A Heterogeneous Digital Signal Processor for Dynamically Reconfigurable Computing

Article *in* IEEE Journal of Solid-State Circuits · September 2010

DOI: 10.1109/JSSC.2010.2048149 · Source: IEEE Xplore

A Heterogeneous Digital Signal Processor for Dynamically Reconfigurable Computing

Davide Rossi, Fabio Campi, Simone Spolzino, Stefano Pucillo, Roberto Guerrieri

***Abstract*- This paper describes a System on Chip implementation of a reconfigurable digital signal processor. The device is suitable for execution of a wide range of applications exploiting a balanced mix of heterogeneous reconfigurable fabrics merged together by a flexible and efficient communication infrastructure based on a 64-bit Network On Chip. The SoC combines a fine grain embedded FPGA, a mid grain configurable processor and a coarse grain reconfigurable array. An ARM processor featuring a resident operating system is the SoC supervisor, managing communication, synchronization and reconfiguration mechanisms. This computational model enables the programmer to manage the high level synchronization and global data of complex signal processing applications through the ARM processor, while allocating most critical computational kernels to the most suitable reconfigurable engines. The SoC has been fabricated in 90-nm technology, the die area being 110 mm^2; it integrates 97 million transistors and has a peak power consumption of 2.5 W. In order to demonstrate the proposed computational model and the reconfigurable signal processor capabilities in a real test case, a video surveillance motion detection application was implemented in the SoC. When running this application, the device proved able to deliver 120 GOPS consuming 1.45 W.**

***Index Terms*—Digital Signal Processors, Globally Asynchronous Locally Synchronous, Network-On-Chip, Reconfigurable architectures, System-On-Chip.**

I. Introduction

Reconfigurable computing is commonly seen as a computing paradigm that combines performance of spatial computations typical of hardware platforms with the runtime programmability of general purpose microprocessors [1]. In many fields of embedded computing, reconfigurable devices are viewed with interest for their ability to update functionality after fabrication, thus reducing the NRE costs associated with fabricating ASIC designs.

This concept has been deployed in the last few years at various levels utilizing different approaches. FPGAs are commonly utilized in many fields of signal processing, because of their fine-grained structure based on SRAM look-up tables (LUTs) that allow designers to implement any kind of logic function [2][3]. However, the bit-level programmability of such designs introduces area and power overheads which often do not match the constraints introduced by most embedded environments. Another issue related to reconfigurable computing is programming productivity: hardware oriented languages are intrinsically more complex and difficult to use than software oriented languages. While it is possible to rely on pre-packaged libraries and IPs for standard computation kernels, development of the top-level wrapping and synchronization stage becomes a significant slowing factor in the application development time.

Coarse Grain Reconfigurable Arrays (CGRA) [4] represents a second class of reconfigurable architectures [5][6][6][8]. CGRA reduce the overheads typical of fine grain FPGAs exploiting a coarser operand grain, replacing LUTs with coarser computational blocks and simplifying interconnect patterns. They can hence provide more aggressive power/area trade-offs, while dispensing with some of the flexibility provided by FPGAs. Unfortunately, in most cases the definition of coarser processing units forces the designer to focus on a specific application field, thus losing the general purpose advantages of the approach.

A mix of the described approaches, has been proposed in [9], which describes a heterogeneous reconfigurable digital signal processor. The device targets baseband wireless signal processing and is capable to achieve significant energy efficiency in this application field combining an embedded microprocessor coupled with an array of computational units of different granularities, address generators and memories, connected by a hierarchical reconfigurable interconnect network.

A novel computation pattern emerging in the last few years is that of processor arrays [10][10][12]. Processor arrays could be described as the upper bound of CGRAs, in the sense that they are reconfigurable architectures of maximum granularity. They exploit parallelism at a thread rather than an instruction level, which is definitely friendlier from the user point of view. However, processor-oriented computation does not match the flexibility of FPGAs for bit-oriented computation, nor the density of CGRAs for massively parallel SIMD computation.

Another category of device that exploits the presence of many simple cores organized in hierarchical structures is the GP-GPU [13][14]. These devices are a generalization on graphics processors, and invaded the market due to their user-friendly programming models and huge floating point computational densities. At the moment, their usage is restricted to the supercomputing field since their intrinsic power and area overheads are not suitable for many embedded applications.

One possible alternative in this flexibility versus performance tradeoff scenario lies in multi-core heterogeneous computing architectures, where different flavours of computing fabrics are mixed, so that any application can be distributed entrusting each subtask to the most suitable computation flavour and grain. In this direction, most commercial Systems-on-Chip for embedded computation-intensive applications strive to match performance and flexibility, exploiting a carefully weighted mix between application-specific ASIC acceleration, powerful programmable DSP cores and controlling processors[15][16]. This approach has proved quite effective for the current generation of high-performance embedded computers, and relies on very careful selection and design of the algorithmic kernels to be mapped on the hardware. Hardwired engines must be specific enough to meet power/area constraints, though general enough to support post-fabrication updates. As a consequence, related products tend to be application-specific and feature limited market windows, related to the trends in the applicative standards they support: any major redesign of the application would make the hardwired block obsolete. Considering the costs related to technology scaling, such a design approach does not appear an ideal solution in the long term.

This paper describes a multi-core, heterogeneous signal processor aimed at a broad range of embedded applications. The proposed SoC is code-named Morpheus [17] and consists of a mix of the above described approaches. The main goal of Morpheus is to re-use the standard SoC template described above, based on a standard processor acting as “house-keeper” and main user interface, surrounded by high-performance acceleration engines to speed-up the critical application kernel. In order to match run-time programmability with performance, Morpheus replaces standard ASIC acceleration with a balanced mix of three different flavours and granularities of reconfigurable hardware fabrics, connected through a communication infrastructure based on an 8-node, 64-bit Network On Chip (NoC) [18].

Application mapping is organized at three levels [19]. The first level consists in exploring the application, dividing it into concurrent kernels and the working out the most suitable of the three reconfigurable fabrics for mapping each kernel, depending on features such as parallelism available, operand bit-width and inter-kernel data dependencies. At the second level each kernel is implemented on the specific fabric making use of the fabric proprietary tools. The third level aims at synchronizing the data flow and relative dependencies. A complete software development kit equipped with a profiler, support for application partitioning and mapping on the reconfigurable devices, data movement abstraction aids the programmers in this exploration, partitioning and mapping process [20]. The extracted kernel can be further optimized making use of the reconfigurable devices proprietary tools that will be introduced in section IV. In order to introduce the proposed programming model a motion detection video surveillance test case application implementation will be described, with particular focus on the application partitioning and data flow structures utilized to massively parallelize the execution of tasks on reconfigurable devices.

This paper is organized as follows. The next section describes the system architecture. Section 3 describes the SoC communication infrastructure. Section 4 reviews the three reconfigurable engines composing the computation core of the system. Section 5 details some quantitative results of chip implementation. Finally, section 6 provides a description of a test case application mapping on the proposed SoC.

II. System Architecture

Fig. 1 describes the system architecture proposed. The SoC is built around three so-called heterogeneous, reconfigurable engines (HREs) which target three different computation styles. These IPs, provided by industrial vendors, were selected due to their complementary capabilities, introduced in the system as RTL entities and finally implemented and integrated in the design as mix of custom and synthesizable standard cell based macros.

XPP-III [21] is a stream processor featuring an array of coarse grain (16-bit) computational elements communicating through a matrix of configurable data channels. It mainly targets streaming, arithmetic oriented computations with regular dataflow structures and operand width such as FFT [22], DCT [23], FIR filtering.

DREAM [24] is a reconfigurable processor composed of a RISC processor coupled with a mid-grain (4-bit) heavily pipelined reconfigurable datapath. The focus of DREAM is to exploit the instruction level parallelism present in the computational kernels of a wide range of applications such as multimedia [25], redundancy check [27], cryptography [28].

Flexeos [29][30] is an embedded Field Programmable Gate Array (eFPGA) based on 1-bit Lookup Tables (LUT). It is suitable for bit-level manipulations, and arbitrary logic implementation. One specific target usage of the eFPGA is to implement programmable IO peripherals [25] by means of 30 GPIO pins exported to the pad frame.

A set of test application kernels implemented on the three reconfigurable IPs together with architectural and area-oriented evaluations driven the selection of the most appropriate size of the introduced IPs. A configuration of a 5x6 coarse-grain computing elements was chosen for the XPP-III processor, an array of 16x24 4-bit computing elements was chosen for the DREAM reconfigurable processor, while a size of 4K LUTs was chosen for the Flexeos eFPGA.

HREs are encapsulated in independent clock islands, dynamically controlled via software. Frequency synthesis for the three islands is performed by three separate PLLs. This solution has the advantage of allowing fine grain selection of operating frequencies for each of the three computational engines, enabling the user to carefully tune the optimal power versus performance trade-off for each application. The drawback of this solution is that each PLL frequency re-setting requires a 400us locking time, but given the long configuration time of each HRE this overhead proves insignificant.

A fourth, non dynamically tunable frequency domain comprises all chip-level communication, peripherals, and an ARM 926EJ-S RISC processor acting as system supervisor. The core is equipped with 16K I-cache and D-cache, plus 16K software-managed D- and I- Tightly Coupled Memories (TCM) and a standard set of peripherals connected through a specific AMBA-APB peripheral bus.

The SoC memory architecture is organized on three levels of hierarchy, that can in turn be logically divided into a data layer and a configuration layer (Fig. 2). ARM TCM, and HRE local buffers (DEB and CEB) represent the first level of memory hierarchy, local to each functional unit. A second level is composed of 512Kb of on-chip SRAM, which is conventionally split into 256 Kb data memory and 256 Kb configuration memory. The third and last level is represented by the external off chip memory which stores both configuration and data.

Communication in the SoC is organized along three independent, orthogonal contexts: system control and synchronization, application data transfers, and configuration bit-stream transfers.

A. Synchronization and Control

ARM runs a resident operating system, and manages all communication, synchronization, and reconfiguration of the SoC by means of a dedicated “Main” AMBA-AHB bus. All computation (HREs), communication (NoC, Peripherals) and Configuration (bitstreams for HREs and relative means of communication) resources in the system are controlled by set of control registers mapped on this bus. The bus is hence critical, but since it carries only control information at computation time, bandwidth is not considered a significant issue. For debugging purposes, the main bus is also capable of accessing all data-storage resources in the system but this feature is not utilized in normal computation.

B. Data Communication

Data are exchanged between each HRE and the ARM/NoC domain by means of a set of Data Exchange Buffers (DEBs). DEBs are dual port, dual clock memory banks that act as local data storage for HREs as well as providing safe clock domain crossing. DEBs are seen by ARM and NoC as a single and coherent addressing space. On the other hand, HREs can only address/access data in the local DEBs and have no other visibility of the external world. Data dependencies and computation synchronization between HREs and the ARM domain are resolved by software via a set of exchange registers (XR) mapped in the DEB addressing space. Depending on the nature of the HRE and of the features of the application kernels deployed, DEBs can be configured by ARM as FIFOs or Random Access Memories (RAM). Configuration bits are transferred similarly through dedicated Configuration Exchange Buffers (CEBs).

C. Dynamic Reconfiguration

Morpheus fully supports dynamic reconfiguration, so that each HRE can be reconfigured while the others are computing. With the exception of the eFPGA, HREs are also multi-context, meaning that configuration bitstreams can be cached into internal configuration memory and are capable to switch their functionality in one clock cycle. Configuration bit-streams flow through an independent AMBA-AHB “Configuration bus”. Apart from the ARM core, masters of this bus are a specific configuration DMA and the so-called Pre-fetch Configuration Manager (PCM).

The configuration DMA is normally programmed by the ARM core so as to handle long bit transfers relating to the HRE reconfiguration process. In cases of intensive dynamic reconfiguration activity, the control work load would become repetitive but too demanding for the core. The PCM is a hardware FSM that allows a list of configuration bit-streams stored in the external memory to be tabulated off-line. During computation it will monitor HRE activity and will schedule bit-stream pre-fetching in order to hide reconfiguration latencies as much as possible.

III. Communication Infrastructure

The Morpheus data communication infrastructure is based on a 64-bit, 8-node STNoC [26] provided as RTL IP. The NoC is composed of three basic blocks: the router (one request and one response router per node), the network interface and the physical link. Connections between NoC routers define the topology of the NoC Fig. 3. The NoC connects up the computational resources of the SoC (XPP-III, DREAM, eFPGA, ARM) and to the available data storage elements (main memory, configuration memory, external memory).

Two kinds of node go to form the NoC infrastructure: initiator nodes are able to generate traffic on the NoC requiring transfers between the local network interface and other nodes; target nodes represent passive elements of the NoC, only used as targets for a given transaction. One fundamental target in the system design is to guarantee a homogeneous programming model capable of abstracting the end user from the specificity of each HRE. In particular, the end user can be provided with appropriate kernel libraries in order to abstract as much as possible from the programming details and tools that are specific to each HRE. But data transfers need to be specified according to a global mapping strategy, and as such they would require the user to utilize different programming styles and languages depending on the HRE issuing the transfer. The definition of the DEBs, topped by a software semaphoring mechanism, allows one to de-couple local access from global transfers: HREs only compute locked chunks of data, but are not involved in global transactions.

Chip level transactions are handled by a set of two-port DMA engines, each local to a given HRE Network Interface (HRE-NI). One port drives the initiator port of the network interface while the secondary port is connected to the HRE local buffers. NoC DMAs are programmed, triggered by ARM via the Main AMBA-AHB bus, and consequently generate traffic on the NoC channels.

In order to handle this distributed architecture, a specific Direct Network Access (DNA) controls all the NoC DMAs. Besides providing a set-up mechanism for data transfer, the DNA is responsible for hardware handshaking between communication and computation. In addition, auto-reload multi-block transfers can implement a full streaming transfer pattern suitable for the XPP-III processor or the eFPGA.

NoC implementation was realized following a hierarchical approach: the router was implemented separately and included in the course of design as a custom macro during top level implementation. The site of the router in the final design was carefully selected in order to constraint the place & route tools for placing the HRE-NI cells, and as far as possible to balance the NoC physical link routing, avoiding congestion areas and unduly long wires.

Fig. 4 describes the logical connections between NoC nodes which define the chip layout topology while Fig. 5 shows the florplanning of the NoC components (routers and network interface). Implementation details of the whole communication infrastructure are reported in TABLE I.

IV. Reconfigurable Cores

The three HREs that compose the computational core of the system were carefully selected in order to provide complementary computational capabilities, both in terms of bit granularity and in terms of exploitation of algorithmic parallelism.

A. XPP-III

The XPP-III [21] is a dynamically programmable coarse grain reconfigurable signal processor geared to stream processing and exploitation of data-level and instruction-level parallelism on 16-bit arithmetic. XPP-III is in itself a heterogeneous core, combining two 16-bit VLIW cores suited to control-flow dominated algorithms (Functional Processing Elements, or FNC-PAEs) and a reconfigurable Array of 16-bit ALUs for highly parallel streaming applications.

XPP-III computation is self-synchronized, with 4 input, 4 output 16-bit data streams, for a peak 9.6Gbit/s bandwidth. Data streams flow to/from the DEBs acting as dual-clock FIFOs, into the XPP proprietary interconnect, and eventually through the Array PAEs or the FNC-PAEs. Fig. 6 shows XPP-III's physical layout. The current implementation of the Array features 5x6 16-bit computational cores (ALU-PAE, centre), plus 2x6 16-bit storage elements (RAM-PAE) on the side columns. Each ALU PAE comprises a 16-bit ALU object providing multiplication, addition, comparison, sort, shift and boolean operations; a forward register path providing data stream control such as multiplexing and swapping; and a backward register object that provides routing paths in vertical direction and a small ALU. RAM PAE is the data storage element of the array, comprised of 512x16-bit SRAM, embedded streaming ports that implement the Array IO plus one forward and one backward register path, equivalents to the ones presents in the ALU-PAE.

XPP-III also provides 4D-DMA controllers that convert chip level data streams into XPP internal memory access patterns for either the FNC-PAEs, the Array, or an 8192x64-bit XRAM used to buffer local data. Configuration bitstream for the array can be either stored in a local 8192x64-bit CEB or in the off-chip SRAM memory which is mapped in the XPP-III addressing space. XPP-III implementation results are summarized in TABLE II.

The XPP-III IP is equipped with a complete Software Development Kit coupled with a cycle-accurate simulator and which allow the programmer to partition and develop applications for both FNC and the Array starting from ANSI-C or C++ code [31].

B. DREAM

DREAM [24] is a reconfigurable digital signal processor composed of a 32-bit RISC core tightly coupled to the PiCoGA-III reconfigurable datapath. It mainly targets heterogeneous, computation intensive kernels with variable data-width, iterative and a complex data addressing mechanism. On the other hand, it is not suited for multiplication-oriented computations, with regular operand sizes (8- or 16-bit).

PiCoGA-III [32] is a matrix of 16x24 Reconfigurable Logic Cells (RLC). Each cell features two 4-bit inputs, and one 4-bit output. It is composed of a 4-bit LUT, a 4-bit ALU, a multiplier slice and logic for implementing Galois-field multiplication. A carry chain logic implements the 16-bit and 32-bit arithmetic composing adjacent RLCs, while . Each row of the PiCoGA is strictly synchronous, and represents a stage in the pipeline implementing an operation or a set of concurrent operations. This heavy pipelined structure allows for a very high computational density in the array. An embedded control unit selectively activates the rows, in order to maintain data dependencies. PiCoGA supports 4 independent configuration contexts: reconfiguration is allowed on one context while another one is running. Each configuration context can occupy up to 2 Kbytes, that can be loaded in 300 clock cycles from the dedicated 9-bank 1024x32-bit CEB memory.

DREAM draws on 16 1034x32-bit dual-port dual clock memory banks of local data storage (DEB). In order to allow high bandwidth between DEBs and PiCoGA, from the DREAM side, DEBs are read by 16 address generators. Addressing patterns are configured by the embedded RISC core, providing step/stride patterns to enable non-contiguous vectorized addressing. A configurable interconnection crossbar allows full flexible routing between each DEB and each PiCoGA input or output.

The embedded 32-bit RISC processor handles configuration, datapath activation and program flow management, DEB addressing and synchronization with the ARM/NoC, as well as working as a computational engine concurrently with the configurable datapath. Fig. 7 shows the DREAM layout. Implementation details are shown in TABLE III.

The device is equipped with a tool chain used to configure the PiCoGA in order to efficiently implement pipelined data flow graph described with a restricted subset of ANSI-C enhanced with some extension to handle variable resizing and register allocation inside the PiCoGA. In addition a cycle accurate simulator of the DREAM environment allow the programmer to profile applications or part of applications mapped on the reconfigurable engine [33].

C. eFPGA

The eFPGA HRE is built around the Flexeos custom macro [30]. Flexeos is a standard FPGA, composed of 4K multi-function logic cells (MFC) based on SRAM 1-bit Lookup-Tables (LUT), implemented using standard CMOS technology. As a result, it can be programmed using HDL to implement bit-oriented, fine grained applications, or to map run-time tunable IO peripherals. As an example, the eFPGA has been successfully utilized to implement a 10 Mbit Ethernet controller [34]. In order to support the described "Programmable IO concept", 30 general-purpose pins (GPIO) from the Flexeos macro were exported to the chip pad frame.

As well as its eFPGA macro functions, the HRE comprises a bitstream loader driven by the configuration bus. The loader is responsible for managing configuration, operating modes and testing. Flexeos does not support dynamic reconfiguration. Hence, CEBs were not included in this island.

The Flexeos macro is connected to 4-input and 4-output 1024x32-bit DEBs, and this allows the engine to achieve a maximum bandwidth of 12.8 Gbit/s in both directions. DEBs can be configured either as FIFOs or as standard addressable memories to exploit device flexibility to the full. Fig. 8 shows the eFPGA-based HRE layout. The Flexeos macro is placed at the top of the design, while the 8 dual port dual clock memories are fitted on the bottom side. The loader and the DEB control units were synthesized at the top level of the island. TABLE IV shows eFPGA implementation details.

The Flexeos device is equipped with a dedicated synthesis and place & route flow, which allows the end user to design accelerators or peripherals on the device starting from a HDL-based description of the selected entities [30].

V. Chip Description And Measurements

A. Overview

The SoC was fabricated using 90 nm 7-metal CMOSGP technology and integrates 97 million transistors in a 110 mm^2 die. Fig. 9 shows a chip photograph, while the chip's main features are summarized in TABLE V. Fig. 10 shows the test environment for the Morpheus prototype, including a printed circuit board hosting a Morpheus chip and a supporting FPGA board to interface on board SRAM with a host system.

The SoC is composed of a mixture of custom-designed digital macros (PiCoGA and eFPGA), embedded SRAM memories and standard cell regions, partitioned as described in Fig. 11. The three reconfigurable engines were designed separately, and re-utilized as hard macro-blocks to partition and better organize the physical design effort [35]. HREs are located on three different clock islands and positioned at the chip corners to ease global routing. The XPP-III macro is flipped horizontally so as to better match the input/output ports with NoC topology; it is placed on the bottom-right side of the chip. The DREAM macro is placed top right, the eFPGA in the top left corner of the die, while the ARM processor macro, working at the system clock frequency is placed in the middle of the chip. The PLLs are placed on the four boundaries of the die in order to avoid coupling noise among their analog supplies. Morpheus area breakdown by logical entities is reported in Fig. 12.

Each HRE features two clock inputs. One global clock is used to feed the system-side of the synchronization barriers (DEBs, CEBs and XRs) and was properly balanced in order to compensate for insertion delays by the internal clocks. Each HRE can be clocked either by the global system clock, or by its private clock, programmed by the ARM setting PLL division factors on specific memory-mapped registers. This mechanism allows one to exploit Globally Asynchronous Locally Synchronous techniques by enabling dynamic frequency scaling on the three auxiliary cores.

B. Power

The power consumption by each reconfigurable device in the system depends not only on the working frequency of each clock island, but also on the number of resources utilized (PAE for XPP, RLC for DREAM, LUT for eFPGA), on the routing path, as well as on the number of concurrent accesses to memory banks or FIFOs. Fig. 13 shows measurement on the DREAM processor when running in idle mode (RISC processor only), when using half of the available rows fetching data from its internal register, when using the all PiCoGA rows, and when using all 24 rows while accessing all I/O memory banks. These experimental results, which can be extended to the other reconfigurable devices of the system, show significant dependence between the power consumption and the quantity of utilized resources. The power characterization of the chip was therefore realized running ad-hoc test vectors that utilizes all the resources of each reconfigurable engine. Fig. 14 reports dynamic power measurements performed running these test vectors on the Morpheus prototype. In details, the ARM clock island power was measured turning off the clocks of all HREs in the system, and programming all DMAs of the system in order to iteratively perform transfers among all the NoC nodes. This number consider power consumed by the ARM, DMAs, the whole NoC, and all the storage elements connected to the NoC. On the other hand, figures relative to the HREs were calculated running the test vectors on the target HRE and subtracting from the measured values the idle power consumed by the ARM clock island. In both ARM and HREs measurement extrapolation the leakage power was considered as an offset subtracted from the measured values.

Although gating techniques could be applied in order to reduce the idle power of the design, a global power reduction strategy has been considered out of the project scope in order to reduce risk associated to the development of the Morpheus prototype. On the other side in several implementation of the PiCoGA datapath as well as in the one realized for the Morpheus system, a set of low-power techniques were applied in order to reduce both idle and static power [36].

C. *Performance Figures*

In order to evaluate the Morpheus performance from a theoretical standpoint, we classified the granularity of operations with increasing complexity and we evaluated the affinity of each reconfigurable device composing the system in relation with each granularity. The operations classification started from logic operation of different output width (1,4), arithmetic operation (sum, sub, shift, comparison..) of increasing operand width (4,8,16,32) up to multiplications with 16- and 32-bit width operands. Fig. 15 shows the theoretical performance delivered by each HRE for a wide range of operation granularities. The eFPGA performance were estimated evaluating synthesis results utilizing the eFPGA proprietary tools, since the most limiting factor has been considered to be the routing in such device. For the coarser reconfigurable engines, less sensitive to routing issues, performance were estimated analyzing the amount of available computing elements. Equivalent GOPS are reported in a logarithmic scale due to the enormous differences when dealing with so heterogeneous application domains. Performance is also normalized according to power density (mW/MHz) of each reconfigurable engine in Fig. 16.

If we define an operation as a 16-bit arithmetic calculation, Morpheus is capable of delivering 50 GOPS @1V with a power efficiency of 20 GOPS/W and an area efficiency of 0.45 GOPS/mm^2. TABLE VI shows a comparison between Morpheus GOPS and other State-of-The-Art devices. The Morpheus SoC exploits a carefully balanced trade-off between flexibility and power/area efficiency which falls between FPGAs and Application Specific Signal Processor (ASSP). The choice of a 16-bit reference operations favors architectures such as processor arrays, which can achieve better performance. But even for some of these, insertion of hardwired accelerators proved necessary for efficient execution of portions of applications not suitable for the processor [12] (e.g. Viterbi decoding, FFT).

A set of different applications including a film grain removal algorithm [40], an in-band reconfigurable network node [41] has been implemented and validated on the Morpheus prototype, while portions of the 802.16e/j mobile wireless application [34] have been implemented on the Morpheus simulator. Mapping of the proposed applications on the SoC demonstrated the platform being able to match computation targets of a wide range of application domains. Moreover implementation results shown an utilization of Morpheus resources bigger than 70%, demonstrating the good trade-off between chosen granularity and sized of the three reconfigurable engines. As proof of concept for the proposed computational model, as well as demonstration of concept usability, the following section will describe the deployment of a video surveillance motion detection application. The section will provide details about application partitioning, dataflow structures and mapping of kernels on the HREs.

VI. Test Case Application

This section describes the implementation on Morpheus of a video motion detection application used in security and surveillance systems. The aim of the proposed algorithm is to detect the presence of external objects on a video transmitted by a camera framing a fixed background. As described in Fig. 17, the application can be partitioned into a few main kernels which are distributed so as to build a balanced streaming pipeline through the NoC over the various different HREs. From such kernels, data/instruction level parallelism is then extracted and exploited on the chosen configurable fabric.

For each video frame the first algorithmic stage performs the sum of absolute differences (SAD) between the current and the background image. The resulting maximum value is extracted and used to calculate the threshold for binarization. The binarized image is then processed by three spatial operators. Erosion and dilatation implement the opening kernel which de-noise the binarized image, while the edge detection implemented through a bi-dimensional Sobel convolution algorithm, creates the external object boundary. If an external object is detected, the final merge kernel returns the highlighting of that object on the original frame. TABLE VII shows the profiling on an ARM 926 EJ-S processor of the proposed application and the kernel mapping on the reconfigurable cores.

The first stage of the application, being strongly arithmetic on 8-bit pixels, fits nicely on the XPP processor. Similarly, the binarization stage is efficiently implemented on the eFPGA. The core of the computation (opening and edge detection) fits well on the DREAM processor. These kernels are implemented with many sequential iterations of morphological operators on the binarized image. Thus, such an image can be stored in the DREAM local buffers and iteratively processed by the datapath. Moreover, these kernels, which have a native nature of 8-bitwidth arithmetic, can be implemented using bitwise operators thanks to the elaboration on the binarized image, perfectly matching the mid-grain nature of the PiCoGA datapath. Finally, the last merging stage again involves 8-bit arithmetic. Mapping on XPP could be an option, but that would require significant NoC transfers, and some time-multiplexing over the XPP array. In addition, being the last stage in the computation, it requires data packaging for which a RISC processor is more suited. Since the stage is not overly critical, it can be performed on ARM without affecting overall performance.

In order to determine the most suitable balance between computation throughput and data transfer, granularity was determined as 80x60 8-bit pixel image chunks, for which an optimized ANSI-C reference software solution [41] implemented on the ARM 9 processor has a cost of 715 cycles/pixel. Image chunks are processed by a 4-stage coarse grain pipeline managed by the ARM processor through specific synchronization events. More precisely, the completion of a transfer stage is notified by the communication DMAs, while the completion of a computation stage is notified by the HRE through the exchange registers. If the HRE DEBs, described in section 2, are utilized in FIFO mode, computation is transparent and the conclusion of a given stage is notified by the communication engine. This is the case with SAD and binarization in this example. If the HRE DEBs are programmed in RAM mode, as occurs with erosion/dilatation and edge detection, the conclusion is notified by the HRE itself.

Fig. 17 describes the application implementation on Morpheus. At the first pipeline stage the reference image and the background image are loaded onto the main on-chip memory. During the second stage, image chunks are processed as a streaming pipe which flows through XPP and the eFPGA, performing sequentially SAD and binarization, and are finally stored on DREAM DEBs. In the third pipeline stage DREAM processes the binarized image chunks, internally iterating erosion, dilatation, Sobel vertical and Sobel horizontal operations. In the last stage the ARM processor merges the reference image with the results of overall computation and stores the final image in the external memory. Configuration management is not necessary for this application, since all the configuration bitstream can be loaded off-line on reconfigurable devices. XPP and eFPGA maintain the same configuration during execution of this whole application, while the DREAM processor can exploit its reconfiguration capabilities, loading each of the four kernel bitstreams onto one configuration context.

Considering a CCTV 640x480, 30 Frames/second grayscale video with 8-bit pixels, the real time bandwidth is 9.2 Mpixel/s (74 Mbit/s). Assuming partitioning as described, the critical kernel remains Opening/Edge Detection, which is performed on the DREAM processor at a computational cost of 1.27 cycles/pixel, thus leading to a real-time frequency for the reconfigurable core of 12 MHz. Considering the other computation cores, PACT can perform Subs/Abs/Max kernel @ 0.25 cycle/pixel, while the eFPGA can perform the Binarization kernel @ 0.125 cycle/pixel. Since the two kernels belong to the same pipeline stage, the real time frequencies for the two devices are respectively 5 and 2.5 MHz. With the described frequency configuration, measurement on the Morpheus test chip shown a power consumption of 600 mW.

On the other hand, when working at the maximum computational power, it is possible to process videos coming from up to 7 cameras. The bottleneck in such case is represented by the external memory controller which is capable of providing 1.6 Gbit/s. A dynamic memory controller would have been more appropriate in this context, but couldn't be inserted in the system for reasons of cost. Removing this limiting factor (accesses to external memory) and using maximum computational power, Morpheus is finally capable of processing videos from up to 16 cameras concurrently, delivering the equivalent of 120 GOPS while consuming a measured power of 1.45W.

VII. CONCLUSIONS

In this paper a reconfigurable heterogeneous digital signal processor has been presented. The device is able to cover many application domains, delivering from tens to hundreds of GOPS thanks to the presence of three reconfigurable engines differing in nature and granularity. Application mapping has demonstrated that the device delivers 120 GOPS on a video surveillance motion detection application with a power consumption of 1.45W. The device size is 110 mm^2 and the peak power consumption is 2.5 W.

REFERENCES

[1] A. DeHon, "The Density Advantage of Configurable Computing", *IEEE Computer*, vol. 33, no. 4, Apr. 2000, pp. 41-49.

[2] www.altera.com

[3] www.xilinx.com

[4] R. Hartenstein, "A Decade of Reconfigurable Computing: A Visionary Retrospective", *Proceedings of the IEEE International Conference on Design, Automation and Test in Europe*, Mar. 2001, pp. 642-649.

[5] E. Mirsky, A. DeHon, "MATRIX: A Reconfigurable Computing Architecture with Configurable Instruction Distribution and Deployable Resources", *Proceedings of IEEE International Symposium on Field-Programmable Custom Computing Machines*, Apr. 1996, pp. 157-166.

[6] T. Miyamori, K. Olukotun, "REMARC (abstract): reconfigurable multimedia array coprocessor", *Proceedings of international Symposium on Field Programmable Gate Arrays*, 1998, pp. 261.

[7] S. C. Goldsteiny, H. Schmit, M. Moe, M. Budiuy, S. Cadambi, R. R. Taylor, R. Laufer, "PipeRench: A Coprocessor for Streaming Multimedia Acceleration", *Proceedings of the IEEE International Symposium on Computer Architecture*, 1999, pp. 28-39.

[8] H. Singh, M.-H. Lee, G. Lu, F.J. Kurdahi, N. Bagherzadeh, E.M. Chaves Filho, "MorphoSys: An Integrated Reconfigurable System for Data-Parallel and Computation-Intensive Applications", *IEEE Transaction on Computers*, vol. 49, no. 5, May 2000, pp. 465-481.

[9] H. Zhang, V. Prabhu, V. George, M. Wan, M. Benes, A. Abnous, "A 1V Heterogeneous Reconfigurable Processor IC for Baseband Wireless Applications", *IEEE Journal of Solid-State Circuits*, vol.35, no. 11, Nov 2000, pp. 1697-1704.

[10] M. B. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffmann, P. Johnson, J. W. Lee, W. Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, A. Agarwal, "The Raw microprocessor: A computational fabric for software circuits and general-purpose programs", *IEEE Micro*, vol. 22, no. 2, pp. 25–35, Mar.-Apr. 2002.

[11] S. Bell, B. Edwards, J. Amann, R. Conlin, K. Joyce, V. Leung, J. MacKay, M. Reif, L. Bao, J. Brown, M. Mattina, C.C. Miao, C. Ramey, D. Wentzlaff, W. Anderson, E. Berger, N. Fairbanks, D. Khan, F. Montenegro, J. Stickney, J. Zook, "TILE64 processor: A 64-core SoC with mesh interconnect", *IEEE International Solid-State Circuits Conference (ISSCC'08)*, Feb. 2008, pp. 88-89.

[12] D.N. Truong, W.H. Cheng, T. Mohsenin, Yu Zhiyi, A.T. Jacobson, G. Landge, M.J. Meeuwsen, C. Watnik, A.T. Tran, X. Zhibin, E.W. Work, J.W. Webb, P.V. Mejia, B.M. Baas, "A 167-Processor Computational Platform in 65 nm CMOS", *IEEE Journal of Solid-State Circuits*, vol. 44, no. 4, April 2009, pp. 1130-1144.

[13] D. Pham, S. Asano, M. Bolliger, M.N. Day, H.P. Hofstee, C. Johns, J. Kahle, A. Kameyama, J. Keaty, Y. Masubuchi, M. Riley, D. Shippy, D. Stasiak, M. Suzuoki, M. Wang, J. Warnock, S. Weitzel, D. Wendel, T. Yamazaki, K. Yazawa, "The Design and Implementation of a First-Generation CELL Processor", *IEEE Journal of Solid-State Circuits*, vol. 41, no. 1, January 2006, pp. 179-196.

[14] E. Lindholm, J. Nickolls, S. Oberman, J. Montrym, "NVIDIA Tesla: A Unified Graphics and Computing Architecture", *IEEE Micro*, vol. 28, no. 2, Mar./Apr. 2008, pp. 39-55.

[15] "Nomadik®: AMobile Multimedia Application Processor Platform", *Asia and South Pacific Design Automation Conference (ASP-DAC 2007)*, Jan. 2007, pp. 749-750.

[16] S. Dutta, R. Jensen, A. Rieckmann, "Viper: A Multiprocessor SOC for Advanced Set-Top Box and Digital TV Systems", *IEEE Design & Test of Computers*, vol. 16, no. 5, Sept.-Oct. 2001, pp. 21-31.

[17] D.Rossi, F.Campi, A.Deledda, S.Spolzino, S.Pucillo, "A Multi-Core Signal Processor for Heterogeneous Reconfigurable Computing", *IEEE Custom Integrated Circuits Conference (CICC'09)*, Sept. 2009, pp 641-644.

[18] A. Deledda, C. Mucci , A. Vitkovski , P. Bonnot, A. Grasset, P. Millet, M. Kuehnle, F. Ries, M. Huebner, J. Becker, M. Coppola, L. Pieralisi, R. Locatelli, G. Maruccia., F. Campi, T. DeMarco, "Design of a HW/SW Communication Infrastructure for a heterogeneous reconfigurable processor", *IEEE International Conference on Design, Automation, and Test in Europe (DATE'08)*, 2008, pp. 1352-1357.

[19] D. Rossi, F. Campi, A. Deledda, C. Mucci, S. Pucillo, S. Whitty, R. Ernst, S. Chevobbe, S. Guyetant, M. Kühnle, M. Hübner, J. Becker, W. Putzke-Roeming "A Multi-Core Signal Processor for Heterogeneous Reconfigurable Computing", *IEEE International Symposium on System-on-Chip (SoC'09)*, Oct. 2009, pp. 106-109.

[20] P. Millet, "Overall MORPHEUS Toolset Flow", *Dynamic System Reconfiguration in Heterogeneous Platforms*, Springer, Jun. 2009, pp. 109-117.

[21] M. Vorbach, J. Becker, "Reconfigurable Processor Architectures for Mobile Phones*", Proceedings of the IEEE Parallel and Distributed Processing Symposium*, April 2003, pp. 6.

[22] A. Rivaton, J. Quevremont, Q. Zhang, P. T. Wolkotte, G. J.M. Smith, "Implementing non power-of-two FFTs on coarse grain reconfigurable architectures", *Proceedings of the International Symposium on System-On-Chip (SoC'05)*, 2005, pp.74-77.

[23] V. Baumgarte, G. Ehlers, F. May, A. Nückel, M. Vorbach and M. Weinhardt, "PACT XPP a Self-Reconfigurable Data Processing Architecture", *The Journal Of Supercomputing*, vol. 26, no. 2, Sept. 2003, pp. 167-184.

[24] F. Campi, A. Deledda, M. Pizzotti, L. Ciccarelli, C. Mucci, A. Lodi, L. Vanzolini, A. Vitkovski, "A dynamically adaptive DSP for heterogeneous reconfigurable platforms", *IEEE International Conference on Design Automation and Test in Europe (DATE'07)*, Apr. 2007, pp. 1-6.

[25] A. Lodi, A. Cappelli, M. Bocchi, C. Mucci, M. Innocenti, C. D. Bartolomeis, L. Ciccarelli, R. Giansante, A. Deledda, F. Campi, M. Toma, and R. Guerrieri, "XiSystem: a XiRisc-based SoC with reconfigurable IO module", *IEEE Journal of Solid-State Circuits*, vol. 41, no. 1, pp. 85–96, Jan. 2006.

[26] M.Coppola, R. Locatelli, G. Maruccia, L. Pieralisi, A. Scandurra, "Spidergron: a novel on-chip communication network", *Proceedings of the IEEE International Symposium on System on Chip*, Nov. 2004, pp. 15-16.

[27] C. Mucci, L. Vanzolini, I. Mirimin, D. Gazzola, A. Deledda, S. Goller, J. Knaeblein, A. Schneider, L. Ciccarelli, F. Campi, "Implementation of Parallel LFSR-based Applications on an Adaptive DSP featuring a Pipelined Configurable Gate Array", *IEEE International Conference on Design Automation and Test in Europe (DATE'08)*, Mar. 2008, pp. 1444-1449.

[28] C. Mucci, L. Vanzolini, A. Lodi, A. Deledda, R. Guerrieri, F. Campi, M. Toma, "Implementation of AES/Rijndael on a dynamically reconfigurable architecture", *IEEE International Conference on Design, Automation and Test in Europe (DATE'07)*, Apr. 2007, pp. 1–6.

[29] Abound Logc Embedded FPGA, http://www.aboundlogic.com.

[30] G. Pulini, D. Hulance, "Flexeos Embedded FPGA Solution", *Dynamic System Reconfiguration in Heterogeneous Platforms*, Springer, Jun. 2009, pp. 217-224.

[31] E. Schüler , M. Weinhardt, "The XPP-III Reconfigurable Processor Core", *Dynamic System Reconfiguration in Heterogeneous Platforms*, Springer, Jun. 2009, pp. 63-76.

[32] A. Lodi, C. Mucci, M. Bocchi, A. Cappelli, M. De Dominicis, L. Ciccarelli, "A Multi-Context Pipelined Array for Embedded Systems", *IEEE International Conference on Field Programmable Logic and Applications (FPL 2006)*, Aug. 2006, pp. 1-8.

[33] C. Mucci, D. Rossi, F. Campi, L. Ciccarelli, M. Pizzotti, L. Perugini, L. Vanzolini, T. De Marco, M. Innocenti, "The Dream Digital Signal Processor", *Dynamic System Reconfiguration in Heterogeneous Platforms*, Springer, Jun. 2009, pp. 49-61.

[34] S. Perissakis, F. Ieromnimon, N. S. Voros, "PHY-Layer of 802.16 Mobile Wireless on a Hardware Accelerated SoC", *Dynamic System Reconfiguration in Heterogeneous Platforms*, Springer, Jun. 2009, pp. 217-224.

[35] F. Campi, R. König, M. Dreschmann, M. Neukirchner, D. Picard, M. Jüttner, E. Schüler, A. Deledda, D. Rossi, A.Pasini, M. Hübner J. Becker, R. Guerrieri, "RTL-to-Layout Implementation of an Embedded Coarse Grained Architecture for Dynamically Reconfigurable Computing in Systems-on-Chip", *IEEE International Symposium on System-on-Chip (SoC'09)*, Oct. 2009.

[36] L. Ciccarelli, D. Loparco, M. Innocenti, A. Lodi, C. Mucci, P. Rolandi, "A Low-Power Routing Architecture Optimized for Deep Sub-Micron FPGAs", *IEEE Custom Integrated Circuit Conference (CICC'06)*, Sep. 2006, pp. 309-312.

[37] C. Cheng, C. Lin, C. Li, L. Chen, "iVisual: An Intelligent Visual Sensor SoC With 2790 fps CMOS Image Sensor and 205 GOPS/W Vision Processor", *IEEE Journal of Solid-State Circuits*, vol. 44, n .1, Jan. 2009, pp. 127-135.

[38] A. Abbo, R. Kleihorst, V. Choudhary, L. Sevat, P. Wielage, S. Mouy, M. 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, Jan. 2008, pp. 192-201.

[39] V. K. Prasanna, "Energy-Efficient Computations on FPGAs", *The Journal of Supercomputing*, vol. 32, no. 2, May 2005, pp. 139–162.

[40] S. Whitty, H. Sahlbach, R. Ernst ,W. Putzke-Roming , "Mapping of a film grain removal algorithm to a heterogeneous reconfigurable architecture", *IEEE International Conference on Design, Automation and Test in Europe (DATE'09)*, Apr. 2009, pp. 27-32.

[41] E. Markert, S. Goller, A. Schneider, J. Knablein, U. Heinkel, "Ethernet Based In-Service Reconfiguration of SoCs in Telecommunication Networks", *Dynamic System Reconfiguration in Heterogeneous Platforms*, Springer, Jun. 2009, pp. 195-203.

[42] C. Mucci, L. Vanzolini, F. Campi, G. Gaillat, A. Deledda, "Intelligent cameras and embedded reconfigurable computing: a case-study on motion detection", *IEEE International Symposium on System on Chip (SoC'07)*, Oct. 2007, pp.1-4.

TABLE I
NOC IMPLEMENTATION DETAILS

Entity	**# of Instances**	**Std Cells count [Kgates]**
Routers	16	292
NoC Initiator NIs	7	202
NoC Target NIs	7	113
AHB to NoC bridges	6	265
NoC to AHB bridges	6	260
DMAs	6	434
DNA	1	65
Other (Bus, mpmc…)		571
Total		2202

TABLE II
XPP-III IMPLEMENTATION DETAILS

Entity	**Std Cell Count [KGates]**	**Macro Area [mm^2]**
ALU PAE	85	0,36
RAM PAE	105	0,6
ARRAY	4130	23
FNC-PAE	283	2,9
Embedded memory	n.a.	5,2
Other (4-D DMA, fifo controller…)	662	n.a.
Total	5358	42,5

TABLE III
DREAM IMPLEMENTATION DETAILS

Entity	**Standard Cell Count [KGates]**	**Area [mm^2]**
PiCoGA macro area	n.a.	9.8
Embedded memory area	n.a.	3.6
µP, address generators, interconnect matrix…	306	n.a.
Total	n.a.	22,5

TABLE IV
EFPGA IMPLEMENTATION DETAILS

Entity	**Standard Cell Count [KGates]**	**Area [mm^2]**
Flexeos macro area	n.a.	2,88
Embedded memory area	n.a.	0,88
Loader, fifo controller	43	n.a.
Total	n.a.	5

TABLE V
CHIP CHARACTERISTICS

Process Technology	90 nm CMOS90GP Process, 7-metal layers
Power Supply	1,0V for core, 3,3 for I/0
Area	110 mm^2
Transistor Count	44M Logic 1,1Mbyte SRAM
Pinout	256, 163 I/O
Operating Frequency	ARM, BUS, NoC: 250 MHz XPP : 0 - 160 MHz DREAM : 0 - 200 MHz eFPGA : 0-140 MHz
Power Consumption	Static Power : 235 mW ARM + NoC : 600 mW @ full speed XPP : 1200 mW @ full occupation - full speed DREAM : 420 mW @ full occupation - full speed eFPGA : 112 mW @ full occupation - full speed

TABLE VI
COMPARISON VS. SOA

Device	Technology	GOPS	GOPS/W	GOPS/mm2
Morpheus	CMOS090	50	20	0,6
ARM9	CMOS090	0,35	0,15	0,2
ST ASIC	CMOS090	n.a.	1000	10
iVisual [1]	CMOS180	77	50	1.2
Xetal II [38]	CMOS090	107	170	1,4
Virtex II [39]	CMOS130	n.a.	2	0,1
AsAP[12]	CMOS090	180	22,5	4,5
CELL [13]	CMOS090	200	0,9	0,5
GPU [14]	CMOS090	576	3,84	1,2

TABLE VII
MOTION DETECTION VIDEO SURVEILLANCE
APPLICATION PROFILING AND PARTITIONING

Kernel	ARM	Computation	Mapping
Sub/Abs/Max	3%	8-bit Arith.	XPP-III
Binarization	2%	Asymm. bit level	eFPGA
Opening	39%	Symm. bit. level	DREAM
Edge Detection	55%	Symm. bit level	DREAM
Final merge	1%	8-bit Arith.	ARM

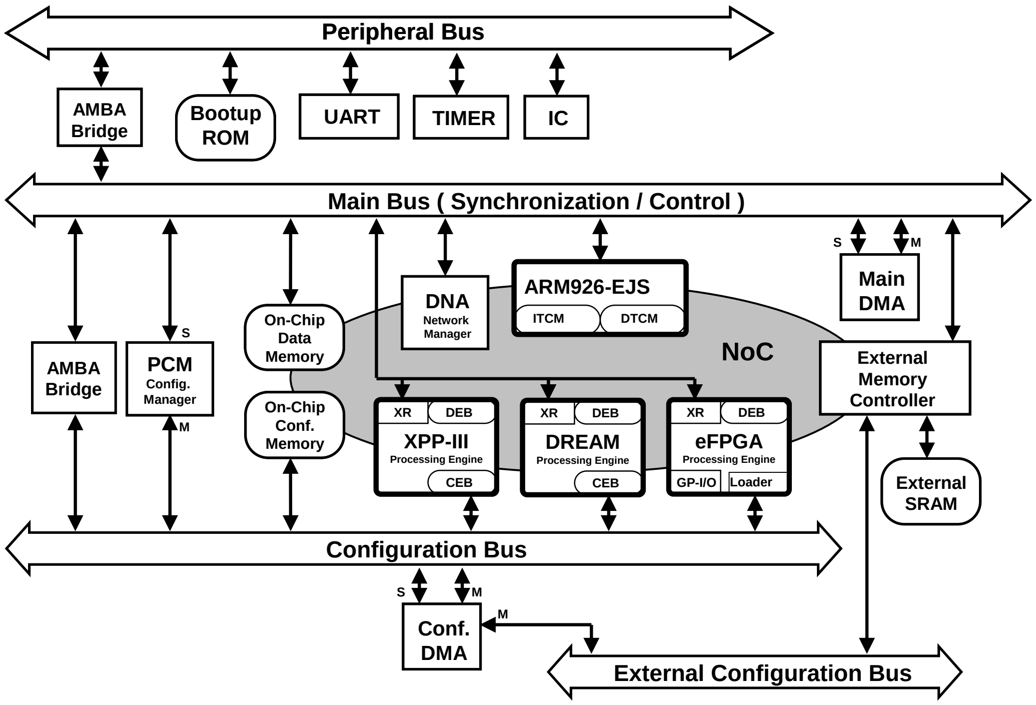

Fig. 1. SoC architecture

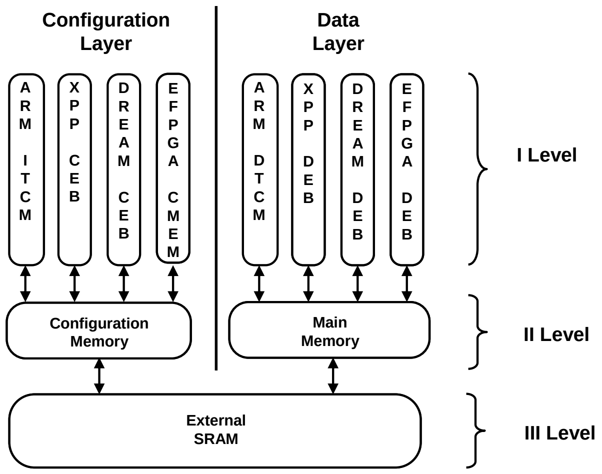

Fig. 2. SoC Memory Hierarchy

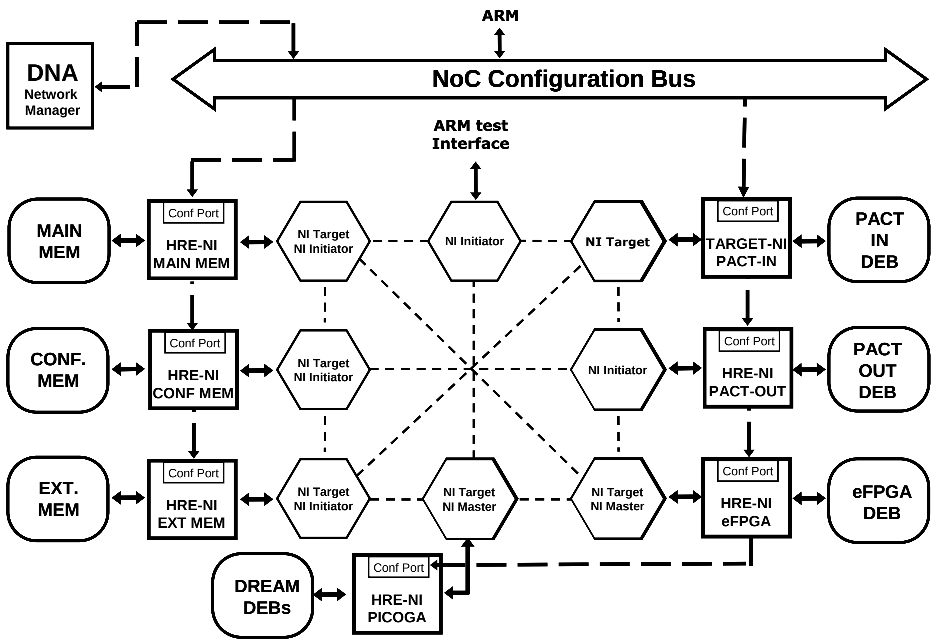

Fig. 3. Morpheus Communication infrastructure Architecture.

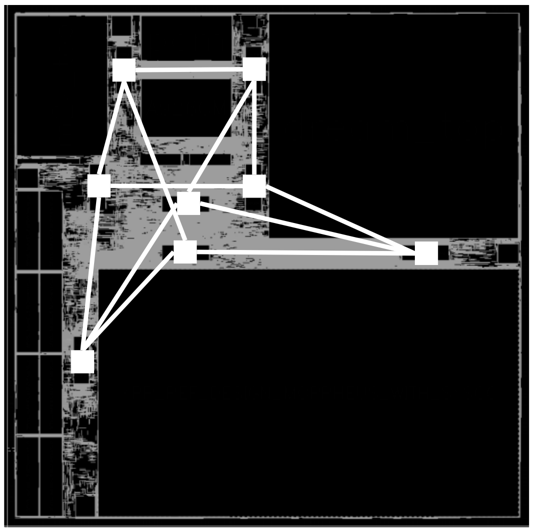

Fig. 4. Morpheus NoC Topology

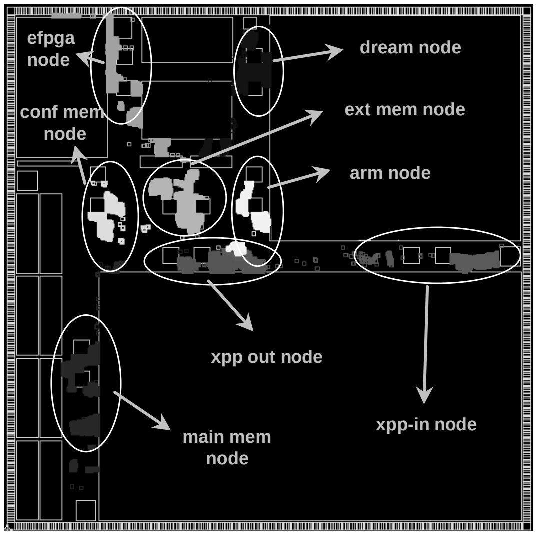

Fig. 5. STNoC floorplan. Each node is composed by one request router, one response router (hard macros) plus the network interface (std. cells).

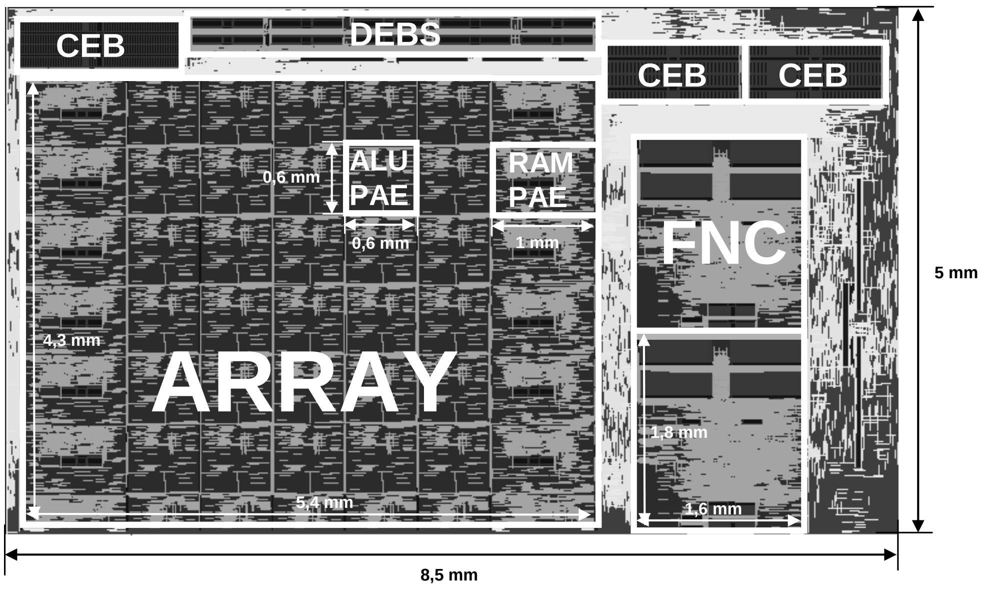

Fig. 6. XPP-III Layout

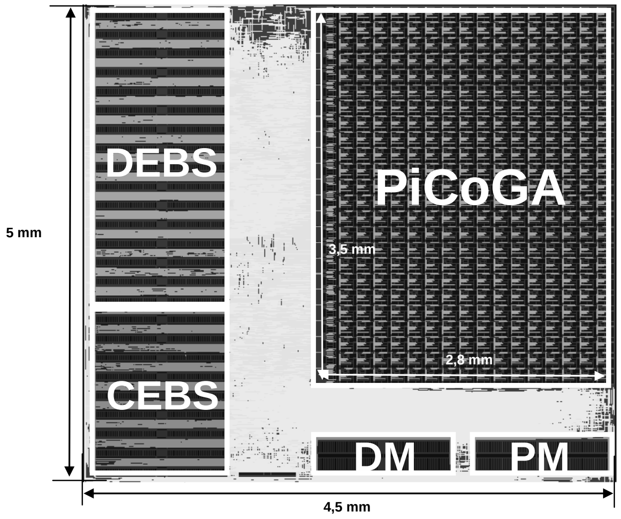

Fig. 7. DREAM Layout

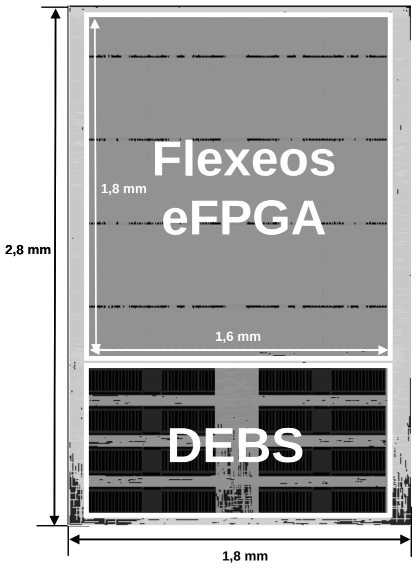

Fig. 8. eFPGA Layout

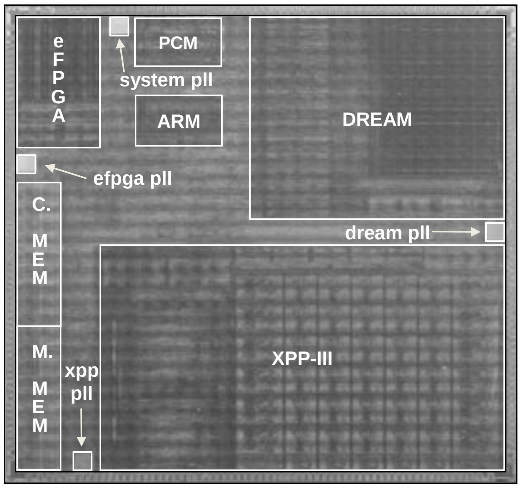

Fig. 9. Morpheus chip photograph

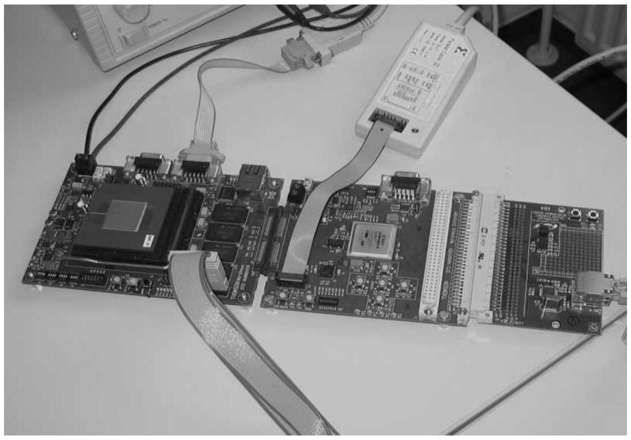
Fig. 10. Morpheus testing environment

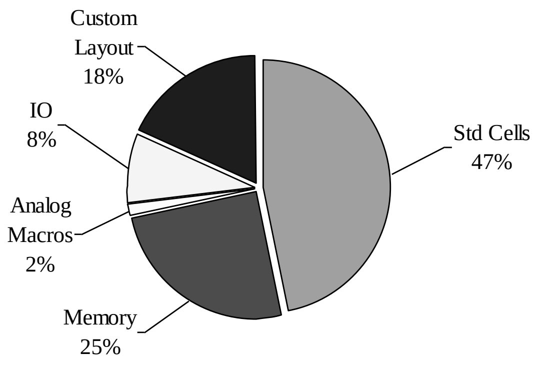

Fig. 11. Morpheus Area by design object

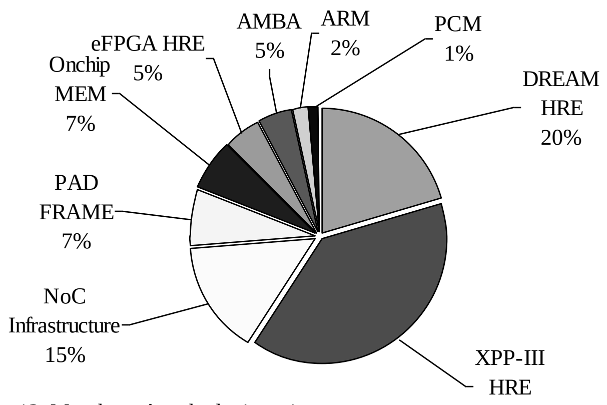

Fig. 12. Morpheus Area by logic entity

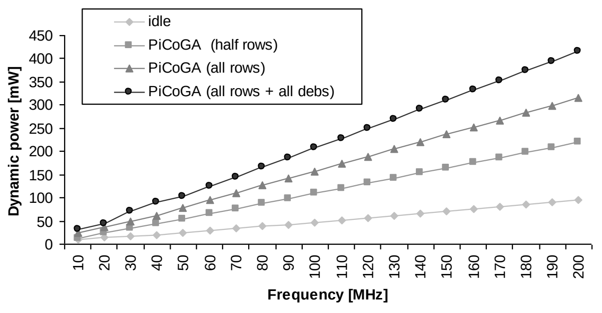

Fig. 13. DREAM dynamic power consumption when varying resources utilization

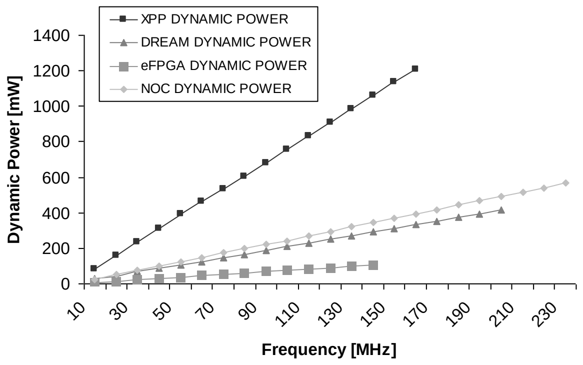

Fig. 14. Overall contributions to Morpheus dynamic power

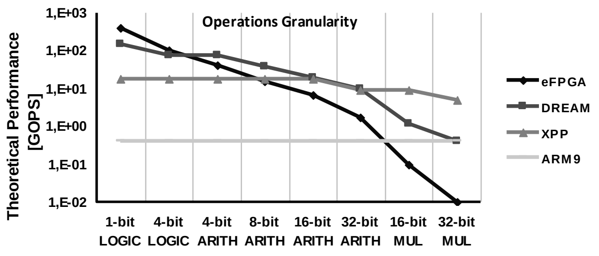

Fig. 15 Morpheus theoretical performance

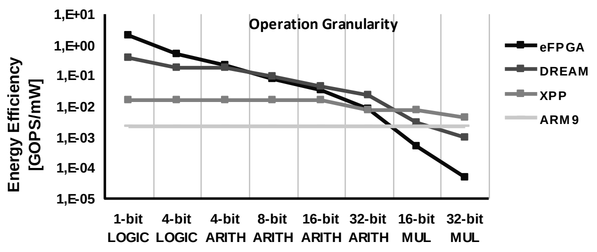

Fig. 16. Morpheus energy efficiency

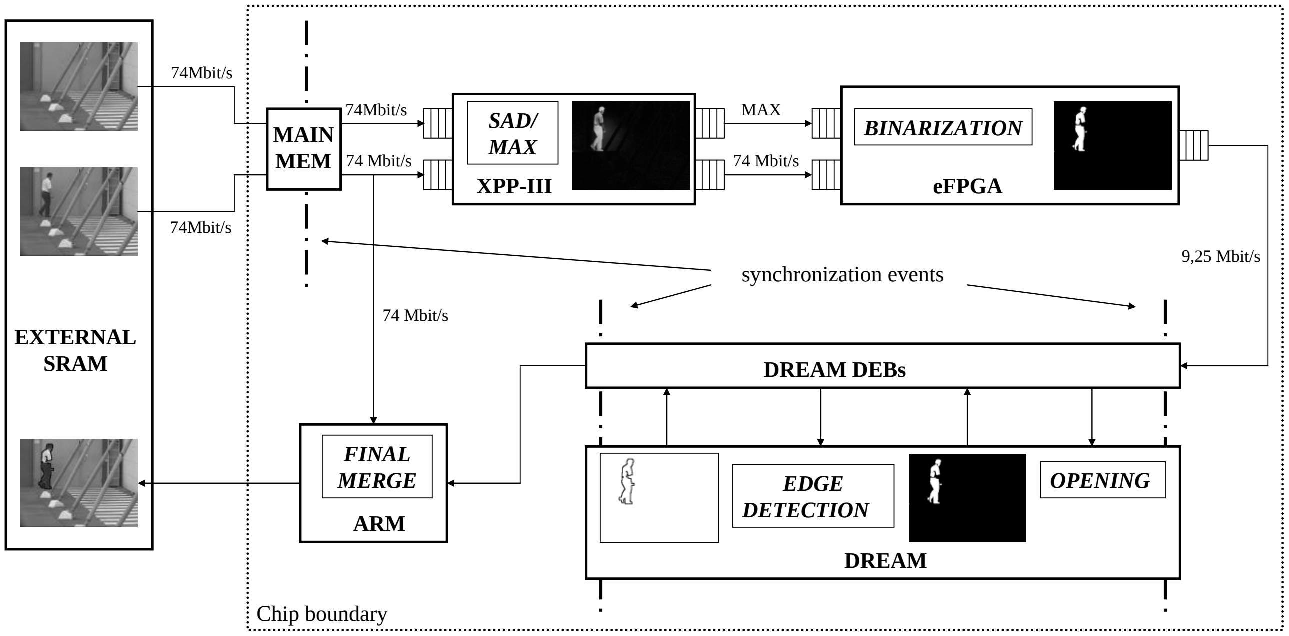

Fig. 17. Implementation of a motion detection video surveillance application on the Morpheus platform.