

Re: WSJ article on software liability

Source: <http://coding.derkeiler.com/Archive/General/comp.programming/2005-02/1338.html>

From: CTips (ctips_at_bestweb.net)

Date: 02/27/05

Date: Sat, 26 Feb 2005 20:19:28 -0500

Traveler wrote:

>>>> *Traveler writes:*

>>>>

>>>>

>>>>> *The Silver Bullet or How to Solve the Software Crisis:*

>>>>

>>>>> *Silver Bullet?*

>

>> *Seems to me what you're advocating here is a natural progression, and*

>> *one that's been on the tip of my tongue for a long time. Hope it*

>> *doesn't get *too* much like hardware though*

>

>

> *It will be better than hardware because of the flexibility.*

I think you've never worked on any mediumish (100k+ logic gate ~ 10kloc program) ASIC-design, particularly recently. Can't say for sure, but I'd bet fairly good money on that.

A quote from your site

(<http://users.adelphia.net/~lilavois/Cosas/Reliability.htm>):

"hardware faults are mostly physical faults, while software faults are design faults, which are harder to visualize, classify, detect, and correct."

Bullshit. In a chip, there are *LOTS* of faults that are design faults. Quite a few of them are bad enough that the chips are never released, and have to be redesigned. Sometimes, they're released with a long list of errata or work-arounds.

In particular, if you consider chips that implement algorithms (ASSPs) as opposed to glorified state-machines (simple CPU), you find more and more design bugs.

I've worked on both hardware and software, and (for the same application) its easier to get the software *design* right. I'm not even going to talk about how mcuh extra pain the synthesis + back-end work adds.

>
>>-- *rather, I fancy something*
>>*more along the lines of "It's alive! It's alive!!"*
>
>
> *You mean you want something tangible? Well, somebody has to build it*
> *first. What I am proposing is a revolution, a radical approach to*
> *software construction that will require new CPUs optimized for a*
> *radically new software model. This kind of stuff does not happen*
> *overnight.*

Again, bullshit. Have you heard of FPGAs? Given the kind of stuff you're proposing, it should be relatively straight-forward to implement it on an FPGA, either by synthesizing your COSA-machine on the FPGA, or by compiling your COSA programs directly for the FPGA.

See the specs on the latest series of FPGAs from Xilinx or Altera. Gazzillions of gates plus gobs of memory.

> *I have been saying essentially the same thing for close to twenty*
> *years: there is a fundamental flaw in the way we program our*
> *computers. It's only now that the software reliability crisis is*
> *regularly making the evening news that Project COSA is beginning to*
> *attract attention from the big software companies. Reliability*
> *engineers have failed to solve the problem and they have failed*
> *miserably. It's time for a change.*

Change? Have a look at Gul Agha's work on Actors. Or, CSP/Occam on the transputers. Or interrupt-driven programming on avionics [there are even a couple of MILSPEC processors that were designed to support this stuff]. Or petri-nets. Or data-flow.

There's a lot of stuff that has been tried in the past. It collapses for several reasons. Many of which have to do with the fact that a parallel decomposition of most problems is very difficult, both because the parallelism may not exist, and because it is difficult for people to generate the parallelism.

Consider sorting N numbers. For small N, one can design parallel hardware to do this efficiently. When N becomes very large [say 256K 4 byte numbers = 1GB of memory], it degenerates into a state machine which does some equivalent of combining smaller sorts [~merge sort] or into a machine which streams data through it, finding the nth-maximum [~bubble sort], or something else. Any way you implement it, it turns out that the hardware is basically a sequential state-machine, possibly with some small parallel components.

Try implementing this problem in any language, and you'll find that the program is about the same, and has a major sequential component.

comp.programming: Re: WSJ article on software liability

> *Louis Savain*

>

> *The Silver Bullet: Why Software Is Bad and What We Can Do to Fix it*

> <http://users.adelphia.net/~lilavois/Cosas/Reliability.htm>