

The [Radio Register has a nice interview](#) with [Kunle Olukotun](#), the man most known for the Afara/Sun Niagara/UltraSparc T1-2-etc. design. It is a long interview, lasting well over an hour, but it is worth a listen. A particular high point is the story on how Kunle worked on parallel processors in the mid-1990s when everyone else was still chasing single-thread performance. He really was a very early proponent of multicore, and saw it coming a bit before most other (general-purpose) computer architects did. Currently, he is working on how to program multiprocessors, at the [Stanford Pervasive Parallelism Laboratory \(PPL\)](#)

. In the interview, I see several themes that I have blogged about before being reinforced

The [Radio Register has a nice interview](#) with [Kunle Olukotun](#), the man most known for the Afara/Sun Niagara/UltraSparc T1-2-etc. design. It is a long interview, lasting well over an hour, but it is worth a listen. A particular high point is the story on how Kunle worked on parallel processors in the mid-1990s when everyone else was still chasing single-thread performance. He really was a very early proponent of multicore, and saw it coming a bit before most other (general-purpose) computer architects did. Currently, he is working on how to program multiprocessors, at the [Stanford Pervasive Parallelism Laboratory \(PPL\)](#)

. In the interview, I see several themes that I have blogged about before being reinforced

The themes are:

- The way there is to provide programming environments that express algorithms in a way that is clear in terms on intent but that does not constrain the implementation too much. It should hide parallelism to the user, and make that a property of the compiler and implementation, not the program. But to make this work, you need higher-level expressivity. For example, Kunle says that you want to say "do a matrix multiply" rather than "here is a pile of loops and expressions implementing a matrix multiply". Very sensible.
- The proper way to do parallel programming this is to be domain-specific. Create domain-specific languages that map to how domain experts think about a problem, and then have a compiler and runtime take care of how to implement it for a particular machine. Rather than provide a low-level language that lets you express parallelism explicitly, like CUDA, Brook, threading libraries, etc, you should be using MatLab if you are doing science (which is something that National Instruments have been doing with their LabView tools).
- Future hardware will be heterogeneous, with a mix of control-oriented simple cores

## Jakob Engblom: Kunle Olukotun Interview: Heterogeneity, Domain-Specific Programming

Written by Jakob Engblom  
Sunday, 20 July 2008 08:20

---

(Niagara-style), data-processing-oriented cores (DSPs, security accelerators, etc.), and the occasional heavy-weight ILP-oriented core (Intel Core 2-style) for the occasional program that just needs maximal single-thread performance.

I fully agree with all of these, and I find the use of domain-specific languages and frameworks especially important. Software people in all ages have become more efficient by designing better languages, more suited for a particular task than a very general language. A language is a just a tool, after all, not a religion (even if some people seem to view it that way), and should be changed depending on the task at hand.

Final note for us embedded folks: when Kunle talks about how he realized around 1995 that lots of simple processors on a chip becoming feasible thanks to Moore's law, the first real multicore chips were already shipping. Motorola had the QUICC 68000+CPM heterogeneous network processors on the market by then, and Texas Instruments had the C80 four-simple-DSPs-plus-a-simple-RISC multicore chip for digital video also out. But the [ISCA](#) crowd really did not notice this development at all, at the time.

Obscure note two: Virtutech Simics was actually used by Afara to help in the design work of the Niagara, since Simics had very good support for 64-bit SPARC architectures thanks to the early work done with Sun.

**Read the original article:** <http://jakob.engbloms.se/archives/157>