Programming with actors

Jörn W. Janneck
Xilinx Research Labs

credits

Dave B. Parlour  Xilinx Research Labs
Thomas A. Lenart  Lund University
Robert Esser  University of Adelaide
The FPGA Platform:
Huge amounts of fine-grained concurrency.

... along with specialized blocks (multipliers, RAM, ALUs, processors...)

The Problem:
Using FPGAs to implement DSP applications requires circuit design expertise.
The Research Goal:
Design and build models and tools that make it possible for application domain experts to program FPGAs with high-quality implementations.

models and tools

What does it take to program with actors?

- actors and dataflow as a concurrent model
  - Cal as the language for writing actors
  - driver application
- tools
  - code generation
    - circuits and software and combinations thereof
  - animation & visualization
models and tools

We are focusing on...

... **hardware** that can be programmed.

... **programming concepts** that can be implemented.

**driver application**

**MPEG-4 decoder**
driver application

MPEG-4 decoder

metrics

• 60 atomic actors
• 22 atomic actor classes
• 3307 LOC (Cal)
• LOC per actor class between 7 and 2054

actor constructs

• variable token rates
• static/cyclostatic rates
• data-dependent choice
• test for absence of tokens
  • non-prefix-monotonic actors

development time approx. 2 months.

approximate sizes of various models

- Cal: 3,300 LOC
- “architectural” C code: 4,200 LOC
- synthesizable VHDL: 15,000 LOC
code generation

2D-IDCT implementation

- first step to complete decoder implementation
  - naive code generator has sufficient language coverage for IDCT compute.

- 10 classes
- 200 LOC
- covers most language features, except...
  - addressable memory
  - multicycle actions

---

code generation

2D-IDCT, version 1

- Starting architecture is very inefficient: 22 multipliers with 12.5% utilization.
code generation

2D-IDCT, version 2
- interleave row and column streams
- pipelined 1D-IDCT
- result:
  - 6 multipliers with 46% utilization
  - more operator re-use costly in terms of operand routing
  - >100 Mhz clock

summary
- good QoR for “naive” code generator
  - redesigned model compares favorably with existing VHDL implementations
    - smaller, faster, simpler to use
    - HDTV rate
  - demonstrates strength of programming model, rather than quality of code generator
- lots of room for improvement
  - pipelining, folding
animation & visualization

**actor animation (1/4)**

![Input Queues](image1)

**actor animation (2/4)**

![Input Queue History](image2)
animation & visualization

actor animation (3/4)

actor state variables

animation & visualization

actor animation (4/4)

action selection status
outlook

- improved hardware code generation
  - language coverage
  - optimizations
- software code generation
- analysis and optimization tools
- debugging/visualization tools
- alternative entry mechanisms
  - “VisualCal”