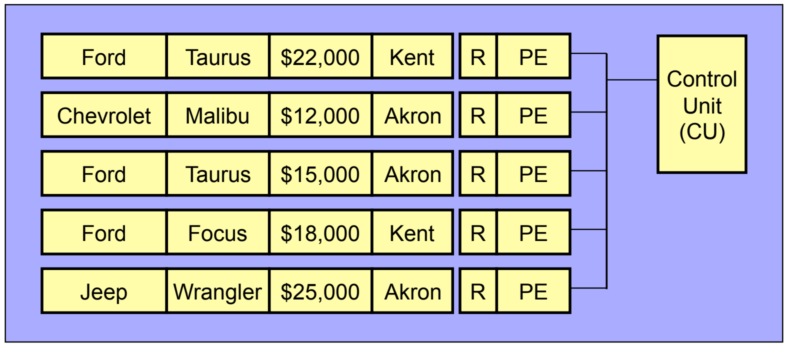
ASC Processor Research

Robert A. Walker, et al.

ASC Processor Group
Computer Science Department
Kent State University

Associative Search



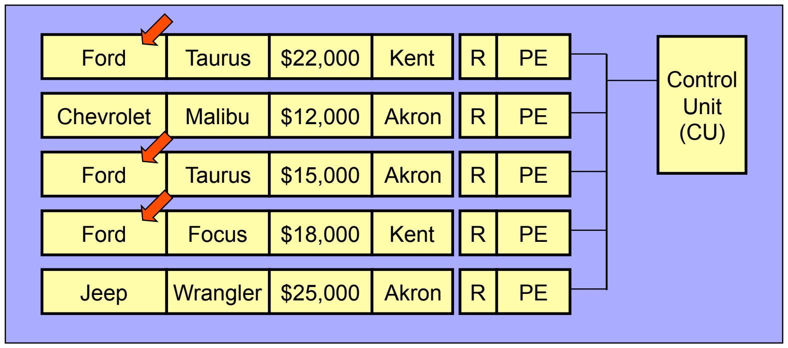
Ford	Taurus	$22,000	Kent	R	PE
Chevrolet	Malibu	$12,000	Akron	R	PE
Ford	Taurus	$15,000	Akron	R	PE
Ford	Focus	$18,000	Kent	R	PE
Jeep	Wrangler	$25,000	Akron	R	PE

Consider this simple automotive database

1 record per PE, each PE searches its local memory

"R" indicates a "responder" (successful match)

Associative Search

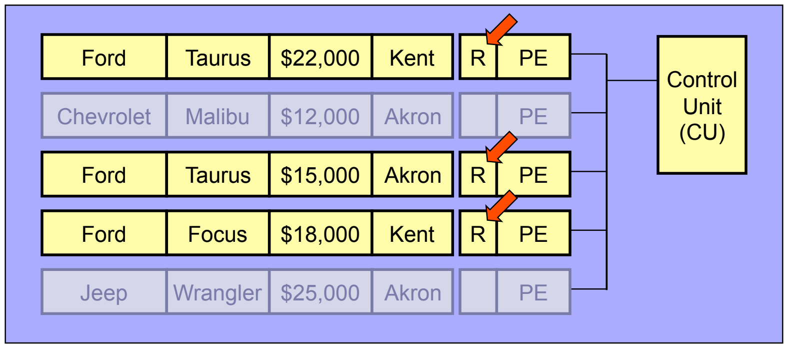


Ford	Taurus	$22,000	Kent	R	PE
Chevrolet	Malibu	$12,000	Akron	R	PE
Ford	Taurus	$15,000	Akron	R	PE
Ford	Focus	$18,000	Kent	R	PE
Jeep	Wrangler	$25,000	Akron	R	PE

PEs search for a key, ones that find it are *responders*

Find all "Ford" cars for sale

Associative Search



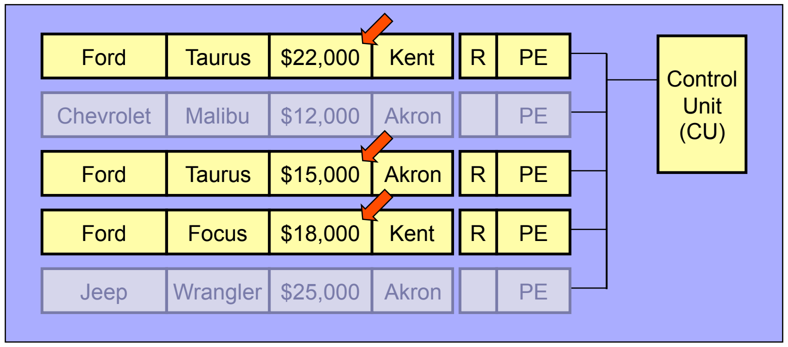
Ford	Taurus	$22,000	Kent	R	PE
Chevrolet	Malibu	$12,000	Akron		PE
Ford	Taurus	$15,000	Akron	R	PE
Ford	Focus	$18,000	Kent	R	PE
Jeep	Wrangler	$25,000	Akron		PE

PEs search for a key, ones that find it are *responders*

Find all "Ford" cars for sale

→ "R" indicates a "responder" (successful match)

Associative Search

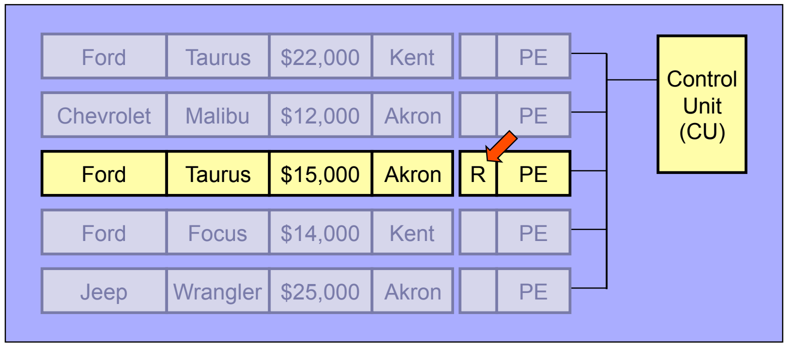


Ford | Taurus | $22,000 | Kent | R | PE
Chevrolet | Malibu | $12,000 | Akron | | PE
Ford | Taurus | $15,000 | Akron | R | PE
Ford | Focus | $18,000 | Kent | R | PE
Jeep | Wrangler | $25,000 | Akron | | PE
Control Unit (CU)

PEs perform a global minimum search

Find the "Ford" car with the lowest price

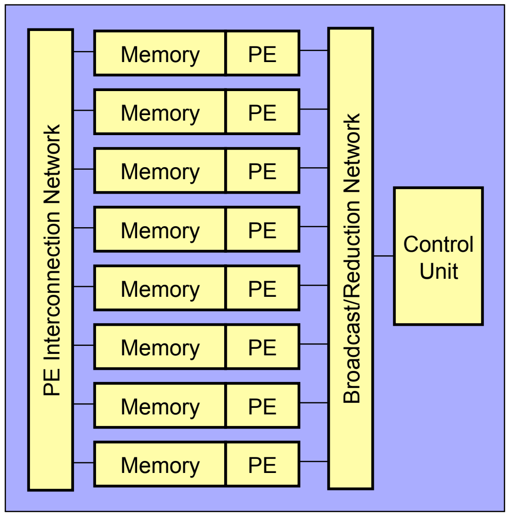
Associative Search



Ford | Taurus | $22,000 | Kent | | PE
Chevrolet | Malibu | $12,000 | Akron | | PE
Ford | Taurus | $15,000 | Akron | R | PE
Ford | Focus | $14,000 | Kent | | PE
Jeep | Wrangler | $25,000 | Akron | | PE
Control Unit (CU)

PE with the minimum value is now the only responder

Find the "Ford" car with the lowest price



Associative SIMD Array

Associative Search
- Broadcast

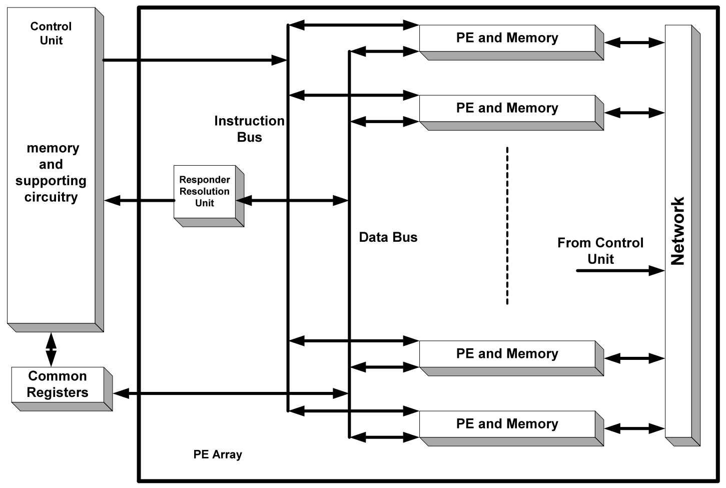
Responder Processing
- AnyResponders
- PickOne

Global Reduction
- Maximum/ minimum

Development of an ASC Processor

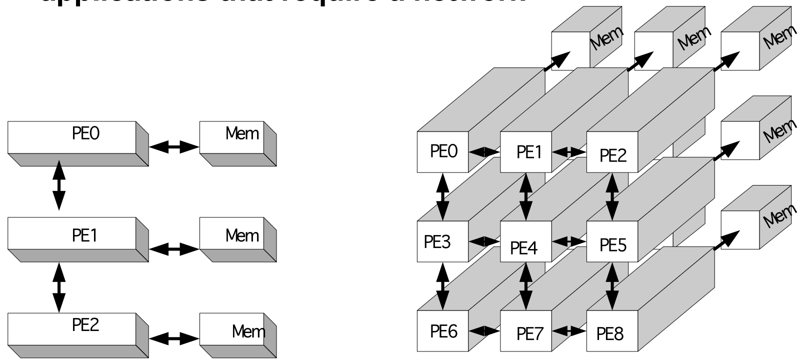
- **2001-02 — First 4-PE prototype w/ associative search, responder resolution, max/min search, not implemented**
- **2003 — Scalable ASC Processor w/50 PEs, implemented on APEX 20K1000E**
- **2004 — Scalable ASC Processor w/ 1-D and 2-D network, demonstrated on VLDC string-matching example & image processing example (edge detection using convolution)**
- **2005 — Scalable ASC Processor w/ pipelined PEs and reconfigurable network, to be demonstrated**
- **2005 — Scalable ASC Processor w/ augmented reconfigurable network and row/column broadcast, demonstrated on exact and approximate match LCS example**
- **2006 — MASC Processor**
- **2008 — Multithreaded ASC Processor**

Scalable ASC Processor

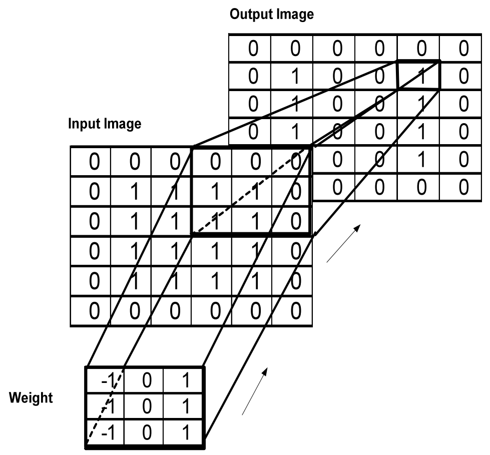


1-D and 2-D PE Interconnection Network

This version of ASC processor supports both a 1-D and 2-D PE interconnection network for those applications that require a network



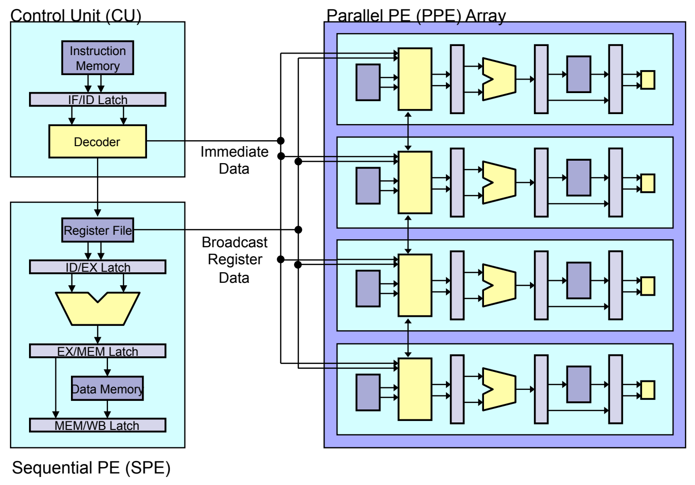
Edge Detection Using Convolution



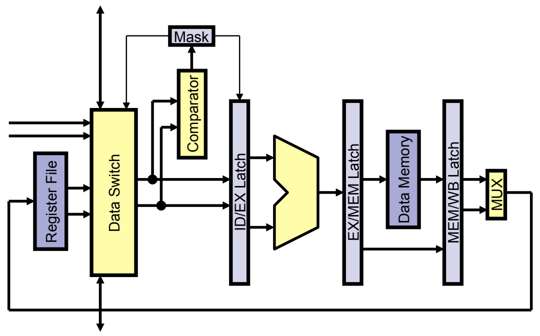
ASC Processor’s Pipelined Architecture

- **Five single-clock-cycle pipeline stages are split between the SIMD Control Unit (CU) and the PEs**
  - In the Control Unit
    - Instruction Fetch (IF)
    - Part of Instruction Decode (ID)
  - In the Scalar PE (SPE), in each Parallel PE (PPE)
    - Rest of Instruction Decode (ID)
    - Execute (EX)
    - Memory Access (MEM)
    - Data Write Back (WB)

Pipelined ASC Processor



Processing Element (PE)



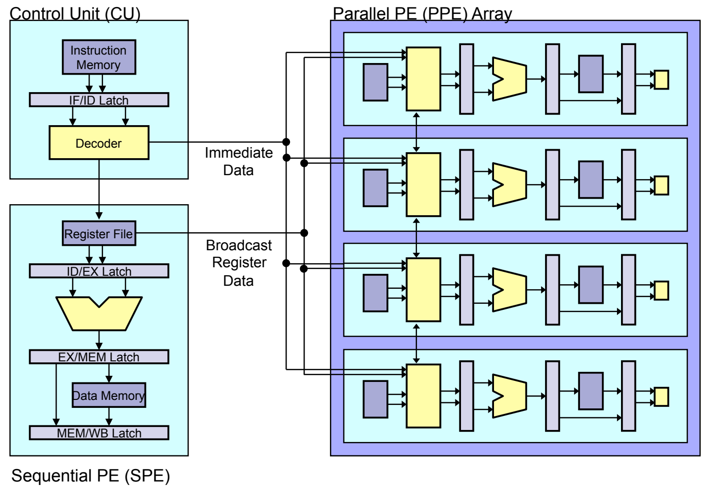
- Comparator implements associative search, pushes '1' onto top of stack for responders, '0' otherwise
- Top of mask of '0' disables ID/EX Latch

Reconfigurable PE Network

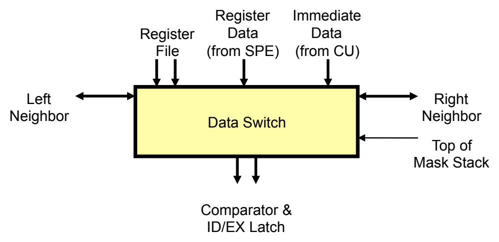
- **Our pipelined ASC Processor also has a reconfigurable PE interconnection network**
- **Reconfigurable PE network supports associative computing by allowing arbitrary PEs in the PE Array to be connected via**
  - Linear array (currently implemented), or
  - 2D mesh (shown in the next chapter)

  without the restriction of physical adjacency
- **Each PE in the PE Array can choose its own connectivity**
  - Responders choose to stay in the PE interconnection network, and
  - Non-Responders choose to stay out of the PE interconnection network, so that they are bypassed by any inter-PE communication

Reconfigurable PE Network



Reconfigurable Network Implementation



- **Data switch**
  - Passes register, broadcast, and immediate data to the PE and to its two neighbors
  - Routes data from the PE's neighbors to its EX stage
- **Reconfigurable network — supports *Bypass Mode* to remove the PE non-responders from the network**
  - Will be needed by MASC Processor

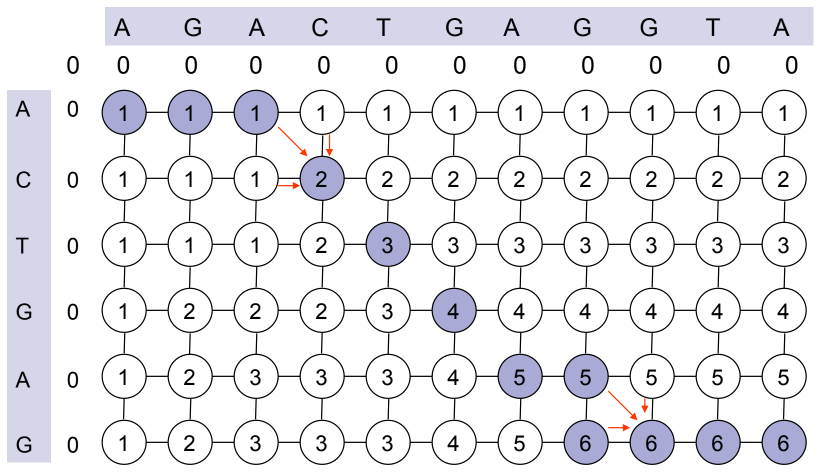
Overview of LCS Algorithm

- **Given two strings, find the LCS common to both strings**
- **Example:**
  - String 1: AGACTGAGGTA
  - String 2: ACTGAG
    - AGACTGAGGTA
    - - -ACTGAG - - -
    - - -ACTGA - G- -
    - A- -CTGA - G- -
    - A- -CTGAG - - -

list of possible alignments

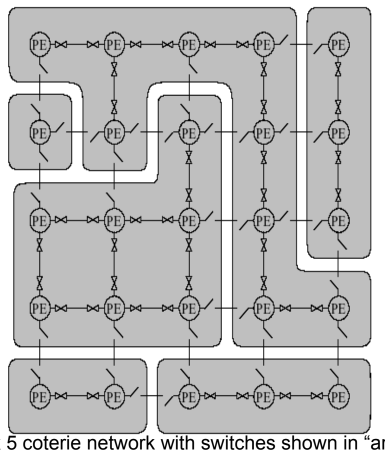
- **The time complexity of this algorithm is clearly O(nm)**

Overview of LCS Algorithm



		A	G	A	C	T	G	A	G	G	T	A
	0	0	0	0	0	0	0	0	0	0	0	0
A	0	1	1	1	1	1	1	1	1	1	1	1
C	0	1	1	1	2	2	2	2	2	2	2	2
T	0	1	1	1	2	3	3	3	3	3	3	3
G	0	1	2	2	2	3	4	4	4	4	4	4
A	0	1	2	3	3	3	4	5	5	5	5	5
G	0	1	2	3	3	3	4	5	6	6	6	6

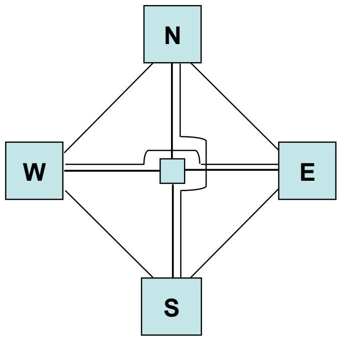
PE's Form Coteries



5 x 5 coterie network with switches shown in "arbitrary" settings. Shaded areas denotes coterie (the set of PEs Sharing same circuit)

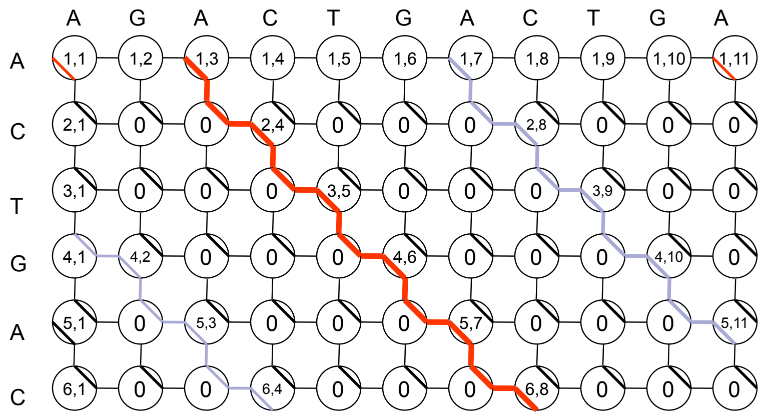
Reconfigurable 2D Network

- Key to reconfigurability is the Data Switch inside each PE:
  - The Data Switch is expanded to connect to its four neighbors (N-E-S-W) to form a 2D Reconfigurable Network
  - Data switch has bypass mode to allow PE communication to skip non-responders, so as to support associative computing



DATA Communication

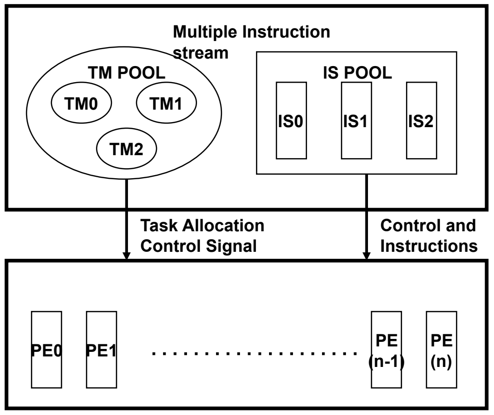
LCS on Reconfigurable 2D Network


A G A C T G A C T G A
A 1,1 1,2 1,3 1,4 1,5 1,6 1,7 1,8 1,9 1,10 1,11
C 2,1 0 0 2,4 0 0 0 2,8 0 0 0
T 3,1 0 0 0 3,5 0 0 0 3,9 0 0
G 4,1 4,2 0 0 0 4,6 0 0 0 4,10 0
A 5,1 0 5,3 0 0 0 5,7 0 0 0 5,11
C 6,1 0 0 6,4 0 0 0 6,8 0 0 0

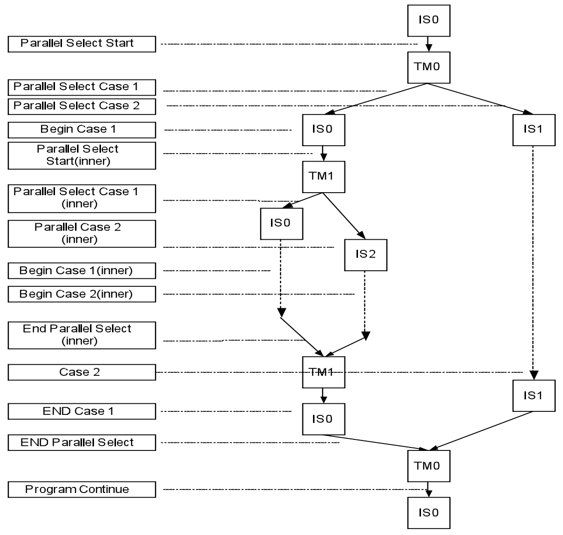
MASC Architecture

MASC is an MSIMD (multiple SIMD) version of ASC that supports multiple Instruction Streams (ISs)

In our dynamic MASC Processor, tasks are assigned to available ISs from a common pool as those ISs become available

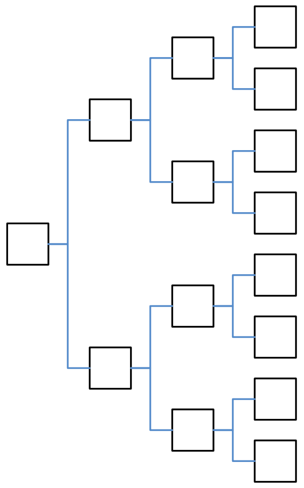


Task Manager (TM) and Instruction Stream (IS) Pools in the MASC Processor



Broadcast/Reduction Bottleneck

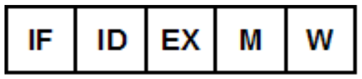
- Time to perform a broadcast or reduction increases as the number of PEs increases
- Even for a moderate number of PEs, this time can dominate the machine cycle time
- Pipelining reduces the cycle time but increases the latency
- Additional latency causes pipeline hazards



Instruction Types

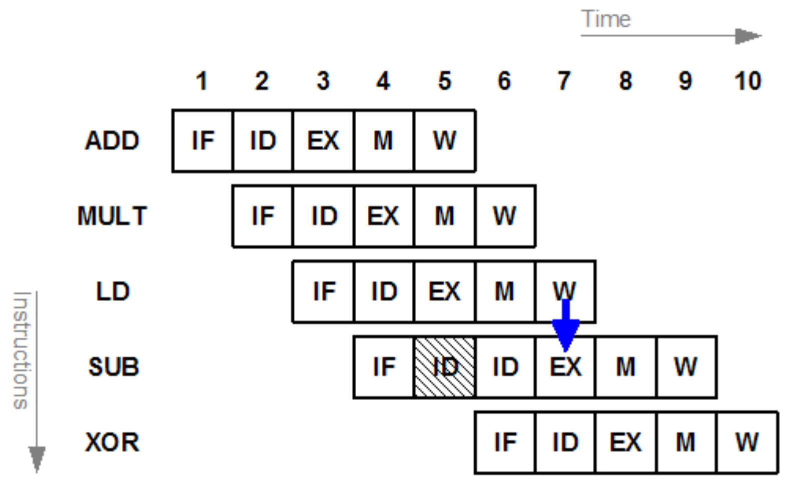
- **Scalar instructions**
  - Execute entirely within the control unit
- **Broadcast/Parallel instructions**
  - Execute within the PE array
  - Use the broadcast network to transfer instruction and data
- **Reduction instructions**
  - Execute within the PE array
  - Use the broadcast network to transfer instruction and data
  - Use the reduction network to combine data from PEs

Scalar Pipeline

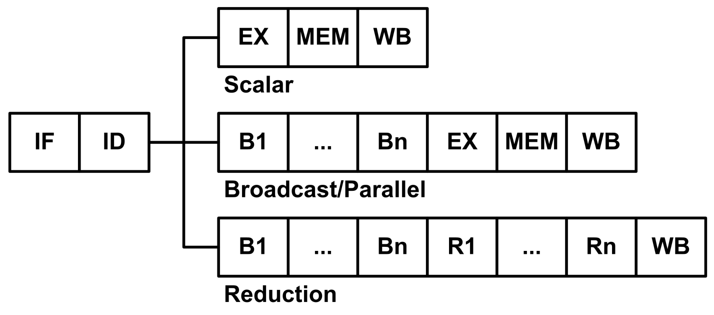


- Instruction Fetch (IF)
- Instruction Decode (ID)
- Execute (EX)
- Memory Access (M)
- Write Back (W)

Hazards in a Scalar Pipeline

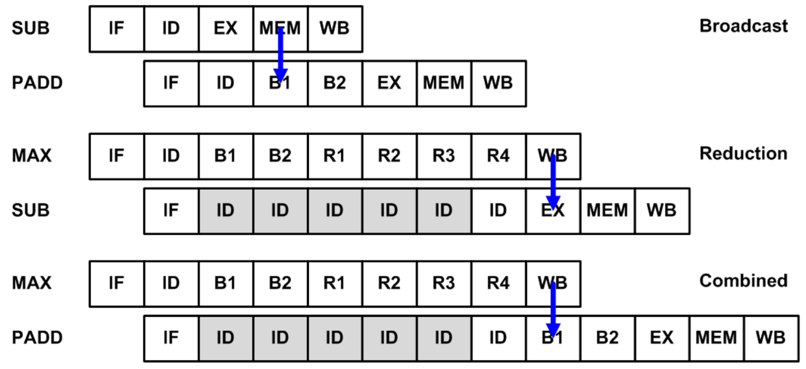


Pipeline Organization



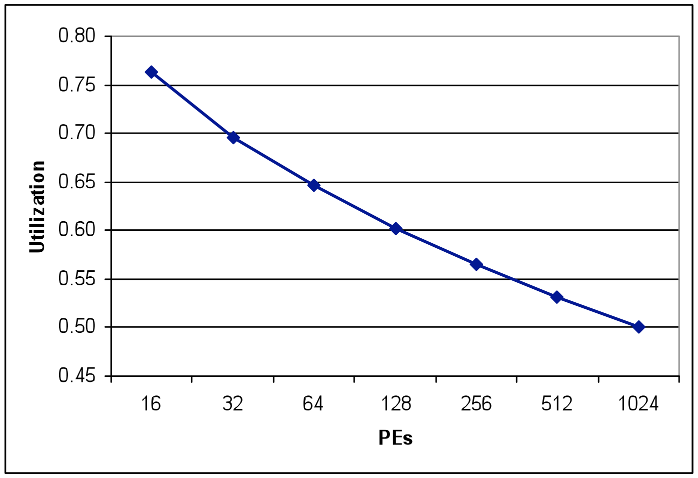
- Separate paths for each instruction type so instructions only go through stages that they use
- Stalls less often than a unified pipeline organization

Hazards



SUB IF ID EX MEM WB
PADD IF ID B1 B2 EX MEM WB
Broadcast
MAX IF ID B1 B2 R1 R2 R3 R4 WB
SUB IF ID ID ID ID ID ID EX MEM WB
Reduction
MAX IF ID B1 B2 R1 R2 R3 R4 WB
PADD IF ID ID ID ID ID ID B1 B2 EX MEM WB
Combined

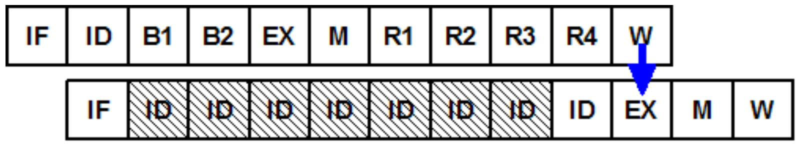
Effect of Hazards on Pipeline Utilization



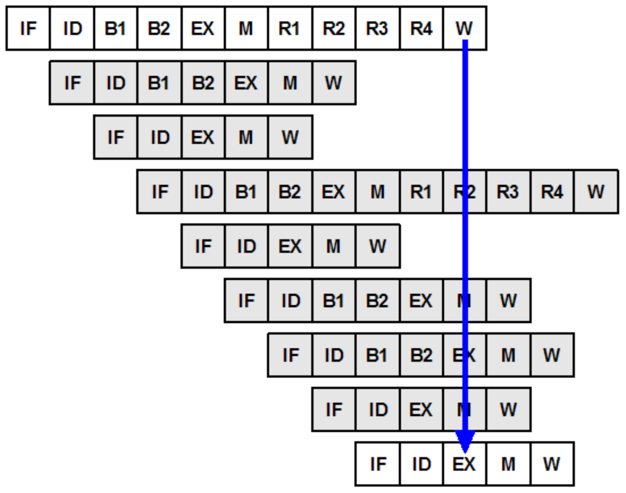
Multithreading

- Pipelining alone cannot eliminate hazards caused by broadcast and reduction latencies
- Solution: use instructions from multiple threads to keep the pipeline full
- Instructions from different threads are independent so they cannot generate stalls due to data dependencies
- As long as there are a sufficient number of threads, it is possible to fill any number of stall cycles

Reduction Hazard with a Single Thread


IF ID B1 B2 EX M R1 R2 R3 R4 W
IF ID ID ID ID ID ID ID ID EX M W

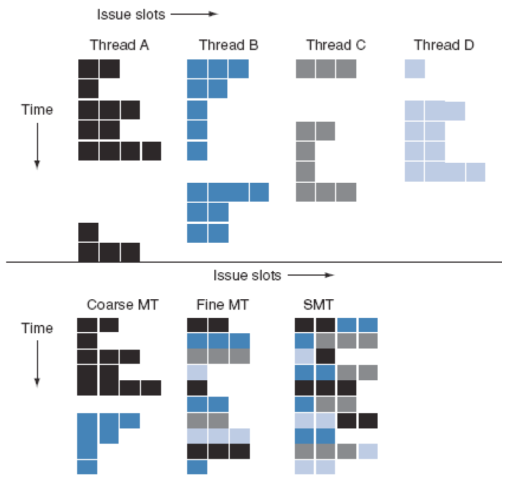
Reduction Hazard with Multiple Threads



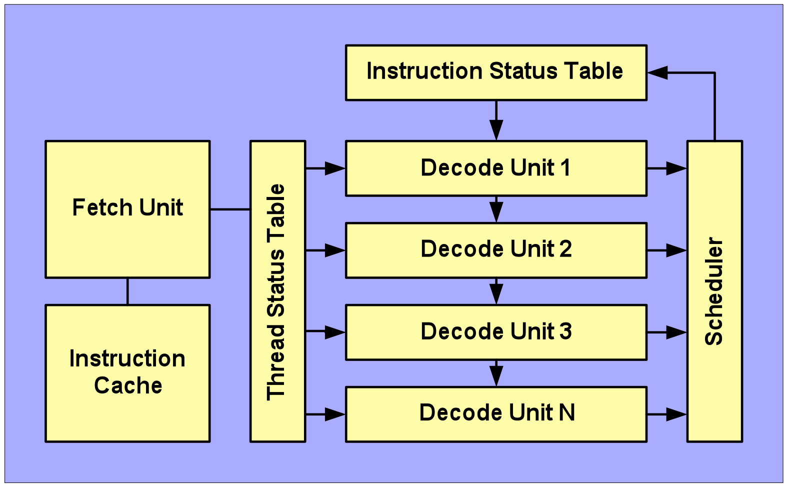
Types of Multithreading

- *Coarse-grain multithreading* switches to a new thread when the current thread encounters a high latency operation
- *Fine-grain multithreading* switches to a new thread every clock cycle
- *Simultaneous multithreading* can issue instructions from multiple threads in the same clock cycle
- For a SIMD processor, fine-grain or simultaneous multithreading is necessary as pipeline stalls are relatively short and occur frequently

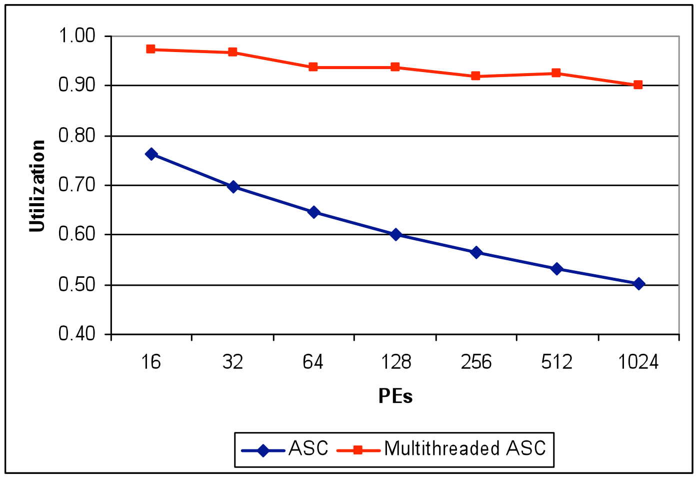
Types of Multithreading



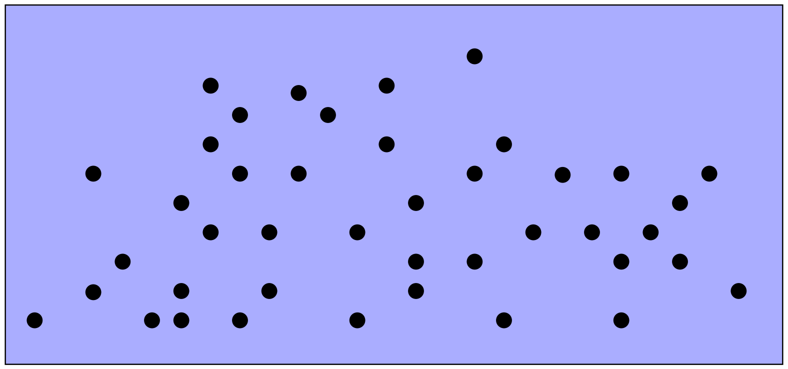
Multithreaded Control Unit



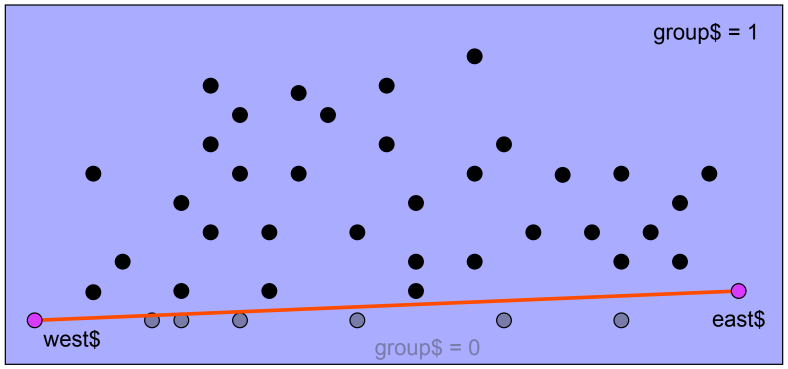
Pipeline Utilization with Multithreading



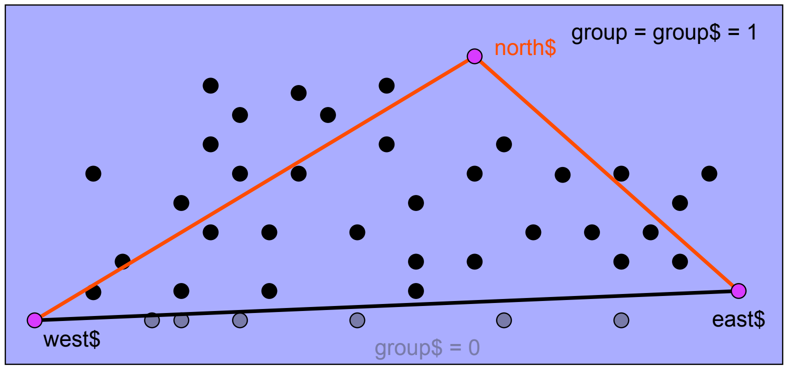
Associative QuickHull



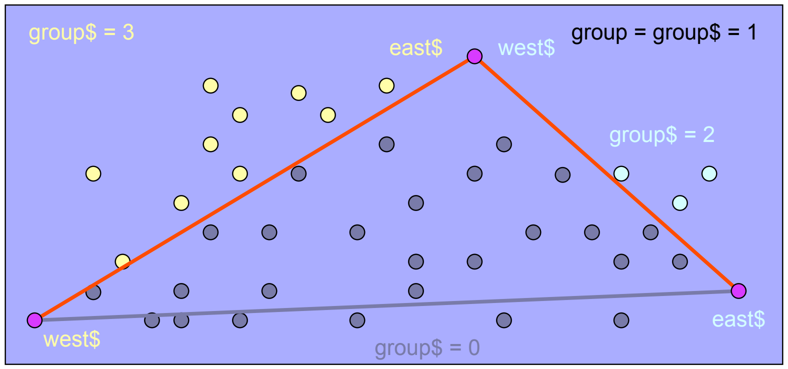
Associative QuickHull



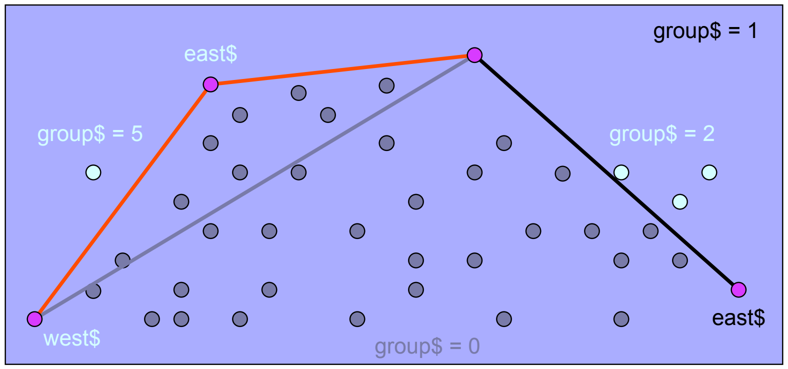
Associative QuickHull



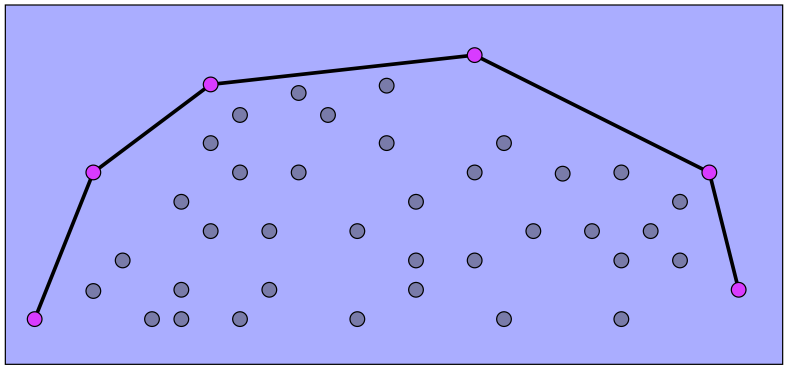
Associative QuickHull



Associative QuickHull



Associative QuickHull



Some Currently Open Problems

- **Implement & demonstrate "virtual PEs" on associative String Match and/or LCS algorithm**
- **Continue LCS algorithm research**
  - Investigate further the presence of "gaps"
  - Find "best" CS instead of "longest" CS
- **Implement & demonstrate one or more associative algorithms on ASC and/or MASC Processor**
  - Convex Hull, video/media processing
- **Augment first MASC Processor to support nested conditionals and loops and to support network operations**

Some Currently Open Problems (cont'd)

- **Modify the ASC compiler to generate assembly language for one of the ASC processor prototypes**
- **Add I/O support to processor**