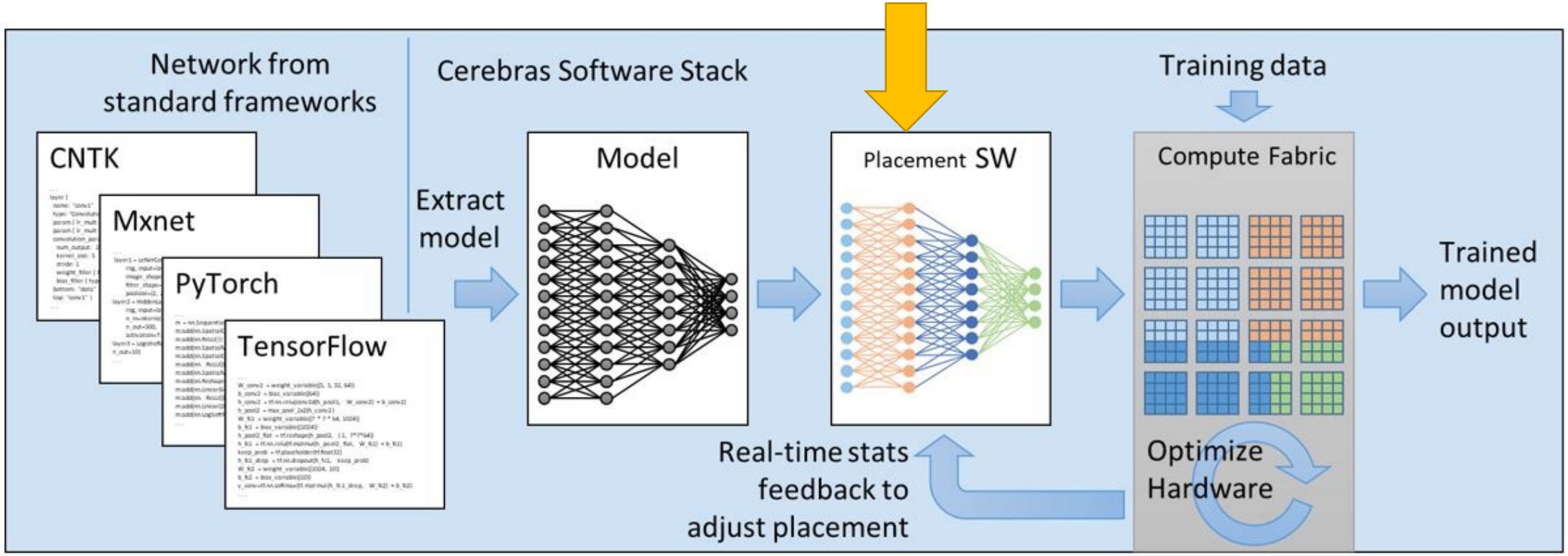
Kernel Mapping Techniques for Deep Learning Neural Network Accelerators

Sarp Özdemir, Mohammad Khasawneh, Smriti Rao, Patrick Madden

SUNY Binghamton CSD

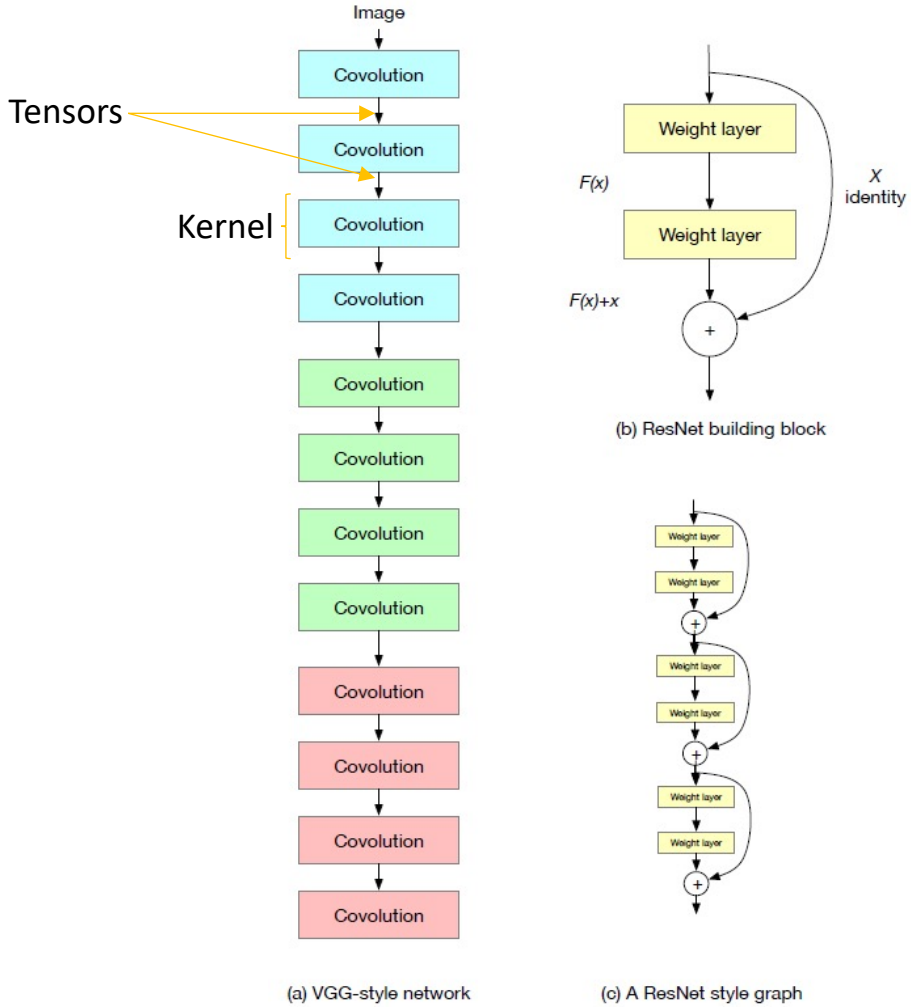
Machine Learning Accelerators: An Overview



Source: https://www.servethehome.com/

Deep learning kernels

- Deep learning applications are built by connecting tens or hundreds of layers of kernels
- Kernels are connected to form a “kernel graph”
  - Many different styles



Kernels and Cerebras CS-1

- Maximizing performance of deep learning applications on CS-1 involves minimization the maximum delay (δ_T) of any compute kernel
- Kernels are compiled and assigned to rectangular grid of compute cores
- Unrolling parallelism of compute kernels:
  - Less time for kernel execution
  - More cores required to perform their operations

Kernels

- Compute cores have fixed amount of local memory
  - Kernels that require more memory are pruned
  - Constrains the important measures of kernel performance to height, width, and time delay
  - We use tuples [h, w, t] (a “performance cube”) to denote performance of a variant

- An implementation of the kernel is a “variant”
  - Large number of possible variants exists
  - very complex solution space for search problem

Kernel Mapping Contest

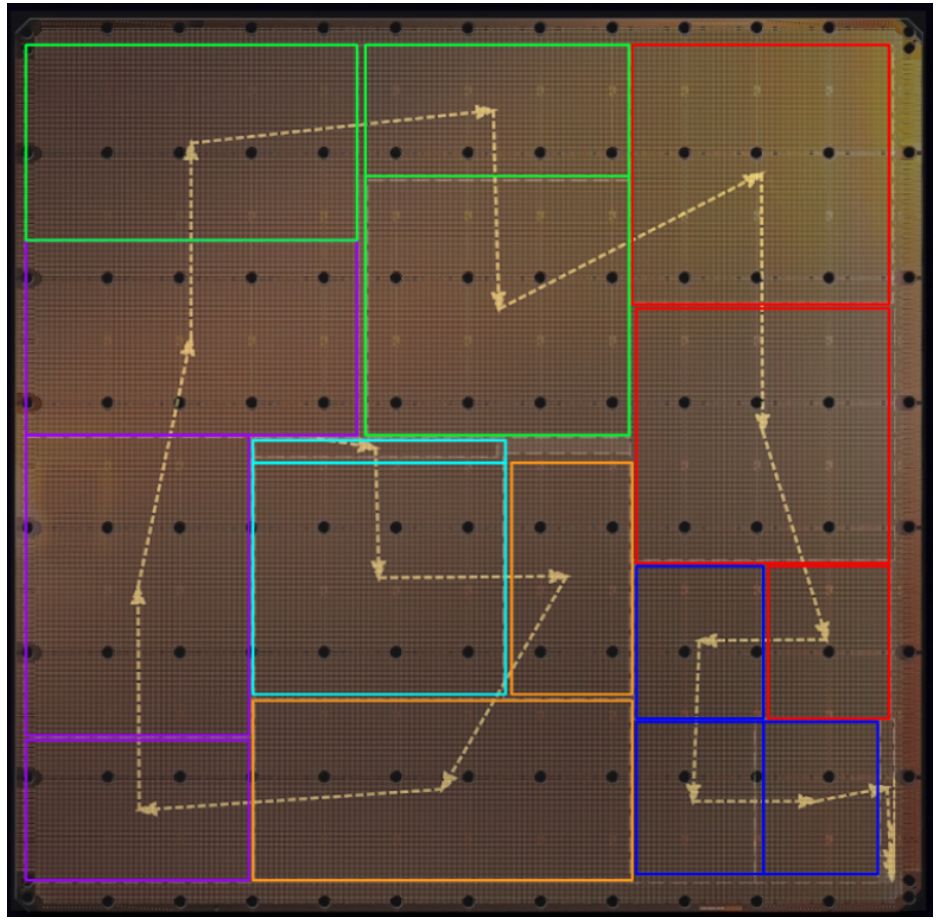


Figure from Cerebras

- For the ISPD contest, the goal was to size each kernel to maximize performance, and then place them to minimize wire length
- Very large potential size differential between kernels

Rent's Rule

- A mathematical measure of a system's complexity
  - Observed by correlating number of connections and a module's internal components.
  - $P = KB^r$
    - P: # External pins of a system
    - K: Average # of pins per block
    - r: Rent's parameter, complexity coefficient of system
  - r==1 → poor (random) organization of system
  - Lower r → better decomposition of the system into modules
- r for deep learning architectures is 0
  - Single tensor flows in and out of any computer kernel

Kernel Mapping

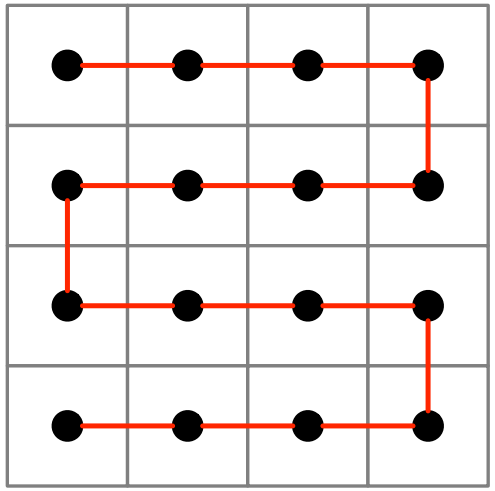
- ISPD contest tracked metrics:
  - δ_T: Max. delay of any kernel.
  - L1 distance between kernel centers → less important, but worthwhile
  - Possible adapters due to difference in loop unrolling → We omit
- The low Rent parameter invites a topological approach for mapping
  - Winners of ISPD contest (CU.POKer) reached the same conclusion as our group
  - CU.POKer team ordered kernels according to a topological traversal.
    - Selected a target δ_T
    - Used binary search to consider different delays until a feasible implementation is found.

Kernel Mapping (2)

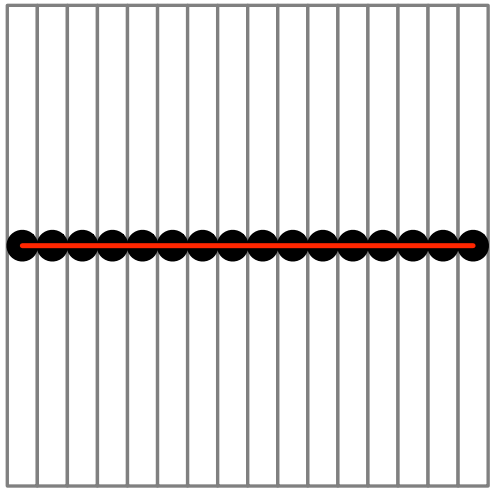
- For each δ_T, the chip area is divided into one or more horizontal rows.
- If a given benchmark can be implemented using a single row, then CU.POKer can find a solution very efficiently.
  - Most contest benchmarks fall into this category.
- If, however, the implementation requires more than one row the solution space explodes.
  - One row: Combinations of cores h_1 tall are considered.
  - Two rows: Combinations of cores h_1 tall , and $(633 – h_1)$ tall.
  - Three rows: Combinations of cores of h_1, h_2, and $(633 – h_1 + h_2)$ height are considered, etc.

Kernel Mapping (3)

- ISPD performance measurements does not penalize designs that implement kernels with high aspect ratios.
- Real-world considerations expected to favor designs that use large number of rows and roughly square kernel implementations.



(a) Low wire lengths within each kernel, high inter-kernel wire lengths

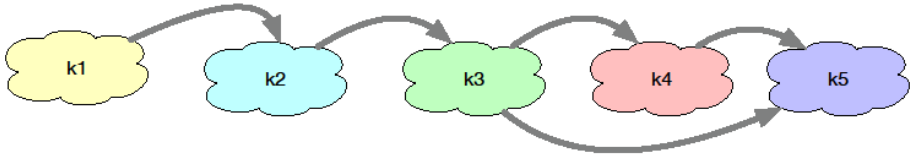


(b) High wire lengths within each kernel, low inter-kernel wire lengths

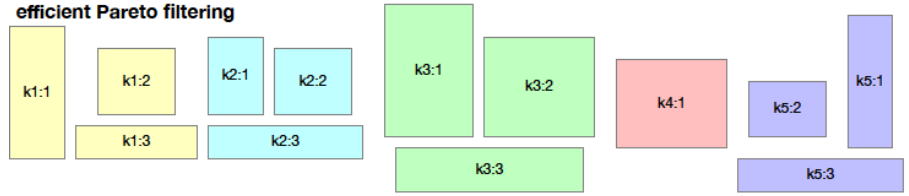
Our Approach

- Key advantages:
  - Support for more complex performance models
  - Multi-row kernel mapping implementations without computational explosion.

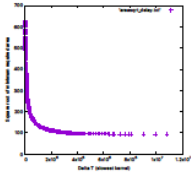
(a) Deep Learning Kernel Graph - ordered by breadth-first search



(b) Non-dominated Kernel Implementations - brute force enumeration, followed by efficient Pareto filtering

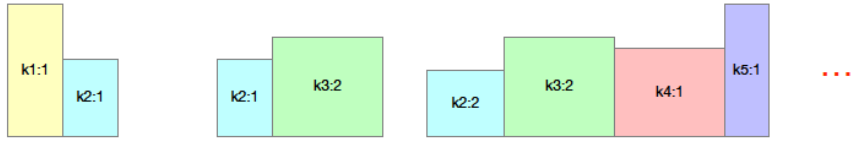


(c) Using the minimum area required for each kernel, for any feasible speed, the area demands for the complete kernel graph can be identified. This leads to a bound on optimal performance in terms of system speed

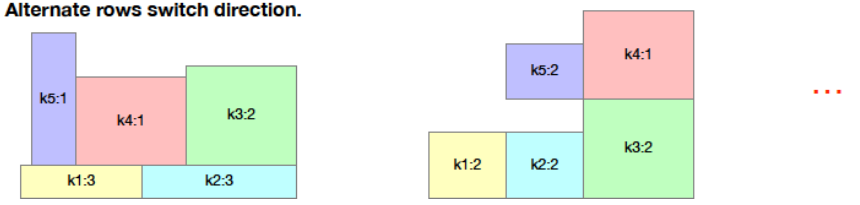


We perform binary search, seeking the fastest possible implementation. When multiple solutions are available, we prefer the solution with minimum wire length.

(d) Optimal single-row strips of kernels - produced with dynamic programming. For each i-to-j range, there is a set of non-dominated strips.



(e) Optimal stacks constructed from strips - produced with dynamic programming. Alternate rows switch direction.

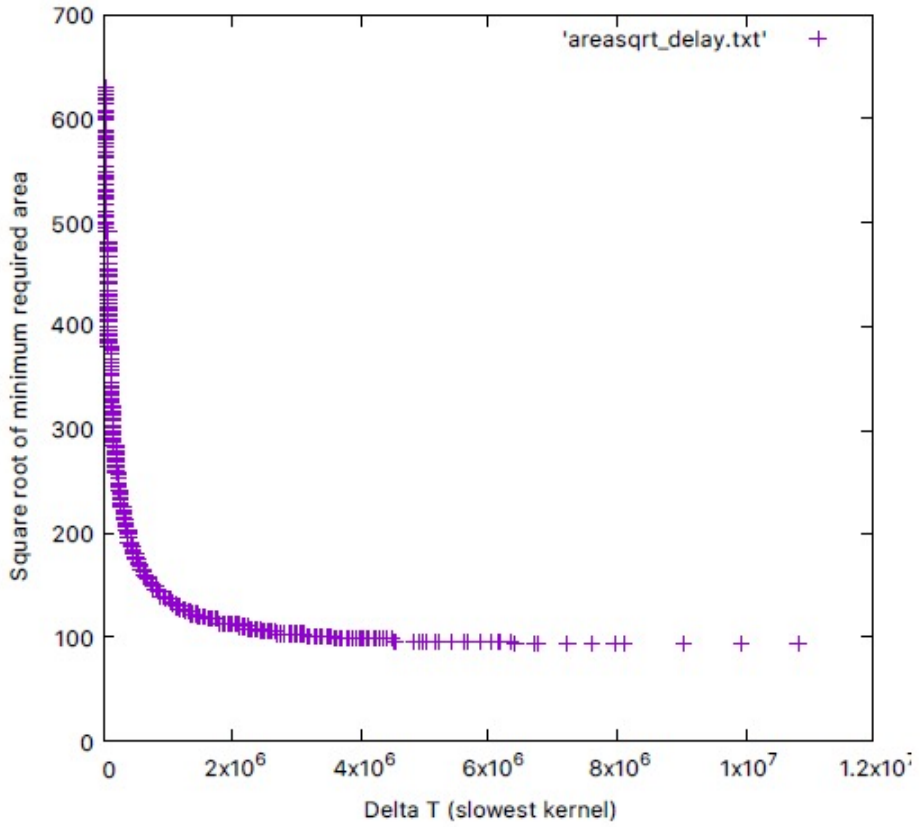


Finding Kernel Variants

- CU.POKer uses numerical methods to find optimal kernel variations for a height target
- We use a simple brute-force method
  - hundreds of thousands of variants
  - Filtered with an efficient implementation of Bentley's divide and conquer
  - Look for the non-dominated kernel implementations
- Variant ***a*** with performance cube $[h_a, w_a, t_a]$ dominates variant ***b*** with performance cube $[h_b, w_b, t_b]$ if:
  - $H_a <= h_b$, AND
  - $W_a <= w_b$, AND
  - $T_a <= t_b$
- Variants are quickly reduced to thousands.

Establishing performance bounds

- By finding non-dominated kernels, we can establish lower bound on δ_T.
- Each potential delay can be correlated with area (#cores) needed to achieve it.



Dynamic Programming – Strips and Stacks

- Horizontal strips
  - All variants of kernel *i* to *i+1* are found by brute force
  - This list of kernels is then reduced to a non-dominated set
  - Using dynamic programming, we construct strips from kernel *i* to kernel *j*
- Stacking vertically
- The entire cluster graph is kernel k_1 to k_n
  - Find the best stacking for kernel k_1 to k_i, and k_{i+1} to k_n
  - Recursive dynamic programming approach, very similar to matrix chain multiplication

Pruning the Solution Space

- Manage computational explosion, AND good solution quality.
- Multiple constraints:
  - Kernel variants exceeding target speed (δ_T) are discarded.
  - Kernels exceeding target area are also discarded.
  - Wasted space is tracked while combining kernels
    - If wasted space exceeds slack, we prune.

Results

- The ISPD contest objective function combined δ_T and interconnect length. CU.POKer won this contest
- Our approach produced smallest δ_T for all but 2 benchmarks (but has higher interconnect length)
- For all benchmarks:
  - We are within few percentage points from optimal δ_T.
  - We achieve almost complete utilization of all compute sources on CS-1.
- Scoring did not take into account kernel aspect ratios; roughly square layouts of kernels (with more rows) may be better in practice

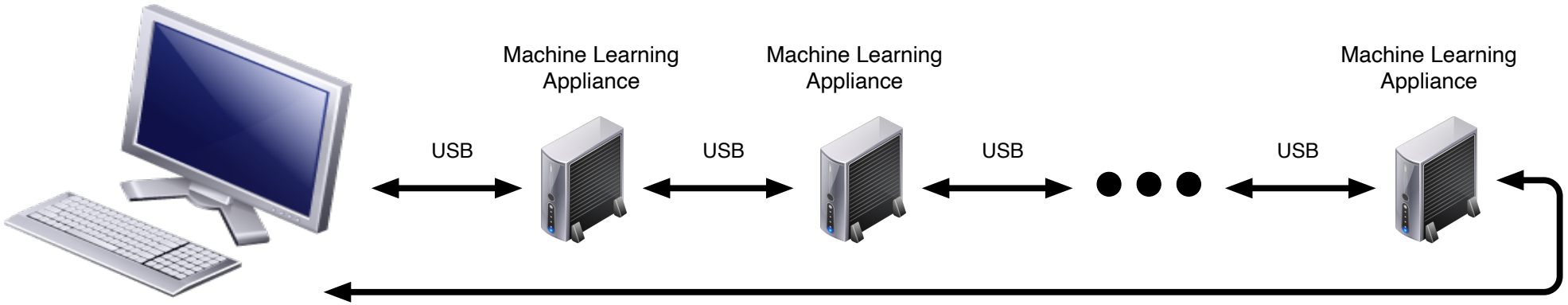
- In the contest, our implementation had a bug; following the contest, the CU.POKer team helped us find and fix it. Many thanks to Prof. Evangeline Young, and her student Mr. Bentian Yang!

Results (2)

			Weighting		δT	CU.POKer			Ours			Ratio vs. CU.POKer				Ratio
BM	$\|k\|$	$\|c\|$	δT	WL	Bound	δT	WL	Score	δT	WL	Score	δT	WL	Score	Util.	δt bnd
A	18	17	1	1	33152	34496	1314	35810	34496	1314	35810	1.000	1.000	1.000	0.972	1.041
B	34	33	1	1	62181	63504	2639.5	66143.5	64512	1913	66425	1.016	0.725	1.004	0.979	1.037
C	102	133	1	1	60270	64512	2408	66920	64512	4108	68620	1.000	1.706	1.025	0.927	1.070
D	54	69	1	1	32256	33713	2722	36435	33712	3588	37300	1.000	1.318	1.024	0.976	1.045
E	17	17	1	10	33152	39312	562	44932	34496	1314	47636	0.877	2.338	1.060	0.972	1.041
F	34	33	1	10	62181	65170	1489.5	80065	64512	1913	83642	0.990	1.284	1.045	0.980	1.037
G	102	133	1	10	60270	64512	2508.5	89597	64512	4108	105592	1.000	1.638	1.179	0.927	1.070
H	54	69	1	10	32256	39520	1104.5	50565	33712	3588	69592	0.853	3.249	1.376	0.976	1.045
I	27	26	1	4	46592	52136	617.5	54606	49280	1504	55296	0.945	2.436	1.013	0.978	1.058
J	81	105	1	4	45752	50274	2472.5	60164	49224	4361	66668	0.979	1.764	1.108	0.951	1.076
K	18	17	1	4	252	432	240	1392	252	423	1944	0.583	1.763	1.397	0.594	1.000
L	54	69	1	4	252	864	279	1980	252	867	3720	0.292	3.108	1.879	0.564	1.000
M	25	28	1	4	2156544	2211840	3610.5	2226282	2322432	3644	2337008	1.050	1.009	1.050	0.956	1.077
N	28	27	1	4	1020	1482	480.5	3404	1035	1490	6995	0.698	3.101	2.055	0.985	1.015
O	27	26	1	40	46592	52136	617.5	76836	49280	1504	109440	0.945	2.436	1.424	0.978	1.058
P	81	105	1	40	45752	57792	1101	101832	49224	4361	223664	0.852	3.961	2.196	0.951	1.076
Q	18	17	1	40	252	882	166	7522	252	423	17172	0.286	2.548	2.283	0.594	1.000
R	54	69	1	40	252	8064	259	18424	252	867	34932	0.031	3.347	1.896	0.564	1.000
S	25	28	1	400	2156544	2396300	1897.5	3155300	2322432	3644	3780032	0.969	1.920	1.198	0.956	1.077
T	28	27	1	40	1020	4102	208.5	12442	1035	1490	60635	0.252	7.146	4.873	0.985	1.015
											Avg:	0.781	2.390	1.554	0.888	1.042

Chaining Together Accelerators

- Because of low Rent parameter, there's much less pressure for complete integration onto a single chip

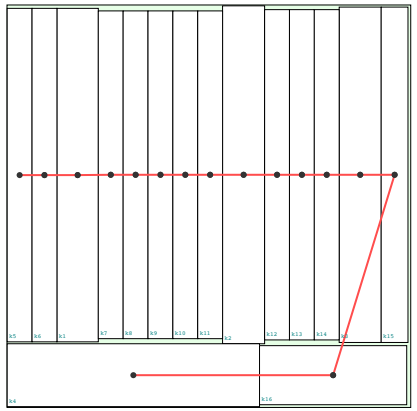


Multiple CS-1s for Further Acceleration

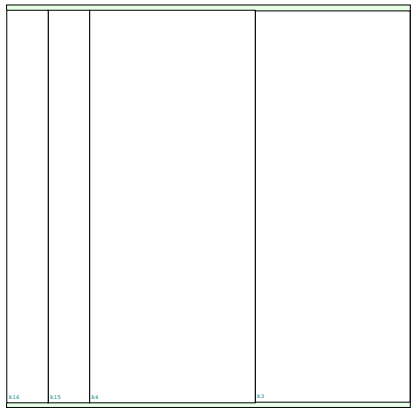
BM	Single CS-1 δT	Multiple CS-1 δT 1	2	3	4	Max δT	Speedup
A	34496	10752	9072	8960	6652	10752	3.21
B	63504	15792	15876	16128	15435	16128	3.94
C	62720	16128	15680	15680	15435	16128	3.89
D	33516	9072	9702	9324	6504	9702	3.45
I	48216	13608	12096	12432	10045	13608	3.54
J	47628	12096	12096	12096	11340	12096	3.94
M	2228224	516096	681984	576576	342133	681984	3.27
						Avg:	3.61

Single vs. Multiple CS-1

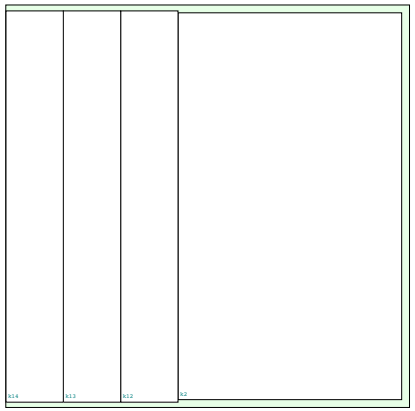
Single WSE implementation: DeltaT=34496



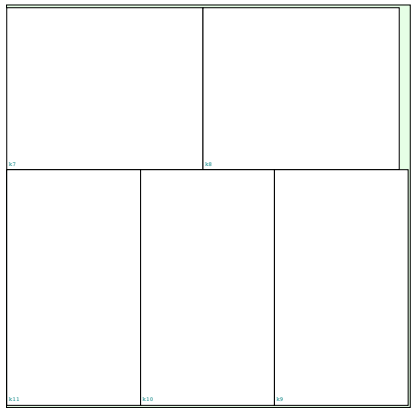
First WSE: DeltaT=10752



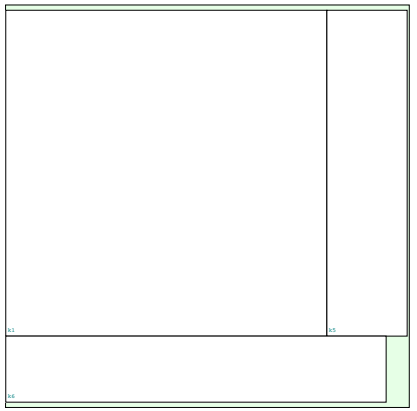
Second WSE: DeltaT=9072



Third WSE: DeltaT=8960



Fourth WSE: DeltaT=6552



Daisy-chaining multiple wafer scale engines together may be practical; tensors flow unidirectionally, as a pipelined flow of information. Communication between WSE units introduce latency, but with buffering, may not impact overall throughput.

Towards Sustainable Deep Learning Acceleration

- We could extend lifetime of accelerators
  - Compared to conventional circuitry that quickly goes out of favor.
- Taking advantage of “clean cuts” in kernel graphs.
  - Chaining multiple processors with interconnects.
  - Map the kernels into the chain.
- Instead of discarding old technology, newer processors could be chained to the old.

Thank you!