DESIGN AUTOMATION CONFERENCE

59

JULY 10 - 14, 2022

MOSCONE WEST CENTER

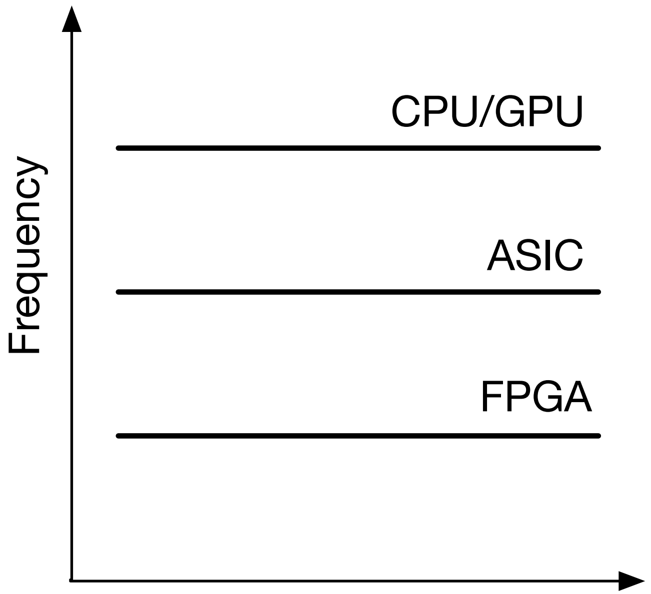
SAN FRANCISCO, CA, USA

Serpens: A High Bandwidth Memory Based Accelerator for General-Purpose Sparse Matrix-Vector Multiplication

Linghao Song, Yuze Chi, Licheng Guo, Jason Cong
University of California, Los Angeles

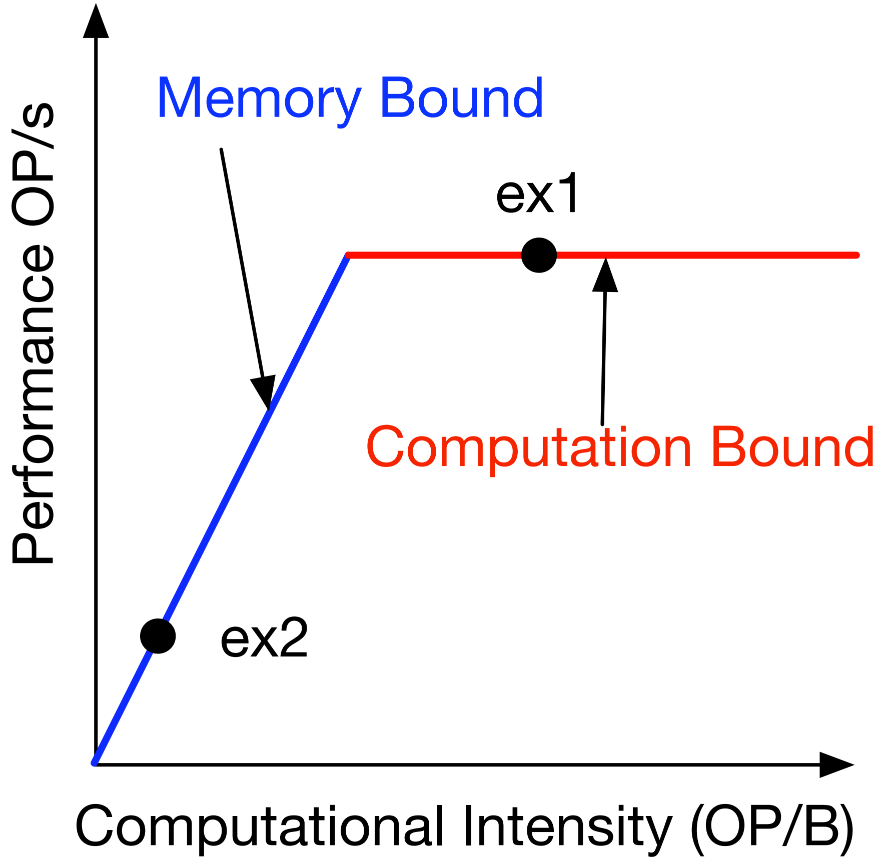
FPGAs as computing devices: the bad news

- Peak Performance:
  - $= \frac{1}{\text{Execution Time}} = \frac{\text{Frequency} \times \text{Instruction(Operation) per Cycle}}{\text{\#Instruction(Operation)}} \propto$ Frequency
- Typical frequency of computing devices:
  - CPUs: Intel Xeon Phi, 1.3 – 1.7 GHz.
  - GPUs: Nvidia V100, 1.23 GHz.
  - ASICs: Google TPU, 800 MHz.
  - FPGAs: 100 – 300 MHz.
- Low frequency -> low performance?



FPGAs as computing devices: the opportunity

- Roofline mode: to help model the performance bottleneck.
  - Bandwidth
  - Peak performance
  - Computational intensity
- Ex1 (computation bound):
  - bdw=10GB/s, PeakPerf=20 GOP/s, Comp.Intsty=10 OP/B
  - -> requires high frequency
- Ex2 (memory bound):
  - bdw=10GB/s, PeakPerf=20 GOP/s, Comp.Intsty=1 OP/B
  - -> requires relatively low frequency because of low computation intensity



Williams, Samuel, Andrew Waterman, and David Patterson. "Roofline: an insightful visual performance model for multicore architectures." *Communications of the ACM* 52.4 (2009): 65-76.

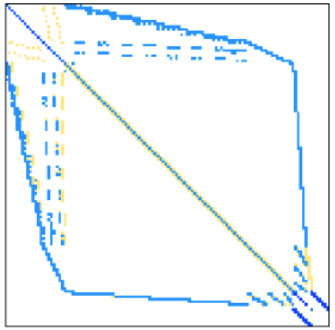
FPGAs as computing devices: the platform & workload

- The computing device: FPGAs (with relatively low working frequency)
- The workload: low computation intensity
- But, we still demand high attainable performance
  - -> high memory bandwidth

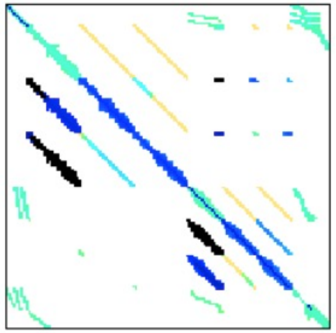
- Sparse matrix-vector multiplication (**low computational intensity**) acceleration on FPGAs with **high bandwidth memory**! -> Perfect!
  - Computational intensity
    - Resnet-50: 20 FLOP/B v.s. SpMV: 0.17 FLOP/B
  - Bandwidth
    - VCU1525: 77 GB/s v.s. U280: 460 GB/s

Sparse Matrix Vector Multiplication (SpMV)

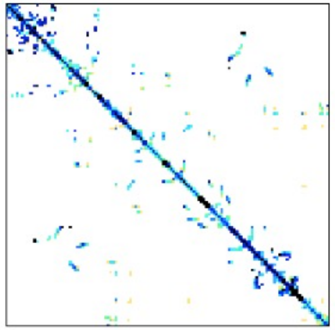
- SpMV: $y = \alpha \cdot A \times x + \beta \cdot y$
  - A: sparse matrix,
  - x, y: dense vectors,
  - α, β: scalar constants.
- SpMV is a key kernel in many applications:
  - Graph processing: scatter-gather processing model,
  - Scientific computing: e.g. iterative solvers.



Sandia/fpga_dcop_01 circuit simulation matrix. Sandia National Lab.



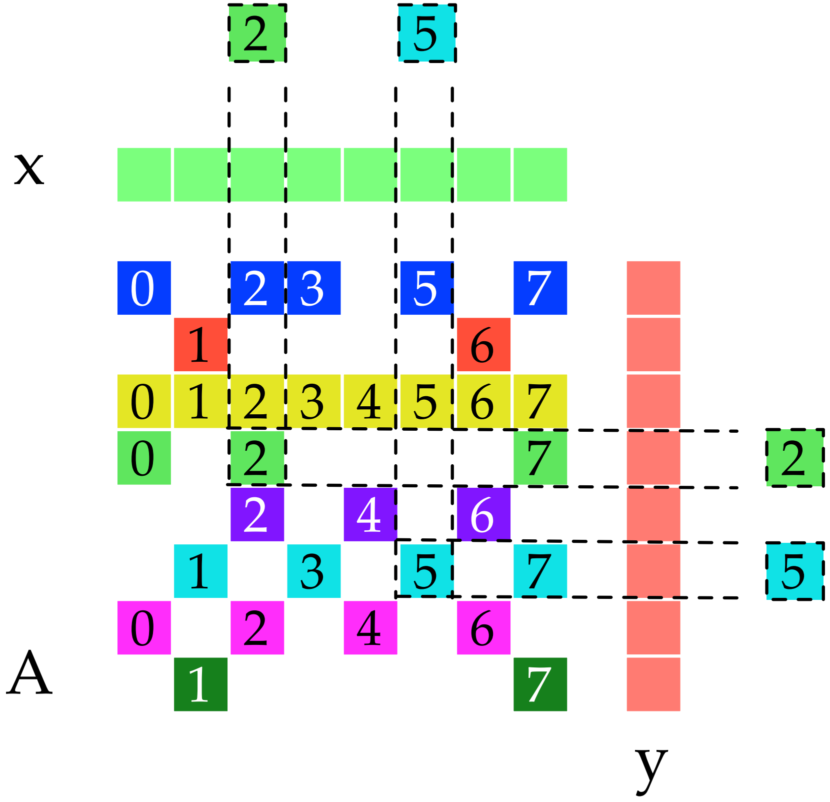
VLSI/nv1 VLSI: semiconductor device and process simulation



Boeing/bcsstk35 STIFFNESS MATRIX, AUTOMOBILE SEAT FRAME AND BODY ATTACHMENT. Boeing

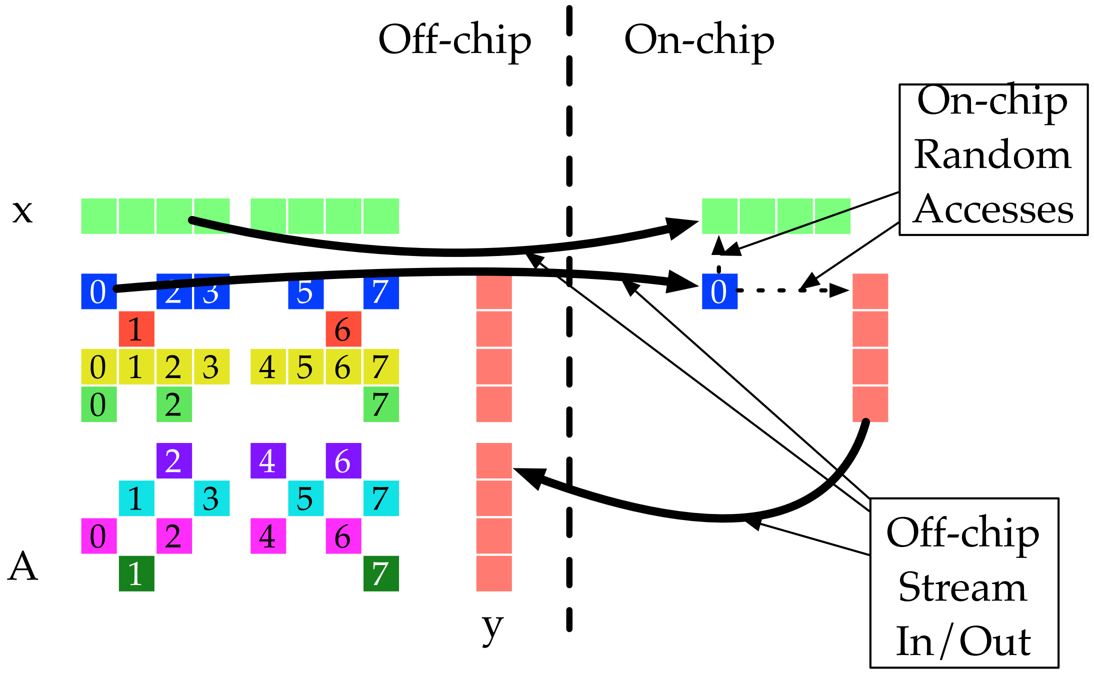
(Sparse matrices from SuitSparse)

Architecture level: Inefficient memory accessing in SpMV



- Suppose the matrix A and vectors x, y are stored in off-chip memory.
- Green 2 -> Lightblue 5:
  - Rd A2 -> Rd A5: random off-chip read,
  - Rd x2 -> Rd x5: random off-chip read,
  - Rd y2, St y2 -> Rd y5, St y5: random off-chip read and write.
- Extensive irregular off-chip accesses -> inefficient!

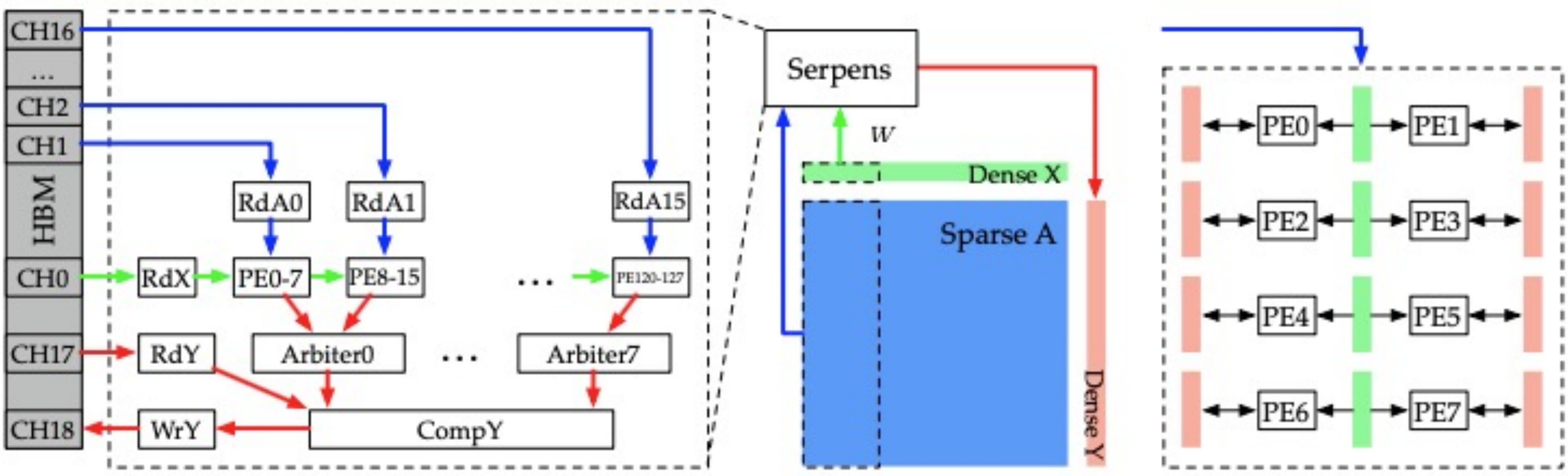
Serpens SpMV processing model



(Song+ HPCA'18; Song+ FPGA'22)

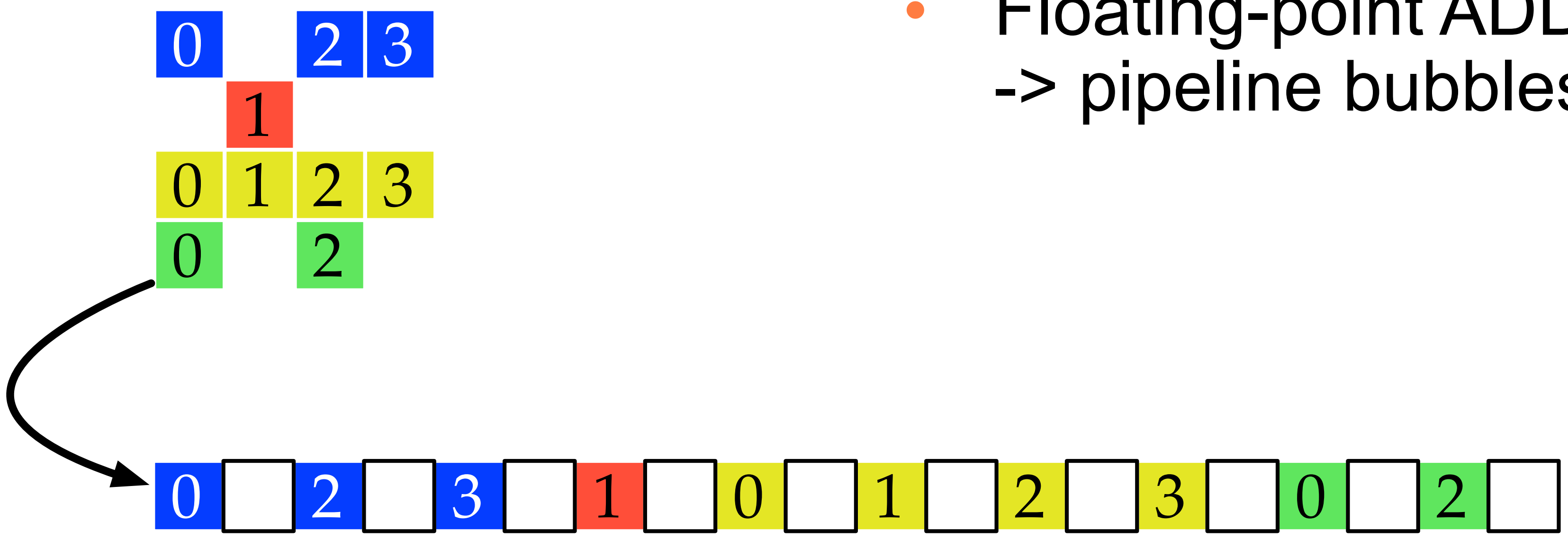
- Partition the A matrix:
  - Keep x, y segments on chip,
  - Stream in A blocks
  - ->Random accesses are always on chip and off-chip memory accesses are sequential streaming read/write.
- Leverage high bandwidth memory (HBM) for memory-level parallelism
  - Memory-centric processing engines

Serpens accelerator: hierarchical architecture



(a) Accelerator architecture. (b) Overall processing. (c) 8 PEs connected to one A channel.

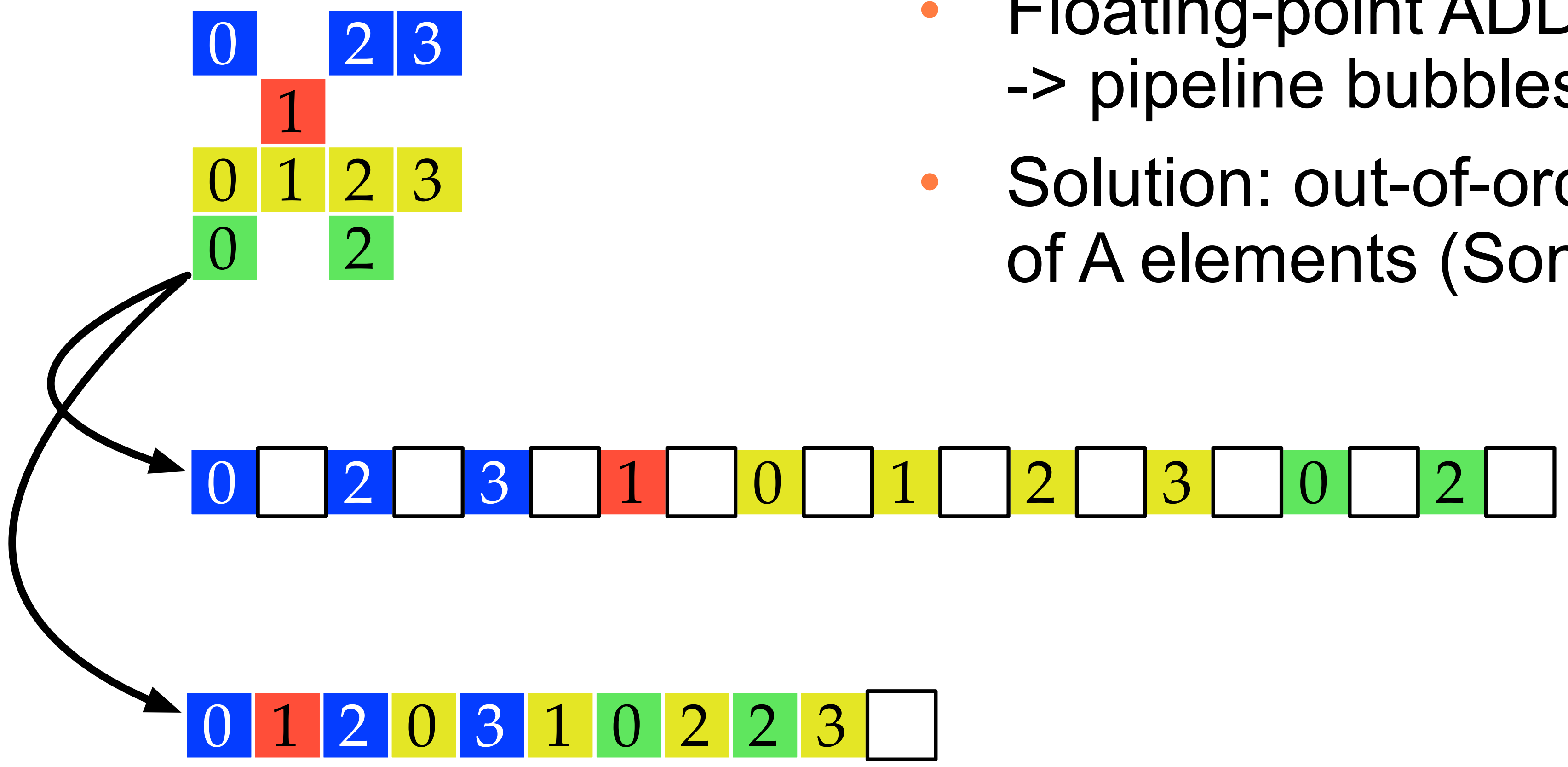
Micro-architecture level:
FP conflict and on-chip memory waste



- Floating-point ADD latency (>1 cycles) -> pipeline bubbles.

Micro-architecture level:
FP conflict and on-chip memory waste

- Floating-point ADD latency (>1 cycles) -> pipeline bubbles.
- Solution: out-of-order (OoO) scheduling of A elements (Song+ FPGA'22).



Micro-architecture level: FP conflict and on-chip memory waste

- However,
  - On-chip FPGA URAM data width 72,
  - FP32 data width: 32,
  - Storing 1 FP32 to 1 URAM entry -> memory waste.
- Serpens solution
  - Store 2 FP32 to 1 URAM entry
  - -> Leading to FP accumulation conflicts on the two FP32.
  - Solution: marking them as conflict and then send to OoO scheduling.

Evaluation

Table 2: The specification of the evaluated accelerators.

	Sextans [27]	GraphLily [18]	SERPENS	Tesla K80
Frequency	197 MHz	166 MHz	223 MHz	562 MHz
Bandwidth	&417 GB/s	&285 GB/s	&273 GB/s	#480 GB/s
Power	52 W	43 W	48 W	130 W

& Utilized bandwidth, # maximum bandwidth.

Table 6: Resource utilization of Sextans, GraphLily, and SERPENS-A16 on a Xilinx U280 FPGA board.

	LUT	FF	DSP	BRAM	URAM
Sextans	331K(29%)	594K(25%)	3233(36%)	1238(68%)	768(80%)
GraphLily	390K(35%)	493K(21%)	723(8%)	417(24%)	512(53%)
SERPENS	173K(15%)	327K(14%)	720(8%)	655(36%)	384(40%)

Table 3: The specification of evaluated matrices.

Twelve Large Matrices/Graphs			
ID	**Matrix**	**#Vertices**	**#Edges**
G1	googleplus [20]	108 K	13.7 M
G2	crankseg_2 [10]	63.8 K	14.1 M
G3	Si41Ge41H72 [10]	186 K	15.0 M
G4	TSOPF_RS_b2383 [10]	38.1 K	16.2 M
G5	ML_Laplace [10]	377 K	27.6 M
G6	mouse_gene [10]	45.1 K	29.0 M
G7	soc_pokec [20]	1.63 M	30.6 M
G8	coPapersCiteseer [10]	434 K	21.1 M
G9	PFlow_742 [10]	743 K	37.1 M
G10	ogbl_ppa [17]	576 K	42.5 M
G11	hollywood [20]	1.07 M	113 M
G12	ogbn_products [17]	2.45 M	124 M

SuiteSparse [10] Matrices			
Number of Matrices	2,519	**NNZ**	1,000 – 89,306,020
Row/column	24 – 2,999,349	**Density**	8.75E-7 – 1

Hu, Yuwei, et al. "GraphLily: Accelerating Graph Linear Algebra on HBM-Equipped FPGAs." ICCAD 2021.

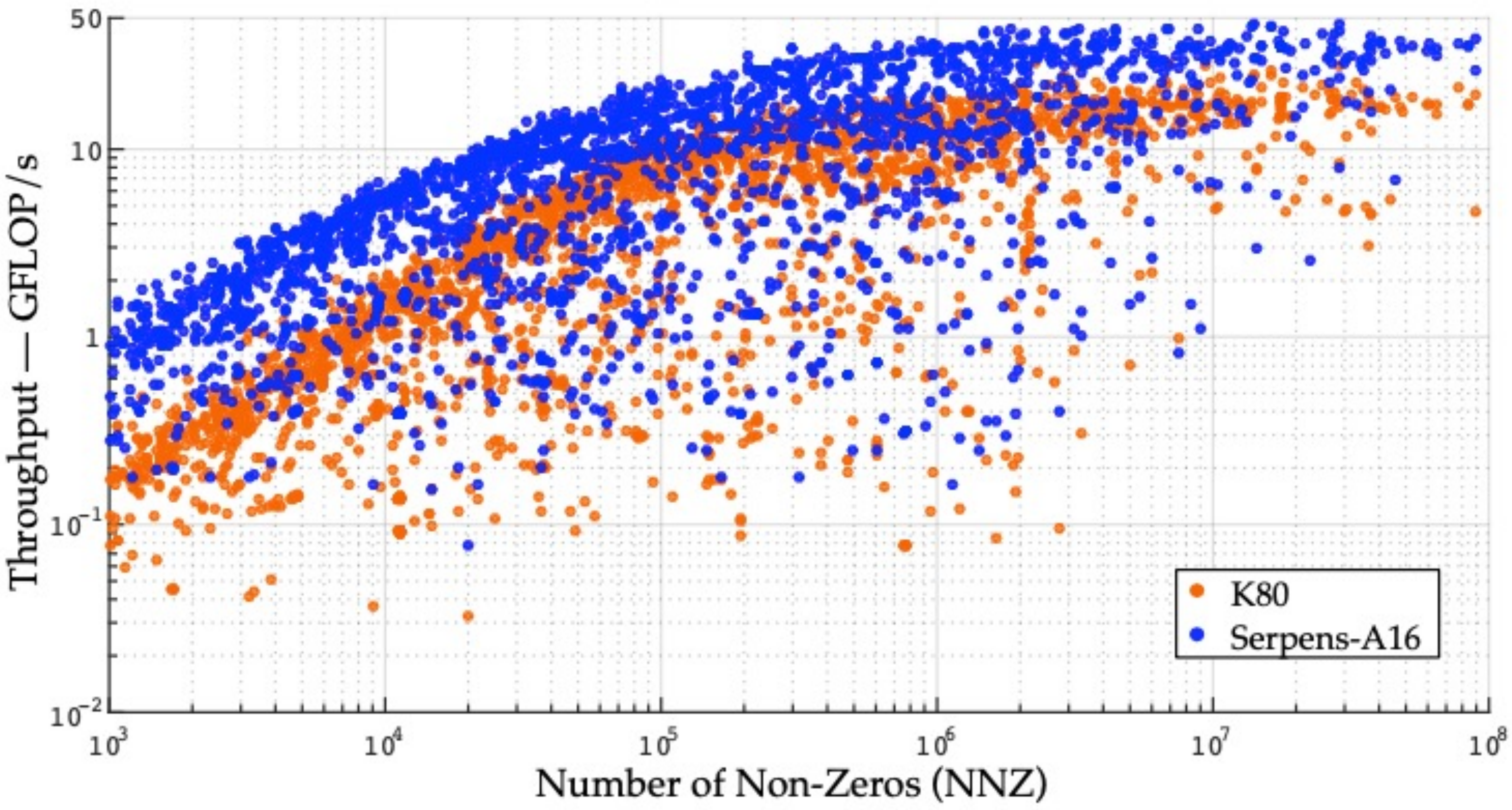
Song, Linghao, et al. "Sextans: A Streaming Accelerator for General-Purpose Sparse-Matrix Dense-Matrix Multiplication." FPGA 2022.

Performance comparison of Serpens, GraphLily(ICCAD'21) and Sextans(FPGA'22)

		G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	**GMN**
Execution Time: ms	**Sextans**	3.06	1.38	1.64	1.36	2.73	2.72	–	3.58	–	–	–	–	2.20
	GraphLily	1.73	1.47	1.85	1.57	2.96	2.80	7.04	3.63	4.52	4.59	12.4	18.6	3.74
	SERPENS-A16	1.87	0.930	0.853	0.730	1.37	1.37	4.52	2.09	2.05	2.04	6.20	6.32	1.96
Throughput: GFLOP/s	**Sextans**	9.01	20.60	18.55	23.81	20.47	21.33	-	18.14	-	-	-	-	18.15
	GraphLily	15.96	19.36	16.44	20.64	18.87	20.69	9.17	17.90	16.75	18.74	18.36	13.60	16.86
	SERPENS-A16	14.71	30.56	35.62	44.39	40.75	42.26	14.29	31.06	37.01	42.26	36.70	39.90	32.21
Throughput: MTEPS	**Sextans**	4,470	10,255	9,162	11,878	10,099	10,651	–	8,951	–	–	–	–	9,005
	GraphLily	7,920	9,639	8,117	10,296	9,305	10,331	4,352	8,828	8,212	9,243	9,094	6,668	8,310
	SERPENS-A16	7,300	15,214	17,594	22,144	20,099	21,098	6,782	15,324	18,142	20,847	18,176	19,565	15,876
	Improvement	0.922×	1.58×	2.17×	2.15×	2.16×	2.04×	1.56×	1.74×	2.21×	2.26×	2.00×	2.93×	1.91×
Bandwidth Efficiency: MTEPS/(GB/s)	**Sextans**	10.7	24.6	22.0	28.5	24.2	25.5	–	21.5	–	–	–	–	21.6
	GraphLily	27.8	33.8	28.5	36.1	32.7	36.2	15.3	31.0	28.8	32.4	31.9	23.4	29.2
	SERPENS-A16	26.7	55.7	64.4	81.1	73.6	77.3	24.8	56.1	66.5	76.4	66.6	71.7	58.2
	Improvement	0.962×	1.65×	2.26×	2.25×	2.25×	2.13×	1.63×	1.81×	2.31×	2.35×	2.09×	3.06×	1.99×
Energy Efficiency: MTEPS/W	**Sextans**	86.0	197	176	228	194	205	–	172	–	–	–	–	173
	GraphLily	184	224	189	239	216	240	101	205	191	215	211	155	193
	SERPENS-A16	152	317	367	461	419	440	141	319	378	434	379	408	331
	Improvement	0.826×	1.41×	1.94×	1.93×	1.94×	1.83×	1.40×	1.56×	1.98×	2.02×	1.79×	2.63×	1.71×

Up to 3.79x performance gain over GraphLily.

Comparison to K80 GPU



- Bandwidth:
  - Serpens: 273 GB/s,
  - K80: 480 GB/s.
- Frequency:
  - Serpens: 223 MHz,
  - K80: 562 MHz.
- Power:
  - Serpens: 48 W,
  - K80: 130 W.
- Serpens / K80:
  - Performance: 2.31x,
  - Energy efficiency: 6.25x.

Comparison with other SpMV accelerators

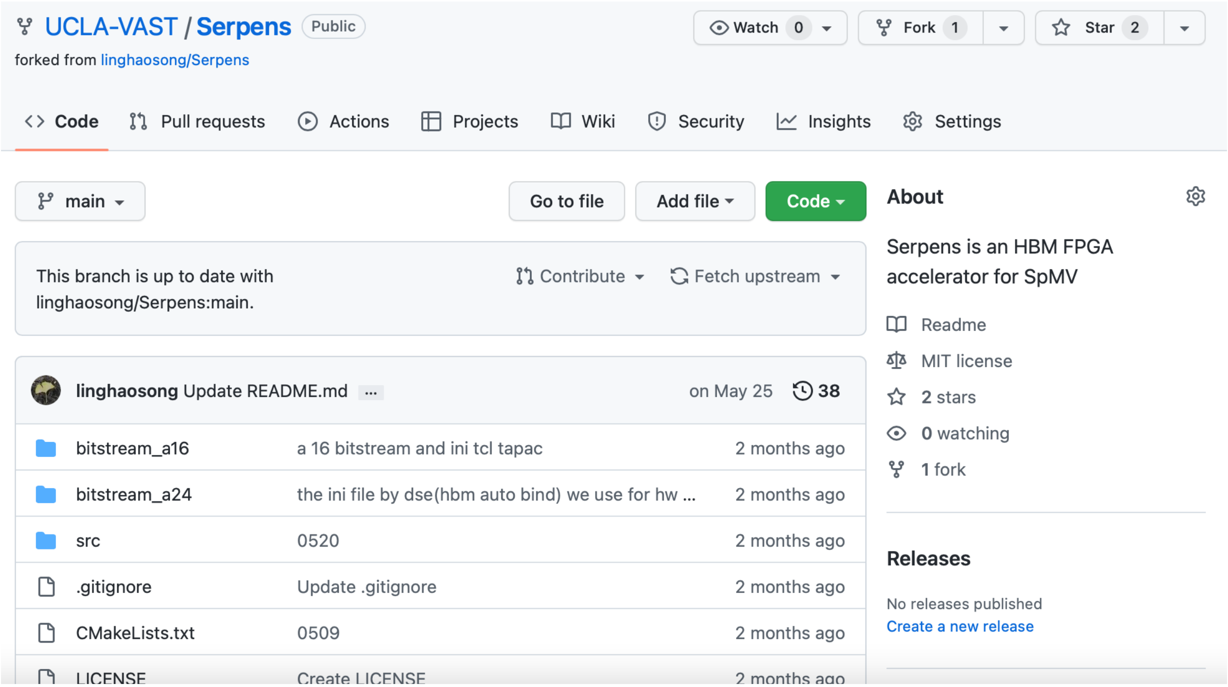
	Bandwidth	Peak Performance
Serpens-A16	273 GB/s	44.2 GFLOP/s
Serpens-A24	388 GB/s	60.4 GFLOP/s
Du et al, FPGA'22 [11]	258 GB/s	25.0 GFLOP/s
Sadi et al, MICRO'19 [25]	357 GB/s	34.0 GFLOP/s
SparseP [13]	1770 GB/s	4.66 GFLOP/s

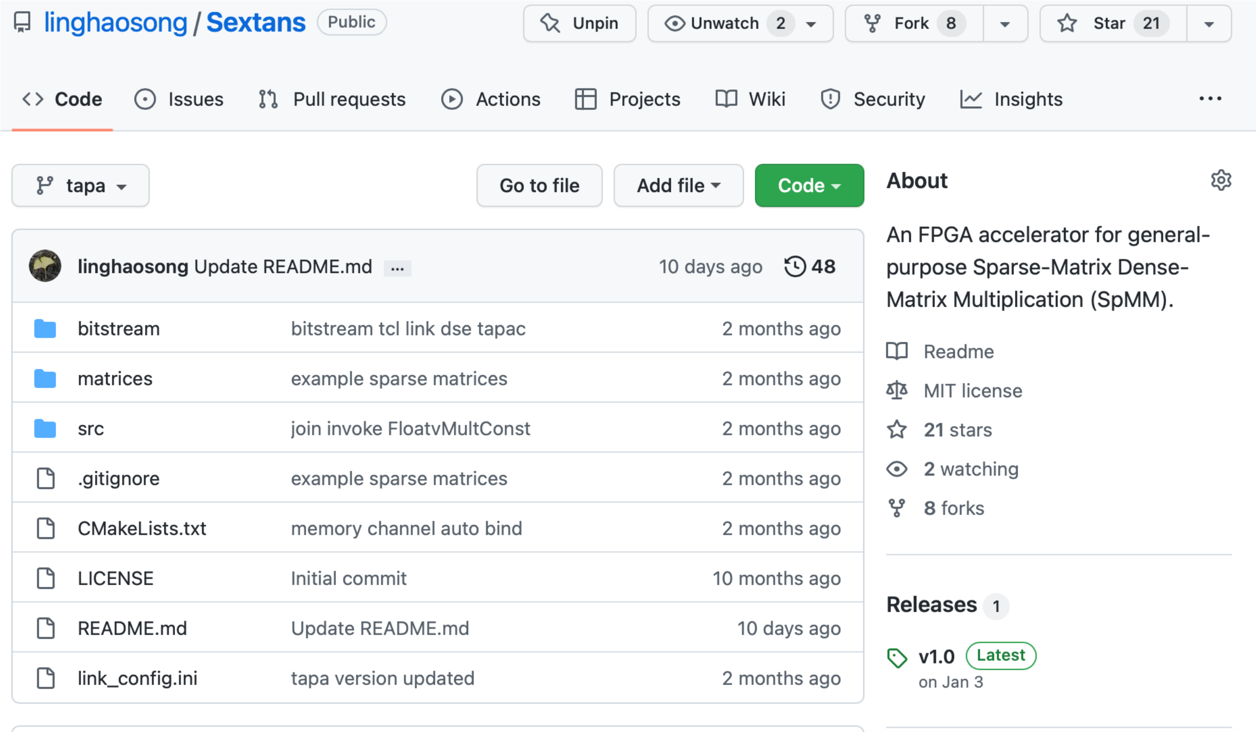
[11] Yixiao Du et al. "High-Performance Sparse Linear Algebra on HBM-Equipped FPGAs Using HLS: A Case Study on SpMV". FPGA'22.

[13] Christina Giannoula et al. "SparseP: Towards Efficient Sparse Matrix Vector Multiplication on Real Processing-In-Memory Systems". SIGMETRICS'22.

[25] Fazle Sadi et al. "Efficient SpMV Operation for Large and Highly Sparse Matrices using Scalable Multi-way Merge Parallelization". MICRO'19.

Source code and bitstream publicly available





DOI 10.5281/zenodo.6555139

https://github.com/UCLA-VAST/Serpens

https://github.com/linghaosong/Sextans

XACC Cluster: https://xilinx.github.io/xacc/ucla.html

Thanks!
Q&A

https://linghaosong.github.io/