J Real-Time Image Proc (2013) 8:111–125
DOI 10.1007/s11554-011-0215-8

SPECIAL ISSUE

A multi-processor NoC-based architecture for real-time image/video enhancement

Sergio Saponara · Luca Fanucci · Esa Petri

Received: 18 November 2010 / Accepted: 2 July 2011 / Published online: 21 July 2011
© Springer-Verlag 2011

Abstract The paper presents a multi-processor architecture for real-time and low-power image and video enhancement applications. Differently from other state-of-the-art parallel architectures the proposed solution is composed of heterogeneous tiles. The tiles have computational and memory capabilities, support different algorithmic classes and are connected by a novel Network-on-Chip (NoC) infrastructure. The proposed packet-switched data transfer scheme avoids communication bottlenecks when more tiles are working concurrently. The functional performances of the NoC-based multi-processor architecture are assessed by presenting the achieved results when the platform is programmed to support different enhancement algorithms for still images or videos. The implementation complexity of the NoC-based multi-tile platform, integrated in 65 nm CMOS technology, is reported and discussed.

Keywords Image enhancement · Real-time image/video processing · Network-on-Chip (NoC) · Digital IP cells · Multi-Processor System on Chip (MPSoC)

1 Introduction

In both professional and high-range consumer products there is a growing interest in image and video enhancement techniques to improve system performance in terms of visualization quality, coding efficiency, error resiliency, and capability of video analysis, understanding and pattern recognition. The application fields include medical imaging, intelligent transport systems and vehicles, video telecommunication and multimedia entertainment, vision-systems in robotics or industrial automation, surveillance and remote sensing.

From the algorithmic point of view, several techniques have been proposed in the literature, such as order statistics and stack filters, median and Retinex-like filters, polynomial and rational operators, morphological algorithms, fuzzy-based and chaotic processing, to name just a few [1–8]. Spatial and spatio-temporal operators, with or without motion compensation, have been proposed. The above techniques represent a mature solution for noise and artifact suppression, for high-dynamic-range management, for spatial and temporal video interpolation, and for image resolution enhancement.

However the implementation of such techniques in real-time and with low power consumption in System-on-Chip (SoC) is a challenging open issue. In most cases the software implementation of the proposed algorithms on a microprocessor is not in real time. Even in the case that real-time processing is obtained by using Graphic-specific Processing Units (GPUs) [9, 10] the relevant power consumption is unsuitable for embedded or mobile systems. Indeed a GPU has a power consumption, depending on the workload [11], ranging from tens to hundreds of Watts. Dedicated integrated circuits (ICs), with power consumption of hundreds mW, have been proposed in [12–16] but they are devoted to a specific task, e.g. dynamic range compression for display of mobile devices in [12]. On the contrary, a programmable solution covering multiple tasks is needed. The availability of low-power and integrated solutions, offering enough programmability to support

S. Saponara (✉) · L. Fanucci
Department of Information Engineering, University of Pisa,
via G. Caruso 16, 56122 Pisa, Italy
e-mail: sergio.saponara@iet.unipi

E. Petri
Consorzio Pisa Ricerche scarl,
C.so Italia 116, 56125 Pisa, Italy

different classes of image/enhancement algorithms, is a strategic target to be achieved to foster the adoption of these techniques in new application fields. The problem is not only the design of the computing architecture but also of the communication infrastructure due to the nature of image enhancement algorithms, often employing irregular data flow and irregular operation scheduling.

To address the above issues this work presents a programmable and scalable MPSoC architecture for image and video enhancement exploiting two main paradigms:

1) the use of an array of multiple heterogeneous tiles;
2) the use of a Network-on-Chip (NoC) as communication infrastructure to overcome the bottleneck of classic circuit-switched bus architectures.

The paper is structured as follows. Section 2 reviews state-of-the-art computing architectures for real-time image/video enhancement. Section 3 describes the proposed NoC-based MPSoC for image/video enhancement, discussing the design of the computing tiles. Section 4 deals with the NoC infrastructure. Section 5 presents some application case studies assessing the functional performances that can be achieved with the proposed MPSoC. Implementation results in 65 nm CMOS technology are discussed in Sect. 6. Conclusions are drawn in Sect. 7.

2 Image/video enhancement architectures

The computing architectures proposed in literature to achieve real-time processing for image/video enhancement tasks can be classified in three main groups, each group having some advantages and some limits still to overcome:

1) Software implementation on general purpose CPUs with clock frequencies in the GHz domain and with multi-cores, e.g. Core 2 in [17], or a single-core, e.g. Atom in [15], depending on the computational load. The power consumption of such solutions can be up to tens of Watts.
2) Software implementation on massively parallel GPUs [9, 10] with computational throughput of billions of floating point operations per second (GFLOPS) but power cost from tens to hundreds of Watts.
3) Hardware design of programmable platforms achieving real time for handheld or mobile devices at power costs lower than 1 Watt, but limited to a specific algorithmic class [12–16, 18–20].

As example of the first group, pure software implementations of 3D DCT tasks, used in literature for video coding or feature extraction applications, have been proposed in [17]. A software optimized design of 3D DCT and IDCT is implemented in real time at 24 frames/s VGA format (640 × 480 pixels) on a general purpose multi-core processor, the Intel Core 2 6300@1.86 GHz. The real time processing of PAL formats (720 × 576 pixels) at 24 frames/s can be achieved by implementing on the Core 2 6300 device only the 3D forward DCT. To be noted that the Core 2 6300 processor, realized in 65 nm CMOS technology, integrates two 64-bit CPU cores and up to 4 MB of L2 cache memory. It has a die size of roughly 143 mm^2 and a transistor count of roughly 300 millions; at 1.86 GHz the power consumption is up to 65 Watts. Therefore, the power consumption of software solutions on general purpose multi-core CPUs, in the order of tens of Watts, is unsuited for battery powered, mobile or handheld terminals.

A lower power consumption can be achieved by targeting single-core CPUs such as the Intel ATOM @ 1.6 GHz used in [15] to implement via software a feature extractor for mobile applications. Although the Atom CPU is more optimized in terms of power efficiency than the Core 2 architecture, its power cost, several Watts, is still high versus the requirements of handheld devices. Moreover, the achieved frame-rate performance in [15] with the Atom solution is less than 6 images/s, not enough even for medium-quality real-time video representations.

Real-time performance can be surely achieved with GPUs, such as the recent Fermi architecture from NVIDIA [10]: GPUs are now evolving as massively parallel platforms for graphic but also for computational-intensive general purpose algorithms with high degree of parallelism. As example, the Fermi architecture is composed of 512 CUDA (Compute Unified Device Architecture) processing cores hierarchically organized in 16 streaming multi-processors each with 64 kB L1 cache and 128 kB local register file and sharing a common 768 kB L2 cache. The total amount of on-chip memory is roughly 4 MB. The Fermi GPU architecture has also a PCIe host interface and 6 64-bit DDR (double data rate) DRAM interfaces. Each CUDA core is capable of both integer and floating point operations with 64-bit results. Each of the 16 streaming processors has also 4 40-bit special function units for fast approximation of non linear operators (square root, sin, cos, exp, log functions) and 16 load/store units. The Fermi NVIDIA GPU leads to high computational power, up to 1,500 GFLOPS in single-precision, orders of magnitude higher than Core 2 or Atom general-purpose processors. Although real-time processing is not an issue for such GPUs, their area and power consumption is suited only for desktop applications and workstations, not for handheld or mobile devices. The Fermi NVIDIA core has an area occupation higher than 300 mm^2 and the power consumption, depending on the computational workload, can be up to several hundreds of Watts.

The considerations done for the Fermi NVIDIA GPU can be applied to other parallel architectures such as the Cell Broadband Engine used for real-time video processing, e.g. as [9] where the Cell realizes a real-time MPEG2 encoder on 128 × 128 frames. The cell includes nine processors: one 64-bit Power PC core in charge of operation scheduling and data flow control and eight Synergistic Processors (SP) dedicated to DSP computing and working according to an SIMD (Single Instruction Multiple Data) scheme. As far as the memory resources are concerned, the cell uses 256 kB of local memory for each SP and 512 kB L2 shared cache. The cell has a capability of more than 200 GFLOPS, with a size higher than 200 mm^2 and a power cost of tens of Watts.

To overcome the power consumption issues of the above computing classes (general purpose CPUs or high-parallel DSPs and GPUs) in academia [13–16, 18–20] and industry [12] several programmable hardware designs have been proposed. Such designs have a power consumption below 1 Watt but achieve real-time processing only for a specific enhancement task and for low/mid-size image and video formats. Most of the proposed solutions use a RISC-like processing core with local instruction/data cache, for operation and data control flow, enhanced by external DRAM controller and hardware accelerators for the most demanding tasks. However, these works have some limits still to overcome:

1) The communication between the RISC core and the coprocessors is based on classic bus architectures, e.g. AHB bus. This limits the scalability of the solution and represents a bottleneck in case of algorithms with irregular data flow, such those used in image enhancement applications.
2) Most of the proposed solutions are customized for a specific algorithmic class: e.g. dynamic image compression in [12], feature extraction in [15], 3D rendering in [16]. Hence, if a mobile device needs a number of these functions several ICs should be used and mounted on a PCB board.

In the following Sections we propose a single-chip architecture to achieve real-time processing for different algorithmic classes (e.g. video motion estimation, image transform, dynamic compression, contrast enhancement, de-blocking/de-noising filtering) at reasonable power cost, within 1 Watt, and for low/mid-size formats typical of mobile devices: up to 30 frames/s VGA quality (640 × 480 pixels) [21]. To this aim the following ideas will be exploited:

– NoC as on-chip communication infrastructure, easing architecture scalability and management of computing tasks with irregular data flow due to the packet-switch communication scheme.
– Use of multiple heterogeneous programmable processors (instead of homogeneous ones as in GPUs and in general-purpose multi-core CPUs) specialized for different algorithmic classes.

3 NoC-based multi-processor SoC for image/video enhancement

3.1 MPSoC architecture

The proposed architecture is sketched in Fig. 1. It is composed of 16 heterogeneous tiles (9 computing/control tiles with local memories plus 7 shared frame memories) connected to each other through a NoC, with Spidergon topology.

The NoC uses 8 5-port Routers (R in Fig. 1): 3 router ports are dedicated to *across*, *right* and *left* router-to-router connections and two ports are dedicated to the connections between the NoC and the computing or memory tiles. Conversion of protocol, data size and clock frequency between the NoC and each tile is managed by 16 Network Interfaces (NIs in Fig. 1).

The computing tiles are in details as follows:

– 1 tile labeled '*CPU*': a 32-bit SPARC V8 core which is in charge of instruction and data flow management. This tile has a 5-stage integer pipeline plus hardware support of multiply and accumulate (MAC) operations and local instruction/data cache memory. The *CPU* tile

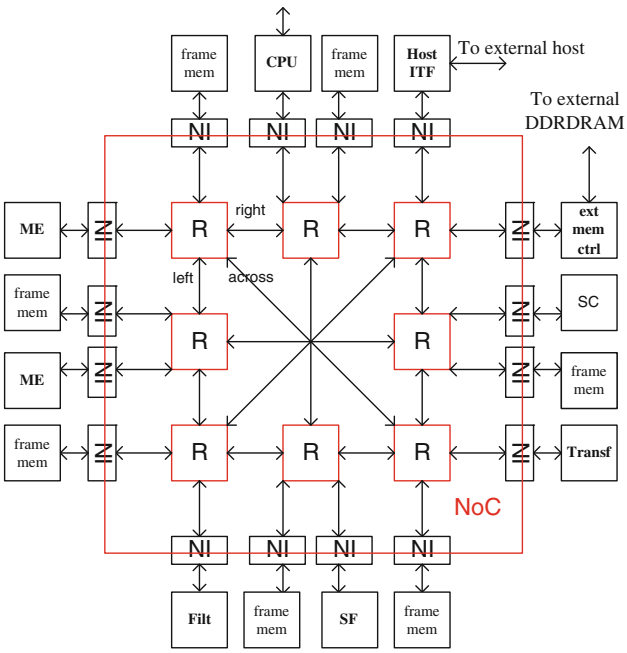

Fig. 1 Block diagram of the NoC-based MPSoC architecture

has also low-speed local peripherals (timers, interrupt controller, UART and GPIO and JTAG interfaces). It is realized as a modified and extended version of a LEON2 IP core and has been enhanced with an IEEE-754 compliant floating point coprocessor. The computational power amounts to roughly 0.85 Dhrystone MIPS/MHz, i.e., 340 MIPS when the tile is clocked at 400 MHz (see CMOS implementation results in Sect. 6). The cache size is kept as a parameter of the HDL description and is configurable at logic synthesis time: 8 kB of instruction cache and 8 kB of data cache are used in the implemented platform. The complexity of the *CPU* tile amounts to 70 k logic gates.

- 2 tiles labeled '*ME*', dedicated to motion estimation processing, working on 2D image blocks, which are detailed in Sect. 3.2;
- 1 tile labeled '*Transf*', dedicated to transform functions, such as Discrete Cosine-, Sine- or Fourier-Transforms and their inverse (*DCT/IDCT*, *DST/IDST*, *FFT/IFFT*) based on radix-2 butterfly units, detailed in Sect. 3.3.
- 1 tile labeled '*FILT*', dedicated to image/video filtering; it supports linear (e.g. FIR) or non-linear (e.g. rational filters) operators working at image block level. The *FILT* tile is detailed in Sect. 3.4.
- 1 tile labeled '*SF*', dedicated to special functions widely used in image processing: pixel transformations such as gamma correction and contrast enhancement [18], color domain conversion (supporting RGB, YUV, YCrCb), frame size conversion, log-linear and linear-log domain conversion, clipping.
- 1 tile labeled '*SC*', in charge of source coding techniques such as variable-length coding or context-adaptive binary arithmetic coding. It has a complexity of 30 k logic gates plus 30 kbits of local memory. The core of this tile is the coding engine we proposed in [19].
- 2 tiles labeled '*Host ITF*' and '*Ext Mem Ctrl*', acting as high-speed external host interfaces and external memory controller (DDR-DRAM or ROM/EEPROM), respectively; both tiles have DMA (Direct frame Memory Access) functionalities and local buffers, 1 Mbits each.

All computing tiles and frame memories have a 32-bit AHB interface towards the corresponding NI. The size of each frame memory (*frame mem* in Fig. 1) is a parameter of the HDL description configurable at synthesis time. In this work a size of 2.35 Mbits, enough to store a VGA (640 × 480) frame is considered for each of the 7 *frame mem* blocks for a total amount of 16.45 Mbits on-chip SRAM.

The architecture in Fig. 1 offers a balanced mix between general-purpose computing capabilities of the *CPU* tile, high-throughput DSP capabilities offered by the other heterogeneous tiles (2 *ME*, the *SF*, the *SC*, the *Transf* tiles) and large on-chip memory resources since image/video applications are data-dominated.

Memory resources are hierarchically organized in a first level of local memories in the computing tiles, plus a second level of shared frame memories (SRAM) accessible via NoC and a third level of off chip DRAMs accessible via the high-speed '*Host ITF*' or the '*Ext Mem Ctrl*' tiles.

Due to the NoC paradigm, the MPSoC architecture can be easily scaled by modifying the configuration discussed in this work. The architecture can be scaled in terms of number of used computing tiles, number and size of the frame memories, size of the tile bus data and of the NoC links. This way the desired trade-off between complexity and performance can be easily set.

Hereafter, we present the architectural block diagram of the main tiles dedicated to signal processing functionalities. By exploiting a design reuse IP approach these tiles have been implemented by modifying the RTL descriptions of IP macrocells we have proposed as stand-alone processors in past works [18, 19, 23–25]. The NoC design instead is a completely new design from scratch. This is why in the following discussion we briefly review the architectures of the tiles derived by previous IP designs (for further details we refer to our published literature) while a deeper analysis is reserved to the NoC in Sect. 4.

3.2 Motion estimation tiles

ME is one of the most diffused and most demanding tasks, in terms of computation and memory resources, in video processing for different applications [19, 22, 26]: video coding, feature extraction, frame-rate up/down conversion, video analysis to name just a few. For each image block in the current frame the best matching block in one or more previous frames is searched, using a full search (FS) or a fast algorithm, minimizing a cost function such as the SAD (sum of absolute difference). Due to the high computational cost required by ME, billions AD operations with a FS algorithm applied to a 30 frames/s VGA video with ±16-pixel search size, we adopted two tiles in our MPSoC architecture, see Fig. 1.

Each tile, as reported in Fig. 2, consists of a 2D hardware search engine plus local buffer memories, a 32-bit AHB interface towards the NoC and a ME controller to support fast search strategies. The search engine is a regular array of 256 processing elements (PE) each implementing at pixel-level AD operations plus an adder tree, a unit for minimum detection calculation and a unit used to compare SAD block matching results with programmable thresholds (in case of fast ME search with early stop criteria). The search engine also supports programmable

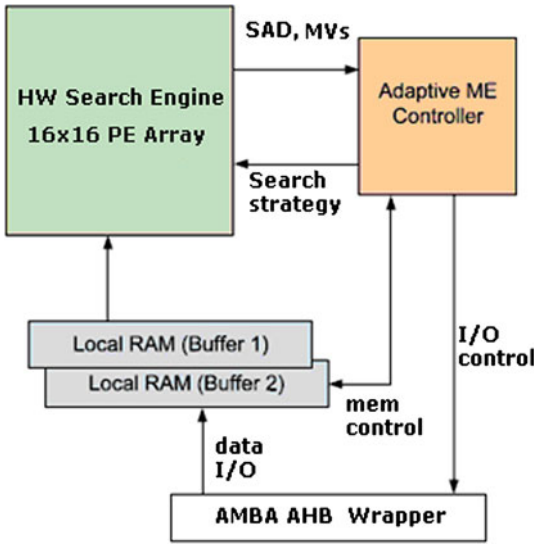

Fig. 2 Architectural block diagram of the ME tile

search size. The search engine provides SAD and Motion Vector (MV) field calculation as results; it is derived from a parametric 2D search engine we previously proposed in [23]. In this work the IP has been configured for a 256 PE array capable of 256 AD operations per clock cycle, i.e., more than 100 GOPS when the tile is clocked at 400 MHz. Hence, the use of both tiles allow for a maximum computational capability of 205 GOPS. The local memory resources for each ME tile have a size of 40 kbits, enough to (i) store a 16 × 16-pixel block and its search area with ±16 pixel displacement in horizontal and vertical directions and (ii) ensuring the prefetch of the next image block and its corresponding search area. The logic complexity of each ME tile amounts to about 115 k logic gates. The architecture proposed in Fig. 2 implements Full Search but also, having a programmable search area and supporting early termination criteria, fast ME algorithms such as the predictive ME in [19, 22, 27, 39]. In case of fast ME algorithms the context-aware control strategies are elaborated by the ME controller reported in Fig. 2 which, analyzing SAD and MV results, adapts the search strategy. The ME controller in Fig. 2 is realized using a simple programmable core, compliant with 8051 instruction set, and available as reusable IP cell from [35, 36].

3.3 Transform tile

The '*Transf*' tile supports frequency transform tasks, among the following 6 possible configurations: DCT, IDCT, DST, IDST, FFT, IFFT. These tasks are used in image and video coding applications for intra-frame data compression, for motion analysis in the frequency domain and for image segmentation. By exploiting the separability

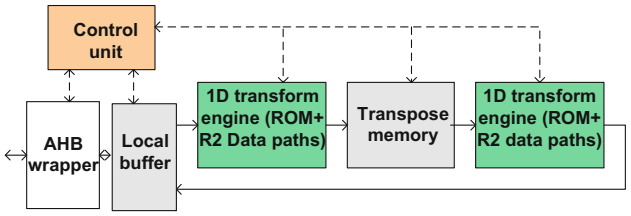

Fig. 3 Architectural block diagram of the *Transf* tile

technique this tile is composed of the cascade of two 1D transform engines, see Fig. 3, working on 8 × 8-pixel blocks plus a transposition buffer memory. The first engine works row-wise while the second engine works on the transposed results. Each engine adopts a decomposition of the 8 × 8-pixel input matrix in 4 radix-2 (R2) butterflies, implementing MAC operations between input pixels and twiddle coefficients. As far as the arithmetic accuracy is concerned, the tile supports MAC operations with a convergent-block floating point (CBFP) arithmetic. CBFP has been proved [24, 38] providing a better trade-off between accuracy and complexity versus fixed-point or floating-point arithmetic. The transform coefficients are uploaded from ROM memories: by selecting the desired transform a multiplexer circuitry connects the proper ROM to the data paths.

The computational throughput of the architecture in Fig. 3 amounts to 1 transform pixel per each clock cycle. Therefore a processing capability up to 400 M transform operations per second is possible by clocking the tile at 400 MHz. The circuit complexity is less than 50 k logic gates. This tile also adopts a local memory buffer of 32 kbits: 16 kbits to prefetch up to 16 8 × 8 blocks, and 16 kbits to store the results of 16 8 × 8 transform image blocks. 8 bit per pixel and 16 bit per sample in the transformed domain are considered. The '*Transf*' tile is connected to the relevant NI by a 32 bit AHB bus. A control unit, realized as a finite state machine, provides all relevant control signals.

The '*Transf*' tile has been described through a parametric VHDL model. Therefore, in case of professional video standards using higher bit depths (e.g., 10 bits in ITU-R BT.601 or 12 bits in Digital Cinema Initiatives) the MPSoC platform can be configured at synthesis time to work with more than 8 bits per pixel.

3.4 Filtering tile

The '*Filt*' tile is a digital macrocell whose global architecture is sketched in Fig. 4. This architecture is made up of the following building blocks:

(i) a programmable filtering core configurable to implement linear or rational non-linear filters;

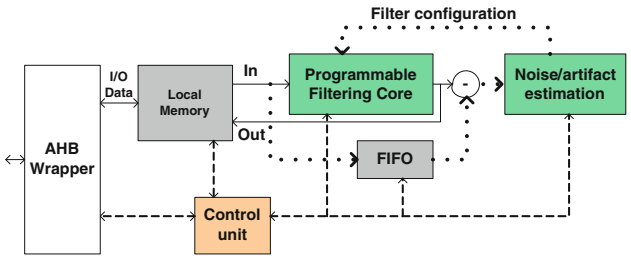

Fig. 4 Architectural block diagram of the *Filt* tile

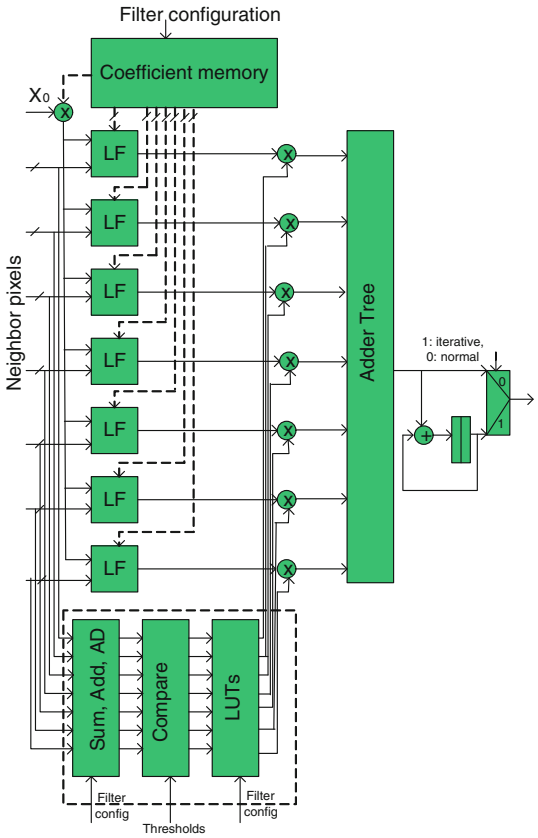

Fig. 5 Block diagram of the programmable filtering core

(ii) a unit for noise/artifact estimation and filter tuning to implement de-noising or de-blocking functionalities;
(iii) memory resources for data flow management;
(iv) a control unit, realized as a finite state machine, providing all relevant control signals.

The *programmable filtering core* block in Fig. 4 is implemented by the circuit sketched in Fig. 5. An array of programmable linear filters (LF in Fig. 5) elaborates the input samples of spatial and temporal masks centered on the pixel to be filtered X_0^t in the current frame t. Each LF is realized using a carry-save multiplier cascaded by an accumulator unit. To realize also rational non linear filters the outputs of the LF blocks are weighted by non-linear terms $1/\beta$. According to Eq. 1 the non-linear weights are produced as a function of spatial neighboring pixels in the current frame t (indexes i, j belonging to the spatial domain S) and/or temporal neighboring pixels in the frames t − 1 and t + 1 (indexes h, k belonging to the temporal domain T). As shown in Fig. 5 in hardware the evaluation of the non linear weights $1/\beta$ is based on look-up-tables (LUTs) receiving as input the results of the comparison between programmable thresholds and a combination (with sum, subtraction and absolute difference operations) of the incoming neighboring pixels. The results of all filtering directions are provided to the adder tree that produces the final output value. The adder tree unit is followed by a programmable output stage that allows the extension of both mask size and filtering directions. The circuit of Fig. 5 processes concurrently up to 7 filtering directions supporting both 2D spatial (4 directions as the algorithm proposed in [25]) and 3D spatio-temporal (as the algorithm proposed in [28]) rational algorithms.

$$Y_0^t = \sum_{i,j\in S} \frac{1}{\beta\left(X_i^t, X_j^t\right)} \cdot \left[a_i X_i^t + .. + a_0 X_0^t + .. + a_j X_j^t\right] + \sum_{h,k\in T} \frac{1}{\beta\left(X_h^{t-1}, X_k^{t-1}\right)} \cdot \left[a_h X_h^{t-1} + .. + a_0 X_0^t + .. + a_k X_k^{t+1}\right] \quad (1)$$

The extension of both mask size and processing directions can be obtained by the iterative use of the filtering unit. For instance, starting from the circuit in Fig. 5 the spatio-temporal rational algorithm with 13 filtering directions in [28] can be implemented using 2 processing iterations of a single *'Filt'* tile. The iterative use of the *'Filt'* tile can occur also when multiple passes of the filter on the same image is required to equalize the phase response in all directions or to achieve a high order filtering effect.

The unit for noise/artifact estimation and filter tuning in Fig. 4 receives as input the difference between the original image and the filtered one. The resulting difference image is an estimation of the noise/artifact affecting the original image to be filtered. Using this difference image the noise statistics are calculated and the type of noise is classified as Gaussian, contaminated-Gaussian, Impulsive or blocking artifact noise. As consequence the filter response is properly tuned in terms of processing coefficients. To reduce complexity, under the hypothesis of a uniform noise/artifact type affecting a whole image, the above described noise estimation phase is done on a 64 × 64 pixel sub-image, obtained by sub-sampling (drop of every second sample) a 128 × 128 area positioned on the center of each frame.

As discussed before, to avoid the use of power-consuming divider and square operators, the generation of the nonlinear weights is based on a LUT approach. In this

work, for each filtering direction 12 possible LUTs are defined, each optimized for a different noise distribution (Gaussian, contaminated-Gaussian or impulsive noise or blocking artifact) and for spatial, temporal or spatio-temporal processing.

The *'Filt'* tile has a throughput of 1 pixel/clock cycle, i.e., 400 Mpixels/s can be processed in real-time with a 400 MHz clock, and has a 32-bit AHB interface towards the NoC. The total local memory plus FIFO amount to 35 kbits while the circuit complexity is around 45 k logic gates. Pixels and coefficients are represented using 8 bits.

4 NoC communication infrastructure design

The need of real time performance in image and video enhancement applications, such as the case studies considered in Sect. 5, can be satisfied not only by reducing algorithms complexity, but also by exploiting parallel computation. Lots of image/video processing algorithms are suitable to follow a parallel operation flow, e.g. in Motion Estimation operations different searches (e.g.: searches for different macroblocks) can be done independently and thus parallelized.

A parallel hardware configuration can take advantage of innovative communication infrastructures that implement a scalable, distributed network in multi-core systems [29]: the so-called Network-on-Chip (NoC), which applies the layered-stack approach to the design of the on-chip inter-tile communications.

A clear overview of NoC capabilities for image/video processing is given in [30], which analyzes the effects of parallelization and communication backbone on an MPEG-2 encoder: when increasing the number of processing cores for parallelization, the NoC communication approach outperforms classic point-to-point and hierarchical bus interconnections in terms of area, power and throughput.

For our design we consider a Spidergon NoC topology [31–33], where each router is connected to its clockwise (Right) and its counter-clockwise (Left) neighbors as in a simple ring topology. In addition, each router is also connected directly to its diagonal counterpart in the network (Across), to minimize the number of nodes to cross before reaching the destination. As an example, a maximum of two hops are necessary to connect any two routers in the Spidergon network in Fig. 1 (i.e., a maximum of three nodes shall be crossed by any packet). Each router in Fig. 1 presents two more connections to the local Network Interfaces, thus resulting in 5-port routers.

The proposed NoC router architecture features wormhole packet-switched routing with credit-based flow control: this approach reduces the amount of network buffering and allows for a deep pipelined packet communication.

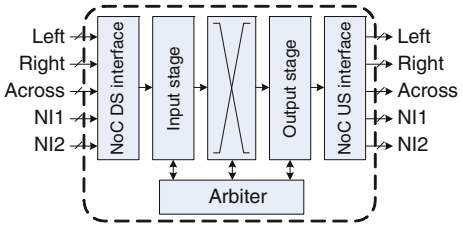

Fig. 6 NoC router architecture

The overall router complexity is kept at a minimum by assigning routing path and QoS (Quality-of-Service) at packet injection.

In fact, the router uses a simple source-based routing, where the entire path is encoded by the NI in the packet header, that has a fixed size due to the symmetry of the topology. This enables fast routing decision, because each router has just to extract the forward information, without any need of computation or look-up tables.

The router also offers best effort and QoS features in terms of both latency and throughput. In the implemented QoS mechanism the requested bandwidth value is programmed in the packet header at the injection point (Network Interface) and is not explicitly linked to the path of a data flow through the router. With this mechanism, the router QoS support is a simple two-step arbitration. When all data flows have the same bandwidth reservation, the arbitration can degenerate into the basic Round Robin, Least Recently Used or fixed priority schemes.

The main blocks of the 5-port router architecture are sketched in Fig. 6. It consists of the NoC *UpStream* (US) interface, i.e., the output to the network, and the NoC *DownStream* (DS) interface, i.e., the input from the network; the Input stage, where the routing and QoS information are extracted from the header; the switching matrix to connect any router input port to any output port; the Output stage, where extra optional buffering is performed; the Arbiter, selecting the winner among the inputs requesting access to the same output. The router has a configurable crossing latency, from zero up to two clock cycles. This is obtained by means of a flexible pipeline in the data path, where registers can be removed and buffers are optional or even bypassable.

Each NoC NI in Fig. 1 interfaces the IP core to the NoC domain, thus constituting the IP core entry point to the communication backbone. Two types of NIs are defined: the Initiator, connecting a Master IP core (e.g. a CPU) to the NoC, and the Target, interfacing a Slave IP core (e.g. a memory) to the NoC. These two NIs are basically dual. In fact, in the NI two data flows can be identified (the pairs of arrows in Fig. 7): *request path* and *response path*. The request path goes from a Master to a Slave, that means from the Master IP core to the NoC in an Initiator NI, and

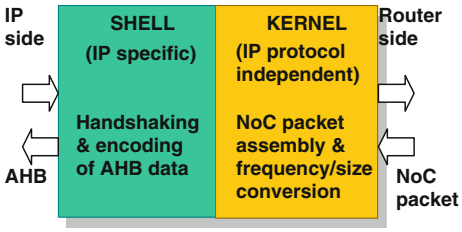

Fig. 7 NI subdivision in Shell and Kernel

from the NoC to the Slave IP core in a Target NI. The other direction, the response path, goes from a Slave to a Master.

The generic NI is made up of two separate components: Shell and Kernel, see Fig. 7. The Shell translates IP transactions (AMBA AHB transactions in our case study) into NoC packets and encodes in the packet header all necessary routing and QoS information. The Kernel is IP-protocol independent and provides optional data bus size and frequency conversion support between the IP and the NoC domain, by means of bisynchronous FIFOs. The FIFOs can also be used to support store & forward transmission, where data are collected up to a specified threshold before being sent; the feature is particularly useful to optimize the use of the network in case of intermittent transmission.

The NI data pipeline is also configurable, from a three-stage one to a zero-latency implementation that removes all input/output registers and FIFOs (thus not supporting size or frequency conversion). The selected configuration for the NoC building blocks implements basic functionalities (conversion of protocols, data size and frequency), plus the store & forward support for transmission to the network, to guarantee efficient parallel communication between cores and memory spaces.

For the sake of completeness, Fig. 8 provides a more detailed insight of a basic Initiator NI architecture with a clear distinction between *request* (upper part of figure, from the IP core to the NoC) and *response* (lower part) paths. From left to right Fig. 8 highlights the different NI components: the Shell, interfacing to the Master IP core and encoding/decoding the NoC packet header; the Kernel, with header and payload FIFOs; the NoC Upstream and Downstream interfaces.

A Target NI presents a dual architecture, with a request path going from a NoC DS (input) interface through a Kernel and a Shell where NoC packet header decoding is performed, while the response path is where the Slave IP core responses are converted into NoC packets to be sent over the network through a NoC US interface.

Figure 9 shows the format of the NoC packet carrying header and payload data. The header field includes both a Network Layer header (HNL) and a Transport Layer header (HTL). The HNL contains the packet routing and

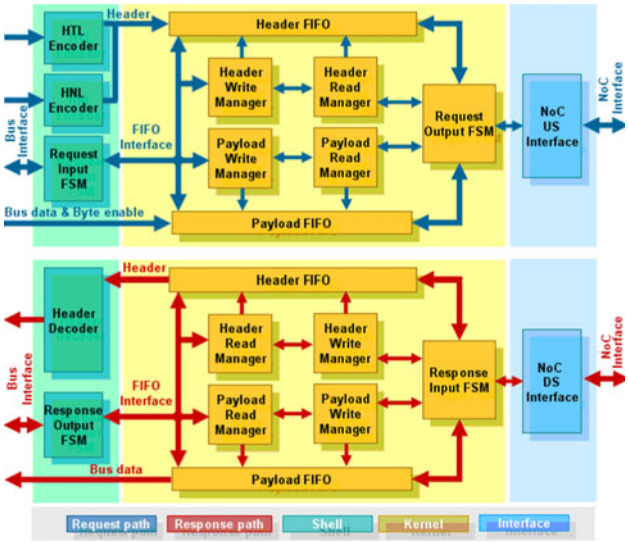

Fig. 8 Main blocks in the NI micro-architecture

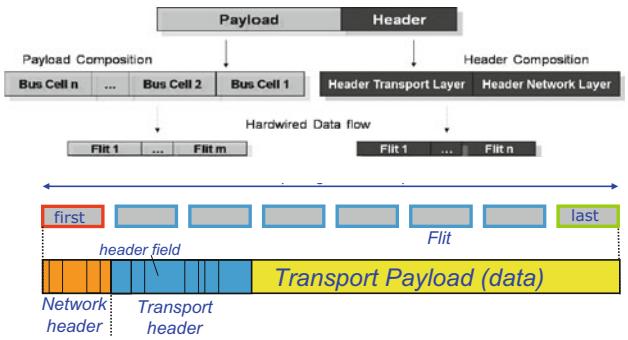

Fig. 9 Internal organization of a Spidergon NoC packet

QoS information to be used by the routers in the network, while the HTL encodes IP protocol information, necessary to convert back the NoC packet into a bus transaction, upon its arrival at destination. The payload of the NoC packet contains the payload data cells of the bus transaction. Both header and payload are physically split in header and payload NoC *flits*; all flits of a packet are routed through the same path across the network. Note also that header and payload information need to travel in separate flits.

In our case study we are not targeting an interconnect for a general purpose system but for specific image/video enhancement algorithms. This way it is possible to estimate the traffic flow and to properly size the interconnection backbone.

In our application the NIs are in charge of converting 32-bit AMBA AHB data transactions into 128-bit NoC transactions. At least the size conversion have to be supported, therefore the zero-latency NI implementation is not suitable. It is also necessary to support frequency conversion, for connecting tiles running at different speed because

of different operation modes (see implementation results in Sect. 6). Since the maximum AHB bus frequency to be supported is 400 MHz, we cannot select the NoC NIs configuration with a single stage of pipeline, constituted by the bisynchronous FIFOs (necessary for conversion), but we need an extra pipeline level given by the bus retiming stage.

The NoC routers have been configured to have one-cycle latency, as this configuration can support the target NoC frequency.

When there is no contention to access the link, any packet entering the network from router i will exit the network from any router k after no more than three cycles (because a maximum of three routers will be crossed, each of them consuming a 1-cycle latency). Moreover, the effect of size and frequency conversion in the NIs adds up to the global latency of the system. However, the latency is not an issue for the target application, as long as the data flow can keep the local frame memories not empty. As a matter of fact, the NoC interconnect should be sized as to support a data flow capable to feed the local memory resources and avoid them to go empty.

By assuming a 400 MHz NoC clock frequency, that can be supported by the selected NoC building blocks configurations, the 128 bits of NoC data size are enough to guarantee a nominal throughput of 128 bits $\times$ 400 MHz $=$ 51.2 Gbps per link, which is sufficient to support all target image/video enhancement functionalities. Moreover, the supported throughput for each link is 4 times higher than the maximum throughput theoretically supported by each IP in Fig. 1 through the AHB interface (considering 32-bit data and 400 MHz clock frequency).

The enabled store & forward feature allows the NI to hold the packet until a certain amount of data is received from the IP. This way, the path in the network is engaged only when a continuous data flow can be guaranteed, thus making full use of the available bandwidth.

NoC packet length depends on the amount of data to be transferred. To utilize the high bandwidth available in the NoC interconnect it is necessary to execute long AMBA AHB transactions: obviously, executing "Store 1 byte" operations would waste the available bandwidth to generate NoC packets composed of a header (whose overhead is about 80 bits per packet, and is transmitted in a separate flit) plus a 128-bit payload flit where only a single byte is significant, thus resulting in a real throughput of 1.6 Gbps. If we execute at least "Store/Load 16 bytes" operations we generate NoC packets composed of a header flit and a payload flit containing 128 data bits, thus achieving a 25.6 Gbps throughput. When further increasing the operation size, for example, to 32 bytes, the throughput increases to 34.1 Gbps. It is therefore clear that the operation size executed in AMBA AHB domain shall be properly selected to take advantage of the high NoC bandwidth.

NoC QoS mechanism is also available, to give higher priority to critical data flows. However, in our application the NoC bandwidth capability is much higher than the AHB one and thus there is no real need to boost critical flows. Therefore, all data flow injected in the NoC by the NIs have the same bandwidth reservation (i.e., the same priority), and the arbitration performed in the routers is a simple Least Recently Used.

5 MPSoC application case studies

To asses the functional performance of the proposed architecture we show the results achieved when programming the MPSoC for different image/video processing and enhancement applications. Comparisons with state-of-the-art techniques addressing similar issues but using different implementation platforms are also reported.

As a first case study we report the performance achieved when the MPSoC platform is used to implement a luminance correction and contrast enhancement algorithm.

The implemented algorithm is the 2D Retinex-like operator originally proposed in [18]. In the current realization the input image to be enhanced (gray scale in the considered example) is processed through the *Filt* and *SF* tiles of Fig. 1 which first extract the luminance component, by applying a recursive low-pass rational filter (4 passes); then by division the reflectance component is obtained. The two image components are separately processed through gamma correction (for the luminance) and detail amplification (for the reflectance) operators. Finally, the two modified components are combined by multiplication in the output enhanced image. To be noted that for this class of algorithms the 2 *ME* tiles plus the *SC* and the *Transf* tiles and the relevant frame memories are not used and are kept in idle mode. Figure 10 presents from left to right the original image to be processed, the output obtained with the Retinex-like algorithm running on the proposed MPSoC platform and the results achieved by another single-scale Retinex algorithm proposed in literature in [5]. This reference algorithm extracts the luminance component by adopting a filter with a large Gaussian kernel of 200 $\times$ 200 taps and has been implemented in real-time in [6] on TI DSP processors: a 32-bit floating-point C6713 DSP clocked at 225 MHz and a 32-bit fixed-point DM642 DSP clocked at 600 MHz, both requiring a power cost of several Watts.

From Fig. 10 it is clear that, although starting from a bad illuminated image, the algorithm originally proposed in [18] and implemented in this work on the MPSoC platform allows an optimal recovery of the image quality

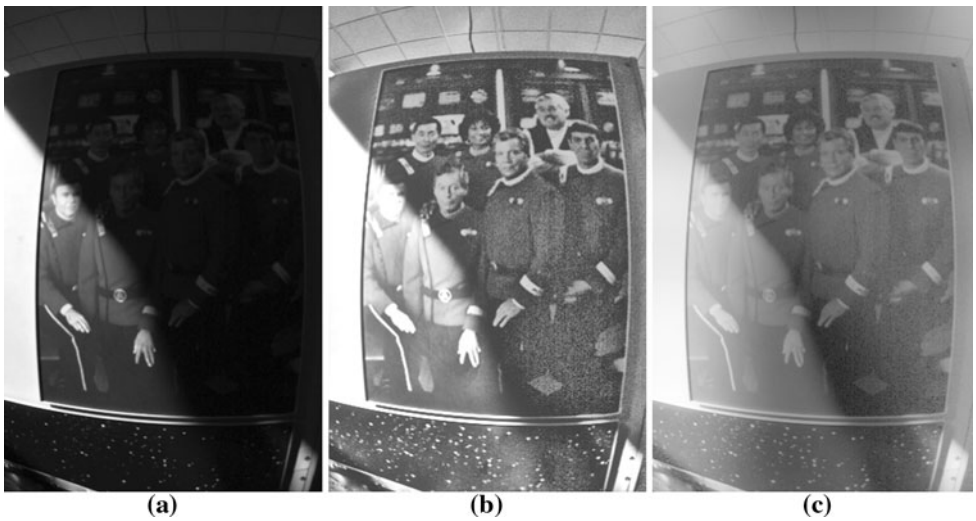

Fig. 10 From left to right: **a** the original image, **b** the output of our algorithm proposed in [18] and implemented in this work on the MPSoC platform, **c** the image processed using the single scale retinex in [5, 6], adopting a large Gaussian kernel of 200 × 200 taps

Fig. 11 Example of Retinex processing for image sequences affected by instantaneous luminance variations. The *left column* shows the original frames, the *right* one the filtered frames

by increasing the luminance level of the whole image but at the same time preserving and enhancing contrasts and edges. On the contrary in the reference single-scale retinex proposed in [5, 6], see Fig. 10, there is a smoothing effect on original image edges and contrast.

The previous technique refers to the use of our MPSoC video platform for enhancement of still images. The same platform can be used also for luminance correction and contrast enhancement of video sequences. As example, by programming the MPSoC to support the 3-branch Retinex-like filtering algorithm proposed in [18] with intra-frame spatial filtering and inter-frame temporal filtering the results of Fig. 11 can be obtained.

Two adjacent frames extracted from a sequence affected by an instantaneous variation of illumination are depicted in Fig. 11. The sequence was acquired by a camera mounted on a car traveling on a road flanked with trees; direct sunlight and shadow rapidly alternate in the scene. The left column shows the original frames, the right one the processed ones, as obtained from the 3-branch algorithm implemented on the proposed MPSoC platform. It can be noted that in the output sequence the luminance variation has been compensated.

Another application case study to assess the functional performances of the MPSoC platform is combining the computational capabilities offered by the *ME*, *FILT* and *SF* tiles to improve the coherency of the MV field extracted from bad illuminated video scenes, e.g. scenes acquired by road-surveillance cameras in real road scenarios with non-controlled light conditions. The implementation of this algorithm [34] on the MPSoC proposed in this work does not require the *SC* and the *Transf* tiles and the corresponding frame memories, which are kept in idle mode.

The top graph in Fig. 12 shows what happens when directly applying a ME algorithm to a real-world video: a car entering a tunnel. A commercial camera working at 30 frames/s SIF video format is accomplishing a horizontal panning to follow the car. To avoid problems of local minima the used motion estimation is a FS algorithm which operates with 16 × 16 blocks, 1 previous frame used for matching, and a search displacement of ±16 pixels in both horizontal and vertical directions. The final resulting MV field in the top graph of Fig. 12 is clearly chaotic and is not fully coherent with the real motion. Instead, the MV field reported on the bottom graph in Fig. 12 is the result of a processing phase realized with our proposed MPSoC platform: a pre-processing filter is first applied for edge-preserving luminance correction and then a ME processing stage is applied on the enhanced video frames.

When using the proposed technique the details are easier to discriminate and the MV filed becomes more accurate (see graph at the bottom of Fig. 12). The MV field is clearly less chaotic and more coherent with the real motion. Hence, this technique is important also in all applications

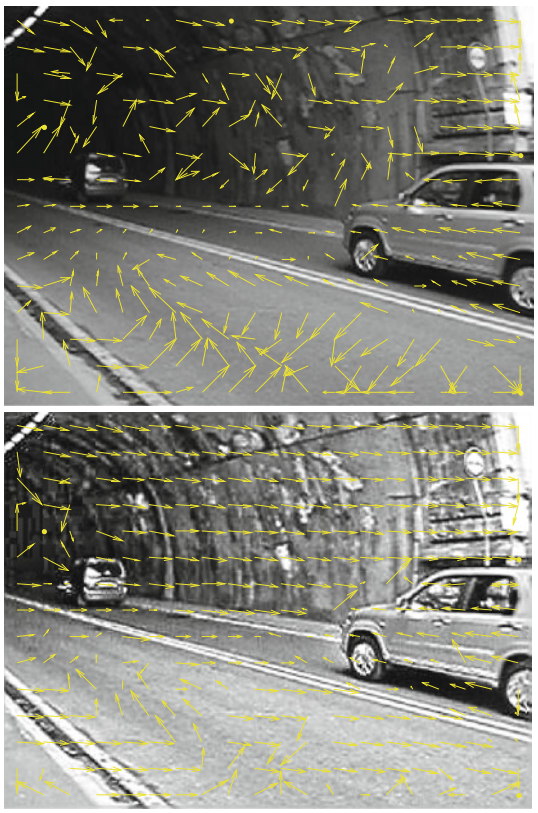

Fig. 12 (*Top*) MV field without preprocessing and (*down*) with preprocess for edge preserving luminance correction before ME

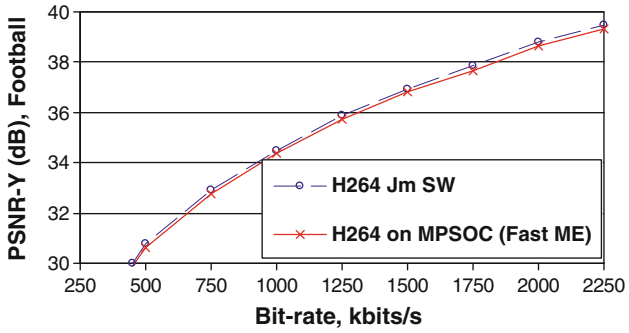

Fig. 13 PSNR versus bit-rate for Football

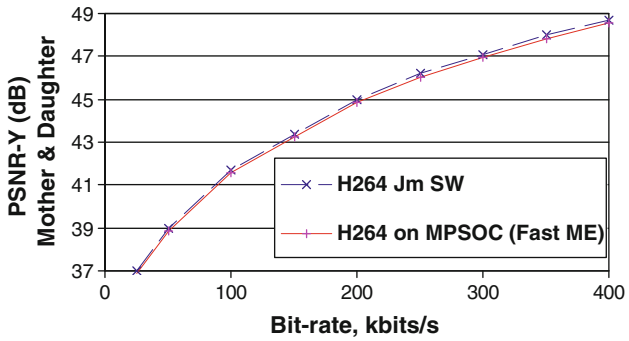

Fig. 14 PSNR versus bit-rate for mother & daughter

where MV field estimation is used as a basis for further enhancement steps.

Finally, a fourth application example of the functional capabilities of the proposed MPSoC platform is its use for video coding. To this aim all tiles have been used to implement a hybrid coding scheme using a fast search adaptive ME [22] with ±16 pixel search displacement, max. 5 previous reference frames for best block matching, SAD error cost function, CABAC entropy coding, 2D Integer DCT transform, in-loop de-blocking filter. With reference to scenes with different degrees of dynamism, such as *Football* 30 frames/s CIF video formats, a sport scene, and *Mother & Daughter* 30 frames/s CIF, a low-dynamic scene, Figs. 13 and 14 present the obtained rate-distortion curves (PSNR in dB vs. bit-rate). The achieved performances are compared to those of a reference MPEG AVC encoder, using JM software implementation [26], configured using CABAC and FS ME with a similar parameter set as described above. From Figs. 13 and 14 it is clear that the proposed MPSoC achieves optimal performances with a PSNR degradation for a fixed bit-rate of less than 0.2 dB versus the reference JM software implementation. As proved in [22, 26, 37], the implementation of the original JM software on general purpose single-core CPUs (Athlon AMD 2400+ in [22], Pentium4@1.7 GHz in [26]) does not allow for real-time processing, even considering small QCIF image formats. Our MPSoC can ensure real-time processing of 30 frames/s VGA videos with a power consumption below 1 Watt (see Sect. 6).

6 CMOS implementation results

The proposed MPSoC architecture has been designed using VHDL language and then synthesized in STMicroelectronics 65 nm low-power CMOS technology. RTL to gate-level synthesis has been accomplished within Synopsys CAD environment while back-end phases have been accomplished within Cadence environment.

The computational and memory complexity of all tiles in Fig. 1 amounts to 520 k logic gates plus 19 Mbits of on-chip SRAM hierarchically organized in 16.45 Mbits level 2 frame memories (*frame mem* blocks in Fig. 1) accessible via NoC and roughly 2.5 Mbits level 1 local memories distributed in the tiles: 1 Mbits each for *Host ITF* and *Ext Mem Ctrl* tiles and 500 kbits distributed in the other heterogeneous tiles. The overall circuit complexity of the MPSoC is determined by both the tiles discussed in Sect. 3 and the resources required by the NoC infrastructure presented in Sect. 4. To meet the functional requirements of the proposed MPSoC each NI is configured to support

32-bit IP AHB bus, 128-bit flits on the NoC side, conversion of protocol, data size and frequency up to a maximum of 400 MHz, FIFOs sized to store 2 locations (data or headers). To meet the 400 MHz requirements a retiming stage has been inserted in the Shell thus each NI presents a 2-stage pipeline. The complexity of this NI configuration is 13.8 k logic gates in 65 nm CMOS technology. As far as the router is concerned it has 5 ports, 128-bit flits data size, an input buffer for each port of 2 locations, and supports the features described in Sect. 4 with a clock frequency of 400 MHz and a complexity in 65 nm CMOS technology of 32.8 K logic gates. The FIFOs in the NoC building blocks are not memory-based, since their small size makes more convenient a register-based implementation. The overall NoC complexity (16 NIs plus 8 Routers) amounts to 483 k logic gates.

Hence, the total MPSoC complexity, due to computing and memory tiles plus NoC infrastructure, amounts to 1,003 k logic gates and 19 Mbits of on-chip SRAM. The number of transistors of the whole platform is about 120 millions.

As far as speed performance is concerned, the maximum achievable frequency depends on the used standard-cells library version. Indeed the used technology provides three types of standard-cells: beside the SVT (standard voltage threshold) version there are also a HVT (high voltage threshold) version optimized in terms of leakage power consumption, and a LVT (low voltage thresholds) optimized in terms of speed but with a much higher power cost. By using a mix of HVT and SVT cells (HVT for all paths with non critical time performances, while SVT only for time-critical paths limiting max. clock speed) we were able to run all processors at 400 MHz while keeping the cost figure *power consumption/MHz*gate* near the minimum value permitted by the technology library. A faster clock, about 800 MHz, could be achieved using LVT cells; however, the power consumption with LVT would be much higher. The leakage power associated to HVT cells is about 12 times smaller than in SVT and 115 times smaller than in LVT. As far as dynamic power is concerned, HVT cells allow for a reduction of a factor of 1.1 if compared with SVT cells and 1.3 if compared with LVT cells. Moreover, the performances achieved at 400 MHz with the HVT/SVT cells are enough for the target mobile and/or handheld applications and hence LVT cells are not used for the synthesis of the proposed MPSoC.

For further power saving three clock configurations are supported by each tile: *power-down mode*, where the clock is deactivated and hence dynamic power consumption is not paid; *low-power mode*, where a tile is clocked at 200 MHz achieving half of the computational power but also spending half of the dynamic power; *high-speed mode*, where the IPs are clocked at full speed. Since the NoC is capable of performing frequency conversion at the NI boundaries, then the tiles can be independently configured in *power-down mode*, *low-power mode* or *high-speed mode*. As example, 3 tiles can be configured to run at 200 MHz, 2 tiles can be kept in idle mode and the others can be clocked at 400 MHz and all can be connected to the NoC running at 400 MHz thanks to the frequency conversion done in the NIs. To be noted that the NoC should provide enough bandwidth for inter-tile communication and hence it is usually clocked at its maximum, 400 MHz, and is kept in idle mode only if all the tiles are in idle mode. Following similar considerations, also the *CPU* tile in Fig. 1, which controls data and operation flows, is usually kept in one of the two active modes.

Table I MPSoC power consumption for 2 different case studies

Resources used	Resources in idle mode	Algorithm	Power, mW
CPU, SF, FILT, Host Itf and Ext mem ctrl tiles and their frame mem, NoC	ME, Transf, SC tiles and their frame mem	2D Retinex-filtering	400
All	None	H.264 encoder with: fast search ME with ±16 search displacement; max. 5 previous reference frames; SAD cost function; CABAC entropy coder; 2D integer DCT transform, in-loop de-blocking filter.	950

The power consumption depends on the workload and on the configuration status of the different tiles. Table I summarizes the power consumption estimated from RTL simulations with reference to the use of the MPSoC for two applications described in Sect. 5: the 2D retinex-like filter, and the H.264 video coding algorithm applied to 30 frames/s VGA videos. With reference to the video coding application example, where all MPSoC computing tiles and relevant frame memories are used, Fig. 15 reports the specific power consumption contribution of each unit: for each block also the contribution of its frame memory is considered; the block 'others' refers to the Host ITF and Ext Mem Ctrl units and their frame memories. From Fig. 15 it is clear that the overall power consumption of 950 mW is dominated by the computing and memory tiles, particularly the ME (up to 40%), while the power cost of the NoC infrastructure is 90 mW, less than 10%.

A comparison of processors with different architectures, computing performances and target technologies is difficult to implement. Just to contextualize the achieved

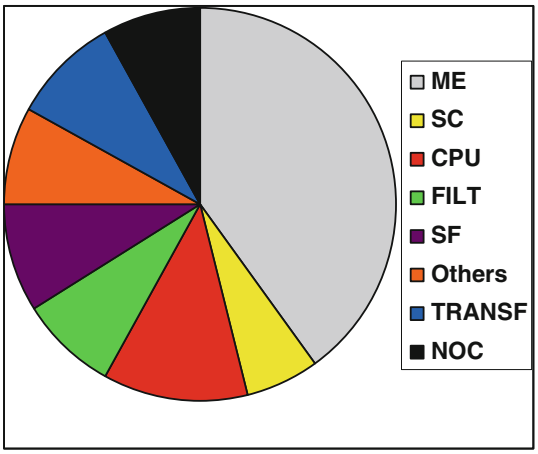

Fig. 15 % contribution of the MPSoC units to the power consumption, H.264 encoder case study of Table 1

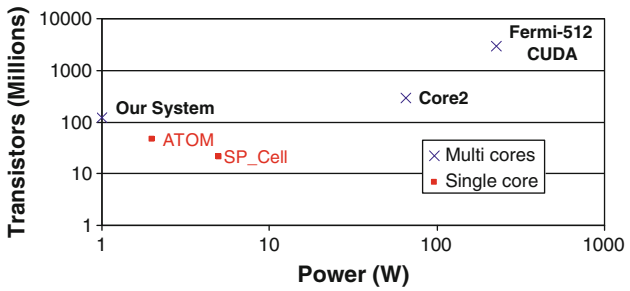

Fig. 16 Transistor count and power cost of state-of-the-art single core (ATOM and 1 SP core of the Cell BE) and multi-core (Fermi with 512 CUDA, Core2, Ours) systems

implementation results Fig. 16 provides a visual representation of the trade-off between transistor count (due to circuit complexity spent for increased parallelism and hence increased performance) and power consumption in state-of-the-art single-core (the Intel AtomZ530 and the SP unit of the Cell BE) and multi-core programmable systems (our MPSoC, the Intel Core 2, the Fermi architecture with 512 CUDA cores). It is worth noting that w.r.t. the 512-CUDA cores Fermi architecture the transistor count and power cost reductions of our architecture are obtained at the expense of a reduced computational capability.

7 Conclusions

A multi-processor architecture for real-time and low-power image and video enhancement applications has been presented. Differently from state-of-the-art parallel architectures, the proposed MPSoC is composed of heterogeneous tiles connected through a novel NoC infrastructure. The architecture offers a good mix between general-purpose computing capabilities of the *CPU* tile, high-throughput DSP capabilities offered by computing tiles dedicated to different classes of algorithms (e.g. motion estimation, special functions for pixel transform, image coding, image transform in the frequency domain, image filtering) and large on-chip memory resources since image/video applications are data-dominated. Memory resources are hierarchically organized in a first level of local memories in the computing tiles plus a second level of shared frame memories accessible via NoC and a third level of off chip DDR-DRAM. Thanks to the packet-switched data transfer scheme the MPSoC architecture avoids communication bottlenecks when more tiles are working concurrently. Moreover, the proposed approach allows an easy scaling of the configuration discussed in this work by modifying the number of used computing tiles, the number and size of the frame memories, the size of IP data and of the NoC links. This is the reason why the desired trade-off between complexity and performance can be easily set. The functional performances of the NoC-based MPSoC have been assessed by presenting the achieved results when the platform is programmed to support 4 different image/video enhancement algorithms.

Finally, the implementation complexity of the NoC-based multi-tile platform, integrated in 65 nm CMOS technology, is reported and discussed: the complexity amount to about 1 million logic gates and 19 Mbits of on chip memory for a total amount of roughly 120 millions of transistors. The complexity overhead is mainly due to the resources required by the computing tiles and frame memories. Running at 400 MHz the MPSoC ensures real-time processing up to 30 frames/s VGA frames. The MPSoC platform supports programmable power-down modes and its power cost, for the proposed case studies, varies from hundreds of mW up to 1 Watt.

As further development of the proposed MPSoC platform an acquisition tile with fast Analog–Digital conversion capability, as in [40], can be added for direct camera interfacing. Given the growing interest of video processing platforms also for aerospace and automotive applications also fast communication bus such as Flexray and/or SpaceWire [41, 42] can be added in the Host ITF tile.

Acknowledgment This work has been partially supported by the EU integrated project SHAPES of the 6th framework programme in collaboration with STMicroelectronics, particularly the group of Dr. M. Coppola (AST, Grenoble).

References

1. Mitra, S., Sicuranza, G.: Non linear image processing. Academic Press (2001). ISBN 0125004516
2. Marshall, S., Sicuranza, G.: Advances in non linear signal and image processing. EURASIP book series, Hindawi Publishing Corp (2006). ISBN 9775945372

3. Marsi, S., Impoco, G., Ukovich, A., Ramponi, G., Carrato, S.: "Using a recursive rational filter to enhance color images". IEEE. Trans. Instrum. Meas **57**, 1230–1236 (2008)
4. Funt, B., Ciurea, F., McCann, J.:"Retinex in Matlab". Proc. IS&T/SID 8th Color Imaging Conf. 112–121 (2000)
5. Jobson, D.J., Rahman, Z., Woodell, G.A.: "Properties and performance of a center/surround Retinex". IEEE. Trans. Image. Process **6**(3), 451–462 (1997)
6. Hines G., et al.: "Real-time enhanced vision system". Proc. SPIE 5802: enhanced and synthetic vision (2005)
7. Shao, Ling, Hao, Hu, de Haan, G.: Coding artifacts robust resolution up-conversion. IEEE. ICIP **5**, 409–412 (2007)
8. Ling Shao, Kirenko, I., Leitao, A., Mydlowski, P.: "Motion-compensated techniques for enhancement of low-quality compressed videos", IEEE. ICASSP. 1349–1352 (2009)
9. N. Parakh, A. Mittal, R. Niyogi, "Optimization of MPEG 2 Encoder on Cell B. E. Processor". IEEE. Int. Adv. Comput. Conf. 423–427 (2009)
10. Nickolls, J., Dally, W.J.: The GPU computing era. IEEE. Micro **30**(2), 56–69 (2010)
11. Xiaohan Ma, Mian Dong, Lin Zhong, Zhigang Deng.: "Statistical power consumption analysis and modeling for GPU-based computing", workshop on power aware computing and systems, Big Sky, MT, USA, October (2009)
12. QuickLogic's Visual Enhancement Engine (VEE™) Brings iridix® to Mobile Devices, (2010)
13. Chang, Chia-Ming., Chien, Shao-Yi., Tsao, You-Ming., Sun, Chih-Hao., Lok, Ka-Hang., Cheng, Yu-Jung .:"Energy-saving techniques for low-power graphics processing unit". ISOCC. **1**, 242–245 (2008)
14. Nam, Byeong-Gyu, Lee, Jeabin, Kim, Kwanho, Lee, Seungjin, Yoo, Hoi-Jun: Cost-effective low-power graphics processing unit for handheld devices. IEEE Commun. Mag **46**(4), 152–159 (2008)
15. Murphy, M., Keutzer, K., Wang, H.: "Image feature extraction for mobile processors". IEEE. IISWC. 138–147 (2009)
16. Nam, Byeong-Gyu, Yoo, Hoi-Jun: "An embedded stream processor core based on logarithmic arithmetic for a low-power 3-D graphics SoC". IEEE. J. Solid-State. Circuits **44**(5), 1554–1570 (2009)
17. Fryza, T.: "Introduction to implementation of real time video compression method". IWSSIP. 217–220 (2008)
18. Saponara, S., Fanucci, L., Ramponi, G., Marsi, S.: Algorithmic and architectural design for real-time and power-efficient Retinex image/video processing. J. Real-Time. Image. Process **1**(4), 267–283 (2007)
19. Saponara, S., et al.: "Motion estimation and CABAC VLSI co-processors for real-time high-quality H.264/AVC video coding". Microprocess. Microsyst **34**, 316–328 (2010)
20. Fanucci, L., et al.: Parameterized and reusable VLSI macro cells for the low-power realization of 2-D discrete-cosine-transform. Microelectron. J **32**(12), 1035–1045 (2001)
21. iPhone 4 Technical Specifications, available at http://www.apple.com/iphone/specs.html
22. Saponara, S., et al.: "Dynamic control of motion estimation search parameters for low complex H.264 video coding". IEEE. Trans. Consumer. Electr **52**(1), 232–239 (2006)
23. Fanucci, L., et al.: A Parametric VLSI architecture for video motion estimation. Integration. VLSI. J **31**(1), 79–100 (2001)
24. Saponara, S., Fanucci, L.: VLSI design investigation for low-cost, low-power FFT/IFFT processing in advanced VDSL transceivers. Microelectron. J **34**(2), 133–148 (2003)
25. Saponara, S., Fanucci, L., Terreni, P.: Design of a low-power VLSI macrocell for non linear adaptive video noise reduction. J. Appl. Signal. Proc **2004**(12), 1921–1930 (2004)
26. Saponara, S., Denolf, K., Blanch, C., Lafruit, G., Bormans, J.: Performance and complexity co-evaluation of the advanced video coding standard for cost-effective multimedia communications. J. Appl. Signal. Proc **2004**(2), 220–235 (2004)
27. Chimienti, A., et al.: "A complexity-bounded motion estimation algorithm". IEEE. Trans. Image. Proc **11**(4), 387–392 (2002)
28. Cocchia, F., Carrato, S., Ramponi, G.: "Design and real-time implementation of a 3-D rational filter for edge preserving smoothing". IEEE. Trans. Consumer. Electronics **43**(4), 1291–1300 (1997)
29. Benini, L., De Micheli, G.: Networks on chip: a new SoC paradigm. IEEE. Comput **35**(1), 70–78 (2002)
30. Gyu Lee H., et al.: "On-chip communication architecture exploration: a quantitative evaluation of point-to-point, bus, and network-on-chip approaches". ACM. Trans. Des. Automation Electron. Syst. vol. **12**, n. 3, (2007)
31. Maruccia, G., et al.: "Method for transferring a stream of at least one data packet between first and second electric devices and corresponding device". US Patent 20090129390 A1, May 2009
32. Vitullo, F., et al.: Low-complexity link micro-architecture for mesochronous communication in networks on chip. IEEE. Trans. Comput **57**(9), 1196–1201 (2008)
33. Grammatikakis, M.D., Coppola, M., Maruccia, G., Locatelli, R., Pieralisi, L.: "Design of cost-efficient interconnect processing units: Spidergon STNoC". CRC Press (2008)
34. Marsi, S., et al.: "Integrated video motion estimator with Retinex-like pre-processing for robust motion analysis in automotive scenarios: algorithmic and real-time architecture design". J. Real.Time. Image. Proc **5**(4), 275–289 (2010)
35. Saponara, S., et al.: Architectural level power optimization of microcontroller cores in embedded systems. IEEE. Trans. Ind. Electron **54**(1), 680–683 (2007)
36. Fanucci, L., et al.: "Power optimization of an 8051-compliant IP micro controller. IEICE. Trans Electron **E 88-C**(4), 597–600 (2005)
37. Ostermann, J., et al.: Video coding with H.264/AVC: tools, performance, and complexity. IEEE. Circuits. Syst. Mag **4**(1), 7–28 (2004)
38. L'insalata, N., et al.: "Automatic synthesis of cost effective FFT/FFT cores for VLSI OFDM systems". IEICE. Trans. Electron **E 91-C**(4), 487–496 (2008)
39. Chimenti, A., et al.: "VLSI architecture for a low-power video codec system". Microelectron. J **33**(5–6), 417–427 (2002)
40. Saponara, S., et al.: "Architectural exploration and design of Time-interleaved SAR arrays for low-power and high speed A/D converters". IEICE. Trans. Electron **E 92-C**(6), 843–851 (2009)
41. Baronti, F., et al.: "Design and verification of hardware building blocks for high-speed and fault-tolerant in-vehicle networks". IEEE. Trans. Ind. Electron **58**(3), 792–801 (2011)
42. Saponara, S., et al.: Radiation tolerant space wire router for satellite on-board networking. IEEE. Aerosp. Electron. Syst. Mag **22**(5), 3–12 (2007)

Author Biographies

Sergio Saponara got the Laurea degree cum laude, and the Ph.D., in Electronic Engineering from the University of Pisa in 1999 and 2003, respectively. In 2002 he was with IMEC, Leuven (B), as Marie Curie Research Fellow. Since 2001, he collaborates with Consorzio Pisa Ricerche in Pisa. He is a senior researcher at the University of Pisa in the field of electronic circuits and systems for telecom, multimedia, space and automotive applications. He holds the chair of electronic systems for automotive and automation at the Faculty of Engineering.

He co-authored more than 130 scientific publications and holds 5 patents. Sergio Saponara is also a research associate of CNIT and INFN and served as a guest editor of special issues of international journals and as a program committee member of international conferences.

Luca Fanucci got the Master of Science and the Ph.D. degrees in Electronic Engineering from the University of Pisa in 1992 and 1996, respectively. From 1992 to 1996, he was with ESA/ESTEC, Noordwijk (NL), as a research fellow. From 1996 to 2004 he was a senior researcher of CNR in Pisa. He is Professor of Microelectronics at the University of Pisa. His research interests include VLSI architectures for integrated circuits and systems. Prof. Fanucci co-authored more than 150 scientific publications and he holds more than 10 patents. He was the program chair of IEEE Euromicro DSD 2008 and IEEE DATE Designer's Forum.

Esa Petri received her M.Sc. degree in Electronic Engineering in 2003 and the Ph.D. degree in Electronic Systems for Automotive Engineering in 2010, both from the University of Pisa (IT). From 2004 to 2005 she was with ESA/ESTEC, Noordwijk (NL). Since 2006 she collaborates with the Microelectronic Systems Division of Consorzio Pisa Ricerche (IT) on several projects of industrial relevance in the fields of HW/SW embedded system architectures and networking.