8. Summary

Systolic and *hyper-systolic* algorithms are schemes to parallelize the computation of a wide range of numerical and non-numerical problems which involve non-local interactions with data exchange between all processing elements. While the original motivation behind the systolic array concept was its potential as to VLSI-chip implementation, systolic algorithms meanwhile have found an attractive implementation medium by massively parallel computers. Hyper-systolic algorithms are a novel scheme recently introduced. They extend systolic algorithms in order to achieve reduction of interprocessor communication. In this thesis, they are worked out in detail.

Systolic and hyper-systolic algorithms offer a number of advantages for parallel computer applications: they can treat problems with non-local interactions whereat they only require a local regular interconnection structure as present on most of the contemporary parallel architectures. All systolic and hyper-systolic algorithms can efficiently be implemented within the SIMD (single instruction, multiple data) model, as far as the regularity of data movement and synchronous operating of all processing elements are concerned.

In this work, a virtual systolic machine model is introduced. It starts from the definition of systolic arrays as cellular automata models of parallel computing structures in which data processing and transfer are pipelined and in which the cells carry out functions of equal load between consecutive communication events. 1-dimensional rings of systolic cells constitute the basic structures that build the fundamental connectivities. 1-dimensional ring connectivities are supported by many contemporary parallel systems or they can readily be emulated via software. Therefore, 1-dimensional systolic algorithms can be mapped onto these rings in a straightforward manner which are formed by real processors. As the number of data elements and the number of processors differ in general, the mapping does not proceed in a bijective manner rather the 1-dimensional chain of systolic cells must be distributed suitably. Standard blocking strategies are given by cyclic-striped or block-striped mapping, or a combination of both approaches. In the present work, it has been demonstrated that many systolic computations can profit from a hierarchical mapping strategy: Large systolic arrays of n

cells can be split into $\frac{n}{p}$ separate systolic arrays of length p, where p is the number of processors.

The design of systolic algorithms originally was guided by the abilities of VLSI chips. Sure enough, there are striking analogies between systolic arrays and processor arrays, however, the interprocessor connectivity of most parallel systems appears to be by far more flexible than communication structures provided by VLSI circuits. Furthermore, the I/O capabilities of parallel machines are much more sophisticated than those of VLSI chips. Thus, it appears natural to take into account the enhanced connectivity of parallel machines for both I/O *and* the reduction of interprocessor communication in the course of the systolic process. Such a reduction can be achieved by a suitable re-organization of partial systolic computations.

The re-organization is based on the observation that one can assess a sort of *intrinsic redundancy* in many systolic algorithms as far as communication between systolic cells is concerned. Avoiding this redundancy has led to the development of *hyper-systolic* algorithms.

Hyper-systolic algorithms define a *novel class* of parallel computing structures. As such they can be considered as a generalization of conventional systolic algorithms, designed to minimize interprocessor communication. They inherit all advantages of systolic algorithms, and, as is the case with the latter, they are very useful on SIMD computers. However, hyper-systolic algorithms are attractive for MIMD (multiple instruction, multiple data) computers as well: their suitability for the SIMD model is rather an advantage than a restriction, as their scope of application is broader than the scope of application of many alternative algorithms, e.g., such ones which would run efficiently on MIMD machines only. Furthermore, on many systems, hyper-systolic algorithms can achieve a considerable reduction of cache to memory data transfer.

In the first part of the present work, the theory behind hyper-systolic computations has been worked out in detail. The appropriate formulation of hyper-systolic algorithms is found in the language of Additive Number Theory. The requirement for minimal interprocessor communication leads to a generalization of the so-called h-range problem, a famous generic problem in Additive Number Theory. In this framework, *shift bases* for hyper-systolic communication can be defined. In general, the determination of optimal bases is only possible for small systems. Optimal bases are produced by direct calculation, quasi-optimal bases can be found by simulated annealing methods or via simulated tempering. A sub-optimal solution is given by the so-called regular base which leads to particularly simple hyper-systolic algorithms. One has to distinguish between bases for 1-array and 2-array problems. The 1-array problem refers to the interaction of the elements of a given systolic array with each other, a prototype is the n-body computation, where all bodies are subject to mutual 2-body gravitative or Coulomb interaction. The 2-array problem is given by the interaction between two

different arrays, a prototype is the convolution. Here, each element of the first array interacts with all elements of the second array.

As is the case with systolic algorithms, the implementation of hyper-systolic algorithms on a parallel system implies a suitable mapping of the underlying systolic array (with n elements) onto the parallel computer (equipped with p processors). For 1-dimensional problems, hierarchy mapping complies very well with the hyper-systolic data flow. For 2-dimensional problems, as found in matrix computations, 1-dimensional projection block mapping or 1-dimensional projection staggered mapping can be chosen.

In the second part of this thesis, hyper-systolic algorithms have been worked out and adapted to four generic applications. These algorithms are implemented on a variety of massively parallel supercomputers where they are benchmarked in comparison to standard systolic computations.

The hyper-systolic algorithm for n-body computations with hierarchy mapping is implemented on a Connection Machine CM5, on a Cray T3D and on an APE100/Quadrics. On the CM5, its theoretical scaling behaviour compared to the conventional systolic method has been confirmed. On the Cray T3D, super-linear scaling has been found, which presumably is due to the reduced memory-to-cache data-transfer for the hyper-systolic method. On the APE100, a genuine quantum chromodynamics number cruncher, a performance of more than 60 % of the machine's peak performance could be achieved for this application!

As a further case study, the correlation problem has been formulated in a hyper-systolic manner. This application demonstrates that hyper-systolic computations are not alone useful for n^2-loop computations but can also be adapted to other 1-dimensional systolic calculations—using the same bases as for the generic n-body computation! It is interesting that, in the case of the correlation problem, the formulation on a ring of systolic cells can achieve 100 % absolute efficiency, whereas standard formulations only reach 50 % absolute efficiency.

Another field of interest is the emulation of routing. On machines with mesh or grid architectures that lack hardware routing facilities, routing can be emulated on the back of the nearest-neighbour connectivity. It is shown that the hyper-systolic algorithm can serve as a simple and effective emulator for a general router. As a benchmark, a hyper-systolic router for the permutation problem has been implemented on APE100/Quadrics.

A very important application field of the hyper-systolic algorithm is the distributed matrix multiplication. Packages like ScaLapack are mainly based on 2-dimensional data layouts. In contrast to these structures, the hyper-systolic matrix product relies on a 1-dimensional column-block data layout. As it exhibits the same complexity as Cannon's algorithm which is footed on a 2-dimensional layout, it might be an interesting alternative to 2-dimensional standard schemes. Moreover, the data layout chosen is ideal for matrix-vector computations, therefore, in applications where a combi-

nation of matrix-matrix and matrix-vector products is required, the hyper-systolic algorithm can be advantageous. The same structures as used for dense matrix-vector multiplications apply to fully coupled neural nets. Furthermore, maximum-likelihood image reconstruction from scatter data, as e.g. exploited in positron-emission tomography, can be computed similarly.

There are quite a few scientific fields where hyper-systolic computations could be of use. I hope that the present work can provide some seminal ideas that will help to spread the new parallel technique among the high-performance computing community.