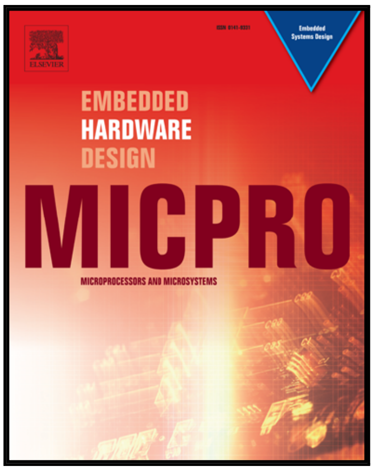


SystemC-based Electronic System-Level Design Space Exploration Environment for Dedicated Heterogeneous Multi-Processor Systems

Luigi Pomante, Vittoriano Muttillo, Marco Santic, Paolo Serri

PII: S0141-9331(19)30100-0
DOI: https://doi.org/10.1016/j.micpro.2019.102898
Reference: MICPRO 102898

To appear in: *Microprocessors and Microsystems*

Received date: 14 February 2019
Revised date: 7 August 2019
Accepted date: 24 September 2019

SystemC-based Electronic System-Level Design Space Exploration Environment for Dedicated Heterogeneous Multi-Processor Systems

Luigi Pomante[a,*], Vittoriano Muttillo[a], Marco Santic[a], Paolo Serri[a]

[a] *Università degli Studi dell'Aquila, Center of Excellence DEWS, Italy*

Abstract

This work faces the problem of the *Electronic System-Level* (ESL) HW/SW co-design of dedicated electronic digital systems based on heterogeneous multi-processor architectures. In particular, the work presents a prototype *SystemC*-based environment that exploits a *Design Space Exploration* (DSE) approach able to suggest an HW/SW partitioning of the system specification and a mapping onto an automatically defined architecture. The descriptions of the reference HW/SW co-design methodology and the main design issues related to the developed DSE SW tools, supported by two reference use cases that allows to understand the role of the DSE step in the whole design flow, represent the core of the paper.

Keywords: Electronic System-Level, HW/SW Co-Design, Design Space Exploration, Heterogeneous Multi-Processor Architectures, Dedicated Systems, SystemC.

1. Introduction

Dedicated Systems (DSs), as intended in the scope of this work, are digital electronic systems with an application-specific HW/SW architecture. They are

*Corresponding author
Email addresses: luigi.pomante@univaq.it (Luigi Pomante), vittoriano.muttillo@univaq.it (Vittoriano Muttillo), marco.santic@univaq.it (Marco Santic), paolo.serri@graduate.univaq.it (Paolo Serri)

specifically designed to satisfy a priori known application requirements (both *functional* and *non-functional*). DSs can be embedded in more complex systems and/or they can be subjected to hard/soft real-time constraints.

DSs based on heterogeneous multi-processor architectures (*Dedicated Heterogeneous Multi-Processor Systems*, D-HMPSs) have been recently exploited for a wide range of application domains, especially in the System-on-Chip (SoC) form factor (e.g., [1, 2, 3, 4]). Such systems can include several heterogeneous processors (i.e., by following the classification provided in [5]: *General-Purpose Processors*, GPPs; *Application Specific Processors*, ASPs; *Single Purpose Processors*, SPPs), memories, and a set of interconnection links among them. Moreover, processors can be adopted in the form of soft, hard or fuse (i.e., *hard-wired*) *Intellectual Property* (IP) cores or as discrete *Integrated Circuits* (ICs) mainly depending on the final system form factor (i.e., *on-chip, on-FPGA, on-board*) and scope (final product or platform).

D-HMPSs are so complex that the adopted *HW/SW Co-Design Methodology* plays a major role in determining the success of a product. Moreover, in order to cope with such a complexity, the selected methodology should allow the designer to start working at the so-called *Electronic System-Level* (ESL) of abstraction. This normally means to be able to start the design activities from an executable model of the system behavior based on a given *Model of Computation* (MoC) that is unifying for HW and SW and that can be described by means of a proper modeling language. For this, in the past years, a remarkable number of research works have focused on the *Electronic System-Level (ESL) HW/SW Co-Design of D-HMPS* (e.g., [6, 7, 8, 9, 10, 11, 12, 13]). In such works, the most critical issues are always related to the *System Specification* and *Design Space Exploration* (DSE) activities. In the first one, the designer models the behavior of the desired system (specifying also possible non-functional constraints), the available basic HW components, and the target HW architecture. The second activity is then related to the approach, automated or not, used to find the best HW/SW partitioning and mapping for the final system implementation. The main differences among the various approaches are related to the different amount of information and actions that are explicitly requested to the designer and that are so heavily influenced by his experience. In particular, a lot of approaches (especially those based on the on the *Y-Chart* principle [14]) explicitly require as an input the HW architecture to be considered for mapping purposes. So, at the best of our knowledge, there are few system-level HW/SW co-design flows, other than the one proposed in this work, that try to fully addresses the problem of both *"automatically suggest an HW/SW partitioning of the system specification"* and *"map the partitioned entities onto an automatically defined heterogeneous multi-processor architecture"*.

Two of the works most similar to the proposed one are [11] and [13]. The first one presents a *SystemC*-based *Co-Design Flow* that try to automate the process of prototypes generation. The approach exploits a commercial synthesis tool to generate automatically hardware accelerators starting from a *SystemC* behavioral model. However, the designer has to manually describe an *Architecture Template* that represents the architectural infrastructure to be used during the DSE. The second one is still more interesting. In fact, it presents a very flexible and extensible system-level MP-SoC design space exploration infrastructure that is also able to automatically generate hardware architectures. In contrast, the approach presented in this work is based on a more integrated and customized environment able to exploit from the very beginning ESL metrics and estimations to perform a DSE step. Moreover, the proposed approach is also able to explicitly consider also SPP; a processing class that is not clearly managed in [13]. Finally, to take a look also to a representative *SystemC*-based commercial product, it is worth citing *Intel CoFluent* [15] as a promising ESL modeling and simulation environment. Other than the model of the system behavior, it explicitly requires a manual modeling of both the hardware architecture and the mapping but, thanks to its *Eclipse-based* architecture, it is possible to think about some future plug-in extensions oriented to support the designer also in such activities.

According with this scenario, this work focuses on a *SystemC-based ESL DSE Environment for D-HMPS* and extends the one presented in [16] by presenting:

- more detailed design aspects about the proposed *SystemC* library extension and its improvement to support the *Alternation* concept (Section 3);
- main SW design issues related to the developed DSE SW tools (Section 4);
- the full integration and exploitation of the main tools (namely, PAM1, PAM2, and HEPSIM) in the context of the whole SystemC-based co-design flow applied to two reference use cases (Section 5).

The paper is organized as follows. Section 2 and Section 3 present the reference HW/SW co-design flow and the developed prototype *SystemC-based* HW/SW co-design environment. Then, Section 4 focuses on the DSE approach and related SW design issues, while Section 5 presents two reference use cases to show the main features of the proposed approach. Finally, Section 6 draws out some conclusions and outlines the future work.

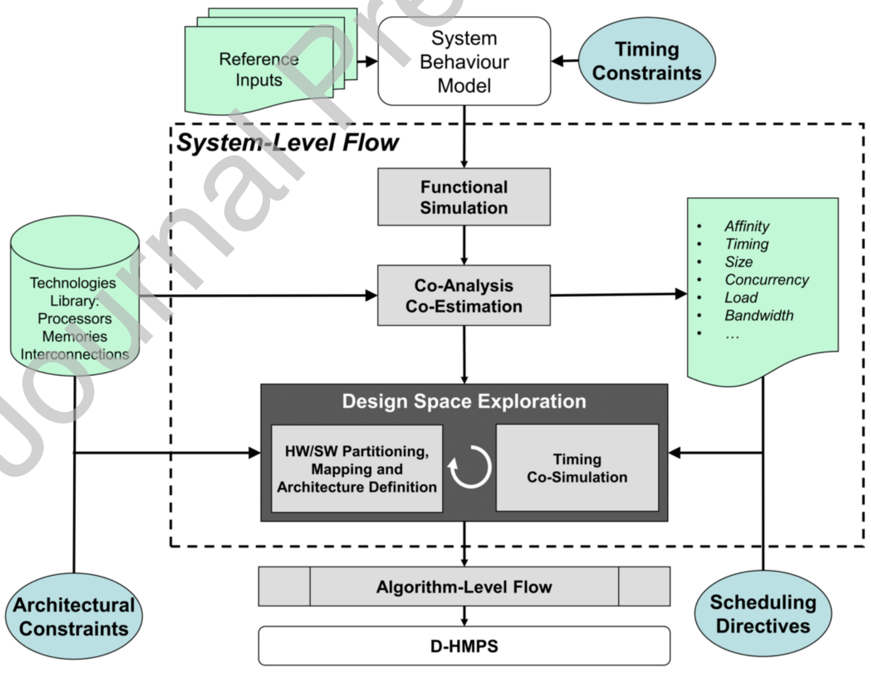


Figure 1: The reference co-design flow.

2. Reference ESL HW/SW Co-Design Flow

The reference ESL HW/SW co-design flow is shown in Figure 1: it reports the main steps and the needed information. The entry point is a model of the system behavior (SBM) based on the *Communicating Sequential Processes* (CSP) MoC [17, 18]. SBM represents the functional requirements while the non-functional ones are currently related only to a *Time-To-Completion* constraint (TTC) and the following architectural ones:

- a fixed set of available processors, interconnection links, and memories contained in a proper *Technologies Library* (TL);
- min and max number of available processors and interconnection links instances;
- available area for chip/board or an equivalent metric for FPGA;
- a reference template HW architecture and a set of rules that the tool has to follow while automatically building the final HW architecture;
- available scheduling policies and possible priorities among processes.

It is worth noting that the final result is considered successful only if all the requirements listed above are satisfied.

The following paragraphs briefly describe each step of the reference co-design flow from a general point of view while the next section shows the main customizations that have been performed to adapt such step to *SystemC* technology and tools.

2.1. System Behavior Model

In the proposed approach, SBM is captured by means of a *procedure-level* internal model called *Procedural INteraction Graph* (PING [6], an example is shown in Figure 2) while each procedure is then described, at *statement-level*, by using a proper modeling language suitable to represent CSP features. As already stated in the title, the language adopted in this work is *SystemC*.

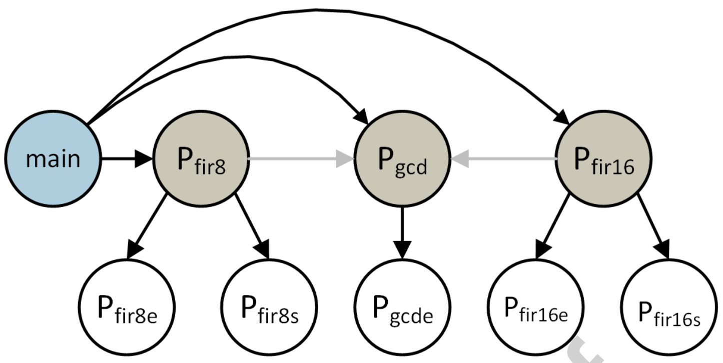


Figure 2: Procedural INteraction Graph (PING) example.

2.2. Functional Simulation

The first step of the proposed HW/SW co-design flow is the *Functional Simulation* where SBM is simulated to check its correctness with respect to some *Reference Inputs*. Such data sets are of critical importance since they have to be as much as possible representative of the actual operating conditions of the system. Such a simulation is normally very fast, and it allows also to take into account timed inputs: there is a concept of simulated time, but it doesn't consider the time needed to execute the statements, related to both computation and communication operations, i.e., statements are executed in 0 simulated time. If SBM is not correct (i.e., wrong outputs or critical conditions such as e.g., deadlocks) it should be properly modified and simulated again. The early detection of anomalous behaviors allows the designer to correct the specification avoiding a late discovery of problems that could lead to time-consuming design loops.

2.3. Co-Analysis and Co-Estimation

This step aims at extracting as much information as possible about the system by analyzing the SBM while considering the provided TL. This step is composed of *Co-Analysis* [6, 19] and *Co-Estimation* activities [19, 20, 21, 22]. *Co-Analysis* provides a set of metrics expressing the *Affinity* of each CSP process towards the given set of processing classes, and some information about

Concurrency. In particular, the latter identify the set of processes and channels that can be potentially executed concurrently. *Co-Estimation* provides a set of estimations about *Timing*, *Size*, *Load* and *Bandwidth*. *Timing* is related to the estimation of the number of clock cycles needed, by each processor in TL, to execute each single statement composing the processes in SBM. *Size* represents the number of ROM/RAM bytes needed for SW implementations and *equivalent gates* (or alternative metrics for FPGA) for HW ones. Finally, by exploiting Timing data and considering the TTC constraint, it is also possible to estimate the *Load* associated with the execution of SBM processes when mapped on a single instance of each processor in TL, and the *Bandwidth* needed to the different processes to communicate while fulfilling the TTC constraint.

2.4. Design Space Exploration

Finally, the reference co-design flow reaches the *Design Space Exploration* (DSE) step [6, 23, 19] that is constituted of two iterative activities: *"HW/SW Partitioning, Mapping and Architecture Definition"* and *"Timing Co-Simulation"*. All the metrics and estimations obtained in the previous steps are then used to drive the DSE, together with additional information/constraints provided by the designer: available *Scheduling Directives* (i.e., available scheduling policies and possible priorities among processes) and possible *Architectural Constraints* (i.e., max number of instances for each available processor and interconnection link). The *HW/SW Partitioning, Mapping and Architecture Definition* activity is decomposed in two phases and it is based on a genetic algorithm that allows to explore the design space looking for feasible mapping/architecture items suitable to satisfy imposed constraints. Then, the *Timing Co-Simulation* activity considers suggested mapping/architecture items to actually check for TTC constraint satisfaction. If the suggested mapping/architecture item doesn't meet such a constraint, the designer should perform again the design space exploration by changing some exploration parameters, by modifying the starting SBM, by enriching the TL with new elements, or by relaxing some constraints.

2.5. Algorithm-Level Flow

When the mapping/architecture item proposed by the DSE step is satisfactory, it is possible to implement the system. For this, the SW-mapped processes are typically transformed in C code, with the support of a possible embedded and/or real-time OS, while the HW-mapped ones are transformed in synthesizable HDL code or implemented by means of existing COTS component depending on the final system form factor (i.e., *on-chip, on-FPGA, on-board*) and scope (final product or platform). It is worth noting that such transformations will be done automatically or manually depending on the language and the coding style adopted to describe the SBM. This step is fully based on existing commercial algorithm-level methodologies and tools that are out of the scope of this work.

2.6. Reference D-HMPS Template HW Architecture

The reference D-HMPS template HW architecture of the proposed methodology is a heterogeneous multiprocessor one with distributed local memory. It is based on some basic HW elements, called *Basic Block* (BB), that represent the minimal computation, storage and communication units in the system. They can be different in their internal components giving so rise to possible heterogeneous multiprocessor systems. In particular, as shown in Figure 3, each BB is composed of three main elements: a *Processing Unit* (PU), a *Local Memory* (LM) and a *External Communication Unit* (ECU). Finally, these elements are interconnected by an *Internal Interconnections Link* (IIL, typically a shared bus where the PU is the master and the ECU is a memory-mapped SPP slave).

All the elements in a BB are characterized by means of the data available in the TL. Currently, PU can belong to one of three different processor classes (i.e., GPP, ASP and SPP) where GPP and ASP are characterized by the cost (€) and the maximum load (L_{max}, typically less than 100% to take into account possible OS overhead), while SPP is characterized by the cost (€) and the max number of equivalent gates Geq_{max} (in the case of reconfigurable logic the last metric can be changed with the max number of available cells or LUTs). LM is the memory directly addressable by the PU (i.e., no CSP channels are

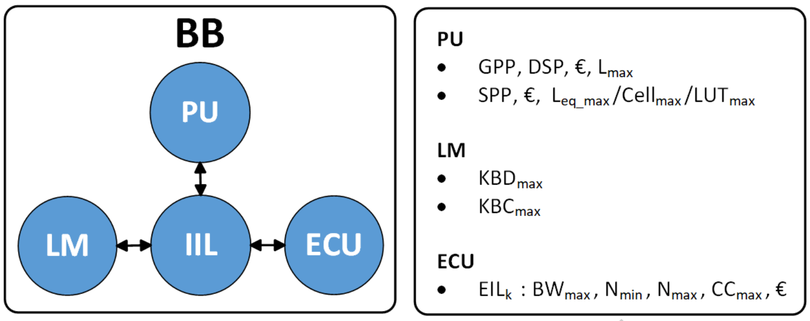

BB
PU
LM
IIL
ECU
PU
GPP, DSP, €, L_{max}
SPP, €, L_{eq_max}/$Cell_{max}$/LUT_{max}
LM
KBD_{max}
KBC_{max}
ECU
EIL_k : BW_{max}, N_{min}, N_{max}, CC_{max}, €

Figure 3: The Basic Block and its parameters.

needed) and it is characterized by a cost (€),max size for data (KBDMAX) and max size for code (KBC_{max}). Finally, ECU is characterized by the set of *External Interconnection Links* (EIL) that it can manage. Moreover, each EIL is characterized by the following parameters:

- the max available bandwidth (BW_{max});
- the min/max number of BBs that shall/can use a single EIL instance (N_{min} and N_{max});
- the max number of allowed concurrent communications (CC_{max});
- the cost (€).

So, considering some instances of BBs and interconnecting them by means of some instances of EILs (i.e., a pair of ECUs shall be able to manage at least a common EIL to allow communication among the related BBs) it is possible to define a feasible dedicated heterogeneous multiprocessor architecture on which the system functionality can automatically be mapped to. In the proposed approach, such an architecture is then represented, for design space exploration purposes, by means of an internal model based on the so-called *Architecture Graph* [24].

3. SystemC-based ESL HW/SW Co-Design Flow

The reference ESL HW/SW co-design flow is shown in Figure 1: it reports the main steps and the needed information as described in the previous section. The following paragraphs briefly describe each step together with the main customizations that have been performed to adapt them to *SystemC* technology and tools.

3.1. SystemC-based System Behavior Model

Since SBM is based on CSP, the *SystemC* library has been extended to properly model such kind of processes and, in particular, CSP channels. In fact, while processes are modeled by exploiting standard SC_THREAD, CSP channels have been modeled by introducing a proper SC_CSP_CHANNEL class in the *SystemC* library. In particular, a process is an SC_THREAD presenting an infinite loop behavior. It is able to directly access only to its local variables and so it communicates with other processes only by means of CSP channels. Moreover, in the considered SC_THREAD, only basic *C/C++* statements and *SystemC* data types are allowed while avoiding a full OOP approach since it can introduce critical issues for estimation and HW synthesis activities (in fact, the adopted restrictions have been inspired by [25]).

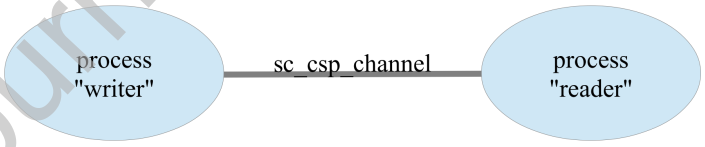


Figure 4: Point-to-point link between writer and reader processes.

The SC_CSP_CHANNEL class has been obtained by modifying the SC_FIFO one while providing an interface that offers blocking *write()* and *read()* methods. Main modifications with respect to the SC_FIFO class are related to the introduction of a full-handshake protocol to allow synchronous data exchange, as expected for a CSP channel. The goal is to create a point-to-point link between two processes; the transfer of data on the channel requires a *rendez-vous* of the two processes: it happens only when the writing process is ready to write at the same time the reading process is ready to read (Figure 4). For such a reason, it is a blocking channel for a process, if the counterpart process is not ready for the data transfer.

As said before, in order to let the data transfer and to synchronize the two processes, SC_CSP_CHANNEL interface exposes the methods *write()* and *read()*, while its implementation makes use of some private attributes (two *bool* flags, called *presence flags*, and *wait()* and *notify()* mechanisms). The situation when the writer process comes first is illustrated in Figure 5. In particular, first of all, the *write()* method sets the presence flag *rd-to-write* to *true* and then it checks for the presence of the reader process (by means of the presence flag *rd-to-read*). Since, for hypothesis, the reader is not present, the *write()* method suspends itself by calling a *wait()* for a *rd-to-read-event*. It will be the standard SystemC scheduler to resume the *write()* method when such an event will be notified by another process (i.e., in this case the reader one). In fact, when the reader tries to access to the channel by means of the *read()* method, it sets the presence flag *rd-to-read* to *true* and then discovers that the writer process is already present (by means of the of the presence flag *rd-to-write*), so it notifies a *rd-to-read-event* and waits for a *rd-to-write-event*. This suspends the *read()* method. When the standard SystemC scheduler will manage the notification of the *rd-to-read-event* event, this will lead to the resume of the *write()* method related to the writer process. Then, the data to be transferred is copied in a temporary data member (i.e., *cps-buf*), the presence flag *rd-to-write* is set to *false*, a *rd-to-write-event* is notified to resume the *read()* method of the reader process, and another *wait()* for a *rd-to-read-event* is called to complete the full handshake. Finally, when the standard SystemC scheduler will manage the notification of the *rd-to-write-event* event, this will lead to the resume of the *read()* method related to the reader process. Then, the data to be transferred is copied from *cps-buf*, the presence flag *rd-to-read* is set to *false*, a *rd-to-read-event* is notified to resume the *write()* method of the reader process, and the *read()* method is completed. When the standard SystemC scheduler will manage the notification of the *rd-to-read-event* event, the *write()* method will be completed too.

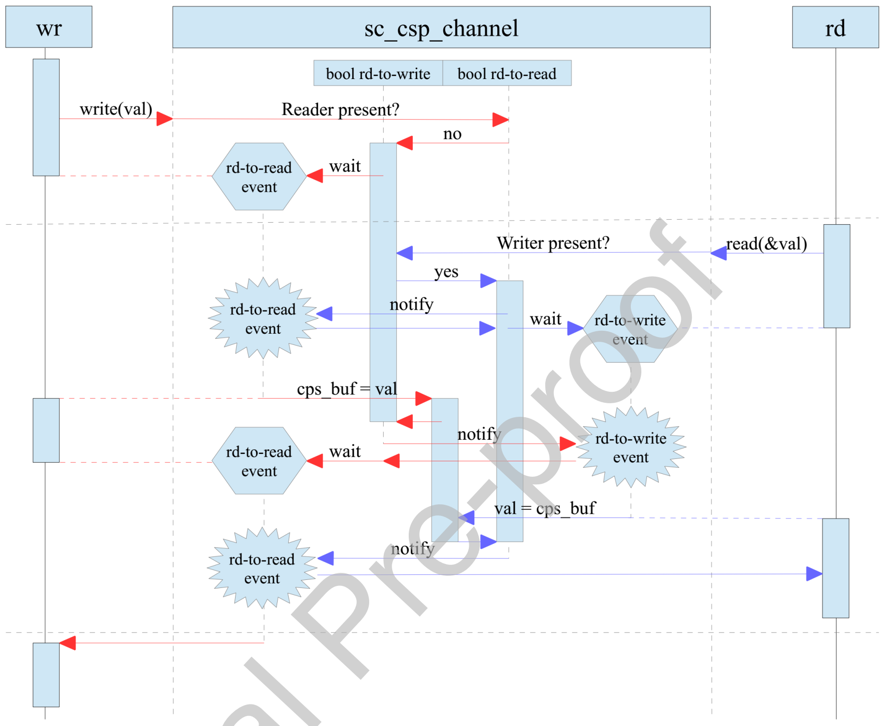


Figure 5: Full Handshake method (writer process comes first).

The other situation, the case of the reader process coming first, is illustrated in Figure 6. Note how the two diagrams have some common phases:

- "waiting for the counterpart to come", in which is the first process accessing the channel, after it has freely set its presence;
- "waiting for the counterpart to put/get data", in which is the second process accessing the channel, after it has set its presence and notified the counterpart of it;
- "waiting for the counterpart to leave", in which is again the first process that has accessed the channel.

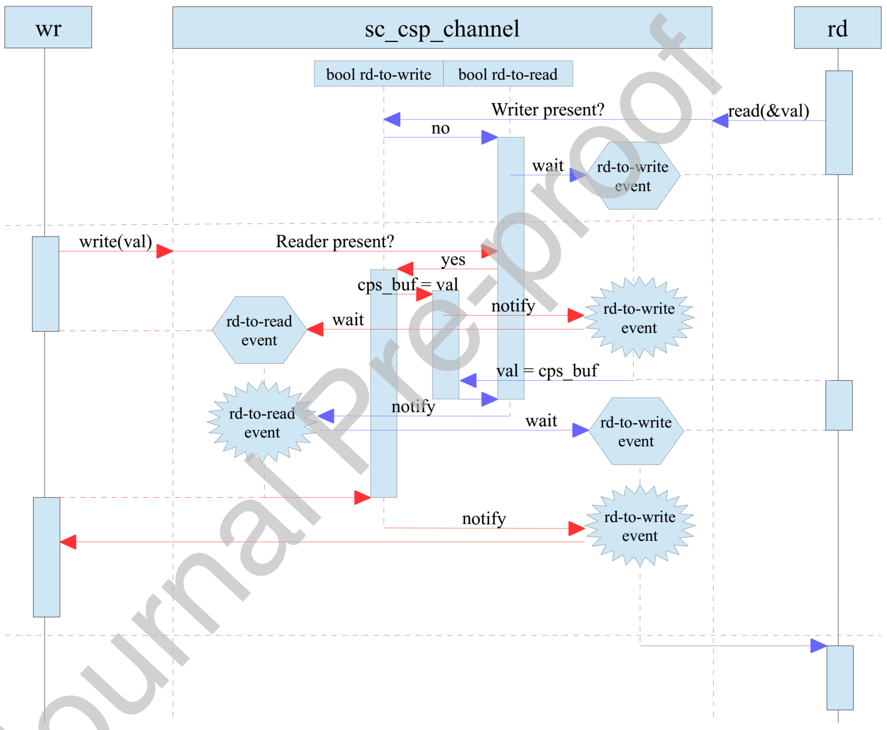


Figure 6: Full Handshake method (reader process comes first).

It is worth noting that the last phase is needed to prevent a "too-early-next-read/write", that would find a not updated presence flag (i.e., a presence flag relative to a previous transfer).

Finally, additional non-blocking *write_test()* and *read_test()* methods have been added to the SC_CSP_CHANNEL class in order to provide a mechanism to build an *Alternation*-like statement (e.g., as the ALT statement available in the OCCAM language [26], also based on CSP): in this way a SC_THREAD is also able to check for data from more than a channel at the same time. In fact, an ALT combines a number of processes guarded by inputs. The ALT performs the process associated with a guard which is ready.

```
ALT
    left ? packet
        stream ! packet
    right ? packet
        stream ! packet
```

Listing 1: OCCAM example

Considering the OCCAM example shown in Listing 1 [27], its effect is to merge the input from the two channels named *left* and *right*, on to the channel stream. The ALT (Figure 7) receives an input from either channel *left* or channel *right*. A ready input is selected, and the associated process is performed. Consider this example in detail. If the channel *left* is ready, and the channel *right* is not ready, then the input *left?packet* is selected. If the channel *right* is ready, and the channel *left* is not ready, then the input *right?packet* is selected. If neither channel is ready then the alternation waits until an input becomes ready. If both inputs are ready, only one of the inputs and its associated process are performed. The main issue for the implementation of an ALT construct is the blocking (reading/writing) nature of the SC_CSP_CHANNEL. To avoid to be blocked on *wait()* functions, a set of non-blocking test methods has been added to the class (i.e., *read_test()* and *write_test()*).

With the above mentioned methods, a process can verify if on the other side

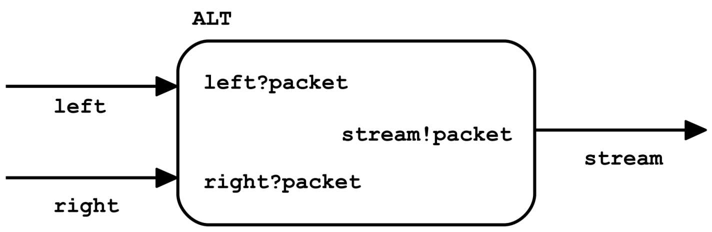


Figure 7: ALT Construct.

of the channel the counterpart is ready for the data transfer; of course only a verification is not sufficient, because the process that is evaluating a set of channels has to stop if none of the transfers are ready. For such a reason a new *wait()/notify()* event has been added to the SC_CSP_CHANNEL class: *sc_event ready_alt_event* that can be enabled on the channel by using the method *void register_alt()* and can be accessed through *const sc_event & get_sc_event()*.

If a process has an ALT on two channels for a read (as in the previous example), the notify will be triggered by one of the processes that writes on such channels; viceversa, if a process has an ALT on two channels for a write, the notify will be triggered by one of the processes that reads from such channels.

Following the scheme illustrated in Figure 5 and Figure 6 about the SC_CSP _CHANNEL *rendez-vouz*, the notification will happen in the phase "waiting for the counterpart to come", since the process with ALT will not make (any) blocking read or write, but only tests, and therefore it will not be waiting blocked on the (single) channel, as shown in Figure 8.

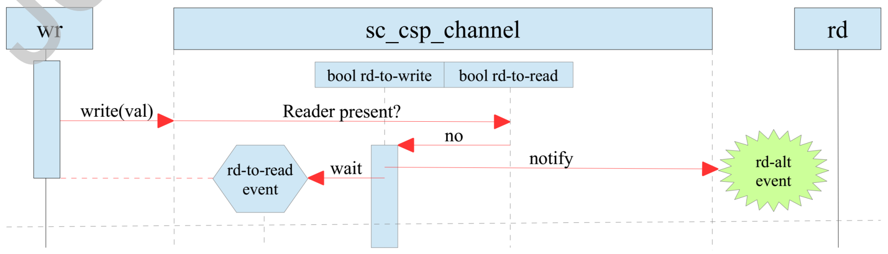


Figure 8: ALT Construct Implementation.

The process with ALT, as said before, will make first tests, calling *read_test()* for the channels in ALT and, only if none is ready for the transfer, will wait for the ready_alt_event. If, instead, one of the channels in ALT is ready, it will be read with *read()* method. Referring to the example of Figure 7, the code in Listing 2 represents a possible implementation.

```
left->register_alt();
right->register_alt();
while(1){
    if (left->read_test() | right->read_test()){
        if (left->read_test()){
            packet = left->read();
            stream->write(packet);
        }else if (right->read_test()){
            packet = right->read();
            stream->write(packet);
        }else{
            // statement not reachable
        }
    }else{
            sc_core::wait(left->get_alt_event() | right->get_alt_event());
    }
}
```

Listing 2: ALT example

It is worth noting how it is possible, in the implementation, to give a priority to the channels in the case in which more than one is ready at the same time; this can be assigned with the order of simple ”if conditions”, without the use of ”if - else if” construct. Also note that with this kind of implementation, when a process with ALT is active, it will find one ready channel or wait on *alt_event* and so there is no possibility to lose a notification. Naturally, it is possible to put in the ALT construct more than two channels and mix input (from which to *read()*) and output (on which to *write()*) channels.

All these elements allow an effective ESL modeling of D-HMPS behavior.

In particular, the whole system behavior is enclosed into a single SC_MODULE containing all the processes and channels. Other SC_MODULE and SC_CSP _CHANNEL objects are then used to model the *Test-Bench* and connected to the system by means of proper SC_PORT. A schematic example is shown in Figure 9. 325

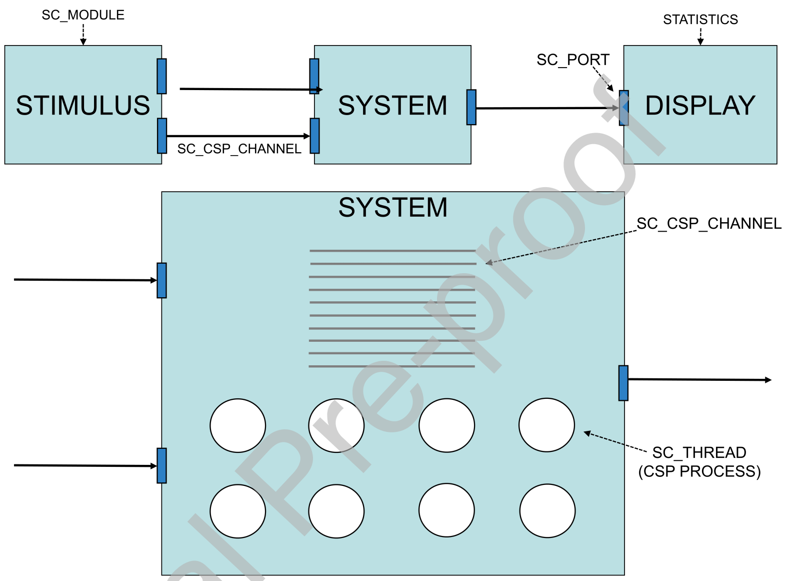


Figure 9: An example of System and Test-Bench modeling.

3.2. SystemC-based Functional Simulation

Since the *SystemC* model representing SBM is executable by construction, this step is straightforward, and it is directly based on the simulation kernel provided by the standard *SystemC* library [28] (commercial simulators can be used as well). 330

3.3. SystemC-based Co-Analysis and Co-Estimation

This step, as described above, is partially based on techniques and tools that are *SystemC* independent. The only *SystemC*-based tool is the simulator that is described in the next section.

3.4. SystemC-based Design Space Exploration

Since such a step is the main focus of this paper, it is described with more detail in the next section.

4. *SystemC*-based Electronic System-Level Design Space Exploration

As described before (see, in Figure 1, the "Design Space Exploration" box), this step is composed of two iterative activities: *"HW/SW Partitioning, Mapping and Architecture Definition"* (that is further decomposed in two phases, as shown in Figure 10 and described in the next paragraphs) and *"Timing Co-Simulation"*. The final goal is the automatic identification of:

- a HW/SW partitioning of the processes;
- a heterogeneous multi-processor architecture composed of several connected BB able to satisfy the architectural constraints;
- a mapping of the partitioned processes onto the identified BB able to satisfy the TTC constraint.

It is worth noting that the reference D-HMPS template HW architecture considered for the described co-design environment is limited to be *heterogeneous mono-core multi-processor* ones (i.e., one PU for BB), where each PU has its own local memory and the system has a distributed memory architecture. Moreover, the ASP class is currently restricted only to *Digital Signal Processors* (DSP).

Then, the goal of this section is to describe the main design aspects related to the SW tools that support the two-phases "HW/SW Partitioning, Mapping and Architecture Definition" activity, and the *"Timing Simulation"* one.

4.1. HW/SW Partitioning, Mapping and Architecture Definition (1st Phase)

The first phase (top of Figure 10) is related to the mapping of SBM (i.e., a CSP specification written in *SystemC* and represented by means of a PING) onto a dedicated architecture by following the approach described in [6]. The internal-model representing the specification, annotated by means of the *Co-Analysis*

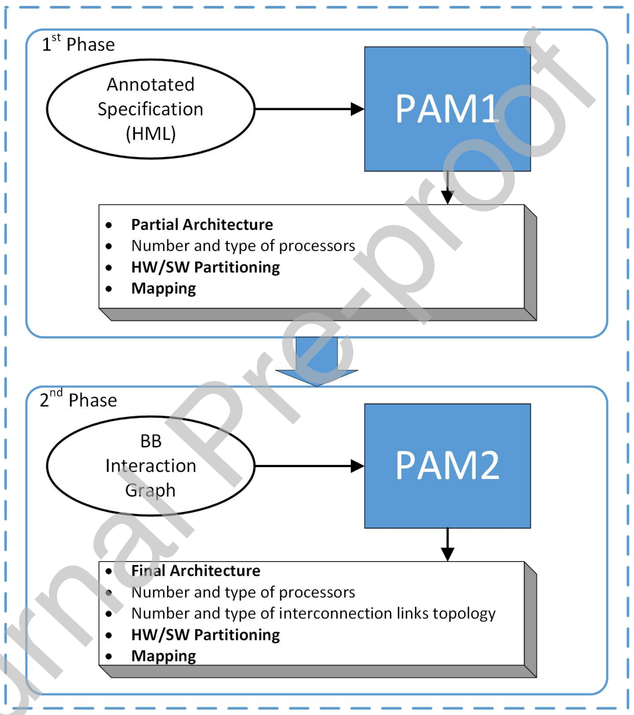
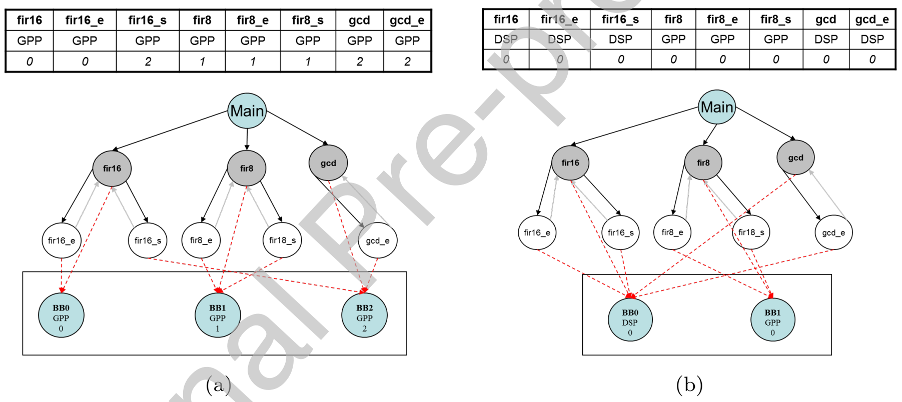

1st Phase
Annotated Specification (HML)
PAM1
Partial Architecture
Number and type of processors
HW/SW Partitioning
Mapping
2nd Phase
BB Interaction Graph
PAM2
Final Architecture
Number and type of processors
Number and type of interconnection links topology
HW/SW Partitioning
Mapping

Figure 10: The two-phases "HW/SW Partitioning, Mapping and Architecture Definition" approach.

and Co-Estimation step, is provided as input to the PAM1 (i.e., *HW/SW **P**artitioning, **A**rchitecture Definition and **M**apping - Phase **1***) tool. So, starting from an annotated PING the first phase goal is to determine number and type of BB/PUs and a mapping of PING procedures onto them while minimizing a cost function by using a genetic approach [29] where each individual of the population represents a possible mapping/architecture item (each procedure is associated with a type of processor and an instance number as shown in Figure 11). The considered cost function is composed of several terms related to system-level metrics that are evaluated for each individual during the genetic evolution [6].


fir16	fir16_e	fir16_s	fir8	fir8_e	fir8_s	gcd	gcd_e
GPP	GPP	GPP	GPP	GPP	GPP	GPP	GPP
0	0	2	1	1	1	2	2

fir16	fir16_e	fir16_s	fir8	fir8_e	fir8_s	gcd	gcd_e
DSP	DSP	DSP	GPP	GPP	GPP	DSP	DSP
0	0	0	0	0	0	0	0

Figure 11: Two individuals and their corresponding architecture.

In fact, the initial population is randomly generated, while during the evolution of the population the algorithm performs the optimizations that minimize the cost function following the classical rules of genetic algorithms (i.e., *crossover* and *mutation*). During the evolution, the individuals that score the worst values tend to be replaced by better ones. Several parameters in the algorithm (e.g., population size, number of generations, mutation probability, etc.) allow a wide exploration of the design space, with the goal to avoid local minima. The output of the first phase is so related to the computation aspects of the architecture: number and type of BB/PUs in the architecture and the mapping of each PING procedure onto them.

To support the 1st *Phase* in the context of the proposed *SystemC*-based co-design environment, the tool developed in [6] (called *EmuP*) has been completely re-designed while keeping in mind two main goals:

- easy integration in the whole co-design flow;
- highly re-use opportunity for the development of PAM2 tool (2nd *Phase*).

The main design issues are briefly described below.

4.1.1. Inputs Modeling

Main inputs to the PAM1 tool (i.e., annotated PING and configuration parameters for the genetic algorithm) are provided by means of two XML files. The UML model related to the annotated PING XML schema is shown in Figure 12. It describes all the information associated to a generic PING procedure (e.g., *Affinity*, *Load*, etc.) allowing to build its runtime representation to be used during the DSE by following a model-driven approach.

4.1.2. Technologies Library Modeling

Other than the annotated PING, there is the need to model also available processors and interconnection links (i.e., the TL). It is worth noting that the adopted approach, shown in Figure 13, takes also into account requirements for PAM2 tool (2nd *Phase*) by exploiting the *ModelEntity* abstract classes and polymorphism. Then, each subclass is able to contain all the data available in the TL. Such data will be stored in an XML file while a runtime representation will be built and exploited during the DSE.

4.1.3. PAM Specification Modeling

The set of inputs and the TL previously described represent the system specification for the DSE and become the starting point to build the initial random population for the genetic algorithm. With the adopted design approach, the genetic algorithm engine is able to evolve the population by means of a *Specification* class, independently from the specific DSE phase (1st *or* 2nd).

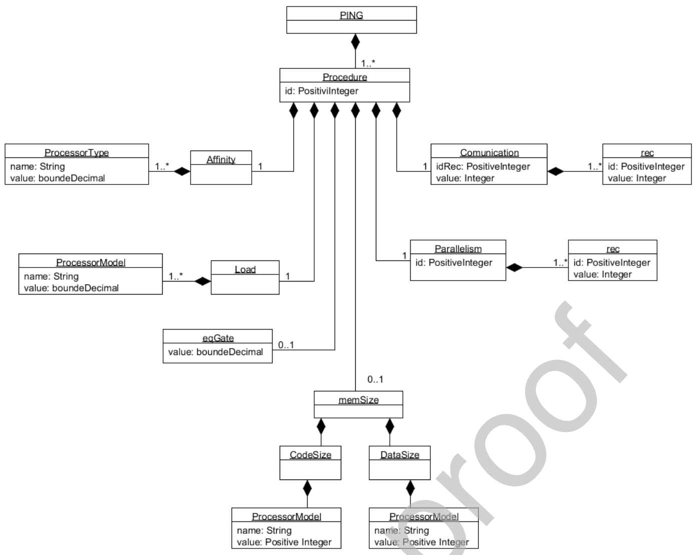


Figure 12: UML model related to the annotated PING XML schema.

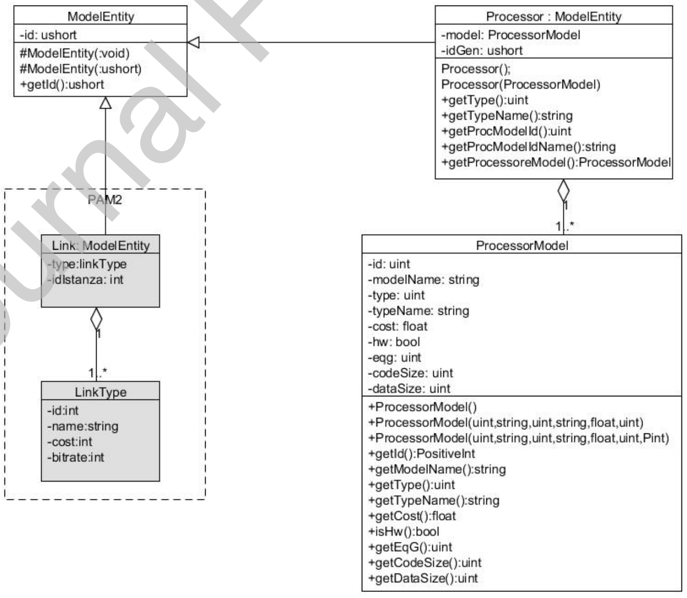


Figure 13: Technologies Library modeling.

4.1.4. Optimization engine, individuals and allocation modeling

Since the genetic approach is the main common element in both PAM1 and PAM2 tools, also if they rely on different individuals to perform different cost functions optimization, the design has been oriented to allow a meaningful re-use. The main issue is that in both cases there is the need to find a (sub)optimal allocation between two sets of objects: procedures to processors in the first case, and channels and links in the second one. So, the individuals and the allocation have been modeled to exploit this common point. In fact, the optimization engine works with a set of generic individuals and so it can be simply reused by specializing the cost function evaluator as shown in Figure 14 for PAM1. This approach makes easy to change the algorithm for the optimization as well as to implement various cost function evaluators to explore the design space in a different way.

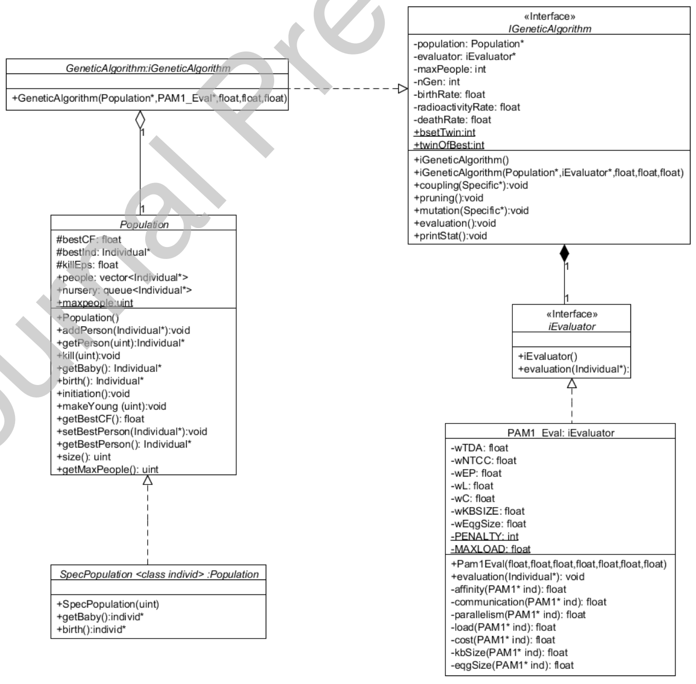


Figure 14: Optimization engine modeling.

4.2. HW/SW Partitioning, Mapping and Architecture Definition (2nd Phase)

The starting point of the second phase (bottom of Figure 10) is the so-called BBs Interaction Graph (BING), i.e., an internal model used to represent the partial system obtained at the end of the first phase [23]. Such a model is provided as input to the PAM2 (i.e., *HW/SW* ***P****artitioning,* ***A****rchitecture Definition and* ***M****apping - Phase* ***2***) tool. Starting from a BING the goal is to determine number and type of EILs among BBs to minimize a cost function by using a genetic approach where each individual of the population represents a possible interconnections/topology item. Such a cost function is composed of several terms related to the system-level metrics [23] that are evaluated for each individual during the genetic evolution.

In order to perform the exploration of the design space, the initial population is randomly generated, while during the evolution of the population the algorithm follows the classical rules of genetic algorithms. However, such an approach could give rise to unfeasible physical solutions, so the feasibility of the children is automatically evaluated and properly taken into account: unfeasible ones get a higher cost function to be probably removed from the population. In order to check such a feasibility, ECU shall be characterized with respect to the BBs belonging to the BING. This is obtained by means of an ECU Characterization Matrix (ECUCM) that explicitly indicates the EILs that each BB is able to manage.

Furthermore, to support the 2^{nd} *Phase* in the context of the proposed *SystemC*-based co-design environment, the tool developed in [23] (called *ETOP3*) has been completely re-designed by fully exploiting the design approach and several of the classes described in the previous paragraph.

4.3. Timing Co-Simulation

Timing Co-Simulation is performed by means of a *HW/SW Timing Co-Simulator* fully described in [19]. Briefly, it has been specifically built on the base of standard *SystemC* library by introducing some additional classes (i.e., *SystemManager*, *SimulationManager*, and *SchedulingManager*, as represented in Figure 15). Thanks to them, the current co-simulator is able to take into account a heterogeneous multi-processor architecture, a processes-to-BB/PU and a channel-to-links mappings, and all the relevant information previously collected, to check if a given TTC constraint is going to be satisfied. Additionally, the designer can also select the scheduling policy to be used for processes implemented in SW and allocated on the same BB/PU. With more details, *SystemManager* allows to take into account all the details about the system to be simulated (i.e., number and type of BB/PUs, number and type of interconnection links, Affinity and Timing data needed for simulation, mapping, etc.), while *SimulationManager* allow to configure the simulation type (i.e., functional or timing) and defines a set of macro used to instrument the simulated *SystemC* code. Finally, *SchedulingManager* allow taking into account the effects of the possible scheduling activities.

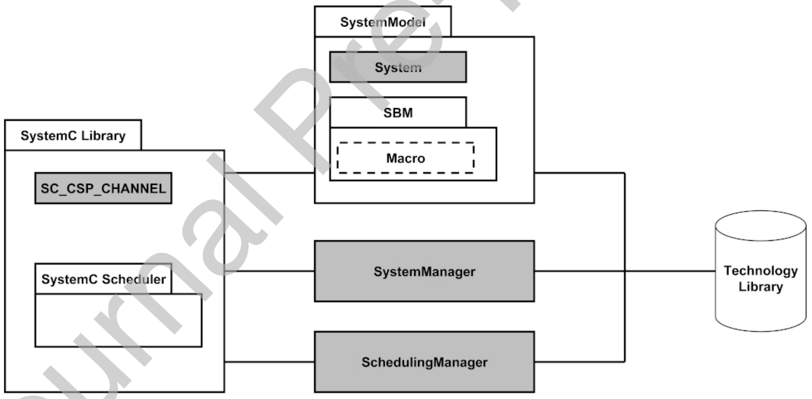


Figure 15: *SystemC* HW/SW Timing Co-Simulator architecture.

As already said, the designer can select a scheduling policy for processes implemented in SW and allocated on the same BB/PU. This scheduler acts as a user-level one with respect to the standard *SystemC* simulation kernel.

5. Reference Use Cases

In order to show the main features of the proposed *SystemC-based ESL DSE Environment*, two use cases are reported below: *Extended FIR-FIR-GCD* and *Sobel Filtering*.

5.1. Use Case #1: Extended FIR-FIR-GCD

This use case is an extension of the one already presented in [16]), called FIR-FIR-GCD. It is worth noting that it doesn't perform a meaningful computation but it is just used as a simple case study (i.e., it is a toy example). In fact, it receives pairs of numbers as input, performs two *fir* (i.e., *finite impulse response*) filtering operations which results are used to evaluate the GCD (*Great Common Divisor*). With respect to the one presented in [16], it has been extended in two ways:

- by exploiting the possibilities to use the ALT mechanism in the modeling activity and to consider different scheduling directives in the DSE (*Extension #1*).
- by exploiting PAM2 to perform DSE with respect to the communication architecture (*Extension #2*).

5.1.1. Extension #1

Let be the SBM, represented by the CSP shown in Figure 16, composed of eight processes and twelve channels (Figure 9 provides also a schematic view of such a system while the related PING is shown in Figure 2). Two more processes and three more channels are then used to describe and connect the test-bench. In this case, thanks to the ALT mechanism, it has been possible to introduce buffers into *fir8*, *fir16* and *gcd* processes to avoid loosing data from the external environment due to the blocking communication mechanism and to improve average timing performances. In fact, by means of the ALT mechanism it is possible to check the availability of inputs without incurring into the block,

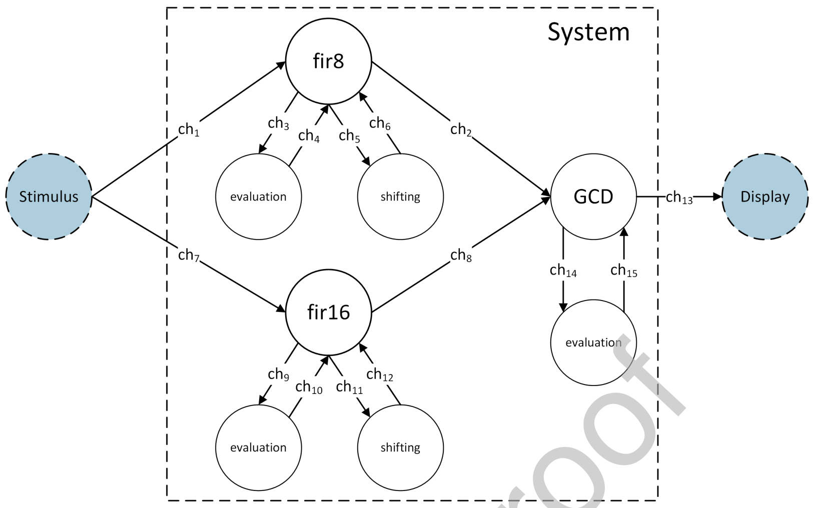


Figure 16: FIR-FIR-GCD CSP.

so allowing the computation to be performed until the buffers contains data to be processed, while still being able to fill the buffers when new data arrives.

The *Technologies Library* considered for this case study is composed of three different PUs: *Intel MPU8051* (16 MHz, GPP), *Microchip PIC24* (32 MHz, DSP) and *Xilinx Spartan3AN* (50 MHz, SPP). TL contains all the relevant information about processors, memories and interconnections needed to perform the DSE. For the moment, as in [16], let the processors being able to communicate only by means of a shared bus (i.e., buses are the only interconnection links actually contained in the TL).

So, the first steps of the flow (i.e., *Functional Simulation, Co-Analysis* and *Co-Estimation*) are the same of [16]. Once collected all the metrics and all the estimations needed for the DSE step, the following constraints are imposed:

- *Timing Constraints* - Given the estimated *Worst-Case Time-To-Completion* (i.e., WCTTC=5.4 ms) obtained by means of a timing co-simulation performed allocating all the CSP processes on a single MPU8051 instance, the DSE step is repeated several times in order to suggest architecture/mapping pairs able to provide each time even more challenging TTCs equal respectively to:

  - 0.75*WCTTC
  - 0.5*WCTTC
  - 0.35*WCTTC
  - 0.25*WCTTC
  - 0.1*WCTTC

- *Architectural Constraints* - The DSE step can use max 4 instances of MPU8051, max 2 instances of PIC24 and max 1 instance of Spartan3AN FPGA;

- *Scheduling Directives* - Processes implemented in SW and allocated on the same processor are subjected to two possible scheduling policies: a preemptive *Round-Robin* (RR) scheduling policy with 10% time overhead for context switches (in this case the results are the ones presented in [16]), and a non-preemptive *First Come First Served* (FCFS) scheduling policy with 10% time overhead for context switches.

As highlighted before, in this case study, the communication infrastructure has been fixed (i.e., BB/PUs with distributed memory and shared bus). So, the timing co-simulator directly takes into account the characterization data, related to the selected shared bus. In this case, it has been adopted an I^2C bus with a bandwidth fixed to 400 Kbps since it is a multi-master one and it is available for all the considered processors.

So, given the previous set of TTC constraints, the DSE step, while trying to satisfy each one of them, to satisfy all the architectural constraints, and to minimize the cost of the solution, provides the results shown in Table 1. Starting from WCTTC (single MPU8051), when the requirements is 0.75* WCTTC, the DSE tool proposes a dual MPU8051 architecture allocating separately processes 0-1-2 and 3-4-5-6-7. The timing co-simulation estimates a TTC equal to 2.94 ms with RR scheduling and 2.352 ms with FCFS one that well satisfy the 4.05

Table 1: DSE step results.

TTC	Proposed architecture/mapping	Simulated Time (RR [16])	Simulated Time (Buffering + FCFS)
#1 (0.75*WCTTC) (4.05 ms)	MPU8051(0): 0, 1, 2 MPU8051(1): 3, 4, 5, 6, 7 (Cost = 2)	2.94 ms	2.352 ms
#2 (0.5*WCTTC) (2.7 ms)	MPU8051(0): 0, 1 MPU8051(1): 3, 4, 5 MPU8051(2): 2, 6, 7 (Cost = 3)	2.87 ms	2.009 ms
#3 (0.35*WCTTC) (1.89 ms)	MPU8051(0): 3, 4, 5 PIC24(0): 0, 1, 2, 6, 7 (Cost = 6)	1.57 ms	1.256 ms
#4 (0.25*WCTTC) (1.35 ms)	PIC24(0): 0, 1, 2 PIC24(1): 3, 4, 5, 6, 7 (Cost = 10)	1.07 ms	0.856 ms
#5 (0.1*WCTTC) (1.35 ms)	SPARTAN3AN(0): 0, 1, 2, 3, 4, 5, 6, 7 (Cost = 50)	0.37 ms	0.37 ms

0: fir16, 1: fir16_eval, 2: for16_shift, 3: fir8, 4: fir8_eval, 5: for8_shift, 6: gcd, 7: gcd_eval ms TTC constraint. Imposing 0.5* WCTTC, the DSE step proposes a triple MPU8051 architecture with the allocations 0-1, 3-4-5 and 2-6-7 (Figure 11-left shows the corresponding PING with related individual). However, this leads to an estimated time a bit greater than 2.7 ms with RR scheduling while, with a FCFCS one, it is 2.009 ms so still able to satisfy the constraint. Imposing 0.35*WCTTC, the DSE step finds a hybrid MPU8051-PIC24 architecture with an allocation that satisfies the new constraint with both scheduling policies (Figure 11-right shows the corresponding PING with related individual). Imposing 0.25* WCTTC, a dual PIC24 seems to solve the problem (with both scheduling policies) while, with 0.1* WCTTC, imposed loads are too high for SW implementations on available processors and so a full HW architecture on FPGA is proposed. It is worth noting that the best results related to FCFS scheduling (i.e., simulated times are 20%-30% less by using the same HW configurations, due to the reduced scheduler overheads) are more than reasonable since the proposed case study does not require particular responsiveness and it is not so useful to force periodic context switches until a process is able to perform computations.

All the steps, prior to DSE one, have been executed in near 30 minutes. It is worth noting that this is a one-time effort, while the described DSE step has been executed in less than 25 minutes by exploiting a *Microsoft Windows7-based* notebook equipped with an *Intel i7* processor and 4 GB of RAM.

5.1.2. Extension #2

This paragraph will exploit the case study described above, to show the integration of PAM2 in the whole HW/SW co-design flow. For this, the *Technologies Library* considered for this case study is extended to consider following EILs:

- UART
  - BW_{max}: 115.2 Kbps
  - $N_{min} = 2$; $N_{max} = 2$ (i.e., point-to-point)

  - $CC_{max} = 2$ (i.e., bidirectional)
  - Relative cost = 1

- I^2C slow (i.e., the only EIL considered in [16])
  - BW_{max}: 400 Kbps
  - $N_{min} = 2$; $N_{max} = 8$ (i.e., an upper limit fixed by design to avoid bottlenecks)
  - $CC_{max} = 1$
  - Relative cost = 2

- I^2C fast
  - BW_{max}: 800 Kbps
  - $N_{min} = 2$; $N_{max} = 12$
  - $CC_{max} = 1$
  - Relative cost = 3

- SPI
  - BW_{max}: 2 Mbps
  - $N_{min} = 2$; $N_{max} = 2$
  - $CC_{max} = 2$
  - Relative cost = 4

Finally, given such EILs and PUs, the ECUCM is shown in Table 2.

Then, as an example of exploitation and integration of PAM2, each solution of the previous DSE - first phase (Table 1 - third column "Simulated Time with RR") have been be used as a starting point for the DSE - second phase by considering all the EILs previously described (and adding the *Architectural Constraint* of max 2 instances for each available EILs). Table 3 summarizes the results.

Table 2: ECUCM related to Extension #2.

	UART	I^2C slow	I^2C fast	SPI
MPU 8051	X	X		
PIC24	X	X	X	X
Spartan3AN	X	X		X

Old solution #1 has been changed by using an UART instead of I^2C slow: the simulated time has increased but it is still under 0.75*WCTTC and the whole cost is reduced. Old solution #2 has been replaced by an architecture that exploits both I^2C slow and UART (please, note that they are the only EILs that can be used by MPU8051). In this way, new solution #2 can satisfy the timing constraint 0.5*WCTTC. Such an architecture is shown in Figure 17. It is worth noting that, in this case, the major benefit is provided by allowing concurrency in the communications by means of two EILs, associated to the fact the 3-6 requires a quite low bandwidth (in fact, by using an UART for 0-6 and 0-2, where 0-2 requires a relevant bandwidth, leads to a discarded solution). Old solution #3 is not changed since using an UART (cheaper) it is not possible to satisfy 0.35*WCTTC. Old solution #4 has been changed by using an I^2C fast instead of slow: it is a little bit more expensive, but it provides a relevant performance improvement. Among other solutions (discarded by PAM2), it is not suggested to use the UART-based (too slow) and the SPI-based (too expensive) ones. Finally, old solution #5 is composed by a single BB (i.e., all the processes are implemented as SPPs on a single Sparta3AN), so PAM2 is not needed since there are no EILs to be considered to connect BBs. Summarizing, PAM2 has allowed to refine the solutions suggested by PAM1 by properly taking into EILs characterization. The described second-phase DSE has been executed in less than 20 minutes by exploiting a Microsoft Windows7-based notebook equipped with an Intel i7 processor and 4 GB of RAM.

Table 3: Second-phase DSE results.

Old **Best Solution**	**New Best Solution** **suggested by PAM2**	**Other Solutions** **discarded by PAM2**
#1 (0.75*WTTTC = 4.05 ms) MPU8051: 0,1,2 MPU8051: 3,4,5,6,7 (2.94 ms) Cost: $2 + 2$	UART instead of I^2C slow (3.25 ms) Cost: $2 + 1$	I^2C slow + UART (dedicated to 0-6 an 0-2) (2.92 ms) Cost: $3 + 3$
#2 (0.5*WTTTC = 2.7 ms) MPU8051: 0,1 MPU8051: 3,4,5 MPU8051: 2,6,7 (2.87 ms) Cost: $6 + 2$	I^2C slow + UART (dedicated to 3-6) (2.37 ms) Cost: $3 + 3$ 1	I^2C slow + UART (dedicated to 0-6 an 0-2) (2.92 ms) Cost: $3 + 3$
#3 (0.35*WCTTC = 1.89 ms) MPU8051: 3,4,5 PIC24: 0,1,2,6,7 (1.57 ms) Cost: $6 + 2$	The Same	UART instead of I^2C slow (2.24 ms) Cost: $6 + 1$
#4 (0.25*WCTTC = 1.35 ms) PIC24: 0,1,2 PIC24: 3,4,5,6,7 (1.07 ms) Cost: $10 + 2$	I^2C fast instead of slow (0.87 ms) Cost: $10 + 3$	UART instead of I^2C slow [1] SPI instead of I^2C slow [2] ([1] 1.11 ms, [2] 0.83 ms) [1] Cost: $10 + 1$, [2] Cost: $10 + 4$
#5 (0.1*WCTTC = 0.54 ms) Spartan3AN: 0,1,2,3,4,5,6,7 (0.37 ms) Cost: 50	None	None

0: fir16, 1: fir16_eval, 2: for16_shift, 3: fir8, 4: fir8_eval, 5: for8_shift, 6: gcd, 7: gcd_eval

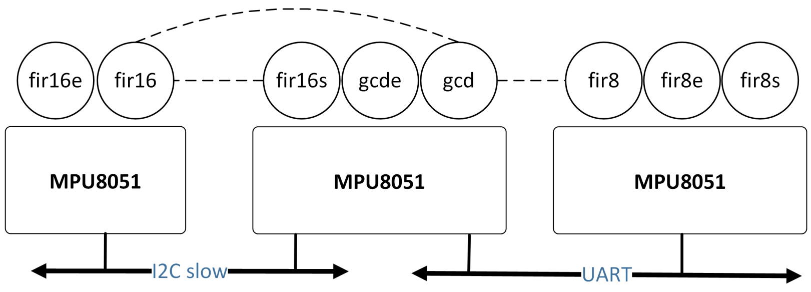


Figure 17: PAM2 New solution #2.

5.2. Use Case #2: Sobel Filtering

This use case considers an application that performs the *Sobel Filtering* [30] on a given input image. The *System Behavior Model* (SBM) is composed of 5 applicative processes (with process IDs from 2 to 6, since 0 and 1 are reserved for 620 *Stimulus* and *Display* processes, i.e. the *TestBed*) and 8 channels (with channel IDs from 1 to 8), as shown in Figure 18

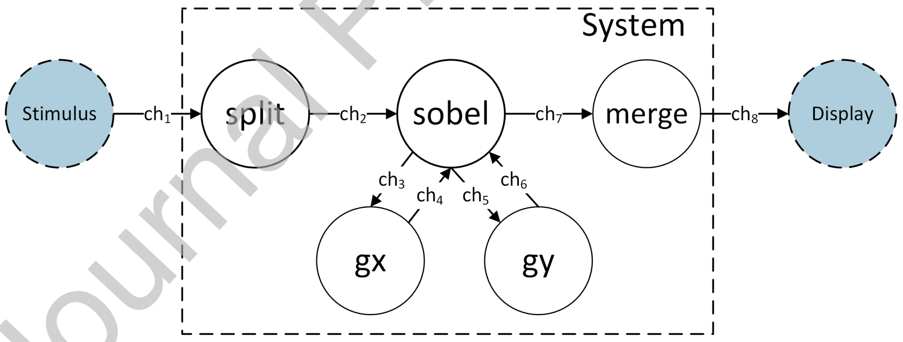


Figure 18: Sobel Filtering CSP.

The *Stimulus* process provides a 64x64 pixel input JPEG image while the other processes are described in the following list:

- *split* (#ID 2): take in input a 64x64 image and split it into several 8x8 625 simple sub-matrices;
- *sobel* (#ID 3): realize the sobel filter using two sub-processes *gx* and *gy*;

- *gx* (#ID 4) and *gy* (#id 5): manage two sub-images which at each point contain the vertical and horizontal derivative approximations, respectively;
- *merge* (#ID 6): combine the sub-images to generate the final result.

Finally, the *Display* process shows the result.

The considered *Technology Library* (TL) is the same used in the *Use Case #1 (Extension #2)*. Other than the SBM, the following *Non Functional Constraints* are considered:

- *Timing Constraints*
  - Given the estimated *Worst-Case Time-To-Completion* (WCTTC) that is obtained by means of a timing co-simulation performed allocating all the CSP processes on a single MPU8051 instance, the DSE step will be repeated several times in order to suggest architecture/mapping pairs able to provide each time even more challenging TTCs equal respectively to:
    - 0.75*WCTTC
    - 0.5*WCTTC
    - 0.35*WCTTC
    - 0.25*WCTTC
    - 0.1*WCTTC
- *Architectural Constraints*
  - The DSE step can use max 4 instances of MPU8051, max 2 instances of PIC24, max 1 instance of Spartan3AN FPGA, and max 2 instances for each available EILs.
- *Scheduling Directives*
  - Processes implemented in SW and allocated on the same BB are subjected to a non-preemptive *First Come First Served* (FCFS) scheduling policy with 10% time overhead for context switches.

The first step of the co-design flow is the *Functional Simulation*, that allows to check by simulation the correctness of the SBM. The process consists in providing an image as input and check if the obtained output is the expected one. The reference 64x64 pixel pair (i.e., the *Reference Input*) for this use case is shown in Figure 19.

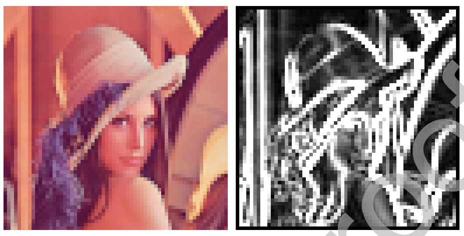

(a) Input Image. (b) Output Image.

Figure 19: Sobel Filtering result.

Then, the *Co-Analysis & Co-Estimation* step allows to gather several information needed for the DSE. In particular: the Affinity has been provided directly by the designer; *Process Concurrency* matrix, *Channel Concurrency* matrix, *Communication* matrix and *Load Estimation* are evaluated by means of HEPSIM [19]; *Timing*, *Size* and *Cost* are the same of [16] (since the TL is the same); *Bandwidth* is evaluated by dividing the *Communication* matrix for the different TTCs.

Once collected all the needed information, the DSE is performed by exploiting PAM1 and PAM2 to obtain suggestions about architecture/mapping pairs able to satisfy the different TTCs (and all the other requirements). Each pair is then simulated by HEPSIM to check if the estimated simulated time satisfies the related TTC. Table 4 reports the DSE results. From their analysis it is possible to highlight some interesting situations:

- solution #3 (0.35*TTC) is cheaper than solution #2 so it should be used also to satisfy the 0.5*TTC requirement. Situations like these can happen since the provided suggestions are related to sub-optimal solutions;

- it is always possible to satisfy the requested TTCs apart from the last one (i.e., 0.1*WCTTC), cfr. solution #5. This typically means that with the imposed constraints (or with the available TL), it is not possible to satisfy the given TTC constraint. In this case, by relaxing the *Architectural* 680 *Constraints* allowing the use of two instances of SPARTAN3AN FPGA, the DSE is able to suggest a (more expensive) solution (i.e., solution #5 bis) able to satisfy the 0.1*WCTTC constraint.

Table 4: DSE step results.

TTC	**Proposed architecture/mapping**	**Simulated Time (FCFS)**	**Cost (PUs+EILSs)**
#1 (0.75*WCTTC) (1314,51 ms)	PAM1: - MPU8051 (0): 2, 5 - MPU8051 (1): 6 - PIC24 (0): 3 - PIC24 (1): 4 PAM2: - I2C slow for all the BBs	1126,38 ms	12+2=24
#2 (0.5*WCTTC) (876,34 ms)	PAM1: - MPU8051 (0): 2 - PIC24 (0): 3, 6 - PIC24 (1): 4 - SPARTAN3AN: 5 PAM2: - I2C slow for all the BBs	632,71 ms	61+2=63
#3 (0.35*WCTTC) (613,438 ms)	PAM1: - MPU8051 (0): 2, 4 - PIC24 (0): 6 - SPARTAN3AN: 3, 5 PAM2: - I2C fast for all the BBs	515,421 ms	56+3=59
#4 (0.25*WCTTC) (438,17 ms)	PAM1: - MPU8051 (0): 2 - PIC24 (0): 3, 6 - SPARTAN3AN: 4, 5	388,992 ms	56+7=63

TTC	**Proposed architecture/mapping**	**Simulated Time (FCFS)**	**Cost (PUs+EILs)**
	PAM2: - I2C fast for all the BBs - SPI between PIC24(0) and SPARTAN3AN		
#5 (0.1*WCTTC) (175,268 ms)	PAM1: - PIC24 (0): 2 - PIC24 (0): 3, 6 - SPARTAN3AN: 4, 5 PAM2: - I2C fast for all the BBs - SPI between PIC24(0) and SPARTAN3AN	322,6586 ms	60+7=67
#5 bis (0.1*WCTTC) (175,268 ms) With relaxed *Architectural Constraints*: max 2 instances of Spartan3AN FPGA	PAM1: - SPARTAN3AN (0): 2, 3, 6 - SPARTAN3AN (1): 4, 5 PAM2: - SPI between SPARTAN3AN (0) and SPARTAN3AN (1)	21,6586 ms	100+4=104

All the steps, prior to DSE one, have been executed in near 90 minutes. It is worth noting that this is a one-time effort, while the described DSE step has been executed in near 60 minutes by exploiting a *Microsoft Windows7-based* notebook equipped with an *Intel i7* processor and 4 GB of RAM.

6. Conclusions

This work has coped with the problem of the electronic system-level HW/SW co-design of dedicated digital systems based on heterogeneous multi-processor architectures. In particular, the work has considered an automatic system-level DSE approach in order to present a prototype "*SystemC*-based ESL DSE Environment for Dedicated Heterogeneous Multi-Processor Systems". The presented tools are still in a prototypal status and several improvements are still possible but the preliminary experimental results are encouraging and justify further research efforts in this direction. In particular, with respect to the whole co-design methodology, it will be of very critical importance a validation based on real-world data coming from lower levels of abstraction. Then, it will be important to consider more non-functional requirements (i.e., power/energy, real-time constraints, mixed-criticality, etc.) and the run-time adaptivity that can be required to the final system, for example in contexts related to *Cyber-Physical Systems* where the working environment can frequently change after system deployment. Finally, it will be interesting also to exploit existing works to introduce in the presented *SystemC*-based co-design environment also the possibility of managing architectures based on *homogeneous/heterogeneous multi-core processors* with shared memory among the cores [31].

Acknowledgement

This work has been partially supported by the ECSEL RIA 2016 MegaM@Rt2 and AQUAS projects.

References

[1] Xilinx Zynq7000, `http://www.xilinx.com`.

[2] OMAP Platform,, `www.omap.com`.

[3] SH Mobile Series, `http://www.renesas.com`.

[4] Intel Stratix 10 SoC FPGA, `https://www.intel.com`.

[5] F. Vahid, T. Givargis, Embedded system design: A Unified Hardware/Software Approach, Department of Computer Science and Engineering University of California, 1999 (1999).

[6] C. Brandolese, W. Fornaciari, L. Pomante, F. Salice, D. Sciuto, Affinity-driven system design exploration for heterogeneous multiprocessor soc,

IEEE Transactions on Computers 55 (5) (2006) 508–519 (May 2006). doi:10.1109/TC.2006.66.

[7] T. Streichert, M. Glaß, C. Haubelt, J. Teich, Design space exploration of reliable networked embedded systems, Journal of Systems Architecture 53 (10) (2007) 751 – 763, embedded Computer Systems: Architectures, Modeling, and Simulation (2007). doi:https://doi.org/10.1016/j.sysarc.2007.01.005.

[8] G. Ascia, V. Catania, A. G. D. Nuovo, M. Palesi, D. Patti, Efficient design space exploration for application specific systems-on-a-chip, Journal of Systems Architecture 53 (10) (2007) 733 – 750, embedded Computer Systems: Architectures, Modeling, and Simulation (2007).

[9] M. Holzer, B. Knerr, M. Rupp, Design space exploration with evolutionary multi-objective optimisation, in: 2007 International Symposium on Industrial Embedded Systems, 2007, pp. 126–133 (July 2007). doi:10.1109/SIES.2007.4297326.

[10] G. Palermo, C. Silvano, V. Zaccaria, An efficient design space exploration methodology for on-chip multiprocessors subject to application-specific constraints, in: 2008 Symposium on Application Specific Processors, 2008, pp. 75–82 (June 2008). doi:10.1109/SASP.2008.4570789.

[11] C. Haubelt, T. Schlichter, J. Keinert, M. Meredith, Systemcodesigner: Automatic design space exploration and rapid prototyping from behavioral models, in: 2008 45th ACM/IEEE Design Automation Conference, 2008, pp. 580–585 (June 2008). doi:10.1145/1391469.1391616.

[12] I. D. L. Anderson, M. A. S. Khalid, Sc build: a computer-aided design tool for design space exploration of embedded central processing unit cores for field-programmable gate arrays, IET Computers Digital Techniques 3 (1) (2009) 24–32 (January 2009). doi:10.1049/iet-cdt:20070120.

[13] Z. J. Jia, T. Bautista, A. Núñez, A. D. Pimentel, M. Thompson, A system-level infrastructure for multidimensional mp-soc design space co-exploration, ACM Trans. Embed. Comput. Syst. 13 (1s) (2013) 27:1–27:26 (Dec. 2013). doi:10.1145/2536747.2536749.

[14] K. Keutzer, A. R. Newton, J. M. Rabaey, A. Sangiovanni-Vincentelli, System-level design: orthogonalization of concerns and platform-based design, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 19 (12) (2000) 1523–1543 (Dec 2000). doi:10.1109/43.898830.

[15] Intel CoFluent Studio - Model and Simulate System Behavior, https://www.intel.it/content/www/it/it/cofluent/cofluent-studio.html (2018 (accessed: 21.05.2018)).

[16] L. Pomante, P. Serri, Systemc-based hw/sw co-design of heterogeneous multiprocessor dedicated systems, International Journal of Information Systems 1 (July 2014).

[17] C. A. R. Hoare, Communicating Sequential Processes, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1985 (1985).

[18] Communicating Sequential Processes, C. A. R. Hoare, May 18, 2015, www.usingcsp.com.

[19] D. Ciambrone, V. Muttillo, L. Pomante, G. Valente, Hepsim: An esl hw/sw co-simulator/analysis tool for heterogeneous parallel embedded systems, in: 2018 7th Mediterranean Conference on Embedded Computing (MECO), 2018, pp. 1–6, best Paper Award (June 2018). doi:10.1109/MECO.2018.8406078.

[20] A. Allara, C. Brandolese, W. Fornaciari, F. Salice, D. Sciuto, System-level performance estimation strategy for sw and hw, in: Proceedings International Conference on Computer Design. VLSI in Computers and

Processors (Cat. No.98CB36273), 1998, pp. 48–53 (Oct 1998). doi:10.1109/ICCD.1998.727022.

[21] C. Brandolese, W. Fornaciari, F. Salice, An area estimation methodology for fpga based designs at systemc-level, in: Proceedings. 41st Design Automation Conference, 2004., 2004, pp. 129–132 (July 2004). doi:10.1145/996566.996606.

[22] C. Brandolese, Source-level estimation of energy consumption and execution time of embedded software, in: 2008 11th EUROMICRO Conference on Digital System Design Architectures, Methods and Tools, 2008, pp. 115–123 (Sep. 2008). doi:10.1109/DSD.2008.43.

[23] L. Pomante, System-level design space exploration for dedicated heterogeneous multi-processor systems, in: ASAP 2011 - 22nd IEEE International Conference on Application-specific Systems, Architectures and Processors, 2011, pp. 79–86 (Sep. 2011). doi:10.1109/ASAP.2011.6043239.

[24] J. Teich, T. Blickle, L. Thiele, An evolutionary approach to system-level synthesis, in: Proceedings of 5th International Workshop on Hardware/Software Co Design. Codes/CASHE '97, 1997, pp. 167–171 (March 1997). doi:10.1109/HSC.1997.584597.

[25] SystemC, http://www.accellera.org (2018 (accessed: 21.05.2018)).

[26] Occam, http://www.wotug.org/occam/.

[27] Barrett, G.: occam 3 Reference Manual. Technical report, Inmos Limited (1992), http://wotug.ukc.ac.uk/parallel/occam/documentation/.

[28] SystemC, http://www.accellera.org.

[29] M. Mitchell, An Introduction to Genetic Algorithms, MIT Press, Cambridge, MA, USA, 1998 (1998).

[30] G. N. Chaple, R. D. Daruwala, M. S. Gofane, Comparisions of robert, prewitt, sobel operator based edge detection methods for real time uses

on fpga, in: 2015 International Conference on Technologies for Sustainable Development (ICTSD), 2015, pp. 1–4 (Feb 2015). doi:10.1109/ICTSD.2015.7095920.

[31] L. Pomante, Hw/sw co-design of dedicated heterogeneous parallel systems: an extended design space exploration approach, IET Computers Digital Techniques 7 (6) (2013) 246–254 (November 2013). doi:10.1049/iet-cdt.2013.0026.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

nplementing mixed-critical applications
xt generation multicore aerospace platfo
Fabio Federici, Danilo A
cci, Luigi Pomante

Luigi Pomante has received the "Laurea" (i.e., BSc+MSc) Degree in Computer Science Engineering from "Politecnico di Milano" (Italy) in 1998, the 2nd Level University Master Degree in Information Technology from CEFRIEL (a Center of Excellence of "Politecnico di Milano") in 1999, and the Ph.D. Degree in Computer Science Engineering from "Politecnico di Milano" in 2002. He had been a Researcher at CEFRIEL from 1999 to 2005 and, in the same period, he had been also a Temporary Professor at "Politecnico di Milano". From 2006, he is an Academic Researcher at Center of Excellence DEWS ("Universit degli Studi dellAquila", Italy). From 2008 he is also Assistant Professor at "Universit degli Studi dellAquila" (he is responsible of the "Embedded Systems" course). His activities focus mainly on Electronic Design Automation (in particular Electronic System-Level HW/SW Co-Design) and Networked Embedded Systems (in particular Wireless Sensor Networks). In such a context, he has been author (or co-author) of more than 100 articles published on international and national conference proceedings, journals, and book chapters. He has been also reviewer and member of several TPCs related to his research topics. From 2010, he has been in charge of scientific and/or technical issues on behalf of DEWS in more than 10 funded European and national research projects.