CONF-900378--1

VLSI Implementation of Moment Invariants for Automated Inspection

CONF-900378--1

DE90 005318

G. A. Armstrong
M. L. Simpson

Oak Ridge National Laboratory*
Oak Ridge, Tennessee 37831-6005

and

D. W. Bouldin
The University of Tennessee
Knoxville, Tennessee 37996

Paper to be presented to the
IEEE Twenty-Second Southeastern Symposium on System Theory
Cookeville, Tennessee

March 11-13, 1990

* Operated by Martin Marietta Energy Systems, Inc., for the U. S. Department of Energy under Contract No. DE-AC05-840R21400.

VLSI Implementation of Moment Invariants for Automated Inspection

G. A. Armstrong **M. L. Simpson**

Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831-6005

D. W. Bouldin

The University of Tennessee
Knoxville, Tennessee 37996

Abstract

This paper describes the design of a VLSI ASIC for use in automated inspection. The inspection scheme uses Hu[1] and Maitra's[2] algorithms for moment invariants. A prototype design was generated that resolved the long delay time of the multiplier by custom designing adder cells based on the Manchester carry chain. The prototype ASIC is currently being fabricated in 2.0-μm CMOS technology and has been simulated at 20 MHz. The final ASICs will be used in parallel at the board level to achieve the 230 MOPs necessary to perform the moment invariant algorithms in real time on 512x512 pixel images with 256 grey scales.

Introduction

Pattern recognition involves processing large amounts of data at speeds from 10 to 20 MHz to attain real-time performance. To efficiently execute the pattern recognition process, the images must be reduced to a small subset of the original data while maintaining a unique description of the image[3]. The moment invariant algorithms as developed by Hu and Maitra reduce the image to six descriptive constants that are invariant to changes in translation, rotation, scaling, contrast, and illumination. The algorithms require all combinations of the zero- through third-order moments for the image to be derived.

Hu's moment invariant algorithm uses the normalized central moments of orders 2 and 3 as input to seven algebraic equations [(1) through (7)] as defined:

$$\phi_1 = \eta_{20} + \eta_{02}; \tag{1}$$

$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2; \tag{2}$$

$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2; \tag{3}$$

$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2; \tag{4}$$

$$\phi_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})\left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})\left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right]; \tag{5}$$

$$\phi_6 = (\eta_{20} - \eta_{02})\left[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}); \tag{6}$$

and

$$\phi_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})\left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})\left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right]. \tag{7}$$

This method produces seven constants for use in pattern recognition that are invariant to changes in orientation, rotation, position, size, and parallel projection (focus). Maitra proposes using the following six equations [(8) through (13)] that produce six descriptive constants that are also invariant to changes in illumination and contrast:

$$\beta(1) = \frac{\sqrt{\phi(2)}}{\phi(1)}; \tag{8}$$

$$\beta(2) = \frac{\phi(3)\mu_{00}}{\phi(2)\phi(1)}; \tag{9}$$

$$\beta(3) = \frac{\phi(4)}{\phi(3)}; \tag{10}$$

$$\beta(4) = \frac{\sqrt{\phi(5)}}{\phi(4)}; \tag{11}$$

$$\beta(5) = \frac{\phi(6)}{\phi(4)\phi(1)}; \tag{12}$$

and

$$\beta(6) = \frac{\phi(7)}{\phi(5)}. \tag{13}$$

For the 256 grey scale 512x512 image to be processed at real time, the pixels must be processed at a stream rate of 7.86 MBps. To derive the necessary moments, the ASICs will be required to perform 230 MOPs. Implementation of the moment derivation by discrete hardware would require a large amount of power and would put constraints on the possible architectures that the designer can select. The pelthora of tools available for custom ASIC design and the capability to simulate the final design makes VLSI design extremely competitive with the development of discrete hardware.

Image-Processing Architectures

This section presents architectures that are considered for implementing pattern recognition. A high priority is given to architectures that allow for a high degree of replication of internal cells while maintaining a simplistic approach to deriving the moment. Array-based architectures that allow for a maximum amount of parallelism, while maintaining simple interconnection paths between array cells, will be sought for implementation of the moment generator. A few of the architectures used in pattern-recognition and image-processing applications are presented in this section. The conclusion proposes an architecture that will allow the optimal combination of performance, area, and regularity of design for a moment generator in the pattern recognition algorithm.

VLSI designs are incorporated into a system when the solution by conventional methods is very time consuming and when the algorithm used to solve the problem can be broken down into simple modules that implemented with a high amount of repetition. As a result, most VLSI architectures use parallelism through arrays and pipelining to optimize the performance[4].

The most typical architectures used in image processing are the one-dimensional and two-dimensional arrays. The one-dimensional linear array and the two-dimensional square and hexagonal arrays are mesh-connected processor arrays. The information paths between these arrays are simple and regular and, therefore, do not require long development times.

Another commonly used architecture is the tree structure. The tree architecture is more complex than the array architectures. For a tree array to be needed for a given application, the problem to be implemented must have a growth pattern that increases exponentially. The node processors in the tree must be able to break down the problem into two more parts. The communications bandwidth of this type of architecture is concentrated at the lowest levels. If a given application requires more communication at the top level than is provided by the one communication path to the root node of the tree, the process will not be able to take advantage of the capabilities of this architecture. The tree structure is particularly useful in applications requiring a search process. The Wallace[5] tree adder is an example of a tree architecture used in multiplier circuits to reduce the propagation time of a long adder bank.

Systolic arrays are a popular architecture used in image processing. Systolic arrays attempt to use data in such a manner to maximize computational efficiency while minimizing throughput. This is achieved by ordering the data so that they are used effectively by each cell that they pass through. In a typical application, a systolic array will be able to produce one piece of output data for every piece of input data clocked into the array. This also stipulates that a systolic-based design must be synchronous rather than transparent or asynchronous. For the multiply function this would mean that for every pair of operands entered into the systolic-based two-dimensional multiplier array, one product would be produced. The series of partial products for a series of given multiplications would ripple through the systolic array in the same manner that a series of waves would ripple through a pond. The method uses the principle of pipelining to allow several multiplications to occur in parallel in the array. As a result, a high level of computational efficiency can be attained with a low I/O bandwidth. To allow the cells to be utilized in this way, the data have to be preprocessed before they can be input into the systolic array and post-processed on the way out. The internal pipelining, preprocessing, and postprocessing will require a large number of register cells. In addition, a pipelined multiplication array will require that a great deal of attention be given to the design of a clock distribution network to ensure that clock skew does not impair the operation.

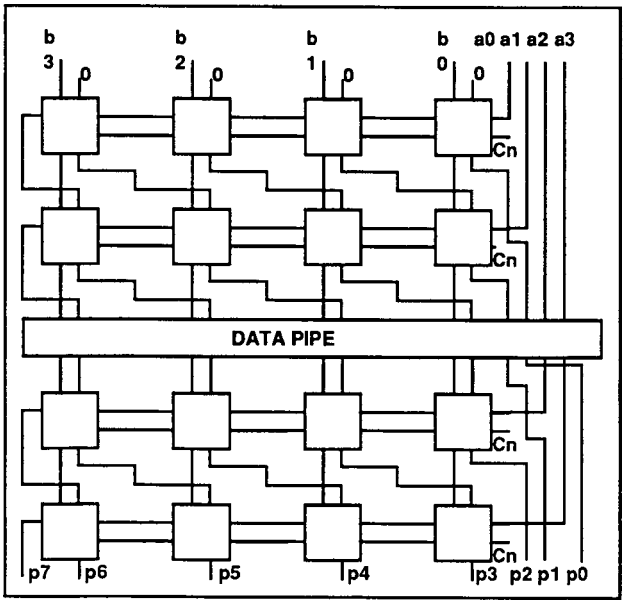

Fig. 1. Pipelined simple multiplier array architecture.

Multiplication Architectures

Figure 1 illustrates a simple array used to implement a multiplication operation. Each element in the array is composed of an adder cell, the exact composition of which does not have to be defined at this point. This array has two input data streams and attempts to make full use of each data item as it passes through the cells. The connections implement a carry-propagate adder (CPA) approach. The effect of the long propagate time for the carry can be reduced by the addition of carry lookahead (CLA) logic that can be added to each row of adders at the expense of an increase in area (typically 60%) and the loss of regularity in the design of the CLA function[6].

The propagation time through the worst case timing path, referred to as the critical path, becomes a limitation of this design. The maximum size of the multiplier becomes limited to the maximum time allotted for the multiplication and the number of cells that the carry signal can propagate through in this time. The critical-path time can be reduced by the addition of pipes in the multiplier array as shown in Fig. 1. This breaks the multiplier array into smaller segments, which allows larger arrays to perform in a shorter period of time by allowing parts of the multiplication to occur in parallel. In addition, the data pipe offers the extra advantage of providing a scan path for implementing built-in testing.

Carry-save adders (CSAs) also could be used to further decrease the dependence of the multiplier array on the size of the multiplication operands. In the CSA approach, each adder cell takes the same three inputs and presents two results to the next row of adders. (The CPA technique takes three inputs and presents one result.) The propagation time for the carry is reduced to the time for the adder to add the three inputs to produce the two results. The last stage of the multiplier would have to be a CPA, but the time delay caused by this technique could be reduced further by the addition of a CLA circuit to the final stage.

One typical approach to reducing the long propagation times associated with multiplier designs based on simple arrays makes extensive use of pipelining to create a systolic-like architecture for multiplication[7,8]. The multiplier arrays typically use CSAs and CLA functions to reduce the time required to compute each partial product of a heavily pipelined array. Pipelined multiplier arrays can use N additional stages (making a NxN multiplier require 2N stages) of half adders to eliminate the use of a CPA bank (and therefore the need for a CLA function) for the last stage of the multiplier. This approach reduces the maximum delay to that of a full adder cell and is a very regular design at the cost of approximately two times the increase in area over a similar approach using a CPA for the last bank of adders.

Another popular multiplier architecture approach is the modified Booth's algorithm[9]. This technique is typically implemented with multiple pipeline stages and can reduce the number of adders to approximately half of what is required for a simple multiplier array approach. A NxN-bit multiplier requires (N+2)/2 stages (with one adder/subtractor per stage). This approach lacks the regularity of the simple multiplier approach and, as a result, requires a long time to design and develop.

The moment generator ASIC presented in this paper is based on a asynchronous simple multiplier array that requires no clocking scheme. The TTL I/O pads of the MOSIS Tiny Chip limits the speed of parallel data transfer into the ASIC to 10 to 20 MHz. This translates to 100 to 50 ns per pipelined function in the ASIC. By implementing the multiplier as an asynchronous design, the clock timing and skew problems of synchronous pipelined designs are bypassed. The disadvantage becomes the long amount of time required to implement a large multiply operation. This requires that the adder cells used in the multiplier array minimize the propagation time of data through the cell and particular attention is given to minimize the carry propagation effects through the adder banks.

The data path design of the multiplier and summation units at the ASIC level allow the ASICs to be cascaded at the board level to derive the higher-order moments. The ASIC is designed such that a first-order moment calculation or a multiplication with no summation can be performed on each ASIC. This would allow the ASICs to be cascaded together to perform the second- and third-order moment calculations necessary for the moment invariants algorithms. The dual function capability of the ASIC would allow the ASICs to be arrayed where each row in the array would generate a different order moment of the image.

The CPA implementation, based on the two-dimensional hexagonal array, was selected. The CPA was chosen over the CSA because of its capability to read the actual intermediate result in each intermediate pipe. This will greatly simplify simulation and testing of the chip because the data going into and coming out of the scan path will not have to be encoded and decoded.

Circuit Design and Layout

Figure 2 depicts the ASIC layout of the moment generator during the initial design phase. The initial layout served as a guide for the desired size of the custom cells. No effort was made to minimize the ASIC area in the design of the cells. The major concern was given to designing fast cells that would be resistant to the effects of noise and loading in an effort to understand the practical problems associated with CMOS VLSI design. As a result, minimum-sized devices were used where possible, but no

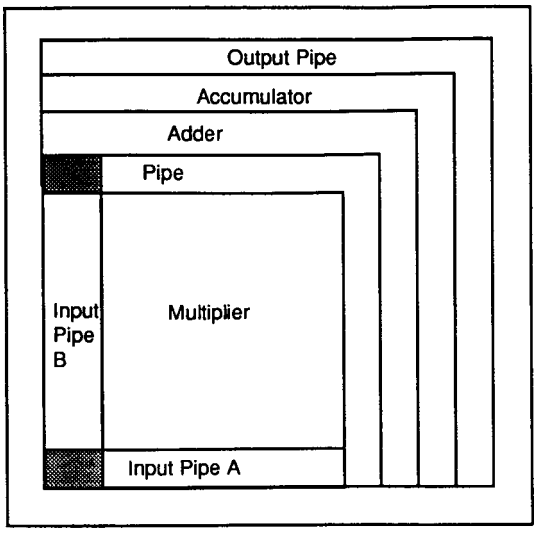

Fig. 2. Initial design for ASIC layout.

optimization was performed on the cell in an effort to reduce the area. Future designs will place more emphasis on minimizing cell area and attempting to reduce the dependency on the input and output buffers used in the cells.

The static design of the adder cells as well as the register cells will allow the test vectors used to test the summation function and the multiplier functions to be scanned at a reduced clock rate with no fear of loosing data because of noise or current leakage at the node. This is the advantage of using static designs over dynamic designs. A dynamic design would stipulate a minimum clock frequency that would need to be running at all times to keep internal storage nodes in the registers from loosing data.

CLAs can be used to decrease the time required for the carry signal to propagate through the adder. This scheme involves computing carries for several sets of bits rather than just one pair at a time. The size of the grouping can be optimized for the application but usually is considered optimum for groups of four, especially if buffers are not included in the carry path. This approach has the disadvantage of adding a significant amount of circuitry to the design. The adder cells can be designed with a method similar to that used to implement the lookahead adder but on a two-bit basis rather than four or more. This is the basis for the Manchester carry chain conditions[10].

Manchester Carry Conditions

Manchester carry conditions concentrate on minimizing the propagation time through the adder cells by studying the adder function solely from the carry signal viewpoint. Given two words, A and B, that must be added together to produce a sum, any two-bit adder cell within the adder bank can have a carry-in signal or a carry-out signal. The conditions of the input operand bits Ai and Bi can have effects on the carry-out signal that are completely independent of the carry-in signal. Equations (14) through (16) summarize these conditions.

$$K_i = \overline{A_i B_i} \text{ (Kill Condition) ;} \quad (14)$$

$$G_i = A_i B_i \text{ (Generate Condition) ;} \quad (15)$$

and

$$P_i = A_i \oplus B_i \text{ (Propagate Condition) .} \quad (16)$$

The summation is expressed as the result of the exclusive-OR (EXOR) of the two operand bits Ai and Bi along with the carry-in signal in an analogous manner to the way summation is performed in the full adder.

$$S_i = P_i \oplus C_{i-1} \text{ (Summation) .} \quad (17)$$

Performance and Test Results

Because of space and time limitations, a prototype ASIC, that was composed of an 8x4 multiplier and an 11-bit accumulator, was designed and simulated and is currently being fabricated.

The worst case propagation delay through the multiplier can be calculated by multiplying the carry propagation time delay of 2.5 ns by 8, and adding that to the product of the SumIn to SumOutBuf propagation delay of 9.5 ns multiplied by 3. This makes the worst case propagation delay through the multiplier 48.5 ns, which was also verified through simulation of the prototype 4x8-bit multiplier. As a result, the moment generator inputs and internal pipes can be clocked at a maximum clock rate of 20 MHz. With the TTL I/O pads of the ASIC, the input data rates will be limited to 10 to 20 MHz.

The worst case multiplication times for the final design can be calculated with the same method used to estimate the prototype. The final design will require a 26x9 multiplier array. Multiplying the carry propagation time of 2.5 ns by 26 allows 65 ns for the carry signal to propagate the maximum length of the first adder bank. Multiplying the worst case internal propagation time through the adder of 9.5 ns by 8 yields 76 ns. The sum of these two propagation times is 141 ns. The longest time allowed for any operation in the moment generator is 127 ns. To keep the final design within this limit, a multiplier based on the custom adder cells used in this design must be broken into two 26x5 multiplier array stages. This can be done by inserting another pipe in the middle of the array. The new worst case propagation time is 112 ns.

Conclusions

Key Features	
Custom cells	
Transistors	3500
Area	1800x1800-μm^2
I/O pins	40 I/O pins
Technology	Double metal 2.0 μm CMOS
Speed	Simualted 20 MHz
Power	Simulated 70 milliwatts

The design and simulation of the cells and architecture used to make the moment generator ASIC has given the authors good insight to the capability of CMOS VLSI technology for reliable high-speed operation.

By combining the performance of custom VLSI special-purpose processors with pattern recognition algorithms, such as moment invariants used in image processing, high-performance computational engines that can process digitized images in real time become possible. This paper describes the design of a prototype ASIC used to implement ordinary and centralized moment calculations used for moment invariant algorithms for pattern recognition applications. The determination of a suitable architecture and the design and simulation of the cells used to build the basic functions through the simulation of the final moment generator ASIC have been presented.

Space limitations on the chip and the number of I/O pins held the design effort to a proof-of-principle effort, which was composed of a multiply function and a summation function necessary to do an ordinary moment calculation.

A fundamental knowledge of the MOSFET mechanics aided in understanding the simulation tools used in the design and simulation of the cells. Extensive simulation of the cells as well as simulations of the complete design gave a high degree of confidence in the success of the fabricated ASIC.

At the chip level, a regular structured design allowed a hierarchy of moment generator ASICs to be used to compute all the moments necessary for the moment invariant algorithms. This design allowed the same ASIC design, at the chip level, to be used multiple times and for different purposes. At the silicon level, a structured-design approach divided the two main functions into two halves of a data-path-based design that allowed the moment at the silicon level to be calculated without the use of a state machine. The data-path design, without any state machines, meant no synchronous feedback paths, which eliminated the need for a two-phase nonoverlapping clock scheme. The design was implemented with a single-phase complementary clock scheme, which reduced the amount of clocking circuitry required in the synchronous register cells. The use of a parallel register as a pipe between the multiplication and summation function made the implementation of a scan path for testing the output of the multiplier and feeding the input to the summation function inexpensive in design effort and in terms of required real estate on the chip. The accumulation register in the summation function allowed the incorporation of a scan path in this function also.

Future Work

As soon as the ASIC is thoroughly tested, a final moment generator will be designed and simulated. The final moment generator will allow the target image (256 grey-scale 512x512 pixels) to be computed at frame rates that can be made with the cells designed in the prototype chip on a larger platform. After getting some feedback as to the success of the custom cells used in the prototype, smaller and faster custom cells and possible improvements on the architecture will be made to achieve much higher data rates in smaller area. The chip necessary to implement a 9x26 multiplier and a 48-bit adder accumulator would need to be much larger and have more I/O pins than provided on the 40-pin 1800x1800 NSF-supplied platform.

References

[1] M. K. Hu, "Visual Pattern Recognition by Moment Invariants," *IRE Transactions on Information Theory*, pp. 179-187, February 1962.

[2] S. Maitra, "Moment Invariants," *Proceedings of the IEEE*, Vol. 67, No. 4, April 1979.

[3] M. L. Simpson, *Moment Invariants for Automated Inspection of Printed Material*, submitted for publication, Instrumentation and Controls Division, Oak Ridge National Laboratory, 1989.

[4] King-sun Fu, *VLSI for Pattern Recognition and Image Processing*, Spring Series in Information Sciences, New York, Tokyo, Springer-Verlag Berlin Heidelberg, 1984.

[5] M. J. M. Pelgrom, H. E. J. Wulms, P. Stokker Van Der, and R. A. Bergamaschi, "FEBRIS: A Chip for Pattern Recognition," *IEEE Journal of Solid-State Circuits,* VOL. SC-22, No. 3, June 1987.

[6] O. L. MacSorley, "High-Speed Arithmetic in Binary Computers," *Proceedings of the IRE,* pp. 67-91, January 1961.

[7] M. Hatamian and G. L. Cash, "A 70-MHz 8-bit X 8-bit Parallel Pipelined Multiplier in 2.5-μm CMOS," *IEEE Journal of Solid-State Circuits*, vol. SC-21, no. 4, August 1986.

[8] T. G. Noll, et al, "A Pipelined 330-MHz Multiplier," *IEEE Journal of Solid-State Circuits*, vol. SC-21, no. 3, June 1986.

[9] N. R. Shanbhag, and P. Juneja, "Parallel Implementation of a 4x4-bit Multiplier Using Modified Booth's Algorithm," *IEEE Journal of Solid-State Circuits*, vol. 23, no. 4, August 1988.

[10] L. A. Glasser, and D. W. Dobberpuhl, *The Design and Analysis of VLSI Circuits,* Addison-Wesley, Mass., 1985.