

FULL-PRECISION BIDIRECTIONAL BIT-SERIAL CONVOLVER for REAL-TIME IMAGE PROCESSING

Lior Djaloshinski, Orly Yadid-Pecht and Ran Ginosar

Dept. of Electrical Engineering
Technion - Israel Institute of Technology
Haifa 32000, Israel

Abstract - A novel chip implementing real-time image convolution is presented. The chip features a pure bit-serial architecture and systolic array structure. It consists of a 3X3 array of modular units that can be easily enlarged. The basic unit heart is a very efficient bidirectional multiplier, which can perform two different multiplications simultaneously. The chip overall size is 5.78mm X 3.18mm (in CMOS 2 μ technology). It is full-precision and has a throughput of one 20 bit convolution every 10 clock cycles, with a latency of 24 clock cycles.

1. Introduction

In this paper we present a VLSI chip for real-time image processing, namely convolution of a TV picture. Originally, the convolver was designed for installation in a specific image processing system of intelligent scan type, but it can be used generally wherever convolution is desired, provided the I/O format is adequate.

Real-time convolution is a computationally intensive operation. A standard (non interlaced) TV picture has 512X512 pixels, and a rate of 30 frames per second. This means that real-time operation requires that each pixel will be fully processed in 125 ns at most. In the case of a convolution with a 3X3 kernel, for instance, multiplication of 9 pixels by 9 weights and summation of the products should be done during this period. In order to cope with this massivity of information the choice of algorithm and implementation is critical. We have chosen the convolver to feature a pure bit-serial architecture and a systolic array structure.

In a pure bit-serial architecture data is fed in, processed, and taken out serially, bit by bit, and the basic computational unit works on the bit level - thus being naturally pipelined to the single bit level [1,2]. By using such an architecture the area occupied by communication paths is greatly reduced (in comparison to word-parallel architectures) and the long carry propagation delays of parallel processors are avoided. Wires are fully utilized and data is circulated in small loops - hence the highest bit rate feasible within technological limitations can be achieved.

Systolic arrays work in a pipelined fashion, very much like the classic industrial mass production line. Each bit is processed by many different units at different time slots and many bits are processed simultaneously. Among their advantages is the ability to speed up compute-bound computations (such as convolution) without increasing the I/O bandwidth - owing to their fully concurrent operation [3]. Systolic structures and bit-serial architecture fit excellently one to another and form an array pipelined to the single bit level.

Another feature that improves the throughput is the use of very efficient multipliers. The multipliers are the heart of the convolver and have dominant influence on its performance. Therefore, special attention was paid to their design. The multipliers are bit-serial and full-precision. Each of them consists of eight simple and similar basic modules. Every multiplier performs two multiplications simultaneously and thus its modules can be utilized 100% of the time.

The result is a full-precision convolver (no approximations or truncations performed), with a high throughput (one 20 bit convolution every 10 clock cycles) and a latency of 24 clock cycles. Nevertheless, the chip is relatively small and compact, thanks to an efficient utilization of the area and a high fill factor. For a 3X3 convolver, implemented in CMOS 2 μ technology, the chip size (including pads) is 5.78 mm X 3.18 mm. Communication with the outside world is via three input, two output and two control pins only - owing to the bit-serial methodology.

Because of the convolver small size, small number of pins and fully-modular and regular design, it can easily be enlarged, both in the

array size (kernel size) and the words length (number of bits used to represent the pixels and weights). Alternatively, a few similar convolvers can be implemented on the same chip and process different parts of the picture in parallel, in order to speed up the operation.

Section 2 presents the general concepts of the convolver structure and operation. Section 3 describes the main component of the convolver basic cell - the multiplier. Section 4 discusses the rest of the components composing the convolver - namely the memory, addition and representation conversion, delay and control units. Section 5 regards implementation, and Section 6 summarizes the paper.

2. The Convolver Design Concepts

The convolver consists of 9 similar basic cells, arranged in a 3X3 array, and a general control unit. The convolver array is composed of three linear systolic arrays, placed and operating in parallel, as shown in Fig. 1. Data flows "horizontally" from cell to cell, while the outputs of each of the basic cells flow "vertically". While flowing, the partial results are summed-up and fanned-in towards the convolver output.

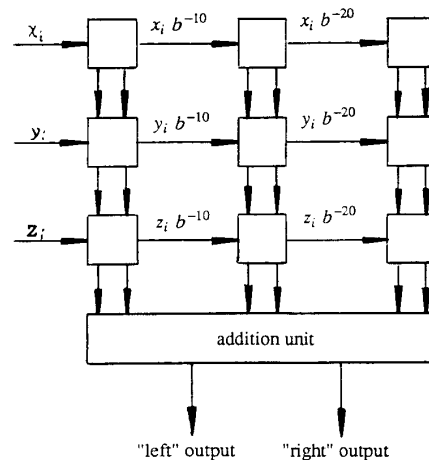


Fig. 1: The convolver array structure.

In every computation cycle the convolver should multiply 9 pixels values by 9 weights and sum the products. As the system scans the picture the pixels change, while the kernel of weights remains constant throughout the computation. Each of the convolver basic cells is responsible for the multiplication of one pixel with one weight in a given convolution cycle. For that purpose a multiplier is included in each basic cell. Since the kernel of weights is constant it is advantageous to preload this kernel to the convolver, one weight per cell, and store it there for the entire computation (until a different kernel is desired). Therefore, a memory unit is also included in every basic cell. The I/O format is determined by the system which the convolver serves. The pixels are represented by an 8-bit positive binary number, the kernel weights - by an 8-bit binary number in "sign and magnitude" format, and the convolution result should be represented in "2's complement" format. A representation conversion unit takes care of this issue. A block diagram of the convolver basic cell is shown in Fig. 2. The function of each block is discussed below.

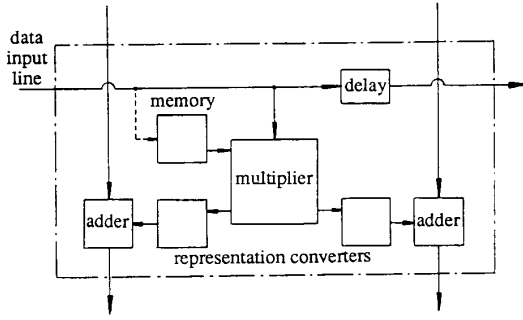


Fig. 2: Convolver basic cell - block diagram.

3. The Multiplier

The multiplier is the core of the convolver basic cell. It is also the most complicated unit in the basic cell and occupies most of its area. Therefore, it is especially important to choose an appropriate algorithm and implementation for the multiplier, in order to reduce its area and accelerate its operation as much as possible.

Most of the bit-serial multiplier implementations found in literature are based on one (5,3) FA (full-adder) in each of the multiplier basic cells. Exceptional is Buric and Mead's [4], which is based on two (3,2) FAs in each cell, working in parallel (A (m,n) FA is one which adds m equally weighted inputs and produces an n-bit output - one sum bit and n-1 carry-out bits of different orders (m>n of course)). We found it advantageous to use (3,2) FAs instead of (5,3) ones for several reasons. To begin with, using two (3,2) FAs enables us to divide the multiplier basic cell into two identical and simpler cells - what contributes to a more modular, regular, and simple design. Since the two (3,2) adders work in parallel, their propagation delay time is that of one regular (3,2) FA, which is shorter than that of a (5,3) one. Two (3,2) bit-serial adders also consume less area than a (5,3) one.

We have adopted Buric and Mead's approach for implementing the multiplier, with slight modifications. Yet we have improved it by making it operate bidirectionally [5], thus increasing its efficiency.

The product of two n-bit binary numbers

$$X = \sum_{i=0}^{n-1} x_i \cdot 2^i \quad \text{and} \quad Y = \sum_{i=0}^{n-1} y_i \cdot 2^i \quad (1)$$

$$\text{is: } Z = X \cdot Y = \sum_{k=0}^{2n-2} z_k \cdot 2^k, \quad \text{where: } z_k = \sum_{i=0}^k x_i \cdot y_{k-i} \quad (2)$$

The product can be written as:

$$Z = X \cdot Y = \sum_{k=0}^{2n-2} (z'_k + z''_k) \cdot 2^k$$

$$\text{where: } z'_k = \sum_{i=0}^{\lfloor \frac{k}{2} \rfloor} x_i \cdot y_{k-i} \quad ; \quad z''_k = \sum_{i=0}^{\lfloor \frac{k+1}{2} \rfloor - 1} y_i \cdot x_{k-i} \quad (3)$$

([t] denotes the integer part of t).

The operation can be achieved by a linear pipelined array of n sections, in which the j-th section computes the following:

$$\begin{aligned} z'_j &= x_j \cdot y_k + z'_{j+1} \cdot b^{-1} \quad ; \quad j \leq k \quad ; \quad j = 0, 1, \dots, n-1 \\ z''_j &= y_j \cdot x_k + z''_{j+1} \cdot b^{-1} \quad ; \quad j < k \quad ; \quad k = 0, 1, \dots, n-1 \end{aligned} \quad (4)$$

where b^{-1} denotes a delay of one clock cycle.

The multiplier is composed of eight similar sections, each of which is constructed of two parts - one for performing z' and the other for z'' . Thus, the multiplier array is actually two linear arrays, working in parallel. A bit-serial FA, external to the array, adds the outputs

of the two arrays ejected from the 0-th section, and generates the total product. The logical scheme of the two parts composing the multiplier basic cells is shown in Fig. 3. Each of the two parts consists of a latch, an AND gate and a bit-serial (3,2) FA (parts 3, 5 and 6 in Fig. 3, respectively).

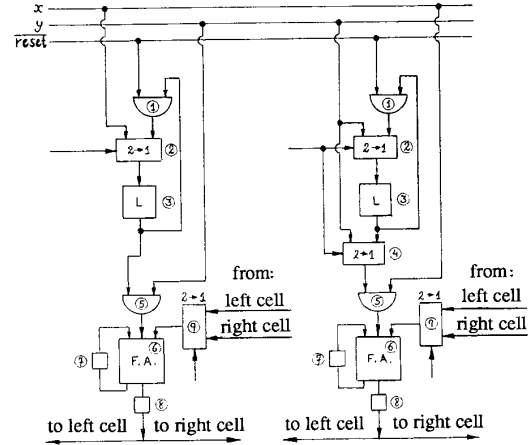


Fig. 3: The two parts of the multiplier basic cell.

The operands X and Y are applied bit-by-bit on two single-bit buses, one bit per clock cycle, LSB first. In the j-th clock cycle of the multiplication x_j and y_j appear on the buses. Therefore, all the single-bit multiplications $x_i \cdot y_j$ and $x_j \cdot y_i$, in which $i \leq j$, are performed in this cycle, and z_j is accomplished one clock cycle later. Also in the j-th cycle, x_j and y_j are latched by the latches of the j-th section for future use. For this purpose a control bit propagates along the array from section 0 to n-1, one section per clock cycle, starting simultaneously with the LSB of the operands (x_0 and y_0). This control bit supplies an enable signal to the j-th section latches in the j-th cycle. The highest order single-bit product, $x_{n-1} \cdot y_{n-1}$, is accomplished in the (n-1)-st cycle in the (n-1)-st section. After this, no more single-bit multiplications need to be done. But all the multiplier sections still hold partial results which should propagate out through section zero within the next n cycles. For this reason we have included a single-bit reset bus and the two AND gates in each section (part 1 in Fig. 3). During the n-th cycle of the multiplication a reset signal is applied and all the latches in all the sections are reset. This causes the outputs of all the AND gates to be zeroed, regardless of the values on the X and Y buses. From now on (until a new multiplication starts) the multiplier basic cells continue to perform:

$$z'_j = z'_{j+1} \cdot b^{-1} \quad ; \quad z''_j = z''_{j+1} \cdot b^{-1} \quad (5)$$

Thus the partial results continue to propagate and are ejected from the 0-th section.

The total result of the convolution can be, in the worst case, a 20-bit binary number. Since the convolver is bit-serial and full-precision, 20 clock cycles are needed for each convolution. However, the 0-th section is busy only 75% of this time, while the other sections are busy less and less time. The (n-1)-st section is busy only 5% of the time. This is a waste of multiplication capability which can be prevented.

As from the 9-th clock cycle of the multiplication the multiplier sections start to become free, one section per cycle, starting from the last (7-th in this case) section towards the first (0-th) section. Hence it is possible to start a new multiplication in the 9-th (or a later) cycle, from the last section towards the first. In this suggested multiplication the direction of propagation is opposite the previous one. Again, in the 9-th cycle of this multiplication it is possible to start a new multiplication with the carry bits propagating in the original direction. In this way all the multiplier sections can be utilized 100% of the time.

The control of the propagation direction is achieved by placing 2→1 multiplexers on the carry lines connecting the multiplier sections (part 9 in Fig. 3). Each of the sections outputs is split into the two nearest neighbours in the array, and drives their multiplexers inputs. The multiplexers determine which of the neighbours outputs will be connected to each of the sections inputs. As we pointed out above, the multipliers sections become free gradually, one section per cycle. This necessitates that the change in the state of the direction multiplexers will also be done gradually, one per clock cycle. The direction control signal which drives the "select" inputs of the direction multiplexers should be of the form shown in Fig. 4a.

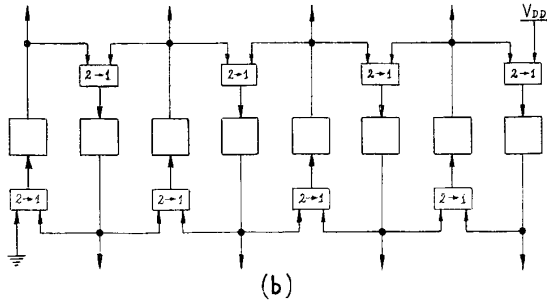
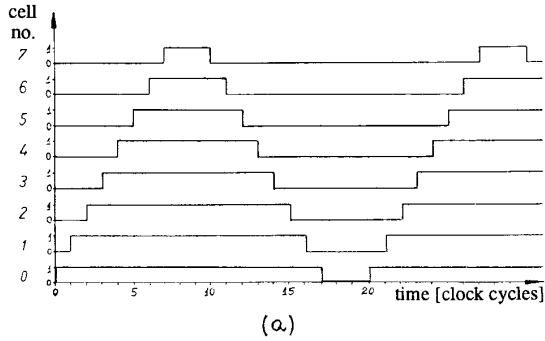


Fig. 4: The direction control signal (a) and its generator implementation (b).

For the implementation of this signal we built the control signals generator shown in Fig. 4b. This generator is an 8-bit bidirectional shift register. The shift direction is determined by the state of the multiplexers. The "select" lines of all the multiplexers are driven together by the counter of the general control unit, and thus the direction control signal will be synchronized with the multiplication cycles.

As we mentioned above, the latches in the j -th section demand a latching signal in the j -th clock cycle of the multiplication. For this purpose a control bus must be added, along which the latching signal propagates. The latching signal should be applied to the multiplier sections successively, one section at a time. For the bidirectional multiplier its form should be as described in Fig. 5a.

The logical scheme of the control bus we built is shown in Fig. 5b. The system is actually a dual 7-bit shift register with its outputs "OR"ed together. The latching control signals come from the outputs of the OR gates. A signal applied to the left input will propagate to the right, and vice-versa. In the complete convolver the left and right inputs of the control bus are driven by the counter of the general control unit, and thus will be synchronized with the multiplication cycles.

This suggested improvement yields an increase of 100% in the convolver throughput, with no effect on the latency. Our estimation for the cost of the bidirectionality is an increase of about 50% in the chip area - i.e., if we would have built a convolver based on unidirectional multipliers in the same configuration, its area would have been about 2/3 of the bidirectional one.

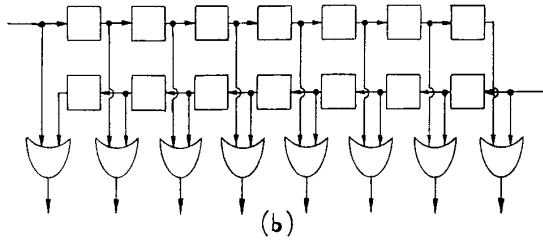
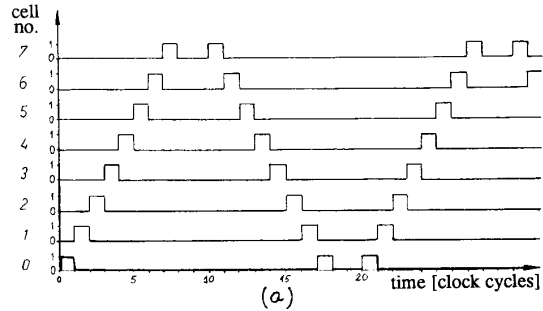


Fig. 5: The latching signal (a) and its control bus (b).

4. Peripheral Components

The Memory Unit

The function of the memory unit is to store the value of one of the kernel weights and feed the multiplier with it bit-serially. Our memory unit is basically a shift register, with several additions. Firstly, the kernel values arrive to the convolver as an 8-bit binary number, represented in "sign and magnitude" format. For the multiplication, however, we treat it as an 8-bit positive binary number with a zero as the MSB. The memory unit separates the sign bit from the magnitude bits and puts the zero instead. The sign bit is used as a signal for the representation conversion unit, for the correct representation of the total product. Secondly, as no computation is performed while a kernel is loaded, and vice versa, the same pin and (single-bit) bus that brings the pixels values can be used to bring the weights values. It only requires the addition of a 2→1 multiplexer. Since a new multiplication starts every 10 clock cycles, the memory unit should feed the multiplier with the weight stored in it at that frequency. The way we accomplished this is by concatenating two more MS bits (actually zeros) to the magnitude bits of the weight, to get a 10-bit positive binary number. These two bits do not affect the multiplication since in the ninth clock cycle of a multiplication the contents of the multiplier latches are reset. Our implementation for the memory unit is shown in Fig. 6.

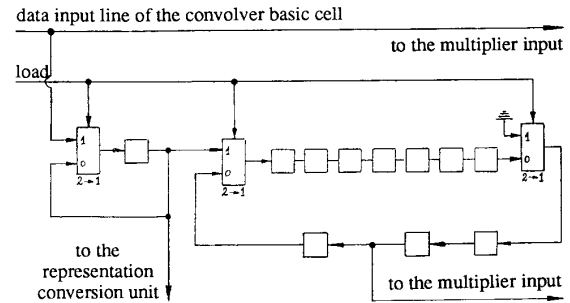


Fig. 6: The memory unit.

The Addition and Representation Conversion Unit

Our approach for the implementation of the multiplier is to multiply positive operands only. Hence, the product is also a positive number. The total result, however, should be represented in "2's complement" format. For that purpose a representation conversion unit is added. The multiplier output is split into two channels. In one channel it flows through an inverter and reaches an adder which adds one to the inverted product. The output of the adder is then the negative of the product coming from the multiplier, represented in "2's complement" format. The other channel is merely a by-pass of the first channel. The two channels are recombined via a 2→1 multiplexor of which the "select" signal is the sign bit of the kernel value. As recalled, this sign bit was stored in the memory unit apart from the magnitude bits.

For reasons of area efficiency we found it advantageous to incorporate the representation conversion unit with the multiplier external adder and the adder of the convolver basic cell into one unit. Since the multiplier performs two multiplications simultaneously, in opposite directions, two units of this kind are needed for each convolver basic cell.

The Delay Unit

The proper timing of the convolver basic cells requires the delay of input bits between neighbour cells of each of its linear arrays. Since a new convolution starts every 10 clock cycles this must be the period of the delay. This function is easily achieved by placing a simple 10-bit shift register on the input bus of each basic cell (see Fig. 2).

The Control Unit

The convolver is governed by a central control unit. This control unit is responsible for supplying all the control signals to all the convolver basic cells and synchronizing their operation.

The heart of the control unit is a circular counter of 20 bits. The counter forms the periodicity of 20 clock cycles. The control signals are produced from the outputs of the counter with the aid of some combinational logic. In addition, the control unit also includes the direction control signals generator and the latching control bus - both of which have been described earlier.

5. Implementation

The convolver was designed using a standard-cell method, with the aid of the PPL software which is a VLSI CAD tool of the standard-cell family [6]. The PPL software provided us with the tools for logic simulation as well. Fig. 7 shows the complete convolver design in PPL representation. The regularity of the design, as well as the high area fill factor are apparent.

The PPL software automatically converts the PPL representation of the circuit into layout, from which a mask set can be prepared. For implementation in a CMOS 2 μ N-well technology the size of the 3X3 convolver chip that we have presented here is 5.78 mm X 3.18 mm. A chip of the convolver is now being fabricated at MOSIS.

6. Summary

In this paper we have presented a novel chip design for image convolution. It was originally designed for a 3X3 kernel but can be easily enlarged to any desired kernel size, as it consists of modular units arranged in a systolic array architecture. Its operation mode is bit-serial, therefore only a few I/O pins are required for the environment interface. The chip has a high throughput due to efficient bidirectional multipliers and can be easily incorporated in any image processing system.



Fig. 7: PPL representation of the convolver chip.

Acknowledgements:

Steve Jacobs from Utah university was of much help with the project submission for fabrication.

References:

- (1) P.B. Denyer, "An Introduction to Bit-Serial Architectures for VLSI Signal Processing", in B. Randell and P.C. Treleaven (ed.), *VLSI Architectures*, Prentice Hall Inc., U.S.A., 1983.
- (2) P.B. Denyer, D. Renshaw, *VLSI Signal Processing: A Bit-Serial Approach*, Addison-Wesley, G.B., 1985.
- (3) H.T. Kung, "Why Systolic Architectures?", *IEEE Computer Magazine*, Vol. 15, No. 1, pp. 37-46, Jan. 1982.
- (4) M.R. Buric and C.A. Mead, "Bit-Serial Inner Product Processors in VLSI", *Caltech Conference on VLSI*, pp. 155-164, Jan. 1981.
- (5) J.T. Scanlon and W.K. Fuchs, "High Performance Bit-Serial Multiplication", *Proc. of the IEEE Intl. Conf. on Comp. Design VLSI*, NY, pp. 114-117, Oct. 1986.
- (6) *PPL (Path Programmable Logic) - An IC Design Methodology*, Design Manual - March 7, 1989, University of Utah, Salt Lake City, Utah.