Cruz Jiménez et al. EURASIP Journal on Advances in Signal Processing (2017) 2017:31
DOI 10.1186/s13634-017-0466-z

EURASIP Journal on Advances in Signal Processing

RESEARCH Open Access

Theoretical lower bounds for parallel pipelined shift-and-add constant multiplications with *n*-input arithmetic operators

Miriam Guadalupe Cruz Jiménez[1*], Uwe Meyer Baese[2] and Gordana Jovanovic Dolecek[1]

Abstract

New theoretical lower bounds for the number of operators needed in fixed-point constant multiplication blocks are presented. The multipliers are constructed with the shift-and-add approach, where every arithmetic operation is pipelined, and with the generalization that *n*-input pipelined additions/subtractions are allowed, along with pure pipelining registers. These lower bounds, tighter than the state-of-the-art theoretical limits, are particularly useful in early design stages for a quick assessment in the hardware utilization of low-cost constant multiplication blocks implemented in the newest families of field programmable gate array (FPGA) integrated circuits.

Keywords: SCM, MCM, FPGA, Multiplication, Lower bound

1 Introduction

Multiplication with constants is a regular operation in digital signal processing (DSP) systems. In hardware, a multiplication is demanding in terms of area and power consumption. However, the single constant multiplication (SCM) and multiple constant multiplication (MCM) operations can be implemented by using only shifts, additions, and subtractions, with the last two being usually referred in general form as additions [1]. The SCM case is when an input is multiplied by a constant coefficient (Fig. 1a), and the MCM operation is when an input is multiplied by a set of constant coefficients (Fig. 1b) [2]. Theoretical lower bounds for the number of adders and for the number of depth levels, i.e., the maximum number of serially connected adders (also known as the critical path), in SCM, MCM, and other constant multiplication blocks that are constructed with two-input adders under the shift-and-add scheme have been presented in [3]. Tighter lower bounds, as well as a new bound, namely, the one for the number of extra adders required to preserve the lowest number of depth levels, were presented in [4] for the SCM case. Nevertheless, there are no theoretical lower bounds for the case of constant multiplication blocks that include multiple input additions/subtractions and pipeline registers in the involved arithmetic operations. However, this type of operations has become very important mainly when the pipelined constant multiplication blocks are implemented in the increasingly demanded field programmable gate array (FPGA) platforms. This is due to the fact that logic blocks of FPGAs include memory elements, and thus, pipelining results in low extra cost [5–12]. Currently, the use of three-input adders has started to gain importance, since the logic blocks of the newest families of FPGAs are bigger and allow to fit more complex adders using nearly the same amount of hardware resources [10–12].

Particularly, in the last two decades, many efficient high-level synthesis algorithms have been introduced for the multiplierless design of constant multiplication blocks. The common cost function to be minimized in these algorithms is given by the number of arithmetic operations (additions and subtractions) needed to implement the multiplications. Nevertheless, the critical path has the main negative impact in the speed and power consumption [13–18]. Therefore, substantial research activity has been carried out currently targeting both, application-specific integrated circuits (ASICs) [19–21]

* Correspondence: miriam.gcj@gmail.com
[1]Department of Electronics, Institute INAOE, Tonantzintla, Puebla, México
Full list of author information is available at the end of the article

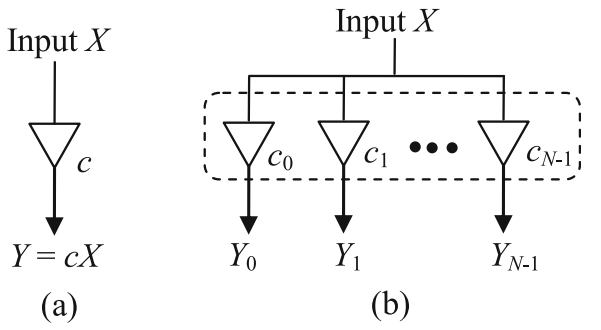

Fig. 1 Block diagram **a** SCM and **b** MCM

and FPGAs [5–10, 22–25], where the minimization of the number of arithmetic operations subject to a minimum number of depth levels is the ultimate goal.

On the other hand, even though ASICs still provides higher performance and low power consumption, the increased development time and manufacturing cost which comes with smaller CMOS transistor technologies have opened a large market for FPGAs. The FPGA technology provides the signal processing engineers with the ability to construct a custom data path that is tailored to the application at hand [26, 27]. FPGAs offer the flexibility of instruction set digital signal processors, while providing the processing power and flexibility of an ASIC, and enable significant design cycle compression and time-to-market advantages, an important consideration in an economic climate with ever-decreasing market windows and short product life cycles [28, 29].

The novelty of this paper is to introduce the theoretical lower bounds for the number of operations necessary to implement pipelined single constant multiplication (PSCM) and pipelined multiple constant multiplication (PMCM) blocks that are constructed with the shift-and-add scheme. For the derivation of these bounds, we consider that either an n-input (where n is an integer) pipelined addition/subtraction or a single pipeline register have the same cost. As mentioned earlier, recently, this assumption fits particularly well for cases where n is set equal to 3 and the target platforms for implementation are the newest FPGAs from the two most dominant manufacturers, Xilinx and Altera. However, it is worth highlighting that $n = 2$ is still under common use in many applications. This contribution is important because the optimality of different algorithms that reduce the number of operations in PSCM and PMCM blocks can be tested using appropriate theoretical lower bounds. Additionally, these bounds can be useful to develop new algorithms.

The paper is organized as follows. In the next section, definitions and methods needed to address the proposal are given. Section 3 presents the new theoretical lower bounds along with theorems and proofs to support the derivation of these bounds. Comparisons with previous theoretical lower bounds from [3] and [4] are provided in Section 4. Finally, conclusions are given in Section 5.

2 Definitions of terms

The constant multiplications referred here are expressed in fixed-point arithmetic because implementations in this number representation have higher speed and lower cost, thus being usually employed in DSP algorithms [1–25, 30–40]. Only integer, positive, odd constants are considered since this is a useful simplification that does not affect the formulation of constant multiplication problems. In this sense, a constant can be expressed simply in binary form, as follows:

$$c = \sum_{i=0}^{B-1} b_i 2^i, \tag{1}$$

where $b_i \in \{0, 1\}$ is the i-th bit and B is the word length [31]. We can express a product of a variable input X by a constant c with the shift-and-add approach using the binary representation of that constant to dictate the multiplier structure. For example, the product $47X$, with $47 = 2^5 + 2^3 + 2^2 + 2^1 + 2^0$ (i.e., a binary string "101111"), needs four additions and has a critical path of three additions, as show in Fig. 2. The implementation cost of a shift-and-add constant multiplier is the number of arithmetic operations since products by powers of two are implemented as hardwired shifts with no practical cost.

It is worth to highlight that additions and subtractions require practically equal amount of resources in hardware implementation. Hence, signed digit (SD) representations of a constant can reduce the aforementioned implementation cost because they employ negative

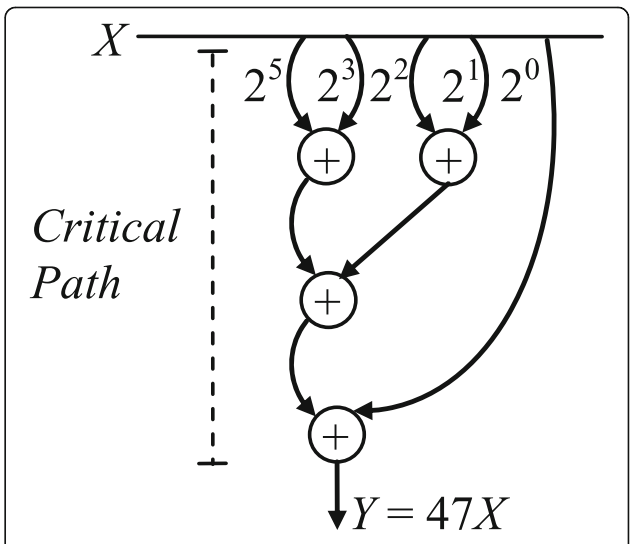

Fig. 2 Implementation structure of the product $47X$ with constant 47 expressed in binary using $n = 2$ input adders

digits, which represent subtractions. An SD representation of a constant is given in the form,

$$c = \sum_{i=0}^{B-1} d_i 2^i, \tag{2}$$

where $d_i \in \{-1, 0, 1\}$, with "−1" usually expressed as $\overline{1}$[32]. Among them, the canonical signed digit (CSD) representation is convenient since its number of non-zero digits is the *minimum number of signed digits* (MNSD) [3]. Besides, each non-zero digit is followed by at least one zero, which makes the representation unique. The CSD form of a constant can be found from binary by iteratively substituting every string of k digits "1" (say, "1111") with a string of $k-1$ digits "0" between a "1" and a "−1" (the string 1111 becomes "$1000\overline{1}$"). In this case, the product $47X$, with $47 = 2^6 - 2^4 - 2^0$ (i.e., a CSD string "$10\overline{1}000\overline{1}$"), needs two subtractions and has two operations in its critical path, as shown in Fig. 3.

In a constant multiplication block, the A-operation [30] represents two-input addition or subtraction along with shifts, and it is defined as

$$A_q(u_1, u_2) = \left|2^{l_1}u_1 + (-1)^{s_2}2^{l_2}u_2\right|2^{-r}, \tag{3}$$

where $l_1 \geq 0$ and $l_2 \geq 0$ are left shifts, $r \geq 0$ is a right shift, s_2 is a binary value, i.e., $s_2 \in \{0,1\}$, q is the set of parameters (so-called the configuration) of the A-operation, i.e., $q = \{l_1, l_2, r, s_2\}$, and u_1 and u_2 are odd integers. For three-input adders the A-operation is [10]

$$A_q(u_1, u_2, u_3) = \left|2^{l_1}u_1 + (-1)^{s_2}2^{l_2}u_2 + (-1)^{s_3}2^{l_3}u_3\right|2^{-r}, \tag{4}$$

where $l_1 \geq 0$, $l_2 \geq 0$, and $l_3 \geq 0$ are left shifts, $r \geq 0$ is a right shift, s_2 and s_3 are binary values, $q = \{l_1, l_2, l_3, s_2, s_3, r\}$ is the configuration of the A-operation, and u_1, u_2, and u_3 are odd integers. Generalizing to n-inputs, the A-operation is expressed as

$$A_q(u_1, \ldots, u_n) = \left|2^{l_1}u_1 + \sum_{i=2}^{n}(-1)^{s_i}2^{l_i}u_i\right|2^{-r}, \tag{5}$$

where $l_1 \geq 0, \ldots, l_n \geq 0$ are left shifts, $r \geq 0$ is a right shift, $s_2, \ldots, s_n$ are binary values, $q = \{l_1, \ldots, l_n, s_2, \ldots, s_n, r\}$ is the configuration of the A-operation, and $u_1, \ldots, u_n$ are odd integers.

An array of interconnected A-operations forms a SCM or a MCM block. The MCM is built upon SCM because the latter is the simplest case. The SCM array is represented using directed acyclic graphs (DAGs) with the following characteristics [33–36]:

- The output of each A-operation is called *fundamental*.
- For a graph with m A-operations, there are $m + 1$ vertices and m fundamentals.
- Each vertex has an in-degree n, except for the input vertex which has in-degree zero.
- A vertex with in-degree n corresponds to an n-input A-operation.
- Each vertex has out-degree larger than or equal to one except for the output vertex which has out-degree zero.
- The constant resulting from the last A-operation is *output fundamental* (OF). The constants resulting from previous A-operations are *non-output fundamentals* (NOFs).

In the MCM case, there are several OFs.

The DAG representation is the most useful for saving arithmetic operations because it allows to exploit structures to interconnect A-operations that cannot be seen in the CSD representation. This expands the opportunity to optimize the constant multiplication blocks. For example, the product $45X$, with $45 = 2^6 - 2^4 - 2^2 + 2^0$ (i.e., a CSD string "$10\overline{1}0\overline{1}01$"), needs three 2-input additions and has a critical path of two additions, as show in Fig. 4a. However, by using the DAG approach, the multiplication $45X$ requires two 2-input additions and has a critical path of two additions. In this case, it is possible to factorize the constant in two factors, namely, 5 and 9, as shown in Fig. 4b.

It is important to mention that a *multiplicative graph* is the graph obtained by cascading subgraphs, and the union point between two cascaded subgraphs in a multiplicative graph is called *articulation point* [37]. This is illustrated in Fig. 5a. A particular case is the *completely multiplicative graph*, where each cascaded subgraph is composed by one A-operation, as shown in Fig. 5b [4]. The graph presented in Fig. 4b is an example of a

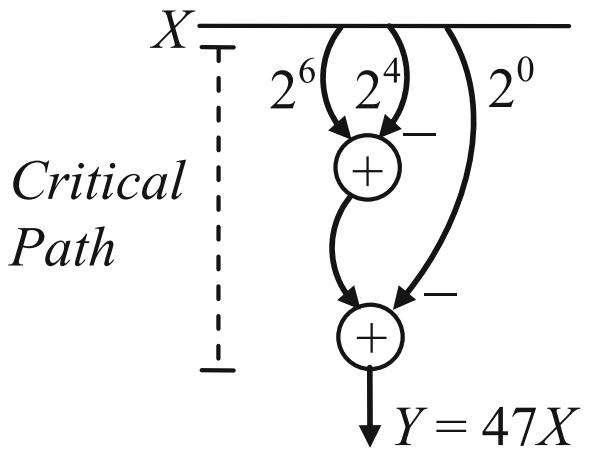

Fig. 3 Implementation structure of the product $47X$ with constant 47 expressed in CSD using two input adders

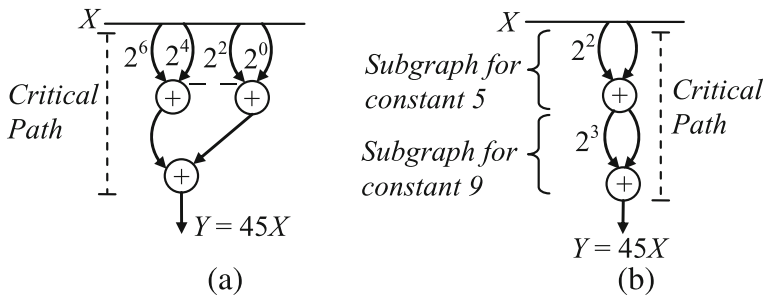

Fig. 4 Structure of the product $45X$ **a** constant 45 expressed in CSD and **b** constant 45 in graph representation using two input adders

completely multiplicative graph with 2-input A-operations. Other graphs without articulation points are referred as *non-multiplicative graphs* [37]. A cascaded interconnection of a completely multiplicative graph with a non-multiplicative graph is called *generalized graph*, see Fig. 5c.

The speed of a design is restricted by the critical path. The pipelining technique allows the reduction of a critical path introducing registers along the data path [38]. In FPGA implementations, the constant multiplications involving shifts-and-add operations can be made fully pipelined with a low extra cost. Pipelining has a small overhead due to the fact that the logic blocks in FPGAs include memory elements, which are otherwise unused [39, 40]. For example, Table 1 shows the amount of logic elements used to implement the multiplier $45X$ (for an 8-bit input) in an Altera Cyclone IV EP4CE115F29C7 FPGA. We observe that only three extra logic elements are needed in the pipelined implementation, which represents an increase of 9.7% in resources utilization compared with the non-pipelined case. Nevertheless, the frequency of operation is increased by 31.7%.

Due to the aforementioned observation, the implementation cost will be accounted by the number of registered operations (R-operations), i.e., either an addition-register pair or a single register, needed to implement constant multiplications. Two R-operations with the same cost are illustrated in a simplified way in Fig. 6. Hence, the PSCM problem consists in finding the pipelined array of A-operations that form a single-constant multiplier using the minimum number of R-operations. Similarly, the PMCM problem consists in finding the pipelined array of A-operations that form a multiple-constant multiplier using the minimum number of R-operations.

To calculate the lower bounds for the number of R-operations required to implement PSCM and PMCM blocks, we need the following information from a constant:

1) Its MNSD, denoted by S. We will also refer to this number in a more informal manner as "the number of non-zero digits".
2) Its number of prime factors (it does not matter if these prime factors are repeated). This number is denoted by Ω.

3 Proposed lower bounds

In the following, we state, in Subsection 3.1, Theorems 1 to 8 to derive the lower bounds of R-operations in

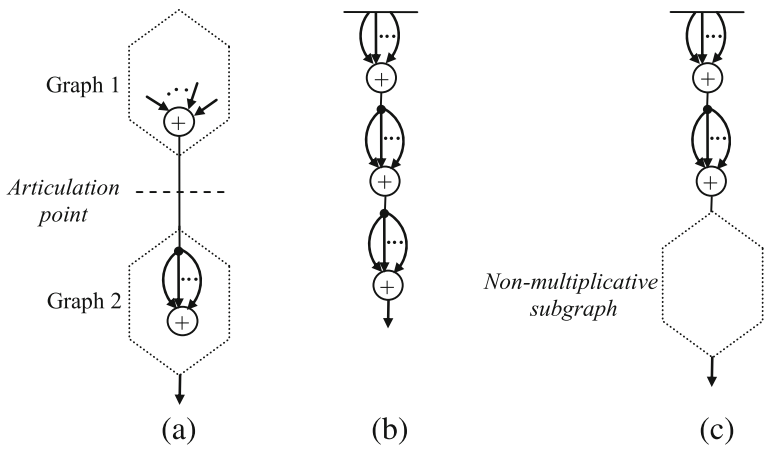

Fig. 5 **a** multiplicative graph, **b** completely multiplicative graph, and **c** generalized graph

Table 1 Synthesis results of pipelined and non-pipelined implementations of a 45*X* multiplier in the Altera Cyclone IV EP4CE115F29C7 FPGA

Pipelined	Total logic elements (LE)	Maximum frequency of operation (MHz)	Area × Time cost metric (LE/MHz)
No	31	285.47	0.1086
Yes	34	376.08	0.0904

PSCM and, in Subsection 3.2, Theorems 9 and 10 for PMCM, along with their corresponding proofs. The pipelining operation, which has not been alluded in the previous works [3] and [4], is explicitly included in the proposed lower bounds through the *R*-operations.

3.1 PSCM case

Whenever a constant c is mentioned in the theorems of this sub-section (Theorems 1 to 8), we consider that the MNSD of that constant is S and its number of prime factors is Ω.

Theorem 1 provides the upper limit of non-zero digits that can be generated by any graph with a given number of depth levels, regardless of its number of *R*-operations. From this, we can know the minimum number of depth levels that a graph must have to implement a constant with a given S.

Theorems 2 and 3 prove the properties of the completely multiplicative graphs, namely, generating the upper limit of non-zero digits mentioned in Theorem 1 with the minimum possible number of *R*-operations. From them, we have that the completely multiplicative graph is a solution with the lower bound for the number of *R*-operations. However, as it is known, this graph has articulation points, and every articulation point represents the union between two cascaded subgraphs, i.e., the product of two smaller constants. Therefore, Theorem 4 uses Ω to identify what constants can be implemented with the completely multiplicative graph (for example, prime constants cannot be factorized into smaller constants; thus, they cannot be implemented by a completely multiplicative graph).

Theorem 5 identifies the minimum number of *R*-operations needed in any non-multiplicative graph with a given number of depth levels, and Theorem 6 proves that non-multiplicative graphs can generate the upper limit of non-zero digits mentioned in Theorem 1 with its minimum number of *R*-operations. Then, Theorem 7 establish the lower bound for the number of *R*-operations needed to implement a prime constant ($\Omega = 1$).

Finally, Theorem 8 completes the information of Theorems 4 and 7, namely, the lower bound of *R*-operations needed to implement non-prime constants that have fewer number of factors than the number of subgraphs used in a completely multiplicative graph.

3.1.0.1 Theorem 1 *A graph with p depth levels can provide at most n^p non-zero digits for a constant.*

3.1.0.2 ***Proof*** The proof is given by induction (see proof of Theorem 6.9 in [39] for the case of 2-input *A*-operations):

1) The base case corresponds to the first depth level, where a n-input *A*-operation can form a constant with at most n non-zero digits. This is true since the input of any graph has one non-zero digit [3, 4, 39].
2) As inductive step, we assume that, in the p-th level, there are n^p non-zero digits at most. In the $(p+1)$-th level, an *A*-operation can form a constant whose number of non-zero digits is the sum of the numbers of non-zero digits at every input of that *A*-operation. This is at most n times the maximum number of non-zero digits available in the previous level, i.e., $n \times n^p = n^{p+1}$ non-zero digits.

Since assuming that the theorem is true for p implies that the theorem is also true for $p+1$, and since the base case is also true, the proof is complete. An adder, regardless of its number of inputs, cannot generate more non-zero digits than the sum of the numbers of non-zero digits in every one of its inputs. Thus, the MNSD can be, at most, n-plicate if the inputs of the n-input adder placed in any depth level come from the immediately previous depth level. ■

3.1.0.3 Theorem 2 *A completely multiplicative graph with p A-operations can generate n^p non-zero digits.*

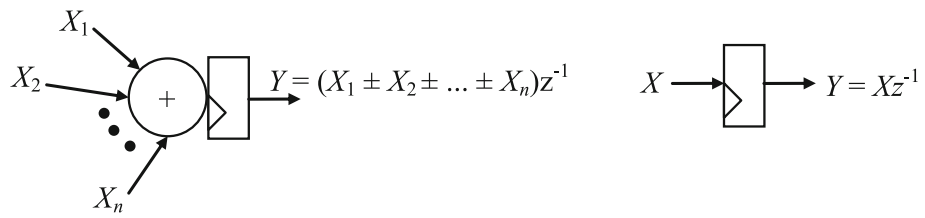

Fig. 6 *R*-operations with the same cost

3.1.0.4 ***Proof*** This proof is an straightforward extension of the proof of Theorem 6.8 in [39], which corresponds to completely multiplicative graphs with 2-input A-operations. As stated earlier, the input of a graph has one non-zero digit. In the completely multiplicative graph, there are at most n non-zero digits after the A-operation placed at the 1st depth level. Cascading an A-operation to that output yields at most $n \times n$ non-zero digits, and so on. The number of non-zero digits at the depth level p is at most the n-tuple of the number of non-zero digits of a fundamental at the $(p-1)$-th depth level. Consequently, the maximum number of non-zero digits at the p-th depth level is n^p. ■

3.1.0.5 Theorem 3 *A completely multiplicative graph with p depth levels needs only p R-operations.*

3.1.0.6 ***Proof*** The completely multiplicative graph with p depth levels has p A-operations, and every A-operation forms a subgraph. Pipelining between two subgraphs needs only one register, according to [38], because the pipelining occurs on the articulation point. This results in every A-operation being followed by a register. Since an A-operation followed by a register is considered an R-operation, there are only p R-operations in total. This is illustrated in Fig. 7. ■

3.1.0.7 Theorem 4 *A constant with* $(n^{p-1}+1) \le S \le n^p$ *and* $\Omega \ge p$ *needs at least p R-operations.*

3.1.0.8 ***Proof*** From Theorem 2, we have that a constant with $(n^{p-1}+1) \le S \le n^p$ non-zero digits can be implemented with at least p depth levels, which implies at least p A-operations. From Theorem 3, we have that a completely multiplicative graph can generate those values for S with only p R-operations. The completely multiplicative graph with p R-operations consists of p cascaded subgraphs; thus, a constant implemented with that graph must have at least p prime factors. Since $\Omega \ge p$ holds, the completely multiplicative graph can be employed to implement that constant using p R-operations. ■

3.1.0.9 Theorem 5 *A non-multiplicative graph with p depth levels needs at least (2p – 1) R-operations.*

3.1.0.10 ***Proof*** According to Theorem 3, if a graph with p depth levels has only p R-operations in total, it must be a pipelined completely multiplicative graph. According to Theorem 2, that graph can generate the maximum possible number of non-zero digits, namely, n^p. To make non-multiplicative that optimal graph, the $(p-1)$ articulation points must be eliminated. From [38], it is known that at least one additional R-operation must be added for every eliminated articulation point. Therefore, at least $(2p-1)$ R-operations are required, i.e., the original p minimum number of R-operations in the form of addition-delay pairs plus the additional $(p-1)$ R-operations in the form of pure delays. Figure 8 shows an example with $p = 3$. ■

3.1.0.11 Theorem 6 *A non-multiplicative graph with p depth levels and (2p – 1) R-operations can generate* n^p *non-zero digits.*

3.1.0.12 ***Proof*** Consider a graph with p depth levels formed by two completely multiplicative graphs of $(p-$

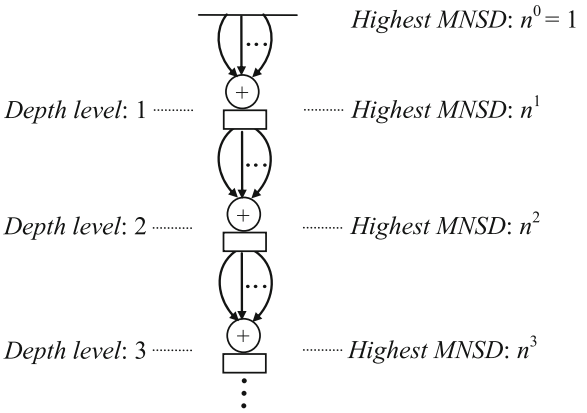

Fig. 7 The pipelined completely multiplicative graph achieves n^p non-zero digits with the minimum number of n-input R-operations, p, and the minimum number of depth levels, p

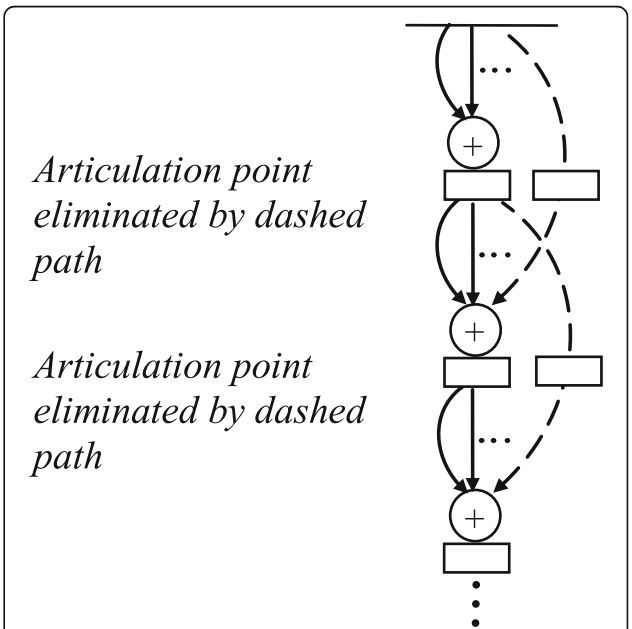

Fig. 8 Non-multiplicative graph with $p = 3$ depth levels and $p - 1$ extra R-operations in the form of pure delay

1) levels each, connected in parallel from the input of the graph, and one A-operation placed in the p-th level summing up the outputs of the aforementioned graphs. The output of one of these graphs is connected to the $n-1$ inputs of the last A-operation, and the output of the other graph is connected to the remaining input of the last A-operation. This is a non-multiplicative graph because it is not formed by cascading subgraphs, and it is composed by $(2p-1)$ A-operations. According to Theorem 2, we can obtain n^{p-1} non-zero digits from the completely multiplicative graphs, and according to Theorem 3, these graphs can be pipelined without requiring extra registers. Since the last A-operation can add n times the n^{p-1} non-zero digits in each one of its inputs and can be pipelined without extra cost, the resulting graph generates n^p non-zero digits using $(2p-1)$ R-operations. An example of this is shown in Fig. 9. ■

3.1.0.13 Theorem 7 *A constant with $(n^{p-1}+1) \le S \le n^p$ and $\Omega = 1$ needs at least $2p-1$ R-operations.*

3.1.0.14 *Proof* Since $\Omega = 1$ holds, the non-multiplicative graph must be employed to implement that constant. From Theorem 6, we have that a constant with $(n^{p-1}+1) \le S \le n^p$ non-zero digits can be implemented with at least p depth levels and at least $2p-1$ R-operations. This is a lower bound for the number of R-operations, since from Theorem 5, we have that a non-multiplicative graph with p-levels needs at least $2p-1$ R-operations. ■

3.1.0.15 Theorem 8 *A constant with $(n^{p-1}+1) \le S \le n^p$ and $1 < \Omega < p$ needs at least $(2p-\Omega)$ R-operations.*

3.1.0.16 *Proof* From Theorem 1, we have that p depth levels are necessary to achieve the values of S in the specified range. Since $\Omega < p$ holds, we can take advantage of a completely multiplicative graph with $\Omega-1$ R-operations at most, which, according to Theorem 2, generates $n^{\Omega-1}$ non-zero digits at most, and represents the product of $\Omega-1$ factors. The last factor can be formed with a non-multiplicative subgraph with $[p-(\Omega-1)]$ depth levels. According to Theorem 5, this subgraph needs at least $2[p-(\Omega-1)]-1$ R-operations, and according to Theorem 6, it can generate $n^{[p-(\Omega-1)]}$ non-zero digits. The total graph, illustrated in Fig. 10, can generate at most $n^{\Omega-1} \times n^{[p-(\Omega-1)]} = n^p$ non-zero digits and uses at least $(\Omega-1)+2[p-(\Omega-1)]-1 = 2p-2(\Omega-1)+(\Omega-1-1 = 2p-(\Omega-1)-1 = (2p-\Omega)$ R-operations. ■

Finally, from Theorem 1, we have that the number of depth levels necessary to achieve S is $p = \lceil \log_n(S) \rceil$. Substituting this value for p and using Theorems 4, 7, and 8, we obtain the lower bound for the number of R-operations needed to form a PSCM block as follows:

$$L_{SCM} = \begin{cases} 2\lceil \log_n(S) \rceil - \Omega; & \Omega < \lceil \log_n(S) \rceil, \\ \lceil \log_n(S) \rceil; & \Omega \ge \lceil \log_n(S) \rceil. \end{cases} \quad (6)$$

3.2 PMCM case

The theorems in this section are stated for N constants $c_1, c_2, \ldots, c_N$, whose respective MNSDs are $S_1, S_2, \ldots, S_N$, and their respective numbers of prime factors are $\Omega_1, \Omega_2, \ldots, \Omega_N$, such that $S_1 \le S_2 \le \ldots \le S_N$.

Theorem 9 indicates the lower bound for the number of n-input A-operations needed to form an MCM block. If pipelining is added, more R-operations than the aforementioned lower bound may be needed because the constants with fewer prime factors may use non-

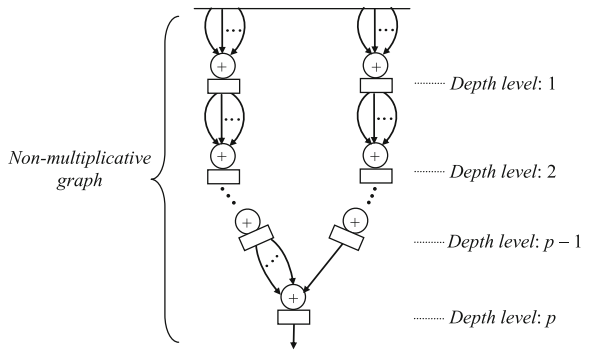

Fig. 9 Non-multiplicative graph that generates the maximum number of non-zero digits, n^p, with the minimum number of R-operations in non-multiplicative graphs

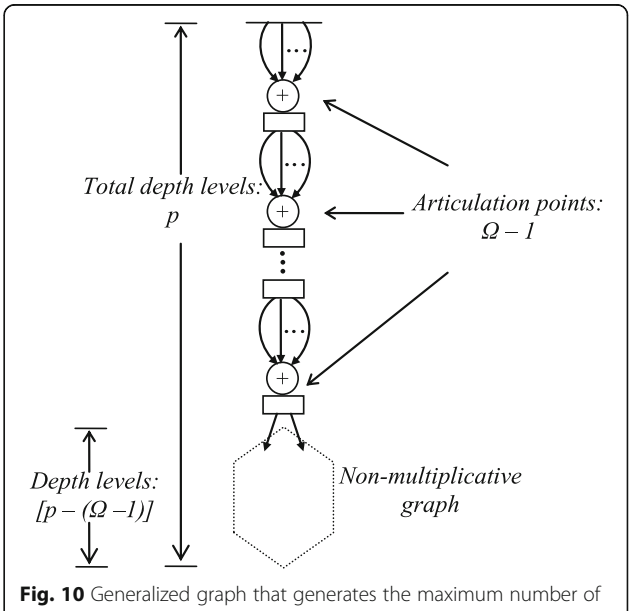

Fig. 10 Generalized graph that generates the maximum number of non-zero digits, n^p, with the minimum number of R-operations in a multiplicative graph for constants with less prime factors than the minimum number of depth levels

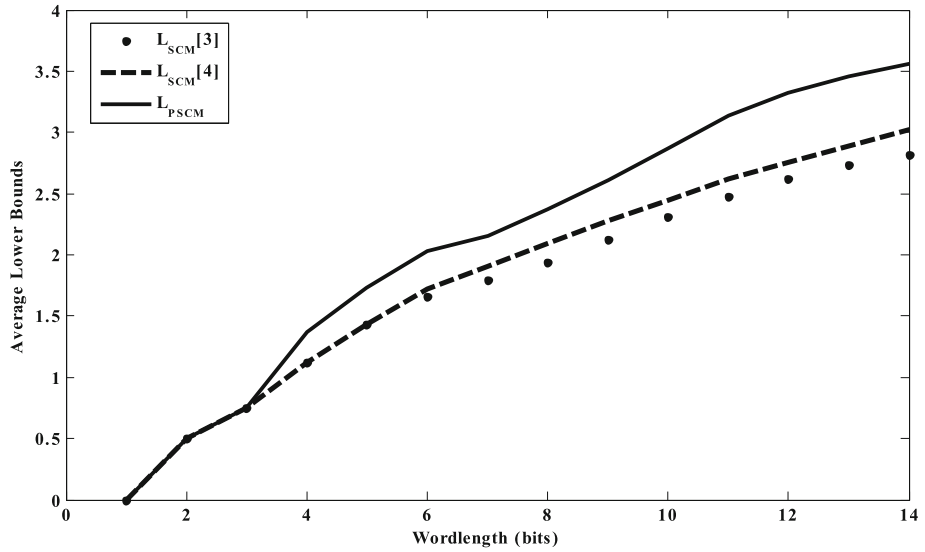

Fig. 11 Average lower bounds

multiplicative graphs, which require extra *R*-operations (see Theorems 5 to 8). Besides, all the outputs of the PMCM block must have equal number of depth levels to balance the input–output delay, which also may require extra *R*-operations. Based on these observations, Theorem 10 extends the lower bound provided in Theorem 9 by identifying at least how many extra *R*-operations would be needed. From these theorems, we obtain the lower bound for the number of *R*-operations needed to form a PMCM block.

3.2.0.1 Theorem 9 *At least K n-input A-operations are needed to build an MCM block, where K is given by*

$$K = \lceil \log_n(S_1) \rceil + \sum_{i=1}^{N-1} E(S_i, S_{i+1}), \tag{7}$$

$$\text{with } E(S_i, S_{i+1}) = \begin{cases} 1; & S_i = S_{i+1}, \\ \left\lceil \log_n \frac{S_{i+1}}{S_i} \right\rceil; & S_i < S_{i+1} \end{cases}.$$

3.2.0.2 ***Proof*** Recall that every *A*-operation has only one possible configuration and therefore can generate only one fundamental. Simply shifted (i.e., scaled by a power of two) versions of that fundamental can be obtained from that *A*-operation. Since the target constants are integer and odd by definition, it is not possible to obtain two target constants from the same *A*-operation. Therefore, there must be at least *N n*-input *A*-operations for the *N* constants. Note that, since the terms S_i are sorted in ascendant order, S_1 corresponds to the simplest constant, i.e., the one with the smallest number of non-zero digits. From Theorem 1, we have that with *p* depth levels we can obtain n^p non-zero digits at most. By using the relation $n^p \geq S_1$, we have that the minimum number of levels necessary to generate S_1 non-zero digits is $\lceil \log_n(S_1) \rceil$, which implies the existence of at least $\lceil \log_n(S_1) \rceil$ *A*-operations for that constant. Finally, if $S_{i+1} > n \times S_i$ holds, we have that a single *A*-operation is not able to generate the constant c_{i+1} if there are only coefficients with at most S_i digits available because the number of non-zero digits at the output of an *A*-operation is at most the sum of the number of non-zero digits at its inputs. Therefore, at least $\lceil \log_n(S_{i+1}/S_i) \rceil$ *A*-operations will be required. This proof is a straightforward extension of the proof given in [3] for the lower bound of 2-input *A*-operations that form an MCM block. ■

Table 2 Percentage of constants with improved lower bounds

Word length	L_{SCM} [3]	L_{SCM} [4]
$B = 14$ bits	54%	45%
$14 < B \leq 32$	63%	55%

3.2.0.3 Theorem 10 *At least L R-operations are needed to build a PMCM block, where L = K + F + G, with*

$$F = \begin{cases} \max_i \{\lceil \log_n(S_i) \rceil - \Omega_i\}; & \forall\, i \text{ such that } \Omega_i < \lceil \log_n(S_i) \rceil, \\ 0; & \text{otherwise.} \end{cases}$$

$$G = \sum_{i=1}^{N-1} \lceil \log_n(S_N) \rceil - \lceil \log_n(S_i) \rceil$$

and K given in (7).

3.2.0.4 ***Proof*** Consider that there is a constant c_m that satisfies $\Omega_m < \lceil \log_n(S_m) \rceil$ and, if there are more constants that satisfy such condition, c_m has the greatest

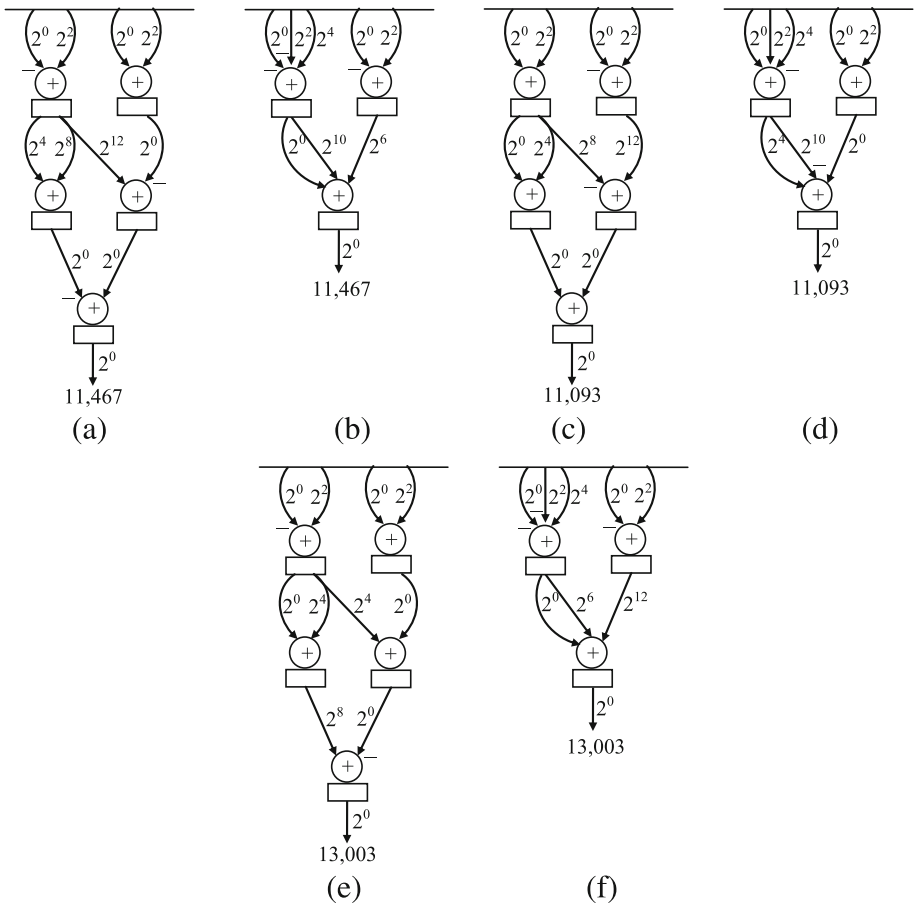

Fig. 12 **a** Two-input adder graph of constant 11,467, **b** three-input adder graph of constant 11,467, **c** two-input adder graph of constant 11,093, **d** three-input adder graph of constant 11,093, **e** two-input adder graph of constant 13,003, and **f** three-input adder graph of constant 13,003

difference $[\lceil \log_n(S_m) \rceil - \Omega_m]$. From Theorem 8, we have that the constant can be formed by cascading a non-multiplicative graph with a completely multiplicative graph, where the non-multiplicative graph needs $2[\lceil \log_n(S_m) \rceil - (\Omega_m - 1)] - 1$ R-operations. Since Theorem 9 has not taken into consideration the number of prime factors, only $[\lceil \log_n(S_m) \rceil - (\Omega_m - 1)]$ A-operations have been accounted in that theorem, under the assumption that the constant c_m can be constructed with the optimal completely multiplicative graph. Therefore, at least $[\lceil \log_n(S_m) \rceil - (\Omega_m - 1)] - 1$ extra R-operations must be included when pipelining is applied, which explains the term F. The term G is explained by the fact that extra R-operations may be needed to achieve the same number of pipelined stages from input to output in every constant. Since the minimum depth level of a constant is given by $\lceil \log_n(S) \rceil$, the differences between the minimum depth level of the constant c_N (which has the greatest depth level among other constants) and the minimum depth levels of the other constants are accumulated in the term G. ■

From Theorem 10, we can express the lower bound for the number of R-operations in the PMCM case as

Table 3 Number of R-operations for Example 1 using $n = 2$ and $n = 3$ input adders

Constant	R-operations $n = 2$			R-operations $n = 3$
	L_{SCM} [3]	L_{SCM} [4]	L_{PSCM}	L_{PSCM}
11,467	3	4	5	3
11,093	3	4	5	3
13,003	3	4	5	3

Table 4 Resulting R-operations for Example 2 using $n = 2$ input adders

Algorithm	R-operations
H_{cub} ADmin (method [30] with additional pipelining)	7
PAG using ASAP pipelining (preliminary solution from [8])	7
Optimal PAG (method [8])	5
L_{MCM} [3]	3
L_{PMCM}	4

Table 5 Resulting *R*-operations for Example 3 using $n = 2$ input adders

Algorithm	*R*-operations
RAG-n (method [36] with additional pipelining)	13
RSG (method [22])	7
OFL (method [7])	6
L_{MCM} [3]	4
L_{PMCM}	6

$$L_{PMCM} = \lceil \log_n(S_1) \rceil + \sum_{i=1}^{N-1} \left(\lceil \log_n(S_N) \rceil - \lceil \log_n(S_i) \rceil \right) + \sum_{i=1}^{N-1} E(S_i, S_{i+1}) + F, \tag{8}$$

$$\text{with } E(S_i, S_{i+1}) = \begin{cases} 1; & S_i = S_{i+1}, \\ \left\lceil \log n \frac{S_{i+1}}{S_i} \right\rceil; & S_i < S_{i+1}, \end{cases}$$

and

$$F = \begin{cases} \max_i \left\{ \lceil \log_n(S_i) \rceil - \Omega_i \right\}; & \forall\, i \text{ such that } \Omega_i < \lceil \log_n(S_i) \rceil, \\ 0; & \text{otherwise.} \end{cases}$$

4 Results and comparisons

In this section, comparisons of the proposed lower bounds with the lower bounds currently available in literature are presented, detailing PSCM and PMCM cases in Subsections 4.1 and 4.2, respectively. In all cases, two and three-input additions were considered.

First, the PSCM case is addressed for $n = 2$ (i.e., 2-input additions) with an illustration of the lower bounds averaged over all the constants with a word length of B bits, where B goes from 1 to 14. This illustration compares the proposed lower bound with the existing lower bounds from [3] and [4], showing that the proposed lower bound is tighter. An example is also included, where the pipelined shift-and-add multipliers for some constants are constructed with 2-input and 3-input additions.

The effectiveness of the PMCM lower bound is demonstrated by examples, where pipelined shift-and-add multiple constant multiplication blocks are constructed using the algorithms from [7, 8, 22, 30] and [36] for the case of 2-input additions and the algorithm from [10] for the case of 3-input additions. The proposed lower bound is compared with the lower bound from [3] in the case of 2-input additions, and in most of the cases, it provides better estimation of the number of required *R*-operations. For $n = 3$ (i.e., 3-input additions), there are no theoretical lower bounds currently available in literature. Thus, the proposed lower bound is only compared with the solution from [10]. In that case, the proposed lower bound falls short only by one *R*-operation.

Table 6 Resulting *R*-operations for Example 4 using $n = 2$ input adders

Algorithm	*R*-operations
PAG (method [8])	9
L_{MCM} [3]	4
L_{PMCM}	4

Table 7 Resulting *R*-operations for Example 4 using $n = 3$ input adders

Algorithm	*R*-operations
PAG for 3-input adders (method [10])	4
L_{PMCM}	3

4.1 PSCM case

The lower bounds from methods [3] and [4], as well as the proposed lower bound L_{PSCM} from (6) are averaged for all constants with B bits, where B is between 1 and 14. These averages are shown in Fig. 11. We can observe the tightening of the proposed lower bound, i.e., the proposed lower bound in general is greater than the lower bounds currently available in literature. Table 2 presents, for $n = 2$, the percentage of constants with improved lower bounds among 10,000 14-bits random constants and among 10,000 B-bits random constants, with B between 15 and 32.

Example 1 presents the pipelined shift-and-add multipliers for constants {11,467}, {11,093}, and {13,003} constructed with 2-input additions (shown in Fig. 12a, c, and e, respectively) and 3-input additions (shown in Fig. 12b, d and f, respectively). In all the cases, the optimal solutions have the number of *R*-operations predicted by the proposed lower bound, as shown in Table 3. Besides, for the case of two-input additions, the proposed lower bound outperforms the ones from [3] and [4] because the lower bound from [3] falls short by 2 *R*-operations and the lower bound from [4] falls short by one *R*-operation.

Example 1 The constants {11,467}, {11,093}, and {13,003} have similar graph and the same lower bounds as shown in Table 3. The corresponding graphs are presented in Fig. 12.

4.2 PMCM case

In Example 2, the multiplier block with constants {44; 130; 172}, formed with 2-input additions, is presented.

Table 8 Resulting *R*-operations for Example 5 using $n = 2$ input adders

Algorithm	*R*-operations
PAG (method [8])	8
L_{MCM} [3]	5
L_{PMCM}	6

Table 9 Resulting *R*-operations for Example 5 using $n = 3$ input adders

Algorithm	*R*-operations
PAG for 3-input adders (method [10])	3
L_{PMCM}	3

In Table 4, the number of *R*-operations obtained by the algorithms H_{cub} [30] with pipelining, PAG using ASAP pipelining [8], and the optimal PAG [8] are listed. Additionally, the lower bound of [3] and the proposed lower bound L_{PMCM} from (8) are given. The proposed lower bound is closer to the number of *R*-operations needed to implement the multiplier block than the lower bound of [3].

Example 3 presents the group of constants {3; 13; 21; 37} that form a multiplier block. The *R*-operations needed to implement the multiplier block using 2-input additions are obtained with the algorithms RAG-n [36] with pipelining, RSG [22], and OFL [7]. The resulting values are shown in Table 5, where it can be observed that the OFL algorithm offers the less number of *R*-operations. Also, the lower bound of [3] and the proposed lower bound L_{PMCM} from (8) are given in Table 5. In this example, the proposed lower bound estimates the same number of *R*-operations used by the OFL algorithm to implement the multiplier block.

A multiplier block formed with the constants {7,567; 20,406} is illustrated in Example 4. The *R*-operations needed to implement the multiplier block using 2-input additions are obtained with the algorithm PAG [8]. Table 6 shows the resulting number of *R*-operations together with the estimated number of *R*-operations using the lower bound of [3] and the proposed lower bound L_{PMCM} from (8). The *R*-operations needed to implement the multiplier block using 3-input additions are obtained with the algorithm PAG for 3-input additions [10]. Table 7 shows the resulting number of *R*-operations along with the estimations using the proposed lower bound L_{PMCM} from (8).

Finally, Example 5 presents the constants {87,381; 689,493} that form a multiplier block. The *R*-operations needed to implement the multiplier block using 2-input additions are obtained with the algorithm PAG [8], and the *R*-operations needed to implement the multiplier block using 3-input additions are obtained with the algorithm PAG for 3-input additions [10]. Table 8 shows the resulting number of *R*-operations together with the estimated number of *R*-operations using the lower bound of [3] and the proposed lower bound L_{PMCM} from (8). Table 9 shows the resulting number of *R*-operations along with the estimations using the proposed lower bound L_{PMCM} from (8). The proposed lower bound presents a reliable estimation of the number of *R*-operations needed to implement the multiplier block.

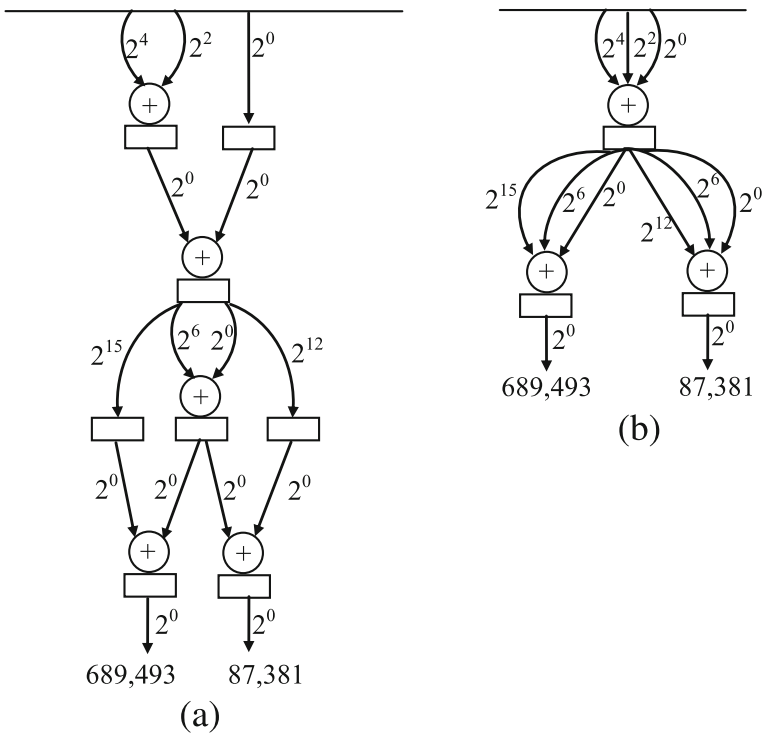

Fig. 13 **a** Two-input adder graph by PAG algorithm for the multiplier block {87,381; 689,493} and **b** three-input adder graph by PAG algorithm for the multiplier block {87,381; 689,493}

Example 2 (example given in [8]) A multiplier block with the constants from the set {44; 130; 172} have the estimate number of *R*-operations as shown in Table 4 (the resulting graphs are shown in Fig. 1 of paper [8]).

Example 3 (example given in [7]) A multiplier block with the constants from the set {3; 13; 21; 37} have the estimate number of *R*-operations as is shown in Table 5 (the resulting graphs can be seen in Fig. 4 of [7]).

Example 4 (example given in [10]) A multiplier block with the constants from the set {7,567; 20,406} have the estimate number of *R*-operations as shown in Table 6 for two-input adders and Table 7 for three-input adders (Fig. 3 of [10] shows the corresponding graphs).

Example 5 A multiplier block with the constants from the set {87,381; 689,493} have the estimate number of *R*-operations as shown in Table 8 for 2-input adders and Table 9 for 3-input adders. The corresponding graphs are shown in Fig. 13.

5 Conclusions

New theoretical lower bounds for the number of *R*-operations in the fully pipelined SCM and the fully pipelined MCM cases for *n*-input adders/subtractions have been presented. The proposed lower bounds are tighter because pipelining registers were explicitly considered. On the other hand, it was observed that the use of articulation points allows a rapid increase of the number of non-zero digits from a depth level to the next depth level. The new theoretical lower bounds achieve better estimation of the number of required operations needed to implement a single multiplier or a multiplier block. The tightening of the new lower bounds was illustrated with examples in the comparisons section.

Acknowledgements

This paper has been supported by CONACYT scholarship no. 224191. The authors are grateful to D. E. T. Romero for his helpful comments during the development of this proposal.

Funding

This work is a result of a doctoral thesis developed in the Institute INAOE; the thesis has been supported with CONACYT's grant.

Authors' contributions

MGCJ contributed to the main development of the theorems and examples in this proposal. UMB is the advisor in the development of low-complexity FPGA-based arithmetic blocks and contributed to the review of theorems and examples. GJD as thesis advisor directed all the work an the paper was written under her supervision. All authors read and approved the final manuscript.

Authors' information

Miriam Guadalupe Cruz Jimenez received the BS degree from the Minatitlan Institute of Technology and the MS degree from the National Institute for Astrophysics, Optics and Electronics (INAOE), Mexico. She received the best paper award at the conference CIIECC 2013. Currently, she is a PhD student in the Institute INAOE. She is a reviewer for the journals IEEE Transactions on Circuits and Systems I and Circuits, Systems & Signal Processing.

Dr. Uwe H. Meyer-Baese (IEEE, S'91-M'93) was born in Kassel, Germany, on July 10, 1964. He received his BSEE, MSEE, and Ph.D. "Summa cum Laude" from the Darmstadt University of Technology in 1987, 1989, and 1995, respectively. In 1994 and 1995, he held a Postdoctoral Position in "Institute of Brain Research," Magdeburg, Germany. In 1996 and 1997, he was a visiting professor at the University of Florida, Gainesville. From 1998 to 2000, he worked as a Research Scientist in the ASIC industry. He joint Electrical and Computer Engineering Department at the FAMU-FSU College of Engineering in 2001 and is now an Associate Professor. He holds 3 patents, has published over 100 journal and conference papers, 5 books, and supervised more than 60 master thesis projects in the real-time DSP/FPGA area. He is author of the best-selling Springer textbook on DSP with FPGAs. He was a recipient of the Max-Kade Award in Neuroengineering in 1997, ECE Department Research Award in 2005, Who's Who in Science member in 2005, SPIE, Best Presentation Award in 2006, FAMU-FSU College of Engineering Teaching Award in 2007, and the Humboldt Fellow in 2009. He has served as Faculty Senator of the FSU senate since Spring 2011. He has been an elected member of the editorial board for the journal Signal, Image and Video Processing for 2011–2015 and has been elected as a board member as well as an associate editor for the EURASIP Journal of Advances in Signal Processing for 2011–2013.

Gordana Jovanovic Dolecek received a BS degree from the Faculty of Electrical Engineering, University of Sarajevo, an Ms degree from the University of Belgrade, and a Ph.D. degree from the Faculty of Electrical Engineering, University of Sarajevo. She was with the Faculty of Electrical Engineering, University of Sarajevo until 1993, as a research assistant, assistant professor, associate professor, and full professor. From 1986 to 1991, she was chairman of the Department of Telecommunication. During 1993–1995, she was with the Institute Mihailo Pupin, Belgrade. In 1995, she joined Institute INAOE, Department for Electronics, Puebla, Mexico, where she works as a professor and researcher. She is the author of three books and more than 100 papers. She is also author of four lectures for TechOnLine University. Her research interests include digital signal processing and digital communications. She is a member of IEEE and The National Researcher System (SNI) of Mexico.

Competing interests

The authors declare that they have no competing interests.

Publisher's Note

Author details

[1]Department of Electronics, Institute INAOE, Tonantzintla, Puebla, México. [2]Electrical and Computer Engineering Department, Florida State University, Tallahassee, FL, USA.

Received: 26 January 2017 Accepted: 19 April 2017
Published online: 03 May 2017

References

1. R Guo, LS DeBrunner, K Johansson, Truncated MCM Using Pattern Modification for FIR Filter Implementation. Paper presented at the Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS), Paris, France, p. 3881–3884, May 30-Jun 2, 2010.
2. L Aksoy, EO Günes, P Flores, Search algorithms for the multiple constant multiplication problem: exact and approximate. Microprocess. Microsyst. 34(5), 151–162 (2010). doi: doi.org/10.1016/j.micpro.2009.10.001.
3. O Gustafsson, Lower bounds for constant multiplication problems. IEEE Trans. Circuits and Syst. II: Express briefs 54 (11), 974–978 (2007). doi: 10.1109/TCSII.2007.903212.
4. DET Romero, U Meyer-Baese, GJ Dolecek, On the inclusion of prime factors to calculate the theoretical lower bounds in multiplierless single constant multiplications. EURASIP Journal on Advances in Signal Processing 122, 1–9 (2014). doi:10.1186/1687-6180-2014-122.
5. S Mirzaei, R Kastner, A Hosangadi, Layout aware optimization of high speed fixed coefficient FIR filters for FPGAs. Int. Journal of Reconfigurable Computing (2010). doi: 10.1155/2010/697625.
6. M Kumm, P Zipf, High speed low complexity FPGA-based FIR filters using pipelined adder graphs. Paper presented at the Int. Conference on Field Programmable Technology (FPT), Indian Institute of Technology Delhi, New Delhi, India, p. 1–4, 12–14 December 2011.
7. U Meyer-Baese, G Botella, DET Romero, M Kumm, Optimization of high speed pipelining in FPGA-based FIR filter design using genetic algorithm. Proc. SPIE 8401, Independent Component Analyses, Compressive Sampling,

Wavelets, Neural Net, Biosystems, and Nanoengineering X, 84010R1-12 (2012). doi:10.1117/12.918934.
8. M Kumm, P Zipf, M Faust, CH Chang, Pipelined adder graph optimization for high speed multiple constant multiplication. Paper presented at the Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS), p. 49–52, COEX, Seoul, Korea, 20–23 May 2012.
9. M Kumm, D Fanghanel, K Moller, P Zipf, U Meyer-Baese, FIR filter optimization for video processing on FPGAs. EURASIP J Adv Sig Process **111**, 1–18 (2013). doi:10.1186/1687-6180-2013-111
10. M Kumm, M Hardieck, J Willkomm, P Zipf, U Meyer-Baese, Multiple constant multiplications with ternary adders. Paper presented at the International Conference on Field Programmable Logic and Applications (FPL), Porto, Portugal, p. 1–8, 2–4 Sept. 2013.
11. M Kumm, P Zipf, Pipelined compressor tree optimization using integer linear programming. Paper presented at the 24th International Conference on Field Programmable Logic and Applications (FPL), p. 1–8, Munich, Germany, 2–4 Sept. 2014.
12. M Kumm, P Zipf, Efficient high speed compression trees on Xilinx FPGAs. Paper presented at the MBMV, IBM Germany Research and Development, Böblinguen, Germany, p. 171–182, 10–12 March 2014.
13. L Aksoy, E Costa, P Flores, J Monteiro, Exact and approximate algorithms for the optimization of area and delay in multiple constant multiplications. IEEE Trans. Comput.-Aided Des. Integr. Circuits **27**(6), 1013–1026 (2008). doi:10.1109/TCAD.2008.923242
14. L Aksoy, E Costa, P Flores, J Monteiro, Finding the optimal tradeoff between area and delay in multiple constant multiplications. Elsevier Journal Microprocessors and Microsystems 35 (8), 729 – 741 (2011). doi: doi.org/10.1016/j.micpro.2011.08.009.
15. AG Dempster, SS Dimirsoy, I Kale, Designing multiplier blocks with low logic depth. Paper presented at the Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS), Scottsdale, Arizona, p. 773–776, 26–29 May 2002.
16. M Faust, C-H Chang, Minimal logic depth adder tree optimization for multiple constant multiplication. Paper presented at the Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS), Paris, France, p. 457–460, May 30-Jun 2, 2010.
17. K Johansson, O Gustafsson, LS DeBrunner, L Wanhammar, Minimum adder depth multiple constant multiplication algorithm for low power FIR filters. Paper presented at the Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS), Rio de Janeiro, Brazil, p. 1439–1442, 15–18 May 2011.
18. AG Dempster, MD Macleod, Using all signed-digit representations to design single integer multipliers using subexpression elimination. Paper presented at the Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS), Vancouver, British Columbia, p. 165–168, 23–26 May 2004.
19. L Aksoy, E Costa, P Flores, J Monteiro, Multiplierless design of linear DSP transforms. VLSI-SoC: Advanced Research for Systems on Chip, ed. by S. Mir, C-Y Tsui, R Reis, O Choy (Springer 2011), p. 73 – 93.
20. YH Ho, CU Lei, HK Kwan, N Wong, Global optimization of common subexpressions for multiplierless synthesis of multiple constant multiplications. Paper presented at the Proceedings of Asia and South Pacific Design Automation Conference, Seoul, South Korea, p. 119–124, 21–24 January 2008.
21. A Hosangadi, F Fallah, R Kastner, Simultaneous optimization of delay and number of operations in multiplierless implementation of linear systems. Paper presented at the Proceedings of International Workshop on Logic Synthesis, Lake Arrowhead, California, p. 1–8, 8–10 June 2005.
22. KN Macpherson, RW Stewart, Rapid prototyping - Area efficient FIR filters for high speed FPGA implementation. IEE Proceedings - Vision, Image Signal Processing 156, 711–720 (2006). doi:10.1049/ip-vis:20045133.
23. U Meyer-Baese, J Chen, CH Chang, AG Dempster, A comparison of pipelined RAGn and DA FPGA-based multiplierless filters. Paper presented at the IEEE Asian-Pacific Conference on Circuits and Systems, Singapore, p. 1555–1558, 4–7 December 2006.
24. L Aksoy, E Costa, P Flores, J Monteiro, Design of low-complexity digital finite impulse response filters on FPGAs. Paper presented at the Proceedings of Design, Automation and Test in Europe Conference, Dresden, Germany, p. 1197-1202, 12–16 March 2012.
25. M Faust, C-H Chang, Bit-parallel Multiple Constant Multiplication using Look-Up Tables on FPGA. Paper presented at the Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS), p. 657–660, Rio de Janeiro, Brazil, 15–18 May 2011.
26. G Botella, A Garcia, M. Rodriguez-Alvarez, E Ros, U Meyer-Baese, M C Molina, Robust bioinspired architecture for optical-flow computation. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 18(4), 616–629 (2010). doi: 10.1109/TVLSI.2009.2013957.
27. G Botella, U Meyer-Baese, A Garcia, M Rodriguez, Quantization analysis and enhancement of a VLSI gradient-based motion estimation architecture. Digital Signal Processing, 22(6), 1174–1187 (2012). doi: doi.org/10.1016/j.dsp.2012.05.013.
28. G Botella, U Meyer-Baese, A Garcia, Bio-inspired robust optical flow processor system for VLSI implementation. Electron Lett **45**(25), 1304–1305 (2009). doi:10.1049/el.2009.1718
29. E Castillo, A Lloris, DP Morales, L Parrilla, A Garcia, G Botella, A new area-efficient BCD-digit multiplier. Digital Signal Processing 62, 1–10 (2017). doi: dx.doi.org/10.1016/j.dsp.2016.10.011.
30. Y Voronenko, M Püschel, Multiplierless multiple constant multiplication, ACM Transactions on Algorithms, 3 (2), 11 (2007). doi: 10.1145/1240233.1240234.
31. I Koren, Computer Arithmetic Algorithms. (Prentice Hall, 1993).
32. U Meyer-Baese, Digital Signal Processing with Field Programmable Gate Arrays, 4th. edn. (Springer, 2014).
33. DR Bull, DH Horrocks, Primitive operator digital filters. IEE Proceedings G - Circuits, Devices and Systems **138**(3), 401–412 (1991). doi:10.1049/ip-g-2.1991.0066
34. K Johansson, O Gustafsson, L Wanhammar, Switching activity estimation for shift-and-add based constant multipliers. Paper presented at the Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS), Seattle, Washington, p. 676–679, 18–21 May 2008.
35. J Chen, CH Chang, High-level synthesis algorithm for the design of reconfigurable constant multiplier. IEEE Trans Computer-Aided Des Integr Circ Syst **28**(12), 1844–1856 (2009). doi:10.1109/TCAD.2009.2030446
36. AG Dempster, MD Macleod, Use of minimum-adder multiplier blocks in FIR digital filters. IEEE Trans. Circ Syst II – Analog Digit Sig Process **42**(9), 569–577 (1995). doi:10.1109/82.466647
37. O Gustafsson, AG Dempster, K Johansson, MD Macleod, L Wanhammar, Simplified design of constant coefficient multipliers. Circuits Syst. Signal Process **25**(2), 225–251 (2006). doi:10.1007/s00034-005-2505-5
38. KK Parhi, VLSI digital signal processing systems: design and implementation, (John Wiley & Sons, 2007).
39. O. Gustafsson, Contributions to Low-Complexity Digital Filters, 837, (Linköping Studies and technology dissertations, 2003).
40. R Kastner, A Hosangadi, F Fallah, Arithmetic Optimization Techniques for Hardware and Software Design, (Cambridge University Press, 2010).