Morphological Co-Processing Unit for Embedded Devices

Jan Bartovsky, Petr Dokládal, Matthieu Faessel, Eva Dokladalova, Michel Bilodeau

To cite this version:

Jan Bartovsky, Petr Dokládal, Matthieu Faessel, Eva Dokladalova, Michel Bilodeau. Morphological Co-Processing Unit for Embedded Devices. Journal of Real-Time Image Processing, Springer Verlag, 2015, pp. 1-12. 10.1007/s11554-015-0518-2 . hal-01117406v2

HAL Id: hal-01117406

https://hal-mines-paristech.archives-ouvertes.fr/hal-01117406v2

Submitted on 25 Jun 2015

A preliminary (accepted) version of an original paper to appear in **Journal of Real-Time Image Processing**
(The final publication is available at Springer via http://dx.doi.org/10.1007/s11554-015-0518-2)

Jan Bartovský · Petr Dokládal · Matthieu Faessel · Eva Dokládalová · Michel Bilodeau

Morphological Co-Processing Unit for Embedded Devices

June 25, 2015

Abstract This paper focuses on the development of a fully programmable morphological coprocessor for embedded devices. It is a well-known fact that the majority of morphological processing operations are composed of a (potentially large) number of sequential elementary operators. At the same time, the industrial context induces a high demand on robustness and decision liability that makes the application even more demanding. Recent stationary platforms (PC, GPU, clusters) no more represent a computational bottleneck in real-time vision or image processing applications. However, in embedded solutions such applications still hit computational limits.

The Morphological Co-Processing Unit (MCPU) replies to this demand. It assembles the previously published efficient dilation/erosion units with geodesic units and ALUs to support a larger collection of morphological operations, from a simple dilation to a serial filters involving a geodesic reconstruction step.

The coprocessor has been integrated into an FPGA platform running a server, able to respond client's requests over the ethernet. The experimental performance of the MCPU measured on a wide set of operations brings as results in orders of magnitude better than another embedded platform, built around an ARM A9 quad-core processor.

J. Bartovský
Centre for Mathematical Morphology, MINES ParisTech, Fontainebleau, France.
Faculty of Electrical Engineering, University of West Bohemia, Pilsen, Czech Republic.
E-mail: jan.bartovsky@mines-paristech.fr

P. Dokládal, M. Faessel and M. Bilodeau
Centre for Mathematical Morphology, MINES ParisTech, Fontainebleau, France.
E-mail: {matthieu.faessel, petr.dokladal, michel.bilodeau}@mines-paritech.fr

E. Dokládalová
Computer Science Laboratory Gaspard Monge, ESIEE Paris, University Paris-Est, Noisy-le-Grand, France.
E-mail: e.dokladalova@esiee.fr

Keywords Mathematical Morphology, Hardware Implementation, Pattern Spectrum, Reconstruction, Parallel Computation

1 Introduction

Mathematical morphology is an image processing framework providing a complete set of tools for filtering, multiscale image analysis, or pattern recognition. It is used in a number of applications, including biomedical and medical imaging, video surveillance, industrial control, video compression, stereology or remote sensing since its very first appearance in the late 1960's, see [21, 27–29].

Considering the hardware implementation context, several different trends have been observed. A recent technological advance of imaging sensors stimulated the development of applications by means of high-resolution images that became a standard. Needless to say large images impose challenging requirements on the computation platform in terms of both performance and memory.

On the other hand, the industrial context often induces severe real-time constraints on applications. Often these demanding image-interpretation applications require a high correct-decision liability, robust but costly multi-criteria and/or multi-scale analyses are used. Given that image processing should not deteriorate industrial productivity, the latency and computational performance are of high interest in this context.

In embedded systems, the most important concerns are low power consumption (and consequently low heat dissipation) and small resources occupation, which allows for better embedding. All these considerations combined together infer overwhelming requirements on the architecture of polyvalent processing units addressing many different contexts. The context of embedded morphology applications includes, for instance, an augmented vision system that improves visual perception [12], or smart cameras [15].

This paper stems from a previous work [1, 2, 9] and shows how to build around the computation pipelines an advanced, polyvalent and user-friendly computation platform for embedded or portable applications. We have enriched the set of supported operators by adding geodesic units, ALUs, a fast DDR2 memory controller and a fast ethernet link to transfer images. The computation is controlled and monitored by a Xilinx MicroBlaze microcontroller that also ensures the communication with the outside world. The entire platform behaves as a server and responds to client's computation requests. A provided software interface (in python and C/C++) offers the user a convenient possibility to interact with the platform.

The text is organized as follows: Section 2 makes a short survey of existing morphological algorithms and architectures. Section 3 outlines the basic definitions of typical mathematical morphology operations. Section 4 describes how these operations can be efficiently computed by processing pipelines and describes the architecture of the proposed coprocessor. The following Section 5 covers the programmability and user interface to the coprocessor server. Finally, Section 8 presents experimental results obtained on an FPGA board and compares them to an ARM A9 embedded platform.

2 State of the art

This section briefly presents the state of the art algorithms for elementary morphology operations dilation and erosion and their hardware implementations in FPGA. The last part discusses the novelty and main contributions of this paper.

2.1 Algorithms

The simplest method to compute a dilation is the exhaustive search for maximum in the scope of a structuring element (SE) B according to the definition Eq. 1 in Section 3. This naive solution tends to need a large number of comparisons, which are on most platforms diadic (with two operands). The number of comparisons is considered as a metric of algorithm complexity, so the naive algorithm has complexity $\mathcal{O}(l)$ as it has to carry out $l-1$ comparisons for a SE containing l pixels. Such complexity suggests that the naive algorithm is inefficient for large SEs. Pecht [25] proposed a method to decrease the complexity based on the logarithmic SE decomposition, thereby achieving $\mathcal{O}(\lceil \log_2(l) \rceil)$ complexity.

The first 1-D algorithm that reduced the complexity to a constant is by van Herk [34], and Gil and Werman [14] (published simultaneously in two papers and often referred to by the authors' initials as HGW). The computation complexity is constant, i.e., of $\mathcal{O}(1)$, which means the upper bound of the computation time is independent of the SE size. Gil [13] proposed an improved version of HGW that lowered the number of comparisons per element, but at the cost of increased memory usage and implementation complexity.

Lemire [18] proposed a fast stream algorithm of $\mathcal{O}(1)$ for causal line SEs. This algorithm uses two queues of length l in order to store the pixels that form locally monotonous signal (i.e., monotonously increasing and decreasing). Although it produces both erosion and dilation simultaneously, it works with causal SEs only. This downside was solved later by Dokládal [9] who proposed another queue-based algorithm. The advantages of these queue-based algorithms are strictly sequential access to data, zero latency, and low memory requirements.

The 2-D dilation is usually obtained by composition of 1-D dilations, see for instance Soille [31] who approximates circle and polygon SEs using rotated line SEs. However, this technique covers only a limited family of shapes. The arbitrary-shaped SE are obtained by either more complex 2-D algorithms (e.g., Urbach [33]), which are suitable for general-purpose processors, or by fine-grained decomposition of the large SE into a set of small 2-D SEs. Xu [41] proposed that any 8-convex polygon (convex on 8-connectivity grid, hence 8-convex) is decomposable into a class of 13 nontrivial indecomposable convex polygonal SEs. Normand [23] reduced the class of shapes to only four 2-pixel SEs by allowing the union operator to take place in the SE decomposition.

2.2 Hardware implementations

One of the first morphology architectures was the texture analyzer by Klein [17]. It was optimized for linear and rectangular SE by decomposition into line segments. More recently, Velten [35] proposed another, delay-line based architecture for binary images supporting arbitrarily shaped 3×3 SEs. The computation of dilation is realized by OR gates (topology was not communicated, probably a tree of diadic OR gates) achieving good performance, which was further improved by spatial parallelism.

Clienti [4] proposed a highly parallel morphological System-on-Chip. It is a set of neighborhood processors optimized for arbitrarily shaped 3×3 SE interconnected in a partially configurable pipeline. Each stage of the pipeline contains 2 processors that can process 2 parallel image streams and an ALU. The reconfiguration allows all the processors to be connected in one chain in order to employ all processors when only one image stream is used. A reconfigurable 3x3 neighborhood morpho processor was recently used in Gibson [12] in a hand-held augmented-vision system for visually impaired.

Another approach is called partial-result reuse (PRR). The morphological operation by some neighborhood B_1 in an early stage is delayed by delay lines in order to be reused later in computation by some other neighborhood B_2 obtaining larger B_3 decreasing thus the number of necessary comparisons. One of the first PRR architectures for 1-D dilation was proposed in [26] and improved in [6]. The principle is based on an exponential growth of the intermediate neighborhoods in the partial-result reuse scheme.

Chien [3] presented more general concept of PRR that builds the desired SE by a set of distinct partial neighborhoods computed by a dedicated algorithm. As a result, it supports arbitrary 8-convex polygon at the cost of some additional comparisons.

A similar approach is based on the Normand [23] decomposition where a convex SE is decomposed into a number of causal 2-pixel SEs, applied in sequence or in parallel. Déforges *et al.* [8] make use of this decomposition, which combined with a stream implementation, allows to conceive a pipeline architecture supporting arbitrary convex SEs.

Recently, Torres-Huitzil [32] designed a linear systolic-like array of processing elements without need for delay-line internal memory storage supporting non-rectangular flat SEs. However, prospective drawbacks can be seen in the chosen column-based image scan requiring significant image storage capability, and the need of deep parallelism to attain real-time performance even for the mentioned 7×7 SE.

The last method mentioned in this overview is the implementation of efficient 1-D algorithms. To our knowledge, there are only few such contributions in the literature. Clienti [5] published architectures for the 1-D Lemonier algorithm [19] and the HGW algorithm [14,34]. Both algorithms require a reverse scan which increases the memory requirements. Regarding the HGW algorithm the line can be reversed per blocks [5], which significantly decreases the latency.

Recently, Bartovský [1] proposed an implementation of the Dokládal algorithm [9] as a processing unit by polygonal SEs with a strictly sequential access to data and small memory requirements. This architecture is mainly beneficial for large SEs because only the memory varies with the size of the SE, the computation logic remains the same.

3 Basic notions

This section describes the operators used in this paper. We are mainly interested in compound operators composed as concatenations of elementary operators, that are therefore costly on sequential machines.

Let δ_B, ε_B: $\mathbb{Z}^2 \rightarrow \mathbb{R}$ be a dilation and an erosion on gray-scale images, parameterized by a structuring element B, assumed to be flat (i.e., $B \subset \mathbb{Z}^2$) and translation-invariant, defined as [27,30]

$$\delta_B(f) = \bigvee_{b \in B} f_b \; ; \qquad \varepsilon_B(f) = \bigwedge_{b \in \widehat{B}} f_b \tag{1}$$

where f_b denotes translation of f by b. The hat $\widehat{\ }$ denotes the transposition of B, equal to the set reflection $\widehat{B} = \{x \mid -x \in B\}$.

For an image f, its internal, external and morphological gradients are given by

$$g_i(f) = f - \varepsilon_B(f) \tag{2}$$

$$g_e(f) = \delta_B(f) - f \tag{3}$$

$$g(f) = \delta_B(f) - \varepsilon_B(f) \tag{4}$$

The concatenation of dilation and erosion forms other morphological operators. The closing and opening on gray-scale images, φ_B, γ_B: $\mathbb{Z}^2 \rightarrow \mathbb{R}$, parameterized by a structuring element B, are defined as

$$\varphi_B(f) = \varepsilon_B[\delta_B(f)] \; ; \qquad \gamma_B(f) = \delta_B[\varepsilon_B(f)] \tag{5}$$

The residue of an opening or closing is called top-hat,

$$th^\gamma(f) = f - \gamma_B(f) \; ; \qquad th^\varphi(f) = \varphi_B(f) - f \tag{6}$$

and is used to detect objects filtered by the opening or the closing.

Closing and opening are filters. Their concatenation forms alternating filters $\gamma\varphi$, $\varphi\gamma$, $\gamma\varphi\gamma$ and $\varphi\gamma\varphi$. Other filters can be obtained by combining *families* of filters. A well known example is the alternating sequential filter (ASF), composed as sequence of closings and openings with a progressively increasing SE λB, with $\lambda > 0$. Let γ^λ and φ^λ denote the change of scale such as $\gamma_{\lambda B}$ and $\varphi_{\lambda B}$. Then λ-order ASF (referred to as ASF$^\lambda$) is composed as

$$\mathrm{ASF}^\lambda = \varphi^\lambda \gamma^\lambda \varphi^{\lambda-1} \gamma^{\lambda-1} \ldots \varphi^1 \gamma^1 \tag{7}$$

starting with opening, and

$$\mathrm{ASF}^\lambda = \gamma^\lambda \varphi^\lambda \gamma^{\lambda-1} \varphi^{\lambda-1} \ldots \gamma^1 \varphi^1 \tag{8}$$

starting with closing.

Let $\delta_f : \mathbb{Z}^2 \rightarrow \mathbb{R}$ be an elementary geodesic dilation of image g (marker) "under" image f (mask) where $g \leq f$, such as [36]

$$\delta_f(g) = f \wedge \delta_{3\times3}(g) \tag{9}$$

Repeating $\delta_f(f)$ until stability represents the dilation-reconstruction of g under f; $g \leq f$,

$$\rho_f(g) = \underbrace{\delta_f \delta_f \ldots \delta_f}_{x \text{ times}}(g) \tag{10}$$

the number of iterations $x = \infty$ by definition, and practically until the idempotence. The marker image g is commonly obtained by morphological opening γ_B. In this case, the operation is called opening by reconstruction γ_B^ρ, defined as

$$\gamma_B^\rho(f) = \rho_f(\gamma_B(f)) \tag{11}$$

Let $\{\gamma^{\lambda_i}\}$, with $\lambda_i > \lambda_{i-1}$ and with $\lambda_i > 0, \forall i$ be a collection of openings, generating a size distribution aka granulometric function (see Matherons' axioms, [21] p. 192) using some measure, e.g. integral (or sum) of the image. One also often uses its derivative, so called granulometric or pattern spectrum, defined as

$$PS_{\lambda_j B}(f) = \sum_D \left(\gamma_{\lambda_i B} f - \gamma_{\lambda_j B} f\right) \tag{12}$$

with $D = \mathrm{spt}(f)$. Notice, that instead of opening $\gamma_{\lambda_i B}$ one also may want to use the opening by reconstruction $\gamma_{\lambda_i B}^\rho$ which even more increases the computation cost.

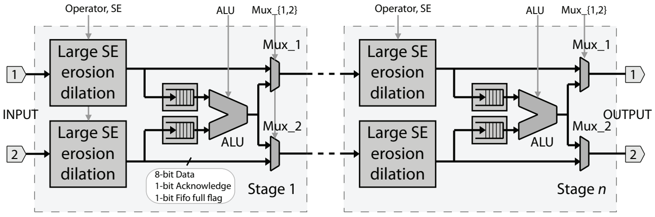


Fig. 1 Large SE pipeline architecture.

4 Hardware architecture

This section describes the hardware architecture of the proposed morphological coprocessor that efficiently implements the aforementioned sequential, costly operators. The following description follows the bottom-up approach, so we start with developing two basic image processing pipelines, one for large SE operators, and one for geodesic operators. Then we build the processing core by surrounding these two pipelines with interconnection busses, configuration registers, and image buffers in such way that it can be used as a peripheral of the Xilinx MicroBlaze. Finally, we describe the top-level architecture of the FPGA evaluation platform.

4.1 Large SE Pipeline

Let us begin with the description of the Large SE pipeline.

One of the most penalizing aspects in morphological operators is the number of iteration on an image. On sequential platforms, this induces an intensive traffic between the CPU and the memory. A considerable increase in efficiency can be obtained when such iterations can be pipelined. This idea is used in Clienti [5]. However, given that its pipeline is only composed of 3×3 blocks it lacks flexibility and must be reconfigured to precisely fit application needs.

In this paper, in order to gain in flexibility we propose two parallel pipelines of programmable large-size SE units. To efficiently execute the largest collection of operators (including those with two parallel branches, such as top hat or gradient) the pipelines are interconnected by programmable ALUs (Arithmetic Logic Unit), see Fig. 1.

Both pipelines contain several identical parts called processing stages connected one after the other. The pipeline is scalable by means of the number of instantiated stages, which is hereafter denoted as n. The heart of each stage is a pair of Large SE erosion/dilation units. The output of both units can be connected to the ALU, or directly to the next stage through the Mux_$\{1,2\}$ multiplexer. The ALU result can be routed to either input port of the next stage (or both). This stream routing capability increases the adaptivity of the architecture to execute different algorithms. See below (Fig. 4), a few examples of possible interconnections.

The Large SE erosion/dilation unit performs one morphological dilation or erosion by a flat rectangular or octagonal SE of programmable size (up to 31 pixels in diameter for rectangles and 43 pixels in diameter for octagons) and the position of the origin. This unit takes advantage of separability of 2-D rectangular and octagonal SEs into a sequence of 1-D SEs. The separability allows that the computation can be separated into rows and columns for rectangles, and four obliquous (rotated by 45^o) lines for octagons. This simplifies the memory manegement since the data can be stored in small-size independent memory blocks. These blocks behave as queues if the 1-D dilation is computed by a queue-based algorithm [9]. A maximum allowable size of the SE needs to be specified prior to the synthesis. Afterwards, the size of the SE is freely programmable by the user within the range of the synthesize maximum size. In the same way, the size of the image is programmable withing the maximum specified prior to the synthesis. Such implementation has proven to be flexible, and beneficial for high-demanding image processing with large SEs. The description of the FPGA architecture and experimental results can be found in [2] for rectangular SEs, and [1] for octagonal SEs, respectively.

The previously published results can be summarized as follows. The Large SE unit computes 2-D rectangular or octagonal erosion/dilation during a single horizontal image scan with minimal latency. The experimentally obtained average processing rate is approximately 2.5 clock cycles per pixel, i.e., approx. 50 Mpx/s at 125 MHz clock frequency. The memory requirement is another important parameter of an image processing implementation because it limits the number of units that fit in the FPGA. The most significant memory requirement of the Large SE unit is given by the set of queues, such as

$$R = NH \times (bpp + \lceil \log_2(H-1) \rceil) \;\; \text{[bits]} \qquad (13)$$

where N denotes the image width, H the height of the SE, bpp the number of bits per pixel, and $\lceil . \rceil$ the ceiling operation. For example, let $N = 1024$ px, $H = 31$ px, and $bpp = 8$ bits. The memory requirement is then

$$R = 31744 \times 13 \;\; \text{[bits]} \qquad (14)$$

The memory occupation of the pipeline is then linear factor of the memory occupation of one stage and the length of the pipeline.

The ALU performs simple dyadic, arithmetic operations on two inputs. Each input can be connected to either an image stream or a programmable constant. The supported operations are as follows: no operation; negation (logical complement); bit-wise AND, OR, XOR; saturated addition, subtraction; infimum and supremum.

In order to ensure that the ALU has both input pixels at the same coordinates in respective images, both image streams have to be synchronized. The synchronization is done at two levels.

1. *Algorithm level*: Consider, e.g. the inner or outer morphological gradient Eqs. 2,3, with B a 3×3 window, or the opening/closing residues Eqs. 6. Computing the dilation introduces a delay in $\delta_B(f)$ of one row plus one column (distance from the center of B to the lower-right

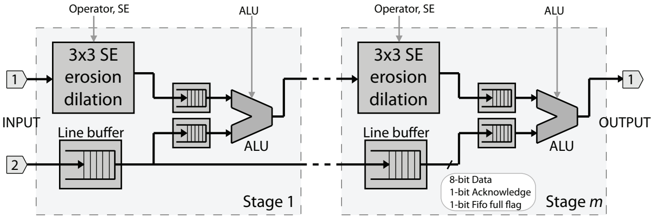


Fig. 2 Geodesic pipeline architecture.

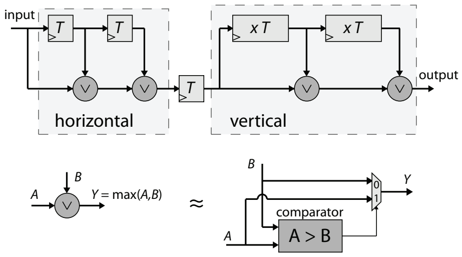


Fig. 3 3×3 dilation unit for the geodesic pipeline.

pixel). This stream f must be delayed accordingly so that the ALU can compute the difference. This is done by turning off the dilation/erosion in the other Large SE block. The computation memory changes into a large FIFO buffer used to delay the stream.

2. *Implementation level*: The previous delay increases with the size of the SE B. Moreover, the dilation/erosion algorithms are asynchronous, see [9] for details. That means that that the delay of the Large SE block can (up to a few pixels) vary. This small delay is compensated by the FIFO memories. In the case that either FIFO is temporarily empty, the empy FIFO stalls the ALU until data become available again.

This synchronization mechanism is automatic, and remains hidden to the user.

The Measurement unit computes simple metrics of the whole image, namely the sum, infimum, and supremum. This measurement is useful in image analysis applications, such as pattern spectrum, and can be obtained on-the-fly at a low cost. The measurement results can be read out through the configuration registers.

All programmable parameters including the SE dimensions, operation, multiplexers and ALU settings, as well as measurement results, are stored in a bank of per-stage configuration registers.

4.2 Geodesic pipeline

A significant subset of morphological operators relies on the reconstruction using the geodesic dilation/erosion by 3×3 SE. Even though the Large SE pipeline supports geodesic operations, using it would be inefficient. A better solution is to devise a dedicated Geodesic pipeline, see Fig. 2.

This Geodesic pipeline contains several equal stages connected one after another. The pipeline is scalable by means of the number of stages instantiated, which is hereafter denoted as m. The heart of each stage is a 3×3 erosion/dilation unit. The output of this unit is connected to the ALU, along with the buffered Mask image. The ALU result is the Marker input of the next stage.

We have used a m=16-stage pipeline in our application. The length of the pipeline is a trade-off between the memory occupation, and the number of iterations before idempotence of a geodesic operation. Given that, geodesic operations require content-dependent number of iterations it is beneficial to have longer pipelines to reduce the number of iterations through the memory.

The 3×3 dilation is outlined in Fig. 3. It also takes advantage of separability of the rectangle into the horizontal and vertical segments, which are implemented using a well-known approach of delay elements (registers T for the horizontal segment, and line buffers xT for the vertical segments) and comparators. The T registers provide a unitary delay, whereas the xT line buffers provide one-image-line delay each. The xT line buffers are parametrized by the the image width to adapt automatically to the image size.

Notice that the delay-line dilation is better adapted for small (3×3 SEs) because it has a smaller area occupation and it is synchronnous. Conversely, the queue-based dilation architecture offers better flexibility due to the programmable SE size and is therefore used in Large SE blocks.

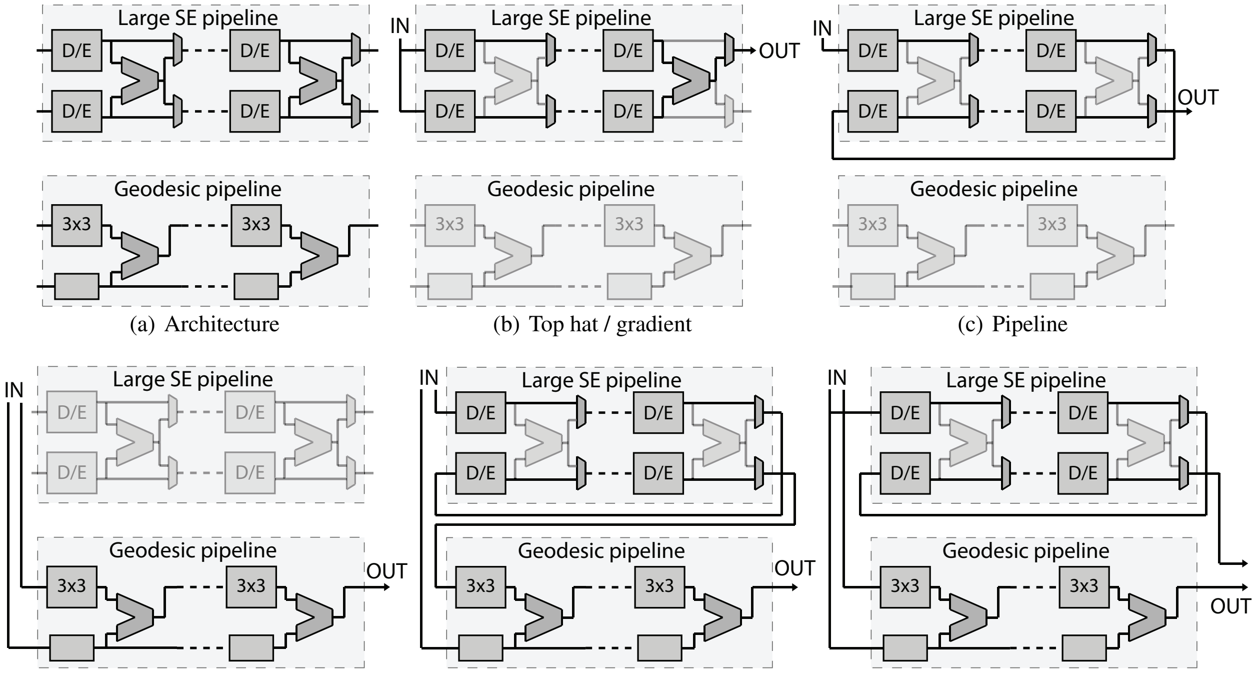
The ALU is the same as described above. The reason for the Line buffer is to synchronize the Mask image and the dilated Marker image, which is delayed by N+1 pixels (recall N is the width of the image).

4.3 Pipeline configurations

The interconnection pattern of the pipelines is thought for maximum polyvalence. The architecture Fig. 4(a), with one, twin Large SE pipeline, and one geodesic pipeline, can be programmed into one of the following patterns (b-f) using global and pipeline multiplexers.

The configuration Fig. 4(b) can be used to compute elementary operators such as the gradient (Eqs. 2-4), an opening, closing (Eqs. 5) or its residues (Eqs. 6). The geodesic pipeline, unused here, appears in grey. The example in Fig. 4(c) connects all the Large SE units into one, long pipeline, suitable for long concatenations of erosions and dilations as in sequential filters (Eqs. 7, 8) or granulometries (Eq. 12). A morphological reconstruction (Eq. 10) is implemented by the geodesic pipeline Fig. 4(d). With both pipelines active and concatenated, as in Fig. 4(e), one can compute openings/closings by reconstruction Eq. 11. The same configuration can also be used in granulometry when the opening by reconstruction Eq. 11 is used Eq. 12.

Such variety of operators allows to fully appreciate the potential of this platform offering a long and flexible pipeline to process data with limited accesses to the memory. Finally, the two pipelines can also be used independently and in parallel on the same input data as in Fig. 4(f).



(a) Architecture (b) Top hat / gradient (c) Pipeline

(d) Reconstruction (e) Opening by reconstruction (f) User ad-hoc

Fig. 4 (a) The interconenction architecture of the pipelines, and (b-f) various possible interconnection patterns of the Large SE and Geodesic pipelines.

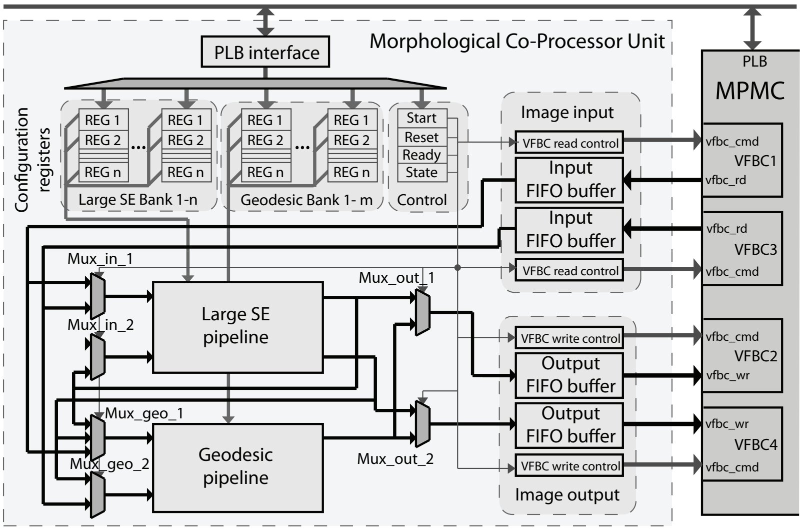


Fig. 5 Architecture of the Morphological Co-Processor Unit. (Legend: Black line denotes an image data bus; grey line denotes configuration and control; MPMC=Multi-Port Memory Controller; VFBC=Video Frame Buffer Controller; PLB=Peripheral Local Bus)

4.4 Morphological Co-Processor Unit (MCPU) Schematics

The pipelines are integrated into the MCPU, supposed to ensure their correct operation, using a set of multiplexers, configuration registers, and image buffers and a memory controller, see Fig. 5.

The configuration registers store the necessary configuration for all the processing units in both pipelines, the global control, and measurement results. Notice that there is one bank of registers for each stage of the processing pipelines. All the registers are accessible to any PLB master by simple read and write instructions.

Image data are transferred by four Video Frame Buffer Controller (VFBC) channels [38]. Two VFBC channels are dedicated to reading input image data from the DDR2 memory, and other two for writing the output image data to the DDR2 memory. The image transfers are independent of each other, so the processing can run in-place (the output image

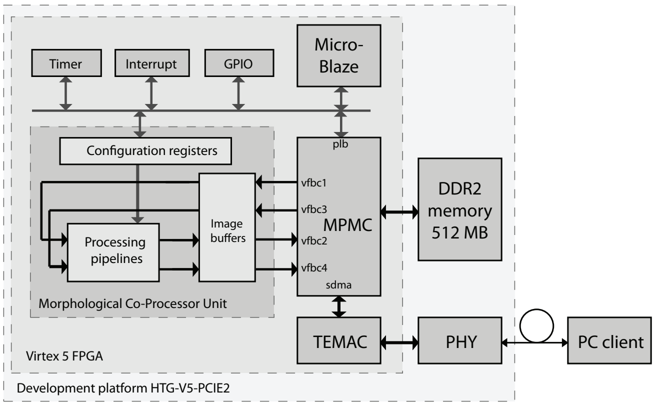


Fig. 6 Overall Platform architecture. Black line denotes image data transfers, grey line denotes configuration and control.

is written in place of the input image). The VFBC allows us to read and/or write image data to the DDR memory with a FIFO-like data-flow control (full, almost full, empty, almost empty flags), so the data stream can be stalled by either endpoint if necessary. The image data in both directions are buffered in Input or Output buffers, respectively.

The MCPU is intended to be used as a peripheral in a higher-level environment. We have tested it as a peripheral of the Xilinx MicroBlaze as described in the following section.

4.5 Top-level architecture

The architecture we have built for evaluation purposes, is outlined in Fig. 6. The proposed MCPU is a coprocessor running as a peripheral of the MicroBlaze CPU synthesized on a Virtex 5 development platform. This platform is also provided with an ethernet link. The architecture consists of two main parts: (i) the MCPU core, and (ii) the MicroBlaze processor environment.

This platform plays several roles: i) the MicroBlaze configures and sends operators to MCPU to execute, ii) provides working memory storage capacity, and iii) handles the communication with the outside world. The images are received via the enternet link and stored in memory.

A very important aspect of every image processing platform is the memory for storing images; either input, output, or intermediate result. The MCPU uses the Xilinx proprietary Multi-Port Memory Controller MPMC [40] that provides a multi-port interface to a high-capacity off-chip DDR2 memory. The MPMC is capable of handling 4 simultaneous image data streams of approx. 50 Mpx/s each that are required by the processing core to sustain the maximal performance.

The MicroBlaze processor uses the Peripheral Local Bus (PLB) to control all the peripherals and to transfer the configuration data, which are small in size, among the peripherals. We used the MicroBlaze version 7.30.b running at 125 MHz with 2048 Bytes of instruction cache and 2048 Bytes of data cache.

5 User interface - programmability

The platform can be accessed via a tri-speed Ethernet interface using either TCP/IP or UDP/IP protocols (implemented as lightweight lwIp). The MCPU runs a server able to accept images and operations to execute via the ethernet link from a superior client.

In order to achieve a proper function of the MCPU server, the client has to perform the following tasks: communicate properly with MicroBlaze, send the image data, configure and run the processing core, read the results.

We have provided a software interface (integrated in MorphM [22]) to handle these tasks. This interface is available at two levels: i) a low-level interface in C/C++, and ii) a high-level interface in C/C++ and Python.

The low-level interface gives the user the possibility to directly control all the features of the computing blocks, such as the sizes of the 1-D segments composing the SE, handle delays, synchronization, multiplexors, turn on/off individual computing units, allocate images in the memory, start and monitor a computation, wait for the end, etc.

The high-level interface offers the user a more intuitive way to execute the most frequent morphological operators. It contains a set of macro functions that hide the low-level programming burden and provide the user the possibility to use the platform on a higher level of abstraction. For example, it can automatically map an arbitrarily long ASF or granulometry to the pipelines. This involves programminc correct SE sizes at every stage, chain the Large SE units in a stream, store the intermediate result in the memory whereever needed, and control the execution and monitor its end.

The MCPU is intended to be as polyvalent as possible, able to perform various operators with different computation schemes. The list of supported operators includes dilation, erosion, opening, closing, reconstruction, opening and closing by reconstruction, top hat, gradient, ASF$^{\lambda}$, pattern spectrum, and pattern spectrum by reconstruction. All these operators proceed in the following steps: send the image (if necessary), calculate the configuration based on the passed arguments (either SE or λ), execute the processing pipelines, and read the output image and/or measurement results. Notice that operators with large values of SE or λ may not fit into the pipelines. In this case, several iteration are automatically executed. This is especially true for the reconstruction that may need hundreds of iterations.

Only a few features of the MCPU are set prior to the synthesis, such as the width of the buses or the number of stages in the pipelines. These features remain immuable afterwards. All other parameters are programmable via the configuration registers.

6 Application Example

In this section, we illustrate how MCPU can be used in a real image processing application.

Consider detecting defects in manufactered surfaces, where the defects appear as changes in local textural patterns. Cord *et al.* [7] follow a probabilistic approach, considering textural variations as realizations of random functions. Taking into account information of pixel neighbourhoods, the texture for each pixel is described at different scales. By means of statistical learning, the most relevant textural descriptors are selected for each application.

The preparatory steps of this technique consist of the following (refer to [7] for details and references):

1. A collection of morphological descriptors is chosen, such as dilations, erosions, openings and closings, each for various SE shapes, such as segments oriented in 0°, 45°, 60°, 90°, 120° and 135°, squares and circles and each for different sizes. Each pixel is assigned a vector with as elements the values from these descriptors.
2. A dimension reduction in a set of independent variables by Principal Component Analysis.
3. A supervised learning using Linear Discriminant Analysis.
4. A variable selection based on forward selection.

Done in this way, the approach is generic, and the selected descriptors application specific. After validation, we know at this stage which descriptors apply best to separate the classes – defect/no defect.

The two-step on line classification application shall execute on the MCPU.

1. The vector of values for all selected sizes of a descriptor is a local pattern spectrum with this SE. The descriptors for every SE shape and size are efficiently computed MCPU and the descriptors stored in the DDR2 memory.
2. The LDA classification is a weighted linear combination of the descriptor values for every pixel which, coded in C, can run on the MicroBlaze controller. The resulting image is then trasferred out via the ethernet link.

7 Experiment setup

The proposed MCPU architecture has been implemented in VHDL and targeted to the Xilinx platform HTG-V5-PCIE2, equipped with one medium-size Xilinx Virtex-5 FPGA chip XC5VSX95T [39]. We report here two different setups with parameters to fully utilize the available FPGA resources and keep high flexibility at the same time. Using the XC5VSX95T – a medium-size Virtex 5 family FPGA, with 14,720 slices and 244 36Kb-size RAM blocks – we could implement 4 to 5 (resp.) Large SE stages and 16 geodesic stages in two pipelines on the chip. The implementation results and setup specification are outlined in Table 1. Notice that the reported “Supported SE per unit” is given by the maximum FIFO length as synthesized on the chip. After the synthesis, the user can freely program his own SE size withing this range.

Table 1 Implementation results, Xilinx Virtex-5 FPGA XC5VSX95T [39]

Parameter	Setup 1	Setup 2
Supported SE per unit	Rectangle 31×31	Octagon 43×43
Large SE stages n	5	4
Geodesic stages m	16	16
Image width N	1024	1000
FPGA Slice	13534	14684
FPGA BRAM	233	243
Clock frequency	125 MHz	125 MHz

Recall that the SE size multiplied by the image width is the determinant factor of the memory occupation per Large SE stage, ref Eq. 13. The memory occupation for the Large SE pipeline is linear factor of its length. The architecture is therefore linearly scalable – factor of the image width, SE size and number of stages (and also bits per pixel) withing the limitation of the FPGA size.

8 Performance Comparison

In this section we compare this architecture with other embedded solutions: software solutions (Sec. 8.1) and existing hardware platforms (Sec. 8.2).

8.1 Software solutions

We compare the performance of the proposed architecture against another embedded solution, an ARM processor. In our case, we use the Sabre platform [11] by Freescale, which can be seen as another example of hand-held platform. The Sabre uses quad-core ARM A9 processor at 1GHz, 1GB of DDR3 memory up to 533MHz, and runs Linux with TCP/IP stack. We have created benchmarks for two image processing libraries: (i) the well-known OpenCV [24], and (ii) highly optimized Smil [20]. For the sake of completeness, we have also included the single-thread results of the OpenCV at desktop PC Intel Xeon E5620, 2.40 GHz, with 24 GB memory, running a Fedora, release 20, linux.

The benchmark in Table 2 includes a set of aforementioned morphological operators on natural gray-scale photos 1000×1000 px. It includes an elementary 3×3 dilation, large opening and opening by reconstruction, alternating sequential filter, pattern spectrum and pattern spectrum by reconstruction. Apart from the 3×3 dilation, the common property of all these operators is the large number of operations, which is even undetermined for the reconstruction, and therefore, a high cost.

The experimental results show that the proposed MCPU architecture delivers performance by orders of magnitude superior to that of the Sabre platform, and even comparable

Table 2 Performance results of selected operators. Image is natural photo 1000×1000 px, time results are in milliseconds (unless seconds are specified).

Operator	Shape of SE	Size of SE or λ	MCPU	OpenCV at Sabre	Smil at Sabre	OpenCV at Xeon
Dilation	Rectangle	3×3	21.9	32.7	8.4	0.58
Opening	Rectangle	151×151	24.3	2450	1083	38.6
Opening	Octagon	151×151	41.9	246 s	2453	2301
Opening by recon.	Rectangle	151×151	544	47.6 s	22.1 s / 2110*	1940
Opening by recon.	Octagon	151×151	512	289 s	21.1 s	4356
ASF	Rectangle	$\lambda = 11$	64.2	4530	1987	57.1
ASF	Octagon	$\lambda = 11$	83.3	77 s	3872	814
Pattern spectrum *PS*	Rectangle	$\lambda = 11$	62.3	2570	1098	53.8
Pattern spectrum *PS*	Octagon	$\lambda = 11$	62.7	21.2 s	1782	249
PS by recon.	Rectangle	$\lambda = 11$	2530	190 s	85.3 s / 18.2 s*	8920
PS by recon.	Octagon	$\lambda = 11$	2410	201 s	81.5 s	8751

note *: The second result is obtained by an algorithm based on hierarchy queues.

Table 3 Comparison of several FPGA and ASIC architectures concerning morphological dilation and erosion. N, M stand for the image width and height of respective architectures.

	Processing unit				Hardware System		Application Example ASF^6		
	Technology	Supported SE	Throughput [Mpx/s]	f_{max} [MHz]	Number of units	Supported image	Image scans	FPS [frame/s]	Latency [px]
Clienti [4]	FPGA	arbitrary 3×3	403	100	16	1024×1024	6	66.7	$5NM + 84N$
Chien [3]	ASIC	disc 5×5	190	200	1	720×480	45	12.2	$44NM + 84N$
Déforges (a) [8]	FPGA	arbitrary 8-convex	50	50	1	512×512	13	14.7	$12NM + 84N$
Déforges (b) [8]	FPGA	arbitrary 8-convex	50	50	13	512×512	1	50	$84N$
This paper	FPGA	regular polygon	195	100	13	1024×1024	1	185	$84N$

with a desktop PC, for all high-cost operations, i.e., all in Table 2 but the 3 × 3 dilation. MCPU outperforms the other platforms (or is at least equivalent) whereever a high number of operators are sequentially applied to the image. Such a significant speed-up is allowed by possibility to thoroughly exploit the inter-operator parallelism via the pipelined computation. This is especially true for the opening and pattern spectrum by reconstruction when $m = 16$ geodesic dilations are computed at the same time. The speed-up becomes less significant for simple operators with small SEs, the performance for 3 × 3 dilation is worse than that of Smil at Sabre. This is due to a much higher clock of the ARM and the Xeon processors (1GHz and 2.40 GHz, respectively). However, the majority of applications of mathematical morphology need a long sequence of operators that take advantage of the proposed parallelism.

Evaluated using the Xilinx Power Estimator tool [37], the total on-chip power consumption is 4.424 W (1.237 W for input/output ports, 1.625 W of dynamic and 1.562 W of static consumption). The consumption of the Sabre platform is ≈3 W during intensive workload [10]. The thermal design power of the Intel Xeon E5620, 2.40 GHz CPU is ≈80 W during intesive workload [16].

8.2 Existing HW solutions

The Table 3 offers a comparison of our architecture with a few others that support flat, non-rectangular SE. The Processing Unit section of this table provides a comparison on the basis on a single 2-D δ/ε unit only. Clienti [4] yields a high throughput for an elementary SE 3×3. The Chien [3] ASIC chip achieves a reasonable throughput with a small 5×5 diamond SE. The Déforges [8] unit supports 8-convex SEs decomposed as a concatenation of elementary 2-pixel SE. (Lower throughput is probably due to a less powerfull device than that of the other implementations.)

For all other architectures (except this paper) the flexibility to control the size and shape of the SE after the synthesis remains unclear. Nonetheless, all architectures can use the homothecy to obtain larger SEs. This requires using a long processing pipeline as in Clienti [4] or Déforge *et al.* [8](b). Iterating over the memory significantly decreases the overall throughput.

The Hardware System section of the table lists the number of units synthesized on the chip. The performance of every architecture for an ASF^6 as example is given in the third section of the table. The FPS decreases drastically as the number of iterations over the image increases, as for Chien [3] or a single unit of Déforges *et al.* [8]. The decrease is less significant for Clienti [4] who uses a 16-unit long pipeline. However, the units are small 3×3, and still 6 images scans are needed.

Only the Déforges *et al.* [8] pipeline and this paper architecture can embed the entire ASF with no or a limited decrease in performance.

9 Conclusions

This paper proposes a novel programmable morphological coprocessor for embedded devices based on FPGA devices. We have integrated previously published efficient dilation/erosion processing units and geodesic units into a MicroBlaze platform, which provides DDR memory storage and Ethernet connectivity, and thus created a very powerful coprocessor that supports a wide range of operators from a simple dilation to the pattern spectrum by reconstruction.

The coprocessor was experimentally evaluated at a Virtex5 development kit and compared to the quad-core ARM9 Sabre platform by Freescale running OpenCV and Smil libraries. The performance results for various compound operators (except the 3×3 dilation) show a significant speed-up of at least one order of magnitude. The results of MCPU do even compare to that of a Xeon desktop workstation.

The future work will be focused on development of a compiler for MCPU that will automatically map a given application to the architecture. The current interface provides a user with a high-level programming interface. In the future, this compiler shall optimize the execution of concurrent operators, branching and simultaneous co-execution of an application on MCPU and the client.

References

1. J. Bartovský, P. Dokládal, E. Dokládalová, M. Bilodeau, and M. Akil. Real-time implementation of morphological filters with polygonal structuring elements. *Journal of Real-Time Image Processing*, 10(1):175–187, 2012.
2. J. Bartovský, P. Dokládal, E. Dokládalová, and V. Georgiev. Parallel implementation of sequential morphological filters. *Journal of Real-Time Image Processing*, 9(2):315–327, 2014.
3. S.-Y. Chien, S.-Y. Ma, and L.-G. Chen. Partial-result-reuse architecture and its design technique for morphological operations with flat structuring elements. *Circuits and Systems for Video Technology, IEEE Transactions on*, 15(9):1156 – 1169, sept. 2005.
4. Ch. Clienti, S. Beucher, and M. Bilodeau. A system on chip dedicated to pipeline neighborhood processing for mathematical morphology. In EURASIP, editor, *EUSIPCO 2008*, Lausanne, August 2008.
5. Ch. Clienti, M. Bilodeau, and S. Beucher. An efficient hardware architecture without line memories for morphological image processing. In *ACIVS '08*, pages 147–156, Berlin, Heidelberg, 2008. Springer-Verlag.
6. D. Coltuc and I. Pitas. On fast running max-min filtering. *IEEE Transactions on Circuits and Systems II*, 44(8):660 –663, aug 1997.
7. A. Cord, F. Bach, and D. Jeulin. Texture classification by statistical learning from morphological image processing: application to metallic surfaces. *J. of Microscopy*, 239:159–166, 2010.
8. O. Déforges, N. Normand, and M. Babel. Fast recursive grayscale morphology operators: from the algorithm to the pipeline architecture. *Journal of Real-Time Image Processing*, pages 1–10, 2010.
9. P. Dokládal and E. Dokládalová. Computationally efficient, one-pass algorithm for morphological filters. *Journal of Visual Communication and Image Representation*, 22(5):411–420, 2011.
10. Freescale. i.MX 6Dual/6Quad Power Consumption Measurement, Rev. 0, 10/2012. http://cache.freescale.com/files/32bit/doc/app_note/AN4509.pdf.
11. Freescale. SABRE reference designs, 2014. http://www.freescale.com/sabre.
12. R. M. Gibson, A. Ahmadinia, S. G. McMeekin, N. C. Strang, and G. Morison. A reconfigurable real-time morphological system for augmented vision. *EURASIP Journal on Advances in Signal Processing*, 2013(1), 2013.
13. J. Gil and R. Kimmel. Efficient dilation, erosion, opening, and closing algorithms. *IEEE Trans. PAMI*, 24(12):1606–1617, 2002.
14. J. Gil and M. Werman. Computing 2-d min, median, and max filters. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(5):504–507, 1993.
15. M. Holzer, F. Schumacher, T. Greiner, and W. Rosenstiel. Optimized hardware architecture of a smart camera with novel cyclic image line storage structures for morphological raster scan image processing. In *Emerging Signal Processing Applications (ESPA), 2012 IEEE International Conference on*, pages 83–86, Jan 2012.
16. Intel. Intel Xeon Processor E5620 (12M Cache, 2.40 GHz, 5.86 GT/s Intel QPI). http://ark.intel.com/products/47925/Intel-Xeon-Processor-E5620-12M-Cache-2_40-GHz-5_86-GTs-Intel-QPI.
17. J.-C. Klein and J. Serra. The texture analyser. *J. of Microscopy*, 95:349–356, 1972.
18. D. Lemire. Streaming maximum-minimum filter using no more than three comparisons per element. *CoRR*, abs/cs/0610046, 2006.
19. F. Lemonnier and J.-C. Klein. Fast dilation by large 1D structuring elements. In *Proc. Int. Workshop Nonlinear Signal and Img. Proc.*, pages 479–482, Greece, Jun. 1995.
20. Faessel M. Smil simple morphological image library. http://smil.cmm.mines-paristech.fr, 2014.
21. G. Matheron. *Random sets and integral geometry*. Wiley New York, 1975.
22. Morph-M. Morph-M documentation. http://cmm.ensmp.fr/Morph-M, 2012.
23. N. Normand. Convex structuring element decomposition for single scan binary mathematical morphology. In *Discrete Geometry for Computer Imagery*, volume 2886 of *LNCS*, pages 154–163. Springer Berlin, Heidelberg, 2003.
24. OpenCV. OpenCV documentation. http://opencv.org, 2014.
25. J Pecht. Speeding-up successive minkowski operations with bit-plane computers. *Pattern Recognition Letters*, 3(2):113 – 117, 1985.
26. I. Pitas. Fast algorithms for running ordering and max/min calculation. *Circuits and Systems, IEEE Transactions on*, 36(6):795 –804, June 1989.
27. J. Serra. *Image Analysis and Mathematical Morphology*, volume 1. Academic Press, New York, 1982.
28. J. Serra. *Image Analysis and Mathematical Morphology, Volume 2, Theoretical Advances*. Academic Press, London, 1988.
29. J. Serra and L. Vincent. An overview of morphological filtering. *Circuits Syst. Signal Process.*, 11(1):47–108, 1992.
30. P. Soille. *Morphological Image Analysis: Principles and Applications*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.
31. P. Soille, E. J. Breen, and R. Jones. Recursive implementation of erosions and dilations along discrete lines at arbitrary angles. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(5):562–567, 1996.
32. C. Torres-Huitzil. FPGA-based fast computation of gray-level morphological granulometries. *Journal of Real-Time Image Processing*, pages 1–11, 2013.
33. E. R. Urbach and M. H. F. Wilkinson. Efficient 2-D grayscale morphological transformations with arbitrary flat structuring elements. *IEEE Trans. Image Processing*, 17(1):1 –8, jan. 2008.
34. M. van Herk. A fast algorithm for local minimum and maximum filters on rectangular and octagonal kernels. *Pattern Recogn. Lett.*, 13(7):517–521, 1992.
35. J. Velten and A. Kummert. Implementation of a high-performance hardware architecture for binary morphological image processing operations. In *Circuits and Systems, 2004. MWSCAS '04. The 2004 47th Midwest Symposium on*, volume 2, pages II–241 – II–244 vol.2, 25-28 2004.
36. L. Vincent. Morphological grayscale reconstruction in image analysis: applications and efficient algorithms. *Image Processing, IEEE Transactions on*, 2(2):176–201, Apr 1993.

37. Xilinx. Xilinx Power Estimator (XPE). http://www.xilinx.com/products/design_tools/logic_design/xpe.htm.
38. Xilinx. Video Frame Buffer Controller v1.0. http://www.xilinx.com/products/devboards/reference_design/vsk_s3/vfbc_xmp013.pdf, October 29 2007.
39. Xilinx. Virtex-5 family overview. http://www.xilinx.com/support/documentation/data_sheets/ds100.pdf, February 6 2009.
40. Xilinx. LogiCORE IP Multi-Port Memory Controller (MPMC) (v6.03.a). http://www.xilinx.com/support/documentation/ip_documentation/mpmc.pdf, March 1 2011.
41. J. Xu. Decomposition of convex polygonal morphological structuring elements into neighborhood subsets. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(2):153–162, 1991.