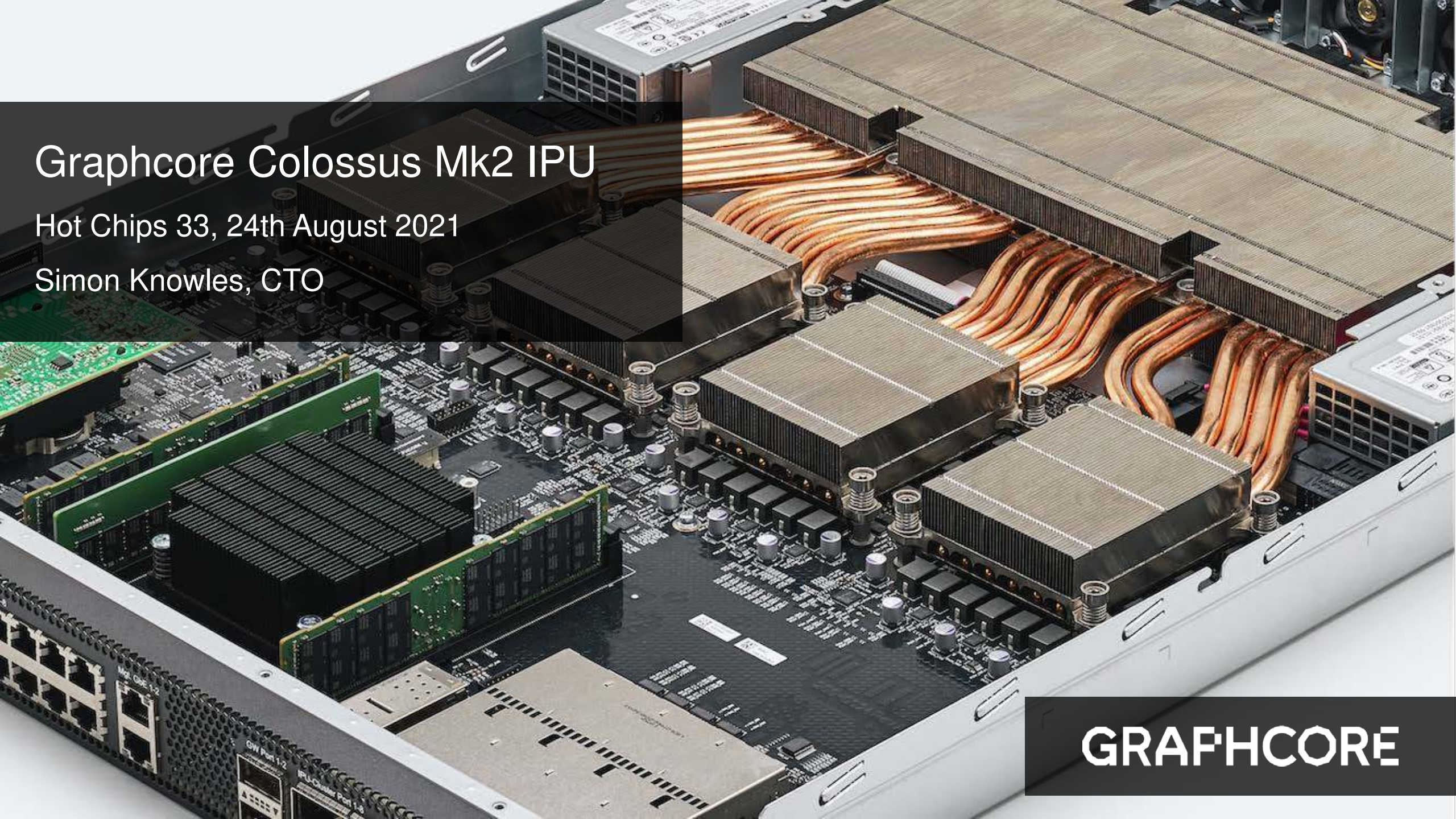
Graphcore Colossus Mk2 IPU

Hot Chips 33, 24th August 2021

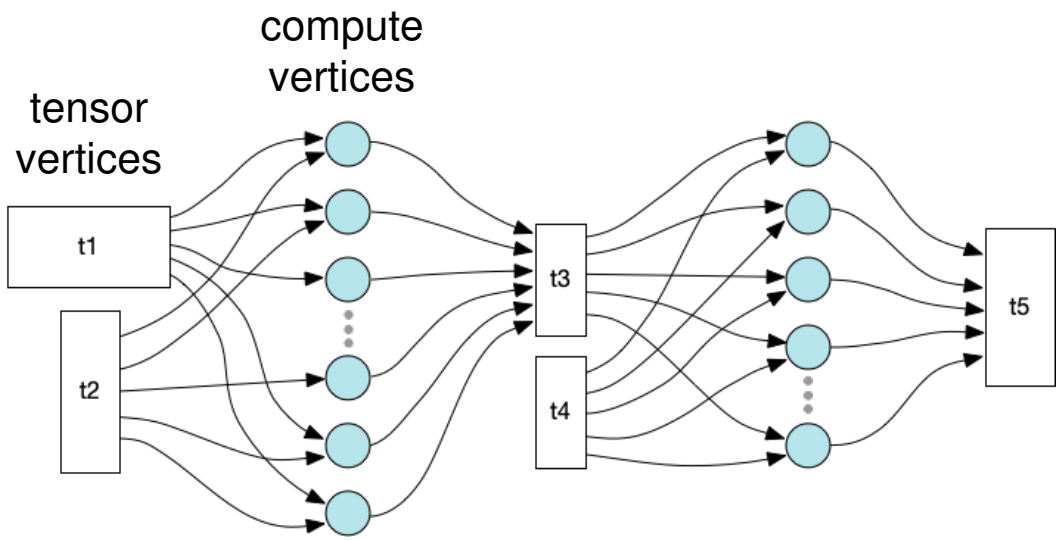
Simon Knowles, CTO



IPU Foundations

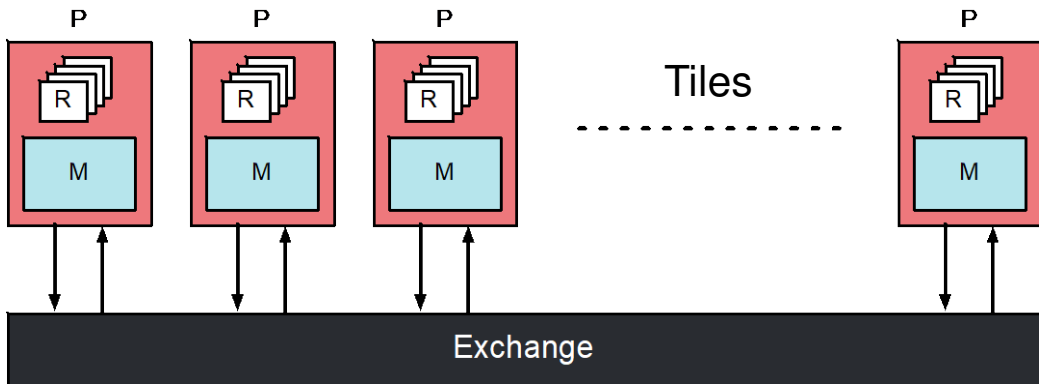
- AI is nascent – facilitating exploration is as vital as executing known algorithms.
- Tera-scale models will be necessary for “super-human” AI.
- Sparse evaluation will be necessary at tera-scale, for $ and Watts.
- Rich natural data is sequences, images, and graphs.
- Post-Dennard, keep memory and logic close.
- Post-Moore, parallel computing over many chips.

IPU Software Abstraction



1) A declared, loopy, bipartite *graph* of compute vertices, tensor vertices, directed edges, and I/O pipes. Persistent vertex state, stateless edges.
2) A library of atomic *codelets,* defining the operation of compute vertices on slices of tensors.
3) A *control program* conditionally executing sets of compute vertices. Set members may execute in parallel.
4) A *host program* terminating IO pipes.

IPU Hardware Abstraction



- Many *tiles*, each containing a multi-threaded processor and local memory.
- Tiles communicate via an all-to-all, stateless *exchange*.
- Any codelet is executable atomically by a single tile thread.
- A tensor vertex may be distributed over many tiles.
- Bulk synchronous alternation between [local compute] and [global communication].

GC2 “Colossus Mk1” IPU [2018 power-on]

23,647,173,309 active transistors in TSMC N16

1216 processor tiles @ 256KiB

Total 125Tflop/s + 304MiB SRAM

62TB/s memory, 7.8TB/s inter-tile, 320GB/s inter-chip

GC200 “Colossus Mk2” IPU [2020 power-on]

59,334,610,787 active transistors in TSMC N7

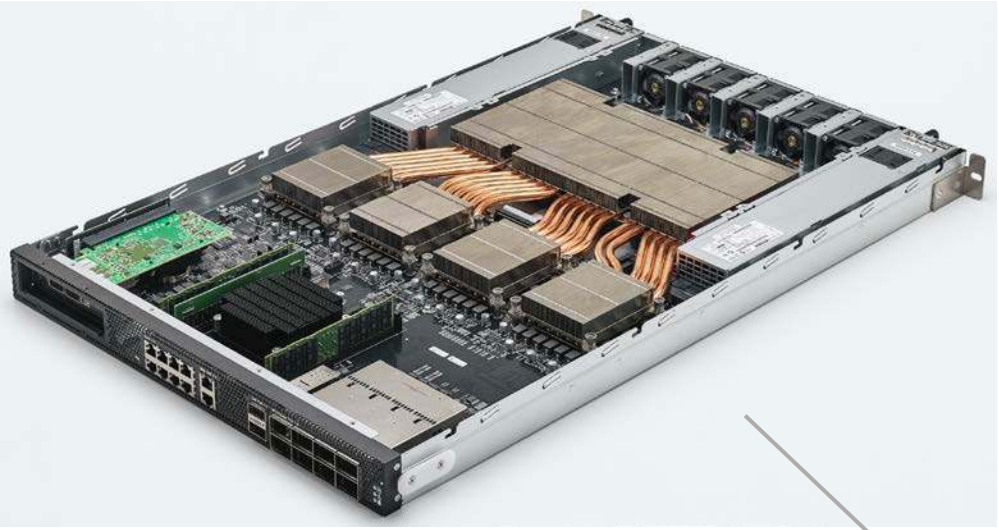
1472 processor tiles @ 624KiB

Total 250Tflop/s + 896MiB SRAM

62TB/s memory, 7.8TB/s inter-tile, 320GB/s inter-chip

Lessons from Colossus Mk1

- We put more features into our “MVP” than we managed to light up with software during its lifetime, eg. sparse tensor arithmetic.
- Not unexpectedly, we spent a lot of time tuning code and workspace to fit 256kB tile memory.
- Efficiently mapping big models across many chips requires computer expertise – most AI programmers need rich automation.
- Whole-graph compilation was initially simplest, but inevitably slow as models grew.
- Bulk synchrony makes it harder to tune out Vdd margin for supply transients, to minimize power consumption. Nevertheless, Mk1 demonstrated good power efficiency.
- PCIe cards severely constrain power density and chip cluster connectivity.
- There’s no one-size-fits-all ratio of host CPUs to AI chips.



M2000 IPU-Machine™ disaggregated AI accelerator

4x Colossus Mk2 IPU ~ 1Pflop/s peak

Local proxy host with max 512GiB[1] DDR

1.2Tb/s inter-chassis breakout.

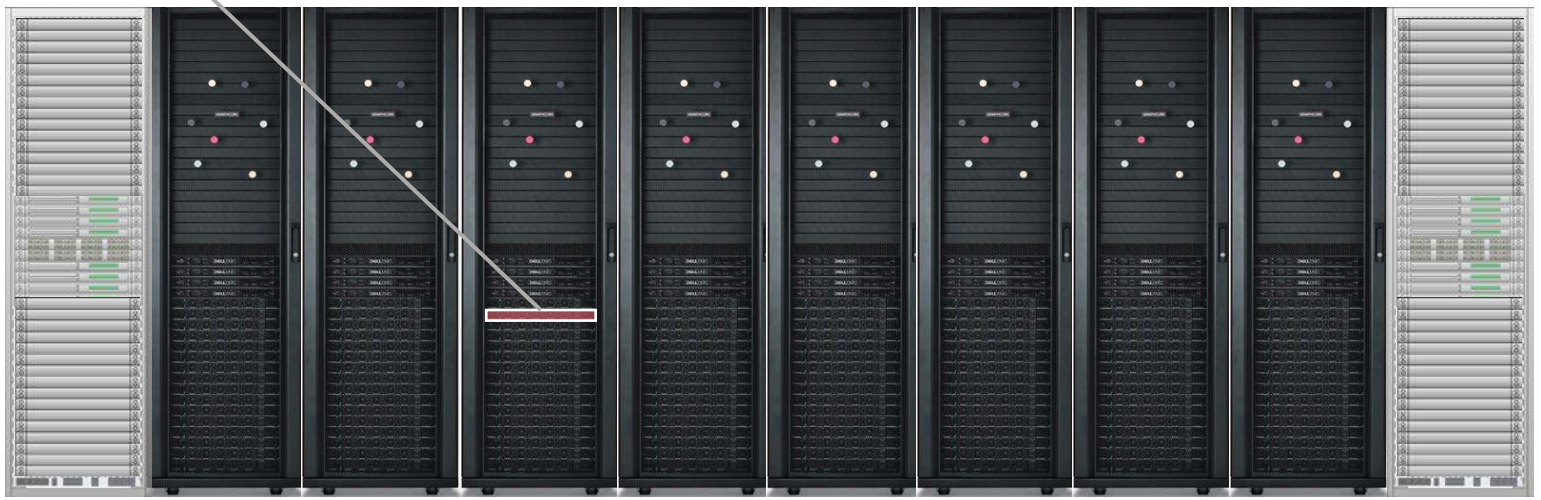
1.5kW TDP, ~1kW typical applications

POPLAR®

PYTORCH

TensorFlow

IPU-POD512™



Structural Headlines

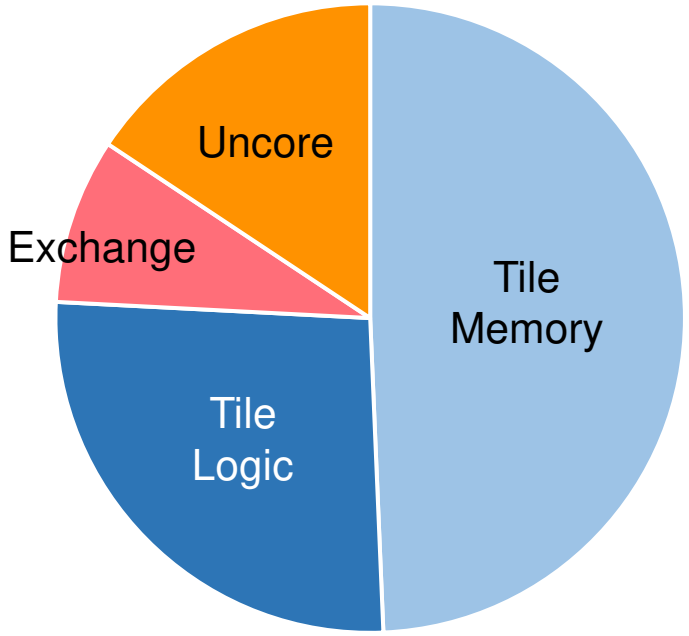
	IPU		GPU		TPU	
Chip	Mk1	Mk2	V100	A100	v2	v3
Cores[2]	1216	1472	320	432	2	2
SIMD width	64b	64b	1024b	1024b	4096b	4096b
Burst fp16 Tflop/s	125	250	125	312	46	123
Burst fp32 Tflop/s[3]	31	62	16	19	3	4
On-die MiB[4]	304	897	20	40	32	32
Off-die GiB	Host	Host	16, 32	40, 80	16	32
Inter-chip duplex Tb/s	1.3	1.3	1.2	2.4	2.0	2.6

IPU is a fine-grained parallel processor with huge distributed SRAM on die.

Colossus Mk2 IPU

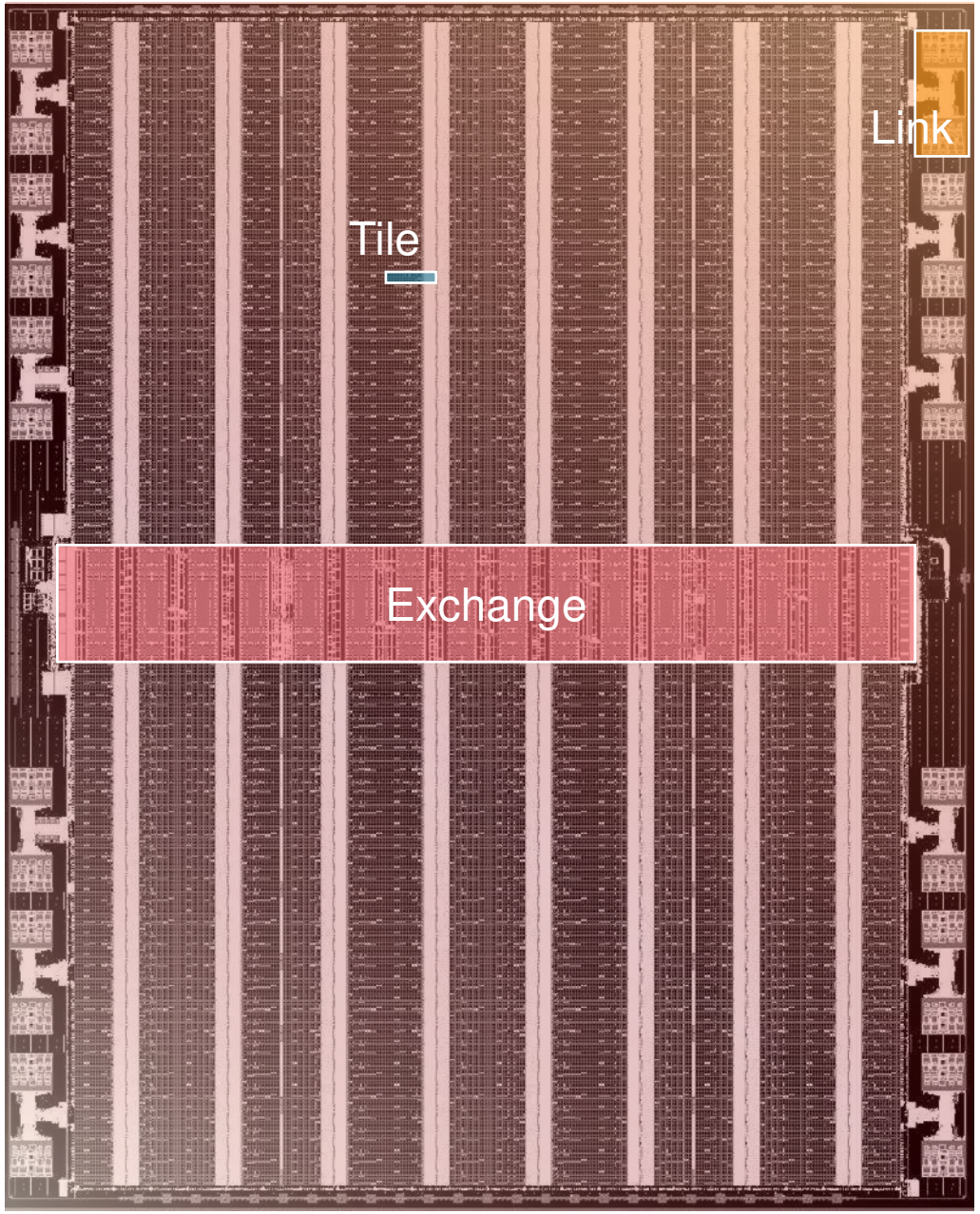
59,334,610,787 active transistors

7nm 823mm^2…



1.325GHz global mesochronous clock

23/24 tile redundancy



Tile Processor

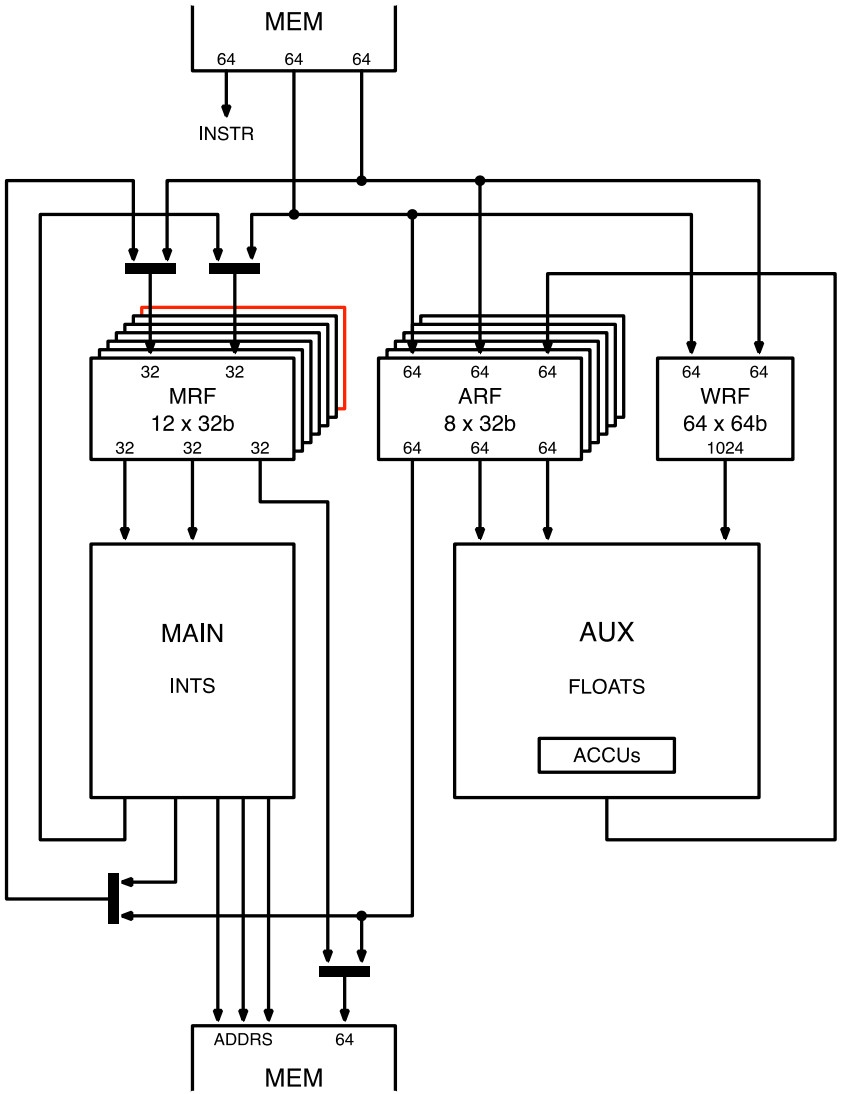
- 32b instructions, single or dual issue.
- Two execution paths, barrel threaded.

MAIN path:

- Control flow, integer/address arithmetic.
- Multi-load/store, to/from either path.

AUX path:

- Floating-point arithmetic co-issued with MAIN.
- Vector and matrix operators with in-line state.
- Transcendentals: e^x, 2^x, ln, $\log_2$, logistic, tanh.
- Random number generation.



N+1 barrel threading

7 program contexts, 6 round-robin pipeline slots.

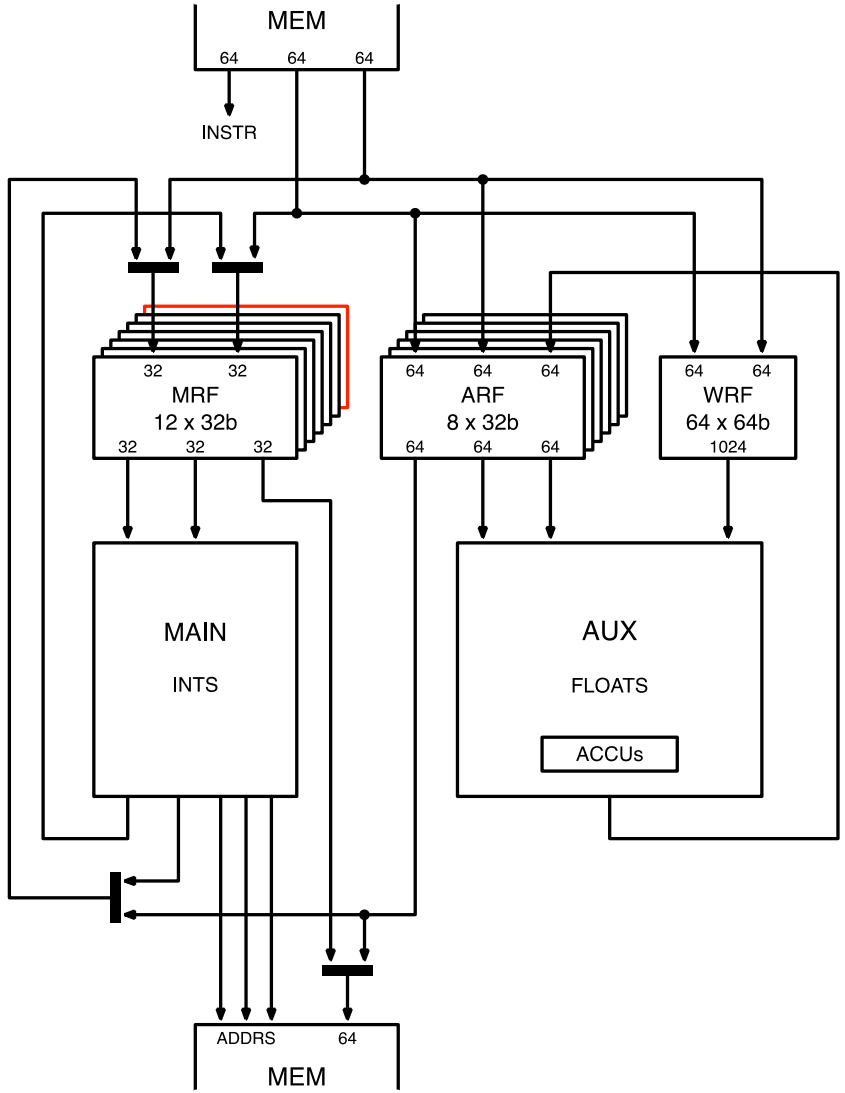
The Supervisor program:

- A fragment of the control program, orchestrating the updating of vertices.
- Executes in all slots not yielded to Workers; sees the pipeline.
- Dispatches Workers by **`RUN`** instruction, yielding that slot.

A Worker program is a codelet updating a vertex:

- Executes in 1 slot at 1/6 of clock; does not see the pipeline.
- Returns its slot to the Supervisor by **`EXIT`** instruction.

Hiding the pipeline from Workers makes vertex execution easy for a compiler to predict, hence to load balance.

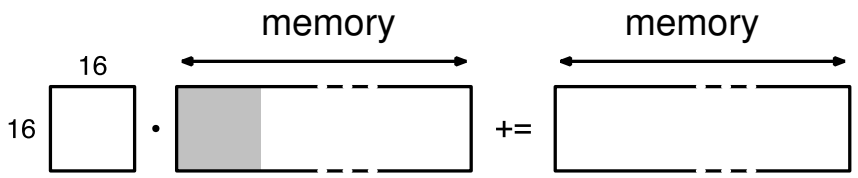


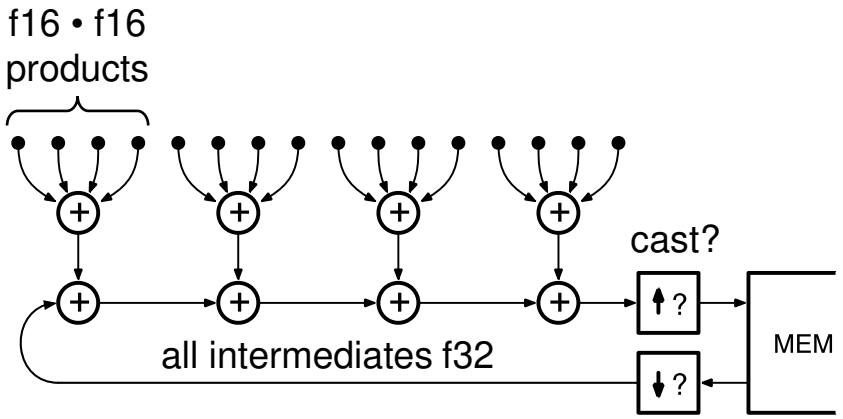
Sparse Load / Store

- 896MiB on-die SRAM at 47TB/s (data-side) provides unprecedented access to arbitrarily-structured data which fits on chip.
- ld/st instructions support sparse gather in parallel with arithmetic at full speed, via compact pointer lists:
  - 16b absolute offsets to a base,
  - 4b cumulative delta offsets to a base.

IEEE f16 and f32 MatMuls and Direct Convolutions

Channels (Kernel)		Multiply	Accumulate		Burst
`AMP` (1x1)	`SLIC` (4x1)		Datapath	Memory	Tflop/s
16x16	4x4	f16	f32	f16	250
16x8	4x2		f32	f32	125
8x8	4x2	f32	f32	f32	62

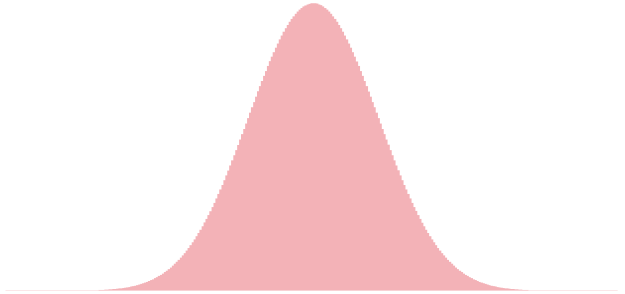




Random numbers and stochastic rounding

Each tile can generate 128 random bits per cycle:

- Private context per worker thread.
- Enhanced xoroshiro128+ PRNG.
- 6th-order Irwin Hall Gaussian shaper.

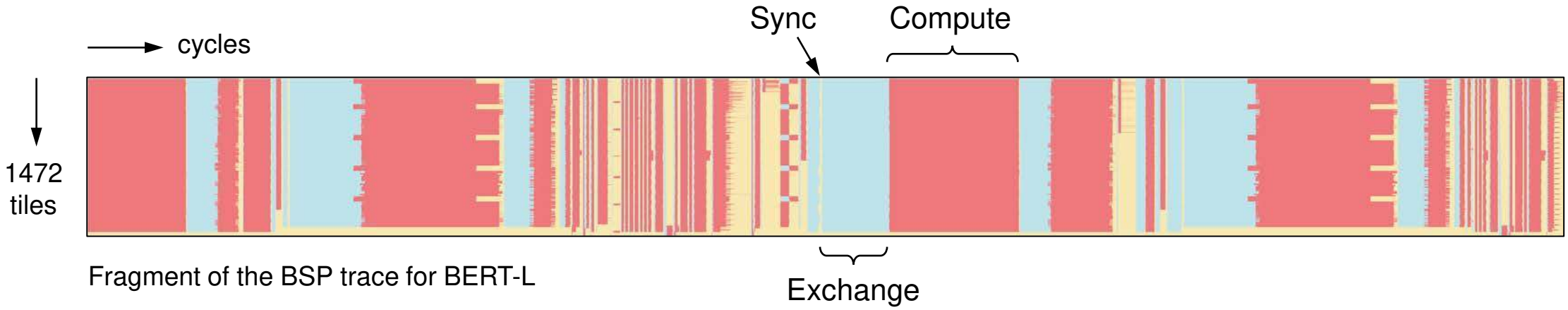


Instructions:

- Generate a vector of random numbers, uniform or Gaussian.
- Randomly puncture a vector with specified probability.
- Stochastically round down-casts at full speed – vital for fast and easy training of f16 models.

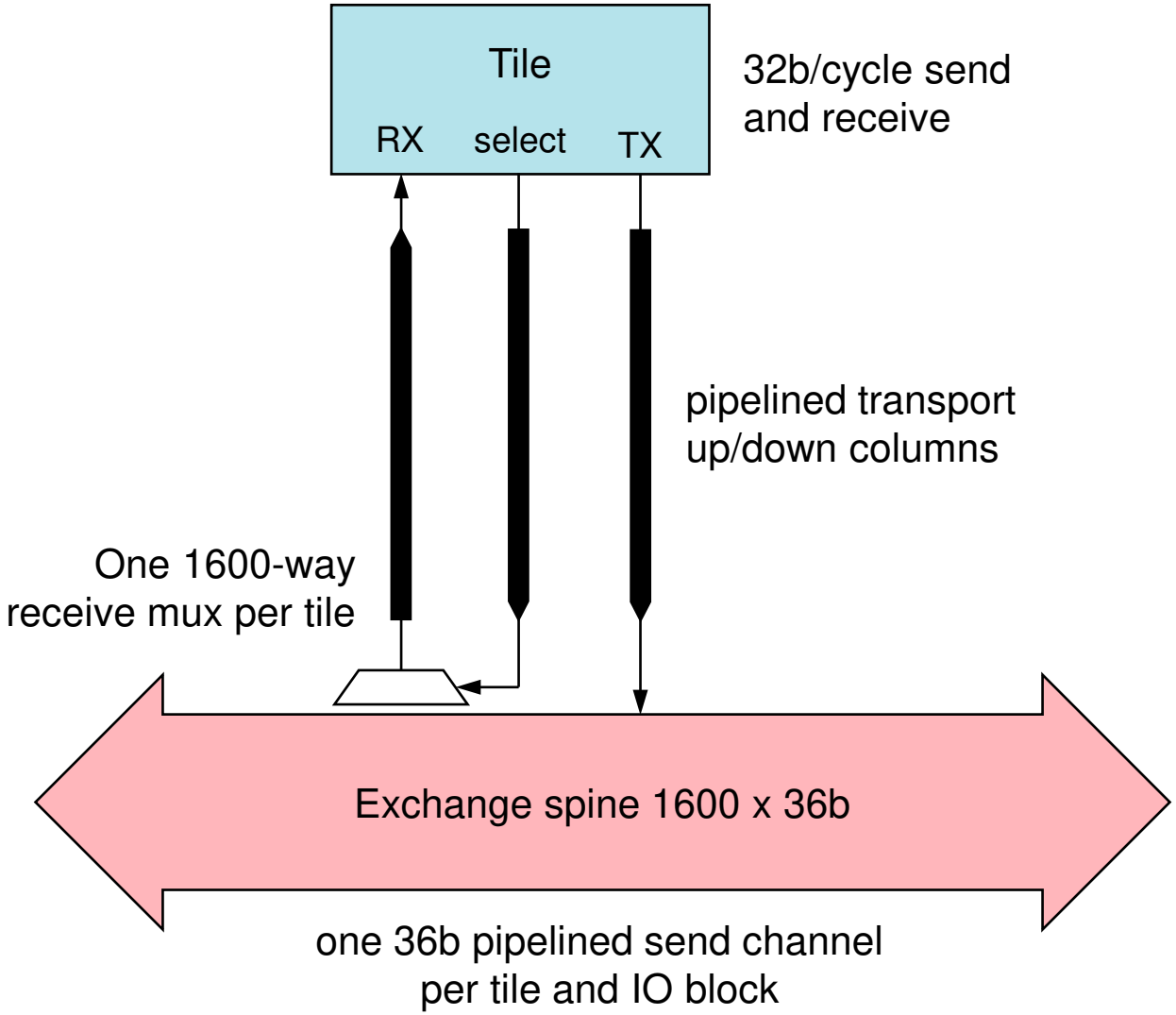
Global Program Order

- Tile processors execute asynchronously until they need to exchange data.
- Bulk Synchronous Parallel (BSP): repeat { Sync; Exchange; Compute }
- Each tile executes a list of atomic codelets in one compute phase.
- Hardware global synchronization in ~150 cycles on chip, 15ns/hop between chips.



Fragment of the BSP trace for BERT-L

Exchange Mechanics

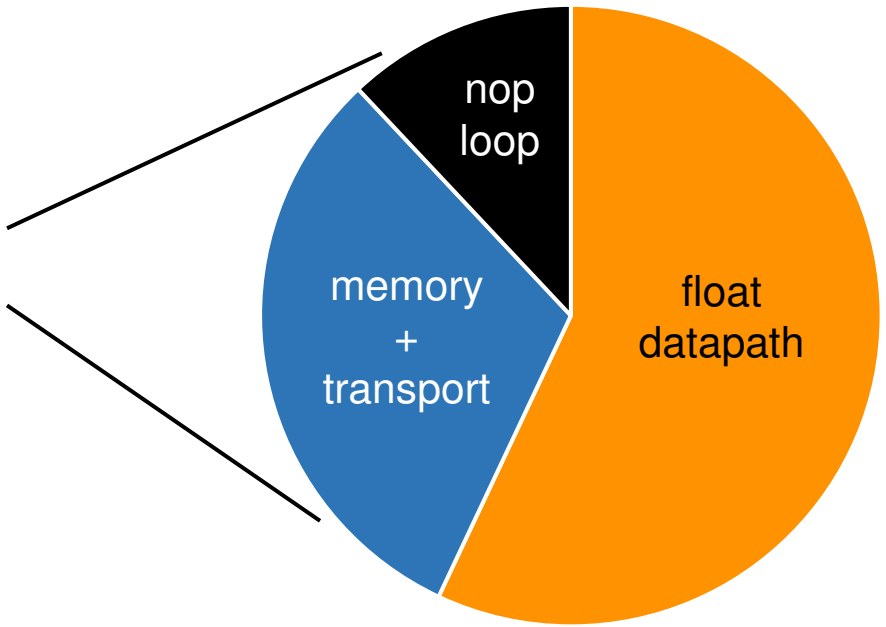


- The POPLAR® compiler schedules transmit, receive and select at precise cycles from sync, knowing all pipeline delays.
- Any pattern of data movement, changeable at every clock cycle.
- Addressing is by time and select state – there are no queues, arbiters, or packet overheads, just data moving at full bandwidth and minimum energy.
- Physically mesochronous; 3 cycles global synchrony drift across chip.

Chip Power

Convolution dynamic power measured at the die with virus data:
(real application data is typically 1/3~1/2 less energetic)

Multiply	Accumulate		pJ/flop
	Datapath	Memory	
f16	f32	f16	1.3
	f32	f32	1.75
f32	f32	f32	3.3



- Distributed SRAM keeps most on-die transport to <1mm.
- Large SRAM collapses the required DRAM bandwidth, and moves DRAM power out of the logic die thermal envelope.

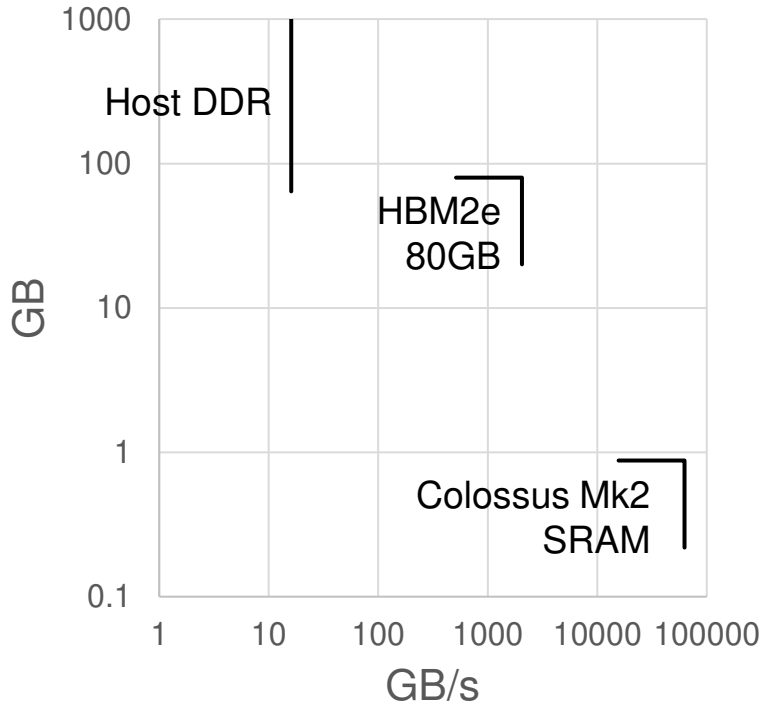
System Power

	IPU	GPU	TPU
Chip	Colossus Mk2	A100	TPUv3
Chip TDP Watts	300	400, 500	450
System w/dual CPU host	Pod16	DGX	Pod16
Chips in system	16	8	16
System TDP Watts	7000	6500	9300
System Watts/chip	437	812	581
System burst fp16 Tflop/s	4000	2496	1968
Nominal Tflop/Watt	0.57	0.38	0.21

Applications typically sustain max ~50% of burst Tflop/s on all platforms. Vendors choose TDP to envelope such applications at full speed; a power virus will slow the clock.

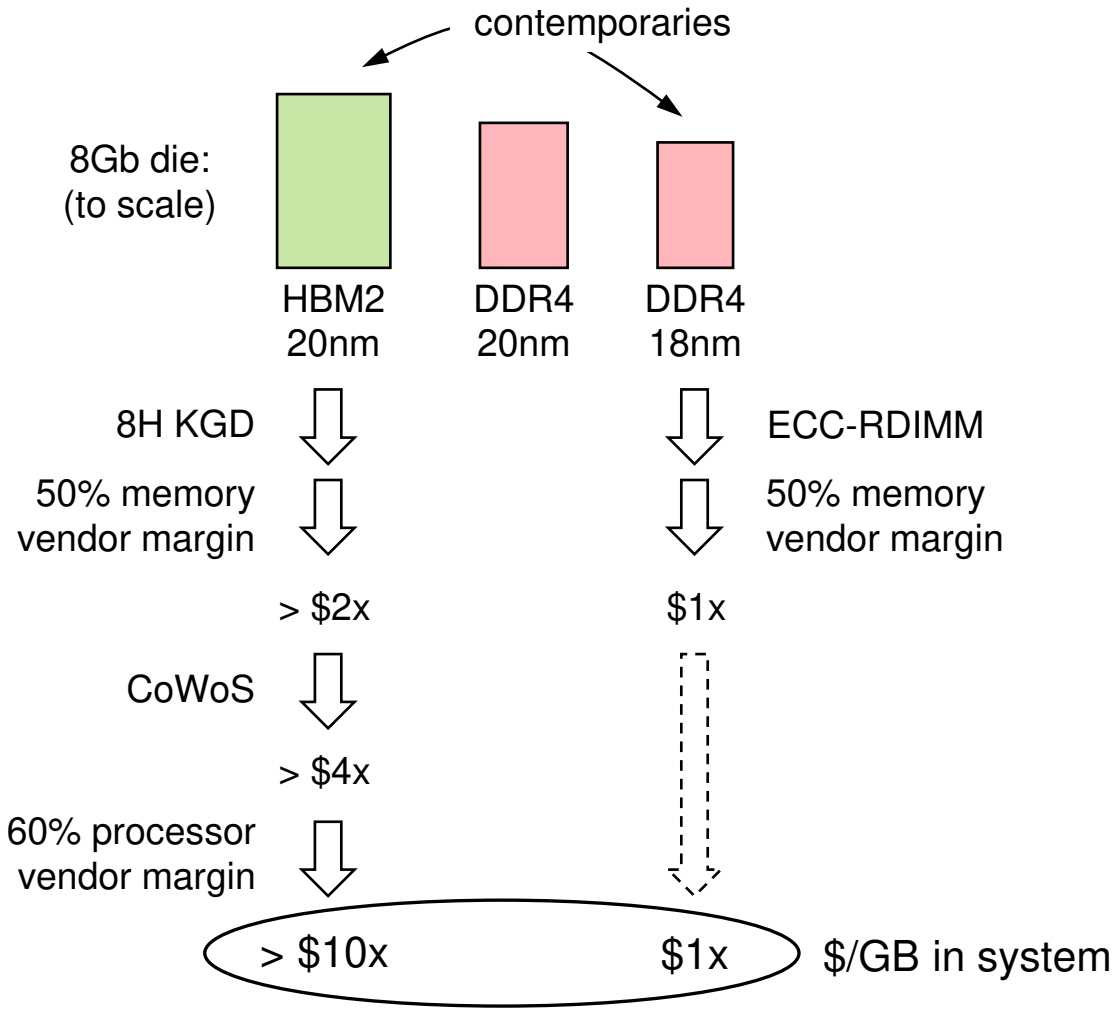
1.5x net efficiency advantage of IPU over GPU implies ~3x transport energy advantage.

Why No HBM?



- Memory capacity determines what an AI can do; bandwidth just limits how fast.
- GPU and TPU try to solve for bandwidth and capacity simultaneously, using HBM.
- HBM is very expensive, capacity-limited, and adds 100W+ to the processor thermal envelope.
- IPU solves for bandwidth with SRAM, and for capacity with DDR.

DRAM Economics

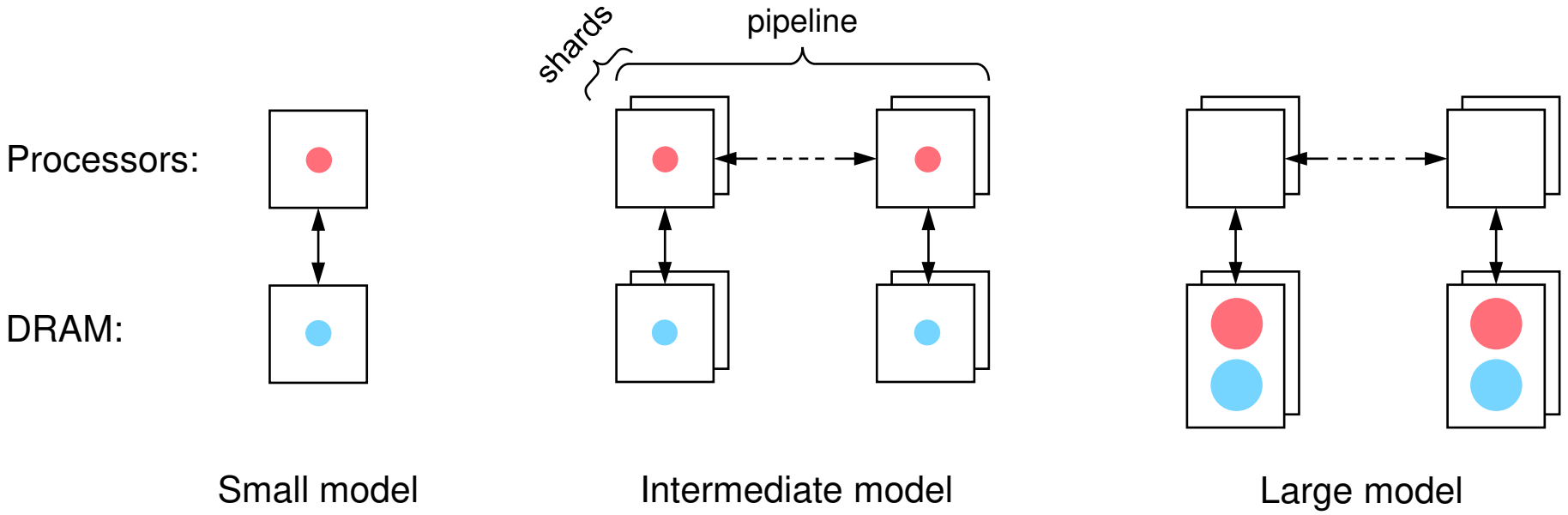


- 40GB HBM ~triples the cost of a packaged reticle-sized processor.
- DDR-based systems like IPU can spend the saved $ on more processors.

Total silicon required for an 8cm^2 processor with 32GB DRAM:

HBM2:	53cm^2
DDR4:	23cm^2

Placing Model State



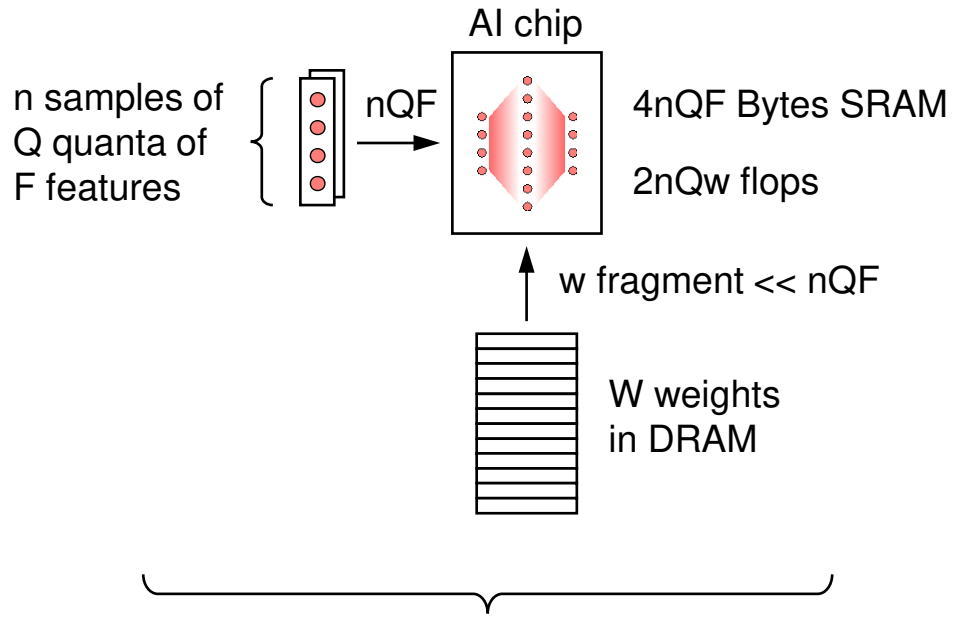
- Model state
- Optimizer state

Off-chip DDR bandwidth suffices for:

- Distributed optimizer states at all scales.
- Streaming weight states for large models.

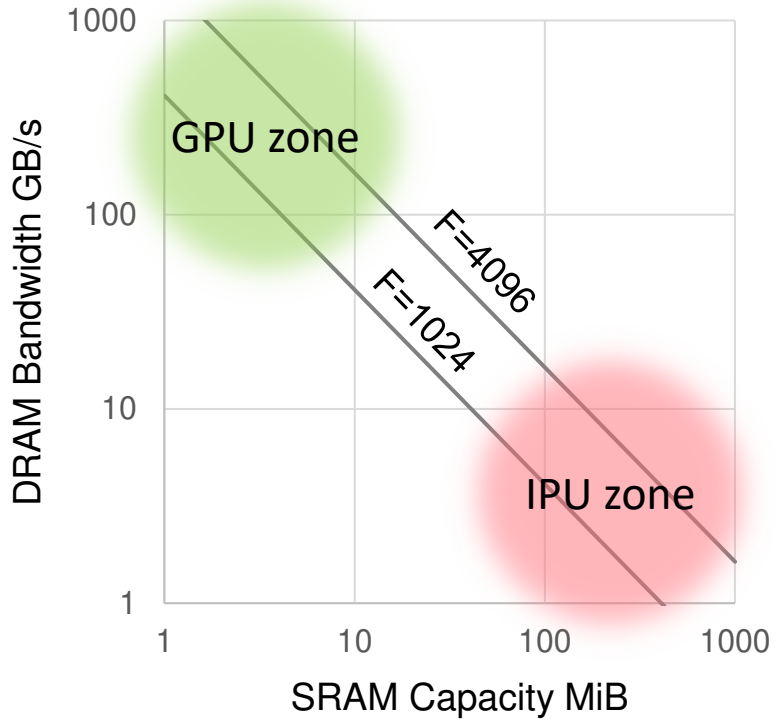
Sufficient On-Die SRAM Collapses the Required DRAM Bandwidth

Crude model of inference with model streamed from DRAM:
(all values 2 Bytes)



$$\text{DRAM bandwidth (Bytes/s)} = 4F \cdot \frac{\text{Compute rate (flop/s)}}{\text{SRAM Capacity (Bytes)}}$$

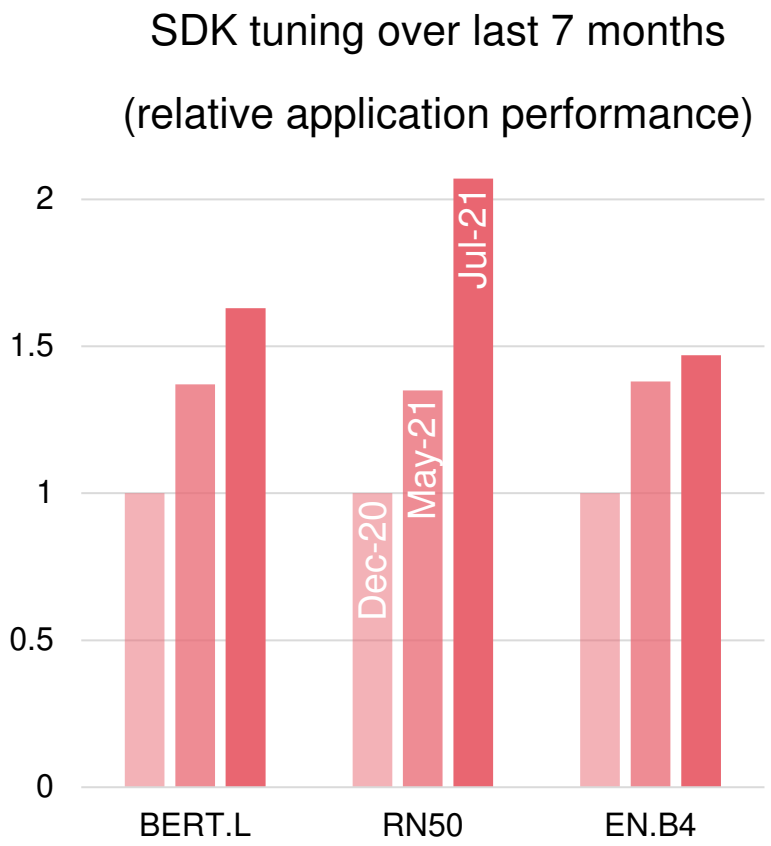
eg. inference at 100Tflop/s…



Hardware helping Software

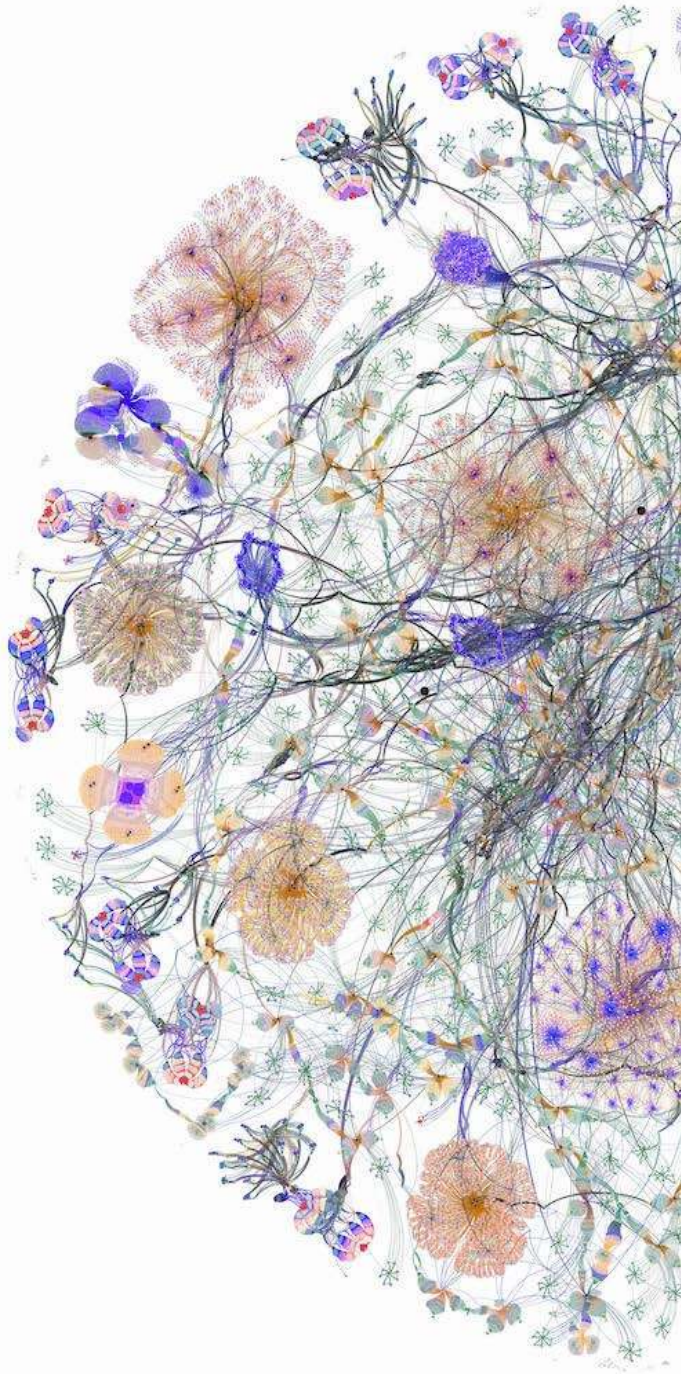
Simple mechanisms allow rapid software evolution:

- Native graph abstraction.
- Codelet-level parallelism.
- Pipeline-oblivious threads.
- BSP eliminates concurrency hazards.
- Stateless all-to-all Exchange.
- Cacheless, uniform, near/far memory.



Key Take-Aways

- Colossus is Graphcore's realization of a new architecture for AI processors, IPUs.
- IPUs minimize the energy of data transport, allowing more processor silicon to be deployed within a power budget.
- IPUs minimize memory cost for AI models, allowing more processor silicon to be deployed within a cost budget.
- IPU's fine-grained parallelism minimizes assumptions about the nature of parallelism in future AI models and data.



Notes:

1) M2000 supports maximum 448GiB available to the 4 IPUs; the balance of DDR is private to the proxy host.
2) We use “core” in the conventional manner, meaning a processor able to run its own program independently of other cores except for communication dependencies, rather than the Nvidia marketing count of “cores” which is all the SIMD lanes of the conventional cores.
3) We list the claimed peak performance using IEEE32 arithmetic, not the Nvidia A100 reduced-precision “tf32” mode which uses only 19 of the 32 bits of its operands.
4) For on-chip memory in a cache hierarchy we count the level with the largest memory, since the other levels will only contain copies of that data. For V100 this maximum is the registers, for A100 it is the L2 caches.

Information presented is believed to be accurate at the time of presentation; however, subsequent events may impact their accuracy. Graphcore undertakes no responsibility to update any information. AI is a rapidly evolving global technology and as such, all information presented herein is subject to change without notice.