IET Circuits, Devices & Systems

Radix-2r Recoding with Common Subexpression Elimination for Multiple Constant Multiplication

CDS-2020-0213 | Research Article

Submitted by: Abdekrim Kamel Oudjida, Ahmed Liacha, Mohammed Bakiri, José Monteiro, Paulo Flores

Keywords: ARITHMETIC, HIGH-SPEED INTEGRATED CIRCUITS, LOW POWER DESIGN, DIGITAL CIRCUITS, FILTERS

Radix-2^r Recoding with Common Subexpression Elimination for Multiple Constant Multiplication

Ahmed Liacha, Abdelkrim K. Oudjida, Mohammed Bakiri, José Monteiro, and Paulo Flores

***Abstract*— In a recent work on multiple constant multiplication (MCM) problem, a fully predictable sub-linear runtime heuristic was introduced, called Radix-2^r MCM. This method shows competitive results in speed, power, and area, comparatively with the leading algorithms. In this paper, we combine Radix-2^r MCM with an exact common subexpression elimination (exact-CSE) algorithm. The resulting algorithm denoted Radix-2^r-CSE allows a substantial reduction in the number of addition/subtraction operations in MCM by maximising the sharing of partial terms after an initial recoding in Radix-2^r MCM. The savings over Radix-2^r MCM ranges from 4.34% up to 18.75% (10% on average) when considering a set of 14 benchmark FIR filters of varied complexity.**

***Index Terms*— 0-1 Integer Linear Programming (ILP) Solver, Common Subexpression Elimination (CSE), High-Speed and Low-Power Design, Multiplierless Single/Multiple Constant Multiplication (SCM/MCM), Radix-2^r Arithmetic.**

I. Backgound and Motivation

The hardware complexity of many linear-time-invariant (LTI) systems, such as finite impulse response (FIR) filters, is dominated by the multiplication of a set of constants $\{C_0, C_1, C_2, \cdots, C_{M-1}\}$ with the same input variable X. This operation is known as multiple constant multiplication (MCM). To be efficiently designed, instead of using full block multipliers, a multiplierless implementation based only on operations of addition, subtraction, and left-shift is used. As adders and subtractors exhibit the same hardware complexity, we will refer to both as adders. Therefore, the MCM problem is defined as the process of finding the minimum number of additions since left-shifts are *costless* in hardware. Another requirement of MCM is the minimization of the critical path across the cascaded adders in order to reduce delay and power. The computational complexity of MCM is *conjectured* to be NP-hard [1].

Based on the Radix-2^r arithmetic, we have developed in a previous work [2][3][4][5] a new heuristic called Radix-2^r MCM. 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. Furthermore, it requires sub-linear computational complexity $O(M\times N/r)$, where M is the number of constants, N is the bit-length of the constants, and r is a function of (M, N). This means that Radix-2^r MCM has *no* limited applicability regarding the complexity of the problem, contrary to most existing algorithms [4][5].

We have demonstrated in [5] that Radix-2^r MCM leads to a near-optimal solution in adder-depth without sacrificing the adder-cost. Furthermore, we have analytically and experimentally proved [5] through physical implementation of many benchmark FIR filters of different complexities the superiority of Radix-2^r MCM in handling medium/high order filters in comparison to the most area efficient algorithms, notably to the cumulative benefit heuristic (Hcub) [6] known for its lowest adder-cost.

In Radix-2^r MCM [4], each constant is implemented apart, independently from the others. But all constants share the same set of nontrivial partial products depending on the used radix (2^r): $\left\{1\times X, 3\times X, 5\times X, \cdots, \left(2^{r-1}-1\right)\times X\right\}$. However, Radix-$2^r$ MCM does not handle the sharing of common subexpressions neither within each constant separately, nor within the set of the whole constants. To make this possible, an additional operation of common subexpressions elimination (CSE) is necessary.

Recently, a new heuristic called H-Radix-2^r has been developed [7] to reduce the adder-cost. This method combines Radix-2^r MCM with Lefevre's CSE algorithm [8]. However, H-Radix-2^r has two serious limitations: only the single constant multiplication (SCM) version of [4] is used (M=1) with just one fixed radix (radix-2^3).

The main purpose of this work is to further reduce the adder-cost by developing an MCM heuristic that copes with the appropriate radix (2^r) in conformity with the complexity (M, N) of the problem. The higher the complexity, the higher the radix. The proposed solution is a mixture of the Radix-2^r MCM [4] and the exact-CSE algorithm introduced in [9]. The resulting algorithm is denoted Radix-2^r-CSE.

After an initial recoding of the constants in Radix-2^r MCM, we apply the exact-CSE to maximize the sharing of partial products terms. We model this problem into a 0-1 integer linear programming (ILP) problem [10] with a cost function to minimize, and some constraints to satisfy. The solution corresponding to the minimum number of operations is provided by the generic 0-1 ILP solver tool available in [11].

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

A. Liacha (Liacha @cdta.dz), A.K. Oudjida, and M. Bakiri are with "Centre de Développement des Technologies Avancées", CDTA, Cité du 20 août 1956, Baba-Hassen, Algiers, 16303, Algeria.

P. Flores and J. Monteiro are with the Instituto Superior Técnico (IST), Technical University of Lisbon, 1000-029 Lisbon, Portugal (e-mail: pff@inesc-id.pt; jcm@inesc-id.pt).

II. Overview of Radix-2^r MCM and Exact-CSE Algorithms

We summarize hereafter the basic theoretical background behind the two combined algorithms.

A. Radix-2^r MCM

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

$$C = \sum_{i=0}^{\lceil (N+1)/r \rceil - 1} \begin{pmatrix} -2^{r-1}c_{r\times i+r-1} + 2^{r-2}c_{r\times i+r-2} + \cdots + \\ 2^2 c_{r\times i+2} + 2^1 c_{r\times i+1} + 2^0 c_{r\times i} + c_{r\times i-1} \end{pmatrix} \times 2^{r\times i} = \sum_{j=0}^{(N+1)/r-1} Q_j \times 2^{rj}, \quad (1)$$

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 [4]. 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 $|OM(2^r)| = 2^{r-2}-1$.

We have proved in [4][5] exact analytic formulas for MCM in terms of upper-bound in adder-cost (*Upb*), average cost (*Avg*), and adder-depth (*Ath*). These metrics are included in Table I. To the best of our knowledge, they are the *unique* exact analytic bounds known so far for MCM.

B. Exact-CSE algorithm

Exact-CSE [9] computes the minimum number of addition operations by maximizing the sharing of partial terms in SCM/MCM. It runs in four main consecutive steps as follows:

1. All possible implementations of the constants are extracted from the non-zero digits defined in binary, minimum signed digit (MSD) [12], or canonical signed digit (CSD) [13] representations.
2. The constants implementations are represented in a Boolean network formed only of AND and OR gates.
3. The MCM problem is formalized as a 0-1 ILP problem with an adder-cost objective-function to minimize, and a number of constraints to comply with.
4. Finally, the solution with minimum number of operations is obtained using a generic ILP solver [11].

III. Radix-2^r-CSE Algorithm

Four main steps are involved in Radix-2^r-CSE. The first one consists in finding the Radix-2^r recoding with a minimum adder-cost of the target constants to be implemented. The Radix-2^r MCM solutions are given by the online version available in [14] allowing three options: 1) adder-cost optimization; 2) adder-depth optimization; 3) trade-off between adder-cost and adder-depth by specifying the radix (fixed value of r). As example, let us consider the constant $(19125)_{10}$, whose bit-length is N=15. Since M=1 (one constant), the formula of r in Table I that optimizes *Upb* gives

Table I
Radix-2^r MCM for a number of M nonnegative constants with different bit-sizes N_i

Metrics	Equations
Adder cost	$Upb(r) = \sum_{i=0}^{M-1} \left\lceil \frac{N_i+1}{r} \right\rceil + 2^{r-2} - 1 - M$ [4]
Adder depth	$Ath(r) = \left\lceil \frac{\max(N_i)+1}{r} \right\rceil + \lceil r/2 \rceil - 2$ with i=0..M–1 [4] [5]
Average cost	$-M + Avg_{pp} + Avg_{om} \le Avg(r) \le -M - 1 + Avg_{pp} + 2^{r-2}$ with $Avg_{pp} = (1-2^{-r}) \times \sum_{i=0}^{M-1} \left\lceil \frac{N_i+1}{r} \right\rceil$, $Avg_{om} = \sum_{j=0}^{\sum_{i=0}^{M-1}\left\lceil \frac{N_i+1}{r} \right\rceil - 1} \left\{ \sum_{k=1}^{2^{r-2}-1} P(m_{jk}) \times [1-P(m_{jk})]^j \right\}$, $P(m_{jk}) = \frac{\log_2 \left\lfloor \frac{2^{r-1}}{2\times k+1} \right\rfloor}{2^{r-1}}$. [4]

$r = 2 \cdot W\left[\sqrt{\left(\sum_{i=0}^{M-1} (N_i+1) \right) \cdot \log(2)} \right] / \log(2)$; W: Lambert function; $\lceil\ \rceil$: Ceiling

r_{opt}=4. Therefore, the Radix-2^r recoding is 19125 = 5<<12 – 5<<8 – 5<<4 + 5<<0 with 5 = 1<<2 + 1. The "<<" denotes a left-shift (a<<b=a×2^b). Note that Radix-2^r solution yields 4 additions.

To further improve the sharing of partial products, the search space can be extended to all possibilities using the "trade-off" option available in [14]. This allows finding all implementations of the constants by varying the value of r from 2 up to r_{opt} + 2. For 19125, r varies from 2 up to 4+2=6.

In the second step, the entire set of implementations of the target constants issued by Radix-2^r MCM solution is represented in a unique Boolean combinational network based only on AND and OR gates. The latter presents the following features:

1. The primary input of the network is the input value (or its shifted versions) to be multiplied with the constants.
2. A two-input AND gate in the network represents an addition/subtraction operation of two partial products and generates a given partial term.
3. An OR gate in the network represents a target constant or a partial term and combines all possible implementations of the constant.
4. The primary outputs of the network are the OR gate outputs associated with the target constants in the MCM problem.

The Boolean network generated for the target constant 19125 implemented with Radix-2^r MCM algorithm is given in Figure 1, where equivalent cases are omitted. The network represents all possible additions/subtractions between partial terms of the target constant 19125.

In the conversion of the MCM problem to a 0-1 ILP problem, we need to include optimization variables in the Boolean network, so that the cost function can be minimized, i.e., the linear function of the optimization variables can be constructed. To this end, the optimization variables are associated with the operations that are required for the implementations of target constants and partial terms. Thus, we add a third input denoting an optimization variable to each AND gate that represents an operation in the network.

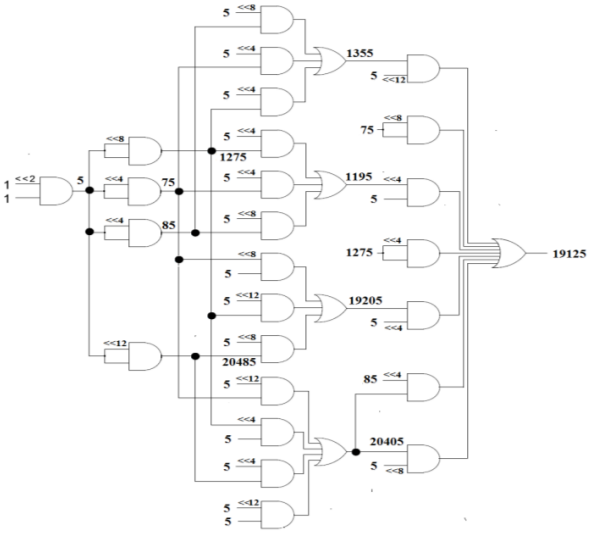

Fig. 1. Boolean network corresponding to the target constant 19125 in Radix-2^r representation with r_{opt}=4 only.

Once we have added the optimization variables to the Boolean network, some network simplifications are applied for minimizing the number of operations and partial terms in the Boolean network [9].

In the third step, the MCM problem is formalized as a 0-1 ILP problem. After the Boolean network is constructed, the conversion of the MCM problem into a 0-1 ILP problem is then straight-forward. The cost function is formed as a linear function of optimization variables where the cost value of each optimization variable is set to 1. The constraints of the 0-1 ILP problem are obtained by finding the conjunctive normal form (CNF) of each gate in the network and expressing each clause in CNF format [15]. For example, a 3-input AND gate, d=a^b^c, is translated to CNF as
$(a \sqcup \bar{d}) \cdot (b + \bar{d}) \cdot (c + \bar{d}) \cdot (\bar{a} + \bar{b} + \bar{c} + d)$.

Each clause is converted into a 0-1 ILP constraint using the straight-forward mapping presented in [10]. The three-input AND gate is described by the following set of restrictions:
$a - d \geq 0,\ b - d \geq 0,\ c - d \geq 0,\ -a - b - c + d \geq -2,$ with
$a,\ b,\ c,\ d \in \{0,\ 1\}$.

In the last step, the obtained model serves as an input to the generic 0-1 ILP solver [11] to find the solution with minimum number of operations. As a result, the algorithm with the additional CSE step induces the following recoding for the constant 19125:
19125 = 75<<8 – 75<<0, with 75 = 5<<4 – 5, 5 = 1<<2 + 1. Therefore, Radix-2^r-CSE yields only 3 additions for 19125, while it produces 4 with Radix-2^r. This is a simple illustrative SCM case, but the saving can be much more substantial in MCM applications as shown hereafter with the following MCM problem: $\{571113, 61161, 17847\}$. The Radix-2^r MCM solution given by [14] corresponds to r_{opt}=4 as follows:
571113 = –7 – 1<<4 + 7<<8 – 5<<12 – 7<<16 + 1<<20,
61161 = –7 – 1<<4 – 1<<8 – 1<<12 +1<<16,
17847 = 7<<0 – 5<<4 + 3<<9 + 1<<14,
with 3 = 1<<1 + 1, 5 =1<<2 + 1, 7 = 1<<3 – 1.

Applying on it the CSE step gives:
571113 = –23 – 73<<8 – 7<<16 + 1<<20,
61161 = –23 – 1<<8 – 1<<12 + 1<<16,
17847 = –73 + 3<<9 + 1<<14,
with 3 = 1<<1 + 1, 5 = 1<<2 + 1, 7 = 1<<3 – 1,
23 = 7 + 1<<4, 73 = 5<<4 – 7.

Note that the Radix-2^r MCM solution requires 15 additions, while the Radix-2^r-CSE needs 13. With an extra runtime, the adder-cost is drastically reduced by extending the search space to all Radix-2^r MCM solutions for r varying from 2 up to r_{opt}+2=6 (Table II). Note that r=5 and r=6 yield better solutions (12 additions) than r_{opt} (15 additions). This is normal since r_{opt} ensures an upper-bound (*Upb*) in number of additions regarding the bit-lengths of the constants (see Table I), independently of their values. Nevertheless, according to the many statistical tests we performed, in *no case* the optimal solution of Radix-2^r MCM goes beyond r_{opt}±2. Consequently, the Radix-2^r-CSE solution comprises only 6 additions as shown hereafter:
571113 = 279<<11 – 279,
61161 = 15<<12 – 279,
17847 = 279<<6 – 9,
with 15 = 1<<4 – 1, 9 = 1<<3 + 1, 279=9<<5 – 9.

TABLE II
RADIX–2^r MCM RECODING FOR THE VALUE OF r VARYING FROM 2 TO 6.

r	Constants	Odd-multiples	Adder-cost
2	571113 = +1<<0 +1<<3 –1<<5 –1<<8 –1<<11 –1<<14 +1<<16 +1<<19 61161 = +1<<0 +1<<3 –1<<5 –1<<8 –1<<12 +1<<16 17847 = –1<<0 –1<<3 –1<<6 –1<<9 +1<<11 +1<<14	None	17
3	571113 = +1<<0 –3<<3+1<<8+3<<9+3<<12+1<<15 +1<<19 61161=+1<<0–3<<3–1<<8–1<<12+1<<16 17847=–1<<0 –1<<3–1<<6+3<<9+1<<14	3=1<<1 +1	15
4 (r_{opt})	571113=–7<<0–1<<4+7<<8–5<<12–7<<16 +1<<20 61161 = –7<<0 –1<<4 –1<<8 –1<<12 +1<<16 17847 = +7<<0 –5<<4 +3<<9 +1<<14	7=1<<3–1 5=1<<2+1 3=1<<1+1	15
5	571113=9<<0–9<<5 +7<<11–15<<15+1<<20 61161=9<<0–9<<5+15<<12 17847=– 9<<0+7<<6 –15<<10+1<<15	9=1<<3+1 7=1<<3–1 15=1<<4–1	12
6	571113 = –23<<0 +7<<8 +11<<12 +1<<19 61161 = –23<<0 –1<<8 +15<<12 17847 = –9<<0 +23<<6 +1<<14	7= 1<<3–1 9=1<<3+1 11=9+1<<1 15=1<<4–1 23=15+1<<3	12

TABLE III
Radix-2^r-CSE VERSUS Radix-2^r: ADDER-COST.

Filter	Lower bound [19]		Hcub [6]		Radix-2^r [4]		Radix-2^r-CSE		Cost saving (%) / Radix-2^r	Depth increase / Radix-2^r
	Cost	Depth	Cost	Depth	Cost	Depth	Cost	Depth		
Low order filters										
FIR_25_13_12 [4]	14	3	16	7	22	3	21	3	4.54	0
FIR4_30_14_13 [16]	14	3	18	7	23	3	22	3	4.34	0
FIR3_30_14_13 [16]	14	3	19	7	27	3	22	3	18.51	0
FIR1_40_19_12 [16]	20	3	23	7	33	3	29	3	12.12	0
FIR2_40_19_13 [16]	20	3	24	6	36	3	31	3	13.88	0
FIR7_40_19_14 [16]	20	3	24	7	35	3	31	4	11.42	1
Medium order filters										
FIR6_60_29_14 [16]	29	3	32	10	47	4	43	4	8.51	0
FIR8_80_36_14 [16]	37	3	38	5	51	4	45	4	11.76	0
FIR5_80_39_15 [16]	40	3	42	8	64	4	52	4	18.75	0
High order filters										
FIR_279_140_24 [17]	141	4	158	26	239	4	224	5	6.27	1
FIR_418_208_22 [17]	208	4	212	9	310	5	285	5	8.06	0
FIR_516_256_24 [17]	256	4	259	9	362	5	333	5	8.01	0
FIR_631_313_23 [17]	313	4	315	6	403	5	370	6	8.18	1
FIR_695_345_24 [17]	345	4	348	6	444	5	416	6	6.30	1
Average									10.04	0.28

The lower bound in cost and depth are given by $\left\lceil log_2\left\lceil\left[\min\left(N_i\right)+1\right]/2\right\rceil\right\rceil+M-1$ and $\left\lceil log_2\left\lceil\left[\max\left(N_i\right)+1\right]/2\right\rceil\right\rceil$, respectively.

IV. EXPERIMENTAL RESULTS

In this section, Radix-2^r-CSE is confronted to Radix-2^r in terms of adder-cost and adder-depth through a set of benchmark FIR filters taken from [4][16][17][18]. Note that we deal with the transposed form of FIR filters. We adopt the following nomenclature: FIR_*L*_*M*_*N*, where *L* is the number of coefficients (*H* set) of the filter, *M* is the number of unique positive odd integer coefficients (H_{min} set) to be solved in MCM, and *N* is the maximum bit length of the coefficients of H_{min} set. The solutions of Radix-2^r were obtained using the online version available in [14] with the "adder-cost" option. The comparison results are reported in Table III.

The results in adder-cost show that Radix-2^r-CSE algorithm achieves a saving over Radix-2^r varying between 4.34% and 18.75% (10.04% on average). While the adder-depth remains the same, or at worse increased by one addition (0.28% on average) as a result of cost minimization effort. Nevertheless, the adder-depth remains near-optimal, too much lower (5) in comparison to Hcub's that yields 26 for FIR_279_140_24.

The MCM block of each benchmark filter in Table III was coded in Verilog with an input data word length (X_b) fixed to 16 bits. All MCM blocks of generated filters were mapped to UMC 65nm standard-cell library (with AND area equal to 1.44μm^2) using Cadence RTL compiler (RC-14 version). The synthesis tool was constrained to a relaxed constraint of 50 ns, using only the worst case library (108°C and 1.08Volt). The place and route was performed with Cadence SoC Encounter (EDI-14 version) using multi-corner multi-mode techniques, including both the worst and best (−40°C and 1.32 Volt) cases library. Dynamic power consumption was evaluated with 2000 random input samples at 25 MHz frequency, since all circuits after place and route were able to run at most up to 50 Mhz. The total power was evaluated in both static and dynamic power, including leakage, switching, and internal power of the design collected from gate simulation level. The post-layout results in speed, power, and area are reported in Table IV.

The results show that for the low-order filters, Radix-2^r-CSE is better than Radix-2^r in area and power with an average saving of 9.27% and 5.23%, respectively. However, for the

TABLE IV
RADIX-2^r-CSE VERSUS RADIX-2^r: POST-LAYOUT IMPLEMENTATION RESULTS OF A NUMBER OF MCM BLOCKS IN 65nm CMOS TECHNOLOGY.

Filter	Delay* (ns)		Power+ (mw)		Area# (μm^2)		Delay Saving (%)	Power Saving (%)	Area Saving (%)
	Radix-2^r-CSE	Radix-2^r [6]	Radix-2^r-CSE	Radix-2^r [6]	Radix-2^r-CSE	Radix-2^r [6]			
Low order filters									
FIR_25_13_12 [4]	11.416	11,438	0.1729	0.1734	3708.360	3750.840	0.19	0.28	1.13
FIR4_30_14_13 [16]	10.891	10,747	0.1784	0.1934	3645.000	4023.720	−1.33	7.75	9.41
FIR3_30_14_13 [16]	11.126	10,985	0.1813	0.1902	3657.600	4139.280	−1.28	4.67	11.63
FIR1_40_19_12 [16]	11.643	10,376	0.2336	0.2434	4914.720	5532.840	−12.21	4.02	11.17
FIR2_40_19_13 [16]	10.695	11,699	0.2363	0.2631	5342.760	6492.960	−8.58	10.18	17.71
FIR7_40_19_14 [16]	11.547	11,941	0.2853	0.2866	5811.840	6091.560	3.29	4.53	4.59
Average							−3.32	5.23	9.27
Medium order filters									
FIR6_60_29_14 [16]	12.989	13,297	0.4070	0.3933	7767.000	7947.720	2.31	−3.48	2.27
FIR8_80_36_14 [16]	13.201	14,022	0.5075	0.5353	9072.720	9892.080	5.85	5.19	8.28
FIR5_80_39_15 [16]	13.751	13,470	0.6994	0.7662	10030.320	11881.080	−2.08	8.71	15.57
Average							2.02	3.47	8.70
High order filters									
FIR_279_140_24 [17]	16.212	16.709	3.7492	3.8475	48509.280	48670.560	2.97	2.55	0.33
FIR_418_208_22 [17]	16.714	17,405	4.9044	5.4010	62349.840	63387.360	3.97	9.19	1.63
FIR_516_256_24 [17]	17.514	19,546	6.8994	7.3457	74817.000	74090.160	10.39	6.07	−0.98
FIR_631_313_23 [17]	16.054	16,363	8.0107	7.8197	83276.640	82598.040	1.88	−2.44	−0.82
FIR_695_345_24 [17]	16.216	18,012	8.9159	9.1470	96746.400	89929.440	9.97	2.52	−7.58
Average							5.83	3.57	−1.48

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

TABLE V
ALGORITHM UTILIZATION VERSUS SIZE OF THE FILTER.

	Speed	Power	Area
Low Order Filters	Radix-2^r	Radix-2^r -CSE	
Medium Order Filters	Radix-2^r -CSE		
High Order Filters	Radix-2^r -CSE		Radix-2^r

speed the opposite is rather true. Radix-2^r is slightly better than Radix-2^r-CSE with an average saving of 3.32%. In medium-order filters, Radix-2^r-CSE yields better results than Radix-2^r in all cases, with an average saving of 8.70%, 3.47%, and 2.02% in area, power, and speed, respectively. In high-order filters, Radix-2^r-CSE is better than Radix-2^r in speed and power with an average saving of 5.83% and 3.57%, respectively. For the area, Radix-2^r is slightly better than Radix-2^r-CSE with an average saving of 1.48%.

As summarized in Table V, the experimental comparison showed that Radix-2^r-CSE is better than Radix-2^r in area, speed, and power for all complexities of benchmark FIR filters except in two cases. The first case is speed in low order filters. Note that the delay as shown in [5] is tightly related to adder-depth (*Ath*): an increase in *Ath* increases the delay. But since there is no increase in *Ath* except for FIR7_40_19_14, the only explanation for the increased delay is the longer routing as a consequence of the increased sharing of partial terms which breaks the placement regularity of the cells in the array. The delay corresponding to small order filters is more sensitive to the slightest increase in routing than in medium/high order filters. This is a negative CSE effect on speed. The second case is area in high order filters. In this case the sharing of partial terms with Radix-2^r-CSE leads to a solution that employs less adder blocks, but with a much larger bit-lengths due to an increased overhead (Δ) in terms of bit-level full-adders (FAs). The term Δ is the number of extra FAs that must be concatenated to the basic X_b FAs to form a complete adder (Δ+X_b FAs), where X_b is the bit-length of the input data X. This is also another negative effect of CSE. Reader is referred to Section IV in [5] for an in-depth analysis of this *critical* issue, which has been reported by many papers [20] insisting on the fact that fewer number of additions does not necessarily lead to fewer number of FAs. For instance, although the Hcub solution for FIR_279_140_24 yields less additions (158) than Radix-2^r-CSE (224), the latter occupies less area (48509.28 μm^2) than Hcub (55152.36 μm^2), which gives a saving of 12.04 %. See similar comparison between Hcub and Radix-2^r in [5].

Radix-2^r-CSE with extended search space (r varying from 2 up to r_{opt}+2) runs in a reasonable amount of time (few hours) on an Intel core i3 computer for the high order filters included in Table IV. These filters are the largest filters that we can find in the benchmark FIRsuite database [18] either in terms of number (up to 695) or bit-length (up to 24 bits) of coefficients.

V. CONCLUSION AND FUTURE WORK

We have combined Radix-2^r recoding to an exact CSE algorithm to decrease the adder-cost. The resulting algorithm Radix-2^r-CSE achieves 10% saving over Radix-2^r on average. We have also proved through circuit implementation of several benchmark MCM blocks that Radix-2^r-CSE performs better than Radix-2^r in speed, power, and area in most complexity cases, especially in medium-order filters.

We are currently exploring the possibility to decrease the density of nonzero digits in Radix-2^r recoding to further reduce the adder-cost.

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 (TCAD), vol. 30, no. 9, pp. 1373-1386, Sep. 2011.

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

[3] 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 (TCAS-II) : Express Brief, vol. 62, no. 4, pp. 372-376, April 2015.

[4] 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 (TCAS-II): Express Brief, vol. 63, no 2, pp. 176-180, February 2016.

[5] A. Liacha, A.K. Oudjida, F. Ferguene, M. Bakiri, and M.L. Berrandjia, "Design of High-Speed, Low-Power, and Area-Efficient FIR Filters," IET Circuits, Devices & Systems (IET-CDS), vol. 12, issue 1, pp.1-11, January 2018.

[6] 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.

[7] B. Khiter, A.K. Oudjida, and M. Belhocine, "H-Radix: a New Heuristic for Single Constant Multiplication," IET Circuits, Devices & Systemes (IET-CDS), vol. 11, issue 3, pp. 256-260, May 2017.

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

[9] L. Aksoy, E. da Costa, P. Flores, and J. Monteiro, "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 (TCAD), vol. 27, no. 6, pp. 1013-1026, June 2008.

[10] P. Barth, "A Davis–Putnam based enumeration algorithm for linear pseudo-Boolean optimization," Max-Planck-Institut Für Informatik, Saarbrücken, Germany, Tech. Rep. MPI-I-95-2-003, Jan. 1995.

[11] Mixed Integer Programming (MIP) Solver, Version LPSolve IDE v5.5.2.3, 2016. Available at : http://groups.yahoo.com/group/lp_solve/

[12] G. W. Reitwiesner, "Binary arithmetic", in Advances in Computers, vol. 1, pp. 231–308, NewYork, NY: Academic Education Press, 1960.

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

[14] A.K. Oudjida, "Radix-2^r MCM Web Page," June 2015. Available at: http://www.cdta.dz/products/mcm/

[15] T. Larrabee, "Test pattern generation using Boolean satisfiability," IEEE Trans. on Computer-Aided Design of Integrated Circ. and Systems (TCAD)., vol. 11, no. 1, pp. 4–15, Jan. 1992.

[16] L. Aksoy, E.O. Günes, and P. Flores, "Search Algorithms for the Multiple Constant Multiplications Problem: Exact and Approximate," Microprocessors and Microsystems Journal, Elsevier, vol. 34, no 5, pp. 151-162, August 2010.

[17] C.H. Chang and M. Faust, "On "A New Common Subexpression Elimination Algorithm for Realizing Low-Complexity Higher Order Digital Filters"," IEEE Trans. on Computer-Aided Design of Integrated Circ. and Systems (TCAD), vol. 29, no. 5, pp. 844–848, May 2010.

[18] Nanyang Technological University, Singapor, "FIRsuite Suite of Constant Coefficient FIR Filters," November 2009. Available at: http://www.firsuite.net

[19] 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.

[20] Ye, W.B., Yu, Y.J.: 'Bit-Level Multiplierless FIR Filter Optimization Incorporating Sparce Filter Technique', IEEE Trans. Circuits and Systems I: Regular Papers, 2014, 61, (11), pp. 3206-3205.