Broadening the Exploration of the Accelerator Design Space in Embedded Scalable Platforms

Luca Piccolboni, Paolo Mantovani, Giuseppe Di Guglielmo, Luca P. Carloni
Department of Computer Science, Columbia University, New York, NY, USA
Emails: {piccolboni, paolo, giuseppe, luca}@cs.columbia.edu

***Abstract*—Accelerators are specialized hardware designs that generally guarantee two to three orders of magnitude higher energy efficiency than general-purpose processor cores for their target computational kernels. To cope with the complexity of integrating many accelerators into heterogeneous systems, we have proposed Embedded Scalable Platforms (ESP) that combines a flexible architecture with a companion system-level design (SLD) methodology. In ESP, we leverage high-level synthesis (HLS) to expedite the design of accelerators, improve the process of design-space exploration (DSE), and promote the reuse of accelerators across different target systems-on-chip (SoCs). HLS tools offer a powerful set of parameters, known as knobs, to optimize the architecture of an accelerator and evaluate different trade-offs in terms of performance and costs. However, exploring a large region of the design space and identifying a rich set of Pareto-optimal implementations are still complex tasks. The standard knobs, in fact, operate only on loops and functions present in the high-level specifications, but they cannot work on other key aspects of SLD such as I/O bandwidth, on-chip memory organization, and trade-offs between the size of the local memory and the granularity at which data is transferred and processed by the accelerators. To address these limitations, we augmented the set of HLS knobs for ESP with three additional knobs, named eXtended Knobs (*XKnobs*). We used the *XKnobs* for exploring two selected kernels of the wide-area motion imagery (WAMI) application. Experimental results show that the DSE is broadened by up to 8.5x for the performance figure (latency) and 3.5x for the implementation costs (area) compared to use only the standard knobs.**

I. Introduction

High-performance embedded systems are increasingly realized with heterogeneous architectures that combine multiple general-purpose processor cores and a variety of special-function hardware accelerators [4], [8], [15]. By being tailored to execute a dedicated function, an accelerator offers major gains in both performance and energy efficiency compared to a corresponding software execution [7], [14]. On the other hand, the integration of many different specialized hardware blocks complicates the design, programming, and verification of the whole system, thereby increasing the non-recurring engineering costs.

To balance the growing demand for more hardware specialization with the need of maintaining helpful degrees of regularity and modularity, we have conceived and developed the concept of *Embedded Scalable Platforms (ESP)* that combines a system architecture and a companion methodology [6]. The architecture simplifies the integration of heterogeneous components because they can be encapsulated into modular sockets and connected through a scalable communication and control infrastructure (SCCI) [22]. The corresponding methodology aims at raising the level of abstraction of hardware design from the register-transfer level (RTL) to system-level design (SLD) [5], [31].

The ESP Architecture. The architecture of an ESP instance consists of a particular mix of tiles. Each tile may implement a processor core (capable of running an operating system like Linux), a hardware accelerator, or some auxiliary functionality, e.g., a DRAM interface. The number and mix of tiles of an architecture vary depending on its target application domain. The choice of a specific combination of tiles is the result of a *design-space exploration (DSE)* process driven by the target set of applications and the desired performance and costs.

The ESP Methodology. The design, programming and verification of an ESP instance is assisted by the ESP methodology [6], which is supported by both commercial computer-aided design (CAD) tools and in-house CAD tools for SLD developed in our group [13], [18], [27], [28], [29]. The methodology promotes the reuse of accelerators across different ESP instances by adopting *high-level synthesis (HLS)* for the automatic generation of multiple RTL implementations of each accelerator [23], [24]. The RTL implementations are derived from a single high-level specification described in SystemC, which is an IEEE-standard object-oriented programming language based on C++ [3], [16]. Replacing RTL designs with higher-level specifications written in SystemC helps designers raise the level of abstraction for the bulk of the design process, reduces the gap between software and hardware, allows fast full-system simulation under more significant application scenarios, and enables the exploration of different micro-architectural solutions by configuring the HLS parameters. These parameters are known as *HLS knobs* (or, simply, *knobs*) and their setting allows designers to explore many alternative RTL implementations. For example, the application of the "loop unrolling" knob leads to RTL implementations with more hardware resources, therefore delivering lower execution time in exchange of a higher area occupation and power dissipation. Conversely, "loop breaking" leads to more sequential executions on shared hardware, resulting in resource savings but lower performance. These implementations are obtained from the same specification, and they represent alternative trade-offs in a multi-objective design space.

Design-Space Exploration. The larger is the region of the design space covered by the different RTL implementations of each accelerator, the more reusable and flexible is their high-level specification. However, as the complexity of hardware accelerators and the number of components integrated in an ESP instance increase, exploring a large portion of their design space and finding a rich set of Pareto-optimal implementations become very challenging tasks. HLS tools, in fact, are limited to datapath and control logic transformations that apply to the loops and function calls present in the SystemC specification. However, they cannot directly operate on other relevant aspects of SLD that have the potential to unlock regions of the design space that are not reachable by only tuning the HLS knobs. These regions of the design space can only be reached by extensively modifying the SystemC specification. Indeed, by using only the HLS knobs the risk is to obtain unbalanced RTL implementations. For example, consider an accelerator implementation generated by applying extensive loop unrolling, which produces a highly-parallel datapath. The computation phase of the accelerator might be able to process data faster than the rate at which it is transferred over the system interconnect. Similarly, a highly-parallel datapath may lead to diminishing returns due the extra area occupation, if the organization of its local memory cannot exchange data at the necessary throughput. The organization and size of the on-chip memory has also a direct impact on the granularity at which data can be transferred between an accelerator and the external memory: larger local memories enable accelerators to operate with

978-1-5386-3472-1/17/$31.00 ©2017 IEEE

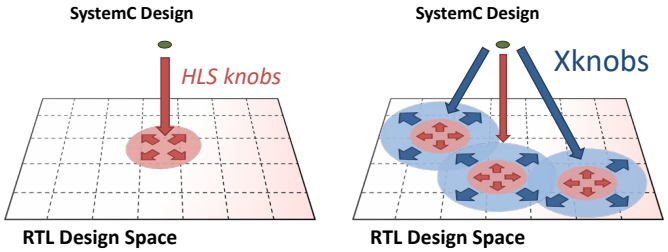


Fig. 1: Standard HLS knobs allow designers to create multiple RTL implementations from the same SystemC specification (left). The ***XKnobs*** further broaden the exploration of the design space (right).

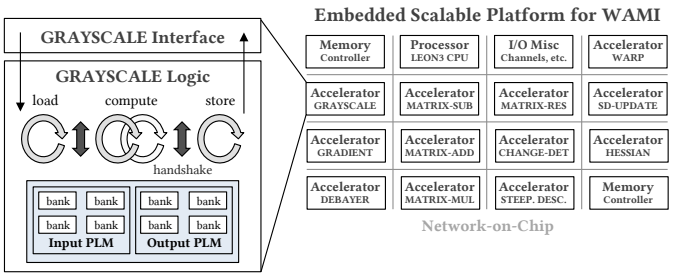


Fig. 2: Loosely-coupled accelerators in ESP for the WAMI application.

few long-burst transactions, as opposed to many short data transfers. When considering several concurrent accelerators contending for the interconnect infrastructure and memory accesses, the granularity of communication can have a relevant impact on the overall performance.

Contributions. To address the current limitations of the standard HLS knobs and to broaden the DSE of the accelerators in ESP, we propose to augment the standard HLS knobs with the *eXtended Knobs* (***XKnobs***). The ***XKnobs*** allow designers to modify significant aspects of the RTL implementations of the accelerators by parametrizing their SystemC specifications. In particular, the ***XKnobs*** allow designers to tune (i) the organization of the accelerators' private local memory (PLM), (ii) the granularity of data transfers, and (iii) the bandwidth of direct-memory access (DMA) channels. Fig. 1 depicts the main differences in applying the standard HLS knobs and the ***XKnobs***. The standard knobs expand the design space locally by operating on loops (e.g., loop unrolling or loop breaking) and functions. Conversely, the ***XKnobs*** expand the design space both locally (by increasing the number of different RTL implementations that can be synthesized) and globally (by unlocking new regions of the design space). We present an application of the ***XKnobs*** to the design of two accelerators for two computational kernels of the wide-area motion imagery (WAMI) application [30]. These accelerators are representative of two general classes: accelerators that have a larger *computation time* compared to the communication time, and accelerators dominated by the *communication time*. The results show that the ***XKnobs*** can broaden the exploration of the design-space spans up to a $8.5\times$ larger performance range and a $3.5\times$ larger area-occupation range.

II. Embedded Scalable Platforms and Accelerators

ESP combines a tile-based architecture and a system-level methodology that simplify both the combination of heterogeneous components (processors, accelerators) and the integration of any chosen alternative implementation of a certain accelerator, i.e., different Pareto-optimal RTL implementations [6]. This is possible thanks to the combined use of (i) the SCCI and (ii) a set of modular sockets [22]. The SCCI is realized with a bus or a network-on-chip (NoC), depending on the communication bandwidth required by the components [17]. A socket is a parameterized module that is synthesized from a generic template in order to encapsulate an accelerator into a tile and simplify its integration with the communication infrastructure. Fig. 2 shows an example of a 4×4 mesh of tiles that implements an instance of ESP realized for the WAMI application [30]. WAMI is an image processing application used in the context of aerial surveillance. It processes a sequence of input frames to extract masks of "meaningfully-changed" pixels. For example, WAMI can be used to detect and track vehicles moving on the ground, while discarding environmental noise, e.g., shadows, surface reflections, etc. A software specification of WAMI is available in C as part of the PERFECT Benchmark Suite [2], which is a collection of applications and kernels targeting energy-efficient high-performance embedded computing. We designed twelve accelerators in SystemC for WAMI as a case study [22]. Each accelerator implements a relevant computational kernel of WAMI. Fig. 2 shows an instance of ESP with the twelve accelerator tiles (**Accelerator**), one processor tile (**Processor**), one auxiliary I/O and miscellaneous tile (**I/O Misc**), and two memory tiles (**Memory Controller**) to provide access to DRAMs.

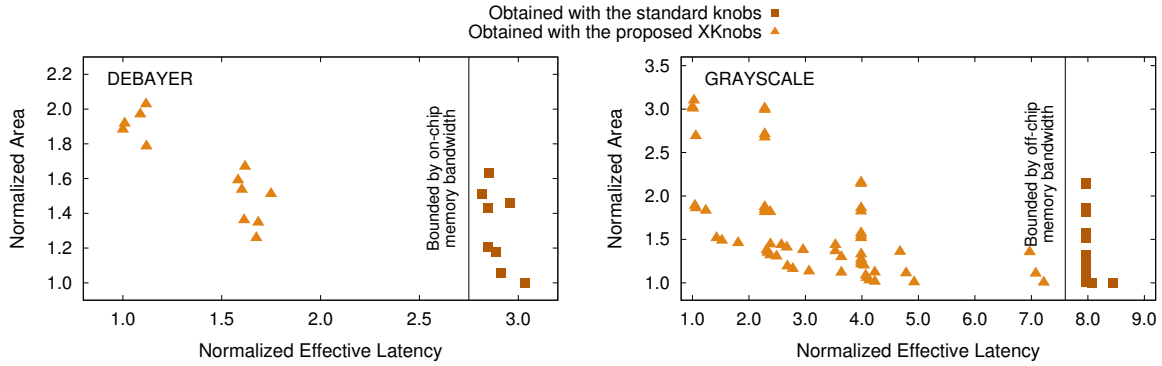
We followed the *loosely-coupled model* [11] to design our accelerators for WAMI: the accelerators are integrated in a ESP instance as devices managed with Linux device drivers. The main advantages of loosely-coupled accelerators consist in operating on large data sets and enabling coarse-grain computation and data transfer phases. Fig. 2 reports the typical structure of a loosely-coupled accelerator. A minimum of three SystemC processes handle the three main phases of the accelerator execution: (i) load input data (load process), (ii) perform the computation (one or more compute processes), and (iii) store the results (store process). The load and store processes interact with a memory controller (DMA) and transfer the data from the external memory to the PLMs of the accelerator. The input PLMs contain the input data, while the output PLMs contain the output data as well as partial results, if necessary. One or more computation processes realize the computational kernel of the accelerator.

For each accelerator, a socket in ESP provides a set of services: (i) memory-mapped registers and control circuitry that allow the device driver to configure and manage the accelerator; (ii) DMA engines that translate the requests from the load and store processes into actual transactions with memory; (iii) interrupt-request mechanisms that allow the accelerator to notify the invoking processor about the completion of its task or the occurrence of an exception. At the circuit-level, the ESP sockets allow the seamless replacement of the synthesized RTL implementation of a given accelerator with any other one taken from its Pareto-optimal set (obtained through HLS). This is simplified by the fact that both the sockets and the accelerators implement latency-insensitive interfaces, instead of relying on strictly synchronous handshake mechanisms specified through manual RTL design [5]. The processor socket has a similar set of services with some differences such as the presence of a cache instead of a DMA engine and the use of bus proxies to interface the processor bus protocol (e.g., AMBA bus) with the protocol of the communication infrastructure.

III. Motivational Examples

HLS allows designers to easily generate multiple RTL implementations with different performance figures and implementation costs by using the same high-level specification. Designers can obtain such a variety of implementations by setting the knobs provided by HLS tools [23], [24]. TABLE I reports some examples of these knobs, including (i) loop manipulations, e.g., loop unrolling, pipelining and

978-1-5386-3472-1/17/$31.00 ©2017 IEEE



(a) Accelerator with higher computation time.

(b) Accelerator with higher communication time.

Fig. 3: Limitations of the standard HLS knobs in exploring the design space of hardware accelerators in ESP.

TABLE I: The standard knobs provided by HLS tools.

Knob	Settings and Effects
LOOP MANIPULATIONS	• unrolling: replicates the operations in the loop body • pipelining: pipelines the operations in the loop body • breaking: inserts additional states in the loop body
ARRAY MAPPINGS	maps the array to registers or on-chip memories
CLOCK PERIOD	sets the target clock period for the synthesis

TABLE II: The ***XKnobs*** we propose in this paper.

Knob	Settings and Effects
PLM PORTS	• sets the number of read ports of input PLMs • sets the number of write ports of output PLMs
DMA WIDTH	• sets the size in bits of the DMA channels • sets the number of write ports of input PLMs • sets the number of read ports of output PLMs
DMA CHUNK	• sets the total size of the input and output PLMs • sets the amount of data exchanged through the DMA

breaking, (ii) array mappings, e.g., mapping the arrays specified in the code to either registers or on-chip memories, and (iii) low-level architectural choices, e.g., the clock period of the RTL descriptions.

The knobs in TABLE I (*standard knobs* for the rest of the paper) enable already a broad DSE as shown for example in [27], [32]. These knobs, however, are not enough for exhaustively exploring the possible implementations of an accelerator in ESP. In fact, these knobs work only on the loops and functions present in the high-level specification, but they do not operate on other important aspects of SLD, including on-chip and off-chip memory bandwidths. To highlight this limitation, Fig. 3 illustrates the DSE results for DEBAYER and GRAYSCALE, which are two representative accelerators of the WAMI application. The two graphs illustrate multiple RTL design points synthesized by targeting an industrial 32nm ASIC technology library and characterized in terms of normalized effective latency and normalized area[1]. In Fig. 3 the squares indicate the design points obtained with the standard knobs (by mapping the arrays in the code to standard dual-port memories). The triangles indicate additional design points obtained by applying the ***XKnobs*** (reported in TABLE II). In both examples, the ***XKnobs*** overcome the limitations of the standard knobs and broaden the DSE.

The DEBAYER accelerator is an image-processing component that takes as input an image in Bayer format. Each pixel of the Bayered image stores information for one of the three channels: red, green and blue (RGB). It restores the missing information by interpolating available data on sliding stencils, each sized 5×5 pixels. Multiple-nested loops characterize the SystemC specification of DEBAYER, and each stencil requires a relevant number of operations on few pixels. A high-performance implementation of DEBAYER requires the use of multi-port memories to enable multiple memory accesses at the same clock cycle (to sustain the parallelism in the multiple-nested loops). Unfortunately, the support for PLM generation and optimization is limited in current HLS tools. In fact, HLS tools often use dual-port memories and rely on third-party memory generators to obtain the RTL implementations of such memories [28]. Some HLS tools allow designers to increase the number of ports of the PLMs of accelerators, but they do not generate highly-optimized descriptions of such PLMs, as those proposed for example in [9], [29]. Thus, the resulting RTL implementations could be limited by the bandwidth to the on-chip memory. As a consequence, in these cases, applying loop unrolling, i.e., augmenting the number of available hardware resources, does not lead to significant performance gains. For example, on the right side of Fig. 3 (a), the area of the DEBAYER design points increases, but the effective latency remains essentially the same. To improve the performance it is necessary to generate multi-port PLMs with external memory generators [9], [29]. We propose to use them with a knob that tailors the PLMs to the needs of accelerators. This allows designers to broad the DSE, as shown on the left side of Fig. 3 (a).

The GRAYSCALE accelerator performs RGB to luminance conversion. The right side of Fig. 3 (b) illustrates a situation similar to DEBAYER. However, in this case the bandwidth to the off-chip memory (DRAM) is the factor that limits the DSE. Differently from DEBAYER, the GRAYSCALE accelerator is characterized by a higher communication time compared to the computation time. When an accelerator is limited by the communication time, applying loop unrolling leads to Pareto-dominated design points (see the squares on the right side of Fig. 3 (b)). HLS tools permit the definition of cycle-accurate interface protocols (e.g., FIFOs, buffers, etc.), but they provide limited support for the configuration of the channels to the off-chip memory [10], e.g., DMA channels. To overcome this limitation, we provide mechanisms that allow designers to easily adapt the off-chip memory bandwidth to the needs of accelerators. The left side of Fig. 3 (b) shows how the exploration of the accelerator design space can be significantly broadened.

IV. XKNOBS FOR THE DESIGN-SPACE EXPLORATION IN ESP

TABLE II reports the main characteristics of the proposed ***XKnobs***. This section describes the effects of each ***XKnob*** separately. Section V explains the effects of the simultaneous application of the ***XKnobs***.

[1]The effective latency is the product of the clock period and the clock cycle count. The area includes the logic area and the memory area. The normalization is applied with respect to the fastest (for the effective latency) and the smallest (for the area) implementations. We use these metrics for the rest of the paper.

978-1-5386-3472-1/17/$31.00 ©2017 IEEE

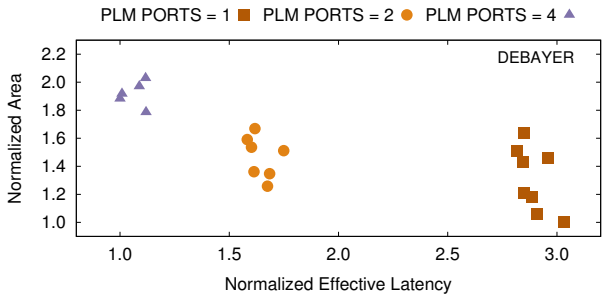


Fig. 4: Application of *PLM PORTS* to DEBAYER.

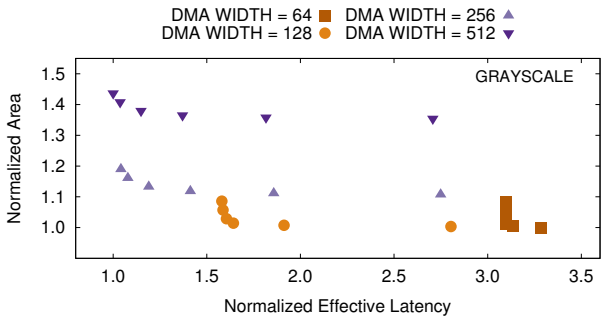


Fig. 5: Application of *DMA WIDTH* to GRAYSCALE.

A. PLM PORTS

Optimizing the PLM is fundamental for sustaining the performance of accelerators [11], [20]. Multi-bank memories allow accelerators to read and write more data per clock cycle by providing multiple read and write ports, respectively. However, to provide such parallelism, multi-bank memories occupy more area due to the larger number of banks they require (e.g., for realizing data duplication or distribution [1], [28]). To represent this trade-off, we define a ***XKnob***, called *PLM PORTS*, that allows designers to select the number of read ports of the input PLMs and the number of write ports of the output PLMs of the accelerators. Hence, designers can choose the number of read and write accesses that can be performed in a single clock cycle by the compute processes of the accelerators. To generate the memories with the specified number of ports we use MNEMOSYNE [29]. Note that *PLM PORTS* not only permits to change the number of ports of the memories for the HLS tools that do not support it, but it also allows designers to generate memories that are optimized for the memory access pattern of the accelerators, as discussed in [28], [29]. Fig. 4 shows an example of application of this ***XKnob*** to the design of DEBAYER. The colors (shapes) differentiate the design points depending on the value of *PLM PORTS*. For each value of *PLM PORTS* we synthesize multiple design points by applying the standard HLS knobs. When the accelerator uses only dual-port memories, with the standard knobs it is not possible to obtain significant performance improvements (see the squares in Fig. 4). In fact, increasing the hardware resources, e.g., with loop unrolling, is ineffective and results often in Pareto-dominated points. Conversely, by assigning different values to *PLM PORTS* we can explore other regions of the design space, achieving a latency span of $3.0\times$ and an area span of $2.0\times$.

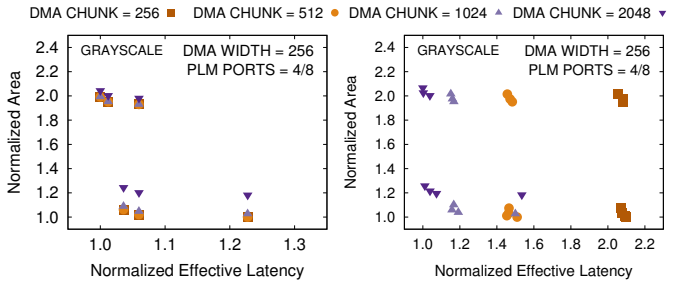
B. DMA WIDTH

Accelerators communicate with the off-chip memory through the DMA controllers. Some accelerators may be dominated by the communication time. Hence, they need to read and write multiple values at the same clock cycle to better balance communication with computation. To optimize the communication time of such accelerators, we define a ***XKnob***, called *DMA WIDTH*, that sets the width of the DMA channels. Note that if we increase the width of the DMA channels, we need also to increase the number of write ports of the input PLMs and the number of read ports of the output PLMs (TABLE II). In this way, the accelerators can access the memory at the same rate of the DMA interfaces. Fig. 5 shows the application of *DMA WIDTH* to the design of the GRAYSCALE accelerator. In this case, the colors (shapes) indicate different widths (in bits) of the DMA channels. For each configuration of *DMA WIDTH* we synthesize multiple design points by using the standard HLS knobs. When *DMA WIDTH* is 64 or 128 bits the accelerator is limited by the communication time and the standard knobs fail to explore significantly-different design alternatives (several points are also Pareto dominated). Instead, by setting *DMA WIDTH* to either 256 or 512 bits, it is possible to move the bottleneck of GRAYSCALE to the computation phase so that using the standard knobs becomes helpful in improving the accelerator performance. As a result, the design points are no more localized in a small region of the design space. In this case, by changing *DMA WIDTH*, we obtain a latency span of $3.3\times$ and an area span of $1.4\times$.

C. DMA CHUNK

The accelerators store the data in the PLMs during computation to optimize the ratio between communication and computation. PLMs can take from 40% to 90% of the area of the entire accelerators [11], [20], but their capacity is often not sufficient for storing the entire workloads. Consequently, the data that can be transferred at every interaction with the DMA controllers is also limited. On the other hand, the smaller is the amount of data transferred at every interaction, the higher is the overhead required for the communication. In ESP, the overhead for the communication increases when multiple accelerators run concurrently and compete for the shared resources, i.e., the communication infrastructure and the off-chip memory. To capture this important trade-off, we define a ***XKnob***, called *DMA CHUNK*, that indicates the amount of data that can be transferred at every interaction with the DMA controller. Hence, the sizes of the PLMs is directly related to its value. The amount of data transferred at each interaction is expressed as a multiple of the size of the data type stored in the PLMs. For example, in the case of DEBAYER, which takes 32-bits floating point numbers as input, if *DMA CHUNK* is equals to 256, then we transfer 1MByte of data at every interaction with the DMA controller. This knob is inspired by loop tiling [26], which is a technique that improves the performance of loops by minimizing cache misses. Similarly, our ***XKnob*** allows accelerators to find a balance between computation and communication time (with the memory). Fig. 6 (a) shows an application of this ***XKnob*** to the design of the GRAYSCALE accelerator in a scenario where there is no contention for accessing the shared resources. This scenario corresponds to the case when there is no traffic on the communication infrastructure and the memory controller is always ready to satisfy the requests of the accelerator (ideal case). By increasing *DMA CHUNK*, we obtain points with higher area, but approximately with the same latency. We use an ideal model of memory for these experiments and, since there is no contention, the value of *DMA CHUNK* influences only the number of handshakes necessary to synchronize the different processes (Fig. 2). Since the time for the handshakes is negligible with respect to the computation and communication times, this results in approximately the same total execution time. Note that for some design points we have the same area with different values of *DMA CHUNK*, e.g., with 256 and 512

978-1-5386-3472-1/17/$31.00 ©2017 IEEE



(a) Without contention. (b) With contention.

Fig. 6: Application of *DMA CHUNK* to GRAYSCALE.

bits. When we generate the PLMs with MNEMOSYNE [29], we map the arrays in the code to the banks available in our technology library. The number of memories in the library is limited, and thus arrays with different sizes could be mapped to the same banks. On the other hand, Fig. 6 (b) considers the case with contention. Specifically, we model: (i) the traffic on the communication infrastructure by using the ESP instance showed in Fig. 2 with a Poisson distribution, and (ii) the delay in accessing the external memory. In this case, increasing *DMA CHUNK* reduces the effective latency and produces several additional Pareto-optimal points as shown in Fig. 6 (b). Finally, note that for applying this ***XKnob*** it is necessary to modify the algorithm of the accelerator so that it can operate on a portion of the data. This modification is simple for all the accelerators of WAMI because the corresponding algorithms express multiple degrees of parallelism as they operate on the pixels, or subset of pixels, of the input images.

V. EXPERIMENTAL RESULTS

We evaluate the combined effects of the ***XKnobs*** in broadening the DSE with respect to using the standard HLS knobs by considering the DEBAYER and GRAYSCALE accelerators introduced in the previous sections. We focus our experiments on these two accelerators of the WAMI application because they are representative of two important classes of accelerators: the first class includes the accelerators that have a larger computation time with respect to the communication time, while the second class includes the accelerators that are limited by the communication time. The other accelerators of WAMI (Fig. 2) can be categorized in these classes. Specifically, MATRIX-SUB, MATRIX-ADD, MATRIX-MUL, MATRIX-RES and SD-UPDATE exhibit behaviors similar to GRAYSCALE, while the rest of the accelerators are similar to DEBAYER. We synthesize the accelerators with the commercial HLS tool Cadence C-to-Silicon, targeting an industrial ASIC 32nm technology library.

Fig. 7 illustrates the results of a DSE that considers two of the ***XKnobs*** described in Section IV: *DMA WIDTH* and *PLM PORTS*. The graphs report the design points characterized in terms of normalized effective latency and normalized area. The colors (or shapes) indicate the design points with different values for *DMA WIDTH*, while the ellipses group the points with a specific value for *PLM PORTS* (for GRAYSCALE we include only the points with *DMA WIDTH* equal to 256 bits in the ellipses). Additionally, to perform a more exhaustive DSE, we synthesize multiple design points for each pair of values of *DMA WIDTH* and *PLM PORTS* by using the standard knobs provided by the HLS tool. As we change the values of *DMA WIDTH* we observe two main types of behavior. For GRAYSCALE, we obtain significant performance improvements by augmenting the value of *DMA WIDTH*, while the area increases as a result of using more banks in the PLM. Since, GRAYSCALE is limited by the communication time, this ***XKnob*** helps to reduce the execution time of the load and store processes (Fig. 2). Conversely, for DEBAYER, the designs with larger values of *DMA WIDTH* are always dominated by the designs with the minimum value of *DMA WIDTH*. For this accelerator we do not obtain a speedup with larger DMA channels since the computation time takes most of its execution time. By increasing the value of *PLM PORTS* we can observe two distinct behaviors for these accelerators as well. In the case of DEBAYER, we obtain a significant impact on the performance and cost. Multi-port memories permit to increase the computation parallelism, thus resulting in a significant reduction of the effective latency. For GRAYSCALE, the number of ports guarantees a speedup only when it is no longer limited by the communication time. For example, for GRAYSCALE we obtain performance improvements in using two ports instead of one port for the PLM, when *DMA WIDTH* is 256 bits. On the other hand, when *DMA WIDTH* is 32 bits, increasing the number of ports results in Pareto-dominated points. These results show that *DMA WIDTH* and *PLM PORTS* are two *complementary* knobs: the former allows the accelerators to load or store more data in parallel, thus reducing the communication time; the latter allows the accelerators to perform more parallel operations, thus reducing the computation time. Consequently, to effectively explore different design alternatives it is fundamental to find a compromise by setting these ***XKnobs*** so that computation and communication are well balanced.

Fig. 8 illustrates the DSE results that consider *DMA CHUNK* and *PLM PORTS*. The results in the two graphs at the top of the figure are obtained under the hypothesis that the accelerators execute without contention. The results in the two graphs at the bottom of the figure correspond to the case when contention is modeled as described in Section IV. For these experiments we set *DMA WIDTH* to 256 bits. The colors (or shapes) indicate the design points with different values for *DMA CHUNK*, while the ellipses include the points for the indicated value of *PLM PORTS* (for GRAYSCALE we include only the points with DMA CHUNK equal to 2048 in the ellipses). Additionally, we synthesize multiple design points for each pair of values of *DMA CHUNK* and *PLM PORTS* by using the standard knobs provided by the HLS tool. In absence of contention, both accelerators behave in a similar way. Increasing the *DMA CHUNK* leads to Pareto-dominated design points. Since there is no contention, varying *DMA CHUNK* changes only the number of handshakes between the processes, which is negligible with respect to the execution time of the accelerators. In the presence of contention we obtain the same results for DEBAYER, while for GRAYSCALE we obtain several Pareto-optimal points with different size of *DMA CHUNK*. The reason is that the first accelerator has a longer computation time that makes the differences in latency for accessing the NoC or the memory be negligible. On the other hand, when the computation is balanced with the communication, as in the case of GRAYSCALE with four or eight ports in the PLM, the differences in latency can be very significant.

VI. RELATED WORK

Several approaches to improve the DSE effectiveness and efficiency have been proposed in literature. For example, Schafer [32] presented a method to accelerate the DSE by using a probabilistic approach to (i) classify the knobs and (ii) drastically reduce the design space to be explored. Liu et al. [18] proposed an approach to identify the Pareto-optimal set of RTL implementations by exploiting a learning-based method. Piccolboni et al. proposed COSMOS [27], an automatic methodology that coordinates HLS and memory optimization tools for the exploration of complex accelerators made of multiple components. Other approaches focused on predicting the relevance of the HLS knobs and determine the Pareto-optimal frontiers by using methods that exploit particle-swarm optimization [25], simulated annealing [33],

978-1-5386-3472-1/17/$31.00 ©2017 IEEE

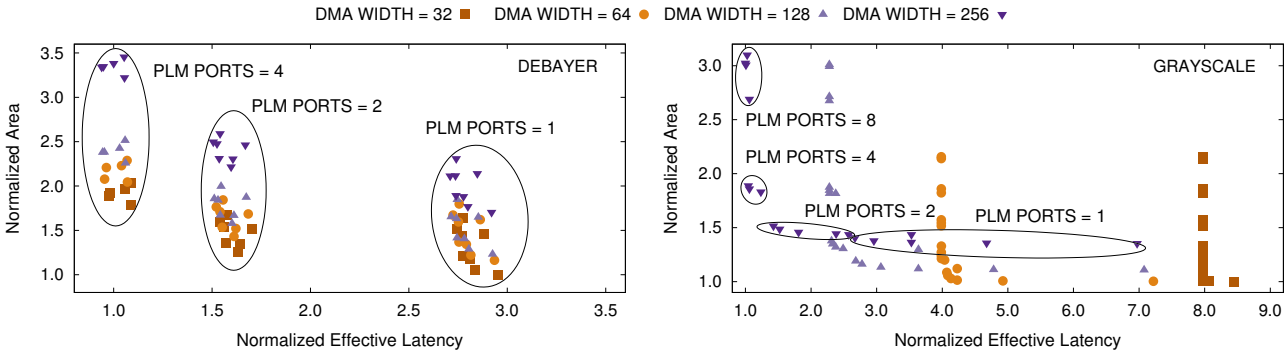


Fig. 7: DSE for the DEBAYER and GRAYSCALE accelerators by varying the *DMA WIDTH* and *PLM PORTS*.

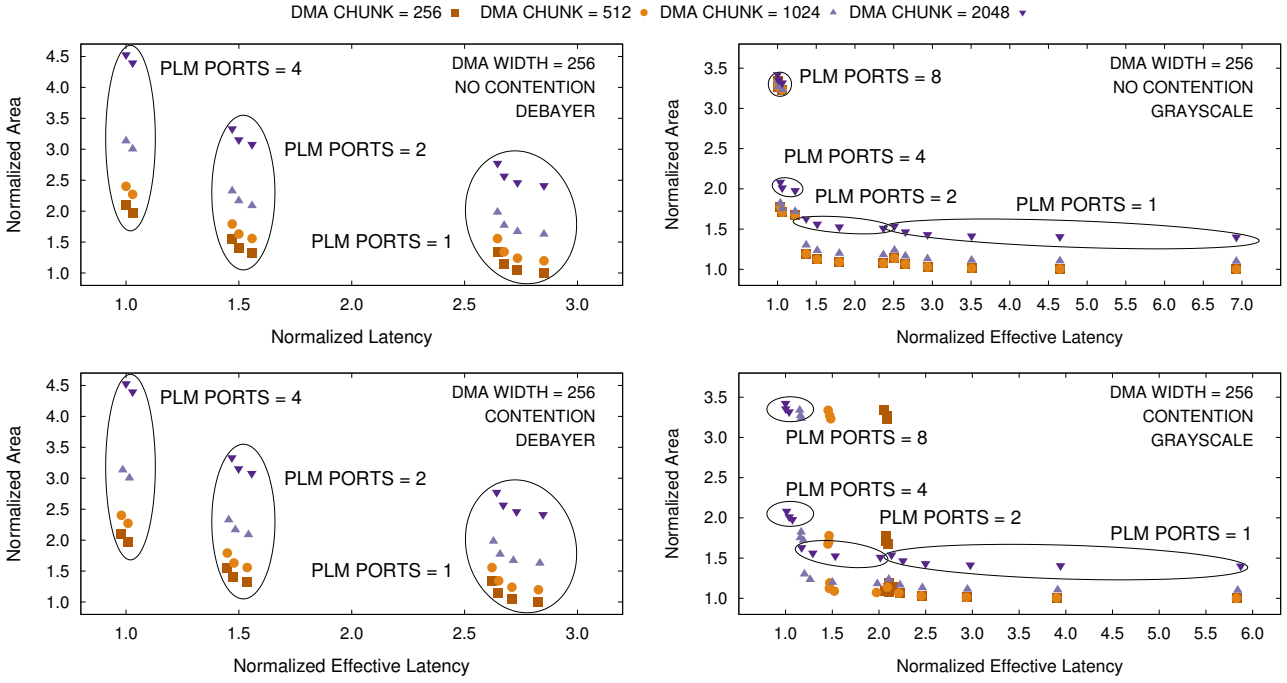


Fig. 8: DSE for the DEBAYER and GRAYSCALE accelerators by varying *DMA CHUNK* and *PLM PORTS*.

genetic algorithms [12], or machine learning [21], [34]. Our work is complementary to these approaches: the ***XKnobs*** can be used by them to further broaden the exploration of the accelerator design space.

Other approaches investigated how to broaden the DSE for specialized architectures. For example, Liu et al. [19] described a set of optimizations and HLS guidelines for the implementation of an H.264 video decoder. Although these optimizations are widely applicable, their work focused on manipulating the C code to obtain high-performance H.264 implementations, rather then offering knobs for HLS. Zhang et al. [36] analyzed Convolutional Neural Networks (CNN) applications. Similarly to us, they focused on the relation between computation and communication, but their work targeted the optimization of CNN algorithms for FPGA. Cong et al. [10] presented buffer restructuring approaches with corresponding analytical models to capture the impact on performance and resource consumption of accelerators. Similarly to us, they highlighted some of the problems of HLS tools in the DSE process. Shao et al. [35] proposed *Aladdin*: a simulator that explores the design space of customized architectures by starting from high-level specifications. *Aladdin*, however, cannot generate RTL implementations, while the ***XKnobs*** when combined with HLS tools allow the synthesis of many Pareto-optimal RTL designs.

VII. CONCLUDING REMARKS

We presented the ***XKnobs***, a set of knobs that aims at extending the standard knobs used in current HLS tools. In particular, we proposed three ***XKnobs***. *PLM PORTS* allows designers to vary the number of ports to access the PLM from the computation processes of the accelerator; this is fundamental for accelerators dominated by the computation time. *DMA WIDTH* indicates the size of the DMA channel in bits; this knob is relevant for accelerators dominated by the communication time. *DMA CHUNK* indicates the amount of data that is transferred at every interaction with the DMA controller; this knob is important to optimize the accelerator data transfers when integrated in complex architectures. We show the effectiveness of the ***XKnobs*** in exploring a broader design space compared to the case in which only the current HLS knobs are used for the WAMI application. The ***XKnobs*** can be integrated in any HLS tools and DSE methods to broaden the DSE of accelerators and enrich their set of Pareto-optimal implementations.

978-1-5386-3472-1/17/$31.00 ©2017 IEEE

ACKNOWLEDGMENTS

This work was supported in part by DARPA PERFECT (C#: HR0011-13-C-0003) and by the Center for Future Architectures Research (C-FAR) (C#: 2013-MA-2384), one of the six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

REFERENCES

[1] N. Baradaran and P. C. Diniz. A Compiler Approach to Managing Storage and Memory Bandwidth in Configurable Architectures. *ACM Transaction on Design Automation of Electronic Systems*, 2008.

[2] K. Barker, T. Benson, D. Campbell, D. Ediger, R. Gioiosa, A. Hoisie, D. Kerbyson, J. Manzano, A. Marquez, L. Song, N. Tallent, and A. Tumeo. *PERFECT (Power Efficiency Revolution For Embedded Computing Technologies) Benchmark Suite Manual*. Pacific Northwest National Laboratory and Georgia Tech Research Institute, 2013. http://hpc.pnl.gov/PERFECT/.

[3] D. Black, J. Donovan, B. Bunton, and A. Keist. *SystemC: From the Ground Up, Second Edition*. Springer, 2009.

[4] S. Borkar and A. Chien. The Future of Microprocessors. *Communication of the ACM*, 2011.

[5] L. P. Carloni. From Latency-Insensitive Design to Communication-Based System-Level Design. *Proc. of the IEEE*, 2015.

[6] L. P. Carloni. The Case for Embedded Scalable Platforms. In *Proc. of the ACM/IEEE Design Automation Conference (DAC)*, 2016. (Invited).

[7] Y. Chen, T. Luo, S. Liu, S. Zhang, L. He, J. Wang, L. Li, T. Chen, Z. Xu, N. Sun, and O. Temam. DaDianNao: A Machine-Learning Supercomputer. In *Proc. of the ACM/IEEE International Symposium on Microarchitecture (MICRO)*, 2014.

[8] J. Cong, M. A. Ghodrat, M. Gill, B. Grigorian, K. Gururaj, and G. Reinman. Accelerator-Rich Architectures: Opportunities and Progresses. In *Proc. of the ACM/IEEE Design Automation Conference (DAC)*, 2014.

[9] J. Cong, P. Li, B. Xiao, and P. Zhang. An Optimal Microarchitecture for Stencil Computation Acceleration Based on Nonuniform Partitioning of Data Reuse Buffers. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2016.

[10] J. Cong, P. Wei, C. H. Yu, and P. Zhou. Bandwidth Optimization Through On-Chip Memory Restructuring for HLS. In *Proc. of the ACM/IEEE Design Automation Conference (DAC)*, 2017.

[11] E. G. Cota, P. Mantovani, G. Di Guglielmo, and L. P. Carloni. An Analysis of Accelerator Coupling in Heterogeneous Architectures. In *Proc. of the ACM/IEEE Design Automation Conference (DAC)*, 2015.

[12] F. Ferrandi, P. L. Lanzi, D. Loiacono, C. Pilato, and D. Sciuto. A Multi-objective Genetic Algorithm for Design Space Exploration in High-Level Synthesis. In *Proc. of IEEE Computer Society Annual Symposium on VLSI*, 2008.

[13] G. Di Guglielmo, C. Pilato, and L. P. Carloni. A Design Methodology for Compositional High-Level Synthesis of Communication-Centric SoCs. In *Proc. of the ACM/IEEE Design Automation Conference (DAC)*, 2014.

[14] T. J. Ham, L. Wu, N. Sundaram, N. Satish, and M. Martonosi. Graphicionado: A High-Performance and Energy-Efficient Accelerator for Graph Analytics. In *Proc. of the ACM/IEEE International Symposium on Microarchitecture (MICRO)*, 2016.

[15] M. Horowitz. Computing's energy problem (and what we can do about it). In *Proc. of the IEEE International Solid-State Circuits Conference (ISSCC)*, 2014.

[16] IEEE. SystemC Standardization Working Group. 1666-2011 - IEEE Standard for Standard SystemC Reference Manual.

[17] H. G. Lee, N. Chang, U. Y. Ogras, and R. Marculescu. On-chip Communication Architecture Exploration: A Quantitative Evaluation of Point-to-point, Bus, and Network-on-chip Approaches. *ACM Transactions on Design Automation of Electronic Systems*, 2008.

[18] H.-Y. Liu and L. P. Carloni. On Learning-Based Methods for Design-Space Exploration with High-Level Synthesis. In *Proc. of the ACM/IEEE Design Automation Conference (DAC)*, 2013.

[19] X. Liu, Y. Chen, T. Nguyen, S. Gurumani, K. Rupnow, and D. Chen. High Level Synthesis of Complex Applications: An H.264 Video Decoder. In *Proc. of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*, 2016.

[20] M. J. Lyons, M. Hempstead, G. Y. Wei, and D. Brooks. The Accelerator Store: A Shared Memory Framework for Accelerator-based Systems. *ACM Transactions on Architecture and Code Optimization*, 2012.

[21] A. Mahapatra and B. C. Schafer. Machine-Learning Based Simulated Annealer Method for High Level Synthesis Design Space Exploration. In *Proc. of the Electronic System Level Synthesis Conference (ESLsyn)*, 2014.

[22] P. Mantovani, G. Di Guglielmo, and L. P. Carloni. High-Level Synthesis of Accelerators in Embedded Scalable Platforms. In *Proc. of the ACM/IEEE Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2016.

[23] G. Martin and G. Smith. High-Level Synthesis: Past, Present, and Future. *IEEE Design Test of Computers*, 2009.

[24] W. Meeus, K. Van Beeck, T. Goedemé, J. Meel, and D. Stroobandt. An Overview of Today's High-level Synthesis Tools. *ACM Design Automation for Embedded Systems*, 2012.

[25] V. K. Mishra and A. Sengupta. PSDSE: Particle Swarm Driven Design Space Exploration of Architecture and Unrolling Factors for Nested Loops in High Level Synthesis. In *Proc. of the IEEE International Symposium on Electronic System Design (ISED)*, 2014.

[26] P. R. Panda, H. Nakamura, N. D. Dutt, and A. Nicolau. Augmenting Loop Tiling with Data Alignment for Improved Cache Performance. *IEEE Transactions on Computers*, 1999.

[27] L. Piccolboni, P. Mantovani, G. Di Guglielmo, and L. P. Carloni. COSMOS: Coordination of High-Level Synthesis and Memory Optimization for Hardware Accelerators. *ACM Transactions on Embedded Computing Systems (TECS), Special Issue presented in CODES+ISSS*, 2017.

[28] C. Pilato, P. Mantovani, G. Di Guglielmo, and L. P. Carloni. System-level Memory Optimization for High-level Synthesis of Component-based SoCs. In *Proc. of the ACM/IEEE International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, 2014.

[29] C. Pilato, P. Mantovani, G. Di Guglielmo, and L. P. Carloni. System-Level Optimization of Accelerator Local Memory for Heterogeneous Systems-on-Chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2017.

[30] R. Porter, A. M. Fraser, and D. Hush. Wide-Area Motion Imagery. *IEEE Signal Processing Magazine*, 2010.

[31] A. Sangiovanni-Vincentelli. Quo Vadis, SLD? Reasoning About the Trends and Challenges of System Level Design. *Proc. of the IEEE*, 2007.

[32] B. Carrion Schafer. Probabilistic Multiknob High-Level Synthesis Design Space Exploration Acceleration. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2016.

[33] B. Carrion Schafer, T. Takenaka, and K. Wakabayashi. Adaptive Simulated Annealer for High Level Synthesis Design Space Exploration. In *Proc. of the IEEE International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, 2009.

[34] B. Carrion Schafer and K. Wakabayashi. Machine Learning Predictive Modelling High-Level Synthesis Design Space Exploration. *IET Computers Digital Techniques*, 2012.

[35] Y. S. Shao, B. Reagen, G. Y. Wei, and D. Brooks. Aladdin: A Pre-RTL, Power-performance Accelerator Simulator Enabling Large Design Space Exploration of Customized Architectures. In *Proc. of the ACM/IEEE International Symposium on Computer Architecture (ISCA)*, 2014.

[36] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong. Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks. In *Proc. of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*, 2015.

978-1-5386-3472-1/17/$31.00 ©2017 IEEE