ESE5320: System-on-a-Chip Architecture

Day 4: September 11, 2024

Parallelism Overview

Board holders pickup boards

Preclass

Today

- Compute Models (Part 1)
  - How do we *express* and reason about parallel execution freedom
- Types of Parallelism (Part 2)
  - How can we slice up and think about parallelism?
  - How *exploit* parallelism

Message

- Many useful models for parallelism
  - Help conceptualize
- One-size does not fill all
  - Match to problem
  - Will want to exploit all of them

Parallel Compute Models

Control Flow, Dataflow

Combining

Explicit, Implicit Parallelism

Term: Operation

- **Operation** – logic computation to be performed

Sequential Control Flow

Control flow

- Program is a sequence of operations
- Operation reads inputs and writes outputs into common store (memory)
- One operation runs at a time
  - defines successor

Model of correctness is sequential execution

Examples

- C (Java, …)
- Finite-State Machine (FSM)
- Finite Automata (FA)
- assembly code (ISA)

Parallelism can be explicit

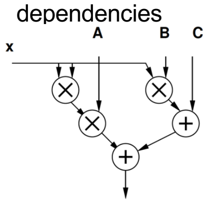
- State which operations occur on a cycle

- Multiply, add for quadratic equation

cycle	mpy	add
1	B,x	
2	x,x	(Bx)+C
3	A,x^2	
4		Ax^2+(Bx+C)

Parallelism can be implicit

- Sequential expression
- Infer data dependencies



T1=x*x
T2=A*T1
T3=B*x
T4=T2+T3
Y=C+T4

- Or

Y=A*x*x+B*x+C

Implicit Parallelism

- d=(x1-x2)*(x1-x2) + (y1-y2)*(y1-y2)

- What parallelism exists here?

Parallelism can be implicit

- Sequential expression
- Infer data dependencies

```
for (i=0;i<100;i++)
   y[i]=A*x[i]*x[i]+B*x[i]+C
```

Why can these operations be performed in parallel?

Dataflow / Control Flow

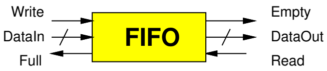
Dataflow
- Program is a graph of operations
- Operation consumes **tokens** and produces tokens
- All operations run concurrently

Control flow (e.g. C)
- Program is a sequence of operations
- Operation reads inputs and writes outputs into common store
- One operation runs at a time
  - defines successor

Token

- Data value with presence indication
  - May be conceptual
    - Only exist in high-level model
    - Not kept around at runtime
  - Or may be physically represented
    - One bit represents presence/absence of data

FIFO



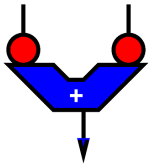
- Hardware Block
- Outputs data in order received
  - First-In, First-Out
- Tell it when you are providing data
  - Write
  - May choose not to insert on a cycle
    - Need to signal
- Tell it when you are consuming data
  - Read
- Tells you when it’s **empty** and has no data to provide
- Tells you when it’s **full** and can hold nothing else

What are data presence indicators here?

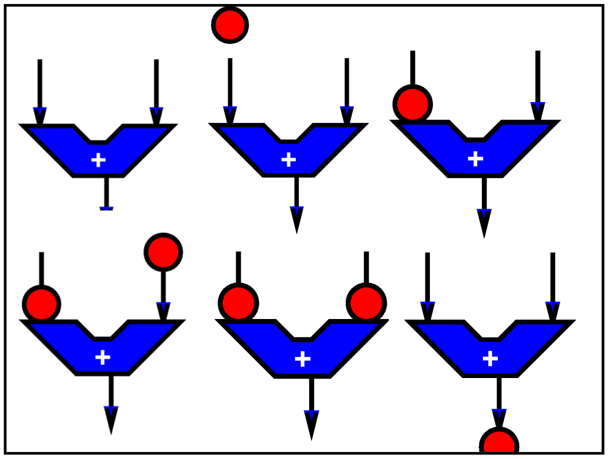
Token Examples?

- How signal miss in processor data cache and processor needs to wait for data?

Operation

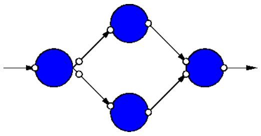


- Takes in one or more inputs
- Computes on the inputs
- Produces results
- Logically **self-timed**
  - “Fires” only when input set present
  - Signals availability of output

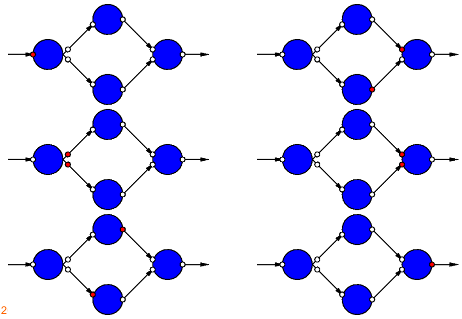


Dataflow Graph

- Represents
  - computation sub-blocks
  - linkage
- Abstractly
  - controlled by data presence



Dataflow Graph Example



Dataflow / Control Flow

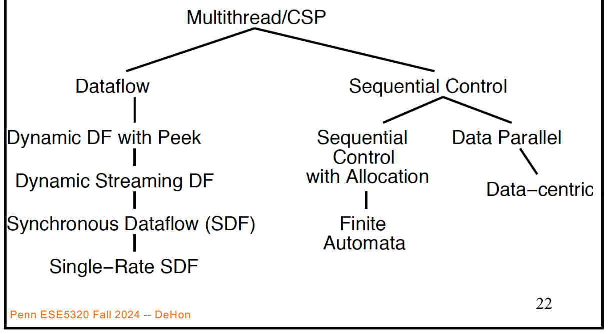
Dataflow
- Program is a graph of operations
- Operation consumes **tokens** and produces tokens
- All operations run concurrently

Control flow (e.g. C)
- Program is a sequence of operations
- Operation reads inputs and writes outputs into common store
- One operation runs at a time
  - defines successor

Communicating Threads

- Computation is a collection of sequential/control-flow “threads”
- Threads may communicate
  - Through dataflow I/O
  - (Through shared variables)
- View as hybrid or generalization
  - Of control flow and dataflow
- CSP – Communicating Sequential Processes → canonical model example

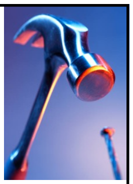
Compute Models



All Used

- All of these things get used in modern CPUs and SoCs
  - Sequential control flow
  - Operation parallelism
  - Data presence and data-driven flow
  - Multiple threads
  - Data Parallel

Value of Multiple Models



- When you have a big enough hammer, everything looks like a nail.
- Many stuck on single model
  - Try to make all problems look like their nail
- Value to diversity / heterogeneity
  - One size does not fit all

Types of Parallelism

Part 2

Types of Parallelism

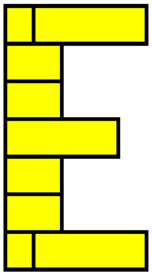
- **Data Level** – Perform same computation on different data items
- **Thread or Task Level** – Perform separable (perhaps heterogeneous) tasks independently
- **Instruction Level** – Within a single sequential thread, perform multiple operations on each cycle.

Pipeline Parallelism

- Pipeline – organize computation as a spatial sequence of concurrent operations
  - Can introduce new inputs before finishing
  - Instruction- or thread-level
  - Use for data-level parallelism
  - Can be directed graph

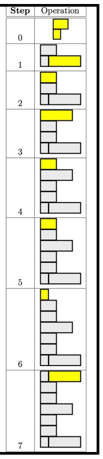
Sequential

- Single person build E
- Latency?
- Throughput?



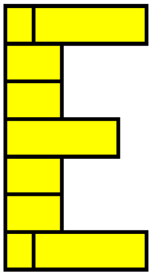
Data Parallel

- Everyone with Legos build own E



Data Parallel

- Everyone in class build own E
- Latency?
- Throughput?
- Ideal speedup?
- Resource Bound?
  - 100 Es, 12 people
- When useful?



Data-Level Parallelism

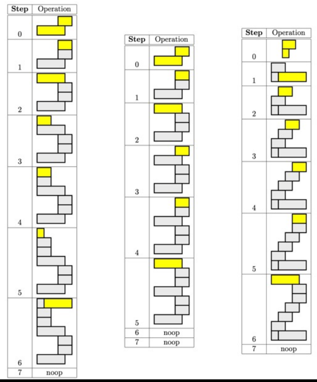
- **Data Level** – Perform same computation on different data items
- Resource Bound: $T_{dp} = T_{seq}/P$
- (with enough independent problems, match our resource bound computation)

Thread Parallel

- Each person build distinct letter or number (e.g. E, S, 5, 3, 2, 0)

Thread Parallel

Likely get
3 volunteers
to help demo.



Thread Parallel

- Each person build distinct letter or number (e.g. E, S, 5, 3, 2, 0)
- Latency? (assume each has <=9 bricks)
- Throughput?
  - Build 6 distinct letters
  - Using 6 people
  - (distinct letters/time-unit)
- Speedup over sequential build of 6 letters?

Thread-Level Parallelism

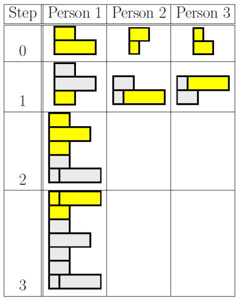
- **Thread or Task Level** – Perform separable (perhaps heterogeneous) tasks independently
- Resource Bound: $T_{tp} = T_{seq}/P$
- $T_{tp}=\max(T_{t1},T_{t2},T_{t3},\ldots)$
  - Less speedup than ideal if not balanced
- Can produce a diversity of calculations
  - Useful if have limited need for the **same** calculation

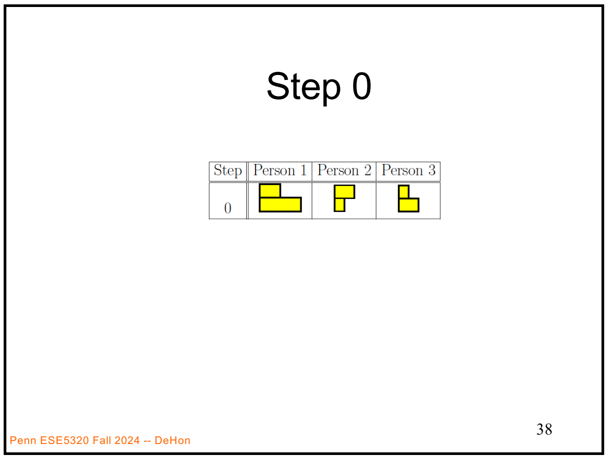
Instruction-Level Parallelism

- Build single letter in lock step
- Group of 3
  - [3 volunteers; steps up front]
- Resource Bound for 3 people building 9-brick letter?
- Announce steps from slide
  - Stay in step with slides

Group Communication

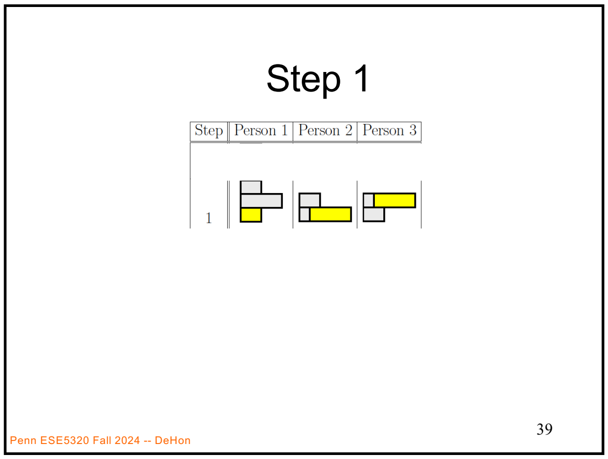
- Groups of 3
- Note who was person 1 task
- 2, 3 will need to pass completed substructures





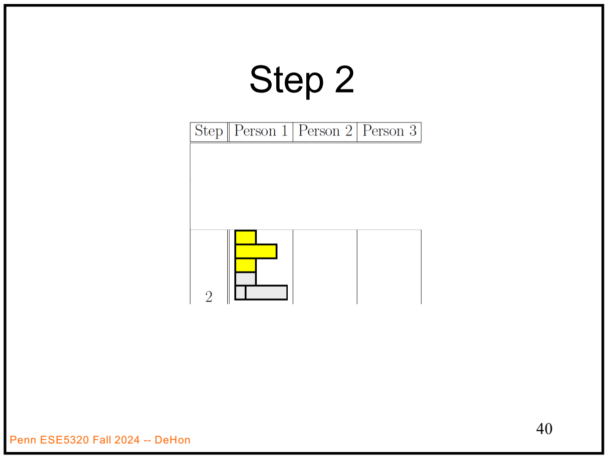
Step 0

Step	Person 1	Person 2	Person 3
0			



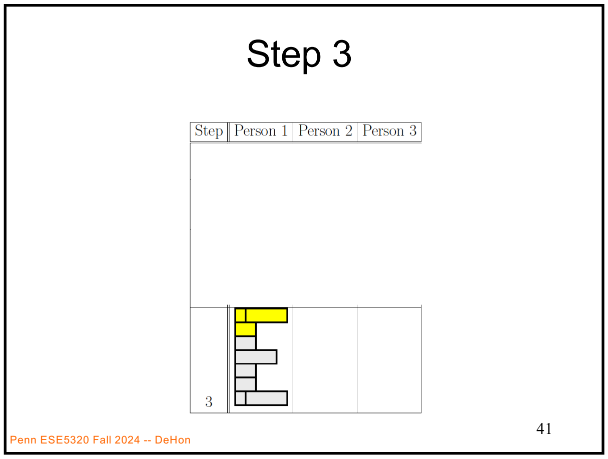
Step 1

Step	Person 1	Person 2	Person 3
1			



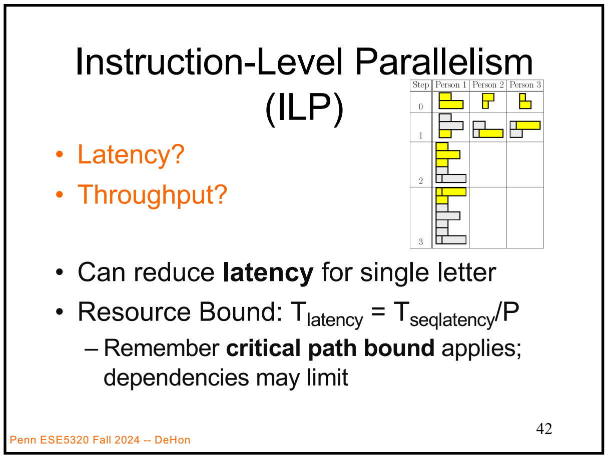
Step 2

Step	Person 1	Person 2	Person 3
2			



Step 3

Step	Person 1	Person 2	Person 3
3			

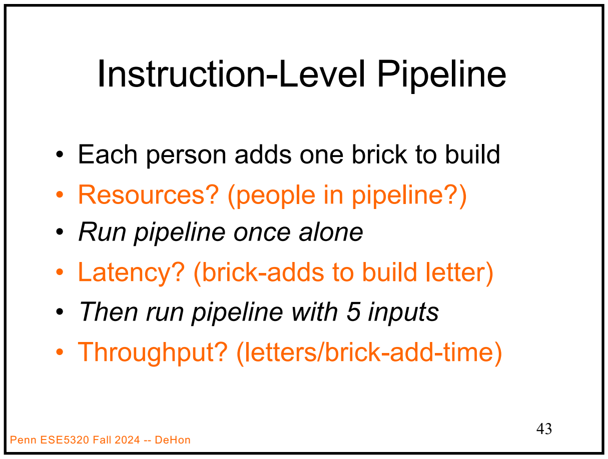


Instruction-Level Parallelism (ILP)

- Latency?
- Throughput?

Step	Person 1	Person 2	Person 3
0			
1			
2			
3			

- Can reduce **latency** for single letter
- Resource Bound: $T_{latency} = T_{seqlatency}/P$
  - Remember **critical path bound** applies; dependencies may limit

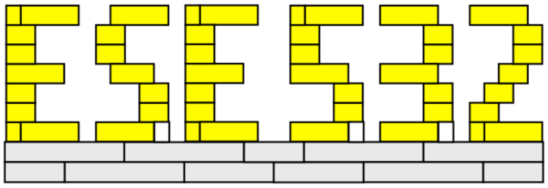


Instruction-Level Pipeline

- Each person adds one brick to build
- Resources? (people in pipeline?)
- *Run pipeline once alone*
- Latency? (brick-adds to build letter)
- *Then run pipeline with 5 inputs*
- Throughput? (letters/brick-add-time)

Thread Graph

- How would we build with task level parallelism?
  - Tasks?
  - Dependencies?



Types of Parallelism

- **Data Level** – Perform same computation on different data items
- **Thread or Task Level** – Perform separable (perhaps heterogeneous) tasks independently
- **Instruction Level** – Within a single sequential thread, perform multiple operations on each cycle.

Pipeline Parallelism

- Pipeline – organize computation as a spatial sequence of concurrent operations
  - Can introduce new inputs before finishing
  - Instruction- or thread-level
  - Use for data-level parallelism
  - Can be directed graph

Big Ideas

- Many parallel compute models
  - Sequential, Dataflow, CSP
- Find natural parallelism in problem
- Mix-and-match
- Likely to need all of them at some point

Admin

- Reading Day 5 on web
- HW2 due Friday
- HW3 out
  - Including partner assignments on canvas
  - Board Holder reach out to partner ASAP