Optimal Single Constant Multiplication using Ternary Adders

Martin Kumm, *Member, IEEE,* Oscar Gustafsson, *Senior Member, IEEE,* Mario Garrido, *Member, IEEE,* and Peter Zipf, *Member, IEEE*

***Abstract*—The single constant coefficient multiplication is a frequently used operation in many numeric algorithms. Extensive previous work is available on how to reduce constant multiplications to additions, subtractions, and bit shifts. However, in the work so far, only common 2-input adders were used. As modern FPGAs support efficient ternary adders, i. e., adders with three inputs, this work investigates constant multiplications which are built from ternary adders in an optimal way. The results show that the multiplication with any constant up to 22 bits can be realized by only three ternary adders. Average adder reductions of more than 33% compared to optimal constant multiplication circuits using 2-input adders are achieved for coefficient word sizes of more than five bits. Synthesis experiments show FPGA average slice reductions in the order of 25% and a similar or higher speed than their 2-input adder counterparts.**

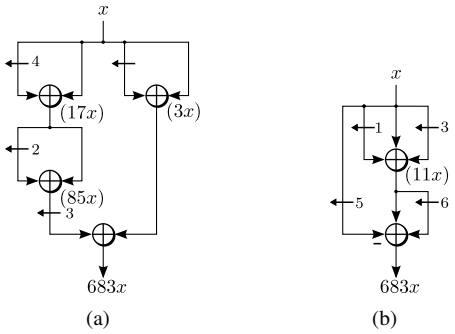

Fig. 1. Optimal SCM adder circuits which realize the multiplication with 683. (a) Using 2-input adders. (b) Using ternary adders.

I. Introduction

The multiplication of a variable with a constant (also known as scaling operation) is a very frequent operation that appears in many numerical algorithms, especially in the domain of digital signal processing (DSP). When implemented in hardware, its complexity can be significantly simplified by reducing the constant multiplication to additions, subtractions and bit shifts. As the shifts are hard-wired, the goal is to find a configuration with minimum adder cost, i. e., minimum number of adders and subtractors. This optimization is commonly called single constant multiplication (SCM) optimization – to distinguish from the generalized multiple constant multiplication (MCM) optimization. Both are NP-hard optimization problems [1].

Many heuristic and optimal methods were developed during the last three decades to reduce the add/subtract operations in constant multiplications [2]–[7]. The first optimal SCM method which is based on an exhaustive graph enumeration and evaluation was proposed by Dempster and Macleod [4]. They showed that all coefficients up to 12 bits can be computed by using up to four adders. Later, it was observed that many of these adder graph topologies compute the same numbers [5]. An additional classification enabled the evaluation of up to five adders, pushing the limit to 19-bit coefficients. Further extensions for up to six adders resulted in optimal SCM circuits for coefficients of up to 32 bits [7].

The optimal SCM circuits described above can be mapped to 2-input adders on field programmable gate arrays (FPGAs) to replace the costly generic embedded multipliers. However, modern FPGAs support ternary adders, i. e., adders with three inputs, that require exactly the same number of look-up table (LUT) resources as common 2-input adders for the same output word size [8], [9]. A detailed description of ternary adders and a comparison for Altera and Xilinx FPGAs is given in [10]. One notable drawback is that their speed is reduced by up to 20% (Altera Stratix IV) and up to 40% (Xilinx Virtex-6) compared to 2-input adders.

Ternary adders have been already used in a heuristic for pipelined MCM circuits [10]. It was shown that the best results compared to 2-input adders can be obtained for a low number of coefficients. However, the method in [10] was designed for MCM. In case of SCM, the design space is small enough to be able to find optimal solutions for practical coefficient word sizes. Nevertheless, none of the previous work on optimal SCM can be used to design optimal ternary adder SCM solutions. To close this gap, the goal of this work is to present a methodology to produce and evaluate optimal SCM circuits with ternary adders. For that, the framework developed in [5] is extended to ternary adders and a novel methodology for the graph classification is proposed for their numerical evaluation. As an example, Figs. 1(a) and 1(b) show the optimal SCM circuits for the multiplication by 683 using 2-input and three-input adders, respectively. Left shifts are denoted by arrows. Even if the two adders on the right of Fig. 1(a) can be merged into a single ternary adder, the resulting circuit still requires one adder more than necessary.

II. Background on Optimal Constant Multipliers

In this section we review the background on optimal constant multiplications, originally introduced in [5], and relevant extensions are provided.

M. Kumm and P. Zipf are with the Digital Technology Group, University of Kassel, 34121 Kassel, Germany (e-mail: kumm@uni-kassel.de, zipf@uni-kassel.de).

O. Gustafsson and M. Garrido are with the Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden, (e-mail: oscar.gustafsson@liu.se, mario.garrido@liu.se).

Manuscript received April 19, 20xx; revised December 27, 20xx.

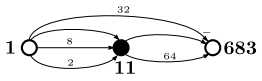

Fig. 2. Adder graph representation of circuit in Fig. 1(b).

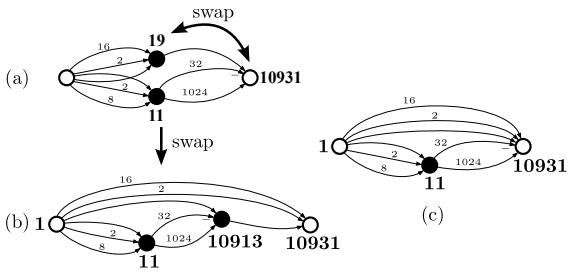

Fig. 3. Two fully specified graphs to multiply with 10931 (a)+(b) and the corresponding vertex reduced graph (c).

A. Adder Graph Representation

A constant multiplier can be represented as a directed acyclic graph (DAG), called adder graph. Each node except the input node corresponds to an adder and each edge has a power-of-two weight which corresponds to a bit shift. The adder graph representation of the adder circuit in Fig. 1(b) is shown in Fig. 2. Each adder computes a constant multiple of the input, which is called a fundamental [4]. The input ('1') and output nodes are hollow, while internal nodes, called non-output fundamentals (NOFs), are filled.

B. Vertex Reduced Graphs

In [5], it was observed that if the out-degree of a node is one, the adder can be swapped with the succeeding adder without changing the output fundamental. It is illustrated in Fig. 3 that this also works for three-input adders. In the adder graph in Fig. 3(a), the node realizing fundamental 19 has an out-degree of one. Hence, it can be swapped with the succeeding node 10931, leading to the adder graph shown in Fig. 3(b).

A topologically unique representation can be obtained by merging each node with out-degree one with the succeeding node until no node with out-degree one exist. The resulting graph is called *vertex reduced*. An example is given in Fig. 3(c). To distinguish between both representations, the original graphs are called *fully specified*.

The number of 2-input adders for each node is the in-degree minus one [5], while it is $(\text{in-degree} - 1)/2$ in case of ternary adders. When the vertex reduced graph representation is topologically identical (i. e., isomorphic) for two fully specified graphs, they can be seen as *redundant* as they can compute exactly the same coefficients.

C. Transposed Graphs

It is well known from the theory of linear networks that the transfer function of a linear single-input single-output system is not affected by transposition, although the structure may be completely different. The transposition is done by reversing the directions of each edge, replacing branches by adders and vice versa, and swapping the input and output. Hence, from a pair of graphs which are isomorphic when one of the graphs is transposed, each graph is redundant compared to the other.

III. Proposed Graph Generation

A. Multiplicative Graphs

Further redundant graphs can be found by evaluating *multiplicative graphs*. Multiplicative graphs were introduced in [5] but only for classification and not for the elimination of redundant graphs. If the graph has an articulation point, i. e., a node which splits the graphs into two disconnected graphs when removed, it can be divided in a series connection of two adder graphs with lower complexity. By exchanging the lower complexity graphs, new topologies can be obtained which are obviously able to compute the identical coefficients and are therefore redundant.

B. Adder Graph Generation

Instead of searching an optimal solution for a given constant we follow the approach of [4], [5] to generate all graphs with a given adder count and to evaluate the coefficients they can realize. Optimality is achieved by making an exhaustive search for all graphs and removing the redundant graphs using the following procedure:

1) From cost $C = 1, \ldots, C_{\max}$, all fully specified graphs with the following properties are computed:
   a) A cost-C graph contains $C + 1$ nodes
   b) Each node has an in-degree of three
   c) Each node lies on a path from the input node to the output node
2) For each graph, the vertex reduced graph is obtained and stored in a list when
   a) the graph itself,
   b) the transposed graph, and,
   c) the graph obtained by interchanging the subgraphs of a multiplicative graph,

   are not isomorphic to any existing graph in the list.

In the implementation, the VF2 algorithm [11] was used for the isomorphism test which is available in the igraph library [12]. The resulting vertex reduced graphs obtained from this computation are shown in Fig. 4 for up to three ternary adders.

IV. Proposed Coefficient Evaluation

As we are interested in the numeric values of the coefficients, each graph has to be evaluated for different combinations of bit shifts. The naive way is to permute the bit shifts on each edge up to a certain limit $s_{\max}$. The computational run time complexity will be $s_{\max}^E$ where E denotes the number of edges which can be up to $E_{\max} = 3C$ for a cost-C graph. This exponential complexity makes this naive evaluation infeasible. Hence, a way to drastically reduce the necessary number of different bit shifts is described after we have introduced some formal definitions.

A. Definitions and Properties

The operation of each node, a single addition/subtraction of shifted inputs and a shifted output, can be formalized as $\mathcal{A}$-operation [6]. For ternary adders, it is extended to three input arguments [10]:

$$\mathcal{A}_q^3(u, v, w) = |2^{l_u} u + (-1)^{s_v} 2^{l_v} v + (-1)^{s_w} 2^{l_w} w| 2^{-r} \quad (1)$$

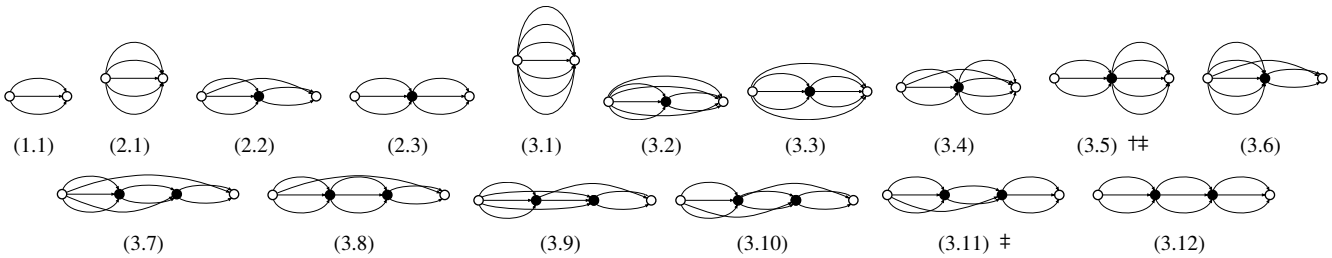

Fig. 4. All vertex reduced graph topologies up to 3 ternary adders. Graphs topologies that can be produced by transposition (marked with †) or rearranging multiplicative sub-graphs (marked with ‡) are not considered.

The arguments u, v and w correspond to the node inputs and $q = (l_u, l_v, l_w, r, s_v, s_w)$ depicts the configuration, where $l_{u/v/w} \in \mathbb{N}_0$ are left shifts, $r \in \mathbb{N}_0$ is a right shift, and $s_{v/w} \in \{0, 1\}$ correspond to the signs of v and w.

In addition, a set $\mathcal{A}^3_*(u, v, w)$ is defined, which contains all possible numbers which can be computed from a triplet (u, v, w) with a single $\mathcal{A}^3$-operation:

$$\mathcal{A}^3_*(u, v, w) = \left\{ \mathcal{A}^3_q(u, v, w) \mid q \text{ valid configuration} \right\} \quad (2)$$

A *valid* configuration q is a combination of l_u, l_v, l_w, r, s_v and s_w such that the result is a positive odd integer with $\mathcal{A}_q(u, v, w) \leq c_{\text{max}}$. Any even coefficient can be obtained by left shifting an odd coefficient. The limit c_{max} has to be introduced to keep the set finite. It is usually chosen as power-of-two value which is set to the maximal bit width of the target values plus one [6].

SCM adder graphs using ternary adders have different lower bounds regarding the adder cost and the adder depth, which is defined as the maximal number of cascaded adders [5]. While the lower bound for adder cost and the adder depth using 2-input adder SCM is both $\lceil \log_2 \text{nz}(c) \rceil$ [13], it is only $\lceil \log_3 \text{nz}(c) \rceil$ using ternary adders, where $\text{nz}(c)$ denotes the minimum number of non-zeros of coefficient c in a signed digit representation [13].

TABLE I
COMPUTATION OF COEFFICIENTS AND NOFS.

adder cost	graph	N_1	N_2	equation
1	1	–	–	$H_{1.1} = \mathcal{A}_*(1, 1, 1)$
2	1	$H_{1.1}$	–	$H_{2.1} = \bigcup_{n_1 \in N_1} \mathcal{A}_*(1, 1, n_1)$
2	2	$H_{1.1}$	–	$H_{2.2} = \bigcup_{n_1 \in N_1} \mathcal{A}_*(1, n_1, n_1)$
2	3	$H_{1.1}$	–	$H_{2.3} = \bigcup_{n_1 \in N_1} \mathcal{A}_*(n_1, n_1, n_1)$
3	1	$H_{2.1}$	–	$H_{3.1} = \bigcup_{n_1 \in N_1} \mathcal{A}_*(1, 1, n_1)$
3	2	$H_{2.2}$	–	$H_{3.2} = \bigcup_{n_1 \in N_1} \mathcal{A}_*(1, 1, n_1)$
3	3	$H_{2.3}$	–	$H_{3.3} = \bigcup_{n_1 \in N_1} \mathcal{A}_*(1, 1, n_1)$
3	4	$H_{2.3}$	$N_{2.3}(n_1)$	$H_{3.4} = \bigcup_{\substack{n_1 \in N_1 \\ n_2 \in N_2(n_1)}} \mathcal{A}_*(1, n_1, n_2)$
3	5	$H_{2.1}$	–	$H_{3.5} = \bigcup_{n_1 \in N_1} \mathcal{A}_*(n_1, n_1, n_1)$
3	6	$H_{2.1}$	–	$H_{3.6} = \bigcup_{n_1 \in N_1} \mathcal{A}_*(1, n_1, n_1)$
3	7	$H_{2.2}$	–	$H_{3.7} = \bigcup_{n_1 \in N_1} \mathcal{A}_*(1, n_1, n_1)$
3	8	$H_{2.3}$	–	$H_{3.8} = \bigcup_{n_1 \in N_1} \mathcal{A}_*(1, n_1, n_1)$
3	9	$H_{2.1}$	$N_{2.1}(n_1)$	$H_{3.9} = \bigcup_{\substack{n_1 \in N_1 \\ n_2 \in N_2(n_1)}} \mathcal{A}_*(n_1, n_1, n_2)$
3	10	$H_{2.2}$	$N_{2.2}(n_1)$	$H_{3.10} = \bigcup_{\substack{n_1 \in N_1 \\ n_2 \in N_2(n_1)}} \mathcal{A}_*(n_1, n_1, n_2)$
3	11	$H_{2.2}$	–	$H_{3.11} = \bigcup_{n_1 \in N_1} \mathcal{A}_*(n_1, n_1, n_1)$
3	12	$H_{2.3}$	–	$H_{3.12} = \bigcup_{n_1 \in N_1} \mathcal{A}_*(n_1, n_1, n_1)$

B. Proposed Classification for Evaluation

The key idea to reduce the complexity in the coefficient evaluation in previous work [5] was to construct coefficients with higher complexity from already known coefficients with lower complexity. For that, the graphs were classified into three classes: 1) additive, 2) multiplicative and 3) leapfrog. The coefficients of additive and multiplicative graphs can be obtained by addition and multiplication of (possibly shifted) coefficients of lower complexity coefficients, respectively. In leapfrog graphs, NOFs of lower complexity coefficients were required in addition.

The same classification can be in principle performed on the ternary adder graphs. However, the vertex reduced graphs contain 2-input adders as subgraphs which have to be evaluated in addition. Furthermore, to construct the solution, NOFs are required which are different from intermediate values obtained by the classification above. Therefore, a different methodology was developed for the ternary adder graphs which computes coefficients with higher complexity from lower complexity coefficients by adding a single ternary add operation.

Let $H_{C.G}$ denote the set of all coefficients that can be realized from the cost-C adder graph number G (up to a certain limit c_{max}). Let $N_{C.G}(c)$ denote the set of all NOFs from the cost-C adder graph number G to realize coefficient c. Now, specific permutations of NOFs and the input '1' can be performed to compute the coefficients. To give an example, consider graph (3.3) in Fig. 4. It contains graph (2.3) as subgraph. Hence, graph (3.3) can be constructed by adding a ternary adder to graph (2.3) and connecting two of its inputs to the input and one input to the output of graph (2.3). Formally, this can be computed by $H_{3.3} = \bigcup_{n_1 \in H_{2.3}} \mathcal{A}_*(1, 1, n_1)$.

The equations to compute the coefficients of all graphs up to three ternary adders are listed in Table I. Up to two NOFs are considered which are contained in sets N_1 and N_2. Note that there are sometimes different ways to compute a given topology, e. g, adder graph (3.2) could also be computed from (2.2). In that case, the simplest computation scheme was selected. The coefficient evaluation is performed according to the order given in Table I which is sorted by adder cost and adder depth to guarantee least cost and adder depth.

TABLE II
PROPERTIES OF ADDER GRAPHS WITH TERNARY ADDERS.

Adder count	Fully specified graphs	Vertex reduced graphs	Further reduced graphs	Max. coefficient	Covered word size
1	1	1	1	42	5
2	3	3	3	2930	11
3	20	14	12	7154954	22
4	231	100	85	–	–
5	4186	1046	914	–	–

V. RESULTS

A. Properties of the Obtained Graphs

The most important properties of the obtained adder graphs for up to five ternary adders are given in Table II. The number of different graph topologies is given for fully specified graphs, vertex reduced graphs as described in Section II-B and the further reductions described in Sections II-C and III-A (column "Further reduced graphs"). A significant reduction compared to the fully specified graphs can be obtained by eliminating redundant graphs. Compared to SCM circuits with 2-input adders, the number of graphs grows much faster with the number of adders. For example, there are 54 vertex reduced graphs for 5 adder graphs with 2-input adders [5], while there are 1046 vertex reduced graphs for ternary adders.

B. Coefficient Evaluation

All adder graphs up to three ternary adders have been evaluated according to the formulas in Table I. The last two columns of Table II list the maximal coefficient and its corresponding word size that can be covered by the given adder count. An SCM circuit with a given adder count can realize any factor between zero and the corresponding maximal coefficient. Hence, 7154955 is the first coefficient which requires at least four ternary adders. The average and maximal adder cost (upper bound) using 2-input and ternary adders up to 19 bits (limit in [5]) and 22 bits, respectively, are plotted in Fig. 5 (top). Note that the minimal adder cost (lower bound) is zero as there is exactly one power-of-two coefficient for each coefficient word size. The percentage average adder cost improvement compared to 2-input adders is shown in Fig. 5 (bottom). For coefficients with more than five bits, an average adder count improvement of at least 33% can be obtained by using ternary adders.

The cost of a ternary adder solution is always less or equal than using 2-input adders. By evaluating the histogram of the number of reduced adders for all coefficients up to 19 bits it turns out that for 62.7% of the coefficients two adders can be saved, in 36.9% of the cases one adder can be saved and there is no improvement for only 0.4% of the coefficients.

Detailed parameters for each topology are listed in Table III. The maximal adder depth is only three compared to five for 2-input adder SCM. The maximal coefficient that can be completely covered by each topology is listed in Table III. One can observe, that there are graphs which have a very limited maximal coefficient like for the topologies (1.1), (2.3) and (3.12) where the maximal coefficient is 42. The reason is that these topologies are multiplicative graphs and 43 is the first prime number with more than three non-zeros in its canonic signed digit (CSD) representation. The opposite extreme is graph (3.9) which can completely cover all 21-bit numbers.

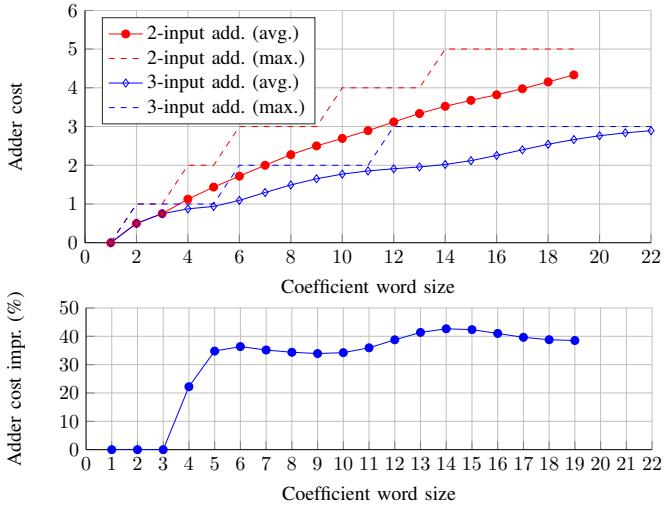

Fig. 5. Adder cost comparison between optimal SCM circuits using 2-input and ternary adders (top) and relative average adder cost improvement compared to 2-input adders (bottom).

TABLE III
PROPERTIES OF DIFFERENT TOPOLOGIES.

Topology	Adder depth	Max. coefficient	Covered word size
1.1	1	42	5
2.1	2	682	9
2.2	2	1376	10
2.3	2	42	5
3.1	2	10922	13
3.2	2	174762	17
3.3	3	632746	19
3.4	3	199484	17
3.5	3	682	9
3.6	3	12914	13
3.7	3	87464	16
3.8	3	13184	13
3.9	3	2775756	21
3.10	3	1774904	20
3.11	3	5668	12
3.12	3	42	5

C. FPGA Synthesis Experiments

The adder count is a good measure for high-level comparisons but the real FPGA resources will also depend on the word size. As adder graphs with ternary adders typically have NOFs with larger magnitude a larger word size and a lower improvement in terms of logic resources is expected. Ternary adders are also known to be slower than 2-input adders [10]. However, this is partially compensated due to a lower adder depth in the adder graphs of ternary adders.

To evaluate all these effects, synthesis experiments were performed. For that, a VHDL code generator for the optimal SCM circuits using 2-input and ternary adders was implemented with the help of the FloPoCo framework [14]. Our implementation is available as open source in the "uni_ks" branch of the FloPoCo Git repository (see [14]). All coefficients up to 12

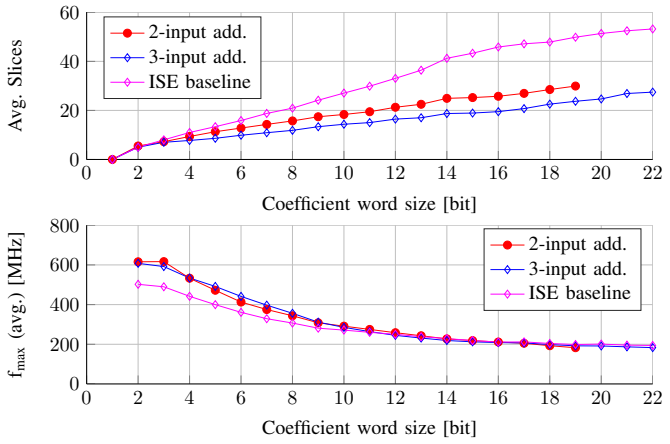

Fig. 6. Comparison between different SCM circuits on Xilinx FPGAs of average slices (top) and average maximum clock frequency, $f_{\max}$ (bottom).

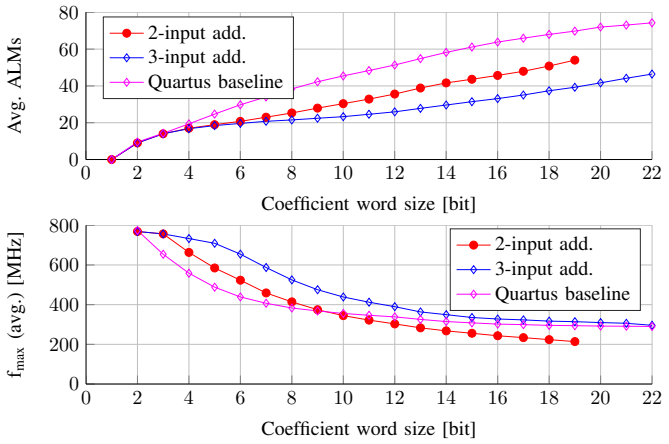

Fig. 7. Comparison between different SCM circuits on Altera FPGAs of average ALMs (top) and average maximum clock frequency, $f_{\max}$ (bottom).

bits were completely synthesized while for word sizes $b > 12$ bits, 1000 random coefficients uniformly distributed between $2^b \ldots 2^{b+1} - 1$ were used for each word size. The input word size was selected to be 16 bits and registers were placed at the input and output to obtain realistic throughput results. The synthesis was performed for Xilinx Virtex 6 (XC6VSX475T-1) and Altera Stratix V (5SGSMD3E3H29C4) FPGAs using ISE 13.4 and Quartus 15.1, respectively, with unconstrained clock. As a baseline, synthesis results from a straight forward VHDL implementation of a constant multiplication were obtained. The synthesis results are plotted in Figs. 6 and 7, respectively. Compared to 2-input adder SCM, Slice and ALM reductions of about 25% are achieved for coefficients with more than 4 and 10 bits word size, respectively. Compared to the baseline from the tools, from which the implementation method is unknown, resource reductions of about 50% are achieved. For the Xilinx FPGA, the slower speed of a ternary adder is nearly exactly compensated by the lower adder depth. For Altera, the speed using ternary adders is considerable higher (25% to 45% for coefficients >5 bits).

Besides resources and speed, power estimations for Xilinx FPGAs were performed for the circuits up to 19 bit using XPower with 1000 uniformly distributed random samples. For combinational circuits, logic and signal power are the relevant components. It turned out that, on average, the logic and signal power components are, respectively, 3.09 mW and 3.41 mW for 2-input adder SCM while they are 3.47 mW (+12.3%) and 2.54 mW (−25.5%) for ternary adder SCM, leading to a total reduction of 7.5%.

VI. Conclusion

A methodology and thorough analysis of optimal single constant multiplication circuits using ternary adders, which became available on modern FPGAs, was presented. The generation of all vertex reduced graphs is based on an extension of previous work [5]. On top of that, a new framework was developed for their numeric evaluation. The evaluation is performed by re-using previously found results from ternary SCM circuits with lower complexity which avoids an exhaustive evaluation of all the shifts. The obtained results show that any coefficient with up to 22 bits can be realized by only three ternary adders, which covers most practical digital signal processing applications. This leads to an average adder reduction compared to 2-input adders of more than 33% for coefficients with a word size of more that four bits. This adder reduction also leads to very compact single constant multiplication circuits when mapped to FPGAs. Extensive synthesis experiments revealed average logic reductions in the order of 25%. Due to the reduced adder depth, the ternary adder SCM circuits are similarly fast (Xilinx) or even faster (Altera) compared to their 2-input adder counterpart.

References

[1] P. R. Cappello and K. Steiglitz, "Some complexity issues in digital signal processing," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 32, no. 5, pp. 1037–1041, Oct. 1984.
[2] R. Bernstein, "Multiplication by integer constants," *Software: Practice and Experience*, vol. 16, no. 7, pp. 641–652, Jul. 1986.
[3] D. R. Bull and D. H. Horrocks, "Primitive operator digital filters," *IEE Proc. Circuits, Devices, Syst.*, vol. 138, no. 3, pp. 401–412, Jun. 1991.
[4] A. G. Dempster and M. D. Macleod, "Constant integer multiplication using minimum adders," *IEE Proc. Circuits, Devices, Syst.*, vol. 141, no. 5, pp. 407–413, Oct. 1994.
[5] O. Gustafsson, A. G. Dempster, K. Johansson, M. D. Macleod, and L. Wanhammar, "Simplified design of constant coefficient multipliers," *Circuits, Syst., Signal Process.*, vol. 25, no. 2, pp. 225–251, 2006.
[6] Y. Voronenko and M. Püschel, "Multiplierless multiple constant multiplication," *ACM Trans. Algorithms*, vol. 3, no. 2, pp. 1–38, May 2007.
[7] J. Thong and N. Nicolici, "An optimal and practical approach to single constant multiplication," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 30, no. 9, pp. 1373–1386, Sep. 2011.
[8] J. M. Simkins and B. D. Philofsky, "Structures and methods for implementing ternary adders/subtractors in programmable logic devices," US Patent 7 274 211, Sep. 25, 2007.
[9] G. Baeckler, M. Langhammer, J. Schleicher, and R. Yuan, "Logic cell supporting addition of three binary words," US Patent 7 565 388, Jul. 21, 2009.
[10] M. Kumm, M. Hardieck, J. Willkomm, P. Zipf, and U. Meyer-Baese, "Multiple constant multiplication with ternary adders," in *Proc. IEEE Int. Conf. Field Programmable Logic Applicat.*, Sep. 2013, pp. 1–8.
[11] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento, "An improved algorithm for matching large graphs," in *Workshop on Graph-based Representations in Pattern Recognition*, 2001, pp. 149–159.
[12] G. Csardi and T. Nepusz, "The igraph software package for complex network research," *InterJournal*, 2006.
[13] O. Gustafsson, "Lower bounds for constant multiplication problems," *IEEE Trans. Circuits Syst. II*, vol. 54, no. 11, pp. 974–978, Nov. 2007.
[14] F. de Dinechin. (2015, Apr.) FloPoCo project website. [Online]. Available: http://flopoco.gforge.inria.fr