SCORE a Scalable Compute Model for Spatial Computing

André DeHon

for

Eylon Caspi, Michael Chu, Randy Huang,
Yury Markovskiy, Laura Pozzi, John Wawrzynek,
Michael Wrighton, Joseph Yeh
Motivation

• What is the right compute model for reconfigurable systems
  – Permits scaling

• What is the right compute model for spatial computing
  – Use vast VLSI area available
  – Exploit parallelism
Capture What can be Done

• What can these systems do well?
  – Build spatial pipelines
  – Perform the **same** computation over and over
  – E.g.
    • Filtering, encryption, image processing…
    • 1-2 orders of magnitude more compute density than processors
Why Not ISA?

• Abstractions Designed for
  – Small machines
  – Small numbers of active computing elements
  – Where
    • compute delay (area) >> interconnect delay (area)

• Interconnect:
  – Implicit
  – Sequentialized
Clock Cycle Radius

- Radius of logic can reach in one cycle (45 nm)
  - Radius 10
    - Few hundred PEs
  - Chip side 600-700 PE
    - 400-500 thousand PEs
  - 100s of cycles to cross
Optimizing

- Must exploit physical locality (placement)
  - Reduce wire requirement
  - Reduce distance traveled over wires
→ new meaning to spatial locality

- Interconnect must show up in our design
  - Run-time management
  - Algorithms
New fixed-point: Stream

- Logical abstraction of a **persistent** point-to-point communication link
  - Has a (single) source and sink
  - Carries data presence / flow control
  - Provides in-order (FIFO) delivery of data from source to sink
  - Logically unbounded
Streams

• Captures communications structure
  – Explicit producer→consumer link up
  – Expose to Compiler and Runtime System

• Abstract communications
  – Physical resources or implementation
  – Delay from source to sink
SCORE (our starting point)

• An attempt at defining a computational model for spatial systems
  – abstract out
    • physical hardware details
    • especially size / # of resources
    • timing

• Goal
  – achieve device independence
  – approach density/efficiency of raw hardware
  – allow automatic scaling
Stream Freedom

- transform
- quantize
- RLE
- encode

Spatial and temporal processes:

- swap
- tran
- qnt
- rle
- enc
SCORE Basics

• Abstract computation is a dataflow graph
  – stream links between operators
  – dynamic dataflow rates
• Allow instantiation/modification/destruction of dataflow during execution
  – separate dataflow construction from usage
• Break up computation into compute pages
  – unit of scheduling and virtualization
  – stream links between pages
• Runtime management of resources
Virtual Hardware Model

- Dataflow graph is arbitrarily large
- Hardware has finite resources
  - resources vary from implementation to implementation
- Dataflow graph must be scheduled on the hardware
- Must happen automatically (software)
  - physical resources are abstracted in compute model
Hardware Abstraction

- Separate:
  - **Logical reconfiguration:** the compute graph changes
  - **Physical reconfiguration:** what’s running on the reconfigurable hardware changes
- Model (user program) supports logical reconfiguration:
  - new operator, new stream, operator ends
- Runtime System responsible for physical scheduling
- Abstracts hardware size → allows scaling
Implementations admitted by Architecture
Basic Organization

- Compute Pages
  - Spatial
  - Temporal (μP)
    - For infrequent code
- Memory Blocks
- Network
- Scale with available capacity
Heterogeneous

- Can accommodate specialized blocks
  - Application specialized
  - IO
- Standard compute model, framework, OS
Challenges
Challenges

- Late Bound Hardware to accommodate scaling, changes in resource mix
- Dynamically evolving computation graph
  - Must move to load-time/run-time
    - Scheduling and Binding
    - Placement
    - Routing
Compute Pages

• Reduce scope of problem
• Group number of blocks together
  – Schedule/route atomically
  – Compare management of pages in VM
  – How many: 100—1000? LUTs
    • Open questions under investigation
• Page generation generally offline
• Reduces size of online problem by 2-3 orders of magnitude
SCORE Page Generation

Programming Model
- Graph of FSMD operators
  - unlimited size, # IOs
  - no timing constraints

Execution Model
- Graph of page configs
  - fixed size, # IOs
  - timed, single-cycle firing

DeHon Stream Workshop Aug. 2003
Quasi-Static Scheduling

- Schedule at load time
- Apply schedule during execution
- Recompute schedule only when significant changes merit
- Contains overhead
  - Allows smarter scheduling
Quasi-Static Scheduler

• A low overhead scheduling solution
  – Scheduler overhead (avg. 14Kcycles)
  – Reconfiguration (avg. 4Kcycles)
• Using on-chip memory blocks

[Markovskiy…/FPGA’02]
JPEG: Dynamic<->Quasi-static

JPEG Decode Makespan

Makespan (MCycles)

Array Size

Dynamic
Static
Qstatic

[Markovskiy…/FPGA’02]
Hardware-Assisted, Online Routing

- Augment network
- Can use network itself to support routing
- Parallel Route search
- Routes JPEG-Decode in ~4K cycles
  - 700× faster than software
Spatial Route Search in Action

[DeHon…/FCCM’02]
Reconfigurable Self-Placement

- Systolic simulated annealing
- Configure compute page as placement engine
- Perform local swaps
- Speedup three orders of magnitude
  - Millisecond placement
Issue
What level is the model?

- Binary / like ISA model
  - Streams present at architecture level
    - strm_read R1,S2
  - Schedule RT graph to fit hardware
- JIT compilation
  - Target operator to different Page types dynamically
  - Runtime Area-time tradeoffs in operator implementation
Summary

• Interconnect key (dominant) resource
  – Should be explicit for optimization
  – Streams first-class abstraction for communications

• Streams key components of scalable compute model
  – For spatial computing systems

• Growing Evidence
  – Can manage mapping tasks dynamically at runtime
    • Scheduling, placement, routing
Additional Information

• SCORE:
  – http://brass.cs.berkeley.edu/SCORE
  – especially see “Introduction and Tutorial”

• CALTECH:
  – http://www.cs.caltech.edu/research/ic/