IP 07

Session : IP Design 3

AMASS CORE: ASSOCIATIVE MEMORY ARRAY FOR SEMANTIC SEARCH (SUPPORTED BY THE FP6 EC RESEARCH PROGRAMME)

J. Carrabina, D. Castells, M. Monton, University Autonoma of Barcelona

Bellaterra, Spain

P. Rujan, F. Vuillod, J. Schwenninger, A. Mages, Learning Computers Int. GmbH

Kirchzarten, Germany

C. Layer, H-J. Pfleiderer, University of Ulm

Ulm, Germany

Abstract :

This paper presents de specification, design and implementation of a high performance search engine core. This core implements a regular Associative Memory Array processing in HW and non-structured data management in SW. This core is being currently integrated in three different software application environments for several search types: text search for data bases, semantic search for web applications and multimedia search for MPEG4/MPEG7 stream management.

The highest speedup of the core comes from the fact that relevant information is coded in pieces of few data bits (1 or 2) that can be easily accessed from RAM memory thanks to the parallel structure of the hardware solution, as supported by complex parallel algorithms..

INTRODUCTION

Due to the enormous increase in stored digital content, any IT device must provide effective and intelligent search-retrieval functionality. Today, in order to retrieve a few relevant KB from a large digital store, one moves in and out several (hundred) GB between memory and processor over a restricted size band bus. The only long-term solution is to delegate this task where it belongs: to the storage medium itself.

By supporting at lowest level simple “broadband” micro-operations performed in parallel, the AMASS
platform provides a hardware platform for error tolerant storage and retrieval of electronic objects described by a set of binary features. The encoding and decoding of objects into/from binary features is performed externally on a general-purpose machine using current software tools. While encoding (indexing) is usually slower and performed off-line, the retrieval phase and the evaluation of the response to a query or chain of queries sets stringent real-time requirements. By identifying and moving the most time-consuming retrieval processes to a hardware platform, this research aims at creating new application-oriented solutions for content addressable database retrieval; ontology based intelligent internet indexers; and multimedia storage management using standard formats (i.e. MPEG4 or MPEG7).

The main goal of that research project has been to develop the HW/SW intellectual property (IP) cores and platforms to implement a general-purpose associative dynamic memory for storing and retrieving binary feature descriptors in HW/SW platforms. This includes creating the general IP and system framework. Beyond developing a C++ abstract hardware/software hybrid model supporting sophisticated pattern recognition and search tasks, the projects also targets a system design approach (SystemC based) for a few concrete application fields. The project delivered a

PCI-attached FPGA demonstrator that can act as a verification framework, for content-based text retrieval system. It includes the corresponding IP core design and system design methodology, the system software, and the middleware resources needed by the full application for the three different companies. Based on this general architecture, AMASS produces application specific platforms for database, multimedia and internet semantic search.

SEARCH ALGORITMS

A search system consists basically of an inverted index, created when data are included into the system via a process called indexing. The inverted index is very similar to the Index found at the end of every manual, except that it contains exactly where all occurring words are located in the manual. An important observation used in all modern search engines is that computationally it is highly advantageous to perform a lossless compression of the inverted index. This reduces the storage requirements and, in addition, searching directly the index in compressed form is faster. Recently, it has been shown that if one compresses the text first (using the Burrows-Wheeler transform), one can build a very effective auto-indexing data structure. There is no need for an inverted index at all – and the systems will find efficiently any part of the text, not only words.

Generalizing this idea to approximate (error-correcting) search, AMASS constructs from the original text first statistically significant features. These features are then encoded using a redundant but lossy compressor. The AMASS encoder deals with "digits" of UNICODE (16 bit integers), and allows for the implementation of highly parallel implementations using operands with few bits. If no exact match to the query is found, a systematic search for the "most similar" number(s) can be done more efficiently in the binary feature representation than in the original one. Hence, the "AMASS" search engine provides also a list of "most similar" hits to the query. By appropriately designing the relevant features of the text, one can also add semantic similarity – in the simplest case by forcing synonyms into the same binary form. Due to these random- and semantically forced collisions, the AMASS system must follow a multistage approach: 1) first filter out the "bad" candidates and 2) make sure that the remaining candidates were not chosen by chance (collisions!). The first phase is called filtering; the second one is part of the relevance scoring. Note that filtering must "penalize" the candidates, while relevance scoring is evaluating their "similarity" to the query. The main challenge here is to define what is meant mathematically by "penalty" and "similarity", especially when comparing natural language texts.

SYSTEM PLATFORM

The best approach when dealing with such a complex design process is to search for parallel abstraction layers, and the corresponding APIs, starting from a standard Application SW stack from the SW-only implementation. From next figures one can identify four such layers.

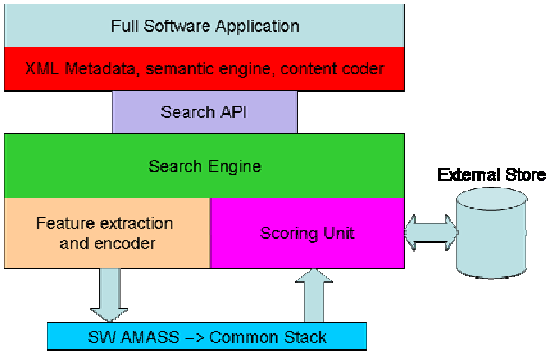


Figure 1. Application SW Stack for AMASS

The most abstract layer refers to the algorithms used for encoding information according to the applications that we will characterize as: database, multimedia, and Internet. Below that, there are two different SW layers and the final HW layer.

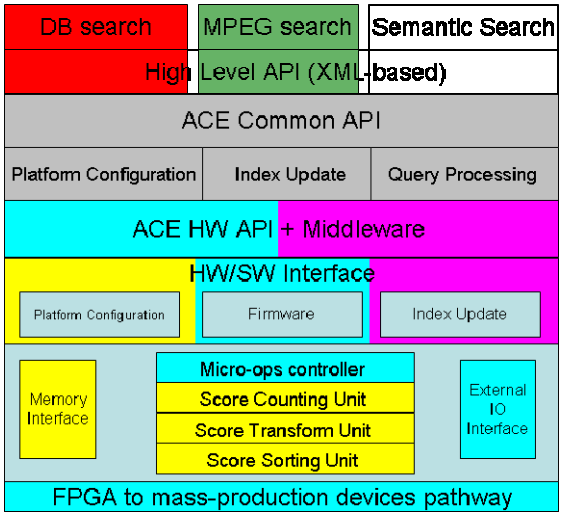


Figure 2. Platform Stack form AMASS

Thanks to that platform description, algorithms and software platforms have been concurrently developed. A high level API accesses the CARE module (Content Addressable Record Extraction) that implements the Encoder and the Relevance scoring. Above the common API, we built one basic and three derived golden models. These golden models consist of executable C/C++ code, which actually performs the desired tasks on a general purpose machine. The golden models must

take into account the desired internal hardware structure and function as blueprint for the next stages.

The other three levels concentrate on the HW/SW implementations in stand-alone platforms. The development followed the standard path suggested by modern methodologies for platforms development, based on a gradual design refinement. This process starts with a general-purpose (but virtual) prototyping platform, designed according to the common API requirements and the golden model (C/C++ code describing the tasks to be performed). This design-phase clarifies different possible architectural choices and specifies the range of attainable specs, as restricted by the particular platform and the set goals. In our case, the platform has been FPGA-based. Design methodology allows mapping to other platforms (in particular ASIC platforms) in order to either deliver better performance (but require large investments) or embed the solution in specific products (like mass storage solution).

Hence, we implemented a dedicated high-performance, optimized-cost implementation of the search engine core. At this level, concrete choices have been made after which the core of the engine, namely the controller and the SCU, STU, and SSU have been built, together with the corresponding embedded software. After passing verification and performance, this core constitutes the unique, stable kern of the ACE. Verification used new and modern techniques, like HW/SW co-simulation using Virtual PCs to execute concurrently C++ (from the SW golden models) and SystemC (for the HW) modules. That helped to isolate and design HW drivers and related MW. In parallel with the core design, different applications were also built.

Once the core integration has been successfully performed, a new iteration started. After linking the system into the original software-based implementations, the common API and the design has been re-evaluated and slightly modified in order to optimize the performance of the whole system. This will lead to products sharing all additional advantages of such HW/SW hybrid systems: performance improvement, intellectual property protection, easy customization, and the availability of new services.

HW SOLUTION

The hardware core is divided into two stages. Stage I lets to a special associative processor (ACE for associative computing environment) for performing filtering, while Stage II includes platform (feature-) specific encoders and IO channels towards the original data store. Note that filtering should reduce the number of candidates to the extent that general purpose processing and that has no real-time constraints in the relevance scoring phase.

The filtering phase works only with “binary” numbers (strings). The query is also encoded into a binary string and send to the ACE processor. This hardware platform supports the parallel computation of the Hamming distance (or other penalty measure) between the query-string and all feature strings stored in the internal memory. The Score Computing Unit performs the parallel computation of the penalties by analyzing the matching bit stream with a window and penalties. The Score Sorting Unit sorts the total penalties and the list of the best candidates is sent back to the mainframe, where it enters the Relevance Scoring process. This now proceeds with a full comparison between the original query and the original form of the potential candidates. One important problem is to find a penalty defined at binary level simulating closely the Relevance Scoring acting on the original text. This penalty function must be also amenable to simple bit-by-bit operations offered by the HW.

Memory management becomes a crucial aspect for the system performance. In order to optimize efficiency, several decisions have been taken into account. The main one refers to the fact that, many indexing data (named SignatureAttributeMatrix) can be downloaded at the same time to the local memory from the ACE core. We selected SRAM as a preferred memory type in order to optimize HW latencies. This is a separate module that can be interchanged according any other architectural requirement (i.e. SDRAM).

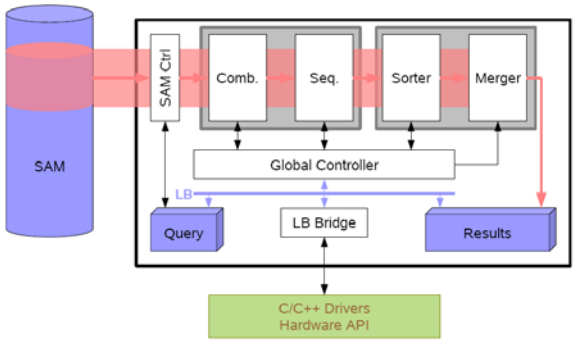


Figure 3. HW CORE structure

Memory management is crucial in order to achieve a large throughput. Two main considerations concerning the SAM access have to be taken into account: (1) features of a given text should be coded in a large string, and (2) in order to compare different indexes with the reference one, we need to access as parallel as possible and accumulate scores. Next figure tries to show this 2D

consideration together with the need of storing both key (to identify the string) and the index (that characterizes it).

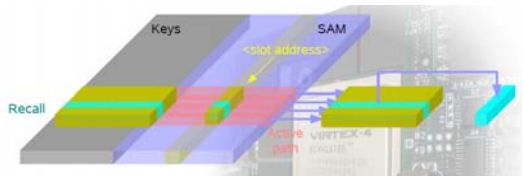


Figure 4. Basic SAM Data Access Scheme

Once again, complex FPGAs allows increasing throughput and to easily manage bit-wise operations with high parallelism.

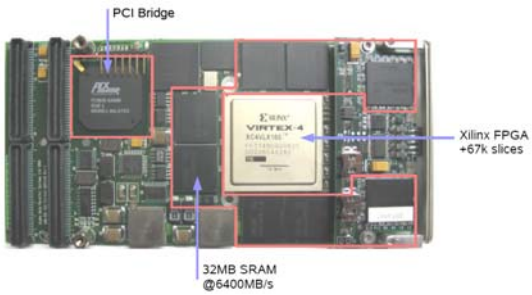
ARCHITECTURAL PLATFORM

The hardware platform does not imply a fixed silicon implementation. Hardware (designed in VHDL and modeled in SystemC) and software (standard C++ libraries) processes have been mapped into one specific prototyping platforms that can be used as part of a global solution for the SW Companies founding the project. Furthermore, concerning the AMASS IP core point of view, a commercial of the shelf FPGA-based PCI-X platform helped us to characterize core performance and to determine the optimal architectural parameters to balance performance and cost according to the required implementation.

Prototype board is an Alpha-Data ADM-XRC-4 platform (see next figure), that is connected to the PCI through a PMC extension. PMC allows us to connect the board to the different PCI standards (PCI-X and PCIe) using different backplanes.

Figure 5. PC-PCI based solution on a PCM slot

Different options were proposed to manage the ACE core to the main PC processor through the PCI bus, and also through the internal bus provided by Alpha-Data, in such a way that can be easily



managed by their SW drivers. Among those options we considered using a soft core processor, coming either by Xilinx (i.e. xBlaze) or technology independent ones (i.e. OpenRisc or similar ones). After analyzing the reduced control complexity and the need of carefully managing timing signals we decided to implement the controller directly on a FSM implementation that can also be considered portable, faster and independent of the OCB architecture.

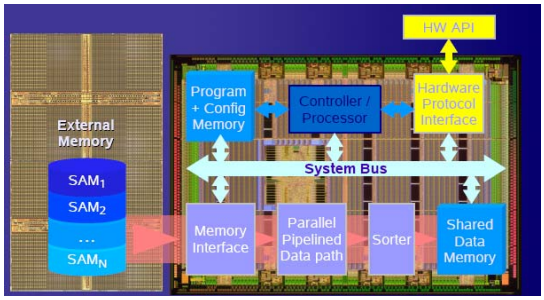


Figure 6. HW IP and Platform Architecture.

APP, SW & FW LAYERS

Link between application specific application (data-base, multimedia or semantic) and fixed HW implementation using our binary coded ACE engine requires some further application dependent data management in order to extract relevant search information from structured data. This is done at the high level API. This structured data can produces either multiple signature matrices (SAM) of specific masking flags that have to be taken into account. Furthermore, both structures require several implementation dependent data management in order to be correctly handled by the HW IP.

VERIFICATION & PERFORMANCE

As described in above sections, we started our IP design from a SW implementation. That led to a complex process to transform SW structures for its hardware implementation. Developing HW modules for standard platforms like PCs or embedded devices required complete system emulator availability to detect and fix bugs on developed HW, O.S., drivers and applications.

At this level, we introduced the use of a plug-in to an open-source CPU emulator, QEMU that enables mixed simulations between platforms emulators and HW modules described in SystemC. QEMU can emulate entire system based on its currently supported CPUs: Intel x86 and x86_64, ARM, SPARC, PowerPC, MIPS, and can also emulate entire systems including most common peripherals. These emulated systems are able to run unmodified standard OS like GNU/Linux and MS-WindowsXP.

In this way, we can connect SystemC to the PCI bus for the HW/SW design and verification of our PC-based platform running both the original and accelerated codes. Unfortunately, QEMU is oriented to functional verification and it does not allow to measure speed performance.

For our purposes, within this framework environment, it is possible to write OS drivers at the same time than the HW is developed, so that we can test the final application running in this virtual platform.
Next figure shows the structure of the complete execution model of the C++ environment running the QEMU that interfaces the SystemC model of our IP using TLM level interfaces.

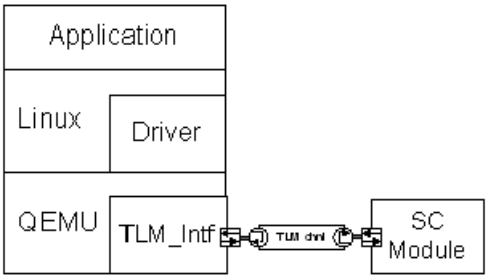


Figure 7. QEMU emulation framework

At TLM emulation level, every access to the bus involving the target device frozes QEMU and starts the SystemC simulation to perform the PCI bus transaction into TLM channel (read/write). Once it's finished, QEMU is resumed, bus access is finished and virtual system simulation continues properly.

This channel can be built simulating the behavior and restrictions of real PCI bus used in order to extract the number of accesses as approximate timing outcome; good enough to test if requirements are met. Moreover, TLM emulation can be used to extract data traces that are passing through the virtual bus. These data values can be used later to do some exhaustive and more realistic simulations of SystemC module that is under development.

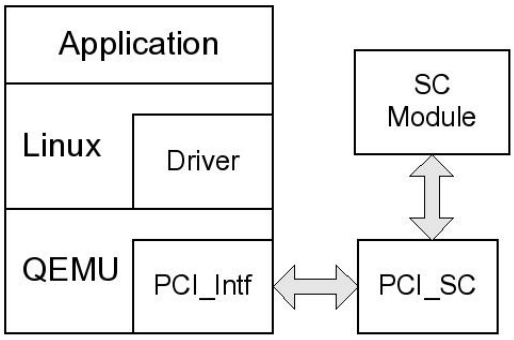


Figure 8. QEMU PC-based emulation framework

Though this figure introduces Linux as OS (used in the early development stages), we have also been running Windows XP on top of QEMU. The main advantage of this approach has been the speedup of the verification. We evaluated that the PC emulation running on a PC is "only" a factor 7 slower that the PC itself running the same application (we also analyzed that for ARM this factor can be reduced down to 1.4).

To implement system level verification, for both functional and performance tests, we implemented a GUI that can configure the different configuration parameters of the system. Next figures show the look and feel of the GUI for a query, that includes the confidence of the produced results.

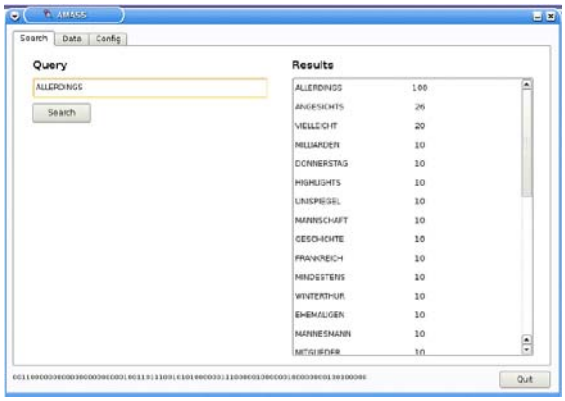


Figure 7. GUI for AMASS Verification

RESULTS

Results of that research come of course at different levels and have been published in different forums and are being exploited bye the founding companies. Basically, sped-up factor for data base search applications (and for large enough data sets) approaches a **factor of 50 in speed performance** compared to a standard of-the-shelf Intel CPU.

We can say that at application level, the final platform consists in an acceleration board (nowadays Alpha Data ADM-XRC-4) connected through the PCI (PCI, PCI-X or PCIe) to a standard racked CPU (Dual Core Xeon, 80GB HD, 1GB RAM) that can run WinXP (either Embedded of Professional) or Linux. Remote access is provided for specific search applications.

Concerning Hardware IP Implementation, current implementation in the Alpha Data platform using the Virtex4LX160 lets to the results shown in the following table.

Resources	Used	Avail.	Util.
Slice FF	20,256	135,168	14%
4 input LUTs	33,068	135,168	24%
Occupied slices	**20,477**	67,584	**30%**
Bonded IOBs	**567**	768	73%
RAMB16s	**76**	288	26%
DSP48s	**65**	96	67%
DCM ADVs	3	12	25%

Table 1. AMASS HW Core Implementation Results

The AMASS core in this platform is running at 100MHZ and internally the data bus width is 256bit that connects the core with the 8 ZBT SSRAM memory banks (100MHz).

CONCLUSIONS

This paper presents the AMASS IP core, Associative Memory Array for Semantic Search that implements search algorithms that can be applied to a different set of domains. Starting from a SW acceleration perspective, the final modular implementation opens its portability to a System on a Chip integration for both general purpose or application specific implementations.

AMASS core contains different design resources: System, Low-level and HW level APIs, functional (timed) SystemC verification model, QEMU-based HW/SW emulation platform, FPGA-based PCM/PCI board implementation, GUI for test and performance measurements. Performance estimations for this IP can be extracted and extrapolated from the implementation results on FPGA.

ACKNOWLEDGEMENTS

This research has been founded by the CRAFT Program (FP6) of the EU as Project CRAFT 18283/AMASS: “IP Core and System Design of Associative Memory Arrays for Semantic Search”.

We also would like to thank researchers from other members of the consortium D. Navarro and J. Contreras from iSOCO, J. Sanchez from VCR; S. Segarceanu and F. Haszler from KEPLER-ROMINFO; and especially D. Lucanu, O. Gheorgies, A. Iftene and V. Radulescu from UAIC.

REFERENCES

C. Layer, D. Schaupp, H.-J. Pfleiderer: ”Area and Throughput Aware Comparator Networks Optimization for Parallel Data Processing on FPGA”. Proc. IEEE ISCAS 2007

C. Layer, H.-J. Pfleiderer “Hardware Implementation of an Approximate String Matching Algorithm Using Bit Parallel Processing for Text Information Retrieval Systems” Proc IEEE Int. W. on Signal Processing Systems, 2005.

C. Layer, H.-J. Pfleiderer, “Efficient Hardware Search Engine for Associative Content Retrieval of Long Queries in Huge Multimedia Databases” Proc. IEEE Int. Conf. on Multimedia and Expo. 2005.

C. Layer, H.-J. Pfleiderer: “A Reconfigurable Recurrent Bitonic Sorting Network for Concurrently Accessible Data”. Int. Conf. FPL. 2004.

C. Layer, H.-J. Pfleiderer, P. Rujan, G. Lapir: “ High Performance System Architecture of an Associative Computing Engine” IFIP Conf. on VLSI of SoC 2003.

Layer, C.; Pfleiderer, H.-J.; A scalable highly parallel VLSI architecture dedicated to associative computing algorithms. IEEE Research in Microelectronics and Electronics, 2005 PhD

L. Ribas-Xirgo, D. Castells, M. Monton, J. Carrabina, "A Rapid Sorting Unit based on Programmable Shifting Register Files" Int. Conf. DCIS, 2005.

A. Portero, L. Ribas, J. Carabina, “Hardware Synthesis of Parallel Machines from SystemC” FDL2005.

Ribas L., Castells D., Carrabina J. "A Linear Sorter Core based on a Programmable Register File" DCIS 2004.

Castells D. et al. “Comparing Design Flows for Structural System Level Specifications facing FPGA Platforms" DCIS2004

M. Monton, D. Castells, A. Portero, J. Carabina “Implementación SystemC sintetizable de un procesador asociativo para elalgoritmo de Dijkstra” JCRA 2005.

M. Monton, A. Portero, M. Moreno, B. Martinez , J. Carrabina“ Mixed SW/SystemC SoC Emulation Framework” Proc. IEEE Conf on Industrial Electronics, 2007.

M. Monton et al. “Accelerating Semantic Search with Application of Specific Platforms” Workshop on Semantic Web Technology for Law. ICAIL 2007.

Y. Nakamura et al.. “A fast hardware/software co-verification method for system-on-a-chip by using a C/C++ simulator and FPGA emulator with shared register communication.”. Proc. DAC2004.

J.-G. Lee et al. “Simulation Acceleration of Transaction-Level Models for SoC with RTL sub-blocks”. Proc. ASP-DAC 2005

I. Granovsky, E. Perlin "Integrating PCI Express IP in a SoC", Proc. IP-SoC 2006.

WEB REFERENCES

http://www.amass-platform.com/applications.html
http://mikro.e-technik.uni-ulm.de/amass/
http://www.lci-software.com/
http://cephis.uab.cat/proj/public/qemu/index.htm
http://fabrice.bellard.free.fr/qemu/
http://www.alpha-data.com/adm-xrc-4.html