

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

Source: <http://sci.tech-archive.net/Archive/sci.electronics.design/2008-08/msg03646.html>

- *From:* Jan Panteltje <pNaonStpealmtje@xxxxxxxxx>
 - *Date:* Sat, 23 Aug 2008 17:18:41 GMT
-

On a sunny day (Sat, 23 Aug 2008 12:53:53 -0400) it happened Phil Hobbs <pcdhSpamMeSenseless@xxxxxxxxxxxxxxxxxxxxx> wrote in <48B040A1.5060209@xxxxxxxxxxxxxxxxxxxxx>:

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

Won't matter, configuring the hardware will be as loading a program into memory, you configure it ONCE for each program. So, if I have some sequential code, with some part optimised in gates, then the OS would load the sequential code in memory, stuff the hardware config bit stream in the FPGA, and have the program do it's thing. With some tricks only part of the FPGA would be used. An other program, loaded at a different time, would use an other part of that FPGA. Until all hardware resources in that FPGA are used, then no more programs that use FPGA real estate can be loaded (OS task to verify that). So the time issue you talk about, is only related to starting up the program. the HDL would have been synthesised at the same time the code was compiled.

So, as to 'task switching', start thinking `_parallel_`, the gate configurations do not change during task-switching, each program has it's own gates (FPGA real estate) as each program has its own memory (neither do you swap out sequential code form memory during a task switch, then reload the code).

What is needed to accomplish this is very large FPGAs that can be partially reconfigured, and compilers that spit out HDL (or bit streams) for those FPGAs for time critical parts of the code, a loading mechanism, OS support to keep track of resources, and hardware interface standards, This is all possible, and in some other postings links have been given.

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

The main problem I see is an IP one, companies often share patents though...
and compiler design, new version, or new type of gcc, now that would be cool.

.