A Many-Core Implementation based on the Reconfigurable Mesh Model

Heiner Giefers, Marco Platzner
Computer Engineering Group
University of Paderborn

{hgiefers, platzner}@upb.de
Motivation

- Investigate the reconfigurable mesh model for many-cores

- Main communication architectures
  - Shared bus
    - Not scalable
    - Uses shared memory as natural communication model
  - NoC
    - Scalable, but routers produce an overhead of area (and time)
    - Uses message passing as natural communication model

- What to expect from a reconfigurable mesh architecture?
  - Light-weight interconnect
  - Scalability
  - Well-investigated programming model
Outline

- Motivation
- Reconfigurable Mesh
  - Reconfigurable mesh model
  - Algorithms
  - Practical issues
- RMESH implementation
  - Reconfigurable switch
  - Processing element
  - RMESH execution
  - Tool chain
  - Host integration
- Case studies
- Summary and Outlook
Reconfigurable Mesh Model

- **Connection autonomy**
  - 15 possible patterns

- **Many model extensions**
  - PARBS [Wang and Chen 91]
  - RMESH [Miller et. al. 93]
  - Polymorphic-Torus [Li and Maresca 89]
  - Constrained RM [Beresford-Smith et. al. 96]

- **Cores execute in a lock step**
  1. Switch configuration
  2. Communication
  3. (Constant time) computation

- **Delay models**
  - $O(1)$ and $O(\log N)$
  - $O(l/k)$
Algorithms

- Basic algorithms
  - Arithmetic [Miller et al 93]
    - Prefix sum of \( N \) elements, \( N \times N \) mesh: \( O(1) \)
    - Minimum of \( N \) elements, \( N \times N \) mesh: \( O(1) \)
  - Sorting of \( N \) elements
    - On \( N \times N \) mesh: \( O(1) \) [Jang and Prasanna 95]
    - On \( N^{1/2} \times N^{1/2} \) mesh: \( O(\sqrt{N}) \) [Thompson and Kung 77]
  - Graph algorithms
  - Image processing

- Simulation of network topologies
  - Pyramid: \( O(T(N)) \)
  - Mesh-of-Trees: \( O(T(N) \cdot \log N) \)

- Algorithmic scalability
  - Self simulation
  - Optimal scaling for restricted models
    - Simulate a mesh of size \( N \) in \( O(N/P) \) steps on a mesh of size \( P \)
Practical Issues

- Reconfigurable mesh has several drawbacks when it comes to an implementation
  1. Communication delays of $O(1)$ or $O(\log N)$ are unrealistic assumptions
  2. Data I/O is not covered by runtime analysis

- Our approach: reduce the reconfigurable mesh model to practice
  - Accept multi-cycle communication delay
Reconfigurable Switch & Processing Element

- **Reconfigurable switch**
  - Directed unbuffered communication
  - Parameterizable model w.r.t. switch patterns and data widths

- **Picoblaze**
  - 8-bit RISC, not pipelined
  - 96 slices on Virtex-II
  - 1024 instructions in BRAM
  - 2 cycles per instruction
  - 64 byte scratchpad RAM

- **Wrapper**
  - Storing data and switch configuration
  - Optional I/O functionality
RMESH Execution

- Communication scheduling
  - Configure switch by one assembly instruction
  - Globally synchronous execution
  - Communication pattern is known at compile time and often independent of local data
  - Worst case for global broadcast on an $N \times N$ mesh is $\Theta(N)$
Tool Chain

- Xilinx EDK flow for HW generation
- RMESH algorithms in C
- Assembly code generator
  - No (suitable) Picoblaze compiler
- Cycle-accurate simulator
  - Simulate arbitrary mesh sizes

```
for (i=1; i<=log(N); i++)
{
    if ((p%N)%i2==0)
    {
        switch_state('E');
        readbus(D_IN, i2/2);
        compute(ADD, ACC, D_IN);
    }
    else
    {
        if ((p%N)%i2==i2/2)
        {
            switch_state('W');
            writebus(ACC, i2/2);
            compute(NOP);
        }
        else
        {
            switch_state('WE');
            donop(i2/2);
            compute(NOP);
        }
    }
}
```

Tool Chain

- Xilinx EDK flow for HW generation
- RMESH algorithms in C
- Assembly code generator
  - No (suitable) Picoblaze compiler
- Cycle-accurate simulator
  - Simulate arbitrary mesh sizes
Host Integration

- RMESH as coprocessor
- MicroBlaze 32-bit RISC core as host CPU
- Connected through FSL channels to an I/O controller
  - Low latency of FSL
  - 32-bit to 4 x 8-bit splitting
  - Load and store operations take $O(N)$ time
- Entire system runs at 100MHz
Synthesis Results for Virtex-II / Virtex 4

- Picoblaze + wrapper
  - 132 slices, 1 BRAM
  - 6ns critical path

- Reconfigurable switch
  - 101 slices
  - 4.5ns critical path

- 4x4 mesh + Microblaze on XC2VP30-7
  - 51% slice utilization
  - 37% of BRAMs used

- Synthesis results for XC4VLX200
  - 4x16 mesh + Microblaze + peripherals:
    - 21% slice utilization
    - 27% of BRAMs used
  - 16x16 mesh
    - 67% slice utilization
    - 76% of BRAMs used
Case Study: Sorting

- Sorting algorithms
  - Several optimal sorting algorithms known for reconfigurable mesh
  - Odd-even transposition sort implemented on 4x4 prototype system
    - Sorts in linear time
    - Optimal $O(N^{1/2})$ algorithms can’t beat odd-even transposition on 4x4 mesh

- Microblaze host
  1. Calculates a random set
  2. Sends frames of 16 values to the mesh
  3. Receives and merges sorted frames

- Runtime comparison to quicksort on Microblaze
  - Single Microblaze
  - Two Microblaze cores connected through FSL
    - Takes nearly the same amount of slices as Microblaze + 4x4 mesh
Sorting Results

- Speedup decreases with a growing part of merging on host due to the dominance of the $O(n \log n)$ merge operation.
Case Study: Binary Tree

- Add up $N^2$ elements on an $N \times N$ reconfigurable mesh
  - Results based on a cycle-accurate simulation
  - Without I/O, data is preloaded

![Graph showing runtime vs. datasynce for different simulations.](image)
# Summary & Future Work

## Summary
- Reconfigurable mesh implementation on FPGA
  - Reconfigurable switch
  - Programmable processing element
- Case study on a small sized mesh shows the potential of the approach

## Ongoing work includes
- More experiments on larger meshes
- Implement self simulation methods
- Communication and synchronization mechanisms for larger meshes
  - Investigate buffering and explicit synchronization
- Exploit HW reconfigurability for reconfigurable meshes
  - Adapt switches to (algorithm-) communication pattern requirements
  - Adapt processing elements to computation requirements
Thanks for your attention!