

INTRODUCTION

Something was different at Hot Chips this year, significantly different. This year Hot Chips had sessions on parallel computing. One of the speakers called multi-core computing the biggest inflection point in computation since the 1940s, just to emphasize how important this is. The IC industry's shift from single processor computing to multi-core has moved us away from the Von Neumann compute model to a world where the compute-model has yet to be defined. In the process it has endangered the progress of Moore's Law.

THE RESEARCH

Several Research centers have been established to address this crisis. University of Illinois, University of California at Berkley and Stanford University were represented at Hot Chips this year. All three have taken a different approach to the problem and as a result have come up with different observations. In the last two years it has become apparent that an Applications approach is necessary. Applications specific programming isn't the answer but Applications will point the way.

GSEDA'S RESEARCH

GSEDA has been following multi-core computing and parallel programming since 2004. During the first three years most of the research came up with no workable solutions. Two years ago the Applications approach allowed the research to start making headway. Work being done at IBM, UC Berkley, University of Illinois and NVIDIA all started coming up with answers to pieces of the puzzle. For awhile it was looking like the old Hindu story of the five blind men and the elephant.

CLOSE ENOUGH TO DRAW ANY CONCLUSIONS ?

The question is do we have enough information to get a good idea what the picture in that puzzle will look like. GSEDA thinks we have, or at least we would like to take a shot at it.

First we need to explore a piece of microprocessor history that no one seems to be talking about. If you look at the first microprocessors you will find that they don't look much like the processors of today. In fact they looked more like microcontrollers than what we now call microprocessors. The first real microprocessors came into being with the introduction of Signetics 2650, followed by Intel's 8080 and Motorola's 6800.

At that point the industry started calling some processors microcontrollers and some processors microprocessors. So you can see that early in the development of processors the concept of domain-specific processors was part of life. Following that there were some network processors and some multimedia processors, but the next important processor was NVIDIA's Graphics Processor. So that's where we stand today, with three domain specific processors or more accurately two domain-specific processors and a general purpose processor.

THE TWELVE DWARFS

This takes us to the research going on at Berkley. If you look at the NVIDIA Graphics processor you will find that it is solving problems in multiple applications; gaming, oil exploration, medical, military, EDA and financial. If you look at their work in EDA you can see they can give us significant performance upgrade in some tools but with the basic graph based tools they don't help much. So that leads us to say that applications point to the answer but are not the answer; domains, or dwarfs and Berkeley calls them (http://view.eecs.berkeley.edu/wiki/Dwarf_Mine) appear to be the answer. (Oops, looks like we have a baker's dozen now.)

1. **Dense Linear Algebra**
2. **Sparse Linear Algebra**
3. **Spectral Methods**
4. **N-body Methods**
5. **Structured Grids**
6. **Unstructured Grids**
7. **Mapreduce**
8. **Combinational Logic**
9. **Graph Traversal**
10. **Dynamic Programming**
11. **Backtrack And Branch-and-bound**
12. **Graphical Models**
13. **Finite State Machines**

It isn't probable that we will have thirteen standard domain specific processors. Some won't be that much more efficient than today's General Purpose microprocessor performance. Some of the Dwarfs may be combined. And others won't be commercially viable as a standard product. We probably will end up with four to six high-volume domain-specific standard product processors. Coming up with a parallel processing capable microcontroller will be a challenge but I think we will do it.

IT'S THE SOFTWARE STUPID

Yes it is the software, but trying to solve the software problem without knowing what compute-engine you'll be using isn't going to work. Much of the work going on today is targeting multi-core versions of today's general purpose processor, possibly the most inefficient processor configuration we can target. Thinking that we are going to fix the parallel processing problem in a general way is a waste of time. Today's power budgets demand that we use the most efficient processing engine available. Our many-core processing environments will be heterogeneous. Will general purpose processors survive; of course. Will they be in the forefront of computing; not likely.

That brings us to languages. The claim that C will be the major language of the future is getting a little stale. The C community is already complaining that C is no longer being taught in most colleges. Will there be a language for each domain? Although that would be the optimum choice it probably won't work, at least not in a heterogeneous compute environment. What we will probably see is a C++ type structure where the language has class libraries for each domain. This could happen under C initially but C will eventually be replaced by a language more in tune with the future of computing.

CONCLUSION

We have a ways to go before the tools we need to solve the parallel processing problem are in place. We think that NVIDIA has shown us that this is a problem that demands a combined Hardware/Software solution. Today's research is pointing the way to the Future of Computing.