Abstract—DCim++ is a C++ library developed for object oriented hardware design, modeling and distributed simulation. DCim++ enables C++ to be used as an OO HDL, which supports concurrency in description, inheritance in design and distributedness in simulation. Design simulation results are obtained by running C++ programs on a network of workstations. The Message Passing Interface (MPI) library has been used in the implementation of DCim++ as the basis of communications required for distributed simulation. In our simulation scheme, we have not considered any central management unit in order to defy performance degradation, instead only a coarse-grain synchronizer is used to keep the distributed components synchronized. This paper explores the structure of the DCim++ library and its mechanisms. The process a designer has to go through in order to design a system using DCim++ and conduct its distributed simulation leaving communication complications to DCim++, has also been presented. Finally, the results of our uniprocessor and distributed simulations for ISCAS benchmark circuits show high degrees of performance gains.

I. INTRODUCTION

OBJECT-ORIENTED and distributed computing concepts have been vastly generalized to all aspects of system design. Besides the ever-increasing application in software engineering, these concepts are now becoming among the main basics of system engineering [2][3]. As a result of the philosophical shift from functionalism to structuralism, the designer needs not focus on performing a task in a top-down manner. The behavior of the total system can be emergent rather than planned. New tasks can be performed as a result of novel collaboration patterns among existing objects of a system. Also, the reusability of the design process can be considerably enhanced by utilizing inheritance. Although hardware engineering has lagged behind the software engineering community in adopting modern bottom-up design technologies with emergent consequences, structuralist thinking is in many respects inherent to the former domain.

Additionally, as the size of the designs to be simulated rapidly grows, the need for fast simulation tools cannot any more be satisfied by sequential simulators [4]. One of the approaches to a solution for this problem consists of parallelizing the system and distributing its execution on more than one processor. This can be facilitated by exploiting the inherent parallelism of the models to be simulated. Hardware systems have many inherent parallel features. This fact together with the large time consumption of the simulation of these systems has led to the utilization of distributed simulation schemes.

In [5], a parallel logic simulation scheme on a network of workstations using a parallel virtual machine (PVM) has been presented. This scheme uses an event-driven engine as its simulation base. Another approach to parallel simulation of digital systems has been by way of distributing sub-tasks of existing simulation-engines such as VHDL which is presented in [6]. There has also lately been work on the distribution of the Verilog simulation engine sub-task for the simulation of ever-growing digital systems [7].

This work proposes DCim++, a C++ library based on Cim++ [1] as a framework for object oriented hardware design and distributed simulation. This framework supports concurrency and inheritance in the phase of design, and eliminates the need for any explicit simulation engine in the simulation phase. A designer using DCim++ utilizes this library without the concern of communications which will eventually be needed for distributed simulation because these are implicitly accounted for in DCim++. This distribution eventually gifts the designer with a gain in time of simulation without complicating his work beyond his own design problem.

DCim++ is extensible in two ways. Firstly because of its C++ nature, it is useful in heterogeneous paradigms such as hardware/software co-design. On the other hand, it can also be extended to a framework for multi-agent system design.

Section II discusses the Cim++, the base of DCim++, its structure and simulation scheme and design examples. Section III describes the approaches and mechanisms employed in DCim++.

Section IV presents the results of applying our method to ISCAS benchmark circuits, and the paper is concluded in Section V.

II. CIM++, OBJECT ORIENTED DESIGN AND SIMULATION

At the first glance, it seems that hardware modules are the most important objects of a hardware system, but in fact, such