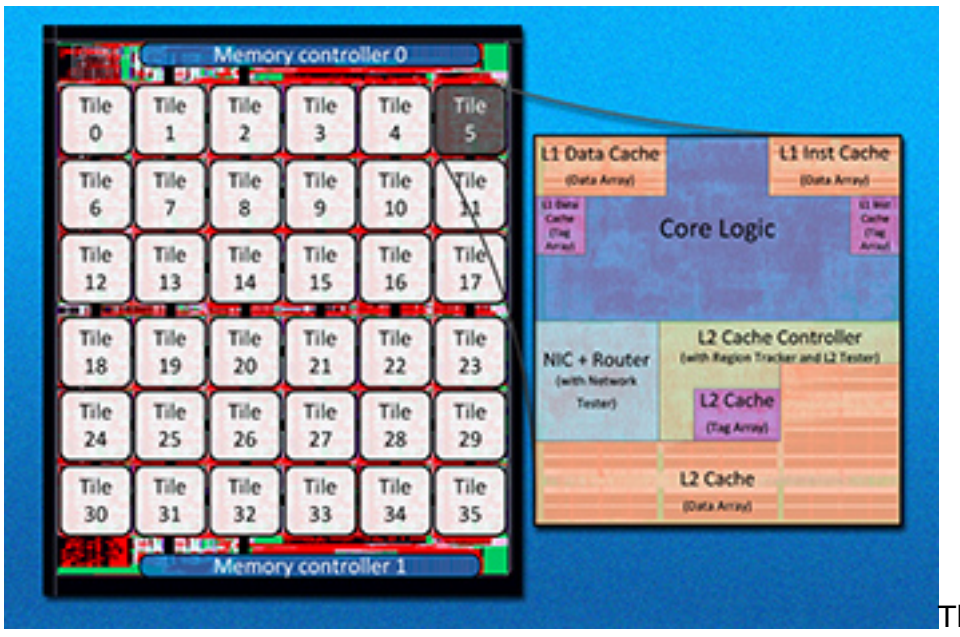
Chip with 36 Cores Unveiled

Larry Hardesty, MIT



The more cores — or processing units — a computer chip has, the bigger the problem of communication between cores becomes. For years, Li-Shiuan Peh, the Singapore Research Professor of Electrical Engineering and Computer Science at MIT, has argued that the massively multicore chips of the future will need to resemble little Internets [1], where each core has an associated router, and data travels between cores in packets of fixed size.

This week, at the International Symposium on Computer Architecture, Peh's group unveiled a 36-core chip that features just such a "network-on-chip." In addition to implementing many of the group's earlier ideas, it also solves one of the problems that has bedeviled previous attempts to design networks-on-chip: maintaining cache coherence, or ensuring that cores' locally stored copies of globally accessible data remain up to date.

In today's chips, all the cores — typically somewhere between two and six — are connected by a single wire, called a bus. When two cores need to communicate, they're granted exclusive access to the bus.

But that approach won't work as the core count mounts: Cores will spend all their time waiting for the bus to free up, rather than performing computations.

In a network-on-chip, each core is connected only to those immediately adjacent to it. "You can reach your neighbors really quickly," says Bhavya Daya, an MIT graduate student in electrical engineering and computer science, and first author on the new paper. "You can also have multiple paths to your destination. So if you're going way across, rather than having one congested path, you could have multiple ones."

Get snoopy

One advantage of a bus, however, is that it makes it easier to maintain cache coherence. Every core on a chip has its own cache [2], a local, high-speed memory bank in which it stores frequently used data. As it performs computations, it updates the data in its cache, and every so often, it undertakes the relatively time-consuming chore of shipping the data back to main memory.

But what happens if another core needs the data before it's been shipped? Most chips address this question with a protocol called "snoopy," because it involves snooping on other cores' communications. When a core needs a particular chunk of data, it broadcasts a request to all the other cores, and whichever one has the data ships it back.

If all the cores share a bus, then when one of them receives a data request, it knows that it's the most recent request that's been issued. Similarly, when the requesting core gets data back, it knows that it's the most recent version of the data.

But in a network-on-chip, data is flying everywhere, and packets will frequently arrive at different cores in different sequences. The implicit ordering that the snoopy protocol relies on breaks down.

Imposing order

Daya, Peh, and their colleagues solve this problem by equipping their chips with a second network, which shadows the first. The circuits connected to this network are very simple: All they can do is declare that their associated cores have sent requests for data over the main network. But precisely because those declarations are so simple, nodes in the shadow network can combine them and pass them on without incurring delays.

Groups of declarations reach the routers associated with the cores at discrete intervals — intervals corresponding to the time it takes to pass from one end of the shadow network to another. Each router can thus tabulate exactly how many requests were issued during which interval, and by which other cores. The requests themselves may still take a while to arrive, but their recipients know that they've been issued.

During each interval, the chip's 36 cores are given different, hierarchical priorities. Say, for instance, that during one interval, both core 1 and core 10 issue requests, but core 1 has a higher priority. Core 32's router may receive core 10's request well before it receives core 1's. But it will hold it until it's passed along 1's.

This hierarchical ordering simulates the chronological ordering of requests sent over a bus, so the snoopy protocol still works. The hierarchy is shuffled during every interval, however, to ensure that in the long run, all the cores receive equal weight.

Proof, pudding

Cache coherence in multicore chips "is a big problem, and it's one that gets larger all the time," says Todd Austin, a professor of

electrical engineering and computer science at the University of Michigan. “Their contribution is an interesting one: They’re saying, ‘Let’s get rid of a lot of the complexity that’s in existing networks. That will create more avenues for communication, and our clever communication protocol will sort out all the details.’ It’s a much simpler approach and a faster approach. It’s a really clever idea.”

“One of the challenges in academia is convincing industry that our ideas are practical and useful,” Austin adds. “They’ve really taken the best approach to demonstrating that, in that they’ve built a working chip. I’d be surprised if these technologies didn’t find their way into commercial products.”

After testing the prototype chips to ensure that they’re operational, Daya intends to load them with a version of the Linux operating system, modified to run on 36 cores, and evaluate the performance of real applications, to determine the accuracy of the group’s theoretical projections. At that point, she plans to release the blueprints for the chip, written in the hardware description language Verilog, as open-source code.

Links:
[1] http://newsoffice.mit.edu/2012/chips-as-mini-internets-0410
[2] http://newsoffice.mit.edu/2014/smarter-caching-0219