Report on regular architectures and selection of test-benches

D1.1

Lead beneficiary	6 (STMicroelectronics SRL)
Synaptic Identifier	SYNAPTIC-1-D1.1
Work package - Task:	WP1 / T1
Document status:	Approved
Confidentiality:	Public

Abstract:

This report contains an incomplete summary of the art of computing architectures and engines found both in state-of-the-art academic proposals and commercially available solutions eligible to deployment in embedded systems. The document attempts at qualitatively evaluate pros vs. cons of the possible choices to identify the ones that bear the most interest in terms of industrial relevance as well as the most potential to expose the benefits envisioned for the SYNAPTIC project. A selection of pragmatic and accessible test-benches is made that will form the bases for the validation of results for the whole project

Document History

Version	Date	Reason of Change
1	*April 15,2010*	*Document structure and Index.*
2	*May 3, 2010*	*First complete draft of the document, collecting contributions from all partners.*
3	*May 28, 2010*	*All contributions included, ready for review.*
4	*May 31, 2010*	*Final version that integrates all comments.*

The following list of authors reflects the major contribution to the writing of the document.

AUTHOR	ORGANIZATION
Andrea Carlo Ornstein	STMicroelectronics
Thomas Boesch	STMicroelectronics
Giuseppe Desoli	STMicroelectronics
Arnaud Grasset	Thales

publication, the user should assume that all materials contained and/or referenced in this document are confidential and proprietary unless otherwise indicated or apparent from the nature of such materials (for example, references to publicly available forms or documents). Disclosure or use of this document or any material contained herein, other than as expressly permitted, is prohibited without the prior written consent of Synaptic or such other party that may grant permission to use its proprietary material. The trademarks, logos, and service marks displayed in this document are the registered and unregistered trademarks of Synaptic, its members and its licensors.

The copyright and trademarks owned by Synaptic, whether registered or unregistered, may not be used in connection with any product or service that is not owned, approved or distributed by Synaptic, and may not be used in any manner that is likely to cause customer confusion or that disparages Synaptic. Nothing contained in this document should be construed as granting by implication, any license or right to use any copyright without the express written consent of Synaptic, its licensors or a third party owner of any such trademark.

Table of Contents

1. Executive Summary

It is becoming common knowledge that the progress and scaling of CMOS technology is hitting several brick walls. The most obvious is the fact that due to scaling the dimensions of silicon devices are approaching the atomic scale and are hence subject to atomic uncertainties. According to the ITRS roadmap, this is becomes of concern at 45nm, and will become critical at the 22nm technology node and below. It is also well-know that other 'brick walls' are likely to impair technology scaling even before this. Lithography resolution, photo resist and electrical field limits (due to power supply voltage fluctuations, thin oxide breakdowns, etc.) are already critical issues for 65nm and 45nm technologies. However, to achieve the predictions of Moore's Law, whilst increased transistor density is of course important, the next key challenge is to optimally integrate foundations such as process technology into the architecture/micro-architecture and system tool flows. Such integration will drastically reduce development cycle and NRE costs, allowing tighter time-to-market windows, and achieve high yield to compensate for the soaring economic investments necessary to develop the next generation nanometer technology nodes and build their manufacturing facilities. Such a trend will dictate a deep rethinking of system architectures and design methodologies, to address the following challenges:

- Low power and power management: Power consumption will be even more critical because of the high-performance requirements, and increased leakage current, at 65nm and below;
- Fault tolerance: The reliability of the components, plus the ability of the total system to operate effectively with degraded elements, are essential;
- Predictability: An early evaluation of process-related effects and variability at the architectural level will provide a direct path from architecture to silicon manufacturing, thus avoiding costly design re-spins;
- Manufacturability: A fast yield ramp with high-performance, cost-effective, and predictable manufacturing process is necessary to optimize the profitability of the overall design-to-silicon flow.
- Programmability and scalability: Deployment of effective and scalable application that map on regular or semi-regular structures of parallel computing nodes

After a qualitative analysis of the state-of-the-art both academic and commercially offered in the industry for a number of computing architecture and engines,; the

final selection of benchmarks for the validation and development of the SYNAPTIC technology is identified with a number of subsystems from both the STMicroelectronics and Thales architectures, namely the xSTream and Ter@pix .

To keep the extra development and refactoring effort low enough to be pragmatically addressed by the project, we have selected a small number of candidate IPs.

Thales will deliver a subset RTL that comprises an array of Processing Elements with their local memories controlled by a sequencer.

While STMicroelectronics will benchmark the full RTL for the xPE processing engine and provide selected subblocks for the early project investigation during the first year (for example a processor vector ALU). For the interconnect the RTL two bus topologies will be leveraged, one for a standard STBus node and an equivalent STNoC interconnect.

If resources and time will allow it, STMicroelectronics will also benchmark the results against the RTL generated automatically for a selected IP functionality with an industry standard data-path generation tool such as Synfora PICO or equivalent (e.g. CatapultC).

2. Introduction

This report contains an incomplete summary of the art of computing architectures and engines found both in state-of-the-art academic proposals and commercially available solutions eligible to deployment in embedded systems. The document attempts at qualitatively evaluate pros vs. cons of the possible choices to identify the ones that bear the most interest in terms of industrial relevance as well as the most potential to expose the benefits envisioned for the SYNAPTIC project.

It has two main focuses: the first one on general purpose computing architectures and accelerators, while the second on the role of regular structures for bus hierarchies and interconnects.

A selection of pragmatic and accessible test-benches is made that will form the bases for the validation of results for the whole project.

3. State of the art of Regular computing

Two different styles of regular computing architectures will be exploited and defined. These include coarse and fine grained regular computing arrays of regular processors and data-paths accelerators as well as fine grained scalar network operand clusters of smaller computational units such as ALUs or similar.

We will analyze regular computing architectures and accelerators for high performance SoCs proposed in the state of the art and then present a qualitative analysis of advantages and disadvantages of different architecture templates in light of the exploitation of the regular design flow methodology defined in the project.

3.1. Coarse grained regular computing

System engineers have to consider more and more parallel computing architectures for embedded systems due to growing processing performance requirements of today's software applications. With tight power budgets and signal speed which is not getting significantly faster with new manufacturing technologies, uni-processor systems are at their limits. Instead, massive parallel architectures which try to exploit parallelism at all levels are getting into focus.

The circuit integration density is still growing in line with Moore's law which means that the system complexity that can be integrated on a single System-on-Chip (SoC) is growing tremendously. Due to this growing complexity the verification effort required and the related risk of design faults are more and more crucial. The identical replication of few, well-tested building blocks such as Arithmetical Logic Units (ALUs), multipliers, or complete processor cores reduce the design and verification effort on the block level significantly (not per se also on the system level).

Traditionally, software has been written for serial computation to be run on a single computer having a single Central Processing Unit. Parallel computing instead is the simultaneous use of multiple compute resources to solve a computational problem to be run using multiple CPUs.

Multiprocessor systems can be subdivided in many ways, depending on the criteria used. They can be grouped looking at node types (heterogeneous vs. homogeneous systems) or at the memory subsystems (shared memory, distributed memory) or at how the nodes are connected (bus, Noc …).

In Homogeneous systems the improvements gained by the regular extraction tools used on one core may have huge effects on the overall system; depending on how many times the core is replicated. The improvements are proportional to the number of instances of the optimized cores.

In heterogeneous systems instead the improvements gained on the single cores contribute only once on the overall system and so the optimization process applied is less effective.

One of the more widely used classifications, in use since 1966, is called Flynn's Taxonomy.

There are 4 possible classifications according to Flynn:

- S I S D: Single Instruction, Single Data
- S I M D: Single Instruction, Multiple Data
- M I S D: Multiple Instruction, Single Data
- M I M D: Multiple Instruction, Multiple Data

The regularity extraction tools will probably be more effective on the SIMD and MIMD architectures since we are doing the same operations in parallel on multiple inputs and so it is probable to find interesting patterns in their implementations.

Main memory in a parallel computer is either shared memory or distributed memory.

Shared memory systems generally have in common the ability for all processors to access all memory as global address space. Multiple processors can operate independently, share the same memory resources and changes in a memory location are visible to all other processors.

Computer architectures in which each element of main memory can be accessed with equal latency and bandwidth are known as Uniform Memory Access (UMA) systems; a system that does not have this property instead is known as Non-Uniform Memory Access (NUMA) architecture.

Most NUMA architectures are distributed memory systems that require communication networks to enable inter-processor memory access. All computing nodes have their own local memory and operate independently of each other. Changes made to a local memory have no effect on the memory content of other processors.

From the point of view of a compound gate design flow methodology parallel computing architectures are of high interest due to the following reasons:

1. Distributed memory approaches have more memory controllers, more memory subsystems; these systems are more interesting for the extraction tools since optimizations in these subsystems have a greater impact on the overall architecture.
2. The data-path of a single programmable computing node is typically timing critical and crucial to achieve high processing performance at maximum system frequency. Improving signal speed with a compound gate implementation could enable a higher maximum data-path frequency and potentially increased data processing throughput.
3. The data-path of a programmable architecture handles wide data operands at high frequencies. Therefore, the overall power consumption of a processor core is often dominated by the data-path power dissipation and heat removal

is a dominant issue in modern processor systems. A compound gates optimization of data-path components with high utilization and large numbers signals with high toggling activity may lead to a significant reduction of power dissipation and/or lower thermal issues.

4. Many critical tasks in multimedia, telecommunication, or signal processing applications have significant level of parallelism which can be exploited with parallel functional units. Functional units which execute low-level mathematical operations are often replicated in large numbers and integrated in a single data-path to handle multiple data values with a single instruction (Single Instruction Multiple Data - SIMD). Due to the replication of identical units the gain of a compound gate optimization on one block would be multiplied by the number of instantiations while keeping the extraction effort limited to a small entity.

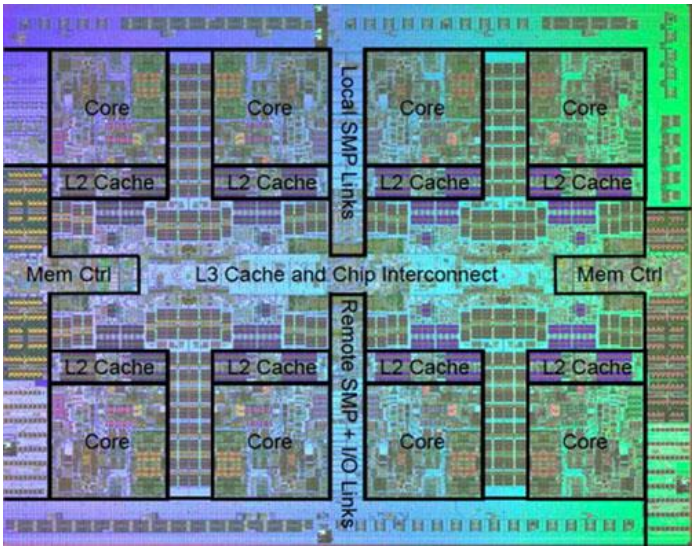
In the following sections, three architecture families which seem to be particularly attractive for compound gate optimization design flow are presented and well-known and/or commercially available implementations are discussed.

3.1.1. Coarse grained multiprocessor architectures

Power7

The Power7 [Power7_10] is the first IBM eight-core processor with four-way simultaneous-multithreading operations per core. It has an advanced memory hierarchy with three levels of on-chip cache.

The following image shows the Power7 chip, including the eight processor cores, each having 12 execution units capable of running four-way-SMT.



Power7 die photo (Credit: IBM)

To feed all the cores the processor has two memory controllers, one on each side of the chip. The processor features a balanced design. Some cores can be turned off, reallocating energy and cache to the remaining cores which all have an independent frequency controller.

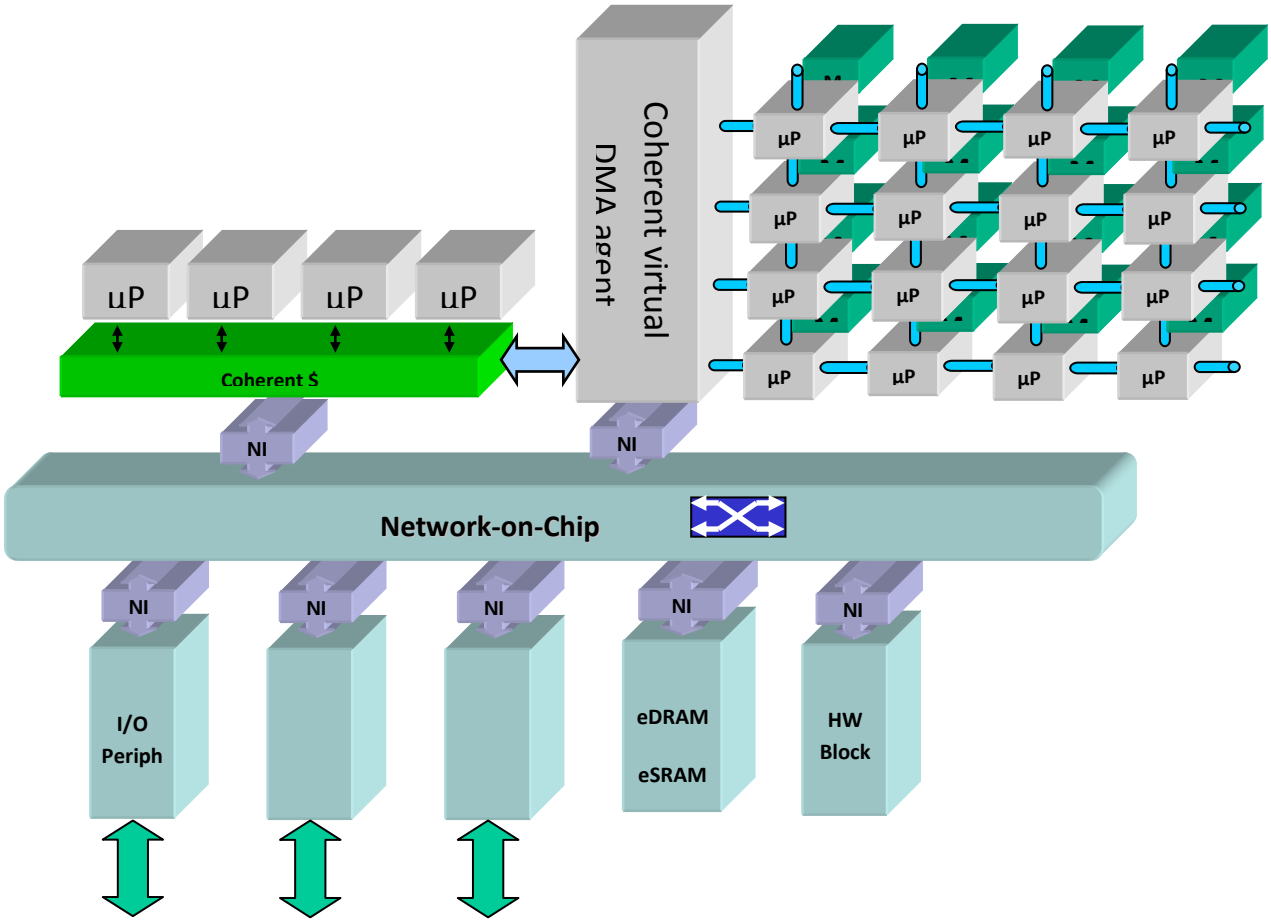
There are a total of twelve execution units per core: two fixed-point units, two load-store units, four FPUs pipelines, one vector, one branch execution unit, one condition register logic unit and one decimal floating point unit pipeline. In a given cycle, each Power7 core can fetch, issue and execute up to eight instructions.

This architecture poses a lot of opportunities for the extraction tools; there are a lot of unit replicated inside the cores where the optimizations can be effective and the cores are instantiated multiple times; the memory is distributed and there are two memory controllers.

xSTream

The STMicroelectronics xSTream architecture is based on the convergence of communication and computing as a way to solve scalability and programmability of high-performance embedded functionalities, such as graphics, multimedia and radio subsystems.

The following image illustrates a high-level view of a complete system embedding an instance of the xSTream processor fabric.



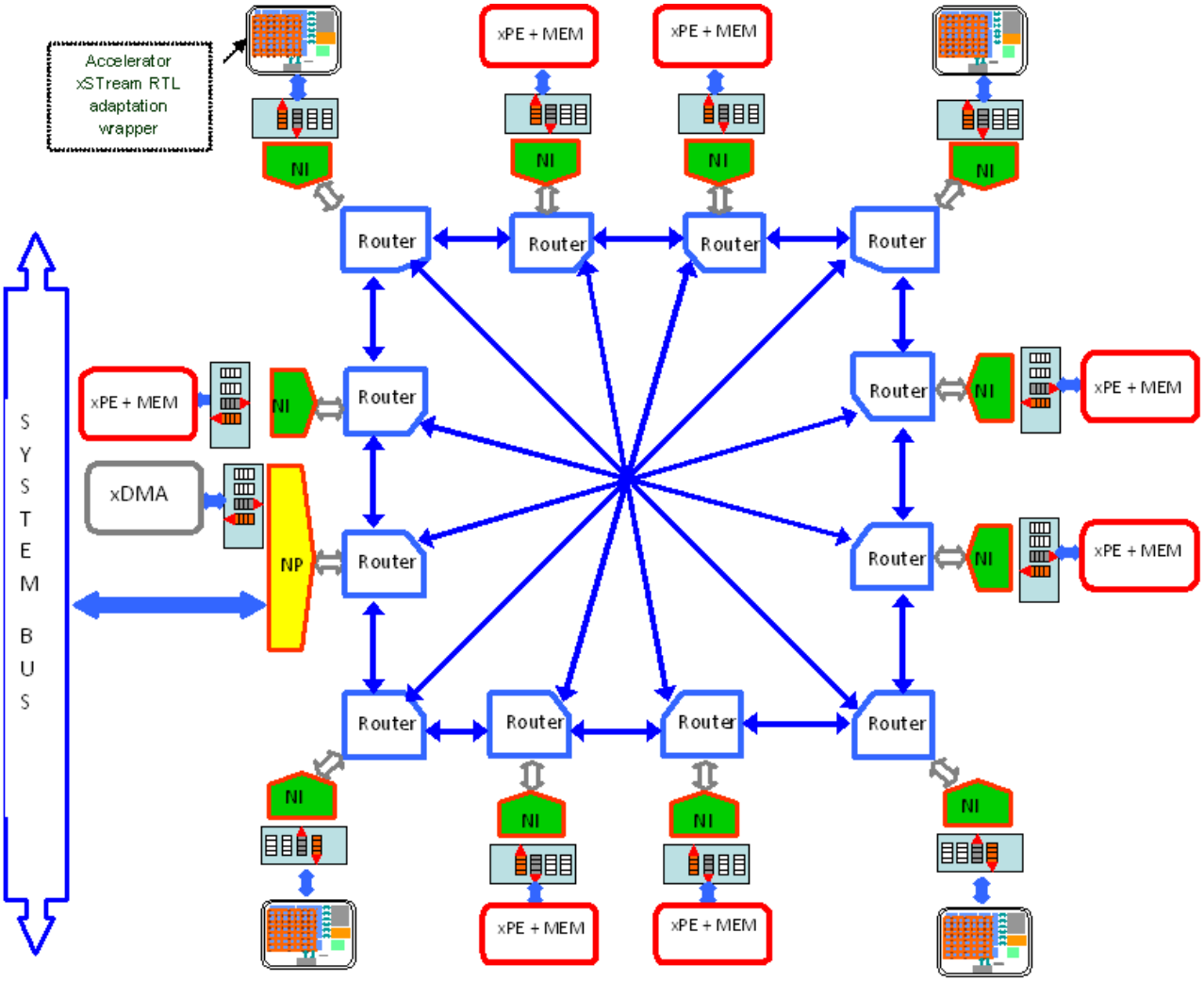
High-level view of an embedded system containing an xSTream computing fabric.

The system is composed of the traditional “host processing” part on the left side, which we have depicted as a Symmetric Multiprocessing (SMP) subsystem, while the entity on the top-right end of the picture above is the ‘streaming engine’ of the xSTream architecture. It’s meant to be addressing the needs of data-flow dominated, highly computational intensive semi-regular tasks, typical of many embedded products. The streaming nature of the kernels mapped onto it makes it possible to design a semi-regular fabric of programmable engines interconnected via a relatively simple network of point to point channels. The fabric supports a number

of simultaneous software pipelines running on the processing elements as to accommodate complex applications and also provide load balancing and latency hiding capability. A property of the streaming fabric is to support very high internal data bandwidth, throughput and computationally intensive tasks.

The processing elements of the streaming fabric (XPEs) are relatively simple programmable processors or engines with a general purpose but simple basic ISA that can be extended with SIMD or Vector mode instructions. They execute instructions fetched from local memories instead of caches, a great simplification at the pipeline forefront. Local memory is also used for wide data accesses. The engines are connected between them, and the interconnect functionality plays one of the critical roles in this picture. In fact it's quite the essence of the system to be able to provide a self synchronizing support for software pipelines. This is achieved with a set of lightweight routers, very similar to the ones being defined for network-on-chip replacements of standard bus infrastructures; but with more freedom for simplification, due to the constrained nature of the communication patterns versus a generic system back-bone NoC.

There are a lot of opportunities to improve this architecture with the gate extraction tools. The host processor is quite regular and supports SMP, the fabric is composed of a large number of replicated cores, each one with local memory. The XPEs have SIMD capabilities for 4 and 8 way vectors, which can allow the optimizing tools to extract regularities in the core pipelines.



An instance of an xSTream computing array

An xSTream symmetrical array with custom accelerator nodes and a single xDMA node, the architecture supports the definition of a hybrid scheme to better suite specific application domains, each node could then be associated with a different kind of computational kernels (e.g. block-matching for motion estimation, DCT/IDCT, FFTs, etc.)

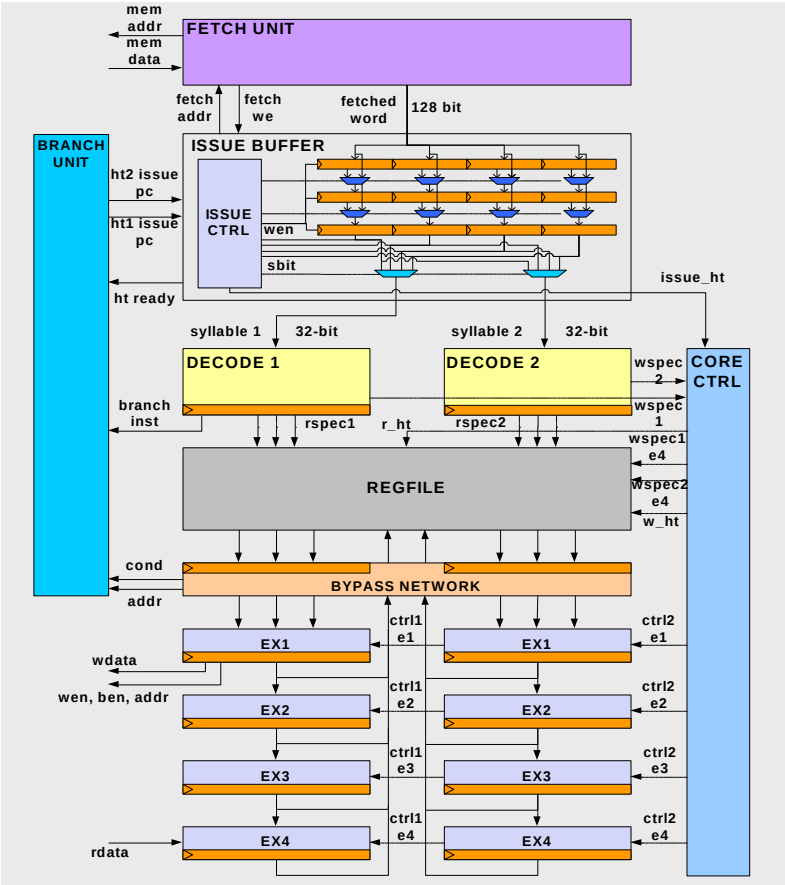
Programmable Processing Element

The processing engine for the demonstrator can be identified in the initial phases of the project functional specification definition, one suitable candidate that was designed with the built-in flexibility for this is the xPE.

The xPE is optimized for stream oriented, data-flow dominated, performance demanding computation, and directly draws many of it's features from the company line of VLIW embedded media processors; even so, it's not a full fledged microprocessor (it can't run a general purpose OS for example). During its design the focus was entirely put on reaching very high compute densities by way of stripping most of the control complexity associated to general purpose cores, and devoting

those gates to useful functional units instead. In addition the xPE is design time parametric and can be instantiated in a number of variants each supporting a different degree of features and capabilities, some of the key aspect of the xPE microarchitecture are:

- Highly streamlined and “minimalist” core architecture to tune system frequency and limit core size.
- VLIW architecture to exploit ILP.
- Configurable VLIW slices which can be chained up to a VLIW cluster.
- Design time configurable to accommodate customized solutions.
- Local memories and interfacing optimized for managing and accessing data streams.
- Vector instructions operating on packed data words to exploit available DLP.
- Fine-grained multithreading to exploit task level parallelism.
- Advanced power saving features for mobile devices.



xPE slice pipeline supporting time-slice multithreading and SIMD extensions

A typical xPE instance summary for the main features of the implementation is as follow:

- Time-slice multithreading with up to 4/8 threads
- Zero latency thread switching
- Standard cell based register file
- 32 128-bit vector GP registers per thread
- 64 32-bit scalar GP registers per thread
- Single slice, 2 issue machine with 8x16-bit or 4x32-bit vector operations per lane
- 9 stage pipeline, 4 stage execution pipeline
- 4x32-bit or 8x16-bit multiplications per cycle
- 128-bit load/stores
- Vector permutation and conditional sub-half register file writes

Theoretical peak performance with 16 bit SIMD operations is 12 Gops in CMOS65LP at a worst case frequency of ~750 MHz with only SVT/HVT libraries

TOTAL AREA REGFILE + CORE (4 threads):

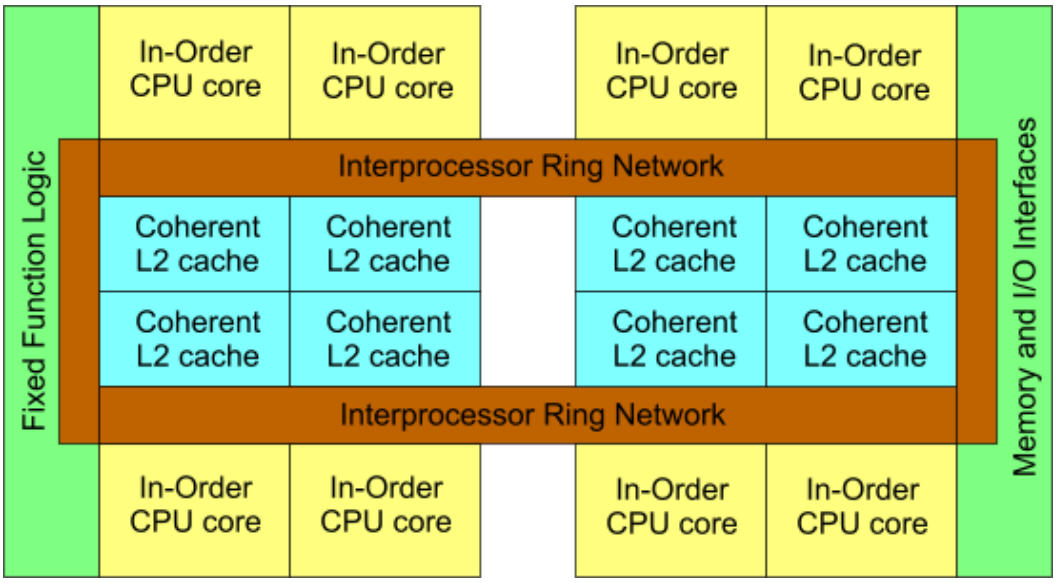
Core:	0.36 mm2
Std.cell Register File:	0.26 mm2
(Custom	~0.10 mm2)
Total Size 4 threads:	**0.62 mm^2**
Total Size 1 thread:	**0.4 mm^2**

If the targeted technology node for the project benchmarking is CMOS40, it's expected that the xPE core size would be around 0.35 mm2 for a four threads instance and somewhere around 0.20 mm2 for a single threaded one.

Larrabee

Larrabee [Larrabee08] is a new many-core visual computing architecture from Intel. It uses multiple in-order x86 CPU cores, each augmented by a 16-wide vector processor unit.

Cores communicate through a high bandwidth interconnect network; they have two separate units, one scalar and one vector unit that communicate through memory. It supports the standard Pentium processor x86 instruction set and so can execute general purpose code.



Larrabee many-core architecture (Source: http://www.viznet.ac.uk)

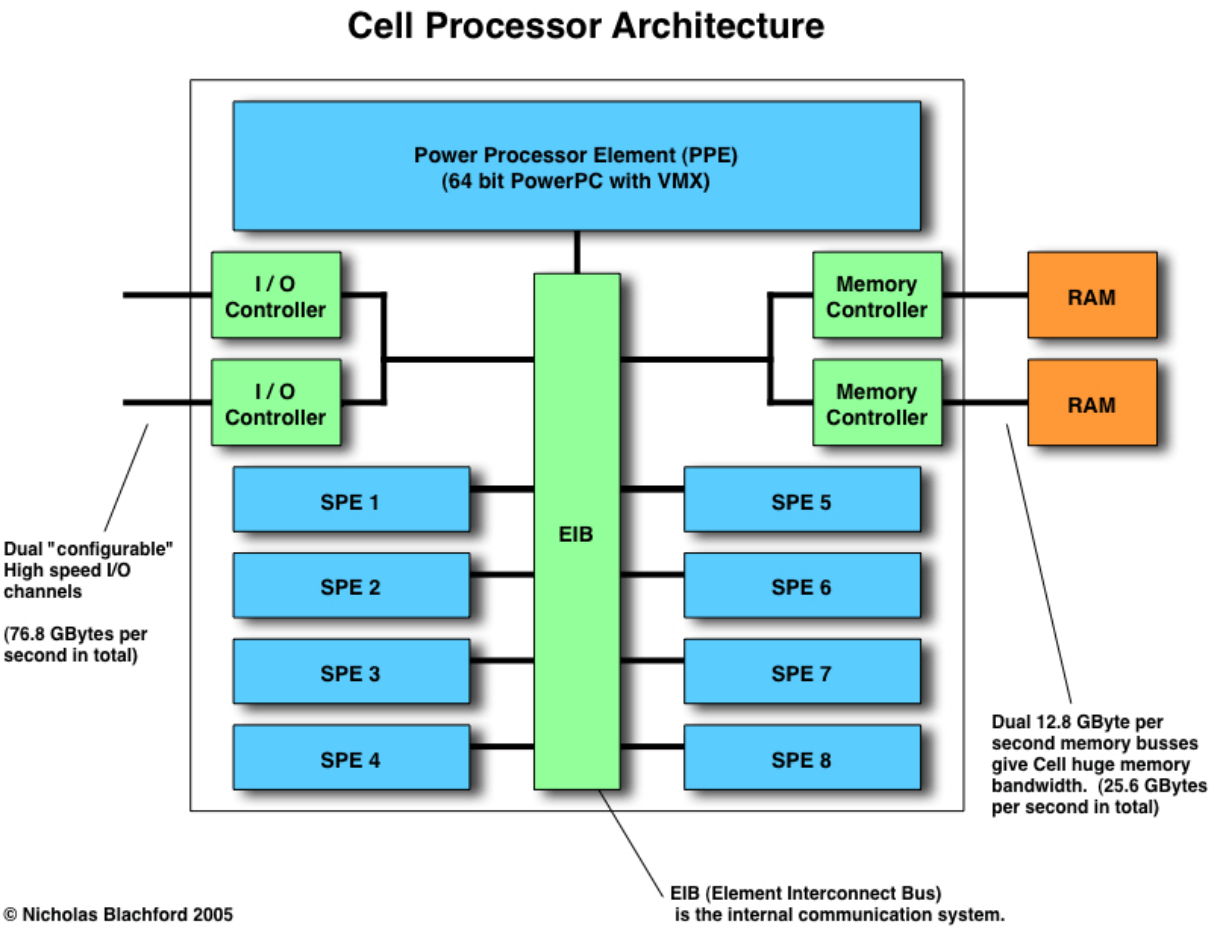
In the previous image we can see the schematic of the architecture with all the cores connected with a bidirectional ring network and L2 coherent cache reserved, for fast accesses, to each core and two memory controllers, one on each side of the chip.

The core replication, the distributed memory and the communication network ring give the extraction tools few opportunities to optimize this architecture.

Cell

Cell [Cell05] is a microprocessor architecture jointly developed by Sony Computer Entertainment, Toshiba, and IBM, an alliance known as "STI".

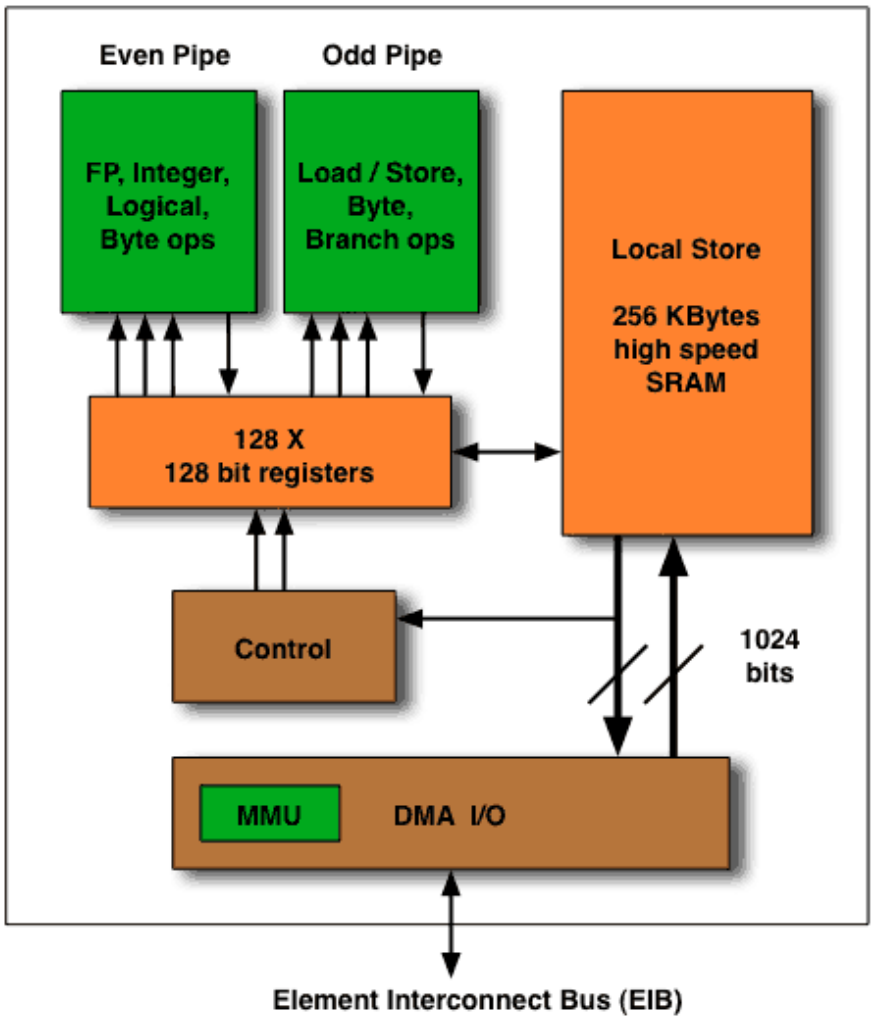
The processor has four main components: external input and output structures, the main processor called the Power Processing Element (PPE), eight fully-functional co-processors called the Synergistic Processing Elements, or SPEs, and a specialized high-bandwidth circular data bus connecting the PPE, input/output elements and the SPEs, called the Element Interconnect Bus. The Cell is a heterogeneous multi-core processor capable of control-intensive processor and compute-intensive SIMD processor cores.



Cell Architecture (Source: http://www.blachford.info/computer/Cell/Cell1_v2.html)

As we can see in the previous image it is composed by one PPE, a conventional microprocessor core, and eight SPEs, each of them a RISC processor with 128-bit SIMD organization for single and double precision instructions.

The multiple instantiations of the SPE with its SIMD support can be optimized successfully with the gate extraction tools.



Synergistic Processor Elements (SPEs)

Each Cell contains 8 SPEs. An SPE is a self contained vector processor which acts as an independent processor. They each contain 128 x 128 bit registers, there are also 4 (single precision) floating point units capable of 32 GigaFLOPS and 4 Integer units capable of 32 GOPS (Billions of integer Operations per Second) at 4GHz. The SPEs also include a small 256 Kilobyte local store instead of a cache. According to IBM a single SPE (which is just 15 square millimeters and consumes less than 5 Watts at 4GHz) can perform as well as a top end (single core) desktop CPU given the right task. Like the PPE the SPEs are in-order processors and have no Out-Of-Order capabilities. This means that as with the PPE the compiler is very important. The SPEs do however have 128 registers and this gives plenty of room for the compiler to unroll loops and use other techniques which largely negate the need for OOO hardware

3.1.2. Massively Parallel Processor Array architectures

Massively Parallel Processor Array (MPPA) architectures are MIMD computing systems with very large numbers of small processors that function asynchronously and independently. The more or less sophisticated computing nodes are often based on a single or very few different processor architectures interconnected by a mesh or hypercube network.

Homogenous systems which are based on large numbers of identically replicated functional blocks have the advantage that the system design, block verification, and programming models can be simplified. This fact is also valid for a compound design flow which gets better manageable while keeping the potential gain worthwhile.

Due to memory bottlenecks and synchronization requirements among the cores, the communication network and data exchange infrastructure are the key elements and main differentiation factor for a multiprocessor system. Therefore, we distinguish in the following between MPPA architectures with distributed memory and systems with shared memory resources.

MPPA architectures with distributed memory

In distributed memory architectures the available memory resources are integrated into or close to each processor node. Thus, the memory architecture is divided in subsystems which are replicated and integrated together with the computing node into the regular array.

Due to the absence of an inherent system memory coherency, the system synchronization has to be handled at software level and data exchange among the processors at low latency with high bandwidth is of high importance. Ideally every node would have a direct point-to-point link to any other node to avoid resource conflicts and communication bottlenecks. However, for a large processor array this is unfeasible due to excessive complexity and un-routable networks.

Many distributed memory MIMD architectures have a reduced but regular structure of interconnection links to provide the required communication infrastructure. Some of these networks follow a regular pattern such as interconnections to all neighbors in a two dimensional mesh which results in a highly regular structure.

Some MPPA architectures include dedicated router devices that are replicated in large numbers to organize the data transportation among the processors and to external IO devices. Data streams are routed with multiple hops across routers to the destination to enable communication among distant nodes.

Distributed data routers or regular point-to-point links with routing capabilities in the processing nodes are ideal for a compound gate optimization design flow. In this

case, the circuits for the communication links are identically replicated in large numbers and can be easily handled with a compound gate optimization flow. This is especially useful due to the importance of the communication infrastructure for the overall performance of MPPA systems.

MPPA architectures with shared memory

Shared memory architectures provide a consistent view on a unique memory space to all computing nodes. The required data access synchronization can be guaranteed by centralized and shared storage unit. However, for large numbers of processing nodes shared memory architectures suffer from inefficiencies due to memory resource conflicts, access latencies, and bandwidth issues.

Thus, shared memory systems are commonly enhanced with memory hierarchies and cache architectures to reduce the access latency and increase data bandwidth. With the integration of caches into the processing elements a part of the memory system is actually distributed into the processing elements and is identically replicated for all nodes.

Cache synchronization mechanisms are implemented to generate a virtually coherent view of the shared memory space for all processing nodes. Distributed cache snooping protocols (e.g. MESI protocol) require sophisticated functionality in the computing nodes. In contrast to centralized cache synchronization mechanisms the memory coherency functionality is handled by replicated blocks which are integrated into the processing nodes.

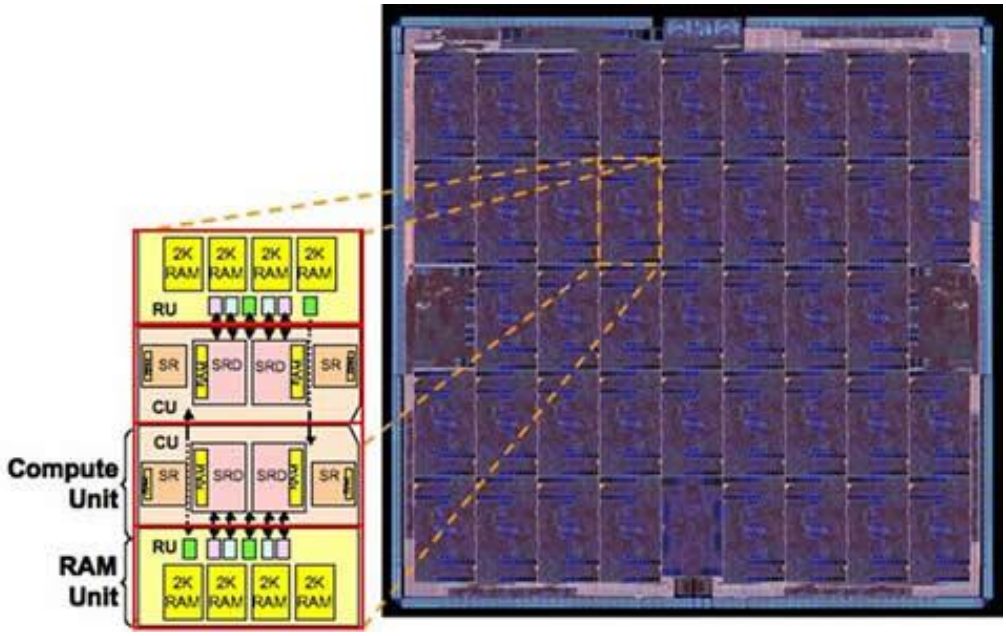
Distributed memory hierarchies and cache synchronization mechanisms are well suitable for a compound gate design flow due to focused optimizations on rather small functional blocks which are identically replicated for all computing nodes.

3.1.3. MPPA Implementation Examples

In the following sections several MPPA systems with many more or less sophisticated software programmable processing elements are discussed. The level of system regularity with extensive replication of key components in the system is analyzed. The focus is to identify highly regular architectures with small and identically replicated building blocks that are potentially attractive for a compound gate optimization flow.

Nethra - Ambric

The company Nethra has developed the MPPA architecture Ambric [Ambric06] which integrates 336 light-weight 32-bit fixed-point RISC processors on the AM2045 device. Two types of processors, one optimized for DSP applications and one extended with specific data streaming operations are clustered with local memories in groups of 8 processors. These units are called bricks and form the building blocks of larger arrays. The bricks are stacked in a two-dimensional mesh and interconnected with a communication infrastructure. The processors run at a system frequency of 350 MHz and include 2KByte of local storage for each processor.



Ambric chip architecture (Source: http://www.nethra.us.com)

The very high regularity of the Ambric system architecture makes it an ideal candidate for the compound gate optimization approach. The optimization of a small and streamlined processor data-path should be easy manageable with a promising impact on the overall system performance due to the identical replication of more than hundred times in the system.

Further, the brinks include local communication networks and memories controllers which can be included into the gate optimization procedure. Due to the regular stacking of the bricks an improvement on a single communication link could have a significant impact on the communication infrastructure of the whole processor array.

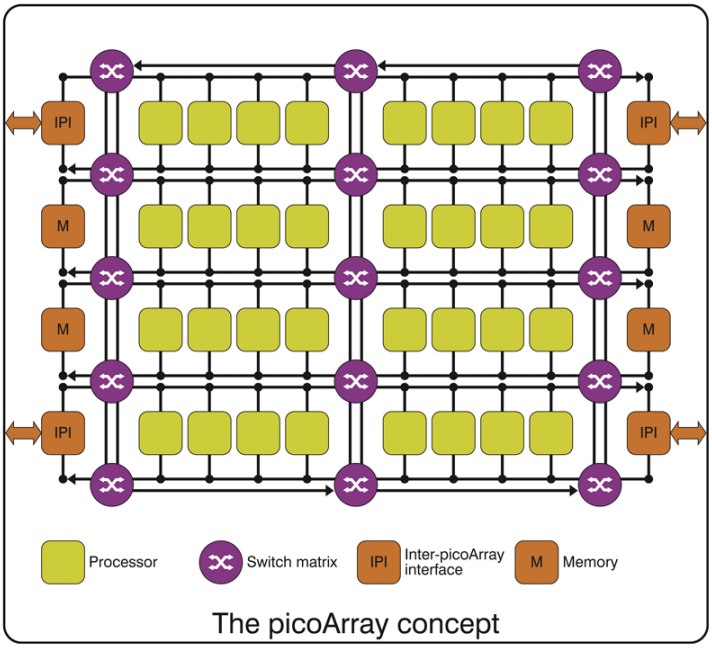
Picochip - picoArray

The picoArray [Pico03] is a multi-core processor system for signal processing applications developed by Picochip. Systems with up to 273 individual processor nodes can be implemented.

All processors consist of a basic building block with a 16-bit wide Harvard architecture supporting three-way long instruction words (LIW). Further, the core can be enhanced with local memories, functional accelerator units, and hardwired execution units for specific applications.

Application specific functional accelerators for correlation and path metric calculations, for turbo code (CTC), FFT/IFFT, Viterbi, Reed Solomon, and cryptographic calculations are available and can be integrated in the picoArray.

The picoArray processor nodes are interconnected by a two-dimensional grid of 32-bit buses and programmable bus switches. Additional IO and communications facilities are connected to the grid and are used to interface to external input/output devices or an external host processor system. Currently, there are also SoC products available that integrate the PicoArray with an ARM926 host processor on the same die.



PicoArray chip architecture (Source: http://www.picochip.com)

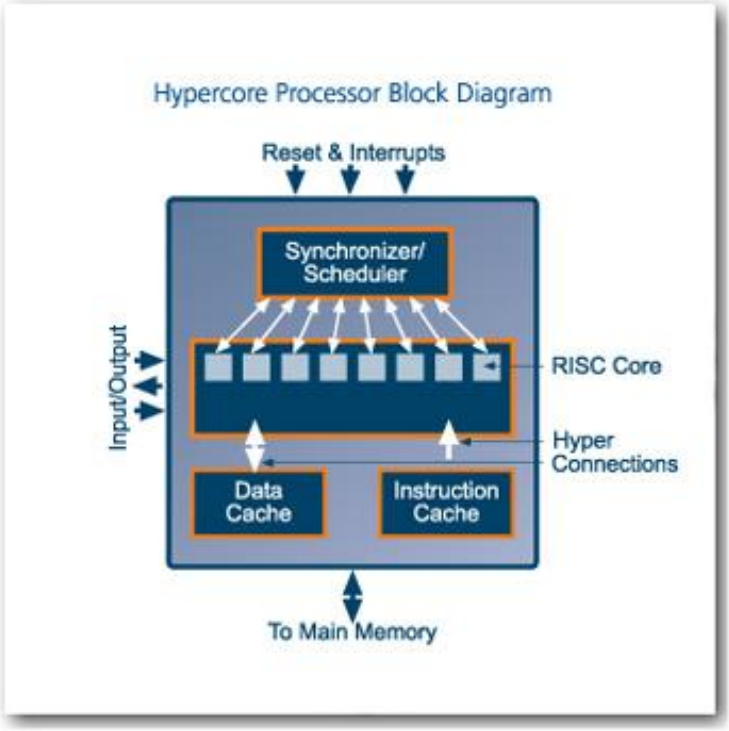
The PicoArray is a highly regular structure with a large number of replicated basic building blocks. A performance improvement of these building blocks due to a compound gate optimization could enable a high potential gain for the overall system performance. Further, there is a regular communication infrastructure that bases on replicated router devices which could be included into the optimization

procedure. However, there is also a significant grade of irregularity in the system due to the large variety of different application specific accelerator components. A compound gate optimization of these components would be more intricate.

Plurality –Hypercore

Plurality's Hypercore is a general-purpose multiprocessor system with 16 to 256 cores. The cores are 32-bit wide RISC processors which have access to a common shared memory space. A centralized hardware scheduler unit distributes and assigns program tasks to the processors. It further synchronizes the task execution and balances the load on the processor cores dynamically. A task map is used to guide the hardware scheduler for an optimal load distribution on the system.

All processing nodes are connected to a centralized shared memory system with a cache subsystem which provides to all cores a coherent view of the available storage resources. Plurality claims that the interconnection network and storage blocks provide non-blocking access to the shared memory resources for all cores in the system without bottlenecks.



Hypercore chip architecture (Source: http://www.plurality.com)

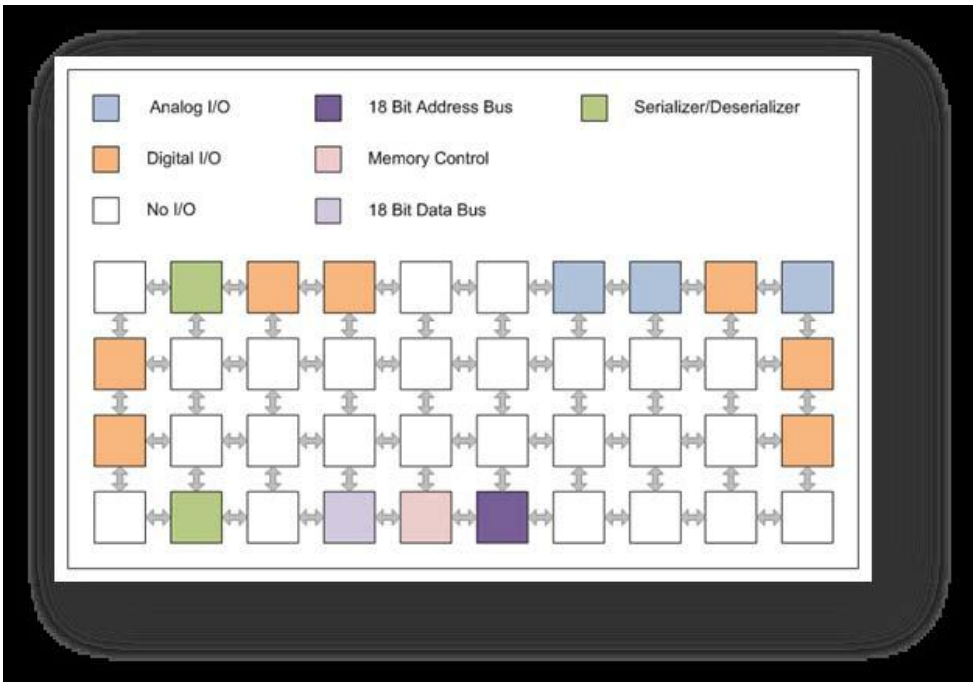
The Hypercore provides a medium level of regularity. Regularity can mainly be found with the replicated RISC cores whereas the synchronizer/scheduler and storage units are centralized blocks with no replication. However, due to the fact that these

centralized units have interfaces to many identical processor nodes it can be assumed that many “sub-blocks” and interfaces within these centralized units are actually replicated building blocks.

Further, one can speculate that many performance critical circuits are located in these two centralized system hot spots. In this case a compound gate optimization on replicated “sub-blocks” and/or specifically mission critical circuits in these units would probably lead to a significant improvement of the overall system performance.

Intellasys - SEAforth

The SEAforth [Intellasys06] multiprocessor systems has been developed by Intellasys. The 40C18 device includes an array of 40 homogeneous 18-bit processors with dedicated ROM, RAM and inter-processor communication interfaces for each processor. The SEAforth has been implemented with an asynchronous circuit design approach to improve the power dissipation and processing speed. The cores use no asynchronous communication method bits or handshake lines but communicate with their neighbor cores via dedicated ports which are used to trigger actions if data is available. The SEAforth chip includes beside the processor array several A/D and D/A converters, I/O interfaces and ring oscillators for clock generation.



SEAforth 40C18 chip architecture (Source: http://www.intellasys.net)

The SEAforth system is build of a large number of identical processors which means that a focused data path optimization on a single processor with the use of a dedicated compound gate library would potentially have a significant impact on the overall system performance.

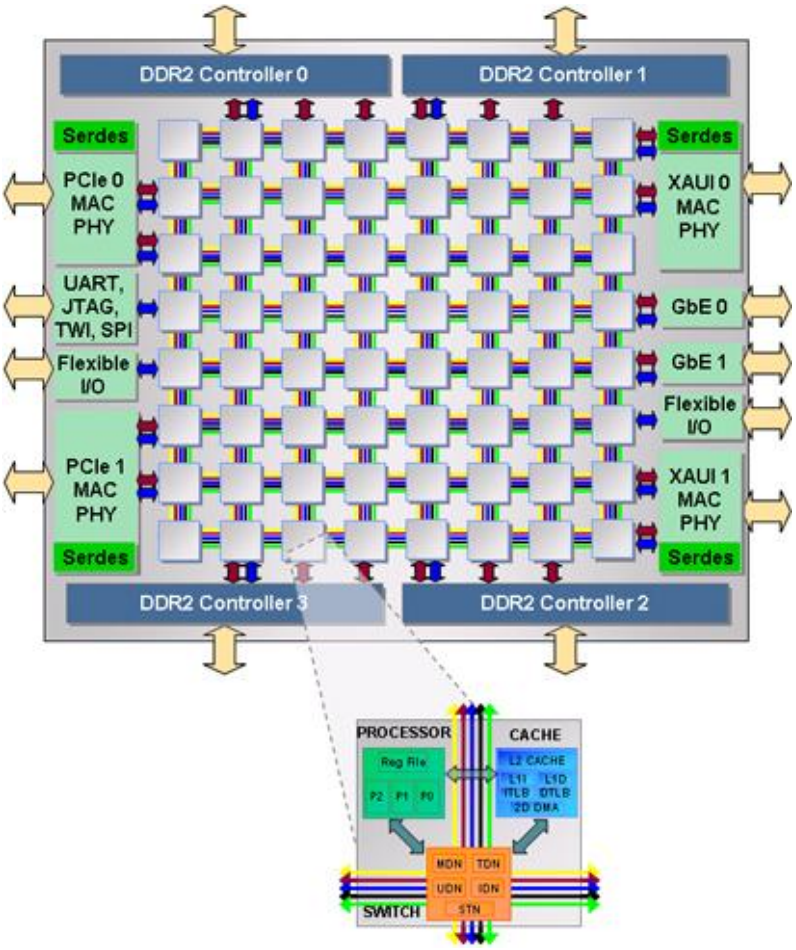
However, due the fact that asynchronous design methods were applied to improve processing performance and reduce power dissipation, the required compound gate extraction process would be far more complex with a hardly predictable gain. Even if one can argue that a compound gate optimization is feasible and theoretically with a potential performance gain comparable to synchronous circuit architectures it has to be left for subsequent studies.

Tilera –tile64

Tilera, a startup company and MIT spin-off provides the TILE64 SoC [Agarwal07] which is based on the technology of the academic RAW project.

64 processors are arranged with router nodes and local cache mechanisms in a two dimensional mesh. Each core has a short three issue in-order pipeline with a VLIW instruction set.

The router nodes provide the communication functionality to exchange data among the cores in the network. Tilera claims that a fully coherent caching mechanism based on a local L1 cache and a distributed L2 cache is providing higher efficiency and concurrent access from multiple cores. Compared to a large centralize L2 or L3 cache access bottlenecks are avoided and power efficiency is improved.



Tilera chip architecture (Source: http://www.tilera.com)

Due to the distributed cache architecture and the large number of equivalent processor cores Tilera til64 is a very regular and homogeneous architecture which makes it an ideal target for a compound gate optimization design-flow.

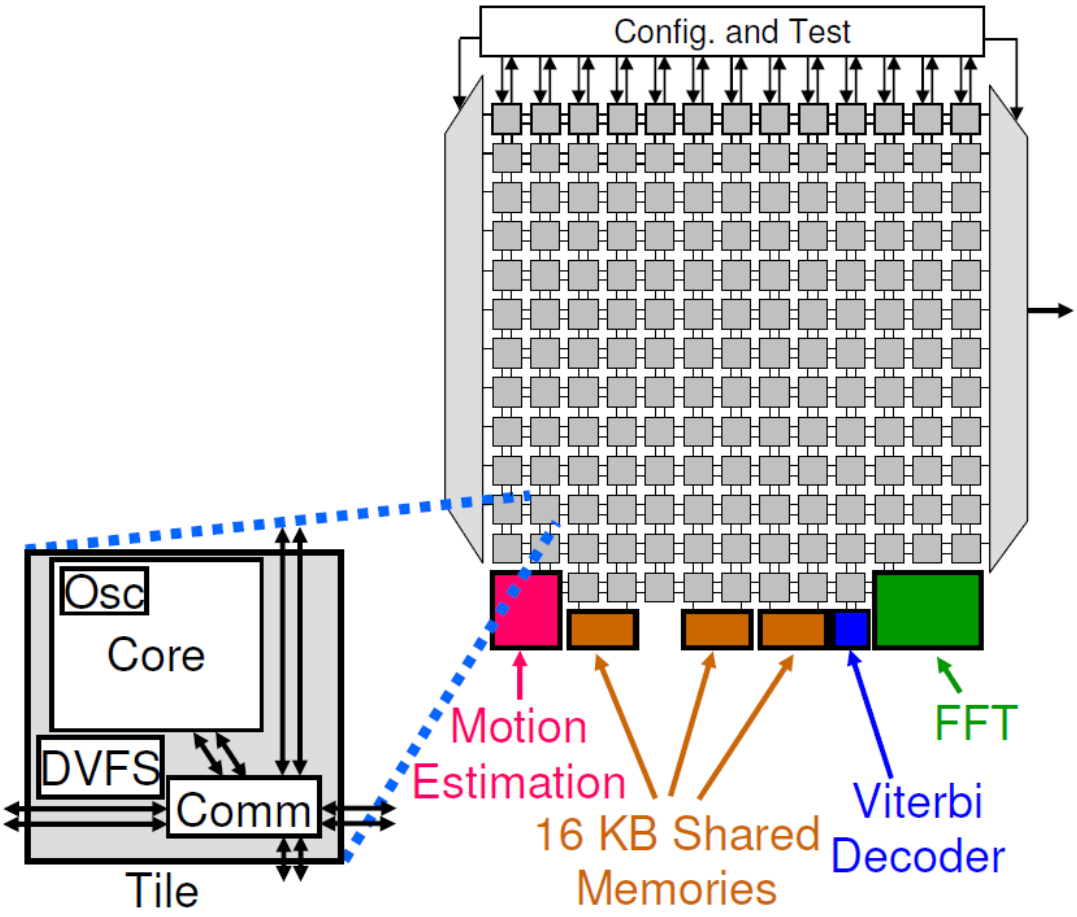
The communication infrastructure is build of replicated data routers and highly regular interconnection links. Further, the caching mechanism is distributed and uniform for all cores. A compound gate optimization on a single package containing the processor data-path, the cache architecture, and the local router would be sufficient for a complete optimization of the Tilera til64 processor array.

AsAP Asynchronous array of simple processors

The Asynchronous Array of simple Processors (AsAP) architecture comprises a 2-D array of reduced complexity programmable processors with small memories interconnected by a reconfigurable mesh network. AsAP was developed by researchers in the VLSI Computation Laboratory (VCL) at the University of California, Davis and achieves high performance and high energy-efficiency, while requiring a relatively small circuit area.

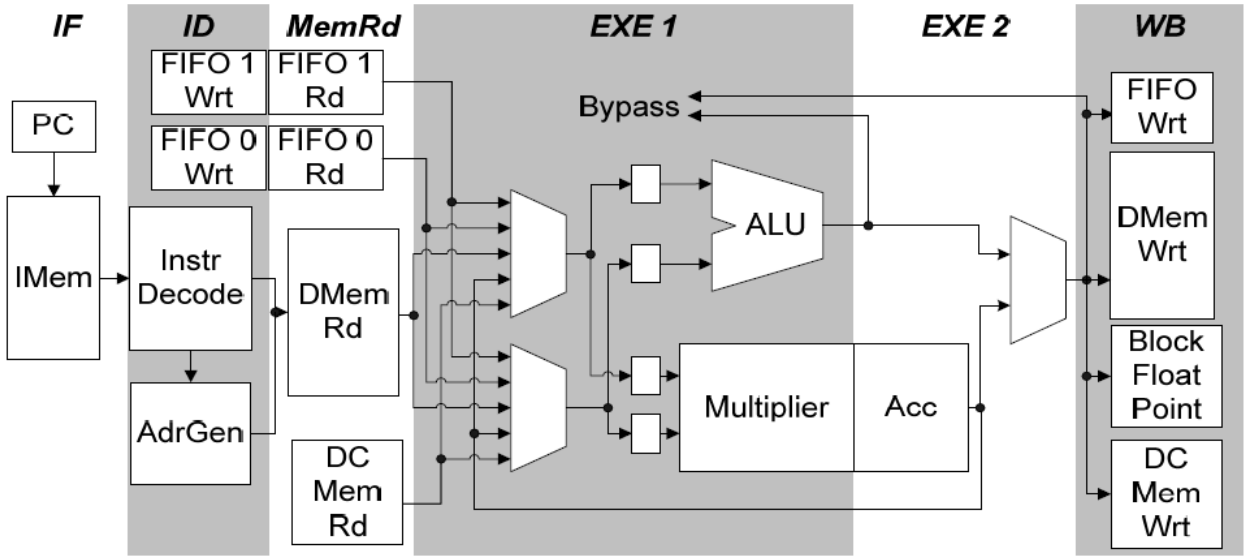
AsAP processors are well suited for implementation in future fabrication technologies, and are clocked in a Globally Asynchronous Locally Synchronous (GALS) fashion. Individual oscillators fully halt (leakage only) in 9 cycles when there is no work to do, and restart at full speed in less than one cycle after work is available. The chip requires no crystal oscillators, PLLs, DLLs, or any global frequency or phase-related signals whatsoever.

The multi-processor architecture efficiently makes use of task-level parallelism in many complex DSP applications, and also efficiently computes many large tasks utilizing fine-grain parallelism.



A 167-processor 65 nm computational platform well suited for DSP,communication, and multimedia workloads [Baas08]

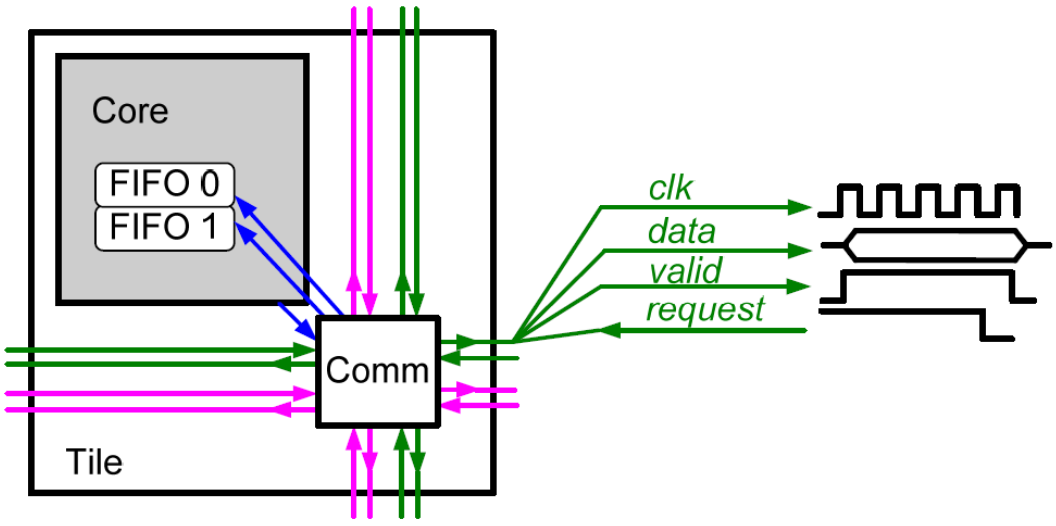
A test chip was manufactured that contains 164 programmable processors with dynamic supply voltage and dynamic clock frequency circuits, three algorithm-specific processors, and three 16 KB shared memories, all clocked by independent oscillators and connected by configurable long-distance-capable links



16-bit datapath with RISC-like instructions, 128x16-bit data memory (DMEM), 128x35-bit instruction memory (IMEM) Two 64x16-bit FIFOs for inter-processor communication

Inter-Processor Communication

Circuit-switched source-synchronous communication, each link has a clock, 16-bit data bus, valid, and request. Core can write to any combination of the 8 outputs under software control and read from any 2 of the 8 inputs using statically configured FIFOs



AsAp Communication node for a single core element

3.1.4. Conclusion

Many MPPA architectures include large numbers of identically replicated processor cores, memory system components, and communication units. The generation of an optimized compound library is more efficient and less complex on small replicated units than on large centralized circuits. Therefore, systems that include beside the processor arrays also communication networks or caches architectures build of identically replicated building blocks are specifically attractive for a compound gate optimization flow.

All presented MPPA systems are promising targets with potential performance improvements that scale with the number of replicated cores. Architectures such as the tile64 from Tilera with replicated distributed cache components or systems with very large numbers of tiny processor nodes such as the PicoArray or Ambric provide significant advantages for the proposed design flow.

3.2. Application specific ultra wide instruction architectures

In the recent years several application specific processor architectures have been presented that operate with very wide instructions and large number of applications-tailored functional units in parallel. Previously, applications such as OFDM base-band processing or multi standard TV applications could only be handled by dedicated hardware circuits and were far out of reach for any programmable processor system.

Such application specific ultra wide instruction architectures often include multiple register files, local memories, complex–value functional units, customizable data width, and a large interconnection network. Due the exponential growth of the complexity with a large number of units with input and output ports it is not feasible to provide a fully connected bypass and interconnection network. From the regularity criteria point of view this means that the functional units in the core are replicated with a regular structure whereas the interconnection of these units is often rather irregular. However, due fact that some functional units are replicated in large numbers we consider these architectures worthwhile for an investigation.

3.2.1. Implementation examples

Two examples of application specific, ultra wide instruction set architectures are presented in the following sections. The focus is on the architecture regularity and the applicability of a compound gate optimization flow. Issues such as programmability, flexibility, and tool support are not part of this study.

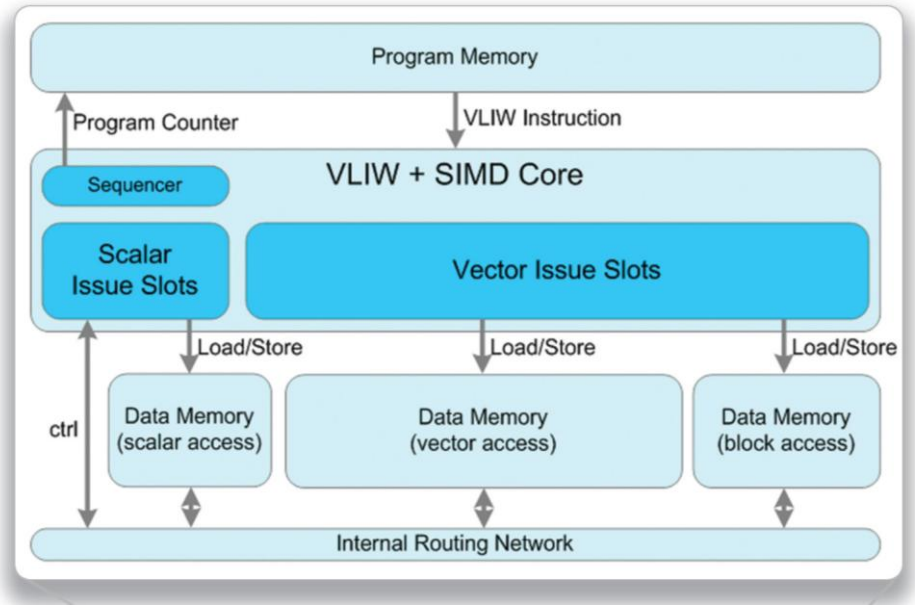
SiliconHive - HiveFlex

The HiveFlex, the successor of the AVISPA core family is a set of Very Long Instruction Word (VLIW) cores provided by the Dutch company Silicon Hive. The Hiveflex cores can be configured and optimized at design-time for application domains such as OFDM base band, multimedia, and multi-standard digital television processing.

The HiveFlex VSP, CSP, and ISP are sub families targeting specific target applications. The ISP series for example is optimized for image processing applications with a separate scalar and vector data-path. The scalar data-path is mainly used to control the program execution flow with a simple RISC instruction set. The vector data-path

includes lots of functional units, register-files, interface modules, and local memories which are controlled by a sequencer. A single instruction stream with very wide instruction words and up to 25 issue slots is decoded by the sequencer to control the operations and interconnections of the functional units and/or storage elements. Some of the functional units can handle complex numbers or multiple data sub words with very wide SIMD operations. Some cores allow a configurable SIMD width of up to 128 ways for a single functional unit.

A local network interconnects all components in the system. The network is not providing a full connectivity (which means every output is connected the every input of all units) to keep the system complexity within reasonable limits.



HiveFlex architecture(Source: http://www.siliconhive.com)

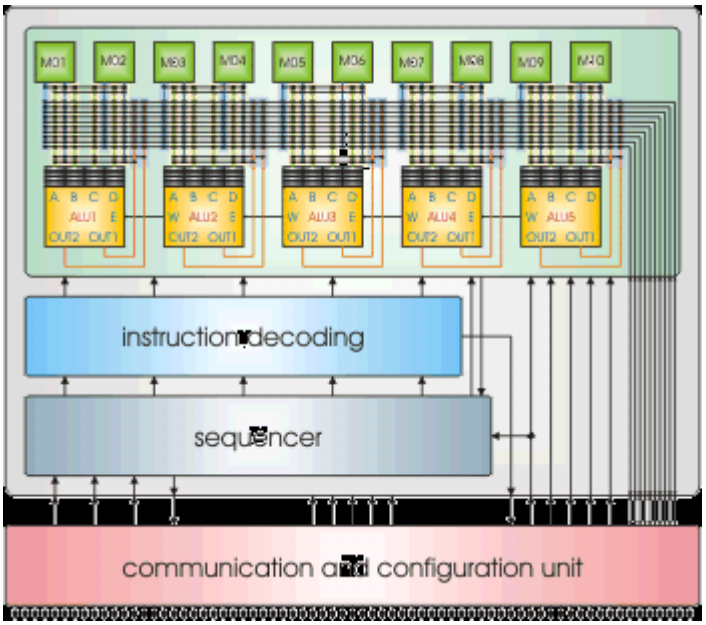
At the first sight, the HiveFlex core's seam to have a rather irregular structure with many application driven optimizations and customized interconnection networks. But keeping in mind that many of the functional units and local storage elements are actually replicated building blocks, a certain level of regularity which can simplify a compound gate optimization could be found.

Further, there are very wide, design-time configurable SIMD operations supported which means that certain sub-blocks such as small ALUs or multipliers are replicated in large numbers within a single functional unit. A compound gate optimization on such components can be reutilized extensively with a potentially great impact on the maximum speed and processing throughput.

Recore Systems - Montium

The Montium [Montium03] Tile processor is a programmable architecture for fixed-point digital signal processing algorithms developed by Recore Systems. The processor data-path includes many local memories and functional units interconnected by a reconfigurable communication network. The data-path width and memory capacity is customizable at design-time. The functional units support various signed integer and fixed-point arithmetic operations with local registers to store intermediate data.

A central program sequencer controls the functional units and interconnection network with a small configuration binary of about 1 kByte which can be updated in less than 5 us. The communication and control unit provides the interfaces to external devices.



Montium architecture (Source: http://www.recoresystems.com)

The Montium Tile processor has a very regular structure with replicated functional units and storage elements. Further, the communication network has an orderly composition interconnecting all functional units in a similar fashion. For this reason, the Montium Tile processor could be a promising target for a compound gate optimization design flow.

The reconfiguration of the interconnection network of the Montium Tile processor resembles more an FPGA than a processor data-path. The interconnection is not updated every clock cycle as it is commonly the case in a processor pipeline and takes with 5 us a rather long period of time.

Compound gates could potentially optimize critical multiplexer circuits and configuration storage cells in the interconnection network and therefore improve the processing and reconfiguration speed of the system.

3.2.2. Conclusion

Typically, application specific ultra wide instruction architectures include many identical functional units and local storage blocks. Therefore, a compound gate design flow optimization on a small block could lead to a greater improvement of the overall processor due to the identical replication of the same block. Further, many architectures have functional units which support vector arithmetic and SIMD operations which provide regulatory and replication on the sub-block level of the functional units.

However, the communication networks among the functional units and storage elements are often tailored to specific application requirements and therefore rather irregular. Many of these architectures include centralized controllers and interfaces which would need a dedicated analysis.

As a conclusion we can summarize that application specific processors provide regularity mainly at the functional unit level where a compound gate level optimization flow could lead to some improvement for the overall system performance.

3.3. Fine Grained regular computing

Power dissipation of integrated circuits is a major issue to continue device scaling of CMOS technology [Fra01]. As a result, parallel architectures [Bor07] have been introduced as a way to deal with increasing power dissipation thanks to their better power efficiency than General Purpose Processors (GPP). The regularity that is often exhibited by these architectures could also be a great opportunity to cope with the extreme process variations of new technologies [Bor05]. Consequently, they are of great interest for the Synaptic project. The parallelism has been introduced at all the levels of granularity of an architecture. For fine-grained architectures, we can distinguish two approaches:

Single-Instruction Multiple-Data (SIMD) architectures are programmable architectures that are still based on a fetch|decode|execute|write-back instruction pipeline. However, a single instruction can execute the same operation on multiple data in this kind of architecture. It takes advantage of the data parallelism of an application to reduce the control overhead of the fetch-decode of instructions. The control part is shared between multiple execution units in these architectures.

Reconfigurable Computing architectures are based on a data-flow execution model. They are roughly composed of a set of processing resources interconnected by a configurable network. So, an operation sequence can be executed on a set of data, generally in a pipelined mode, with a reduced overhead control.

The better efficiency (performance density, power efficiency) of these architectures is generally gained at the cost of a loss flexibility. The common philosophy behind these two approaches is to reduce the part of the chip dedicated to control, and to devote a bigger part of the chip to the operating part. It maximizes the peak processing power, but in turn this processing power is more difficult to exploit. They are thus generally restrained to an application domain, or to a type of processing.

3.3.1. Massively parallel Single-Instruction Multiple-Data architectures

The memory bandwidth and the limited scaling of Instruction Level Parallelism (ILP) are two major bottlenecks of single-core processors. In order to also take advantage of data-level parallelism, SIMD extensions are sometime available in common general-purpose processors [Die98][Obe99][Die00]. But, these extensions are mostly short vector (SIMD) processing engines. We focus here on massively parallel SIMD architectures. These architectures scale efficiently thanks to a scattering of data in many small memories, which allows to scale the internal memory bandwidth. The more they are scaled, better is their peak power efficiency as the

control overhead is shared between more processing units. All the more, they are characterized by their high regularity, which is our primary concern.

The Ter@pix processor

Processor architecture

The Ter@pix processor [Bon08] was designed by Thales Research & Technology for image processing application that must conciliate strong requirements in term of high processing power, limited size, limited power consumption, limited production volumes and long lifecycles. The low production volumes and long lifecycles of the targeted applications have motivated the selection of a Xilinx FPGA [Xilinx07] as the underlying technology. For products with long lifecycle, the risks of severe procurement issues and redesign costs all over the lifecycle of the system are the primary concern. An architecture built on the top of a FPGA can be expected to be reproduced more or less the same during years on successive generations of FPGA and hence benefit from increasing performances due to technology evolution without requiring a too costly effort. Compared to the traditional FPGA approach consisting in application specific designs, the two main advantages of a programmable architecture on FPGA are first a dramatic reduction in development costs, evolution costs and time to market and second the capability to implement sophisticated data dependent processing algorithms.

The Ter@pix architecture is organized in three subsystems (see Figure 1):

- A general-purpose processor (a MicroBlaze softcore [Xilinx08]) controlling the whole platform and executing the main application code.
- A SIMD processing unit acting as a coprocessor and executing a set of basic image processing operators (e.g. erosion, dilation, Sobel).
- A communication unit, called Data Mover Unit (DMU), in charge of the data transfers between the two external memories (i.e. storage RAM) and the internal memories of the chip.

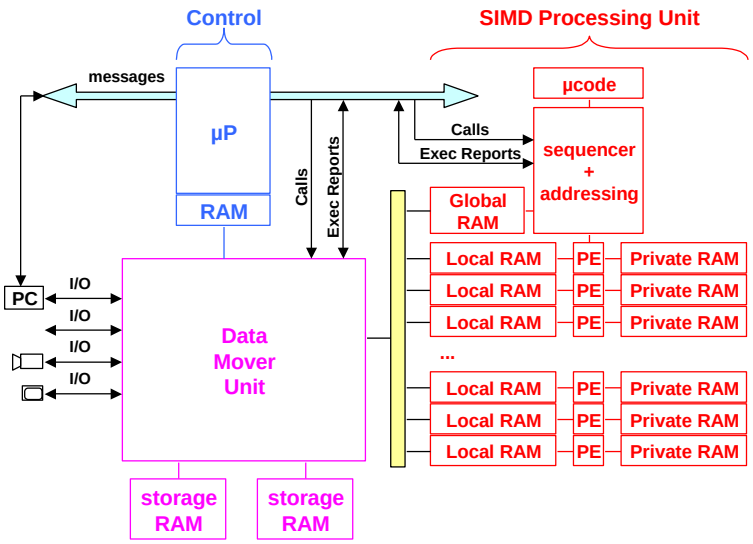


Figure 1. The Ter@pix architecture

The SIMD Processing Unit is composed of 128 Processing Elements (PE), which are all equipped with a local and a private RAM. All the PEs are interconnected in a ring, so that each PE can directly communicate with two neighbors. The SIMD code is stored in a µ-code memory whose the capacity is of 2048 instructions of 108 bits. A sequencer fetches SIMD instructions from this memory, and broadcast them to the PEs. The content of a 16Kx18b global RAM can also be broadcast to all the PEs by the sequencer.

The DMU is roughly composed of DMA engines, mux/demux logic and an external memory controller. The local RAMs and the DMU are interconnected though 2 ring networks. This solution avoids sophisticated logic and simplifies the P&R.

The SIMD Processing Unit is controlled by the general purpose microprocessor. The MicroBlaze has 32 KB of instruction and data memory (32 bits). It is in charge of activating the SIMD and the DMU.

Processing Element micro-architecture

Figure 2 represents the micro-architecture of the Ter@pix PE. The datapath bit precision is of 18 bits. This value has been chosen in accordance with the bit precision of the RAM blocks and DSP macros integrated into the FPGA. The private RAM is a 36-bit memory of 512 words that acts as a Register File. The use of a RAM instead of a Register File was driven by the FPGA implementation. The PE also contains a local RAM of 1024 words of 18 bits.

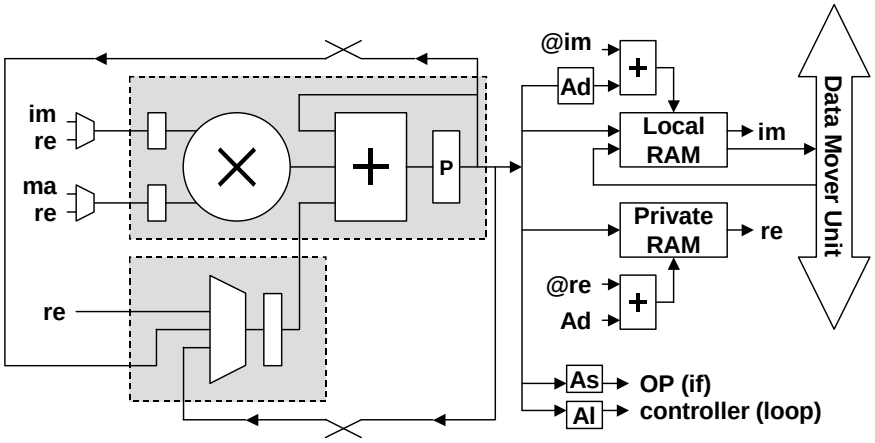


Figure 2. Micro-architecture of the Ter@pix PE

Programming model

The programming model of the Ter@pix architecture is based on a library of image processing operators. An operator is a piece of microcode running on the SIMD. The microcode of the operators is stored in a micro-code SRAM memory, and is loaded from the external memory at boot time by the MicroBlaze. An operator can be activated by the GPP through a remote call which indicates the starting address in the microcode plus call parameters. Note that due to memory size limitations inside the FPGA, operators generally do not work on a full image but on a window. The role of the GPP is to split images into windows and to make remote calls in order first to get a window loaded by the DMU from the storage RAM, second to get it processed by the SIMD, third to get the result unloaded by the DMU into the storage RAM. Of course, data transfer and processing tasks are done in parallel. A middleware, executed on the uBlaze, provides control and communication services to support the development of applications on the Ter@pix processor.

Performance results

This processor has been implemented on a Xilinx Virtex-4 SX-55 [Xilinx07]. This FPGA is realized in a 90 nm CMOS technology, and integrates 24 576 slices, 512 DSP macro [Xilinx04] and 320 blocks of RAM. The SIMD array has been sized to 128 PEs in order to maximize the occupation of the FPGA. In this configuration, the resources occupation of the selected FPGA is of 82% for the logic block slices. The critical resource is the memory with 95% of the RAM blocks occupied. Three quarters of the DSP blocks are used (2 DSP macros are used per PE and the implementation of the DMU takes 128 DSP blocks).

The SIMD unit operates at a peak frequency of 150 MHz. It achieves thus a processing power of 19.2 GMACS at this peak frequency (i.e. 1 GMAC corresponds to 109x MAC operation executed in 1 second). The power dissipation has been measured to 14 W. It gives a power efficiency of 1.37 GMACS/W.

The Xetal-II processor

Processor architecture

The Xetal-II processor [Abb08] is a 16-bit processor, which is also dedicated to image processing. Its architecture (see Figure 3) is organized as following:

- A linear processor array is the core of the architecture. It corresponds to the SIMD unit. It is composed of 320 Processing Elements. This number of PE is an integral divisor of image lines for most standard video formats, and is thus convenient for the mapping of applications on the processor.
- A frame memory allows to store 2048 lines of 320 pixels.
- A global control processor executes the application code, fetches and broadcasts the SIMD instructions and control IOs.
- A Data Input Processor (DIP) and a Data Output Processor (DOP) control respectively three input and three output independent video channels. Each of them contains a memory buffer of 240 kb. The video channels have a 10-bit precision.
- A sequential I/O memory composed of 2 shift-registers lines to feed the array from the DIP interface, and to dump the results to the DOP interface.

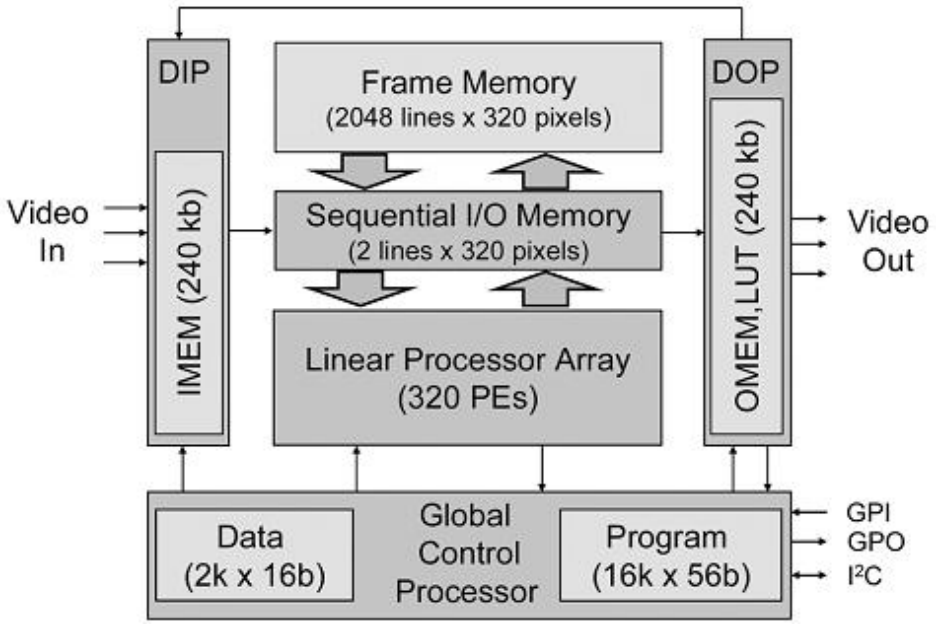


Figure 1

Figure 3. XetaL-II top-level architecture [Abb08]

A Look-Up Table (LUT) is implemented in the Data Output Processor for an efficient execution of non-linear image processing tasks. A global LUT has been preferred to many local LUTs integrated in each PEs because it is a cheaper solution. In this way, a LUT with 4096 entries has been implemented inside the DOP.

The global processor controls the DIP and the DOP processors, as well as the program flow. It fetches 2 instructions by cycle. If possible, they are executed in parallel by the processor array and the global processor.

Processing Element micro-architecture

The PEs are grouped in tiles of 8 PEs (see Figure 4). This organization in tiles simplifies the placement & routing, as a place and routed tile can easily be replicated inside the layout. There is one instruction decoder per tile. It avoids to have a dramatic rise of the control logic, as it would be the case with one instruction decoder per PE. On the other side, it reduces the number of control logic signals to route compared to an implementation with only one instruction decoder.

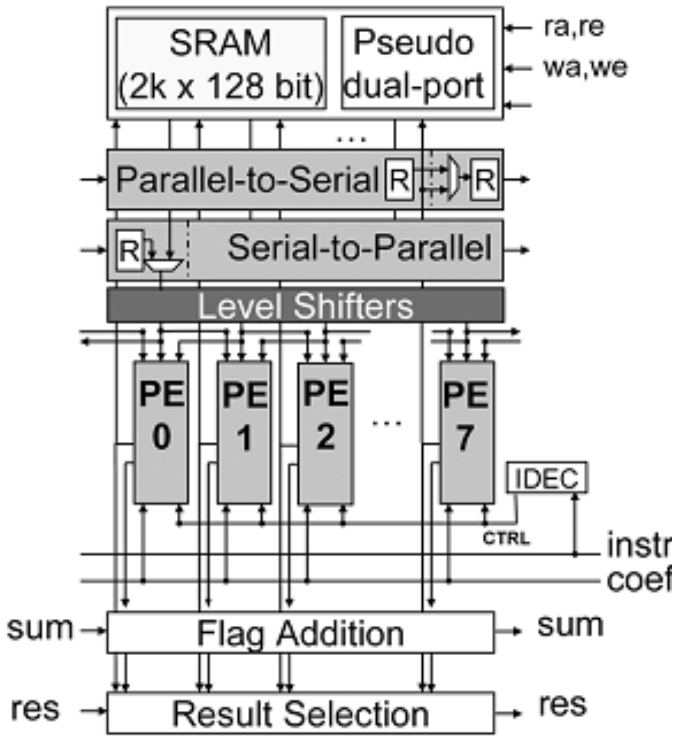


Figure 4. Xetal-II tile architecture [Abb08]

The pseudo dual-port RAM is implemented physically by a single-port memory. As the logic operates at half the frequency of this memory, two memory accesses are possible during one clock period of the logic domain. In this way, the single-port memory can mimic the behaviour of a dual-port memory.

The voltage domain of the linear processor array is different from the one of the frame memory and of the sequential line memory. So, level shifters in the tile adapt these voltage domains.

Figure 5 represents the micro-architecture of the elementary processing elements. It is made up of a basic ALU, MUX/DEMUX logic, a register/accumulator and some flags. The instruction set of the ALU consists of 64 instructions, which are all

executed in one cycle. The available instructions range from simple arithmetic and logical operations to MAC.

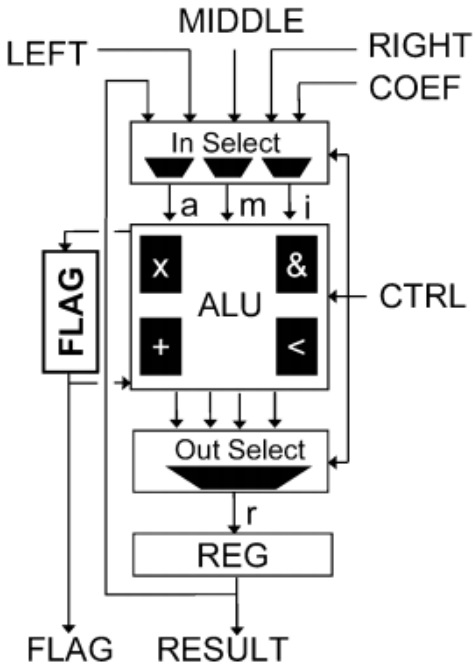


Figure 5. Xetal-II Processing Element micro-architecture [Abb08]

Programming model

The programming is done in a subset of C extended with a vector data type (corresponding to a 320-element array of 16-bits integers). Operations on vector data are automatically assigned to the Linear Processor Array. Other operations are executed on the Global Control Processor.

Performance results

The chip implementation occupies an area of 74 mm² in a 90 nm technology. The processor runs at a peak frequency of 84 MHz. At this frequency, it delivers a peak performance of 107 GOPS for a power dissipation of only 600 mW. It makes it convenient for low power applications, including portable consumer applications.

This peak performance considers the case where the PEs execute 4 basic operations per instructions (4*320*84=107 520). If we evaluate the peak performance in number of MAC per second, the processor delivers a peak performance of 26.88 GMACS (84*320 = 26 880). The reported peak performance of 107 GOPS gives a power efficiency of 178.33 GOPS/W. If we evaluate it in term of MAC operations, it gives a power efficiency of 44.8 GMACS/W.

The IMAPCAR processor

Processor architecture

The IMAPCAR processor [Kyo08a] [Kyo08b] from NEC has been designed for in-vehicle image recognition applications. It is notably used for the obstacle detecting

pre-crash safety system of the TOYOTA LEXUS. Figure 6 shows its architecture that consists of an array of Processing Elements, a RISC control processor (CP) and an external interface block (EIF).

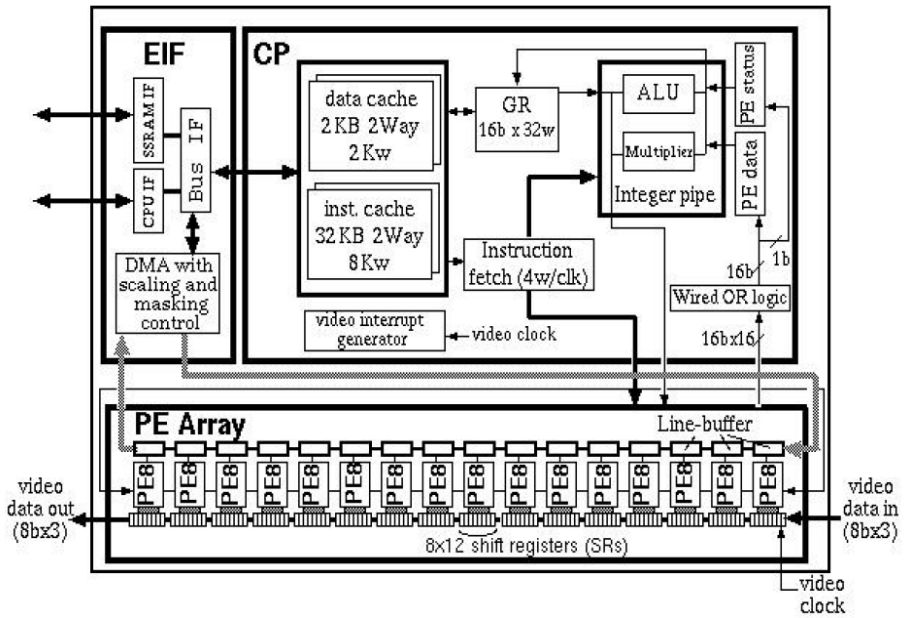


Figure 6. Architecture of the IMAPCAR processor [Kyo08a]

To ensure the reliability required for automotive use, instructions memories are protected by ECC (Error Correction Code) while data memories are protected by parity bits. The control processor features a 32KB instruction cache and a 2 KB data cache. It executes the scalar operations and broadcasts other instructions to the PE array. The PE array contains 128 PEs connected in a ring. The array of Processing Elements is organized in 16 tiles of 8 PEs. The set of data RAM attached to each PE composes the internal memory of the chip, which is called IMEM. A DMA controller transfers data between the IMEM and the external memory through a line of buffer. Shift registers are also available to store data in the PE from input/output video channels.

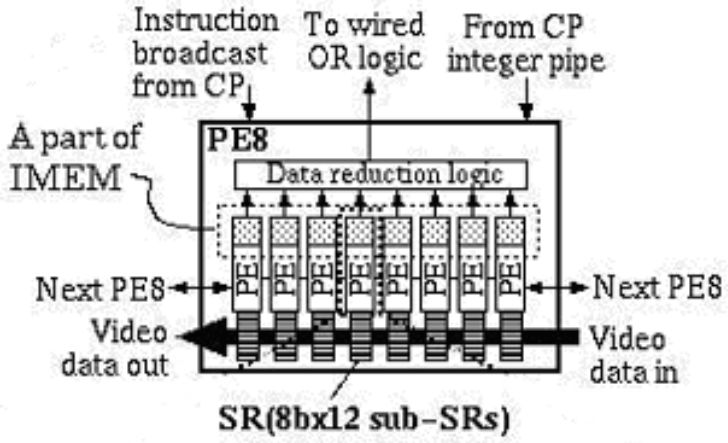


Figure 7. Tile architecture of the IMAPCAR processor [Kyo08a]

Processing Element micro-architecture

A PE is a 4-Way VLIW unit with a MAC operator. Three data arithmetic operations and one memory access operation can be accomplished within one clock cycle. Its micro-architecture is represented in the Figure 8. Each PE has a 2 KB RAM (so the total capacity of the IMEM is 256 KB).

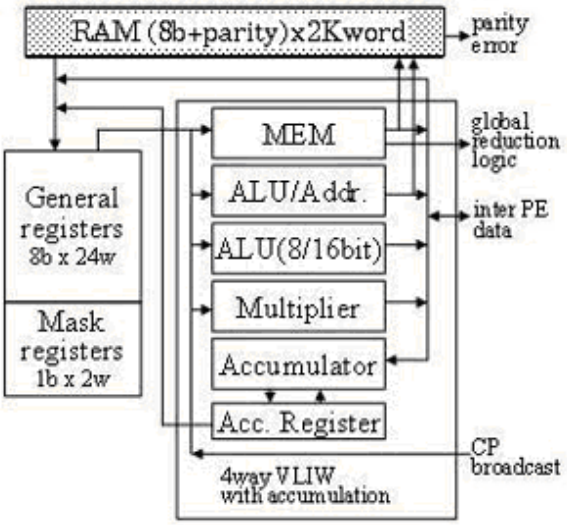


Figure 8. Micro-architecture of the IMAPCAR Processing Element [Kyo08a]

Programming model

The programming language of IMPCAR is a C extension called 1DC. The data items manipulated by the PE array are explicitly specified by using the keyword "sep". A VLIW compiler is available to generate VLIW code from 1DC descriptions.

Performance results

The chip is realized in a 0.13 µm technology, and contains 26.8M of transistors. It can achieve a 100 GOPS peak performance (with 8 bit operations) at a peak frequency of 100 MHz. The power dissipation is approximately of 2 watts. If we consider that each PE can do one MAC by cycle thanks to its MAC unit, the peak performance is of 12.8 GMAC/s in term of MAC operations.

The Storm-1 processor

Processor architecture

The Storm-1 processor [Kha08][SPI] targets stream processing for the realization of image and signal processing applications. It has been designed to exploit both the data-level and the instruction-level parallelisms of applications. Two GPPs control the platform, and execute the application code (see Figure 9).

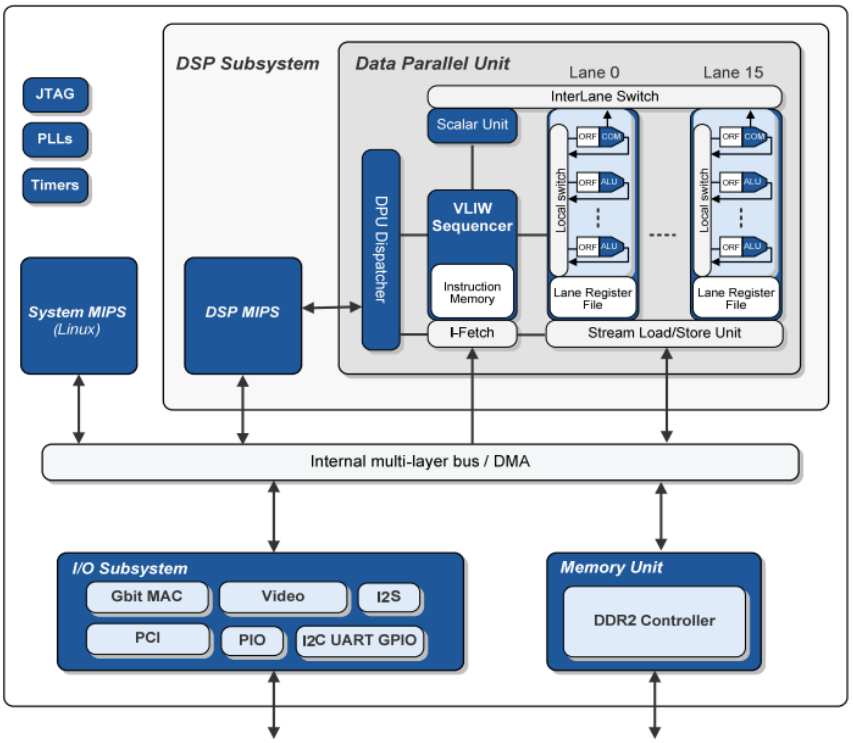


Figure 9. Storm-1 processor architecture [SPI]

One of them (System MIPS) is dedicated to the control of the I/Os, and the other one (DSP MIPS) executes the application and controls a Data-Parallel Unit (DPU). The I/O subsystem integrates a set of interfaces commonly used for embedded systems. The SoC also contains a DDR2 memory controller. The DPU executes compute-intensive kernels. It is controlled by the application threads running on the DSP MIPS. The DSP MIPS sent commands to the CPU. Four commands are used to control the DPU: (1) load kernel, (2) execute kernel, (3) load stream and (4) store stream. A command dispatcher sequences the commands and dispatches them between the communication and execution units. The instruction fetch unit loads the kernel microcode into a 96 KB on-chip SRAM.

The Data Parallel Unit is organized in 16 lanes with a 5-ALU VLIW per lane. So, it contains a total of 80 parallel ALUs. There is one 16KB register file per lane. A stream load/store unit transfers data between the external memory and these register files. It supports sequential, strided and scatter/gather access patterns. Data can also be exchanged between lanes through an inter-lane switch during kernel execution. The register files in the lanes are implemented by 16 KB single-ported 128-bit-wide

SRAM, which can be accessed on any half clock-rate cycle. All the units of a lane are interconnected by a compiler managed full crossbar switch.

Processing Element micro-architecture

In this architecture, the processing element consist of a VLIW unit composed of five ALUs, one load/store unit and one communication unit. The DPU uses a 15 stage pipeline. The ALU supports SWAR (SIMD Within A Register) operations, so that the architecture can also take advantage of data parallelism at the ALU level. There is one 16-word two-port register file per input of the ALU (see Figure 10).

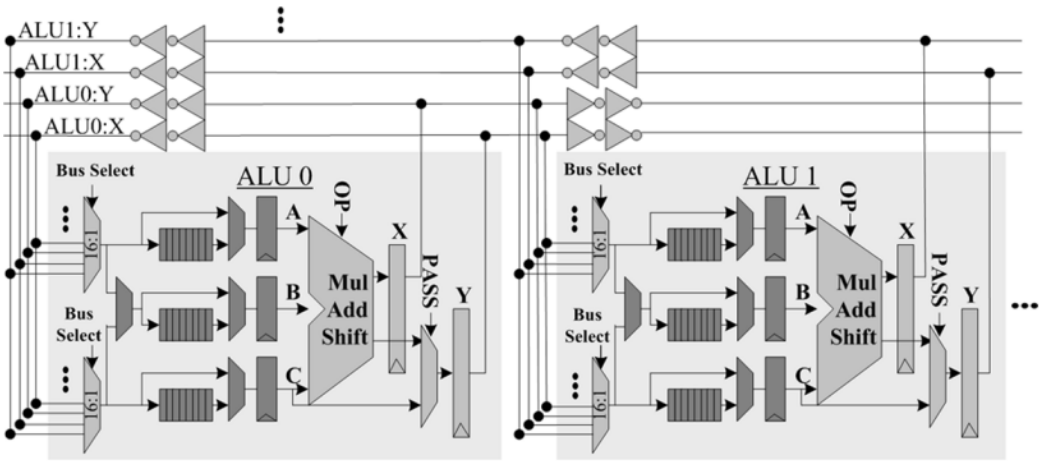


Figure 10. VLIW micro-architecture [Kha08]

Programming model

The DSP MIPS, which runs the main application threads, is programmed in C language. Keywords are used in the code to indicate kernel function calls that run on the DPU. A single kernel is active at a time. The kernels only access to input/output data streams that are locally stored in the lane register files. The load/store of the streams in the register files are explicitly described by the programmer. The compiler supports optimized arithmetic operations through the use of intrinsic functions.

Performance results

The chip counts 34 millions of transistors, and takes up 155 mm² in a 0.13 µm technology. The lanes operate at a frequency of 800 MHz. It allows the chip to deliver a performance of 256 GOPS (16 bit operations) or 128 GMACS. This processor dissipates a power of 10.5 W. So, its power efficiency is of 12.2 GMACS/W or 24.4 GOPS/W.

3.3.2. Comparison of regular SIMD architectures

The processors presented in this section are mostly dedicated for image processing. Indeed, low-level processing in this application domain often exhibits data

parallelism at the level of pixels or of macro-blocs of pixels. This characteristic has made of SIMD architecture a relevant choice for implementing image processing applications.

Performance and power efficiency

Table 1 presents the traditional figures of merit and the characteristics of presented processors. The differences of implementation technologies have however to be considered when comparing these raw performances figures.

	Ter@pix	Xetal-II	Storm-1	IMAPCAR
Number of PEs	128	320	80	128
Datapath resolution (bits)	18	16	32	8
On-chip data memory (Mb)	5	10	2	3
Clock (MHz)	150	84	800	100
Peak performance (16-bit MAC x10^9 /s)	19.2	27	128	12.8 *(24-bit)*
Power efficiency (GMACS/W)	1.37	44.8	12.2	6.4~12.8

Table 1. Comparison of SIMD architecture characteristics

The Ter@pix processor is the only one to be implemented on FPGA. It follows logically from this that it has the worst power efficiency of all the studied SIMD processors (due to the bad efficiency of FPGA compared to ASIC implementations). On the contrary, the Xetal-II is the most power efficient processor. The primary reason of its good power efficiency is probably the chip implementation in a 90 nm technology, whereas the Storm-1 and IMAPCAR processors are realized in a 0.13 µm technology.

We could also note that the Xetal-II has no external memory controller, on the contrary of other chips. The IMAPCAR processor has a SRAM controller; the Storm-1 processor has a DDRAM controller whereas Ter@pix has both a SRAM and a DDRAM controller. This parameter has probably also an influence on the power dissipation (even if this factor is minor relatively to the implementation technology). In term of performance, the large amount of on-chip data memory in the Xetal-II processor aims to compensate the absence of memory controller.

The Storm-1 processor delivers the highest peak performance. To achieve this raw peak performance, it operates at a peak frequency of 800 MHz, whereas other processors have a frequency that does not exceed 150 MHz. The IMAPCAR has the lowest raw peak performance but the peak performance of the IMAPCAR corresponds to 24-bit MAC whereas the peak performance of others processors corresponds to 16-bit MAC. All the more, this figure does not take into account the

VLIW architecture of the IMAPCAR PEs. Other operations can be done in parallel with a MAC in a PE.

Beside the performance considerations, the aim of the Synaptic project is to combine performance and manufacturability aspects thanks to regular approaches. No objective figures and quantitative metrics are available to measure and evaluate the regularity of an architecture and its tolerance to PV. So, an evaluation of the regularity of the architectures can only be subjective and qualitative.

Processing Elements granularity and homogeneity

The analysis of the presented SIMD architecture clearly shows similarities between them. They are roughly composed of an array of basic processing elements and a control unit. The major differences between these architectures are observed at the PE level. The analysis of the presented architectures shows two approaches in the granularity of the processing elements.

The Ter@pix and the Xetal-II processors use basic PEs, which roughly consists of an ALU, one or few registers and mux-demux logic. They are thus built from the replication of simple units. They are particularly efficient for low-level image processing, which consists in the processing of a large number of pixels in a similar manner. However, a good platform should mix them with other processing architecture.

The IMAPCAR and the Storm-1 processors use VLIW units in their SIMD array. These architecture try thus to exploit no only the data level parallelism of the application, but also the instruction level parallelism. The penalty is an increase of the architecture and compiler complexity. In the case of IMAPCAR, the objective of this design choice is to extend the application domain of the chip to medium and high-level image processing. In the case of the Storm-1 processor, the application domain covers not only image processing but also signal processing. The purpose of these chips is to be used as a standalone component

In term of regularity, the first approach seems more advantageous as the PEs are smaller (ALU instead of VLIW processor). The more small the blocks are, the more number of PEs can be integrated into the chip. It is true if we consider that the PEs are not regular and that the regularity of the architecture is created by the replication of the PEs. But we have also to consider the (non-)regularity of the PEs. In the Storm-1 processors, the five ALUs composing the VLIW unit of the PE are identical. So, we have also some regularity inside the PE.

The datapath bit precision of the Storm-1 processor is of 32 bits, whereas other processors have lower bit precisions. The difference of application domains justifies this difference of bit precision. The Storm-1 is not restricted to image processing but also targets Signal Processing application domain.

All the more, the Storm-1 and Xetal-II processors use the SWAR technique (SIMD Within A Register) in order to be efficient with data of different granularities. For instance, the Storm-1 processor has the highest peak performance expressed in number of 16 GMACS per second, despite it is a 32 bit processor.

Interconnection network topology

The topology of the PE interconnection networks is a major parameter for the regularity of the architecture. The study of the presented architectures shows some similarities. The PEs are interconnected in a line in the Xetal-II processor. The end-points of this line are interconnected in the IMAPCAR and Ter@pix processors to form a ring. A ring topology can help to implement some applications, but is more complex to place and route. The interconnection network of the Storm-1 architecture is more complex. It is based on 2 levels of hierarchy. A crossbar connects the ALUs inside the lanes and an “InterLane Switch” connects the lanes.

For both Ter@pix and Storm-1, the platform IO are done through the intermediary of the external memory and the IOs of the SIMD array are done though load/store in local memories. In the case of Ter@pix, the memory is local to each PEs. Data are transfered from and to these local memories through a 2-ring network. Data transfers are done between these local memories and the external memory. Direct data transfer between two local memories is not possible. Direct data transfers between PEs are done through the PEs interconnection. In the case of the Storm-1 architecture, the IOs are done through the register files located in each lane. A memory switch allows data transfers between the external memory and these register files.

In the IMAPCAR architecture, a shift-registers line (with 12 shift-registers by PE) is used for the IOs and for inter-PE communication. A separate line-buffer is used for data transfers between the external memory and the local memories.

The IOs are done through 2 shift-register line directly from and to the PE in the Xetal-II processor. Qualitatively, the Xetal-II architecture seems to have the more regular interconnection network and the easier to route. The network is linear, used for the IO as well as for the memory, the array can feed or dump directly the IO.

Hierarchical organization of the PE array

The PEs of the Xetal-II and IMAPCAR processors are grouped in tiles of 8 PEs. Such an organization in tile is clearly suitable for the step-and-repeat template approach of the project. It allows to trade-off the cost of the control logic with the number of control signal to route. Implementing one instruction decoder by PE is a costly solution. On the other hand, implementing only instruction decoder for all the processor involves to route a large number of control signals between all the PEs and the instruction decoder. In a tiled approach, two levels of replication co-exist. PEs are replicated in a tile, and the tiles are replicated in the chip.

The Storm-1 processor is organized in 16 lanes of 5-ALU VLIW unit. This organization is also hierarchical, but with different interconnection networks at the different levels.

In the Ter@pix processor, the PEs are not grouped in tiles. But, the place and route procedure is less complex on FPGA than for an ASIC. So, a "flat" organization is probably less problematic than for an ASIC.

Execution and programming models

The analysis of the presented architectures shows that two different approaches in term of execution and programming model have been adopted.

The IMAPCAR and Xetal-II processors are SIMD processors, with scalar processing capabilities as well. There is only one fetch unit for both the scalar and the SIMD instructions. So, the instruction memory is common to the SIMD unit and the scalar unit.

The SIMD unit acts as a coprocessor in the Ter@pix and Storm-1 architectures. It is under the control of a GPP, but has distinct instruction memory and fetch unit from the GPP. This execution model requires a different programming model than the other architectures. These two processors are based on the (explicit) notion of accelerated kernels, where the code of the kernels represents the micro-code of the machine.

Regarding the aims of Synaptic, this aspect has however no impact.

3.3.3. Reconfigurable Computing architectures

The reconfigurable architectures are also composed of a massively replicated element, which can be more or less complex. They are generally based on some kind of dataflow execution model. The elementary cells are dynamically reconfigured to execute the same operation until the next reconfiguration. The interconnections between the cells are also dynamically reconfigured. Data transfers are fixed at reconfiguration time. These architectures are very efficient to execute dataflow dominated kernels.

However, the reconfiguration time between the execution of two different kernels can be a performance killer. If the ratio between the reconfiguration time and the execution time is in favor of the execution time, then the reconfiguration overhead is cost-effective. They are thus efficient and well suited for applications requiring performance accelerations for kernels with medium temporal utilization. Performance density is improved for those applications.

Field Programmable Gate Arrays

Architecture description

FPGAs have been used since the mid-80s [Com02], and are probably the most used reconfigurable architectures. They are appreciated for their flexibility (compared to ASIC) and their performance (compared to GPPs). These architectures are roughly built by a large scale replication of small elementary blocks called slices. For instance, Figure 11 is a simplified view of the Xilinx Virtex-6 FPGA slice. It contains Look-Up Tables (LUT), multiplexers and registers/latches. All these slices are interconnected by a highly flexible network. This network is however the source of a high-overhead in term of area, power efficiency and performance compared to a custom SoC/ASIC implementation.

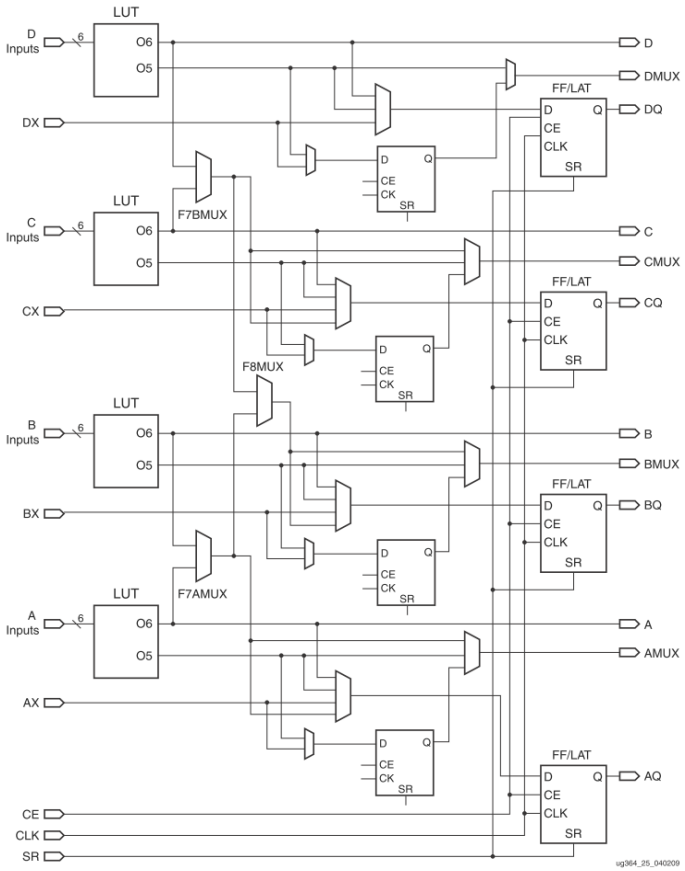


Figure 11. Simplified Virtex-6 FPGA Slice [Xilinx09]

High-performance FPGA architectures now integrates HW macros like DSP blocks or GPPs to overcome the limitations of FPGAs. Lastest generations of FPGAs also exploit partial reconfiguration to deal with their limitations. FPGAs can also be integrated into heterogeneous SoC architectures (e.g. [Ros09]). Such embedded FPGAs [Bar04] is a solution to take advantage of the flexibility of FPGA architectures and of the performance of System-on-Chips.

Programming model

Circuits implemented on FPGAs are described in traditional Hardware Description Languages (e.g. VHDL, Verilog). They are then synthesized and placed & routed from this description. Hardware design flows for FPGAs are similar to ASIC design flows. However, they rely on FPGA proprietary tools to generate the configuration bitstreams.

The DREAM architecture

Architecture description

The DREAM processor [Muc09] has been designed to be integrated as a subsystem inside a SoC. Its architecture is based on a 4-bit reconfigurable datapath, called PicoGA. A 32-bit RISC processor controls the reconfiguration and the execution of this reconfigurable array, but also the address generators and the interconnect crossbar used to stream data to and from this array. A memory bank can be used both as a communication buffer or as a temporary storage buffer.

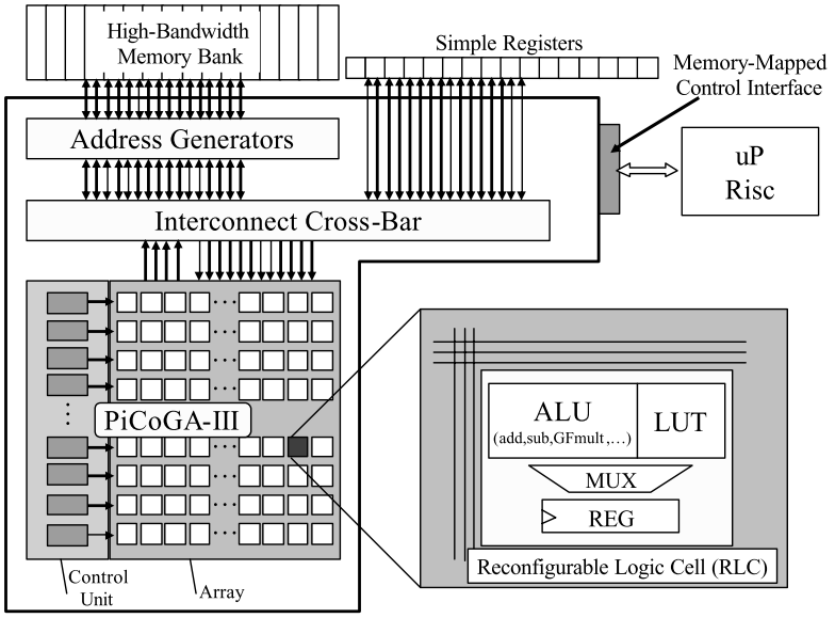


Figure 12. The DREAM architecture [Muc09]

Programming model

The architecture is programmed in a subset of C language called GriffyC.

Performance results

The DREAM core is integrated a SoC prototype implemented in 90 nm technology. It can run at a peak frequency of 200 MHz for a power consumption of 375 mW. It occupies an area of 22.5 mm².

The ADRES architecture

Architecture description

ADRES is a customizable architecture consisting of a 2D array of basic components (see Figure 13). There are three types of basic components: functional units (FUs), storage resources such as register files and memory blocks, and routing resources that include wires, multiplexers and busses. Different values can be specified for the communication topology, the number and size of local register files and functional units and supported instruction set. In fact, ADRES is a template more than a concrete processor architecture.

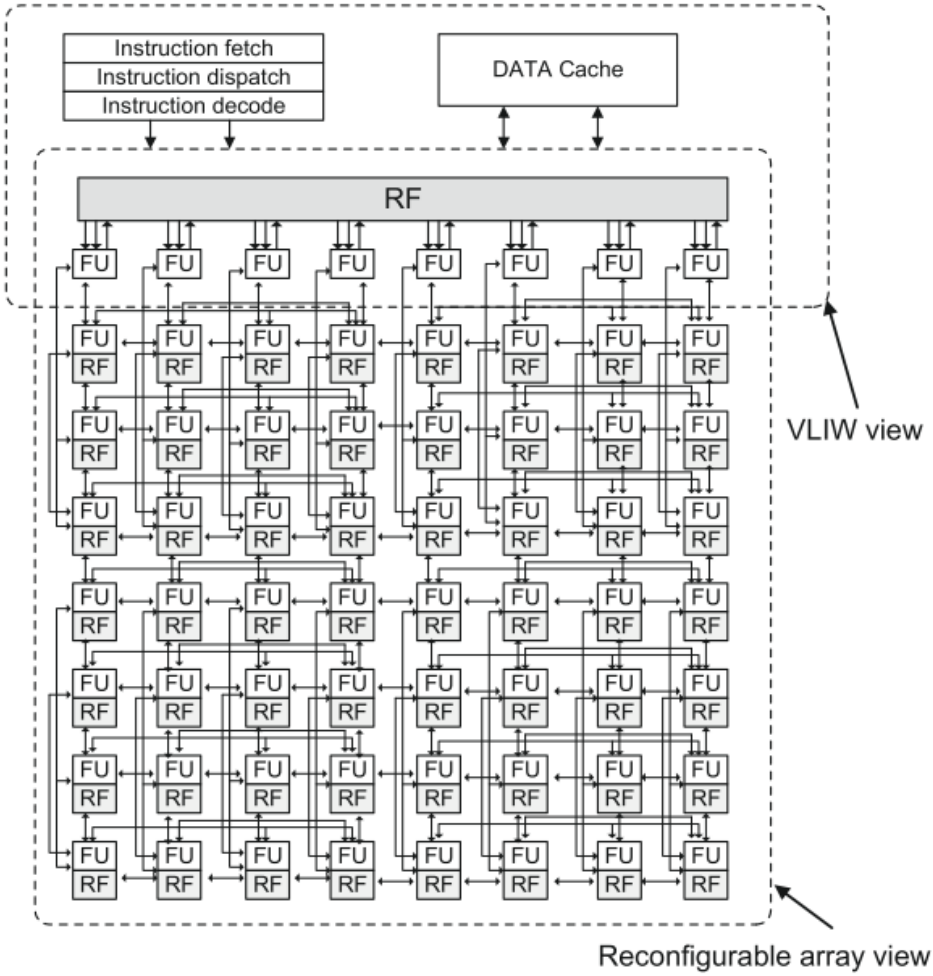


Figure 13. Architecture of the ADRES coarse-grain reconfigurable array [Ber09]

An original aspect of the architecture is the coexistence of two functional view of the same physical resources. The top row of the reconfigurable array can act as a tightly coupled VLIW processor in the same physical entity. The central register file, which is shared between the two modes, is used for the data communication between the VLIW processor and the reconfigurable array.

The FUs (Figure 14) execute coarse-grained operations on 32-bits data, and support predicated operations for conditional executions. Their results can either be written to a local register file (RF), or be routed directly to the inputs of other FUs.

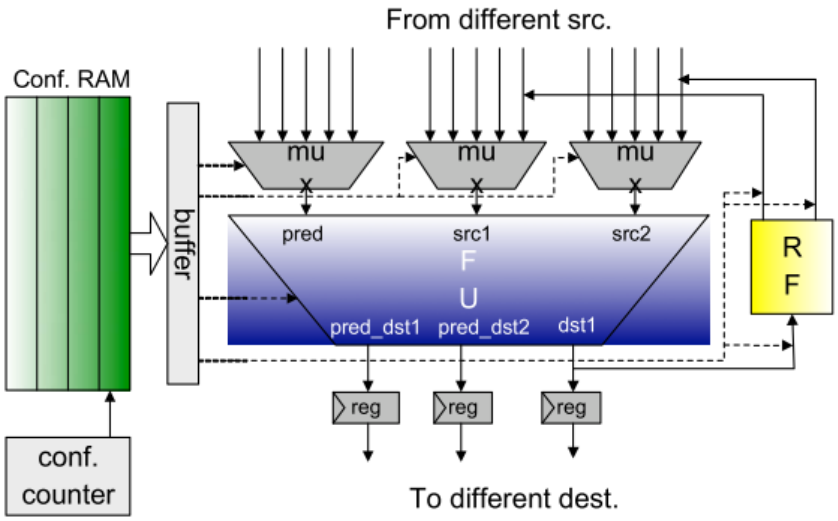


Figure 14. ADRES array node [Ber09]

It is also a multi-context reconfigurable array. Many configuration contexts are stored locally, and loaded on a cycle-by-cycle basis.

Programming model

A C compiler has been developed for the ADRES architecture, which can thus be fully programmed in C language. This compiler is able to find SW pipelining, do the automatic placement and the scheduling. The computation intensive kernels, typically the dataflow loops, are mapped onto the reconfigurable array. To achieve an efficient mapping of loops to the CGRA, some source–level transformations must however be performed by the programmer

Performance results

The operating frequency has been estimated after synthesis to 600 MHz in 90 nm technology. A power efficiency in the range of 30 GOPS/W has been reported. This power estimation has been done considering 2 KB of instruction and data cache and 1 MB of L2 cache.

In a 90 nm technology, the size of a 8 x 8 array is estimated to 4 mm² without cache, to 6 mm² with L1 cache and to 22 mm² with L1 and L2 caches.

The PACT-XPP architecture

Architecture description

The PACT-III architecture is the combination of a-dataflow array for computation intensive loop kernels, and of VLIW cores for control-dominated code. The array is composed of two kinds of elements, namely the ALU-PAEs and the RAM-PAEs.

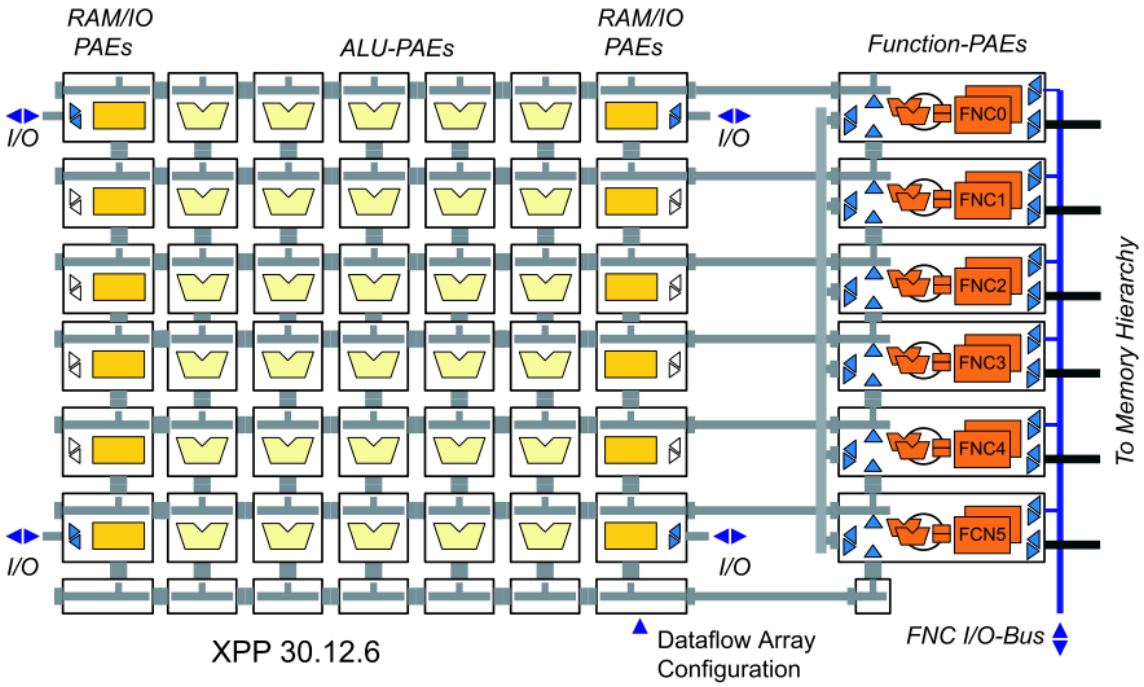


Figure 15. PACT-III dataflow array [PACT]

Each processing element contains an ALU configured to execute operations on words of 16, 24 or 32 bits words. The bit precision is parameterized at design-time. The XPP array is also sized at design time.

The operation to execute is fixed by the configuration. Each PE is synchronized with its neighbours. The elements of the arrays forward synchronization signals with the forwarded data packets. These signals trigger the execution of the next/following PEs. The RAM/IO-PAEs act as input/output buffers and IO controllers.

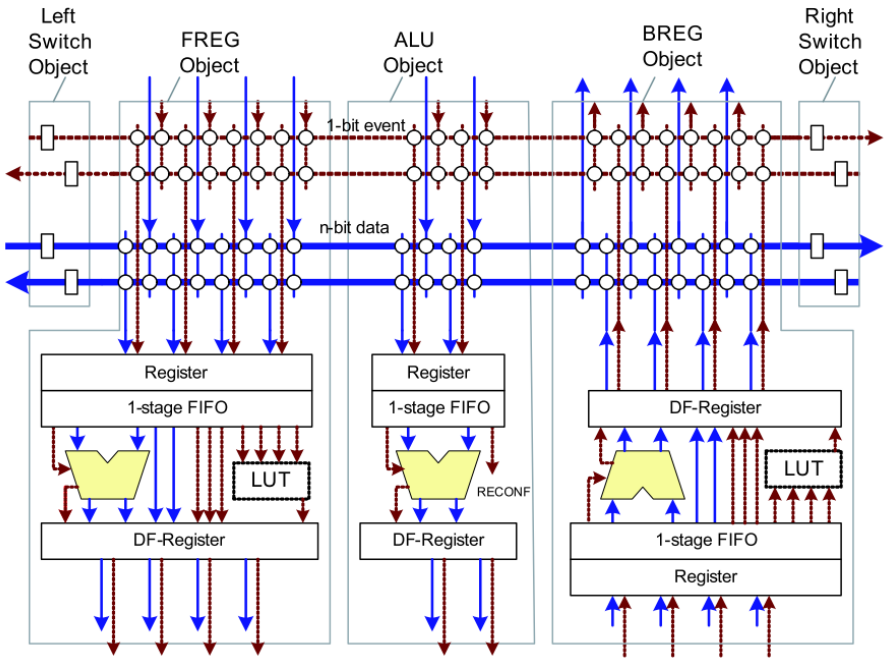


Figure 16. Structure of an ALU-PAE [PACT]

Programming model

The XPP architecture can be programmed at low-level using the NML language or at high-level in C language [PACTb]. When using the NML language, the resource allocation and the configuration of the ALU-PAEs are described by the programmer in the code. When using the C language, a compiler is available to semi-automatically generate the configuration bitstream from the C code.

The RICA architecture

Architecture description

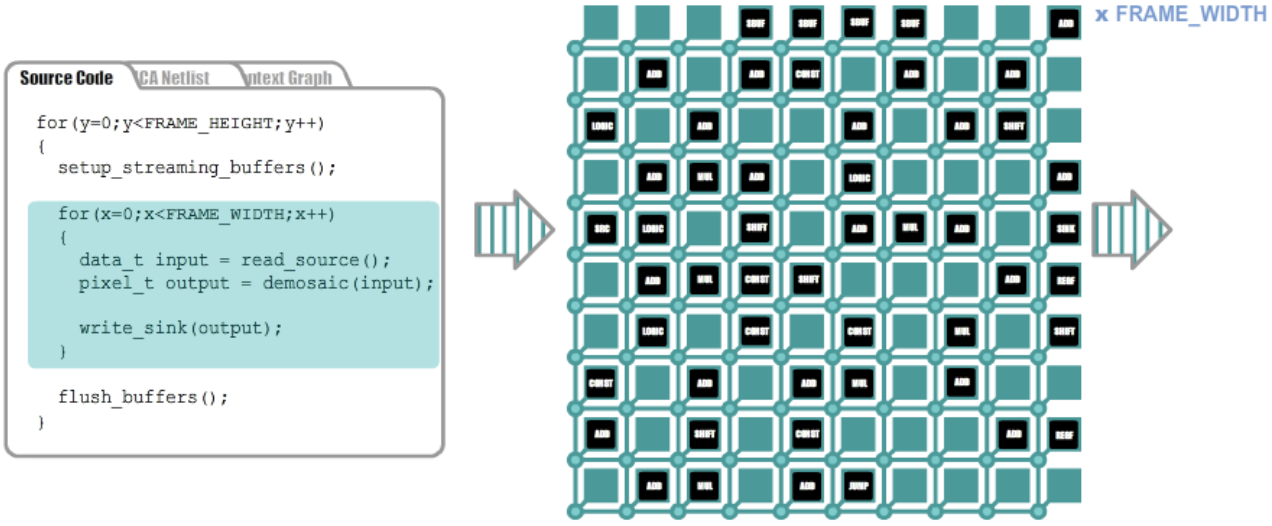
The Reconfigurable Instruction Cell Array (RICA) is a high performance programmable core with a natural inclination to streaming applications. The benefits it offers are:

Easy programming model from high-level C

High speed (throughputs similar to ASICs using software defined pipelining)

Low power compared to other programmable solutions

High silicon efficiency (performance/area) compared to other programmable solutions



Rica is an heterogeneous array of instruction cells (http://www.ricatek.com)

Rica is an heterogeneous array of instruction cells with an Harvard Architecture with data memory and program memory, reconfigurable rate controller to control delay in datapaths, General I/O Ports (mapped as Instructions-Cells) and Multiple-memory banks

Programming model

The RICA programming framework consists of an image-processing library (written in C) along with tools for compilation (C/C++ frontend), implementation and emulation. For testing, debugging and profiling, both a functional emulator and a cycle accurate simulator exist. The simulation model is able to accept detailed timing information from the mapper backend for accurate performance and power estimation. The RICA tools are bundled with a RICA specific Eclipse IDE plugin.

3.3.4. Comparison of RC architectures

PE granularity

The granularity of the elementary cells is one of the main differentiators between these architectures. A versatile platform like [MORPHEUS] mix RC units with different granularities, and is thus able to tackle a large scope of application by selecting the most suitable unit when programming an application. Figure 1 shows the theoretical performance of the different reconfigures cores integrated inside this SoC. It clearly shows the difference of performance according to the granularity of data processed.

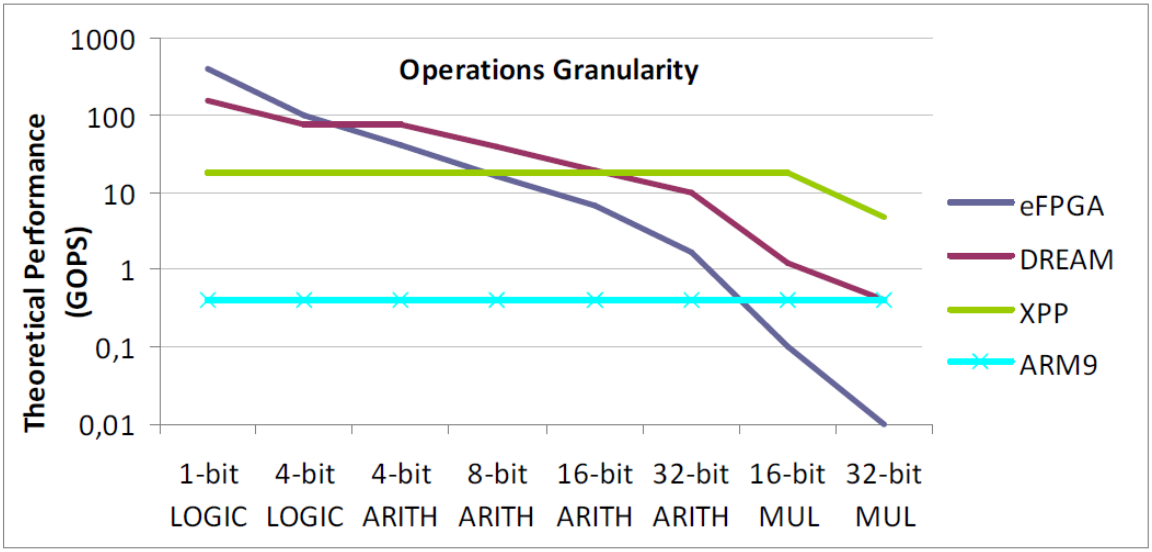


Figure 17. MORPHEUS SoC theoretical performance

Network topology

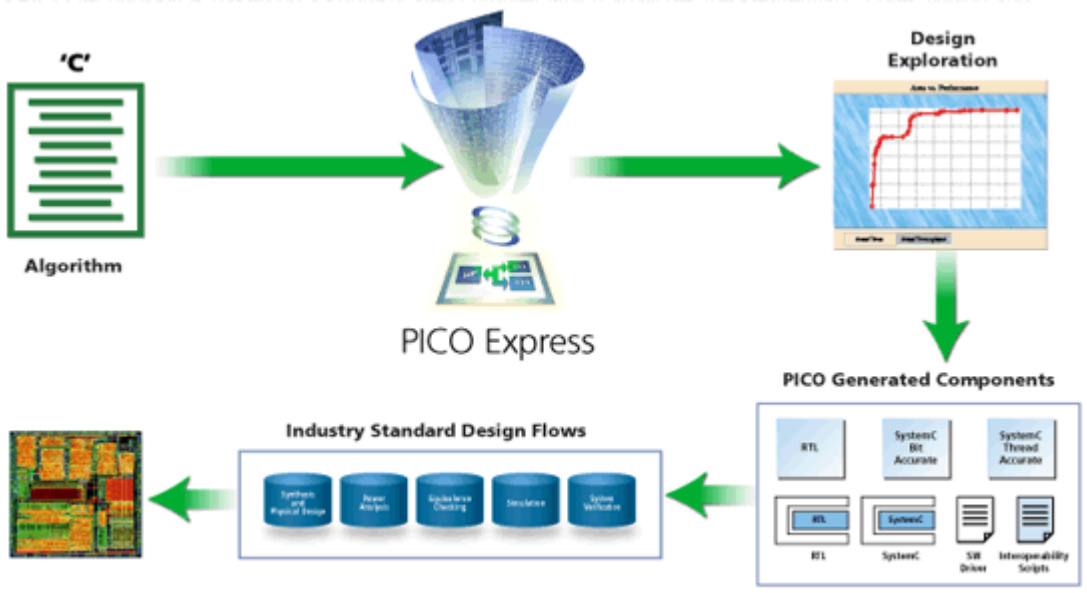
The analyzed RC architectures have a 2D topology unlike the analyzed SIMD architecture. They are also characterized by their great regularity.

4. Analysis of datapaths and interconnects

4.1. Datapaths

In order to evaluate and validate the results of the SYNAPTIC compound design flow it might be worthwhile to consider also RTL automatically generated by advanced data-path and compute engines generators such as the ones available from a number of leading EDA/IP provider companies, for example CatapultC, Esterel, Pico (Synfora), Forte Design, etc.

As an example of the kind of technology in this domain, Synfora provides PICO Express, an algorithmic synthesis tools able to generated hardware from untimed C algorithms. The tool is aimed at allowing the designer to work at a higher level of abstraction.



PICO Express generates hardware from an untimed C algorithm; can support both RTL and SystemC models and testbenches (source: http://www.synfora.com/products/picoexpress.html)

4.2. Interconnects

A full analysis of on-chip interconnects is well beyond the scope of this document; for the purpose of selecting representative testbenches associated to the interconnect infrastructure of modern SoC to be fed into the project benchmarking and validation tasks, we will classify interconnects in two broad categories, circuit switched busses with a centralized arbiter but with the support for segmentation and multiple hierarchies; and more modern packetized on-chip networks a.k.a Networks-On-Chip (NoCs) with distributed routing and arbitration.

To the first category belong the industry standard technologies such as the ARM AMBA bus (APB,AHB,AXI) or the STMicroelectronics STBus, the IBM OPB and many others; while to the latter the commercial offerings are continuously increasing such as the ones from the likes of Arteris [Arteris10], STMicroelectronics [STNoC05].

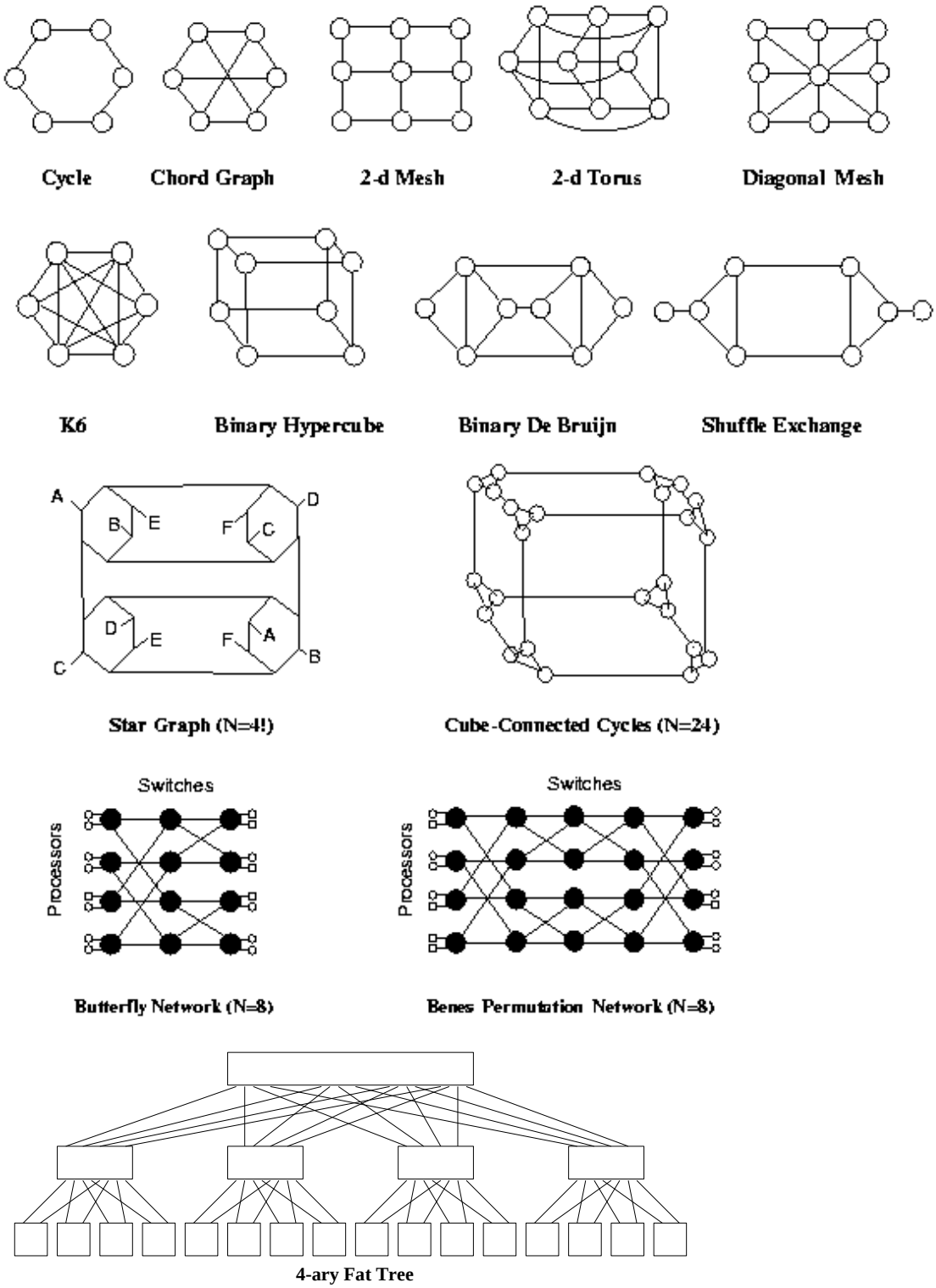
Traditional on-chip buses, such as IBM CoreConnect, STMicroelectronics STBus, and AMBA AXI are generally believed to have both strengths and weaknesses compared to NoC architectures.

For example the widely deployed AMBA AXI protocol targets low latency, high bandwidth, reduced power submicron interconnect design, while retaining previous AMBA 3.0 (ASB, AHB) modularity, component reuse and a simple interface. AXI includes optional extensions that cover signaling for low-power operation and is backward compatible with existing AHB and APB interfaces. Key features of AXI protocol include:

- separated address & data phase,
- burst transactions,
- separate read and write data channels,
- ability to issue multiple outstanding addresses,
- out-of-order transaction completion, and
- addition of register stages to enable simple time closure

With increasing SoC complexity and performance, NoC is clearly the best IP block integration solution for high-end SoC designs today and into the foreseeable MPSoC era. This distributed on-chip communication network architecture reduces communication resource sharing by replacing the traditional bus architecture with a distributed, low-cost, point-to-point, packet-based pipelined network architecture that incorporates a layered network protocol stack analogous to Open System Interconnect (OSI). MPSoC will generally benefit from hybrid communication solutions, i.e. intra-processing element (PE) communication with local storage based on a traditional bus and inter-PE communication based on a NoC interconnect. The

topology supported by the network significantly impacts many aspects of NoC cost and performance.



Different NoC topologies

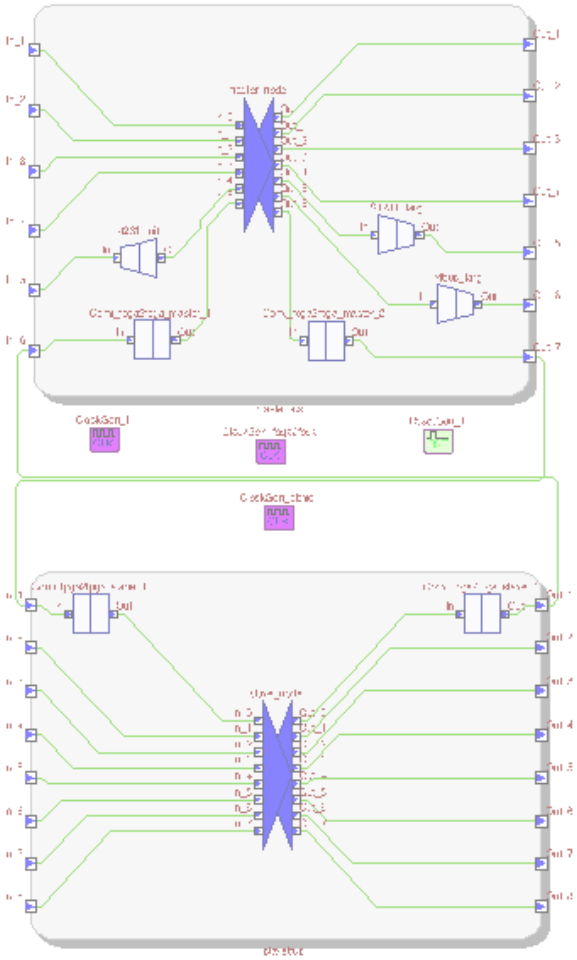
For an excellent survey of existing academic proposals on NoC architectures and variants see [Salminen08].

In order to focus on a pragmatic test case selection we will focus on the technologies readily available for the STMicroelectronics partner of Synaptic, namely the STBus and STNoC [STNoC04] interconnects.

The STBus is a set of protocols, interfaces and architectural specifications defined to implement the communication network of digital systems such as microcontrollers for different applications (set-top box, digital camera, MPEG decoder, GPS).

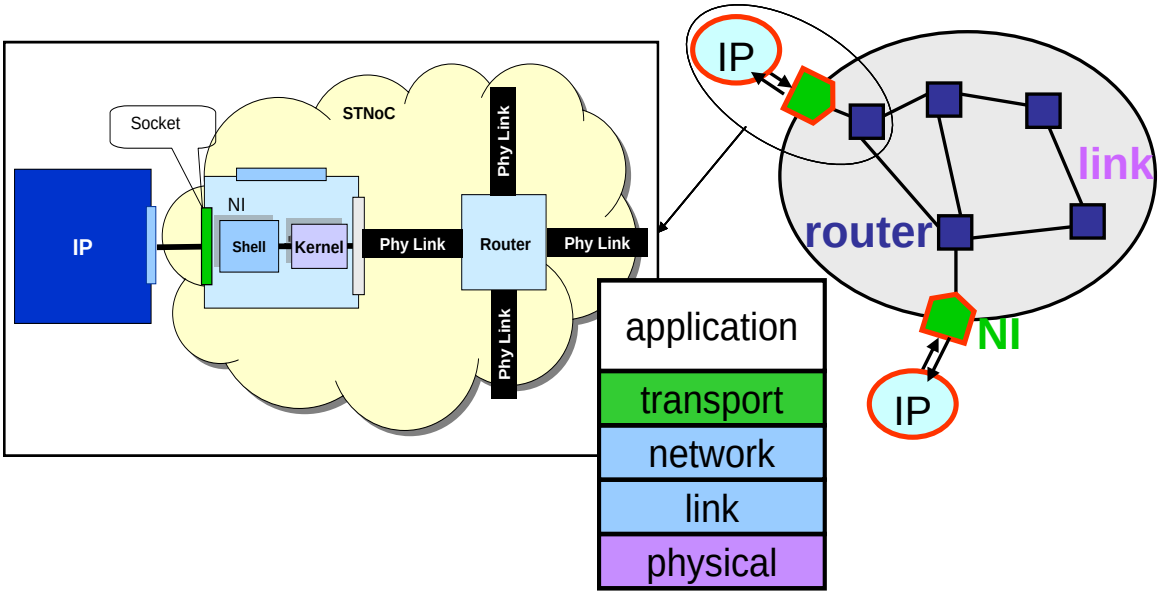
The STBus protocol consists of three different types (namely Type I, Type II and Type III), each one associated with a different interface and capable of differing performance levels. Type I is a simple synchronous handshake protocol with a limited set of available command types, suitable for register access and slow peripherals. Type II is more efficient than type I because it supports split transactions and pipelining. The transaction set includes read/write operation with different sizes (up to 64 bytes) and also specific operations like ReadModifyWrite and Swap. Transactions may also be grouped together into chunks to ensure allocation of the slave and so ensure no interruption of the data stream. It is typically suited for External Memory controllers. A limitation of this protocol is that the traffic must be ordered. Type III is the most efficient, as it adds support for out-of-order transactions and asymmetric communication (length of request packet different from the length of response packet) on top of what is already provided by Type II. It can therefore be used by CPUs, multichannel DMAs and DDR controllers.

The STBus architecture is modular and allows master and slaves of any protocol type and data size to communicate, through the use of appropriate type/size converters. A wide variety of arbitration policies is also available, to help system integrators meet initiators/system requirements. These include bandwidth limitation, latency arbitration, LRU, priority-based arbitration and others.



Example of an STBus interconnect with two arbitrers, size converters and asynchronous links between the bus nodes

The STNoC is an industrially proven Network on Chip technology developed by STMicroelectronics and provides a large number of features to cope with highly complex MPSoC requirements comprising of all of the blocks required to assemble a very sophisticated interconnect topology (Network Interfaces, Routers, synchronizers, repeaters, concentrators, different kind of links, etc.).



Spidergon STNoC is a set of Network Interfaces (NI) (layer 4), on-chip routers (layers 2,3) and physical links (layer 1)

5. Conclusions

5.1. Selection of test bench

To keep the extra development and refactoring effort low enough to be pragmatically addressed by the project, we have selected a small number of candidate IPs.

Thales will deliver a subset RTL that comprises an array of Processing Elements with their local memories controlled by a sequencer.

One of the key objectives of the regular architecture analysis is the selection of test benches to validate the Synaptic design flow and assess its benefits. The comparative analysis should help to select amongst the possible choices the ones that bear the most interest in term of industrial relevance as well as the most potential to expose the benefits envisioned. Two categories of fine-grained architectures have been identified: the SIMD architectures and the reconfigurable architectures. Each of them has its advantages and disadvantages. But, none of them seems more regular than the other.

As a major concern of the testbench is the related time of development, we have selected a SIMD architecture as a testbench of the Synaptic methodology. The

availability of the Ter@pix IP, which has been presented previously, is a major criteria of choice, while being a guarantee of industrial relevance.

The testbench has to be sufficiently significant while not involving too much development efforts. To this end, only a subset of the Ter@pix SIMD architecture is selected. It comprises an array of Processing Elements with their local memories controlled by a sequencer.

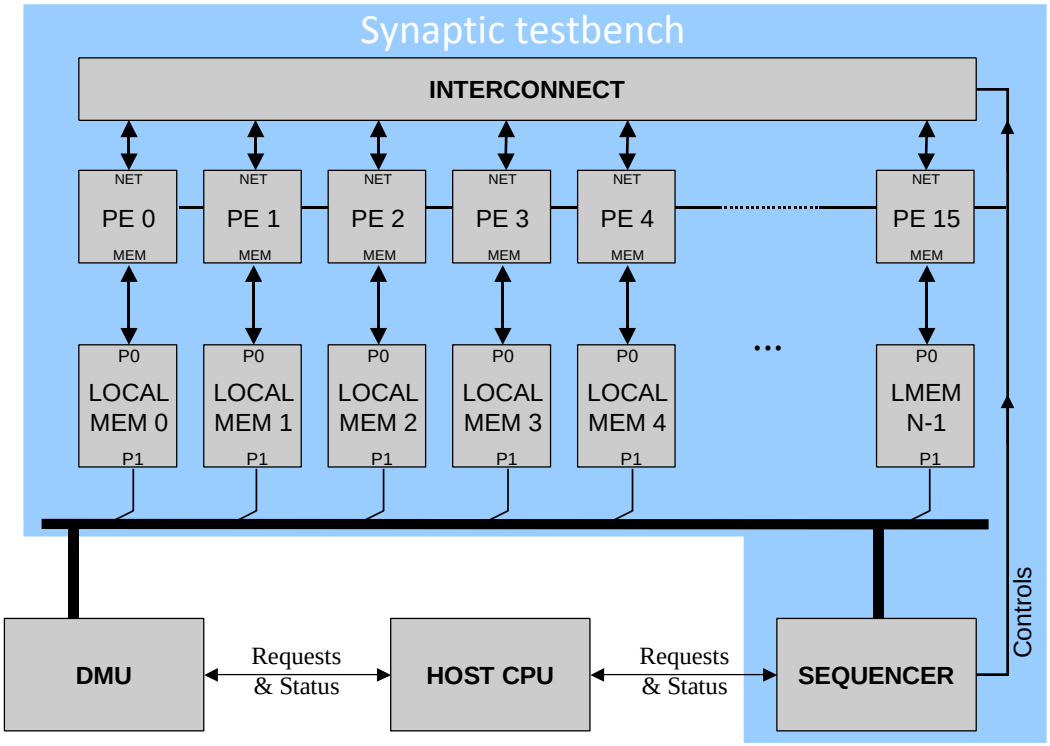


Figure 18. SIMD architecture testbench

The current implementation of the Ter@pix architecture and some architectural choices are dependant of the FPGA technology used. Redesign (at the RT level) is thus necessary for a standard-cell implementation.

The architecture will also be extended and adapted according to the results of our comparative analysis. The results of the analysis have emphasized the interest of some architectural choices in the perspective of the Synaptic approach:

- A tiled organization is an interesting approach for a step and repeat methodology at the layout level.
- A shift-register is an interconnect network that is probably easier to place and route than a ring network.

While STMicroelectronics will benchmark the full RTL for the xPE processing engine and provide selected subblocks for the early project investigation during the first year (for example a processor vector ALU). For the interconnect the RTL two bus

topologies will be leveraged, one for a standard STBus node and an equivalent STNoC interconnect.

If resources and time will allow it, STMicroelectronics will also benchmark the results against the RTL generated automatically for a selected IP functionality with an industry standard data-path generation tool such as Synfora PICO or equivalent (e.g. CatapultC).

6. References

[Power7_10] Ron Kalla, Balaram Sinharoy, William Starke, Michael Floyd, "POWER7TM: IBM's Next Generation Server Processor," IEEE Micro, 16 Feb. 2010. IEEE computer Society Digital Library. IEEE Computer Society, http://doi.ieeecomputersociety.org/10.1109/MM.2010.2

[Larrabee08] Seiler, L., Carmean, D., Sprangle, E., Forsyth, T., Abrash, M., Dubey, P., Junkins, S., Lake, A., Sugerman, J., Cavin, R., Espasa, R., Grochowski, E., Juan, T., and Hanrahan, P. 2008. Larrabee: a many-core x86 architecture for visual computing. ACM Trans. Graph. 27, 3 (Aug. 2008), 1-15. DOI= http://doi.acm.org/10.1145/1360612.1360617

[Cell05] Kahle, J. A.; Day, M. N.; Hofstee, H. P.; Johns, C. R.; Maeurer, T. R.; Shippy, D.; "Introduction to the Cell multiprocessor", IBM Journal of Research and Development, July 2005, Vol 49 Issue:4.5 , page(s) 589 – 604

[Ambric06] M. Jones, Michael Butts. TeraOPS Hardware: A New Massively-Parallel MIMD Computing Fabric IC. IEEE Hot Chips Symposium, August 2006.

[Pico03] A. Duller, G. Panesar, and D. Towner, "Parallel Processing: The picoChip Way!" Proc. Communicating Process Architectures, 2003, IOS Press, pp. 125-138.

[Intellasys06] http://www.intellasys.net/templates/trial/content/WPMeshComp.pdf

[Agarwal07] Agarwal, Anant. "The Tile Processor: A 64-Core Multicore for Embedded Processing" Available: http://www.tilera.com ,2007

[Montium03] P. M. Heysters, G. J. M. Smit, and E. Molenkamp. A flexible and energy-efficient coarse-grained reconfigurable architecture for mobile systems. Journal of Supercomputing, Kluwer Academic Publishers, 26(3): 283–308, November 2003

[Baas08] Dean Truong, Wayne Cheng, Tinoosh Mohsenin, Zhiyi Yu, Toney Jacobson, Gouri Landge, Michael Meeuwsen, Christine Watnik, Paul Mejia, Anh Tran, Jeremy Webb, Eric Work, Zhibin Xiao, Bevan M. Baas. "A 167-processor 65 nm Computational Platform with Per-Processor Dynamic Supply Voltage and Dynamic Clock Frequency Scaling." Symposium on VLSI Circuits, (VLSI '08), June 2008, pp. 22-23.

[Abb08] A. Abbo, R. Kleihorst, V. Choudhary, L. Sevat, P. Wielage, S. Mouy, B. Vermeulen, and Marc Heijligers, *"Xetal-II: A 107 GOPS, 600 mW Massively Parallel Processor for Video Scene Analysis"*, IEEE Journal of Solid-State Circuits, Vol. 43, No. 1, January 2008, pages 192-201.

[Bar04] M. Baron, "M2000's Spherical FPGA Cores", in MicroProcessor Report, December 2004.

[Ber09] M. Berekovic, A. Kanstein, B. Mei and B. D. Sutter, *"Mapping of nomadic multimedia applications on the ADRES reconfigurable array processor"*, Microprocess. Microsyst. 33, 4 (Jun. 2009), page(s) 290-294, 2009.

[Bon08] P. Bonnot, F. Lemonnier, G. Edelin, G. Gaillat, O. Ruch and P. Gauget, *"Definition and SIMD implementation of a multi-processing architecture approach on FPGA"*. In Proceedings of the Conference on Design, Automation and Test in Europe (DATE '08), Munich, Germany, March 10 - 14, 2008, p. 610-615.

[Bor05] S.Borkar, *"Designing reliable systems from unreliable components: the challenges of transistor variability and degradation"*, IEEE Micro, IEEE Computer Society, 2005, 25, page(s) 10-16.

[Bor07] S. Borkar, *"Thousand core chips: a technology perspective"*, in Proceedings of the 44th annual Design Automation Conference (DAC'07), ACM, 2007, page(s) 746-749.

[Cam07] F. Campi, A. Deledda, M. Pizzotti, L. Ciccarelli, P. Rolandi, C. Mucci, A. Lodi, A. Vitkovski and L. Vanzolini, *"A dynamically adaptive DSP for heterogeneous reconfigurable platforms"*, In Proceedings of the Conference on Design, Automation and Test in Europe (Nice, France, April 16 - 20, 2007), 2007.

[Com02] K. Compton and S. Hauck, *"Reconfigurable computing: a survey of systems and software"*, in ACM Computing Surveys (CSUR), vol. 34(2), 2002, pp. 171-210.

[Die98] K. Diefendorff, *"Katmai Enhances MMX"*, Microprocessor Report, Oct. 5, 1998, pp. 1, 6-9.

[Die00] Keith Diefendorff, Pradeep K. Dubey, Ron Hochsprung, Hunter Scales, *"AltiVec Extension to PowerPC Accelerates Media Processing",* IEEE Micro, vol. 20, no. 2, pp. 85-95, March/April 2000.

[Fra01] D. J. Frank, R. H. Dennard, E. Nowak, P. M. Solomon, Y. Taur, & H. P. Wong, *"Device scaling limits of Si MOSFETs and their application dependencies"*, in Proceedings of the IEEE, 2001, 89, 259-288.

[Kha08] B. K. Khailany, T. Williams, J. Lin, E. P. Long, M. Rygh, D. W. Tovey, and W. J. Dally, *"A Programmable 512 GOPS Stream Processor for Signal, Image, and Video Processing",* IEEE Journal of Solid-State Circuit, Vol. 43, No. 1, January 2008, pages 202-213.

[Kyo08a] S. Kyo, S. Okazaki, T. Koga, F. Hidano, *"A 100 GOPS in-vehicle vision processor for pre-crash safety systems based on a ring connected 128 4-Way VLIW processing elements"*, IEEE Symposium on VLSI Circuits, 18-20 June 2008, pages 28-29.

[Kyo08b] S. Kyo, and S. Okazaki, *"In-vehicle vision processors for driver assistance systems"*. In Proceedings of the 2008 Asia and South Pacific Design Automation Conference (Seoul, Korea, January 21 - 24, 2008), pages 383-388, 2008.

[Muc09] C. Mucci, D. Rossi, F. Campi, L. Ciccarelli, M. Pizzotti, L. Perugini, L. Vanzolini, T. De Marco, M. Innocenti, *"The Dream Digital Signal Processor"*, chapter 5 in "Dynamic System Reconfiguration in Heterogeneous Platforms - The Morpheus Approach" Edited by Nikolaos S. Voros - Alberto Rosti - Michael Henry Hubner, Lecture Notes in Electrical Engineering / Springer Publisher, June 2009.

[Obe99] S. Oberman, G. Favor, F. Weber, *"AMD 3DNow! technology: architecture and implementations"*, IEEE Micro, March 1999.

[PACTa] PACT XPP Technologies, "XPP-III Processor Overview", white paper, version 2.0.1, July 13, 2006.

[PACTb] PACT XPP Technologies, "Programming XPP-III Processors", white paper, version 2.0.1, July 13, 2006.

[Ros09] D.Rossi, F.Campi, A.Deledda, S. Spolzino, S.Pucillo, *"A Heterogeneus Digital Signal Processor Implementation For Dynamically Reconfigurable Computing"*, CICC 2009, San Jose, USA.

[Sch09] E. Schüler and M. Weinhardt, *"XPP-III: The XPP-III Reconfigurable Processor Core"*, chapter 6 in "Dynamic System Reconfiguration in Heterogeneous Platforms - The Morpheus Approach" Edited by Nikolaos S. Voros - Alberto Rosti - Michael Henry Hubner, Lecture Notes in Electrical Engineering / Springer Publisher, June 2009.

[SPI] Stream Processors, Inc., "Stream Processing: Enabling the new generation of easy to use, high-performance DSPs", white paper, 2008.

[Xilinx04] Xilinx, *"XtremeDSP Design Considerations User Guide"*, UG073 (v1.1) September 9, 2004.

[Xilinx07] Xilinx, *"Virtex-4 Family Overview"*, datasheet, v3.0, 2007.

[Xilinx08] Xilinx, *"MicroBlaze Processor Reference Guide - Embedded Development Kit EDK 10.1i"*, UG081 (v9.0), January 17, 2008.

[Xilinx09] Xilinx, *"Virtex-6 FPGA Configurable Logic Block User Guide"*, UG364 (v1.1) September 16, 2009.

[Salminen08] E. Salminen,A. Kulmala and T. D. Hämäläinen, "Survey of Network-on-chip Proposal," White paper, OCP-IP, Mar. 2008.

[STNoC05] http://www.st.com/stonline/press/news/year2005/t1741t.htm

[Arteris10] http://www.arteris.com/technology.php

[STNoC04] M. Coppola, R. Locatelli, G. Maruccia, L. Pieralisi, and A. Scandurra. Spidergon: A Novel On-Chip Communication Network. In SOC 2004: International Symposium on System-on-Chip, November 2004.

[STBus] Overview on the STBus – available on http://www.stmcu.st.com

7. Glossary

ALU	Arithmetic Logic Unit
ASIC	Application-Specific Integrated Circuit
CMOS	Complementary Metal–Oxide–Semiconductor
CP	Control Processor
CPU	Central Processing Unit
DCT	Discrete Cosine Transform
DIP	Data Input Processor
DLL	Delay Dock Loop
DLP	Data Level Parallelism
DMA	Direct Memory Access
DMU	Data Mover Unit
DOP	Data Output Processor
DPU	Data Parallel Unit
DSP	Digital Signal Processing
ECC	Error Correction Code
EIF	External Interface
FFT	Fast Fourier Transformation
FIFO	First In First Out
FPGA	Field Programmable gate array
FPU	Floating-Point Unit
FU	Functional Units
GALS	Globally Asynchronous Locally Synchronous
GP register	General Purpose Register
GPP	General Purpose Processors
ILP	Instruction Level Parallelism
IO	Input/Output
ISA	Instruction Set Architecture
LIW	Long Instruction Word

LUT	Look-Up Table
MAC	Multiply/Accumulate
MIMD	Multiple Instruction, Multiple Data
MISD	Multiple Instruction, Single Data
MPPA	Massively Parallel Processor Array
MPSoC	Multiprocessor System-on-Chip
NoC	Network on Chip
NUMA	Non-Uniform Memory Access
OFDM	Orthogonal frequency-division multiplexing
OOO	Out-Of-Order
OS	Operating System
P&R	Place and Route
PE	Processing Element
PLL	Phase Lock Loop
PPE	Power Processing Element
RAM	Random Access Memory
RF	Register File
RF	Register File
RISC	Reduced instruction set computing
ROM	Read Only Memory
SIMD	Single Instruction, Multiple Data
SISD	Single Instruction, Single Data
SMP	Symmetric Multiprocessing
SMT	Simultaneous Multi Threading
SoC	System on Chip
SPE	Synergistic Processing Element
SWAR	SIMD Within A Register
UMA	Uniform Memory Access
VLIW	Very Long Instruction Word
Term	Explanation