ESE532: System-on-a-Chip Architecture

Day 3: January 23, 2017
Parallelism Overview

Today

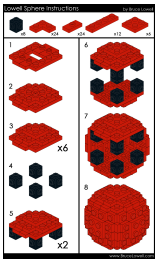
- Parallelism in Tasks
- Types of Parallelism
- Compute Models
- System Architectures

Message

- Many useful models for parallelism
  - Help conceptualize
- One-size does not fill all
  - But maybe 6—10 do?
  - Match to problem

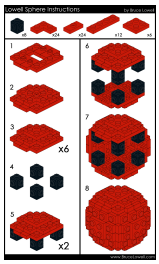
Preclass 1

- How do 6 people collaborate on sphere building?



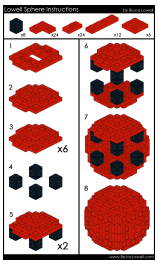
Preclass 2

- How do 12 people collaborate on sphere building?



Preclass 3

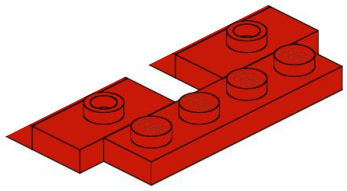
- How do 6 people collaborate on building 3 spheres?
- (alternate solution?)



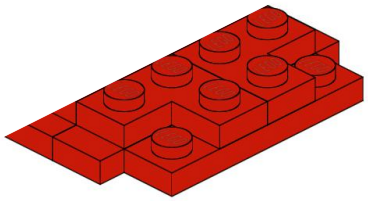
In Class Exercise

- Distribute 24 piece sets for building Red and Yellow Sphere
  - [if have more than 24 people, have pairs build a different model]
- Follow instructions from slides to come

Step 1: Build half of L1



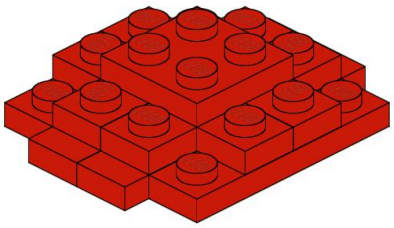
Step 2: Build half of L2



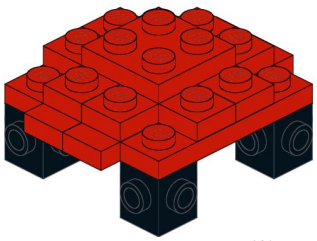
Step 3:

- Pass half to builder with 2x2 plate

Step 4: Build L3



Step 5: Build L5 (ends)



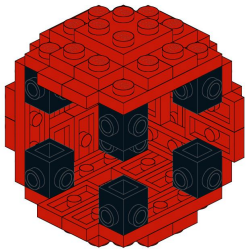
(if have pieces)

Step 6:

- Pass both “L5: ends” to builder with side

Step 7: half of L7

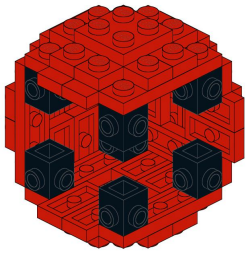
Install one side



Step 8:

- Pass assemble to builder with unused side

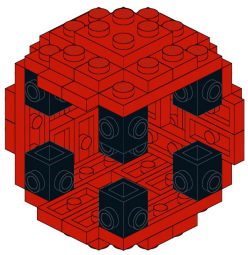
Step 9: finish L7



Step 10:

- Pass assemble to builder with unused side

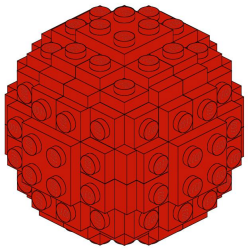
Step 11: add 3rd side



Step 12:

- Pass assemble to builder with unused side

Step 13: add final side



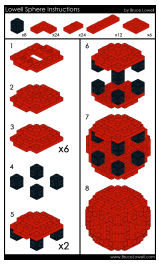
Finish

- Check status of all builds

Types of Parallelism

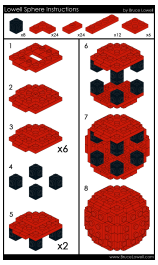
Types of Parallelism

- What kind of parallelism did we see for steps 1—3?



Types of Parallelism

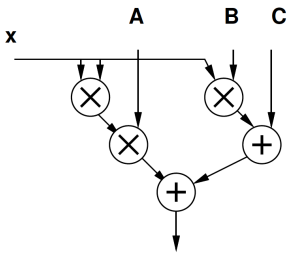
- What parallelism when some folks built different model?



Types of Parallelism

- What could we build independently here?
- Kind of parallelism?

Type of Parallelism



- Latency multiply = 1
- Latency add = 1
- (different Day2)

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

Kind of Parallelism?

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.

Parallel Compute Models

Sequential Control Flow

Control flow

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

Model of correctness is sequential execution

Examples

C (Java, …)

FSM / FA

Can be explicit

- Sphere Build example Step 2
- Coordinate data parallel operations
- Multiply, add for quadratic equation

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

- Coordinate ILP

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
```


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
```


Term: Operation

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

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

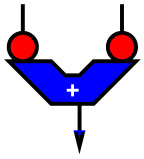
- Data value with presence indication
  - May be conceptual
    - Only exist in high-level model
    - Not kept around at runtime
  - Or may be physically represented
    - One bit represents presence/absence of data

Token Examples?

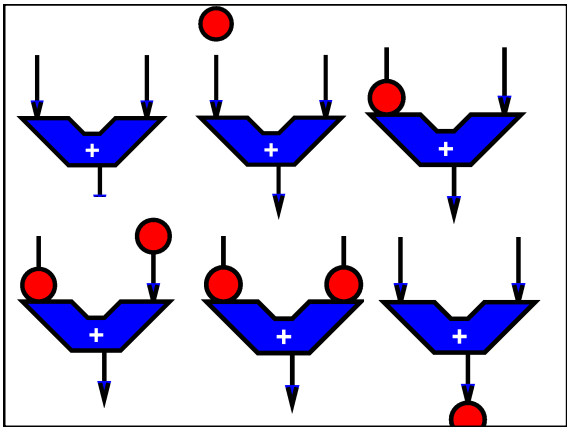
- What are familiar cases where data may come with presence tokens?
  - Network packets
  - Memory references from processor
    - Variable latency depending on cache presence
  - Start bit on serial communication

Operation

- Takes in one or more inputs
- Computes on the inputs
- Produces results

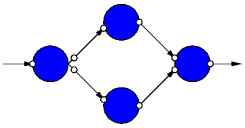


- Logically **self-timed**
  - "Fires" only when input set present
  - Signals availability of output

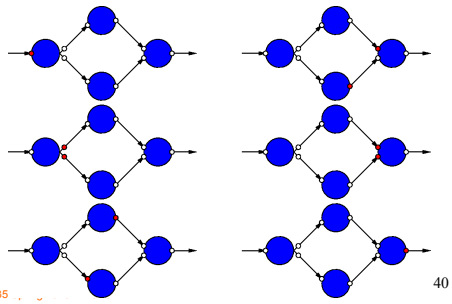


Dataflow Graph

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



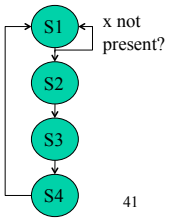
Dataflow Graph Example



Sequential / FSM

- FSM is degenerate dataflow graph where there is exactly one token

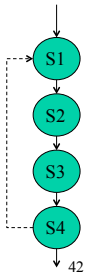
cycle	mpy	add	next
S1	B,x		x-->S2, else S1
S2	x,x	(Bx)+C	S3
S3	A,x^2		S4
S4		Ax^2+(Bx+C)	S1



Sequential / FSM

- FSM is degenerate dataflow graph where there is exactly one token

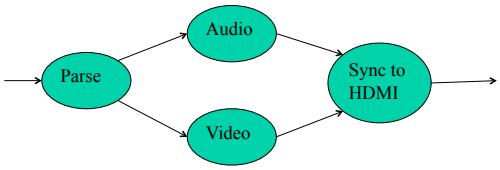
cycle	mpy	add	next
S1	B,x		x-->S2, else S1
S2	x,x	(Bx)+C	S3
S3	A,x^2		S4
S4		Ax^2+(Bx+C)	S1



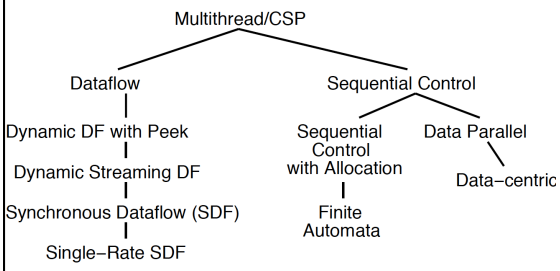
Communicating Threads

- Computation is a collection of sequential/control-flow "threads"
- Threads may communicate
  - Through dataflow I/O
  - (Through shared variables)
- View as hybrid or generalization

Video Decode



Compute Models



System Architectures

System Architecture Hypothesis

- There are a small number of useful system architectures
- These architectures
  - Give guidance for organizing resources
  - Make manageable
  - Allow share lessons between applications
  - Provide basis for scalability
  - Point toward efficient solutions

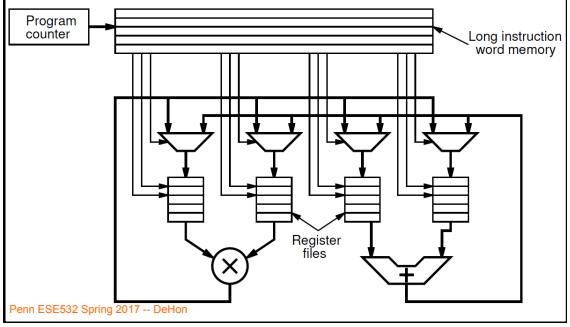
Unconstrained Model

- Multithreaded programming (equivalently Communicating Sequential Processes)
  - Application is collection of threads
  - Communicate with each other
  - May or may not have shared memory
  - Programmer responsible for
    - Synchronization
    - Parallelism
    - Data layout
    - Communications…

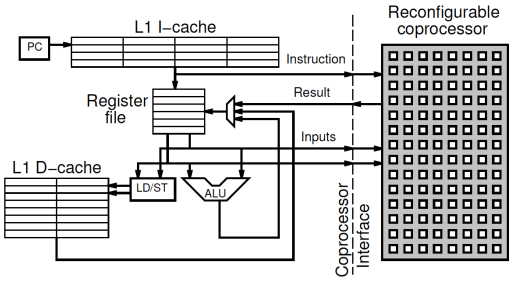
Architectural Restrictions

- Sequential Control
  - Data Parallel → all parallel processing does the same thing
  - Lock-Step → all parallel processing does different things at synchronized time (e.g. VLIW)
  - Bulk Synchronous → periodic barrier synchronization
  - Instruction Augmentation – control accelerators from seq. instruction stream

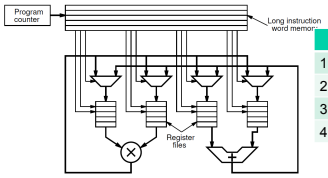
Very Long Instruction Word (VLIW)



Instruction Augmentation Co-Processor



Very Long Instruction Word (VLIW)

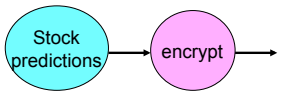


cycle	mpy	add
1	B,x	
2	x,x	(Bx)+C
3	A,x^2	
4		$Ax^2+(Bx+C)$

Architectural Restrictions (2)

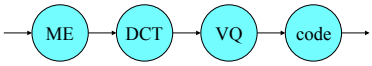
- Dataflow interactions
  - Allow multithreaded operation
  - Use data presence for synchronization
- E.g.
  - Pipe-and-filter / Streaming Dataflow
  - Synchronous Dataflow (SDF)

Producer-Consumer Parallelism



- Can run concurrently
- Just let consumer know when producer sending data

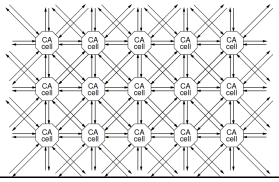
Pipeline Parallelism



- Can potentially all run in parallel
- Like **physical** pipeline
- Useful to think about **stream** of data between operators

Architectural Restrictions (3)

- Regular Communication Patterns
  - Systolic
  - Cellular Automata → regular grid of homogeneous FSMs

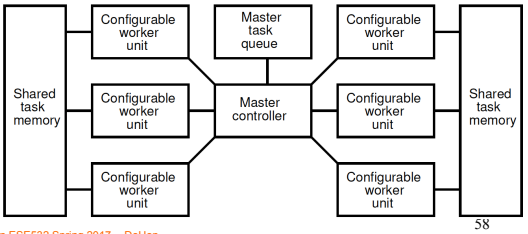


Architectural Restrictions (4)

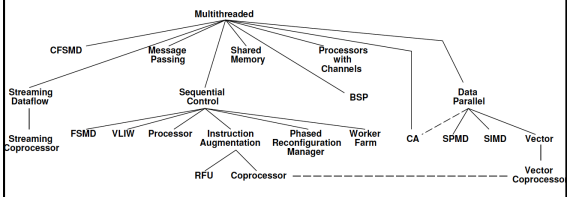
- Memory/Data Centric
  - Computation is collection of objects in memory
  - Each object triggered by input changes
  - Compute and potentially trigger other objects
- E.g.
  - Repository models
  - GraphStep
  - App: network flow, routing…

Work Farm

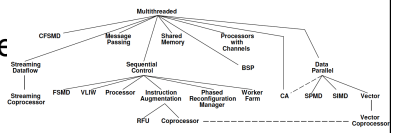
- Central controller farms out work



System Architecture Taxonomy

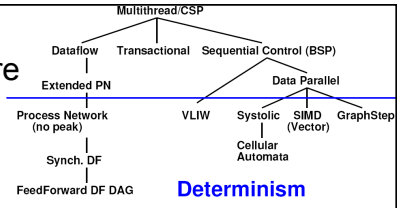


System Architecture Taxonomy

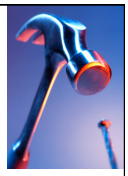


- Further down the hierarchy
  - – More restricted the model
  - + More guidance provided
  - + More efficient potential implementation
  - + More amenable to analysis
    - → tools and optimizations
- Restrictions provide power

System Architecture Taxonomy



- Further down the hierarchy
  - – More restricted the model
  - + More guidance provided
  - + More efficient potential implementation
  - + More amenable to analysis
    - → tools and optimizations
- Restrictions provide power

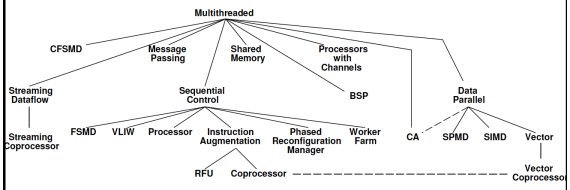


- When you have a big enough hammer, everything looks like a nail.
- Many stuck on single model
  - – Try to make all problems look like their nail
- Value to diversity / heterogeneity
  - – One size does not fit all

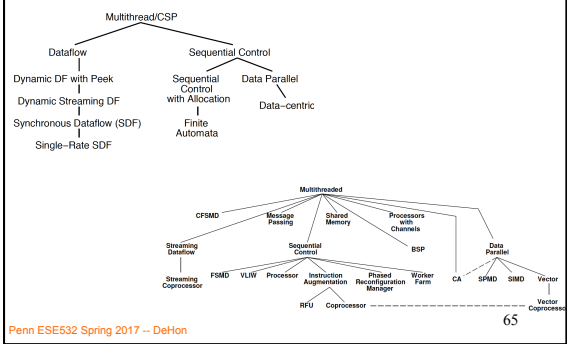
System Architecture Hypothesis

- There are a small number of useful system architectures
- These architectures
  - – Give guidance for organizing resources
  - – Make manageable
  - – Allow share lessons between applications
  - – Provide basis for scalability
  - – Point toward efficient solutions

System Architectures



Model → Architecture not 1:1



Big Ideas

- Many parallel compute models
  - – Sequential, Dataflow, CSP
- Useful System Architectures
  - – Streaming Dataflow, VLIW, co-processor, work farm, SIMD, Vector, CA, FSMD, …
- Find natural parallelism in problem
- Mix-and-match

Admin

- Reading for Day 4 on web
- Talk on Thursday by Ed Lee (UCB)
  - 3pm in Wu and Chen
- HW2 due Friday