SLS team (System Level Synthesis)

Themes

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

Expertise

Scientific
Integration and optimization of multiprocessor hardware/software systems

Fields of expertise
Multiprocessor architecture, Memory subsystems, On chip embedded operating system, Fast simulation of digital systems, Methods and tools for system level synthesis, Reconfigurable and *ad-hoc* neuronal architectures

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

Industrial transfer
Patent on predictive cache (N° 16201190.2-1953), 5 CIFRE theses over the 4 last years (2012–2016), Creation of the Antfiled spin-off company in 2016.

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.

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 : F. Rousseau, F. Pétrot, O. Alcantara de Lima Junior, Th. Baumela, C. Deschamps, M. France-Pillois, P. Njoyah Ntafam, M. Payet

Cooperations: STMicroelectronics, CEA LETI, Magillem Design Services, LIG

Contracts: CIFRE, SoC-trace (FUI)

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: Device driver generation, software task migration in multi-tile architecture, Network-on-Chip (NoC) based systems, and traces analysis.

1. Device driver generation

Sharing the device driver development knowledge is complex (see *Figure 1*), and writing drivers is an error prone and a time consuming activity. Therefore, we are currently working on automatic driver generation from high-level hardware and software descriptions. From a practical point of view, this would simplify this task much.

In simplest terms, device drivers are means of communication between the kernel and hardware devices. In more advanced terms, a device driver is a specific type of software component/module in the OS that converts requests from higher-level software (e.g., the kernel or an application program) into a series of low-level input/output (I/O) operations specific to a hardware device (e.g., a network interface controller) abstracting the functionality of a physical or virtual device and managing their operation. It hides completely the details of how the device works.

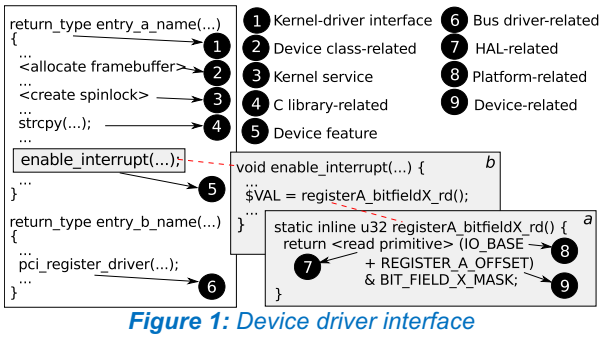


Figure 1: *Device driver interface*

One of the most elementary pieces of information about a device and the driver that manages it, is what function the device accomplishes. Different devices carry out different tasks and need information at different semantic level to work out. There are devices that play sound samples, devices that read and write data on a magnetic disk, and still other devices that display graphics to a video screen, and so on. Thus, because of the various and interrelated sources of information, driver generation is intrinsically complex.

A method, called Me3D, has been investigated to help in device driver generation. Step by step, information is requested from the designers (IP-XACT model of the platform or device, behavior of the driver, …) for the final generation. Based on that methodology, a work on device driver portability is on-going. The idea is to extract from a C code the main properties of a driver for a specific OS, in order to generate a new version for another OS. And finally, as most of current IPs includes a processor, we are also working on the idea of a driverless OS. This means that the driver is embedded with in the IP, and the OS uses standard API to access the driver interface, making what used to be a HW/SW interface a SW/SW interface.

2. Task migration in non-SMP architecture

Task migration is a well know feature in the context of SMP (Symmetrical Multi-Processor) architecture. This becomes very challenging when dealing with non-SMP architecture, and we are developing method and tool to provide such a feature. The main goal is to provide an answer to load balancing, thermal and fault tolerance awareness of processors in MPSoC.

The main idea of task migration is the transfer of a process state plus its address space from the source core (referred to also as home or host node) to the destination one (referred to also as destination node). Its significant cost is coming mainly from the process of transferring the address space. The address space is usually composed of the task stack and heap. However, this is not the only task attribute to be transferred. The whole task state including the address space and the CPU registers, open files and ports, have to be transferred to the destination core to be eventually resumed/ restarted properly.

As a solution, we consider task replication: It consists in having replicas of tasks in the system. When a migration is needed, the task is suspended in its source processor and resumed in the destination processor after the transfer of the process state. The destinations for migrating tasks are determined statically prior to compilation and

linking. This enables the system level design process more open for optimum locations of migrating tasks. Although the system level designer must be aware of the sources and corresponding possible destinations of all migrate-able tasks, our solution has been designed so that there is no involvement from the programmer side whatsoever regarding the migration process. Our solution is implemented as a layer between the operating system and the application. It has been chosen to utilize OS application software interface APIs without modifying the operating system itself and that's to make the solution capable of being plugged over any operating system.

One of the issues of task migration is the inconsistency in the communication arising from the change of the location where the task is supposed to be running in. This consequently, requires informing all the adjacent tasks to the migrating one in the task graph with its new location after migration to keep sending and receiving tokens correctly. Not only location update is important to keep the communication consistent, but also any remaining unprocessed tokens left in channels FIFOs have to be forwarded/re-sent to the right corresponding FIFOs after migration to ensure consistent completion.

3. Network-on-Chip based architecture

Network-on-Chip (NoC) is an interesting communication fabric for multi processing element architectures that benefits from the parallelism of algorithms.

We developed a method that uses a symbolic execution technique to extract the parallelism of an application to be mapped on FPGAs using the flexibility of a NoC communication infrastructure and the properties of a high level programming language. An application specific hardware is then generated using a High Level Synthesis flow. We provide a dedicated mechanism for data paths reconfiguration that allows different applications to run on the same set of processing elements. Thus, the output design is programmable and has a processor-less distributed control. This approach of using NoCs enables us to automatically design generic architectures that can be used on FPGA servers for High Performance Reconfigurable Computing.

Another research domain concerns the automatic generation of traffic generators, obtained from the analysis of NoC simulation traces. The idea is to use an emulation platform in order to re-play faster the original traces for a design space exploration of the NoC architecture or NoC parameters (routing algorithm, mapping, power consumption…). Replacing IP by traffic generators could do this. But traditional trace-based traffic distorts the injection rate and the effects of congestion due to the lack of packets dependency information. We propose a new framework to process traces generated by message passing applications modeled as acyclic task graphs. This framework builds dependency-aware traffic generators (DATGs) by retrieving the packet dependencies from traces in a single simulation. The DATGs accurately replace the application nodes in emulations or simulations to explore the NoC design space. Our experimental analysis showed that our framework is more accurate than trace-based simulation for a broad range of NoC configurations. Moreover, our proposed framework uses only 3 % of the data storage required by the traces.

4. Simulation driven prefetching instructions insertion

Consumer applications are less and less regular, and may have hard to predict memory behaviour in the presence of multiple caches levels. Using cycle accurate simulation, we experiment the automatic detection of high latency load instructions to insert prefetching instructions at the "right" place. Due to its inherent complexity, our goal is merely to point out what is the reason for these high latencies memory access, and not to automatically insert the prefetch instructions. Indeed, reports on automatic insertion of such instructions by compilers tend to show that it is very hard to find the right place and that, for general purpose benchmarks, performance more often degrades than improves.

5. References

[1] P.S. Paolucci and all, Fast and Standalone Dynamic Many-process Applications on Many-tile Embedded Systems and HPC Clusters: The EURETILE programming Environment and Execution Platforms, Journal of System Architecture, Elsevier, published on-line November 24th, 2015

[2] O. Alcantara, V. Fresse, F. Rousseau, H. Sheibanyrad, Synthesis of Dependency-aware Traffic Gererators from NoC Simulation Traces, Journal of System Architecture, Elsevier, available on-line 9 November 2016, pp. 102-113, Vol. 71 (November 2016)

[3] Perrin N. Ntafam, Eric Paire, Alain Clouard, and Frédéric Petrot. Simulation driven insertion of data prefetching instructions for early software-on-SoC optimization. In Proceedings of the 27th International Symposium on Rapid System Prototyping, October 2016, ACM.

Reconfigurable: virtualization, prototyping and archirtectures

Keywords: HLS tools, dynamic reconfiguration, FPGA virtualization, Accelerator Architectures

Members : O. Muller, F. Pétrot, F. Rousseau, A. Bourge, G. Christodoulis, A. Prost-Boucle, A. Wicaksana

Cooperations: ST Microelectronics, LIG, CEA, GIPSA, STEI-ITB

Contracts: HEAVEN (Persyval), Esprit (Nano2017)

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. Enabling context switch on reconfigurable system

To provide better computing efficiency, in terms of execution time, power consumption and area in a given technology, MPSoC architectures often integrate application-specific processors (IP). The nature and number of these dedicated hardware tasks are limited and have to be set up at design time. The reconfigurable computing paradigm gives the opportunity to a virtually infinite hardware task pool.

Typical reconfigurable systems are based on a fabric of cells (chunk of hardware resources, which can be dynamically reconfigured to host a hardware task while the rest of system is still running and computing), an interconnect, CPUs and an operating system. Thus, by time-multiplexing the reconfigurable cells, a reconfigurable system can tend towards better performance that heterogeneous one, while preserving flexibility. In spite of their tremendous potential, reconfigurable systems still fail to convince application programmers.

One key reason is the cooperative multitasking typically used in reconfigurable systems to relax integration constraints of hardware tasks, while preemptive multitasking is the norm elsewhere. In preemptive multitasking, the operating system can start, pause and resume tasks at demand. In order to satisfy system demands, tasks have to rely on the context-switch technic.

In this work, we have proposed a high-level design flow, that automatically generates hardware tasks with context-switch ability from a C description. The design flow manipulates the intermediate representation of a High-Level Synthesis (HLS) tool to build the context extraction mechanism and to optimize performance for the circuit produced. The method is based on efficient checkpoint selection and insertion of a powerful scan-chain into the initial circuit as illustrated Figure 1.

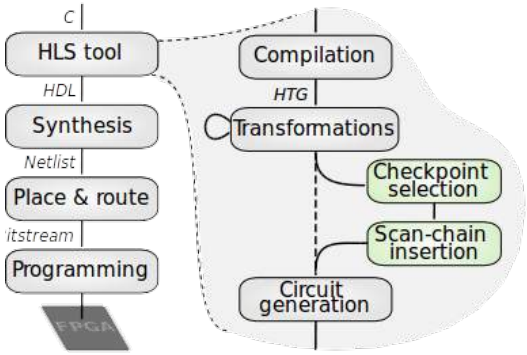


Figure 1: *Proposed design flow*

The first step of the method consists in selecting good execution points, called hardware checkpoints, to perform a context switch on a reconfigurable resource. The obtained selection ensures that the context switch mechanism will respect the latency demanded by the system and tries to minimize the mechanism costs. The checkpoint selection method relies on a static analysis of the finite state machine of a circuit, accessible through the intermediate representation of the HLS tool.

This scan-chain insertion step manages the extraction of flip-flops or memory content, involved in the previous step. Experiments with the system produced show that it has a low hardware overhead for many benchmark applications, and that the hardware added has a negligible impact on application performance. Comparison with current standard methods highlights the efficiency of our contributions. The prototype tool, called CP3, is open-source.

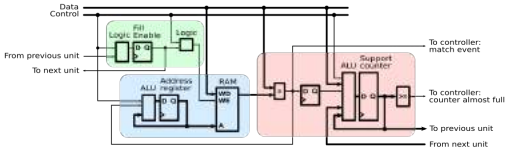
An extension of this work to enlarge its usefulness consists in the communication consistency management. Indeed, the previous work has a strong hypothesis: all the input data has been consumed by the IP and all produced data has been consumed. This new work deals

with the management of all unconsumed data when the context switch happens. A small new piece of hardware is added in the context extraction component to extract as well all the unconsumed data in the input and output FIFOs. In the same time, the management of data producers and consumers are informed of the changes. This is done to be as transparent as possible for the user.

2. Hardware acceleration for mining frequent Item set on streams

Extracting information from huge unorganized data has taken an increasing importance in the last decades. This information extraction takes place in a context in which more and more data sources are data streams, i.e. the data is produced by a continuous and uninterruptible source, leading to a virtually infinite amount of data.There exist many algorithms to extract specific patterns from data batches, such as frequent occurrences, association rules, etc. However, many algorithms require the ability to store the entire data set in order to scan it several times, in an unconstrained processing time.

In this work, we focus on the *a priori* algorithm, the seminal frequent item set mining algorithm, because it requires few or no preprocessing steps and exhibit a high level of fine grain parallelism that can be efficiently used by a specialized hardware device. We specified and implemented a parallel support counting accelerator. We tuned it using specific placement directives allowing us to reach a global LUT usage of around 90 %. The detail of the counting unit is given in the next Figure.



Our accelerator is implemented on a VC709 board (Xilinx), that support the PCIe standard, so that all our experimentations, conducted using the FIMI data set, take into account the whole path from the PC to the hardware.

We compared our implementation to the most efficient software implementation available today (LCM), and to the Micro Automata Processor board (AP). Depending on the dataset, we reach gains of 2.8x compared to LCM, and gains of above 12x compared to Micron's circuit. In all cases, we were much more energy efficient, the AP consuming at worst 192W when the FPGA was consuming 24W.

3. FPGA integration in heterogeneous architectures

As the evolution of programming environments towards heterogeneous programming mainly focuses on CPU/GPU platforms, hardware accelerators on FPGA are still difficult to exploit from a general-purpose parallel application, despites its energy efficiency.

To improve portability, we are extending the standard parallel programming environment OpenMP to support FPGA. We implemented a software framework that enables some computing kernels of an OpenMP application to run on a wide variety of FPGA boards with very little effort from the application programmer. The framework architecture is depicted on Figure 2.

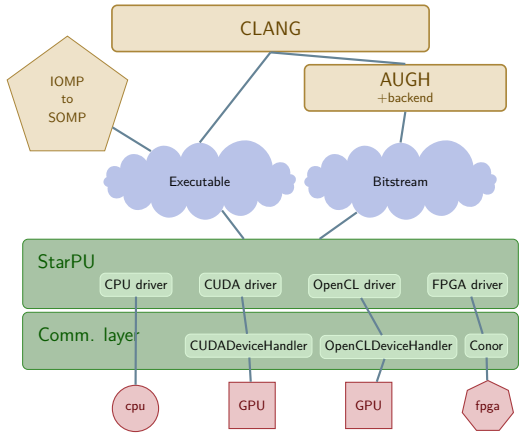


Figure 2: *HEAVEN software framework*

4. 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, and we designed a highly efficient CNN architecture on FPGA. We are currently patenting an ad-hoc asic architecture.

5. References

[1] A. Bourge, O. Muller, F. Rousseau, Generating Efficient Context-Switch Capable Circuits Through Autonomous Design Flow, ACM Transactions on Reconfigurable Technology and Systems, Vol. 10, Issue 1, art. 9, December 2016

[2] A. Prost-Boucle, F. Pétrot, V. Leroy, H. Alemdar : Efficient and versatile FPGA acceleration of support counting for stream mining of sequences and frequent itemsets. ACM Transactions on Reconfigurable Technology and Systems, 2017

[3] H. Alemdar, N. Caldwell, V. Leroy, A. Prost-Boucle, F. Pétrot. (2016). Ternary Neural Networks for Resource-Efficient AI Applications. arXiv preprint arXiv:1609.00222.

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

Keywords: simulation, debug, profiling

Members: H. Bel-Hadj-Amor, A. Bullich, C. Deschamps, J. Dumas, A. Faravelon, O. Matoussi, L. Michel, F. Pétrot, G. Sarrazin, A. Temani.

Cooperations: STMicroelectronics, Magillem Design Services, Kalray, CEA-LETI, UPMC-LIP6

Contracts: COVADEC (FUI), BENEFIC (CATRENE), CAPACITES (Investissements d'avenir), STMicroelectronics, Kalray

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.

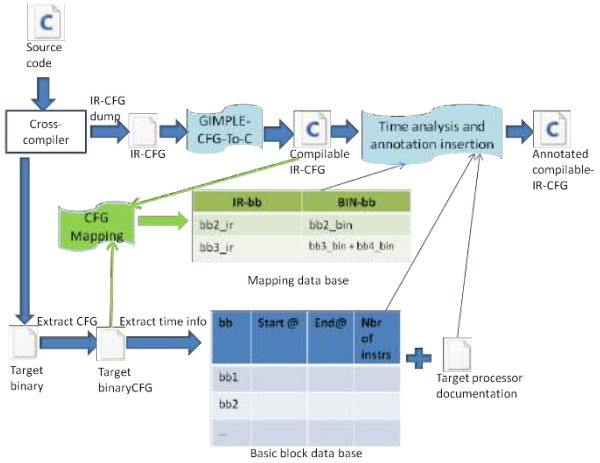


Figure 2: *Approximate CFG Mapping*

During this year, we have kept on working on dynamic binary translation, targeting a) optimization by studying firstly the behaviour of the translation caches, and secondly the interest of using an intermediate representation which follows the single state assignment principles, and b) accurate time modelling, by taking into account branch predictors. We also applied more broadly our simulation technique to evaluate several hardware/software architectures, by varying the level of parallelism of a given application. We experimented coarse grain parallelism with 1 to 8 processors, and fine grain parallelism though the use of SIMD instructions, to support high throughput applications in the context of driver assistance algorithms. Overall, using 8 cores and SIMD instructions led to a tenfold performance improvement compared to a scalar sequential implementation.

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, as illustrated on Figure 2.

Another topic we focused on has been the native simulation of floating point software. As a matter of fact, and even though floating point representation has been standardized by the IEEE, target to host mapping is not as trivial as it seems, and even the simple example below leads to different results if the machine implements a fuse multiply and add or not.

```
void main(void)
{
   volatile float a[2]
        = {0x1.000002p10,0x1.000002 p10};
   volatile float acc = 0.0f;
   for (int i = 0; i < 100000; ++i)
      acc += -(0x1 .000004 p20f - a[0] * a[1]);
   printf("iterations results = %e\n", acc);
}
```

We devised strategies to perform correct computations under several hypotheses, and we showed that guarantying a given behavior is difficult when avoiding full floating point emulation, which costs a lot in term of simulation performance.

2. 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. Our preliminary results show that the optimization of software using a few prefetch instructions can lead to significant performance gains (up to 25 % on data intensive benchmarks). However, our goal is to not to optimize code per see, but to automate the process of finding where to place these instructions taking benefit from the high observability provided by cycle accurate simulation platforms.

3. Memory hierarchy evaluation and design

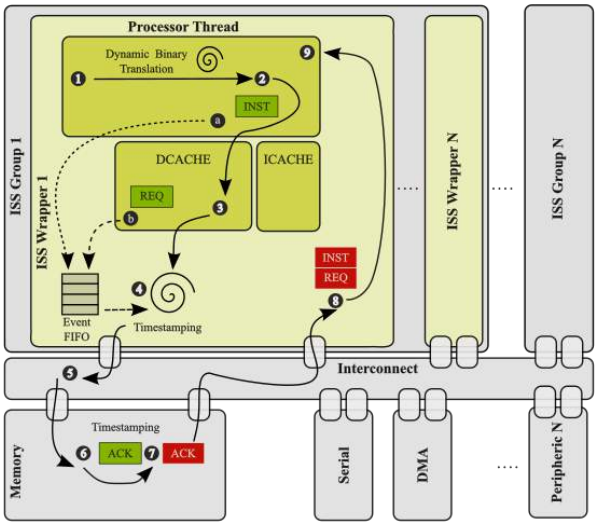
Cache coherence has been a long lasting subject in computer architecture. With the emergence of integrated massively parallel machines, this subject is regaining importance, as the available throughput on network-on-chip is much higher than the one available on circuits that were previously implemented on different boards or on multi-chip modules.

A first line of work, done in cooperation with CEA-LIST, aims at quickly evaluating the performance of the handling of the list of sharers in distributed protocols. Indeed, the original way of maintaining this list is to maintain an exact count of the sharing processor by devoting a bit per processor per cache/memory line and perform multicast. Unfortunately, this hardly scales with hundreds or thousands of processors, as the list requires more memory resources than the data itself! Therefore, other ideas have been proposed, such as limiting the number of sharers to, lets say, 5, and perform a broadcast if the list overflows. However, doing experimentations to define the appropriate list size is very time consuming with cycle accurate models, and going up in abstraction is not simple, as contention has to be modeled. We devised a trace based approach which replays the traces and performs cache simulation, allowing to rank the protocols depending on their performance.

A second line of research, done in cooperation with the LIP6 lab in Paris, aims at introducing support for Network-on-Chip virtualization, allowing the implementation of a set of logical networks on top of a physical network. We call this virtualization since its goal is similar for network to the goal of processor virtualization. The technique leverages the Elevator First routing algorithm that we introduced a few years ago, by giving the user of a planar network, for example, the access to the topology of a cube. These virtual networks can be used for running independent applications and guarantying that their traffic will stay independent (from a functional point of view). It can also be used to support several traffic classes within a system. In that context, we devised a new cache coherence protocol implementation which uses the virtual network to share data between all caches at a given level, saving resources while still providing the traffic independence and thus the deadlock and livelock freeness properties of the underlying algorithms.

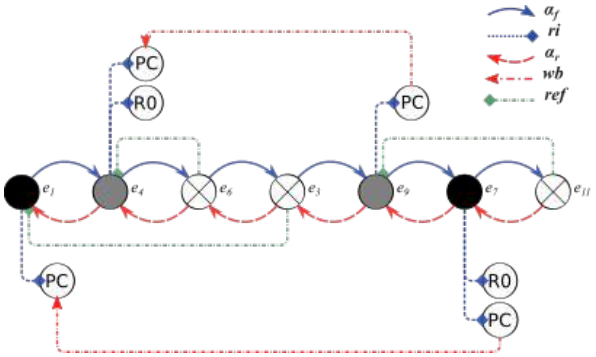
4. Traces analysis

The increasing complexity of MPSoC makes the software developers life harder when chasing bugs. Even if executing a program many times is a conventional debugging process, the non-determinism due to parallel execution often leads to different execution paths and different behaviors.



We propose an approach based on simulation, to ease pin-pointing bugs in a parallel execution. To that aim, we collect traces using a virtual platform based on dynamic binary translation, as illustrated above. Producing the traces is not as easy as it seems, because the components generate their own traces, and we do not want to rely on timestamps since the notion of time in the platform depends on the quality of timing annotations, that can be very approximates. Therefore, we build a partial order based on event dependencies, that we make total using constraints so that walking the trace is straightforward and deterministic. When an execution fails, we re-execute the traces, in either forward or reverse direction. We define a trace model suitable for this task, and detail a strategy for providing forward and reverse execution features to avoid long simulation times during a debug session. The Figure below displays a trace in which the forward (blue) and reverse (red and green) arcs are visible. The green arcs directly point to the last instruction which modified a register that is the

source of the instruction currently executed, allowing to backtrack in a faster way.



We demonstrate experimentally that re-execution is a deterministic process which, when debugging using the usual trial and error developer approach, is much faster than simulation. Building the entire graph of an execution takes from two to five time the time it takes to simulate, but then going forward or backward into the trace is two to seven times faster, and it is not possible to go one step too far, which otherwise would require a new re-execution.

5. References

[1] Cunha, M.A.P., Fournel, N. & Pétrot, F. Des Autom Embed Syst (2016) 20: 47. doi:10.1007/s10617-015-9167-8.

[2] G. Sarrazin, N. Brunie, and F. Pétrot. 2016. Virtual prototyping of floating point units. In Proceedings of the 2016 Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools (RAPIDO '16). ACM

Architecture Algorithm Adequacy for Image Processing and Embedded Vision

Members: S. Mancini, V. Schwambach-Costa, K. Hadj-Salem, M. Bernard, L. Vincent, N. Grataloup, S. Guy, J. Hans

Production: 1 Patent, 9 Conference Papers

Collaborations: ST-Microelectronics, CEA LETI DTBS, CEA LETI/LIST- DACLE, CSUG

Funding: Persyval Labex, AGIR

1. Multi-Core Platforms for High Performance Embedded Video Analytics

Multi-Core platforms are likely to be embedded as hidden accelerators in more complex systems. Such a platform acts as an accelerator from the point of view of the system but has the great capability to be re-programmable for each target system. Then, the challenge is to produce efficient implementation of video analytics application in order to increase the performance and reduce the power consumption. When it comes to video analytics with complex data-dependent algorithms, it is quite difficult to estimate the performance of the system unless one has completely designed the parallel system.

In this thematic, conducted with ST-Microelectronics, an early performance estimator is proposed. The performance estimator, Parana, takes as an input a sequencial code of the application, that may include either OpenMP constructs or pragmas, and produces an estimation of the abstractparallelization on a virtual platform. The estimated performance is computed by incorporating overhead metrics that are produced by micro-benchmarking a reference platform. Results showed that Parana is able to produce estimations as close as 5 % to a reference FPGA prototype of the STxP70-ASMP multi-core platform.

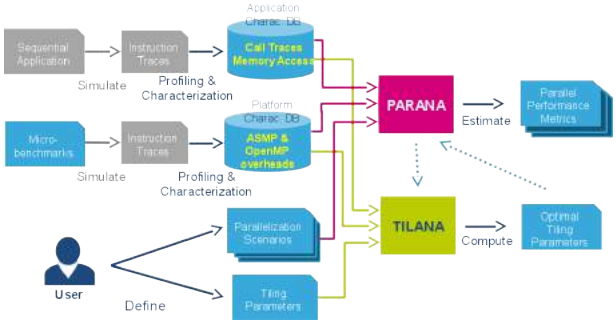


Figure 1: *The Parana flow for abstract early estimation of parallel performance*

2. Adaptive Sampling in 3D SPECT Imaging

SPECT imaging is a nuclear medical imaging modality which relies on the reconstruction of the observed body from the detection of gamma photons. In the CEA-DTBS system, the CdZnTe detectors, grouped in in an array, called a head, allow to measure the direction of the incoming gamma rays. The grid collimators in front of each of the heads allow to further increase the resolution.

The studied system **[1, 6]** is built of several moving heads focused to the active parts of the body thanks to the adaptive sampling system. The latter is accomplished with innovative on-line reconstruction algorithms.

The investigations showed that instead of brute-force hardware accelerated reconstruction of the full data set, it is more interesting to reconstruct the 3D data as the gamma rays are detected.

As the system is validated with a single head, the next step is to build a fully adaptive SPECT camera.

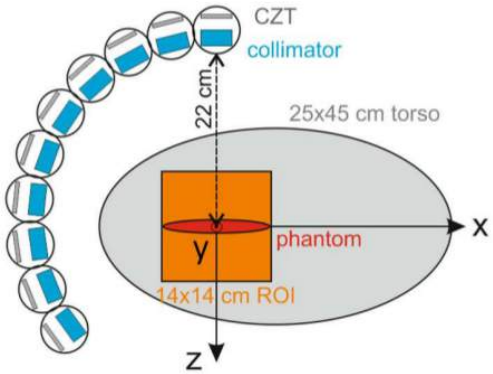


Figure 2*: The CnZnTe Adaptive SPECT camera*

3. Data Management for Image Processing

Image processing algorithms have some very specific memory reference patterns because of the natural use of arrays to represent images. Then, it is interresting to design prefetcherx that exploit this specificity. In this thematic, both static, ie off-line, optimized prefetcher and on-line, ie hardwired, prefetchers are studied.

On-line Adaptive Prefetching

To guess the next memory reference even in presence of irregular fetch sequence, the patented HYP prefetcher **[7, 10]** relies on the Kolmogorov-Smirnov Hypothesis test (KS test). This very new strategy allows the dynamic adaptation of the prefetcher's internal parameters without a prior on the fetch sequence model. Indeed, with classic prefetchers, some fixed parameters of some pre-designed model have to be set, which reduce their portability. The HYP prefetcher learns the fetch sequence characteristics on-line and follows its evolution. The idea is to make the assumption that the probability distribution CDF – Cumulative Distribution Function- of the deltas of the fetch sequence evolves relatively slowly. HYP measures the probability distributions of both a current observation window and a previous window. The

data to prefetch is the one which maximise the KS hypothesis test between the two distributions. Adaptivity is enabled by automatically setting the size of the observed windows.

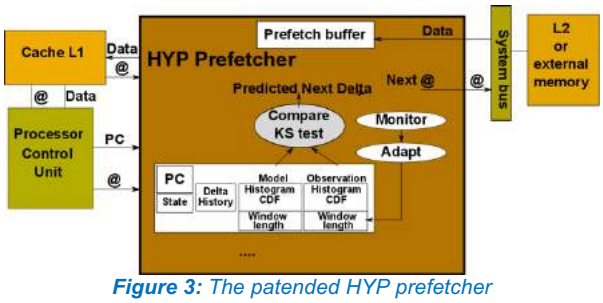


Figure 3: *The patended HYP prefetcher*

Conjoint Scheduling and Prefetching Optimization

When image processing kernel's schedules do not depend on the data values, then it is interesting to optimize the prefetching off-line, at compile time for example. In this thematic, Operations Research is at the heart of the prefetcher's optimization. When the kernel's memory reference do not follow a linear law along the loop indexes, then standard linear optimizations are helpless. In this work, the multi-objective optimization problem of prefetching tiles of data to compute output tiles produces by the kernel is formalized. The new formalization **[2, 4, 5, 8, 9]** allowed to make the link between this problem and known Operations Research problem. Then a very new optimization strategy was proposed to reduce the number of loaded tiles at constant embedded memory amount, namely the KTNS "Keep Tiles Needed the Soonest" strategy. These strategies are highly valuable when the references follow non-linear laws. When it comes to standard linear kernels, standard commercial HLS tools are near optimal.

4. High Performance Hardware Panorama Stitching

Panorama Stitching aims at gathering images from multiple video cameras to form a single image that covers a wider field of view than individual cameras. It is performed with algorithms that performs non-linear mapping of all the images to the final geometric space of the produced imaged. In this work, the produced image is mapped to the spherical coordinates by considering a perfect stitching at a long range focus. In the designed FPGA prototype **[3]**, images come from MIPI-CSI micro camera modules as found in mobile phones. Real time stitching is performed with as little as 6 % of the FPGA resources. The algorithm was implemented by a commercial HLS tool.

The next step is to add geometric and gain auto-calibration to the system. The auto-calibration will be partitioned between hardware and software.

5. Nano-satellite OBC

At CSUG, this field of expertise contributes to the design of the OBC of a student nano-satellite for auroral spectral analysis.

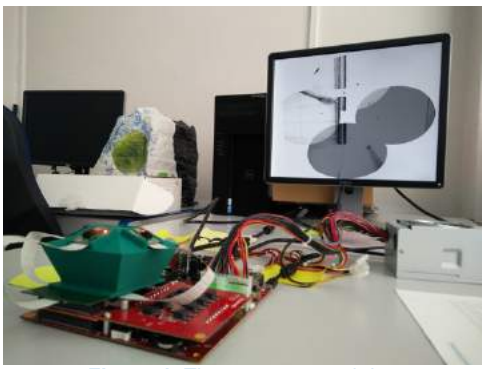


Figure 4: *The prototype real-time FPGA panorama stitching engine*

6. References

[1] M. Bernard, G. Montémont, S. Stanchina, S. Mancini, "Real-Time Processing for an Adaptable SPECT System Based on CZT Detectors", IEEE Nuclear Science Symposium (NSS/MIC'16), Strasbourg, France, October 26-Nov. 6, 2016

[2] K. Hadj Salem, Y. Kieffer, S. Mancini, "Memory Management in Embedded Vision Systems: Optimization Problems and Solution Methods", The Conference on Design, Architectures for Signal and Image Processing (DASIP'16), Rennes, France, October 12-14, 2016

[3] S. Guy, S. Mancini, "Prototyping a Panoptic Camera by means of High Level Synthesis: Demo", International Conference on Distributed Smart Camera, Paris, France, September 12-15, 2016

[4] K. Hadj Salem, Y. Kieffer, S. Mancini, "Formulation and Practical Solution for the Optimization of Memory Accesses in Embedded Vision Systems", The 9th International Workshop on Computational Optimization (WCO'16), Gdansk, Poland, September 11-14, 2016

[5] K. Hadj Salem, Y. Kieffer, S. Mancini, "Optimisation du Fonctionnement d'un Contrôleur de Buffers pour les Systèmes de Vision Embarquée", Conférence d'informatique en Parallélisme, Architecture et Système (ComPAS'16), Lorient, France, July 5-7, 2016

[6] M. Bernard, G. Montémont, S. Stanchina, S. Mancini, "Enabling real-time reconstruction for high intrinsic resolution SPECT systems", IEEE-NPSS Real Time Conference (RT'16), Padova, Italy, June 5-10, 2016

[7] L. Vincent, S. Mancini, S. Lesecq, H.P. Charles, "Model Free Adaptive Data Prefetching using Hypothesis Tests", DAC 2016 conference, WIP workshop, Austin, TX, USA, June 5-9, 2016

[8] Y. Kieffer, K. Hadj Salem, S. Mancini, "Multi-objective optimization for the scheduling of embedded vision accelerators", The 29th Conference of the European Chapter on Combinatorial Optimization (ECCO'16), Budapest, Hungary, May 26-28, 2016

[9] K. Hadj Salem, Y. Kieffer, S. Mancini, "Efficient Algorithms for Memory Management in Embedded Vision Systems", The 11th IEEE International Symposium on Industrial Embedded Systems (SIES-WIP'16), Krakow, Poland, May 23-25, 2016

[10] S. Lesecq, H.P. Charles, S. Mancini, L. Vincent, Patent "Procédé de prédiction ...", Patent number 16201190.2-1953