A Novel Approach for Interactive Debugging of Dynamic Dataflow Embedded Applications

Kevin Pouget[†*], Patricia López-Cueva[†*], Miguel Santana[†] and Jean-François Méhaut[*]

[*]University of Grenoble, CEA, LIG and [†]STMicroelectronics

March 18-22 2013 — SAC — Symposium on Applied Computing Embedded Systems Special Track — Coimbra Portugal

Debugging multicore applications running on embedded platforms is challenging. Dynamic dataflow programming makes it a daunting task.

Embedded System Development

HD Multimedia Applications

⇒ High performance expectations
- H.265 HEVC
- Augmented reality
- 4K digital television
- ...

Sharp Time-To-Market Rqrmts

⇒ Important demand for:
- Powerful parallel architectures:
  - MultiProcessor-on-Chip (MPSoC)
- Convenient programming models:
  - Dynamic dataflow programming
- Efficient verif. and valid. tools:
  - Our research contribution

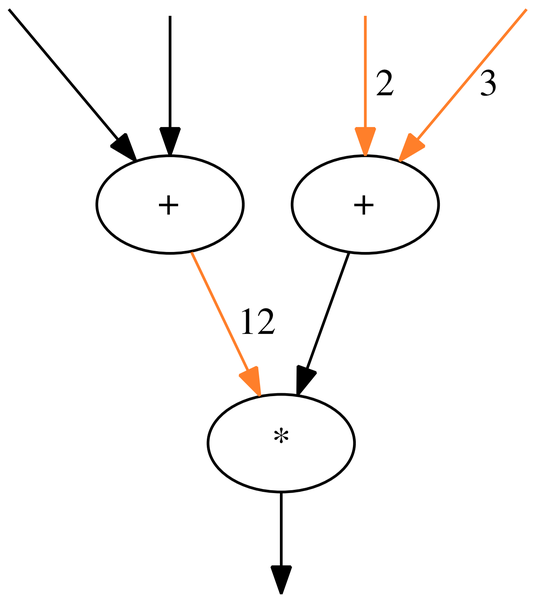
Dataflow Programming

- Alternative to *von Neumann* imperative model (↔ C/ASM)
- Instructions executed when their operands are ready, not when the instruction pointer (aka. program counter, %PC) reaches it

⇒ Inherently parallel

× HW did not follow the trend

⇒ Only hybrid imperative/object + dataflow frameworks available



Decidable Dataflow

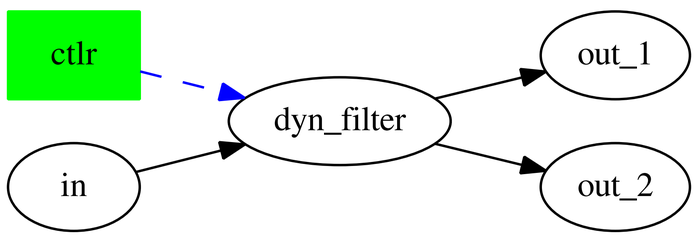
- Correctness analysis
- Deadlock-free static scheduling
- Powerful optimization

but:
- Strong constraints imposed to dev.
- Reduced expressiveness
  - no dynamic problem

Dynamic Dataflow

- Increased modeling flexibility
- Conditional token emission/rception
- Variable actor input/output rates
- Adaptive signal processing

but:
- Debugging is not straightforward ☺




```
WORK() { /* dyn_filter.c */
  flg = ctlr.next()
  if (flg)
   dat = ctlr.next()
   out_1.send(treat(dat))
  else
   ctr = ctlr.next()
   for (i in 0:ctr)
     nxt = in.next()
     out_2.send(treat(nxt))
```

Dataflow Debugging Challenges

Single-threaded applications

- sequential exec controlled by %PC
- only one execution context
- simple flow-control mechanisms:
  - functions, if-conditions, loops

Multi-threaded applications

- multi-sequential execution
- system view: 1 thrd ⇏ 1 filter
- flat organization:
  - no inter-thread relationship

Dataflow applications

Graph-Based Architect.

Token-Based Execution
- concurrent filt[r] exec.
- no function calls
- tokens sent/received

Non-Linear Instructions

after this instruction:

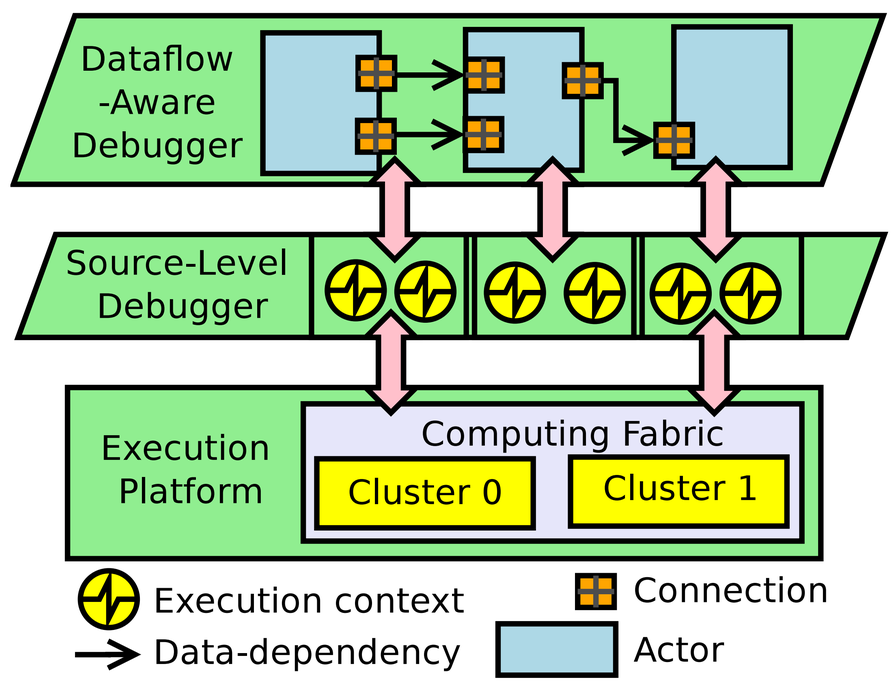
`out_1.send(treat(dat))`,
- `dyn_filr` continues
- `out_1` can run

Objective

Provide debugger users with means to better understand the state of the dataflow execution and easily reach key transition events.

Contribution

Dataflow Aware Debugging



- Graph of dataflow data-dependencies
- Breakpoints on dataflow-related events
- Information about actor interactions
- Hide the inherent complexity of system low-level aspects
  - focus on the execution of user-relevant code

Proof-of-Concept Environment

The Gnu Debugger — GDB

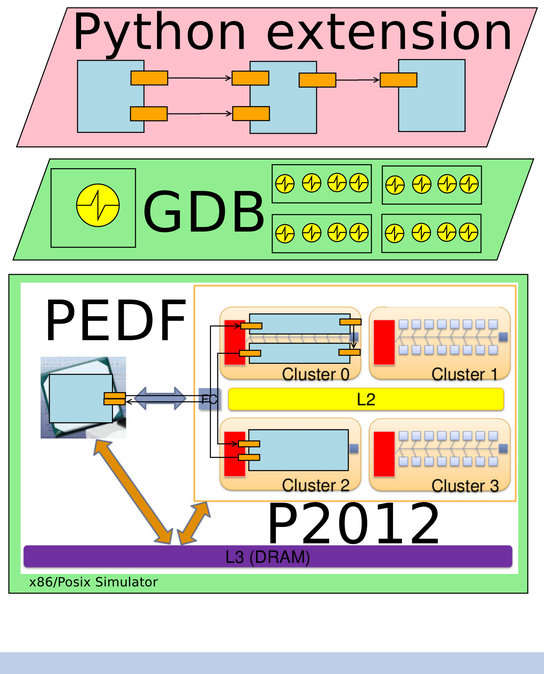
- Extendable with Python API
- Adapted to low level debugging
- Patches contributed to FSF

Predicated Execution Dataflow — PEDF

- P2012 dataflow framework
- Dynamic dataflow model
- Designed to exploit heterogeneity

Platform 2012 — P2012

- ST/CEA MPSoC research platform
- Heterogeneous environment
- 4x16 CPU OS-less computing fabric



Perspectives

- Investigate other programming models
  - OpenCL and GPU computing
- Generalize the approach to programming-model centric debugging
- Use visualization tools to better represent the execution details
- Analyze performance slowdown vs. bug localization speedup

[*]firstname.lastname@imag.fr http://www.liglab.fr/ [†]firstname.lastname@st.com http://www.st.com/