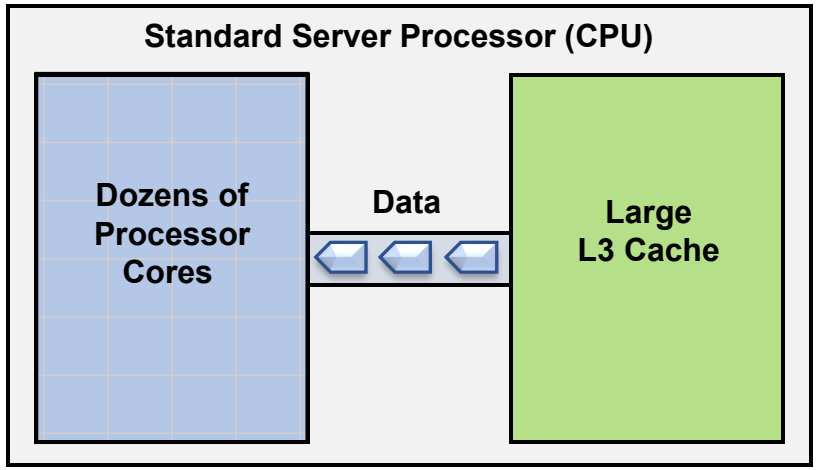
An Associative Processing Structure Challenges von Neumann Architecture for Latency and Energy Efficiency

Linley Fall Processor Conference, October 29, 2020

Standard CPUs vs. Gemini APU

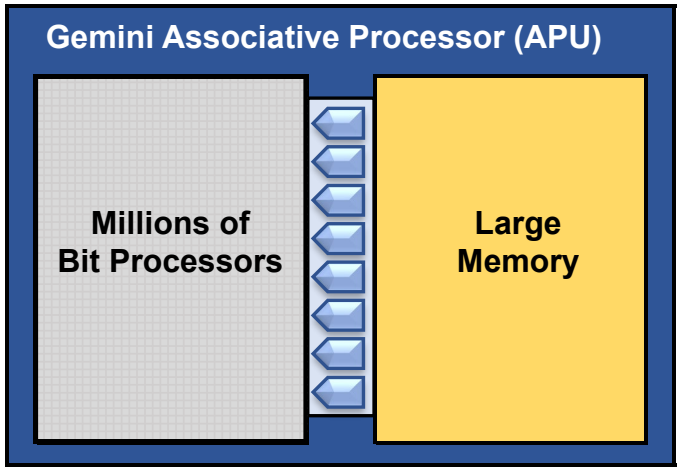
Standard CPUs

- Highly efficient at complex computations on small datasets
- Memory bottleneck throttles performance for large datasets
- Energy associated with data transfers is higher



Gemini APU

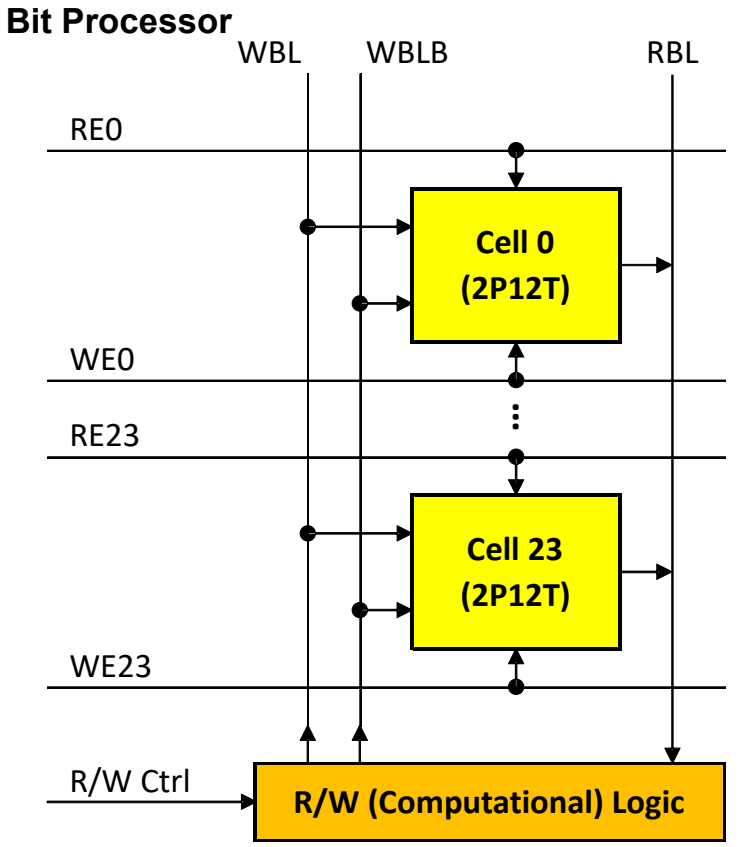
- Highly efficient at simple computations on large datasets
- Huge bandwidth to local memory
- Energy associated with data transfers is lower



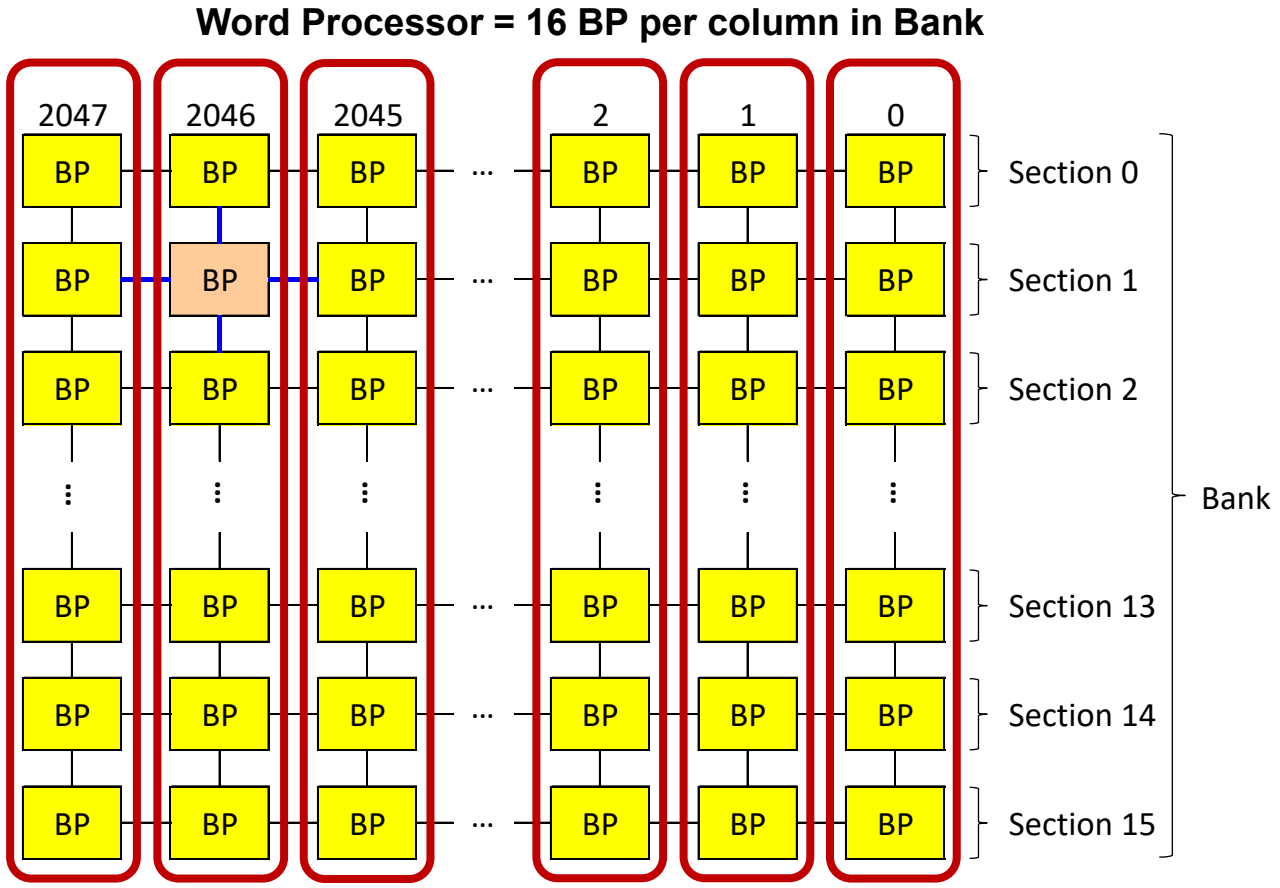
Gemini APU : Key Parameters

Bit Processors	2 Million
Associative memory	48 Mb
Boolean Ops/sec (@ 400 MHz)	840 Trillion
Peak Compute (8b ADD) Performance	25 TOPS
Peak Search (128b TCAM) Performance	1 Trillion/sec
Local Memory (L1)	96Mb, distributed
L1 <-> BP Data Bandwidth	210 Tb/sec
Power (TDP)	60W
Status	LEDA-G PCIe Card Shipping

Bit Processor



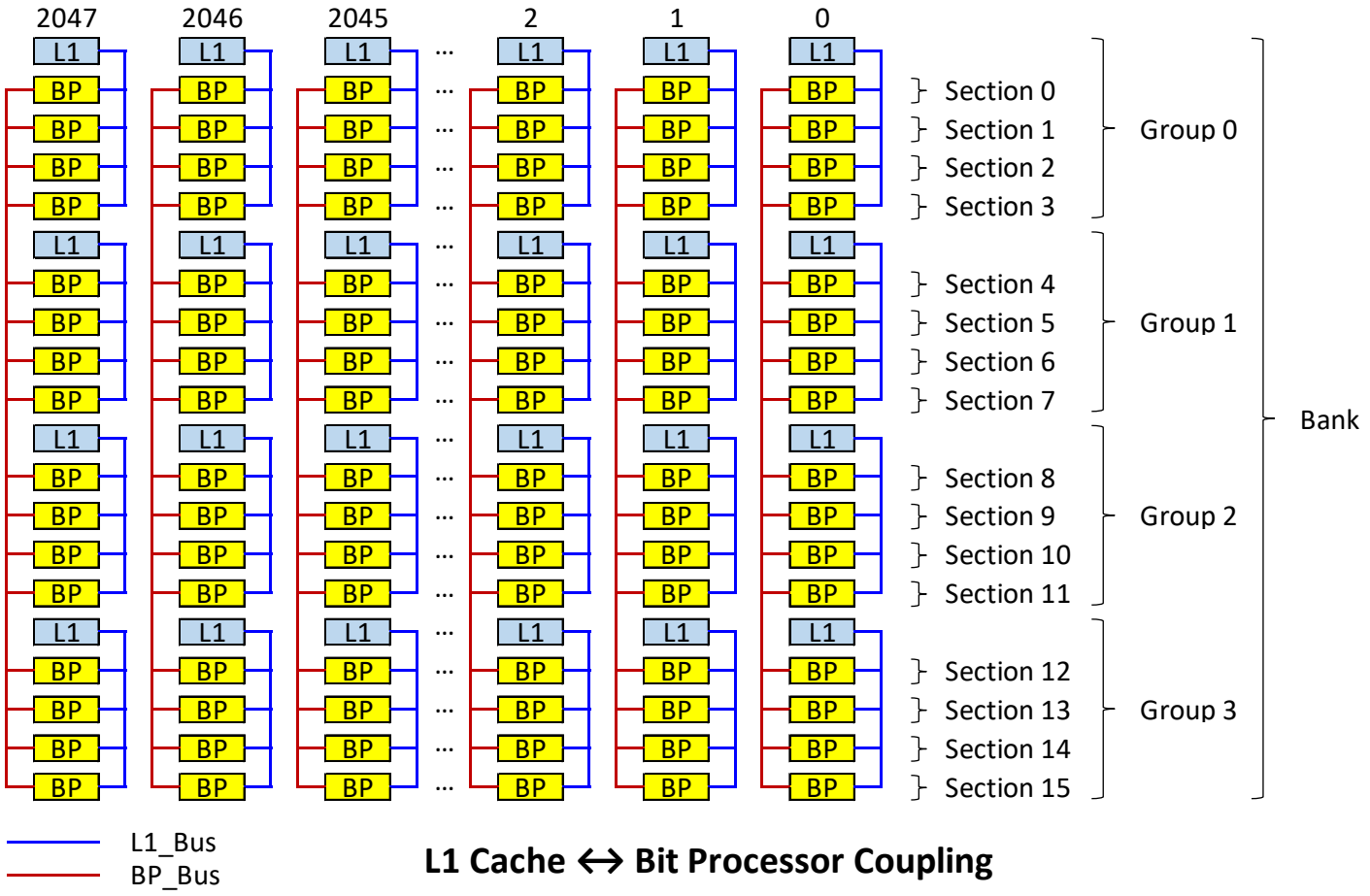
RBL = Logical AND when Multiple Cells are Read



Section = 2K columns (bit-lines) = 2K BP
Bank = 16 Sections = 32K BP (2K WP)

BP Instruction = RE[23:0] + WE[23:0] + R/W Ctrl. All 2K BPs per section receive the same instruction.

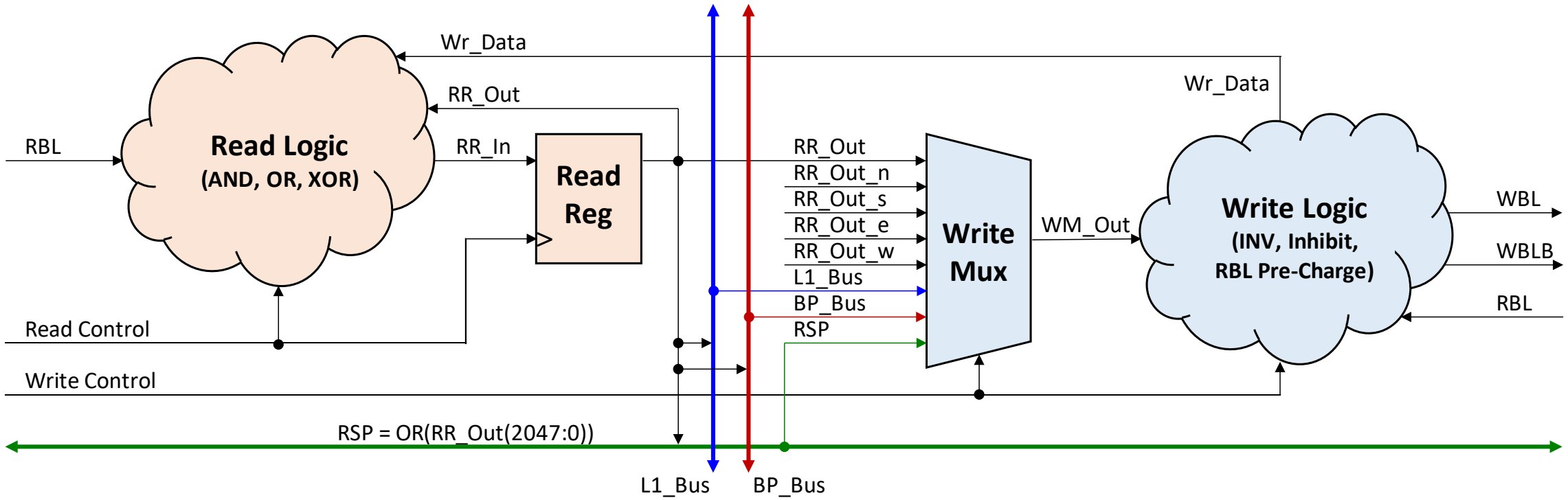
Distributed L1 Cache



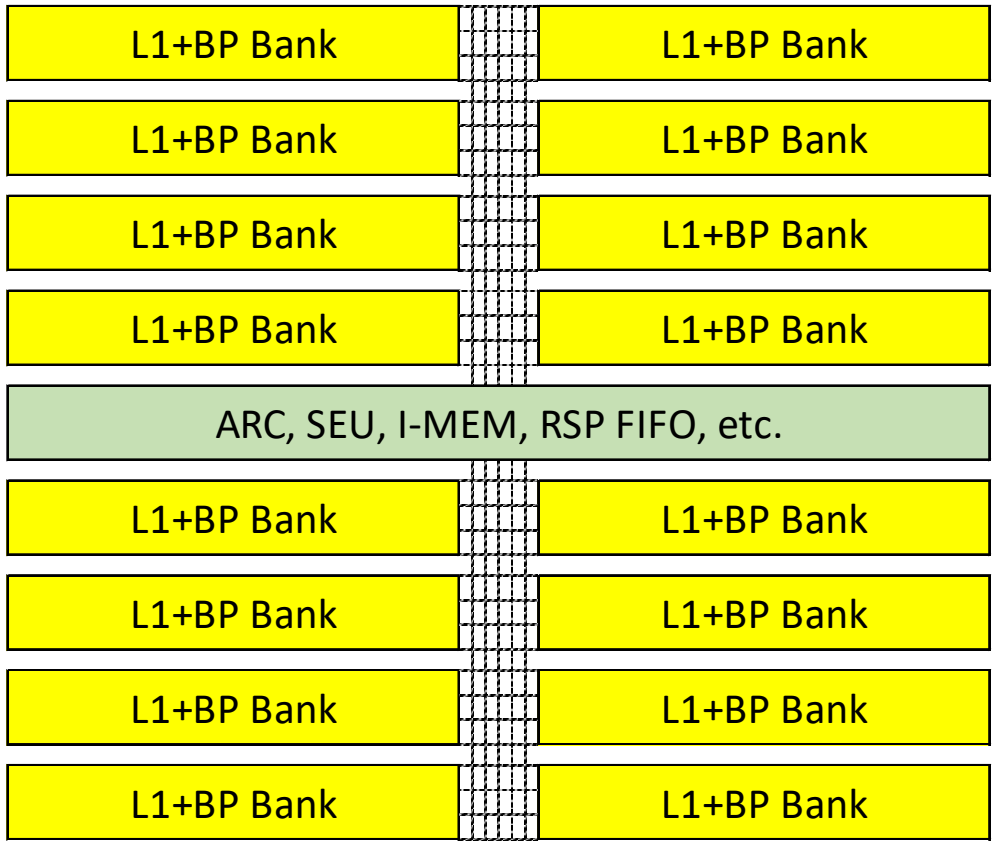
L1 Cache ↔ Bit Processor Coupling

- L1 Block Size = 192 mcells
- L1 Group Size = 2K * 192 = 384Kb
- L1 Bank Size = 4 * 384Kb = 1.5Mb
- L1 ↔ BP Bandwidth
  - 8Kb per cycle per bank
  - 210 Tb/s per chip @ 400 MHz

Bit Processor Computational (R/W) Logic



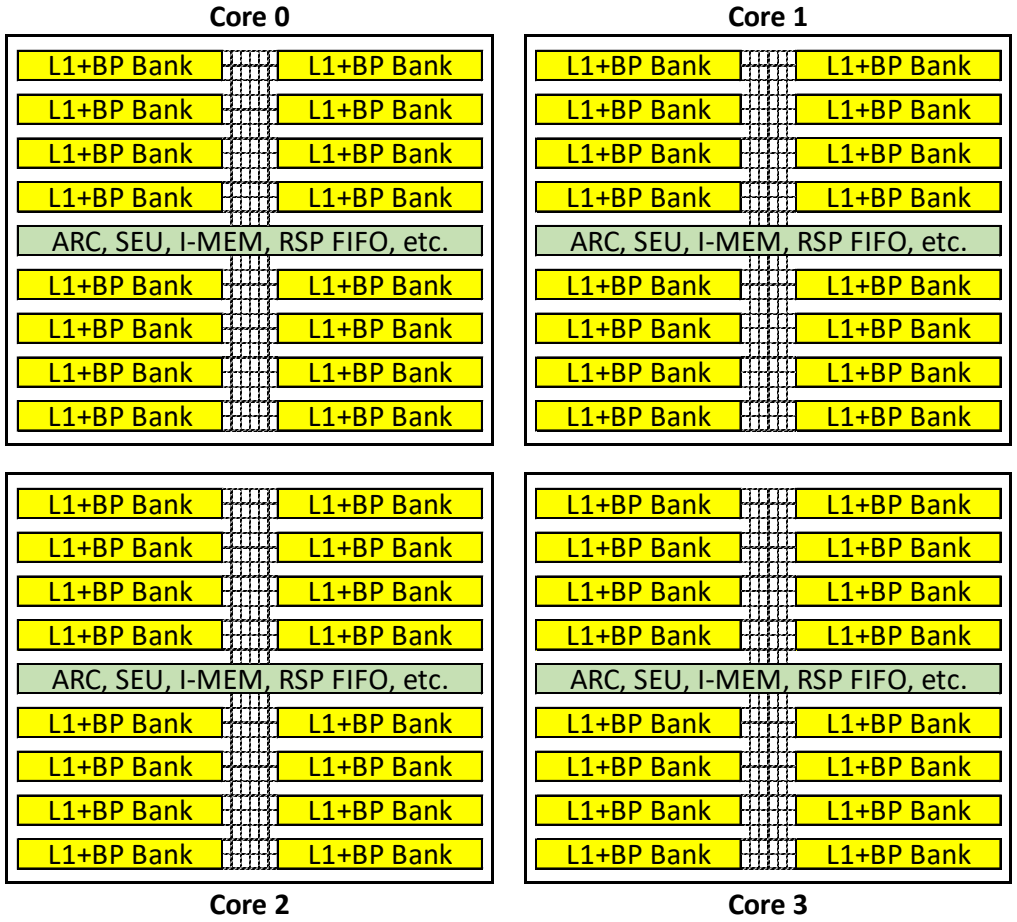
Core Configuration



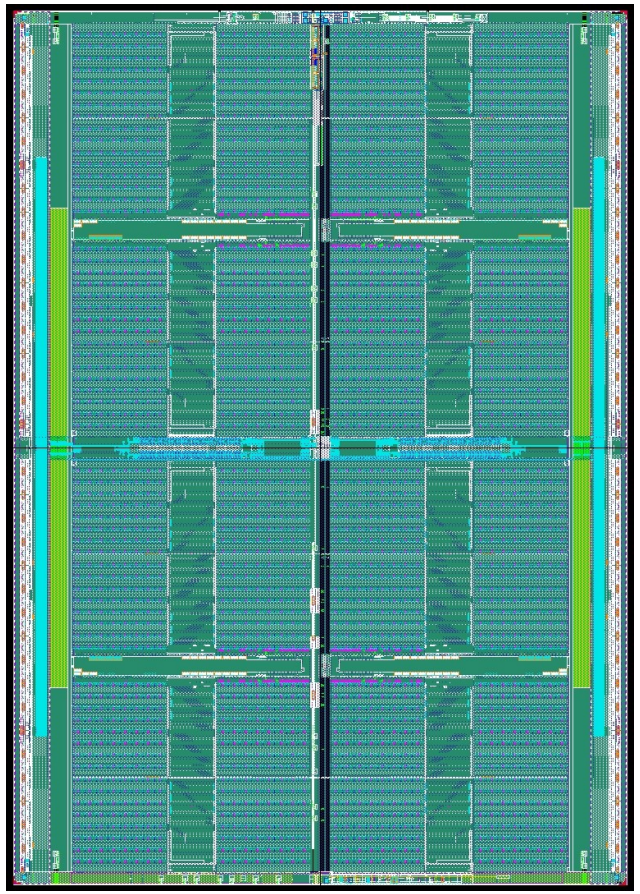
Core = 16 Banks = 512K BP (32K WP) + 24Mb L1

- The core's **Section Execution Unit** (SEU) can generate up to 4 unique BP instructions per cycle.
  - A particular instruction (any of the 4) can be mapped to any of the 16 BP sections in a bank.
  - All 16 BP banks per core execute the same instructions, with the same section mapping.
- The core's SEU can generate 1 L1 instruction per cycle.
  - All 4 L1 groups in all 16 L1 banks per core execute the same instruction.

Chip Configuration



Chip = 4 Cores = 64 Banks = 2M BP (128K WP) + 96Mb L1



Chip Layout

Applications of The Gemini® APU

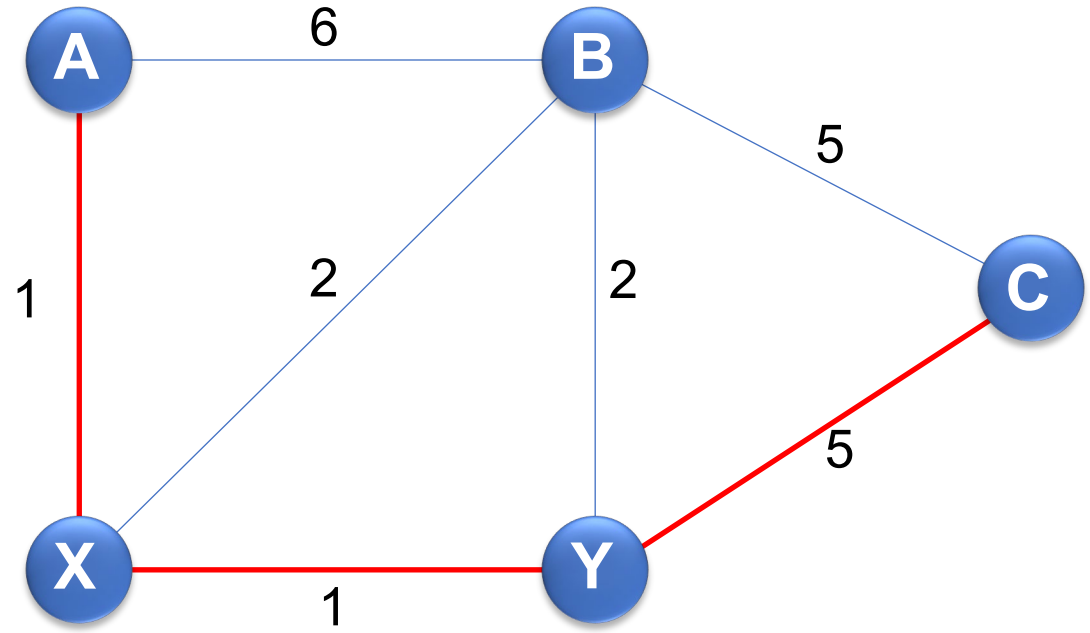
Architecture advantages:

- Parallel processing
- Associative computing
- Energy efficiency

Graph Theory: Shortest Path Problem

- Given graph (V,E), find the shortest path between source node to destination node. We shall see how Dijkstra's algorithm is computed in associative memory
- Nodes are represented as keys in dictionary
- Edges are represented as values in list associated with a source key
- Each associative word processor store a key and a list of values

Graph Theory: Shortest Path Problem



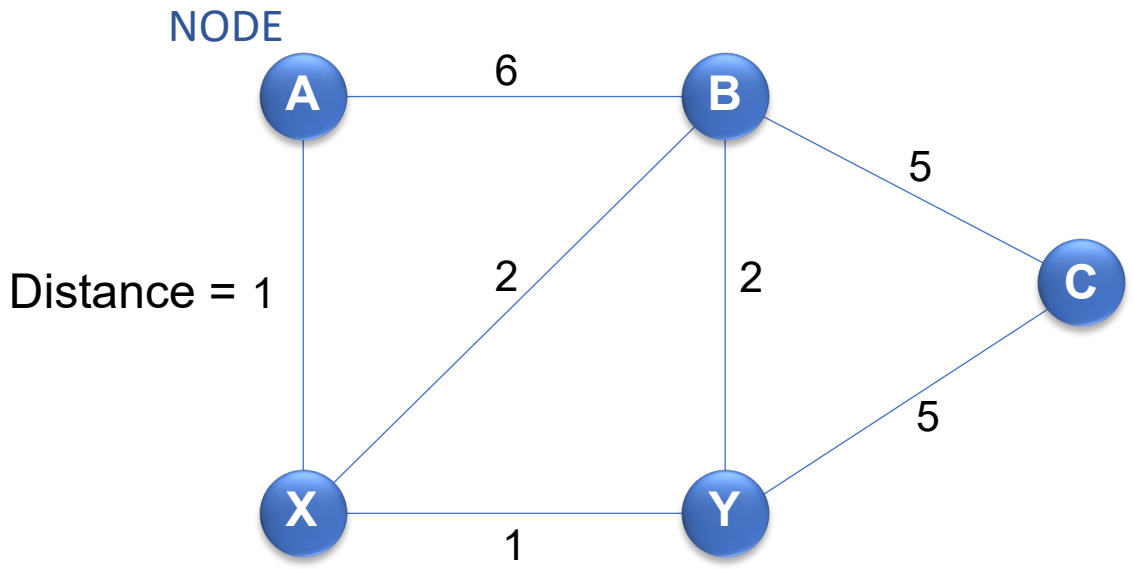
WP index	0	…	i	j	k	m	n	…	32K-1
Key (Node)			A	B	C	X	Y		
Values			(B,6)…	(A,6),…	(B,5),…	(A,1),…	(B,2),…		

Dijkstra: Start

Load graph to associative memory

Query: Find shortest path from A to C

Set current node to source (A)

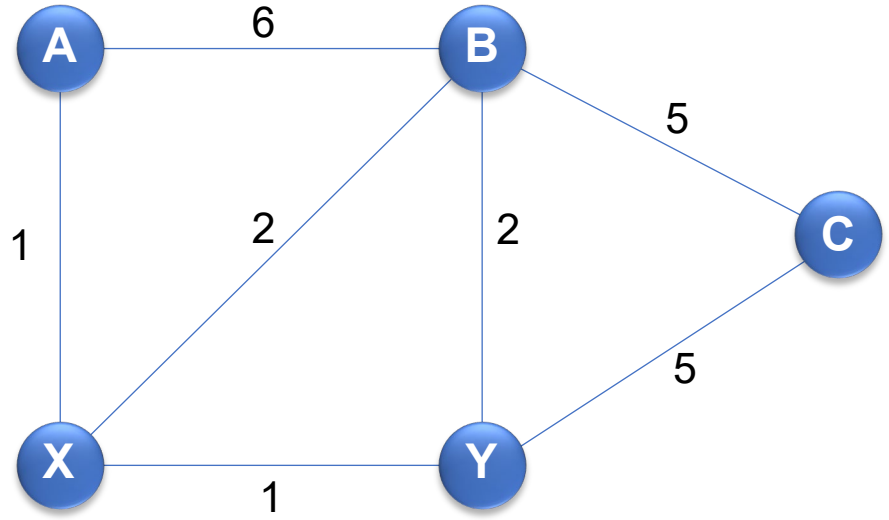


Node (Key)	Visited	Distance to A	Parent	Edges (Values)
A	0	0	A	(B,6,0), (X,1,0)
B	0	INF		(A,6,0), (Y,2,0), (C,5,0), (X,2,0)
C	0	INF		(B,5,0), (Y,5,0)
X	0	INF		(A,1,0), (Y,1,0), (B,2,0)
Y	0	INF		(B,2,0), (C,5,0), (X,1,0)

Dijkstra Iteration (1)

Update all unvisited neighbors of current node:

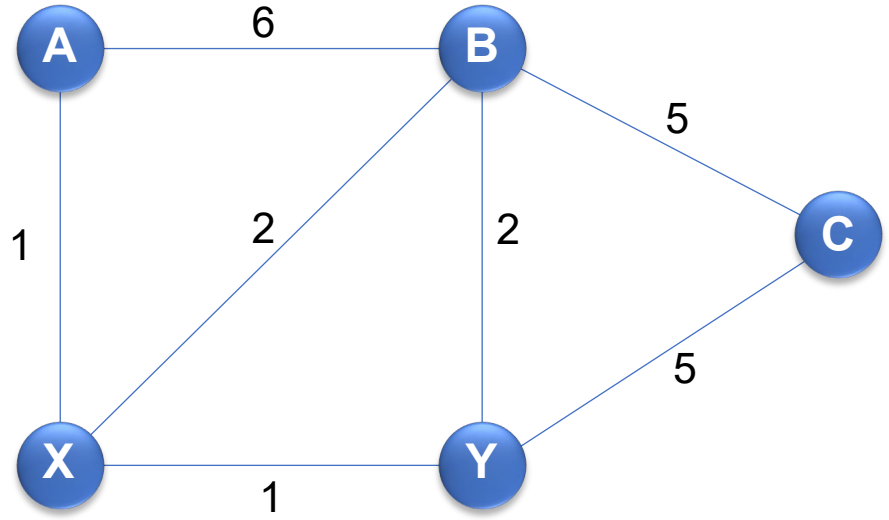
Associative search and selective, in-memory update



Node	Visited	Distance to A	Parent	Edge
A	1	0	A	(B,6,0), (X,1,0)
B	0	INF->6	A	(A,6,1), (Y,2,0), (C,5,0), (X,2,0)
C	0	INF		
X	0	INF->1	A	(A,1,1), (Y,1,0), (B,2,0)
Y	0	INF		

Dijkstra Iteration (2)

Set current node to unvisited node with <u>minimal</u> distance to source (X in this example): Associative in-memory global minimum.



Node	Visited	Distance to A	Parent	Edge
A	1	0	A	
B	0	6	A	
C	0	INF		
X	0	1	A	
Y	0	INF		

Dijkstra (next iterations)

Iterate over steps 1,2 until reaching destination (C) or until running out of options

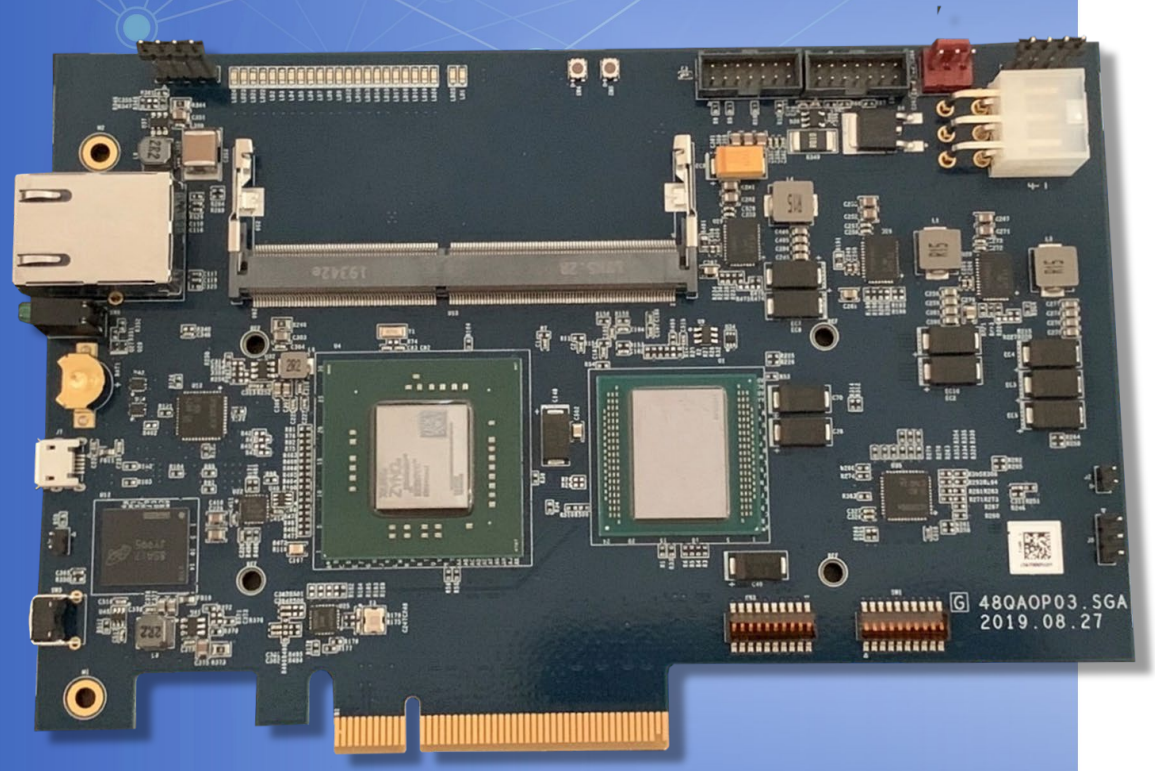
Summary: How is this done?

- From Boolean functions to Arithmetic functions
- SIMD: Data level parallelism
- Selective update: Data dependency
- Associative search: Key->Value
- Global operations: Min/Max

APU vs CPU: Energy Efficiency

- CPU “Power wall”: Energy cost of moving data between the memory cache and the core of a CPU is much higher than the energy cost of ALU operations on the data inside the core.
- Table below compares the dynamic energy of APU with a CPU
- The marked efficiency of APUs is explained by Short data traces between the memory (L1) and the Bit Processors (BPs)

Operation	CPU (data in L3 Cache)	APU (data in L1 memory)	APU vs CPU
Copy (4KB)	1500 (nJ)	15 (nJ)	1%
Logical OR (4KB)	2000 (nJ)	24 (nJ)	1.2%



Advantages of The Gemini® Architecture

- **Parallel processing**
  - **2 Million Bit Processors**
- **Associative computing**
  - **1 Trillion TCAM search/sec peak**
- **Energy efficiency**
  - **~1% Dynamic Energy Consumption vs. CPU (for data transfers and Boolean operations)**

www.gsitechnology.com/APU

associativecomputing@gsitechnology.com