Application Development for Hybrid Pipelined Systems

Mark A. Franklin*, Patrick Crowley*, Roger D. Chamberlain*, Jeremy Buhler*, and James H. Buckley†

*Department of Computer Science and Engineering
†Department of Physics
Washington University in St. Louis
Compute Resource #1: General-Purpose Processors
Compute Resource #2: Chip Multiprocessors

Examples include:
- Intel IXP
- ClearSpeed CSX600
- Tensilica Xtensa
- IBM Cell processor
Compute Resource #3: Reconfigurable Hardware

- Field Programmable Gate Arrays (FPGAs) provide custom logic function capability
- Operate at hardware speeds
- Can be altered (reconfigured) in the field to meet specific application needs
Applications

Our applications of interest:
• Are computationally expensive
• Have large data volumes to process
• Typically are pipelined in structure
• Are constrained in
  ➢ time and/or
  ➢ resources
• To be deployed on hybrid architectures
VERITAS Telescope

- Array of 12 m telescopes being constructed in Arizona.
- Looking for Cherenkov radiation from 50 GeV to 50 TeV gamma-ray interactions with upper atmosphere.
- Early indicator of supernovae, so timely data analysis is central to scientific mission.
VERITAS Signal Processing

Smooth filtering
matched filter
Parameter extraction
Threshold check

FFT → LPF → MF → IFFT → Parm Ext → Test

... → ... → ... → ... → ... → ... → ... → Compute Moments → Image Params

Pixel processing

Washington University in St. Louis
Biosequence Similarity Search

*BIG* Genome Databases

DNA pattern

DNA sequence

Match?
BLAST Biosequence Alignment

As we move down the pipeline:
• Data volume shrinks
• Compute requirements grow
How do we map the application to the computational resources?
Related questions

• What algorithmic pipeline stages are appropriate (i.e., what are the candidate decompositions)?
• What computational resources are available?
• What performance constraints must be met?
• Two possible optimization problems:
  – Given resource constraint, maximize performance
  – Given performance constraint, minimize resources
• In all of above, understanding performance is critical to good design
Given (What this talk is not about)

Native implementation of modules for each compute resource

```
fft () {
    while(1) {
        rcve(array time[N]);
        perform_fft(time,freq);
        send(array freq[N]);
    }
}
```

```
entity fft is
    time: in std_logic_vector(...);
    freq: out std_logic_vector(...);
end fft;

architecture foo of fft is
begin
    ...
end foo;
```
Module design constraints

Module designs must conform to common interface specification

```vhdl
entity fft is
  time: in std_logic_vector(...);
  freq: out std_logic_vector(...);
end fft;

architecture foo of fft is
begin
  ...  
end foo;
```
Specify Processing Topology

- Generate formal interconnect specification
- MPI-like semantics between modules
Design Tool

• Inputs:
  – Individual module designs
  – Interconnect specification
  – Mapping of modules to compute resources

• Outputs:
  – Software-only, pipelined “golden model”
  – Instrumented functional RTL simulation
  – Parameterizable discrete-event simulation model
  – Deployable implementation
Software Golden Model

Also usable as a software pipelined implementation, if multiple general-purpose processors available.
Collect performance data on:
- compute times
- communications volume
Discrete-Event Simulation Model

Parameterized from RTL-level simulation and/or deployed implementation.

Washington University in St. Louis
Deploy on Actual Hardware

Supported platforms:
- Exegy K•Appliance
- SGI Altix w/FPGA

- Intel IXP network proc.
- Traditional workstations
Summary

• We are building an application development environment that:
  – Supports investigation of many application-to-hardware mappings
  – Develops models and deployable application from common specification
  – Handles unique needs of hybrid systems
• Focus is on performance evaluation early in the design cycle