King Fahd University of Petroleum and Minerals
College Of Computer Sciences and Engineering
Computer Engineering Department

COE 485 – Senior Project 1 (T051)

Parallel Communicating Models Simulator

1. Introduction

The process of designing high performance architecture requires a lot of experimentation and evaluation. There are many possible ways of conducting the experimentations like:

- Describe the architecture using HDLs (Hardware Description Languages):
- Use of Network Simulators like OpNet.
- Describe the architecture is HLL (like C).

In the HDL approach, the architecture designer is limited by the nature of the HDL itself. HDL simulators are by construction targeted towards the simulation of digital logic circuits and models. Their event-driven paradigm is unsuited for applications where the designer is trying to find his way. In the process of high performance architecture design, the design needs to rapidly experiment many aspects of the architecture. The HDL description is unsuited for such rapid change of description. High Level Languages such as C are well adapted for such task. The designer of high performance architectures also needs to conduct a large amount of data collection to be able to evaluate many ways of implementing the requirements. This is only done through executing a large number of simulations that go far beyond the traditional HDL simulation time. In HDLs, the designers simulate just few tens to hundreds of milliseconds to be able to verify the design of the circuits implementing the blocks of an already validated architecture. In high performance architecture design, the architecture itself is the subject of the study. This is the reason why simulations in such context need to go far beyond the few milliseconds in being able to simulate up to tens or hundreds of seconds of architecture behavior. This means that the simulator that will carry these tasks should be able to realize that.

2. The Simulator

An open, cycle-based simulator written in C is the proposed solution to produce a high performance architecture CAD tool. The simulator will simulate instances of parallel communicating finite state machines. Each finite state machine is an instance of a model that has the following attributes:

- An interface which represent generic communication ports. Every interface comprises a multitude of communication ports. Each port has:
  - An identifier
  - A direction: input, output or bidirectional
  - A size expressed in number of bytes

    - A position on the icon for graphic representation: North, South, East and West.
- An instance data area defined using standard C (typedef-like) record structure that will contain values on a per-instance basis
- A model data area holding parameters that will hold values which scope covers all instances of the same model at the same time.
- Probe area. This is a data area that is used to compute data collection and analysis through the simulation by providing means of collecting data and step-by-step computations of significant measurements.

The project comprises mainly two tools:

- An editor: GUI based. Used to edit architectures by being able to build the architecture itself. It will contain many features of user-based or automatic instantiation.
- A model editor used to generate model files and modify the configuration of the simulator.
- A compiler to put together the models of the simulator along with the simulator library. The compilation step can be invoked from the editor.
- A simulator that is command line based and that is generated from the compiler.

The simulator is independent from the architectures and is capable of executing any architecture that uses instances of models which are included in it.

3. Division of Work

This project can be thought of as a single project in which case it will need a large team to be able to complete it in one semester (5 or 6 students).
Because it is more difficult to manage one large team, students may also decide to segment the project as sub-projects that will be conducted in parallel by several smaller groups of students. In this case, the sub projects will be:

- The Editor
- The simulator
- The model editor, the compiler and the integration between the editor and the simulator.

4. Tools

The editor is a full GUI tool that should be written in Java. The simulator should not have any GUI and should be written in C. The model editor should also be written in Java since it is GUI-based. The compiler is a text generator (Generates Makefiles) and can either be written in Java or C.