SLS team (System Level Synthesis)

Themes

HW/SW Architectures and CAD software for multiprocessor systems on chip
Specification, modeling, simulation and verification of embedded systems on chip
Reconfigurable and prototyping for cloud and artificial intelligence

Expertise

Scientific
Integration, optimization, and verification of multiprocessor hardware/software systems

Fields of expertise
Multiprocessor architecture, Memory subsystems, Fast simulation of digital systems, Methods and tools for system level synthesis, Formal and semi-formal verification, Reconfigurable and *ad-hoc* neuronal architectures

Know-how
Design of integrated circuits and systems, FPGA, Operating systems, Software/hardware integration, simulation and verification

Industrial transfer
Patent on multiprocessor synchronization (n°1858803, sept. 2018), 6 CIFRE theses over the 4 last years (14-18). Merging of the Antfield SLS spinoff into the GreenSocs company.

Research keywords

Digital system architecture: network-on-chip, memory hierarchies (prefetching and coherence), reconfigurable computing, artificial neural network architectures, low-level software and drivers.
System simulation: dynamic binary translation, native and compiled simulation.
System-Level and software verification.

Contact

Frédéric PÉTROT
GRENOBLE INP Professor
(+33) 4 76 57 48 70
frederic.petrot@univ-grenoble-alpes.fr

HW/SW architecture and CAD software for MPSoC

Keywords: Embedded software, device driver, MPSoC, Network-on-Chip, task migration

Members: T. Baumela, J. Dumas, M. France-Pillois, O. Muller, F. Pétrot, F. Rousseau

Cooperations: CEA-LETI, LIG

Contracts: CIFRE, CAPACITES (Investissements d'avenir)

Designing a Multi-Processor System-on-Chip (MPSoC) implies a lot of knowledge specifically when the software interacts with the hardware in terms of architecture and tools. This domain is usually called hardware/software (HW/SW) interfaces.

The long-term expertise acquired in the last 10-15 years in this HW/SW interface domain leads to deep scientific and technological breakthroughs: synchronization in MPSoC, driverless operating systems, memory hierarchy organization and design, ...

1. Optimization of Synchronization barriers in Multiprocessor architecture

Providing high-performance synchronization mechanisms is a key issue to leverage hardware parallelism offered by MPSoCs. The study focuses on the synchronization barrier mechanism and the impact of hardware contention for shared memory clustered MPSoC.

Based on a real implementation of the TSAR platform on Veloce2 Quattro emulator, we have developed a non-intrusive tool-chain in order to observe certain internal signals. We implemented a spy module in the platform to extract at runtime, by this side channel, useful signals like processors program counters and registers [3]. Hence software execution is not impacted by the signal extraction, which occurs only on the hardware side. Extracted signals are dumped into files. The content of these files is then processed by different tools that analyze signal values and provide relevant information of time spent in the studied mechanisms. Leveraging accurate non distorted timing measurements, this tool-chain generates time annotated function calls stack and cross-processors timing analysis exposing synchronization mechanisms slowdowns in realistic working condition.

With this methodology, we have identified hardware modules restrictions and Linux kernel sub-optimal services both in active and passive wait processes. We show how the introduction of delays in the thread awakening process improves the overall synchronization mechanism. We have provided a combined HW/SW optimization. For the passive wait, our proposal provides a 60 % gain for 64 threads running on a 64-core architecture, and about 85 % gain for active wait on 16 and 32-core architecture [1].

Another work concerns an efficient hardware support for synchronization locks. We have confirmed a fundamental hypothesis for the optimization of the lock mechanism: although during the run time a lock may be used by various cores belonging to different clusters, it is often reused by the last core which has released it. Based on this observation, we propose an innovative decentralized solution to manage dynamic re-homing of locks in memory close to the last access-granted core, thus reducing overall access latency and network traffic in case of reuse of the lock by the same cluster [2].

2. Redefinition of the HW/SW interface

Given their unbeatable energy efficiency, Systems-on-a-Chip (SoC) are the solution to integrate computing in devices. However, all devices do not need the same hardware, hence the impressive number of different SoC produced by the system integrators. The Table below gives the approximate number of different SoC produced in volume per integrator:

Time frame	Nb of SoCs	Device	Device Maker
2012-2018	22	Kirin	Huawei
2007-2018	29	APLx	Apple
2012-2016	33	Atom	Intel
2000-2018	46	Exynos	Samsung
2003-2019	120	MTX	Mediatek
2007-2018	136	Sndg	Qualcomm

From a hardware point of view, the integration techniques have evolved drastically in the last decade. However, the definition of the frontier between peripheral devices and the kernel of the operating system, structuring how input/output operations occur at runtime, has remained mostly unchanged since the very first days of computers, even though the number of IPs have risen tremendously, has had the number of registers per IP.

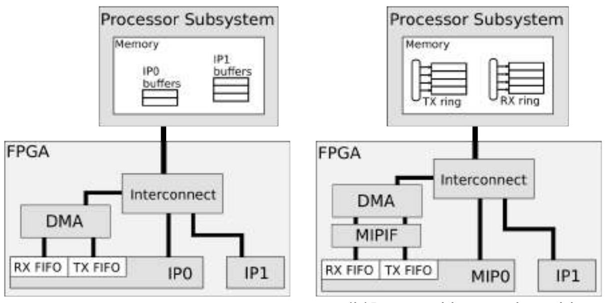
We believe that there is a huge value in revisiting the hardware/software frontier, simply because complexity is getting out of hand: tens to hundred of IPs per chip, 10 to hundreds of registers per IP.

Research reports that more than 70 % of Linux code is due to drivers, and 70 % of the bugs are due to drivers too. Therefore, we propose a simple idea: have each driver run on the device it controls, and exchanging information not through a complex set of ordered accesses to registers, but through simple sequential messages, typically as USB does on serial links for USB slaves.
The immediate benefits are increased performance and lower power consumption by better exploiting the inherent parallelism available in the SoC.
Additionally, devices will host and run their own drivers, which is beneficial to device makers: they can deliver an out-of-the-box solution combining the software driver and the hardware device.

In particular, each driver is written only once and can be updated independently of the official OS releases, thus increasing robustness and shortening time-to-market.

As a side effect, we can expect that kernels will be greatly simplified and more robust, as device specific drivers are no longer part of their codebase, ultimately greatly simplifying and improving virtualization techniques.

As a first step, we have studied this year the applicability of this proposal to FPGA devices [4], as illustrated on Figure 1.



(a) Typical integration

(b)Proposed integration with MIPIF and MIP

Figure 1: Proposed integration approach

As the approach is message-based while the IPs are not, specific hardware is required to demonstrate our idea.

Given the large amount of IP available in FPGA development platforms and the need for supporting them in operating systems, we have designed and developed a hardware wrapper that translate messages to the IP into a set of register access to evaluate the hardware and performances costs. As a preliminary result, both area and performances are minimally impacted. A fact of life is that many complex IPs today embed MCU or even full fledge processors. As a next step, we plan to use these processors in order to run IP local drivers interfaces with the host processor, to limit the number of host drivers to a minimum.

3. Cache simulation and sharing set representation in shared memory multiprocessors

Cache coherence has been a long lasting subject in computer architecture. With the emergence of VLSI integration of massively parallel machines, this subject is regaining importance, and this for two reasons. First, the available throughput on network-on-chip is much higher than the one available on circuits that were previously implemented on different boards or on multi-chip modules. And second, the scalability issues are more stringent on-chip because power consumption and area are of high importance for cost sensitive applications.

In cooperation with CEA-LIST, we worked on a new sharing list management strategy for cache. The sharing list is a set of metadata that each L2/L3/LLC cache maintains to know which lower level caches have a copy of a given memory line. As the number of processor grows, the size of these meta-data becomes just too large. To limit this size, many proposals have been developed. However, we believe it is possible to be more scarce on data while being at least as efficient in term of performances. For that, we developed a new representation that is constituted by a rectangle, whose size and shape can vary dynamically, and of a list stored in a shared heap. The maximal size of the rectangle is determined experimentally by analyzing the level of sharing of classical shared memory benchmarks. It appears that, consistently for all benchmarks, around 90 % of the cache blocks are shared by at most 8 processors, as can be seen on Figure 2.

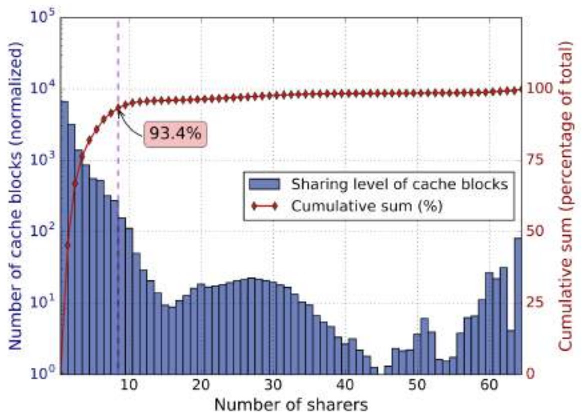


Figure 2: Experimental results

So we arbitrarily fix this maximal rectangle size to 8.

The idea is then to cover as much sharers as possible with the rectangle, by dynamically adjusting its position and shape (here the shape also determines the size). Should any cache stay uncovered, then we store these outliers in the list. This list has a maximum size so as to avoid preventing other blocks to store their own outliers.

The appropriate size of the list is also determined experimentally, and it appears that a value of 4 is a good performance vs access time trade-off .If the list overflows, in last resort, we use broadcast to send the protocol messages to all cores, making sure no copies stay uninformed of state changes [5].

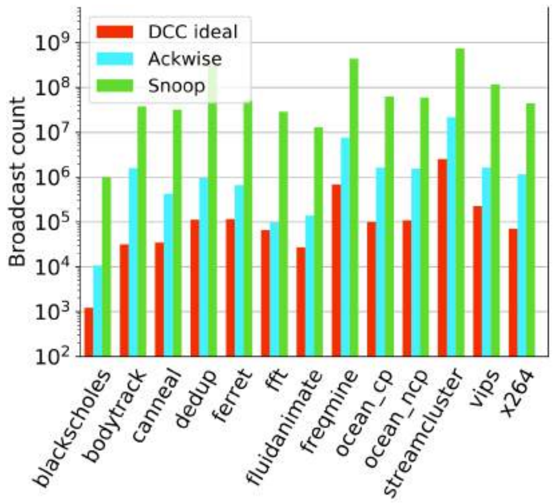


Figure 3 : Comparison of the number of broadcasts for parallel benchmarks

Experiment on architectures with up to 64 cores show that the approach is competitive in terms of performance with the state of the art (see Figure 3), and that is scales better in terms of resource requirements, as shown in Table 1.

Sharing set management strategy	Directory entry size			
	Analytical formula	Cores		
		64	128	1024
Bit vector	N	64	128	1024
Ackwise	$\log_2(N) \times k + 1$	31	36	51
Linked List	$\log_2(N) + \log_2(H)$	13	14	17
DCC	$\log_2(N) + C + l + \log_2(H) + 1$	33	34	37

Table 1: Resources

4. References

[1] M. France-Pillois, J. Martin, F. Rousseau, "Optimization of the GNU OpenMP Synchronization Barrier" in MPSoC. 31st Int. Conference on Architecture of Computing Systems (ARCS), April 2018, Springer

[2] M. France-Pillois, J. Martin, F. Rousseau, "Linux synchronization barrier on MPSoC: Hardware/software accurate study and optimization", 29th Int'l Conference on Application-Specific Systems, Architectures and Processors, ASAP 2018: 1-4, Milano, Italy July 2018

[3] M. France-Pillois, J. Martin, F. Rousseau, "Accurate MPSoC Prototyping Platform and Methodology for the Studying of the Linux Synchronization Barrier Slowdown Issues", Int'l Symposium on Rapid System Prototyping (IEEE RSP 2018), pp. 56-62, Oct. 2018, Torino, Italy

[4] T. Baumela, O. Gruber, O. Muller, F. Pétrot. "Message-Oriented Devices on FPGAs", 2018 International Symposium on Rapid System Prototyping (RSP), pp. 8-14, Oct. 2018, Torino, Italy

[5] J. Dumas, E. Guthmuller, F. Pétrot, Dynamic Coherent Cluster: "A Scalable Sharing Set Management Approach". 29th Int'l Conference on Application-Specific Systems, Architectures and Processors, pp. 1-8, Milano, Italy July 2018.

Reconfigurable and prototyping for cloud and artificial intelligence

Keywords: Dynamic reconfiguration, FPGA virtualization, accelerator architectures

Members: L. Andrade, J. Bruant, G. Christodoulis, B. Ferres, O. Muller, F. Pétrot, F. Rousseau, A. Wicaksana

Cooperations: LIG, GIPSA, STEI-ITB, OVH

Contracts: HEAVEN (Persyval), CIFRE

Reconfigurable architectures, such as FPGA, are commonly used in research activities for a wide range of applications. During the report period, the SLS team has mainly conducted its research on the virtualization of reconfigurable architecture and on the use of FPGA prototype to demonstrate innovative application-specific accelerators. Both research activities are presented in the following sections.

1. Portable infrastructure for heterogeneous reconfigurable devices in a cloud-FPGA environment

Field-Programmable Gate Arrays (FPGAs) have been gaining popularity as hardware accelerators in heterogeneous architectures thanks to their high performance and low energy consumption. This argument has been supported by the recent integration of FPGA devices in cloud services and data centers. The potential offered by the reconfigurable architectures can still be optimized by treating FPGAs as virtualizable resources and offering them multitasking capability. The solution to preempt a hardware task on an FPGA with the objective of context switching it has been in research for many years. In previous works, we proposed a strategy to extract the context of a running task from the FPGA to provide the possibility of its resumption at a later time.

An extension of this work to enlarge its usefulness consists in the communication consistency management. This communication management is necessary to ensure the consistency in the communication of a task with context switch capability in a reconfigurable system. Otherwise, a hardware context switch can only be allowed under restrictive constraints which may lead to a considerable penalty in performance; context switching a task is possible after the communication flows finish and the input/output data have been consumed. Furthermore, certain techniques demand homogeneity in the platform for a hardware context switch can take place.

We have proposed a mechanism which preserves the communication consistency during a hardware context switch in a reconfigurable architecture, as illustrated in Figure 1 [1]. The input/output communication data are managed together with the task context to ensure their integrity. The overall management of the hardware task context and communication data follows a dedicated protocol developed for heterogeneous reconfigurable architectures. This protocol thus allows a hardware context switch to take place while the task still has ongoing communication flows on Reconfigurable System-on-Chips (RSoCs). Our experiments show that the overhead due to managing the communication data becomes negligible since our mechanism provides the necessary high responsiveness for preemptive scheduling, besides the consistency in communication. Finally, our solution has been integrated in a task migration prototyping and in a hypervisor-based system.

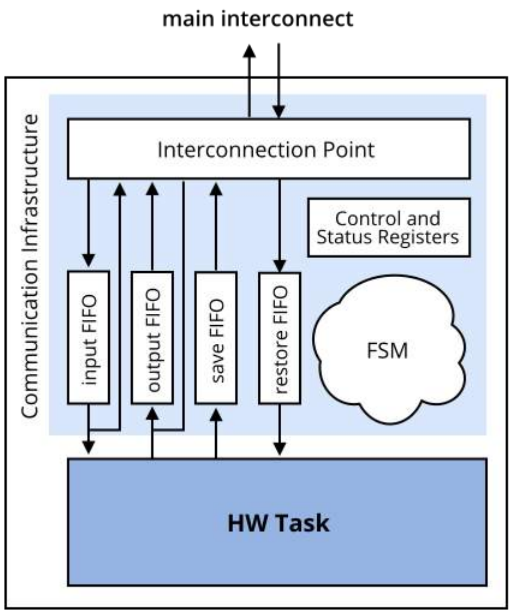


Figure 1: Proposed communication infrastructure

2. FPGA integration in heterogeneous HPC architectures

Heterogeneity in HPC nodes appears as a promising solution to improve the execution of a wide range of scientific applications, regarding both performance and energy consumption. Unlike CPUs and GPUs, FPGAs can be configured to fit the application needs, making them an appealing target to extend traditional heterogeneous HPC architectures. However, exploiting them requires an in-depth knowledge of low-level hardware and high expertise on vendor-provided tools, which should not be the primary concern of HPC application programmers. We proposed a framework, which require the minimum knowledge of the underlying architecture, as well as fewer changes to the existing code. To fulfill these requirements, we extend the StarPU task programming library that initially targets heterogeneous architectures to support FPGA. We use Vivado HLS, a high-level synthesis tool to deliver efficient hardware implementations of the tasks from high-level languages like C/C++. A proof-of-concept implementation and preliminary results have been revealed in [2].

3. High-Throughput Inference with Ternary Neural Networks

Deep neural networks are requiring a lot of computation and storage resources. To alleviate this, we are advocating the use of ternary weights and low bit representation of data. A cooperation with LIG has led to the definition of a new training approach in which all weights are ternary and all activations are ternary, the input data being 8 bit wide.

This training approach, based on a new usage of the teacher/student strategy, leads to relatively small degradation as compared to the state of the art on academic benchmarks (we tested image classification for CIFAR10/100 and GTRSB), but with a need for resources order of magnitude smaller than usual floating point approaches.

We developed a Convolutional Neural Network (CNN) in which weights and activations have been quantified on 2-bits to represent the values {-1,0,1}. This architecture, called Convolutional Ternary Neural Network [3, 4], has been designed to be implemented in a Xilinx Virtex-7 FPGA VC709 prototyping board. It integrates 14M of parameters into the BRAM and LUTRAM of FPGA board and it is able to obtain a performance of 18TOP/s. In addition, the architecture supports different parallelism levels with small memory cut, leading to a very high-throughput in terms of frame per second, and with low latency. The goal of this architecture is to have all weights on-chip, so that power efficiency is excellent, since there is no need to access an external DRAM. Also, given the ternary nature of the weights and the usage of creative low level optimizations, the network can be fully unrolled on the FPGA, leading to very high-throughput (up to 60k frame per second) at low power (11 watts measured on the board, including PCIe IP). Figure 2 gives an overview of the design points in terms of power consumption and throughput for a given TNN architecture and level of parallelism.

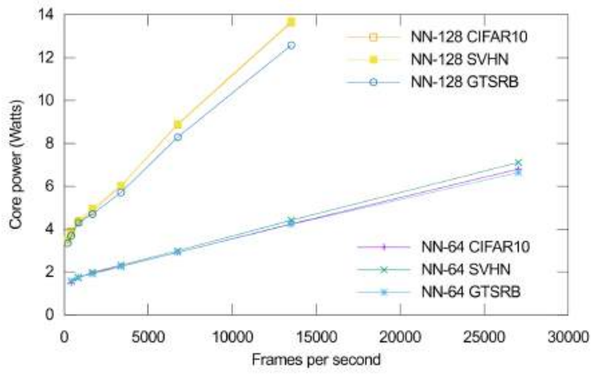


Figure 2: Power consumption as a function of throughput

The application of our quantification approach demonstrated the possibility of improving the performance of CNN using a reduced number of bits. So far, the quantification of weights and activations, the design of a dedicated architecture and its implementation in FPGA were demonstrated on an architecture inspired from a VGG model. The next phase of this work will focus on the application of the ternary approach in dedicated architectures inspired from other traditional deep CNN configurations as GoogLeNet ou ResNet, and other small configurations as MobileNet or SqueezeNet.

4. References

[1] A. Wicaksana, "Portable infrastructure for heterogeneous reconfigurable devices in a cloud-FPGA environment", PhD thesis, Univ. Grenoble Alpes. October 2018

[2] G. Christodoulis, M. Selva, F. Broquedis, F. Desprez, O. Muller, "An FPGA target for the StarPU heterogeneous runtime system", 13th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (RECOSOC'2018), pp. 1-8, Lille, France, 2018

[3] A. Prost-Boucle, A. Bourge, F. Pétrot, “High-Efficiency Convolutional Ternary Neural Networks with Custom Adder Trees and Weight Compression”, ACM Transactions on Reconfigurable Technology and Systems (TRETS'2018) 11 (3), 15

[4] L. Andrade, A. Prost-Boucle, F. Pétrot, “Overview of the state of the art in embedded machine learning”, Design, Automation and Test in Europe (DATE'2018), 1033-1038

Specification, modeling, simulation and verification of embedded systems on chip

Keywords: Simulation, debug, profiling, verification

Members: L.Andrade, L.Bonicel, M.Chabot, A.Faravelon, B.J.Fernandez, F.Pétrot, L.Pierre, F.Rousseau

Cooperations: LIP6, Kalray, Schneider Electric, Dolphin Integration

Contracts: CAPACITES (Investissements d'avenir), SPICA (FUI), CIFRE

Simulation and verification have been long lasting research topics of the SLS team. During this past year, we classically still focused on simulation speed, trace analysis, etc, but also bootstrapped new activities around cyber-physical systems. This includes the modeling of requirements with the goal of high-level testing of ESL systems, modeling approach and simulation of analog and mixed signal systems, and verification of cyber-physical systems.

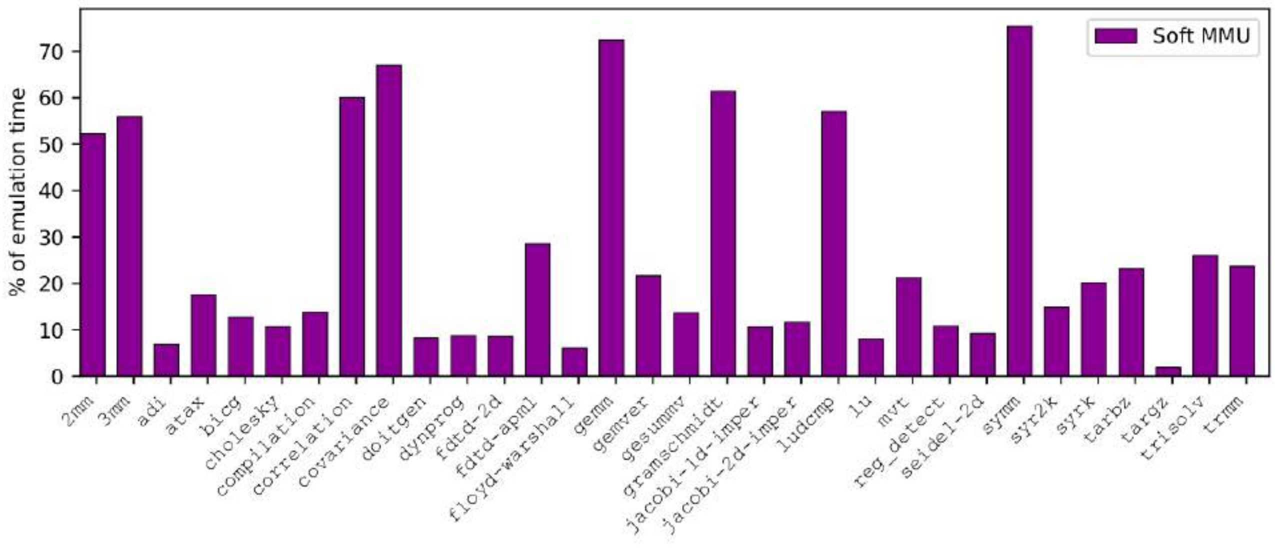
1. Fast simulation strategies

Simulation of large scale integrated multiprocessor platforms is a long lasting theme of the SLS group. This work is of primary necessity as the number of processors in integrated circuits is rising, and therefore the simulation times are increasing constantly. Current manycore systems contains tenths or even hundreds of processors, usually with a VLIW architectures (e.g. Tilera TileGx72 or Kalray MPPA256) for a high performance per watt ration. As a result, the execution of the software on Instruction Set Simulators during simulation (making the processor the ultimate hardware/ software interface) is not viable anymore.

During this year, we have kept on working on dynamic binary translation, with the goal of accelerating the simulation of the load/store accesses, and more specifically to speed-up virtual to physical address translation that occurs on each and every read or write in the target code. This focus stems from the fact that the simulation of the logical to physical address translation takes a non-negligible part of the simulation time, as can be seen on the figure below. This is due to two factors: a) load and store are done using *helpers*, *i.e.* functions that are called to perform a complex action instead of being inlined, and b) the actual translation can be very complex. It has a fast path that searches within a hash table caching the most recent translations, and a slow path traversing the target page table structure to retrieve the translation.

The principle of our idea is to translate the target load or store directly into a host load or store (technically, one more load is necessary in both cases) and use the possibility given by Linux to integrate, though a kernel module, our own way to manage a set of pages of a given process. Thanks to that, we can simply map the physical target address space (obtained by a simple memory allocation to create the memory model in the simulation environment) into the host virtual memory. Using a simple offset, all memory access can then be done through an indirect + register type memory access of the host, at the cost of a page fault on the first access. Many details have however to be taken care of, because the mappings change over time due to the target operating system, which incurs also a non-negligible overhead.

We have also continued our work on native simulation making use of the hardware assisted virtualization support available on modern processors. We were the first to point out in 2009



that the control flow graph (CFG) of code compiled on the target processor was not isomorphic to the CFG of the same code compiled on the host processor, making therefore timing estimates during native simulation at best a wild guess. Thanks to the fact that most compilers use an intermediate representation on which most of the non-machine dependent optimizations are done, we defined an approach which does an approximate mapping between a fully optimized target CFG and the partially optimized host code in intermediate representation. To that aim, we had to study aggressive compiler optimizations such as loop unrolling and code motion. We defined an algorithm to map basic blocks of the intermediate representation onto basic blocks of the optimized binary, and back-propagate the relevant information into the intermediate representation basic blocks, or even restructure the intermediate representation control flow graph so that the execution of the C generated view of the intermediate representation is fully faithful to the execution of the binary in terms of traversed basic blocks. This focus on loop is particularly important as any error in a loop is replicated by the number of iteration of the loop [1].

2. Speed up heterogeneous simulation

Exploring possibilities for fast modeling and simulation of heterogeneous (software and digital/analog hardware) systems is a recent interest of the SLS group. During the last year, we worked on the feasibility of combining different simulation techniques (QEMU + SystemC AMS) in order to accelerate the joint development of such heterogeneous systems. Based on the experience of the SLS members about simulation techniques, and modeling/simulation of digital and analog/mixed-signal systems at high level of abstraction, we developed a heterogeneous virtual platform that allowed us to demonstrated our approach. As shown in Figure 1, using the IEEE 1666.1-2016 SystemC AMS Standard, we implemented the behavior of an analog component, in our case a DC motor, by means of a scaled continuous-time linear transfer function G(s) provided by the Timed Data Flow Model of Computation (MoC).

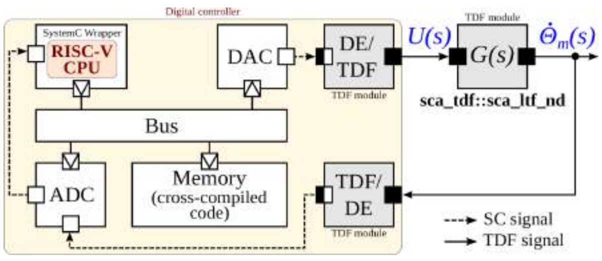


Figure 1: SystemC/TLM/AMS Heterogeneous Virtual Platform

This analog component was interconnected, by means of user-defined converter modules, with a digital SystemC/TLM platform implemented using the IEEE 1666-2011 SystemC standard. The TLM platform, which implements a digital version of a motor speed PID controller, is composed of a RISC-V CPU model, a memory, an ADC, a DAC and a memory-mapped bus. The ADC samples the motor speed signal (Θm') and sends an interruption to the CPU. The CPU handles the interruption by executing the binary cross-compiled code stored in the memory: read the sampled data from the ADC, execute a discretized controller equation, write back the control signal value to the DAC and suspend until the next interruption arrives. The code of the controller is written in C and cross-compiled to target a RISC-V 32 bits ISA (RV32IMF). The CPU model, provided by Rabbits, was initially implemented as a SystemC module, which instantiates a RISC-V QEMU virtual machine and coordinates its execution as a SystemC process in a way that is transparent to the user. In order to evaluate the simulation performance, we implemented another CPU version, a RISC-V ISS wrapped in a SystemC module exposing a TLM initiator socket and an interruption port, which transforms the load and store requests into TLM transactions. We executed the heterogeneous virtual platform using the two CPU versions and we demonstrated that the QEMU RISC-V model speeds up simulation by almost a threefold factor relative to the RISC-V ISS. In addition, the control system time response was validated using as reference three analog versions of the PID controller, implemented using the TDF, LSF, and ELN SystemC AMS MoCs. Simulation results showed no significant difference in accuracy. This work demonstrates that QEMU can be coupled with SystemC and SystemC AMS models to enable early and fast development of software and the heterogeneous design space exploration.

3. Using simulation for software performance improvement

The goal of this research, done in cooperation with STMicroelectronics, is to help optimize software performance during hardware/software integration, by using simple metrics such as processor stalls and transactions latencies. To than aim, pretty time accurate simulation models are required since the effects of pipeline stalls, cache misses, memory contention and so on are very dependent on the moment at which they occur. Although very time consuming, we choose to use a cycle-accurate/bit accurate model of the processor and its environment, obtained from the VHDL description thanks to a commercial tool. Using this model, stalls due to the non-availability of data into registers are identified, allowing to compute a prefetch distance at which a prefetch instruction could be added to limit stall latency. Note that we consider here superscalar processor in which the instruction that stall is not the load instruction, but the instruction that consumes the register that is the target of that load. Trace analysis is necessary to identify which instructions are responsible for the stalls.

4. Testing framework for cyber-physical system design

This project, realized in cooperation with Schneider Electric, advocates a *requirement driven testing approach* to improve the design of cyber-physical systems. It enables to specify test infrastructures dedicated to requirements (distinctive of IEC or other standards, or user requirements) and to automate the generation of the corresponding executable tests.

We have specified and implemented a tool called ARES (Automatic geneRation of Executable tests from SysML). It *automatically transforms models of test scenarios* originating from requirements *into automated tests* [2], [3]. Test scenarios are modeled as SysML Sequence Diagrams that specify ordered sequences of interactions between "test emulators" and the system under test. From such a behavioural description of the test scenario, ARES automatically generates an executable scenario used to drive the simulation or the verification on a (TestStand) testbench, thus guaranteeing the application of the same scenario in both contexts. The method consists in generating a finite state machine (FSM) able to communicate with the system through specific ports.

The overall method (requirement driven test specification + transformation into executable tests) has been applied to several use cases of Schneider Electric, for example a model of breakers network (busbars) with its protection algorithm (Figure 2).

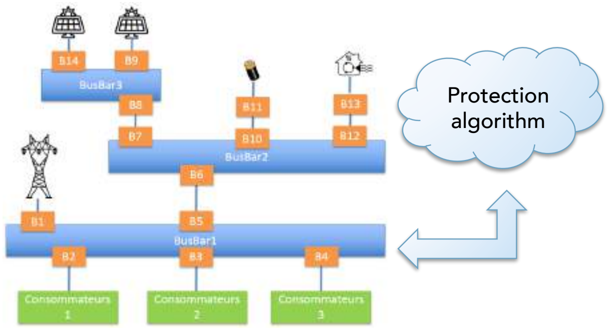


Figure 2: Use case - Protection for a network of breakers

5. Assertion-Based Verification for HW/SW systems

Verifying the correctness and the reliability of C or C++ embedded software is a crucial issue. To that goal, *runtime Assertion-Based Verification* (ABV) can be used to check whether applications obey temporal properties ("assertions") that capture the design intent. Such assertions can also be used for the online monitoring of fault-tolerance, security, or performance properties.

With our approach, assertions are formalized in the IEEE standard language PSL, and our tool OSIRIS automatically transforms them into C++ assertion checkers. Associated with the embedded software, these checkers can be triggered during execution to report property satisfaction or violation.

A major challenge is the *instrumentation of the embedded application* so that the assertion checkers can be triggered upon specific events during execution. The events of interest are (1) global/local variable readings and updatings, and (2) function calls and returns. OSIRIS implements two solutions to automatically instrument object files of embedded programs in order to enable the event-driven activation of the temporal assertion checkers [4]. One of them simply transforms the original binary file as pictured on Figure 3: each instruction that corresponds to an event of interest is replaced by a branch instruction to the function that triggers the checker evaluation.

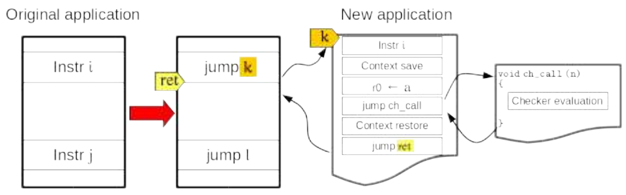


Figure 3: Binary instrumentation with PSL checkers

Experiments have been performed for several use cases, among them a path following lateral controller for an autonomous car, and an audio decoding platform. The CPU time overhead for the instrumentation mechanism itself is negligible, and instrumentation with the assertion checkers ranges from 0.4 % to 25 % CPU time overhead, for complex properties.

6. References

[1] O. Matoussi, F. Pétrot, “A mapping approach between IR and binary CFGs dealing with aggressive compiler optimizations for performance estimation”, Asia South-Pacific Design Automation Conference, pp. 452-457. Jeju Island, Korea, 2018

[2] M. Chabot, L. Pierre, A. Nabais-Moreno, "Automated Testing for Cyber-physical Systems: From Scenarios to Executable Tests", FDL'2018, Sept. 2018

[3] M. Chabot, "Tests automatisés dirigés par les exigences pour systèmes cyber-physiques", PhD thesis, Univ. Grenoble Alpes. October 2018

[4] E. Brignon, L. Pierre, "Assertion-Based Verification through Binary Instrumentation", DATE'2019, March 2019