Submitted, rejected and still posted online by the Wild-and-Crazy Session at ASPLOS 1998

RDF: Reconfigurable Data-Flow

Oskar Mencer, Michael J. Flynn, Martin Morf
{oskar, flynn,morf}@umunhum.stanford.edu
Computer Systems Laboratory, Stanford, CA 94305

Problem:

General-purpose microprocessors are to slow for top-of-the-line multimedia applications. As microprocessors become more powerful, the performance requirements of stream-based multimedia applications grow as well.

We observe that the amount of data that has to be processed by a specific dataflow-graph grows; e.g. display resolution increases, computer animations become more realistic, etc. At the same time available silicon area on one chip grows exponentially due to improvements in process technology.

Proposed Solution:

A static dataflow-centered architecture using off-the-shelf programmable logic (PL) chips; e.g. SRAM-based Field-Programmable Gate Arrays (FPGAs), configurable datapaths, etc. The proposed solution targets data-intensive, latency tolerant applications such as multimedia. Large numbers of physically small, high-throughput arithmetic units with a large selection of different number representations (e.g. various logarithmic, rational, redundant, etc.) and FIFO-style delay elements are used as building blocks for entire data-flow graphs. From the architectural perspective, the distributed delay-FIFOs implement a variable-size distributed register file. The distributed register blocks ensure that the architecture scales well to very large dataflow-graphs.

Data streams through the pipeline with relatively long latency and *high throughput*. Control flow is implemented by executing all branches in parallel and selecting the right result; increasing the latency tolerance of the architecture. Fast bit-parallel reconfiguration creates the impression of Ultra-Large-Instruction-Word (ULIW) execution, where 1 configuration = 1 ULIW instruction.

Digit-serial methods can be employed to tailor the size of the dataflow-graphs to the physical limitations of the reconfigurable devices. Given the pace of advancement in VLSI technology, it will be feasible to compile entire commercial applications to one static configuration of a PL-chip.

Our current successes with FPGAs at 33MHz include 0.5 Gbits/s *low-power* IDEA (International Data Encryption Algorithm) encryption and 10X speedup of boolean satisfiability (SAT) solvers for sufficiently large data sets. Using the CORDIC algorithm, a single 300 US$ chip achieves 200 million 2D-16 bit vector rotations per second.

Serious Crazy Idea: Large Scale RDF

Compiling large data-flow graphs onto multiple PL chips requires partitioning, which has been shown to be a computationally hard problem. In addition, the resulting designs are highly inefficient in terms of area utilization and clock speed.

Instead we design a supercomputer system where all PL-chips in one node have the same configuration - computing *UMA-style* on a data-stream from distributed memory, e.g. encryption: each PL-chip encrypts one block of data. This organization enables us to exploit highly parallelizable AND highly pipelinable applications.

Each node has one communication processor (CP) which handles local memory streams to FPGAs and message passing style communication with other nodes. For example: 256 nodes with 16 FPGAs per node result in >100 TOPS peak performance. Using a combination of rational and logarithmic number represenation could result in >100 Tera-FLOPS peak performance. Neglecting memory system issues, a large scale RDF could reach *1Peta-FLOP* by the time we reach 0.1 micron technology. Such a computing system would be well suited for supercomputer customers requiring absolute top performance for applications that can be mostly statically scheduled.

some issues to be resolved:

- Currently, the communication processor is programmed separately in the runtime environment.
  Ideally, a dataflow-graph compiler schedules memory accesses automatically.
- PL architectures are not well suited for reconfigurable computing.
- Programmability is another major unresolved issue.