IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 7, 2019, accepted September 11, 2019, date of publication October 7, 2019, date of current version October 31, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2946054

Reconfigurable Adaptive Multiple Transform Hardware Solutions for Versatile Video Coding

CARLO SAU[1], DARIO LIGAS[1], TIZIANA FANNI[1], LUIGI RAFFO[1], (Member, IEEE), AND FRANCESCA PALUMBO[2], (Member, IEEE)

[1]Dipartimento di Ingegneria Elettrica ed Elettronica, Universitá degli Studi di Cagliari, 09123 Cagliari, Italy
[2]Dipartimento di Chimica e Farmacia, Universitá degli Studi di Sassari, 07100 Sassari, Italy

Corresponding author: Carlo Sau (carlo.sau@diee.unica.it)

This work was supported by the EU Commission's ECSEL Programme under Agreement 783162.

ABSTRACT Computer aided design is nowadays a must to quickly provide optimized circuits, to cope with stringent time to market constraints, and to be able to guarantee colliding constrained requirements. Design automation is exploited, whenever possible, to speed up the design process and relieve the developers from error prone customization, optimization and tuning phases. In this work we study the possibility of adopting automated algorithms for the optimization of reconfigurable multiple constant multiplication circuits. In particular, an exploration of novel reconfigurable Adaptive Multiple Transform circuital solutions adoptable in video coding applications has been conducted. These solutions have also been compared with the unique similar work at the state of the art, revealing to be beneficial under certain constraints. Moreover, the proposed approach has been generalized with some guidelines helpful to designers facing similar problems.

INDEX TERMS Circuit optimization, design automation, digital signal processing, digital systems, reconfigurable architectures, video coding, adaptive multiple transform, multiplierless constant multiplication.

I. INTRODUCTION

In modern society, devices are every day smarter and more strongly connected, frequently exchanging a variety of information. Video and images, due to their expressiveness, are natural and powerful instruments to allow communication at different levels and in different contexts [1], to support users taking decisions [2], as well as to improve the capabilities of a system to take autonomous decisions [3]. Therefore, video technologies are subjected to continuous changes and improvements to follow the pace of social trends and human needs. This turns out into periodic increase of data, in terms of resolution, frame rate, and bit width.

In a world dominated by information exchange and communication, video coding is fundamental, allowing a substantial data compression at the price of a negligible loss in video quality. To achieve better compression, the complexity of video coding algorithms is growing accordingly to the increased demand in terms of resolution, frame rate and bit width. Such complexity has become so high that most of the mass electronic devices, like smartphones, requires an application-specific unit to handle it. A dedicated video coding specific chip is then embedded on most of them. For instance Snapdragon 800 processors involve the HD, which has a High Efficiency Video Coding (HEVC) H.265 encode and decode Qualcomm® Snapdragon™ 4k Ultra hardware module for maximum file compression with high power saving [4]. Exynos 9820 mobile processor, which is also powering the Samsung Galaxy S10, supports the encoding/decoding of 8k videos at 30 fps for a multi-format codec: 10-bit HEVC(H.265), H.264, VP9 [5].

Optimizing those application specific chips, keeping them updated with the successive video coding algorithms and meeting the constraints according to the context is extremely challenging for designers. Indeed, there exists a trade-off between *specialization*, needed to meet stringent and colliding constraints related to quality, real-time and/or low-power execution, and *design flow complexity*. The more the circuit is specialized, the more will be the effort required to designers in order to optimize it, which will lead to solutions that are

The associate editor coordinating the review of this manuscript and approving it for publication was Prakasam Periasamy.

technology- and algorithm-specific. The less, in turn, will be the re-usability of specialized designs among successive video technology generations, negatively impacting on both designer efforts and development costs.

According to the already explained trend, the next video coding standard, known as the Versatile Video Coding (VVC) and expected to be released by the end of 2020, is intended to improve its predecessor, the HEVC, on both compression efficiency and video quality. To do so, it is introducing lots of enhancements on several aspects of the video coding algorithm. One of them is related to the transform phase, that in HEVC involves two different types of transform, selected according to the specific processed data, e.g. depending on the assigned prediction mode. In VVC, the Adaptive Multiple Transform (AMT) will be adopted, which is based on five different types of transform. To cope with such an increased complexity, designers are required to focus on transforms to improve their implementation. Being these transforms mutually exclusive for a given data, a novel and promising solution to tackle this issue is to leverage on reconfiguration and to adopt, for example, the same circuit to execute one of the five transforms at a time [6].

Within the described context, the innovative contributions of this paper are listed hereafter.

1) Exploration and assessment of several solutions for the implementation of reconfigurable AMT circuits exploiting multiplierless optimization algorithms [7]. These latter can be used since transforms require Multiple Constant Multiplication (MCM). The exploration based on a reconfigurable approach, where different architectural parameters are tuned, is not yet present in literature.
2) Quantitative comparison among the proposed implementations and the unique reconfigurable AMT solution present in literature by Mert *et al.* [6]. Results show that the adoption of the MCM optimization algorithms is doable and beneficial under some constraints.
3) Generalization of the approach to adopt the MCM optimization algorithms in any hardware implementation involving MCM, where parallelism and multiplexing of the outputs is necessary. Besides in other two-dimensional transform cases (e.g. Fast Fourier Transforms), it can be used in two-dimensional filtering [8], color space conversion [9] or polyphase filter banks [10]. To this aim, a set of guidelines has been derived, considering different metrics of interest, such as resources minimization, operating frequency maximization, designer effort reduction and mandatory skills/background.

The rest of the document is organized as follows: in Section II a short overview of the background, application context and available works that are already at the state of the art is provided. Section III discusses the different architectural solutions that constitute the first novel contribution of this work; while Section IV reports on the second contribution showing the results obtained both in terms of architectural exploration and comparison with the state of the art. At last, Section V reports on the third contribution, educing some general guidelines from the results achieved specifically for the AMT case, prior to conclude in Section VI with some final remarks.

II. LITERATURE AND BACKGROUND

A. ADAPTIVE MULTIPLE TRANSFORM HARDWARE IMPLEMENTATIONS

In literature, several works tried to implement video coding transformation phase in hardware. A common implementation scheme for two-dimensional transform hardware implementation is to split it into two one-dimensional cascaded transforms, as shown in Figure 1. The implemented operation is described by Equation 1:

$$D = (DTT \times S)^T \times DTT = T^T \times DTT \tag{1}$$

where DTT is the Discrete Trigonometric Transform (DTT) matrix, S is the source image block, $T = DTT \times S$ is the temporary intermediate one-dimensional transformed image block, and D is the resulting two-dimensional transformed image block. The elements of D and T are then calculated by means of matrix multiplications, as in Equation 2:

$$t_{ij} = \sum_{j=1}^{N} c_{ij} \times s_{ji} \tag{2}$$

where t_{ij} is the i-th row and j-th column element of T, c_{ij} is the i-th row and j-th column element of DTT, s_{ji} is the j-th row and i-th column element of S, and N is the block size.

In the past, the Discrete Cosine Transform II (DCTII) was primarily adopted in video coding; whereas recently researchers found out that adopting different transforms, according to the data and elaboration to be performed, may be beneficial to improve coding efficiency [11]. For this reason HEVC, which is currently the latest video coding standard, uses also the Discrete Sine Transform VII (DSTVII) for intra-predicted image blocks. Next generation video coding, VVC, is moving further by adopting five different transforms in its AMT approach. DCTII, DCTV, DCTVIII, DSTI and DSTVII are all included in the AMT. Moreover, VVC is also bringing the maximum block size from 32×32 to 64×64, increasing processing complexity and, in turn, resource demand of such part of the algorithm.

Within this context, to cope with complexity, the scientific community is then extremely active. On the one hand, researchers are working at the algorithm level, trying to find a set of transforms capable of reducing complexity, while maintaining a high coding quality [12], or to decompose transform matrices in order to simplify the corresponding hardware implementation [13]. On the other hand the

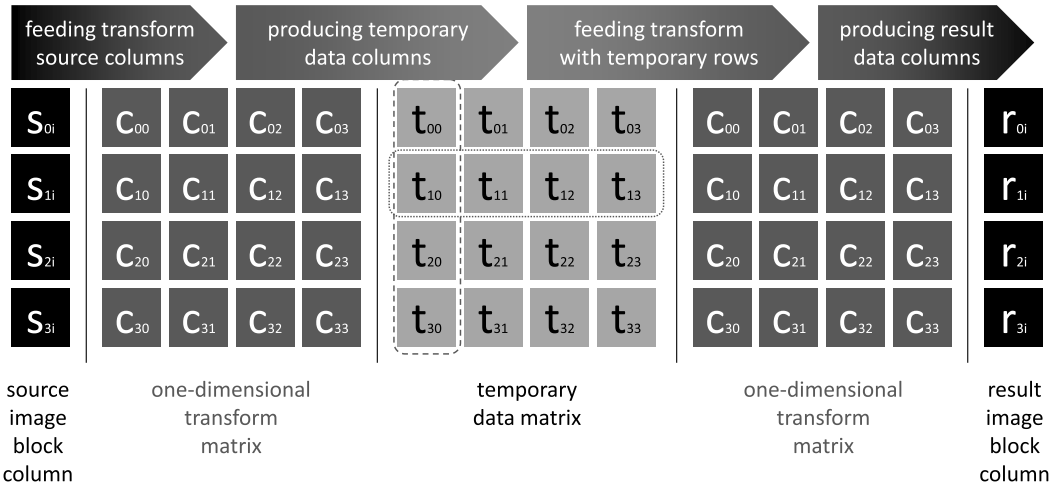

FIGURE 1. General scheme for the two-dimensional transform performed as the cascade of two one-dimensional transforms.

focus is on facilitating and optimizing the hardware implementation, by leveraging whenever possible on automated instruments [14]–[16]. Classical digital system design solutions, such as pipelining [17] and target board primitives exploitation [18], [19] are still valid. However, the main issue to be faced is related to multipliers, and it is still open. Transformation is basically a matrix multiplication and to perform it efficiently lots of multipliers are necessary. Nevertheless, some features can be exploited to simplify the problem.

1) Multipliers, usually very resource demanding operations, can be efficiently implemented with adders and shifters when one of the operands is constant, such as for transform matrix coefficients [13], [18].
2) Being row by column matrix multiplications, transforms require Multiple Constant Multiplication (MCM). Such operations, where the same input number is multiplied by several constant numbers at the same time, offer additional optimization possibilities, either employing multipliers [19] or not [6].

One important aspect of the AMT with respect to HEVC is the increased number of adopted transforms, going from two to five. The optimal transform to be applied to the current image block is chosen according to the prediction mode it is subjected to. This means that the five AMT transforms have not to be applied in parallel, but one at a time, for a given image block. Additional optimizations can then be achieved by considering this time exclusive property of the different transforms. In [6] a first step in this direction has been done by proposing a reconfigurable AMT hardware implementation capable of switching among the different transforms, elaborating one of them at a time. As far as we know, this is the first and unique reconfigurable AMT solution present at the state of the art. In this work we are going to further explore reconfiguration possibilities on AMT and to combine them with multiple constant multiplication optimizations. Due to the fact that [6] is the only work in literature delivering a reconfigurable AMT hardware solution, this is also going to be the unique term of comparison for the proposed architectures.

B. RECONFIGURABLE AND MULTIPLIERLESS HARDWARE

Reconfigurable hardware constitutes a good trade-off between general purpose systems (e.g. CPUs), offering maximum flexibility for executing applications, and dedicated circuits, guaranteeing maximum efficiency for the application they have been conceived for [20]. Reconfigurable systems can be differentiated according to the granularity the reconfiguration is applied to: fine-grained systems reconfigure functionalities and connections of the single bits within the system (the most common fine-grained devices are the Field Programmable Gate Arrays, FPGAs [21]), while coarse-grained ones reconfigure at word level [22]. The former favours flexibility at the price of reconfiguration time, while the latter limits flexibility but reconfigures very quickly. In our case, due to the fact that flexibility is limited by construction to the five AMT transforms to be supported, coarse-grained reconfiguration has been adopted.

Integer multiplications are known to be extremely resource demanding when implemented in hardware. For this reason, digital system designers often focus on optimizing their architecture [23]–[25]. One of the most effective and widely adopted solutions to reduce hardware resources of integer multipliers is adopting multiplierless approaches. By definition, these approaches substitute multiplications with a set of shifters and adders [26], [27]. Indeed, by shifting and accumulating a given integer number, it is possible to calculate its product by any other integer number. This solution is mainly suitable for constant multiplications, where one of the two operands is known a priori. Optimizing the number of adders and shifters adopted to perform a multiplication does not have a standardized optimal solution yet. Moreover, in cases like AMT, it is necessary to multiply the same number by different

coefficients at the same time, that basically means to have parallel MCMs with the input in common. Here, it is possible to exploit common operations (intermediate shifts and sums) between the datapaths accounted for a specific multiplication, resulting in further optimizations of the architecture.

In [7] algorithms for the generation of optimized shifters and adders architectures serving as constant multipliers are presented. All the proposed algorithms deliver automatically Register Transfer Level (RTL) descriptions of the resulting architecture, in Verilog Hardware Description Language (HDL). Besides optimizing the single constant multiplication, algorithms for minimizing resources of parallel and multiplexed MCM are provided. MCM-parallel [28] is capable of optimizing the parallel multiplications of the same number by a set of different coefficients, and it has been already adopted in video coding transform circuits [6]. MCM-multiplexed [29], on the contrary, is capable of optimizing the multiplication of the same number by a set of different coefficients, but providing one of the products at a time as output, according to an additional selector input. Differently from MCM-parallel, so far MCM-multiplexed has never been adopted for video coding transform circuits.

In this work, both the above-mentioned MCM approaches will be exploited for the design of novel reconfigurable AMT hardware implementations. As far as we know, this is the first work that adopts and compares, through a comprehensive architectural exploration, the algorithms proposed in [7] to derive such kind of circuits.

III. NOVEL PROPOSED ARCHITECTURES

This section presents several different implementations of AMT, grouped according to the capability of performing over the same circuit either one single transform, standalone architectures in Section III-A, or multiple transforms, reconfigurable architectures in Section III-B. Please note that standalone architectures have been built mimicking state of the art solutions presented in [6], while reconfigurable ones represent a novel contribution to the state of the art and are derived from the attempt of implementing in hardware the optimization algorithms presented in [7] with a reconfigurable approach.

The setting of the different architectures is the same: the two-dimensional transform is implemented as a cascade of two one-dimensional transforms with a transposing bank of registers between the two stages (see Figure 1). Each one-dimensional transform is capable of performing matrix multiplication of the input 4 × 4 image block with a 4 × 4 transform matrix. The circuit processes one column of the input image block per clock cycle, thus requiring four cycles to finalize the whole convolution and produce a 4 × 4 block as output. The block coming from the first one-dimensional transform circuit is stored in the transposing bank of registers. The second one-dimensional transform circuit, which is identical to the first one, reads the data row by row from the bank of registers performing transformation all along the second direction.

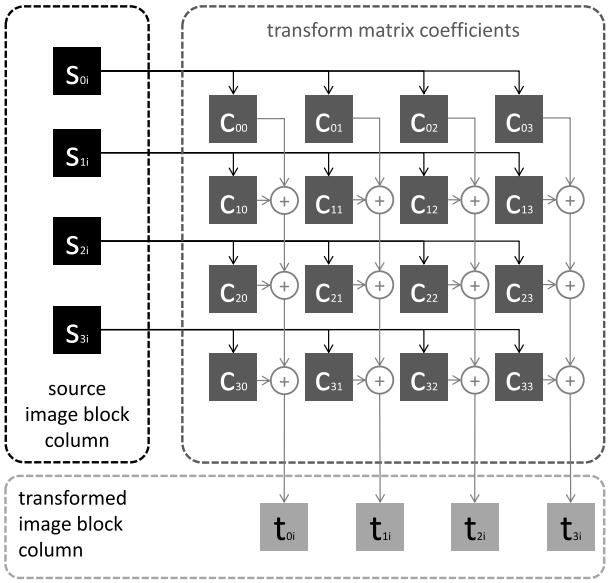

FIGURE 2. One-dimensional transform circuit: row by column matrix multiplication of one source image block column by the transform matrix.

A. STANDALONE ARCHITECTURES

Standalone architectures are capable of executing only one specific transform. Focusing on the one-dimensional transform circuits, that are the core of the developed architectures, they are basically composed by multiply-and-accumulate operations performing the row by column matrix multiplication. The multiplication of one column of source image by all the rows of the coefficients matrix is performed at each clock cycle. Being the coefficients 4 × 4 matrices, 16 parallel multiplications are required for each one-dimensional transform circuit, as shown in Figure 2. Due to the fact that multiplications are extremely resource and power consuming operations, a multiplierless implementation has been preferred. In this case, leveraging on the fact that coefficients are constant, it is possible to replace multiplications with shifters and adders, thus simplifying the circuit, as depicted in Figure 3.a.

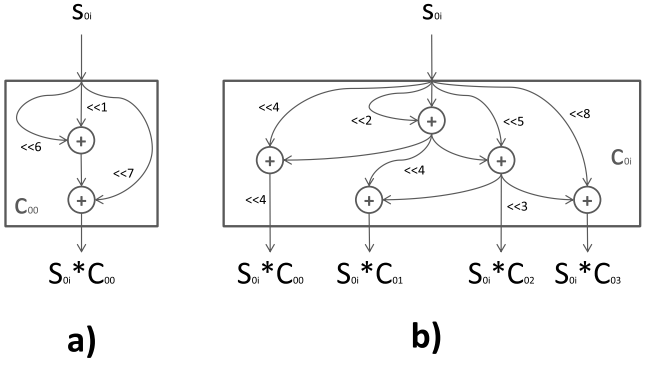

FIGURE 3. Example of multiplierless implementation: a) simple multiplication by 194; b) MCM-parallel multiplication by 336, 269, 219 and 117.

One of the algorithms proposed in [7], MCM-parallel, is capable of optimizing the number of shifters and adders

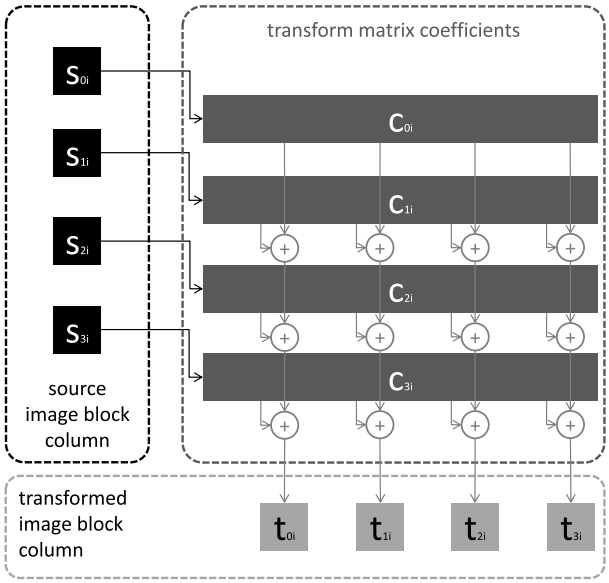

FIGURE 4. One-dimensional transform circuit with MCM-parallel blocks.

when the same data has to be multiplied by several coefficients. In the considered row by column matrix multiplication, each element of the input image block has to be multiplied by 4 different coefficients, one per each row of the transform matrix. Thus, the 16 parallel multiplications theoretically required by the circuit can be simplified by 4 MCM-parallel blocks, each providing 4 products in parallel (see Figure 3.b). The resulting one-dimensional transform circuit is depicted in Figure 4, and it resembles the same implementation proposed in [6].

B. RECONFIGURABLE ARCHITECTURES

Reconfigurable architectures are capable of elaborating all the five transforms (DCTII, DCTV, DCTVIII, DSTI, DSTVII) involved in AMT, one at a time, with the same circuit. Starting from the standalone architecture with 16 blocks seen in Section III-A, reconfigurability can be reached by letting blocks perform a different multiplication, with different coefficients, according to the desired transform, which can be specified by a proper identifier, the *transform_ID*. Then, a first reconfigurable version of the architecture can be derived directly from the standalone one, combining all the blocks of the different transforms that correspond, in terms of position, to the same coefficient. A simplified example of such combination is depicted in Figure 5, limited to the first coefficient of transforms DCTV (194) and DSTII (117). Of course, doing it manually takes lot of time and digital system design skills. In fact, here 194 is obtained as:

$$\begin{aligned} S_{0i} * 194 &= S_{0i} * (2 + 64 + 128) \\ &= S_{0i} * (1 \ll 1 + 1 \ll 6 + 1 \ll 7) \end{aligned} \quad (3)$$

as depicted in Figure 5.a, while 117 is resulting from:

$$\begin{aligned} S_{0i} * 117 &= S_{0i} * (1 + 4 + 16 + 32 + 64) \\ &= S_{0i} * (1{+}1 \ll 2{+}1 \ll 4{+}1 \ll 5{+}1 \ll 6) \end{aligned} \quad (4)$$

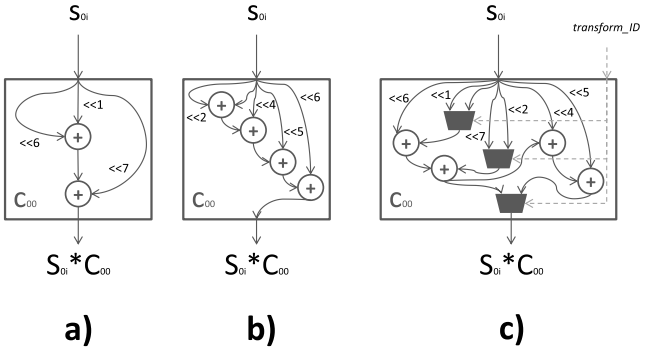

FIGURE 5. Example of multiplierless implementation: a) simple multiplication by 194; b) simple multiplication by 117; c) MCM-multiplexed multiplication: 194 or 117.

as shown in Figure 5.b. Combining such kind of circuits, in a way that one out of the two products is provided at a time, is not that trivial, considering also the fact that in AMT you need to deal with four products instead of two. Finally, more than one solution is possible since: i) the output of one adder can, in turn, be shifted; and ii) subtractions may be used, having the same circuital complexity of adders.

One simple possible combination is illustrated in Figure 5.c, where two adders and one shifter are shared between the two datapaths. In order to combine them, three multiplexers have been inserted, which are driven by the configuration signal *transform_ID*. When product of S_{0i} by 194 is required, *transform_ID* drives multiplexers so that they consider only left side inputs; while, when 117 has to be implemented, right side inputs are enabled by *transform_ID*.

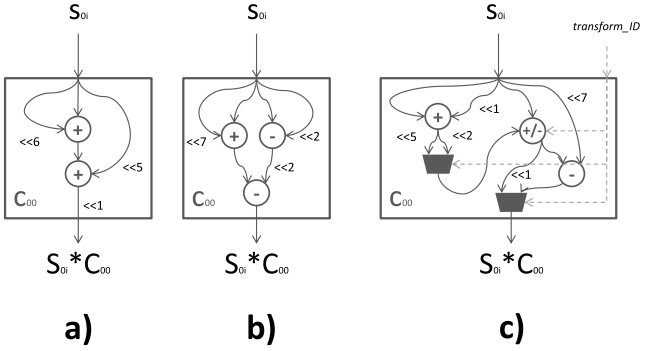

FIGURE 6. Example of optimized multiplierless implementation: a) simple multiplication by 194; a) simple multiplication by 117; c) MCM-multiplexed multiplication: 194 or 117.

Combining multiplierless circuits for constant multiplications to implement time-multiplexed products is again a complex and error prone task by construction. Adding optimization makes things even worst. Figure 6 depicts a possible optimized version of the same circuits of Figure 5. Here, while the product by 194 is basically the same (see Figure 6.a), the one by 117 is obtained with less shifters

and adders by leveraging on subtractors and shifting the adders/subtractors outputs, as depicted in Figure 6.b:

$$\begin{aligned} S_{0i} * 117 &= S_{0i} * ((128 + 1) - (4 - 1) * 4) \\ &= S_{0i} * ((1 \ll 7 + 1) - (1 \ll 2 - 1) \ll 2) \end{aligned} \quad (5)$$

With respect to the implementation shown in Equation 4, one less shifter and one less adder/subtractor are employed. The resulting combined circuit, capable of multiplexing in time products by 194 and 117, is provided by Figure 6.c. Here, the benefits present on the optimization of multiplication by 117 are still visible since the final circuit is requiring one less resource respectively in terms of shifters and adders/subtractors. Please note that the number of multiplexers, or configuration points, is the same since the $+/-$ operator embeds a multiplexer to decide if sum or subtraction has to be implemented. The MCM-multiplexed algorithm [29] is capable of relieving designers from the burden of designing and optimizing such circuits, by automatically combining an arbitrary number of constants, which products have to be multiplexed in time. However, it is not possible to guarantee the advantages offered by MCM-parallel algorithm [28], that pushes throughput and resource sharing among parallel products.

1) MCM_PAR ARCHITECTURE

Dealing with reconfigurable architectures, it is then still possible to exploit multiplierless circuitry algorithms presented in [7]. Considering the MCM-parallel algorithm, a trivial solution for a comprehensive AMT reconfigurable architecture is multiplexing in time standalone MCM-parallel circuits (with 1 input and 4 outputs multiplierless blocks, as explained in Section III-A) according to the *transform_ID*. An example of such kind of architecture is depicted in Figure 7. In this case each one-dimensional transform circuit will involve 20 different MCM-parallel blocks: for every image input among the 4 belonging to the same column, 5 different blocks are required, one per transform. Please note that this solution requires a low effort to the designer since MCM-parallel blocks have only to be multiplexed in time. Thus just a bit more additional logic, for multiplexing and connecting modules, has to be placed externally to the blocks, provided directly by the algorithm.

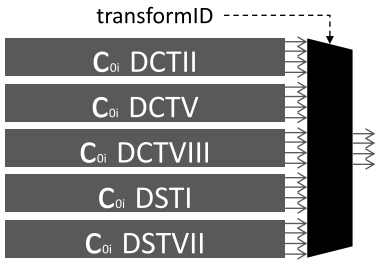

FIGURE 7. Reconfigurable 1-input 4-outputs multiplication block for the MCM_PAR architecture.

2) MCM_MUX ARCHITECTURE

Besides MCM-parallel, the authors of [7] also presented an algorithm to optimize circuits that are still capable of performing different multiplications, but providing only one product at a time. Such algorithm, the MCM-multiplexed, has been adopted to derive a second reconfigurable architectural solution composed by 16 multiplierless blocks. Each of them, obtained with MCM-multiplexed, performs one out of five products, corresponding to the five coefficients of the AMT matrices in the same specific position, according to the current *transform_ID*. The resulting 1 input by 4 coefficients multiplication circuit is illustrated in Figure 8.

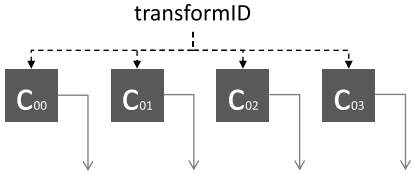

FIGURE 8. Reconfigurable 1-input 1-output multiplication blocks for the MCM_MUL and MCM_MIX0 architectures.

In this case the effort required to the developer is also extremely low. It is only necessary to connect the overall 16 MCM-multiplexed blocks generated by the automated algorithm, without touching them internally.

3) MCM_MIX0 ARCHITECTURE

MCM-parallel and MCM-multiplexed optimize different aspects of the circuit: the former focuses on performance, parallelizing computation of several outputs all together; the latter improves resources, by multiplexing in time the same adders and shifters used to perform different products. While trying to exploit the benefits of both MCM approaches, additional architectures have been derived. One first solution can be obtained transforming the MCM-parallel circuits in multiplexed ones. More precisely, by exploiting the MCM-parallel approach, multiplierless blocks capable of providing the parallel multiplication of a given input by the coefficients on a specific position of the 5 different transform matrices have been generated. These blocks produce 5 parallel outputs corresponding to the multiplication of the input by 5 coefficients in the same position coming from the 5 AMT transform matrices. In order to make it reconfigurable and perform one transform at a time, these outputs have to be multiplexed according to the *transform_ID* signal (multiplexers have been inserted internally with respect to the MCM-parallel blocks), leading to the same architectural scheme shown in Figure 8. Overall, 16 different multiplierless blocks will be embedded within the one-dimensional transform circuit. This solution is a bit less cheap than previous ones in terms of design effort, requiring both connection (external to the blocks) and multiplexing (internal to the blocks) modifications.

4) MCM_MIX1 ARCHITECTURE

Another solution to get the best of both MCM approaches [7] can be achieved by modifying the MCM-multiplexed circuit, which is reconfigurable by definition, to make it parallel. Indeed, it is possible to define a multiplierless block capable of providing, for a given source image element, the 4 required products for all the 5 AMT matrices, resulting in 20 different possible outputs by the same block. According to the MCM-multiplexed approach, these outputs are multiplexed in time, meaning that only one of them can be computed at a time. Four MCM-multiplexed blocks, one for each source image column element, are necessary to provide a complete one-dimensional transform circuit. Starting from this circuit, providing 1 out of 20 outputs at a time, it is possible to derive a solution producing 4 out of 20 outputs in parallel at a time, according to the given *transform_ID*, as shown in Figure 9.

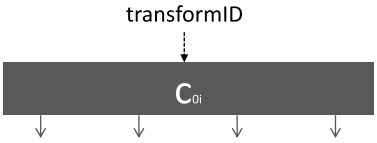

FIGURE 9. Reconfigurable 1-input 4-outputs multiplication block for the MCM_MIX1 architecture.

Here, besides performing a selector merging, since the 4 selectors of the involved parallel outputs have to be activated together, also a circuital modification is needed. In fact, some of the shifters and adders that were shared due to their mutual exclusivity when the corresponding products were multiplexed, have to be replied to allow parallelization. These modifications require to manipulate the design internally to the generated MCM-multiplexed blocks, since reconfigurability and optimization management goes beyond the external connection logic required in the other cases. Thus, contrarily to the other solutions, to obtain the MCM_MIX1 architecture, a significant design effort is required since 4 different 20-output MCM-multiplexed blocks have to be modified. Please note that MCM_MIX1 architecture is similar to the manually developed work at the state of the art described in [6]. In fact, this latter exploits a reconfigurable approach, without leveraging on any automated optimization strategy.

5) MCM_MIX2 ARCHITECTURE

The last solution tries to exploit the MCM-multiplexed approach without block modifications. MCM_MIX2 is composed of four blocks capable of providing 1 out of 20 possible products at a time for each source image column element, which is the same principle applied in MCM_MIX1. In order to have together all the 4 outputs required by the specific row by column matrix product, this circuit can be overclocked with a frequency that is four times faster than the one of the rest of the system. In such a way, on each quarter of the main clock period, one different product will be computed and, at the end of the whole period, all the 4 necessary products will be available. Due to the fact that the circuit

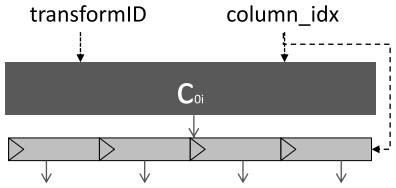

FIGURE 10. Reconfigurable 1-input 4-outputs multiplication block for the MCM_MIX2 architecture.

is re-used for computing different products in different time periods, additional storing resources, a bank of 4 registers for a total amount of 76 bits overall, are required to store the already computed products while the others are processed, as can be noticed in Figure 10. An additional selector signal, *column_idx*, is adopted to identify the current quarter period, corresponding to an element on a different column of the coefficients row.

For the MCM_MIX2 architecture, the designer effort is all external to the 4 involved MCM-multiplexed blocks, and it is limited to the additional logic necessary for overclocking and connecting them. Internally, the blocks are kept as they are provided by the MCM-multiplexed algorithm.

C. DEVELOPED ARCHITECTURES SUMMARY

In this section a short summary about the developed architectures, their composition, features and required effort to the designer is presented. We overall developed 10 architectures: 5 standalone and 5 reconfigurable. Table 1 depicts all the data related to these architectures. Please note that for standalone architectures only one entry is provided for all the 5 transforms, since they have been designed in the same way and present the same composition, features and effort.

TABLE 1. Summary of the developed architectures (numbers refer to one-dimensional transform circuits). * Blocks parallel outputs (number of outputs produced together). ** Blocks multiplexed outputs (number of outputs or groups of parallel outputs multiplexed in time). *** Kind of modification (internal or external to the automatically provided MCM blocks).

design	blocks number	blocks* par out	blocks** mux out	mod***	effort
standalone	4	4	0	external	low
MCM_PAR	20	4	0	external	low
MCM_MUX	16	1	5	external	low
MCM_MIX0	16	1	5	internal external	medium
MCM_MIX1	4	4	5	internal external	high
MCM_MIX2	4	1	20	external	medium

From Table 1 it is possible to see how the first three architectures, standalone, MCM_PAR and MCM_MUX, required a low effort since only modifications external to the multiplierless blocks have been necessary. The following three cases, instead, require both internal and external additional logic (MCM_MIX0, MCM_MIX1) or sequential external logic (MCM_MIX2) and, in turn, a medium to high design effort. It is worth to mention that effort is also an index of the digital

hardware design skills needed to derive the architectures from the output of the automated algorithms. Dealing with the remaining columns, related to the adopted multiplierless blocks and considering a one-dimensional transform circuit, it is possible to see how MCM_MIX1, among the reconfigurable architectures, is the unique solution providing parallel multiplexed circuits, that are blocks capable of multiplexing 5 different groups of 4 parallel outputs each. All the other cases give, instead, alternative solutions by adopting less parallel or less multiplexed blocks.

IV. RESULTS

In this section all the proposed solutions for standalone and reconfigurable AMT hardware implementations are evaluated. First of all, the benefits of MCM-parallel algorithm are assessed on the five different standalone architectures (see Section IV-A.1). Then, the exploration of the proposed novel reconfigurable solutions is evaluated (see Section IV-A.2); and finally, the reconfigurable architectures are compared with the unique reconfigurable AMT implementation available in literature (see Section IV-B).

All the reported data are gathered with Xilinx Vivado, targeting an Artix-7 XC7A100T FPGA board. Resource numbers are referred to post-implementation designs setting the frequency to the maximum achievable one, always individually reported.

A. ARCHITECTURAL EXPLORATION

In this section, the conducted architectural exploration considering both standalone and reconfigurable architectures is presented. All the designs discussed here are capable of performing a whole 4×4 two-dimensional transform.

1) STANDALONE ARCHITECTURES

Standalone architectures are circuits dedicated to the elaboration of one specific transform out of the five AMT ones. In order to understand the performance benefits of the adopted optimization algorithms [7], the MCM-parallel technique has been used to optimize multiplierless architectures manually developed. Please note that we are using the following format for naming: *DTT original* designs, where *DTT* has to be replaced with the specific transform implemented by the design, are the manually developed solutions; *DTT MCM_PAR*, where *DTT* has the same meaning of the previous case, are the optimized solutions. The considered *DTT* here are the five involved in the AMT. The assessment of these architectures is present in [6], but it has been repeated here since they are not available open-source.

Table 2 depicts the resource and frequency performance for all the five AMT transforms standalone designs. In terms of resources, LUTs and FFs, the MCM-parallel adoption is leading to savings with respect to all the *original* architectures. DCTVIII and DSTVII constitute the best cases since here MCM-parallel reduces the number of LUTs by more than 30%. This is due to the specific DCTVIII and DSTVII transform matrices that are more complex than the others.

TABLE 2. Standalone architectures evaluation.

design	LUTs	FFs	Freq [MHz]
DCTII original	2452	338	65.06
DCTII MCM_PAR	1948	338	63.98
original vs *MCM_PAR*	-20.6%	-0.0%	-1.7%
DCTV original	3178	365	62.81
DCTV MCM_PAR	2396	357	63.37
original vs *MCM_PAR*	-24.6%	-2.2%	+0.9%
DCTVIII original	2975	358	63.09
DCTVIII MCM_PAR	2031	357	66.09
original vs *MCM_PAR*	-31.7%	-0.3%	+4.8%
DSTI original	2705	360	62.66
DSTI MCM_PAR	2285	359	69.78
original vs *MCM_PAR*	-15.5%	-0.3%	+11.4%
DSTVII original	2975	359	62.38
DSTVII MCM_PAR	2041	357	67.20
original vs *MCM_PAR*	-31.4%	-0.6%	+7.7%

Their rows have often all the elements different, thus the sharing of the related common resources is more effective than having rows with equal elements, where synthesizer is capable of performing similar optimizations by itself. Dealing with frequency, benefits are not always present, e.g. for DCTII the *MCM_PAR* architecture is losing 1.7%. It seems that a valid and general motivation for the related trend is not present, but it depends on the synthesizer choices to find the best trade-off between resources and frequency.

2) RECONFIGURABLE ARCHITECTURES

Reconfigurable architectures are capable of elaborating all the five AMT transforms, one at a time, with the same circuit. Taking as reference a reconfigurable design, *reconf original*, manually derived from standalone *original* designs, five different possible architectural solutions have been developed. They have been designed trying to exploit the benefits of both MCM-parallel and MCM-multiplexed optimization techniques [7]. The format for names is the same presented in Section III, where these architectures are described in detail, preceded by *reconf* in order to better differentiate them from standalone designs.

TABLE 3. Reconfigurable architectures exploration.

design	LUTs	FFs	Freq [MHz]
reconf original	5723	356	58.14
reconf MCM_PAR	8021	356	58.14
original vs *MCM_PAR*	+40.2%	+0.0%	**+0.0%**
recof MCM_MUX	6603	355	54.35
original vs *MCM_MUX*	**+15.4%**	**-0.3%**	-6.5%
reconf MCM_MIX0	6697	367	55.07
original vs *MCM_MIX0*	+17.0%	+3.1%	-5.3%
reconf MCM_MIX1	7192	355	52.08
original vs *MCM_MIX0*	+25.7%	**-0.3%**	-10.4%
reconf MCM_MIX2	4777	1093	8.09
original vs *MCM_MIX2*	**-16.5%**	+207.0%	-86.1%

Table 3 depicts the results, in terms of resources and maximum operating frequency, for all the considered

reconfigurable architectures. In terms of LUTs, the MCM-multiplexed overclocked design (*reconf MCM_MIX2*) is the best solution, capable of reaching more than 15% of saving with respect to the *reconf original* design. However, this saving is heavily paid in terms of the other metrics: the overclocking requires more than 200% more FFs and a frequency drop of more than 85% with respect to the *original* design. In general the FF amount for all the other architectures, including *reconf original* is almost the same , due to the fact that they do not differ for the sequential logic, but for the combinational one, and the small differences are due to synthesis choices in order to optimize resources and frequency. The frequency, that is directly impacting on performance of the system, is lower than the *original* one for all the developed architectures but the *MCM_PAR* where it is equal, 58.14 MHz, with respect to the considered reference. One important, additional consideration has to be done here, regarding the effort required to the designer for developing each solution. The *reconf original* architecture has been manually designed and optimized, which implies big effort and advanced digital hardware design skills. On the contrary, all the proposed solutions exploit automated algorithms to derive the architecture. In particular, most of the proposed solutions, such as *reconf MCM_PAR*, *reconf MCM_MUX*, *reconf MCM_MIX0* and *reconf MCM_MIX2*, require little to medium designer intervention on the automatically generated circuits; while the other, *reconf MCM_MIX1*, still needs a considerable development effort (see also Table 1).

B. COMPARISON WITH MERT ET AL. [6]

In this section, a comparison with the sole literature work proposing a reconfigurable AMT implementation [6] is provided. To have a fair comparison, only reconfigurable AMT hardware architectures are considered, while traditional, non-reconfigurable ones are omitted. Here reconfigurable is intended as the capability of executing all the five AMT transforms, one at a time, with the same circuit. In order to perform a direct comparison with [6], the reported data is referred to one-dimensional transform circuits. Moreover, due to the different target boards and technologies adopted by authors in [6], but also to the fact that our proposed solutions are not yet integrated within the full video codec, the comparison has been made simply in terms of numbers of required instances of adders and shifters. Adders and shifters numbers in [6] are given for a circuit composed of two 4×4 one-dimensional transforms capable of performing two 4×4 one-dimensional transforms in parallel or one 8×8 one-dimensional transform. Therefore, it has been necessary to combine two of our 4×4 designs to obtain the same behavior.

From Table 4, the reconfigurable circuit proposed in [6], *reconf [6]*, and manually developed without the direct exploitation of the algorithms proposed in [7], seems to be generally very efficient with respect to the ones proposed in this work. However, there are some exceptions: the *reconf MCM_MIX1* solution is capable of achieving better results than *reconf [6]* in terms of shifters, with 16% of saving,

TABLE 4. Comparison of proposed designs with the ones presented in [6].

design	adders	shifters
reconf ([6])	168	248
reconf (this work)	271	278
this work vs [6]	+61.3%	+12.1%
reconf MCM_PAR (this work)	205	256
this work MCM_PAR vs [6]	+22.0%	+3.2%
reconf MCM_MUX (this work)	169	306
this work MCM_MUX vs [6]	+0.6%	+23.4%
reconf MCM_MIX0 (this work)	249	300
this work MCM_MIX0 vs [6]	+48.2%	+21.0%
reconf MCM_MIX1 (this work)	183	208
this work MCM_MIX1 vs [6]	+8.9%	**-16.1%**
reconf MCM_MIX2 (this work)	67	130
this work MCM_MIX2 vs [6]	**-60.1%**	**-47.6%**

while requiring only about 8% more adders. Please note that, even if *reconf MCM_MIX1* leverages on MCM-multiplexed for automatically deriving 16 blocks, each with 5 multiplexed outputs, it requires manual modifications to group them by 4, combining the internal adders and shifters. Another exception is given by the *reconf MCM_MIX2* solution, that is actually overperforming *reconf [6]* on both adders and shifters metrics, with 60% and 48% saving respectively. In this case, the full multiplexed solution generating 4 blocks with 20 multiplexed outputs per each 4×4 one-dimensional transform circuit employs a very small amount of resources. This is due to the fact that resources adopted to perform different multiplications can be shared even if they belong to the same row by column product. Please note that, differently from *reconf MCM_MIX1* and also from *reconf [6]*, *reconf MCM_MIX2* does not require much effort to the designer, since the circuits generated by MCM-multiplexed are adopted as they are without any further internal modification. To have a fair overview of the comparison, it is necessary to consider also that *reconf MCM_MIX2* requires additional sequential logic and has a significantly lower operating frequency with respect to all the other designs reported in Table 4.

V. GUIDELINES

In this work an exploration of possible reconfigurable implementations of the AMT, that will be adopted in the future video codecs, has been presented. More specifically, the addressed problem is related to the efficient implementation of multiple constant multiplication (required during transform operations) by means of circuits composed only by adders and shifters. It has to be noticed that what has been found here can be generalized to any case where multiple constant multiplication with multiplexed parallel products are required. Here parallel means that $N > 1$ products are required at a time, while multiplexed means that one out of M possible groups of parallel products has to be output according to the current value of a given selector (*transform_ID* in our case). In such cases, it is possible to derive a kind of guideline to provide an optimized multiplierless implementation exploiting the algorithms presented in [7]. If the most important metric is combinational logic (LUTs) and designer

effort, while an increase of sequential logic (FFs) and a drop in frequency can be afforded, then:

1) adopt MCM-multiplexed to generate multiplierless circuits with all the $N \times M$ products multiplexed;
2) overclock the generated circuits with a frequency M times faster than the main clock one, and let them generate one of the N products (that are required in parallel) at each fast clock cycle;
3) store the N products generated progressively by the circuit on additional sequential logic.

Conversely, it is possible to sacrifice the performance in terms of combinatorial logic and designer effort, to not get penalties in terms of sequential logic and maximum achievable frequency:

1) adopt MCM-multiplexed to generate multiplierless circuits with all the $N \times M$ products multiplexed;
2) combine together groups of N outputs, so that they produce one of the groups out of M overall groups at a time.

If all the considered metrics are important, including the designer effort, a suitable choice is to:

1) adopt MCM-multiplexed to generate N separated MCM-multiplexed circuits, each capable of providing one out of M different products.

Following such guidelines the designer, depending on the requirements and on his/her own skills and time, can decide which solution is mostly suitable for the specific context.

VI. CONCLUSION

Several application fields in electronics require strong hardware implementation optimization due to the ever increasing complexity and demands of the constantly evolving underlying algorithms. To meet time to market constraints, designers are additionally required to quickly perform such optimizations, pushing further their effort. Automated optimization algorithms and re-use can then be useful to reduce the effort, reach more efficient solutions and save money.

Starting from a video coding transformation case, in this work an exploration of different ways to apply optimization algorithms on multiple constant multiplication circuits has been conducted. Please note that, such kind of circuits is adopted in several fields of applications, such as two-dimensional filtering, color space conversion or polyphase filter banks.

An architectural exploration of five novel and reconfigurable hardware solutions for the transformation phase in the future video coding standard has been provided. Results demonstrated that, according to the way algorithms are applied, the corresponding architectural solutions reach different trade-offs, then representing viable options under certain metrics. Moreover, such solutions, almost automatically generated, quantitatively compared with the unique similar literature work, manually developed and optimized, revealed to be similar and, in some cases, better than this latter.

Finally, a set of guidelines that, generalizing the algorithms adoption process, open to the possibility of availing on them in different application contexts facing similar issues (multiplierless constant multiplications where multiplexing and parallelization of the products is required) have been also provided.

ACKNOWLEDGMENT

The authors would like to thank all the people in UNICA and UNISS who contributed to the FitOptiVis project (https://fitoptivis.eu/), ECSEL Programme under Agreement 783162.

REFERENCES

[1] E. Keating and G. Mirus, "American sign language in virtual space: Interactions between deaf users of computer-mediated video communication and the impact of technology on language practices," *Lang. Soc.*, vol. 32, no. 5, pp. 693–714, 2003.

[2] J. Spitz, P. Moors, J. Wagemans, and W. F. Helsen, "The impact of video speed on the decision-making process of sports officials," *Cogn. Res., Principles Implications*, vol. 3, no. 1, 2018, Art. no. 16.

[3] M. Singh, S. Singh, J. Jaiswal, and J. Hempshall, "Autonomous rail track inspection using vision based system," in *Proc. IEEE Int. Conf. Comput. Intell. Homeland Secur. Pers. Saf.*, Oct. 2006, pp. 56–59.

[4] *Qualcomm Snapdragon 4k Ultra HD*. Accessed: Jul. 30, 2019. [Online]. Available: https://www.qualcomm.com/media/documents/files/snapdragon-4k-datasheet.pdf

[5] Samsung. *Exynos 9820 Mobile Processor*. Accessed: Jul. 30, 2019. [Online]. Available: https://www.samsung.com/semiconductor/minisite/exynos/products/mobileproce/ssor/exynos-9-series-9820/

[6] A. C. Mert, E. Kalali, and I. Hamzaoglu, "High performance 2D transform hardware for future video coding," *IEEE Trans. Consum. Electron.*, vol. 63, no. 2, pp. 117–125, May 2017.

[7] *SPIRAL Project—Multiplierless Constant Multiplication*. Accessed: Jul. 30, 2019. [Online]. Available: https://www.spiral.net/hardware/multless.html

[8] L. Aksoy, P. Flores, and J. Monteiro, "A novel method for the approximation of multiplierless constant matrix vector multiplication," in *Proc. IEEE 13th Int. Conf. Embedded Ubiquitous Comput.*, Oct. 2015, pp. 98–105.

[9] K. Holm and O. Gustafsson, "Low-complexity and low-power color space conversion for digital video," in *Proc. NORCHIP*, Nov. 2006, pp. 179–182.

[10] O. Gustafsson, K. Johansson, H. Johansson, and L. Wanhammar, "Implementation of polyphase decomposed FIR filters for interpolation and decimation using multiple constant multiplication techniques" in *Proc. IEEE Asia Pacific Conf. Circuits Syst.*, Dec. 2006, pp. 924–927.

[11] S. De-Luxán-Hernández, D. Marpe, H. Schwarz, K.-R. Müller, M. Wien, J.-R. Ohm, and T. Wiegand, "Block adaptive selection of multiple core transforms for video coding," in *Proc. IEEE Picture Coding Symp.*, Dec. 2016, pp. 1–5.

[12] T. Biatek, V. Lorcy, P. Castel, and P. Philippe, "Low-complexity adaptive multiple transforms for post-HEVC video coding," in *Proc. IEEE Picture Coding Symp.*, Dec. 2016, pp. 1–5.

[13] A. Kammoun, S. B. Jdidia, F. Belghith, W. Hamidouche, J. F. Nezan, and N. Masmoudi, "An optimized hardware implementation of 4-point adaptive multiple transform design for post-HEVC," in *Proc. Int. Conf. Adv. Technol. Signal Image Process.*, Mar. 2018, pp. 1–6.

[14] C. Lucarz, J. Piat, and M. Mattavelli, "Automatic synthesis of parsers and validation of bitstreams within the MPEG reconfigurable video coding framework," *J. Signal Process. Syst.*, vol. 63, no. 2, pp. 215–225, 2011.

[15] C. Sau, L. Fanni, P. Meloni, L. Raffo, and F. Palumbo, "Reconfigurable coprocessors synthesis in the MPEG-RVC domain," in *Proc. Int. Conf. Reconfigurable Comput. FPGAs*, Dec. 2015, pp. 1–8.

[16] C. Pilato and L. P. Carloni, "DarkMem: Fine-grained power management of local memories for accelerators in embedded systems," in *Proc. Asia South Pacific Design Autom. Conf.*, Jan. 2018, pp. 696–701.

[17] A. Kammoun, W. Hamidouche, F. Belghith, J.-F. Nezan, and N. Masmoudi, "Hardware design and implementation of adaptive multiple transforms for the versatile video coding standard," *IEEE Trans. Consum. Electron.*, vol. 64, no. 4, pp. 424–432, Nov. 2018.

[18] S. B. Jdidia, A. Kammoun, F. Belghith, and N. Masmoudi, "Hardware implementation of 1-D 8-point adaptive multiple transform in post-HEVC standard," in *Proc. 18th Int. Conf. Sci. Techn. Autom. Control Comput. Eng. (STA)*, Dec. 2017, pp. 146–151.

[19] A. C. Mert, H. Azgin, E. Kalali, and I. Hamzaoglu, "Efficient multiple constant multiplication using DSP blocks in FPGA," in *Proc. 28th Int. Conf. Field Program. Logic Appl. (FPL)*, Aug. 2018, pp. 331–334.
[20] K. Compton and S. Hauck, "Reconfigurable computing: A survey of systems and software," *ACM Comput. Surv.*, vol. 34, no. 2, pp. 171–210, 2002.
[21] S. M. S. Trimberger, "Three ages of FPGAs: A retrospective on the first thirty years of FPGA technology: This paper reflects on how Moore's law has driven the design of FPGAs through three epochs: The age of invention, the age of expansion, and the age of accumulation," *IEEE Solid State Circuits Mag.*, vol. 10, no. 2, pp. 16–29, Jun. 2018.
[22] M. Wijtvliet, L. Waeijen, and H. Corporaal, "Coarse grained reconfigurable architectures in the past 25 years: Overview and classification," in *Proc. Int. Conf. Embedded Comput. Syst., Archit., Modeling Simulation (SAMOS)*, Jun. 2016, pp. 235–244.
[23] R. Zendegani, M. Kamal, M. Bahadori, A. Afzali-Kusha, and M. Pedram, "RoBa multiplier: A rounding-based approximate multiplier for high-speed yet energy-efficient digital signal processing," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 2, pp. 393–401, Feb. 2017.
[24] S. Hashemi, R. Bahar, and S. Reda, "DRUM: A dynamic range unbiased multiplier for approximate applications," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2015, pp. 418–425.
[25] C. Rafferty, M. O'Neill, and N. Hanley, "Evaluation of large integer multiplication methods on hardware," *IEEE Trans. Comput.*, vol. 66, no. 8, pp. 1369–1382, Aug. 2017.
[26] D. De, A. Ghosh, K. G. Kumar, A. Saha, and M. K. Naskar, "Multiplierless hardware realization of trigonometric functions for high speed applications," in *Proc. IEEE Appl. Signal Process. Conf.*, Dec. 2018, pp. 149–152.
[27] S. S. Sarwar, S. Venkataramani, A. Raghunathan, and K. Roy, "Multiplierless artificial neurons exploiting error resiliency for energy-efficient neural computing," in *Proc. Conf. Design, Autom. Test Eur.*, 2016, pp. 145–150.
[28] Y. Voronenko and M. Puschel, "Multiplierless multiple constant multiplication," *ACM Trans. Algorithms*, vol. 3, no. 2, p. 11, 2007.
[29] P. Tummeltshammer, J. C. Hoe, and M. Püschel, "Time-multiplexed multiple-constant multiplication," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 9, pp. 1551–1563, Sep. 2007.

CARLO SAU received the degree in electronic engineering and the Ph.D. degree from the University of Cagliari, in 2012 and 2016, respectively, where he is currently an Assistant Professor. Since 2012, he has been involved in automated methodologies for dataflow-based reconfigurable platforms generation. His main research focus is related to reconfigurable system design and development of code generation tools for advanced reconfigurable hardware architectures.

DARIO LIGAS received the degree in electronic engineering from the University of Cagliari, in 2019, with a work entitled: Optimization of Reconfigurable Multiplierless Devices for Video Coding Applications.

TIZIANA FANNI received the degree in electronic engineering and the Ph.D. degree from the University of Cagliari, in 2014 and 2019, respectively. She is currently a full-time Researcher with the Department of Electrical and Electronic Engineering, University of Cagliari. Since 2014, she has been involved in power saving methodologies in dataflow-based reconfigurable platforms. Her main research focus is related to reconfigurable systems design and development of code generation tools for low power reconfigurable hardware architectures.

LUIGI RAFFO received the Laurea Degree in electronic engineering and the Ph.D. degree in electronics and computer science from the University of Genoa, Italy, in 1989 and 1994, respectively. In 1994, he joined the Department of Electrical and Electronic Engineering, University of Cagliari, Italy, as an Assistant Professor and as an Associate Professor, in 1998. He has been a Full Professor of electronics with the Department of Electrical and Electronic Engineering, University of Cagliari, Italy, since 2006. He teaches courses on system/digital and analog electronic design and processor architectures for the Courses of studies in electronic and biomedical engineering. He was a Coordinator of the project EU IST- FET - IST-2001-39266 - BEST and the MADNESS EU Project (FP7/2007-2013). He was also an Unit Coordinator of the project EU IST-FET - SHAPES - Scalable Software Hardware Architecture Platform for Embedded Systems. He is also a Local Coordinator of industrial projects in the field (among others: ST-Microelectronics - Extension of ST200 architecture for ARM binary compatibility and ST-Microelectronics - Network on chip). He is responsible for the cooperation programs in the field of embedded systems with several other European Universities. He was a Local Coordinator of the ASAM (ARTEMIS-JU) and ALBA projects (national founded project) and RPCT (regional founded project).

FRANCESCA PALUMBO received the Laurea Degree *(summa cum laude)* in electronic engineering from the University of Cagliari, in 2005, the master's degree in advanced in embedded system design from the Advanced Learning and Research Institute, University of Lugano, in 2006, and the Ph.D. degree in electronic and computer engineering from the University of Cagliari. She is currently an Assistant Professor with the Intelligent System Design and Applications (IDEA) Group, University of Sassari. At the moment, she is also a Scientific Coordinator of the CERBERO (ID: 732105) H2020 European Project on Smart Cyber Physical System Design. Her research focus is related to reconfigurable systems and to code generation tools and design automation strategies for advanced reconfigurable hardware architectures. For her studies in the fields of dataflow-based programming and hardware customization, she received the two best paper awards from the Conference on Design and Architectures for Signal and Image Processing, in 2011 and 2015, with the works entitled "The Multi-Dataflow Composer tool: A runtime reconfigurable HDL platform composer" and "MPSoCs for real-time neural signal decoding: A low-power ASIP-based implementation". She is a permanent Steering Committee Member of the ACM Conference on Computing Frontiers and became a member of the IEEE VLSI Systems & Applications Technical Committee (VSATC), in 2018. She serves in several different Technical Committee of international conferences. She also serves as an Associate Editor of the *Springer Journal of Signal Processing Systems*.