PARALLEL MICROPROGRAMMING AS A TOOL FOR MULTI-MICROPROCESSOR SYSTEMS

O.L. Bandman, S.V. Piskunov

1. INTRODUCTION

Since microprocessors are cheap and easy to reach, the idea is exciting of introducing them into high-performance systems designed to solve complicated computation tasks. Here belong signal processing, image processing, process simulation in media, etc. These tasks usually incorporate subtasks related to computing on data arrays such as matrices, vectors, sequences, etc. As the subtasks have many similar operations and constitute a good proportion of computation, it is reasonable to consider them as independent operators to be implemented in homogeneous structures of the cell type. Parallel-sequential or parallel-pipeline composition of these operators matches best the natural parallelism of a task.

When designing special-purpose processors from serial microprocessors, much labour is required to design architecture. The purpose of architecture design is to produce the structural scheme of a computing system, i.e. a set of processor elements with microprogram sets assigned to them and connections between them.

A range of tools for architecture design are referred to as the architecture compiler. The basis for the architecture compiler is provided by the ways in which problem solution algorithms are represented in the parallel form. They should meet the following requirements:

a) make the system architecture as similar as possible to computation processes and
b) make automation and the use of the experience gained (architecture libraries) possible.

Here we suggest a new approach to systematic design of multi-microprocessor systems. Its underlying postulates are as follows.

1) Computing and control algorithms are represented in the form of parallel microprograms [1]. Parallel microprogram notation as well as the simulation language and structural synthesis techniques are based on the fundamental algorithmic system - the parallel substitution algorithm [2]. This algorithmic system is, on the one hand, a parallel version of the Markov algorithm and, on the other hand, the cellular automaton generalization.

2) Asynchronous composition of algorithmic descriptions of operators over arrays and parallel microprograms is based on the model called the Petri net [3]. The well-developed tool for Petri net analysis allows an automated test for parallel algorithm correctness, a formalized synthesis of control over interactions between microprocessors and assessment of hardware performance.

1) Computing Center, Siberian Division, the USSR Academy of Sciences, Novosibirsk, 630090, USSR.
3) The design is carried out in the interactive mode with the use of the extendable library of architecture solutions already made to implement basic algorithms. Interactivity implies that all important decisions such as the choice of an original architecture hypothesis, synchronization techniques, composition and control are the designer's responsibility, mapping them into parallel microprograms texts. All other manipulations with microprogram description such as validity test and structural scheme plotting are computer-aided.

The process of architecture design has three stages. The first one, which is preliminary and non-formalized is to determine basic operators of array processing and make a general algorithmic schedule. The second stage includes the construction of parallel microprogram operators and the control microprogram and combining them into a complete microprogram description of a special-purpose processor. The third stage incorporates formal computer-aided procedures converting parallel microprograms into structural schemes, mapping of these schemes into microprocessor net and their microprogramming.

The paper has three main sections. Section 1 dwells on theoretical substantiation of the approach in question. It introduces fundamental concepts of parallel programming and offers the techniques for compiling parallel microprograms. In Section 2, we describe the composition of operators and the synthesis of the control microprogram based on Petri net simulating asynchronous interactions among operators. In Section 3, we present a formal way of transfer from microprogram description to a structural scheme.

2. PARALLEL MICROPROGRAM

Methods of the structure synthesis of a special-purpose processor rely on microprogram representation of parallel algorithms implemented by it.

The major distinction between a parallel microprogram and an ordinary (sequential) one is that the order of microinstructions in a parallel microprogram is not defined. Instead, each microinstruction includes the condition of its execution in the form of data values list with the indication of their coordinates in the array processed. At every step, all the microinstructions are executed either simultaneously or in an arbitrary order for which execution conditions are satisfied.

The parallel microprogram class is a subclass of parallel substitution algorithms [2]. Formally, parallel substitution algorithms are defined as follows.

Let $A$ be a finite alphabet, $M$ - a set of names whose cardinality is but enumerable. The pair $(a,m)$ belonging to $A \times M$ is called a cell, $a$ is the state of a cell, and $m$ is the name of a cell. In a parallel substitution algorithm, the object for conversion is the cellular array $W$, a finite set of cells having not a single pair of cells with the same