Proceedings

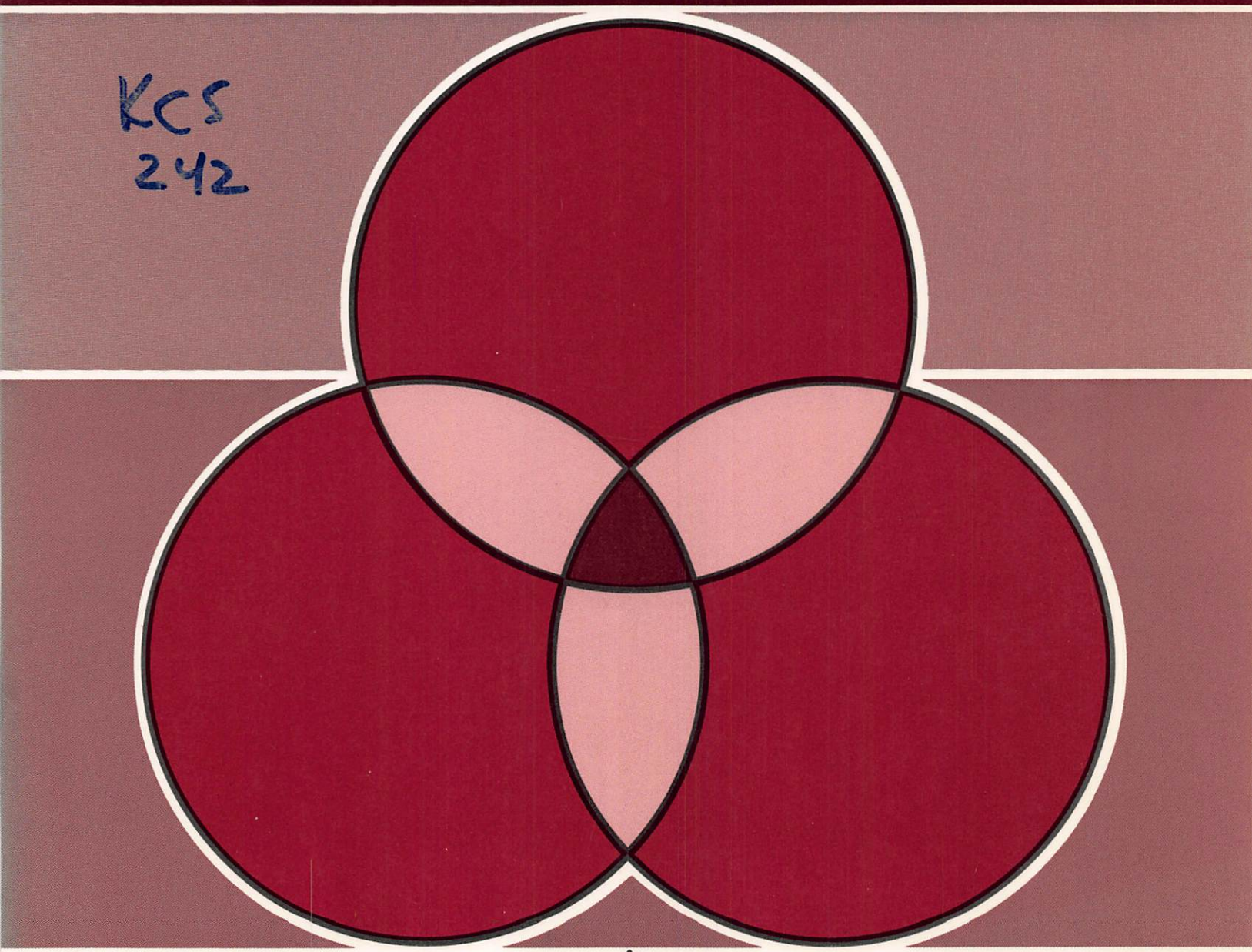
IEEE INTERNATIONAL CONFERENCE ON Computer Design: VLSI in Computers

Sponsored by **IEEE Computer Society and IEEE Circuits and Systems Society**

ISBN 0-8186-0642-8
IEEE Catalog Number 85CH2223-6
Library of Congress Number 85-62335
IEEE Computer Society Order Number 642

ICCD' 85

Rye Town Hilton, Port Chester, New York
October 7–October 10, 1985



IEEE COMPUTER SOCIETY

IEEE
THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

IEEE COMPUTER SOCIETY PRESS

PROBABILISTIC SYSTOLIC ARRAYS*

Marius V.A. Hâncu and Kenneth C. Smith

Department of Electrical Engineering, University of Toronto

Toronto, Ontario M5S 1A4, Canada

Abstract

A random sequence generator is proposed to condition the clock distribution in linear and bi-dimensional systolic arrays. The streams of data so provided to the processing elements are thus subjected to probabilistic selection and propagation. Applications include running probabilistic (randomized) algorithms, data encryption and digital signal processing.

1. Introduction

Systolic arrays[1,2] have seen an increasingly widespread use as highly parallel computing organizations suitable for VLSI implementation of specialized processors for various applications.

In a majority of designs, control of the movement of data is synchronous[2]. Characteristically, a multiphase clock is used to direct the sequence of incremental data transfer and processing steps. Thus, the clock line is the only acceptable global signal connection, all data transfers being of a local, neighbour-to-neighbour, nature.

Under these assumptions, data flow within a systolic array is in highly organized and predictable (deterministic) streams, where data interactions occur in the *PE*'s (processing elements) "while they travel" through the array. For each data item, the inherent assumption is made that it can be localized at any moment in time and space. This means that we can specify the propitious moments at which to inject data into the array (from the interface with the host computer) and the exact operands to be processed by each *PE* at a particular time. Correspondingly, we can predict the full composition and timing of the stream of output results.

While this deterministic data propagation and processing approach is quite appropriate in many applications, there exist applications where there is a need for a probabilistic (randomized) treatment of data.

Thus, when implementing probabilistic algorithms[3], some random process, such as random sampling, is necessary in addition to an otherwise deterministic computation. Such a step can be also used in data encryption where the multiplication (or "modulation") of data (before transmission) via a random (or pseudo-random) generator is frequently employed.

If large amounts of data are to be processed in this way a systolic array will remain the VLSI solution of choice. This is especially true for reasons of intercompatibility, in the event that the rest of the VLSI chip under design is conceived as a systolic structure.

In this paper we propose systolic structures able to perform such random operations. The major modification required involves a random conditioning of the clock signal reaching the input data registers in each *PE*.

*Ref. no. 85052410.

In the modified system, the systolic rhytmicity of the data flow and the functions of each *PE* are conserved, but the operands are sampled from the (passing) data streams on a random basis. This process is equivalent to the modulation of the data stream with a random sequence.

2. Randomization of a Linear Systolic Array

In the "randomized scheme", the original systolic array of *PE*'s is paralleled ("cushioned") by a shift register, meant to serve as a medium for a travelling random sequence (Fig.1). We assume that the random sequence is shifted through the register on clock phase Φ1 while the *PE*'s accept input data on clock phase Φ2. For simplicity, only one input data stream is presented. The *PE*'s perform any of the predefined operations on this data stream, combining this data eventually with other data streams, travelling on the same left-to-right direction (or alternately, coming from the right). The way *the results* accumulate and travel through the array is irrelevant to the point we are presently making.

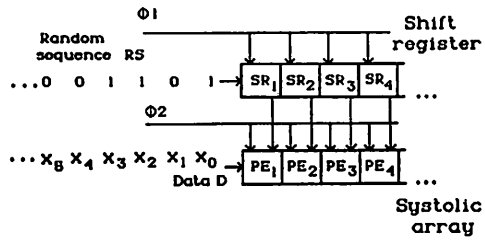


Fig. 1. Random conditioning of input data in the systolic array.

The random sequence ripples from left to right through the shift register cells (*SR*) under the control of Φ1. The outputs of the *SR* cells are AND-ed with Φ2 inside each *PE*, controlling the sampling of new data in the input data register(s) of each processor element.

Thus, only the data input registers receiving 1 on their clock inputs as a result of the above-mentioned AND will be updated. The rest of the *PE*'s will remain with the content of their input registers unmodified.

We present in Fig.2 successive displacements of the random sequence (RS) exemplified in Fig.1 through the conditioning shift register in parallel with the evolution of the input data stream through the *PE*'s, during six successive clock periods. We assume that initially the input data registers (D) have been reset.

(after) Φ_{1_1}	RS:	1	0	0	0	0	0	...
Φ_{2_1}	D:	X_0	0	0	0	0	0	...
Φ_{1_2}	RS:	0	1	0	0	0	0	...
Φ_{2_2}	D:	X_0	X_0	0	0	0	0	...
Φ_{1_3}	RS:	1	0	1	0	0	0	...
Φ_{2_3}	D:	X_2	X_0	X_0	0	0	0	...
Φ_{1_4}	RS:	1	1	0	1	0	0	...
Φ_{2_4}	D:	X_3	X_2	X_0	X_0	0	0	...
Φ_{1_5}	RS:	0	1	1	0	1	0	...
Φ_{2_5}	D:	X_3	X_3	X_2	X_0	X_0	0	...
Φ_{1_6}	RS:	0	0	1	1	0	1	...
Φ_{2_6}	D:	X_3	X_3	X_3	X_2	X_0	X_0	...

Fig. 2. Random sequence and input data movement in the systolic array.

From this example, it can be seen that the data advance systolically through the array of processors, in step with the movement of the random sequence through the shift register. However *some input data are dismissed on a random basis*. Those dismissed are data for which PE_1 gets a 0 from SR_1 at the time when sampling from the host processor interface occurs. At the same time, also on a random basis (corresponding to the distribution of 1's and 0's in the random sequence), some data are replicated a number of times (we might say that their weight in the input data stream has been increased). They replace the data dismissed from the stream.

Other random transformations can be applied to the input data stream by minor modifications of the logic function by which the outputs of *SR* cells and Φ2 are processed. For example, for each zero in the random sequence travelling through the shift register we might force a reset of the input data register in the corresponding *PE*. Randomly distributed groups of *zeros* will thus substitute for the dismissed x_i values.

3. Randomization of Multiple Input Data Streams

To avoid any intercorrelation, multiple input data streams could be conditioned by separate (independent) random sequence generators. Extra layers of shift register cells should be necessary in such cases (a schematic representation is given in Fig. 3).

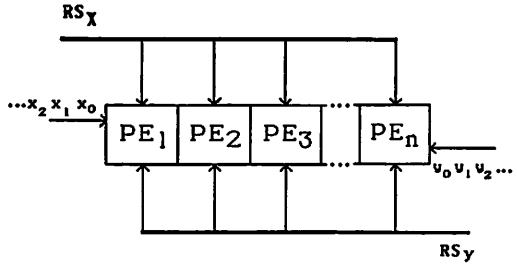


Fig. 3. Randomization of multi-input data streams.

For the two-dimensional systolic arrays, a different circulation of the random sequence would minimize the number of independent random sequence generators necessary (Fig. 4). The a's and b's are the input data streams, while D's represent register cells. The clock distribution is not represented, but it is similar to the one shown in Fig. 1.

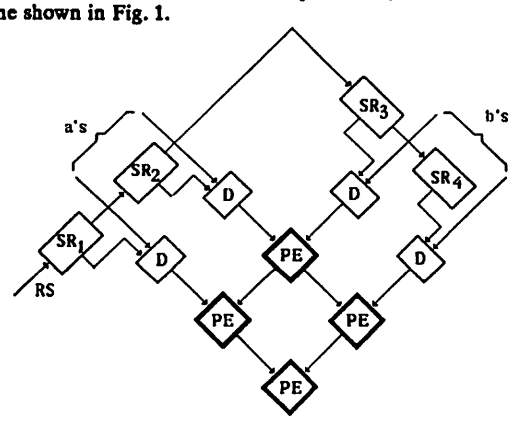


Fig. 4. Randomization of a two-dimensional systolic array.

It is obvious that this scheme could have been used also with the linear array depicted in Fig.1 and Fig.3, in which case the shift-register is reduced to two randomly conditioned flip-flops.

However, it is believed that the availability of one or more random sequences *at the PE level* will increase the number of novel possibilities for random conditioning. For example, we might want to "randomly modulate" the very *functions* implemented by the *PE*'s. In order to reduce cross-correlation effects it is recommended, in this case too, that the random sequences involved be statistically independent.

The overhead of implementing the extra conditioning shift register layers and, eventually, (pseudo) random generators on chip is dependent on the complexity of the *PE*s and the overall structure of the systolic array. However, we estimate it should not represent more than 15% of the overall costs.

4. Conclusions

We have proposed a new type of systolic array, *the probabilistic systolic array*, in which data streams and/or functions are "modulated" by a random sequence. The implementation is fully compatible with VLSI systolic arrays presently contemplated. Possible applications lie in the areas of VLSI implementation of probabilistic algorithms, data encryption, data compression and digital signal processing.

5. Acknowledgment

This research was supported by the Natural Sciences and Engineering Research Council of Canada under Grant A1753.

REFERENCES

1. H.T. Kung and C.E. Leiserson, "Systolic Arrays for VLSI", in C.A. Mead and L.A. Conway, *Introduction to VLSI systems*, Addison-Wesley, Reading, MA, 1980.
2. C.E. Leiserson, "Systolic and Semisystolic Design", *Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers (ICCD '83)*, 1983, pp.627-632.
3. M.O. Rabin, "Probabilistic Algorithms", in J.F. Traub, (Ed.), *Algorithms and Complexity: New Directions and Recent Results*, Academic Press, New York, 1976.
4. S.Y. Kung, H.J. Whitehouse and T. Kailath, (Eds.), *VLSI and Modern Signal Processing*, Prentice Hall, Englewood Cliffs, NJ, 1985.