Reconfigurable computing for future vision-capable devices

Miguel Bordallo López[1], Alejandro Nieto[2], Olli Silvén[1], Jani Boutellier[1], David López Vilariño[2]
[1] University of Oulu, Finland
[2] CITIUS, Universidad de Santiago de Compostela, Spain
miguelbl@ee.oulu.fi

Abstract

Mobile devices have been identified as promising platforms for interactive vision-based applications. However, this type of applications still pose significant challenges in terms of latency, throughput and energy-efficiency. In this context, the integration of reconfigurable architectures on mobile devices allows dynamic reconfiguration to match the computation and data flow of interactive applications, demonstrating significant performance benefits compared to general purpose architectures. This paper presents concepts laying on platform level adaptability, exploring the acceleration of vision-based interactive applications through the utilization of three reconfigurable architectures: A low-power EnCore processor with a Configurable Flow Accelerator co-processor, a hybrid reconfigurable SIMD/MIMD platform and Transport-Triggered Architecture-based processors. The architectures are evaluated and compared with current processors, analyzing their advantages and weaknesses in terms of performance and energy-efficiency when implementing highly interactive vision-based applications. The results show that the inclusion of reconfigurable platforms on mobile devices can enable the computation of several computationally heavy tasks with high performance and small energy consumption while providing enough flexibility.

1. Introduction

Mobile devices have been identified as promising platforms for interactive vision-based applications [1]. As their computational resources grew, they became increasingly suitable for tasks related to the analysis and understanding of images and videos. However, this type of applications still pose significant challenges. Many times, interactive vision-based applications are in fact impractical, since their requirements of computational power and energy make them unusable for extended periods of time. Consequently, their computational throughput needs should be carefully balanced with the energy-efficiency of the system. This becomes even more apparent in the most interactive cases, such as vision-based user interfaces, where the latency and energy-efficiency needs are even more pronounced.

The computing and sensing platforms integrated in current mobile devices still present some limitations in the scalability for higher resolutions and more complex algorithms, while still maintaining energy-efficiency. Future high-resolution cameras and high-performance applications are likely to require more specific solutions such as dedicated image processors or reconfigurable hardware architectures.

This paper presents concepts laying on platform level adaptability, exploring the acceleration of vision-based interactive applications through the utilization of three reconfigurable architectures. Based on the analysis of interactive applications and user interfaces, several computationally expensive image processing kernels are implemented using three different reconfigurable architectures. In this context, a processor with a reconfigurable accelerator is proposed as a low-power high-efficiency alternative, or complement to the current ARM processors and NEON units. A hybrid reconfigurable SIMD/MIMD platform is proposed to complement mobile GPUs. Lastly, the inclusion of the flexible Transport-Triggered Architecture-based processors is proposed as a low-power complement to current DSP-based solutions. All three architectures are evaluated and compared with their current counterparts, analyzing their advantages and weaknesses in terms of performance and energy-efficiency when implementing highly interactive vision-based applications.

2. Reconfigurable architectures

A reconfigurable processor is a processor with erasable hardware that can rewire itself dynamically. This allows the chip to adapt effectively to the programming tasks demanded by the particular software they are interfacing with at any given time. Ideally, a reconfigurable processor can transform itself to run applications across different fields with the highest possible performance.

The scalability of the performance of general purpose processors has been recently declining. Even with transistor densities improving according to Moore's law, the failure of Dennard Scaling [2] and the lack of proportional improvements in battery technology will prevent future devices from utilizing the whole die area at the same time. Alternative processors such as the GPU could be used as energy-efficient architecture alternatives. Thus, their use is likely to rapidly increase. However, the current architectures included in mobile devices have noticeable drawbacks either in future scalability or lack of flexibility. In this context, reconfigurable computing has the chance of becoming a future mainstream alternative as a part of the future scalable mobile architectures [3].

Over the years, numerous reconfigurable architectures have been proposed to fill the gap between the performance of ASICs and the flexibility of General Purpose Processors. Computer Vision algorithms and applications are inherently comprised of very variable of tasks that range from low-level pixel processing to high-level inference of abstract representations. Reconfigurable computing, oriented in performance, but with specific flexibility in mind, adapts extremely well to this paradigm [4].

Several processors aim to meet the processing requirements of camera pipelines without compromising the costs. This is the case of the CRISP stream processor (Coarse-grained Reconfigurable Image Stream Processor) [5] which outperforms modern DSPs in these kinds of tasks by a factor up to 80.

Other processors focus on the inherent parallelism of image data to enhance the performance of computation-intensive tasks by including SIMD units. The MorphoSys processor [6] adds a reconfigurable SIMD coprocessor based on a 2-dimensional mesh with enhanced connectivity to a RISC core utilized for control tasks. To exploit task parallelism, other architectures are designed with the focus on the execution of different tasks at the same time [7] [8].

Low-level image processing, inherently data parallel, usually consumes most of the computation time. However, subsequent tasks are also time-consuming, and custom accelerators that allow task parallelism are often a requirement. Hybrid architectures permit facing both processing stages, reducing hardware requirements and taking advantage of the interaction between these stages to improve performance, instead of considering them independently. Embedding FPGAs in modern SoCs composing an heterogeneous system provides for flexibility and high performance [9].

Exploiting both data and task parallelism, heterogeneous reconfigurable architectures such as the HERA processor [10] have been developed. Usually composed of two complementary units and a data sharing network, they present some limitations in dataflow control. Some of these limitations can be overcome by the inclusion of a RISC core that executes sequential parts of the algorithm and takes care of control flow [11].

Since many architectures still require a general purpose or domain specific processor, other reconfigurable architectures focus on assisting a more general counterpart such as a DSP to perform specific tasks in a faster manner [12].

Utilizing an example architecture of each type, the rest of this paper analyzes three different styles of reconfigurable architectures, a reconfigurable accelerator for a RISC processor (EnCore), a task/data parallel reconfigurable architecture (Hybrid) and reconfigurable application-specific processors to assist the general processor (TTA).

3. Experimental setup

For experimental purposes, to provide a comparison with the selected reconfigurable architectures, we benchmark different computer vision kernels across several current platforms. In this context, we have measured and estimated the performance on two different devices based on the Texas Instruments OMAP3 family, OMAP3430 and OMAP3530. The platforms, a Beagleboard revision C and a Nokia N900 mobile phone, include an ARM Cortex-A8 CPU, a PowerVR SGX530 GPU and a TMS320C64x+ DSP. The OMAP3530 SoC can be set to use at least six different operating points, with frequencies of the main processor ranging from 125 to 720MHz and DSP frequencies up to 520MHz. The chosen operating point features a 600MHz ARM core, with a 430MHz DSP and a 110MHz GPU. For the selected operating point, the single-core ARM processor presents a maximum power consumption of 550mW. The utilization of the NEON coprocessor increases the consumption to 670mW. The mobile GPU consumes about 93mW alone and about 110mW including the overheads of the memory readings. Lastly, the DSP core consumes about 248mW [13] [14].

4. EnCore processor with a Configurable Flow Accelerator

The EnCore processor [15] [16] is a configurable 32-bit single-issue RISC core which implements the ARCompact instruction set [17]. The processor can be integrated on a System on Chip, together with an extension interface for reconfigurable accelerators. The specific reconfigurable accelerator of the EnCore architecture is called the Configurable Flow Accelerator (CFA). It defines a small Instruction Set Architecture (ISA) which allows the customization of Application-Specific Instruction-set Processors (ASIP) through the use of user-defined Instruction Set Extensions (ISE).

Figure 1 shows a simplified schematic of the EnCore Castle datapath. Although not shown, the current implementation has a 5-stage pipeline. The Fetch block manages instruction supply. There are two banks of registers. The first, a general-purpose-processor register bank (GPP) is employed for the standard ALU of the CPU. The second register bank stores data for the CFA extension datapath.

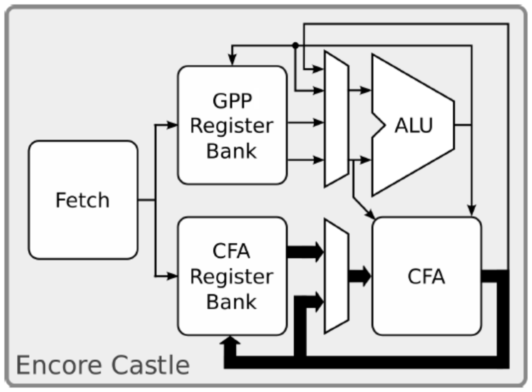


Figure 1. EnCore Processor simplified organization scheme.

In a similar manner as SIMD coprocessors in ARM processors, the EnCore CFA defines an Instruction Set Extension (ISE) which includes specific operations that can be used to accelerate algorithmic computations. The particularity of the CFA is that it enables the definition of custom instructions that specifically adapt to the algorithm in hand. The instructions include additional arithmetic operations or a combination of them, facilitating the speed-up of the most critical parts of the application.

The inclusion of the CFA entails only a limited increase in hardware resources and power consumption, but usually implies a large increase in performance. This is achieved by making use of several single-function ALUs that allow spatial and temporal parallelism through resource sharing and pipelining. In addition, the CFA is fully programmable, supporting up to 64 reconfigurable extension instructions.

Figure 2 presents a simplified scheme of the CFA unit. A set of ALUs and multiplexers allow the data alignment through shuffling. The CFA is highly configurable and its datapath runs multi-cycle operations. The CFA has a 3-stage pipeline and is able to handle 4 independent arithmetic operations according to the configuration of the particular ISE under execution. The CFA register bank supplies a vector of 4 elements to the CFA, storing up to 10 of them.

The programmability of the CFA is exploited by the definition of new specific instructions that adapt to the desired algorithm or application. The process of analyzing the applications to identify candidate instructions, can be done in a manual or automatic manner, resulting in a series of templates that adapt to an existing CFA or the generation of a new one. In this context, the EnCore processor employs a design flow for automated construction of ISEs [15]. Figure 3 shows how a custom ISE is mapped in the CFA unit.

Using an adapted *gcc* compiler to identify and exploit the more suitable ISEs, the resulting CFA mapping can reuse the results across different applications. The resulting ISEs are larger and more complex than the standard RISC instructions. It has to be noted that in order to adapt to the available CFA hardware resources, not all the ISEs identified by the compiler are necessarily matched to the CFA. In practice, there is a trade-off between specific ISEs with low latency and high resource usage and reusable shared instructions with higher instruction latencies. Thus, data allocation becomes critical in the performance maximization process [18].

4.1. Improving vision-applications using an EnCore processor

To provide for an example of the possible improvements obtained by using the EnCore processor to execute vision-based interactive applications, the execution of several operations has been measured. The evaluation of the several image processing kernels has been done using two setups, the EnCore processor alone and the Encore processor utilizing a CFA. The setup for the EnCore processor consists on an EnCore Castle chip that has been used to obtain the measurements in terms of performance and energy efficiency. The Castle test-chip, a second iteration of the architecture, is fabricated with a generic 90nm CMOS process, and occupies only 2.25 mm^2, including the CFA and two 32KB caches. Embedded within a SoC providing a generic 32-bit memory interface, the processor features an operation clock-rate of 600MHz, with a maximum power consumption of 70mW on typical conditions. The results are compared with a mobile CPU (ARM Cortex-A8), with and without the use of a NEON unit. Table 1 presents a summary of the experiments. The energy measurements are based on increase of power compared against an idle processor, and do not include static power consumption.

The experiments show how the performance of the EnCore processor is comparable to the ARM. However, EnCore, designed with energy-efficiency in mind, consumes much less power. The energy consumption of the EnCore processor represents from 8 to 33% of the ARM consumption, depending on the image processing kernel. However,

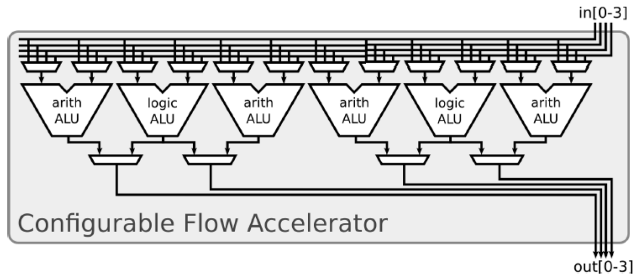


Figure 2. A simplified scheme of the Configurable Flow Accelerator (CFA).

Table 1. Cycles per pixel (CPP) and nanoJoules per pixel (nJPP) needed by several algorithms in the ARM and EnCore processors including accelerators.

	CPP				nJPP			
Operation	ARM	NEON	EnCo	CFA	ARM	NEON	EnCo	CFA
Grayscale conv.	216,4	156,1	240,2	66,0	197,7	174,2	28,0	7,7
Image displac.	78,4	56,1	50,0	47,3	71,5	62,5	5,8	5,5
Alpha Blending	141,2	100,1	85,9	20,0	128,8	111,7	10,0	2,3
Blur detection	72,8	52,3	84,1	19,5	66,6	58,0	9,8	2,3
2D-Convolution(3x3)	422,8	301,9	199,2	57,9	405,1	337,2	23,2	6,8
Histogram	21,4	21,4	29,0	20,1	19,25	19,25	3,4	2,3
Image Rotation	546,1	390,1	607,8	234,0	500,0	435,5	70,9	27,3
Image Scaling	384,3	250,0	390,3	143,2	352,2	279,2	45,5	16,7

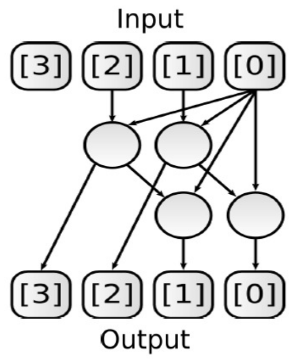


Figure 3. An example of a custom instruction of the Configurable Flow Accelerator. Four input registers and four independent arithmetic operations result in four output registers.

it has to be noted, that the comparison with newer ARM models (e.g. Cortex A15) could reduce the gap to about the half.

Depending on the operation, the use of the NEON co-processor increases the performance of the ARM core up to 50% while increasing the total power consumption only 20%. The potential of a reconfigurable accelerator can be seen in the comparison of the NEON unit with the CFA of the EnCore processor. The performance of the EnCore processor increases up to 4 times for certain kernels, such as grayscale conversion or alpha blending. The difference is more noticeable in simpler kernels, where the needed arithmetic operations can be mapped directly into a single CFA instruction.

Computationally expensive kernels can benefit from a reconfigurable co-processor. The integrated nature of the CFA unit can be included in the tool-chain in a transparent manner. However, for memory intensive operations, with bottlenecks mainly dependent on fast data access, the performance gains are expected to be smaller.

The low-power design of the EnCore/CFA configuration also implies a very important gain in energy consumption. The reconfigurable setup outperforms the ARM/NEON combination, consuming only 5% of the energy.

The EnCore processor with its CFA proves to be a very good alternative to reduce the power consumption of mobile microprocessors. In this context, an asymmetric configuration of one or two ARM cores with several EnCore processors in a multicore architecture could be a viable option for future SoCs.

5. SIMD/MIMD dynamically-reconfigurable architecture

Interactive vision based applications integrate complex computer vision algorithms that include a wide range of operations, data dependencies and program flows. Current GPU devices, although extremely performance-efficient in certain tasks, lack the flexibility of an unrestrained program flow control that can adapt to different types of parallelism when faced with looping and branching. In this context, a reconfigurable architecture that is able to reorganize its processing elements is a suitable candidate to complement current mobile GPUs.

A hybrid SIMD/MIMD dynamically-reconfigurable architecture is essentially an image coprocessor designed to take advantage of the different types of parallelism (data parallelism and task parallelism) present on each algorithm by adding a flexible datapath to the processor. Keeping certain similarities with GPUs, the hybrid platform is essentially a many-core architecture able to process many operations concurrently. However, the addition of the flexible data path allows the architecture to reconfigure during the program flow to select the best characteristics for SIMD and MIMD (Multiple Instruction Multiple Data) computing paradigms.

The hybrid architecture features general purpose capabilities, dynamic and at-runtime reconfiguration that can select the SIMD or MIMD modes as needed. The architecture is completely modular and scalable, adaptable according to the requirements of the algorithm or the target platform. In addition, it aims to reduce the set-up time and ease algorithm migration by automatically managing tasks such as data I/O or synchronization between the computing units. The architectural details can be found in the work of Nieto et al [19].

Figure 4 depicts the main elements of the Image Coprocessor. It is composed of three main elements: two I/O Processors, the Programmable Input Processor (PIP) and the Programmable Output Processor (POP), and a set of Processing Elements (PEs). Depending on the configuration of the coprocessor, the set of PEs can execute both SIMD and MIMD computing paradigms. In the SIMD mode, all PEs execute the same instruction, exploiting the spatial (data) parallelism. In the MIMD mode, each PE executes a small kernel of the whole algorithm, making it possible to take advantage of the temporal (task) parallelism. Two different networks enable data sharing between the PEs.

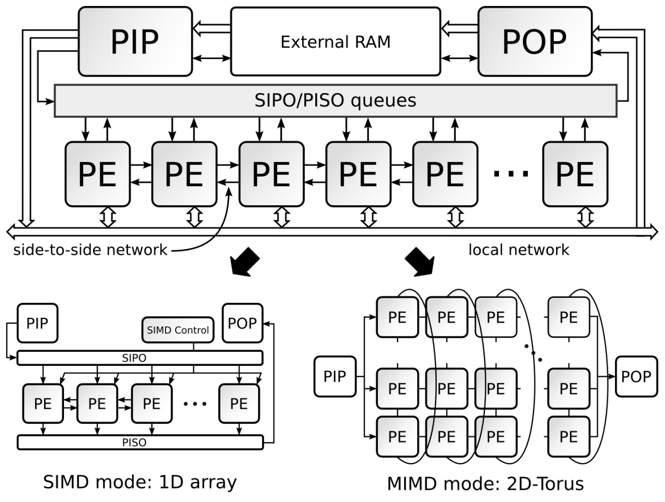


Figure 4. Schematic view of the Hybrid Image Coprocessor and the operation modes. [20].

In the SIMD mode, adjacent PEs can exchange data synchronously using the *side-to-side network*, while in the MIMD mode, the different kernels executed on the different PEs are chained, employing the *local network*. This mode uses the Stream Queues to enable automatic synchronization, therefore no additional operations are needed. The different modules of the architecture are enabled, depending on the operation mode, and this selection depends on the algorithm characteristics and how the different tasks are scheduled.

5.1. Accelerating vision-based applications with a Hybrid reconfigurable architecture

The hybrid SIMD/MIMD architecture is currently prototyped on an FPGA for evaluation purposes. The target device is a Xilinx Virtex-6 X240T, included on the Xilinx ML605 Base Board [21]. An AXI4-based MicroBlaze SoC with Multi-Port Memory Controller and 10/100-Ethernet units was implemented to support the Image Coprocessor. It was configured with 128-PEs of 32-bit each. Their ALUs only support integer and fixed-point arithmetic in order to save FPGA resources. Due to the FPGA characteristics, turning on the prototype consumes a static power of 1.97W. Clocked up to 150MHz, the peak performance is 19.6GOP/s and the maximum power consumption is 7.197W. More details of the hardware prototype are available in the article by Nieto *et al.* [19]. The experimental architecture includes a basic linker and an assembler type programming interface.

To provide an example of the possible benefits of employing a Hybrid SIMD/MIMD platform for accelerating vision-based interactive applications, this section evaluates its use in several computer vision kernels utilized in many interactive camera applications and vision-based UIs. The performance is compared with an ARM Cortex-A8 processor and a PowerVR530 GPU, both included on the OMAP3430 SoC. The energy measurements are based on increase of power compared against an idle processor, and do not include static power consumption. The results show that the Hybrid SIMD/MIMD platform can outperform mobile CPUs and GPUs in scenarios requiring a flexible data path and parallel computations. Table 2 summarizes the performance of the platform compared with a mobile CPU and a mobile GPU.

The measurements show that the Hybrid architecture, designed with emphasis in performance, outperforms the ARM processor in speed and energy efficiency for all the implemented image kernels. When compared with a mobile GPU, the flexibility of the Hybrid platform offers a considerable advantage in operations that require a more complicated program flow, such as feature extraction, 2D-convolution or LBP computation. Its flexibility is better exploited with long image pipelines that can take more advantage of its task-parallel capabilities through the SIMD configuration.

However, when data access patterns become irregular, the performance of the Hybrid platform is hindered. For example, for pixel-wise operations typically present in graphics processing such as image rotation and scaling, the well optimized GPU still outperforms the Hybrid platform.

The speedups obtained by the hybrid platform imply a smaller energy consumption for the less parallelizable kernels, such as the image histogram or the feature matching even when implemented on an FPGA. Although the FPGA implementation employed techniques to lower the power, the Virtex-6 family is not the most suitable platform for power-critical systems. It is expected that the implementation of the architecture in silicon can drop the energy consumption by at least one order of magnitude [22] [23].

The results suggest that a Hybrid SIMD-MIMD platform is a good alternative to be used in conjunction with a GPU, providing for a flexible architecture, able to exploit different types of parallelism and supporting different program flows.

Table 2. Cycles per pixel (CPP) and nanoJoules per pixel (nJPP) needed by several algorithms in the mobile CPU, mobile GPU and Hybrid platforms.

	CPP			nJPP		
Operation	ARM	mGPU	Hybrid	ARM	mGPU	Hybrid
Grayscale conversion	156,1	13,4	2,1	174,2	11,3	72,8
Image displacement	56,1	13,6	1,3	62,5	11,5	45,0
Alpha Blending	100,1	13,6	1,0	111,7	11,5	34,7
Feature Extraction	548,7	75,5	0,7	613,0	63,8	24,3
Blur detection	52,2	100,7	1,0	58,0	85,1	34,7
LBP extraction	37,0	17,9	0,2	41,3	15,2	6,9
iLBP extraction	76,8	17,9	0,3	85,8	15,2	11,2
2D-convolution(3x3)	301,9	160,2	1,0	337,2	135,3	34,7
Histogram	21,4	-	2,4	19,25	-	83,2
Image Rotation	390,1	13,6	12,0	435,5	11,5	416
Image Scaling	250,0	20,3	136,7	279,1	16,9	4714

6. Transport-triggered architecture

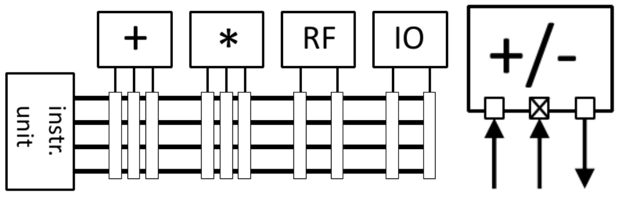
Current mobile Image Signal Processor (ISP) architectures are based in a combination of programmable Digital Signal Processors with monolithic and inflexible hardware *codecs*. Future vision-capable mobile platforms are expected to provide for energy-efficient solutions with enough flexibility and programmability to adapt to several scenarios. In this context, the inclusion of reconfigurable architectures in future devices, designed for computing and sensing tasks is a suitable solution.

Transport-Triggered Architecture (TTA) is a processor technology that is fundamentally different from conventional processor designs [24]. TTA resembles the VLIW processor architecture and exploits instruction-level parallelism, executing multiple instructions simultaneously in the same clock cycle. While in mainstream embedded and signal processors computations are triggered by processor instructions that are accompanied with operands, in TTA processors there is only one instruction: move data. Computational operations are triggered as side-effects of data moves. TTA resembles the VLIW processor architecture and exploits instruction-level parallelism executing multiple instructions simultaneously in the same clock cycle

TTAs fetch and execute several instructions in parallel every clock cycle. This makes TTA processors well-suited for computationally intensive signal processing-style computations that offer abundant instruction-level parallelism.

An example TTA processor is depicted in Figure 5. In a TTA design, there is no theoretical limit to the number of buses (and respectively, number of instructions executed in parallel). However, the maximum operating frequency goes down as the number of buses increases.

Also, the maturity of the TTA design tools, that include a standard C compiler, makes the platform especially attractive for high performance applications with moderate devel-



(a) Toy TTA (b) Function Unit

Figure 5. A toy TTA and one of the function units. [20]

opment times, and easy to integrate with other chip designs.

6.1. Improving intensive operations with TTA processors

To provide for an example of the possible benefits of employing TTA processors for accelerating vision-based interactive applications, this section evaluates the use of two different programmable application-specific instruction processor (ASIP). The first one is capable of performing LBP feature extraction for HDTV resolution video at a modest clock frequency of 304MHz in real time [20] [25]. The second one, presented by Pitkänen *et al.* [26] is designed to accelerate 1024-point fast Fourier transforms with minimum latencies, employing a maximum frequency of 400MHz.

The custom TTA processors, have been designed with the open source TCE tool-set [27] that provides a complete co-design flow all the way to ASIC synthesis. The programmability of the processor enables changing its operation solely by software updates. To verify the functionality and to measure the power consumption to test the suitability for mobile devices, the processors can be synthesized on an FPGA board.

In the case of the LBP processor, for evaluation purposes, the FPGA used for measurements and testing was Altera EP4CE115F29C7 Cyclone IV. The FFT processor was synthesized using 130nm CMOS standard sell ASIC technology.

The resulting processor, a TTA-based ASIP, can be integrated in numerous interactive applications such as face detection and tracking systems. The TTA processor, designed to minimize the dissipated power while keeping the programmability requires about 11 Cycles per Pixel (*CPP*) for the non-interpolated LBP, while the interpolated LBP requires 20 *CPP*. This implies a figure of energy consumption equivalent to only 1,1 pJ/pixel, which proves that the approach is extremely power efficient. This means that, even on the FPGA prototype, the real-time processing of HD720p frames at 30 fps. can be achieved while keeping the power consumption below 30mW. It is expected that the synthesis of the processor in silicon could mean a possible increase in the power efficiency of about one order of magnitude.

To provide a comparison, the LBP implementation is compared with highly optimized implementations on a DSP [28] [29] and optimized ARM and NEON implementations [20] [30]. The DSP core, explicitly designed for signal processing tasks, offers a performance about four times faster than the ARM and NEON implementation. The DSP code is a carefully optimized code and makes use of DSP intrinsics.

For the LBP processor, the experiments suggest a *CPP* count of 6.7 and 11.8 for LBP and interpolated LBP, respectively. These numbers show that the DSP is actually faster than the reconfigurable TTA processor, which make it still a very suitable candidate for high resolution and high performance applications. However, when distributing the performance over power, the TTA processor is about 3,5 times more efficient than the DSP, making it especially suitable for continuous sensing tasks. The energy efficiency of the TTA processor could be improved further with the synthesis of the processor in silicon [22] [23].

On the FFT processor, the TTA-architecture outperforms the DSP in terms of *CPP*. Implementation in silicon makes also the processor about 3.5 times more energy efficient. Table 3 shows a summary of the experiments.

Table 3. Cycles per pixel needed by the two TTA processors compared with a DSP and an ARM processor.$^{(1)}$ FPGA. $^{(2)}$ 130nm technology

	CPP			nJPP		
Operation	ARM	DSP	TTA	ARM	DSP	TTA
LBP	37,0	6,7	11,0	41	3,9	1,1$^{(1)}$
iLBP	76,8	11,8	20,0	86	6,9	2,0$^{(1)}$
FFT	160	6	5,0	146	3,5	1,1$^{(2)}$

The analysis of the results show that the TTA processors outperform the ARM processor in both CPP and JPP metrics. Compared with the more specific DSP, designed to take advantage of instruction parallelism, the performance is comparable. However, the TTA processors prove to be more energy efficient, even when implemented on an FPGA. Future implementations of the LBP processor in silicon are expected to reach even better energy-efficiency.

Although not a replacement for current DSPs, the inclusion of several reconfigurable TTA processors in a mobile architecture enables the computation of several specific tasks with small energy consumption, while still providing enough flexibility.

7. Discussion

The main contribution of this article is the analysis of the characteristics of reconfigurable computing platforms in the context of mobile vision-based interactivity. The comparison of different architectures and the characterization of their advantages and shortcomings is achieved through the implementation of several compute-intensive image processing kernels. The analysis of the trade-off between the flexibility of general purpose processors and the high performance of dedicated hardware circuits provides advances towards future devices that include platform level adaptability concepts.

The use of reconfigurable processors in future mobile platforms is a very attractive opportunity. However, there are still major challenges that have to be addressed. The main target of future research is the integration of reconfigurable computing into existing hardware and software systems. In this context, the investment in the development of efficient tools that help the exploitation of such devices is imperative. The identification of novel emerging applications and their possible bottlenecks and constrains in terms of size and energy consumption is paramount for the definition of the boundaries between software and hardware, and the apparition of new paradigms of reconfigurable computing. In this context, the future of reconfigurable computing on vision-capable mobile devices will be determined by the same trends that affect the development of these systems today. System integration, dynamic reconfiguration and high-level compilation are still major areas that require development [31].

References

[1] F. Zhou, H. B.-L. Duh, and M. Billinghurst, “Trends in augmented reality tracking, interaction and display: A review of ten years of ismar,” in *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*. IEEE Computer Society, 2008, pp. 193–202. 1

[2] H. Esmaeilzadeh, E. Blem, R. St Amant, K. Sankaralingam, and D. Burger, “Dark silicon and the end of multicore scaling,” in *Computer Architecture (ISCA), 2011 38th Annual International Symposium on*. IEEE, 2011, pp. 365–376. 2

[3] E. Chung, D. Burger, M. Butts, J. Gray, C. Thacker, K. Vissers, and J. Wawrzynek, "Reconfigurable computing in the era of post-silicon scaling," in *Field-Programmable Custom Computing Machines (FCCM), 2013 IEEE 21st Annual International Symposium on*. IEEE, 2013, pp. xvi–xvi. 2

[4] K. Bondalapati and V. K. Prasanna, "Reconfigurable computing systems," *Proceedings of the IEEE*, vol. 90, no. 7, pp. 1201–1217, 2002. 2

[5] J. C. Chen and S.-Y. Chien, "Crisp: coarse-grained reconfigurable image stream processor for digital still cameras and camcorders," *Circuits and Systems for Video Technology, IEEE Trans.*, vol. 18, no. 9, pp. 1223–1236, 2008. 2

[6] H. Singh, M.-H. Lee, G. Lu, F. J. Kurdahi, N. Bagherzadeh, and E. M. Chaves Filho, "Morphosys: an integrated reconfigurable system for data-parallel and computation-intensive applications," *Computers, IEEE Transactions on*, vol. 49, no. 5, pp. 465–481, 2000. 2

[7] M. Lanuzza, S. Perri, P. Corsonello, and M. Margala, "A new reconfigurable coarse-grain architecture for multimedia applications," in *Adaptive Hardware and Systems, 2007. AHS 2007. 2nd NASA/ESA Conf. on*, 2007, pp. 119–126. 2

[8] I. Uzun, A. Amira, and F. Bensaali, "A reconfigurable coprocessor for high-resolution image filtering in real time," in *Electronics, Circuits and Systems, 2003. ICECS 2003. Proceedings of the 2003 10th IEEE International Conference on*, vol. 1. IEEE, 2003, pp. 192–195. 2

[9] T. von Sydow, B. Neumann, H. Blume, and T. G. Noll, "Quantitative analysis of embedded fpga-architectures for arithmetic," in *Application-specific Systems, Architectures and Processors, 2006. Int. Conf.*, 2006, pp. 125–131. 2

[10] X. Wang and S. G. Ziavras, "Hera: A reconfigurable and mixed-mode parallel computing engine on platform fpgas," in *16th International Conference on Parallel and Distributed Computing and Systems (PDCS)*, 2004, pp. 374–379. 2

[11] A. Prengler and K. Adi, "A reconfigurable simd-mimd processor architecture for embedded vision processing applications," SAE Technical Paper, Tech. Rep., 2009. 2

[12] C.-Y. Hung, L. W. Estevez, and W. A. Rabadi, "Multiplexer reconfigurable image processing peripheral having for loop control," Mar. 4 2003, uS Patent 6,530,010. 2

[13] M. Bordallo López, H. Nykänen, J. Hannuksela, O. Silvén, and M. Vehviläinen, "Accelerating image recognition on mobile devices using gpgpu." in *Proceeding of SPIE Electronic Imaging 2011, 7872*, 2011. 2

[14] Texas-Instruments, "Omap3530 power estimation spreadsheet," Tech. Rep., 2011. 2

[15] O. Almer, R. Bennett, I. Böhm, A. Murray, X. Qu, M. Zuluaga, B. Franke, and N. Topham, "An end-to-end design flow for automated instruction set extension and complex instruction selection based on gcc," in *Proceedings of the First International Workshop on GCC Research Opportunities (GROW'09)*, 2009, pp. 49–60. 2, 3

[16] PASTA project, University of Edinburgh., http://groups.inf.ed.ac.uk/pasta/, 2014. 2

[17] Arc International., http://www.arc.com, 2014. 2

[18] M. Zuluaga and N. Topham, "Resource sharing in custom instruction set extensions," in *Application Specific Processors, 2008. SASP 2008. Symposium on*. IEEE, 2008, pp. 7–13. 3

[19] A. Nieto, D. Vilariño, and V. Brea, "SIMD/MIMD dynamically-reconfigurable architecture for high-performance embedded vision systems," in *23rd IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP 2012)*, July 2012. 5

[20] M. Bordallo López, A. Nieto, J. Boutellier, J. Hannuksela, and O. Silvén, "Evaluation of lbp computing in multiple architectures," *Journal of Real-Time Image Processing*, pp. 1–34, 2014. 5, 6, 7

[21] Xilinx Inc., http://www.xilinx.com, 2014. 5

[22] I. Kuon and J. Rose, "Measuring the gap between fpgas and asics," *Computer-Aided Design of Integrated Circuits and Systems, IEEE T.*, vol. 26, no. 2, pp. 203–215, 2007. 5, 7

[23] D. G. Chinnery and K. Keutzer, "Closing the power gap between asic and custom: an asic perspective," in *Proceedings of the 42nd annual Design Automation Conference*. ACM, 2005, pp. 275–280. 5, 7

[24] H. Corporaal, *Microprocessor Architectures : From VLIW to TTA*. John Wiley & Sons, Dec. 1997. 6

[25] J. Boutellier, I. Lundbom, J. Janhunen, J. Ylimäinen, and J. Hannuksela, "Application-specific instruction processor for extracting local binary patterns," in *Conf. on Design and Architectures for Signal and Image Processing*, 2012. 6

[26] T. Pitkänen, R. Mäkinen, J. Heikkinen, T. Partanen, and J. Takala, "Low-power, high-performance tta processor for 1024-point fast fourier transform," in *Embedded Computer Systems: Architectures, Modeling, and Simulation*. Springer, 2006, pp. 227–236. 6

[27] O. Esko, P. Jääskeläinen, P. Huerta, C. S. de La Lama, J. Takala, and J. I. Martinez, "Customized exposed datapath soft-core design flow with compiler support," in *20th International Conference on Field Programmable Logic and Applications*, Milano, Italy, 2010, pp. 217–222. 6

[28] M. Humenberger, C. Zinner, M. Weber, W. Kubinger, and M. Vincze, "A fast stereo matching algorithm suitable for embedded real-time systems," *Computer Vision and Image Understanding*, vol. 114, no. 11, pp. 1180–1202, 2010. 7

[29] T. Patyk, D. Guevorkian, T. Pitkanen, P. Jaaskelainen, and J. Takala, "Low-power application-specific fft processor for lte applications," in *Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIII), 2013 International Conference on*. IEEE, 2013, pp. 28–32. 7

[30] M. Bordallo López, J. Hannuksela, O. Silven, and M. Vehvilainen, "Interactive multi-frame reconstruction for mobile devices," *Multimedia Tools and Applications*, pp. 1–21, 2012. 7

[31] R. Tessier and W. Burleson, "Reconfigurable computing for digital signal processing: A survey," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 28, no. 1-2, pp. 7–27, 2001. 7