SLAP: A Split Latency Adaptive VLIW Pipeline Architecture which enables on-the-fly Variable SIMD Vector Length

Ashish Shrivastava, Alan Gatherer, Tong Sun, Sushma Wokhlu, Alex Chandra

The Problem

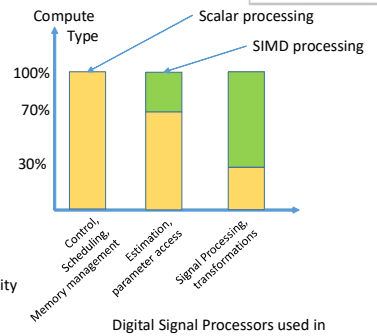
5G (even more than 4G) is a collection of disparate, *evolving* requirements

Short latency: URLLC, factory4.0

High Capacity: eMBB, Virtual Reality

Intense Processing: FFT, mMIMO

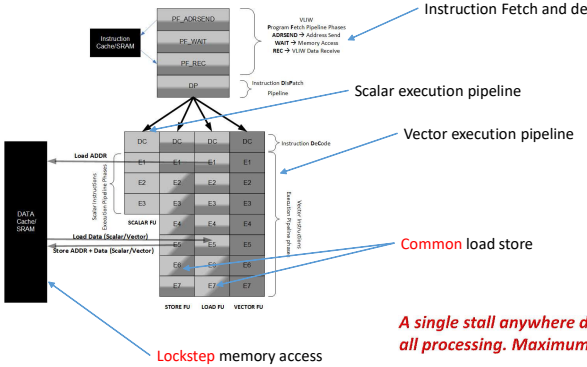
Complex Control: short TTI, mMTC, CatM



Digital Signal Processors used in Baseband Infrastructure support a mix of control and vector processing using Single Instruction Multiple Data (SIMD) architectures

How do we vary Scalar/SIMD ratio on the fly with little memory organization overhead?

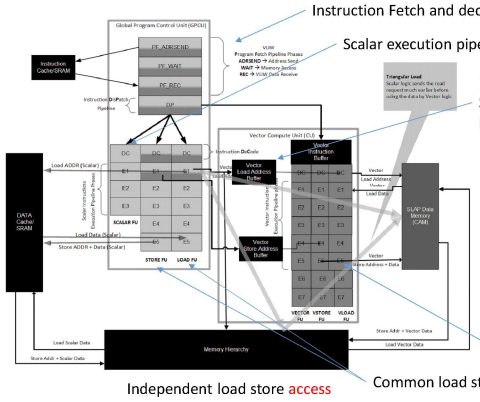
Evolving SIMD to SLAP: Current SIMD architecture



- SIMD Vector processing built for worst case vector load is often unused
- Control and vector data access often collide in L2
- Wide vector access has variable access time across vector to shared L2

A single stall anywhere due to memory access will stall all processing. Maximum access stall experienced

Evolving SIMD to SLAP: SLAP SIMD architecture

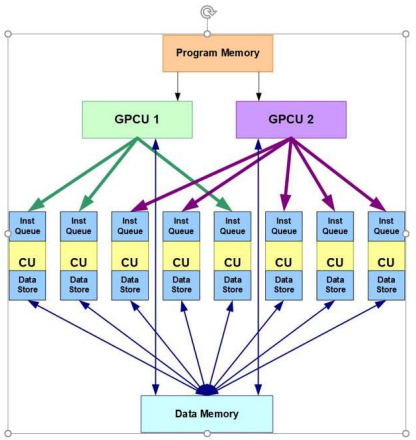


- Each lane of SIMD and the scalar unit process independently in time off a single instruction stream

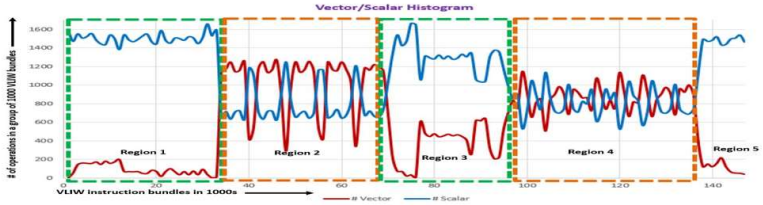
Average access stall is experienced as lanes run independently

SLAP based Variable SIMD Vector Architecture

- Single GPCU program fetch with asynchronous CU program execution.
- Variable length Asynchronous SIMD (in the form of CU) on single executable code
- Micro local DU memory access decouples from micro local CU execution
- Asynchronous execution of CUs /DUs, which increases parallelism and reduces stalls
- 0-L1 for vector, no data organization. CU natural lag to DU auto synchronizes compute/access
- Different Scalar and Vector access mechanism to L2
- Data access is randomized providing probabilistic parallelism



Results for Production Memory Traces



Performance improvements due to reduction in effective stalls with realistic L2 access.

FIFOs to implement elastic timing are low energy and size compared to, for instance adding more registers.

Name	% Improvement
Region 1	7.10%
Region 2	**30.50%**
Region 3	10.50%
Region 4	4.10%
Region 5	7.70%
All Regions (Overall)	**11.79%**

Results Compared to Increasing L1 cache

DSP Architecture	% Degradation (compare to flat memory)	% Reduction in Cache Degradation	% Improvement (compare to VLIW-DSP with 32K DataS + Memory Subsystem)	%Area improvement	Performance-Area Efficiency (Compare to VLIW-DSP with 32K DataS + Memory SubSystem)
VLIW-DSP (Flat memory)	0.00%	N/A	N/A	N/A	N/A
VLIW-DSP (32K DataS)	33.64%	0.00%	0.00%	0.00%	0.00%
SLAP_VLIW_1 (24-FIFO, 8K DataS)	26.12%	22.36%	8.04%	6.40%	5.79%
SLAP_VLIW_2 (24-FIFO, 16K DataS)	23.69%	29.60%	9.81%	2.69%	7.39%
SLAP_VLIW_3 (24-FIFO, 32K DataS)	22.42%	33.36%	10.74%	-4.74%	7.75%
SLAP_VLIW_4 (32-FIFO, 8K DataS)	25.13%	25.32%	9.15%	4.88%	6.41%
SLAP_VLIW_5 (32-FIFO, 16K DataS)	22.79%	32.26%	10.85%	1.16%	7.88%
SLAP_VLIW_6 (32-FIFO, 32K DataS)	21.49%	36.13%	11.79%	-6.27%	8.22%

- Memory stalls can be mitigated in a classic architecture by increasing the L1 cache size
- In our comparison with real 5G algorithms we compared classic SIMD with 32KB data cache to variations of cache and FIFO size for SLAP
- Performance/area improvements of up to 8% are observed with improvements with as little as 8KB L1 cache.