Enabling Extreme-Scale Circuit Modeling for Massively Parallel Discrete-Event Simulations

Outline

Thank Chris. Today I'm talking about enabling circuit simulations. My key contribution is a *complete* software-based workflow. I provide the means for automatically transforming domain-specific designs into an accurate PDES model, one that can be run on today's supercomputer systems.

In order to do this, I created three integrated tools. 1st I automatically generate the model of the logic gates. 2nd I map the full circuit model onto the simulation world view 3rd I designed an API which can efficiently load the circuit model into our simulation and create restart checkpoints.

Background

Before I get into my contributions, I'm going to define some terms for both parallel discrete-event simulation (PDES) and circuit design

Logic Gates

Logic gates are a small component of logic within a circuit. Many people here may remember the AND, NOT, OR boolean logic functions. These gates operation on incoming electrical signals, in the form of 1's or 0's, True or False.

Each of these AND gates has 2 input pins and one output pin. The AND gate itself is described in a library file written in the Liberty language. This collection of gates is a circuit design, and is expressed in a netlist.

DES

In discrete-event simulation, the state of the simulation is created from logical processes (LPs) and the events they send to each other. Here, each gate is one LP and the electrical signals passing between them are called events or messages. The simulation engine I use is called ROSS.

Watch the electrical signals flow through this circuit.

!! ANIMATION !!

PDES: Conservative

In *parallel* discrete-event simulation, the whole simulation is broken apart across multiple compute nodes. This adds some extra overhead for synchronization, and there are two options. First we have conservative synchronization, through the YAWNs algorithm. We constrain the system with a global lookahead window: the whole system synchronizes before it starts simulation of the current window, and no LP can send an event into the current window.

!! ANIMATION !!

PDES: Optimistic

The other option for global synchronization is an optimistic approach through the Time Warp algorithm. Each local compute node is allowed to process events as fast it receives them. When an out of order event arrives, the local node preforms. recovery. It rolls back time, the moves forward again with the correct event ordering. With ROSS, we use reverse computation (+1 becomes -1) to undo LP state changes.