Exploiting Dependencies as Concepts for Parallel Programming

Eva Burrows, Magne Haveraaen

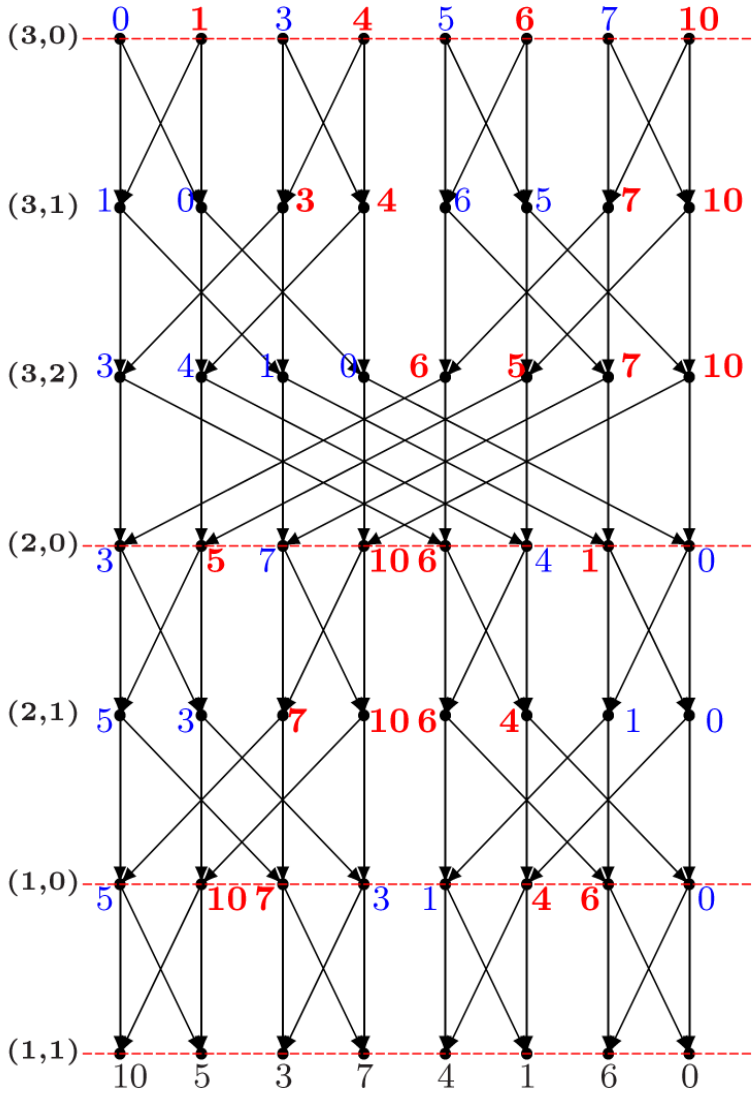
Bergen Language Design Laboratory (BLDL)
Department of Informatics, University of Bergen, Norway

Workshop on Concepts, Bergen, November 5, 2009

Programming Parallel Architectures

- Traditional: OpenMP and MPI
  - Communication architecture ignored
- Architecture-aware programming
  - Existing codes cannot be ported directly to new architectures
  - New architectures come along with new programming models: one for GPUs, one for Cell and so on.
- Hardware independent programming
  - Map computations to new architectures without rewriting the problem solving code
  - Higher level abstractions needed

Bitonic Sort Dependency



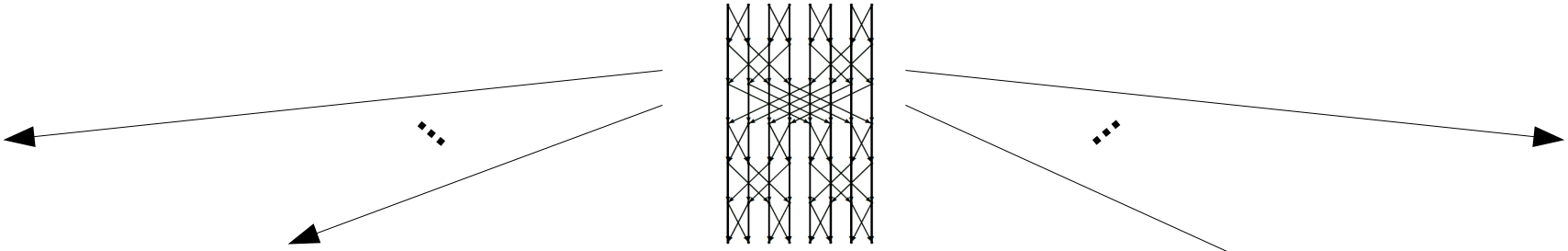
OUTPUT: sorted sequence

1 bitonic sequence of lenght 8

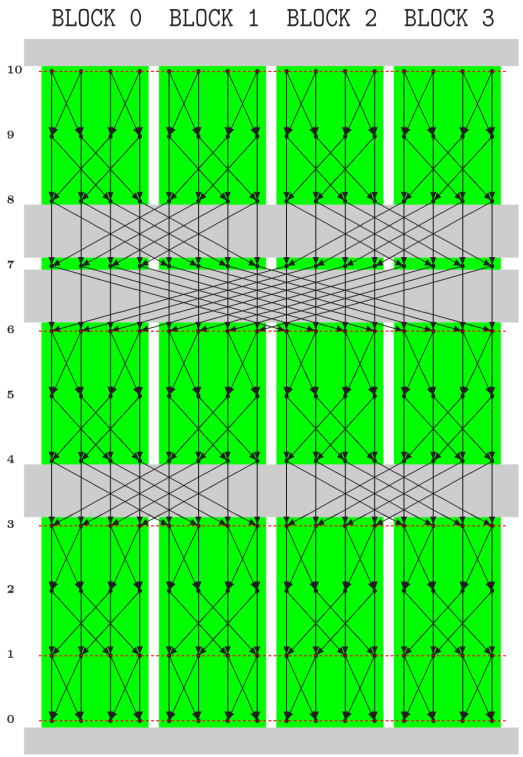
2 bitonic sequences of lenght 4

INPUT: viewed as 4 bitonic sequences of lenght 2

Embedding BS DDA onto Various Hardware

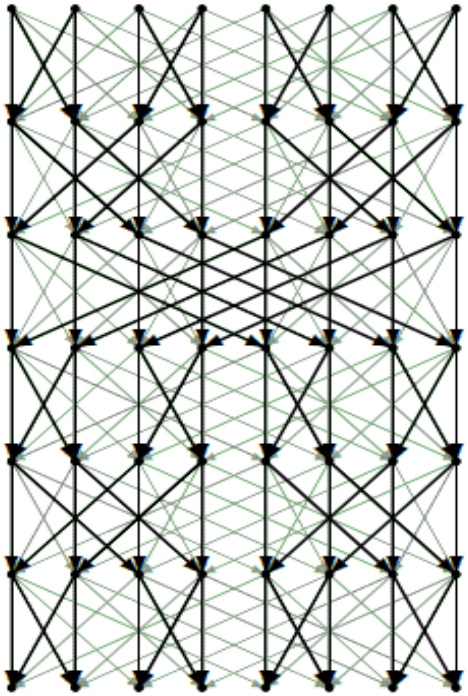


Bitonic Sort DDA



GPU: NVIDIA's CUDA

MPI-code



Hypercube

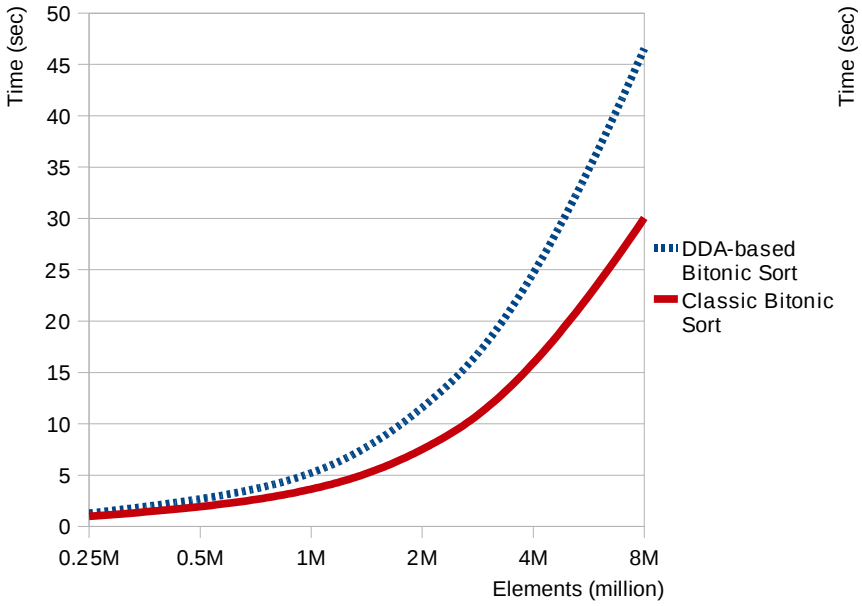
DDA concept (API) definition in Magnolia

```
module DDA
imports Equivalence;

concept DDA<type P, type B> {
requires Substitutable<P>; requires Substitutable<B>;

    /** The signiture */
        predicate rg (P p, B b);
        function P rp (P p, B b) guard rg(p,b);
        function B rb (P p, B b) guard rg(p,b);

        predicate sg (P p, B b);
        function P sp (P p, B b) guard sg(p,b);
        function B sb (P p, B b) guard sg(p,b);

    /** The axioms */
        axiom Receives (P p, B b) {
            assert sg(rp(p,b),rb(p,b));
            assert sp(rp(p,b),rb(p,b)) <-> p;
            assert sb(rp(p,b),rp(p,b)) <-> b;
        }
        axiom Supplies (P p, B b) {
            assert rg(sp(p,b),sb(p,b));
            assert rp(sp(p,b),sb(p,b)) <-> p;
            assert rb(sp(p,b),sb(p,b)) <-> b;
        }
}
```

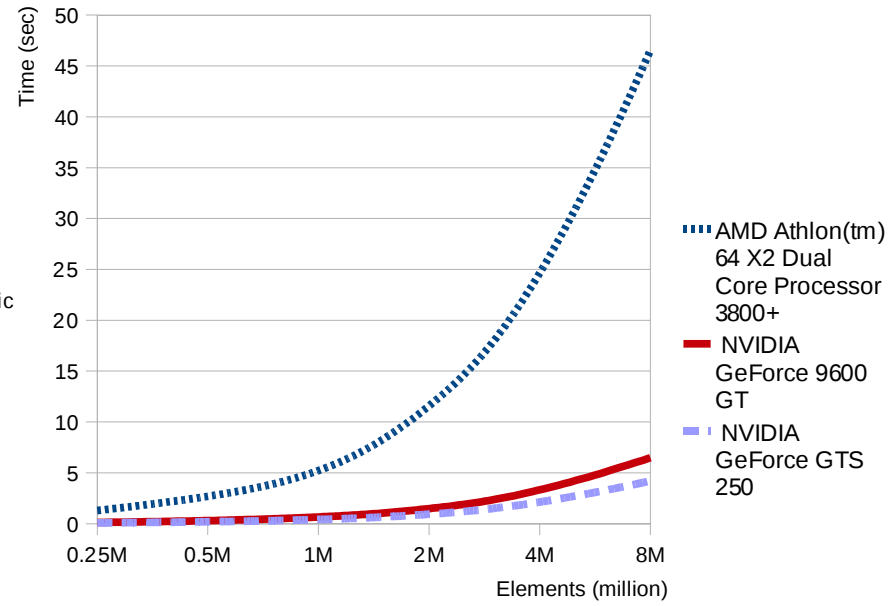
DDA Concepts and Compilation Schemes

- Plain DDA concept
  - Hashmap based implementation
    - Time control, no space control
- DDA with space-time projections concept
  - Sequential implementation
    - Time and space (memory layout) control
  - Parallel execution model using MPI
    - Time and space (parallel distribution) control
    - No communication structure control (limitation of MPI)
  - CUDA / OpenCL execution model using threads
    - Time and space (kernel/block/thread/memory) control
    - Communication structure control

Run Times for DDA-based Bitonic Sort



a)



b)

DDA-concept implementations are portable across platforms.

DDAs as Concepts

- Application domain – compiler construction
- DDA concept – API for the user
- Predefined collection of concepts with associated computational mechanism:
  - CUDA-execution model
  - MPI
  - Hypercube
  - FPGA,
  - etc
- Portability
- User benefits from axiom-based testing tools.