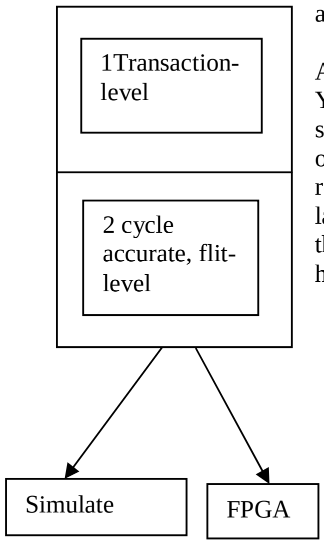
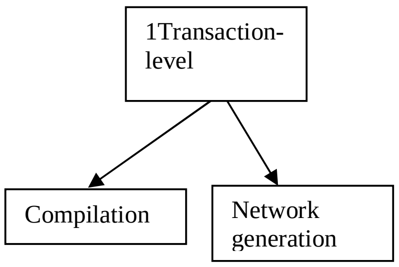
Simulation flow

Premadona meeting 9-9-2005



We tend to distinguish three levels of simulation. Note that depending on the available man power in Premadona, we can probably not have all levels available at TUE.

At the highest level (transaction) we have available POOSL en YAPI/HAPI. POOSL allows statistical analysis en co-simulation of streaming and control. It doesn't have a C-entry, so functional verification of C-code is not possible. HAPI has a C-entry. For both options numbers are required with respect to execution time on the computational elements and latency/throughput of the communication channels. In order to generate these numbers a compilation/profiling step and a network generation step has to be performed.



Because of the relatively small simulation times at this level, it is probably most suitable for DSE, both wrt mapping and architectural exploration. We distinguish two forms of exploration wrt architecture: the generation of (pareto) points on parameters like the number of processors, the bandwidth made available for a channel, the buffer size in a network interface, etc. The other type is more conceptual; what kind of arbitration mechanism is applied in router, network interface, memory access, and processor? Do we support preemption? What routing algorithm is used? The token buffer memory in the processing tiles; does it consist of fixed-size fifos, or are buffers dynamically configured? Do we use point-to-point connections besides an aethereal network? What does a run-time resource manager look like? Can we find SDF models of our architectural concepts?
For the first type of exploration it is convenient to have an automated mapping with a machine description file (MDF) and a mapping file. For the latter type of exploration it is convenient to have a high-level flexible environment that allows to incorporate user-defined models of architectural components. Note that these kind of architectural features always require a low-level simulation in the end, in order to verify the implementation.
The CASSE simulation environment from Victor Reyes does work with an MDF, so that would allow architectural exploration on parameters, but a model of a network is not incorporated. It is however, sufficiently flexible to allow incorporation of user-defined component models.

The second level is cycle-accurate. Also the aethereal flit-level simulator is at this level. Two portugese masters students have built a simulator at this level in the Hijdra project. The platform is uniform SWArm (Software Arm). I will go talk with them.

The lowest level is RT-level, like system-C (or Hendel-C ?). Because of extensive run-times, this is only suitable for verification of components. Alternatively, an FPGA implementation is a lot faster. An estimation is testing 1 implementation in an hour. This is not sufficient for exploration on parameters, but it might be good enough for exploring concepts.

For a Ph.D. (architecture) topic for Akash I see two possibilities: High-level architecture exploration (parameters, but I prefer concepts). This is a deviation from his current activity, whereas his FPGA activity is unique for TUE and Philips. The alternative is to explore concepts (not parameters) in FPGA directly with the advantage of providing detailed verification of an architectural concept in the context of the completely implemented system. A potential disadvantage is the amount of work to implement a number of applications. We preferably don't want to restrict ourselves to a single application because we might tune our architectural concepts to a single application, and because we also would like to investigate resource arbitration and linked to that, an on-chip resource manager (HW or SW) that can handle multiple applications. This is still possible with a single application if we duplicate jobs, eg multiple rectangles or multiple objects for a video-graphics decoder, or multiple MPEG4 streams for the shape-texture decoder from Milan. One interesting issue that can be explored (performance & flexibility vs.cost) after an application is implemented in FPGA is resource arbitration; priority-based, TDMA, credit based, ordered-transaction (Sriram) or a hybrid. Are some arbitration schemes better for arbitrating processors, memory accesses, or the communication network? Can these mechanisms be conveniently controlled by a resource manager? At what order of time intervals should this be performed? Can we prevent application artifacts occurring during system reconfiguration (entering or exiting of a job, or adaptability to changing computation requirements)?

On the topic of these architectural concepts there is potential overlap with the work of Arno, but I think there are plenty issues to be explored in that area.