NoC Simulation in Heterogeneous Architectures for PGAS Programming Model

Sascha Roloff, Andreas Weichslgartner, Frank Hannig, Jürgen Teich
University of Erlangen-Nuremberg, Germany

Jan Heißwolf
Karlsruhe Institute of Technology, Germany

M-SCOPES 2013, St. Goar, Germany, June 20, 2013
Introduction

- Networks-on-Chip (NoC) are state of the art in industry and academia

- Benefits of NoCs
  - Better scalability
  - Higher throughput
  - Increased power efficiency
  - No single bottleneck
  - Higher parallelism
Introduction

- NoCs introduce tiled architectures
  - Tile = Node in the NoC
  - Tiles contain resources
    (Memory, CPUs, etc.)
  - Distributed memory
  - Tiles may be
    homogeneous
    or heterogeneous

- Heterogeneous NoC-based MPSoCs induce big challenges in their design and programming
  ➔ Hardware/Software co-design problem
Motivation

- **Hardware architect**
  - Explores design space of NoC-based architectures using simulators
    - Number and type of tiles, topology of the NoC
    - Clock frequency, latency, throughput, buffer sizes
  - NoC-Simulators use models of applications or only traffic generators, no "real" applications
Motivation

- **Software programmer**
  - Writes parallel programs for exploiting parallel architectures
    - Programming model supporting distributed memory
  - Performance evaluation on physically not available architectures via simulators
    - Requires translation into an appropriate application model

```java
public static def main() {
    val dist = Dist.makeBlock(1..512);
    val array = DistArray.make[int](dist);
    val globalSum = GlobalRef[Cell[int]](new
        Cell[int]());
    finish for(p in dist.places()) async at (p) {
        Console.OUT.println("work on place "+p.id);
        val localArray = array.getLocalPortion();
        var sum:int = 0;
        for([i] in localArray)
            sum += localArray(i);
        val result = sum;
        at (globalSum) atomic globalSum() += result;
    }
}
```
Proposed Solution

- Unified simulation framework that combines
  - NoC-based MPSoC modeling
  - Simulation of parallel software on the modeled MPSoC architecture

- No translation into an appropriate model necessary!
- Instead using actual communication demand inherently given by the parallel program
PGAS Programming Model

- Partitioned global address space (PGAS)
  - Single logical global address space
  - Locality awareness (Place = shared memory location)
    - Local and remote accesses
  - PGAS languages: X10, Chapel, Fortress, UPC, CAF
X10 Programming Language

- Parallel, object-oriented programming language
- Developed by IBM (since 2004)
- Implements PGAS programming model

- Key features
  - Classical object-oriented language features (like in Java)
  - Functional language features (like in Scala)
  - Parallel programming constructs
    - Intra-tile parallelism
    - Inter-tile parallelism
    - Synchronization
X10 Programming Language

- Intra-tile parallelism
  - Exploiting multiple on-tile CPUs by activities
  - Activity = light-weight thread
  - Created by `async {S}`

```java
1 val list = new ArrayList[int](0);
2 /* intra-tile parallelism */
3 finish {
4 async {
5     atomic list.add(1);
6 }
7 async {
8     atomic list.add(2);
9 }
10 }
```
X10 Programming Language

- Inter-tile parallelism
  - Distribute workload to several tiles
  - Creating remote activities on other places
    - Causes communication overhead (closure)
  - Initiated by `at (p) {S}`

```java
val list = new ArrayList[int](0);
/* inter-tile parallelism */
finish {
    async at (Place(2)) {
        list.add(1);
    }
    async at (Place(3)) {
        list.add(2);
    }
}
```
X10 Run-time

- Each X10-specific language construct is replaced by the X10 compiler to a call into the X10 run-time (XRX)
  - Written in X10
- Parts for communication (runAt) written in native code (C++ or Java)
  - Serialization of all data objects that are accessed by the closure
  - Transmission of the data to the remote place by the X10RT

```
Activity:
    ...........
    ...........
    at(2) print ("Hallo");
    ...........
    ...........

Place 1

Run-time

    runAt
    ↓
    serialize
    ↓
    x10rt_send_msg

X10RT

    x10rt_lgl_send_msg
    ↓
    deserialize
    ↓
    CUDA, MPI, ...

Place 2

Activity:

    print ("Hallo");
```
X10 Run-time

- **X10RT**
  - Library that implements actual communication between places
  - Only *target place* and *payload* are passed to the X10RT
    - Input to our network simulation model
- Coupling with our framework by replacing X10RT with our network simulation model
Target Architecture

- Heterogeneous tiled many-core architecture

- A flexible NoC interconnects the tiles
  - Tiles are equipped with a network adapter
  - NoC routers are connected by point to point connections (links)
    - Time multiplexed through *virtual channels* (VC)
  - Supports best effort (BE) and guaranteed service (GS) data transfers
NoC Architecture

- Router structure
  - Five input and output ports connected by a crossbar (mesh topology)
  - One buffer per virtual channel
  - Transmission control
    - Decides how often a VC is scheduled for transmission (round-robin)
    - Depends on the number of reserved timeslots

- Routing mechanism
  - XY-routing: Packet is routed first in x direction then in y direction
NoC Architecture

- Flit size
  - Amount of data that can be transmitted over a link in one machine cycle
- Every packet is split up into flits

- HEAD flit and TAIL flit mark begin and end of a transmission
  - HEAD flit contains destination address and BE or GS flag
  - Allows routing decision and VC reservation within a router
- BODY flits carry the payload
NoC Architecture

- **BE connection**
  - Exactly one timeslot reserved per router link
  - BODY flits follow directly the HEAD flit
  - No guaranteed bandwidth

- **GS connection**
  - Service level defines number of reserved timeslots
  - BODY flits wait until HEAD flit reaches destination
  - Guaranteed bandwidth after connection establishment
  - Allows Quality of Service (QoS)
GS Guarantees

- For GS connections upper bounds for throughput ($TP_{GS}$) and delay ($t_{GS}$) for a certain service level (SL) can be given as follows:

$$TS_{util,max} = \max_{i \notin 0,k-1} \{TS_{util,i}\}$$

$$TP_{GS} = \frac{SL}{TS_{util,max}} \cdot TP_{link}$$

$$t_{GS} = \left[\frac{TS_{util,max}}{SL} \cdot (H + S_{pkt} - 1)\right] \cdot T_{hop}$$

- $TS_{util,i}$ – Utilized time slots at port $i$
- $k$ – Number of ports
- $H$ – Number of hops
- $S_{pkt}$ – Packet size in flits
NoC Simulation

- Written in X10 to be fast and easy to couple to existing framework
- Two operation modes:
  - Flit-accurate simulation
  - Formula-based simulation in case of GS channels
- X10 Run-time integration
  - Providing as much places to the X10 program as tiles in the modeled architecture
  - But using only one physical X10 place for simulation
NoC Simulation

- Simulation framework integration
  - Multiple simulation threads maintain their own simulation clock and advance in different time steps
    - Activity simulation threads
    - NoC simulation thread
  - Synchronization between simulation threads at synchronization points
Experiment I

- Comparison with SystemC flit-based simulation
  - Picture-in-Picture (PiP) task graph is mapped to a 4x4 NoC architecture
  - $C_0…C_7$ denote bandwidth requirements between the tasks in Mb/s
  - GS channels with SL 1 are used for all connections
Experiment I

- Test environment:
  - Intel Core i7-870 CPU
  - 2.93 GHz

- Average flit delay
  - SystemC simulator: 8.3 cycles
  - X10 simulator: 8.33 cycles
  → Error: 3.15% - 6.86%

- Simulation time
  - SystemC simulator: 4012 ms
  - X10 simulator: 406 ms
  → Speedup: 9.88
Experiment II

- Evaluation of different simulation modes
  - Histogram calculation on a 2x2 NoC architecture
    - Distributed X10 version
  - Input image is block-based partitioned to all available places
    - Processed in parallel and reduced at the initial place
Experiment II

- Evaluation of flit-based vs. formula-based simulation modes
  - Flit-based simulation is reference
  - Latency for computation and communication
  - GS Channels up to SL 4

- Results
  - Error: 0.02% - 7.8%
  - Speedup: 9.5 - 695
Conclusions

● Trend to heterogeneous NoC-based architectures induces simulation challenges

● Our framework combines the two worlds of
  ● NoC-based MPSoC modeling
  ● Simulation of parallel software using the PGAS programming model

● Different simulation modes allow tradeoff between accuracy and simulation speed
• Thank you for your attention!

• Questions?