RECONFIGURABLE, REAL TIME, SINGLE CHIP CONVOLUTION DEVICE FOR IMAGE PROCESSING

Sergiu Nedevschi, Octavian Creţ, Zoltan Baruch

Technical University of Cluj-Napoca, ROMANIA, 24-26 Bariţiu St., Cluj-Napoca
Sergiu.Nedevschi@cs.utcluj.ro, Octavian.Cret@cs.utcluj.ro, Zoltan.Baruch@cs.utcluj.ro

Abstract: This paper continues a previous work on the design and implementation of a real time, single chip, generalized convolution device for image processing. The objective of this work is to improve the effectiveness of the previous solution by reducing the complexity and introducing reconfigurability features. The results achieved are based on the features offered by the XILINX XC4000 FPGA circuit family, the most important among them being the distributed memory matrices. This allows the use of the serial distributed arithmetic in implementing a reconfigurable convolution device in a single FPGA chip.

Keywords: real time image processing, reconfigurable convolution device, serial distributed arithmetic, matrix multiplication, RAM based shift registers, XILINX FPGA.

1. INTRODUCTION

The architecture of the convolution device working with a 3×3 kernel that was implemented in [1] is shown in Figure 1.

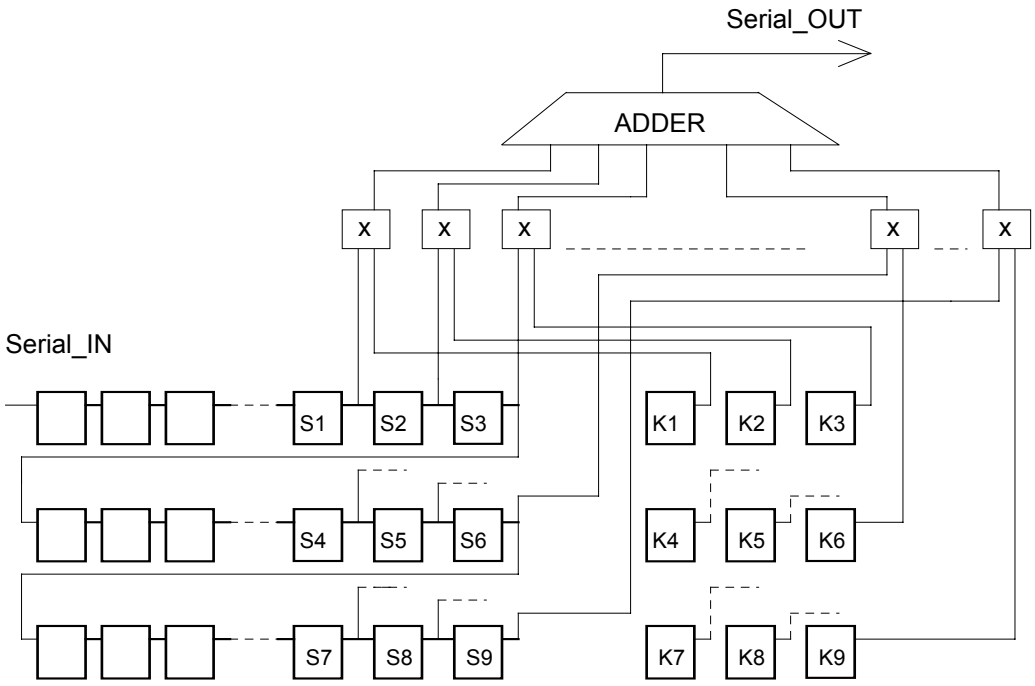

Figure 1. The architecture of the 3×3 convolution system.

A conventional implementation approach implies a high number of circuits. Since the purpose was a single chip, compact, easy reconfigurable and real time working implementation, a FPGA based solution was chosen. This option determined the reorientation of the design towards exploiting the FPGA families features.

The most important advantage of the XC4000 family used in the implementation is the double functionality of the elementary CLB cells as logical function generators or as RAM or ROM memory matrices. The available memory at a CLB cell level is organized as a 32×1 block or two 16×1 blocks. The access time to these memories is under 10 ns. The existence of these memory blocks, distributed at the whole FPGA chip level, allowed the implementation of the three chains of 8-bit serial shift registers storing three image lines, the achievement of a serial distributed arithmetic for bit serial manner parallel multiplication of the neighborhood pixels with kernel coefficients, and in the same time the summing of the partial products [1], [4], [5].

The block diagram of the generalized convolution device [1] is shown in Figure 2.

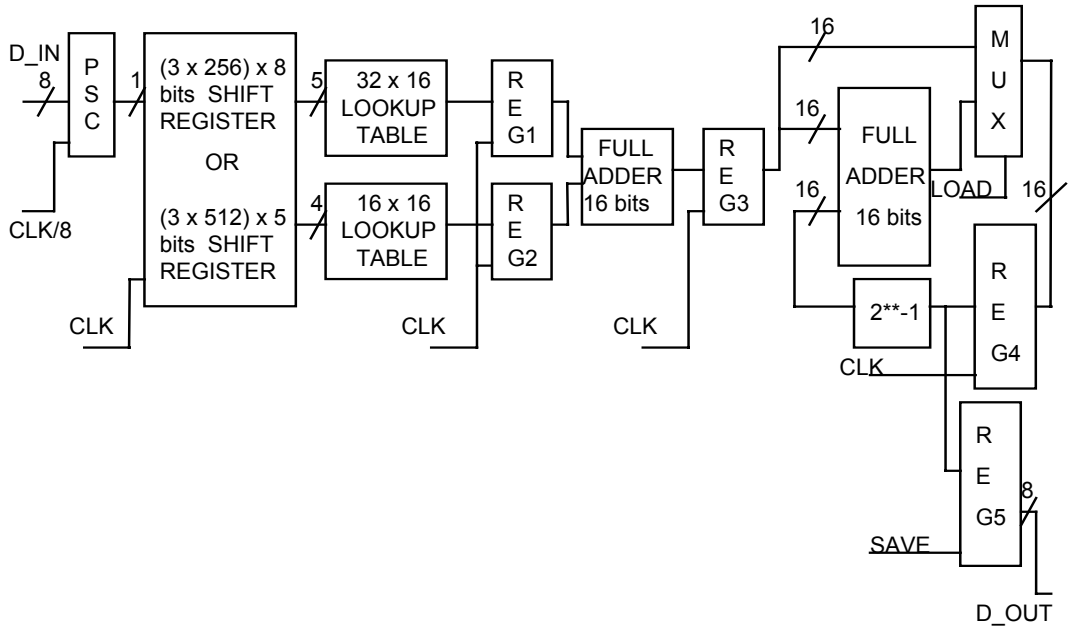

Figure 2. Block diagram of the generalized convolution device.

2. THE PROPOSED ARCHITECTURE

The objective of this work is to improve the effectiveness of the previous solution by reducing the complexity and introducing reconfigurability features.

The analysis of the previous solution leads to the conclusion that the chain of 8-bit shift registers, although it offers significant advantages, consumes a great amount of FPGA resources. Finding a better solution to this problem could save these resources and would allow the implementation of new facilities as reconfigurable kernels.

The replacement of the chain of 8-bit shift registers is possible only if the source image memory is fast enough to supply data at a rate that is comparable to the shift registers pipe rate.

A mixed solution is proposed, in which a pipe of 8-bit shift registers having the dimension of the convolution kernel is maintained, and an address generation logic able to load from the source image memory the needed operands is added. This solution frees a large amount of resources and allows the implementation of a set of dynamically reconfigurable convolution kernels.

2.1 The convolution device

The chain of 8-bit shift registers that allowed access, in each clock period, to a pixel neighborhood, must be replaced by an 8-bit registers matrix having the same dimension as the processed neighborhood, a supplementary column of 8-bit registers used as an *input buffer* and an *address generator* able to load a new column of pixels from the source image memory in each convolution step.

In the case of a 3×3 neighborhood, the three chains of 8-bit shift registers [1] are eliminated and replaced by nine 8-bit shift registers that store only the exact neighborhood of the currently processed pixel and three supplementary 8-bit shift registers used as an input buffer.

For ensuring the access to a new neighborhood, in each processing step, a left-to-right shift of the matrix elements and a device able to provide new input elements is necessary.

The general block diagram of the system is shown in Figure 3. The convolution device contains three main functional blocks: the *address generation logic*, the *serial distributed matrix multiplier* and the *dynamically reconfigurable convolution kernel*.

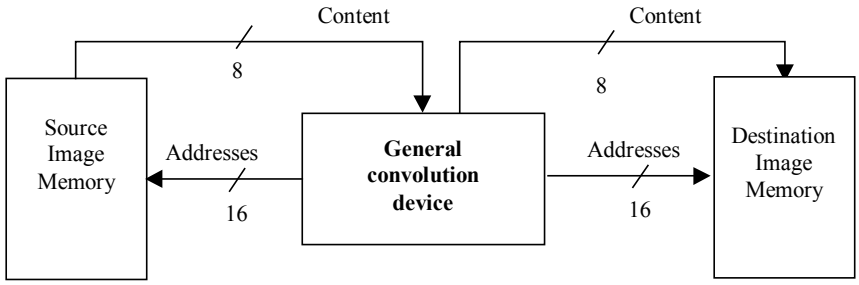

Figure 3. Block diagram of the general convolution system.

2.2 The address generation logic

This logic will read three new elements per cycle from the source image memory using the following address generation algorithm: (0, 256, 512), (1, 257, 513), (2, 258, 514), etc. (for a 256×256 pixels image). A pixel identified by its Column and Row address is located in the memory at the following address:

$$Current\_Address = Column\_Counter \cdot 256 + Row\_Counter \quad (1)$$

The block diagram of the address generation logic is shown in Figure 4. It generates the following three addresses for loading, in each processing step, a new neighborhood-sized column in the input buffers (R_0, R_1, R_2):

$$\begin{aligned} Addr0 &= Column\_Counter \cdot 256 + Row\_Counter \\ Addr1 &= (Column\_Counter + 1) \cdot 256 + Row\_Counter \\ Addr2 &= (Column\_Counter + 2) \cdot 256 + Row\_Counter \end{aligned} \quad (2)$$

The *Address_Multiplexer_Counter* N_3 is used to select the addresses (located on the busses *Addr0*, *Addr1* and *Addr2*) that access the source image memory. These addresses access pixels located on consecutive rows of the image. The N_2 and N_1 counters are used to build the addresses according to the equations (2).

The multiplication by 256 is implemented by a concatenation of the *Column_Counter* (N_2) bus (on the most significant 8 bits) with the *Row_Counter* (N_1) bus (on the least significant 8 bits of the *Addr* busses).

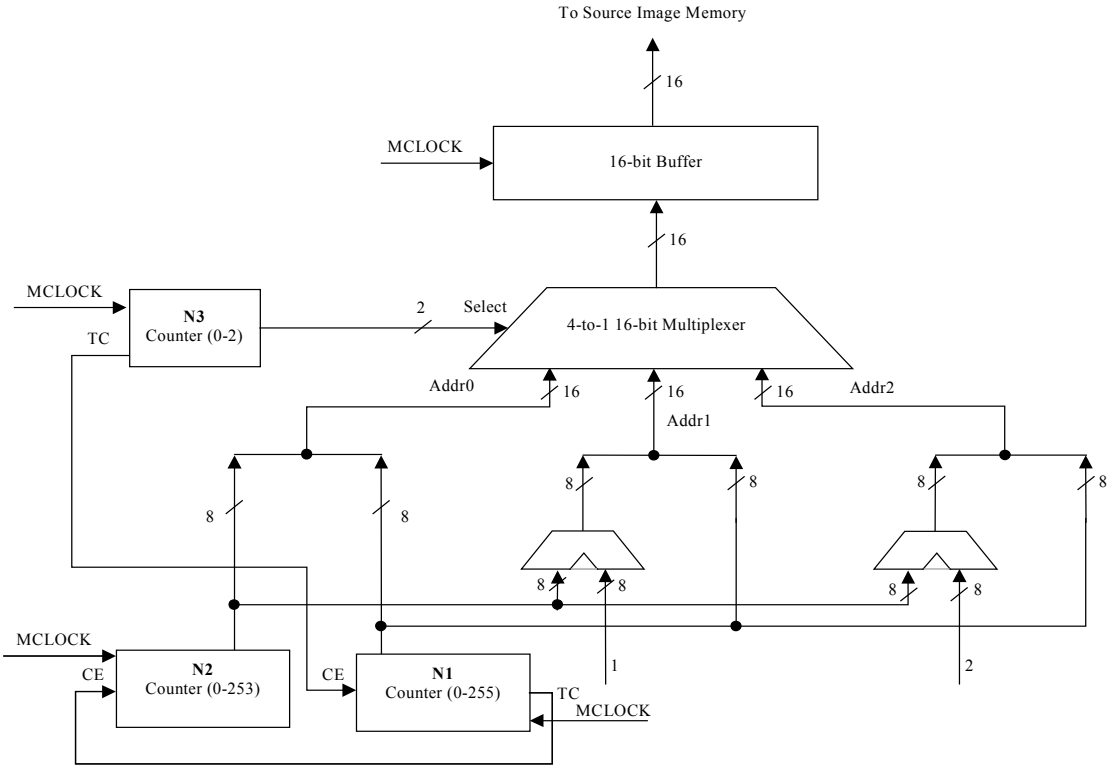

Figure 4. The address generation logic.

Two supplementary 8-bit full adders are necessary to ensure the correct access to the desired pixel, according to the equations (2).

The ranges for the *Address_Multiplexer_Counter* (N_3), the *Column_Counter* (N_2) and the *Row_Counter* (N_1) are [0-2], [0-253] and [0-255], respectively. The counters are cascaded using the *Terminal Count* (*TC*) and *Clock Enable* (*CE*) signals. The MCLOCK period must be equal to or greater than the memory access time (T_{mem}).

2.3 The serial distributed matrix multiplier block

The two-dimensional convolution between an image neighborhood and a convolution kernel can be represented by an one-dimensional sum of products:

$$\sum_{i=1}^{m^2} S_i \cdot K_i = \sum_{i=1}^{m^2} [\{...\{(S_{i,0} \cdot K_i)2^{-1} + (S_{i,1} \cdot K_i)\}2^{-1} + \\ + ... + (S_{i,n-1} \cdot K_i)\}2^{-1} + (S_{i,n} \cdot K_i)] \quad (3)$$

where: S_i represents the neighborhood elements, K_i represents the kernel coefficients, m^2 represents the number of pixels in a neighborhood, $S_{i,j}$ represents the j-th order bit of the S_i pixel, $j \in [1,...,n]$, n represents the number of bits of a pixel.

The j-th order term can be developed as the following sum of products:

$$\sum_{i=1}^{m^2} S_{i,j} \cdot K_i = S_{1,j} \cdot K_1 + S_{2,j} \cdot K_2 + ... + S_{m^2,j} \cdot K_{m^2} \quad (4)$$

The $S_{i,j}$ terms represent the j-th order bits of the neighborhood elements, taking 0 or 1 values. The K_i terms represent the constant coefficients of the convolution kernel.

For a given set of coefficients these sums of products can be memorized in a ROM used as a lookup table. The access to this lookup table will be made using the j-th order bits of neighborhood pixels.

Figure 5 shows the block diagram of such a multiplier [1]. The use of the shift registers allows simultaneous access to all the bits of the same order of the current neighborhood pixels. These bits are used to address the partial sums in the lookup table.

The final result of the matrix multiplication is obtained by accumulation of the sums of partial products in a number of steps equal to the number of bits in a pixel.

The matrix multiplication period T_p is determined by the number of bits used for pixel representation n, and the period in which a step of the multiplication algorithm is performed, T_c:

$$T_p = n \cdot T_c \tag{5}$$

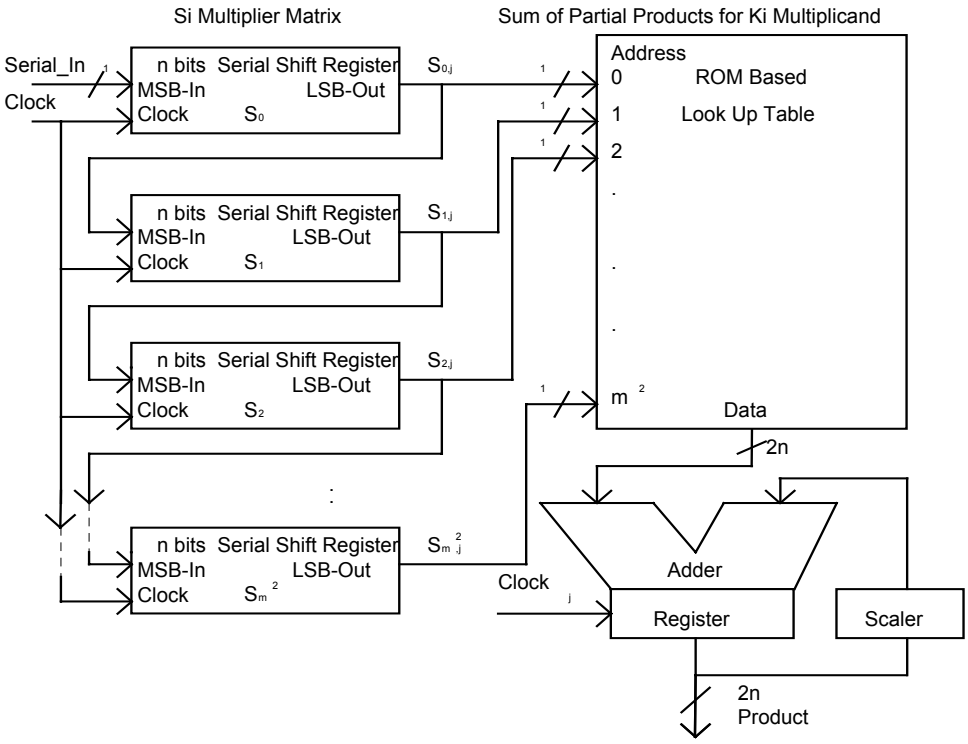

Figure 5. Block diagram of the serial distributed matrix multiplier [1].

The convolution products are computed in 8 clock cycles, using the lookup table to stores the partial products. For scaling the result with the sum of the convolution kernel terms, K_i, the values stored are previously divided by $\left|\sum_{i=1}^{m} K_i\right|$. This is possible because all these values are smaller than 1, which makes the decimal point to be in the same position. A two's complement fixed point arithmetic with an 8 bit fractional part is used for normalized coefficients representation. The final result will be an integer on 8 bits.

ADDRESS	**DATA**
0...000	0...000
0...001	K_1 / Σ
0...010	K_2 / Σ
0...011	$(K_1 + K_2) / \Sigma$
0...100	K_3 / Σ
0...101	$(K_1 + K_3) / \Sigma$
0...110	$(K_2 + K_3) / \Sigma$
0...111	$(K_1 + K_2 + K_3) / \Sigma$
…	…
1...111	$(K_1 + K_2 + ... + K_m) / \Sigma$

Figure 6. The contents of the lookup table.

Figure 6 shows the contents of the lookup tables, where Σ represents $\left|\sum_{i=1}^{m} K_i\right|$.

The registers *R*0, *R*1, *R*2, *S*1, *S*4 and *S*7 (Figure 7) are implemented with flip-flops (because a parallel load operation is necessary), while for the remaining shift-registers (*S*2, *S*3, *S*5, *S*6, *S*8 and *S*9) we can use the previous solution: the use of the Select-RAM modules as shift-registers [1].

The internal structure of the convolution kernel is shown in Figure 7.

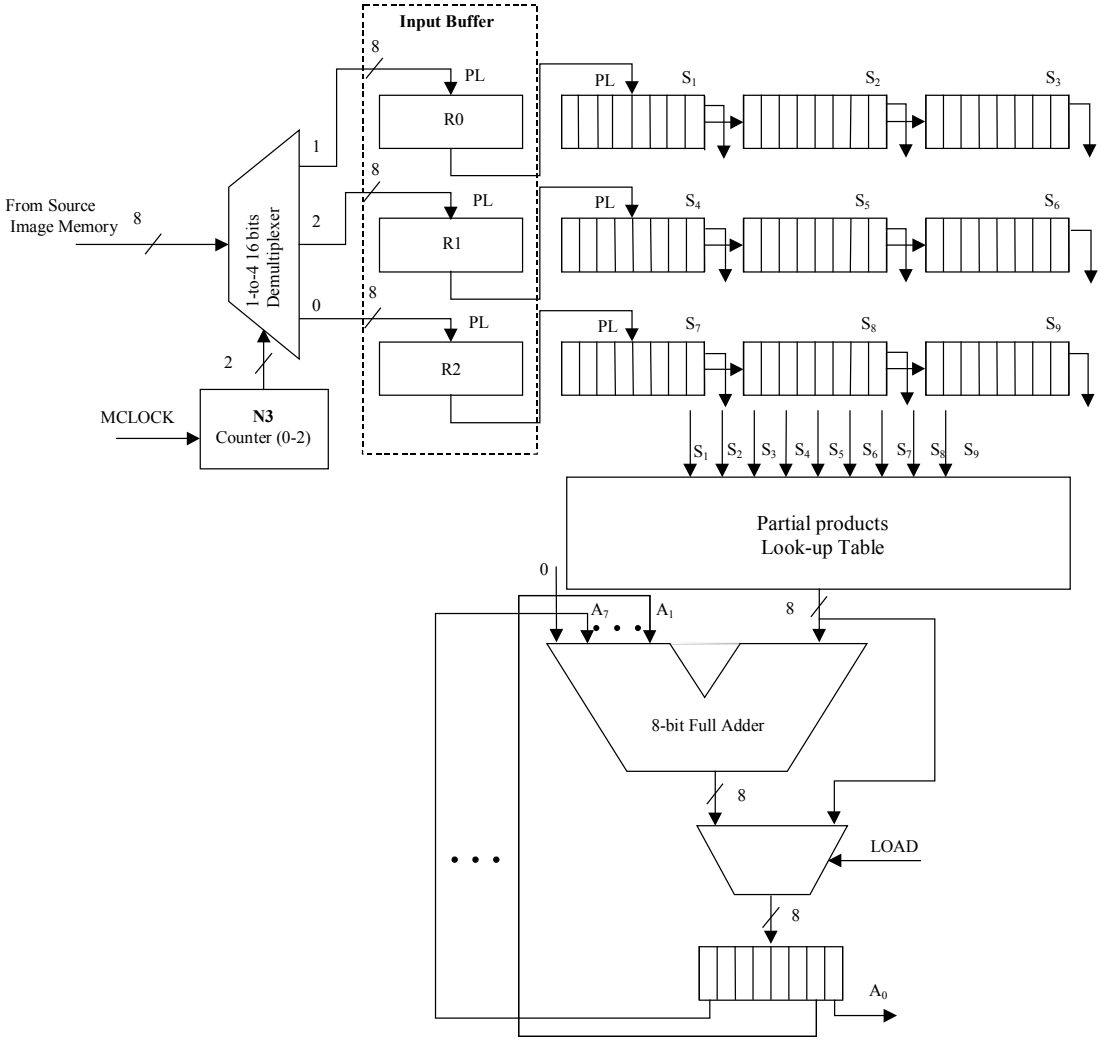

Figure 7. The internal structure of the convolution kernel.

2.4 Implementation details

Because the serial distributed arithmetic was used for matrix multiplication, the access to the 9 neighborhood pixels is serial, starting with the LSBs.

The sum of partial products for each order of the current neighborhood pixels must be stored in the lookup table. The 9 pixels imply a 9-bit address bus, and in consequence a lookup table with 2^9 locations. In order to reduce the size of the lookup table, the address vector is split into two, 5-bit and 4-bit fields. This leads to the use of a memory with 2^5 locations for storing the sums of the partial products for the first 5 terms, and the use of a memory with 2^4 locations for storing the sums of the partial products for the last 4 terms. Also, it is necessary to append an adder to calculate the total sums (Figure 8).

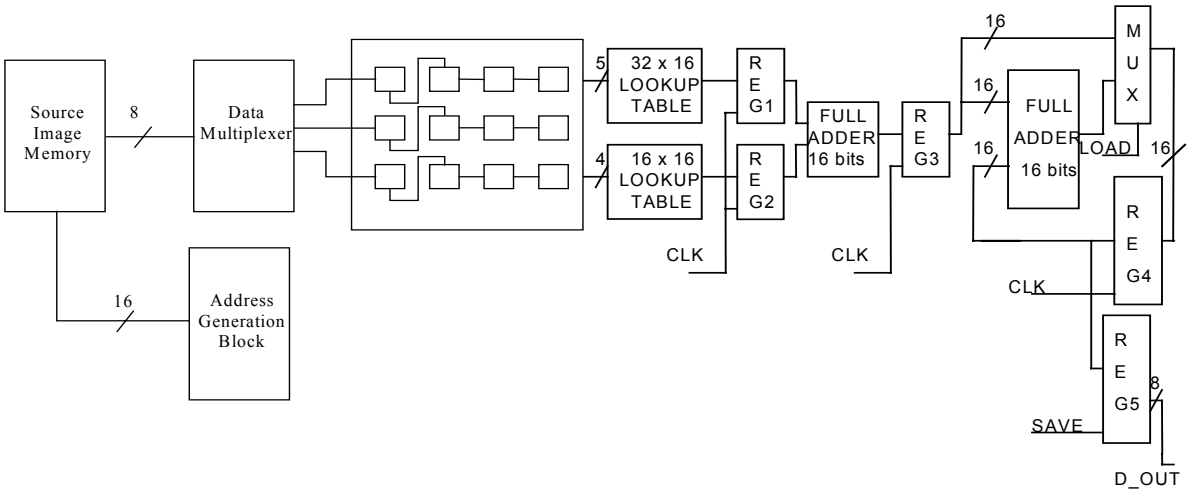

Figure 8. The schematic of the 3×3 convolution device data path.

The timing diagram of the system is presented in Figure 9. Assuming T_{mem} = 80 ns (corresponding to MCLOCK), the maximum clock period is 30 ns (corresponding to CLOCK).

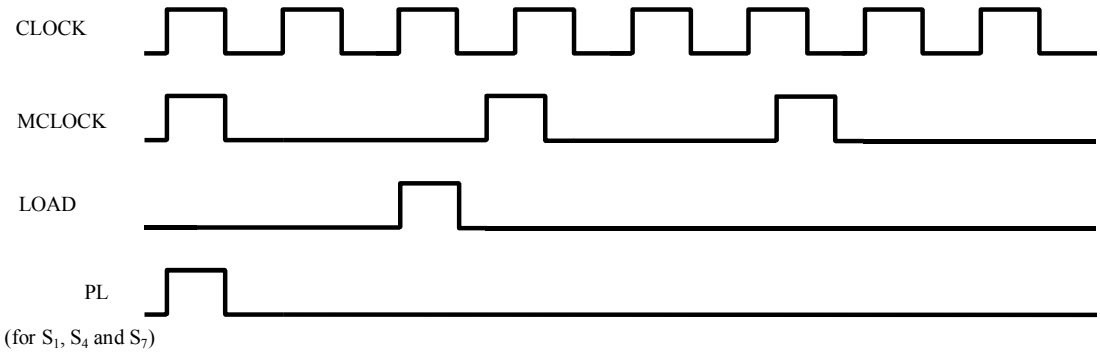

Figure 9. The time diagram of the control block signals.

Because this pixel period is not long enough for the execution of all algorithm step operations (as the access of the sums of the partial products, their summing, the final accumulation), a pipeline solution is necessary for parallel execution of operations. Thus, REG1 and REG2 registers are introduced for storing the partial sum of partial products, REG3 for storing the sum of partial products, and REG4 for accumulating the sum of partial products. The command block associated with this data path generates the signals shown in Figure 9.

2.5 Reconfigurability opportunities

With all these optimizations, the available silicon area increases considerably, allowing the implementation of a reconfigurable system in the following way. Several convolution sets of partial products, corresponding to different convolution kernels, can be stored in a separate configuration memory. For implementing the desired convolution, the corresponding set of partial products can be used to reconfigure the lookup table.

A second variant consists of storing the different kernels in separate lookup tables and to multiplex the kernel being computed.

This way, the system becomes *dynamically reconfigurable*, all the configurations being stored on-chip and the switch between configurations being achieved at request in very short time, compared to the time necessary for a full reconfiguration in ordinary XC4000 FPGA devices.

2.6 Experimental results

The presented scheme was implemented and tested on a XILINX PC development system. The test was successfully carried out for a 3×3 convolution kernel applied to a 256× 256 image matrix.

The reconfigurability capabilities are still under development.

The use of the Virtex-EM family FPGA devices allows a faster implementation (the source and destination image memories being stored on-chip, with considerably shorter access times).

3. CONCLUSIONS AND FUTURE DEVELOPMENTS

The XILINX XC4000 family features and the chosen design solution allow the development of a real time, single chip, generalized convolution device for image processing.

Future developments can be achieved in the following directions.

- The growth of the spatial resolution to 512×512 pixels. This would imply only small changes of the address generation block, the serial distributed matrix multiplier block remaining unchanged.
- The possibility to choose between different convolution kernels. This can be achieved by storing in PROM memories the sums of partial products for different kernels and creating the possibility to select the corresponding PROM for a desired kernel, making the system dynamically reconfigurable.
- The real time execution of complex processes by serially linking several convolution devices and selecting the corresponding convolutions sequence. In a Virtex–EM FPGA device, all the convolution kernels can be stored on the same chip and the entire computation can be realized by a unique device.

REFERENCES

[1] Nedevschi, S., Samways, P., Marian, M., Hall, T. (1996): Real-time, Single-chip, Generalised Convolution Device for Image Processing. *ACAM Scientific Journal*, Vol. 5, No.1, pp.11-22.

[2] Sternberg, S. R. (1982): Pipeline Architectures for Image Processing, *Multicomputers and Image Processing-Algorithms and Programs*, L. Uhr, ed., Academic Press, pp. 291-305.

[3] Pratt, W. K. (1991): Digital Signal Processing, Second Edition, Wiley Interscience Publication.

[4] Mintzer, L. (1987): Mechanization of Digital Signal Processors, Handbook of Digital Signal Processing, pp. 941-973.

[5] Goslin, G. (1995): Using XILINX FPGAs to Design Custom Digital Signal Processing Devices, *Xilinx University Program Workshop*, Fall 1995, XILINX Inc.

[6] Goslin, G., Newgard, B. (1995): 16-Tap, 8-Bit FIR Filter Application Guide, *Xilinx University Program Workshop*, Fall 1995, XILINX Inc.

[7] Alfke, P. (1995): Efficient Shift Register, LFSR Counters and Long Pseudo-Random Sequence Generators, *Xilinx University Program Workshop*, Fall 1995, XILINX Inc.