6.189 Multicore Programming Primer, January (IAP) 2007

6.189 IAP 2007

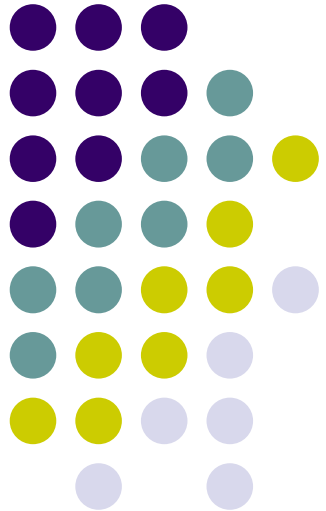
Student Project Presentation

Software Radio

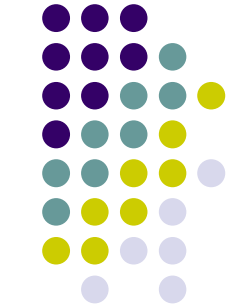
Flexible Stream Processing On the Cell

Case Study: Software Radio

Arvind Thiagarajan and Micah Brodsky



Motivation

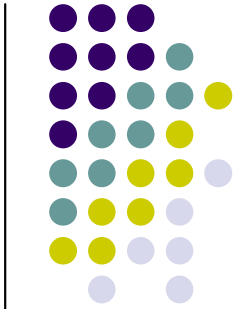


- Cell isn’t easy to program
  - No shared mem, messy msg passing
- Extracting parallelism is nontrivial
  - E.g., pipelining can be quite tricky
- Stream programming (as discussed) can help address both issues

What We Built

- Lightweight, but expressive streaming framework targeted at DSP apps
  - Data model based on WaveScope streaming DBMS
- Case study:
  - Simple Software Radio (Incoherent ASK)
- Main Goals:
  - Simplify life for developers
  - Automate as much parallelism as possible

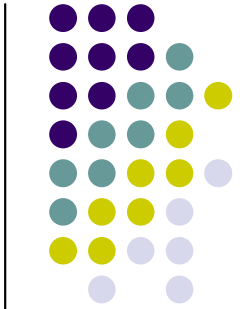
Programming Model



- Basic execution unit is the “operator”
  - Analogous to StreamIt work fn, or GNURadio block
- Can be arbitrary C++ classes, with state
  - Overload iterate() to process block of data
- Apps built by chaining operators:

```
CREATE_BOX(FIRFilter<float>, filter1, args…)
CREATE_BOX(WhiteNoiseGen, noisegen, args…)
CONNECT(filter1, noisegen)
....
```

Framework Components



- Lightweight Scheduler on PPE and SPEs
  - Static operator mapping to SPEs, but easy to extend
- Signal Blocks (adapted from WaveScope)
  - Ref counting, avoid in-memory copies
  - Convenient API, with “append” and “subseg”
- Queue, and remote heap mgmt library for Cell
  - Automatic pipelining for streaming, SPE-SPE
  - Autonomous memory mgmt (not PPE controlled)

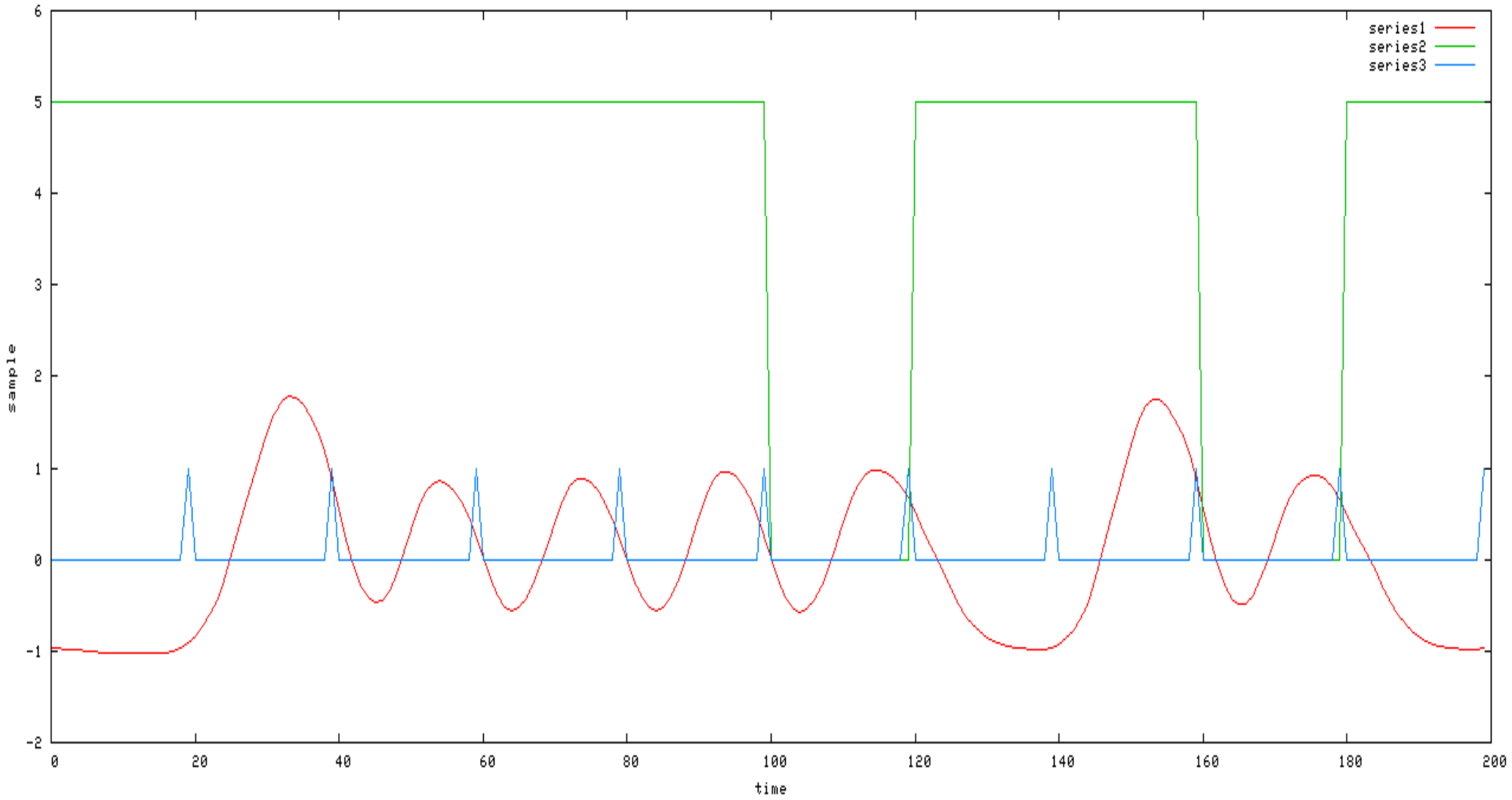
S/W Radio Implementation

- Simple prototype to evaluate framework
- 25 Operators, mapped to PPE + 5 SPEs
- ~3K lines of code (2K framework, 1K radio)

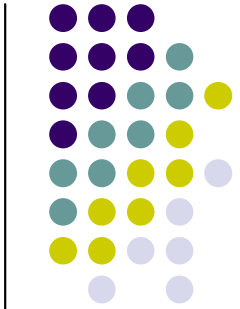
S/W Radio (Contd.)

- Simulated Channel
  - Random FIR Filter (emulate multipath)
  - Additive Gaussian white noise
- Simple ASK modulation
- Incoherent demodulation (quick and dirty)

Example Decoded Waveform



Challenges



- Distributed, almost zero-copy objects
- Lock-free remote heap for streaming data
- Low code footprint on SPE
- Efficient scheduling, SPE-SPE flow control
- Race conditions and memory corruption
  - Not completely solved yet ☹

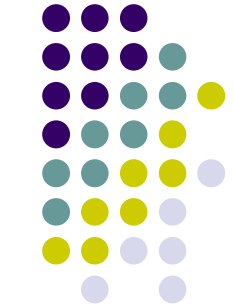
Prelim Results (S/W Radio)

# of Processors Used	Throughput (-O2) (x1000 samples/sec)
1 (Only PPE)	~ 170
6 (1 PPE + 5 SPEs)	~ 640

Speedup with max #SPEs ~ 4

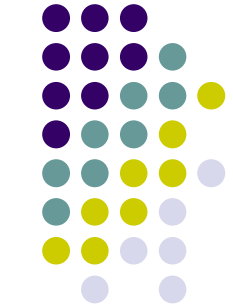
Code footprint of framework ~ 75K

Issues and Bottlenecks



- Flow control not completely resolved
  - PPE spends 50% of its time blocked for SPE queues to drain
- Code footprint needs further reduction
  - Restricts queue sizes, worsens flow problem

Future Work



- Reduce code footprint
- Use framework to investigate dynamic/static operator → SPE assignment algorithms
- Automatic data parallelism
  - Run same op in parallel
- Build more apps for Cell using framework

Project Summary

- Dynamic, flexible streaming framework
- Convenient for DSP apps
  - Block passing abstraction
- Reasonably scalable (Pipeline parallelism)
- Lots of work remains…