A Multiwindow Partial Buffering Scheme for FPGA-Based 2-D Convolvers

Hui Zhang, Mingxin Xia, and Guangshu Hu

***Abstract*—Two-dimensional (2-D) convolution is widely used in image and video processing. Although the operation is simple, 2-D convolution is however both computationally expensive and memory-intensive. Field-programmable-gate-array (FPGA)-based parallel processing architectures were proposed to accelerate calculations for 2-D convolution. And data buffers implemented with FPGA on-chip resources were used to avoid direct access to external memories. Full buffering and partial buffering (PB) schemes were adopted in previous works. The former would consume a large amount of FPGA resources, while the latter would cause a sharp increase in external memory bus bandwidth. In this brief, we present a multiwindow PB scheme for FPGA-based 2-D convolvers. Compared with the aforementioned methods, the new buffering strategy exhibits a good balance between on-chip resource utilization and external memory bus bandwidth, and therefore is suitable for low-cost FPGA implementation.**

***Index Terms*—Field-programmable gate arrays (FPGAs), full buffering (FB), multiwindow (MW), partial buffering (PB), two-dimensional (2-D) convolution.**

I. Introduction

TWO-DIMENSIONAL (2-D) convolution is a basic operation in image and video processing [1]. Despite its simple representation, the implementation of 2-D convolution is not trivial because it is not only computationally expensive but also memory-intensive. With a $R \times S$ convolution mask, it requires $R \times S$ multiplications and $R \times S - 1$ additions, as well as $R \times S$ accesses to the input image data for the calculation of a single output pixel. A huge demand for computational power and memory bandwidth would be resulted if real-time processing of input images is required [2].

Field-programmable-gate-array (FPGA)-based 2-D convolvers, with specially designed parallel architectures to exploit the inherent data and instruction level parallelism of 2-D convolution [2]–[4], were proposed to speed-up calculations of 2-D convolution. For these convolvers, a total of $R \times S$ input image pixel values must be simultaneously available in a single cycle to reach a throughput data rate of 1 clock/pixel [2]–[4]. External memory devices such as SRAMs and DRAMs are generally used to hold input images, but their memory bandwidths can't directly satisfy the aforementioned requirement. In such cases, internal data buffers implemented with FPGA on-chip resources were used to avoid direct access to external memories [3], [5]. Internal data buffers could be designed to provide a sufficiently large bandwidth, for example, multiple data ports could be attached to internal data buffers to feed as much data as needed to the 2-D convolver in a single cycle. Besides, when a pixel value was loaded into the internal buffers, it could be reused for successive windows' calculation to avoid fetching it repetitively from external memories. As a result, the requirement for external memory bandwidth was reduced to an acceptable level.

A full buffering (FB) scheme was adopted in the $R \times S$ convolver architecture in [3], where internal data buffers were used to hold $R-1$ full raster lines of the input image. Most part of the data buffers functioned as delay lines to temporally hold data. Storing several lines of image on-chip would consume a large amount of FPGA resources, which is evidently not an economic solution, especially when the input image size and/or convolution mask size are large [3], [5], [6]. A partial buffering (PB) scheme, which was named single-window (SW) PB scheme in this brief, was also proposed in [3] to reduce resource consumption. Similar buffering strategies were used in [5], [6]. Delay lines were eliminated in SWPB schemes and the total amount of resources utilized for data buffers was greatly reduced, but a sharp increase to as much as R times in external memory bus bandwidth was resulted in order to the keep the 1 clock/pixel throughput rate [6].

It is always desirable to design a 2-D convolver that is both high-performance and area-efficient, especially when a low-cost FPGA device with limited on-chip resources is targeted, or when 2-D convolution is only a part of the whole image processing procedure. In this brief, we present a MWPB scheme, which uses much less FPGA resources compared with FB schemes, and at the cost of a much lower elevation in external memory bus bandwidth compared with SWPB schemes. The good balance between on-chip resource utilization and external memory bus bandwidth exhibited by the proposed buffering strategy makes it suitable for low-cost FPGA implementation.

II. MWPB Scheme

In this section, we will first briefly introduce the implementation strategy for 2-D convolution. Then we will discuss the advantages and disadvantages of existing FB and PB schemes for FPGA-based 2-D convolvers. And finally we will present the new MWPB scheme. Throughout this section, an input image size of $M \times N$ and a convolution mask size of $R \times S$ are assumed.

A. 2-D Convolution Implementation Strategy

2-D convolution with the input image I is defined by

$$I'_{m \cdot n} = \sum_i \sum_j w_{i,j} \times I_{m+i,n+j} \qquad \forall (i,j) \in R \times S \quad (1)$$

Manuscript received May 14, 2006; revised August 4, 2006. This work was supported in part by the National Basic Research Program of China (973 Program under Grant 2006CB705700). This paper was recommended by Associate Editor T. Stouraitis.

The authors are with the Department of Biomedical Engineering, Tsinghua University, Beijing 100084, China (e-mail: hzhang@tsinghua.edu.cn).

Digital Object Identifier 10.1109/TCSII.2006.886898

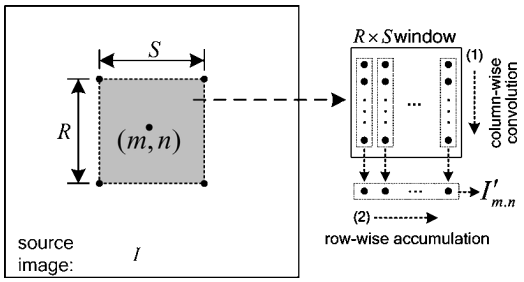

Fig. 1. Concept view of dividing a 2-D convolution into separate 1-D convolutions.

where I' is the output image, and $w_{i,j}$ is the convolution kernel weight. To calculate an output pixel $I'_{m,n}$, a $R \times S$ window centered on (m, n) is extracted from the input image, and each pixel in the window is multiplied by the corresponding kernel weight and the products are then added to produce the output pixel value.

Basically, 2-D convolution can be divided into separated 1-D convolutions. Fig. 1 shows such a concept. For a $R \times S$ window extracted from the input image, column-wise (or row-wise) 1-D convolutions are first performed for all the columns (or rows) in the window, and then the intermediate results are accumulated to obtain the final output pixel value. This indicates that 1-D convolution modules that perform convolution on a column (or row) of data within the window area can be used as basic building blocks for 2-D convolvers [2], [3].

To speed up the computation of 2-D convolution, simultaneous access to multiple data in a window is required, so that calculations on several sites could be performed simultaneously. For example, if data in a column (R pixels) are simultaneously available, a single 1-D convolution module can be utilized to perform the calculation of a window in a column-by-column manner, and the throughput rate in this case is S clocks/pixel. When more data are simultaneously available, multiple 1-D convolution modules can be employed to increase the throughput rate. When data in S columns, i.e., a whole window, are simultaneously available, S 1-D convolution modules can be linked together to form a complete 2-D convolver and a throughput rate of 1 clock/pixel is achieved.

B. FB Scheme

It is common practice to use FPGA internal memories as data buffers to avoid direct access to external memories by the 2-D convolver. A FB scheme was adopted for the 2-D convolver in [3]. Pixels are fetched from external memories and shifted into internal buffers line by line until $R-1$ raster lines and the first S pixels in the next line are loaded. At that point, all the pixels belonging to the first $R \times S$ window are available for the 2-D convolver. From that moment on, each new pixel shifted in will effectively move the convolution window to a next position (see Fig. 2). And a throughput rate of 1 clock/pixel is achieved.

In the FB scheme, $R-1$ delay-lines, each with a length of $N-S$, are employed to temporally hold data before they are fed to the 2-D convolver, and R sets of register arrays, each consisting of S shift registers, are used to assemble the $R \times S$ convolution window that is currently used by the 2-D convolver. Delay-lines are basically shift registers and can be implemented with flip-flops or Block-RAMs depending on the specific FPGA device.

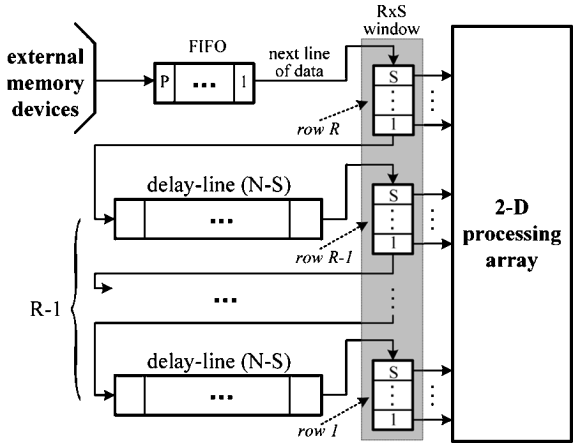

Fig. 2. FB scheme for a $R \times S$ convolver. P represents the depth of FIFO.

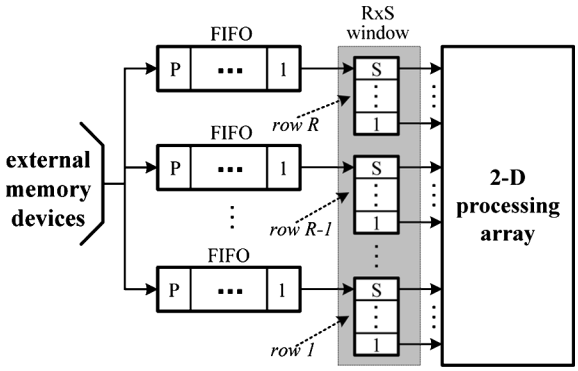

Fig. 3. SWPB scheme for a $R \times S$ convolver. P represents the depth of FIFO.

The advantage of a FB scheme is that only a single dataflow is needed to feed data to the internal buffers. Each pixel in the input image needs to be read from external memories only once, and the required external memory bandwidth is at a minimum level (1 pixel/clock) in such a case. The disadvantage is that, because of the delay-lines, it is very expensive for FPGA-based implementations. A total of $(R-1) \times N + S$ shift registers are needed for the buffering strategy, which accounts for a significant part of FPGA resources when the input image size and/or convolution mask size are large [5], [6].

In Fig. 2, an input first-in first-out (FIFO) is also included in case that external memory bus word length is larger than the pixel data length, which is common for most applications [3].

C. SWPB Scheme

An alternative to FB schemes is to store only a small number of image pixels on-chip [3], [5], [6]. In the SWPB scheme proposed in [3], only a small portion of pixels in a raster line are stored in on-chip buffers. Each set of shift register array in the convolution window receives the pixels belonging to consecutive rows of the input image through a FIFO. With R pixels shifted from FIFOs into shift register arrays, a column of data is read into the convolution window, and consequently the window moves to a next position (see Fig. 3).

Compared with Fig. 2, delay-lines are completed eliminated for the SWPB scheme in Fig. 3. Instead, a much smaller FIFO is used to feed data to each shift register array. Therefore, a great reduction in terms of shift registers will be achieved. However, multiple dataflow must be provided to update the convolution

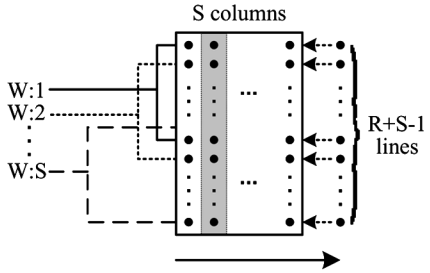

Fig. 4. Conceptual view of assembling S windows in a $(R + S - 1) \times S$ window area. $W : i$ represents the ith window.

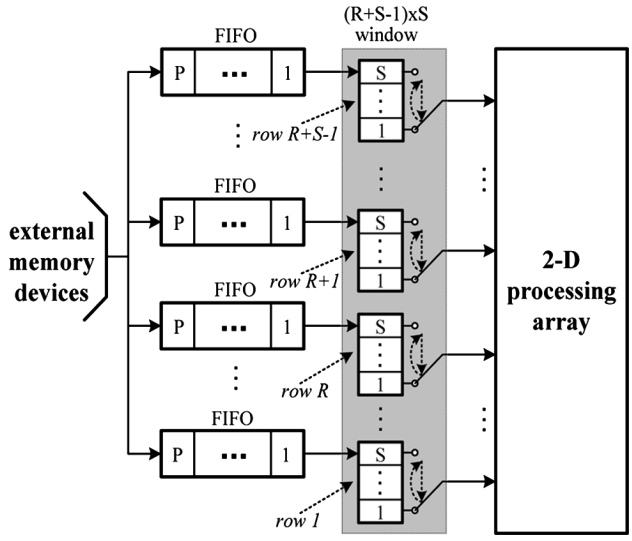

Fig. 5. MWPB scheme for a $R \times S$ convolver. P represents the depth of FIFO.

window. Each pixel in the input image needs to be read R times from external memories for the R consecutive windows in the column-wise direction. And this leads to a sharp elevation in external memory bus bandwidth. In the SWPB scheme, a total of R pixels must be fetched from FIFOs, and eventually from external memories, in a single cycle to keep the 1 clock/pixel throughput rate. The resulting external memory bus bandwidth requirement is R pixels/clock.

D. MWPB Scheme

The basic idea of the multiwindow (MW)PB scheme we propose is to reuse data that are already stored in internal buffers as many times as possible. It is known that two neighboring windows in the column-wise direction share $R - 1$ common rows, therefore, when pixels from $R + S - 1$ raster lines are loaded into internal buffers, a total of S neighboring windows can be assembled, as shown in Fig. 4. If the calculations for these S windows are performed at the same time, a much higher level of data reusing will be achieved compared with SWPB schemes.

Fig. 5 illustrates the MWPB scheme we proposed. The number of shift register arrays is extended to $R+S-1$ to hold all the pixels in the $(R+S-1) \times S$ area as depicted in Fig. 4. Unlike the full-buffering scheme and the SWPB scheme, the S pixel data in each set of shift register array are not simultaneously fed to the 2-D convolver, but in a serial manner instead. Only one register in the shift register array is accessible in each cycle, and a rotationally incremented pointer is used to address the output register. Therefore, a total of $R + S - 1$ pixels of a same column in the input image, belonging to S neighboring windows in the column-wise direction, are provided to the 2-D convolver in each cycle. After S cycles, all the data in the current $(R + S - 1) \times S$ area have been fed to the 2-D convolver, and the shift register arrays will then be updated. A new column of data will be shifted in from the FIFOs and effectively moves the $(R + S - 1) \times S$ area to a next position.

The architecture for the 2-D convolver using MWPB strategies may be a little different from that under other aforementioned buffering strategies (see Fig. 6). For each $R \times S$ convolution window, data are fed in a column-by-column manner, therefore a single 1-D convolution module can be employed, and it will take S cycles to complete the calculation for each window. Since there are S neighboring windows available, a total of S independent 1-D convolution modules can be used. With pipelined arrangement, a throughput rate of 1 clock/pixel can still be achieved.

For the MWPB scheme, multiple dataflow must also be provided to update the convolution window. But unlike the SWPB scheme, the convolution window in the MWPB scheme is updated every S cycles, which means that the shift register arrays work in a much lower frequency (shift every S cycles). Therefore, a total of $R + S - 1$ pixels will be fetched from external memories every S cycles, and the resulting memory bus bandwidth is $(R+S-1)/S$ pixels/clock. For most 2-D convolution masks, this means only an approximately 2 times increase in external memory bus bandwidth compared with FB schemes.

One disadvantage for the MWPB scheme is that output pixels are no longer in the raster scan format. Instead, a column-major zigzag scan format will be generated. Row-major zigzag scan path may also be generated by making some modifications to the buffering scheme shown in Fig. 5 so that the each FIFO will contain column-wise data.

III. Performance Analysis

A. Case Study

When choosing a buffering strategy for FPGA-based 2-D convolvers, cost functions such as area utilization and external memory bus bandwidth, with respect to throughput data rate, must be evaluated. In Table I, we have summarized the main features of the three buffering schemes discussed in the above section: throughput, given in terms of clocks/pixel; area utilization, measured in terms of shift registers in the window area and the memory pixels for delay-lines and FIFOs (FIFOs are assumed to have a depth of P for all the buffering schemes); and external memory bus bandwidth requirements, given in terms of pixels/clock.

For a case study, we assumed an input image size of 1024×1024 and a convolution mask size of 5×5. Images were read from external memories, and stored back to external memories after the 2-D convolution. We considered a 32-bit SRAM as the external memory, and a single read (or write) operation will fetch (or store) 4 byte-size pixels from (or to) the external memory. A FIFO depth of 8 i.e., $P = 8$, was chosen for all the buffering schemes. Shift registers, FIFOs and delay-lines can be implemented with different resources depending on the specific FPGA targeted in the design. But for comparison, they were all measured in terms of flip-flops in the case study. Flip-flop count was obtained by multiplying the number of shift registers and memory pixels by bits per pixel (8 in the case study). We have reported the results of flip-flop

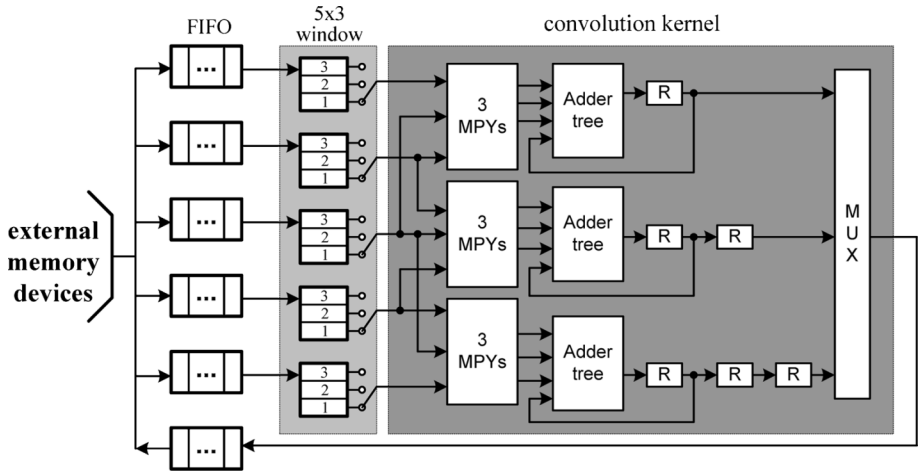

Fig. 6. Architecture of a $R \times S$ convolver using the MWPB scheme with $R = 3$ and $S = 3$. MPY and R in the figure represent multiplier and register respectively.

TABLE I
FEATURES OF DIFFERENT BUFFERING SCHEMES FOR A $R \times S$ CONVOLVER

scheme	throughput (clocks/pixel)	area utilization		bandwidth (pixels/clock)	case study	
		shift registers	memory pixels		area utilization (ff count)	bandwidth (pixels/clock)
FB	1	$R \times S$	$(R-1)\times(N-S)+P$	1	32872	1
SWPB	1	$R \times S$	$R \times P$	R	520	5
MWPB	1	$(R+S-1)\times S$	$(R+S-1)\times P$	$(R+S-1)/S$	936	1.8

count and external memory bus bandwidth requirement for the case study in the last two columns of Table I. It should be noted that although the SWPB scheme is most area-efficient, the external memory bus bandwidth requirement of it, however, can't be satisfied in the case study. As a result, the actual throughput rate of it will become 1.5 clocks/pixel (considering an output bandwidth requirement of 1 pixel/clock).

FPGA implementations were also done and 5×5 convolvers with three different buffering schemes were realized on a low-cost XILINX Spartan-3 XC3S1000 FPGA [7] using XILINX Integrated Software Environment (ISE) 7.1i. The complete architecture for each convolver mainly consists of an input buffer module, a convolution kernel module and an output buffer module. All modules were written using the hardware description language Verilog. Delay lines in the full-buffering scheme were implemented using the RAM-based shift registers (SRL: Shift Register LUT) [6], [7] while FIFOs were implemented using flip-flops. The designs were synthesized using XILINX Synthesis Tool (XST). Hardware resource utilization was obtained and summarized in Table II. For the proposed MWPB scheme, the total hardware resource utilization is 29%, which is only a slight increase compared with the 24% for the SWPB scheme, but a sharp decrease compared with the 46% for the FB scheme.

TABLE II
HARDWARE RESOURCES UTILIZATION FOR THE 5×5 CONVOLVERS WITH DIFFERENT BUFFERING SCHEMES ON A XC3S1000 DEVICE

module	**MWPB**	FB	SWPB
input buffer	**686 slices**	2129 slices	376 slices
convolution kernel	**1562 slices**	1427 slices	1427 slices
output buffer	**40 slices**	40 slices	40 slices
complete architecture	**2290 slices**	3598 slices	1845 slices
percentage for the total occupied slices	**29%**	46%	24%

B. Architecture Selection

In order to choose the optimum architecture for a particular design point, a performance metric that consists in maximizing the throughput with respect to the area utilization may be used. It was proposed in [6] that the product throughput (in terms of clocks/pixel) times flip-flop number is a suitable metric. By minimizing the metric value, a maximum degree of area efficiency will be achieved for this particular design point. We have adopted the same concept in our work. In Table III, we have reported corresponding flip-flop count and throughput rate for window sizes from 3×3 to 15×15 for the three buffering schemes. The remaining variables are the same as described for the case study, and an output memory bandwidth requirement of 1 pixel/clock is assumed. The actual throughput rate is calculated as $\max((\mathrm{BW_{in}}+1)/4, 1)$, where $\mathrm{BW_{in}}$ represents the memory bandwidth requirement of each buffering scheme. In Fig. 7, we have shown the aforementioned metric for the three buffering schemes. We can observe that, for the design parameters as described for the case study, PB schemes are always more area efficient than FB schemes. For window size below or equal

TABLE III
AREA UTILIZATION AND THROUGHPUT RATE OF DIFFERENT BUFFERING SCHEMES FOR VARIOUS SIZE 2-D CONVOLVERS

convolution size	FB		SWPB		MWPB	
	area utilization (ff count)	throughput (clocks/pixel)	area utilization (ff count)	throughput (clocks/pixel)	area utilization (ff count)	throughput (clocks/pixel)
3 x 3	16472	1	264	1	440	1
5 x 5	32872	1	520	1.5	936	1
7 x 7	49272	1	840	2	1560	1
9 x 9	65672	1	1224	2.5	2312	1
11 x 11	82072	1	1672	3	3192	1
13 x 13	98472	1	2184	3.5	4200	1
15 x 15	114872	1	2760	4	5336	1

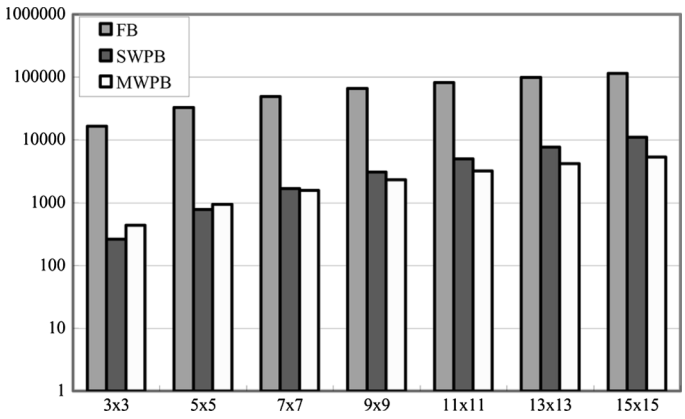

Fig. 7. Bar diagram comparing the area efficiency metric for different buffering schemes and for window sizes from 3×3 to 15×15 using the parameters of the case study. The lower the bar, the more efficient.

to 5×5, SWPB scheme is superior; while for larger window size, MWPB scheme is the optimum one.

It should be noted that area utilization for PB schemes depends mainly on the depth of FIFOs (see Table I). If SRAMs are used as external memories to hold image data, a small FIFO is generally adequate, as shown in the case study. But if low-cost DRAM devices are used as external memories, the situation will become a little more complicated. DRAM devices have a rather long latency time for read and write operations when random accesses are performed. If data in different rows are to be accessed consecutively, a so-called page-miss penalty will be incurred. In this case, a large FIFO that allows a single transaction to fetch as many data as possible within a row in the DRAM device could effectively improve the overall memory bandwidth efficiency. But as FIFOs get larger, resource utilization will also arise abruptly. Tradeoffs must be made, depending on FPGA resources and available external memory bandwidth as well.

IV. CONCLUSION

In this brief, we have presented a MWPB scheme for FPGA-based 2-D convolvers. Comparisons are made between the proposed technique and prior-arts. The new buffering strategy uses much less FPGA resources compared with FB schemes, and at the cost of a much lower elevation in external memory bus bandwidth compared with SWPB schemes. The good balance between on-chip resource utilization and external memory bus bandwidth exhibited by the proposed buffering strategy makes it suitable for low-cost FPGA implementation.

ACKNOWLEDGMENT

The author would like to thank the associate editor and the reviewers for helpful comments that greatly improved this brief.

REFERENCES

[1] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 2002.
[2] C. Torres-Huitzil and M. Arias-Estrada, "Real-time image processing with a compact FPGA-based systolic architecture," *Real Time Imaging*, vol. 10, no. 3, pp. 177–187, Jun. 2004.
[3] B. Bosi, G. Bois, and Y. Savaria, "Reconfigurable pipelined 2-D convolvers for fast digital signal processing," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 7, no. 3, pp. 229–308, Sep. 1999.
[4] S. Perri, M. Lanuzza, P. Corsonello, and G. Cocorullo, "A high-performance fully reconfigurable FPGA-based 2-D convolution processor," *Microprocess. Microsyst.*, vol. 29, pp. 381–391, 2005.
[5] X. Liang, J. S. N. Jean, and K. Tomko, "Data buffering and allocation in mapping generalized template matching on reconfigurable systems," *J. Supercomput.*, vol. 19, no. 1, pp. 77–91, 2001.
[6] F. Cardells-Tormo and P. Molinet, "Area-efficient 2-D shift-variant convolvers for FPGA-based digital image processing," *IEEE Trans. Circuits. Syst. II: Exp. Briefs*, vol. 53, no. 2, pp. 105–109, Feb. 2006.
[7] Xilinx Inc. , Spartan-3 FPGA Family: Complete Datasheet [Online]. Available: http://www.xilinx.com/