Agenda Internal Kick-Off Meeting

- Determine date + agenda official PreMaDoNa kick-off meeting (+ Invitees + Presentations)
- Discussion on the different tracks based on the living PreMaDona document
- Allocation of people to tasks
- Weekly meetings
- Status vacancies
- Web-site
- Relation with Progress
- Other external links

Action Points and Discussion Points

General action points:
- Website (BartT, ongoing)

Track 1 (Architecture, implementation):

The current proposal emphasizes the demonstrator.
This demonstrator is an implementation containing:
- A Risc processor (Arm or Mips)
- An accellerator
- A VLIW processor
- I/O peripherals
- Level 2 memory (scratch pad or cache)
- A communication network where Aethereal components are reused as much as possible (routers and network interfaces)

Operational issues:
We plan a simple FPGA demonstrator in two years. It should be simple because FPGA delivers performance and area 1 order of magnitude worse than silicon. One problem is that the reusability of Aethereal components for FPGA is low because they use special cells in the buffer. On the other hand, the new Aethereal routers and network interfaces are quite simple, which eases reusability.
We also plan a more extensive demonstrator in four years that also embodies research results from Premadona. This demonstrator can be FPGA or silicon (layout). Silicon is preferred but implies certain riscs. For real silicon we probably require a partner to cover the huge mask costs.
The Aethereal tools generate verilog for the network components based on the requested bandwidth and traffic type per connection. This implies a high degree of reusability.
Availability of these tools is not obvious and may contain some political issues.

Research issues:
- Often it is more convenient to have direct wires between two processors. So we consider architectures with both router-based and wire-based communication. For the two-year demonstrator this is probably not possible because we have to stick to a fixed template for an implementation on such a short time scale.
- Can we physically map a non-homogeneous NW on chip? One problem is that the delays between tiles is different, which can lead to problems with timing closure in the physical layout.
- How do we enforce and exploit hierarchy in the layout?

- What kind of tile structure is convenient for physical design? Squares are convenient because they imply a very regular structure, but the tile size would be the maximum required tile size for all tiles (unless a tile can be easily split up in two), causing a potentially very low usage of tile area. Rectangles are more flexible and yield higher usage, but are also more difficult to combine, and may cause unused silicon area between the tiles.
- Can we conveniently exploit information regarding the amount of communication between two tiles (large bandwidth implies putting the tiles close together) to get better time delayus and power numbers.
- Can we conveniently trade-off power and delay, eg by partial memory switch-off? Where is most of the power being consumed?

Action points:

- Write an add for the position (Bartm, done)
- See what is available for reuse (Jef?)

Track 2 (methods for predictable design, mapping tools):

The proposal comprises a design flow as sketched in **Figure 1**. We would like to take C as entry point, because that is the de-facto industry standard. We assume in this flow that a platform is given (as a machine description), although this may be open for discussion. Note that not all input is streaming. First we want to make the task-level parallelism (TLP) explicit. This can be done in YAPI or POOSL. Each task is then assigned to a processor and compiled. After compilation we have the timing delay associated with a task (for static tasks) or with an instruction (for dynamic tasks). Subsequently we follow an analysis path (left hand) and a simulation path (right hand). The analysis path comprises a tool (Facts or Heracles) that computes guaranteed upper bounds for a self-timed schedule, as well as the corresponding buffer sizes. If possible we also want to analyse power consumption. The simulation path comprises a high-level simulator (HAPI or POOSL) with timing that verifies the analysis results for static jobs, and also gives timing and buffering numbers for dynamic jobs. Furthermore, the simulator should be able to say something about quality in the case of soft RT jobs.

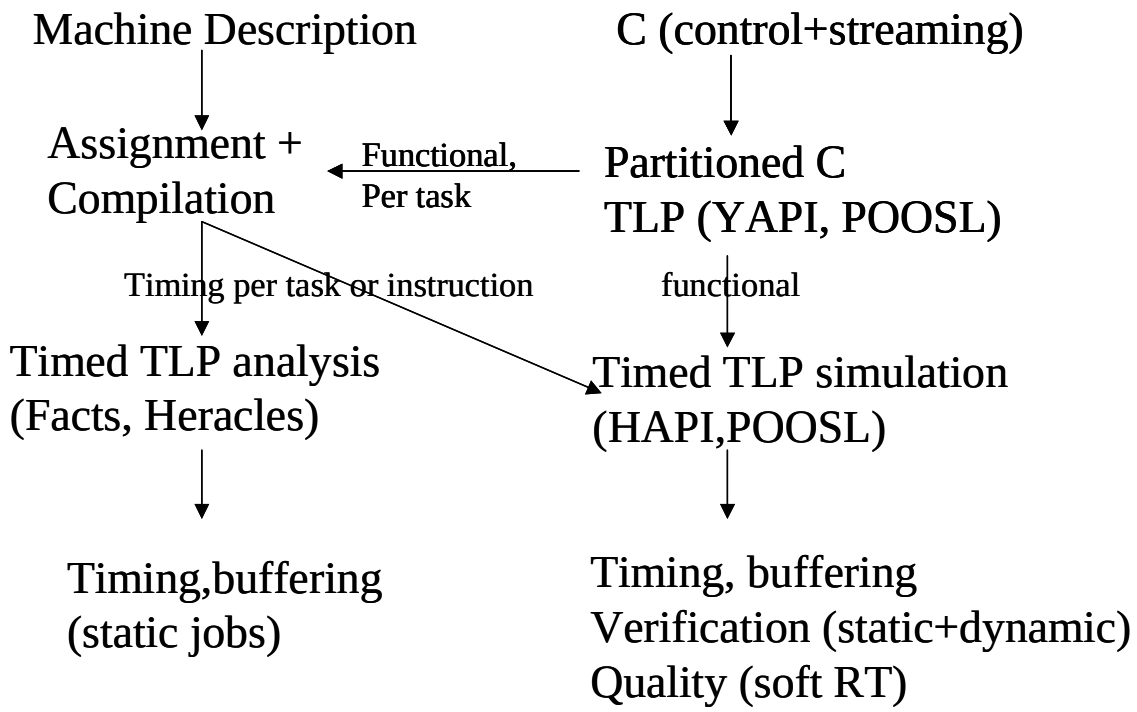

Figure 1 PreMaDoNA Design flow (tentative)

Operational issues:

- Mapping using a machine description file (MDF).
- Interfacing between assignment/compilation and analysis tools
- An important issue for automating the design flow is the step from YAPI to HAPI. Sander thinks this step can be automated (but is difficult) is possible if the designer makes explicit how many tokens are produced/consumed.
- In HAPI we can only model guaranteed throughput (GT) channels.

Research issues:

- Design-space exploration (DSE) for timing, buffering, and energy. In order to explore different solutions, we distinguish the cost for computation and communication, and we distinguish static and dynamic jobs. The computation cost for each static job can be derived by compiling that job for each 'suitable' processor in the architecture. Note that a homogenous platform significantly reduces the compilation effort. The computation cost for a dynamic job is data dependent, and can be derived in a statistical manner using simulation (the right-hand path in **Figure 1**) and/or using modes or scenarios. Furthermore, the timing and buffering cost depends on the amount of resource sharing and arbitration schemes. The communication cost depends on the length of the physical path through the router network. This path depends on the amount of resource sharing in the routing network, and therefore also has a dynamic nature. An important issue is the potential for viewing the communication cost individually as virtual platform. In all cases fragmentation will occur in a dynamic environment both for computation and communication.
- Deriving upper bounds for the cost factors in the analysis tool.
- (RE-)configuration in a dynamic environment.

Action points:

- Converge to a design flow (BartT, BartM, Marc, Sander, first draft in this document)
- See what is available for reuse (BartT, BartM, Sander, ongoing)

Track 3 (Application development with QoS and mapping that application as main case study):

In this track an application will be developed with a quality of service (QoS) layer. Furthermore, the application will be mapped on the multi-processor network. The application is part of the MPEG4 decoder standard. This track is supervised by Peter de With. Milan Pastrnak will also play a supervising role, because he already has developed the MPEG4 arbitrary shape decoder in quite some detail. Current topics for Milan are the decoding of multiple objects, growing objects, dynamic execution times, and modelling the application in Hapi. In the flagship project the encoder will also be considered and the application will be extended with some pixel processing at the decoder side.

Operational issues:

- Currently, QoS is limited to frame skipping. This will be extended with QoS methods from research.
- Modelling the application for the tools in **Figure 1**.

Research issues:

- Quality management
- Perceptive quality: eg, which objects can be skipped
- QoS for the several subtasks

External track:

Philips Nijmegen, Business line Car Audio Infotainment Systems develops multi-processor platforms (using a DSP from Philips called Epics) called sea of Epics. Arno Moonen is a PhD student working alternately in Nijmegen and Research, and one day a week at TUE. He investigates architectural trends and bottlenecks in earlier generations (HW and SW). (Re-)configuration is a main research topic.