National Défense
Defence nationale

UNCLASSIFIED
UNLIMITED DISTRIBUTION

CRDV RAPPORT 4513/89
DOSSIER: 3632D-018
AVRIL 1989

DESIGN OF A SYSTOLIC VLSI CONVOLUTION PROCESSOR

Y. Boudreault

RESEARCH AND DEVELOPMENT BRANCH
DEPARTMENT OF NATIONAL DEFENCE
CANADA

BUREAU - RECHERCHE ET DÉVELOPPEMENT
MINISTÈRE DE LA DÉFENSE NATIONALE
CANADA

Defence Research Establishment
Centre de recherches pour la Défense,
Valcartier, Québec

Canada

SANS CLASSIFICATION
DISTRIBUTION ILLIMITÉE

DREV R-4513/89
FILE: 3632D-018

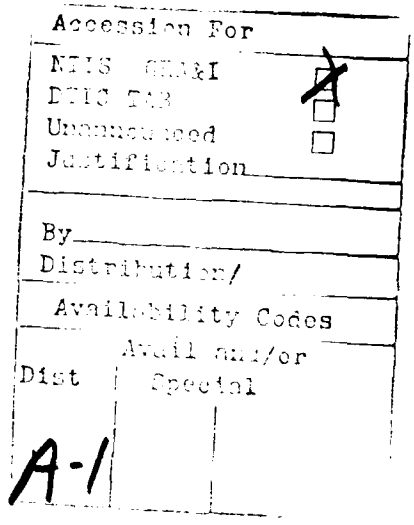
UNCLASSIFIED

CRDV R-4513/89
DOSSIER: 3632D-018

DESIGN OF A SYSTOLIC VLSI CONVOLUTION PROCESSOR

by

Y. Boudreault



DEFENCE RESEARCH ESTABLISHMENT

CENTRE DE RECHERCHES POUR LA DÉFENSE

VALCARTIER

Tel: (418) 844-4271

Québec, Canada

April/avril 1989

SANS CLASSIFICATION

ABSTRACT

This report describes the design and implementation of a systolic convolution processor. It uses a number of bit-serial processors, each operating on 8-bit signed coefficients and 8-bit positive pixel intensities. The processing time is independent of the coefficient array size due to the architecture selected. Overflow detection and output scaling capabilities are provided for autonomous applications. Two processors were placed on a CMOS custom integrated circuit using 5μ design rules. Simulation results were used to estimate the processing time to be under 0.5 s for a 512 x 512 pixel image.

RÉSUMÉ

Ce rapport décrit la conception et la réalisation d'un processeur systolique spécialisé pour la convolution. Étant donné l'architecture utilisée, le temps de calcul est indépendant du nombre de coefficients. Les processeurs systoliques sont sériels et utilisent des coefficients signés de 8 bits ainsi que des pixels positifs de 8 bits. Une certaine capacité de détecter les erreurs de calcul existe pour les applications autonomes. Le processeur fut réalisé en circuit intégré CMOS suivant des règles de conception de 5μ. À l'aide de simulations, on a estimé qu'une image de 512 x 512 pixels pourrait être traitée en moins de 0.5 s.

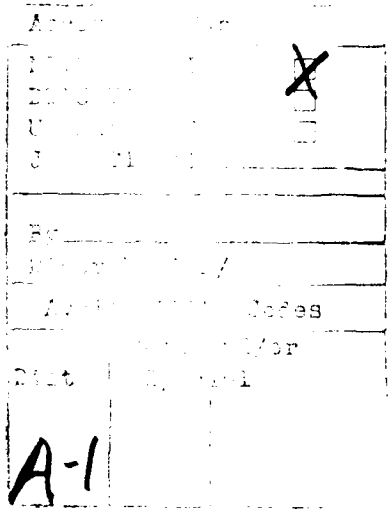


TABLE OF CONTENTS

1.0 INTRODUCTION

Over the past years, Defence Research Establishment Valcartier (DREV) has been involved in the automatic detection and tracking of targets in imagery. These efforts have resulted in the development of many algorithms, some of which use digital convolution to implement linear and certain non-linear filters (e.g. Sobel edge detector).

The two-dimensional convolution may be described by eq. 1, where y(m,n) is the output image, x(m,n) the input image, and h(m,n) is the system's impulse response. In practice, the convolution is always summable because (1) the image and the impulse response are defined only in a limited range of k and l; and (2) the image intensity and the impulse response coefficients are bounded.

$$y(m,n) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} x(k,l)\, h(m-k,\, n-l) \qquad [1]$$

Throughout this report, the terms impulse response, coefficient array, and convolution kernel will be used interchangeably.

Because of its importance in image processing applications and its heavy computational demand, many people have implemented digital convolution in hardware. Among such implementations, it is worth mentioning the convolution computer described by Heuft and Little (Ref. 1), which uses a number of identical processors in parallel, each working on a different point of the output image. Other implementations include the ones by Comtal Corp. (Ref. 2) and Quantex Corp. (Ref. 3) which were real-time versions limited to 3 x 3 coefficient arrays. More recently, systolic implementations have been proposed by Kung et al (Refs. 4-6), McCanny et al (Refs. 7-9), Danielson (Ref. 10) and many more.

A survey of image processing techniques indicated the need to rapidly process large convolution kernels, typically up to 15 x 15. For compatibility with current digitizers, a convolver should be able to handle images with up to 8 bits/pixel and varying image size. Such a processor should be as compact as possible and some capabilities for autonomous operation should be provided.

This report describes the design and implementation of a systolic convolution processor that operates bit-serially and provides an overflow detection circuitry which enables it to perform unsupervised applications. The design was targeted for Very Large Scale Integrated (VLSI) circuit implementation in order to maximize the convolution array size while minimizing the total area requirements of the convolver. The circuit was manufactured by Northern Telecom through Canadian Microelectronics Corporation (CMC).

The main part of this work was performed while the author was on educational leave at McGill University, and was completed at DREV between March 1985 and May 1986 under PCN 32D18 Real-Time Image Filters.

2.0 ARCHITECTURE

This chapter introduces the basic concepts used and describes the architecture selected for this convolution processor. Section 2.1 introduces the systolic arrays, Section 2.2 describes the advantages and disadvantages of bit-serial communications and processing. Section 2.3 deals with the architecture selected for the individual processors while Section 2.4 describes how these processors may be interconnected to perform the desired two-dimensional convolution.

2.1 Systolic Arrays

The architecture used in this design is based on the principle of a systolic array as introduced by Kung and Leiserson (Refs. 4, 11 and 12). This approach uses a number of identical processors organized in a regular array. A global (synchronous) clock "pumps" data through the array of processors in the same way as the heart pumps blood through the arteries of the body. This approach has numerous advantages, including:

1) The reduction of the input/output requirements through the use of the same data many times;

2) The ease of expansion through modularity and regular flow of data and control signals;

3) The ease and speed of the design and implementation cycles because of the repetitive use of simple cells; and

4) The use of local communications, eliminating the need for global interconnections.

There are also a few disadvantages to systolic arrays, including the large number of I/O signals they usually require, as well as their lack of programmability in most cases.

The basic structure of a systolic processor is a regular array of identical processors, each of which may only communicate with their nearest neighbors. For example, let us now investigate the computation of a one-dimensional convolution (eq. 2a) using a systolic processor, and let us limit the impulse response of the system to only 3 coefficients (eq. 2b).

$$y(m) = \sum_{k=-\infty}^{+\infty} x(k)\, h(m-k) = \sum_{k=-\infty}^{+\infty} h(k)\, x(m-k) \qquad [2a]$$

$$y(m) = h(0)\, x(m) + h(1)\, x(m-1) + h(2)\, x(m-2) \qquad [2b]$$

Figure 1(a) shows a linear convolver described by Kung and Picard (Ref. 5). Each processor is used to perform part of the computation for a single output point. Figure 1(b) shows the functionality of an individual processor. In particular, the processor stores the value of the coefficient in an internal register, computes the product of h(k) times x(1) and adds this product to the Y_{in} input. The X_{out} output is simply a delayed version of the input signal X_{in}, necessary for the proper synchronization of the processors.

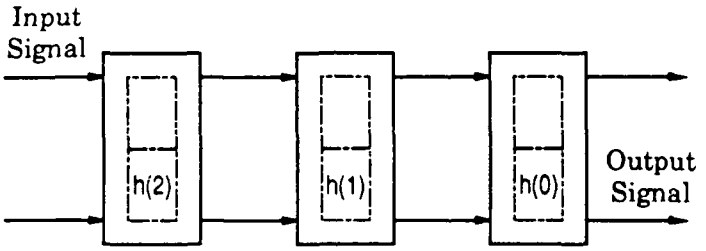
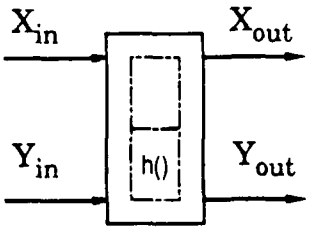


FIGURE 1(a) - Systolic linear convolver

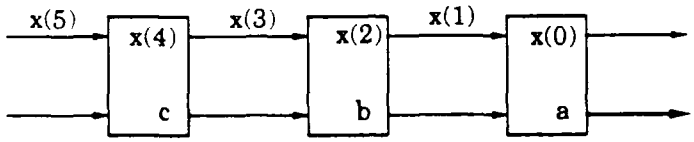


$$X_{out}(t) = X_{in}(t-2)$$

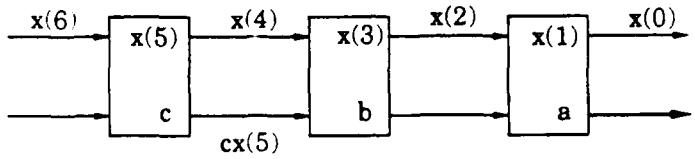
$$Y_{out}(t) = Y_{in}(t-1) + h()\, X_{in}(t-1)$$

FIGURE 1(b) - Systolic processing element

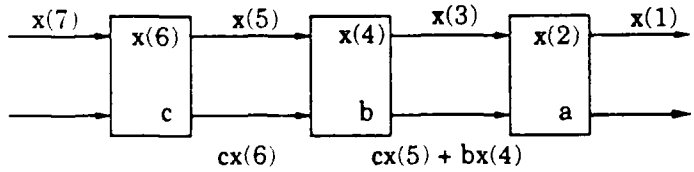
Figure 2 shows an example of the operation of such a convolver through a number of clock cycles. The example shows a three-point, linear convolution with coefficients a, b and c, on clock cycles 5 through 9. The desired result is of the form of eq. 2b, where $h(0) = a$, $h(1) = b$ and $h(2) = c$.



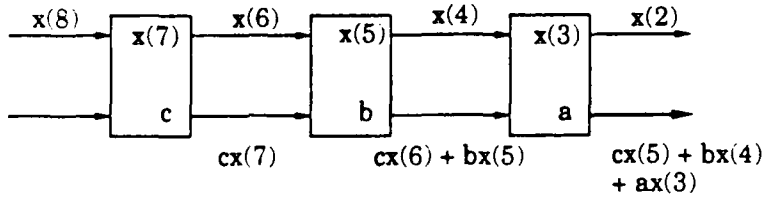
(a) 5th clock cycle



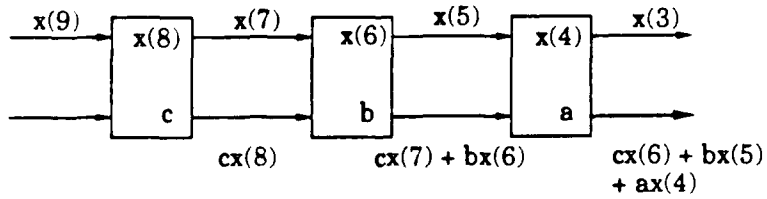
(b) 6th clock cycle



(c) 7th clock cycle



(d) 8th clock cycle



(e) 9th clock cycle

FIGURE 2 - Example of the operation of a 3-coefficient linear convolution processor

The inputs x(n) are entered on every cycle. Each processor computes part of the convolution for every output point. In particular, the first processor computes cx(5) on the 5th cycle, the second processor computes bx(4), and adds it to the result from the first processor, resulting in the partial sum cx(5) + bx(4) on the 6th cycle, while the last processor computes the missing element ax(3), completing the computation of the convolution point with the desired result on the 7th cycle: cx(5) + bx(4) + ax(3) = y(5).

2.2 Bit-Serial Communications

Bit-serial computing consists in processing the data one bit at a time. This method is often used in highly parallel architectures such as the MPP (Ref. 13), the CLIP4 (Refs. 13-14) and many more. Its advantages include:

- Reduced number of I/O signals,
- Smaller processor size for higher density,
- Simpler elements for faster design cycle, and
- Less propagation delay for a shorter clock cycle.

However, these advantages are offset by the following inconvenients:

- Slower operation, and
- More complex control structure.

Even though parallel computations are certainly much faster, it is interesting to note that bit-serial computations will execute at a faster clock rate, but require more cycles to complete the operations.

Through simulations, it was estimated that the bit-serial processor would meet the design requirements of less than 1 s to process a 512 x 512 pixel image. It was then decided to use this approach in order to minimize the size of the processors and hence maximize the size of the coefficient array that could be handled in a specific area.

2.3 Processor Architecture

The basic architecture of this convolver is similar to the one described in Section 2.1, but with a number of modifications to accommodate our choices of using bit-serial communications and processing, as well as our need to include some form of scaling of the multiplier output and overflow detection. This section describes the processor architecture adopted for this project.

2.3.1 Data Representation

In such a design, it is important to determine the accuracy and resolution that will be required at the different steps of the processing. In the case of convolution, the output is often an image with the same gray scale resolution as the input image; however, it is important to obtain results with a greater dynamic range in order to reduce the rounding errors to a minimum.

Let us now investigate the number of bits and the data formats required to represent the data. First, the input pixels should be represented in a format compatible with today's digitizers. Certainly the most widely accepted format is to represent pixels as 8-bit, unsigned numbers ranging in values from 0 to 255. For synchronization purposes, this 8-bit value is stored in the least significant portion of a 16-bit word.

On the other hand, representing the coefficients as positive values would not be satisfactory, since it would restrict the flexibility of the processor. It was determined that a two's complement representation of 8-bit (values ranging from -128 to +127) would allow sufficient range for most applications.

Finally, one must represent the partial convolution sum (Y) data. To simplify the control structure of the processors, it was decided to assign it the same format as the output of the multiplier, i.e. 16-bit, two's complement (values ranging from -32768 to +32767).

2.3.2 Processor Block Diagram

The structure of the processor is shown in Fig. 3, and a short description of the signals is given in Table I. The processor contains a serial-parallel multiplier used to compute the product of the coefficient with the input pixel intensity, and a serial adder circuit for the partial convolution sum input and the output of the multiplier. It also contains an 8-bit shift register used to store the coefficient, a 32-bit (2 words) shift register to synchronize the pixel intensity data, and a 16-bit shift register (1 word) to synchronize the flow of the partial convolution sum data.

Two additional blocks have been included for autonomous operation. One is the overflow detection circuit used to flag overflows and underflows at the output of the serial adder circuit. The OV_{out} flag is generated by a logical OR of the output of this circuit along with the OV_{in} signal; this flag indicates the occurrence of an error while computing the current output point Y_{out}.

TABLE I

Description of processor signals

Symbol	Name	Description
X_{in}	Pixel input	Input gray level, 16-bit positive value ($0 \leq X_{in} < 255$).
C_{in}	Coefficient Input	Input coefficient, 8-bit two's complement value.
Y_{in}	Partial Sum Input	Input partially computed convolution sum, 16-bit two's complement number.
OV_{in}	Overflow Input	Input flag indicating whether an overflow occurred while computing Y_{in}.
X_{out}	Pixel Output	Output gray level, 16-bit value ($0 \leq X_{in} < 255$).
C_{out}	Coefficient Output	Output coefficient, 8-bit two's complement number.
Y_{out}	Partial Sum Output	Output partial convolution sum, 16-bit two's complement number.
OV_{out}	Overflow Output	Output overflow flag indicating if an overflow occurred during the computation of Y_{out}.
Ld	Load Coefficient	When active, the coefficient shift-register is activated and the coefficient is loaded through C_{in}.
Pe	Processor Enable	When active, the whole processor is activated and the value of Y_{out} computed.
ResetA	Adder Reset	When active, the adder circuit is reset.
ResetM	Multiplier Reset	When active, the multiplier is reset.
Sd	Sum Disable	When active, the output of the multiplier is replaced with the sign of the coefficient. This is used for dynamic range extension.
Φ_1 and Φ_2	Clocks	Two phase, non-overlapping clock.

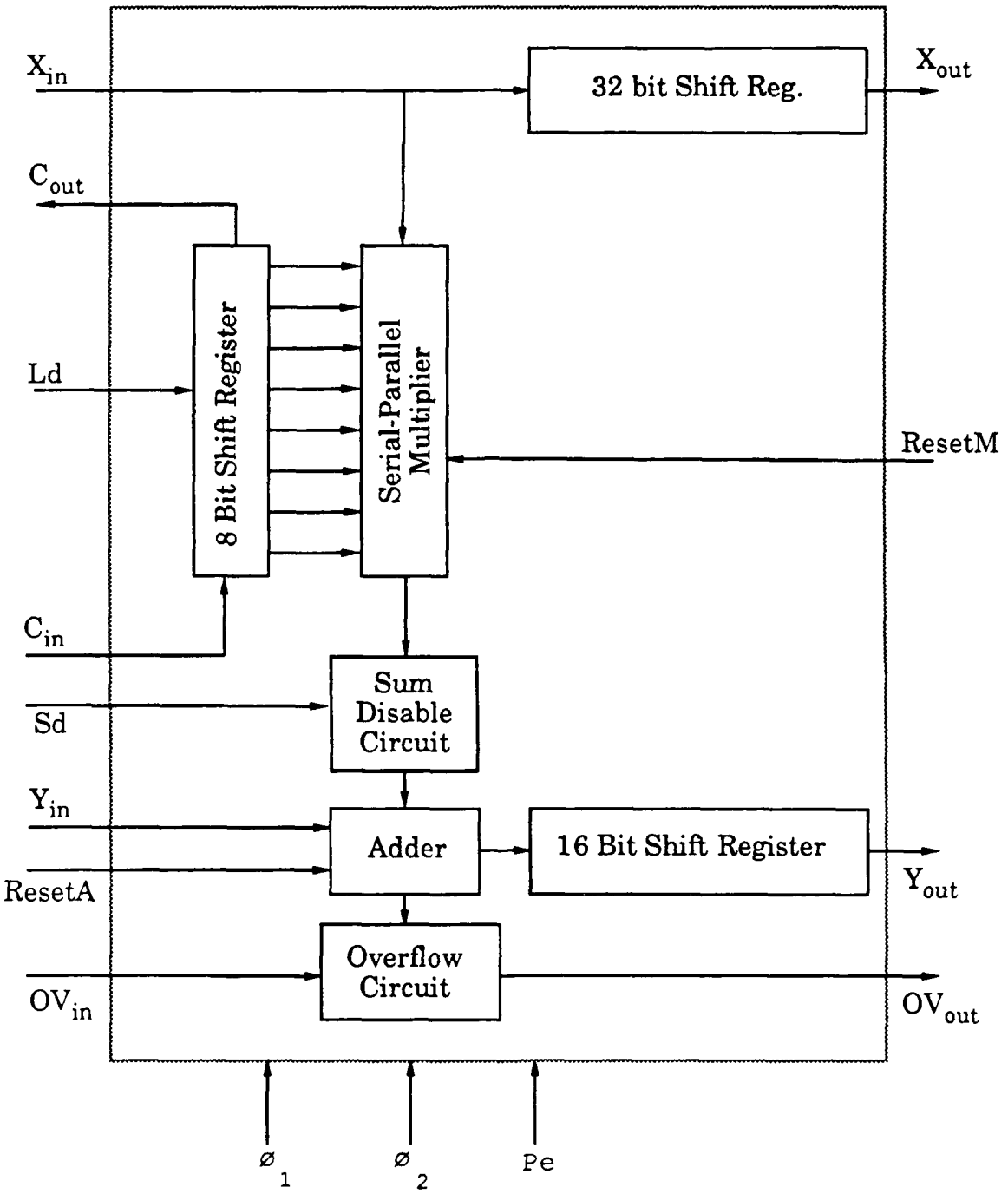


FIGURE 3 - Block diagram of processor architecture

The second circuit is a sum-disable circuit used to scale the output of the multiplier by a factor of 2^{-n}, where n is a positive integer representing the number of cycles during which the signal Sd is active. This circuit is useful to scale down the output of the multiplier in order to reduce occurrences of overflows. In a typical system, all of the processors would share a common Sd signal.

2.3.3 Processor Operation

In order to simplify the control structure, all the processors are synchronized on word boundaries, each operating on the same bit at the same time. The correct operation of the circuit requires 16 clock cycles. Figure 4 shows a typical operating waveform.

The pixel intensities and the partial convolution sums are entered in the processor one bit at a time, starting with the least significant bit. On the last cycle, the most significant bit is entered to complete the processing. It is important to note that while bit 3 is being entered in the processor, bit 3 of another word is also available at the output so that the processors may easily be chained to perform convolutions of arbitrary lengths.

2.4 Convolver System

The structure of a 1-D convolver using this processor is very similar to the one described in Section 2.1. Figure 5 shows such a convolver based on the processor described in Section 2.3. The overflow signals are connected to form a shift register in order for the OV_{out} flag to always represent the validity of the Y_{out} result.

Based on the processor described in the previous section, it is now possible to design a two-dimensional convolver. The convolution sum of eq. 1 may be separated into a sum of 1-D convolutions. Hence, a 2-D convolution may be performed using a structure as shown in Fig. 6.

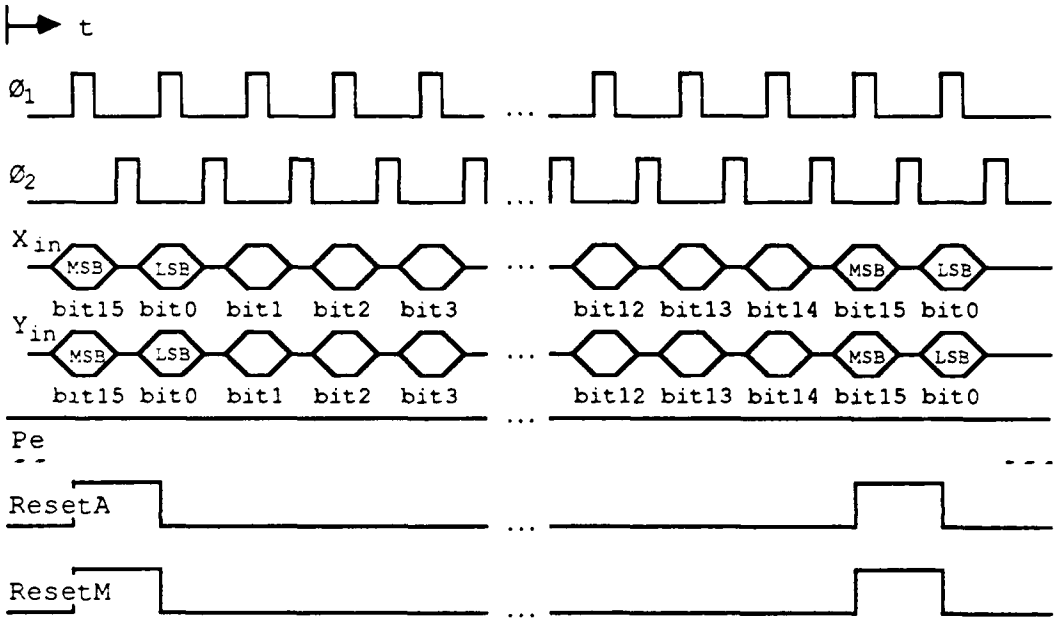


FIGURE 4 - Waveform of convolution operation

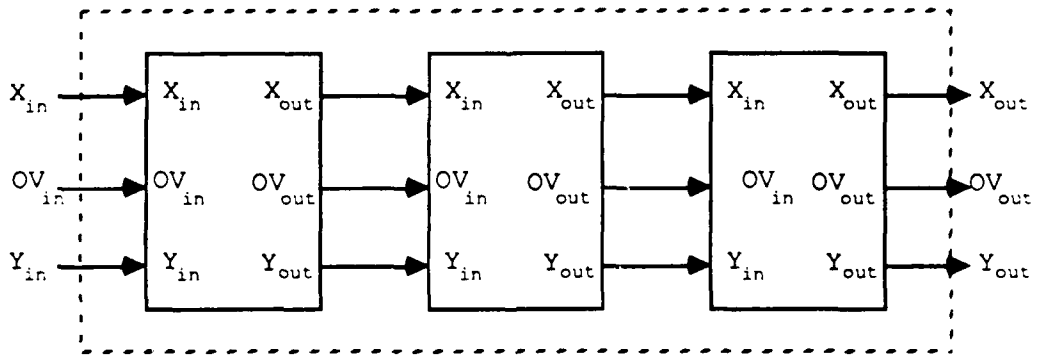


FIGURE 5 - One-dimensional convolver

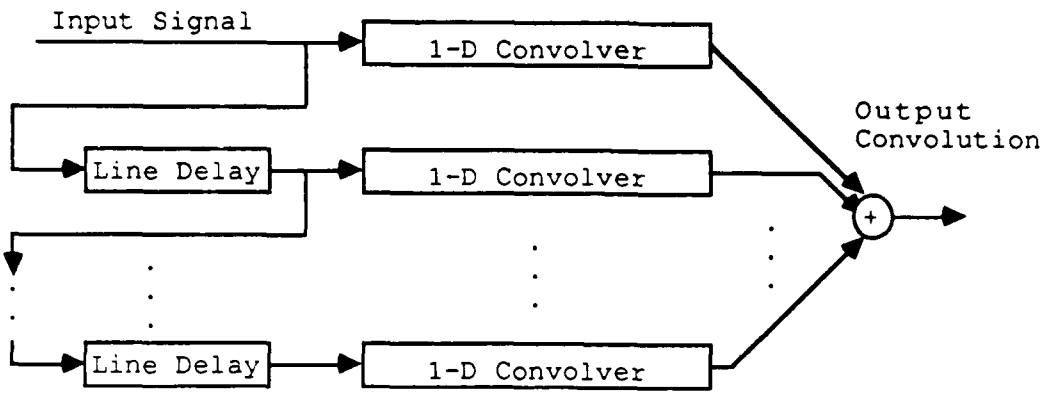


FIGURE 6 - Two-dimensional convolver

One important characteristic of systolic arrays is their expansibility, which is due to their basic regular structure. The proposed structure maintains this characteristic. It enables the user to employ the number of processors required for the particular application, using one processor per filter coefficient. The dimension of the image may also be changed without affecting the array of processors, although the delay structure would have to be modified accordingly.

3.0 DESIGN

This chapter introduces some of the CMOS design concepts and describes the main building blocks used in this design, including a serial-parallel multiplier, an adder, a shift register and some other basic circuits. This chapter also gives a description of how these blocks were assembled to form an integrated circuit.

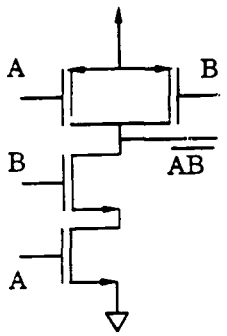
3.1 CMOS Gates

This design was implemented through an agreement between Northern Telecom and Canadian Microelectronic Corporation. There were two technologies available: a 5µm nMOS and a 5µm CMOS process.

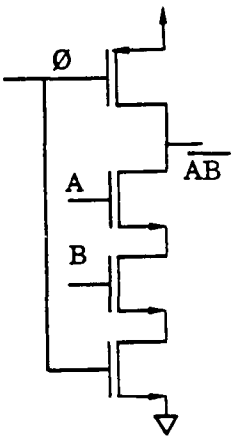
Because Northern Telecom was phasing out their nMOS process, it was thus decided to use CMOS for this design. The new 3μ process was not available when this circuit was designed.

There are many ways to implement CMOS circuits. One approach is to use static gates, which are very similar to normal digital gates in that they are not clocked. Figure 7(a) shows a static NAND. The advantages of a static implementation include the relatively low power consumption and the small number of transistors switching simultaneously. However, the disadvantage is certainly the large area required for static CMOS gates, since the logic must be duplicated for both the n and the p-type transistors. To avoid this duplication process, dynamic CMOS gates are often used. An example of a dynamic NAND is shown in Fig. 7(b). Notice that certain transistors are controlled by a pre-charge clock (Φ). Dynamic gates use the gate capacitance of the following transistors to store a pre-charge value, which may then be modified (charged or discharged) during the evaluation phase. The use of dynamic gates often permits more compact designs, but there are a number of problems associated with their use. In particular, race conditions may often yield inaccurate results. Also, the large flow of charges during the pre-charge phase may induce a condition known as latch-up, which prevents proper operation of the circuit (Ref. 15). Many approaches or design techniques have been devised to reduce these problems. One method is the domino logic (Ref. 16) in which dynamic gates are always followed by a static inverter. This method eliminates the race problem because the pre-charge phase always disables all the n-type transistors. An example of a domino AND gate is shown in Fig. 7(c). However, this method precludes the use of inverting logic. Another method proposed by Goncalves and De Mann (Ref. 17) uses alternating n-logic and p-logic gates, so that the pre-charge of one gate always disables the logic transistors in the next gate. A number of rules are described in this report to ensure that race conditions do

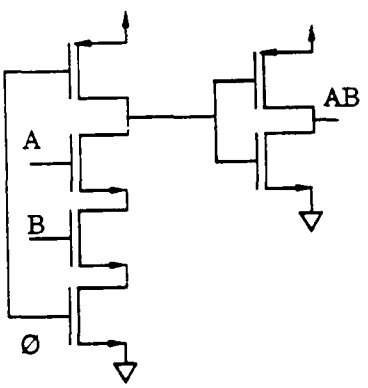
not occur during pre-charge. This method has been termed NoRa, standing for "No Race". An example using three NAND gates is shown in Fig. 7(d).



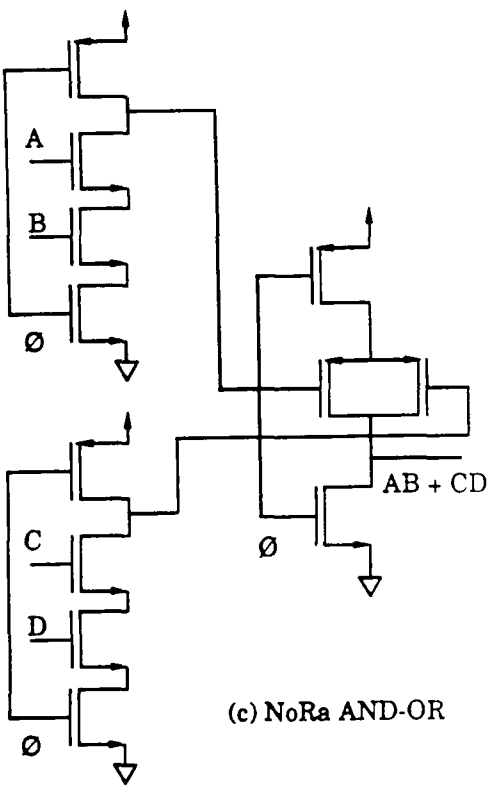
(a) Static NAND



(b) Dynamic NAND



(c) Domino AND



(c) NoRa AND-OR

FIGURE 7 - CMOS logic gates

It was decided to use static CMOS gates for this project because they are relatively easy to use and are less likely to induce a substrate latch-up. Also, because most of the gates needed are relatively simple, the use of static gates is not expected to significantly increase the layout requirements.

3.2 Common Cells

There are three basic cells that are used throughout this design. They are a full-adder and two types of flip-flops, with and without preset. This section will describe these three cells and assess their performance based on circuit simulations performed using SPICE, an electrical circuit simulator developed at the University of California, Berkeley.

3.2.1 Full-Adder

A full-adder is a subcircuit that is used to encode three bits into a sum and carry format. The truth table of this function is shown in Table II, while the logic equations (eqs. 3, 4) are shown below:

$$K = AB + AC + BC \qquad [3]$$

$$S = ABC + \overline{K}(A + B + C) \qquad [4]$$

TABLE II

Truth table of a full-adder

A	B	C	S	K
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

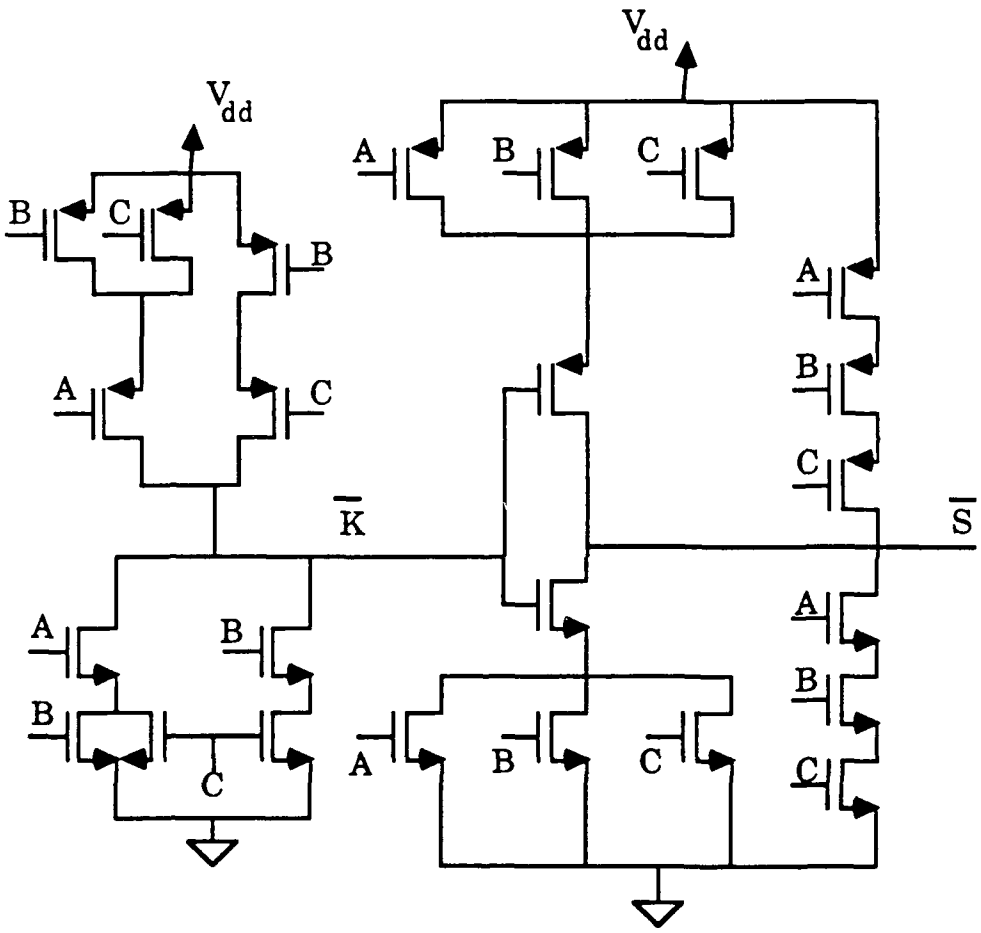


FIGURE 8 - Full-adder circuit diagram

In Table II and eq. 4, A, B and C are the inputs, S is the sum and K is the output carry. The advantage of the formulation shown is that no inverted signals are necessary except for $\overline{K}$. A transistor diagram of a CMOS circuit implementing this function is shown in Fig. 8. One should notice that K is generated as an active low signal ($\overline{K}$) so that it may be used directly to generate S.

The circuit was laid out in 5μ CMOS and required 211 x 193 μm^2. The worst case propagation delay through this circuit occurs when a change in the input signal triggers a change in the carry which, in turn, forces a change in the sum output; for example, when the input switches from ABC = 100 (K=0, S=1) to ABC = 110 (K=1, S=0). This situation was simulated using SPICE. The maximum propagation delay was found to be approximately 15 ns.

3.2.2 Flip-Flop Without Preset

The next basic cell required was a flip-flop. This cell was required to store the coefficients and also to form the shift registers. In order to reduce the chances of race conditions, master-slave circuits were designed. The circuit diagram is shown in Fig. 9.

There are two similar subcircuits, the master and the slave. The first one stores the data when Φ_1 is active, otherwise the data is recirculated through the feedback. The slave is controlled by Φ_2 and operates in a similar way. In order to operate this flip-flop correctly, Φ_1 and Φ_2 are pulsed sequentially. It is important to note that if the clocks do overloap, a race condition may occur causing the latch to become invisible.

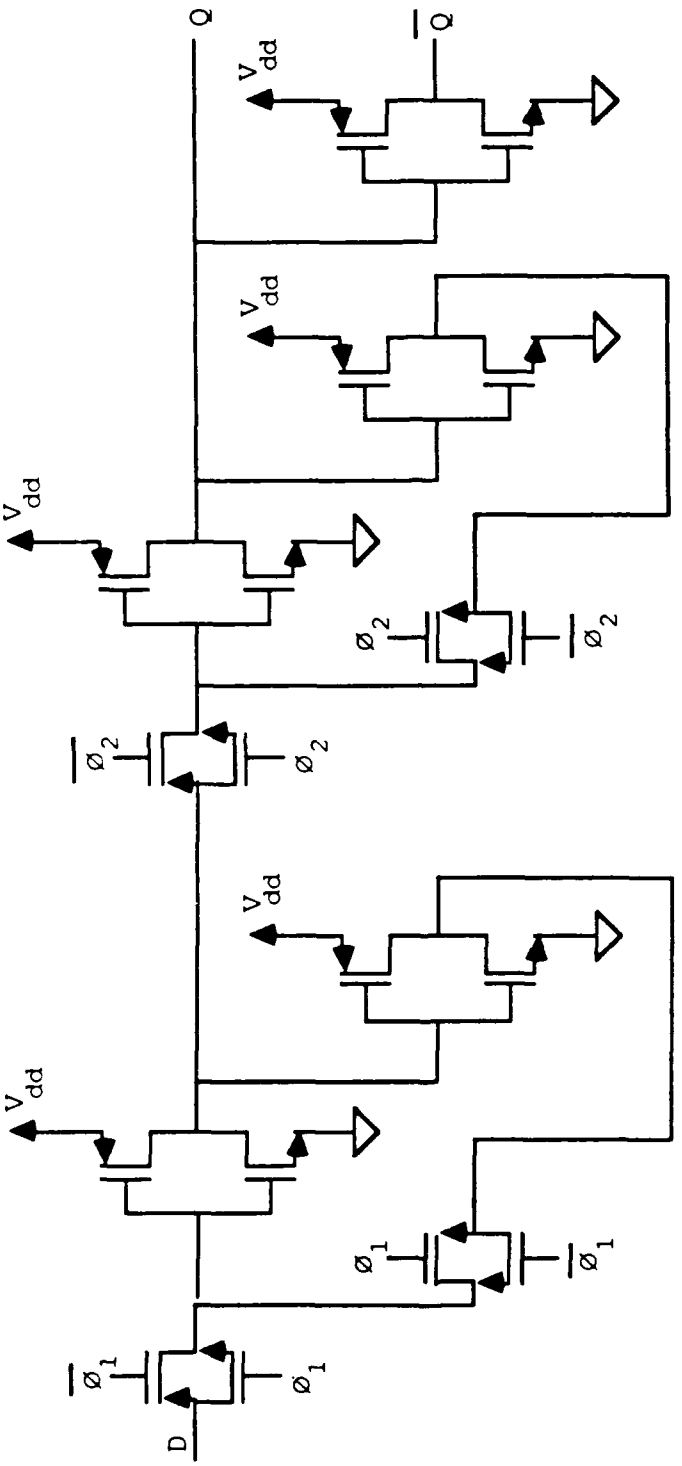


FIGURE 9 - Circuit diagram of flip-flop without preset

The layout of this circuit required 365 x 178 μm^2. Simulation results indicate that the flip-flop hold time is approximately 12 ns. This was computed from the time the input is valid until the time the output of the feedback is stabilized. The output is valid 7 ns after the leading edge of Φ_2.

3.2.3 Flip-Flop With Preset

The design of the multiplier (Section 3.3) required flip-flops that could be preset. The circuit diagram is shown in Fig. 10. The basic structure and operation are similar to the other flip-flop, except for the presence of a preset signal in the master section.

The layout required 375 x 204 μm^2. Simulation results indicated a preset hold time of 4 ns. The flip-flop hold time is approximately 12 ns and the output is valid 7 ns after the leading edge of Φ_2.

3.3 Serial-Parallel Multiplier

The next circuit that must be designed is the multiplier. The serial-parallel structure was selected to minimize computation time while maintaining a minimum number of I/O signals. This structure was possible because the coefficients are rarely modified and may easily be stored in a parallel form on chip.

Let us now investigate how this multiplier may be implemented. First, the coefficient is a two's complement number (sect. 2.3.1) which may be represented by eq. 5. The c(i) represent the i^{th} bit of the coefficient C.

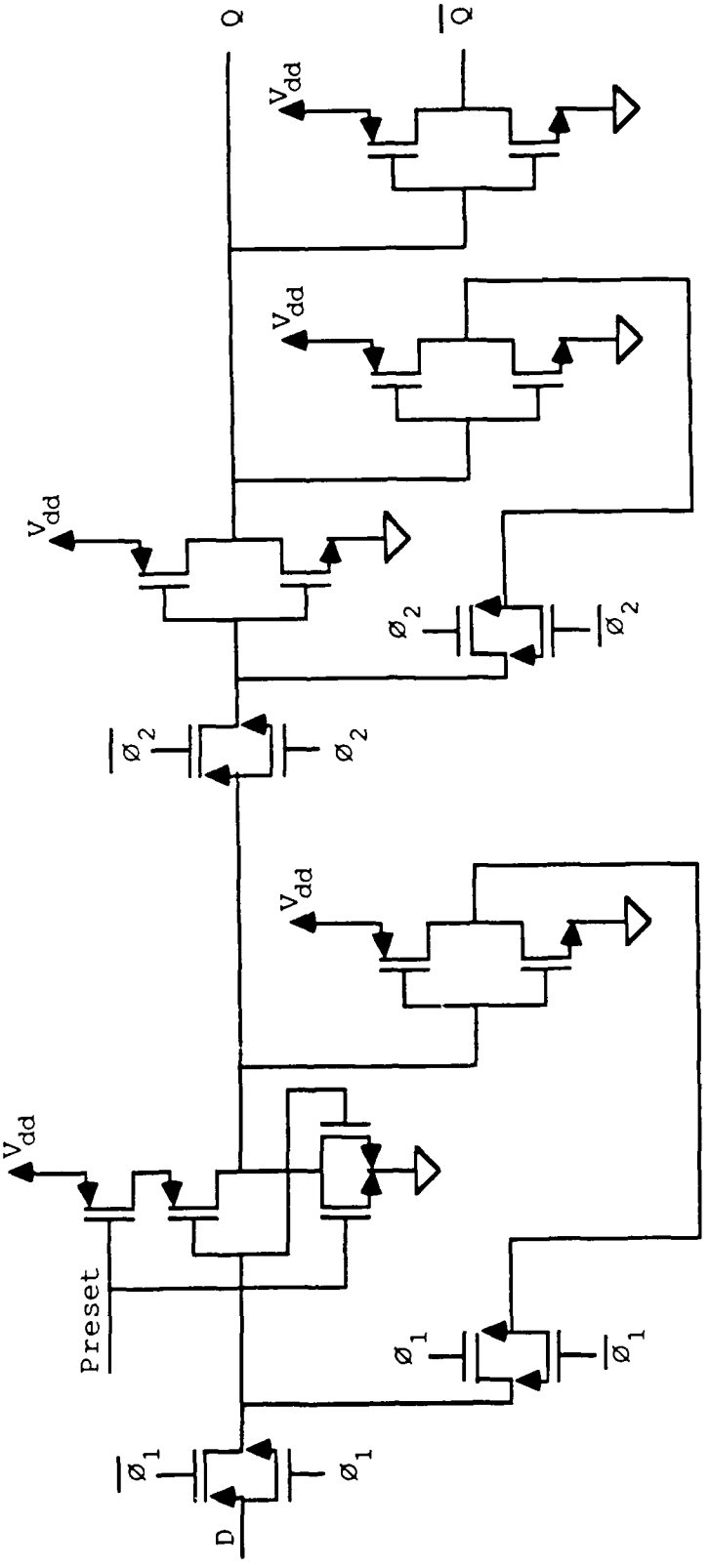


FIGURE 10 - Circuit diagram of flip-flop with preset

$$C = -c(7)\ 2^7 + \sum_{i=0}^{6} c(i)\ 2^i \qquad [5]$$

The product may then be described by eq. 6, in which X' is the two's complement representation of X.

$$CX = -X\ c(7)\ 2^7 + \left(\sum_{i=0}^{6} c(i)\ 2^i\right) X \qquad [6a]$$

$$CX = c(7)\ X'\ 2^7 + \left(\sum_{i=0}^{6} c(i)\ 2^i\right) X \qquad [6b]$$

A hardware structure to implement this product is shown in Fig. 11. It consists of seven identical blocks, with an eighth similar but modified to properly handle the sign bit (perform the two's complement of the X input). Each block computes one of the $c(i)\ 2^i X$ product. A control signal (Ctl) is used to identify the first cycle and perform part of the computation of the two's complement of X.

The actual implementation of the multiplier was accomplished using the basic cells described in Section 3.2. Two cells were necessary, one for the main block of the multiplier and one to handle differences between the sign cell and the normal cell. The design of the main cell is shown in Fig. 12 and requires 962 x 454 μm^2, while the sign cell is shown in Fig. 13 and is 959 x 192 μm^2. Figure 14 shows how these cells are combined to form the desired multiplier.

Results from a simulation using SPICE allowed us to estimate the worst multiplier cell delay at 32 ns. Because of the intrinsic pipeline used in the multiplier, this value also corresponds to the multiplier worst delay during a single cycle. Hence, an 8 x 8 multiplication (16 cycles) can be performed in 512 (=16 x 32) ns.

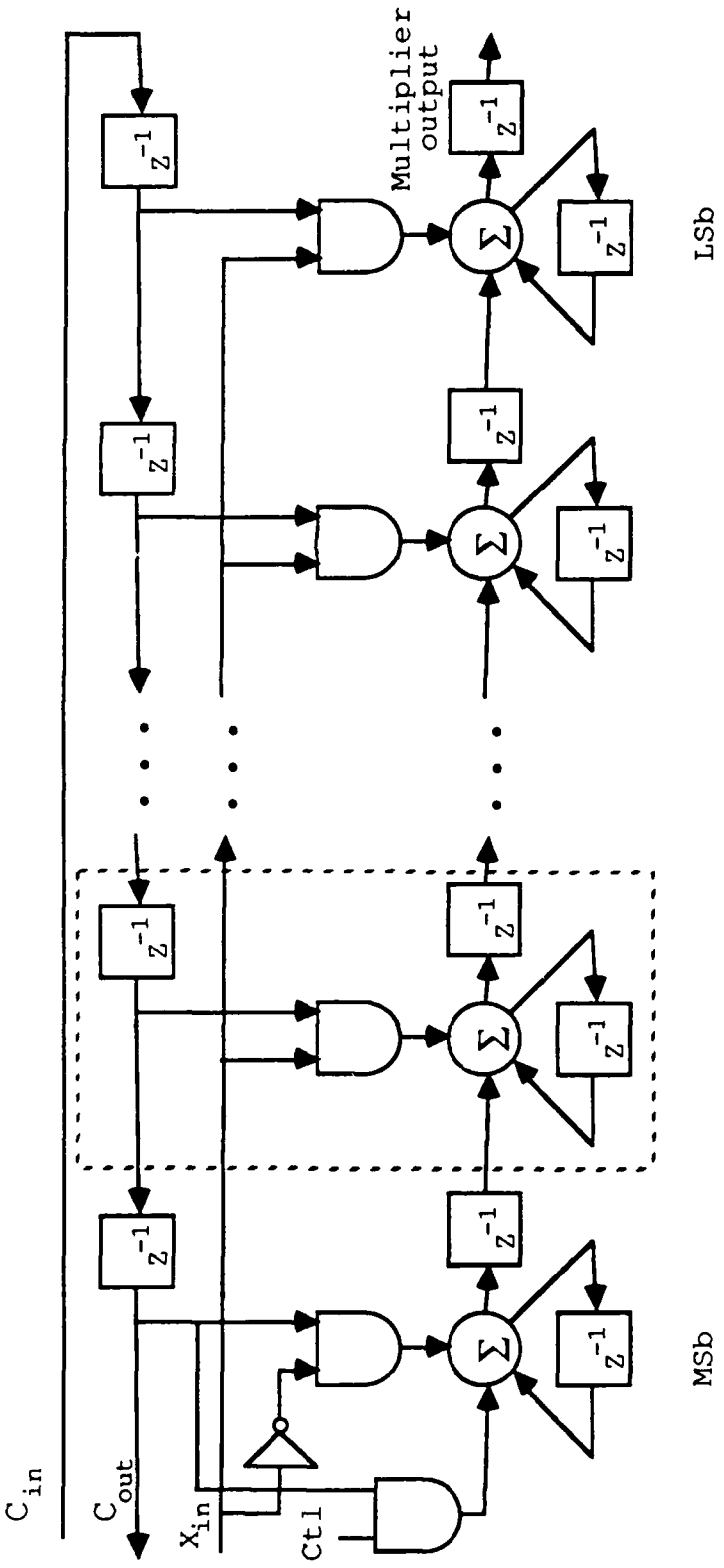


FIGURE 11 - Block diagram of serial-parallel multiplier

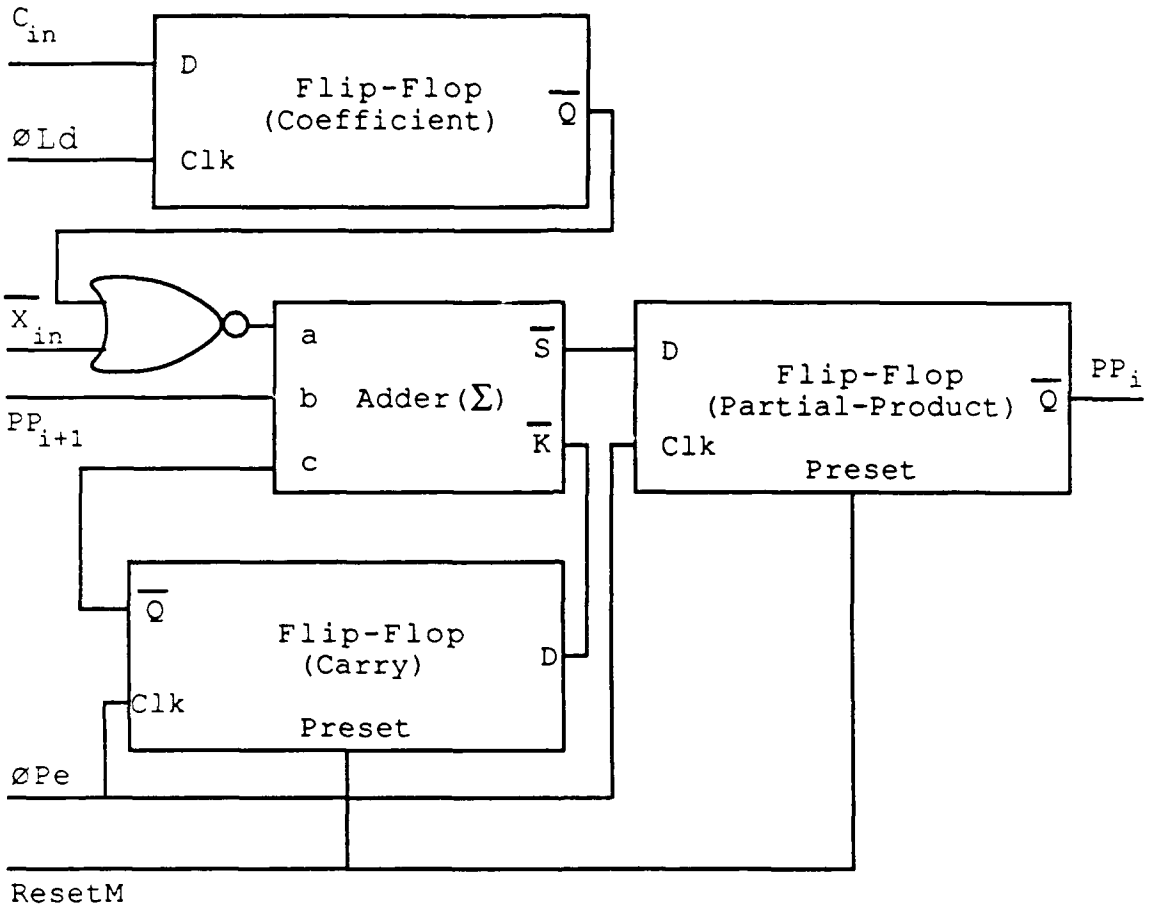


FIGURE 12 - Main cell circuit of serial-parallel multiplier

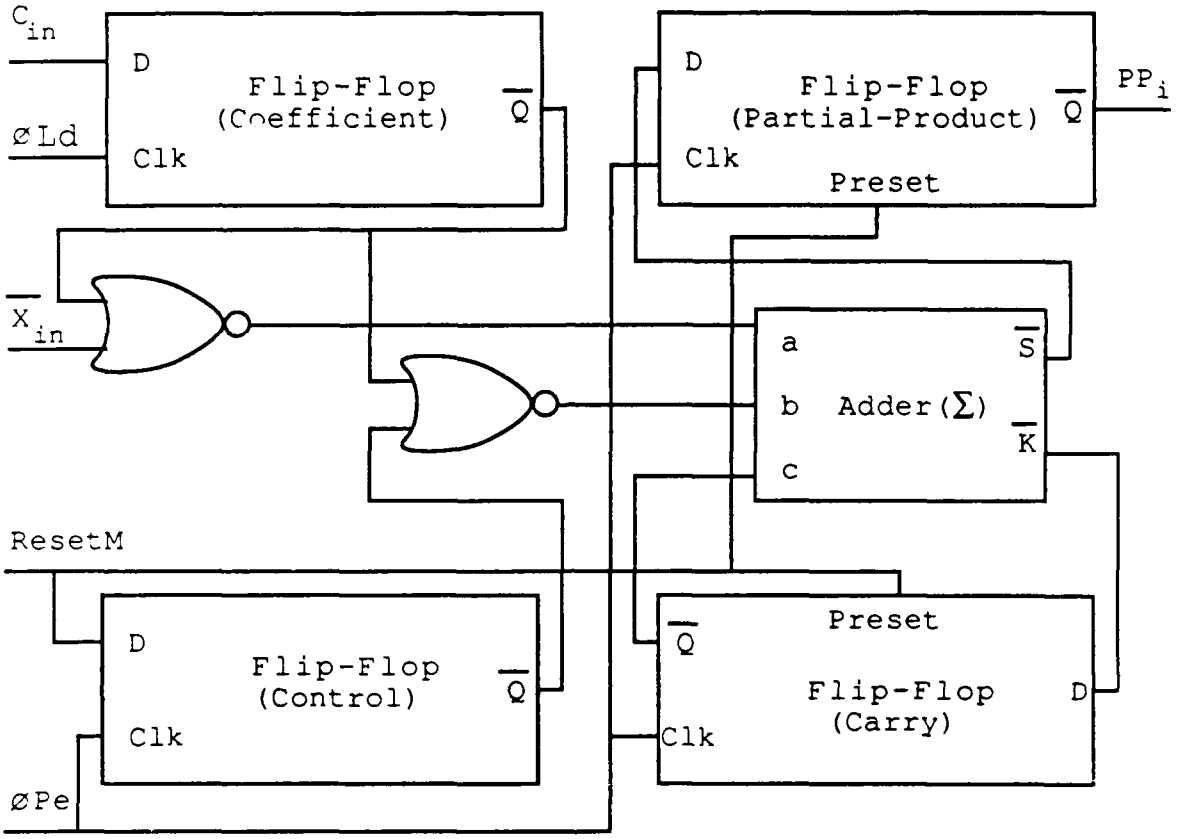


FIGURE 13 - Sign circuit of serial-parallel multiplier

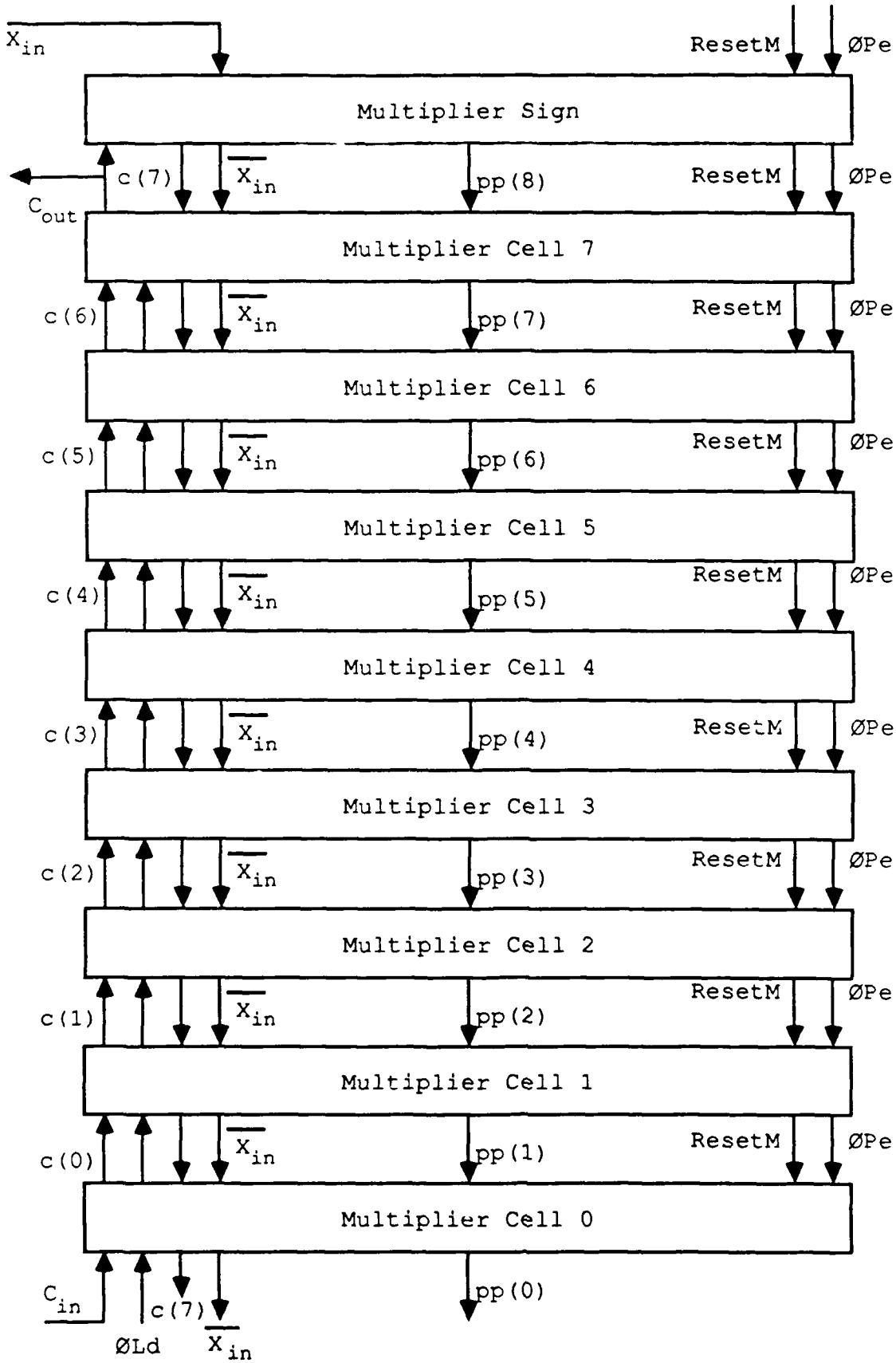


FIGURE 14 - Block diagram of multiplier layout

3.4 Shift Registers

The shift registers were simple to design using the master-slave flip-flops described in Section 3.2. However, in order to save real-estate on the chip, another layout was performed using the same circuit diagram as shown in Fig. 9. The basic cell could then be duplicated to form long shift registers. Figure 15 shows a 16-bit version, which required 704 x 1327 μm^2.

3.5 Sum-Disable

The circuit used for sum-disabling is shown in Fig. 16. Sign-extension is done by replacing the multiplier output bits with the coefficient sign. This is valid because the input is always positive and hence, the multiplier output sign is always the same as that of the coefficient.

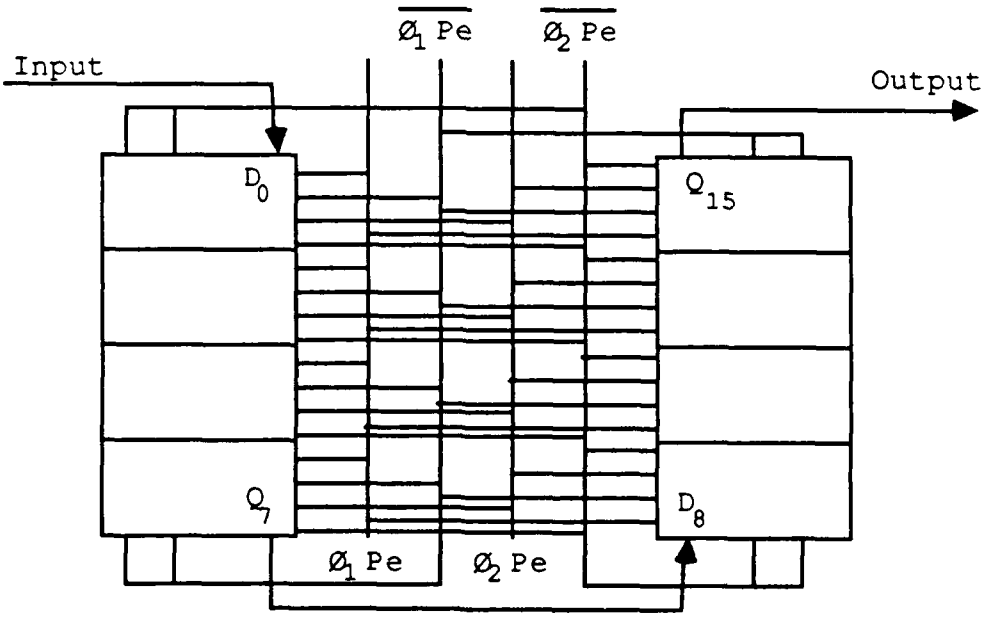


FIGURE 15 - Block diagram of 16-bit shift register

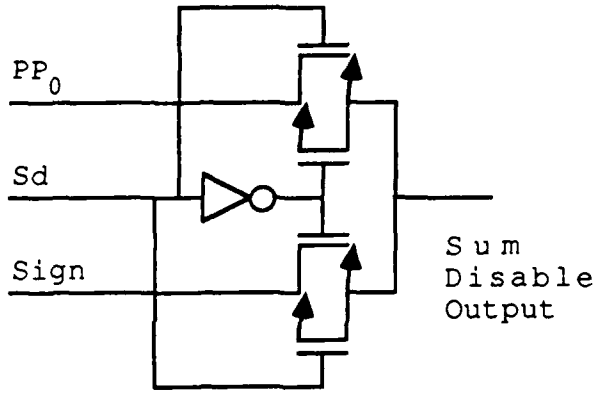


FIGURE 16 - Circuit diagram of sum-disable circuit

3.6 Adder

A serial adder is used to sum the multiplier output with the partial sum input. This circuit is shown in Fig. 17. It incorporates a full-adder, a flip-flop for the carry, and a flip-flop for the input (for synchronization with the multiplier output). The actual layout also incorporates the sum-disable circuit as well as a flip-flop used by the overflow circuit. The cell requires 963 x 465 μm^2.

3.7 Overflow

When two n-bit numbers are added, an overflow or underflow is detected when the carry from the $(n-1)^{th}$ addition is different from the carry from the n^{th} addition. The overflow circuit diagram is shown in Fig. 18. The layout of this circuit requires 938 x 490 μm^2, and the circuit delay was estimated at 17 ns.

3.8 Clock Drivers

Because of the large number of transistor gates that must be driven from the clocks, special drivers were designed. They are used to produce the control signals (and their inverse) to enable the different flip-flops.

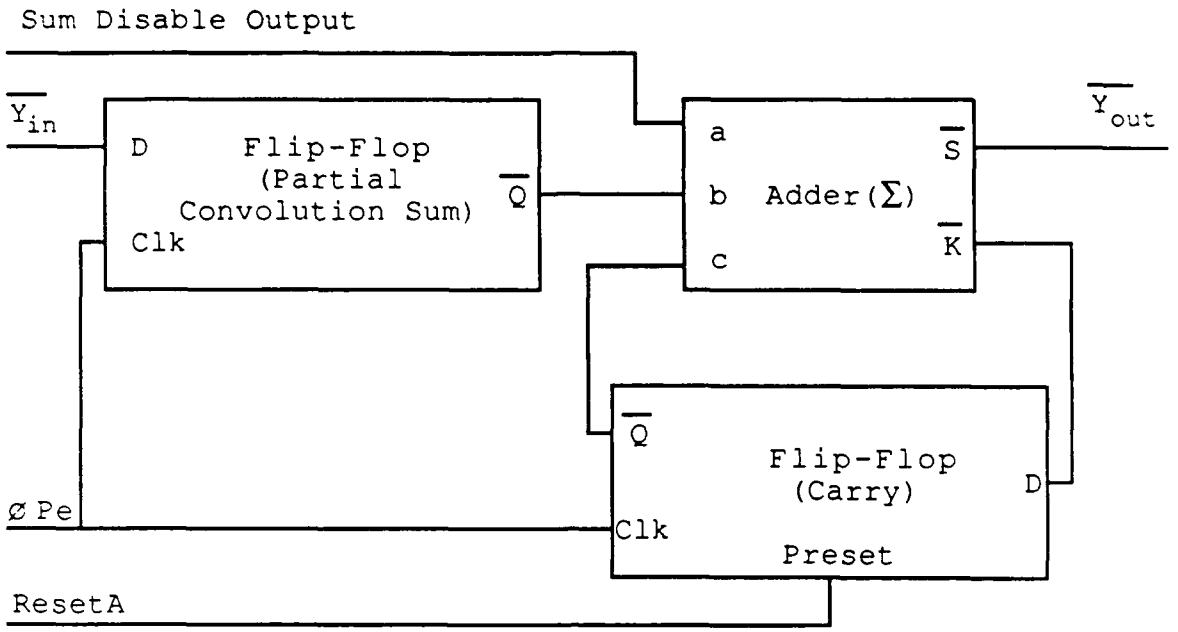


FIGURE 17 - Circuit diagram of serial adder

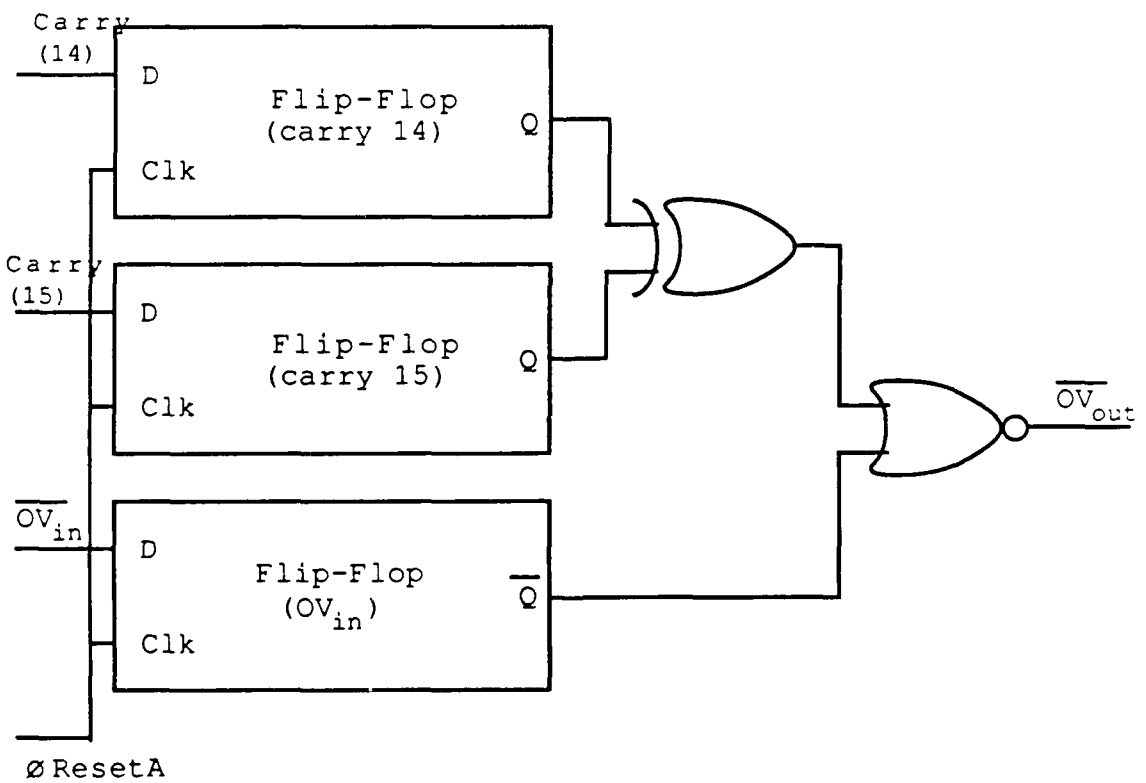


FIGURE 18 - Circuit diagram of overflow detection

3.9 Processor

All the cells were designed and laid out to minimize the number of connections required when assembling the processor. Figure 19 shows the basic layout of a single processor. The size of the processor is 1752 x 4682 μm^2.

3.10 Chip

Finally, the circuit layout was completed at the chip level with the inclusion of two processors on a single chip. The layout is shown in block diagram form in Fig. 20. On the diagram, the small boxes on the periphery are the pad drivers, which provide the interface to the outside world. The boxes marked (Φ_1Pe, Φ_2Pe, etc.) generate the control signals required by the processors, which are represented by the large boxes.

The chip dimensions are 4700 x 4700 μm^2. A total of 19 pads were used, 2 for power, 12 input, and 5 output. A total of 3380 transistors are used in this design. The worst case propagation delay was estimated at 92 ns, for a maximum clock frequency of 10.9 MHz. The power requirements for this circuit is estimated to be 70 mW at its maximum operating frequency.

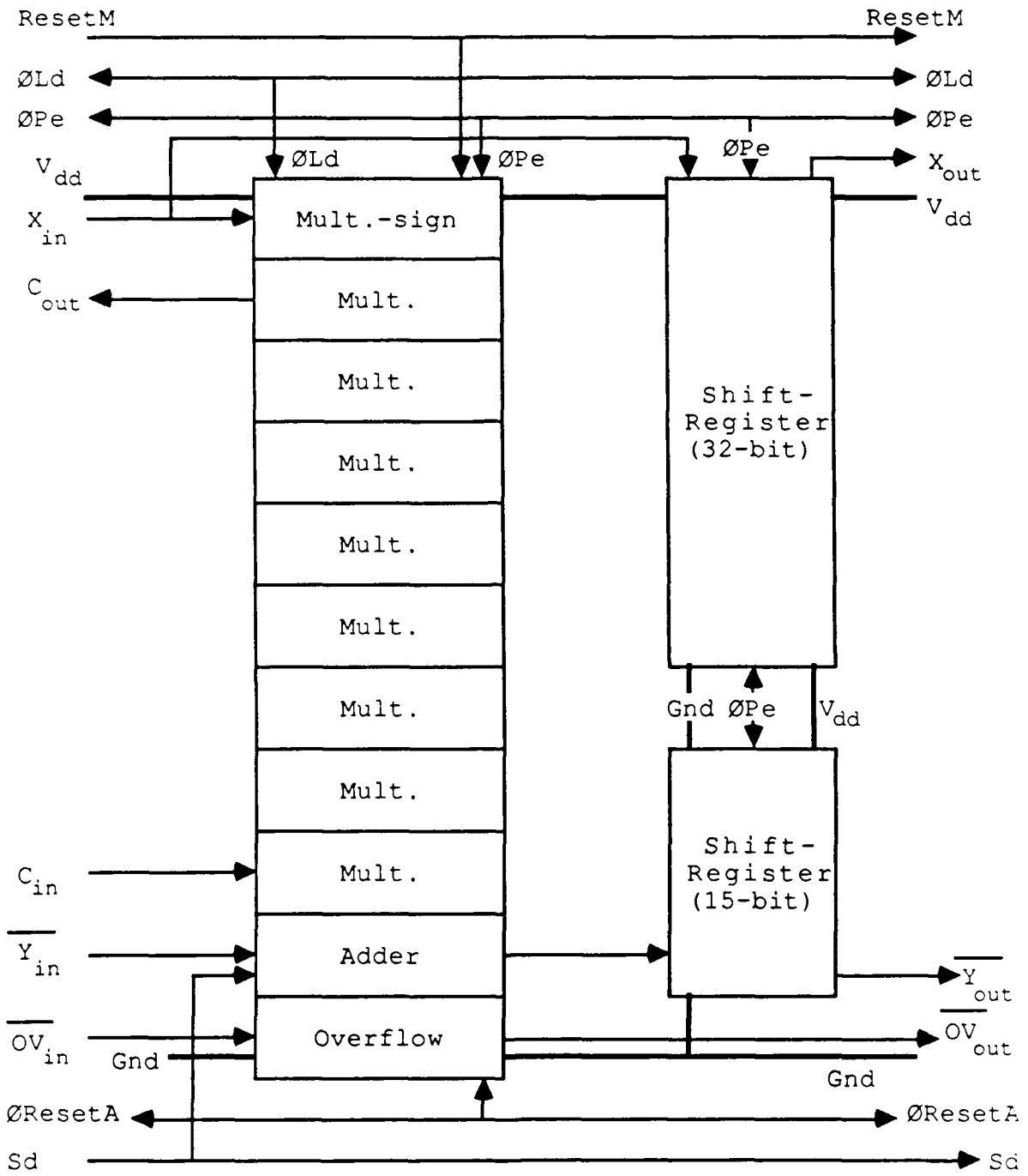


FIGURE 19 - Block diagram of processor layout

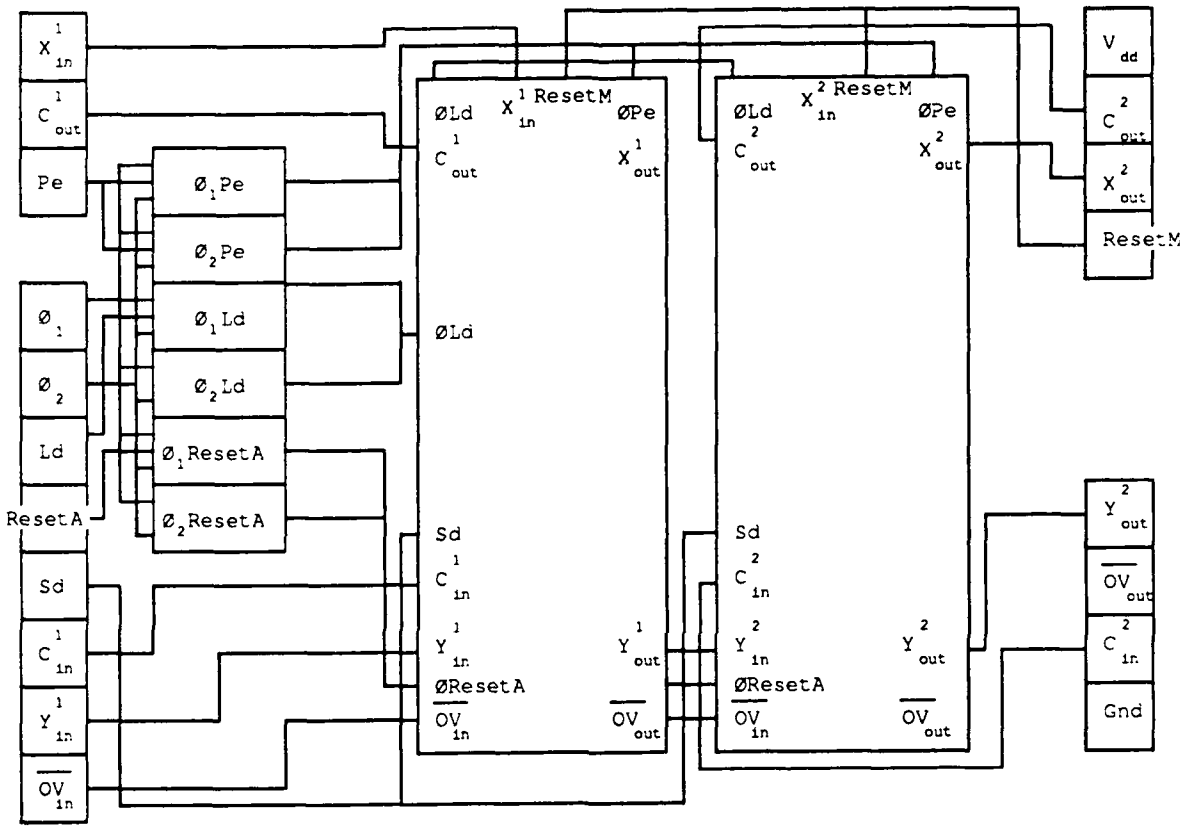


FIGURE 20 - Block diagram of chip layout

4.0 CONCLUSION

Digital convolution is certainly among the most commonly used processing functions in image processing. However, it is computationally expensive; a total of (M x N) x (I x J) multiplications and (M x N) x (I x J - 1) additions are required to process an M x N image by an I x J coefficient array.

This report described the design of a VLSI systolic convolution circuit which was implemented in a 5μ CMOS process. It uses a pipelining technique to divide the computational task among a number of identical, synchronous processors. The time required to perform digital convolution is proportional to the size of the image because the system uses one processor per coefficient.

Bit-serial communications and processing were used throughout this design to reduce the processor size (maximize density) as well as the number of I/O pins, thus allowing the use of smaller packages. A total of 19 pins were used.

The coefficients were represented as 8-bit, two's complement numbers while the pixel values were restricted to a range from 0 to 255. This was done to allow for easy interfacing with standard digitizers as well as a reasonable range of coefficients.

Overflow detection and multiplier scaling capabilities were included in the design so that this processor can be used for autonomous applications. Moreover, with these capabilities, the user can carry out simple or sophisticated error recovery.

Two systolic processors were put on a single integrated circuit of 5 mm x 5 mm with all interconnections done internally. The circuit simulations show that it should be able to operate at up to 10 MHz, allowing it to process a 512 x 512 pixel image in approximately 0.35 s. Power consumption was estimated to be 25 mW when operating at 4 MHz. More details about this design may be found in Ref. 18.

5.0 REFERENCES

1. Heuft, R.W. and Little, W.D., "Convolution Computer", IEEE Transactions on Computers, Vol. C-29, pp. 738-740, August 1980.

2. Comtal Vision One/20 Display System, User's Manual, Comtal Corp., California.

3. Mimaroglu, T., "A High-Speed Two-Dimensional Hardware Convolver for Image Processing", Proceedings of the Pattern Recognition and Image Processing Conference, pp. 386-389, Las Vegas, Nevada, 1982.

4. Kung, H.T. and Leiserson, C.E., "Systolic Arrays (for VLSI)", Sparse Matrix Proceedings, 1978.

5. Kung, H.T. and Picard, R.L., "Hardware Pipelines for Multi-Dimensional Convolution and Resampling", Proceedings of the IEEE Computer Society Workshop on Computer Architecture for Pattern Analysis and Image Database Management, pp. 273-278, November 1981.

6. Kung, H.T. and Song, S.W., "A Systolic 2-D Convolution Chip", Proceedings of the IEEE Computer Society Workshop on Computer Architecture for Pattern Analysis and Image Database Management, pp. 159-160, November 1981.

7. McCanny, J.V., McWhirter, J.G. and Wood, K., "Optimized Bit Level Systolic Array for Convolution", Proceedings of IEEE, Vol. 131, Pt. F, No. 6, October 1984.

8. McWhirter, J.G., McCanny, J.V. and Wood, K.W., "Novel Multibit Convolver/Correlator Chips Design Based on Systolic Array Principles", Proceedings of SPIE, Vol. 341, 1982.

9. McWhirter, J.G., Wood, K., Evans, R.A., McCanny, J.V. and McCabe, A.P.H., "Multibit Convolution Using a Bit Level Systolic Array", IEEE Transactions on Circuits and Systems, Vol. CAS-32, No. 1, January 1985.

10. Danielson, P.E., "Serial/Parallel Convolvers", IEEE Transactions on Computers, Vol. C-33, p. 652, July 1984.

11. Kung, H.T., "Why Systolic Architectures?", IEEE Computer, pp. 37-46, January 1982.

12. Leiserson, C.E., "Area Efficient VLSI Computations", Ph.D. Dissertation, CMU CS-82-108, Computer Science Department, Carnegie-Mellon University, Pittsburgh, Pa., USA, 1981.

13. Gerritsen, F.A., "A Comparison of the CLIP4, DAP and MPP Processor-Array Implementations", in Computing Structures for Image Processing (M.J.B. Duff, ed.), Academic Press, pp. 15-30, 1983.

14. Fountain, T.J., "CLIP4: A Progress Report", in Languages and Architectures for Image Processing (M.J.B. Duff and S. Levialdi, eds.), Academic Press, pp. 283-293, 1981.

15. Weste, N. and Eshraghian, K., "Principles of CMOS VLSI Design, A Systems Perspective", Addison Wesley, 1985.

16. Krambeck, R.H., Lee, C.M. and Law, H.-F.S. "High-Speed Compact Circuits with CMOS", IEEE Journal on Solid-State Circuits, Vol. SC-17, No. 3, pp. 614-619, June 1982.

17. Goncalves, N.F. and DeMann, H.J., "NORA: A Racefree Dynamic CMOS Technique for Pipelined Logic Structures", IEEE Journal on Solid-State Circuits, Vol. SC-18, No. 3, pp. 261-267, June 1983.

18. Boudreault, Y., "Design of a VLSI Convolver for a Robot Vision System", M. Eng. Thesis, McGill University, March 1986.

INTERNAL DISTRIBUTION

DREV R-4513/89

1 - Chief
1 - Deputy Chief
1 - Military Assistant
1 - Director Armaments Division
1 - Director Command and Control Division
1 - Director Electro-optics Division
1 - Director Energetic Materials Division
6 - Document Library
1 - Y. Boudreault (author)
1 - J.G.G. Dionne
1 - J. F. Boulter
1 - L. Sévigny
1 - A. Blanchard
1 - G. Picard
1 - A. Morin
1 - L. Demers
1 - D.S. Galbraith
1 - J.-C. Labbé
1 - D. Heckman
1 - M. Lévesque
1 - R. Charpentier
1 - L. Paquet
1 - G. Morley
1 - P. Côté
1 - J.P. Ardouin

EXTERNAL DISTRIBUTION

DREV R-4513/89

DSIS Standard Distribution

3 - DSIS
1 - CRAD