Available online at www.sciencedirect.com

SCIENCE DIRECT®

Microprocessors and Microsystems 29 (2005) 381–391

MICROPROCESSORS AND MICROSYSTEMS

www.elsevier.com/locate/micpro

A high-performance fully reconfigurable FPGA-based 2D convolution processor

Stefania Perri[a], Marco Lanuzza[a], Pasquale Corsonello[b,*], Giuseppe Cocorullo[a]

[a]DEIS, University of Calabria, Arcavacata di Rende, 87036 Rende (CS), Italy
[b]DIMET, University of Reggio Calabria, Loc. Feo di Vito, 89060 Reggio Calabria, Italy

Received 17 October 2003; revised 11 February 2004; accepted 8 October 2004
Available online 19 November 2005

Abstract

This paper presents a new fully reconfigurable 2D convolver designed for FPGA-based image and video processors. The proposed architecture operates on image pixels coded with different bit resolutions and varying kernel weights avoiding power and time-consuming reconfiguration. This is made possible by using new SIMD arithmetic modules purposely designed for the new circuit. When optimized for the XILINX VIRTEX device family, the convolver presented in this work requires just 18.4 ms to perform a 5×5 convolution on a 1024×1024 8-bit pixels image and dissipates only 102.1 mW/MHz.

The new circuit can be exploited in all the real-time applications in which adaptive convolutions are required and it can be realized also in many other FPGA device families.

Keywords: Image processing; Convolution; Single instruction multiple data circuits

1. Introduction

Digital image processing and computer vision are rapidly evolving research fields with a growing number of applications for commercial, medical and military purposes. In these areas, improving perceptual information for human vision and processing image data for efficient storage, transmission, and representation are two of the main goals.

Modern image processing and computer vision algorithms require high computational capability, especially when high-resolution images have to be elaborated under real-time requirements. In such applications (e.g. image filtering, image restoration, feature recognition, object tracking, template matching, etc.), the spatial domain two-dimensional (2D) convolution plays a fundamental role [1,2]. For these reasons, the design of efficient convolvers receives great interest [3–7].

In implementing image and video processing algorithms, intensive computations are typically required [3]. Moreover, operations on different precisions data are frequently needed. This happens because certain steps of the algorithms have to operate on high precision data, whereas other steps can operate on lower precisions. In addition, many computations needed in image processing (e.g. image convolution) involve local image transformations resulting in thousands of potentially parallel operations.

In order to accommodate in hardware the requirements for elaborating both high and low precisions data with extended parallelism, appropriate data-paths have to be supported and extreme flexibility has to be guaranteed. The exploitation of Single Instruction Multiple Data (SIMD) circuits represents a good solution. In fact, as is well known, SIMD architectures are able to efficiently elaborate the highest precision data and guarantee fine-grained parallelism in processing data on lower precisions [8,9].

In this work, a new fully reconfigurable 2D convolver for FPGA-based image and video processors is presented. The proposed architecture exhibits extreme flexibility and very high computational capability.

* Corresponding author. Address: DEIS, University of Calabria, Arcavacata di Rende, 87036 Rende (CS), Italy. Tel.: +39 0984 494708; fax: +39 0984 494834.
E-mail address: corsonello@ing.unirc.it (P. Corsonello).

0141-9331/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.micpro.2004.10.004

The core of the convolution processor contains a grid of four 16-bit SIMD 2D 3×3 convolvers. Each one exploits new SIMD arithmetic circuits purposely designed and optimized for the FPGA platform. The SIMD modules adapt their structures at run-time to different bit resolutions. Thanks to this, the new convolution processor is able to operate on 2D images with different bit resolutions and varying kernels avoiding the time and power consuming reconfiguration process.

The paper is organized as follows: design motivations of the proposed circuit are explained in Section 2, then its architecture and the basic modules used in it are described in Sections 3 and 4. Finally, results and conclusions are provided.

2. Research motivations

In image and video processing, convolution is a basic operation. Thus, it can strongly influence the overall performance. A convolution operation is usually performed as illustrated in Fig. 1: for each pixel $P(x,y)$ (with $x=1,\ldots,M$ and $y=1,\ldots,N$) of a $M\times N$ input image a $K\times R$ sliding template, called convolution kernel, is convolved with the $K\times R$ window centered on $P(x,y)$. That is, each value into the pixels window is multiplied by the corresponding signed weight into the convolution kernel. Then, the $K\times R$ products obtained in this way are added to produce the output pixel value.

In several applications, better results are achieved if in a single image enhancement task progressive execution of 2D convolutions with differently sized kernels is exploited.

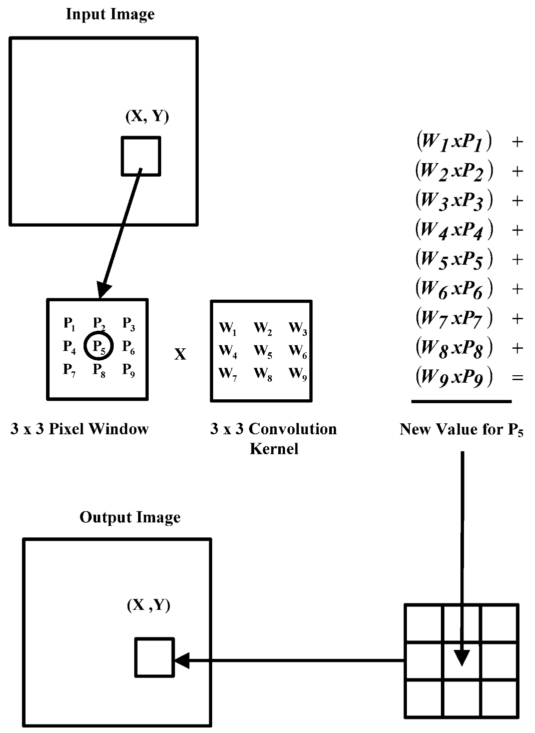

Fig. 1. Image convolution for $K=R=3$.

For example, this happens in medical applications. There, as shown in [10], the enhancement task is useful in scanning skeleton images. The computational flow typically used in these cases is illustrated in Fig. 2, which shows how successive convolutions with differently sized kernels enhance the perceptual quality by sharpening the input image and bringing out more of the skeletal details.

It is worth pointing out that in order to obtain the highest image quality and to avoid inaccurate results, also different precisions have to be supported.

In applications requiring real-time image convolutions, software implementations on general-purpose microprocessors appear to be very time consuming. Moreover, commercial DSPs are often unable to efficiently support image convolution. For example, the TMS320C40 DSP microprocessor [11] requires about 20 instruction cycles per pixel when a 3×3 kernel is used [3]. It is then clear that special purpose parallel circuits for convolution can represent efficient solutions to provide high computational capabilities and to ensure high throughput data rates.

Several hardware implementations of convolvers able to satisfy real-time constraints exist in literature. Many of them [3–7] take advantages of FPGAs to accelerate convolution operations. Even though FPGA devices lead with extremely flexible hardware, previous proposals appear inflexible from an application point of view. In fact, they operate on kernels with fixed-size and fixed-precision pixels (usually 8- or 16-bits). Moreover, just restricted set of kernel weights are typically supported. As an example, the 2D 3×3 convolver presented in [3] supports convolutions in which kernel weights can assume only the values −4, −2, −1, 0, 1, 2, 4. This approach yields excellent performance and device utilization, since only multiplications by constants have to be executed. But, as a drawback, this kind of circuits is useless in applications like video processing with special effects and real-time image manipulation, where the kernel weights and the convolution window size could vary at run-time. For these applications changing convolution window size, pixel resolution and/or kernel weights at run-time avoiding the conventional FPGA reconfiguration process is very useful to save time and power consumption.

3. The architecture of the new 2D run-time reconfigurable convolver

Run-time reconfigurability makes the proposed specialized image convolution processor able to perform user programmable 2D image convolution operations.

The possible operation modes and the parallelism levels supported by the new convolver are summarized in Table 1. The control signals *rec* and *Select_window* establish what pixels resolution and convolution window size the convolver has to operate on, respectively. For example, when both the control signals are low, four adjacent 3×3 image convolutions on 16-bit pixels and 16-bit kernel

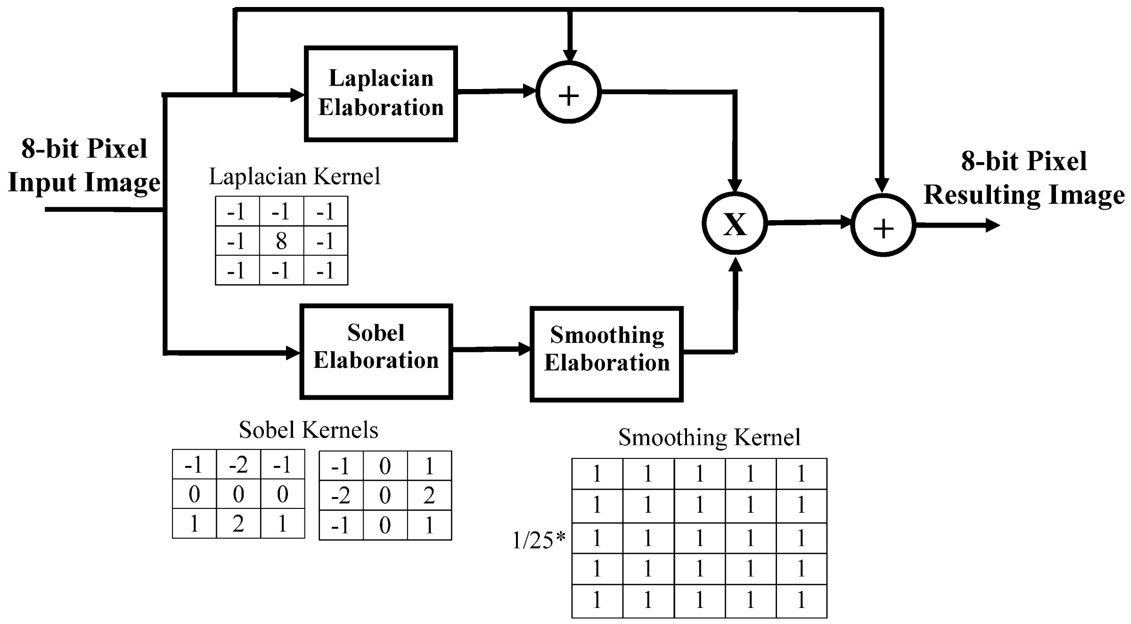

Fig. 2. Enhancement task.

weights are executed in parallel per clock cycle. The new circuit can provide such high parallelism levels thanks to a flexible grid used as computational core and consisting of four 16-bit 2D 3×3 SIMD convolvers. In the following, the latter are referred to as basic convolvers.

In Fig. 3, the whole structure of the new circuit is illustrated. The input image pixels are transferred to the convolver row-by-row in accordance with the usual raster-scan order [3,12]. On the contrary, the kernel weights are initially loaded through the *Pixel_Weights* bus from an external source (e.g. a host computer or an external memory). Then, they are internally stored.

In all the supported operation modes, the arrival of input pixels to the four basic convolvers has to be synchronized to ensure that the pixels involved in the operations are processed at the same time. In order to do this, the *Buffer interfaces*, appropriate internal registers, and *FIFO* memories are used. In this way the basic 3×3 convolvers can operate simultaneously. In fact, at the same time they multiply the received image pixels by the stored kernel coefficients and add the resulting products to form the final output. FIFOs play an important role. In fact, they allow a large number of time-consuming memory accesses to be avoided when the input image is stored in an external RAM memory [12]. To efficiently support all the above shown operation modes, FIFOs have been organized as variable depth 64-bit FIFOs. That is, they can be configured in accordance with the control signals *rec* and *Select_window*.

The 2×2 grid of 3×3 convolvers is organized as shown in Fig. 4. It allows 3×3 and 5×5 2D convolutions to be supported on both 16-bit and 8-bit image pixels and kernel weights. This is made possible thanks to purposely designed SIMD arithmetic circuits and to appropriate multiplexing stages exploited in the basic 2D 3×3 convolvers. The SIMD submodules are able to perform both 16-bit and 8-bit additions and multiplications. As a consequence, each basic convolver can execute one 16-bit 3×3 convolution or two parallel and independent 3×3 8-bit convolutions. When a 5×5 convolution is required, each basic convolver elaborates a 3×3 window, which is a subarray of the larger 5×5 image window. The multiplexing stages visible in Fig. 4 allow the input data of the basic convolvers to be properly formed as a function of the chosen kernel size. In the referred case, adjacent convolvers have to simultaneously elaborate different portions of the 5×5 convolution window [3]. In order to ensure data arriving at appropriate time for both 16-bit and 8-bit pixels resolutions three 16-bit delay registers are used. Partial results produced by the first three convolvers are inputted to the fourth convolution circuit, which then computes the expected final output.

Table 1
Supported instructions

Rec	Select_window	Instruction	Parallelism level
0	0	16-bit 3×3 image convolution	4
1	0	8-bit 3×3 image convolution	8
0	1	16-bit 5×5 image convolution	1
1	1	8-bit 5×5 image convolution	2

4. The 3×3 basic SIMD convolver

In order to describe the implementation strategy exploited in the basic 2D 3×3 SIMD convolver, let us consider a 3×3

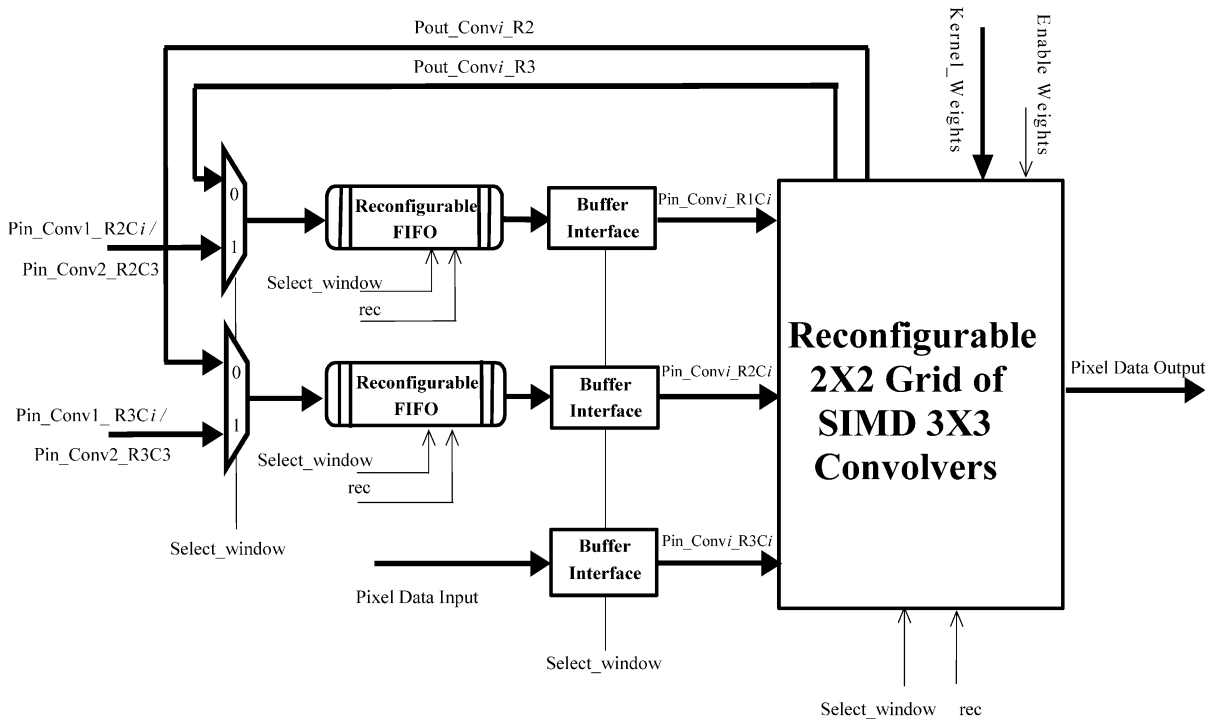

Fig. 3. The new convolution processor architecture.

convolution mask with 16-bit kernel weights and 16-bit pixels. In this case, the relationship between the input pixels $P(x,y)$, the convolution kernel weights $W(i,j)$ and the convolved pixels $P'(x,y)$ is given by Eq. (1)

$$P'_{(x,y)}[15:0] = \mathrm{SAT}\left\{ \sum_{i=-1}^{+1} \sum_{j=-1}^{+1} P_{(x+i,y+j)}[15:0]W_{(i,j)}[15:0] \right\} \quad (1)$$

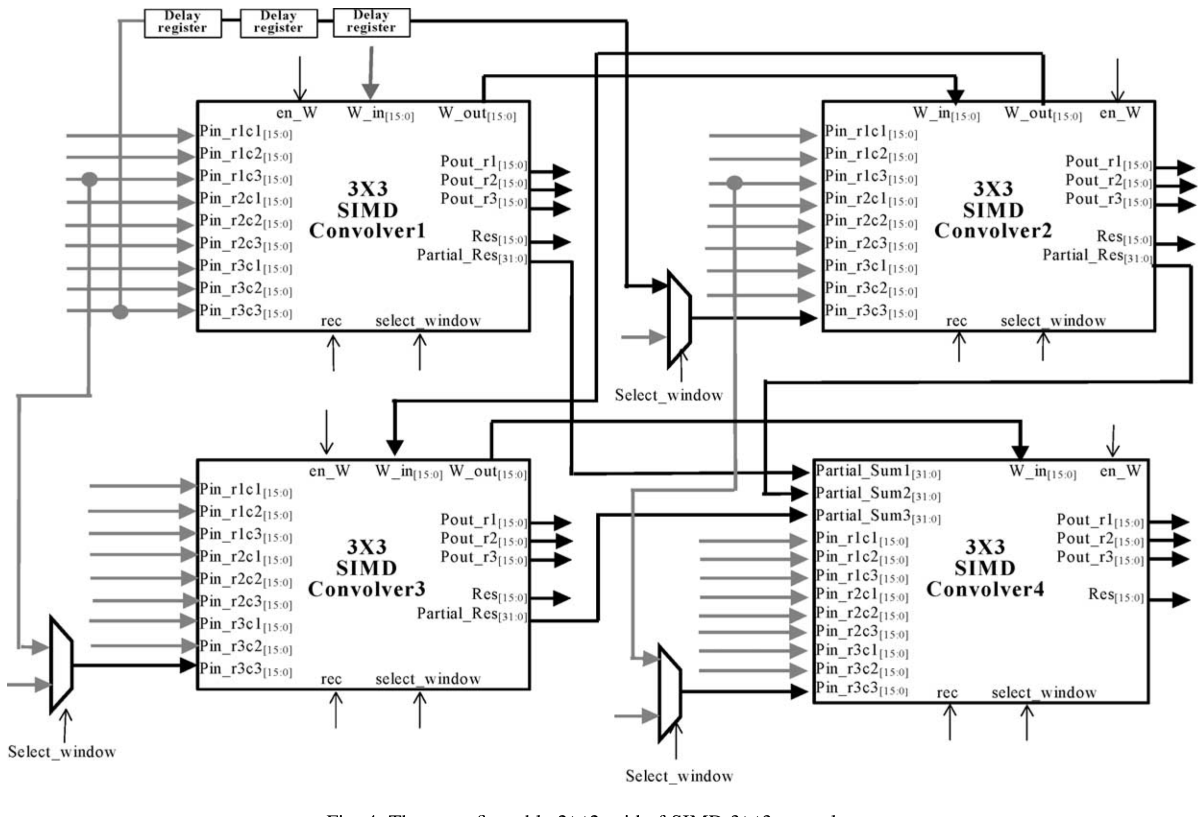

Fig. 4. The reconfigurable 2×2 grid of SIMD 3×3 convolvers.

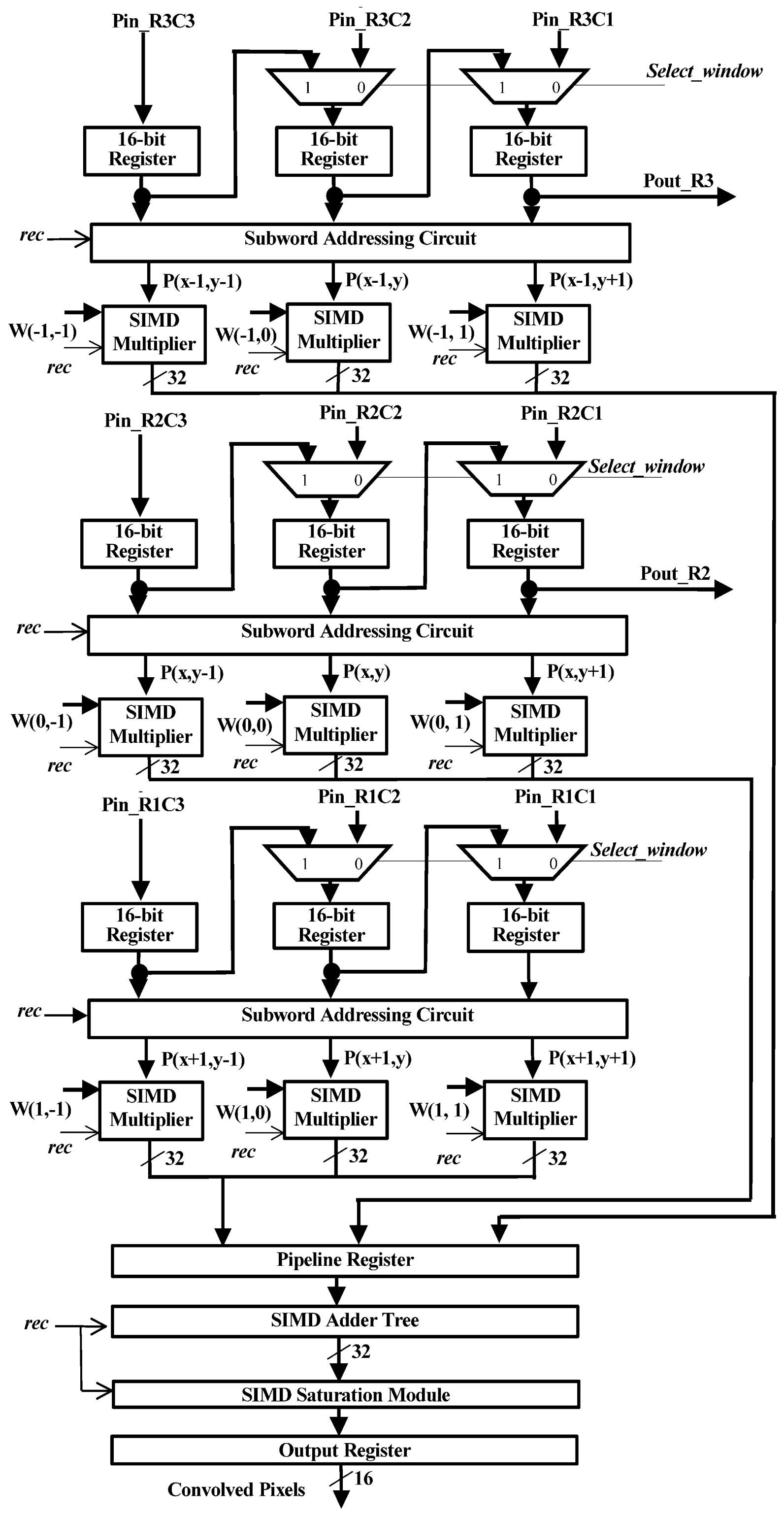

Fig. 5. The 2D SIMD 3×3 basic convolver.

where x and y are the coordinates of pixels into the input image; i and j are the coordinates of weights into the kernel mask; and SAT indicates a saturation operation performed on the output pixel. Saturation is used to fix the bit resolution of the resulting pixel to 16-bits. Negative results are saturated to zero, whereas overflowing positive results are saturated to the maximum 16-bit positive value.

As shown in [8,9], a 16×16 binary multiplication can be performed operating on the 8-bit subwords of the operands. Therefore, Eq. (1) can be rewritten as Eq. (2)

$$P'_{(x,y)}[15:0] = P'_{(x,y)}[15:8]\text{link}P'_{(x,y)}[7:0]$$
$$= \text{SAT}\left\{ \left[\sum_{i=-1}^{1}\sum_{j=-1}^{1} W_{(i,j)}[15:8]P_{(x+i,y+j)}[15:8] \right] \times \text{link} \left[\sum_{i=-1}^{1}\sum_{j=-1}^{1} W_{(i,j)}[7:0]P_{(x+i,y+j)}[7:0] \right] + \sum_{i=-1}^{1}\sum_{j=-1}^{1} [\text{LLS8}(W_{(i,j)}[15:8]P_{(x+i,y+j)}[7:0]) + \text{LLS8}(W_{(i,j)}[7:0]P_{(x+i,y+j)}[15:8])] \right\} \quad (2)$$

where link and LLSk indicate a link action and a logical left shift by k bit positions, respectively.

Eq. (2) suggests that a 16-bit 2D convolver can be realized exploiting SIMD arithmetic circuits, able to operate simultaneously on the operands subwords. The independent results obtained in this way can be then combined to generate the whole expected 16-bits saturated result. The main advantage of this approach resides in the possibility of performing also two parallel independent 8-bit convolutions using the same circuit.

In Fig. 5, the top-level architecture of the basic 2D 3×3 SIMD convolver is depicted. As explained in Section 3, each basic convolver can operate in autonomous (*Select_window* = 0) or in linked mode (*Select_window* = 1). When *Select_window* is low the image pixels are inputted in parallel to the nine 16-bit input registers. On the contrary when *Select_window* is high only the first 16-bit register of each convolution row receives a pixel per clock cycle. Kernel weights are pre-loaded from an external source. It is worth noting that each 16-bit register contains one 16-bit or two 8-bit pixel values in accord to the signal *rec*.

In order to operate in SIMD fashion, the *Subword Adressing Circuit* illustrated in Fig. 6 has been purposely designed. It implements a multiplexing stage, which dispatches the appropriate operands to the SIMD multipliers depending on the signal rec. When convolutions on 16-bit pixels are required, each SIMD multiplier performs one 16×16 pixel-by-weight product. The nine 32-bit independent results obtained in this way, are then added by the *SIMD Adder Tree*, which generates a 32-bit word. The latter is inputted to the *SIMD Saturation Module* that saturates the output to the appropriate 16-bit value. On the contrary, when convolution operations on 8-bit pixels have to be performed, each multiplier executes two independent 8×8-bit multiplications. Thus, two separate sets of nine 16-bit products are obtained in parallel. Then, they are independently added by the *SIMD Adder Tree*, which produces two independent 16-bit results that are separately saturated to the appropriate 8-bit values by the *SIMD saturation module*.

From Fig. 5, it can be seen that each basic convolver is structured as a two stages pipeline. Thus, the whole convolution processor exhibits a latency of just one pixel line plus two clock cycles.

4.1. The SIMD multiplier

As it is well known designing efficient multipliers for an FPGA platform is not trivial. In fact, many of the existing multiplier architectures are typically inadequate to take advantages of dedicated carry-propagate logic and fast routing resources available in FPGA devices. As an example, referring to ASIC designs, the well-known Booth and Wallace Tree multipliers are faster than other multiplier schemes, but their irregular structures result unsuitable to exploit dedicated routing resources available

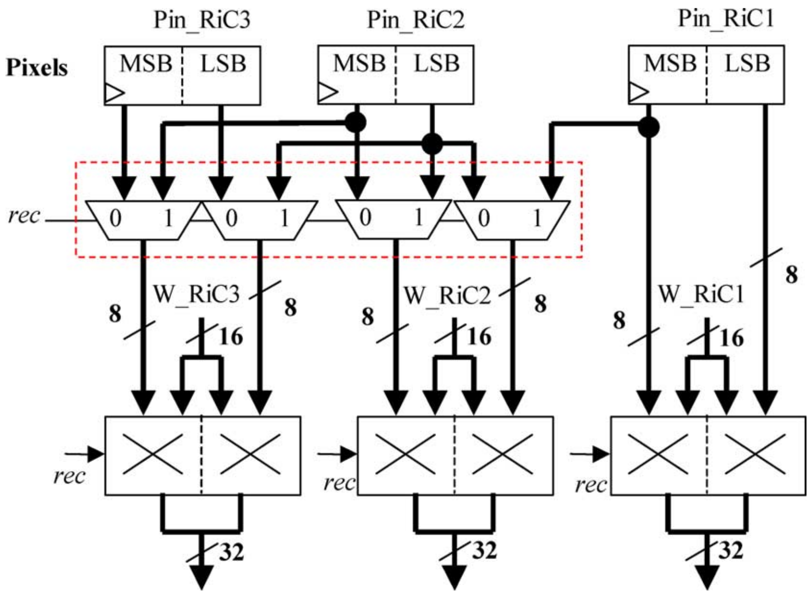

Fig. 6. The subword addressing circuit.

in FPGAs. On the contrary, Carry Propagate Array Multipliers allow the special resources available into FPGAs to be efficiently exploited [13].

As previously mentioned, a 16×16-bit binary multiplication can be computed by combining the 16-bit subproducts between the most significant and least significant 8-bit subwords ($P=P[15{:}9]\ P[7{:}0]$; $W=W[15{:}8]\ W[7{:}0]$) of the operands as shown in Fig. 7. This approach suggests that a 16-bit Carry propagate Array Multiplier can be partitioned to compute also two 8×8-bit independent binary multiplications.

In Fig. 8, the block diagram of the proposed SIMD multiplier is depicted. The *Multiplicand Management Unit* properly forms the input data of the *Reconfigurable Partial Product Generators*. In particular, as summarized in Eqs. (3) and (4) it provides either the whole 16-bit operand P or its 8-bit subwords properly extended to 16-bit

$$\mathrm{PZ}[i] = \begin{cases} P[i], \ \mathrm{rec}=0 \\ 0, \ \mathrm{rec}=1 \end{cases} i=0,\ldots,7 \tag{3}$$

$$\mathrm{SI\_PZ}[i] = \begin{cases} P[i], \ \mathrm{rec}=0 \\ P[7], \ \mathrm{rec}=1 \end{cases} i=8,\ldots,15 \tag{4}$$

Eight 2×16-bit *Reconfigurable Partial Product Generators* generate partial products, which are then added two by two to generate the 32-bit multiplication results. As shown in Fig. 8, to this aim three levels of addition are needed. Among the used adders four are *Reconfigurable Ripple Carry Adders*, which operate on different precision data thus making the partition of the Array Multiplier possible. The remaining three adders are fixed-precision *Carry Propagate Adders*.

All of the modules used in the SIMD multiplier optimally exploit the dedicated fast logic and routing resources available into a FPGA device. In the following, the Xilinx Virtex family is referred to, but a similar approach can be applied in many other device families. In Xilinx Virtex FPGAs [13] each slice contains two four input Look-Up Tables (LUTs), implementing logic function generators, two flip-flops and additional AND components (MULT_AND), multiplexers (MUXCY) and exclusive OR (XORCY) gates for high-speed arithmetic circuits. MULT_AND gates are used for implementing fast and small multipliers, whereas MUXCY and XORCY gates are suitable to realize 1-bit high-speed full-adders. All these auxiliary elements have been used in the realization of the high speed carry chains involved in the arithmetic circuits purposely designed for the new convolver.

Fig. 9 details how dedicated FPGA's logic has been efficiently exploited to realize the 2×16-bit *Reconfigurable Partial Product Generator*. It can be noted that two partial products are generated ANDing the operand $P[15{:}0]$ with the operand bits $W[i+1,i]$. Then, the obtained results are added. To perform the above operation 4-bits are inputted to each LUT in the slice. The output of the LUT drives the selection control line of MUXCY and one input of XORCY. The former generates the carry propagated to the next 1-bit addition stage whereas the latter provides the sum bit of the current addition stage. It is also important to underline that the MULT_AND gate guarantees fast generation of the carry generate signal in the current sum position.

From Fig. 9, it can be seen that a MUXCY controlled by the signal *rec* is used to break the carry propagation in an established position. When 16×16-bit multiplication is required (*rec*=0) this MUXCY propagates the carry signal to the next addition stage, otherwise a logic zero is propagated. In this way, two independent results are generated for lower precision data.

In Fig. 10, the structure of the *Reconfigurable Ripple Carry Adders* is illustrated. Each LUT in the slice receives two operand bits (e.g. $A[i]$ and $B[i]$) as input and generates the corresponding propagate signal $P[i]=A[i]\ xor\ B[i]$. The LUT output drives the selection input of MUXCY and one input of XORCY, which generate the carry-out and the sum

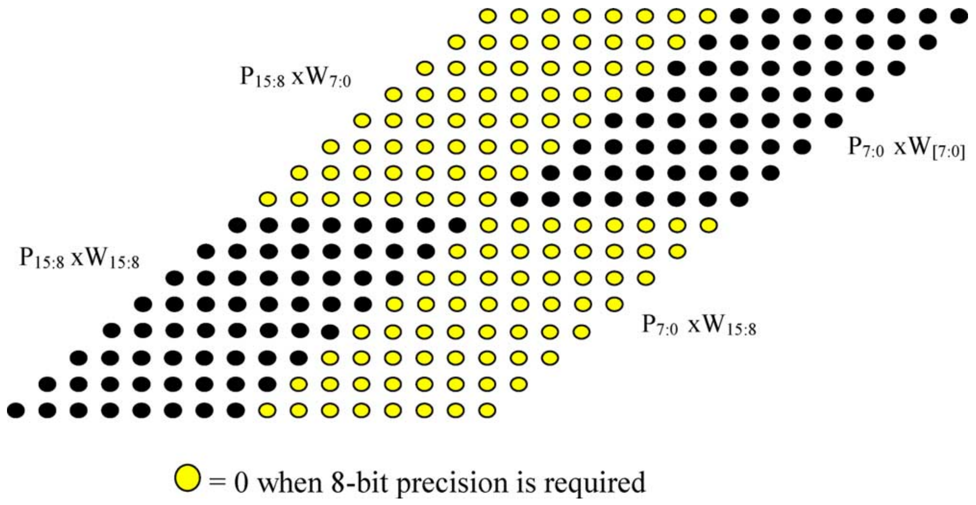

Fig. 7. The SIMD multiplier tree.

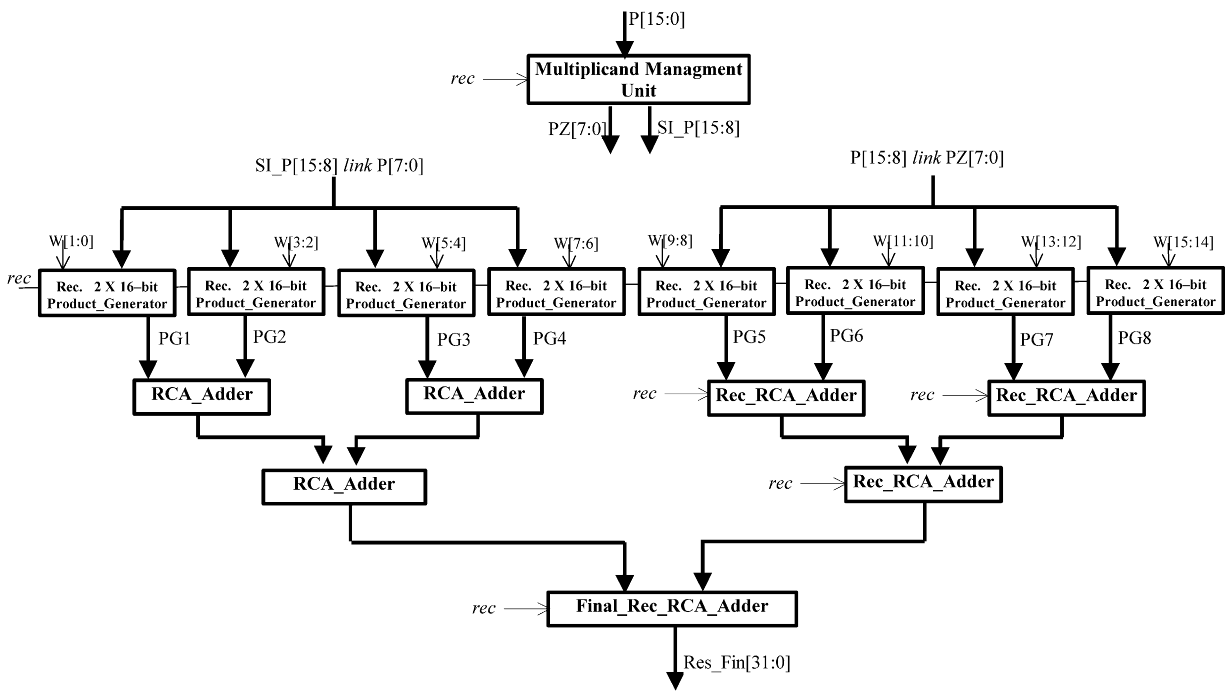

Fig. 8. The 16-bit SIMD multiplier.

bit of the full-adder, respectively. Using this full-adder scheme the length of the carry chain is 2-bits for slice and the carry propagation occurs quickly through the special routing resources. Also for the *Reconfigurable Ripple Carry Adders*, a MUXCY controlled by the signal *rec* has been used to guarantee run-time partitioning of the Array Multiplier.

4.2. The SIMD Adder Tree

To efficiently add the 32-bit partial results coming from the nine SIMD multipliers belonging to a basic 3×3 convolver the adder tree shown in Fig. 11 has been used. It consists of eight SIMD Ripple Carry Adders implemented using the FPGA dedicated carry chains. Depending on the signal *rec*, each adder generates either one 32-bit or two independent 16-bit results.

4.3. The SIMD saturation module

The SIMD saturation Module performs a saturation operation on the output pixels. Negative results are saturated to zero, whereas overflowing positive results are saturated to a maximum positive value depending on the actual precision. Relations Eqs. (5) and (6) show how saturation is performed when $rec=0$ (i.e. one 16-bit saturated result

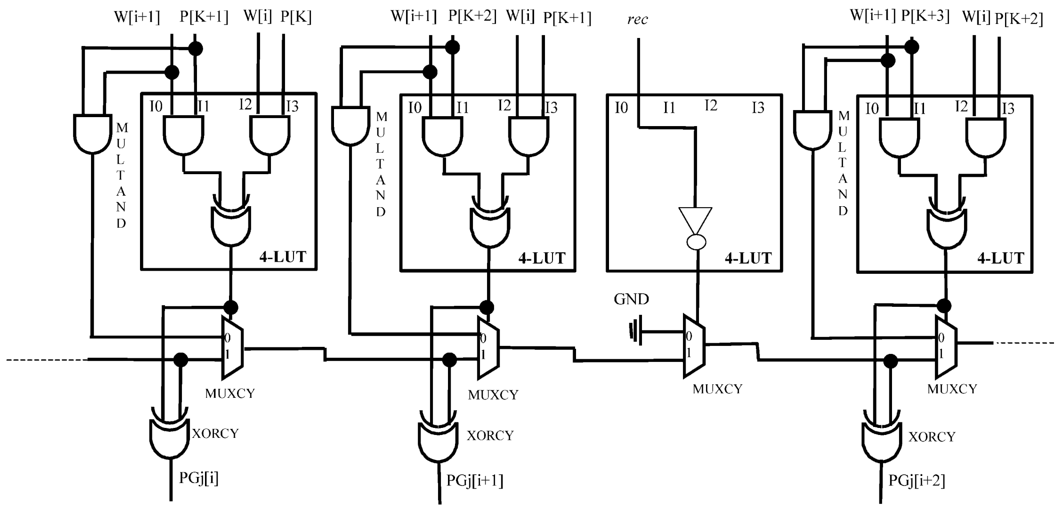

Fig. 9. The reconfigurable 2×16 partial product generator.

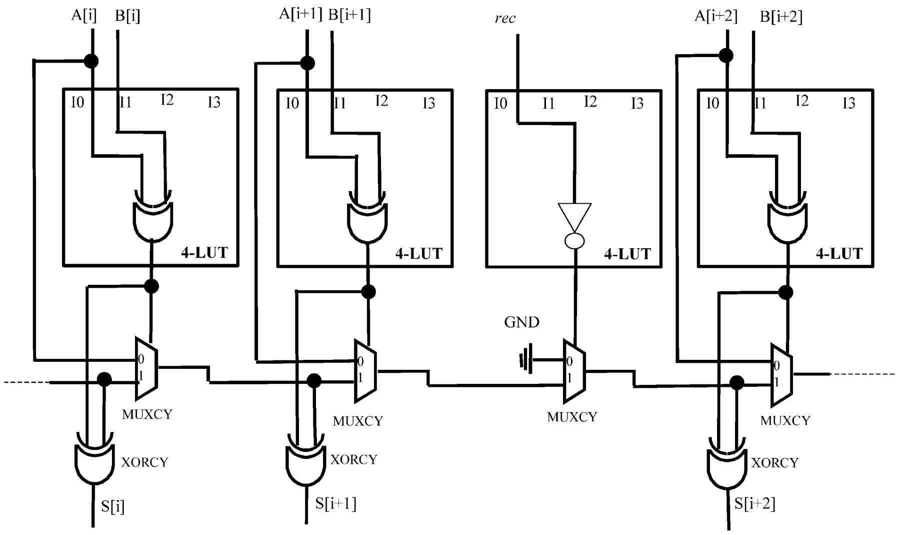

Fig. 10. The reconfigurable ripple carry adder.

has to be generated) and $rec=1$ (i.e. two 8-bit saturated results have to be generated), respectively

$$\begin{aligned}&\text{If } S[31]=1\rightarrow \text{Saturated Result}[15:0]=0\\&\text{If } S[31]=0\ \&\ \text{OR}(S[30],\dots,S[16])=1\\&\quad\rightarrow\text{Saturated Result}[15:0]=1\\&\text{Else Saturated Result}[15:0]=S[15:0]\end{aligned}\tag{5}$$

$$\begin{aligned}&\text{If } S[15]=1\rightarrow \text{Saturated Result}[7:0]=0\\&\text{If } S[31]=1\rightarrow \text{Saturated Result}[15:8]=0\\&\text{If } S[15]=0\ \&\ \text{OR}(S[14],\dots,S[8])=1\\&\quad\rightarrow\text{Saturated Result}[7:0]=1\\&\text{If } S[31]=0\ \&\ \text{OR}(S[30],\dots,S[24])=1\\&\quad\rightarrow\text{Saturated Result}[15:8]=1\\&\text{Else Saturated Result}[7:0]=S[7:0]\\&\qquad\text{Saturated Result}[15:8]=S[23:16]\end{aligned}\tag{6}$$

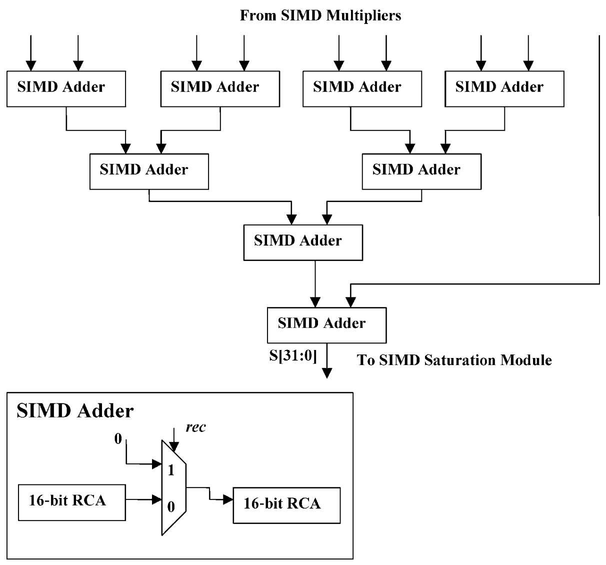

Fig. 11. The SIMD Adder Tree.

5. Results

The proposed FPGA-based convolver processor has been realized on a XILINX VirtexE XCV2000e device using the XILINX™ Integrated Software Environment (ISE) 5.2i and it has been prototyped for the elaboration of 256×256 image block size. In this case, to properly manage 3×3 and 5×5 convolution windows for both 8-bit and 16-bit pixels and kernel weights, FIFOs visible in Fig. 3 are made configurable as 32-, 64-, 128- or 256-depth 64-bit FIFOs. Obviously, what condition arises is established in accordance with the control signals *rec* and *Select_window*.

The layout of the new circuit is illustrated in Fig. 12. It has been realized utilizing 2048 slices for the two Reconfigurable FIFOs and 7262 slices for the basic convolvers and buffer interfaces. In other words, only 48% of the available slices have been occupied. Such compact layout has been obtained by a very careful place and route that also allowed performances of each basic 3×3 convolver to be optimized. In fact, also the carry-chains existing in the SIMD computational elements of the proposed convolver optimally exploit fast logic and routing resources available in Virtex FPGAs.

Timing Analyzer and XPower tools from XILINX ISE 5.2i software have been used for post-layout measuring the worst-case delay and the power consumption, respectively. For the dissipated power measurement the default parameters values provided by XPower have been used with uniform distribution of the operands (i.e. all activity rates equal to 100%) and a capacitive load of 10 pF for all the output signals.

Post-layout characterization results show that the new convolver reaches a maximum running frequency of 28.6 MHz. Thus, it can execute up to 228.8 MOPS and 114.4 MOPS when performing 8-bit and 16-bit 3×3 2D convolutions, respectively. Whereas, when 5×5 2D

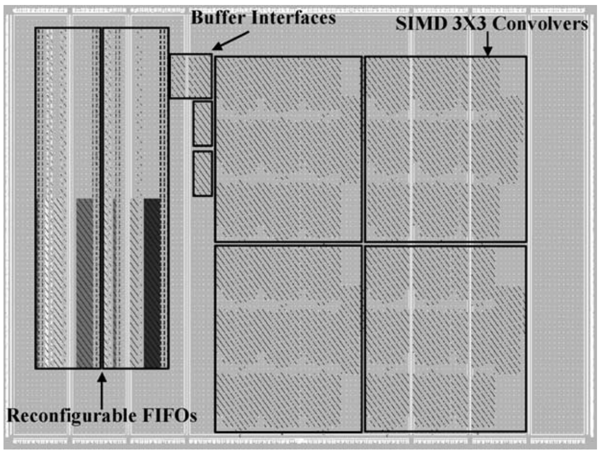

Fig. 12. The layout of the new convolution processor.

convolutions are required up to 57.2 MOPS and 28.6 MOPS can be supported by the proposed architecture. This very high computational capability is achieved with an average energy dissipation of just 102.1 mW/MHz.

The proposed circuit has been compared to the FPGA-based convolvers presented in [3,5,6]. Comparison results are summarized in Table 2. It is important to note that characteristics of the compared circuits are related to the elaboration on specific image block sizes. Nevertheless, the advantages of the convolver presented in this work are evident with respect to the previous proposals. The new circuit exhibits an extreme flexibility accomplished with the highest computational capability. In particular, it performs both 3×3 and 5×5 convolutions on both 8-bit and 16-bit data guaranteeing the maximum parallelism level for all the supported operation modes and avoiding time- and power-consuming bit-stream uploading. On the contrary, implementations described in [3,5] can perform just one 8-bit 3×3 convolution. The circuit recently proposed in [6] provides an intermediate flexibility since it can execute one 16-bit 3×3 convolution or two parallel 8-bit 3×3 convolutions. Furthermore, it is worth underlining that the convolvers described in [3,5] supports only fixed sets of kernel weights. More exactly, the architecture described in [3] requires the execution of only multiplications by constants, whereas the convolver presented in [5] requires only multiplications by power of two (i.e. simple left shifts). Due to this, lower area occupancy is obtained but at the expense of flexibility and computational capabilities. On the contrary, the new convolver and that reported in [6] can operate on any kernel weights since they use general multipliers.

In order to demonstrate the efficiency of the new convolution processor with respect to previous proposals, the computational time required to perform a 3×3 convolution on a 1024×1024 image with 8-bit pixels has been evaluated for all the compared circuits. As shown in Table 2, the proposed approach allows a much lower elaboration time to be reached. When performing a 5×5 convolution on a 1024×1024 image with 16-bit pixels the proposed circuit requires just 36.7 ms. It is worth pointing out that previously proposed convolvers do not support this operation mode. Thus, comparison data are not available.

6. Conclusions

Modern digital image processing and computer vision applications can take advantage from 2D convolutions on different kernel sizes and bit-resolution image pixels. SRAM-based FPGAs seem the obvious candidates to achieve an extreme flexibility.

In this paper, the design of a new high-speed energy efficient FPGA-based 2D convolver has been presented. The new architecture supports 3×3 and 5×5 2D convolutions for both 16-bit and 8-bit pixels and kernel weights avoiding the conventional time and power consuming reconfiguration process. This nice feature makes the new circuit very suitable to efficiently support

Table 2
Comparison results

	New	[3]	[5]	[6]
Block size	256×256	68×68	128×128	68×68
Device	XCV2000e	XC4013	XCV300	XCV400
Energy (pJ)	102.1	NA	NA	15.7
No. of device	1	2	1	1
Area for logic	7262 slices	484 CLBs[a]	584 slices	1758 slices
Memory size	32 Kbits	1040-bits	7.68 Kbits	2080-bits
MOPS 8-bit 3×3	228.8	25	49	78.8
MOPS 16-bit 3×3	114.4	–	–	39.4
MOPS 8-bit 5×5	57.2	–	–	–
MOPS 16-bit 5×5	28.6	–	–	–
3×3 convolution on 1024×1024 8-bit pixels image (ms)	4.6	42	21	13

[a] It has been obtained from data given in [3] ignoring the number of slices used for FIFOs, control and interface modules.

modern image processing algorithms in which adaptive 2D convolutions are required.

References

[1] V. Hecht, K. Rönner, P. Pirsch, An advanced programmable 2D-convolution chip for real time image processing, Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS), 1991, pp. 1897–1900.

[2] Y. Leblebici, A. Kepkep, F.K. Gürkaynak, H. Özdemir, Fully programmable real time (3×3) image filter based on capacitive threshold-logic gates, Proceedings of IEEE International Symposium on Circuits and Systems, 1997, pp. 2072–2075.

[3] B. Bosi, G. Bois, Y. Savaria, Reconfigurable pipelined 2D convolvers for fast digital signal processing, IEEE Transactions on VLSI Systems 7 (3) (1999).

[4] R.G. Shoup, Parameterised convolution filtering in an FPGA in: W. Moore, W. Luk (Eds.),, More FPGAs, Abingdon EE&CS Books, Abingdon, England, 1993, pp. 274–280.

[5] A.E. Nelson, Implementation of Image Processing Alghorithms on FPGA Hardware, PhD thesis, Faculty of the Graduate School of Vanderbilt University, Nashville, TN, May 2000.

[6] S. Perri, M. Lanuzza, P. Corsonello, G. Cocorullo, SIMD 2-D convolver for fast FPGA-based image and video processors, Proceedings of the Military and Aerospace Programmable Logic Devices (MAPLD) International Conference, Washington, USA, September 2003.

[7] A.S. Dawood, S.J. Visser, J.A. Williams, Reconfigurable FPGAS for real time image processing in space. Proceedings of the14th International IEEE Digital Signal Processing (DSP) Conference, vol. 2, 2002, pp. 845–848.

[8] S. Perri, M. Lanuzza, P. Corsonello, G. Cocorullo, Fully-synthesizable reconfigurable multiplier for high-performance multimedia processors, Proceedings of the International Signal Processing Conference, Dallas, Texas, March 31–April 3, 2003.

[9] A. Farooqui, V.G. Oklobdzija, A programmable data-path for MPEG-4 and natural hybrid video coding, 34th Annual Asilomar Conference on signals, Systems and Computers, Pacific Grove, California, October 29–November 1, 2000.

[10] R. Gonzalez, R. Woods, Digital Image Processing, second ed., Prentice-Hall, Englewood Cliffs, NJ, 2002.

[11] Texas Instruments, TMS320C40 DIGITAL SIGNAL PROCESSOR Datasheet, January 1996.

[12] A. Benedetti, A. Prati, N. Scarabottolo, Image convolution on FPGAs: the implementation of a multi-FPGA FIFO structure, Proceedings of the 24th Euromicro Conference, Vesteras, Sweden, 1998.

[13] Xilinx Inc., Constraints Guide—ISE 5, USA, 2002.