ARQUITECTURAS ESPECÍFICAS DE DOMINIO

TPU de Google

Diego Dorado Galán
Esther Camacho Caro

TABLA DE CONTENIDOS

01 INTRODUCCIÓN

02 UNIDADES DE PROCESAMIENTO TENSORIAL

TPUs

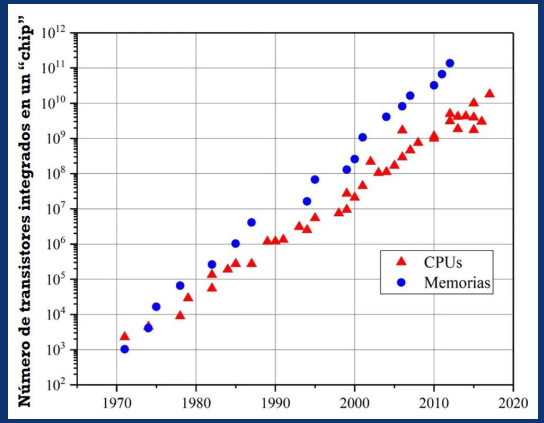
03 TPU DE GOOGLE

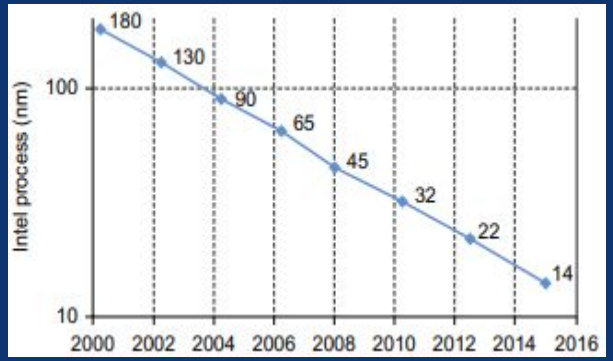
TABLA DE CONTENIDOS

04 GUÍA DE DISEÑO DE DSA

05 BIBLIOGRAFÍA

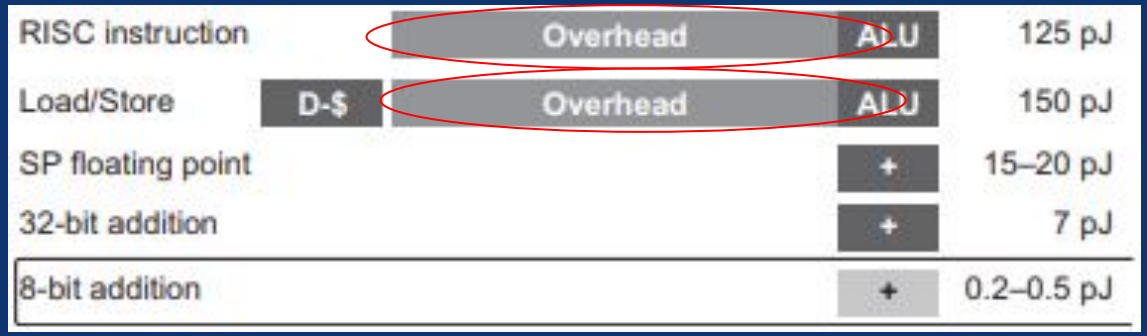
1. INTRODUCCIÓN





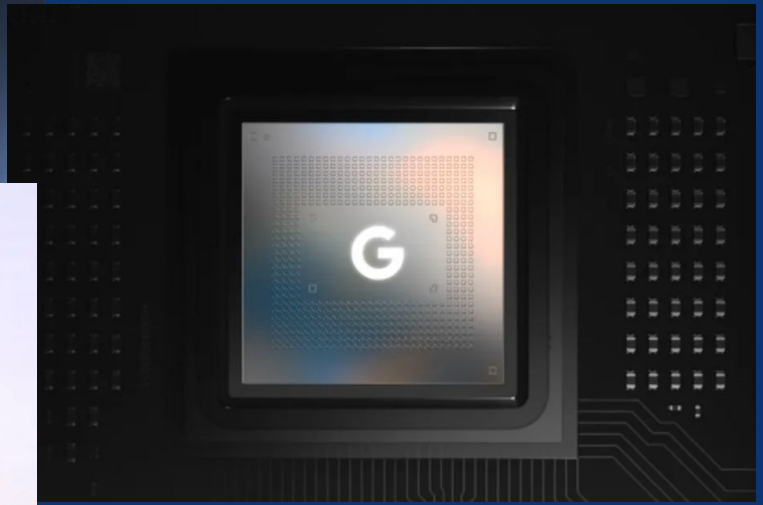
Ley de Moore:

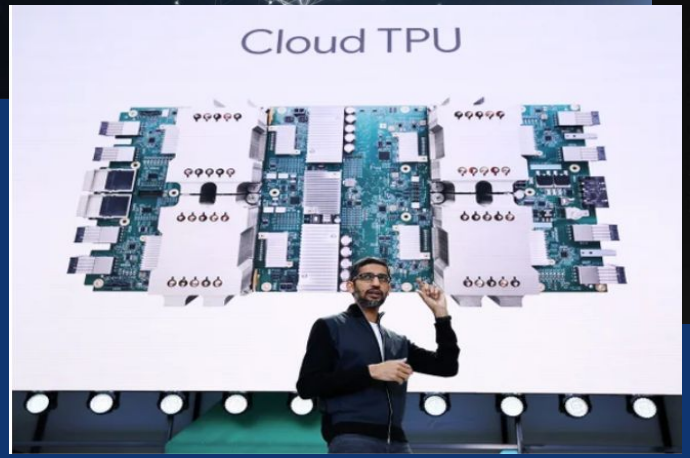
- Niveles de caché
- 512-bit SIMD floating-point units
- Branch prediction
- Out-of-order execution
- Multithreading
- Multiprocessing
- Speculative prefetching



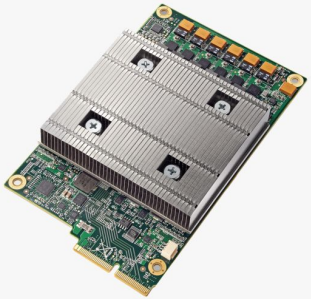
Coste energético en pico julios para un procesador de 90 nm para obtener instrucciones o acceder a caché de datos en comparación con el costo de las operaciones aritméticas

SOLUCION: DSA





2. UNIDADES DE PROCESAMIENTO TENSORIAL



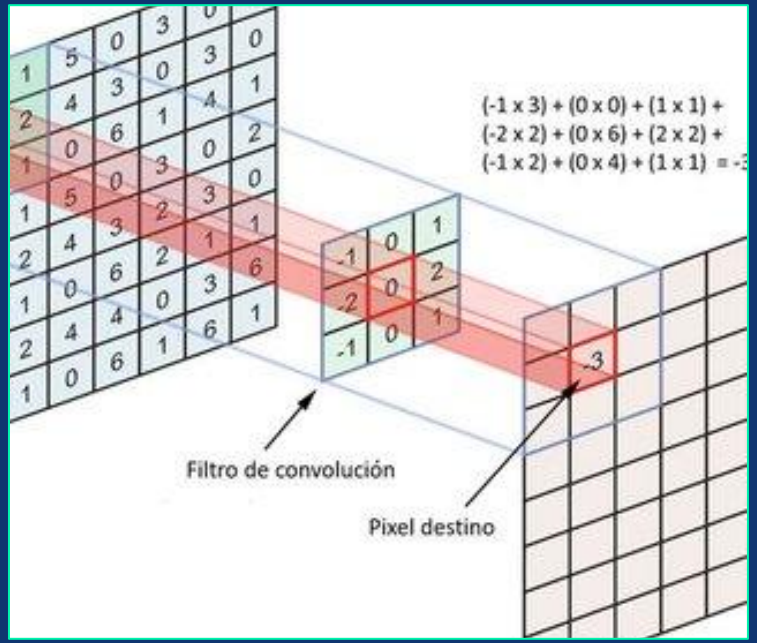
- Unidad de procesamiento ASIC
- Circuitos integrados desarrollados para el aprendizaje de máquinas
- Mayor volumen de cálculo
- Adaptadas a la librería TensorFlow

PRINCIPALES USOS

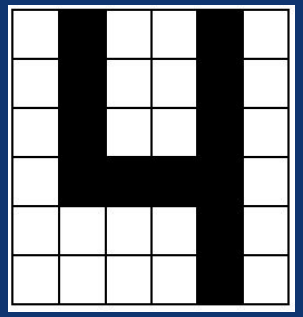
TPU se centra en operaciones vectoriales y operaciones sobre matrices con números con baja precisión.

MATRICES DE CONVOLUCIÓN

Presentes en modelos que trabajan con imágenes (clasificación o detección); software de tratamiento de imágenes, …



Queremos aplicar el siguiente filtro a una matriz de 6x6 de una imagen en blanco y negro traducida a 1’s y 0’s



0	**1**	0	0	**1**	0
0	**1**	0	0	**1**	0
0	**1**	0	0	**1**	0
0	**1**	**1**	**1**	**1**	0
0	0	0	0	**1**	0
0	0	0	0	**1**	0

1	0	1
0	1	0
1	0	1

PASO 1:

Matriz m

0	1	0	0	1	0
0	1	0	0	1	0
0	1	0	0	1	0
0	1	1	1	1	0
0	0	0	0	1	0
0	0	0	0	1	0

Matriz n

1	0	1
0	1	0
1	0	1

Matriz r

1			

$$r_{(1,1)} = m_{(1,1)}*n_{(1,1)} + m_{(1,2)}*n_{(1,2)} + m_{(1,3)}*n_{(1,3)} + m_{(2,1)}*n_{(2,1)} + m_{(2,2)}*n_{(2,2)} + m_{(2,3)}*n_{(2,3)} + m_{(3,1)}*n_{(3,1)} + m_{(3,2)}*n_{(3,2)} + m_{(3,3)}*n_{(3,3)}$$

$$r_{(1,1)} = 0*1 + 1*0 + 0*1 + 0*0 + 1*1 + 0*0 + 0*1 + 1*0 + 0*1 = 1$$

PASO 2:

Matriz m

0	**1**	0	0	**1**	0
0	**1**	0	0	**1**	0
0	**1**	0	0	**1**	0
0	**1**	**1**	**1**	**1**	0
0	0	0	0	**1**	0
0	0	0	0	**1**	0

Matriz n

1	0	1
0	1	0
1	0	1

Matriz r

1	**2**		

$$r_{(2,1)} = m_{(1,2)}*n_{(1,1)} + m_{(1,3)}*n_{(1,2)} + m_{(1,4)}*n_{(1,3)} + m_{(2,2)}*n_{(2,1)} + m_{(2,3)}*n_{(2,2)} + m_{(2,4)}*n_{(2,3)} + m_{(3,2)}*n_{(3,1)} + m_{(3,3)}*n_{(3,2)} + m_{(3,4)}*n_{(3,3)}$$

$$r_{(1,1)} = 1*1 + 0*0 + 0*1 + 1*0 + 0*1 + 0*0 + 1*1 + 0*0 + 0*1 = 2$$

PASO 3:

Matriz m

0	**1**	0	0	**1**	0
0	**1**	0	0	**1**	0
0	**1**	0	0	**1**	0
0	**1**	**1**	**1**	**1**	0
0	0	0	0	**1**	0
0	0	0	0	**1**	0

Matriz n

1	0	1
0	1	0
1	0	1

Matriz r

1	2	**2**	

$$r_{(3,1)} = m_{(1,3)}*n_{(1,1)} + m_{(1,4)}*n_{(1,2)} + m_{(1,5)}*n_{(1,3)} + m_{(2,3)}*n_{(2,1)} + m_{(3,4)}*n_{(2,2)} + m_{(2,5)}*n_{(2,3)} + m_{(3,3)}*n_{(3,1)} + m_{(4,4)}*n_{(3,2)} + m_{(3,5)}*n_{(3,3)}$$

$$r_{(1,1)} = 1*1 + 0*0 + 0*1 + 1*0 + 0*1 + 0*0 + 1*1 + 0*0 + 0*1 = 2$$

RESULTADO:

Matriz m

0	**1**	0	0	**1**	0
0	**1**	0	0	**1**	0
0	**1**	0	0	**1**	0
0	**1**	**1**	**1**	**1**	0
0	0	0	0	**1**	0
0	0	0	0	**1**	0

Matriz n

1	0	1
0	1	0
1	0	1

Matriz r

1	2	2	1
2	3	3	2
1	2	3	1
1	2	3	2

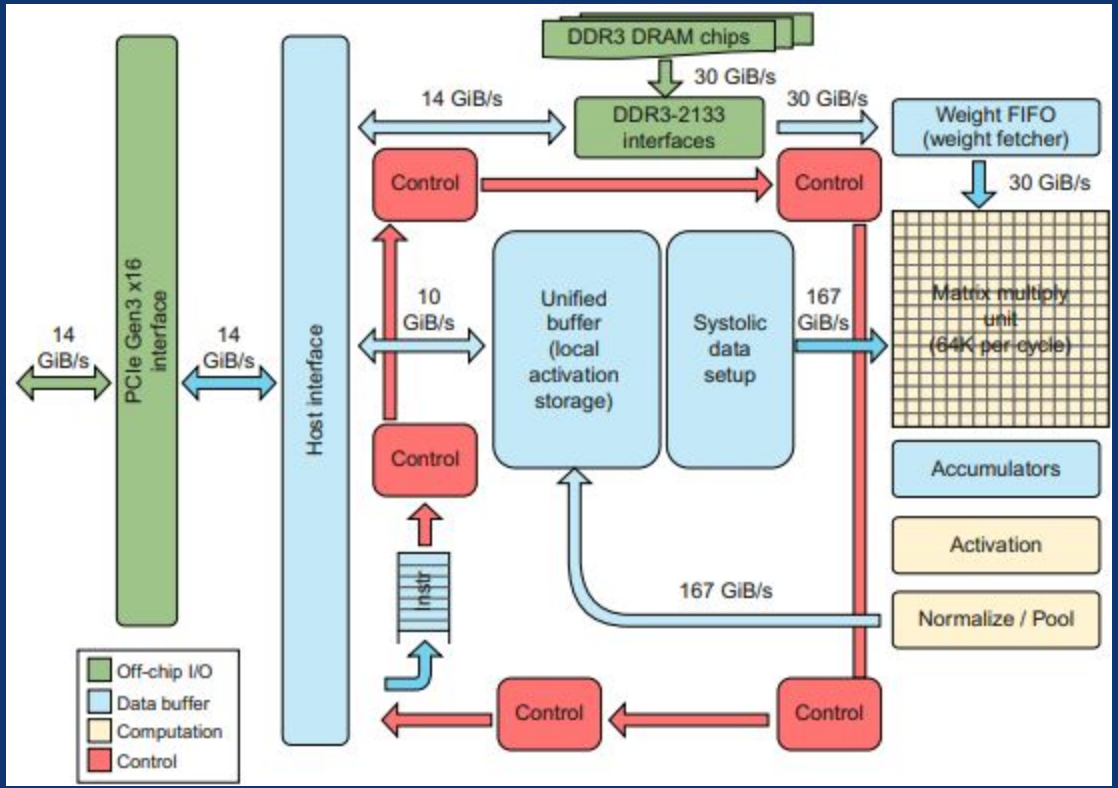
3. TPU DE GOOGLE

- ORIGEN

Primer TPU de Google

TPU implementado en un centro de datos de Google

- ARQUITECTURA

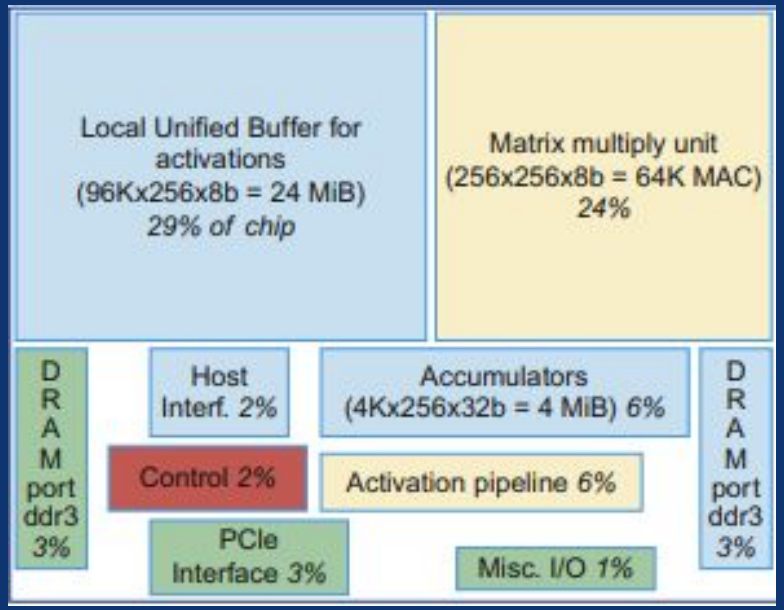


- TPU ISA

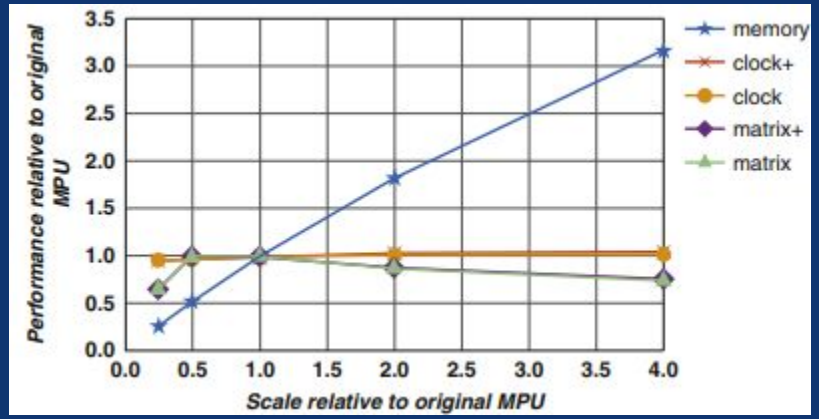
  - READ_HOST_MEMORY
  - READ_WEIGHTS
  - MatrixMultiply / Convolve
  - ACTIVATE
  - WRITE_HOST_MEMORY

- SOFTWARE

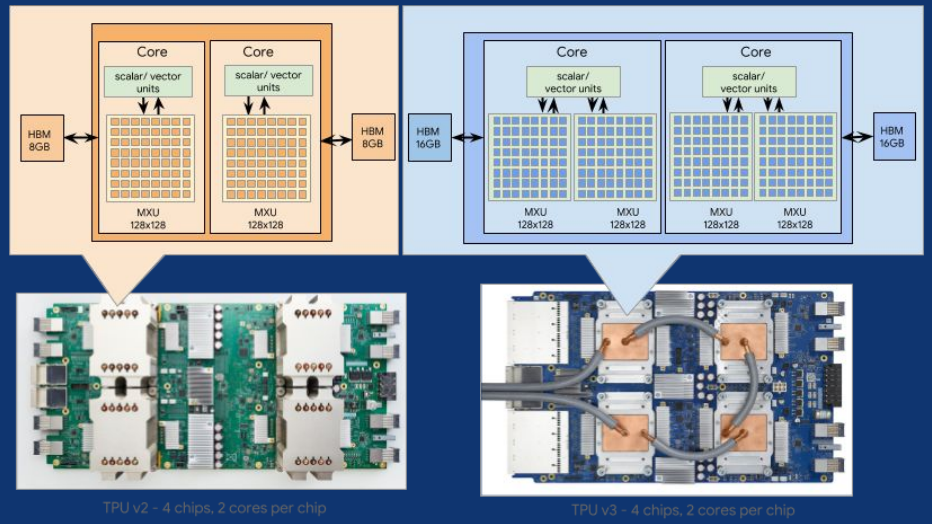
- DISEÑO



- MEJORAS


Performance relative to original MPU
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Scale relative to original MPU
memory
clock+
clock
matrix+
matrix

4. GUÍA DE DISEÑO DE ARQUITECTURAS DE DOMINIO

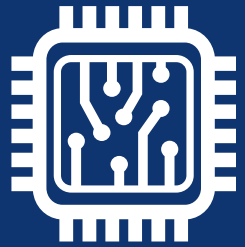


5 PRINCIPIOS

- Usar memorias dedicadas para minimizar distancias
- Invertir los recursos ahorrados en más unidades aritméticas o memorias más grandes
- Usar la forma más fácil de paralelismo que coincida con el dominio
- Reducir el tamaño y el tipo de datos al mínimo necesario para el dominio
- Usar un lenguaje de programación específico del dominio para portar el código al DSA

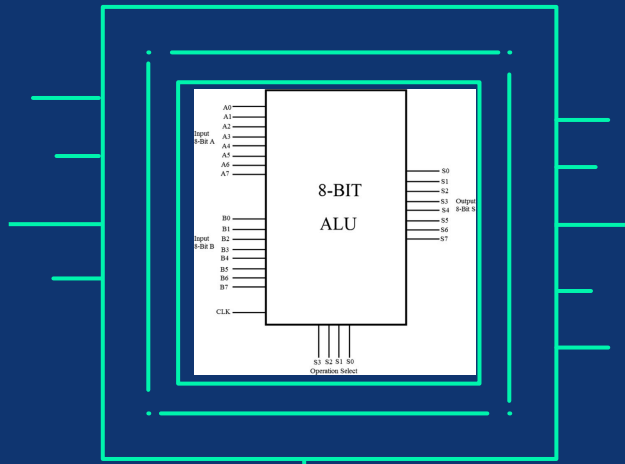
1. USAR MEMORIAS DEDICADAS PARA MINIMIZAR LA DISTANCIA SOBRE LA CUAL SE MUEVEN LOS DATOS

- Búfer unificado de 24 MiB
- 4 Acumuladores MiB
- Memoria DRAM
- Cola FIFO



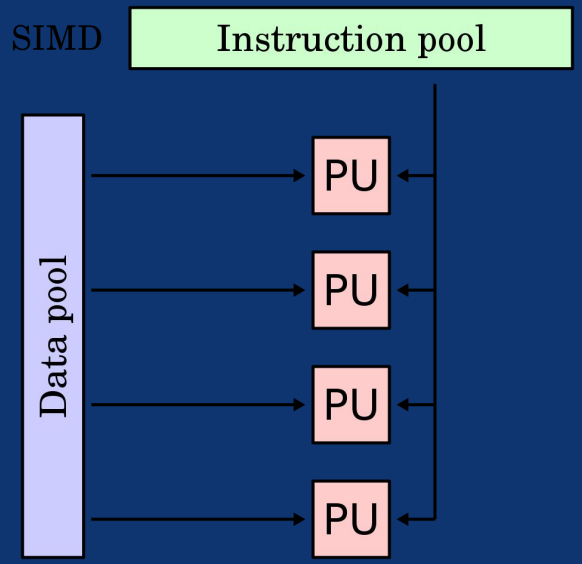
2. INVERTIR LOS RECURSOS AHORRADOS AL COLOCAR OPTIMIZACIONES MICROARQUITECTÓNICAS AVANZADAS EN MÁS UNIDADES ARITMÉTICAS O MEMORIAS MÁS GRANDES

- 28 MiB de memoria dedicada
- 65.536 ALU’s de 8 bits



3. USAR LA FORMA MÁS FÁCIL DE PARALELISMO QUE COINCIDA CON EL DOMINIO

- Paralelismo SIMD bidimensional



4. Reducir el tamaño y el tipo de datos al mínimo necesario para el dominio

- Cálculos en enteros de 8 bits

5. USAR UN LENGUAJE DE PROGRAMACIÓN ESPECÍFICO DEL DOMINIO PARA PORTAR EL CÓDIGO AL DSA

- TensorFlow

5. BIBLIOGRAFÍA

- https://www.paradigmadigital.com/dev/tpus-google-imparable-avance-hardware-ml/
- The Pixel 6’s Tensor processor promises to put Google’s machine learning smarts in your pocket - The Verge
- https://www.muycomputer.com/2016/05/19/tpu-chip-maquinas-aprendan/