In-Place Associative Computing:

An Introduction to the APU Architecture

www.gsitechnology.com

Abstract

The world is being reshaped by artificial intelligence, with deep learning algorithms now driving everything from personalized health monitoring solutions to individualized online retail experiences.

As computation becomes increasingly limited by data movement and energy consumption, the market is seeing an increase in demand for artificial intelligence hardware solutions. The Associative Processing Unit (APU) is GSI Technology's patented processing technology and a new breed of processor that computes in-place—directly in the memory array—thus significantly reducing the overheads associated with data movement. GSI's APU drives robust performance on inference stage word2vec, image2vec, and other feature extraction models, and accelerates similarity search applications.

This whitepaper covers the APU's overall architecture, including the APU card, host to device interfaces, hierarchical structure, and the main computational memory—the core of this solution—and its main components. The paper also describes the other memory components and the vertical and horizontal data lines. Finally, there is brief overview of the APU's internal and external data movement.

Overview

To perform in-place computing in the memory array, GSI Technology has developed the Associative Processing Unit (APU).

The APU processor is composed of vector memory for data storage, and a computational memory unit known as the Main Memory Block (MMB). The computational memory array combines SRAM memory cells and patented bit-level processing elements. The architecture enables program instructions to be broken down into basic operations that can be performed by the processor in parallel. The microcode orchestrates these parallel operations utilizing Single Instruction, Multiple Data (SIMD), allowing multiple processing elements to perform the same operation on multiple data points, simultaneously.

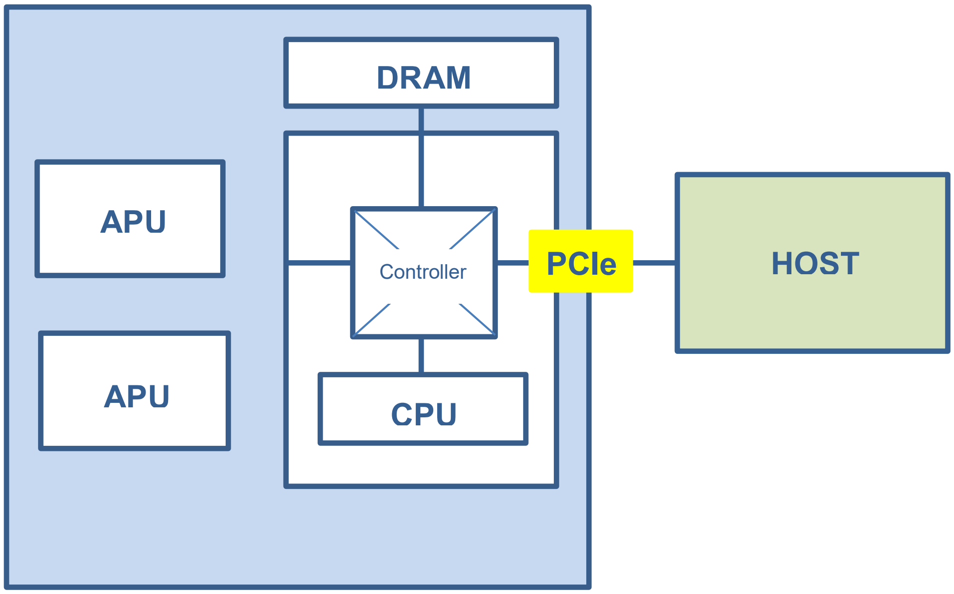
Associative Processing Unit (APU) Card

The APU Card is a PCIe add-in card which is attached to a server as an associative computing acceleration device. The APU card consists of 2 APU chips, a card controller device, and device memory (DRAM). The controller connects the APU chips and the host to the card's DRAM.

The APU card's form factor is a standard height, full length PCIe Gen3 x16, with a width of 2 PCIe slots.

The APU card's DRAM[1] memory capacity is up to 32 GB.

APU Card



Host-Device interface

The APU Card Controller

The APU card controller includes a PCIe interface for connecting with the host, 2 DDR4-DRAM channels to device memory, 2 UBus channels to connect with the APU Chips, a CPU for the card's firmware, and APU Chip Controller (ACC) logic to connect the APU with the DRAM channels and with the firmware.

The UBus is a bi-directional multi-master DDR bus connecting the APU Chip with the APU Card Controller (and with the DRAM, through the controller).

The data path between device memory (DRAM) and APU memory (L1) is split between APUC DMA (L2DMA) and APU Card DMA (L4DMA).

[1] From the host's perspective, the card's DRAM is usually referred to as "device memory." However, this paper uses the term "system memory" to refer to the card's DRAM from the APU's perspective.

The controller uses a RT-CPU and a L4DMA engine to move data efficiently between ACCs and the device memory. The APU cores use a L2DMA engine to move data efficiently between the L2 memory and the ACCs.

Host to Device Interactions

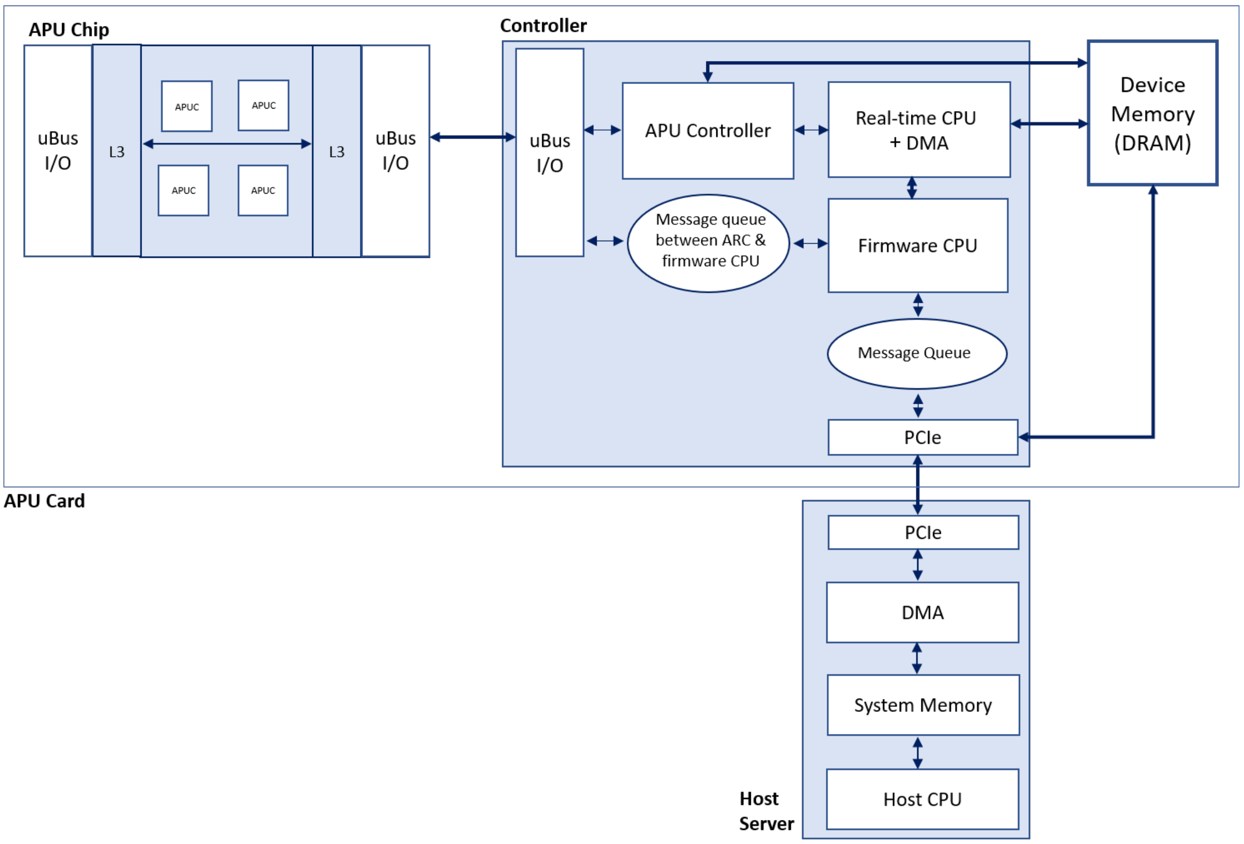
The host software and card firmware exchange command and completion descriptors over the host message queues. The host manages the data path between the system memory and the device memory, using the host side DMA.

The firmware sends task descriptor pointers to the APUC over the UBus. The APUC sends an interrupt message back to the firmware over the UBus (similar to MSI-X) to indicate task completion.

The firmware manages the APU control plane but not the data path.

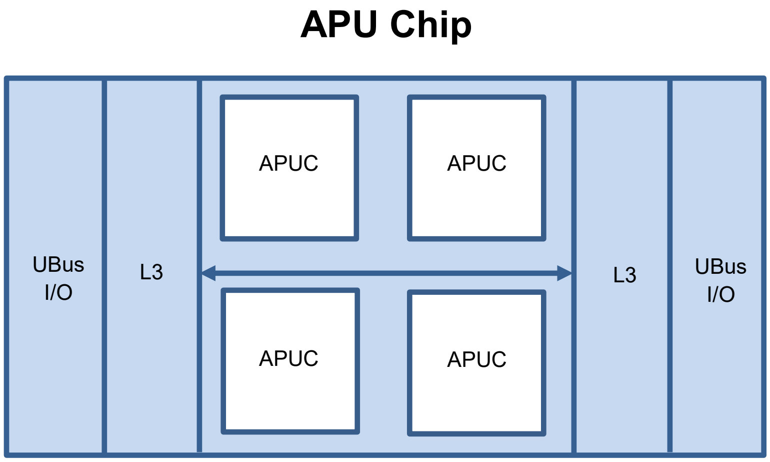
The APU cores send I/O tasks to the RT-CPU via shared memory to activate the L4DMA, and the RT-CPU sends completion interrupt message back to the APU cores.

Host to Device Interface



APU Chip

The APU chip consists of 4 APUCs. Each APUC is connected to 1MB of on-chip SRAM (L3 storage) memory and to the controller via the APU I/O. The L3 storage (L3S) memory is used as a low latency memory for the APUCs.

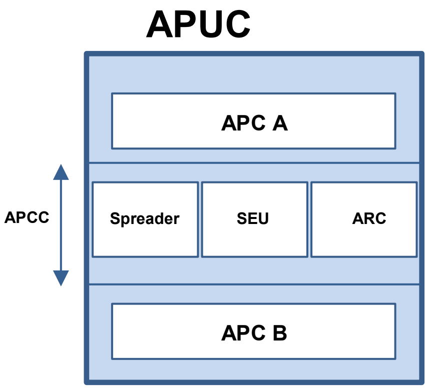


APUC

The APUC is the core processing unit in the APU. It consists of an APCC and 2 APCs. The APCC controls the two APCs, enabling them to perform in-place logic operations on data stored in the APC's Main Memory Block (MMB).

Within the APCC, there is an ARC microprocessor which runs the program code, a Section Execution Unit (SEU) which is a 4-lane microcode unit that runs the microcode, and a spreader.

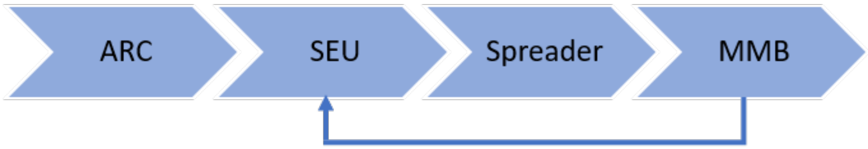
The spreader receives 4 sets of MMB commands and section masks—plus other miscellaneous MMB, L1, and L2 control signals—from the SEU. It uses them to generate the APC control signals for the MMB, L1, and L2.



The SEU is an APCC component that fetches microcode instructions from the SEU Instruction Memory (SEU I-Mem). It then decodes these instructions into MMB commands and section masks, as well as other miscellaneous MMB, L1, and L2 control signals, and sends them to the spreader.

The SEU I-Mem is the microcode memory in the SEU that stores the microcode instructions for each execution cycle.

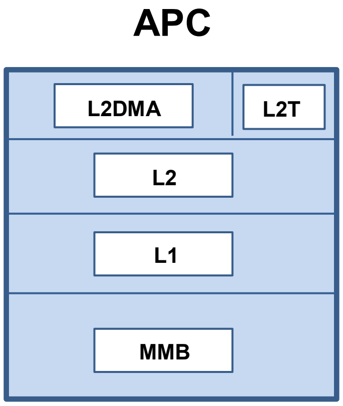
The following diagram illustrates the data flow from the ARC processor to the MMB:



APC

The Associative Processor Core (APC) is a collective term for the several components that comprise each main processing core in the APU. There are 2 APCs per APUC, and they are controlled by one APCC. Each APC consists of MMB, L1, L2, and L2T.

The L2DMA is a DMA engine that controls the L2T and L2 during data transfer to and from the system memory.



MMB

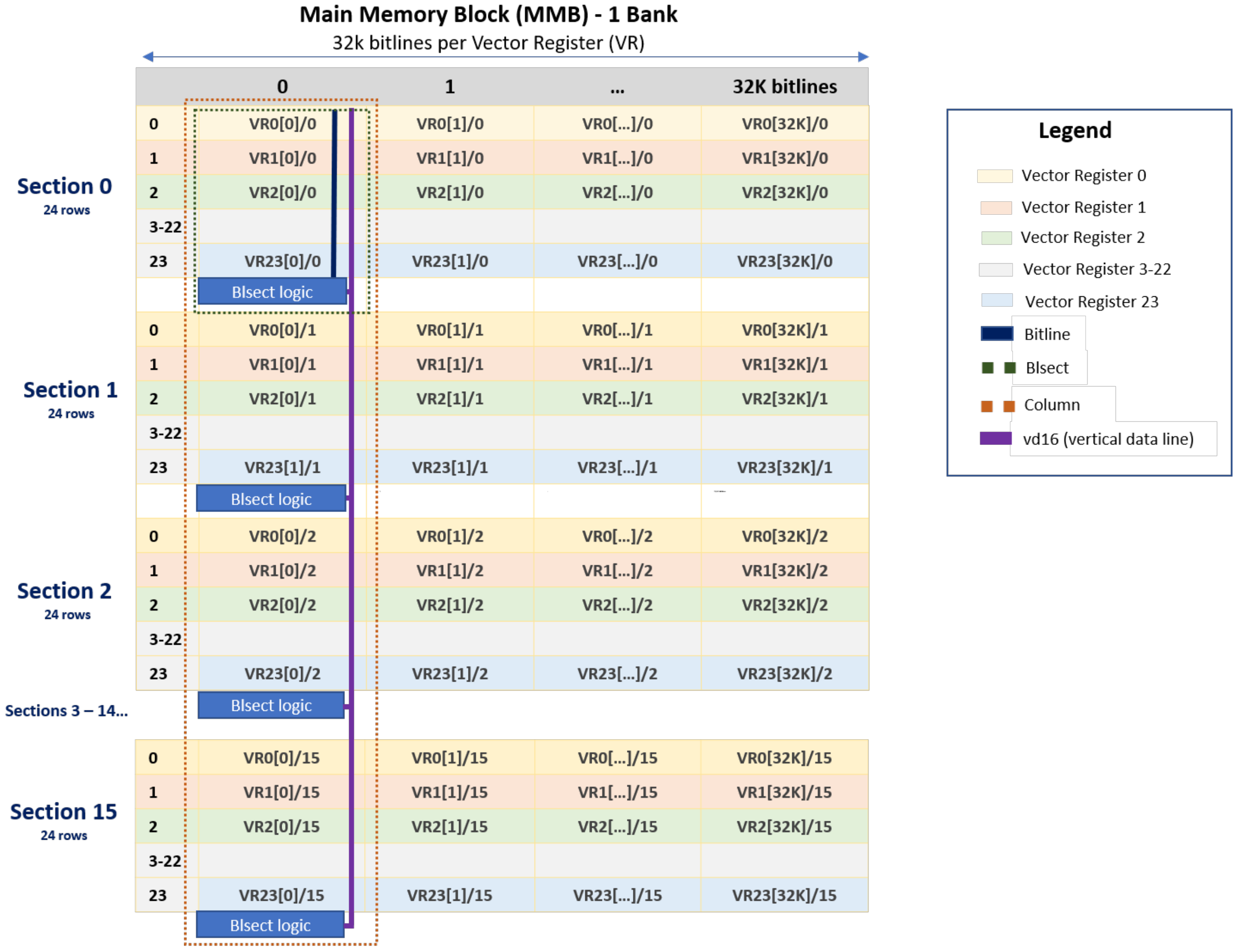
The MMB is the block of memory in the APC where computations take place. The MMB receives read/write control signals from the spreader in order to perform the desired operations by MMB, L1, and L2.

Architecture

The MMB is 1536 rows tall and 4096 columns (or bitlines) wide. All 6M memory cells in the MMB are constructed with custom 10T cells to support selective write functionality.

The 1536 rows in the MMB are sub-divided into 4 *banks* of 384 rows per bank. The 4 *banks* share the same set of read/write control signals and therefore always perform the same computations (on different data). The 384 rows in each MMB bank are sub-divided into 16 *sections*, each with 24 rows. Therefore, each bitline in an MMB *bank* is composed of 16 *bitline sections* (or *blsects*) containing 24 mcells/*blsect*. Each *section* receives a unique set of read/write control signals from the spreader.

The 16 *sections* in each MMB *bank* are sub-divided into 4 *groups* of 4 *sections*. Each MMB *group* is connected to a distinct L1 *group* via a dedicated vd4 vertical data line. Consequently, 4 L1-MMB data transfers can occur simultaneously per *bank* of L1 & MMB.



Main Memory Block (MMB) - 1 Bank

32k bitlines per Vector Register (VR)

		0	1	...	32K bitlines
Section 0 24 rows	0	VR0[0]/0	VR0[1]/0	VR0[...]/0	VR0[32K]/0
	1	VR1[0]/0	VR1[1]/0	VR1[...]/0	VR1[32K]/0
	2	VR2[0]/0	VR2[1]/0	VR2[...]/0	VR2[32K]/0
	3-22				
	23	VR23[0]/0	VR23[1]/0	VR23[...]/0	VR23[32K]/0
		Blsect logic			
Section 1 24 rows	0	VR0[0]/1	VR0[1]/1	VR0[...]/1	VR0[32K]/1
	1	VR1[0]/1	VR1[1]/1	VR1[...]/1	VR1[32K]/1
	2	VR2[0]/1	VR2[1]/1	VR2[...]/1	VR2[32K]/1
	3-22				
	23	VR23[1]/1	VR23[1]/1	VR23[...]/1	VR23[32K]/1
		Blsect logic			
Section 2 24 rows	0	VR0[0]/2	VR0[1]/2	VR0[...]/2	VR0[32K]/2
	1	VR1[0]/2	VR1[1]/2	VR1[...]/2	VR1[32K]/2
	2	VR2[0]/2	VR2[1]/2	VR2[...]/2	VR2[32K]/2
	3-22				
	23	VR23[0]/2	VR23[1]/2	VR23[...]/2	VR23[32K]/2
Sections 3 – 14...		Blsect logic			
Section 15 24 rows	0	VR0[0]/15	VR0[1]/15	VR0[...]/15	VR0[32K]/15
	1	VR1[0]/15	VR1[1]/15	VR1[...]/15	VR1[32K]/15
	2	VR2[0]/15	VR2[1]/15	VR2[...]/15	VR2[32K]/15
	3-22				
	23	VR23[0]/15	VR23[1]/15	VR23[...]/15	VR23[32K]/15
		Blsect logic			

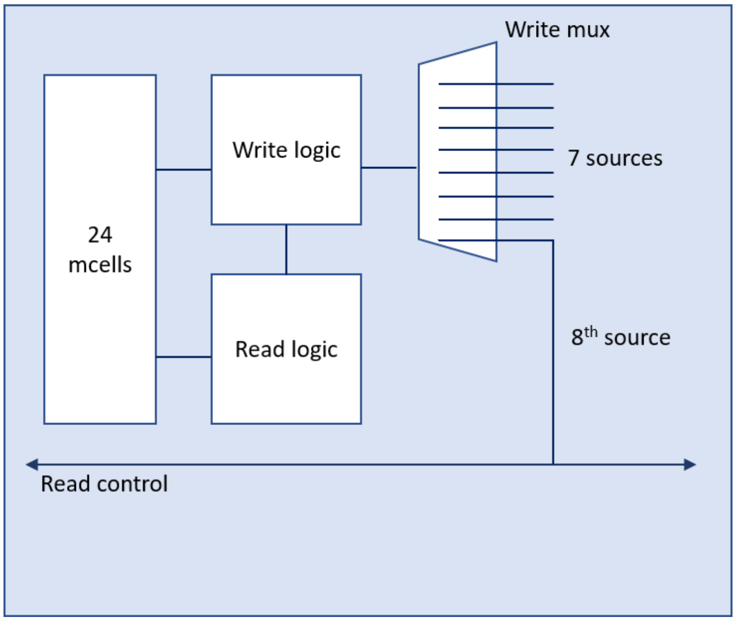
External Interfaces

Each *blsect* shares data and control interfaces (see under **Architecture**, above) with the rest of the MMB. Each *blsect* is connected to seven data interfaces: Four adjacent neighbor *blsects* (North/South/East/West), two vertical data lines (vd4/vd16), and one horizontal data line (RSP). These connections enable the *blsect* to drive the content of its read latch to any of the data interfaces, or source the content of any of the interfaces into its write logic.

Memory Cells

Each *blsect* includes 24 memory cells, one per vector register bit.

Bitline Section & External Interfaces



Write Logic

The write logic selects which input source to write (from eight sources) into the memory cells and which memory cells to write to. The output of the write logic is the write data and write controls.

The eight sources include the seven sources described previously (see External Interfaces, above) as the data interfaces of the *blsect*, and the *blsect's* self-read latch.

Read Logic

The read logic selects which memory cells to read data from, and which logical operations to do on the read data. The read logic operates on data from memory cells and from the write data. The output is stored in a read latch.

Memory

L1 Memory

L1 is the vector memory of the APU. It is a block of memory in the APC that serves as closely-coupled data storage/cache to the MMB. The L1 stores 96 Mbits of data per APU (not counting parity bits).

L1 receives addresses and read/write control signals from the spreader to control data transfers to and from the MMB and L2.

L1 is 3456 rows (wordlines) tall and 4096 columns (bitlines) wide. All 13.5M mcells in the L1 are constructed with standard 8T cells from the TSMC library. The 3456 rows in the L1 are sub-divided into 4 banks, each of 864 rows. Each bank of L1 is closely coupled to a corresponding bank of MMB. The 864 rows in each L1 bank are sub-divided into 4 sections, each with 216 rows.

Each section in an L1 bank also represents an L1 group (Thus, for L1, *section* and *group* are synonymous). Each L1 group is connected to a distinct MMB *group* via a dedicated vd4 vertical data line. Consequently, 4 L1 to MMB data transfers can occur simultaneously per bank of L1 & MMB.

All L1 control signals are driven by the spreader.

The ability to transfer data between L1 and MMB is provided via the vd4 data lines. The ability to transfer data between L1 and L2 is provided via the vd64 data lines.

L2 Memory

L2 is a block of memory in the APC that serves as a data buffer between the L2T and the L1. L2 receives addresses and read/write control signals from the spreader to control data transfers to and from the L1. L2 receives addresses and read/write control signals from the L2DMA to control data transfers to and from the L2T.

L2 is 72 rows (wordlines) tall and 4096 columns (bitlines) wide. It has enough rows to store exactly one L2T.

All 288K mcells in the L2 are constructed with standard 8T cells from the TSMC library. The 72 rows in the L2 comprise one *section*.

L2 control signals are driven by the spreader and L2DMA.

The L1 and L2 memory blocks are connected via a 4K bus.

L2 Transposers (L2T)

L2Ts are bi-directional shift registers in the APC that buffer data traffic between the system memory and the L2. They are used to transpose rows of system memory read data into columns of L2 write data, and to transpose columns of L2 read data into rows of system memory write data.

L2DMA

L2DMA is a control block in the APUC that controls the L2Ts and L2 in an APC during system memory to L2 data transfers. There are 2 L2DMAs per APUC—one per APC.

Vertical Data Lines

Four types of vertical data lines (vd) are utilized to transfer data within and between MMB, L1, L2, and L2T. There is a distinct set of 4 vd for each column (bitline) of MMB, L1, and L2. Each MMB, L1, and L2 *blsect* is connected to exactly 2 of the 4 vd. Each column of the L2T is connected to only 1 vd.

1. vdtr: For L2T – L2 transfers. '*tr'* indicates *transposer*.

2. vd64: For L2 – L1 (or possibly L1-L1) transfers. The '64' in 'vd64' indicates the number of MMB *blsects* crossed by any particular vd64 (none of which are connected to it).

3. vd4: For L1 – MMB or MMB – MMB transfers. The '4' in 'vd4' indicates the number of MMB *blsects* connected to any particular vd4.

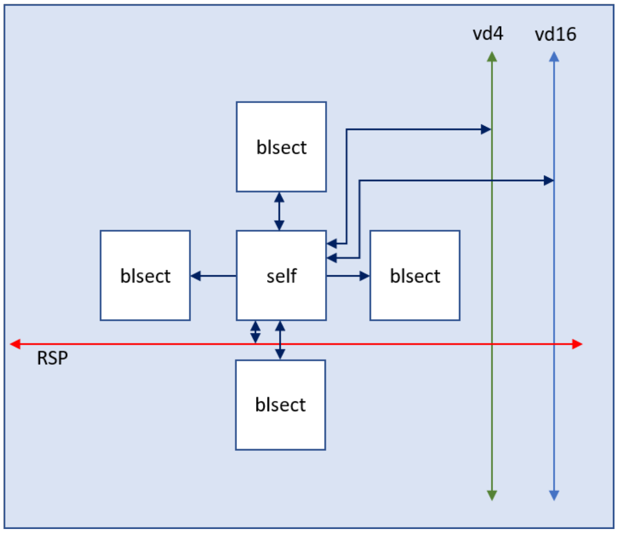
4. vd16: For MMB – MMB transfers. The '16' in 'vd16' indicates the number of MMB *blsects* connected to any particular vd16.

RSP: Horizontal Data Lines

RSP functionality provides the ability to logically OR data from multiple columns in an MMB section together to generate a single result. The software considers the single result as a response from the MMB—hence the abbreviation RSP.

The RSP is vital for providing software feedback on MMB computed outputs for the flow control and for reading back computed data from MMB as part of the algorithm.

MMB Bitline Section (blsect) Overview



Data Integrity

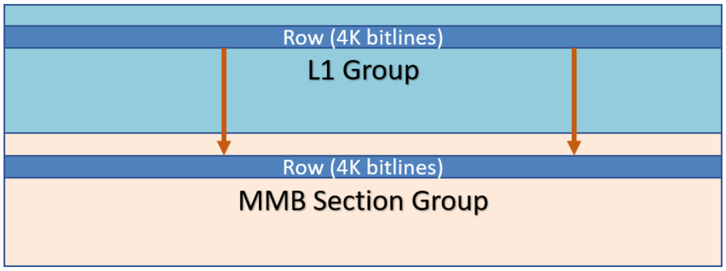
The APU uses industry standard data integrity solutions.

These include ECC-memory for the device memory (L4), APU memory (L3), CPU caches, and microcode I-MEM; and parity bits for the APU bus and vector memory (L1, L2). The Main Memory Block (MMB) is not protected by a parity bit or ECC because it is designed using periphery design rules in the same way as registers, and not with memory design rules. It is, therefore, not susceptible to cosmic ray-induced soft errors in the way that memory bit-cells are.

APU I/O Internal Data Movement

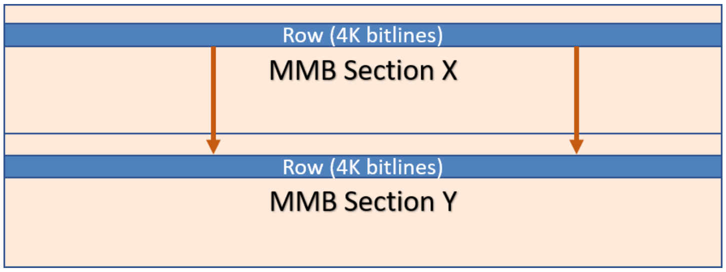
The microcode moves vectors between the L1 memory and the MMB

- Each transfer is 4K bit lines/group/bank
- 4 groups/bank/APC
- Total: 32 rows/128K bit lines transferred per I/O
- Moving a vector takes 4+1 I/O operations



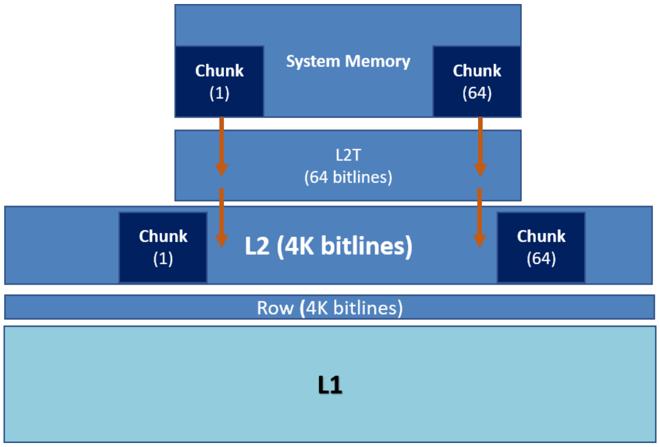
The microcode also moves data between MMB sections:

- Each transfer is 4K bit lines/bank
- 4 banks/APC
- Total: 8 rows/32K bit lines transferred per I/O



APU I/O External Data Movement

- SW controls vector movement between system memory and L1 memory
- L2DMA engine moves vectors between system memory and L2 buffer
- Microcode moves vectors between L1 memory and L2 buffer


System Memory
Chunk (1)
Chunk (64)
L2T (64 bitlines)
Chunk (1)
L2 (4K bitlines)
Chunk (64)
Row (4K bitlines)
L1