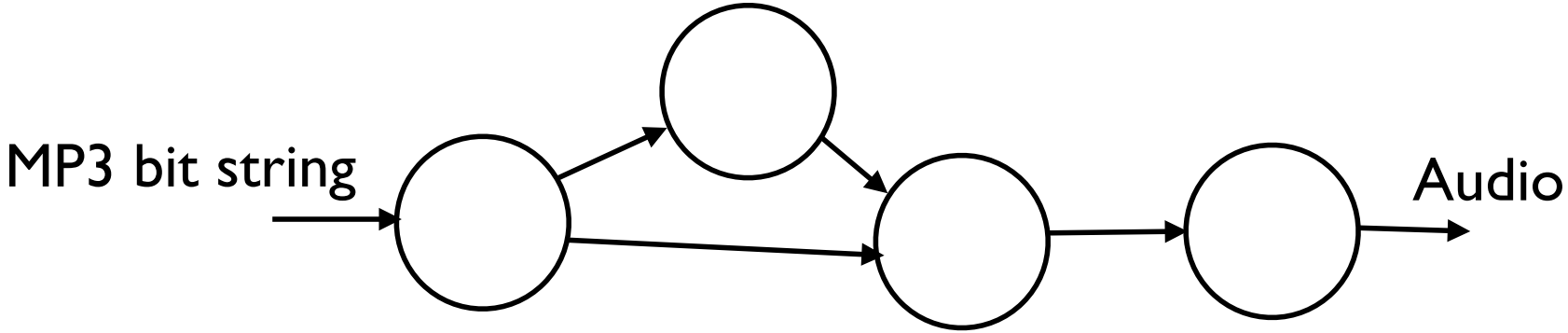
CS294-48: Hardware Design Patterns
Berkeley Hardware Pattern Language Version 0.5

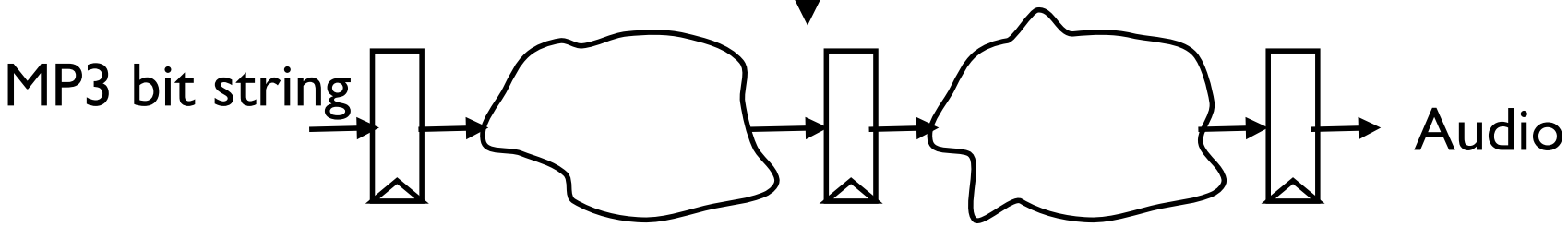
Krste Asanovic
UC Berkeley
Fall 2009

Overall Problem Statement



Application(s)

(Berkeley) Hardware Pattern Language



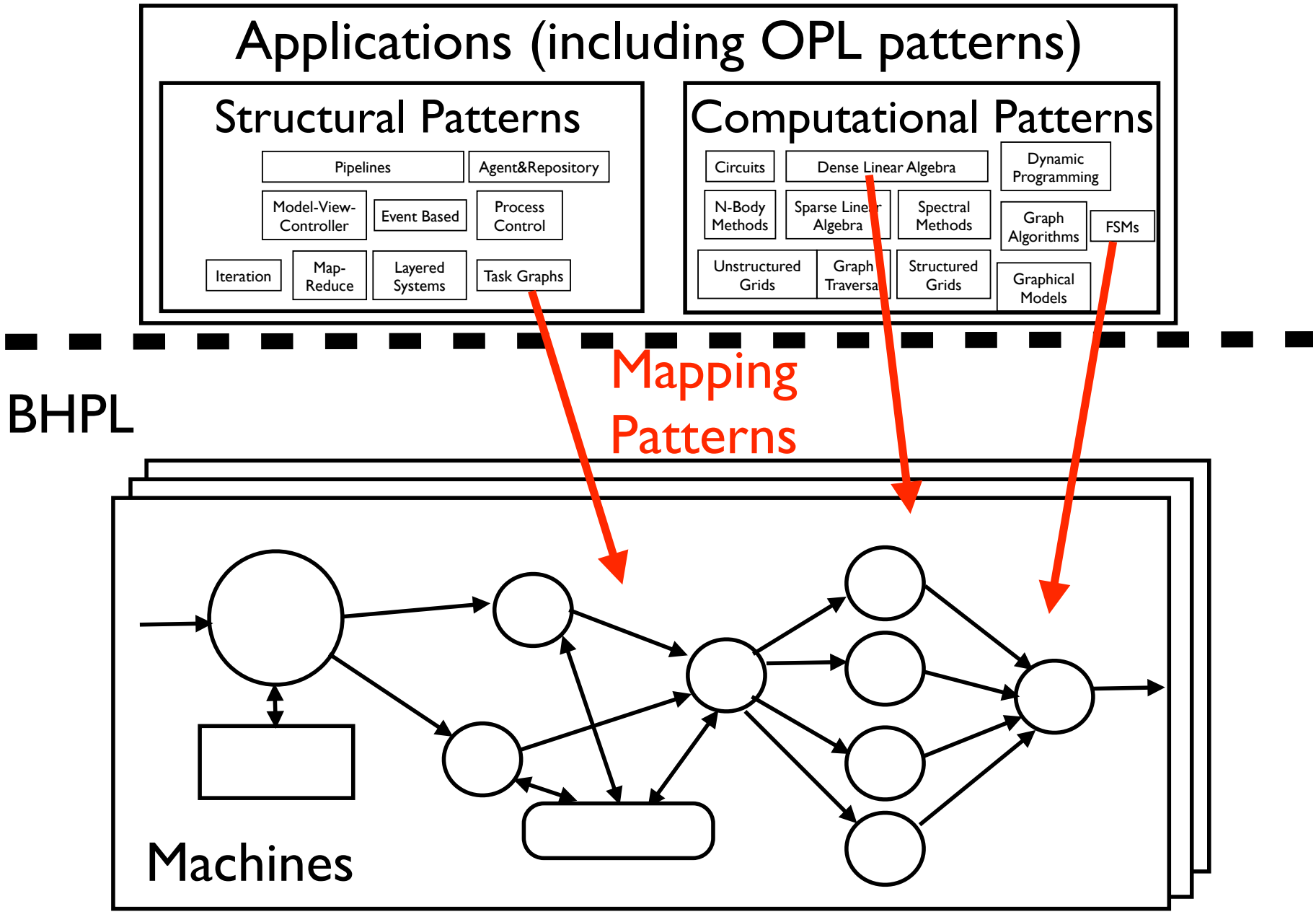
Hardware (RTL)

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



Machine Vocabulary

Why a Vocabulary?

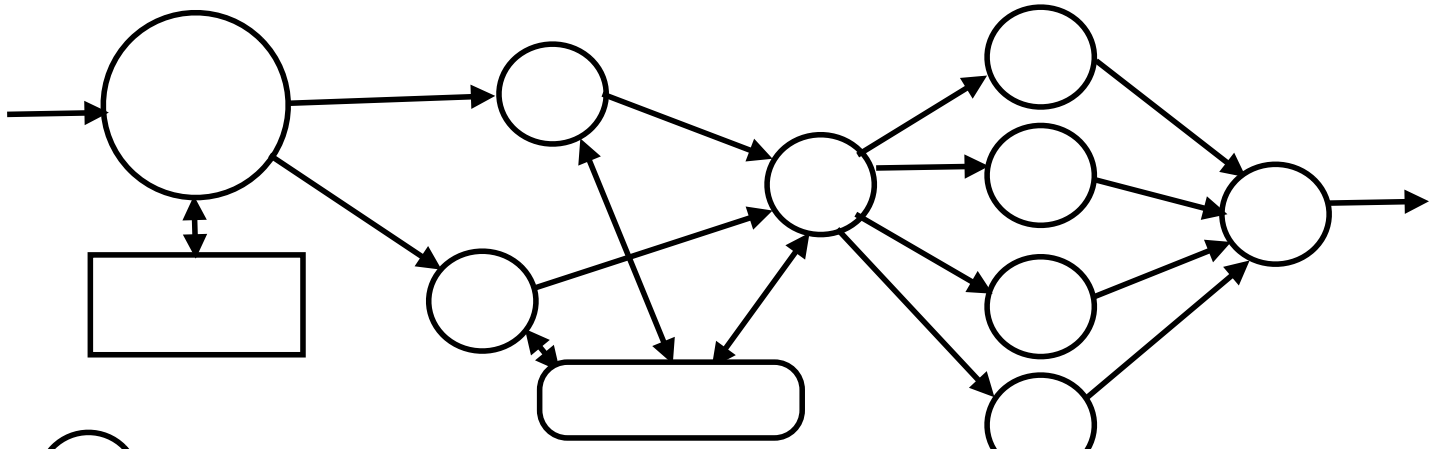
- Need a standard graphical and textual language to describe the problems and solutions in our pattern language

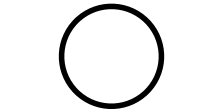
Unit-Transaction Level (UTL)

- Can describe machines abstractly as actor networks, where each actor is a "unit" that executes "transactions"
  - Useful subclasses of actor are SDF actors, Kahn process networks, and transactional actors
  - SDF << KPN << xactor in expressivity
  - SDF >> KPN >> xactor in analyzability
- This higher level description admits various mappings into RTL with different cycle timings
- These actor networks must represent machines, so must be implementable (finite)

Machine Vocabulary

- Machines described using a hierarchical structural decomposition

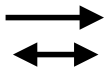




Units (processing engines)

Memories

Networks (connect multiple entities)



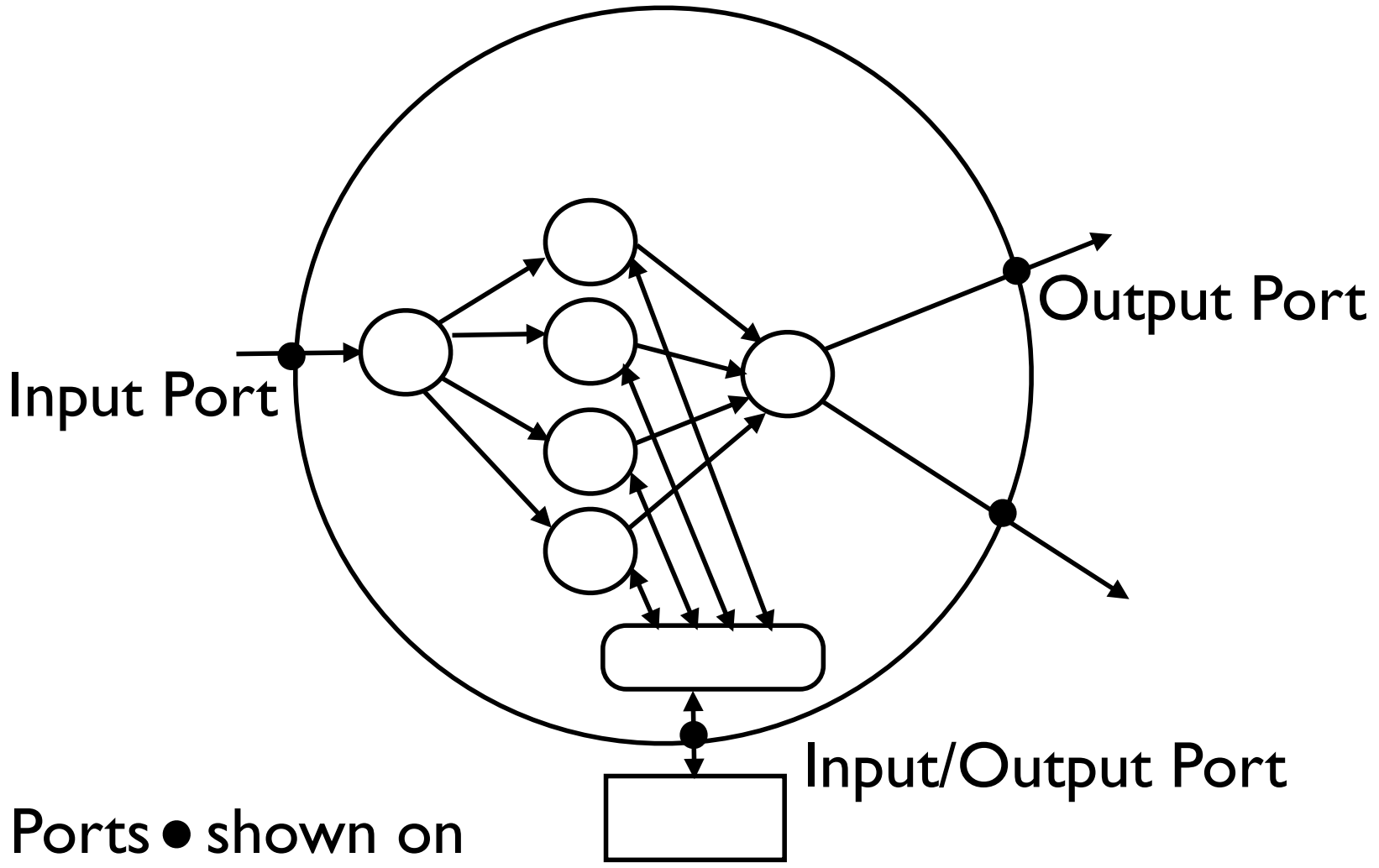
Channels (point-to-point connections)

(Memories, Networks, and Channels are just specialized Units)

Unit types

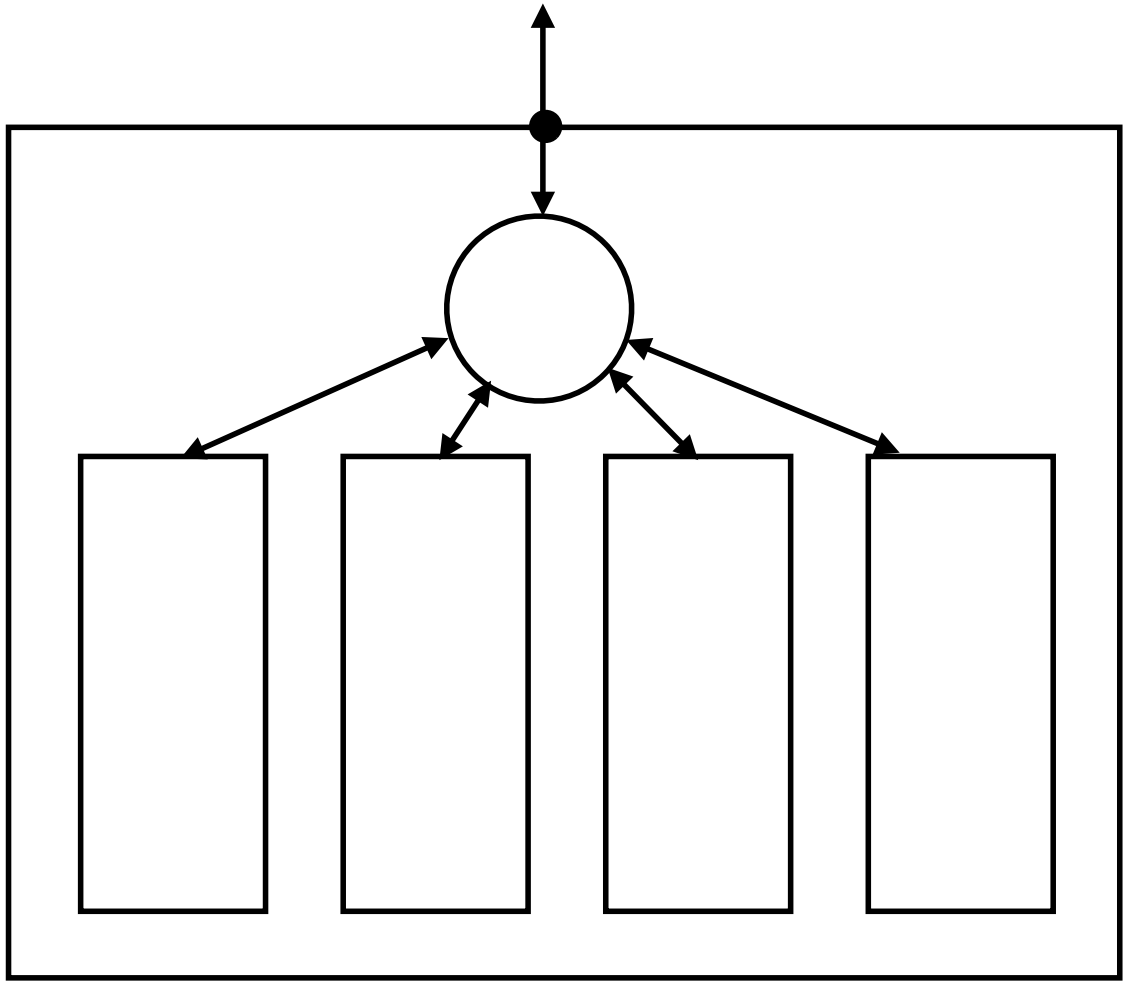
- Memories, Networks, and Channels are also units, just with specialized symbol to convey their main intended purpose.
  - Memories store data.
  - Networks connect multiple entities
  - Channels are communication paths
- High-level channels show primary direction of information flow
  - Might have wires in other direction to handle flow-control etc.

Hierarchy within Unit

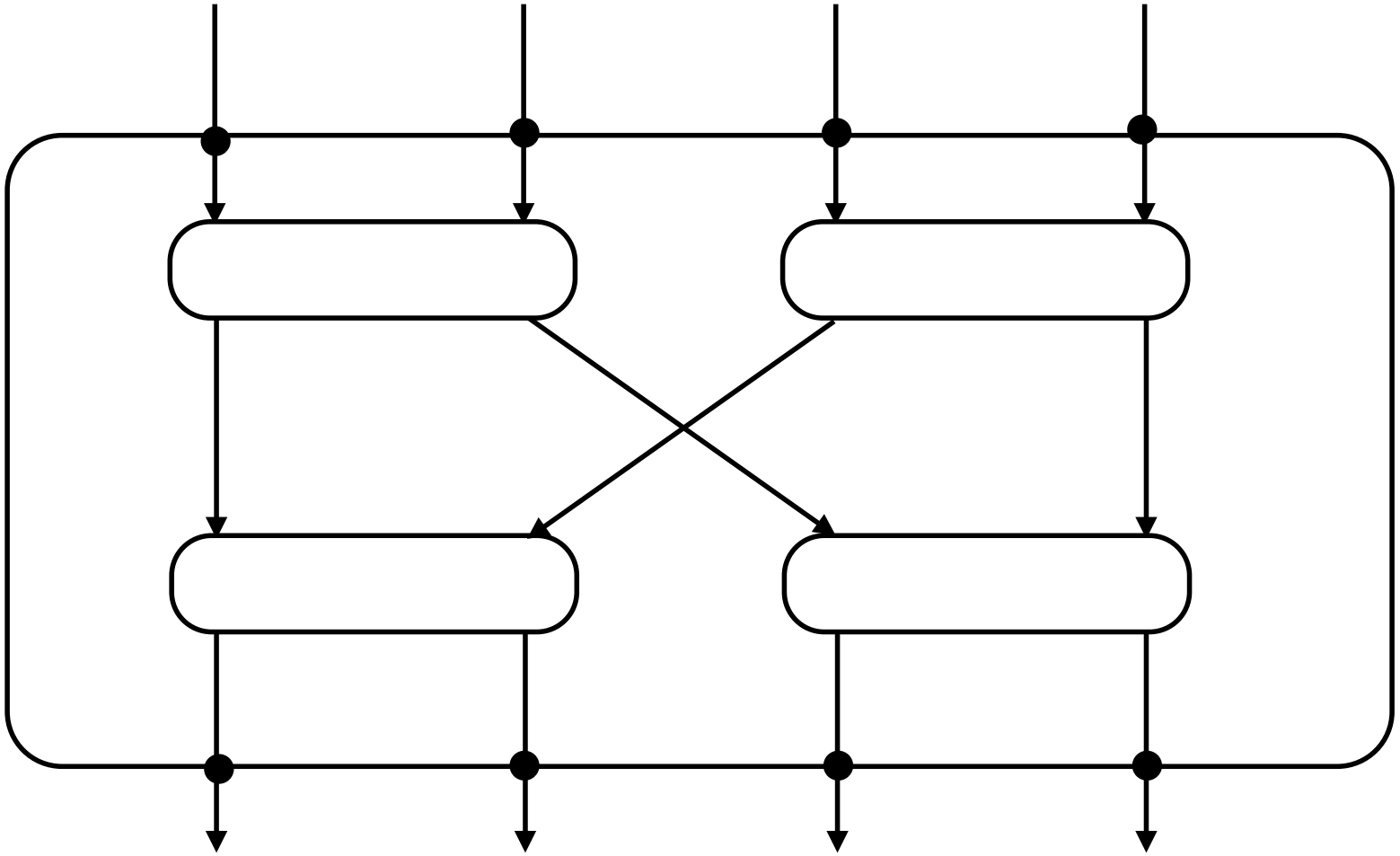


Ports ● shown on edge of unit

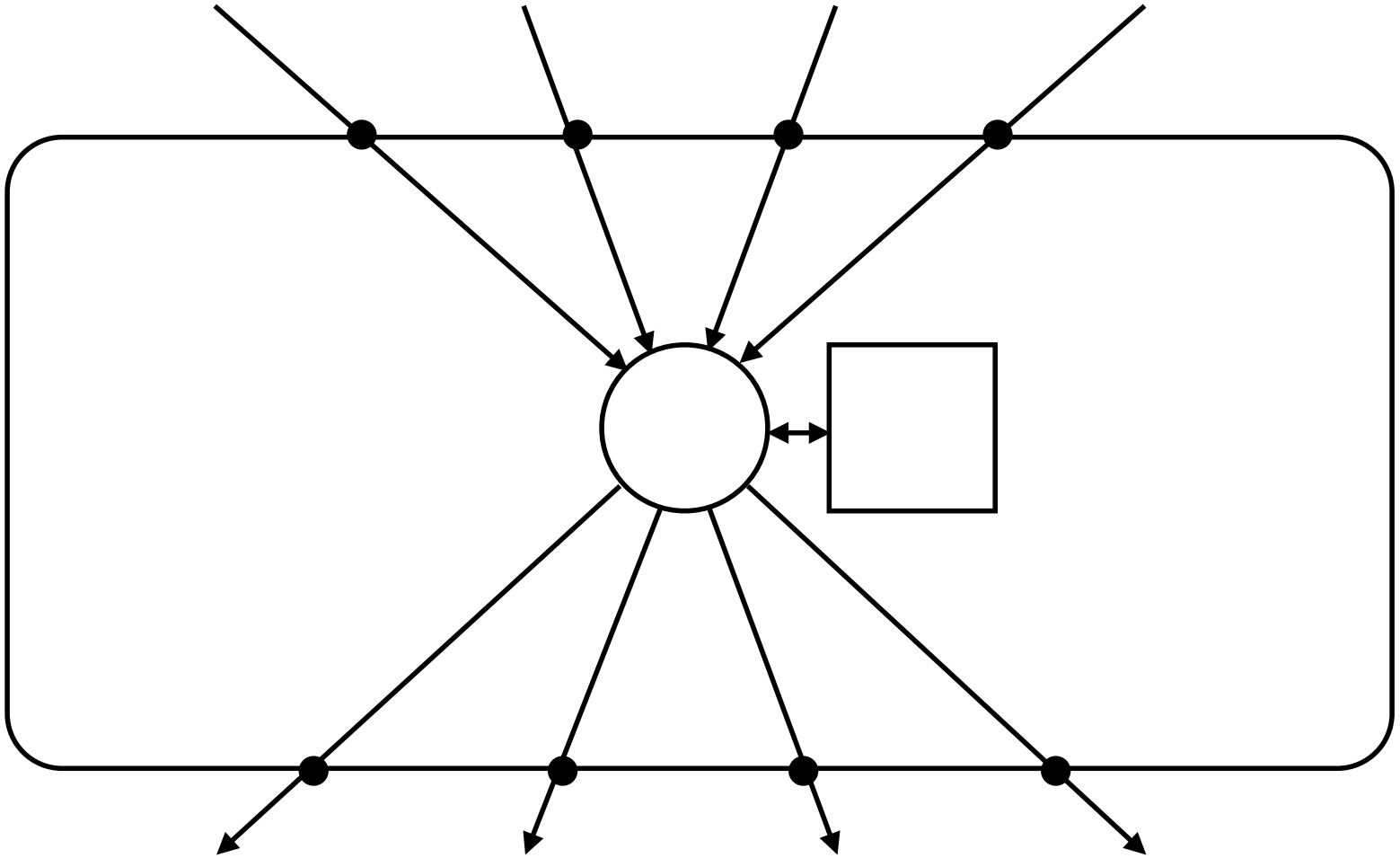
Hierarchy within Memory



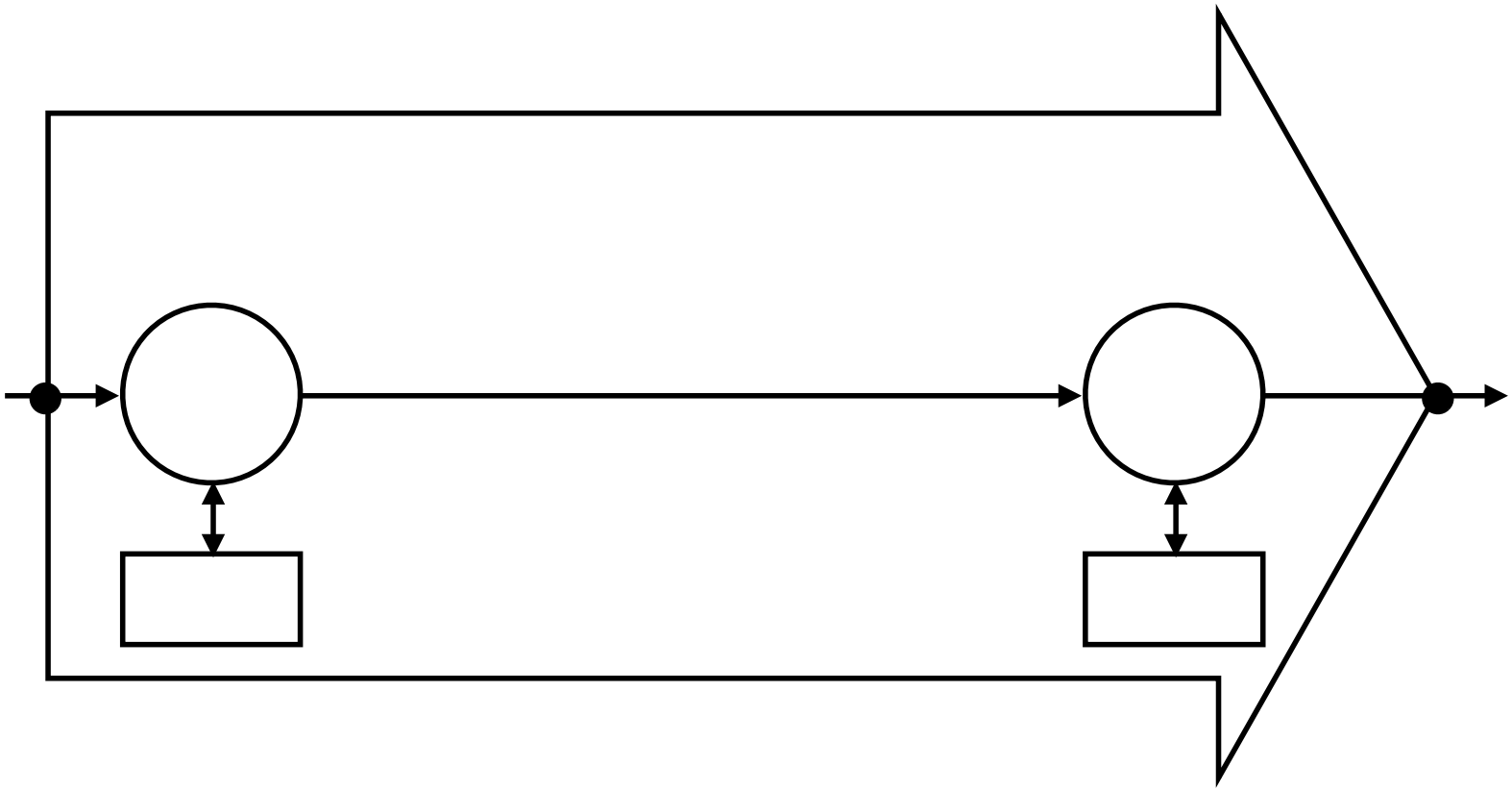
Hierarchy within Network



Hierarchy within Network (2)



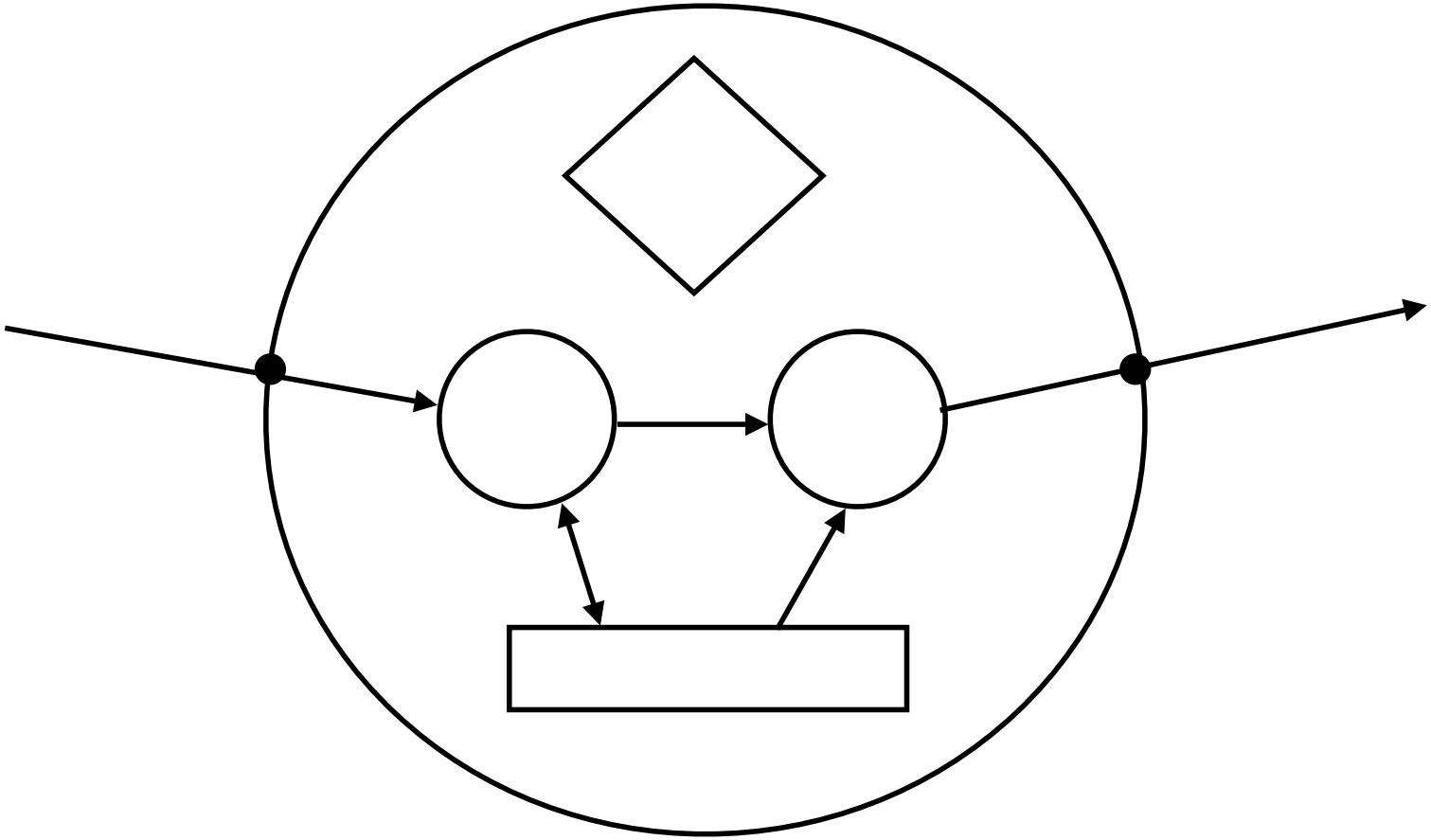
Hierarchy within Channel



Eventually refine UTL to RTL

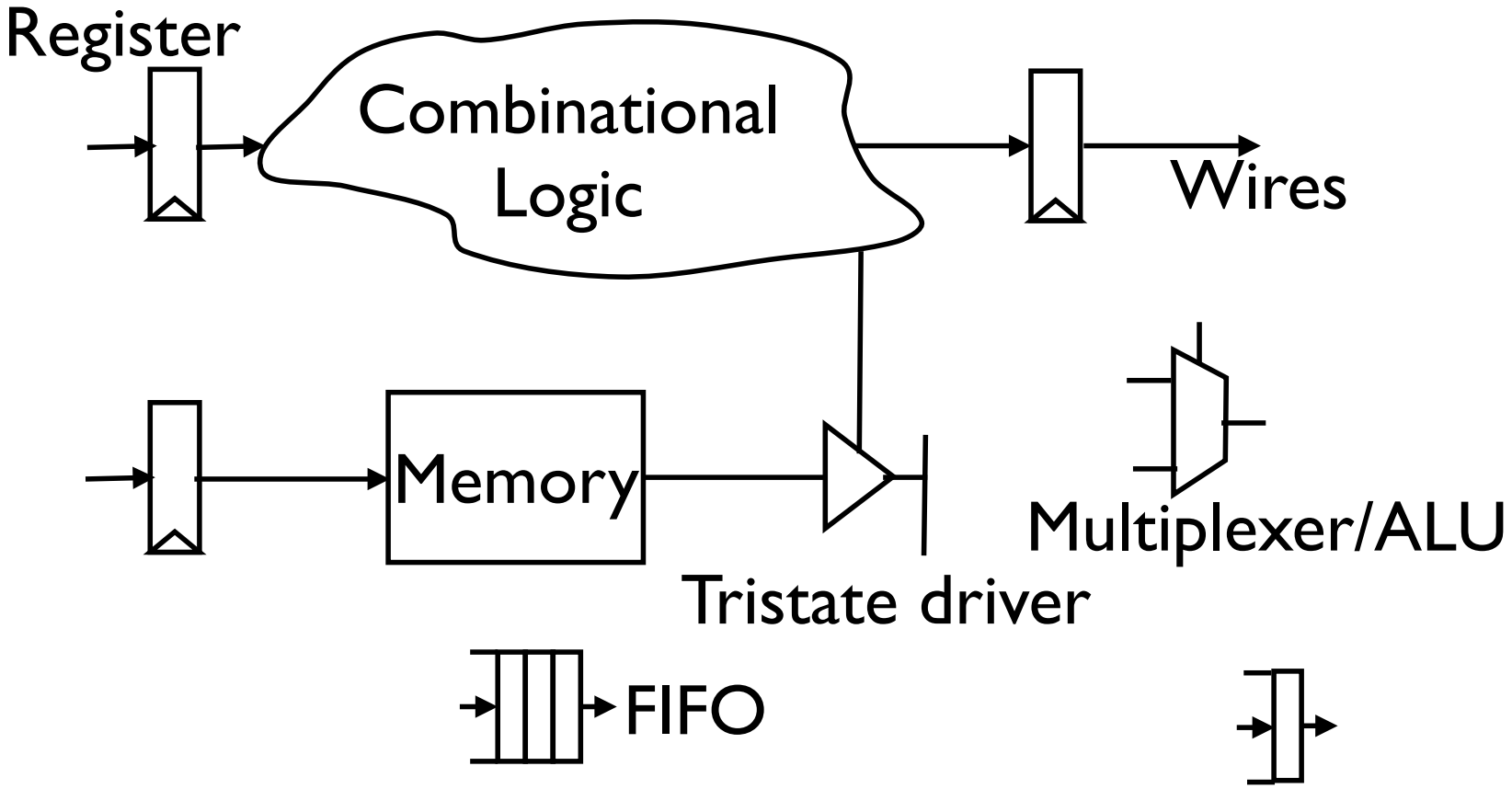
- All units are eventually mapped into digital hardware, i.e., ultimately describable as a finite-state machine (FSM)
- Different ways of factoring out the FSM description of a unit
  - Structural decomposition into hierarchical sub-units
  - Decompose functionality into control + datapath
- All factorings are equivalent, so pick factoring that best explains what unit does

Unit with Single Controller



Diamond represents unit-level controller, implied control connections to all other components (datapaths+memories) in unit

Leaf-Level Hardware



- Conventional schematic notation
- (Need additional notation for asynchronous logic?)

Hardware Pattern Language

BHPL 0.5 Overview

Applications (including OPL patterns)

BHPL

App-to-UTL Mappings Layer

FFT to SIMD array

Problem: Application Computation
Solution: UTL Machine

UTL-to-UTL Transformation Layer

Time-Multiplexing

Unrolling

Problem: UTL violates constraint (too big, too slow)
Solution: Transformed UTL

UTL-to-RTL Transformation Layer

Microcoded Engine

In-Order Pipeline Engine

Problem: UTL design
Solution: RTL behavior

RTL-to-Technology

Interleaved Memory

FIFO

CAM

Problem: RTL behavior
Solution: Structural design

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.

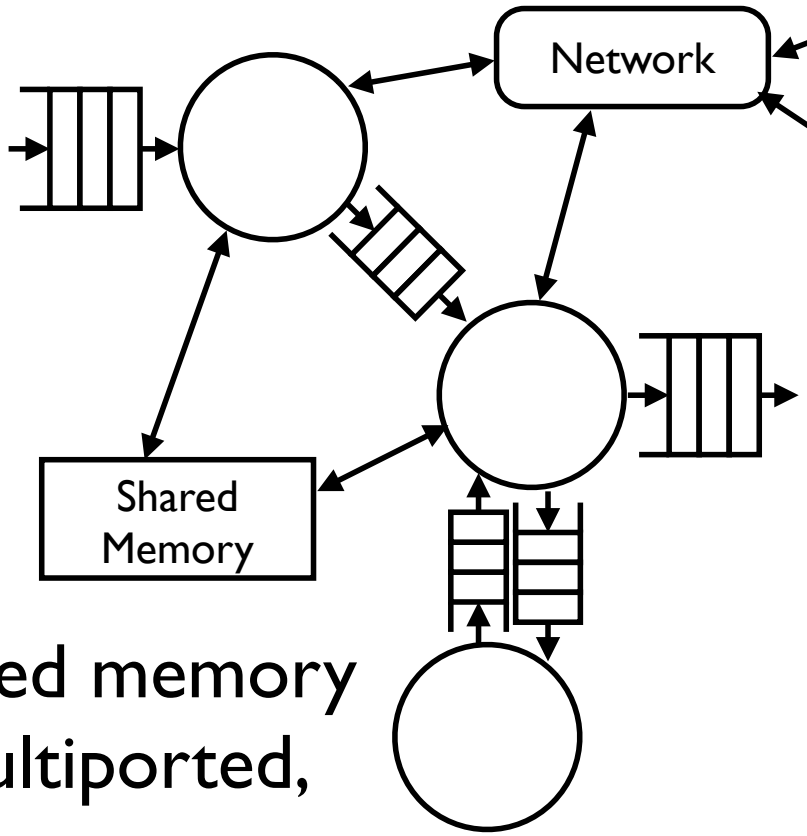
Solution: Break large unit into smaller sub-units where each sub-unit has a separate controller and all channels between sub-units have some form of decoupling (i.e., no assumption about timing of sub-units).

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

Channels to network are usually decoupled in any case



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

Pipelined Operator

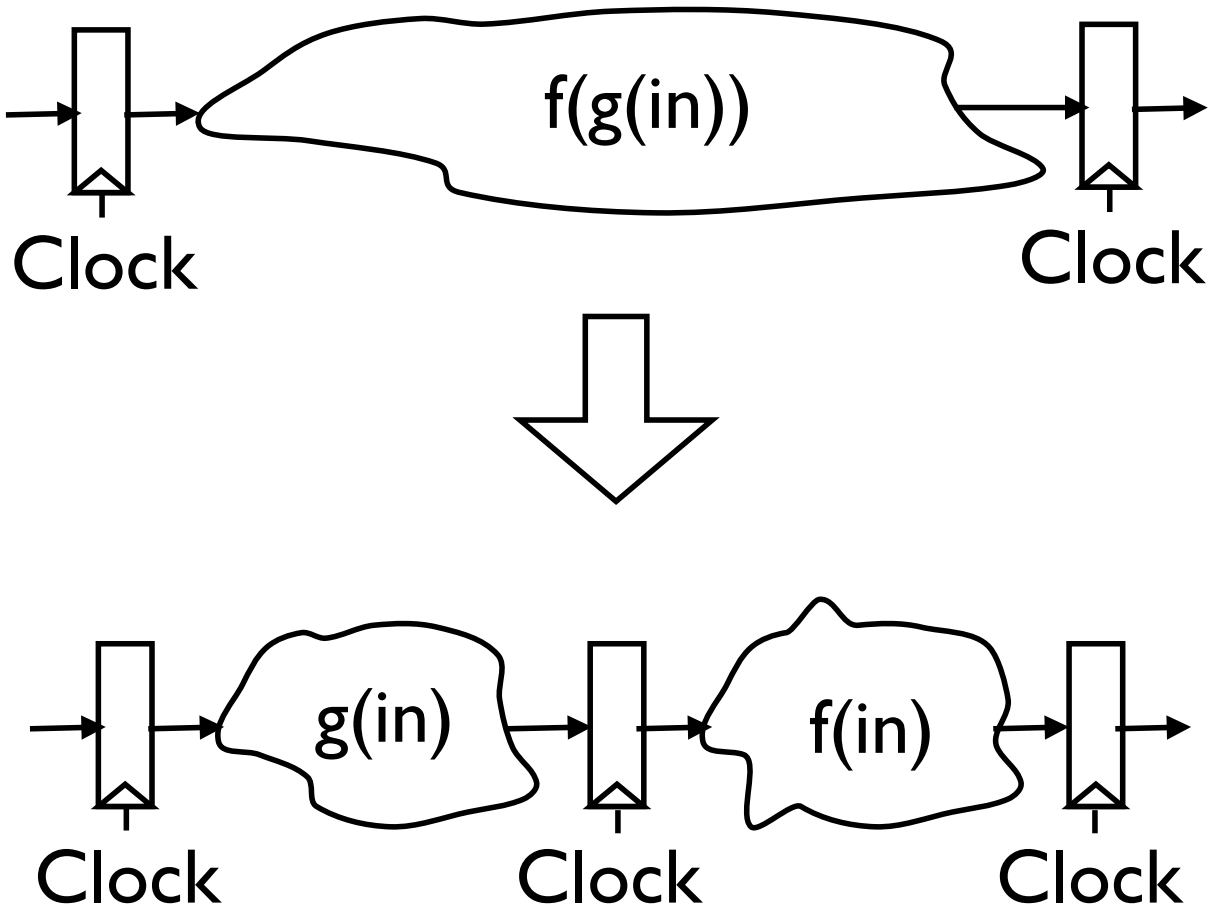
Problem: Combinational function of operator has long critical path that would reduce system clock frequency. High throughput of this function is required.

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



Multicycle Operator

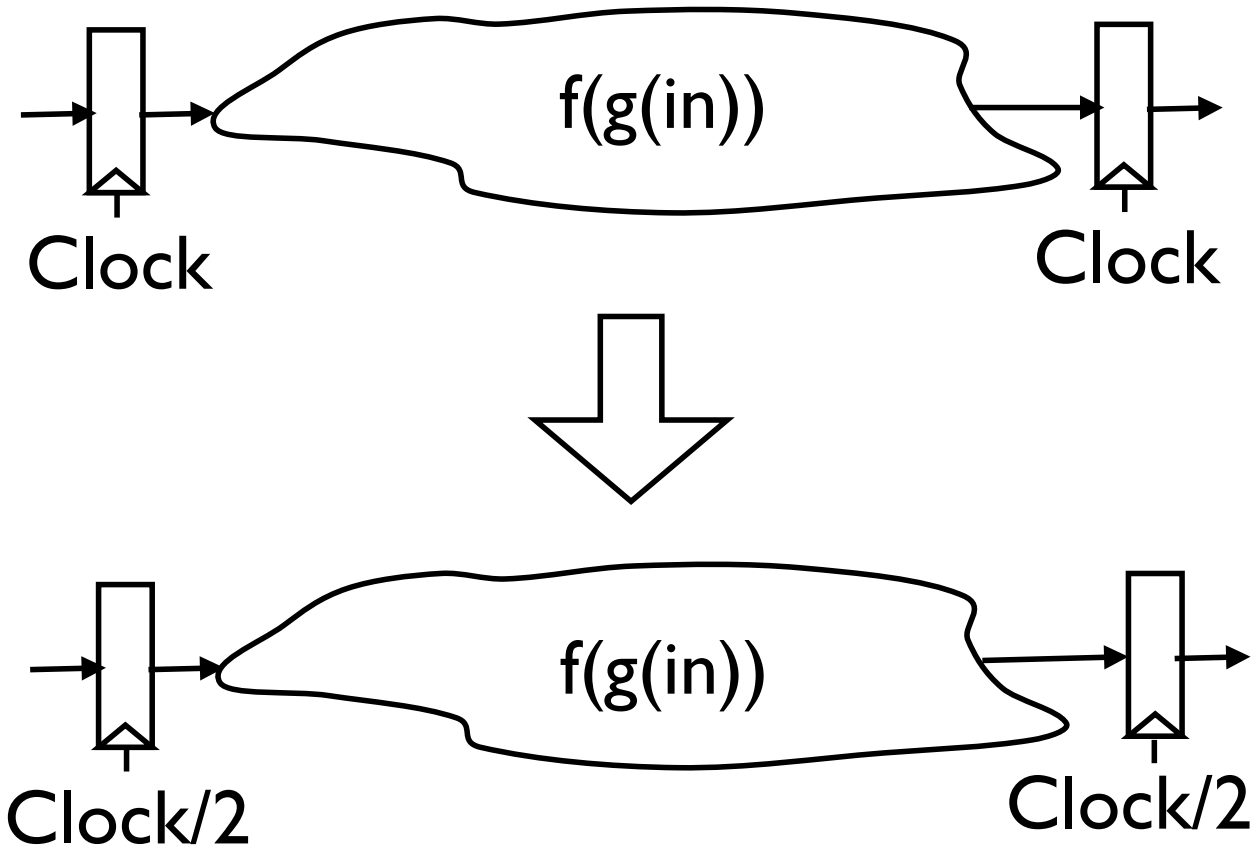
Problem: Combinational function of operator has long critical path that would reduce system clock frequency. High throughput of this function is not required.

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



Controller Patterns

- 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

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