

## Re: Simulating simple electric circuits

---

*Source:* <http://coding.derkeiler.com/Archive/Python/comp.lang.python/2007-05/msg01075.html>

---

- *From:* Dave Baum <Dave.Baum@xxxxxxxxxxxx>
  - *Date:* Mon, 07 May 2007 14:56:56 -0500
- 

In article <5a9838F2nlbanU1@xxxxxxxxxxxxxxxxxxxx>, Bjoern Schliessmann <usenet-mail-0306.20.chr0n0ss@xxxxxxxxxxxxxxxx> wrote:

Hello all,

I'm trying to simulate simple electric logic (asynchronous) circuits. By "simple" I mean that I only want to know if I have "current" or "no current" (it's quite digital) and the only elements need to be (with some level of abstraction to my specific problem)

- sources (here begin currents)
- ground (here end currents)
- joints
- switches (which are able to let current pass or not, depending on outside influence)
- loads (which can signal change in current flow to the outside --- just like a light bulb)

Are you trying to do logic simulation (digital) or analog circuit simulation? The only reason I ask is that the simulation techniques are very different, and you used both "logic" and "current" in your description, so I'm not quite sure which direction you are heading.

For analog:

If you are ignoring time related effects (no inductance or capacitance), then the system is solvable as a set of linear equations. Basically your circuit consists of a set of nodes and edges. Wires are edges, joints are nodes. An open switch is nothing, a closed switch is an edge. A load is an edge. You'll have to assign resistances to the edges (anything non-zero will do) in order for the equations to make sense. Then you can use Kirchoff's laws to analyze the circuit and construct the equations to solve. A good linear algebra library (numpy) will help in solving the equations.

## Re: Simulating simple electric circuits

Opening or closing a switch would result in a new set of equations, and thus a new solution. You might be able to get clever and model open switches as edges with infinite resistance, which would allow you to skip the Kirchoff stuff each time a switch is flipped. You'd only have to change coefficients and solve the system of equations. However, the system of equations would be singular, so you'd have to do something like an SVD rather than an inverse.

I don't want to go into too much more detail about this one because I have a hunch you are really looking at digital, but if you're interested in the analog approach let me know and I'll fill in more of the details.

For digital:

Event based simulation is typical here. These simulations generally are concerned with voltage, not current. A circuit consists of signals and devices. At any given time, each signal has a certain state (high/low, on/off, 9 level logic, whatever). Devices are connected to one another by signals. You'll also need events (a signal, a time, and a new state), and an event queue (events sorted by time). This is easier if each signal is driven by at most one device:

- 1) pop the next event off the queue
- 2) if the event's signal's state is the same as the new state, go to 1
- 3) set the event's signal's state to the new state
- 4) for each device that is attached to the signal, run the device's code, which should look at all of its inputs, and post new events to the queue for any outputs (do this even if the computed output is the same as the current output). These events are usually posted for some time in the future (1 simulation 'tick' is fine).
- 5) go to 1

This approach is pretty simple to do in Python. I wrote a sample digital simulator a while back and the core of the simulator was around 50 lines of code. Rounded out with some basic logic gates and helper functions to probe the simulation, it was around 150 lines. It was only 2 level logic and signals could only be driven by a single device.

The devices that you want to model (switches, loads, etc) don't have explicit inputs and outputs, and you'll need to deal with a signal being driven from multiple sources, so it will get a bit more complicated. You will probably also need 9 level logic (or something equivalent) to deal with the fact that ground beats a source through a load when determining a node's state.

The basic idea is that each signal has multiple drivers, each of which has an internal state. When a device wants to set an output, it only changes its driver's state. The signal then has code that looks at the state of all drivers and combines them in some way (this is called a

## Re: Simulating simple electric circuits

resolution function). That combined state is what devices see when they read the signal.

It isn't \*that\* complicated to implement, but if you can turn your problem into one with 2 level logic and no multiple drivers, then it will be easier to write and debug.

Dave

.