CS294-48: Hardware Design Patterns
Berkeley Hardware Pattern Language Version 0.3

Krste Asanovic
UC Berkeley
Fall 2009
Overall Problem Statement

MP3 bit string → Application(s) → (Berkeley) Hardware Pattern Language → MP3 bit string → Hardware (RTL) → Audio
BHPL Goals

BHPL captures problem-solution pairs for creating hardware designs (machines) to execute applications

BHPL Non-Goals

- Doesn’t describe applications themselves, only machines that execute applications and strategies for mapping applications onto machines
BHPL Overview

Applications (including OPL patterns)

Structural Patterns
- Pipelines
- Model-View-Controller
- Event Based
- Process Control
- Iteration
- Map-Reduce
- Layered Systems
- Task Graphs

Computational Patterns
- Circuits
- Dense Linear Algebra
- N-Body Methods
- Sparse Linear Algebra
- Spectral Methods
- Unstructured Grids
- Graph Traversal
- Structured Grids
- Graph Algorithms
- Graphical Models
- Dynamic Programming
- FSMs

Structural Patterns
- Pipelines
- Model-View-Controller
- Event Based
- Process Control
- Iteration
- Map-Reduce
- Layered Systems
- Task Graphs

Computational Patterns
- Circuits
- Dense Linear Algebra
- N-Body Methods
- Sparse Linear Algebra
- Spectral Methods
- Unstructured Grids
- Graph Traversal
- Structured Grids
- Graph Algorithms
- Graphical Models
- Dynamic Programming
- FSMs

Machines

Mapping Patterns
Machine Vocabulary
Machine Vocabulary

- Machines described using a hierarchical structural decomposition

Units (processing engines)

Memories

Networks (connect multiple entities)

Channels (point-to-point connections)

(Memories, Networks, and Channels are really just specialized Units)
Hierarchy within Memory
Hierarchy within Network
Hierarchy within Network (2)
Hierarchy within Channel
Transaction Scheduler

- A single controller that executes a stream of transactions
- Each transaction is an atomic (i.e., indivisible) computation that reads and writes machine state and sends messages on inputs and outputs
- Where does this fit in taxonomy?
- How to show graphically?
Controller Types

- **State Machine Controller**
  - control lines generated by state machine

- **Microcoded Controller**
  - single-cycle datapath, control lines in ROM/RAM

- **In-Order Pipeline Controller**
  - pipelined control, dynamic interaction between stages

- **Out-of-Order Pipeline Controller**
  - operations within a control stream might be reordered internally

- **Threaded Pipeline Controller**
  - multiple control streams one execution pipeline
  - can be either in-order (PPU) or out-of-order
Leaf-Level Hardware

- Conventional schematic notation
- (Need additional notation for asynchronous logic?)
Hardware Patterns
Decoupled Units

Problem: Difficult to design a large unit with a single controller, especially when components have variable processing rates. Large controllers have long combinational paths.

Solution: Break large unit into smaller sub-units where each sub-unit has a separate controller and all channels between sub-units have some form of decoupling (i.e., no combinational path between units on each side of channel).

Applicability: Larger units where area and performance overhead of decoupling is small compared to benefits of simpler design and shorter controller critical paths.

Consequences: Decoupled channels generally have greater communication latency and area/power cost. Sub-unit controllers must cope with unknown arrival time of inputs and unknown time of availability of space on outputs. Sub-units must be synchronized explicitly.
Decoupled Units

Unless shared memory is truly multiported, channels to memory must be decoupled.

Channels to network are always decoupled in any case.
Pipelined Operator

**Problem:** Combinational function of operator has long critical path that would reduce system clock frequency. High throughput of this function is required.

**Solution:** Divide combinational function using pipeline registers such that logic in each stage has critical path below desired cycle time. Improve throughput by initiating new operation every clock cycle overlapped with propagation of earlier operations down pipeline.

**Applicability:** Operators that require high throughput but where latency is not critical.

**Consequences:** Latency of function increases due to propagation through pipeline registers, adds energy/op. Any associated controller might have to track execution of operation across multiple cycles.
Pipelined Operator

Clock → f(g(in)) → Clock

Clock → g(in) → Clock → f(in) → Clock
Multicycle Operator

**Problem:** Combinational function of operator has long critical path that would reduce system clock frequency. High throughput of this function is not required.

**Solution:** Hold input registers stable for multiple clock cycles of main system, and capture output after combinational function has settled.

**Applicability:** Operators where high throughput is not required, or if latency is critical (in which case, replicate to increase throughput).

**Consequences:** Associated controller has to track execution of operation across multiple cycles. CAD tools might detect false critical path in block.
Multicycle Operator

Clock → \( f(g(in)) \) → Clock

Clock/2 → \( f(g(in)) \) → Clock/2
Memory Patterns

- True Multiport Memory
- Banked Memory
  - Interleave lesser-ported banks to provide higher bandwidth
- Cached Memory
  - Memory hierarchy to provide higher-bandwidth, lower latency for predictable accesses
- Bypassed Memory
  - Reduce latency of pipelined dependent memory accesses
Network Patterns

- Connects multiple units using shared resources
- Bus
  - Low-cost, ordered
- Crossbar
  - High-performance
- Multi-stage network
  - Trade cost/performance
Control+Datapath

Problem:
Solution:
Applicability:
Consequences:
Machine Types

- If SCSD, SCMD, MCMD machines are patterns, what is the problem-solution?
- If they’re solutions, what’re the problems?
Examples: MPP, ICL DAP, CM-1, CM-2, MasPar, Sony Playstation-2 Graphics Engine, Vision processing chips
SCMD Shared Memory

Examples: STARAN, BSP, TI ASC, CDC
Star-100, Multi-Lane Vector Machines
MCMD Shared Memory

- Examples: Burroughs B5x00 series, Network Packet Routers
Homogeneous MCMD
Distributed Memory

Examples: Caltech Cosmic Cube, Transputer, nCube, Clusters
Heterogeneous MCMD Distributed Memory

\[ P = C + D \]

- Examples: Signal Processing Pipelines,
Systolic

\[ P = C + D \]

- Examples: Warp, Raw, Motion Estimation Engines,
Channels

- Control->Datapath
  - direct
  - pipelined? (maybe don’t need pipelined controller?)

- Datapath<->Memory
  - fixed latency
    - cannot have shared memory without true multiport
  - decoupled in-order
  - out-of-order

- Control<->Network<->Control
  - fixed latency
  - FIFOs
  - addressable messaging