Architecting the Future: Dr. Marc Tremblay

By Brian Neal – March 2003

Introduction

Late last month, Sun held a press conference detailing its microprocessor roadmap, featuring a range of chips including the current UltraSPARC III, the dual-core UltraSPARC IV, the SMT-based UltraSPARC V, and the highly multithreaded Niagara processor. Last week we had the chance to sit down with Dr. Marc Tremblay and discuss some of the new processors and technologies on Sun's roadmap for the future. Dr. Tremblay is a senior Distinguished Engineer at Sun and Chief Architect for the Processor and Network Products Group. He was a Co-Architect of Sun's first 64-bit processor, the UltraSPARC-I, and was Chief Architect for the UltraSPARC-II and the MAJC program. He holds 55 patents in various areas of computer architecture and has spent a lot of his time at Sun studying Chip MultiProcessing (CMP) and Chip MultiThreading (CMT).

The topics of CMT and CMP, or what Sun calls "Throughput Computing" are a key focus of the discussion, so if you are not already familiar with the technologies or the implementations of them discussed here, you may want to refer to the following news posts:

- Sun Details UltraSPARC Roadmap (Feb. 26, 2003)
- Sun's Multi-Core Plans (Feb. 13, 2003)

CMP and CMT are techniques to scale the performance of multithreaded applications (or multiple processes/programs running on one CPU) through the integration of multiple cores onto a single silicon die and the execution of instruction streams from multiple threads on each core. Implementations of these technologies exist on the market today in the form of the Intel Pentium 4 ("HyperThreading" / SMT) as well as the IBM POWER4 and Sun MAJC-5200 (dual-core / CMP), to provide a few examples. The upcoming processors Dr. Tremblay discusses feature somewhat more radical implementations of these technologies, including designs that have fully embraced the multithreaded approach like the Niagara processor (4 threads per core, 8 cores per die, 32 threads per chip). Processors like Niagara are designed to provide high-performance on heavily multithreaded, branch-heavy code like that of many Enterprise/Server applications. In the case of Niagara, it is expected a single chip will provide performance equivalent to 15 current uni-processor blades -- essentially allowing it to equal the performance of a blade shelf on one piece of silicon.

A forthcoming article, "Multithreaded CPU Design," will explore the forces behind this shift in design style, its impact on CPU design, and what CPUs and systems various vendors are fielding with these technologies. Until then, we present this interview with Dr. Tremblay, who's currently working on a high-performance multithreaded microprocessor expected to deliver 30 times the performance of the current 1.2 GHz UltraSPARC III.

Chris Rijk [Ace's Hardware]: Marc, I was wondering if you are more of a researcher or a developer? In terms of what you actually do in your day-to-day work.

> **Dr. Marc Tremblay:** My job comes in [as the] Chief Architect for the Processor and Network Products Group and I'm leading the next generation of high-end super computing chips. So I'm doing some research but definitely applying it to the next generation of products.

Brian Neal [Ace's Hardware]: So what are you doing right now, Marc? Can you say?

> **Dr. Marc Tremblay:** You guys saw the announcement at the worldwide analysts conference and we talked about a part that was 15x faster than blades and another part that was 30x faster than our current systems. I'm working on the second one.

Chris Rijk [Ace's Hardware]: Sounds like it would be called the UltraSPARC VII.

> **Dr. Marc Tremblay:** Well, there's a whole process in finding a name, but we'll see. So far it's still a codename.

Chris Rijk [Ace's Hardware]: Recently you were raised to being a Sun Fellow, and I guess you are quite proud of that. I actually saw the press release and among other things it attributed to you it listed chip multiprocessing, multithreading, hardware scout, space-time computing, transactional memory, and execute ahead. I remember from the MAJC documents several years ago about space-time computing, which is speculatively using multiple threads to speed up the main thread or words to that effect. And I know what chip multiprocessing is, but I'm not quite sure what hardware scout, transactional memory, and execute ahead are. So I was wondering if you could explain those a bit more.

> **Dr. Marc Tremblay:** I'm surprised some of that technology made it out in the press release, but I'll spend a little bit of time on hardware scout because I know we talked a little bit about it.

Brian Neal [Ace's Hardware]: Yeah that was in the chalk talk the other week [(press teleconference on Throughput Computing)].

> **Dr. Marc Tremblay:** Well the main idea is that while running large commercial programs we find the processors sometimes waiting 70% of the time, so being idle 70% of the time.

Chris Rijk [Ace's Hardware]: Stalled on waiting for data, basically.

> **Dr. Marc Tremblay:** Yes. In large multiprocessor servers, waiting for data can easily take 500 cycles, especially in multi-GHz processors. So there are two strategies to be able to tolerate this latency or be able to do useful work. One of them is switching threads, so go to another thread and try to do useful work, and when that thread stalls and waits for memory then switch to another thread and so on. So that's kind of the traditional vertical multithreading approach.
>
> The other approach is if you truly want to speed up that one thread and want to achieve high single thread performance, well what we do is that we actually, under the hood, launch a hardware thread that while the processor is stalled and therefore not doing any useful work, that hardware thread keeps going down the code as fast as it can and tries to prefetch along the program path. Along the predicted execution path [it] will prefetch all the interesting data and by going along the predicted path [it] will prefetch all the interesting instructions as well.

Chris Rijk [Ace's Hardware]: So it's a bit like an intelligent prefetch.

> **Dr. Marc Tremblay:** Exactly.

Chris Rijk [Ace's Hardware]: And that's entirely hardware based -- does not need any special compilation or software support?

> **Dr. Marc Tremblay:** That's correct.

Brian Neal [Ace's Hardware]: Now is that on a per-core basis in a multi-core chip or is there a central memory controller there where that logic will deal with requests for every chip -- every core essentially.

> **Dr. Marc Tremblay:** So, I won't describe the details, but you can imagine that the hardware thread needs a lot of information needs a lot of information about the current running state of the program. It's basically taken over the execution and it's going to use the running PC, the branch prediction state and all of this. So, it's basically more effectively taking the place of the previously running thread.

Chris Rijk [Ace's Hardware]: The way I'm kind of thinking of it is when you get a data stall, you do a checkpoint and continue running the same program except you treat all loads as non-faulting and don't do any stores -- just running ahead to get data into the cache, but without using an extra thread.

> **Dr. Marc Tremblay:** Yes that's along those lines. Your ultimate prefetch that has runtime information. Blows away what we can do with software prefetch.

Chris Rijk [Ace's Hardware]: Ok so hardware scout is primarily designed to improve the performance of a single thread, and mostly for databases or would that be for stuff like HPC as well?

> **Dr. Marc Tremblay:** Well it will work for both. HPC is [bandwidth limited] and it's easier to predict so it software prefetches pretty well. The spaghetti code is where we truly see benefits.

Chris Rijk [Ace's Hardware]: There's lots of that.

Can you describe transactional memory or execute ahead? They sound [like] related technologies to using multiple threads or advanced ways of improving performance. Though I'm not quite sure about transactional memory, that's rather vague.

> **Dr. Marc Tremblay:** At this point I'd prefer not to talk about it since we're filing intellectual property patents on that and it hasn't really been disclosed.

Chris Rijk [Ace's Hardware]: My other guess was that it was related to very high reliability stuff. For example, lockstepping CPU cores so that if you get an error in one the other can take over or you can double check results or anything like that.

Going back a bit from Brian and my point of view, it's actually very welcome to see Sun being very open about future CPUs and directions because it's actually really hard to get information.

> **Dr. Marc Tremblay:** That was the whole purpose of doing the disclosure [in] that a lot of us within Sun feel real good about the future and the whole throughput computing path [has been with us] for a number of years and people outside of Sun did not really understand while we had a little bit of a smile while we're talking to them. So now [that] we're [explaining] to people like you and financial analysts, I would think we're working on the right strategy here.

Chris Rijk [Ace's Hardware]: I certainly hope you will [disclose] more information in the future.

> **Dr. Marc Tremblay:** Yeah no doubt about it.

Chris Rijk [Ace's Hardware]: Several years ago you were working on the MAJC design, which was dual-core and there was a discussion about multithreading. How long have you been back in the SPARC group, and were you mostly instrumental in pushing multithreaded and multi-core designs within Sun or is that something that kind of happened all at once?

> **Dr. Marc Tremblay:** Well, it's basically been my mission. So, MAJC started in '95-'96, and even while doing MAJC a couple years later I was involved in the proposal within Sun, interestingly enough with Bill Joy, where we were proposing a multithreaded multi-core chip. And you know, Sun's a big company and it takes time to move things around, so at some point when the first MAJC chip, the 5200, was taped out and almost shipping, I took the position of Chief Architect for the Processor and Network Products Group and from then on I almost exclusively spent my time on SPARC. So that was about two and a half years ago and since that day I've been pushing really hard to move our entire roadmap in the CMP and CMT space. So, within Sun, I'm probably the one person carrying that torch.

Brian Neal [Ace's Hardware]: There seem to be some similarities, at least spiritually, between the MAJC-5200 and the upcoming chips, at least in terms of dual-core, agnostic functional units (in some sense). How much have you carried from that design into what you're working on now.

> **Dr. Marc Tremblay:** Fundamentally, there are some MAJC docs that you guys never saw that really push the multithreading and so whenever we proposed something like this or started a project, we did some very detailed cycle-accurate simulators, and then we spend months understanding the data, understanding the performance, calculating IPC, covering stalls, and kind of doing the whole research work of what works, what doesn't work, how do we tune things. And that whole R&D [effort] is leading to that second dot on that roadmap I talked about [("30x current performance")] and is being led by the same people who did that second MAJC chip.

Chris Rijk [Ace's Hardware]: One of the things actually occurred to me more recently is that once you start going down that kind of road is that the effect to some extent [involves] developing benchmarks which are representative of multithreaded/server workloads. Compared to SPEC's benchmark suite, which is actually what a lot of people use these days to design CPUs; to some extent you [might] completely ignore that. So the cycle-accurate simulators that you are working on, is that similar or are you only working with server code with that sort of thing, or?

> **Dr. Marc Tremblay:** No. So, we'll have a variety of CMP chips and some of them really cater to throughput computing and heavy duty server-side processing. I personally like to run SPECint and SPECfp to give us really another indication of how well these processors perform across the board and can somewhat be called general-purpose.
>
> So in the project I'm working on, we have been running SPECint/SPECfp for a long time, as well as TPC-C, SPECjbb, SPECweb, app server stuff, and everything you find in the server room. So anything we can grab our hands on from low miss-rate applications like SPECint to fairly high miss-rate applications that can be predicted fairly well such as SPECfp, to spaghetti code like the more commercial-oriented stuff.
>
> So, we are trying to optimize pretty much across the board for some of these parts because if you start with reasonable IPC cores, and then you add throughput computing on top of that, boy, you end up with a chip that can do a lot of work per minute.

Chris Rijk [Ace's Hardware]: Niagara seems a bit more radical, seems much more server-only. From what Brian told me of the chalk talk, it sounds like the 30x chip was like what you described. It sounded like, say 3-4-way issue and simultaneous multithreading on top [with] lots of cores. So pretty good at everything and especially good compared to current stuff for server [code], whereas Niagara seems pretty much server only, like you wouldn't really want to run SPECint or fp on it.

Kind of related to that, I'm curious to see some actual examples of IPC in real programs because that's quite important from my point of view to try and estimate how Niagara can be so fast because it seems the cores themselves are very small. So I don't know if you can actually give us some examples of that, of IPC in various programs and how much time is wasted on branch mispredictions and how much time is wasted on data stalls.

> **Dr. Marc Tremblay:** These processors, as you notice, are fairly simple so we're not talking about the deep very [aggressive] pipeline like Pentium 4. These chips, by the way, will waste like 28 cycles on misprediction. So here we're talking about a fairly shallow pipeline that doesn't suffer much from branches. So the focus is truly on hiding the much higher cost of cache misses that go to main memory. And there, you take an application like TPC-C or a large database app, and it's typical to see at least a 1% per instruction miss-rate. Meaning that once you normalize the instructions, you're saying that if you have 1 instruction every 100 that misses, and that one takes 300 cycles to service, and assuming all other instructions execute in one clock, so the first 99 take 99 clocks, and 100 takes 300 clocks. So, roughly the whole thing takes 400 clocks. Your IPC is 0.25, and your CPI is 4.

Chris Rijk [Ace's Hardware]: I was hoping that you could actually give us some examples of current stuff at least so that we can have a way to [see] where things are currently having problems. So if you at like that 25% efficiency value, if you raise that to 75%-80%, the performance will naturally go up a lot. One of the things I want to try and be able to explain to our readers is effectively from basics is why the current methodology isn't the best way to go about things.

> **Dr. Marc Tremblay:** Well, the example I gave you is not far-fetched. So that should give you an example and then say 'wow, that processor is idle a lot', in this particular case 75% of the time. So then you start sizing how many threads [you] need to cover that latency, and then notice that if you do this with multiple threads and multiple cores, you also end up with this constructive sharing. Each thread has a pretty small cache, the miss-rate might actually go up, but that's ok.

Chris Rijk [Ace's Hardware]: As long as you have enough memory bandwidth.

> **Dr. Marc Tremblay:** That's right, so we turn a latency problem into a bandwidth problem. And as you guys know, trying to cut the latency of DRAM in half or by 75% is basically impossible, but providing a lot of bandwidth is a lot easier. And especially if you look at our current systems, we have a lot of bandwidth already; we just don't use it very well. So the factors we gave at the conference, the 15x and 30x, they don't require 15 times more bandwidth, they require that a) we use it more effectively and b) yes we're [ramping] it up.

Brian Neal [Ace's Hardware]: To some extent, it seems like hardware scouting is converting bandwidth into latency, or lower latency.

> **Dr. Marc Tremblay:** So in this particular example, yes, it's using bandwidth to reduce effective latency. Notice that the beauty of the hardware scout is that it's not speculative space. Basically speculation is your enemy if you're worried about power. Because if you speculate in parallel and try various paths, or do like Itanium and do predication [to] try both sides of a branch, you're wasting half your power. And the hardware scout and the techniques we're using are speculating in time so the processor was idle anyway, so we might as well use the power that was already budgeted.

Chris Rijk [Ace's Hardware]: With regards to branch prediction, one thing that occurred to me is that perhaps in very throughput optimized designs, maybe for something like Niagara, you could do something like when you have a branch coming out, instead of trying to predict it, you could switch threads instead and later when you can actually know which way that branch was going to go you could switch back to the old thread. Effectively, you could actually remove much of your branch prediction code. I think for what it sounds like for the 30x design it doesn't sound like you're going to go with that at all, but do you think that that might work for something like Niagara.

> **Dr. Marc Tremblay:** Well as you notice, the event or events that lead to thread switching can be much more sophisticated that what we've [indicated to] the press. The granularity at which you switch threads, for instance. Switching a thread is a 0 cycle process. So once you have provided that hardware capability, you're much more amenable to switching on events like you mentioned. Anything that would cause a stall for that matter.

Chris Rijk [Ace's Hardware]: Through it sounds like Niagara is just 1-way issue, it seems like it could achieve an average IPC of close to that in real server programs. Maybe around 70 or 80% efficient, which would be far better than anything else.

Brian Neal [Ace's Hardware]: There was an article in EETimes that said the core would be based on the UltraSPARC II for Niagara, and it seems like you're going for a much more simplified design that's targeted for a highly multi-core chip. So, is that incorrect [about] the USII?

> **Dr. Marc Tremblay:** Yes, that's incorrect.

Chris Rijk [Ace's Hardware]: That article actually said the chip was 340 mm^2, and in David Yen's presentation it actually had [a floor plan] showing the size of one core, and from that I actually estimated that the core would be just 8sqmm in size, which is really tiny. So that's kind of why I was wondering whether it was 1-way issue or not.

> **Dr. Marc Tremblay:** We haven't disclosed the internals, but as you can extrapolate from the floor plan, the core is smaller than an UltraSPARC II core.

Chris Rijk [Ace's Hardware]: Another thing I've been looking at is that once you start having multiple cores you need to synchronize data and access to the rest of the system, shared cache, and so on. The actual overhead of that would start to become an issue -- it's not close to 0. Would that be correct (the actual logic you need to dedicate to having multiple cores)?

> **Dr. Marc Tremblay:** Ok, so notice that's kind of where our expertise is in doing scalable systems -- large multiprocessor systems. So the good news here is that this is all happening [on die]. So [it's] between 1 order and 2 orders of magnitude faster [than off-chip interconnects]. So the beauty of this is that in a blade system (single chip based system), all the coherency is happening on chip and with much faster busses and much lower latencies. So we actually will see many more opportunities to even expand the scalability of lots of multithreaded applications and even to do auto-parallelization of programs.

Chris Rijk [Ace's Hardware]: Somewhat related to that, a question that sometimes comes up, instead of doing fairly simple cores, you could do a huge number of extremely simple cores, and one of the reasons I've thought of not doing that is that the overhead of actually managing stuff between them would start to become quite large if you had extremely tiny cores, as well as just the fact that you can have higher single-thread performance.

> **Dr. Marc Tremblay:** Right, so, the art is really to balance the whole thing. When we saw the introduction of RISC processors (us [SPARC], MIPS, then DEC Alpha, and HP), you easily saw differences of 2x with these offerings. So with CMT and CMP, I think we'll see that ratio go even higher because if you start out with cores that have a delta of 2x, the aggregation of these on a chip could easily lead to factors of four between two different implementations. The art is optimizing and balancing the cores versus the bandwidth of the L2 and the memory bandwidth, and a lot of it is done through science and running benchmarks and measuring everything. The rest is having a good understanding of what is efficient on our part and what applications will run well.

Brian Neal [Ace's Hardware]: Do you see out-of-order execution (OOOE) to be complementary to on-chip multithreading or is that something that's kind of obsolete or not worth the logic?

> **Dr. Marc Tremblay:** I think there are various forms of out-of-order execution and the stuff that tries to speculate in parallel is not really well suited because of the power requirements. It's really burning power and speculating to try to optimize single-thread performance at the expense of running other stuff that's non-speculative. So, we'll have to be very careful about applying that old style architecture to the CMTs.
>
> Now notice that for x86, you don't have much of a choice. [People] keep saying RISC technique won the war against CISC. Obviously CISC chips are doing just fine considering how well Intel is doing, but Intel had to go to hardware translation to go from CISC to RISC and basically they have a RISC pipeline using micro-ops. The enabler for that was advanced branch prediction techniques that allowed them to stretch the front-end of the pipeline to be able to do translation over multiple cycles or to introduce a trace cache to cache the translation and so on, and then they run the RISC pipeline in the back-end. The ordering of instructions is unrelated almost to how software scheduled them originally, at least the micro-ops are, so therefore an out-of-order engine in that space makes a lot of sense, although it does cost Intel a lot of power.

Chris Rijk [Ace's Hardware]: To clarify slightly, with regards to out-of-order, it's not so much you're necessarily against it inherently, it's just to some extent more specific implementations, particularly reducing speculation overhead?

> **Dr. Marc Tremblay:** Yes.

Chris Rijk [Ace's Hardware]: One of the critical things you have once you start having multiple cores is that the power consumption per core has to become a lot smaller. So if you have four cores and a 100W power budget, each cores going to have to be around 20W max. But complementary to that, because each core becomes simpler, the power's going to be a lot lower, and because you don't need to have as high a clock rate, which helps reduce power consumption per core.

It seems like you would need fewer engineering resources per core [with this approach]. For example, it sounds like the UltraSPARC V core is actually quite complicated and I heard a comment that there were something like 500 engineers working on it, whereas the Niagara design which you bought from Afara had I think a bit over 100 engineers working on it. So, it seems that it leaves you a lot more resources to focus on low-level implementation and getting the most out of your silicon with regards to design because there's a lot of low level techniques you can use to improve the performance of a design regardless of the high-level architecture. For example, using low-voltage transistors and other stuff.

> **Dr. Marc Tremblay:** Yes. That's absolutely correct. And then it's up to the Enterprise to use these extra resources or to do it with fewer resources. The key thing for us, for the path we're on, is much faster turn around time on these designs.

Brian Neal [Ace's Hardware]: I was reading a paper on CELL actually, the IBM/Toshiba/Sony project, and there was actually a comparison between Itanium II and CELL with regards to multiple cores or execution units versus cache space for die size. The conclusion drawn there was it was delivering much higher returns from adding more cores than adding more cache. So I'm wondering what kind of research you done in terms of balancing cache versus cores in a multi-core design.

> **Dr. Marc Tremblay:** So basically that's doing very detailed cycle-accurate simulators and starting with the MAJC base, so we've got like 7-8 years of R&D in that space and running apps. I hate to do this, but I have to defend the Itanium design a little bit in the sense that these two chips will be running drastically different applications. So it will be easy for CELL to claim some pretty outrageous numbers, but it will be hard to exploit this while running TPC-C or whatever.

Brian Neal [Ace's Hardware]: I'm just talking in general, perhaps versus Niagara. Is it better to have 10 MB of cache or is it better to have 10 cores, or does it depend on the application?

> **Dr. Marc Tremblay:** After talking to a lot of people about this for the last couple years, one thing that I find myself having to explain multiple times is that [even] if the size of the cache is reduced because there's just so much real estate on a die, the key thing is memory bandwidth. So, say that per thread the cache size is small, as long as we can get to the memory and sustain high bandwidth, then we're doing useful work. As long as that bandwidth is not always replacing data, and that we haven't increased the miss-rate tremendously, then we do a lot of useful work. So the sizing is definitely less traditional and more skewed to smaller cache per core / per thread than in the past.

Chris Rijk [Ace's Hardware]: One thing that helps with server code, though, is that a fair amount of the data would actually be shared. For a database, there would be a lot of main memory that is just cache for the storage. So because a fair amount of that would be shared, it's not quite as bad as it would necessarily appear to be. But with say something like HPC code, the amount of data would increase proportionally with the number of cores whereas with database code, a certain amount of it would be shared, so it wouldn't be quite so bad.

> **Dr. Marc Tremblay:** Yes. It's pretty astonishing when you walk into a place like a Google, or I used to go to Exodus a lot to see all the server rooms, and you see racks of Lintel boxes, for example. And all

running the same code and a lot of the data could be shared, but there's no sharing because it's all a discrete implementation. Here, as you observe, we have a tremendous amount of sharing and actually, we run a lot faster. Let's say we had 5 cores on a chip and each one had 1 MB of cache, we run a lot faster than if we have 5 discrete cores each with 1 MB of cache by the fact that they're consolidated, the instruction space is not replicated and a lot of data sharing is not replicated either.

Brian Neal [Ace's Hardware]: So you don't have to replicate bandwidth either.

Dr. Marc Tremblay: Yes.

Brian Neal [Ace's Hardware]: So speaking of bandwidth, I'm curious what technologies you're looking at for Niagara, for the future to deliver the kind of bandwidth you need. Things like DDR-II, RDRAM, Yellowstone, etc. What are you guys looking at for high-bandwidth memory interfaces?

Chris Rijk [Ace's Hardware]: Also with regards to having the main memory connection on-die, I think it was said in David Yen's talk that the memory interface was on-die was well as Ethernet.

Dr. Marc Tremblay: Obviously, we're trading off latency for bandwidth so therefore we have to have the latest, greatest DRAM available and by putting the controllers on the chip, we save a lot of latency. So, it's a huge benefit if you don't have to go across chips like in an SMP [system].

Chris Rijk [Ace's Hardware]: It also helps with integrated things like blades as well [in terms of] providing a smaller form factor. It would also help reduce power consumption a bit wouldn't it?

Dr. Marc Tremblay: Yes [and] yes.

Brian Neal [Ace's Hardware]: Well, in MAJC you used RDRAM, which is targeting bandwidth per pin, but I know with UltraSPARC IIIi you're going with DDR. With MAJC you have an advantage were your memory can be scaled one chip at a time versus a more traditional workstation approach with USIIIi. So, could the same thing happen with Niagara, where you want to keep things small embedded and integrated?

Dr. Marc Tremblay: We love the high bandwidth per pin, but the Rambus question is always the business model and how the DRAM manufacturers are supporting it.

Chris Rijk [Ace's Hardware]: You need a lot of memory.

Dr. Marc Tremblay: They do allow scaling, chipkill, and all the traditional things you'd like in a server -- EV7 from DEC is based on Rambus, and that's a large server -- but one thing has always upset us, and even though I went down that path with MAJC, as a processor manufacturer you're creating a demand for the DRAM if you put the controller on chip. And somehow the licensing model [still means] you have to get a license on the processor and pay royalties and the DRAM manufacturer will have to get a license and pay royalties.

Brian Neal [Ace's Hardware]: So both you and the DRAM manufacturer are paying licensing fees?

Dr. Marc Tremblay: Right, while here I am creating the demand for [Rambus].

Chris Rijk [Ace's Hardware]: When you have multiple cores, the actual speed you would have to set for the chip would have to be the slowest core, right?

Dr. Marc Tremblay: Yes, they're all synchronous.

Chris Rijk [Ace's Hardware]: With regards to asynchronous logic, one of the things it sounds like you might do is to isolate each core in a synchronous island. I did some research and found something similar, it was described as locally synchronous / globally asynchronous. So you have individual cores that just run by themselves and talk to the rest of the chip asynchronously. I don't know if in that case you would run them at different speeds, but that would make a great difference as to how complicated it is to design the clock tree and having clock domains with regards to synchronizing cores with external I/O, which is actually quite a problem for the [future].

> **Dr. Marc Tremblay:** Absolutely. We are pursuing both approaches, so locally synchronous, globally synchronous, locally asynchronous, and globally asynchronous. They both have their [usage] in various parts of these large chips.

Chris Rijk [Ace's Hardware]: With regards to cache, it sounds like Niagara is going to have all the cache on die, but for larger high-end chips like the 30x chip, it seems like the better tradeoff is to have a reasonable about of shared L2 cache on die and then having a large external L3 cache. Would you agree with that philosophy?

> **Dr. Marc Tremblay:** A large external cache in many scenarios is very useful, but there are costs associated with it.

Chris Rijk [Ace's Hardware]: The external cache sounds like something for high-end systems only, like the s-series (large SMP systems).

> **Dr. Marc Tremblay:** Yes, it's very useful there.

Chris Rijk [Ace's Hardware]: Do you have plans of using large multi-chip modules (MCMs) or similar?

> **Dr. Marc Tremblay:** We keep pursuing this.

Chris Rijk [Ace's Hardware]: How much of an advantage is being able to do this [processor design] together with the system design and OS design? This is a significant difference to Intel, [for instance] [(i.e. vertical vs. horizontal)].

> **Dr. Marc Tremblay:** I think we're entering an era where that's becoming even more obvious, in the sense that Solaris has been scaled for years to be a multithreaded kernel, to provide lightweight communication and lightweight synchronization. We're adding stuff even at the instruction level in our processors to be able to do these CMPs really well, and it was a one-hour conversation with both the system software guys, the Solaris side, and the JVM guys saying "Oh yeah, we love this, we'll put it in!" I can just imagine [what it must be like] for an Intel, having to convince Microsoft to do this while Hammer's not doing it. So we see this to be a tremendous advantage over here, the agility to enable this technology.

Chris Rijk [Ace's Hardware]: The 15x performance increase for Niagara, is that over the 650 MHz UltraSPARC IIi in the server blades or the UltraSPARC III?

> **Dr. Marc Tremblay:** The first one [(UltraSPARC IIi)].

Chris Rijk [Ace's Hardware]: And the 30x increase was compared to the current UltraSPARC III?

> **Dr. Marc Tremblay:** Yes, 1.2 GHz [(UltraSPARC III)].

Chris Rijk [Ace's Hardware]: So that's actually kind of like 60 times the performance of the UltraSPARC IIi. With regards to power consumption, what would the UltraSPARC V be like compared to things today and then also compared to the 30x chip [(total power consumption)].

> **Dr. Marc Tremblay:** We're approaching things in the sense that we're going to go to somewhat exotic cooling technology, but we're not bringing any water pipes into the systems. That gives us a certain power budget and we stay within that envelope. The turning point is that DRAM is starting to spend more power than the processor anyway. So, we were discussing "How do you size the L2 versus the number of cores?" and the other question is "How do you share the power budget between the DRAMs (the DIMMs) and the processor, and can you with another 10W get higher performance but have to reduce the number of DRAMs and therefore have more I/O?" So that's really a black art.

Chris Rijk [Ace's Hardware]: Was that for the UltraSPARC V only or the USV and the 30x chip?

> **Dr. Marc Tremblay:** I would say they are all within a reasonable power envelope, but these won't be water-cooled chips.

Chris Rijk [Ace's Hardware]: The UltraSPARC V, from previous comments, is said to have two SMT threads. Are you planning to do four at a later date? Four threads per core, [that is].

> **Dr. Marc Tremblay:** We haven't disclosed beyond what you have mentioned.

Chris Rijk [Ace's Hardware]: With regards to the 30x chip, it seems like it's supposed to come quite soon after the UltraSPARC V -- like within two years. It seems like that was [the case] from the graph. Is that correct, approximately?

> **Dr. Marc Tremblay:** Yes.

Brian Neal [Ace's Hardware]: Marc, thanks for talking with us, we really appreciate it.

> **Dr. Marc Tremblay:** I hope that was useful...

Brian Neal [Ace's Hardware]: Thanks for your time.