Google Tensor Processing Unit

- Google getting into hardware design. Three spins of TPU already!
- TPU is a specialized chip for AI → matrix-matrix/tensor operations
- **Key Idea**: Systolic design
  - Lots of simple parallel computational units
  - Data movement strictly localized to immediate neighbours
- Tightly coupled with TensorFlow (programming APIs)
- $6.50 USD per TPU per hour (as of today!)

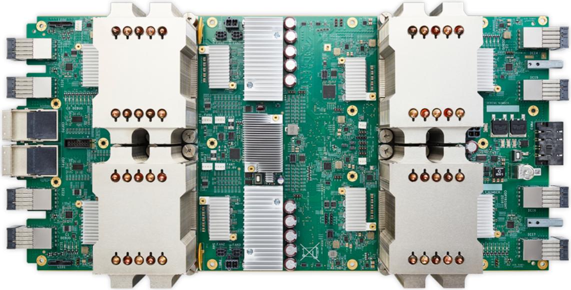


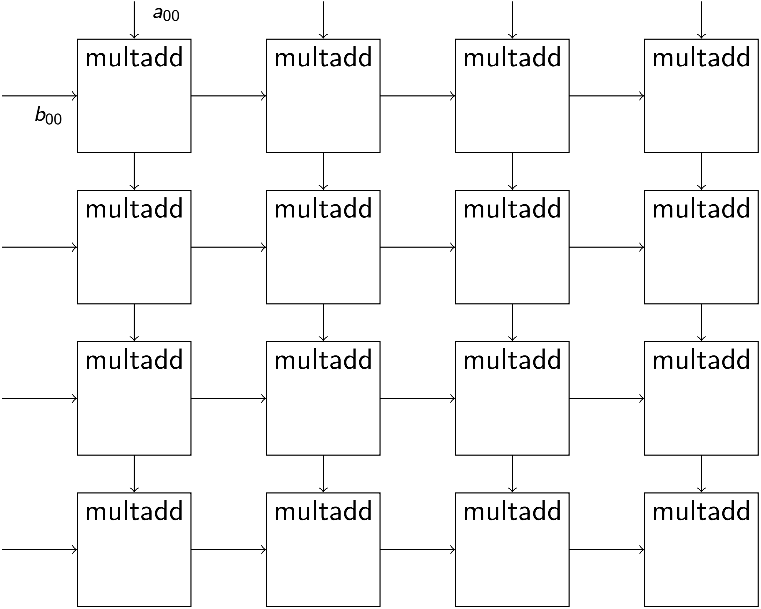
Figure: TPU v3 device → liquid cooled!

https://cloud.google.com/tpu/

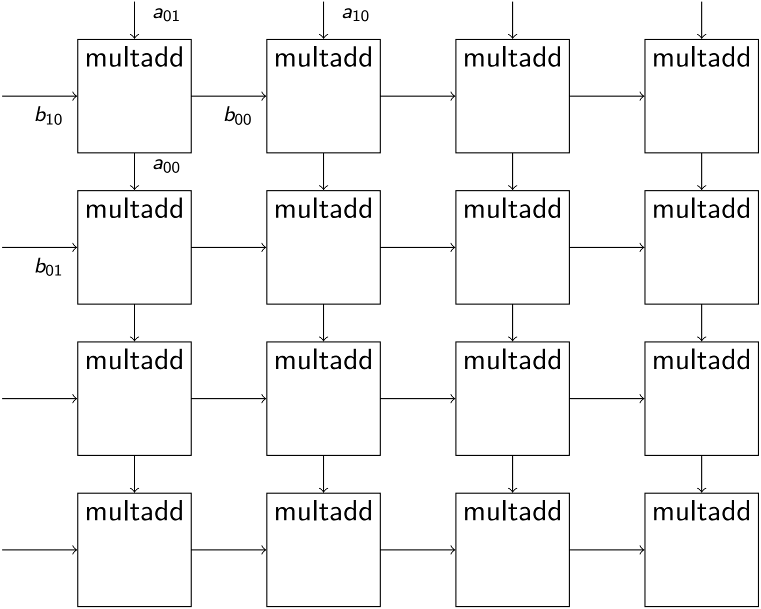
TPU History

- First generation
  - 64K 8-bit multiply-add operations @700 MHz → 44 Tops/s
- Second generation
  - 32K 32-bit (and single-precision float) multiply-add operations @700 MHz → 45 TFLOPS (float)
- Third generation
  - Not a lot of details → 2× better than TPU v2

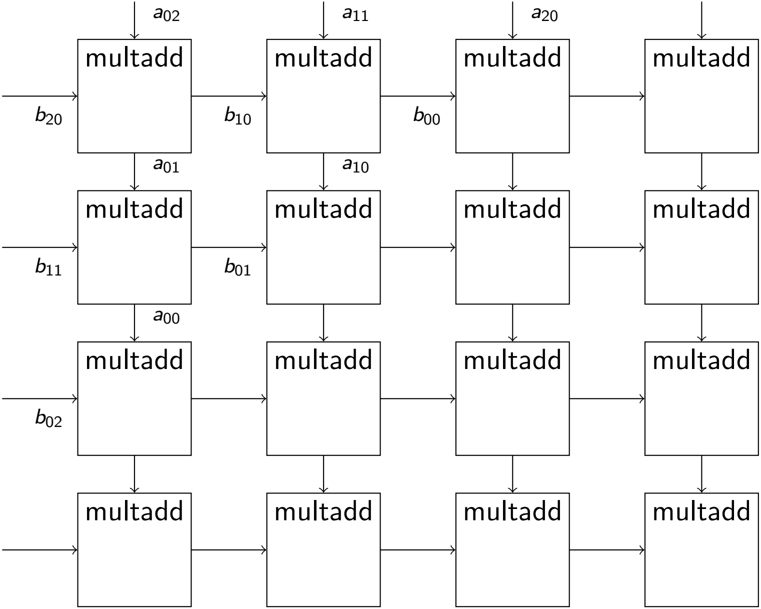
Systolic Array (Step-by-step Matrix Multiply)



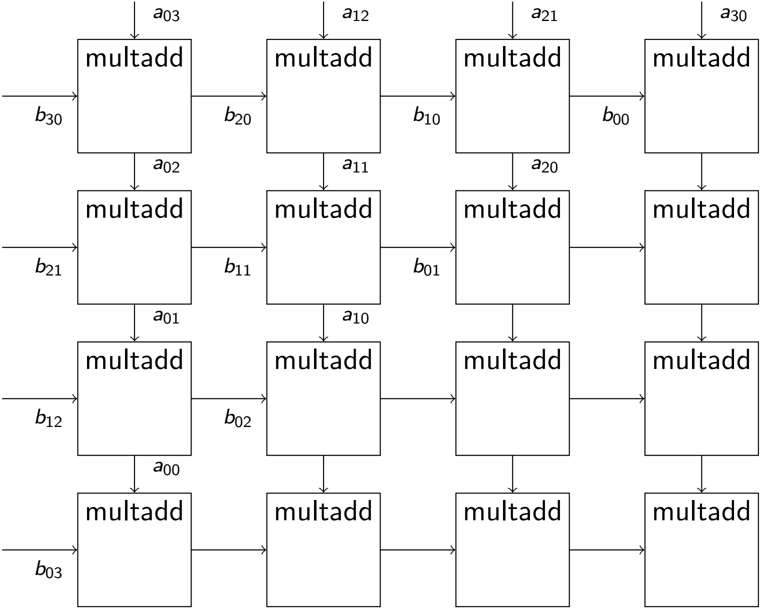
Systolic Array (Step-by-step Matrix Multiply)



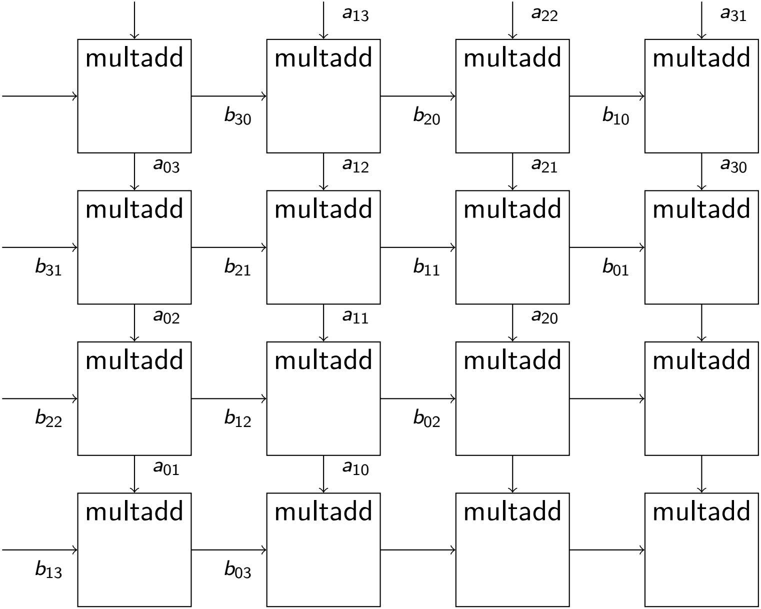
Systolic Array (Step-by-step Matrix Multiply)



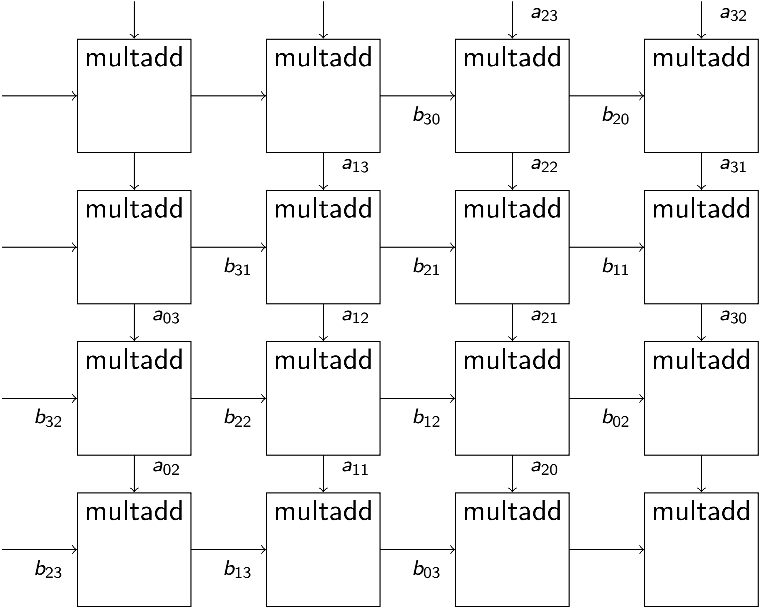
Systolic Array (Step-by-step Matrix Multiply)



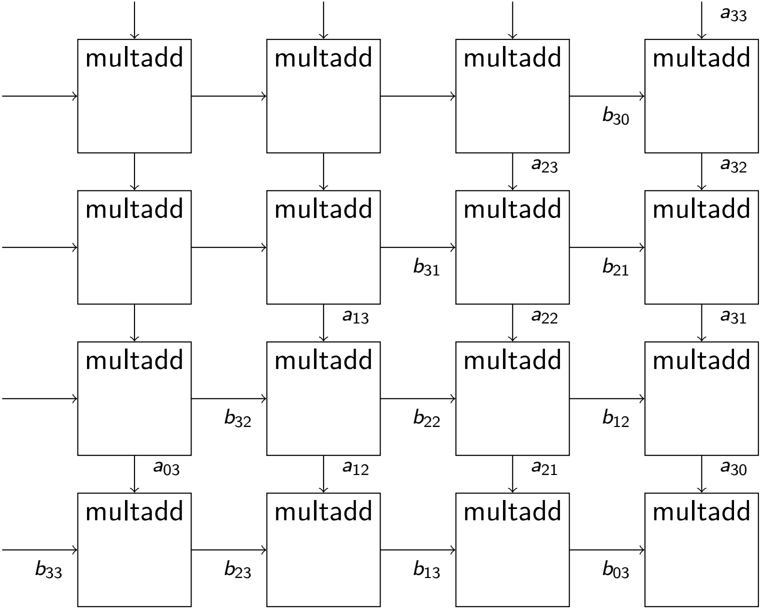
Systolic Array (Step-by-step Matrix Multiply)



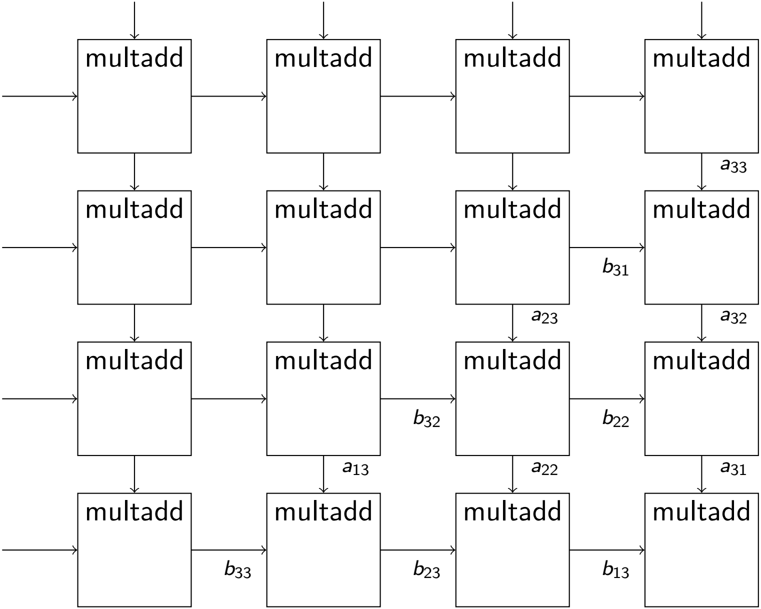
Systolic Array (Step-by-step Matrix Multiply)



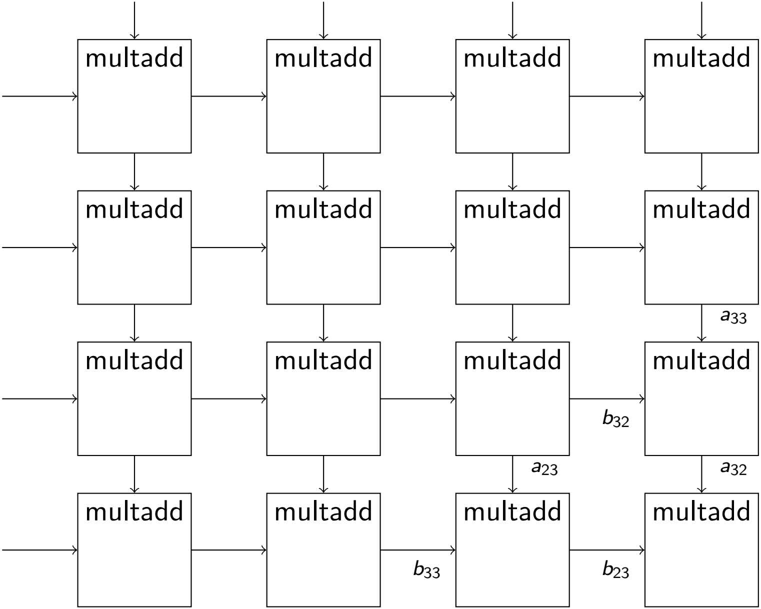
Systolic Array (Step-by-step Matrix Multiply)



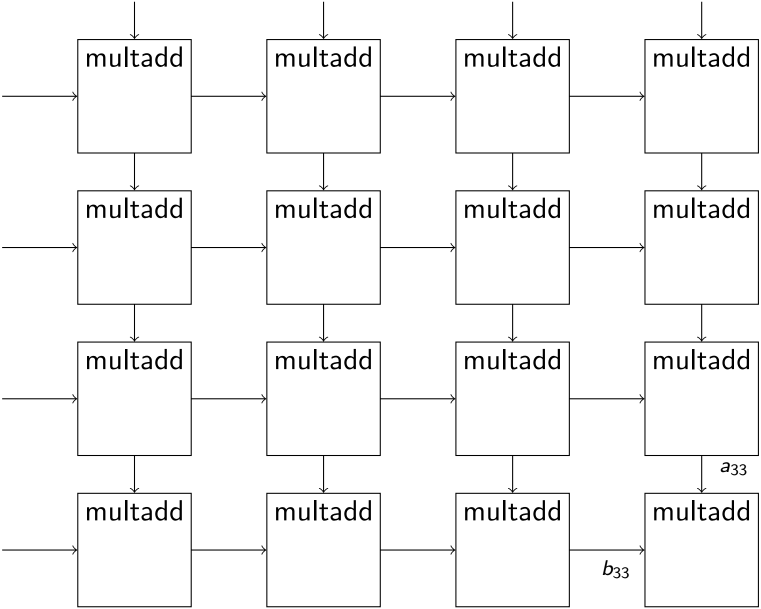
Systolic Array (Step-by-step Matrix Multiply)



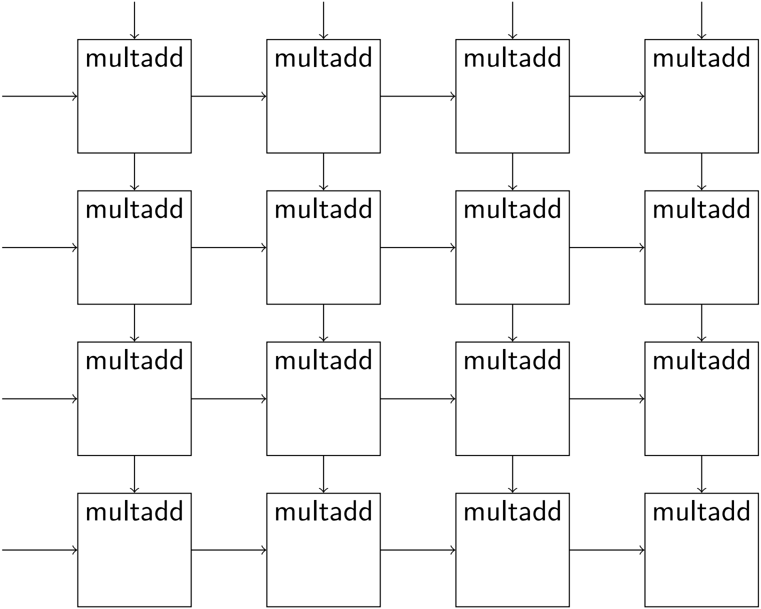
Systolic Array (Step-by-step Matrix Multiply)



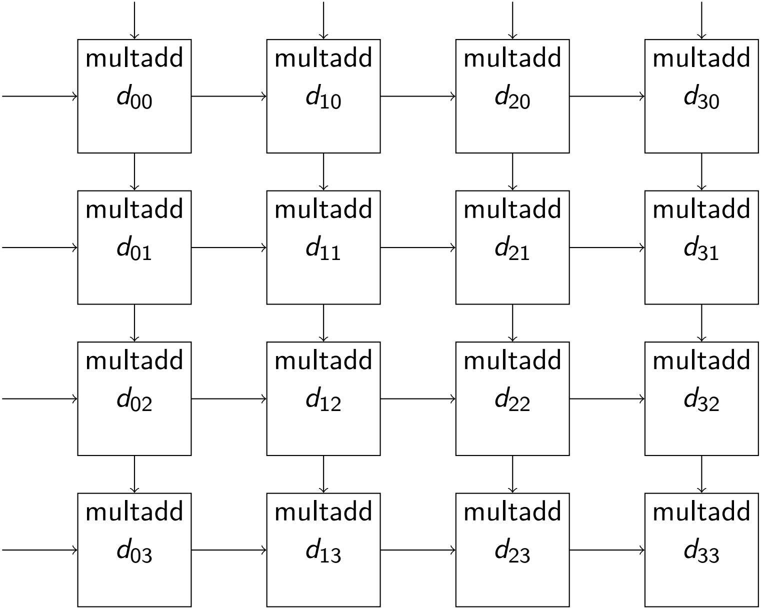
Systolic Array (Step-by-step Matrix Multiply)



Systolic Array (Step-by-step Matrix Multiply)



Systolic Array (Step-by-step Matrix Multiply)



multadd d_{00}

multadd d_{10}

multadd d_{20}

multadd d_{30}

multadd d_{01}

multadd d_{11}

multadd d_{21}

multadd d_{31}

multadd d_{02}

multadd d_{12}

multadd d_{22}

multadd d_{32}

multadd d_{03}

multadd d_{13}

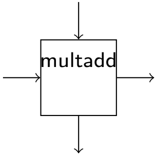
multadd d_{23}

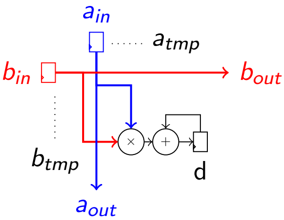
multadd d_{33}

Matrix-Matrix Multiplication on Systolic Array

- Send A matrix along column, and B matrix along rows of a 2D array
- Each step along the way, the compute block performs a small multiply-add operation
- Data movement is co-ordinated to ensure correct data arrives at correct compute block in the correct time!
- All data movement is local $\rightarrow$ hardware design is super easy!
- In the end, final product D is available in every compute block $\rightarrow$ result must be read out (not shown)

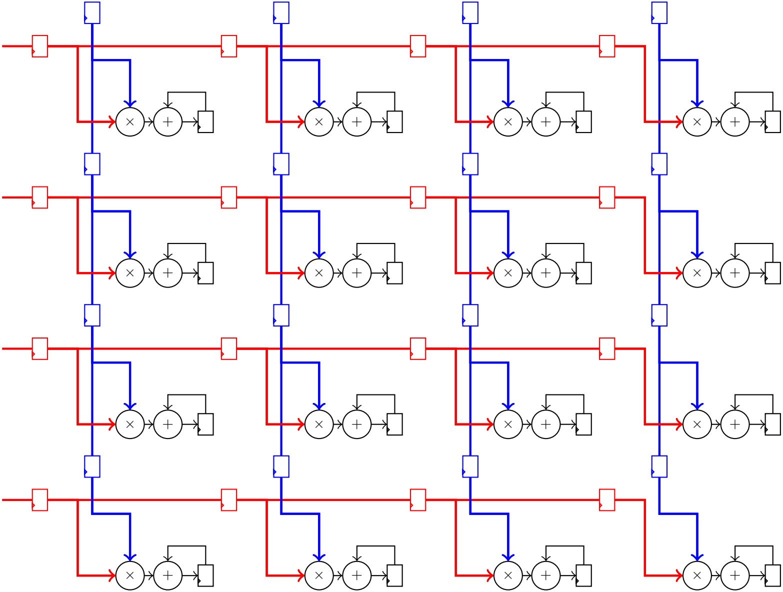
The `multadd` Building Block





- The `multadd` block is a simple 8b×8b multiplier followed by a 32b accumulator
- All communication in the chip is strictly local → nearest neighbour → no long wires!
  - Matrix elements *streamed* into systolic core from top+left links
  - Data forwarded to links below+right for further use by neighouring elements
- The building block in the TPUv1 was just 8b multiplication of neuron weights, and input → later revisions added floating-point and custom formats
- Design is so easy, even Google engineers can build this :)

Systolic Array Design (4×4 design)



Caveat: Above pic is simple, actual TPUv1 is 256×256 blocks

Wrapup

- Top-down design of hardware design → NVIDIA Tensor Cores, Google TPU Systolic array case studies
- It is possible to approximate (to the first order) even complex hardware designs
- Hardware design complexity can be tamed with replication via `generate` statements