Analyzing Benefits of FPGA Libraries for Scientific Applications

Maya B. Gokhale, Christopher D. Rickett, and Chung Hsing Hsu

Los Alamos National Laboratory

July 12, 2006
Outline

• Motivation for library approach to hardware.
• Descriptions of selected scientific applications.
• Profiles of scientific applications.
• Conclusions
Motivation for Library Approach

- The path to continued performance appears to require heterogeneous computing.

- Best alternative, if available.
  - Application developer doesn’t have to design hardware.
  - Hardware/algorithm expert optimizes library function.

- FFT “IP” modules have been implemented that outperform older ASICs.
  - Good performance as FFT is in data flow of signal processing pipeline

- But ...
  - Desired library functions may not exist.
  - Difficult to chain library functions with other desired processing
    - e.g., matrix multiply as part of conjugate gradient solver
Goal of Study

- Want to assess whether a specific large scientific application can be accelerated by FPGAs.
- What parts of the code should be ported to the FPGA?
- Need at least 2X application speedup to make it worthwhile.
- Study includes two NSF HPC Scientific Grand Challenge problems and other codes.
  - Representative of applications of interest to the scientific community.
Profiled Codes

- **MILC**
  - studies of the mass spectrum of strongly interacting subatomic particles, the weak interactions of these particles, and the behavior of strongly interacting matter under extreme conditions (quantum chromodynamics). **57,000 lines of C code** counting header files and comments.

- **GAMESS**
  - ab initio studies of quantum chemistry. Computes molecular geometries, energies, etc. **> 100,000 lines of Fortran.**

GAMESS: Flexibility and Molecular Recognition in the Immune System
Profiled Codes (cont.)

- Parallel Ocean Program (POP)
  - ocean circulation model used as the ocean component of a system climate model; used to resolve eddies in global ocean and ocean-ice models. almost 48,000 lines of Fortran 90.

- GROMACS
  - molecular dynamics simulations of complex biomolecules; simulates the Newtonian equations of motion upon large systems of particles (i.e. ensemble molecular dynamics). 316,000 lines of C code.
Locating acceleration regions of code

- Study execution profile
  - TAU, gprof
  - Quantify time spent at routine, loop, or even line granularity
  - Find representative data sets
    - Execution profile may vary greatly depending on data set
    - Want 80% time in a small region, but that doesn’t occur too often

- Study code of likely acceleration candidates
  - Data type - integer, single precision FP, double FP
  - Types of operations - divides, transcendental functions
  - Numbers of operations - how many FP units are needed
  - Dependency graph

- Study data profile
  - Data consumed and produced in a region must be communicated between global microprocessor memory and FPGA board memory
  - Need to know amount of data transferred (per loop iteration)
  - Need to know if communication and computation can be overlapped

- We look first for library acceleration opportunities, next for compute kernels
TAU – Tuning and Analysis Utilities

• TAU automatically inserts timer calls at the start and end of a function.
• The instrumented source then gets compiled into the resulting object.
• At run time, each timer is started at the beginning of a routine and stopped at any point of return for the routine.
• TAU gives
  – The total time spent in each routine, and
  – Call-path information to show what execution path(s) caused what amount of run time.
• Compensates for overhead of profiling.
• www.cs.uoregon.edu/research/tau
LINPACK

• Most popular benchmark, uses linear algebra library, especially DGEMM (double precision, dense matrix multiply).

• 74% time spent in DGEMM.

• 5X DGEMM acceleration gives 2.45X overall speedup.

• 10X DGEMM acceleration gives 3X overall speedup.
MILC Profile – 4 Processes

- Timing results from running on 4 dual socket 2.0 GHz Opteron (2GB memory, gcc/gfortran 4.1.0)
- Approx. 75% in matrix algebra routines, and taking > 5% run time each.
- Ignoring data transfer time, speedup of 3X needed in those routines to get 2X speedup on the application.

Recent results show 12GF on V2-6K for dense matrix multiply (Delft); 4GF on 2.4GHz Xeon for DGEMM = 3X
MILC Profile – 64 Processes

- Timing results from 32 dual socket 2.0 GHz Opteron (2 GB memory, gcc/gfortran 4.1.0).
- Over 20% time in MPI_Wait
  - In comparison, the 4 process was approx. 7.4%
- Approx. 55% time in matrix algebra routines, each > 5%.
  - Scaling effects.
- Speedup of 10X would be needed for a 2X application speedup.
GAMESS profile

- Largest time spent in EDIMER for one data set.
  - Long, complex routine with lots of control flow
- `dxpy` and `dgmm` (wrappers to `DAXPY` and `DGEMM`) account for 10%
  - Using optimized BLAS.
- Not enough library routine usage to benefit from library-based acceleration
- Must examine other routines for possible compute kernel acceleration
- Dual socket 2.4 GHz Xeon, 2GB memory, ifort 9.0
Data Profile

GAMESS: DGEMM array sizes
• Vary with the data sets
  • Many calls with relatively small size arrays
  • Many calls with large array sizes
POPOP Profile – 4 Processes

• Top library routine is conjugate gradient solver (PCG).
• Average 10.7% of the run time.
• Max speedup of 12%.
• Hardware PCG library would have little impact on POP run time.
• Combination of all kernels < 30%.
• Poor candidate for FPGA acceleration.
POP Profile – 32 and 64 Processes

- Top routine is no longer the conjugate gradient solver.
  - Solver is < 5% for 32 processes; even less for 64!
  - Runtime is dependent on input parameters.

- Very complex routines, with large amount of control flow.
  - Conditionals within inner-loops.
  - Difficult to map to FPGAs.

- Approx. 63% in routines > 5% for 32 processes; 48% for 64.
  - Scalability effects.
  - Should you use 32 or 64 processors?
GROMACS Profile

- Profiled with gprof on Xeon for a palmitic acid simulation.
- No library routines used.
- Specially coded sse routines used - inl3300, inl330.
- Up to 55% in 5 routines
  - solve_pme very complex
  - sse routine already highly optimized
Conclusions

• Acceleration using only library routines in general will be negligible for scientific codes.
  – Even Linpack needs at least 5X DGEMM acceleration, which has not yet been demonstrated for double precision FP.
  – Scientific codes designed to handle large number of parameters and often don’t have single compute-intensive regions.

• Application compute kernels can be long and complex.
  – Not a good fit for FPGAs.

• Must re-think application’s algorithms and data access patterns.