Frequently Asked Questions

Questions

Answers

1. General

1.1. The simulator sucks. There isn't even a graphical interface! Why?

The simulator engine is designed as a learning tool in object-oriented analysis, design and implementation. The engine is implemented in less than 300 lines of Java source code (not counting comment and blank lines).

1.2. It's hard to believe the whole thing is implemented in only 300 lines of code (even if the user interface is primitive). How is it possible?

It is the simulation engine package (`ca.ryerson.kclowes.digsim`) that is implemented in a few hundred lines of code. There is not even a single line of code in that package that implements any kind of user interface.

The primitive command line user interface is implemented in a separate subpackage (`digsim.ui`). It has no code to perform simulation; rather, it uses the mechanisms available in the `digsim` package to implement the user-interface.

1.3. Simulation should be fast. Doesn't the use of an essentially interpretive language like Java compromise the speed?

Perhaps...but I think not. By the way, the premise that Java is an *interpretive language* is wrong. It is compiled (albeit to a virtual machine language rather than native machine language). Yes, the virtual machine language is in priciple interpreted; however, modern interpreters that exploit the just-in-time compilation technique do translate the virtual code to native code on the fly and achieve respectable performance.

An advantage of using Java is the platform-independence of the code.

1.4. What are the performance metrics?

A simple metric for an event-driven simulator is the number of events processed per second. On my laptop, the engine can process about three million events per second.

1.5. How was that measured and under what conditions?

I used the simplest possible circuit that would generate an infinite number of events: an inverter with its output connected to its input. The circuit was described and simulated with the command-line user interface with:

```
        Nand a a
        step 100000000  //simulate 100 Million events
```

This took about 30 seconds—hence the claim of 3,000,000 events per second.

The metric depends on various factors including:

Computer hardware:
A laptop based on a 1.6GHz Centrino processor with 512 MB of RAM.
Operating system:
Microsoft Widows XP.
Virtual Machine:
Java 1.4.2 VM from Sun.

The exact details:

Java VM client mode:
  Invocation:
  `java -jar digsim.jar`
  Time:

23.5 seconds
Events/sec:
4,250,000/second

Java VM server mode:
Invocation:
`java -server -jar digsim.jar`
Time:
14.9 seconds
Events/sec:
6,890,000/second

1.6. What are the differences between Java "server" and "client" modes and what are these modes?

There are many options that can be specified when the Java VM is invoked. The "client mode" (which is the default) has a relatively quick start-up overhead at the cost of performing fewer runtime optimizations. The "server mode" (which must be explicitly set) has a slower start-up because it performs more aggresive optimizations.

When the simulation involves only a small number of events, the client-mode version will be faster since the total time will be swamped by the start-up time with very little time spent on the actual simulation. However, the opposite situation occurs when there are many events to process in the actual simulation; in this case, the "server mode" is better.

2. Future

2.1. Will there be future versions of digsim?

I don't know.

The current version is good enough for its primary purpose: a demonstration teaching tool in Object Oriented techniques. Sure, there are bugs and glaring limitations; if the intent of the *digsim* project were to produce a real *Digital Simulation Application*, there is a lot of work to do.

2.2. What the "bugs" and "glaring limitations"?

There are few (to my knowledge) bugs in the simulation engine itself.

The glaring limitations include:

Simulation engine:
No node values other than 0 and 1; something like a tri-state device (with a

high-impedance output in addition to 0 and 1) is difficult to simulate. (Fixing this would also require the engine to know about outputs that drove a common node.)
User interface:
Besides the obvious lack of a Graphical User Interface