A SYSTOLIC 2-D CONVOLUTION CHIP[1]

H. T. Kung[2]
S. W. Song[3]

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pa. 15213

I. INTRODUCTION

With recent technological advances in the VLSI circuitry, the chip capacity (or component count on a chip) is increasing at an astonishing rate (7). Both the opportunities and challenges regarding effective use of VLSI are tremendous.

[1]This research was supported in part by the Office of Naval Research under Contracts N00014-76-C-0370, NR 044-422, and N00014-80-C-0236, NR 048-659, in part by the National Science Foundation under Grant MCS 78-236-76, and in part by the Defense Advanced Research Projects Agency under Contract F33615-78-C-1551 (monitored by the Air Force Office of Scientific Research).

[2]Currently on leave from Carnegie-Mellon University at ESL's Advanced Processor Technology Group in San Jose, California. (ESL is a subsidiary of TRW.)

[3]Supported in part by CNPq, Conselho Nacional de Desenvolvimento Cientifico e Tecnologico, Brazil, under Contract 200.402-79-CC, and is on leave from the Institute of Mathematics and Statistics of the University of Sao Paulo, Brazil.

ISBN 0-12-564480-9

Systolic design is an architectural concept proposed for VLSI (3). Chip designs based on systolic architectures tend to be simple, modular, and of high performance. In (6) a general discussion on the attractiveness of systolic architectures is given. Systolic architectures are particularly suited to chip implementation of operations in signal and image processing such as filtering, correlation, and discrete Fourier transform (5). In this paper we describe a chip, based on a novel systolic design, for performing the 2-D convolution operator. The chip consists of essentially only one type of simple cells, which are mesh-interconnected in a regular and modular way, and achieves high performance through extensive concurrent and pipelined use of these cells. Denoting by u the cycle time of the basic cell, the chip allows convolving a kxk window with an nxn image in $O(n^2u/k)$ time, using a total of k^2 basic cells. The total number of cells is optimal in the sense that the usual sequential algorithm takes $O(n^2k^2u)$ time. Furthermore, because of the modularity of the design, the number of cells used by the chip can be easily adjusted to achieve any desirable balance between I/O and computation speeds.

Examples of application of the 2-D convolution operator include noise smoothing, linear edge enhancement, edge crispening, etc.. Designs similar to the one described in this paper can be used for digital filtering and numerical relaxation.

II. DESCRIPTION OF THE SYSTOLIC DESIGN

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} & \cdots & x_{1n} \\ x_{21} & x_{22} & x_{23} & x_{24} & \cdots & x_{2n} \\ x_{31} & x_{32} & x_{33} & x_{34} & \cdots & x_{3n} \\ & & \vdots & & & \\ x_{n1} & x_{n2} & x_{n3} & x_{n4} & \cdots & x_{nn} \end{bmatrix}$$

w_{ij} = WEIGHTING COEFFICIENTS

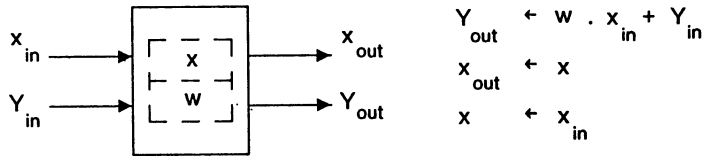
w_{11}	w_{12}	w_{13}
w_{21}	w_{22}	w_{23}
w_{31}	w_{32}	w_{33}

3x3 WINDOW

Figure 1. nxn matrix and kxk window with k=3.

To define the 2-D convolution problem, consider a matrix (or image) $X = \{x_{ij}\}$ and a kxk window with weighting coefficients w_{ij} as shown in Figure 1. For the purpose of illustration, we use k=3. Now slide the 3x3 window along the matrix. For each position of the window, the weighted sum of the entries (or pixels) in the submatrix covered by the window is to be computed. More precisely, if the entry at the center of the submatrix is x_{rs}, then we wish to compute

$$\sum_{i=1}^{3} \sum_{j=1}^{3} w_{ij}\, x_{r+i-2,\, s+j-2}.$$



[EACH X STAYS INSIDE THE CELL FOR ONE CYCLE]

[VALUE W IS A PRELOADED WEIGHTING COEFFICIENT]

Figure 2. Basic cell.

The 2-D convolution chip is based on a variant of the systolic FIR filtering array proposed in (4) and (5). The chip uses essentially only one type of basic cells (see Figure 2), which are interconnected in a regular and modular way to form a two-dimensional systolic array. The function of a basic cell is to update a result Y as indicated in the figure. Note that in each cell, w is a preloaded weighting coefficient and each x stays inside the cell for a cycle.

The three kernel cells that form the systolic array are shown in Figure 3. Each kernel cell is composed of nine basic cells and one row-interface cell whose function will become clear later. Five rows of the matrix X advance synchronously from left to right. The small dots in the figure denote appropriate delays needed to make each column advance as a whole. (This does not mean that these delays have to be implemented on-chip; the inputs can be spaced accordingly when fed into the device.) During the cycle subsequent to that an entry x_{ij} enters the upper right basic cell of a kernel cell, the weighted sum corresponding to a submatrix with that entry as

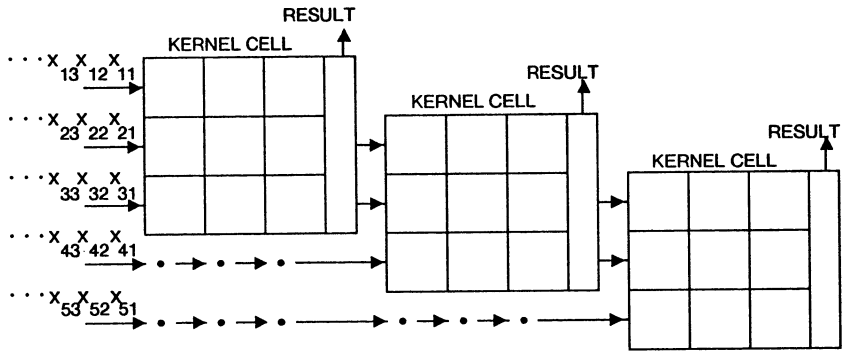
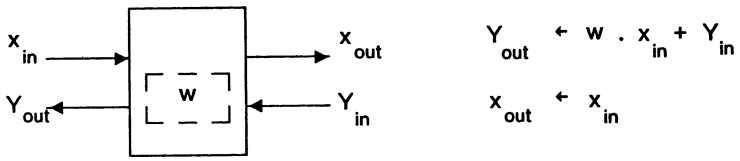


Figure 3. Systolic array consisting of three kernel cells.

the upper left entry will be output from that kernel cell. Therefore by sweeping the first five rows, a systolic array consisting of three kernel cells can compute the first three output rows on-the-fly. In general, by sweeping rows $3i+1$ to $3i+5$ the array computes rows $3i+1$ to $3i+3$. Thus an entry in the nxn matrix is input to the cell array at most twice. In the rest of the section, we describe the underlying idea of the kernel cell design.



[VALUE W IS A PRELOADED WEIGHTING COEFFICIENT]

Figure 4. A two-way flow basic cell.

Suppose we want to compute the following weighted sum in the first row of a kernel cell:

$$Y = w_{11}\, x_{i\,j-2} + w_{12}\, x_{i\,j-1} + w_{13}\, x_{ij}.$$

One way to do this is to use a special case of the linear systolic FIR filtering array discussed in (5). We use a two-way flow basic cell (to distinguish it from the one-way flow basic cell defined earlier) as in Figure 4, and the desired value for Y can be obtained as illustrated in Figure 5 (a) through (c). In Figure 5 (a), we have denoted by Y the particular Y_{in} to the rightmost basic cell, with its value initialized to zero.

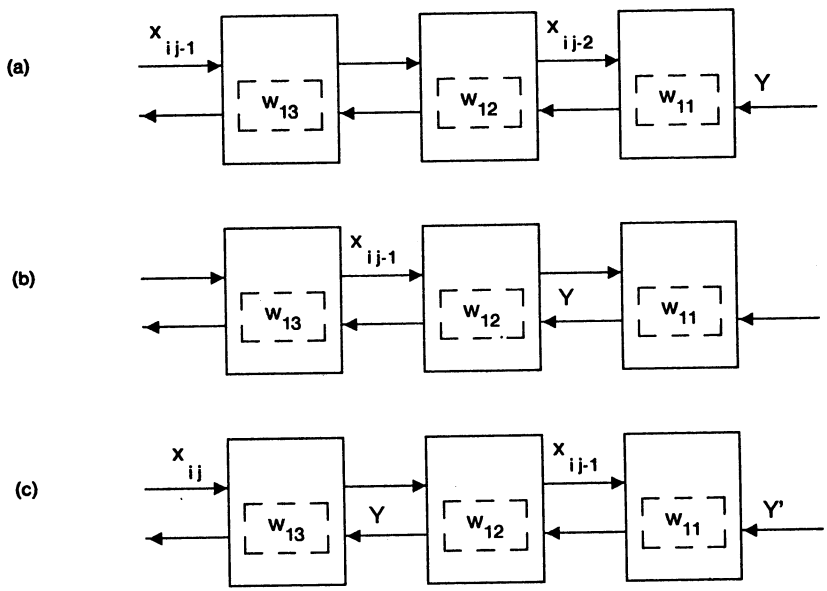


Figure 5. Computing Y with two-way flow.

An improved way is to use the one-way flow basic cell as defined in Figure 2. Consider Figure 6 (a) where three basic cells in the first row of a kernel cell are shown. Consider the moment the entry x_{ij} enters the leftmost basic cell. Denote by Y the particular Y_{in} to this cell, whose value is initialized as zero. During the cycle x_{ij} enters the leftmost cell, Y becomes $w_{13}\, x_{ij}$. In the next cycle $w_{12}\, x_{i\, j-1}$ is accumulated to Y, as shown in Figure 6 (b). Finally, as in Figure 6 (c), $w_{11}\, x_{i\, j-2}$ is further accumulated to Y. As a result, when Y emerges from the rightmost cell, it has already accumulated the weighted sum of the first row of a submatrix. In the meanwhile, the same has

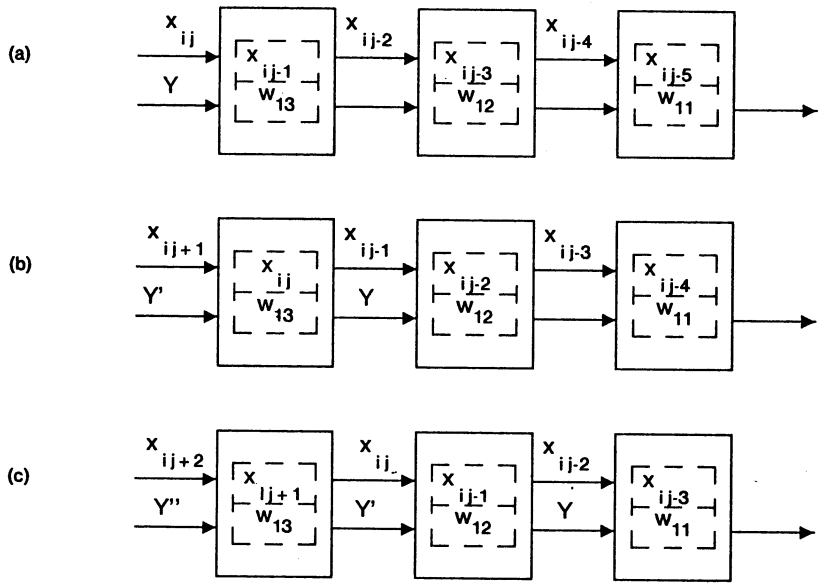


Figure 6. Computing Y with one-way flow.

been happening at the two bottom rows of the kernel cell. Therefore, by summing the partial results at the row-interface cell, the desired output, which is the weighted sum of all the nine entries in a 3x3 submatrix, is obtained (see Figure 3).

The filtering array of Figure 6 differs from the previous one in that both the x-stream and Y-stream now move from left to right and the x-stream travels at half the speed of the Y-stream. This variation results in the active use of every cell at any given time. As a result each kernel cell can produce a pixel every cycle rather than every other cycle, though each basic cell now requires the additional storage to provide the needed delay for the x-stream.

III. BALANCING I/O AND COMPUTATION

Because of the modularity of the basic design of the

systolic convolution array, its size can easily be adjusted to match the bandwidth between the array and the host to which it is attached. Let u denote the cycle time of the basic cell. Suppose that the image is nxn, and that in time u, 2k-1 pixel values can be transferred from the host to the array, and k output pixels can be sent back to the host. For processing the whole image, the I/O time alone is thus $O(n^2u/k)$. To balance this, the convolution array allows convolving a kxk window with the whole image in $O(n^2u/k)$ time, using k kxk kernel cells. We thus see how I/O complexity can be used to guide the design of special-purpose devices. The reader is referred to (2) for several lower bound results on the I/O complexity for a number of problems including the fast Fourier transform. The total number of cells, k^3, used in the proposed convolution array is optimal in the sense that the usual sequential algorithm takes $O(n^2k^2u)$ time.

IV. IMPLEMENTATION

Using part of a chip we have implemented a 3x3 kernel cell, consisting of nine basic cells plus one row-interface cell. The implementation uses a bit-serial word-parallel organization. Pixels are input as 8-bit samples and output with 16 bits. The weighting coefficients can be changed during the loading phase prior to the convolution computation. Their values are restricted to be powers of two in $[-16, 16]$, to reduce the area of multiplier circuits in each basic cell. (See the remark in Section IV.D for a technique of using the chip to handle cases where weighting coefficients are not powers of two.)

We expect that based on this prototype design a full chip can contain three 3x3 kernel cells and output a pixel in less than 175 ns. The anticipated high computation rate is mostly due to the high degree of concurrency inherent in the overall design.

A. I/O Description

A total of twelve I/O pins are used. Eleven of these are for input; only one is for output. Weighting coefficients are loaded into the convolution chip prior to the computation proper and therefore the inputs of these coefficients and the pixel values share the same pins. A cycle is composed of 16 minor cycles each of which includes phases ϕ_1 and ϕ_2. During each

cycle one pixel value is input and one result is output. Since an input pixel value has only 8 bits which are input bit serially, input will occur every other half-cycle. In the following we give a list of names of each pin accompanied by a brief description. Some of the control signals can in fact be generated on-chip.

Name	Description
Vdd	Power
Gnd	Ground
ϕ_1	Clock phase 1
ϕ_2	Clock phase 2
$xorw_1$	Pixel value or weighting coefficient for row 1
$xorw_2$	Pixel value or weighting coefficient for row 2
$xorw_3$	Pixel value or weighting coefficient for row 3
Load	Control signal indicating weighting coefficients are being loaded
LSB	Indicates the least significant pixel bits are being input
Circulate	Control signal indicating the 2nd half cycle (absence of input)
Y_{aux}	Optional input to be accumulated to the computed result (to allow repeated computation for non-power-of-2 weights)
Y	Result of the weighted sum

B. A General Layout

In Figure 7 we illustrate a general layout of a kernel cell. Power and ground, as well as the clock signals, have been omitted in the illustration.

C. Implementation of the Basic Cell

The basic cell is represented schematically as in Figure 8. Each pixel value remains in a basic cell for two cycles while each result stays there for one cycle. Two 8-bit shift registers are provided for the storage of two 8-bit pixel values. During the first half cycle, pixel bits entering a basic cell advance from left to right into the first 8-bit shift

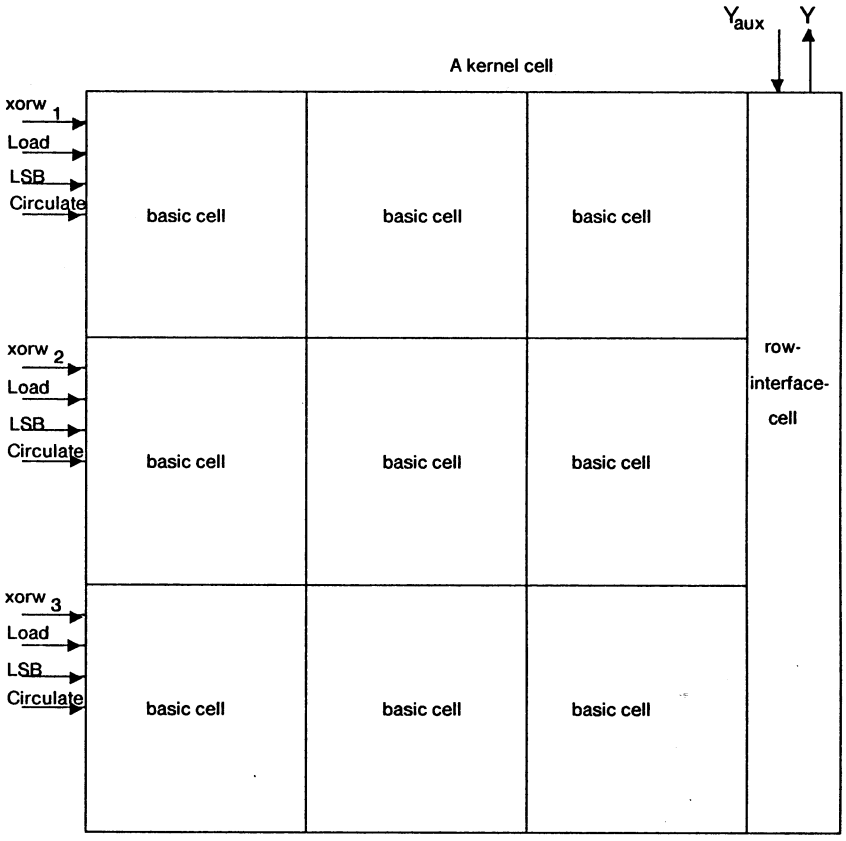


Figure 7. General layout of a kernel cell.

register, and at the same time enter the multiplier, as indicated by the solid lines. During this same time, pixel bits already inside the first 8-bit shift register are moved into the second 8-bit shift register, while pixel bits originally inside the second 8-bit shift register are moved to the next basic cell. During the second half cycle all pixel bits circulate inside the 8-bit shift registers. Flow during the second half cycle is indicated by the dash lines.

Multiplication in this particular case is merely a shift

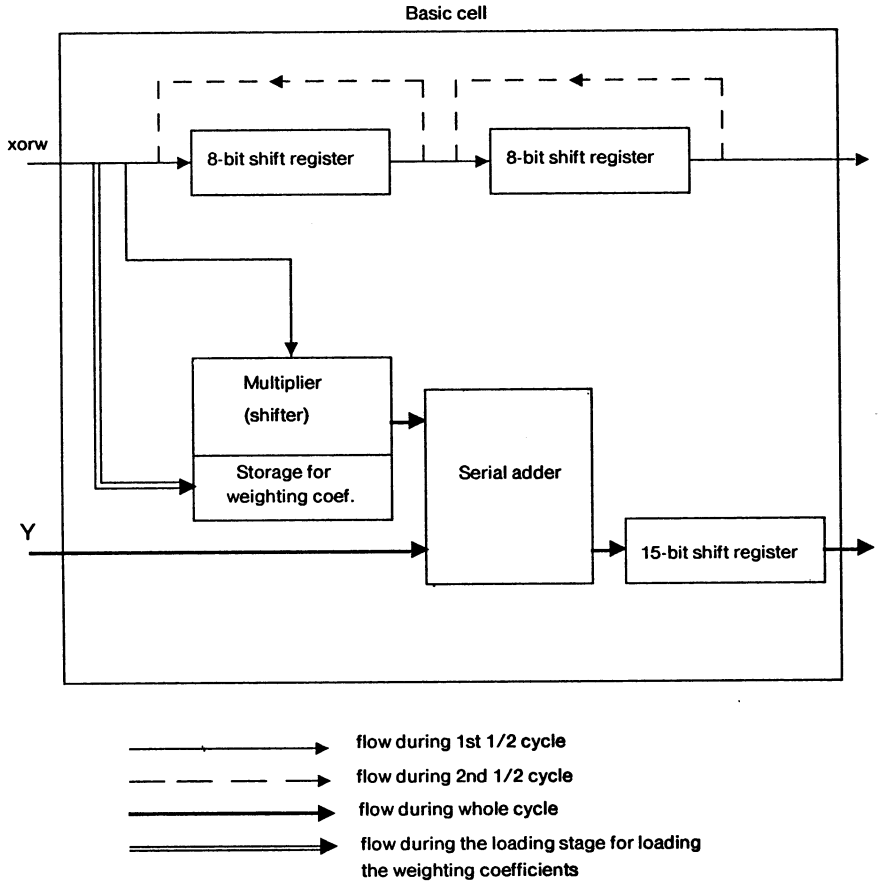


Figure 8. Implementation of the basic cell.

operation. It takes one minor cycle for an input bit to enter the shifter and the result bit to exit the serial adder. A 15-bit shift register therefore is provided to hold the result bits so that they will arrive at the next basic cell at the appropriate time.

D. The Row-Interface Cell

This cell computes the sum of the three partial results from the three rows of basic cells. It also takes an optional input Y_{aux} which will be accumulated to the computed result. This will allow the handling of cases in which weighting coefficients are not powers of two by repeating the computation several times. In most image processing applications it is sufficient to have weighting coefficients as powers of two and, in such cases, Y_{aux} will be zero.

V. CONCLUDING REMARKS

We have developed a chip design for the 2-D convolution operator, which uses essentially only one type of basic cells interconnected in a regular and modular way. While this experiment is about a particular design, we wish to make the following general remark concerning the balance of I/O and computation. Since a special-purpose device is typically attached to a host, from which it gets the data to be processed and to which it outputs results, I/O considerations play an important role on the overall performance. The ultimate goal of a special-purpose hardware design is (and should be no more than) that of achieving a computation rate that balances the available I/O bandwidth. Thus it is important that the design be modular so that its size can easily be adjusted to match the bandwidth between the device and the host. In the design presented basic cells are implemented in a bit-serial manner. As a result, the time for a basic cell to process a pixel is likely to be longer than that for the memory to input and output a pixel. This calls for parallel inputs and outputs of multiple pixels at each cycle of the basic cell. This is the reason why we made the systolic array of this paper input five pixels and output three pixels every cycle. If, on the other hand, basic cells are implemented by bit-parallel schemes (which imply a much shorter cycle time for the basic cell), then the chip is likely able to input and output only one pixel during each cycle. A one-dimensional, rather than two-dimensional, systolic array would be an appropriate design for this case.

ACKNOWLEDMENTS

We would like to thank all those people who made the MPC79

project (1) possible, through which a prototype design of our chip could be implemented. We would also like to thank the following people at CMU who helped on this research. Dave McKeown and John Kender provided valuable information regarding typical requirements in image processing applications. Dan Hoey provided the design of the serial adder. Mike Foster and Dan Hoey checked for design rule violations.

REFERENCES

1. Conway, L., Bell, A., and Newell, M. E., MPC79: The Large-Scale Demonstration of a New Way to Create Systems in Silicon. LAMBDA 1(2), pages 10-19, 1980.

2. Hong, J.-W., and Kung, H. T., I/O Complexity: The Red-Blue Pebble Game. In Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing, pages 326-333, May, 1981. Also available as a CMU Computer Science Department technical report CMU-CS-81-111, March, 1981.

3. Kung, H. T., and Leiserson, C. E., Systolic Arrays (for VLSI). In Duff, I. S., and Stewart, G. W. (editors), Sparse Matrix Proceedings 1978, pages 256-282. Society for Industrial and Applied Mathematics, 1979. A slightly different version appears in Introduction to VLSI Systems by C. A. Mead and L. A. Conway, Addison-Wesley, 1980, Section 8.3.

4. Kung, H. T., Let's Design Algorithms for VLSI Systems, In Proceedings of Conference on Very Large Scale of Integration: Architecture, Design, Fabrication, pages 65-90. California Institute of Technology, January, 1979. Also available as a CMU Computer Science Department technical report, September, 1979.

5. Kung, H. T., Special-Purpose Devices for Signal and Image Processing: An Opportunity in VLSI. In Proceedings of the SPIE, Vol. 241, Real-Time Signal Processing III, pages 76-84. The Society of Photo-Optical Instrumentation Engineers, July, 1980.

6. Kung, H. T., Why Systolic Architecture. To appear in Computer Magazine, 1981.

7. Mead, C. A., and Conway, L. A., Introduction to VLSI Systems, Addison-Wesley, Reading, Massachusetts, 1980.