SCM. In Section II we define RADIX-2^r MCM and determine the respective metrics. RADIX-2^r MCM is confronted in Section III to some of the best published MCM heuristics. Finally, Section IV provides some concluding remarks and suggestions for future work.

II. Radix-2^r MCM

A nonnegative *N*-bit constant *C* is expressed in radix-2^r as

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

where $c_{-1} = c_N = 0$ and $r \in \mathrm{N}^*$. In (1), the two's complement representation of *C* is split into $\lceil (N+1)/r \rceil$ slices (Q_j), each of r+1 bit length (see Fig.1.a). Each pair of two contiguous 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 $|OM(2^r)| = 2^{r-2} - 1$.

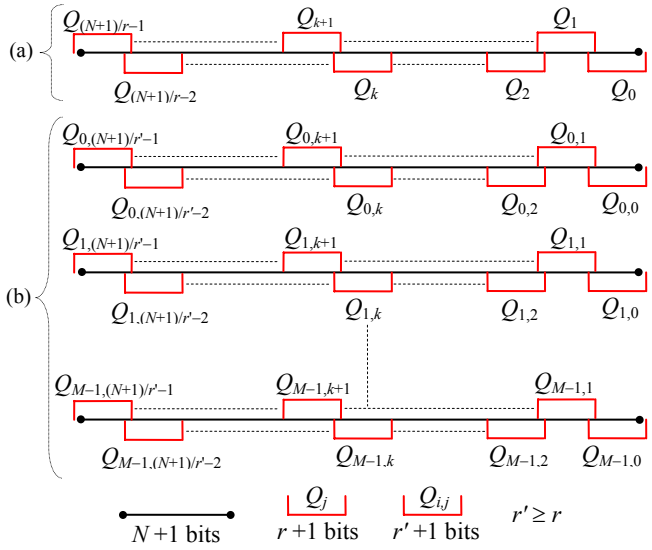

Fig. 1. Slice partitioning of N-bit constants in radix-2^r.
(a) RADIX-2^r SCM, (b) RADIX-2^r MCM.

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

Metrics	Equations
Adder cost	$Upb(r)=\left\lceil \frac{N+1}{r} \right\rceil + 2^{r-2} - 2$ with $r=r_1$ or $r=r_2$
Adder depth	$Ath(r)=\left\lceil \frac{N+1}{r} \right\rceil + r - 3$ with $r=r_1$ or $r=r_2$
Average	$-1+Avg_{pp}+Avg_{om} \leq Avg(r) \leq -2+Avg_{pp}+2^{r-2}$ with $Avg_{pp}=(1-2^{-r})\times\left\lceil \frac{N+1}{r} \right\rceil$, $Avg_{om}=\sum_{j=0}^{\left\lceil \frac{N+1}{r} \right\rceil-1}\left\{\sum_{k=1}^{2^{r-2}-1}P(m_{jk})\times[1-P(m_{jk})]^j\right\}$, $P(m_{jk})=\frac{\log_2\left\lceil \frac{2^{r-1}}{2\times k+1} \right\rceil}{2^{r-1}}$, and $r=r_1$ or $r=r_2$
$r_1=2\cdot W\left[\sqrt{(N+1)\cdot\log(2)}\right]/\log(2)$	$r_2=W[4.(N+1).\log(2)]/\log(2)$

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

RADIX-2^r SCM can be easily extended to MCM. In MCM a single variable X is simultaneously multiplied by a set of M constants $\{C_0, C_1, C_2, \cdots, C_{M-1}\}$, having all the same bit-size N. In RADIX-2^r MCM, each constant C_i is split into $\lceil (N+1)/r' \rceil$ slices (Q_{ij}), each slice of a bit-length r'+1 (see Fig. 1.b). Thus, the maximal number of partial products (PP) is $M\times\lceil (N+1)/r' \rceil$, plus a maximum of $2^{r'-2}-1$ nontrivial PP $\{3\times X,\ 5\times X, 7\times X, \ldots, (2^{r'-1}-1)\times X\}$ that can be invoked during the PP generation process. Aided by Fig. 1.b and using the same reasoning based on Theorem (1) in [3], we can easily demonstrate that the maximum number of additions (Upb) in RADIX-2^r MCM is

$$Upb(r') = M\times\lceil (N+1)/r' \rceil + 2^{r'-2} - 1 - M \ .$$

$Upb(r')$ is minimal for $r' = 2\cdot W\left(\sqrt{M\cdot(N+1)\cdot\log(2)}\right)/\log(2)$.

Note that $r' \geq r$ (Fig. 1), due to the product $M\times(N+1)$. In fact RADIX-2^r SCM is a particular case of RADIX-2^r MCM for M=1. Pursuing also the same reasoning developed in [4], we can straightforwardly derive the analytic expressions of Avg and Ath (see Table II). But in real-life applications of MCM, such as for instance in the transposed form of a FIR filter, the coefficients will most likely have different bit-sizes. Assume that for each constant C_i corresponds a bit-size N_i, the total number of PP for a set of M constants will be equal to $\sum_{i=0}^{M-1}\lceil (N_i+1)/r' \rceil$. Likewise, taking this fact into account, we can easily prove the analytic equations of Table III.

TABLE II
RADIX-2^r MCM FOR A NUMBER OF M NONNEGATIVE CONSTANTS WITH THE SAME BIT-SIZE N

Metrics	Equations
Adder cost	$Upb(r') = M\times\left\lceil \frac{N+1}{r'} \right\rceil + 2^{r'-2} - 1 - M$ with $r'=r_1$ or $r'=r_2$
Adder depth	$Ath(r')=\left\lceil \frac{N+1}{r'} \right\rceil + r' - 3$ with $r'=r_1$ or $r'=r_2$
Average	$-M+Avg_{pp}+Avg_{om} \leq Avg(r') \leq -M-1+Avg_{pp}+2^{r'-2}$ with $Avg_{pp}=(1-2^{-r'})\times M\times\left\lceil \frac{N+1}{r'} \right\rceil$, $Avg_{om}=\sum_{j=0}^{M\times\left\lceil \frac{N+1}{r'} \right\rceil-1}\left\{\sum_{k=1}^{2^{r'-2}-1}P(m_{jk})\times[1-P(m_{jk})]^j\right\}$, $P(m_{jk})=\frac{\log_2\left\lceil \frac{2^{r'-1}}{2\times k+1} \right\rceil}{2^{r'-1}}$, and $r'=r_1$ or $r'=r_2$
$r_1=2\cdot W\left[\sqrt{M\cdot(N+1)\cdot\log(2)}\right]/\log(2)$	$r_2=W[4.M\cdot(N+1).\log(2)]/\log(2)$

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

Unlike existing MCM algorithms, in RADIX-2^r MCM each constant is implemented apart, independently from the others. But all constants share the same set of nontrivial PP. This is illustrated by the following MCM example: C_0=(84AB5)$_H$, C_1=(64AB55)$_H$, C_2=(5959595B)$_H$. To the constants C_0, C_1, and C_2 corresponds the bit-sizes N_0=20, N_1=23, and N_2=31, respectively. Thus, for $\sum_{i=0}^{2}(N_i+1)=77$ the $r'=r_1$ formula in Table III gives r'=4, which is the value that minimizes Upb.

Hence, the solution given by the online version [11] of RADIX-2^r MCM is

$C_0\times X = X\times 2^{19}+X_1\times 2^{12}-X_1\times 2^8-X_1\times 2^4+X_1$,
$C_1\times X = X_0\times 2^{21}+X_1\times 2^{16}-X_1\times 2^{12}-X_1\times 2^8+X_1\times 2^4+X_1$,
$C_2\times X = X_0\times 2^{29}-X_2\times 2^{24}+X_0\times 2^{21}-X_2\times 2^{16}+X_0\times 2^{13}-X_2\times 2^8+X_0\times 2^5-X_1$
with $X_0=3\times X=X\times 2+X$, $X_1=5\times X=X\times 2^2+X$, $X_2=7\times X= X\times 2^3-X$.

Note that the online version offers three solutions. The one given above corresponds to the optimization of adder-depth.

In [4], we introduced a variant of RADIX-2^r called R3. It has a better Avg with the same Upb and Ath. R3 MCM gives

$C_0\times X = X\times 2^{19}+U_{75}\times 2^8-U_{75}$, $U_{75}= X_1\times 2^4-X_1$,
$C_1\times X = U_{101}\times 2^{16}-U_{85}\times 2^8+U_{85}$, $U_{101}=X_0\times 2^5-X_1$, $U_{85}=X_1\times 2^4+X_1$,
$C_2\times X = U_{89}\times 2^{24}+U_{89}\times 2^{16}+U_{89}\times 2^8+U_{91}$, $U_{89}=X_0\times 2^5-X_2$,
$U_{91}=X_0\times 2^5-X_1$.

These two solutions are compared in Table IV to the ones provided by the most efficient MCM algorithms. Note that CSD, RADIX-2^r, and R3 MCM, as digit-recoding algorithms, allow both serial and parallel implementations, while Hcub, BHM and Lefèvre's CSP are limited to serial implementation only due to the shared terms. This issue will be detailed further in the next section.

TABLE III
RADIX-2^r MCM FOR A NUMBER OF M NONNEGATIVE CONSTANTS WITH DIFFERENT BIT-SIZES N_i

Metrics	Equations
Adder cost	$Upb(r')=\sum_{i=0}^{M-1}\left\lceil\frac{N_i+1}{r'}\right\rceil+2^{r'-2}-1-M$ with $r'=r_1$ or $r'=r_2$
Adder depth	$Ath(r')=\left\lceil\frac{\max(N_i)+1}{r'}\right\rceil+r'-3$ with $i=0..M-1$, $r'=r_1$ or $r'=r_2$
Average	$-M+Avg_{pp}+Avg_{om}\le Avg(r')\le -M-1+Avg_{pp}+2^{r'-2}$ with $Avg_{pp}=(1-2^{-r'})\times\sum_{i=0}^{M-1}\left\lceil\frac{N_i+1}{r'}\right\rceil$, $Avg_{om}=\sum_{j=0}^{\sum_{i=0}^{M-1}\lceil\frac{N_i+1}{r'}\rceil-1}\left\{\sum_{k=1}^{2^{r'-2}-1}P(m_{jk})\times\left[1-P(m_{jk})\right]^j\right\}$, $P(m_{jk})=\frac{\log_2\left\lceil\frac{2^{r'-1}}{2\times k+1}\right\rceil}{2^{r'-1}}$, and $r'=r_1$ or $r'=r_2$
$r_1=2\cdot W\left[\sqrt{\left(\sum_{i=0}^{M-1}(N_i+1)\right)\cdot log(2)}\right]/log(2)$	$r_2=W\left[4.\left(\sum_{i=0}^{M-1}(N_i+1)\right).log(2)\right]/log(2)$

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

III. EXPERIMENTAL RESULTS

CSD is a widespread technique used in designing the vast majority of SCM/MCM blocks. It has many attractive features, such as: 33% reduction over binary, ease of use, full-predictability, suitable for serial/parallel reduction of adder depth, and especially, adaptable to carry-save-array (CSA) implementation because the addition terms are independent.

We have first compared RADIX-2^r MCM to CSD in the case of a carry-propagate-adder (CPA) implementation for a number of 32-bit constants varying from 1 to 1000. The results are reported in Table V. Note that the savings in *Avg* and *Upb* are not asymptotically bounded, i.e., they keep increasing as the product $M\times N$ increases. For N=32, a saving of 50% over CSD is attained in *Avg* and *Upb* for M=28, and M=4, respectively. For serial *Ath*, an average of 40% is achieved over CSD, independently of M. The reason is that for a fixed N, high variations of M produce almost the same value of r'.

As for a parallel implementation based on a tree structure, CSD *Ath* is lower than RADIX-2^r for any value of M. The reason is that in the parallel *Ath* formula, which is $\left\lceil log_2\left\lceil (N+1)/r'\right\rceil\right\rceil+r'-2$, the term r' prevails over the term log as r' increases. Therefore, lower values of r' decrease RADIX-2^r *Ath* as indicated in Table VI. Note that r'=7 is the value that optimizes *Upb*. For r'=3, RADIX-2^r and CSD exhibit the same $Ath^{//}$ while RADIX-2^r *Upb* and *Avg* are still better.

TABLE IV
MCM SOLUTION FOR THE FOLLOWING CONSTANTS: C_0=(84AB5)$_H$, C_1=(64AB55)$_H$, C_2=(5959595B)$_H$

Algorithm	Adder-cost	Serial adder-depth		Parallel adder-depth	
		Max	Avg	Max	Avg
Hcub [9]	13	11	8.33	–	
BHM [8]	15	6	5.33	–	
Lefèvre's CSP [7]	15	5	4.66	–	
CSD [5]	34	15	11.33	4	4.00
RADIX-2^r MCM	19	8	6.33	4	4.00
R3 MCM	15	5	4.33	4	4.00

$r'=r_1$ (see Table II).

TABLE VI
RADIX-2^r MCM VERSUS CSD: DECREASING VALUES OF r' FOR A NUMBER OF 80 NONNEGATIVE CONSTANTS WITH THE SAME BIT-SIZE N=32

r'		3	4	5	6	7
Avg	RADIX-2^r	691	598.00	469.50	407.49	347.85
	CSD	808.88				
	Saving (%)	14.57	26.07	41.95	49.62	56.99
Ath$^{//}$	RADIX-2^r	5	6	6	7	8
	CSD	5				
	Saving (%)	00.00	–20.00	–20.00	–40.00	–60.00
Upb	RADIX-2^r	801	643	487	415	351
	CSD	1280				
	Saving (%)	37.42	49.76	61.95	67.57	72.57

We have also compared RADIX-2^r MCM to the best MCM heuristics. Fig. 2 gives an idea on how *Avg* evolves with respect to the number of constants for a variation span from 1 to 100. Note that for RADIX-2^r MCM the *Avg* curve is exact as it directly derives from the expression given in Table II, while for the other curves; *Avg* is taken *only* over 50 uniformly distributed random constant sets [9]. Here a *big question* arises on the *accuracy* of these three curves, given that a set of 50 samples is relatively insignificant compared to billions of billions of samples when considering for example M=100. But despite the inaccuracy of the curves, RADIX-2^r MCM shows almost the same *Avg* as Lefèvre's CSP [7]. This means both exhibit the same compression performance, but with an important runtime overhead for Lefèvre's CSP (see Table VII). Note that we proved in [4] that R3 is better than RADIX-2^r in *Avg*. Because of their lower runtime, RADIX-2^r and R3 MCM are very useful for huge complexity problems ($M\times N>>$).

TABLE V
RADIX-2^r MCM VERSUS CSD: *Avg*, *Ath*, AND *Upb* FOR A NUMBER OF M NONNEGATIVE CONSTANTS WITH THE SAME BIT-SIZE N=32

M			1	2	10	20	40	60	80	100	200	500	1000
r'			4	5	6	7	7	7	7	9	9	9	11
Avg	RADIX-2^r	min	8.96	16.57	62.78	106.53	188.73	268.51	347.84	411.29	722.58	1623.06	2495.50
		max		18.56	64.06	110.21	189.43	268.65	347.87	426.21	725.43	1623.09	2509.53
	CSD		10.11	20.22	101.11	202.22	404.44	606.66	808.88	1011.11	2022.22	5055.55	10111.11
	Saving (%)		11.37	13.15	37.27	46.40	53.24	55.72	56.99	58.58	64.19	67.89	75.24
Ath	RADIX-2^r	···	10	9	9	9	9	9	9	10	10	10	11
		//	6	6	7	8	8	8	8	9	9	9	11
	CSD	···	16	16	16	16	16	16	16	16	16	16	16
		//	5	5	5	5	5	5	5	5	5	5	5
	Saving (%)	···	37.50	43.75	43.75	43.75	43.75	43.75	43.75	37.50	37.50	37.50	31.25
		//	–20.00	–20.00	–40.00	–60.00	–60.00	–60.00	–60.00	–80.00	–80.00	–80.00	–120.00
Upb	RADIX-2^r		11	19	65	111	191	271	351	427	727	1627	2511
	CSD		16	32	160	320	640	960	1280	1600	3200	8000	16000
	Saving (%)		31.25	40.62	59.37	65.31	70.15	71.77	72.57	73.31	77.28	79.66	84.30

RADIX-2^r MCM formulas for *Avg*, *Upb*, and *Ath* are taken from Table II with $r'=r_1$. For $r'\ge 5$, the saving in *Avg* is calculated considering (min+max)/2.

...: Serial implementation (adders connected in series); //: Parallel implementation based on a tree structure. For RADIX-2^r, $Ath^{\cdots}=\left\lceil (N+1)/r'\right\rceil+r'-3$, and $Ath^{//}=\left\lceil log_2\left\lceil (N+1)/r'\right\rceil\right\rceil+r'-2$. For CSD MCM, $Avg=M\times\left[(N+1)/3-8/9\right]$, $Upb=M\times\left[\left\lceil (N+1)/2\right\rceil-1\right]$, $Ath^{\cdots}=\left\lceil (N+1)/2\right\rceil-1$, and $Ath^{//}=\left\lceil log_2\left\lceil (N+1)/2\right\rceil\right\rceil$

TABLE VII
RUNTIME COMPLEXITY OF MCM ALGORITHMS

Hcub [9]	BHM [8]	Lefèvre CSP[7]	RAGn[9]	MADc [14]	NAIAD [15]	SIREN [15]	CSD [5]	RADIX-2^r	R3
$O(M^4\times N^5\times log(M\times N)+M^3\times N^6)$	$O(M^3\times N^4)$	$O(M^3\times N^3)$	$O(M^2\times N^3\times log(M\times N))$	$O(M\times 2^N)$	$O(M\times 2^N)$	$O(2^{M\times N})$	$O(M\times N)$	$O(M\times N/r')$	$O(M\times N)$

M: number of constants. N: bit-size of the constants. $r'=r_1$ or r_2 (see Table II).

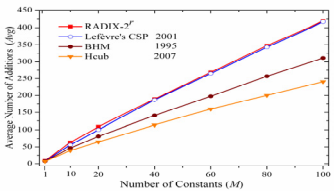

Fig. 2. *Avg* comparison for 32-bit constants ($r'=r_1$).

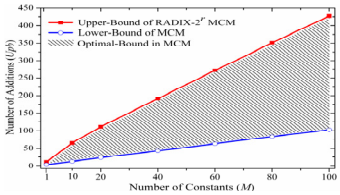

Fig. 3. *Upb* comparison for 32-bit constants ($r'=r_1$).

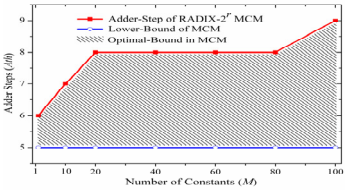

Fig. 4. *Ath* comparison for 32-bit constants ($r'=r_1$).

Neither *Upb* nor *Ath* are known at the current state-of-the-art of MCM. This is not only because the existing heuristics are unpredictable (no analytic formulas), but also because the calculation of *Upb* and *Ath* requires a prohibitive runtime. While it is feasible to roughly estimate *Avg* using small sets of uniformly distributed random constants, in the case of *Upb* and *Ath*, all possible combinations of the constants must be explored, causing a combinatorial explosion. Therefore, the only remaining possibility is to confront RADIX-2^r *Upb* and *Ath* to Gustafsson's lower-bounds for MCM [12], which are $\lceil log_2\lceil (N+1)/2\rceil\rceil+M-1$ and $\lceil log_2\lceil (N+1)/2\rceil\rceil$, respectively. A lower bound does not state that there is a solution that actually has an *Upb* or an *Ath* equal to the lower-bound; it only states that there are no solutions with an *Upb* or an *Ath* lower than the bound. Hence, if a heuristic finds a solution with an *Upb* or an *Ath* equal to the lower bound then that solution is optimal.

The hatched regions in Fig. 3 and 4 represent the space for an optimal solution of *Upb* and *Ath*, respectively. In [3], we came to a significant conclusion on the metrics: a lower *Avg* does not guarantee a lower *Upb* or *Ath*. We observe that RADIX-2^r $Ath^{\cdots}$ and $Ath^{//}$ remains somewhat constant; insignificantly altered by the number of constants M that grows from 1 up to 1000 (see Table V and Fig. 4). Such a significant feature leads to shorter critical paths in MCM blocks, comparatively to the existing heuristics. Note that the serial $Ath^{\cdots}$ can be further decreased taking $r'=r_2$ (see Table II) but to the detriment of *Avg* and *Upb* as shown in [4].

In fact, *Ath* is not only a measure of the critical-path (speed), but also a *good* indicator of the power consumption. It has been proved in [13][14] that a lower *Ath* results in lower power consumption. This is because shorter paths reduce the number of glitches which are the main factor in power consumption. Therefore, from a power point of view, it is often beneficial to have more adders (*to a certain extent*) if it means that the *Ath* can be decreased [14].

Note that the best MCM algorithms can not compete with RADIX-2^r in *Ath*. The reason is that RADIX-2^r allows a *logarithmic* reduction of the adder steps ($Ath^{//}$), while it is not possible in the other case. The best MCM algorithms aim to maximize the shared terms, making the parallel reduction of the different paths impossible. To achieve a better adder cost, they rely either on the acyclic directed graphs (DAG) [9] or common subexpression elimination (CSE) [7]. However, both approaches obey to the same calculation pattern described hereafter. A finite sequence of integers u_0, u_1, u_2, ..., u_q is acceptable for C_m (m=1..M) if it satisfies the properties:

- Initial value: $u_0=1$;
- For all i>0, $u_i=s_i\times u_j\times 2^{a_i}+r_i\times u_k\times 2^{b_i}$; with $j,k<i$; $s_i, r_i\in\{-1,0,1\}$; and $a_i, b_i\in \mathrm{N}$;
- Final value: $u_q\times 2^{c_q}=C_m$, with $c_q\in \mathrm{N}$.

The objective is to generate an acceptable sequence $(u_i)_{0\le i\le q}$ that is as short as possible. The value q is called the quality, or length [7]. It is the condition $j,k<i$ that is responsible of the subsequent connection of adders in series.

As a practical example, we considered the 24th-order linear-phase FIR filter used in [14]. The input data wordlength was fixed to 16 bits. The symmetric impulse response is H={–710, 327, 505, 582, 398, –35, –499, –662, –266, 699, 1943, 2987, 3395, 2987, ...}/2^{14}. The set of unique positive odd integer coefficients to be realized contains 13 elements: H_{min}={355, 327, 505, 291, 199, 35, 499, 331, 133, 699, 1943, 2987, 3395}. H_{min} is a well-known set of coefficients for which it is difficult to find a good solution in terms of adders [14].

To H_{min} corresponds $\sum_{i=0}^{12}(N_i+1)=134$, which gives r'=5. The RADIX-2^r MCM solution [11] for H_{min} yields 22 operations:
355=3+11×2^5; **327**=7+5×2^6; **505**=–7+2^9; **291**=3+9×2^5;
199=7+3×2^6; **35**=3+2^5; **499**=–13+2^9; **331**=11+5×2^6;
133=5+2^7; **699**=–5+11×2^6; **1943**=–9–3×2^5+2^{11};
2987=11–3×2^5+3×2^{10}; **3395**=3+5×2^6+3×2^{10};
with 3=1+2; 5=1+2^2; 7=–1+2^3; 9=1+2^3; 11=9+2; 13=9+2^2.

Note that RADIX-2^r requires half as many additions as CSD (see Table VIII). It is normal that R3 gives similar results because the bit-size of the constants is small (N_{max}=12) [4]. The maximum and average adder-depths in RADIX-2^r are 3 and 2.46, respectively. These are the optimal values for H_{min} as proved by the lower depth formula $\lceil log_2\lceil (N_{max}+1)/2\rceil\rceil$ of [12].

Twelve Verilog versions of the transposed form of the filter were implemented, based each on a different MCM recoding of the coefficients. They were synthesized using the Cadence RTL Compiler with TSMC 1P6M 0.18µm CMOS process (TSMC-Artisan standard-cell-library). The synthesis tool was constrained to a relaxed time period of 100ns. The place & route was performed using Cadence SoC Encounter. The power was evaluated at 10 MHz frequency. The post-layout results in speed, power, and area are reported in Table IX.

A strong correlation between speed/power and adder-depth can be observed. This confirms once more the results, and specially, the conclusion drawn in [14]. Though CSD induces the shortest paths, the delay is slightly higher due to a longer routing (area). The results given by RADIX-2^r and R3 [4] are among the best when considering the Speed/Power Product

TABLE VIII
ADDER COST AND ADDER DEPTH OF THE FIR FILTER

Algorithm	Adder cost	Maximum adder depth	Average adder depth
DIFFAG [14]	16	7	3.92
C1 [14]	19	5	2.84
Pasko [14]	23	4	2.69
MADc [14]	20	**3**	**2.46**
Hcub* [9]	16	7	4.46
BHM* [8]	19	5	3.38
RAGn* [9]	18	10	5.07
NAIAD [15]	18	**3**	2.69
SIREN [15]	16	7	4.00
CSD [5]	44	**3**	**2.46**
RADIX-2^r MCM	22	**3**	**2.46**
R3 MCM	22	**3**	**2.46**

*: Values taken from www.spiral.net. The lower bound in adder cost is 14. The lower bound in adder-depth is 3. **x**: Optimal value.

TABLE IX
POST-LAYOUT IMPLEMENTATION RESULTS OF THE FIR FILTER

Algorithm	Delay* (ns)	Power+ (mw)	Area# (μm^2)	Delay×Power (ns×mw)
DIFFAG [14]	20.42	3.32	95845	67.79
C1 [14]	18.63	3.21	98012	59.80
Pasko [14]	18.17	3.13	98640	56.87
MADc [14]	17.41	2.89	98157	50.31
Hcub [9]	20.63	3.38	95715	69.72
BHM [8]	19.41	3.17	97541	61.52
RAGn [9]	21.57	3.37	97967	72.69
NAIAD [15]	17.19	3.01	97876	51.74
SIREN [15]	20.06	3.28	96612	65.79
CSD [5]	18.19	3.22	110343	58.57
RADIX-2^r MCM	17.48	2.93	98355	51.21
R3 MCM	17.17	3.02	98399	51.85

*: Minimum clock period. +: Total dynamic power dissipation.
#: Total area.

(SPP). The gap in SPP will be more significant for high-order filters as RADIX-2^r and R3 are more efficient in problems of high complexity (see $Ath^{\cdots}$ and $Ath^{//}$ formulas in Table V).

While the conventional metric of adder-depth was proved as a reliable measure of the critical-path, a more accurate delay model has been recently introduced in [16]. The latter is based on a bit-level propagation of signals for a fine-grained analysis of the critical path of MCM blocks. It shows that the delays of the shift-add network estimated at bit-level are *shorter* than the delays at adder level (cascaded additions).

So far, only common two-input adders were considered. As FPGAs today support ternary adders, i.e., adders with three inputs (3:2 compressors), the adder-cost as well as the adder-depth can be reduced further. This is made possible by exploiting the unused look-up-tables (LUTs) in the FPGA carry-chain [17]. Applied to the FIR filter, the number of additions is reduced to 19 instead of 22 since the term $1943\times X$, $2987\times X$, and $3395\times X$ requires each just one adder. The maximum adder-depth remains the same (3), while the average adder-depth is reduced to 2.23. Using ternary adders, the maximum adder-depth in RADIX-2^r MCM becomes $\lceil log_3\lceil (N+1)/r' \rceil \rceil + r' - 2 = \lceil 0.631 \cdot log_2\lceil (N+1)/r' \rceil \rceil + r' - 2$ since three partial products can be added in each stage. However, in smaller designs the gain in adder-depth is somewhat offset as the ternary adders are slower due to the internal routing [17]. Conversely, it is expected that in larger designs the gain prevails, but this has to be practically investigated.

Another design paradigm is the hybrid MCM on FPGA. By combining embedded multipliers (hard macros) with additions/subtractions and shifts, efficient implementation of the MCM block can be achieved. It has been shown in [18] that minimal speed and power values are attained with a number m of embedded multipliers, such that $0<m<M$. In RADIX-2^r, we propose to map the whole set of odd-multiple PP to embedded multipliers to reduce the adder-depth. In this case, it becomes $\lceil log_2\lceil (N+1)/r' \rceil \rceil + \theta$, where θ is the multiplier delay. Note that θ can be finely reduced by inserting pipeline stages as in, for instance, the 18×18 bits Xilinx's multipliers.

IV. CONCLUSION AND FUTURE WORK

A fully-predictable and sublinear-runtime MCM heuristic with the shotrest adder-depth has been developed (RADIX-2^r) and improved (R3). Its proved limits with an exact number of additions for the average, adder-cost, and adder-depth, are the unique analytic-bounds known so far for MCM. However, optimal bounds remain an open research problem.

While the presented results apply only to a carry-propagate-adder (CPA) implementation of MCM blocks, we are currently exploring the radix-2^r arithmetic for a carry-save-adder (CSA) realization, required in high-speed applications.

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] L. Aksoy, E. da Costa, and P. Flores, "Exact and Approximate Algorithms for the Optimisation of Area and Delay in Multiple Constant Multiplication," IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 27, no. 6, pp. 1013-1026, June 2008.

[3] A.K. Oudjida and N. Chaillet, "Radix-2^r Arithmetic for Multiplication by a Constant," IEEE Trans. on Circuits and Systems II: Express Brief, vol. 61, no 5, pp. 349-353, May 2014.

[4] A.K. Oudjida, N. Chaillet, and M.L. Berrandjia, "Radix-2^r Arithmetic for Multiplication by a Constant: Further Results and Improvements," IEEE Trans. on Circuits and Systems II: Express Brief, vol. 62, no. 4, pp. 372-376, April 2015.

[5] 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.

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

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

[8] 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.

[9] 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.

[10] 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.

[11] A.K. Oudjida and M.L. Berrandjia, "RADIX-2^r MCM Web Page," June 2015. Available at: http://www.cdta.dz/products/mcm/

[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] M. Fraust, O. Gustafsson, and C. Chip-Hong, "Reconfigurable Multiple Constant Multiplication Using Minimum Adder Depth," Proceedings of IEEE ASILOMAR Conference on Signals, Systems, and Computers, pp. 1293-1301, CA, USA, November 2010.

[14] K. Johansson, O. Gustafsson, L.S. DeBrunner, and L. Wanhammar, "Minimum Adder Depth Multiple Constant Multiplication Algorithm for Low Power FIR Filters," Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1439-1442, Rio de Janeiro, Brazil, May 2011.

[15] L. Aksoy, P. Flores, and J. Monteiro, "Exact and Approximate Algorithms for the Filter Design Optimization Problem" IEEE Trans. on Signal Processing, vol. 63, no 1, pp. 142-154, January 2015.

[16] X. Lu, Y.J. Yu, and P.K. Meher, "Fine-Grained Critical Path Analysis and Optimization for Area-Time Efficient Realization of Multiple Constant Multiplications," IEEE Trans. on Circuits and Systems I: Regular Papers, vol. 62, no 3, pp. 863-872, March 2014.

[17] M. Kumm et al., ". Multiple Constant Multiplication with Ternary adders," Proceedings of the IEEE 23rd International Conference on Field Programmable Logic and Applications (FPL), pp. 1-8, Porto, Portugal, Septembre 2013.

[18] L. Aksoy, P. Flores, and J. Monteiro, "Efficient Design of FIR Filters Using Hybrid Multiple Constant Multiplications on FPGA," Proceedings of the IEEE International Conference on Computer Design (ICCD), pp. 42-47, Seoul, South-Corea, October 2014.