

Re: A chip too far? Where is your solution Mr Larkin?

## Re: A chip too far? Where is your solution Mr Larkin?

---

*Source:* <http://sci.tech-archive.net/Archive/sci.electronics.design/2008-08/msg03644.html>

---

- *From:* Phil Hobbs <[pcdhSpamMeSenseless@xxxxxxxxxxxxxxxxxxxxx](mailto:pcdhSpamMeSenseless@xxxxxxxxxxxxxxxxxxxxx)>
  - *Date:* Sat, 23 Aug 2008 12:53:53 -0400
- 

Jan Panteltje wrote:

On a sunny day (Tue, 19 Aug 2008 18:41:38 GMT) it happened Rich Grise  
<[rich@xxxxxxxxxxxxx](mailto:rich@xxxxxxxxxxxxx)> wrote in <[pan.2008.08.19.18.40.10.445123@xxxxxxxxxxxxx](mailto:pan.2008.08.19.18.40.10.445123@xxxxxxxxxxxxx)>:

On Tue, 19 Aug 2008 15:07:51 +0000, Jan Panteltje wrote:

A chip too far? Where is your solution Mr Larkin?

[http://money.cnn.com/2008/08/13/technology/microchips\\_copeland.fortune/index.htm?postve](http://money.cnn.com/2008/08/13/technology/microchips_copeland.fortune/index.htm?postve)

Blowing in the wind, all blowing in the wind.

Faster chips (and the concomitant bloatware) have been making slower computers since CP/M. ;-)

Cheers!  
Rich

In my view, that professor, if he is real, must at one point be willing to say:

**YOU CANNOT ALWAYS CONVERT SEQUENTIAL CODE TO PARALLEL CODE.**

But if he is into his research grant, OK, that may take a while.

As a programmer I say:

Find where the sequential program is spending [most of] it's time.

Try to optimise that code area, use hardware if need be,

FPGA is the perfect solution here.

So we have then mother boards with on the fly programmable logic as part of the programs that run on it.

Maybe the obstacle here is, that for both IBM and Intel, they have cores.

More cores is simple for them, 'If I had a hammer, I would hammer in the morning, hammer in the evening, all over this land'.

This land is my land ....

Now this land WAS their land.

They have an IP problem, have to team up with Altera, or Xilinx, others, and

Re: A chip too far? Where is your solution Mr Larkin?

combine their forces to write a compiler that finds the time consuming part of the sequential code, checks the available hardware resources as in FPGA, and converts that time critical part to gates, unfolding as many loops as possible.

Use gates (not the person ;- ) as we now use memory resources.

Now here is a solution that can WORK.

I am no professor, but this can be realized.

HELLO IBM? HELLO INTEL?

Intel is lead by a sales druid, IBM too?

Somebody should try that way.

Very late to this party, having been incommunicado in rural Alaska for a week—but the biggest problem with reconfigurable hardware is the inconceivably vast task-switching overhead. If you can dedicate reconfigurable cores to tasks, a la JL's model, this is reasonable at the margins, but it won't ever be a mainstream programming model.

Cheers,

Phil Hobbs

.