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 (2013–2017).
The Antfiled company created in 2016 as a spin-off of the SLS team was acquired by GreenSoCs in November 2017.

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, Th. Baumela, C. Deschamps, M. France-Pillois, P. Njoyah Ntafam

Cooperations: STMicroelectronics, CEA LETI, Magillem Design Services, 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: Device driver generation, software task migration in multi-tile architecture, Network-on-Chip (NoC) based systems, and traces analysis.

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 foccuses 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. 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.

2. HW/SW interface redefinition

Systems-on-a-Chip (SoC) are more and more important in computing systems, with the integration of the various subsystems becoming the number one challenge.

Albeit the integration techniques have evolved drastically in the last decade, 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.

However, we believe that there is value in revisiting this frontier; we propose a simple idea: have each driver run on the device it controls, and exchanging information not through a complex set of accesses to registers, but through simple sequential messages à la USB.
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 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. Given the large amount of IP 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.

3. 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

Reconfigurable: virtualization, prototyping and architectures

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

Members : L. Andrade, 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.

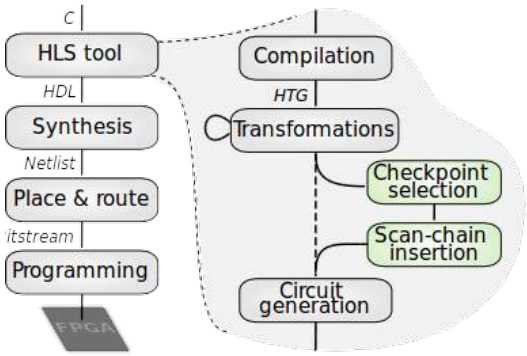


Figure 1: Proposed design flow

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 %.

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.

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 in which all weights are ternary and all activations are ternary, the input data being 8 bit wide.

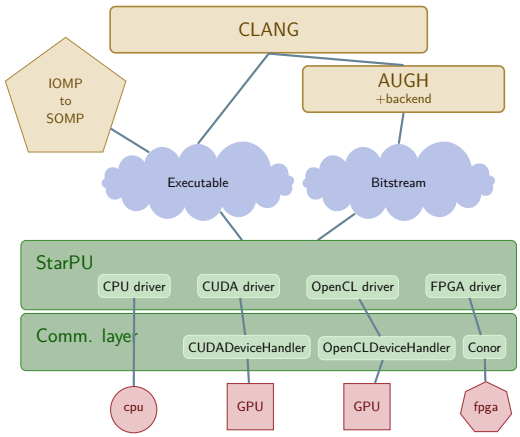
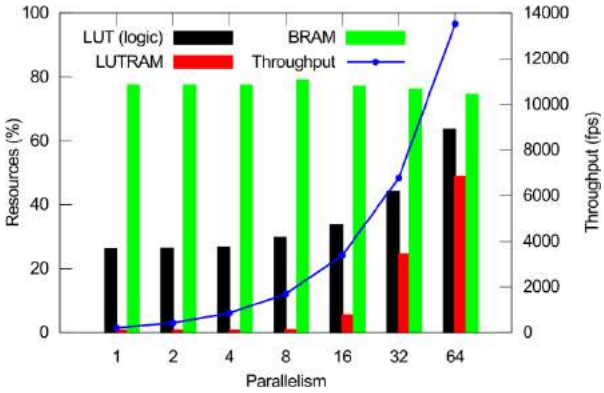


Figure 2: HEAVEN software framework

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 an architecture supporting 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.



5. References

[1] A. Wicaksana, A. Bourge, O. Muller, A. Sasongko, F. Rousseau: "Prototyping Dynamic Task Migration on Heterogeneous Reconfigurable Systems", International Symposium on Rapid System Prototyping (IEEE RSP 2017), pp. 16-22, Oct. 2017, Seoul, South Korea

[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, V. Leroy, A. Prost-Boucle, F. Pétrot. (2016): "Ternary neural networks for resource-efficient AI applications". IJCNN 2017: 2547-2554

[4] A. Prost-Boucle, A. Bourge, F. Pétrot, H. Alemdar, N. Caldwell, V. Leroy: "Scalable high-performance architecture for convolutional ternary neural networks on FPGA". FPL 2017: 1-7

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

Keywords: simulation, debug, profiling

Members: L. Andrade, H. Bel-Hadj-Amor, M. Cunha, C. Deschamps, J. Dumas, A. Faravelon, O. Matoussi, L. Michel, F. Pétrot

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.

During this year, we have kept on working on dynamic binary translation. Our first focus was on support of VLIW architectures, for which to the best of our knowledge no previous approach was published. We devised an approach to handle delayed updates of registers, due to non-unit assumed latency functional units. To limit the impact on simulation speed, all that was possible to do at translation time was done. However, some specific behaviour required action to be taken during the execution of the translated basic block, for example branching. At the end of the day, we were able to demonstrate the feasibility of the approach on a subset of the C6x instruction set under the QEMU dynamic binary translator [1]. The second path we follow was dedicated to accelerating simulation per see. Our idea stems from the fact that the simulation of the logical to physical address translation takes a non-negligible part of the simulation time. 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 hardware assisted virtualization support to build our own page tables mapping the target address space into the appropriate region of the host address space. Our experiments with bare metal software show a speedup of from a few percent to up to 40 % for certain applications [2].

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 [4].

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. 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.

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.

In cooperation with CEA-LIST, we worked on a new sharing list management strategy for cache. The sharing list is a set of meta-data 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 share can vary dynamically, and of a list stored in a shared heap. The idea is to cover as much sharers as possible with the rectangle and store the outliers in the list. This list has a maximum size, and when the list overflows, in last resort, we use broadcast. Experiment on architecture with up to 64 cores show that the approach is competitive in terms of performance with the state of the art, and that is scales better in terms of resource requirements.

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 [5].

4. Traces analysis

Following many years of work around trace analysis, we focused this year on the analysis of traces to detect errors in cache coherence protocols, either hardware or software ones. The idea is to gather execution traces from a multiprocessor execution, typically using Rabbits (a QEMU+SystemC environment), SoCLib or Gem5. Then we analyse these traces using automata to check that at no moment a non-up-to-date memory access could have occurred. This is a bit harder than checking that no staled memory access occurred, as we need to move backward and forward in the trace to identify the earliest moment a data is written, and the latest moment at which it can be read, depending on the instructions being executed on the processors and the state of the memory hierarchy. We designed verification automata for write-through and write-back caches, taking as serialization points accesses by several caches to the same upper level cache or memory. To demonstrate the interest of the approach, we ported the Splash2 benchmark on a cache-non-coherent machine, like, i.e., a tile of the Kalray MPP256. We then ran the program as is and identified cache coherence violations that we corrected by software by either forcing writes to memory or invalidation of a line before reading its content.

5. References

[1] L. Michel, F. Pétrot: "Dynamic Binary Translation of VLIW Codes on Scalar Architectures". IEEE Trans. on CAD of Integrated Circuits and Systems 36(5): 789-800 (2017)

[2] M. Cunha, O. Matoussi, F. Pétrot: "Detecting Software Cache Coherence Violations in MPSoC Using Traces Captured on Virtual Platforms". ACM Trans. Embedded Comput. Syst. 16(2): 30:1-30:21 (2017)

[3] A. Faravelon, O. Gruber, F. Pétrot: "Optimizing Memory Access Performance Using Hardware Assisted Virtualization in Retargetable Dynamic Binary Translation". DSD 2017: 40-46.

[4] O. Matoussi, F. Pétrot: "Loop aware IR-level annotation framework for performance estimation in native simulation". ASP-DAC 2017: 220-225.

[5] H. Belhadj Amor, A. Sheibanyrad, F. Pétrot: "A Distributed NUCA Architecture Using an Efficient NoC Multicasting Support". DSD 2017: 184-191.

Architecture Algorithm Adequacy for Image Processing and Embedded Vision

Members: S. Mancini, K. Hadj-Salem, M. Bernard, L. Brillet-Fernandez, N. Leclaire

Production: 1 Patent, 9 Conference Papers

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

Funding: Persyval Labex, AGIR

1. 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.

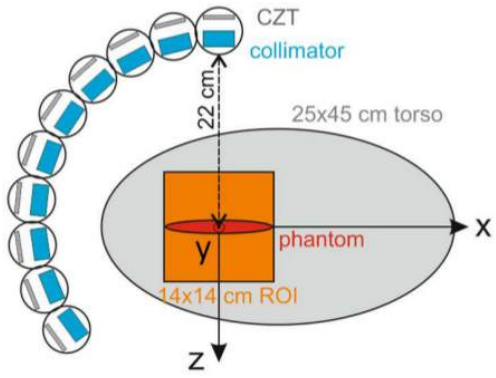


Figure 1: The CnZnTe Adaptive SPECT camera

2. 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.

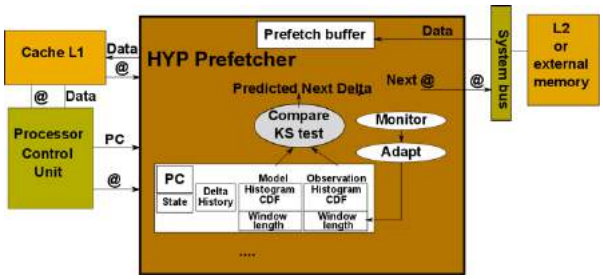


Figure 2: The patended HYP prefetcher

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.

3. 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 image. In this work, the produced image is mapped to the spherical coordinates by considering a perfect stitching at a long range focus. As shown figure4, in the designed FPGA prototype **[3]**, the 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 geometry and gain auto-calibration to the system. The auto-calibration will be partitioned between hardware and software. Also, real-time multi-band blending, to smooth the transition between cameras, is the new challenge to face.

4. Nano-satellite OBC

At CSUG, this field of expertise contributes to the design of the ATISE OBC of a student nano-satellite for auroral spectral analysis. The proposed electronic design based on hardened Xilinx Zynq hardwarre/software platform has been validated by the CNES Janus Program and its implementation is on the process.

5. Efficient Deep Learning & CNN

This new research topic concerns the design of both new deep learning training process and efficient execution of network inference at the software and hardware multi-processor architecture. Conducted with ST-Microelectronics with two PhDs, this new research encompasses all the aspects of deep learning in order to prevent counter-productive too-specific optimization and envision a coherent and balanced strategy.

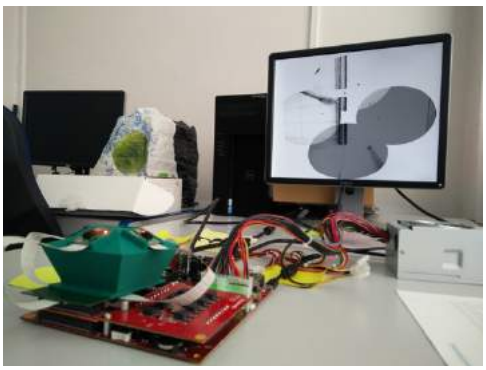


Figure 3: The prototype real-time FPGA panorama stitching engine

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