Dataflow programming for heterogeneous computing systems

Jeronimo Castrillon
Cfaed Chair for Compiler Construction
TU Dresden
jeronimo.castrillon@tu-dresden.de

Tutorial: Algorithmic specification, tools and algorithms for programming heterogeneous platforms. PACT Conference. San Francisco, October 19th 2015
Outline

- Heterogeneous systems
- Dataflow models
- Analysis and synthesis
- Summary
Outline

- Heterogeneous systems
- Dataflow models
- Analysis and synthesis
- Summary
Heterogeneous computing systems

- Today’s heterogeneity
  - Desktop/HPC: GPGPU, GP + ACC
  - Embedded: RISCs, DSPs, ASIPs, …
  - Resources: different performance/energy characteristics

- Tomorrow’s heterogeneity: Emerging technologies
  - Heterogeneity beyond performance and energy
    - Reliability/error tolerance
    - Computation model
German Excellence Cluster: Goal – “to explore new technologies for electronic information processing which overcome the limits of CMOS technology”
Heterogeneity: Example SiNW

- SiNW: Silicon Nanowires
- Multi-gate devices with less performance penalty
- Reconfigurable P/N functionality

Possibilities
- New micro architectures
- New pipeline structures
- New field programmable devices
Heterogeneity: Example chemical processing

- Lab-on-Chips: for sensing, analysis and test
- Also for computing?
  - Different kinds of transistors
  - Oscillators and other components

Fundamentally different way of computation

[Voigt14]
Heterogeneity: Example DNA origami

- DNA origami: Self-assembled 2D/3D structures made of DNA strands
- Use structures to build advance electronic devices
- Example: Plasmonic waveguides

Courtesy: Thorsten Lars-Schmidt
https://cfaed.tu-dresden.de/schmidt-home
Consequence of heterogeneity

Already difficult to program them today, what about tomorrow?  
Need for models and abstraction
Outline

- Heterogeneous systems
- Dataflow models
- Analysis and synthesis
- Summary
Models: Introduction

- Von Neumann model makes things complicated
  - Sharing state
  - Data races

- Task graphs: A simple parallel programming model
  - Intel TBB, .NET Task parallel library (TPL), OpenMP Tasks, ...
  - Runtime and data management, e.g., StarPU [Augonnet10]
Directed acyclic task graphs

- Very well studied, see for example [Kwok99]
- Difficult problem for heterogeneous systems
Dataflow models

- Also a graph representation: Nodes & edges are called actors & channels
- Implicit repetition, common in streaming, signal processing applications
- Communication: only through channels
- Multiple flavors of models: Rules that determine when an actor fires
- A graph models multiple possible executions:

cf. LabVIEW models
Dataflow models (2)

- Also a graph representation: Nodes & edges are called **actors & channels**
- Implicit repetition, common in streaming, signal processing applications
- Communication: **only** through channels
- Multiple flavors of models: Rules that determine when an actor **fires**
- A graph models multiple possible executions:

What now?
Dataflow models (3)

- **Synchronous Dataflow (SDF):** every actor has a **fixed behavior**
  
  ![Synchronous Dataflow Diagram]

  - a3 always writes 1 token to e4

- **Cyclo-Static Dataflow (CSDF):** every actor has a **set of fixed behaviors**
  
  ![Cyclo-Static Dataflow Diagram]

  - a1: writes 1 token, then 0, then 0 to e2
Dynamic models and Kahn Process Networks

- Dynamic dataflow: set of firing rules per actor

- Kahn process networks (KPN): nodes are now called processes

```
Firing rules:
...
R_{a_2,1} = ([\ast, \ast], [\ast])
R_{a_2,2} = ([\bot], [0])
R_{a_2,3} = ([0], [\bot])
R_{a_3,2} = ([\ast, \ast, \ast])
...
```

p1: writes any amount of tokens to e2 at any time
Outline

- Heterogeneous systems
- Dataflow models
- Analysis and synthesis
- Summary
Analysis and synthesis

cf. Silexica’s tool flows

Analysis

Synthesis

Code generation

Property models (timing, energy, error, ...)

Non-functional specification

Application

Architecture model

Analysis and synthesis

Example for SDFs

- Compute topology matrix, and solve system of equations
- Solution: repetition vector serve to unroll the graph \([1 \; 3 \; 2]\)
- Perform mapping and scheduling on the resulting directed acyclic graph (DAG)
Example for KPNs: Static & dynamic analysis

- Need to understand process interactions

[Diagram showing process interactions and annotated code segments]

Unrolled CFG for $P_1$

[Diagram with unrolled control flow graph for process $P_1$]

Unrolled CFG for $P_1$

[Diagram with unrolled control flow graph for process $P_1$]
KPN & DDF: Tracing

- Dynamic analysis based on execution traces [Castrillon10/13, Brunet15, Singh15]

DAG representation for further analysis and synthesis
Models from functional specification

- Inspect functional specification of actors/processes (cf. 2nd talk)
  - Instrumentation, emulation, cost models, …
  - For timing, energy, errors, …
- Example

---

For timing, energy, errors, …

Example:

```
... for (;i < x;i++) {
  write(&c2);
  f1(...);
  read(&c1);
  f2(...);
  read(&c1);
  ...
```

SSA-IR (LLVM)

```
%ptrx = getelementptr i32
%y = load i32* %ptrx
%z = mul %x, i32 10
%z = add %x, %y
```

Profiler:
- Execution counts, branch stats
Models based on algorithmic descriptions

- When functional specification is not meant for synthesis
  - Common/required in heterogeneous systems (special components)
  - Need to match algorithms to hardware components [Castrillon10b, Castrillon11]

Diagram:
- Algorithm library
- Platform model + characterization of special components
- N: Algorithmic actors
- F: Existing implementation in target platform
Multiple-applications

- Use traces and mappings to reason about platform sharing
Multiple-applications (2)

- Quickly discard “bad” multi-application configurations by observing the platform utilization profiles.
Outline

- Heterogeneous systems
- Dataflow models
- Analysis and synthesis

Summary
Summary

- Need programming models (and HW/SW stacks) to handle heterogeneity
- Even more dramatic in the post-CMOS era

- Dataflow models
  - Natural way to describe some applications
  - Amenable to analysis and synthesis for parallel execution
  - Discuss different kinds of models and required analysis
  - Need models of hardware for synthesis
Acknowledgements

- German Cluster of Excellence: Center for Advancing Electronics Dresden (www.cfaed.tu-dresden.de)
- Collaborative research center (CRC): Highly Adaptive Energy-Efficient Computing (HAEC)
- Silexica Software Solutions GmbH
References


