Systolic Origins of the Polyhedral Model

S. Rajopadhye*

This is a brief survey of the development of the polyhedral model based on the work on systolic synthesis from recurrence equations and recent extensions. It does not pretend to be complete, and is definitely *not* a survey of all systolic synthesis research—many results are not even mentioned. The objective is to enable the non-specialist to trace the main developments leading to the use of a linear algebraic framework, and polyhedra in particular. References in the papers cited here will give a more complete picture.

The starting point of systolic synthesis is a single *recurrence equation* (RE), or a *system* of recurrence equations (SRE). A single RE specifies in a purely functional manner, how the value of a variable at a point in an index space is defined (recursively) in terms of its value at some other points. In an SRE, there is more than one variable, and the equations are in general, mutually recursive (REs are thus a proper subset of SREs). The *domain* of an RE is the set of index points at which it is defined. Each variable in an SRE may have its own domain. Some authors have investigated SREs whose domains are polyhedral, while others do not make such restrictions. Since an SRE is purely functional, all its *dependencies* are flow dependencies, and are obtained directly (syntactically) from the SRE. The dependencies may be either simple translations (for *uniform recurrences*, or SUREs), or affine functions of the index point (SAREs). Thus there is a wide variety in the nature of the initial specification.

Systolic synthesis consists essentially of assigning to every computation, a *time* and a *place*, in such a manner that the dependencies become local and may be interpreted as dedicated interconnection links between processors. This can be more or less involved, depending on the initial SRE.

We now discuss four main aspects of systolic synthesis, namely scheduling, optimality, communication/localization and transformations. We do not discuss allocation of computations to processors (the so-called *allocation functions*).

1 A Brief History of Time

To *schedule* an SRE is to assign a time instant at which every computation specified by the SRE can be performed. For systolic synthesis, one seeks schedules that are expressed as

*IRISA, Campus de Beaulieu, 35042 Rennes Cedex, France

(one-dimensional) affine functions of the index points, although their generalization to multi-dimensional schedules is interesting for parallelizing compilation.

Undoubtedly, the most important work on this subject is that of Karp, Miller and Winograd [KMW67], which predates systolic arrays by well over a decade! They studied (in fact, defined) SUREs over a particular class of domains (the domain of all variables is the same and is the entire positive orthant), and gave techniques for determining (one- and multi-dimensional) affine schedules[1]. This work is only recently being matched and improved.

Though many people did systolic synthesis by space time transformations in the early 80's, the first use of recurrence equations was in 1983 by Quinton [Qui87], who studied a single URE[2] defined over a polyhedral index space. He expressed the space of one-dimensional affine schedules as the feasible space of an integer linear programming problem, thus using establishing the link to the work of Karp et al. Rao [Rao85] and Roychowdhury [Roy88] investigated multidimensional schedules for SUREs, and Rao also improved the algorithm given by Karp et al.

Delosme and Ipsen first studied the scheduling problems for *affine* REs (they defined the term AREs), without considering the domains over which they were defined [DI86]. They showed that all (one-dimensional) affine schedules belong to a cone. Rajopadhye et al. addressed the problem of scheduling a single ARE defined over a polyhedral index domain and showed that the space of valid one dimensional affine schedules is described in terms of linear inequalities involving the extremal points (vertices and rays) of the domain [RPF86, RF90]. Quinton and van Dongen also obtained a somewhat tighter result [QV89]. Mauras et al. extended this result to *systems* of AREs, for which they proposed *variable dependent* schedules (now also called affine-by-variable or affine-by-statement schedules) [MQRS90]. Rajopadhye et al. further extended this and proposed *piecewise* affine schedules for SAREs [RMK92].

Feautrier [Fea92b] gave an alternative formulation using Farkas' lemma, for determining (one-dimensional, variable dependent) affine schedules for a SARE. He further extended the method to multidimensional schedules [Fea92c]. Recently, Darte and Vivien [DV95] showed that multidimensional schedules as proposed by Feautrier can be obtained (for SUREs) by a dual linear programming formulation of the one used by Karp et al. They also showed that what Feautrier proposed as a greedy heuristic, turns out to be optimal. Furthermore, they showed by introducing what are called *conic approximations* of the dependencies, one can reduce the SARE scheduling problem to that of scheduling SUREs [DV94].

At the other end of the spectrum, Joinnault [GJ87] showed that the problem of scheduling an SURE whose variables are defined over arbitrary domains is undecidable. Quinton and Saouter have shown [SQ93] that scheduling even parameterized families of SAREs (each of whose domains is bounded) is also undecidable.

[1] Today we know them to be multi-dimensional schedules, but this understanding was slow in coming.

[2] Actually Quinton worked on a a particular restricted form of SUREs, but this can always be viewed as a single URE for analysis purposes.

2 Optimality

Apart from [KMW67] and and the recent results on multidimensional schedules, none of the work above addressed the problem of optimality of the schedules. The emphasis was on defining the space of feasible schedules, rather than the choice of schedule. In general, there are two notions of optimality: minimizing the total completion time (which is relevant for parallelizing compilers), and minimizing the throughput or the *rate* at which outputs are produced (this is useful typically in signal and/or image processing applications where the domains are infinite).

Karp et al. had shown that the space of linear schedules for a single URE (over the positive orthant) satisfies a finite number of linear inequalities. By adding a cost function one can obtain an optimal linear schedule. They also showed that the optimal linear schedule was close to being globally optimal (i.e., it was close to the fastest possible schedule). Shang and Fortes showed that for a single URE over a polyhedral index space (actually, they used nested loop programs, but the ideas carry over easily), the optimal linear schedules can be determined by solving a finite number of linear programming problems [SF91]. Darte, Kachiyan and Robert linked these two results by showing that such optimal linear schedules for a single URE over a polyhedral domain was also close to being globally optimal [DKR91]. Darte and Robert formulated linear programming problems to determine the optimal variable-dependent one-dimensional affine schedule for a SURE defined over *parameterized* polyhedral domains [DR94a], and also for SAREs.

Feautrier [Fea92b] and Darte and Robert [DR95] extended these ideas to one-dimensional affine schedules for a single ARE and variable-dependent affine schedules for SAREs over parameterized families of domains (piecewise affine schedules for Feautrier). Both papers are excellent tours de force (though Darte and Robert incorrectly credit Rajopadhye [RF90] and Rao [Rao85] for inventing variable-dependent affine schedules: Rajopadhye studied only a single ARE, and Rao studied multidimensional schedules for SUREs).

Optimality of multidimensional schedules for SUREs (up to a multiplicative factor) is assured by Karp et al. and was reiterated by Darte and Vivien [DV95]. Optimal multidimensional schedules for SAREs (parameterized or not) is an open question.

3 Localization/Communication

In systolic array synthesis, the target architecture imposes constraints of locality in space as well as in time. It turns out that if one uses affine schedules and affine allocation functions, and apply them to UREs (which are inherently local), one preserves locality. However, AREs are a different story, since their dependencies are not local to start with. Fortes and Moldovan [FM84] first identified (not in the framework of SREs, but the ideas carry over) that broadcasts correspond to singular (or rank-deficient) indexing functions where multiple uses of the same data value are scheduled at the same instant. They also gave methods to eliminate (or reduce) broadcasts, by appropriately changing the schedule and/or the processor allocation.

Rajopadhye et al. [RPF86] gave a constructive method for localization (or broadcast elimination) by propagation along a basis vector of the null space of the dependency matrix. A similar idea was presented by Quinton and Van Dongen [QV89]. Roychowdhury et al. [Roy88, RTRK88] investigated other cases of localization. In particular they showed that null space propagation works if and only if the dependency matrix is idempotent, and gave a method to localize *any* singular dependencies when this does not hold. Wong and Delosme [WD88] also presented different techniques for this problem. Rajopadhye described *multistage pipelining* where the propagation of the values needed for a particular dependency made use of propagating values of *another* dependency [Raj89].

Yaacoby and Cappello [YC88] considered an interesting case, namely when the dependency matrix is non-singular (obviously, there is no broadcast, since the dependency is invertible). They showed that whenever the dependency matrix, A, is the generator of a group and hence, there exists an positive integer, k such that $A^k = I$, there are precisely $k-1$ *folds* in the iteration space that can be used to localize the dependency. Folding was also initially studied by Choffrut and Culik [CC83]. A classic and simple example is the transpose, where the computation at point $[i,j]$ depends on the point $[j,i]$. Here $A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, and $A^2 = I$, and sure enough, folding the domain about the $i = j$ line will localize the dependency!

Many kinds of static analysis of communication can be done in the linear algebraic framework of the array accessing functions of a system of recurrences. Rajopadhye defined a SARE-like notation for specifying communication and illustrated these ideas [Raj93].

4 Transformations/Languages

The early work on systolic synthesis from UREs was based on determining a unimodular transformation, Λ, based on the schedule and the allocation function. Then, for each dependency vector d, the corresponding interconnection link was simply Λd. It was known that the space-time description of a systolic array was a URE, or more precisely, a SURE with a one dimensional affine schedule where all variables had the *same* linear part (Rao, who showed this formally [Rao85], called them UREs with a strongly separating hyperplane). Initially, there was no effort on obtaining this SURE by a systematic transformation of the original specification. This started to change when researchers began to address AREs, since the specifications became more and more complicated due to the pipelining (localization) variables.

Rajopadhye et al. [RPF86] showed that when a unimodular space time transformation Λ is applied to an ARE with a dependency, $Az + a$, the (linear part of) the new dependency becomes $\Lambda A \Lambda^{-1}$. This also holds when applying the same unimodular transformation to all variables in a SARE.

Chen defined the language Crystal based on recurrences defined over data fields [Che86], and Choo and Chen developed a theory of domain transformations for Crystal. Unimodular space-time transformations of AREs (or all variables in an SARE) turn out to be particular cases.

Mauras defined the language ALPHA [Mau89] which can be viewed as a particular sub-language of Crystal where all domains are (possibly parameterized) polyhedra (or unions of finitely many polyhedra), and all dependencies are affine. As a result of these restrictions, *all* ALPHA programs can be simplified into a normal form, by using a special library for manipulating polyhedra. Such transformations are also possible in Crystal, but they require a special algebraic metalanguage and associated machinery. ALPHA thus achieves simplicity and greater degree of automation by sacrificing generality. Wilde's PhD thesis deals with the problem of compiling ALPHA to sequential or parallel machines [WR95, QRW95a, QRW95b, Wil95, WR96].

References

[CC83] C. Choffrut and K. Culik II. Folding of the plane and the design of systolic arrays. *Information Processing Letters*, 17:149–153, 1983.

[CGST93] S. Chatterjee, J. Gilbert, R. Schreiber, and S-H. Teng. Automatic array alignment in data-parallel programs. In *Principles of Programming Languages (POPL)*, pages 16–28, Charleston, S.C., Jan 1993. ACM.

[Che86] Marina C. Chen. A parallel language and its compilation to multiprocessor machines for VLSI. In *Principles of Programming Languages.* ACM, 1986.

[DG96] C. G. Diederich and M. Gengler. Solving the constant degree alignment problem. In *Europar 96: Parallel Processing Conference*, volume I, pages 451–454, Lyon, France, August 1996. Springer Verlag.

[DI86] Jean-Marc Delosme and Ilse C. F. Ipsen. Systolic array synthesis: Computability and time cones. In M. Cosnard, P. Quinton, Y. Robert, and M. Tchuente, editors, *Int. Workshop on Parallel Algorithms and Architectures*, pages 295–312. Elsevier Science, North Holland, April 1986.

[DKR91] A. Darte, L. Khachiyan, and Y. Robert. Linear scheduling is nearly optimal. *Parallel Processing Letters*, 1(2):73–82, December 1991.

[DR55] M. Dion and Y. Robert. Mapping affine loop nests: New results. In *High Performance Computing and Networking Conference*, volume LNCS 919, pages 184–189. Springer Verlag, 1955.

[DR94a] A. Darte and Y. Robert. Constructive methods for scheduling uniform loop nests. *IEEE Transactions on Parallel and Distributed Systems*, 5(8):814–822, Aug 1994.

[DR94b] A. Darte and Y. Robert. On the alignment problem. *Parallel Processing Letters*, 4(3):259–270, Sept 1994. Prelimininary version: LIP RR 93-20, entitled "A graph-theoretic approach to the alignment problem".

[DR95] A. Darte and Y. Robert. Affine-by statement scheduling of uniform and affine loop nests over parametric domains. *Journal of Parallel and Distributed Computing*, 29(1):43–59, February 1995.

[DV94] A. Darte and F. Vivien. Automatic parallelization based on multidimensional scheduling. Technical Report 94-24, Laboratiore de l'Informatique du Parallelisme, ENS Lyon, France, Sept 1994.

[DV95] A. Darte and F. Vivien. Revisiting the decomposition of karp, miller and winograd. In *International Conference on Application–Specific Array Processors*, pages 12–25, Strasbourg, France, July 1995. IEEE.

[Fea92a] P. Feautrier. Toward automatic distribution. Technical Report 92-95, Laboratoire MASI, Institut Blaise Pascal, Paris, Dec 1992.

[Fea92b] Paul Feautrier. Some efficient solutions to the affine scheduling problem, Part I, one-dimensional time. Technical Report 28, Labaratoire MASI, Institut Blaise Pascal, April 1992.

[Fea92c] Paul Feautrier. Some efficient solutions to the affine scheduling problem, Part II, multidimensional time. Technical Report 78, Labaratoire MASI, Institut Blaise Pascal, October 1992.

[FM84] Jose A. B. Fortes and D. Moldovan. Data broadcasting in linearly scheduled array processors. In *Proceedings, 11th Annual Symposium on Computer Architecture*, pages 224–231, 1984.

[GB91] M. Gupta and P. Bannerjee. Demonstration of automatic data partitioning techniques for parallelizing compilers on multicomputers. *IEEE Transactions on Parallel and Distributed Systems*, ?(?):179–193, Mar 1991.

[GJ87] P. Gachet and B. Joinnault. *Conception d'algorithmes et d'architectures systoliques.* PhD thesis, Université de Rennes I, September 1987. (In French).

[KLD92] K. Knobe, J. D. Lukas, and W. J. Dally. Dynamic memory alignment on distributed memory systems. In *Proc. Workshop on Compilers for Parallel Computers*, pages 394–404, Vienna, Austria, July 1992. Austrian Center for Parallel Computation.

[KLS90] K. Knobe, J. D. Lukas, and G. L. Steele. Data optimization: Allocation of arrays to reduce communication on SIMD machines. *Journal of Parallel and Distributed Computing*, 8(?):102–118, ? 1990.

[KMW67] R. M. Karp, R. E. Miller, and S. V. Winograd. The organization of computations for uniform recurrence equations. *JACM*, 14(3):563–590, July 1967.

[LC91a] J. Li and M. Chen. The data alignent phase in compiling programs for distributed memory machines. *Journal Parallel Distributed Computing*, 13:213–221, 1991.

[LC91b] J. Li and M. C. Chen. Compiling communication efficient programs for massively parallel machines. *IEEE Transactions on Parallel and Distributed Systems*, 2(3):361–376, July 1991.

[Mau89] Christophe Mauras. *ALPHA: un langage équationnel pour la conception et la programmation d'architectures parallèles synchrones.* PhD thesis, L'Université de Rennes I, IRISA, Campus de Beaulieu, Rennes, France, December 1989.

[MQRS90] Christophe Mauras, Patrice Quinton, Sanjay V. Rajopadhye, and Yannick Saouter. Scheduling affine parameterized recurrences by means of variable dependent timing functions. In S. Y. Kung and E. Swartzlander, editors, *International Conference on Application Specific Array Processing*, pages 100–110, Princeton, New Jersey, Sept 1990. IEEE Computer Society.

[QRW95a] P. Quinton, S. Rajopadhye, and D. Wilde. Derivation of data parallel code from a functional program. In *9th International Parallel Processing Symposium*, pages 766–772, Santa Barbara, CA, April 1995. IEEE.

[QRW95b] P. Quinton, S. Rajopadhye, and D. Wilde. Deriving imperative code from functional programs. In *7th Conference on Functional Programming Languages and Computer Architecture*, pages 36–44, La Jolla, CA, Jun 1995. ACM.

[Qui87] Patrice Quinton. *The Systematic Design of Systolic Arrays*, chapter 9, Automata Networks in Computer Science, pages 229–260. Princeton University Press, 1987. Preliminary versions appear as IRISA Tech Reports 193 and 216, 1983, and in the proceedings of the IEEE Symposium on Computer Architecture, 1984.

[QV89] Patrice Quinton and Vincent Van Dongen. The mapping of linear recurrence equations on regular arrays. *Journal of VLSI Signal Processing*, 1(2):95–113, 1989.

[Raj89] S. V. Rajopadhye. Synthesizing systolic arrays with control signals from recurrence equations. *Distributed Computing*, 3:88–105, May 1989.

[Raj93] S. V. Rajopadhye. LACS: A language for affine communication structures. Technical Report 712, IRISA, 35042, Rennes Cedex, April 1993.

[Rao85] Sailesh Rao. *Regular Iterative Algorithms and their Implementations on Processor Arrays.* PhD thesis, Stanford University, Information Systems Lab., Stanford, Ca, October 1985.

[RF90] S. V. Rajopadhye and R. M. Fujimoto. Synthesizing systolic arrays from recurrence equations. *Parallel Computing*, 14:163–189, June 1990. First presented as [RPF86].

[RMK92] S. V. Rajopadhye, L. Mui, and S. Kiaei. Piecewise linear schedules for recurrence equations. In K. Yao, R. Jain, W. Przytula, and J. Rabbaey, editors, *VLSI Signal Processing, V*, pages 375–384, Napa, Oct 1992. IEEE Signal Processing Society.

[Roy88] Vwani P. Roychowdhury. *Derivation, Extensions and Parallel Implementation of Regular Iterative Algorithms.* PhD thesis, Stanford University, Department of Electrical Engineering, Stanford, CA, December 1988.

[RPF86] S. V. Rajopadhye, S. Purushothaman, and R. M. Fujimoto. On synthesizing systolic arrays from recurrence equations with linear dependencies. In *Proceedings, Sixth Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 488–503, New Delhi, India, December 1986. Springer Verlag, LNCS 241. Later appeared in Parallel Computing, June 1990.

[RTRK88] Vwani Roychowdhury, Lothar Thiele, Sailesh K Rao, and Thomas Kailath. On the localization of algorithms for VLSI processor arrays. In Robert W. Brodersen and Howard S. Moscovitz, editors, *VLSI Signal Processing, III*, pages 459–470, Monterey, Ca, November 1988. IEEE Accoustics, Speech and Signal Processing Society, IEEE Press. A detailed version is submitted to IEEE Transactions on Computers.

[SF91] W. Shang and J. Fortes. Time optimal linear schedules for algorithms with uniform dependencies. *IEEE Transactions on Computers*, 40(6):723–742, June 1991. preliminary results reported in ASAP '88.

[SQ93] Y. Saouter and P. Quinton. Computability of recurrence equations. *Theoretical Computer Science*, 114, 1993. (to appear: preliminary version available as IRISA TR-1203, April, 1990).

[WD88] F. C. Wong and Jean-Marc Delosme. Broadcast removal in systolic algorithms. In *International Conference on Systolic Arrays*, pages 403–412, San Diego, CA, May 1988.

[Wil95] D. Wilde. *From Alpha to Imperative Code: A Transformational Compiler for an Array Based Functional Language.* PhD thesis, Oregon State University, July 1995.

[WR95] D. Wilde and S. Rajopadhye. The naive execution of affine recurrence equations. In *International Conference on Application-Specific Array Processors*, pages 1–12, Strasbourg, France, July 1995. IEEE.

[WR96] D. Wilde and S. Rajopadhye. Memory reuse analysis in the polyhedral model. In *EUROPAR 96: Parallel Processing Conference*, pages 389–397, Lyon, France, August 1996. Springer Verlag.

[YC88] Yoav Yaacoby and Peter R. Cappello. Converting affine recurrence equations to quasi-uniform recurrence equations. In *AWOC 1988: Third International Workshop on Parallel Computation and VLSI Theory.* Springer Verlag, June 1988. See also, UCSB Technical Report TRCS87-18, February 1988.