Adapting a SDR environment to GPU architectures

06/22/2011 - 06/24/2011
SDR’11 - WinnComm - Europe

Pierre-Henri Horrein
Frédéric Pétrot (TIMA)
Christine Hennebert
Contents

1. Context and aim
2. Approaches
3. Results
4. Conclusion
Outline

1. Context and aim
2. Approaches
3. Results
4. Conclusion
OpenCL architecture

- Centralized management on host
- SIMD architecture: same kernels applied on large vectors
GNURadio

Context and aim

Approaches

Results

Conclusion

- **SDR framework**

- **Provides:**
  - a large set of SDR basic operations
  - runtime management of the operations
  - I/O integration (Ettus Research, audio, ...)

![Diagram of SDR framework]

IQ Samples

Task 0 → Task 1 → Task 2 → Task 3 → Applications

Task 4 → Task 5
Aim

Context and aim

Approaches

Results

Conclusion
Outline

1. Context and aim
2. Approaches
3. Results
4. Conclusion
Use GPU as a single very efficient CPU
Per-block optimization
Efficient for some operations on very large data set
Mapping to GPU: parallelism

- Use each PE as a small CPU
- Apply an optimized sequential operation on each data set
- Launch operation on multiple data sets
- Efficient for streaming applications, requires more memory
Outline

1. Context and aim
2. Approaches
3. Results
4. Conclusion
Test platform and method

Context and aim

Test platform
- Intel Core i5 760 CPU (4 cores, 2.8GHz, 8MB cache)
- 4GB DDR3 memory
- Linux 2.6.36 kernel
- NVidia GTS 450 GPU, Asus DirectCU Card, 1GB DDR5 memory

Method
- 3 single operations:
  - FFT
  - IIR
  - Mapping
- Sequences of operations

Approaches

Results

Conclusion
FFT

Context and aim

Approaches

Results

Conclusion

- Straightforward solution inefficient on considered vector sizes
- Small gain for GPU solution
- Data transfer reduces performance
- GPU monitoring:
  - 10% for straightforward solution
  - 98% for parallel solution
No optimized algorithm for straightforward solution

~ 50% gain for GPU solution

High block size requires more memory
Demapping

Context and aim

Approaches

Results

Conclusion

- No need for high processing power
  → GPU core is sufficient

- Very efficient on GPU, even for large data set
Multitasking

Context and aim

Approaches

Results

Conclusion

- No multitasking on GPU: sequential execution
- Issue on buffer management reduces performance
- 20% gain for 4 tasks for size 1024
Conclusion and perspectives

Contributions

- Study of two possible solutions for GPU integration
  - an existing solution, with disappointing results
  - a new solution for streaming application, with promising performance

Perspectives

- Resolve the buffer management issue
- Experiment in a real radio application