IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 6 December 2023, accepted 16 January 2024, date of publication 22 January 2024, date of current version 1 February 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3357352

RESEARCH ARTICLE

Design Space Exploration of HW Accelerators and Network Infrastructure for FPGA-Based MPSoC

BOUTHAINA DAMMAK[1], MOUNA BAKLOUTI[2], AND DEEMA ALSEKAIT[1]
[1]Department of Computer Science, Applied College, Princess Nourah bint Abdulrahman University, Riyadh 11671, Saudi Arabia
[2]Computer Embedded Systems Laboratory (CES-Lab), University of Sfax, Sfax 3038, Tunisia

Corresponding author: Deema Alsekait (DMAlSekait@pnu.edu.sa)

This work was supported by Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2024R435), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

ABSTRACT Supercomputing systems are increasingly reliant on heterogeneous Multiprocessors System on-Chip (MPSoCs), merging multiple processors and hardware accelerators (HWAcc) on the same die to achieve power and performance needs. Due to CPU complication and timing closure challenges of tightly coupled design approach, the state-of-the art of HWAcc design methodology relies on coupling the processors with loosely coupled HWAccs. Loosely-coupled HWAccs can be shared or private accelerators running custom instructions to form a heterogeneous multi-processor system. Some works discussed the determination of the sharing degree of the HWAcc over the processors, however the impact of the integrated communication infrastructure is not discussed. Thus, we propose a high-level design exploration tool to select the accelerators and generate the adequate communication interconnect under performance and area constraints. Different homogeneous and heterogeneous multi-processor configurations are evaluated and compared running different signal processing benchmarks. Experimental results show the efficiency of the proposed exploration tool to rapidly explore and select the adequate architecture.

INDEX TERMS Multi-processor system-on-chip, custom instructions, shared hardware accelerator, network communication, high-level exploration.

I. INTRODUCTION

New embedded applications demand high-performance multiprocessor design to meet real time deadlines while respecting other critical constraints such as low power consumption and low area [18], [22]. Energy efficient computing constraint has led to the rapid adoption of hardware accelerators (HWAccs) [1], [13], [18], [28]. In fact, large scale multiprocessors SoCs [2] can integrate several heterogeneous processing elements, or cores, in the same die. Since hardwired specialized accelerators are inflexible and cannot be suited to changing requirements, some processor cores are coupled with HWAccs running custom instructions that can be easily modified by the user [30].

In this paper, not only the number of HWAccs and the sharing degree is flexible but also the communication infrastructure is chosen in an area and performance aware manner. The proposed multi-processor architecture is implemented in a Field Programmable Gate Array (FPGA) fabric, to take advantages from its flexibility and reconfigurability.

Our proposed scalable architecture integrates a parametric number of soft-cores and HWAccs interconnected together via a configurable interconnection network. The soft-cores execute the software tasks and the HWAccs execute application specific instructions. A HWAcc could be shared between a group of processor cores. In fact, the purpose of this resource sharing between cores [3] is to reduce the number of hardware resources on the FPGA while preserving better performance by benefiting from similar tasks, commonly used in signal processing applications [11] or multimedia applications.

The communication between cores and shared accelerators is ensured by a configurable interconnection network. In Multiprocessors System on-Chip (MPSoCs), hierarchical bus-based network infrastructure are popular because of simplicity and moderate power and area consumption. The

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

simplicity of the bus-based architecture design, moderate area usage and predictable access overhead are the selling keys. However, state-of-the-art demonstrates that beyond a number of processor cores, the performance of the hierarchical interconnect degrades notably.

For our shared HW accelerators architecture, bus-based communication infrastructure is adopted for a convenient number of cores that share a set of HW accelerators. Beyond a sharing limit, the bus-based architecture may cause a communication bottleneck. In this case, our methodology rely on a more scalable architecture based on crossbar network.

As mentioned before, the HWAcc sharing degree and the choice of communication infrastructure have to be appropriately parametric to find the best trade-offs between area consumption and execution time, deemed of designers interest. In this context, Design Space Exploration (DSE) is a tool by which the optimal parametric configuration for a given system can be found.

In this paper which is a continuation and refinement of our earlier works [6], we present the following contributions:

- **Flexible Multi-Processor Architecture with Enhanced Communication Infrastructure:** We introduce a novel multi-processor system-on-chip (MPSoC) architecture that seamlessly integrates an adaptable communication infrastructure specifically tailored for the required shared hardware accelerator (HWAcc) configuration. Unlike previous works, our architecture introduces innovative communication paradigms, emphasizing the efficient sharing of HWAcc with a well-suited network interconnect.
- **Enhanced Design Space Exploration (DSE) for HWAcc Sharing and Interconnection Structure:** Our DSE incorporates new metrics, including the consumed logic area and the delay of the interconnect, providing a more comprehensive evaluation of area utilization and performance delay. This refined approach to DSE enables accurate predictions and optimizations, setting our work apart from existing literature.
- **Case Study Implementation on Cyclone V 5CEA7 FPGA Board:** To validate our proposed architectures, a case study is conducted implementing multi-processor architectures based on the NIOS II embedded processor on the Cyclone V 5CEA7 FPGA board. This practical implementation demonstrates the real applicability and effectiveness of our innovative MPSoC designs.

The paper is structured as follows. Section II reviews state-of-the-art multi-processor architectures, mainly used to accelerate signal processing applications. Section III describes the heterogeneous multi-processor SoC and gives a brief outlook on its developed high-level design exploration framework. Experimental results are discussed in Section IV. Conclusions are drawn in Section V with a brief outlook on future works.

II. RELATED WORKS

Newer MPSoCs are mainly heterogeneous, with the integration of application specific instructions also called MPSoCs customization [14], [21], [23]. MPSoCs customization consists on coupling the processors to Specific Functional Units (SFU). These SFU are either closely tied to the processor data path, as custom functional units, or loosely related to the processor as HWAcc. Many researchers adopted the second design methodology, where each processor is extended with Reconfigurable Fabric (RF) [10], [22]. Many works provide tools that make it simple for system designers to compare several software and hardware options for executing the same algorithm [9], [17]. In [28], the authors take into consideration several HWAccs in an FPGA and their testbeds as input of the exploration algorithm. The result is a collection of dominant systems that trade off area and performance. In [25], the authors propose to help a system designer to quickly compare several hardware/software configurations using the Kwok List scheduling heuristic [12]. The heuristic kwok algorithm uses intelligent search techniques to quickly find a solution. However, such solution is approximate while, in comparison, Mixed Integer Linear Programming (MILP)-based exploration offers an exact one. Moreover, in our proposed work, two additional metrics are considered, namely HWAccs sharing between cores and the network interconnect

To enhance performance and to tackle the problem of hardware resource usage of heterogeneous architectures, other works proposed to share RF among the processors to implement HWAccs. The work in [4] proposed to share coarse grained custom instruction between processors. This work defined a metric to evaluate the impact of FU (Functional Units) sharing between the processors. The metric checks the fraction of execution time in which different processors would be competing to access the shared resource. Thereby, this metric can be used to estimate the possible speedup with the extension of new HWAccs in different configurations. Despite the use of Shared Accelerator Concurrency Level (SACL) metric, the authors limited the number of shared accelerators to symmetric configurations. For example, for 8-processors, only the configurations with 2, 4 or 8 shared hardware accelerators are considered. In [19], the authors propose a framework to explore the design of multiprocessor heterogeneous architecture based on fuzzy MIP and Graph theory based Traffic Estimator (GTE). The authors propose to share HWAcc, through the integration of bridges and a fuzzy-controller capable of solving sharing problems. Their ILP model is based on previous work [7] and no improvements have been proposed. In [16], the authors propose a 3-steps Hardware/software partitioning design flow for single-processor architecture. The first step consists on Graph conversion that translates an application program into an SDF graph. The second step consists on sub-graph clustering which is based on the hill climbing heuristic search [24] that identifies candidate nodes for hardware acceleration. The third step consists on graph scheduling that

uses HEFT algorithm [26] for implementing multiple custom accelerators in a single-core. Their proposed approach has been evaluation for the NIOS II processor executing neural networks applications.

The flexible MPSoCs design exploration methodology presented in this paper is different from the cited ones at different levels. First, the works in [16] is proposed for single-core architecture and works in [4] and [25] are not considering sharing of the same computational task between the different processors. Second, most cited works focus on a given fashion (SPMD or MPMD) whereas our methodology considers both SPMD and MPMD configurations, to cover the wide variety and needs of different signal processing applications. Adding to that, the MPSoC architectures explored are parametric and flexible in terms of the number of processors, the used interconnection network and the number and sharing of the integrated HWAcc. Based on our high-level exploration tool, the most adequate configuration could be rapidly generated under area and performance constraints.

III. FLEXIBLE MULTI-PROCESSOR SOC

In this work, both SPMD and MPMD paradigms are explored in a multi-processor system to work with private and shared HWAccs. As depicted in Figure 1, it is composed of multiple processor processors, each one is connected to its memory. The processor can also be connected to local peripherals such as timer, memory controller for connection to external memory, etc. The first processor can act as the controller and synchronizer of the whole system. The other processors execute the software tasks and a group of HWAccs is coupled to the ones running application specific instructions. The HWAcc can be private to a processor,

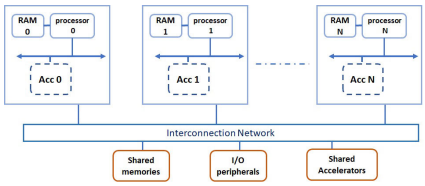

FIGURE 1. Heterogeneous multi-processor SoC architecture.

this means that is coupled to only one, or shared over two processors or more. The number of shared HWAcc and the sharing type of HWAccs are different from one processor to another. The scope of sharing HWAccs between processors is to reduce circuit complexity regarding logic elements usage while reserving the performance and minimizing the energy consumption [3]. In this work, the sharing methodology is explored for architecture with SPMD and MPMD fashions. In SPMD fashion, all processors execute the same applications so the set of computational tasks to be explored as HWAcc is the same for all processors. In MPMD fashion, HW accelerators that can be attached to the different processors differ from one processor to the other depending on the applications tasks.

A. MPSoC ARCHITECTURE FOR SPMD PARADIGM

In SPMD paradigm, multiple processors simultaneously process the same program on different data. For such embedded systems, the straightforward implementation of computational tasks as HWAccs is an excessive area-consuming solution. For this type of architecture, the sharing methodology will preserve logic resources by implementing a moderate number of HWAcc. Indeed, different processors that execute the same task can access the same HWAcc of a computational task as long as this sharing does not give rise to a considerable delay. In SPMD fashion, a number of homogeneous processors share a main memory, used for data communication. Each processor can be also connected to peripherals over its local bus. Further, it can have a set of private and shared HW accelerators. To ensure the symmetric aspect for the architecture, the number of processors sharing a HW accelerator is a power of two. An interconnection network, which will be covered later, connects a shared HW accelerator to processing cores. Figure 2 is an example of 4-processor architecture designed to work in SPMD fashion. In this example, four HWAccs of T1 task are implemented in private, whereas only two HWAccs of T2 task are shared between a couple of processors.

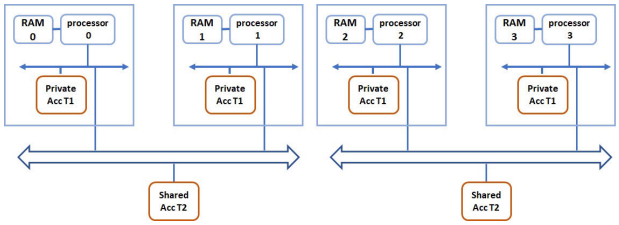

FIGURE 2. Example of 4-processors MPSoC architecture for SPMD paradigm.

B. MPSoC ARCHITECTURE FOR MPMD PARADIGM

In large and complex MPSoC systems, the computation becomes irregular and the MPMD paradigm is preferred. In a MPMD system, the processors are executing different programs and a full implementation of computational tasks is non-trivial. However, we argue that the different programs contain similar tasks, in particular for signal processing applications, and a HWAcc designed and implemented at low abstraction level can be shared across the different programs. A given HWAcc implementation, without exploring the possibility to share it, may bloat hardware resources with insufficient performance amelioration. Our proposed architecture for MPMD systems is an MPSoC architecture where each processor may have private HWAcc and shared HWAcc. Each processor has a local memory and a local bus. The peripherals and private HWAcc are connected to local bus. The HWAcc of a common task that is a part of two or more programs, can be shared between the processors. A network infrastructure is implemented to ensure communication of different processors to a shared HWAcc. The number of private and shared HWAccs for

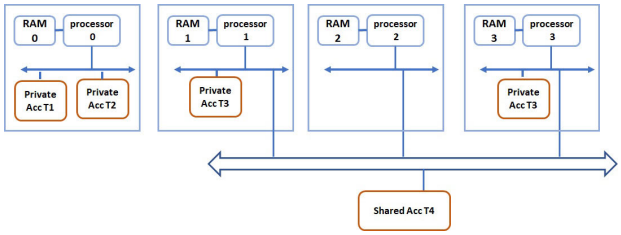

FIGURE 3. Example of 4-processors MPSoC architecture for MPMD paradigm.

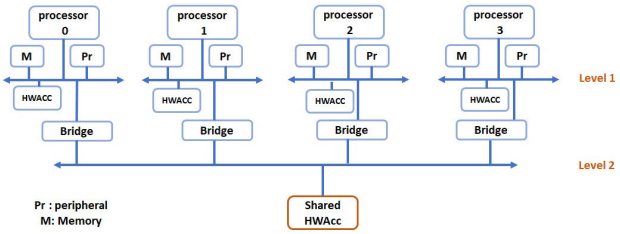

FIGURE 4. MPSoC architecture with a two-level hierarchical bus interconnect.

each processor depends on the performance requirement for each application. Figure 3 depicts a 4-processor architecture designed to work in MPMD fashion. Processors 0, 1 and 3 have private HWAccs. Processors 1, 2 and 3 run a similar task T4, thus, they share its HWAcc. The private HWAcc for each processor is placed on its local bus; whereas, the shared one is connected via the network.

C. INTERCONNECTION NETWORK

A critical design part of our proposed architecture is the communication infrastructure. We adopt hierarchical bus for architecture with moderate number of processors and shared HW accelerators. The hierarchical bus is simple to build and cost-effective. However, the cost of this interconnect will be high, in terms of interconnect wires and bottleneck, depending on the number of processors sharing a HW accelerator. In consequence, to face this limitation, our design methodology shifts to the use of a crossbar network for the architecture with high sharing degrees.

1) HIERARCHICAL BUS BASED-ARCHITECTURE

A hierarchical bus interconnect consists of different levels of buses connecting various components. For our MPSoC architecture, the simplest way is to rely on two-levels. The higher level is composed of processors local buses that allow processors to communicate to their peripherals and private HWAcc. The second level is composed of shared buses that communicate the shared HWAcc to the processors sharing this latter. Between these two levels, bridges are placed to ensure transactions. The level one buses are considered as primary buses whereas level two buses are secondary ones. Figure 4 is a an example of 4-processors MPSoC architecture with a shared HWAcc connected through a 2-levels hierarchical bus.

The bridge is a slave on the primary bus and is a master on the secondary bus. It is responsible to pass signals between network levels. This bridge is composed of the following blocks:

- Slave interface: it is a bi-directional slave interface to primary bus. It decodes address from the bridge registers and for the HWAcc on the secondary bus.
- Slave buffer interface: it is responsible of the conditional read/write operations from secondary bus to primary one. It decodes the access-request from the HWAcc and raises it to the primary bus.
- Write FIFO (First In First Out): it is a FIFO memory that stores the data from the Slave buffer interface during a write transaction.
- Read FIFO: it is a FIFO memory that stores the data from the Master buffer during a read transaction.
- Master interface block: it is a bi-directional master interface to secondary bus. It decodes address from HWAcc.

When a processor sends a request to the shared HWAcc, the slave interface receives the request and the primary interface decodes it. Then, the secondary interface forwards the signals to perform the request on the HWAcc.

2) NoC BASED-ARCHITECTURE

In this case, the different processors are interconnected via a crossbar: a fully connected network. The interconnection network architecture is sketched in Figure 5. It is composed of routing elements (RE), where the number is equal to the number of processors in the system. These routers allow each processor to communicate with the desired one. Each RE integrates two main blocks: n input routers and n output routers, where n is the number of processors in the system. The input router plays the role of a switch which will route the request and the data to communicate to the appropriate output router according to the address decoding. As soon as the output router detects a data request on its input port, it stores it in a FIFO (of size 2) and sends it afterwards to the requested receiver. This router contains a routing and arbitration unit (based on priority mechanism) that performs the routing function and handles conflict situations. Although the increased logic area, compared to the hierarchical bus (as shown in Tables 3 and 4), the crossbar offers a better bandwidth and thus a better latency since the different sender-receiver interconnection paths are separated from each other.

IV. HIGH-LEVEL EXPLORATION FRAMEWORK

The proposed framework, is based on a high-level modeling approach to generate the adequate MPSoC architecture satisfying area and performance constraints, specifically in term of HWAcc design. In fact, the hardware accelerators configuration of selected tasks and the network infrastructure

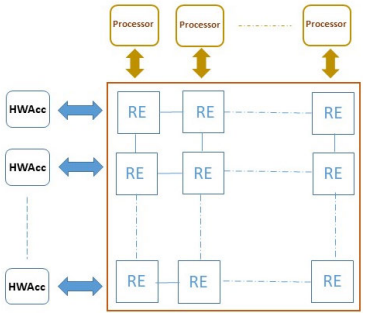

FIGURE 5. Network connections.

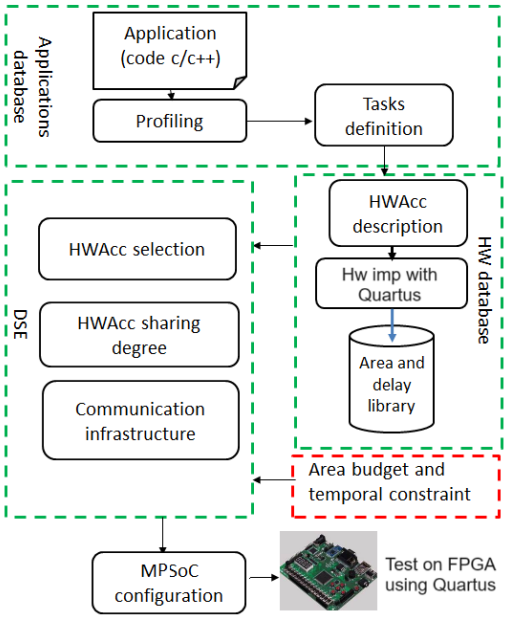

FIGURE 6. Exploration Tool.

are explored to select the configurations that are good enough to satisfy design requirements. It is worth to mention that the proposed DSE tool automatically generates the MPSoC configuration that is optimal for a given application. In comparison, existing synthesis tools, like Quartus, does not automatically generate the suitable MPSoC architecture for a given application. It just helps the user to simulate and test a given architecture he has already implemented himself. In fact, the Quartus tool is a synthesis and simulation tool that allows the implementation of various architectures to test and evaluate their performances. Conversely, our proposed tool utilizes an Integer Linear Programming (ILP) algorithm based on an objective function and performance constraint (as detailed in sections IV-B and IV-C). It automatically explores the space of MPSoC configurations using combinations of private and/or shared hardware accelerators and a configurable interconnect, which can be either a hierarchical bus or a crossbar. Therefore, prior to implementing the architecture using tools like Quartus, our tool assists the designer in automatically generating the optimal MPSoC configuration for a specified number of cores. That is, our framework helps the user to easily and rapidly implement the needed MPSoC architecture that will fit his application needs.

Figure 6 illustrates the proposed framework, which comprises three main steps: the applications database, HW database, and the Design Space Exploration (DSE) process. It begins by profiling C/C++ codes to identify the most computational tasks, contributing to the applications database. Subsequently, the computational tasks from step 1 undergo synthesis as Hardware Accelerators (HWAcc), forming the HW database. In this step, we employ the Quartus tool to implement the HWAccs and gather information regarding their area usage and speed-up. These details serve as input for the second step (DSE), which generates the HWAcc sharing configuration and the interconnect network. The final step involves the execution of Design Space Exploration using the ILP algorithm to meet a well-defined objective function and a performance constraint (see sections IV-B and IV-C). The ILP yields an optimal configuration for HWAcc and a suitable network infrastructure that preserves optimal area utilization while meeting the required performance criteria. The resulting architecture is then implemented in Quartus for testing.

In this section, we call a common task, the task that is executed on two processor cores or more. $C = \{C_1 \dots C_i \dots C_n\}$ is the set of n homogeneous cores. Let $\{T_1 \dots T_j \dots T_m\}$ is the sequence of tasks to be executed as HWAcc, and $N = \{1, 2, \dots, n\}$ and $M = \{1, 2, \dots, m\}$. Each common task T_j, $(j \in M)$ has a list of parameters:

- a_j : is the number of HW resources consumed by the HWAcc of T_j task. For each task, the value of a_j is measured with an implementation of T_j as private HW accelerator.
- AN: represents the Area, in terms of HW resources, consumed by the crossbar Network.
- AB: is the number of HW resources consumed by a bridge.
- E_{ji}: is a binary parameter which is equal to 1 if the task T_j is executed by the processor C_i, otherwise it is equal to 0.
- ts_{ji} and te_{ji}: are two parameters respectively for the start-time and the end-time of T_j on processor C_i. These parameters are expressed in seconds and are determined through profiling step.
- G_j: the execution time gain of T_j when implemented as HWAcc. It is expressed in seconds.

The space exploration solution is based on a Mathematical MILP (Mixed Integer Linear Programming) formulation. The formulation section is divided into three sub-sections dedicated to detail variables, optimisation functions and the defined constraints.

A. VARIABLES

The decision variables of our MILP are of two types: binary assignment variables and continuous flow variables:

- Acc_j is a binary variable that denotes whether task Tj is implemented on Hardware or Software.

$$\forall j \in M$$
$$Acc_j = \begin{cases} 1, \text{if } T_j \text{ is implemented on HWAcc} \\ 0, \text{else} \end{cases}$$

- $CShT_{jik}$ is a binary decision variable to decide whether the accelerator of task Tj is shared between processors Pi and Pk or not.

$$\forall j \in M, \forall (i,k) \in N^2$$
$$CShT_{jik} = \begin{cases} 1, \text{if Acc of } T_j \text{ is shared between } C_i \text{ and} C_k \\ 0, \text{otherwise} \end{cases}$$

- CI: a binary decision variable that denotes weather the interconnect is a Crossbar interconnect or hierarchical bus interconnect.

$$CI = \begin{cases} 1, \text{if crossbar interconnect is implemented} \\ 0, \text{if hierarchical-bus interconnect is implemented} \end{cases}$$

B. OBJECTIVE FUNCTION

The considered optimization problem aims to minimise the area objective function, which depends on software/hardware implementation, the sharing of HWAccs between the different cores and also the type of network to be adopted. Think about the n-cores architecture that executes applications containing a set of the same patterns, which can be implemented either on SW (Software) or HW. Obviously, HW resources are not used when a pattern is implemented on SW. However, if a pattern is implemented on HW, the consumed area depends on whether a private or shared hardware accelerator is used. Following the HWAccs configuration, the suitable connection network is decided.

Given the decision variables defined in subsection IV-A, the resulting objective function is:

$$MinTotal\_Area = \sum_{j=1}^{m}\sum_{i=1}^{n} \frac{E_{ji} Acc_j a_j}{\sum_{k=1}^{n} CShT_{jik}} + \sum_{j=1}^{m}\sum_{i=1}^{n} CI * AB * sh_{ji} + (1 - CI) A_N \quad (1)$$

subject to

$$CShT_{jii} = 1 \quad \text{for } j = 1..m \quad \text{and} \quad i = 1..n \quad (2)$$

Sh_{ji}: a continuous variable that denotes the sharing degree of the processor C_i for the task T_j and is calculated as follow:

$$sh_{ji} = \sum_{k=i+1}^{n} CShT_{jik} \quad \text{for } j = 1..m \quad \text{and} \quad i = 1..n \quad (3)$$

$$CShT_{jik} \leq E_{ji} \quad \text{for } j = 1..m \quad \text{and} \quad i = 1..n \quad \text{and} \quad k = \{1..i-1\} \cap \{i+1..n\} \quad (4)$$

$$CShT_{jik} - CShT_{jki} = 0 \quad \text{for } j = 1..m \quad \text{and} \quad (i,k) = 1..n \quad (5)$$

$$r_{jikh} = CShT_{jik} * CShT_{jih} \quad \text{for } j = 1..m \quad \text{and} \quad (i,k,h) = 1..n \quad (6)$$

$$CShT_{jih} \geq r_{jikh} \quad \text{for } j = 1..m \quad \text{and} \quad (i,k,h) = 1..n \quad (7)$$

$$CShT_{jkh} \leq 1 + 2 * r_{jikh} - CShT_{jik} - CShT_{jih} \quad \text{for } j = 1..m \quad \text{and} \quad (i,k,h) = 1..n \quad (8)$$

Equation 2 guarantees that denominator in Equation 1 is non-zero. Equation 3 calculates the area usage of the bridge for a hierarchical interconnect. Equation 4 guarantees that the task T_j will be implemented only for the processor executing this task. Equation 5 guarantees the symmetry of $CShT_{jik}$ matrix. In fact, if the HWAcc of T_j is shared between C_i and C_k, both $CShT_{jik}$ and $CShT_{jki}$ must be equal. Equations 6, 7 and 8 guarantee that a core C_i share a HW Accelerator of T_j with C_k and C_h then C_i, C_k and C_h share the same HWAcc of T_j.

C. PERFORMANCE CONSTRAINT

In the proposed DSE, the area consumption must be minimized following a performance constraint that needs to be satisfied. *acci* is a variable that calculates the execution-time gain or acceleration of the core C_i for the generated solution. This acceleration must be equal or upper the required acceleration named $limit_i$. For each core, the acceleration *acci*

In Equation 9 is computed based on the acceleration of each task T_j executed on this core when implemented on HWAcc ($tacc_j$) and the delay DL_{ji} to access the shared HWAcc of T_j.

$$acc_i = \sum_{j=1}^{m} E_{ji} Acc_j (tacc_j - DL_{ji}) \geq limit_i \quad (9)$$

The continuous variable DL_{ji} denotes the delay of t he core i to access the T_j HWAcc. This variable is a crucial parameter to consider in the performance constraint and is calculated through the overlapping delay Ov_{ji}, crossbar interconnect delay DN and Hierarchical-bus delay DH. The overlapping delay Ov_{ji} is the delay of executing task Tj on the core i competing an other core to access Tj HWAcc.

$$DL_{ji} = Ov_{ji} + CI * DN + (1 - CI) * DH, \quad \text{for } j = 1..m \quad \text{and} \quad (i,k) = 1..n \quad (10)$$

V. SIGNAL PROCESSING BENCHMARKS

The experimental results, reported in this section, are divided into three parts. In the first part, we present the adopted methodology as well as some preliminary results.

In the second part, a discussion on the impact of HWAcc sharing on MPSoC performance is presented. The third Section presents an investigation of the MILP model. The results are built on different sets of instances, carefully chosen and covering various systems.

A. EXPERIMENTAL SET-UP

We consider different MPSoC architectures, each architecture has a number $n \in \{2, 4, 8, 16, 32, 64\}$ of cores. The various configurations are evaluated running different SW benchmarks. Table 1 highlights the hardware environment settings used in this work: the targeted FPGA, the used tools and the integrated core. It is to be noted that no synthesis nor placement and routing optimizations were applied. Quartus tool, based on default settings, is used to synthesize and simulate the generated architecture. Thereby, we maintain transparency and ensure that the obtained results accurately reflect the inherent capabilities of the chosen architecture without the influence of additional optimizations.

TABLE 1. Configuration settings.

Tool–version	Intel Quartus Prime standard edition v17.0 equipped with: - NIOS II EDS (Embedded Design Suite) - Intel PowerPlay Early Power Estimator tool
Settings Parallel compilation Compiler	 Use all available processors Preserve fewer node names to save disk space Balanced (Normal flow)
Soft-core Frequency	NIOS II 100 MHz
FPGA Resources	Cyclone V 5CEA7 149500 logic elements & 7.696 Mb embedded memory
Operating System	Windows 10

B. PRELIMINARY RESULTS

Table 2 shows synthesis results for different homogeneous multi-processor configurations (without accelerators). For example, the architecture with 16 cores consumes 18% of the FPGA logic resources. Tables 3 and 4 show simultaneously the implementation results of hierarchical bus and crossbar architectures depending on the size of the network. In fact, synthesis results in terms of the consumed logic resources on the FPGA as well as the flow rate and delay metrics are presented. Delay refers to the time, expressed in number of cycles, required to route a message from a sender core to a recipient core. Flow rate reflects the network's traffic flow capacity in terms of quantity of data per unit of time, expressed in Mega Bytes per second. The delay and flow rate metrics are mainly obtained after different simulations via the Quartus synthesis tool.

These different results are detailed for different number of cores and will be used in the MILP model to generate for each system the convenient network infrastructure.

TABLE 2. Synthesis results.

Number of processors	Logic utilization %	Power mW
8	10	1069
16	18	1475
32	35	2020
64	66	2633

TABLE 3. Hierarchical-bus network implementation results on the cyclone V board.

Number of processors	Logic resources ALM	Logic resources registers	Flow rate Mb/s	Delay cycles
4	92	442	347.16	5
8	204	739	303.42	11
16	363	1319	229.64	23
32	748	2467	185.91	48
64	1491	4781	120.25	96
128	2220	6888	63.5	190

TABLE 4. Crossbar network implementation results on the cyclone V board.

Number of processors	Logic resources ALM	Logic resources registers	Flow rate Mb/s	Delay cycles
4	141	483	527.16	2
8	274	806	463.8	4
16	479	1469	420.57	8
32	953	2818	360.64	17
64	2096	5917	340.22	33
128	5092	13609	63.5	59

C. MPSOC EVALUATION AND MOTIVATION

To investigate the impact of HWAcc integration and sharing for MPSoC, we tested two applications: ICT (Inverse Cosine Transform) application and JPEG codec application. In the first experiment, we executed the integer forward transform ICT that constitutes a part of the H.264/AVC video encoder. It uses integer arithmetic and is less complex than the DCT (Discrete Cosine Transform). The application is implemented in a MPMD fashion. We evaluated the results for different number of processor cores (2, 4, 8 and 16). The implemented integer forward transform is based on the butterfly method [5] generally applied on macroblocks of size 4×4 pixels [20] and often on macroblocks of size 8×8 pixels [29] in the high profiles (like High Definition profiles). Figure 7 presents ICT execution results on homogeneous multi-core configurations varying the number of integrated cores as well as the picture format [CIF (Common Intermediate Format), QCIF (Quarte-CIF) and HD]. The vertical axis is displayed with logarithmic scale.

In figure 7, we show the execution time in milliseconds for the ICT application on different architectures. The

results demonstrate that significant speedup over single-core implementation can be achieved. For example, for 2-cores and 4-cores implementation, the execution times of a QCIF picture are respectively 1200 and 900 milliseconds. These results show that we achieved a speed up to 1.27 for 2 cores and 1.22 for 4-cores architectures. Such results are in line with similar implementations [8], [27] that achieve comparable speedup for the same number of cores.

The implementation of a HWAacc for ICT task yields a slower speedup. This is due to the fact that the butterfly method is simple and based on arithmetic computations consisted mainly of addition, subtraction and shift operations [15].

The second experimentation presents results executing a JPEG codec application on different multi-processor configurations. The application is implemented in a SPMD fashion. The JPEG profiling results show that the HDCT (Horizontal discrete cosine transform), VDCT (Vertical discrete cosine transform) tasks are the most computational tasks. Different MPSoC architectures are synthesized on Cyclone V board. In figure 8, we present the synthesis results for different number of cores, $n = 2, 4, 8$ *and* 16, accounting for all cases from a fully shared configuration to a fully private one (one HWAcc per core).

For n-cores architecture executing Jpeg application, we have different configurations based on the number of implemented HWAccs. For each notation (i, j), i and j are respectively the number of HDCT and VDCT HWAccs. The (n, n) configurations mean that n HDCT and n VDCT HWAccs are implemented, so each core is connected to a private HWAcc and this configuration is called fully private configuration. Whereas (1,1) notation means that all the cores are sharing only one HDCT and one VDCT HWAcc and it is called a fully shared configuration. The (i, j) configuration, where $1 < i, j < n$, has i HDCT HWAcc and j VDCT HWAcc shared between the different cores. In figure 8, for $n = 4$, we presented the evaluation of three different configurations namely (1,1), (2,2) and (4,4).

For $n = 8$, different configurations between the two extreme configurations have been evaluated. Considering 8-core MPSoC, the (8, 8) configuration is a fully private one as each core is connected to one HDCT HWAcc and one VDCT HWAcc. This configuration is a consuming solution compared to the others configurations, it consumes almost 4x more logic resources than the fully shared configuration (1,1) and approximately 2x more than the configuration (4,4).

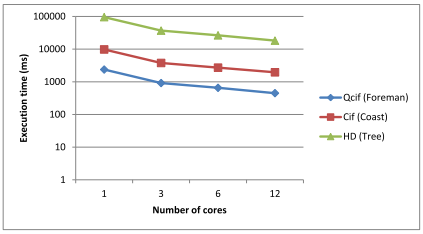

FIGURE 7. Execution results for integer encoder forward benchmark.

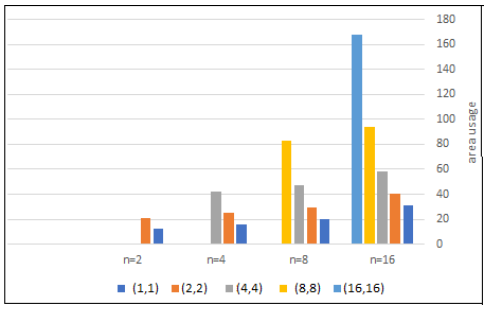

FIGURE 8. Area usage measured on Cyclone V for different MPSoCs (n = 2, 4, 8, 16) and (*HDCTHWAcc*,*VDCTHWAcc*).

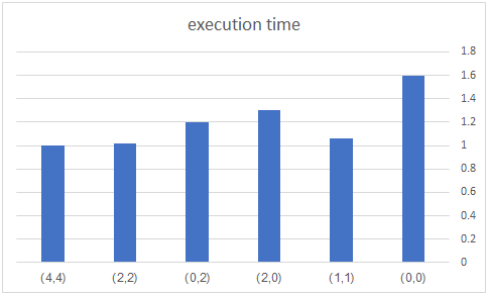

FIGURE 9. Execution time (y-axis) in seconds to encode 20 images measured for MPSoC with n=4 and different (*HDCTHWAcc*,*VDCTHWAcc*) configurations.

In Figure 9, we demonstrate the execution time improvement for MPSoC for $n = 4$ and different HWAccs configurations. The pure SW execution is represented by the configuration (0,0) and it spends 1.6s to encode 20 images whereas the fully private configuration (4,4) takes only 1s. The configurations (1,1) and (2,2) show a slight increase compared to the fully private configuration due to the overlapping delay to access a shared resource.

From figures 8 and 9, we conduct that configurations for n-cores architecture are bounded between the two extreme solutions: the fully private solution and the fully shared solution. For 4-cores architecture, the fully private solution, (4,4) configuration, preserves the best speed-up, reaching 1.6 over the SW solution to execute the jpeg encoder. However, the intermediate solutions, can also offer the required speed-up with a less-area consumption such as configuration (2,2), which consumes the half area resources and provides roughly the same speed-up. When the number of cores increases, the space of shared solutions increases dramatically and the search of the best solution would be infeasible. Moreover, the selection of the suitable connection network will thereby enlarge the design space further. This justifies the need for an automatic exploration tool that can generate the best configuration under area and time constraints.

D. MILP MODEL EVALUATION

In this part, we validate the efficiency of the proposed DSE to rapidly explore the design space and generate the best multi-processor configuration. The MILP model is run for JPEG codec application. The results for different number of cores are summarized in figure 10. In this figure, we depict the logic area usage of the HWAccs and the appropriate interconnection infrastructure while varying the required speed-up. Here, an area unit corresponds to 150 slices.

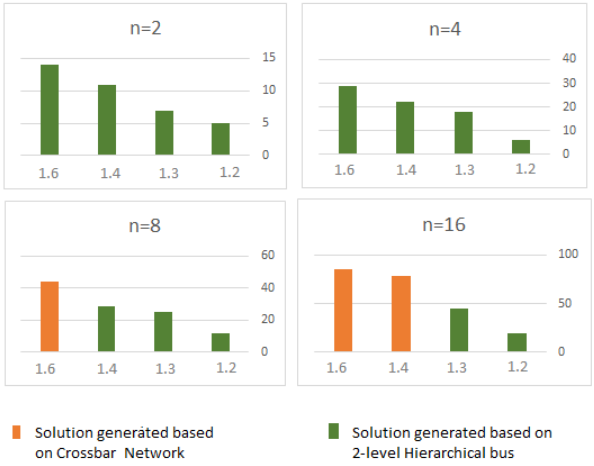

FIGURE 10. Estimated Area consumption (y-axis) of DSE solutions of different architectures (number of cores n=2, 4, 8 and 16) generated for different speed-up (x-axis).

For $n = 2$ and $n = 4$, the DSE tool generates hybrid configurations with shared and private HWAccs implemented with hierarchical bus. For these systems, for each performance constraint, the area usage is increased due to additional private HWAccs implemented or an increased number of shared HWAcc. For example, for $n = 4$ and speedup = 1.3, the DSE generates in 7 minutes a hybrid configuration based on hierarchical bus connected to two shared HWAccs for HDCT. To reach a speedup equal to 1.4, the DSE generates a configuration with one additional shared HWAcc for VDCT.

DSE results are compared to the real results for the MPSoC with $n = 4$ already presented in figure 9. We note for the configuration (2,2) the speedup provided by the implementation of this MPSoC configuration in the Cyclone V board is equal to 1.58. The same configuration is generated by our model for $n = 4$ and a speedup greater than 1.4. For speedup=1.6, the generated solution integrates four private HDCT HWAccs and one shared VDCT HWAcc. This solution provides approximately the same speedup measured for the configuration (4,4) in figure 9 with a moderate area usage. For $n = 8$ and $n = 16$, the DSE generates solutions with hierarchical interconnect for lightened constraint and more consuming-area solutions integrating a crossbar network for more strict performance constraints. For example, for $n = 16$ and speedup = 1.3, the MILP solver generates in 37 minutes a solution based on a hierarchical bus that connects one shared HDCT HWAcc and two shared VDCT HWAccs.

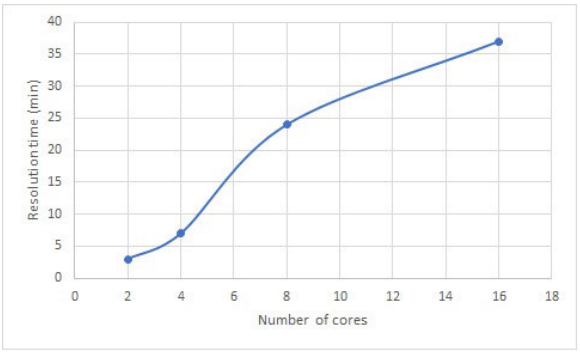

FIGURE 11. MILP resolution time run on CPLEX for different number of cores for JPEG application.

In figure 11, we report the time the Cplex solver takes to generate the feasible solution. We note that this time is light for a small number of cores and it increases substantially while increasing the number of cores. Even if the DSE tool consumes more time (in the order of minutes) to deal with complex MPSoC configurations, it remains fast and considerably helps the designer to choose the best configuration targeting a HW implementation.

These aforementioned experimental results clearly prove the efficiency of the developed high-level DSE framework to rapidly generate the best MPSoC hybrid configurations, varying different parameters: number of cores, used interconnection network, number and configuration of HWAcc, which minimizes the area consumption while respecting the required performance. MILP-based generated solutions have been also compared to real FPGA-based solutions, which demonstrate and confirm the validity of the developed DSE tool.

VI. CONCLUSION AND FUTURE WORK

Coupling the processor with an accelerator has been used to accelerate many applications and to obtain energy-efficient solutions. This paper described a heterogeneous multi-processor SoC architecture that can integrate accelerators according to the application needs. The proposed design methodology is based on a high-level DSE tool that assesses the trade-off between the execution time and area cost that are explored at two levels: HWAcc's sharing level and communication infrastructure level. Two real applications were executed in different systems composed of different cores number. Thereby, a direct correlation has been shown between sharing degree, communication infrastructure and performance/area trade-off.

In future work, we plan to test the system with other types of benchmarks. The proposed framework can be extended in a number of ways to take into account other constraints such as power distribution, energy and other constrained resources. Exploring the architecture with hard-cores is another interesting future work. We also plan to develop analytical model to automatically select the optimal architecture.

REFERENCES

[1] *An 531: Reducing Power With Hardware Accelerators*, ALTERA, San Jose, CA, USA, 2001.

[2] M. Baklouti, P. Marquet, J. L. Dekeyser, and M. Abid, "FPGA-based many-core system-on-chip design," *Microprocess. Microsys.*, vol. 39, nos. 4–5, pp. 302–312, Jun. 2015.

[3] D. Bouthaina, M. Baklouti, S. Niar, and M. Abid, "Shared hardware accelerator architectures for heterogeneous MPSoCs," in *Proc. 8th Int. Workshop Reconfigurable Commun.-Centric Syst.-Chip*, 2013, pp. 1–6.

[4] M. Brandalero, T. D. Souto, L. Carro, and A. C. S. Beck, "Predicting performance in multi-core systems with shared reconfigurable accelerators," *J. Syst. Archit.*, vol. 98, pp. 201–213, Sep. 2019.

[5] V. Britanák, "On the discrete cosine transform computation," *Signal Process.*, vol. 40, nos. 2–3, pp. 183–194, Nov. 1994.

[6] B. Dammak, M. Baklouti, R. Benmansour, S. Niar, and M. Abid, "Framework for a selection of custom instructions for ht-MPSoC in area-performance aware manner," *IEEE Embedded Syst. Lett.*, vol. 7, no. 4, pp. 105–108, Dec. 2015.

[7] B. Dammak, M. Baklouti, R. Benmansour, S. Niar, and M. Abid, "Hardware resource utilization optimization in FPGA-based heterogeneous MPSoC architectures," *Microprocess. Microsyst.*, vol. 39, no. 8, pp. 1108–1118, Nov. 2015.

[8] R. Grubisic and V. Zadrija, "Design of a system-level pipelined jpeg coder for a homogeneous multiprocessor platform using replication," Tech. Rep., May 2009.

[9] J. Rettkowski and D. Göhringer, "SDMPSoC: Software-defined MPSoC for FPGAs," *J. Signal Process. Syst.*, vol. 92, no. 10, pp. 1187–1196, Oct. 2020.

[10] L. Jia, Z. Luo, L. Lu, and Y. Liang, "Tensorlib: A spatial accelerator generation framework for tensor algebra," 2021, *arXiv:2104.12339*.

[11] G. G. Kumar, S. K. Sahoo, and P. K. Meher, "50 years of FFT algorithms and applications," *Circuits, Syst., Signal Process.*, vol. 38, no. 12, pp. 5665–5698, Dec. 2019.

[12] Y.-K. Kwok, "High-performance algorithms for compile-time scheduling of parallel processors," Ph.D. thesis, Hong Kong Univ. Sci. Technol., 1997.

[13] J. D. Lopes, M. P. Véstias, R. P. Duarte, H. C. Neto, and J. T. de Sousa, "Coarse-grained reconfigurable computing with the versat architecture," *Electronics*, vol. 10, no. 6, p. 669, Mar. 2021.

[14] K. Lubeck and O. Bringmann, "A heterogeneous and reconfigurable embedded architecture for energy-efficient execution of convolutional neural networks," in *Architecture of Computing Systems—ARCS*, M. Schoeberl, C. Hochberger, S. Uhrig, J. Brehm, and T. Pionteck, Eds. Cham, Switzerland: Springer, 2019, pp. 267–280.

[15] S. Lubobya, M. E. Dlodlo, G. Jager, and K. Ferguson, "Optimization of 4×4 integer DCT in H. 264/AVC encoder," Tech. Rep., Sep. 2011.

[16] E. Manor and S. Greenberg, "Using HW/SW codesign for deep neural network hardware accelerator targeting low-resources embedded processors," *IEEE Access*, vol. 10, pp. 22274–22287, 2022.

[17] R. S. Molina, V. Gil-Costa, M. L. Crespo, and G. Ramponi, "High-level synthesis hardware design for FPGA-based accelerators: Models, methodologies, and frameworks," *IEEE Access*, vol. 10, pp. 90429–90455, 2022.

[18] N. Paulino, J. C. Ferreira, and J. M. P. Cardoso, "Improving performance and energy consumption in embedded systems via binary acceleration: A survey," *ACM Comput. Surv.*, vol. 53, no. 1, pp. 1–36, Jan. 2021.

[19] A. P. Raveendran, J. A. Alzubi, R. Sekaran, and M. Ramachandran, "A high performance scalable fuzzy based modified asymmetric heterogene multiprocessor system on chip (AHt-MPSOC) reconfigurable architecture," *J. Intell. Fuzzy Syst.*, vol. 42, no. 2, pp. 647–658, Jan. 2022.

[20] B. A. Ringnyu, A. Tangel, and E. Karabulut, "Implementation of different architectures of forward 4×4 integer DCT for H. 264/AVC encoder," in *Proc. 10th Int. Conf. Electr. Electron. Eng. (ELECO)*, 2017, pp. 423–427.

[21] D. Rodriguez, D. Gomez, D. Alvarez, and S. Rivera, "A review of parallel heterogeneous computing algorithms in power systems," *Algorithms*, vol. 14, no. 10, p. 275, Sep. 2021.

[22] I. Skliarova, "A survey of network-based hardware accelerators," *Electronics*, vol. 11, no. 7, p. 1029, Mar. 2022.

[23] E. D. Sozzo, D. Conficconi, A. Zeni, M. Salaris, D. Sciuto, and M. D. Santambrogio, "Pushing the level of abstraction of digital system design: A survey on how to program FPGAs," *ACM Comput. Surveys*, vol. 55, no. 5, pp. 1–48, Dec. 2022.

[24] P. Stanicek and R. Farana, "Chosen optimization methods for search data," Tech. Rep., May 2011.

[25] L. Suriano, F. Arrestier, A. Rodríguez, J. Heulot, K. Desnos, M. Pelcat, and E. D. L. Torre, "DAMHSE: Programming heterogeneous MPSoCs with hardware acceleration using dataflow-based design space exploration and automated rapid prototyping," *Microprocess. Microsyst.*, vol. 71, Nov. 2019, Art. no. 102882.

[26] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Task scheduling algorithms for heterogeneous processors," in *Proc. 8th Heterogeneous Comput. Workshop (HCW)*, 1999, pp. 3–14.

[27] W. Wolf, "Multimedia applications of multiprocessor systems-on-chips," in *Proc. Conf. Design, Autom., Test Eur.*, vol. 3, 2005, pp. 86–89.

[28] S. Xu, S. Liu, Y. Liu, A. Mahapatra, M. Villaverde, F. Moreno, and B. C. Schafer, "Design space exploration of heterogeneous MPSoCs with variable number of hardware accelerators," *Microprocess. Microsyst.*, vol. 65, pp. 169–179, Mar. 2019.

[29] F. Zargari and S. Ghorbani, "Fast calculation of 8×8 integer DCT in the software implementation of H.264/AVC," in *Proc. 7th Int. Conf. Appl. Inf. Commun. Technol.*, Oct. 2013, pp. 1–4.

[30] X. Zhang, H. Ye, and D. Chen, "Being-ahead: Benchmarking and exploring accelerators for hardware-efficient ai deployment," 2021, *arXiv:2104.02251*.

BOUTHAINA DAMMAK was born in Sfax, Tunisia, in 1985. She received the engineering and M.S. degrees from the National Engineering School of Sfax (ENIS), Tunisia, in 2009, and the Ph.D. degree in computer science from the ENIS and the University of Valenciennes and Hainaut Cambresis, France, in December 2016. She is currently an Assistant Professor with the Department of Computer Science, Applied College, Princess Nourah bint Abdulrahman University. Her research interests include multiprocessor architecture, embedded systems, and the IoT applications.

MOUNA BAKLOUTI was born in Sfax, Tunisia, in 1983. She received the engineering and M.S. degrees from the Tunisian Polytechnic School, Tunisia, in 2006 and 2007, respectively, the Ph.D. degree in computer science from the National Engineering School of Sfax (ENIS), Sfax, and the University of Lille 1, France, in December 2010, and the HDR degree in electrical engineering from the ENIS, in June 2016. From 2012 to 2020, she was the Co-Founder and the Coordinator of the Research Master in Embedded Systems, ENIS, in cooperation with the University of Chemnitz, Germany. She is currently an Associate Professor of computer science with the University of Sfax. She is also a Research Member of the Computer Embedded Systems Laboratory (CES-Lab), ENIS. Her research interests include smart embedded systems design, the IoT applications, machine learning, and e-health.

DEEMA ALSEKAIT received the Doctor of Philosophy (Ph.D.) degree in information technology. She is currently a talented Assistant Professor with an excellent background in the field of information technology (IT). With the Ph.D. degree in information technology, she has firmly established herself as a leading figure in academia and research. She possesses a versatile skill set, excelling in strategic planning, data analytics, program development, public speaking, and research. In addition, she is a leading Advocate for improving access to careers in science, technology, engineering, and math (STEM). In the ever-changing IT field, she is a shining example of an outstanding professional. She perfectly combines her research expertise, pedagogical knowledge, and uncompromising commitment to social advancement. Her tireless efforts continue to inspire innovation, ensuring that the future of information technology remains bright and inclusive.