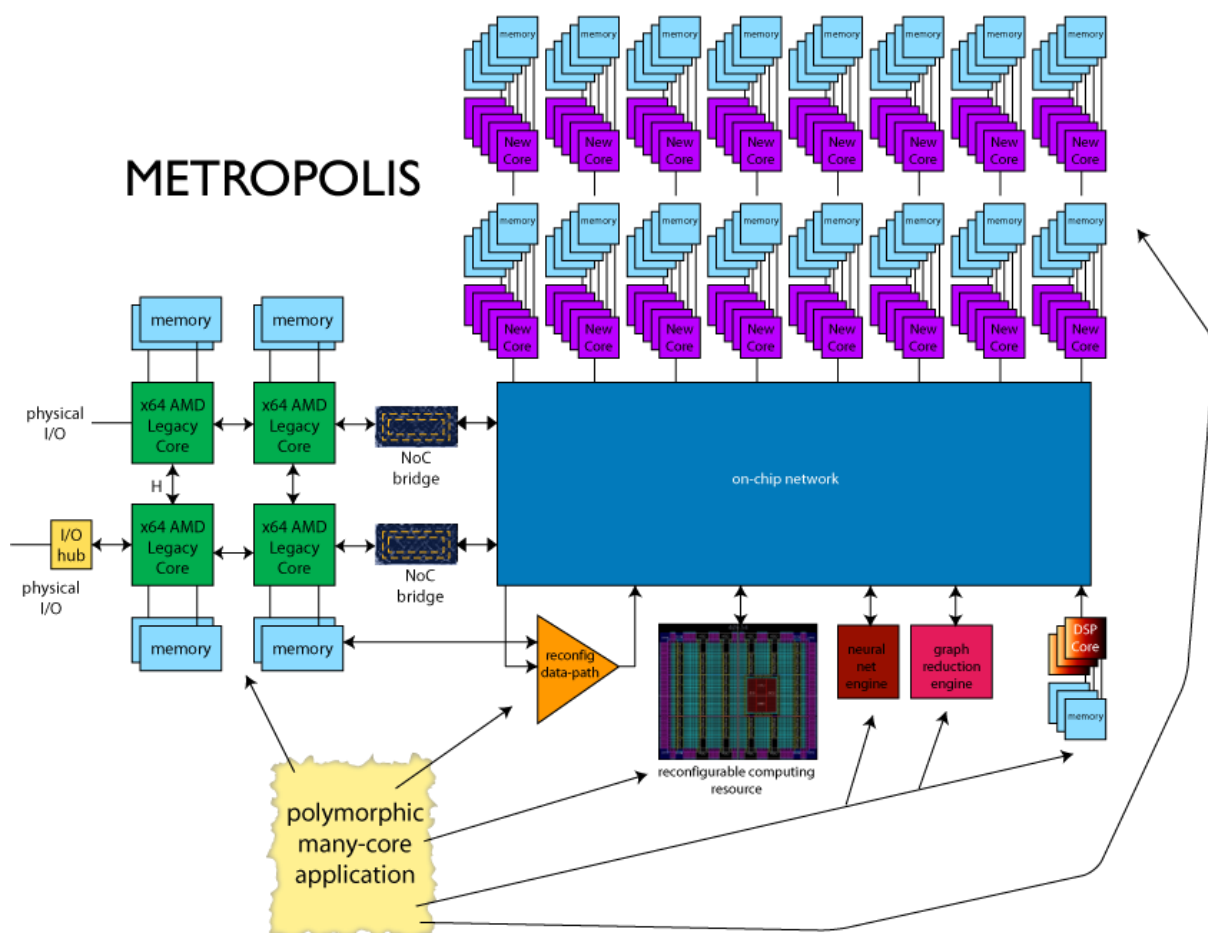


# Many-core Architecture and Programming Challenges

*Satnam Singh, Microsoft Research, Cambridge*

I argue that future many-core processor architectures will be necessarily heterogeneous and pose even more significant programming challenges than we are facing today with multi-core systems. These many-core architectures pose important research problems varying from how to model, design, evaluate and verify many-core hardware to how to model, write, and verify software that has to execute on a variety of processing elements. An example of a many-core architecture may include a small number of legacy x86 processors, many more simpler processors (perhaps with varying micro-architectures and instruction sets), configurable data-paths which are the evolution of today's GPUs, reconfigurable logic to add custom co-processing in hardware or for instruction set extensions and dynamically configurable memory architectures that vary to suit a particular application. It is necessary to take an interdisciplinary approach to tackling the issues of many-core systems by drawing together expertise in architecture, languages, verification and modelling. A mock many-core system called Metropolis is shown in the figure below.



I think it is important for academic researchers to collaborate with industrial researchers to make progress in this area. It is very important to have effective workloads for architecture evaluation and

these workloads have often been lacking in computer architecture research done at universities. Through academic and industrial collaboration we may be able to get representative and substantial commercial workloads and use them to evaluate proposed many-core architectures.

What can be done now to help us prepare for a many-core future? Here are a few suggestions.

- Current multi-core architectures are putting pressure on us to write **concurrent software** and this trend will only get stronger with many-core. Devising techniques which make it easier to write concurrent applications in a composable way is now an important research challenge. My work on developing workload applications for software transactional memory is a step in the direction of developing a shared memory concurrency mechanism to support composable software construction without explicit locks.
- A key technique for exploiting the parallel processing elements in a many-core architecture will be **data-parallelism**. In particular, we need to devise language level techniques to support nested data-parallelism which arises naturally when one wants to compose libraries of data-parallel operations. Simon Peyton-Jones and others are adding nested-data parallelism to the Haskell compiler and I hope to use this technology to develop libraries of common data-parallel operations. Researching data-parallel programming techniques is an obvious area of collaboration with Chalmers.
- **System level modelling** techniques will need to be devised to help describe and analyze proposed models at a high level of abstraction for the purpose of architecture evaluation. I believe there is a lot of potential to exploit functional languages as “executable specifications” that serve as effective architecture modelling languages. I have served as the PC chair of the system level modelling track at DATE for three years now and by looking at the papers submitted I believe techniques based on UML and System-C are either too weak or overly complicated to meet the needs of system level modeling for many-core.
- **Portable programming techniques** will need to be developed to allow a calculation designed to run on one device (e.g. a CPU) to migrate to another device (e.g. a GPU). Although in general it is probably too hard a problem to translate an arbitrary binary for a processor to run on a data-path oriented architecture I do believe that one can cast calculations in domain specific languages in a way which captures just enough information to allow a calculation to be instantiated on a variety of computing devices. I’ve already experimented with data-parallel descriptions in the Lava system (an embedded domain specific language for hardware developed in conjunction with Chalmers) by compiling the same parallel sorter descriptions to FPGA hardware, GPU code and to multi-threaded code.
- **Verification** will become increasingly important especially due to the interaction between many different kinds of computing elements through a variety of communication mechanisms e.g. shared memory, message-passing, FIFOs, interrupts etc. Much of the verification work at Chalmers is directly relevant for such verification.
- **Applications and workloads.** To be successful it is essential to have effective workloads drawn from legacy systems and projected future requirements. Workloads is an important area in which academic and industrial collaboration is essential.