ABSTRACT: A new scheme for partitioning systolic algorithms is presented. It is based on the time-sharing properties of the c-slow circuits. The technique is amenable to formalization and holds high potential for automatization.

INDEXING WORDS: systolic algorithms, partitioning, c-slow circuits, convolution

1. INTRODUCTION

Well balanced load, high efficiency, and minimal communication with the host are features which make the systolic algorithms especially attractive for parallel computer implementation. Since the explicit formulation of the systolic array notion [1-3], a great number of systolic algorithms have been proposed for different computational problems (the reader is referred to [4] for extensive introduction to systolic algorithms and arrays and for many examples). One of the major problems in implementing systolic algorithms is that they usually require a processor array whose size depends on the size of the problem to be solved. Different partitioning schemes have been proposed to solve this problem [5,6]. They are, however, either applicable only to algorithms with unidirectional data flow and the resulting structure depends radically on the ratio between problem size and array size [5], or the criteria used for partitioning have heuristic nature and can be effectively applied only on relatively simple problems [6].

A new partitioning technique is presented in this paper. Our approach is based on the theory of the so called c-slow circuits and gives a discipline for the partitioning of systolic algorithms. Due to the high degree of formalization our approach is very suitable for automatization.

* This paper has been first submitted to the Conference on Vector and Parallel Processors in Computational Science III, Liverpool, August 1987, and subsequently accepted for presentation. The author could, however, not attend the conference and present the paper, so that it was not included in the proceedings and could thus not be widely disseminated. Only slight corrections have been made in the original text and a comment has been added about a similar partial result this work shares with another paper which appeared elsewhere meanwhile.
tization. The paper is organized as follows. Elements of the theory of c-slow circuits are presented in section 2. In section 3, the technique is illustrated on systolic arrays for convolution. Some characteristics of the technique are summarized in section 4.

2. USING C-SLOW CIRCUITS FOR PARTITIONING

The partitioning technique proposed here is a sort of a time-sharing method for the use of logic circuits. It will be illustrated on automata consisting of combinatorial circuits and registers (delay elements). The automata abstraction should, however, not be considered as a proposal for implementation, but rather as a model of an algorithm. This art of algorithm representation is widely used in the literature and it is especially useful and comprehensive in the representation of parallel systolic algorithms [4]. It might be more illuminating to illustrate the technique on a simple example first instead of giving a general theory. Fig.1a shows an accumulator which consists of an adder and an accumulation register (shown by a small black bar). The circuit is capable of accumulating and outputting the running sum \( y_i = \sum_{k=-2}^{\infty} x_k \) of an input digital signal \( x_k, k = -2, -1, 0, 1, 2, \ldots \). Now, we change the circuit in Fig.1a by replacing the register by two (in general by c) chained registers, Fig.1b. The new circuit can realize the same function as the original one, if the time periods measured in number of cycles of the clock which controls the registers is increased twice (in general c times), Fig.1b. Since the new circuit is two (generally c) times slower than the original one, it is called after [7] a 2-slow (generally c-slow) version of the original circuit. A 2-slow (in general c-slow) circuit can be used for the concurrent execution of two (generally of c) independent computational problems. Fig.1c gives an illustration: the operations \( y_i = \sum_{k=-2}^{\infty} x_k \) and \( y'_i = \sum_{k=-2}^{\infty} x_k \) are executed during the clock cycles for which holds \( t \mod 2 = 0 \) and \( t \mod 2 = 1 \), respectively.

Fig.2 shows how the method described can be applied on array structures. The array shown in Fig.2 consists of three independent accumulators. These three array parts are active on three independent accumulation processes: \( y_i = \sum_{k=-2}^{\infty} x_k \), \( y'_i = \sum_{k=-2}^{\infty} x_k \), and \( y''_i = \sum_{k=-2}^{\infty} x_k \), respectively. A 3-slow version of one accumulator can execute all three accumulation processes, Fig.2b. It is active on the first, second and third accumulation process in the clock cycles for which holds \( t \mod 3 = 0 \), \( t \mod 3 = 1 \), and \( t \mod 3 = 2 \), respectively. In this way, a c-slow version of an appropriate (1/c)-th part of an array can execute the task of the whole array. (In doing this, it is, however, c times slower.)

The method can be generalized for the case in which the parts of the