

## Re: a dozen cpu's on a chip

---

*Source:* <http://sci.tech-archive.net/Archive/sci.electronics.design/2008-05/msg01878.html>

---

- *From:* Jan Panteltje <pNaonStpealmtje@xxxxxxxx>
  - *Date:* Fri, 16 May 2008 10:25:54 GMT
- 

On a sunny day (Thu, 15 May 2008 14:24:57 -0700) it happened John Larkin <jjlarkin@xx> wrote in <le8p24pni8jr47uahd75f82f3hi189112v@xxxxxxxx>:

Nanometer transistors are fast and free.

Actually they are not, those 80-cores will be difficult to make (yield),

If you want 250 cores, build 300 and use the 250 that work. So a chip can have 50 defects and you can still sell it. Or build one giant CPU on the same silicon and toss it if has a single defect.

Well, how much you get from a wafer... and you still need to test 300, testing is complicated even with single cores, takes time, configure the chip... Intel needs you!

I think AMD is already selling multicore processors that have one bad core. Xilinx sells FPGAs with bad cells, but that work for a given compiled design.

Yes I have heard about 3 cores, have not used or seen any though.

look at all the stuff IBM had to throw away with Cell, that is why the Sony PS3 has one peripheral processor less enabled in the Cell.  
The 100% chips were too expensive,

Re: a dozen cpu's on a chip

gigabit world. The majority of guys here are adamant that things will never change, a pretty radical position for engineers to take.

mm you keep sticking that in every bodies mouth, but when I asked how you would spread a monolithic resources sucking application over 'n' CPUs you remained silent.

I already suggested that a few of the cpu's could be floating-point monster number crunchers, and most could be dumber, slower integer machines. A TCP/IP stack doesn't need much floating point power.

If I understand this right (but it is no answer to my question), then that is what we already have.

For example the Cell [http://en.wikipedia.org/wiki/Cell\\_\(microprocessor\)](http://en.wikipedia.org/wiki/Cell_(microprocessor)) has one Powerprocessor core and 8 SPU.

This year, quote:

'In May 2008, IBM introduced the high-performance double-precision floating-point version of the Cell processor, the PowerXCell 8i[16], at the 65 nm feature size.

Why do I carry on about Cell? At least I know something about that processor.

But also notice the size of the chip, in the picture in the above link.

Now imagine 80 cores, add some FPU real estate too...

Even at 45 nm the 80 core should be huge = expansive.

And that is one issue.

The other one you conveniently forget is that, if each core has its own memory,

where is the overhead in moving data... sync. etc.

They'd surround a shared cache. They wouldn't bother the common cache when they execute out of local cache, or when they use the small local stack and variables rams. That makes the shared cache much more efficient, since it not being invalidated by a lot of unnecessary traffic.

Well, like I said, Intel needs you :-)

Re: a dozen cpu's on a chip

## Re: a dozen cpu's on a chip

One thing I've always thought that CPUs should have is hardware task switching, a register that declares which task or thread the core is running. That would instantly remap everything... the registers, the memory mapping, everything. That would make context switching have zero overhead, and allow full hardware protection. But nowadays, one might just as well have multiple cores. That would be faster, and avoids some cache efficiency and pipeline issues.

mm

There are many architectures, I dunno all of them.

It is interesting that 100% of the responses to my posts have been destructive, and none additive. I sure hope you guys don't actually work that way.

I noticed Mr Brown's remark to you, and I second it:

<quote>

You are beginning to take on the air of a netkook harping on how a CPU per thread will solve all problems and bring sweetness and light to the world.

<end quote>

Some other aspects, you likely remember the 'transputer', it died.

When you write a threaded program, the thread can access the same memory (say variables) as the main routine.

If however the thread was to run on a different CPU with its own memory, then you'd have to move data (copy memory).

Look at the Element Interconnect Bus in the Cell link above.

I think the English expression is 'bottle neck' for such data paths between the different cores.

What it in reality boils down to, is that one will have to make very conscious choices what to run on what core, when and how to transfer data between the different units, something that could perhaps not be done by a compiler or OS without guidance from a designer.

So, to put it simply, many cores (> 10, 80, whatever) may have no practical advantage, even if some of those cores are only FPU, pipelines need to be filled, prefetches, the whole thing needs to be synchronised.

It REALLY is VERY complicated.

Especially for a 'general purpose' application and computer.

For a media box: demodulation, decryption, decoding, graphics, can simply in many cases be assigned to a few cores, but after 6 ???

And those PCs will be media boxes, things like VOIP are small things that can run in the background too.

Re: a dozen cpu's on a chip

Re: a dozen cpu's on a chip

I am already way out here, sure, but the guys do HD editing on an Intel dual core, fast enough, real time.

You do not see any washing machines with 200 HP engines either.....

So unless some new application surfaces, 10 cores should be enough?

I put the question mark because of Bill Gates '640kB is enough for everybody'.

But he never had vision! He even gambled wrong on Blu-Ray.

Good salesman though.

.