Proc. Int. Conf. on Par. and Distr. Comp. and Netw. (PDCN'97), p.144-150, 1997.

PARAMETERIZED PARTITION VALUATION FOR PARALLEL LOGIC SIMULATION

KLAUS HERING, REINER HAUPT and UDO PETRI
Institute of Computer Science
University of Leipzig
Augustusplatz 10-11, 04109 Leipzig
Germany

ABSTRACT

Parallelization of logic simulation on register-transfer and gate level is a promising way to accelerate extremely time-extensive system simulation processes during the design of whole processor structures. The background of this paper is given by the functional simulator *parallelTEXSIM* realizing simulation based on the *clock-cycle algorithm* over loosely-coupled parallel processor systems. In preparation for parallel cycle simulation, partitioning of hardware models is necessary, which essentially determines the efficiency of the following simulation.
We introduce a new method of *parameterized partition valuation* for use within model partitioning algorithms. It is based on a formal definition of parallel cycle simulation involving a model of parallel computation called *Communicating Processors*. Parameters within the valuation function permit consideration of specific properties related to both the simulation target architecture and the hardware design to be simulated. Our partition valuation method allows *performance estimation* with respect to corresponding parallel simulation. This has been confirmed by tests concerning several models of real processors as, for instance, the *PowerPC 604* with parallel simulation running on an *IBM SP2*.

KEYWORDS:
Parallel logic simulation; Model partitioning algorithms; Performance estimation; *IBM SP2*

1 INTRODUCTION

Due to challenging technological capabilities the attainable complexity of *VLSI* designs is growing rapidly. Therefore, the employment of verification tools in all design phases is unavoidable. Simulation is a very important *VLSI* design verification method. The background of our work is given by functional simulation on register-transfer and gate level (logic simulation) without consideration of timing aspects. In [1] a simulation strategy is presented with underlying hardware models embodying complete processor structures and simulation stimuli representing microprogrammes or machine instruction sequences. During system simulation, time-extensive simulation runs for final validation of complex designs are considered. Aiming at significant run-time reductions for such processes we have parallelized the sequential functional simulator *TEXSIM*[1] which operates on the basis of the *clock-cycle algorithm*. The simulator *parallelTEXSIM* is documented in [2]. We chose a parallelization approach making use of model inherent parallelism. Within a corresponding parallel cycle simulation, several simulator instances co-operate over a loosely-coupled processor system, each instance simulating a part of a synchronous hardware design. Therefore, in preparation for parallel simulation, partitioning of the whole hardware model is necessary which essentially determines the run-time behaviour of a following simulation.

Starting from ideas of D.Zike and W.Roesner presented in [3] we have developed a hierarchical partitioning strategy outlined in [4] based on *fan-in cones* as elementary building blocks for partitions. Related work to cone-based partitioning can be found in [5] and [6]. Within the taxonomy of partitioning techniques given in [7] our strategy embodies a *bottom-up clustering* approach. Partitioning algorithms regarded at the end of this paper follow a special two-stage strategy, where *Evolutionary Algorithms* are applied after a fast pre-partitioning phase which reduces the problem complexity.

In the context of the formulation of partitioning problems as combinational optimization problems, partitions are related to quantities via cost functions (*partition valuation*). With respect to parallel simulation as our special subject, assigned values more or less directly express a connection to parallel simulation run-time. In this paper, we introduce a parameterized partition valuation function on the basis of a formal definition of parallel cycle simulation realized

[1] developed by *IBM*

by *parallelTEXSIM*. The corresponding framework of formal concepts is fully represented in [8]. Within our valuation method aspects of *workload* and *interprocessor communication* are combined. Parameters derived from pre-simulation and machine benchmarking allow specific properties of both the hardware design and the simulation target architecture to flow into partition valuation. The *time-driven* nature of the underlying simulation allows direct deduction of *run-time estimations* for sequences of cycles from estimations concerning one cycle.

Our paper is organized as follows. In Section 2 the simulator *parallelTEXSIM* is briefly introduced. Section 3 provides the definition of a *Structural Hardware Model (SHM)*. Furthermore, model partitions are introduced as families of cone sets and partition-based *Parallel Structural Hardware Models (PSHMs)* are constructed as sets of *SHMs*. In Section 4 *PSHMs* are combined with a model of parallel computation called *Communicating Processors (CP)* to define *Parallel Cycle Simulation (PCS)* as special behaviour of *CP*. Based on the preceding concepts a parameterized partition valuation function is developed in Section 5. Experimental results related to model partitioning and corresponding parallel simulation on an *IBM SP2* system for real processor designs are given in Section 6. Finally, Section 7 summarizes the work presented and outlines future objectives.

2 THE PARALLEL SIMULATOR

The simulator *parallelTEXSIM* was implemented by D.DÖHLER under the *AIX Parallel Environment*[2] starting from the sequential simulator *TEXSIM* which was developed by D.ZIKE (*IBM*). *TEXSIM* performs logic simulation on the register-transfer and gate level in zero-delay mode using the *clock-cycle algorithm*. The simulation of one clock-cycle mainly consists of two parts: evaluation of (rank-ordered) combinational logic and updating of storing elements (latches) which represent cycle boundaries. During the simulation of one cycle with *parallelTEXSIM* several simulator instances *sTEXSIM* (sequential *TEXSIM* enriched by a communication shell) co-operate over a loosely-coupled processor system, each simulating a part (model block) of the whole design under consideration. At cycle boundaries collective communication takes place. All interprocessor communication is realized making use of the *Message Passing Library (MPL)* of the *AIX Parallel Environment*. The parallel simulator is both running on *IBM SP2* systems with processors coupled via a *High-Performance Switch* and on *IBM RS/6000* workstation clusters. Besides the *sTEXSIM* components one master component *mTEXSIM* is responsible for the realization of the *parallelTEXSIM Application Programming Interface (API)* providing the possibility of simulation control by dynamically linked clients. Furthermore, *mTEXSIM* co-ordinates the *sTEXSIM* components within parallel simulation.

[2] product of IBM

3 HARDWARE MODELS

The graph models introduced in the following provide a formal basis for the development, investigation and implementation of model partitioning algorithms in the context of parallel *cycle-based* simulation. They are not restricted to the special simulator *TEXSIM*.

3.1 THE BASIC MODEL

Essential components of our basic structural hardware model are given by a family of pairwise disjoint sets which comprises the set M_E of logical boxes (logical gates, multiplexers, ...), the set M_I of input boxes, the set M_O of output boxes, the set M_L of storing boxes (latches) and the set M_S of nets representing wires. With $M_B = M_E \cup M_I \cup M_O \cup M_L$ and $M_B, M_S \neq \emptyset$, a directed bipartite graph

$$M = (M_B, M_S, M_{\mathcal{R}})$$

is called **Structural Hardware Model (SHM)**, if M_I is the set of all sources of M, M_O is the set of all sinks of M and any directed cycle in M includes at least one element of M_L.

$M_{\mathcal{R}} \subseteq (M_B \times M_S) \cup (M_S \times M_B)$ describes the connection of elements (boxes) of M_B with nets of M_S. There are no directed cycles within *SHMs* covering boxes belonging to $M_E \cup M_S$ exclusively. With respect to underlying hardware designs this reflects the absence of asynchronous feedbacks in combinational logic. Figure 1 roughly illustrates a *SHM*. Thick arrows represent sub-sets of M_S.

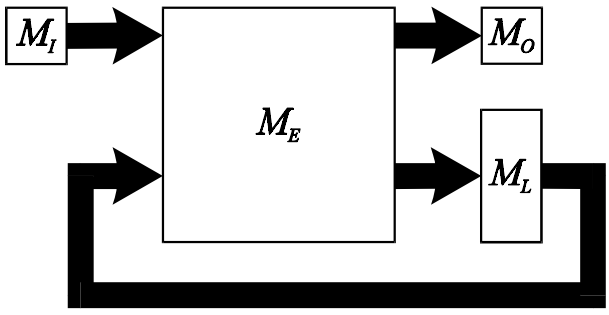

FIGURE 1: STRUCTURAL HARDWARE MODEL

For a formal description of (sequential) clock-cycle simulation over a *SHM* M we regard the boxes of M_B as elemental carriers of simulation activities. To identify these activities we introduce a set of abstract actions

$$\mathcal{A} = \mathcal{A}_I \cup \mathcal{A}_E \cup \mathcal{A}_O \cup \mathcal{A}_L$$

with a bijective assignment function $a : M_B \rightarrow \mathcal{A}$ assuming $a(M_\omega) = \mathcal{A}_\omega$ for $\omega \in \{E, I, O, L\}$. With $\|$ denoting sequence concatenation and $\mathcal{A}^+$ representing the set of all finite non-empty sequences over $\mathcal{A}$, special sequences

$$s_{seq} \in {\mathcal{A}_I}^+ \| {\mathcal{A}_E}^+ \| {\mathcal{A}_O}^+ \| {\mathcal{A}_L}^+$$

are introduced as **Sequential Cycle Simulation (SCS)** with respect to M. In particular, each action of $\mathcal{A}$ appears exactly once within a *SCS* and the order of actions belonging to $\mathcal{A}_E$ is restricted by a levelizing of M_E which is determined by $M_\mathcal{R}$. Remark, that our consideration is restricted to the simulation of one clock-cycle. In the context of this paper actions are not supplied with semantic details as, for instance, state transferring functions. According to partition valuation presented in Section 5 they are considered as sources of simulation expense.

3.2 CONE-BASED PARTITIONS

Within our model partitioning approach for parallel cycle simulation we take *fan-in cones* as basic building blocks for partitions. This choice allows building of sub-models for parallel cycle simulation over which *sTEXSIM* instances in their simulation kernel can work in the same way as *TEXSIM* instances.

Let M be a *SHM* and $x \in M_E \cup M_L \cup M_O$. The **fan-in cone co(x)** (with respect to M) is defined as sub-set of M_B containing x itself and all logical boxes of M_E for which a directed path to x exists which does not cover elements of $M_I \cup M_L \cup M_O$. For partitioning of M special *fan-in cones* (shortly called *cones*) given by

$$Co(M) = \{co(x) \mid x \in M_L \cup M_O\}$$

are taken into consideration. Then, a partition Π of $Co(M)$ in mathematical sense is called a **partition** of M.

Different elements (cones) of $Co(M)$ may have common boxes (*cone overlapping*). If we assume, that a partition component determines the model part to be handled by a simulator instance on a single processor during parallel cycle simulation, then overlapping cones as elements of different partition components stand for replication of simulation work. Besides this drawback the cone-based partitioning approach bears the advantage that interprocessor communication during parallel cycle simulation is concentrated at cycle boundaries.

3.3 THE PARALLEL STRUCTURAL HARDWARE MODEL

Partitions of a *SHM* M are now translated into sets of "sub-models" of M. Thereby, for each partition component (representing a cone set) a corresponding *SHM* is built.

Let us consider a *SHM* M, a partition Π of M and a cone set $\mathcal{C} \in \Pi$. The construction of **sub-models** of M with respect to Π as triplets

$$M^\mathcal{C}_\Pi = \left(M^\mathcal{C}_B, M^\mathcal{C}_S, M^\mathcal{C}_\mathcal{R}\right)$$

with $M^\mathcal{C}_B = M^\mathcal{C}_E \cup M^\mathcal{C}_I \cup M^\mathcal{C}_O \cup M^\mathcal{C}_L$ is described in detail in [8]. For a short consideration of the component determination we assume $B^\mathcal{C} = \bigcup_{c \in \mathcal{C}} c$. Then $M^\mathcal{C}_E$ and $M^\mathcal{C}_L$ are defined as follows:

$$M^\mathcal{C}_E = B^\mathcal{C} \cap M_E,\ M^\mathcal{C}_L = B^\mathcal{C} \cap M_L. \qquad (3.1)$$

$M^\mathcal{C}_{O,O} = B^\mathcal{C} \cap M_O$ yields the set of global output boxes of $M^\mathcal{C}_\Pi$ $\left(M^\mathcal{C}_{O,O} \subseteq M^\mathcal{C}_O\right)$. All elements of M_I "feeding" $\mathcal{C}$ (elements from which a path of length 2 exists leading to an element of $B^\mathcal{C}$) form the set $M^\mathcal{C}_{I,I}$ of global input boxes of $M^\mathcal{C}_\Pi$ $\left(M^\mathcal{C}_{I,I} \subseteq M^\mathcal{C}_I\right)$. In the context of communication processes at cycle boundaries between $M^\mathcal{C}_\Pi$ and other sub-models related to Π "artificial" boxes (not present within M_B) are introduced. Thereby, input and output boxes appear as ordered pairs $(\mathcal{C}^*, s)$ and $(s, \mathcal{C}^{**})$, respectively ($s \in M_S$; $\mathcal{C}^*, \mathcal{C}^{**} \in \Pi$; $\mathcal{C}^*, \mathcal{C}^{**} \neq \mathcal{C}$). $M^{\mathcal{C}^*}_\Pi$ is to be interpreted as communication source and $M^{\mathcal{C}^{**}}_\Pi$ represents a communication target (with respect to $M^\mathcal{C}_\Pi$). s embodies a physical connection in the context of M "leaving" $\mathcal{C}^*$ and "leading" to $\mathcal{C}$ or "leaving" $\mathcal{C}$ and "leading" to $\mathcal{C}^{**}$. With $M^\mathcal{C}_{I,L}$ built from all input boxes $(\mathcal{C}^*, s)$ and $M^\mathcal{C}_{O,L}$ built from all output boxes $(s, \mathcal{C}^{**})$, we set

$$M^\mathcal{C}_I = M^\mathcal{C}_{I,I} \cup M^\mathcal{C}_{I,L},\ M^\mathcal{C}_O = M^\mathcal{C}_{O,O} \cup M^\mathcal{C}_{O,L}. \qquad (3.2)$$

$M^\mathcal{C}_S$ and $M^\mathcal{C}_\mathcal{R}$ are constructed in a straight forward manner (see [8]). The resulting sub-model $M^\mathcal{C}_\Pi$ is proved to be a *SHM*. The set

$$M^\Pi = \left\{M^\mathcal{C}_\Pi \mid \mathcal{C} \in \Pi\right\} \qquad (3.3)$$

is called **Parallel Structural Hardware Model (PSHM)** with respect to Π. In Figure 2 a sub-model $M^\mathcal{C}_\Pi$ in the context of a *PSHM* is represented schematically.

As for *SHMs* not standing in the context of a *PSHM*, the behaviour of *PSHM* components is defined as action sequence again. For a sub-model $M^\mathcal{C}_\Pi$ of M with respect to Π we consider an action set

$$\mathcal{A}^\mathcal{C} = \mathcal{A}^\mathcal{C}_E \cup \mathcal{A}^\mathcal{C}_{I,I} \cup \mathcal{A}^\mathcal{C}_{I,L} \cup \mathcal{A}^\mathcal{C}_{O,O} \cup \mathcal{A}^\mathcal{C}_{O,L} \cup \mathcal{A}^\mathcal{C}_L \cup \{c\} \quad (3.4)$$

with a bijective assignment function $a : M^\mathcal{C}_B \rightarrow \mathcal{A}^\mathcal{C} \setminus \{c\}$ assuming $a\left(M^\mathcal{C}_\omega\right) = \mathcal{A}^\mathcal{C}_\omega$ (ω representing an arbitrary variant of the lower indices appearing in (3.4)). Different from *SCS*, a special action c not bound to a special

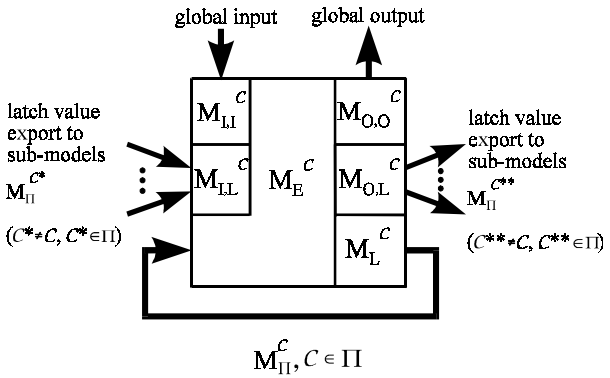

FIGURE 2: PARALLEL STRUCTURAL HARDWARE MODEL

box is involved representing component communication at cycle boundaries. $\mathcal{A}^{\mathcal{C}}$ reflects the splitting of the sets of input and output boxes within $M_{\Pi}^{\mathcal{C}}$. A sequence $s_{seq}^{\mathcal{C}} \in \left(\mathcal{A}^{\mathcal{C}}\right)^{+}$ is called **Extended Sequential Cycle Simulation (ESCS)** with respect to $M_{\Pi}^{\mathcal{C}}$ if each action of $\mathcal{A}^{\mathcal{C}}$ appears exactly once within $s_{seq}^{\mathcal{C}}$, actions of $\mathcal{A}_{E}^{\mathcal{C}}$ are ordered with respect to a levelizing of $M_{E}^{\mathcal{C}}$ (determined by $M_{\mathcal{R}}^{\mathcal{C}}$) and $s_{seq}^{\mathcal{C}}$ has the following structure:

$$s_{seq}^{\mathcal{C}} = s_{cycle}^{\mathcal{C}} \| s_{comm}^{\mathcal{C}}, \quad (3.5)$$

$$s_{cycle}^{\mathcal{C}} \in {\mathcal{A}_{I,I}^{\mathcal{C}}}^{+} \| {\mathcal{A}_{E}^{\mathcal{C}}}^{+} \| {\mathcal{A}_{O,O}^{\mathcal{C}}}^{+} \| {\mathcal{A}_{L}^{\mathcal{C}}}^{+}, \quad (3.6)$$

$$s_{comm}^{\mathcal{C}} = s_{pre}^{\mathcal{C}} \| (c) \| s_{post}^{\mathcal{C}}, \quad (3.7)$$

$$s_{pre}^{\mathcal{C}} \in {\mathcal{A}_{O,L}^{\mathcal{C}}}^{+}, s_{post}^{\mathcal{C}} \in {\mathcal{A}_{I,L}^{\mathcal{C}}}^{+}.$$

(3.5) reflects the restriction of communication between components involved in parallel cycle simulation to cycle boundaries. $s_{cycle}^{\mathcal{C}}$ (3.6) appears as *SCS* with respect to $M_{\Pi}^{\mathcal{C}}$ modified by omitting the actions from $\mathcal{A}_{I,L}^{\mathcal{C}} \cup \mathcal{A}_{O,L}^{\mathcal{C}}$. $s_{comm}^{\mathcal{C}}$ (3.7) represents three phases of communication related work. $s_{pre}^{\mathcal{C}}$, the first one, is devoted to the preparation of interprocessor communication under sending aspect with respect to $M_{\Pi}^{\mathcal{C}}$ (extraction of sub-model data with following placement in communication related structures). Then, c stands for a collective communication action at cycle boundaries. Finally, $s_{post}^{\mathcal{C}}$ represents post-processing of interprocessor comunication under receiving aspect with respect to $M_{\Pi}^{\mathcal{C}}$ (extraction of data from communication related structures with following placement in submodel structures).

4 THE MODEL OF PARALLEL COMPUTATION

For combining the behaviour of *PSHM* components to a behaviour of the whole *PSHM* we make use of a model of parallel computation introduced in [8].

4.1 COMMUNICATING PROCESSORS

Strongly inspired by the *LogP* model described in [9], we have developed *Communicating Processors* (*CP*) as model of parallel computation related to message passing architectures. *CP* allows the consideration of architecture dependent properties via parameters. Different communication mechanisms can be integrated into the model corresponding to topical needs. The behavioural capabilities of single processes are described in terms of sequences of abstract actions to have the possibility of relating them to several interpretations (for instance, to simulation time amount as basis for partition valuation).

The *Communicating Processors (CP)* model is defined as triplet $\mathcal{P} = (P_P, P_A, P_C)$ where

- $P_P = \{P_1, \ldots, P_n\}$ is a set of (abstract) processors working asynchronously,
- $P_A = \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ is a family of finite processor-bound action sets and
- $P_C = \{\mathcal{M}_1, \ldots, \mathcal{M}_l\}$ is a finite set of communication mechanisms. A communication mechanism is given as an ordered pair with a *qualitative characteristic* as first component and a (possibly empty) set of *quantitative characteristics* as second component. A qualitative characteristic comprises
  - the determination of actions related to the corresponding mechanism
  - the determination of source/target relations within a set of involved processors
  - the determination of synchronization conditions

  A quantitative characteristic appears as a real function or constant, valuating a communication-related aspect.

In the context of *CP*, actions represent the execution of operations on the processors under consideration. There is nothing said about their complexity. The execution of an extensive high-level procedure can be regarded as well as handling a microcode instruction.

Quantitative characteristics of communication mechanisms are introduced within P_C for allowing communication properties of real parallel architectures to flow into the *CP* model. For instance, time boundaries of special communication-related events (see latency, gap, overhead within *LogP*) could be such characteristics. Another example is given by functions

yielding run-time estimations for communication processes in dependence of the number of involved processors, message lengths, network load situation and similar arguments.

Within P_C elementary *point-to-point* communication mechanisms built on *send-* and *receive-* actions can be considered as well as *collective* communication mechanisms with (usually) more than two actions (on different processors) involved. *CP* behaviour is determined by given (sequential) component behaviour and communication mechanisms integrated into the model. Concrete behaviour appears as sequence of action sets which are interpreted as maximum sets of simultaneous active actions on different processors. In [8], *Unrestricted Parallel Behaviour (UPB)* is defined as general framework which is restricted to concrete *CP* behaviour by inclusion of synchronization conditions according to the communication mechanisms considered.

4.2 PARALLEL CYCLE SIMULATION

In the following we outline the definition of *Parallel Cycle Simulation* with respect to a *PSHM* $M^{\Pi} = \{M_1, \ldots, M_n\}$ as introduced in (3.3). Connected to this, a *CP* model

$$\mathcal{P} = (P_p, P_A, P_C) \tag{4.1}$$

is constructed with $P_P = \{P_1, \ldots, P_n\}$, each M_i corresponding to P_i. As processor-bound action sets $\mathcal{A}_i$ those action sets are chosen, which belong to M_i according to (3.4). Furthermore, we integrate exactly one communication mechanism $\mathcal{M}$ into $\mathcal{P}$ ($P_C = \{\mathcal{M}\}$). $\mathcal{M}$ does not depend on the concrete *PSHM* under consideration. It is related to the *mpc_index*-command belonging to the *Message Passing Library* of the *AIX Parallel Environment*. This command is used for the implementation of interprocessor communication at cycle boundaries during simulation with *parallelTEXSIM*. The qualitative characteristic of $\mathcal{M}$ in the framework of $\mathcal{P}$ is as follows:

- The only action engaged in $\mathcal{M}$ is c which is an element of every action set $\mathcal{A}_i$.
- The whole processor set P_P is involved in $\mathcal{M}$. Each processor sends to each of the remaining processors individual messages (*all-to-all* personalized communication).
- $\mathcal{M}$ is a collective communication for which n actions c (one at each processor) have to synchronize.

In addition, there is one quantitative characteristic of $\mathcal{M}$. It is given by a function estimating the time needed for a corresponding collective communication in dependence of the number of processors involved and the message length under supposition of a *SP2* configuration with nodes connected via a *High-Performance Switch*.

With $\mathcal{B}_i$ denoting the set of all *Extended Sequential Cycle Simulations* belonging to M_i $\left(\mathcal{B}_i \subseteq {\mathcal{A}_i}^+\right)$ we set

$$\mathcal{B} = \{\mathcal{B}_1, \ldots, \mathcal{B}_n\}\,. \tag{4.2}$$

Then, every *unrestricted parallel behaviour sequence* $\overline{s}$ of $\mathcal{P}$ with respect to $\mathcal{B}$ which contains an action set consisting of n actions corresponding to the communication action c is called **Parallel Cycle Simulation (PCS)**. In Figure 3, a *PCS* related to a three-component *PSHM* with s^i_ω denoting sequences of ${\mathcal{A}^i_\omega}^+$ and i identifying *PSHM* components is depicted schematically. A sub-sequence of *SCS* structure is shaded grey. Areas represented in black are related to pre- or post-communication sub-sequences.

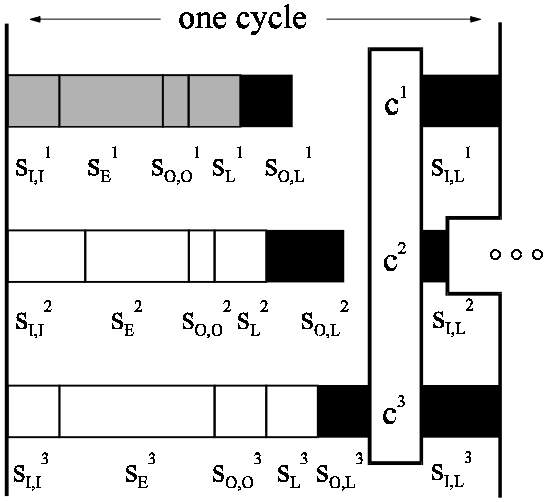

FIGURE 3: PARALLEL CYCLE SIMULATION

5 PARTITION VALUATION

In the following, partitions are related to run-time estimations for *PCS* realized by *parallelTEXSIM*. Actions are regarded as basic elements consuming simulation time.

Let us consider an arbitrarily chosen partition Π of a *SHM* M. This implies a *PSHM* M^{Π} (3.3) together with a family $\mathcal{B}$ (4.2) of sets of component behaviour sequences (*ESCSs*) assuming a fixed rule of assigning actions to boxes. Furthermore, a *CP* model $\mathcal{P}$ (4.1) can be constructed which in dependence of $\mathcal{B}$ delivers a set of *PCSs* with respect to M^{Π} as its behaviour. From each *PCS* $\overline{s}$ for an arbitrary $\mathcal{C} \in \Pi$ a "local" behaviour sequence $s^{\mathcal{C}}_{seq}$ can be deduced with

$$s^{\mathcal{C}}_{seq} \quad = s^{\mathcal{C}}_{cycle} \| s^{\mathcal{C}}_{pre} \| (c) \| s^{\mathcal{C}}_{post} \tag{5.1}$$

according to (3.5) and (3.7). We estimate component simulation time $t^{\mathcal{C}}_{seq}$ on the basis of assigning execution time values to corresponding sub-sequences omitting possible idle intervals between their execution:

$$t^{\mathcal{C}}_{seq} = t^{\mathcal{C}}_{cycle} + t^{\mathcal{C}}_{pre} + t_c + t^{\mathcal{C}}_{post}.$$

Due to the synchronization effect of c we obtain an estimation of the *PCS* execution time $t^{\overline{s}}_{par}$ as follows:

$$t^{\overline{s}}_{par} = \max_{\mathcal{C}\in\Pi} \left(t^{\mathcal{C}}_{cycle} + t^{\mathcal{C}}_{pre} + t^{\mathcal{C}}_{post}\right) + t_c. \qquad (5.2)$$

$t^{\mathcal{C}}_{cycle}$ is supposed to be determined by the evaluation of logical boxes and latch updates, global input and output actions are neglected. We set

$$t^{\mathcal{C}}_{cycle} = t^{M}_{B} \left| M^{\mathcal{C}}_{E} \cup M^{\mathcal{C}}_{L} \right|, \qquad (5.3)$$

with $M^{\mathcal{C}}_{E}$ and $M^{\mathcal{C}}_{L}$ defined as in (3.1) and t^{M}_{B} representing an average execution time for actions of $\mathcal{A}^{\mathcal{C}}_{E} \cup \mathcal{A}^{\mathcal{C}}_{L}$. The parameter t^{M}_{B} is obtained from (sequential) pre-simulation of the model M.

Pre- and post-processing of interprocessor communication is related to actions of $\mathcal{A}^{\mathcal{C}}_{O,L}$ and $\mathcal{A}^{\mathcal{C}}_{I,L}$. The amount of time for each such action is supposed to be the same, expressed by the parameter t^{M}_{comm} which is inquired by (parallel) pre-simulation according to a reference partition of M. With $M^{\mathcal{C}}_{O,L}$ and $M^{\mathcal{C}}_{I,L}$ as described in the context of (3.2), we get

$$t^{\mathcal{C}}_{pre} + t^{\mathcal{C}}_{post} = t^{M}_{comm} \left| M^{\mathcal{C}}_{O,L} \cup M^{\mathcal{C}}_{I,L} \right| . \qquad (5.4)$$

Finally, we have to consider t_c, which is estimated by a function given as quantitative characteristic of the collective communication mechanism $\mathcal{M}$ integrated into $\mathcal{P}$. We suppose

$$t_c = m\,(t_a + t_b n) \qquad (5.5)$$

with t_a,t_b being architecture-dependent parameters, m denoting the maximum number of values which have to be transferred between any pair of processors and $n = |\Pi|$.

Taking together (5.3), (5.4) and (5.5) we have determined our estimation of $t^{\overline{s}}_{par}$ under (5.2) completely. The result is the same for all possible *PCS* sequences related to M^{Π}, because corresponding component behaviour sequences only differ in the order of actions within sub-sequences occuring in (5.1). This is without relevance for the particular estimations given above. Due to the *time-driven* nature of simulation considered we can immediately derive run-time estimations for complex simulation processes covering a sequence of cycles from $t^{\overline{s}}_{par}$. With such run-time predictions we are able to avoid expensive sub-model building and following simulation runs for partitions resulting in bad simulation performance. The problem of early performance prediction in general is considered, for instance, in [10].

6 EXPERIMENTAL RESULTS

We have integrated the above partition valuation method into our two-level partitioning scheme outlined in [4]. At the first level fast pre-partitioning algorithms are applied to reduce problem complexity by concentrating cones into super-cones. Here we consider the algorithm of MUELLER-THUNS et al. (*MT*) described in [6] and our *STEP* algorithm. Both algorithms yield partitions which are balanced with respect to the number of cones within the partition components. There is no explicit partition valuation at the first level of our scheme. At the second level, *Evolutionary Algorithms* (*EAs*) are applied starting with an initial population of individuals representing encoded partitions. These initial partitions are constructed applying the *MOCC* algorithm sketched in [4]. Within the *EAs*, partition valuation appears as realization of the *fitness function*.

In the following we present results of simulating models of real processor structures with *parallel-TEXSIM* on an *IBM SP2* system. Measured run-time is compared with estimated run-time (5.2) resulting from partition valuation (related to one clock-cycle in both cases). We consider essential parts of two processor designs: *IBM S/390 G1* and *PowerPC 604*.

The S/390 G1 model contains 181 418 boxes yielding 22 034 cones. Our prediction for the sequential simulation run-time of one clock-cycle amounts to 14.513 *ms* in comparison to a measured value of 15.393 *ms*. The following two tables give the predicted and measured (by simulation) parallel run-times and the corresponding speedups for two different pre-partitioning procedures and three different block numbers:

pre-partitioning algorithm	MT to 500 super cones		STEP to 500 super cones	
	predicted run-time	measured run-time	predicted run-time	measured run-time
2 processors	10.02	10.60	9.01	9.99
3 processors	9.81	10.48	7.42	8.12
4 processors	8.26	8.89	6.22	6.92

pre-partitioning algorithm	MT to 500 super cones		STEP to 500 super cones	
	predicted speedup	measured speedup	predicted speedup	measured speedup
2 processors	1.45	1.44	1.61	1.54
3 processors	1.48	1.47	1.95	1.90
4 processors	1.76	1.73	2.33	2.22

The PowerPC 604 model contains 319 543 boxes yielding 42 176 cones. The predicted sequential run-time related to one clock-cycle amounts to 38.345 *ms* in comparison to a measured value of 36.382 *ms*. The

following two tables contain experimental results organized as in the case above.

pre-partitioning algorithm	MT to 500 super cones		STEP to 500 super cones	
	predicted run-time	measured run-time	predicted run-time	measured run-time
2 processors	25.24	24.47	22.25	21.20
3 processors	21.21	21.12	17.75	17.41
4 processors	19.37	19.06	14.38	13.82

pre-partitioning algorithm	MT to 500 super cones		STEP to 500 super cones	
	predicted speedup	measured speedup	predicted speedup	measured speedup
2 processors	1.52	1.49	1.72	1.72
3 processors	1.81	1.72	2.16	2.09
4 processors	1.98	1.91	2.67	2.63

The coincidence of predicted and measured run-times for both processor models are encouraging for applying this method of run-time estimation (5.2) to the valuation of partitions of actual processor models with essentially more boxes.

7 CONCLUDING REMARKS

We have presented a new method of parameterized partition valuation for application in model partitioning in the context of parallel logic simulation based on the *clock-cycle algorithm* over loosely-coupled processor systems. It represents a combination of workload and interprocessor communication aspects and allows performance estimation of corresponding simulation. We have developed our method in the context of *parallelTEXSIM*. First experiments with models of real processor structures using this simulator are encouraging for further development of our valuation method and its integration into new partitioning algorithms. Our technique is not bound to a special simulator.

Currently, work is in progress to optimize algorithms for the realization of partition valuation. In future work we want to distinguish classes of actions regarding to their contribution to simulation run-time. Furthermore, heterogeneous target architectures for parallel simulation will be subject of our investigations.

ACKNOWLEDGEMENTS

This work was partly supported by DEUTSCHE FORSCHUNGSGEMEINSCHAFT (DFG) under grant **Sp 487/1-2**. We would like to thank K. Lamb, H.-W. Anderson (both at *IBM* Böblingen), D. Zike and W. Roesner (both at *IBM* Austin, TX) for providing *SP2* access, software support and many valuable discussions. Moreover, we are grateful to our students D. Döhler, R. Reilein, H. Hennings and Th. Siedschlag for their efforts in algorithm implementation and testing.

REFERENCES

[1] W. G. Spruth, *The Design of a Microprocessor* (New York, Berlin, Heidelberg: Springer, 1989).

[2] D. Döhler, *Entwurf und Implementierung eines parallelen Logiksimulators auf Basis von TEXSIM* (Leipzig: University of Leipzig, Department of Mathematics and Computer Science, Diploma Thesis, 1996).

[3] W. Roesner, *TEXSIM for loosely coupled multi-processors - performance estimates, sizing* (Böblingen: IBM, 1993).

[4] K. Hering, R. Haupt, and T. Villmann, Hierarchical strategy of model partitioning for VLSI-design using an improved mixture of experts approach, *Proc. of 10th Workshop on Parallel and Distributed Simulation*, 1996, 106–113.

[5] N. Manjikian, *High performance parallel logic simulation on a network of workstations* (Waterloo: University of Waterloo, Department of Electrical and Computer Engineering and Computer Communications Network Group, Technical Report CCNG T-220, 1992).

[6] R. B. Mueller-Thuns, D. G. Saab, R. F. Damiano, and J. A. Abraham, VLSI logic and fault simulation on general purpose parallel computers, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* , 12(3), 1993, 446–460.

[7] C. J. Alpert and A. B. Kahng, Recent directions in netlist partitioning : a survey, *INTEGRATION the VLSI Journal*, 19, 1995, 1–81.

[8] K. Hering, *Parallel cycle simulation* (Leipzig: University of Leipzig, Institute of Computer Science, Technical Report 13, 1996).

[9] D. Culler, R. Karp, D. Patterson, A. Sahay, K. E. Schauser, E. Santos, R. Subramonian, and T. von Eicken, LogP: Towards a realistic model of parallel computation, *4th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, 1993, 1–12.

[10] Z. Xu and K. Hwang, Early prediction of MPP performance : The SP2, T3D and Paragon experiences, *Parallel Computing* 22(7), 1996, 917–942.