PROPHID: A Data-Driven Multi-Processor Architecture for High-Performance DSP

Jeroen A.J. Leijten[1,2], Jef L. van Meerbergen[1], Adwin H. Timmer[1], and Jochen A.G. Jess[2]

[1] Philips Research Laboratories, WAY4, Prof. Holstlaan 4, 5656 AA Eindhoven, The Netherlands
[2]Design Automation Section, Department of Electrical Engineering, Eindhoven University of Technology, The Netherlands

PROPHID is a design method for high-performance systems with a focus on high-throughput signal processing applications. It makes use of a novel stream-based multi-processor architecture, consisting of data-driven autonomous processors interconnected by a programmable connection network. The key element is the communication arbiter, which controls the flow of data between processors. Variable rates and data-dependent processing times are handled efficiently by performing scheduling at run time. We give an overview of the characteristics and advantages of the architecture as well as some implementation results.

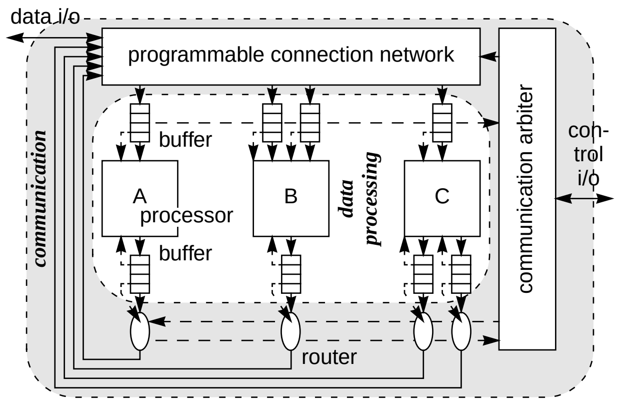
The computing and bandwidth requirements of large real-time applications such as video compression and decompression, graphics, and speech recognition implemented on a single chip are very demanding. Furthermore, these applications often contain algorithms that tend to be *dynamic*, with data-dependent processing times and variable data rates. Another important aspect of these applications is the demand for *programmability.*

The high performance requirements lead to *task-level parallelism*, exploited in multi-processor architectures. The *dynamic nature* results in poor efficiency of compile-time schedules that are aimed at worst-case behavior. Therefore, *run-time scheduling* techniques aimed at *average-case* behavior must be used. Finding a good balance between *programmability*, performance and cost-efficiency leads to multi-processor systems with application domain specific processors that are each tuned towards a specific type of function. The key issue is to keep the overhead of interprocessor *communication* and *synchronization* small.

Our applications require in the order of 20 processors delivering 1 GOPS each, with a total interprocessor communication bandwidth of 30 to 40 Gbit/sec. With these requirements in mind we developed the PROPHID multi-processor target architecture template. This template can be customized to many different applications. A varying number of processors of any type ranging from hard-wired data paths to fully programmable processors can be used. Therefore, programmability at the processor level is supported. Processors are data-driven and communicate via *streams* that are split up into labelled *data packets*. Via the data implicit synchronization of parallel tasks is obtained. All interprocessor communication is handled by a *reconfigurable communication network* to which the processors are connected by means of buffers. This network provides programmability at the interprocessor level. Thus, processing is performed in parallel and is controlled at a *local level* by each autonomous processor, while communication is performed and controlled at a *global level* by the communication network.

By separating data processing from data communication and performing these tasks in parallel, reduction of communication overhead is possible. However, to ensure that the interprocessor communication does not lead to a bottleneck in the design a fast and simple communication mechanism is crucial. The key element of this mechanism is the *communication arbiter* that determines at run time which packets can be transmitted to their destination. This allows for an efficient execution of dynamic data flow tasks, while optimizing throughput for average-case behavior. The user can influence the decisions made by the arbiter by imposing schedule constraints. These constraints, together with a task-graph representation of the application, and information regarding the binding of task nodes to processors and graph edges to buffers, form the program of the arbiter.

We have implemented a communication arbiter that requires a program memory of less than 1 kilobyte for task graphs containing up to 256 edges. It needs 4 clock cycles to acknowledge a data packet transfer. Therefore, even for relatively small packet sizes arbitration overhead is negligible. A VHDL description of the arbiter was synthesized with Cadence's Synergy using standard cells in a 3-metal layer, 0.5-micron process. This resulted in an area of about 4 mm^2 with the arbiter running at 64-MHz clock frequency. Therefore, only 63 ns are needed to acknowledge a request. These results show that a fast arbiter with negligible arbitration overhead can be created at low area costs.



The authors thank Yann Bonhomme, Fons de Lange, and Peter de With for their contributions.