PROCESSING CONVOLUTIONAL NEURAL NETWORKS ON CACHE

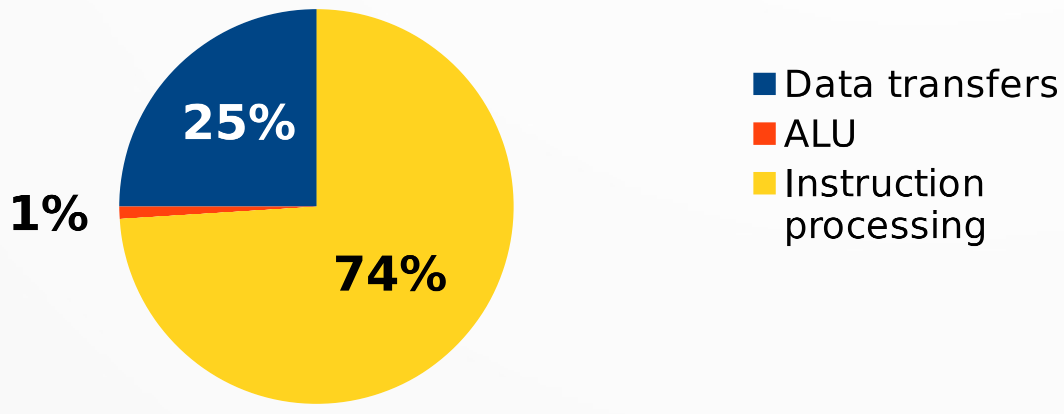
João Vieira, Nuno Roma*, Gabriel Falcão†, and Pedro Tomás**

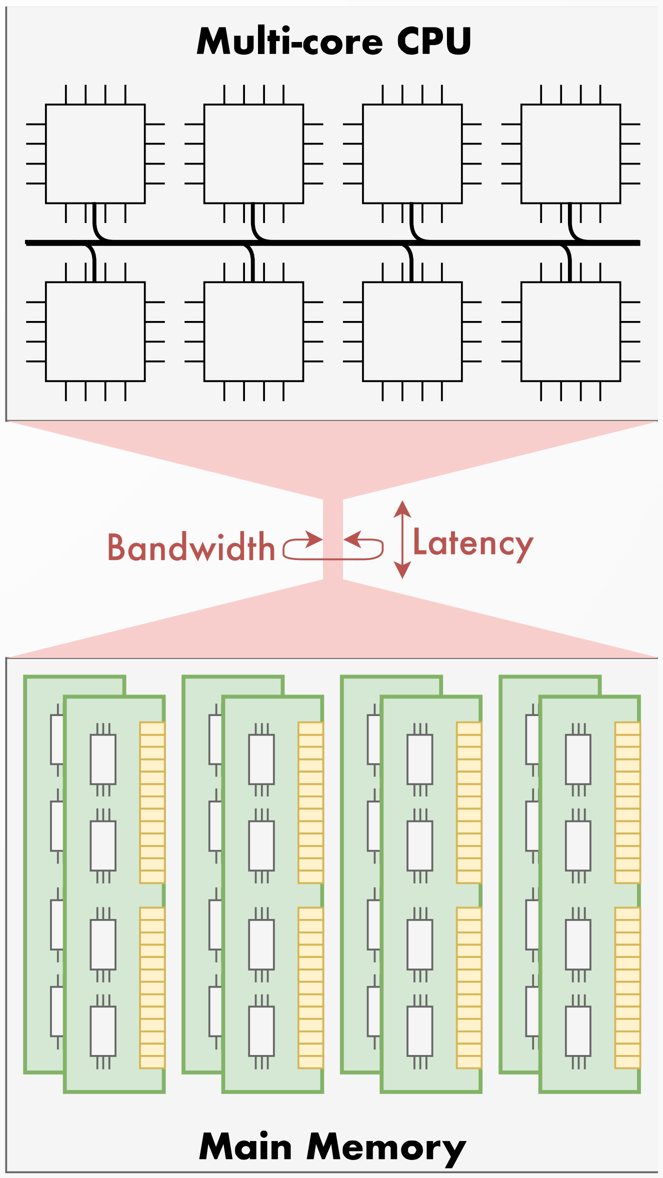
* INESC-ID, Instituto Superior Técnico, University of Lisbon, Portugal
† Instituto de Telecomunicações, University of Coimbra, Portugal

Motivation

- **The memory subsystem can limit the system's performance:**
  - Bandwidth;
  - Latency.

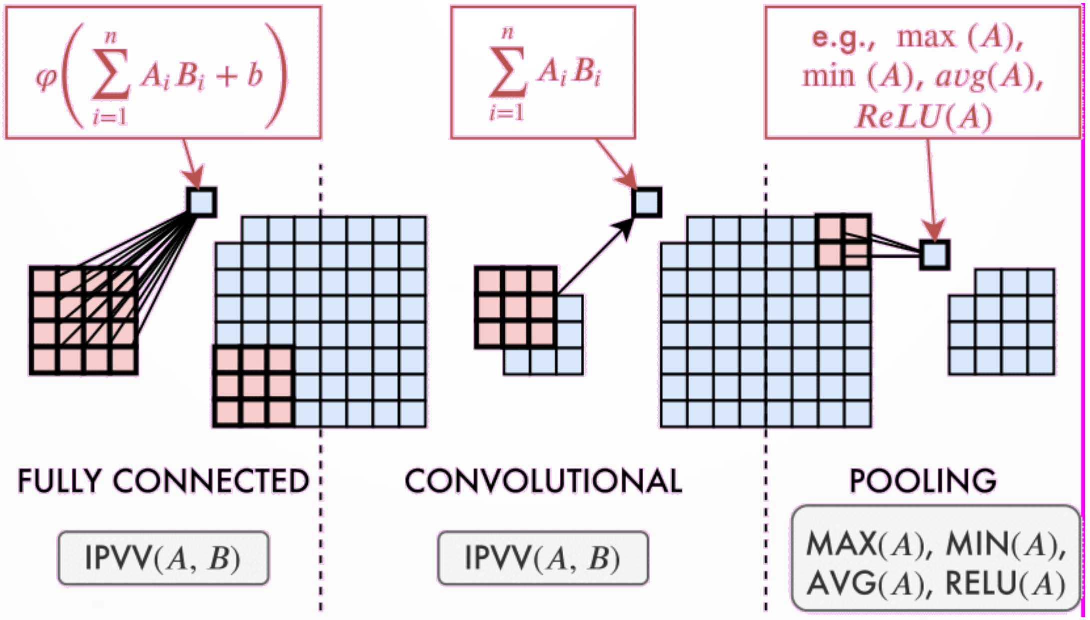
Energy distribution of a processing core





R. Das et al., "Compute caches," in HPCA. IEEE Computer Society, 2017, pp. 481–492;

Convolutional Neural Networks



- *Convolutional Neural Networks* (CNNs) are characterized by massive datasets;
- Depending on the CNN architecture, thousands of elementary operations are performed:
  - Usually, over **90% of the CNN's workload is due to convolutions** (convolutional and pooling layers).

J. Cong et al., "Minimizing Computation in Convolutional Neural Networks," in *ICANN*, 2014, vol. 8681 of *Lecture Notes in Computer Science*, pp. 281-290, Springer.

It would be more efficient not to transfer data between the memory subsystem and the processing elements, but instead **perform the computation in-place, in parallel**

Near-Data-Processing

- ***Processing-In-Memory* (PIM)** solutions appeared during the 1980s;
  - Mostly based on **bit-line processing**;
  - Hard to integrate before **3D-stacking** (introduced much later);
  - **Hard to use** (each solution had its own programming paradigm);
  - Recently, PIM evolved to enable analog and digital processing in *Resistive Random Access Memories* (RRAMs)
- Some solutions use custom hardware to perform computation near the memory:
  - Not as efficient as bit-line computation;
  - Allow more **complex operations**.
- *Near-Data-Processing* (NDP) was recently ported to **cache**.

A. Shafiee et al., “ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars,” in ISCA, 2016, pp. 14-26;

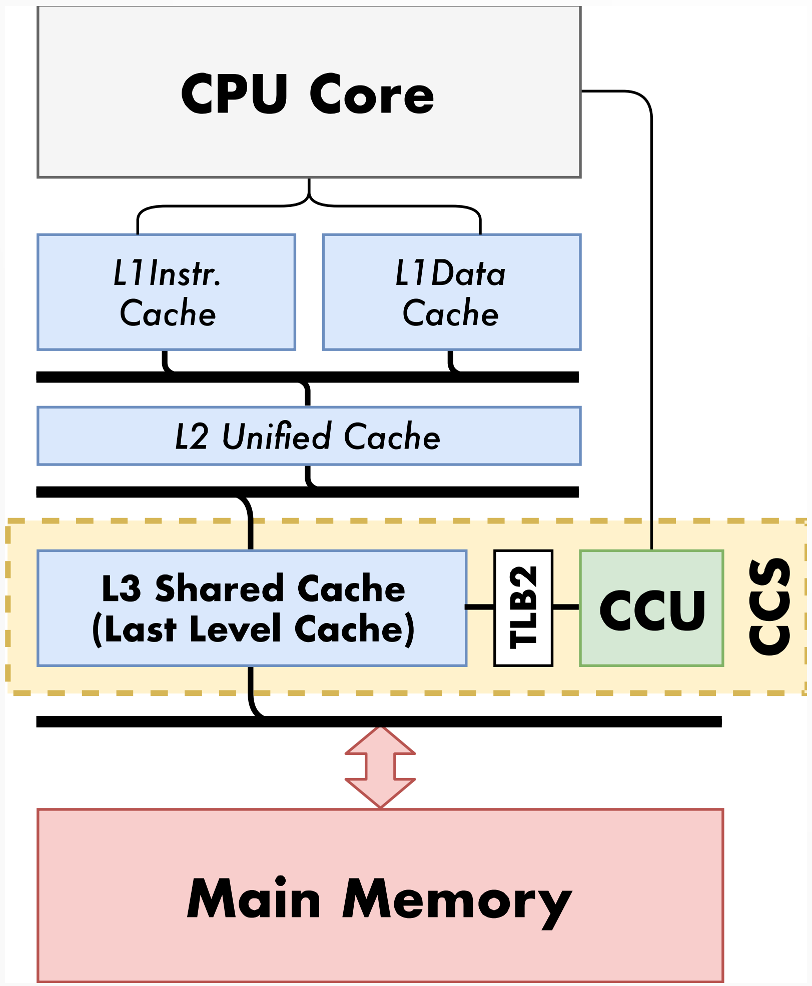
R. Das et al., “Compute caches,” in HPCA. IEEE Computer Society, 2017, pp. 481–492;

T. Mowry et al., “Fast bulk bitwise AND and OR in DRAM”, Computer Architecture Letters, vol. 14, no. 2, pp. 127–131, 2015;

O. Mutlu et al., “Pim-enabled instructions: a low-overhead, locality-aware processing-in-memory architecture,” in ISCA. ACM, 2015, pp. 336–348;

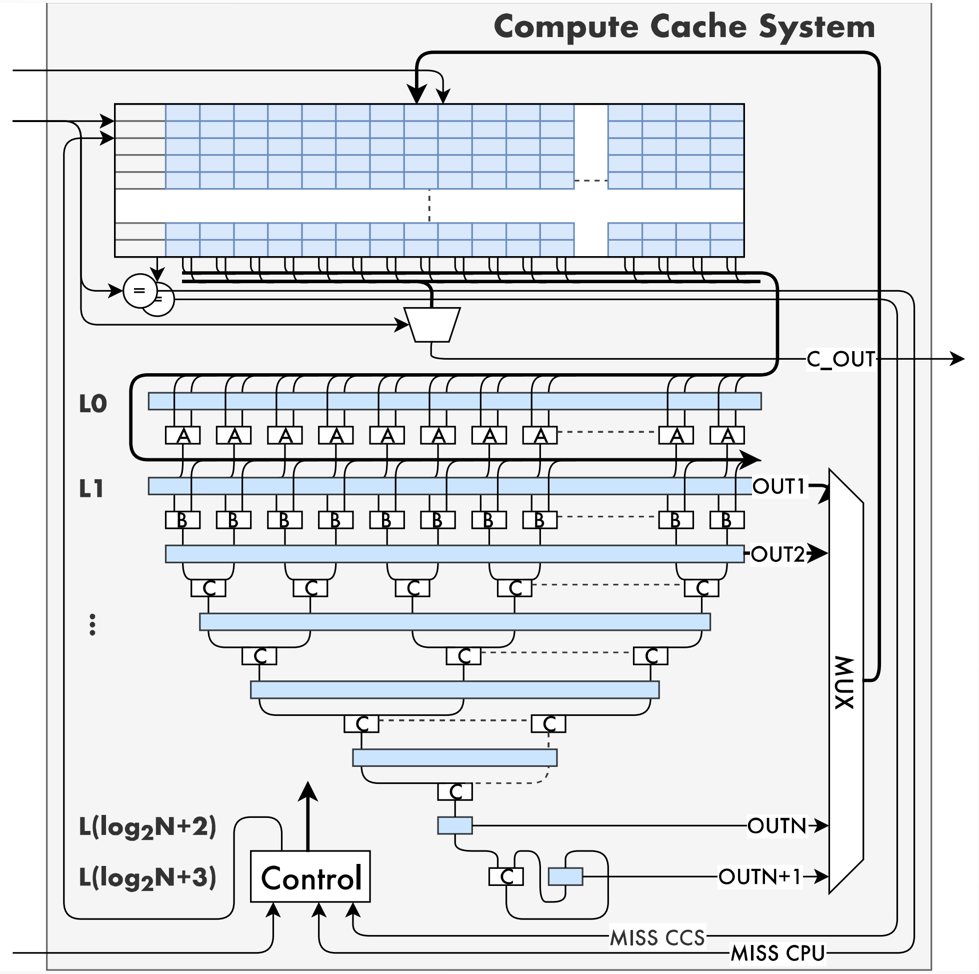
R. Das et al., “Cache automaton,” in MICRO. ACM, 2017, pp. 259–272.

Compute Cache System

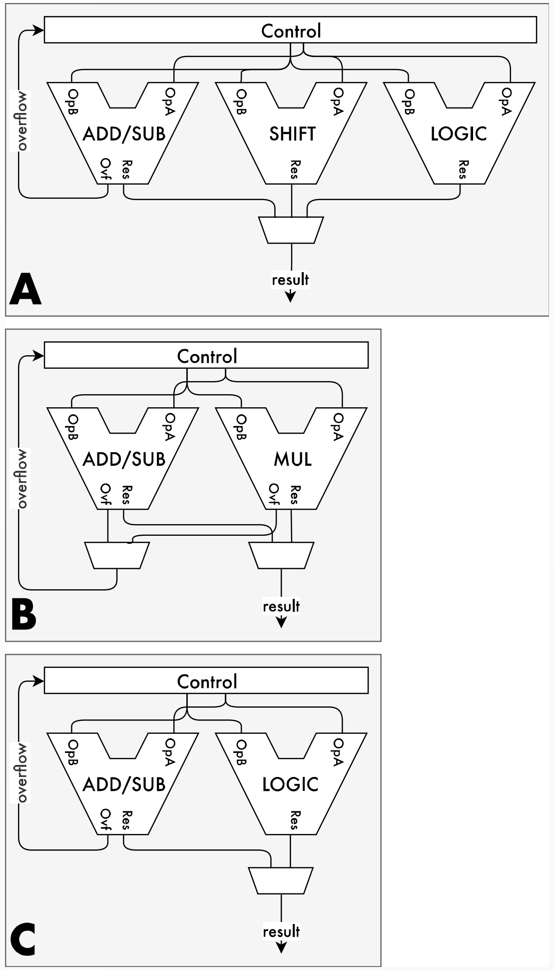


- **Dedicated hardware to perform parallel computation near the cache,** taking advantage of:
  - Operands locality;
  - Long cache lines;
  - Lower latency to the main memory;
  - Existing cache coherency protocols.
- **The CCS is integrated with an existing processing system:**
  - As a slave of the processing core;
  - Communicating with the LLC to get the operands;
  - Communicating with the main memory in case of cache miss.

System Schematics

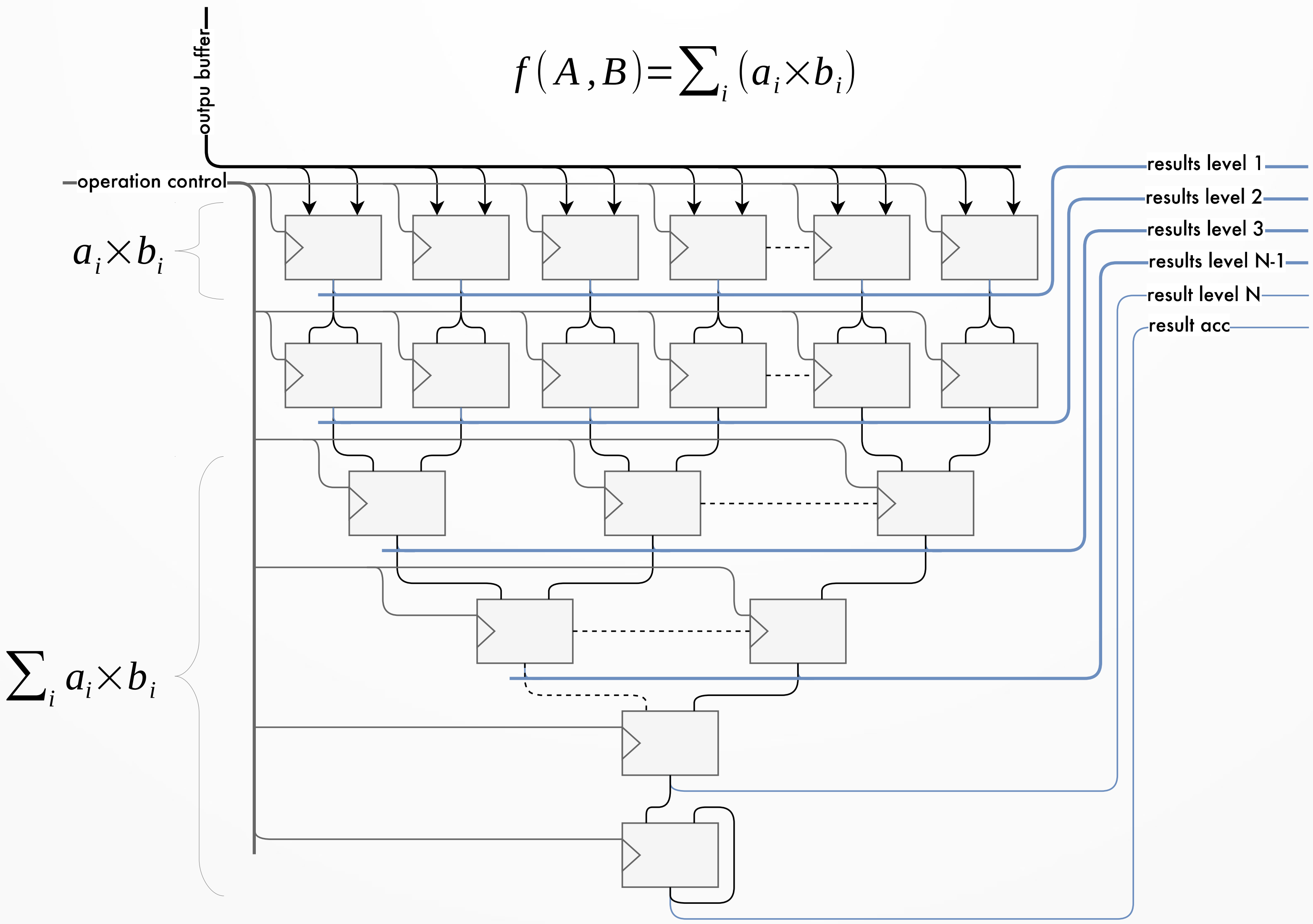


47 Commands

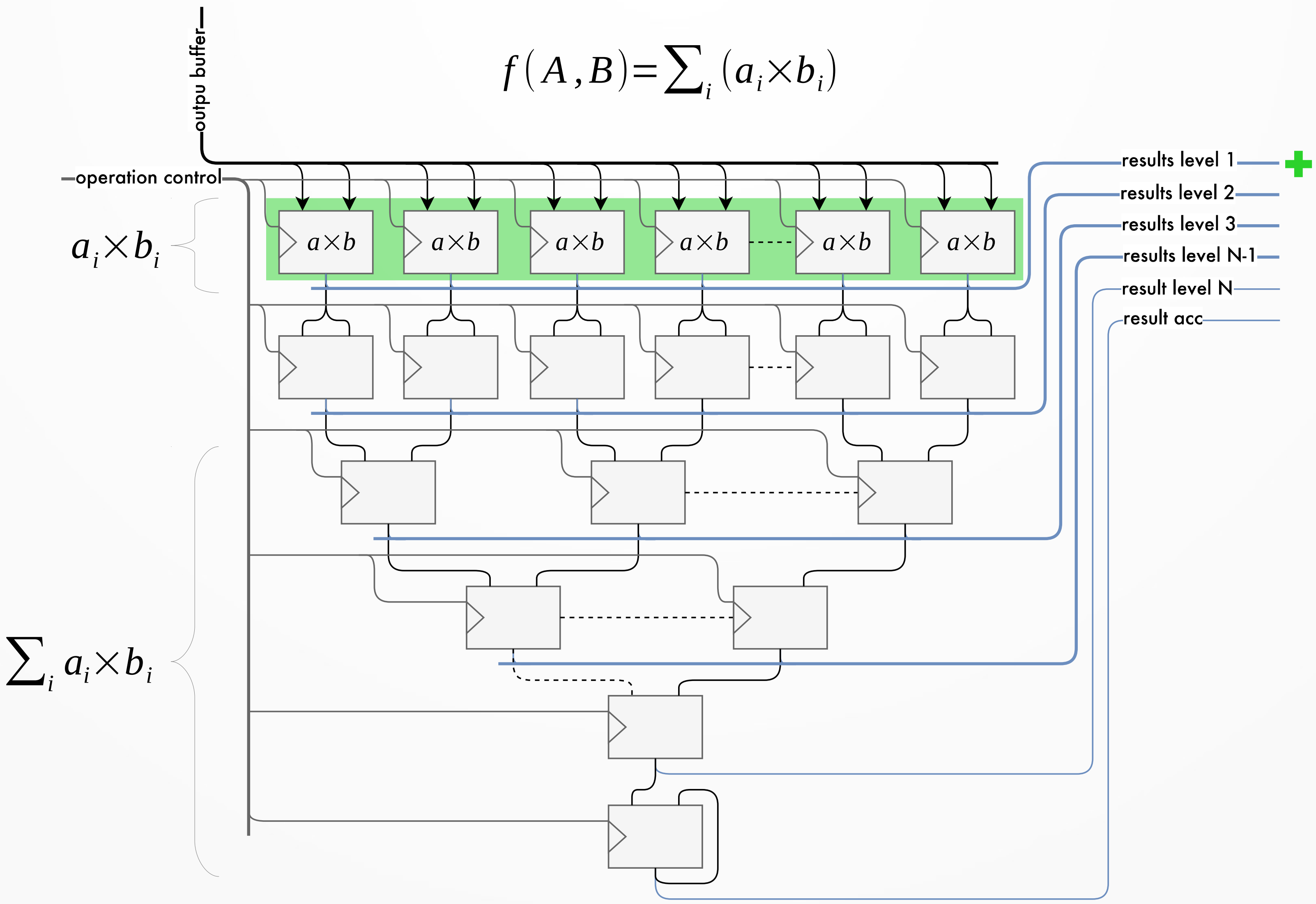


Lets check an operation example!

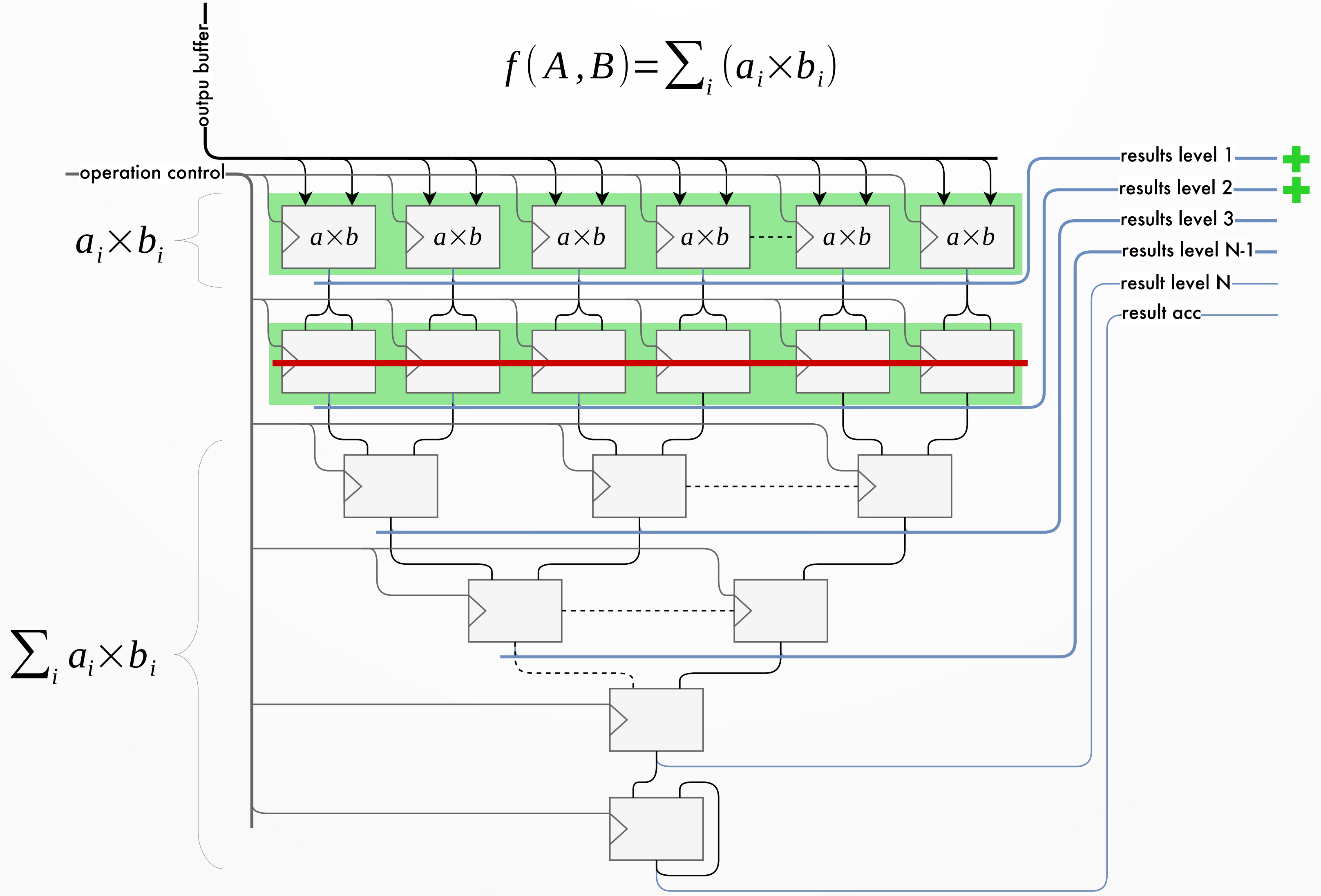
Example: 2-D convolution (1)



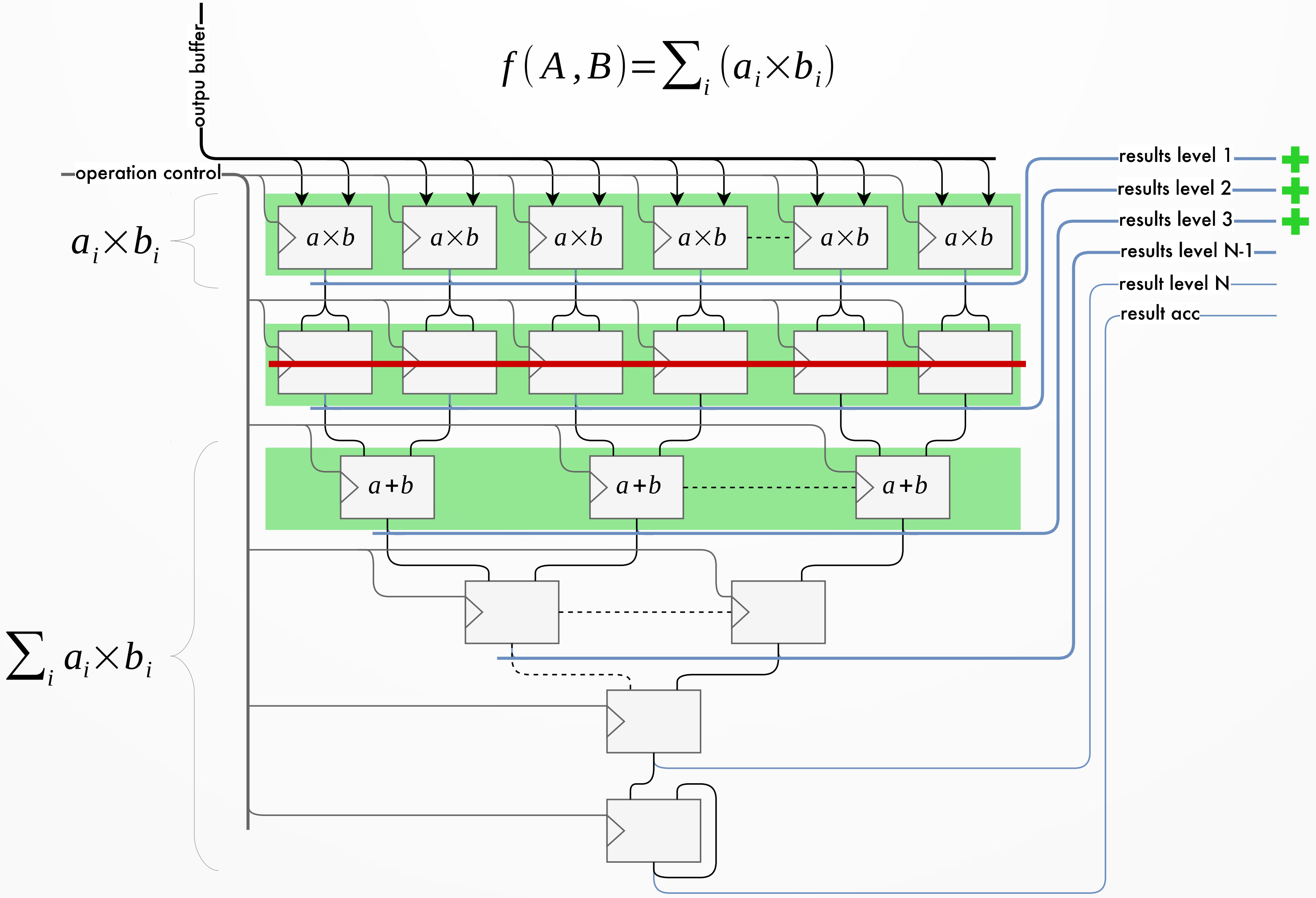
Example: 2-D convolution (2)



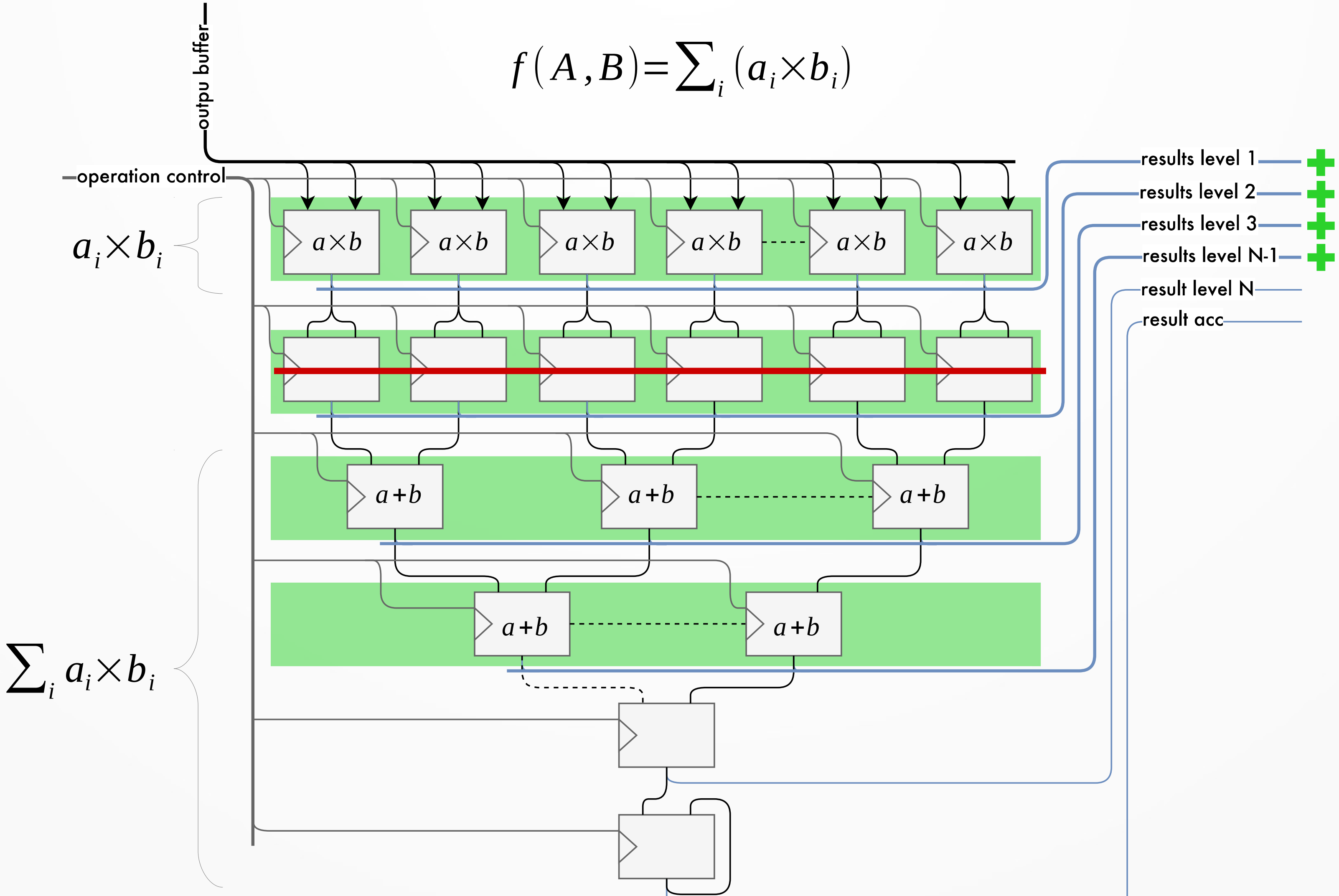
Example: 2-D convolution (3)



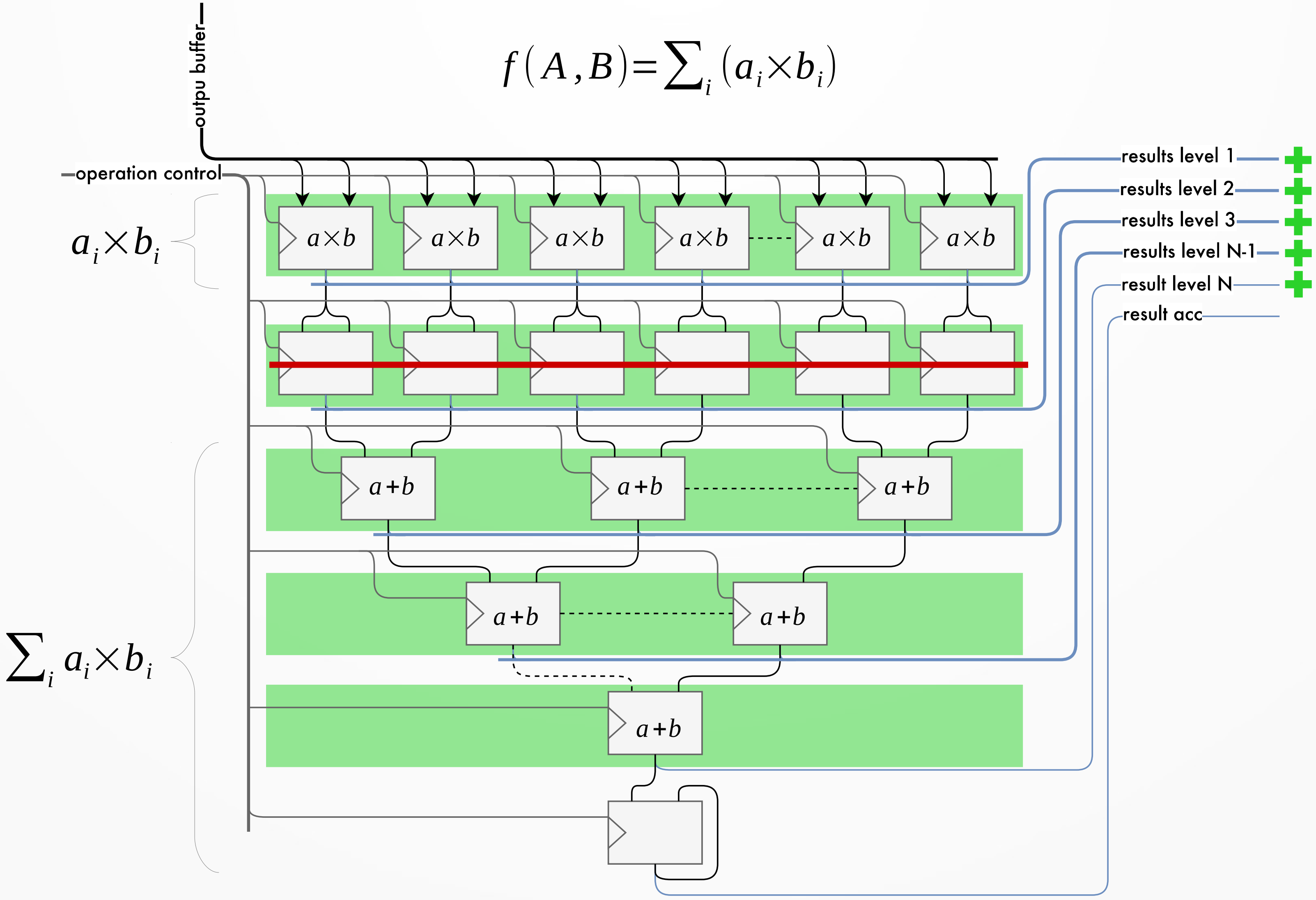
Example: 2-D convolution (4)



Example: 2-D convolution (5)

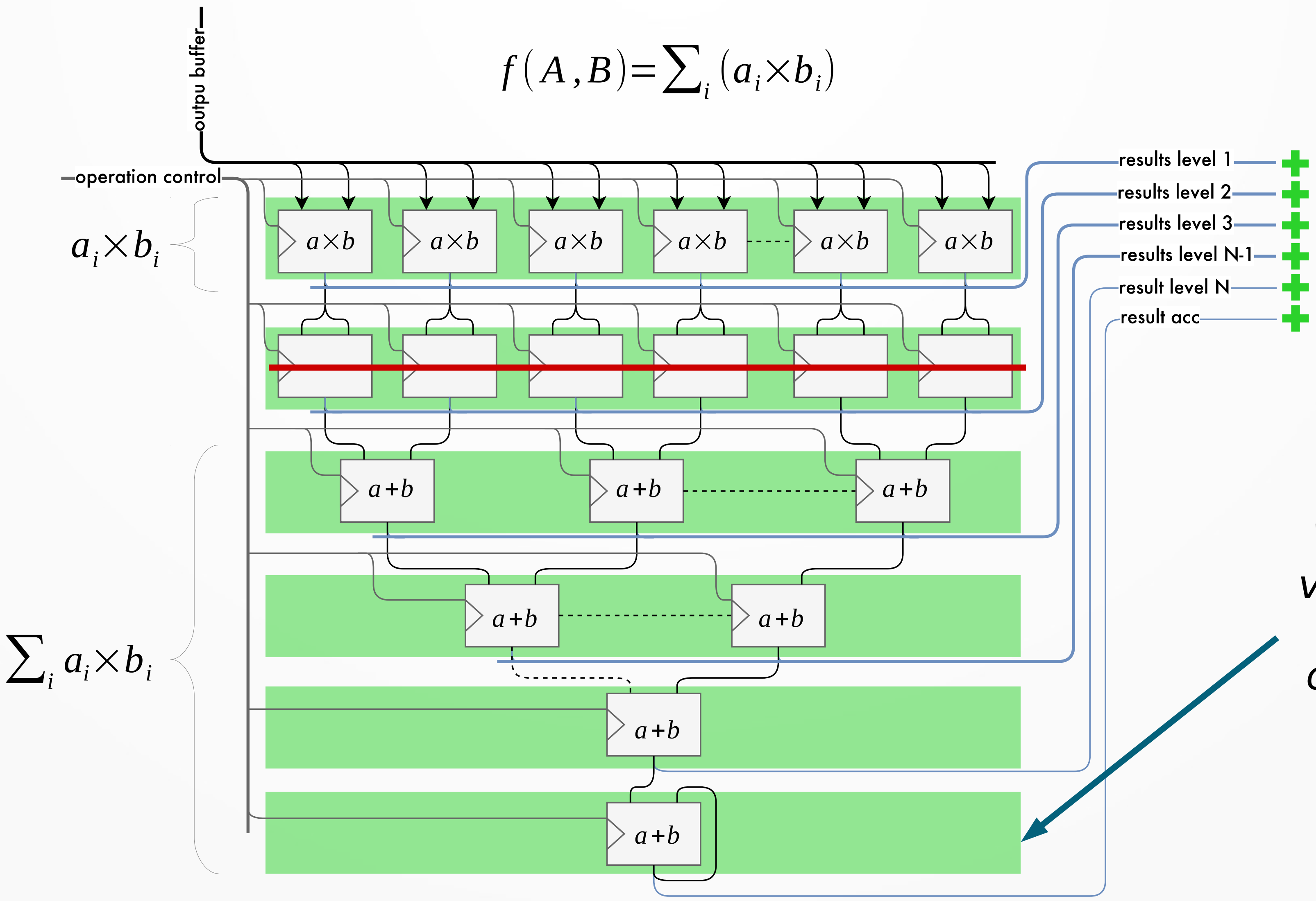


Example: 2-D convolution (6)



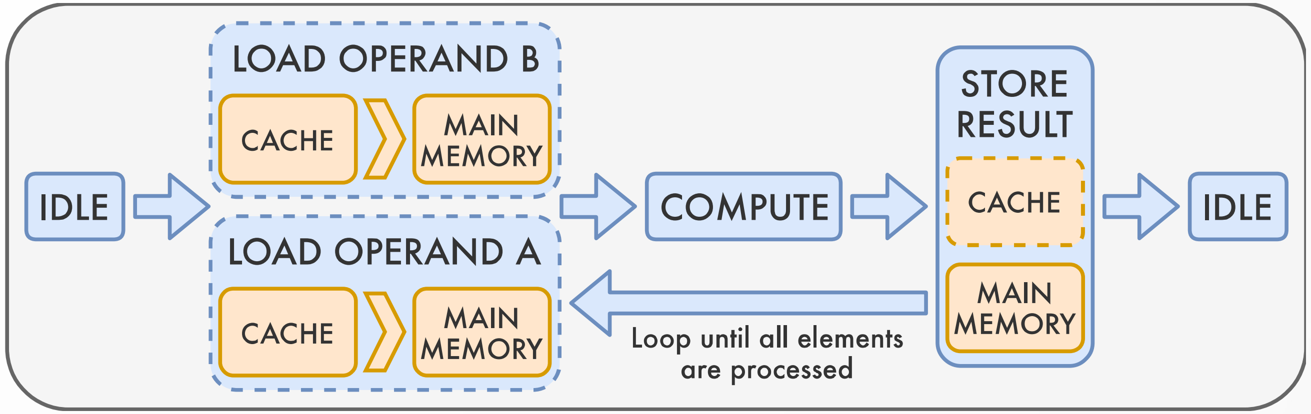
Example: 2-D convolution (7)

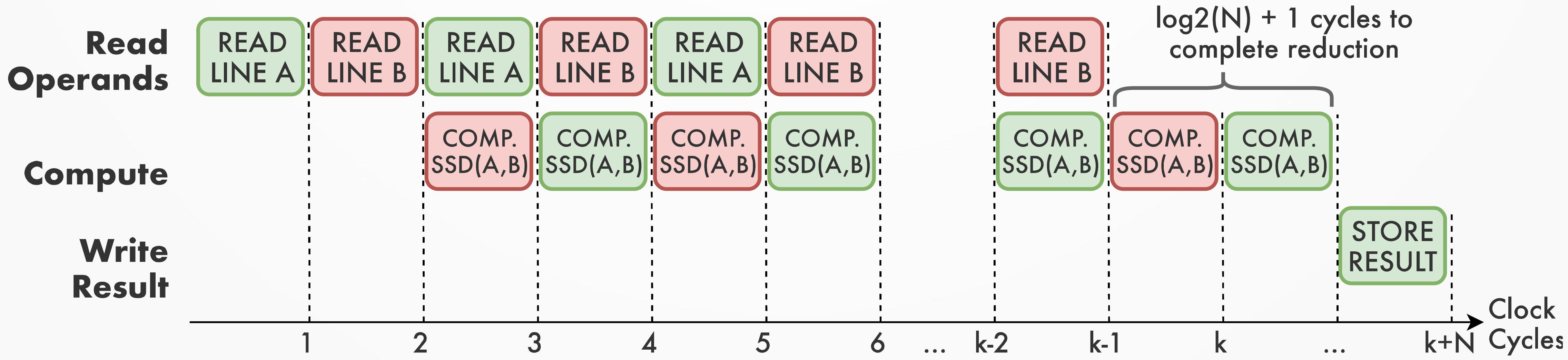
$$f(A,B)=\sum_i (a_i \times b_i)$$



What if the vector is too big to be operated at once?

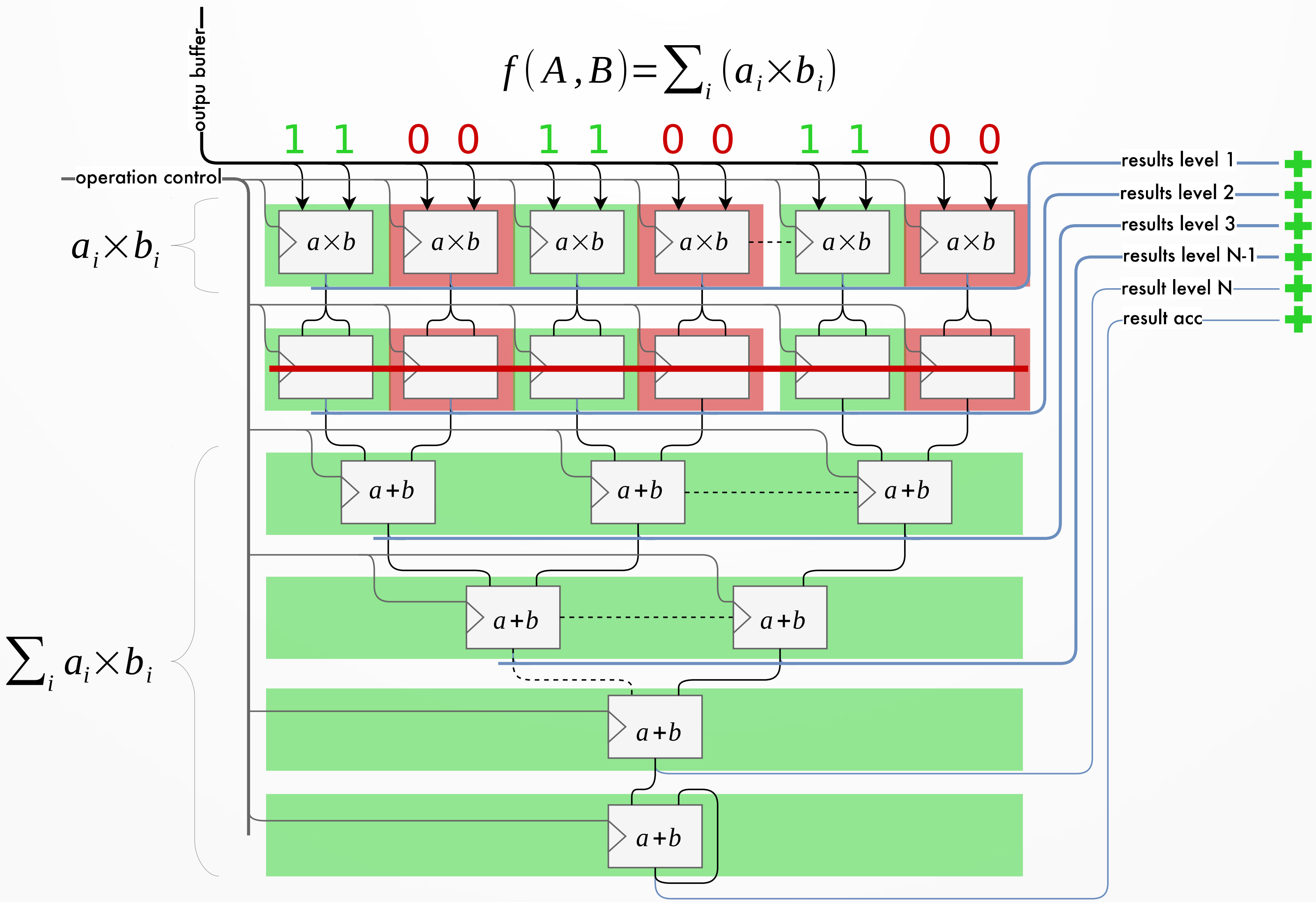
Pipeline execution/hardware loops





But what if the vectors are not aligned or continuous in memory?

Execution mask

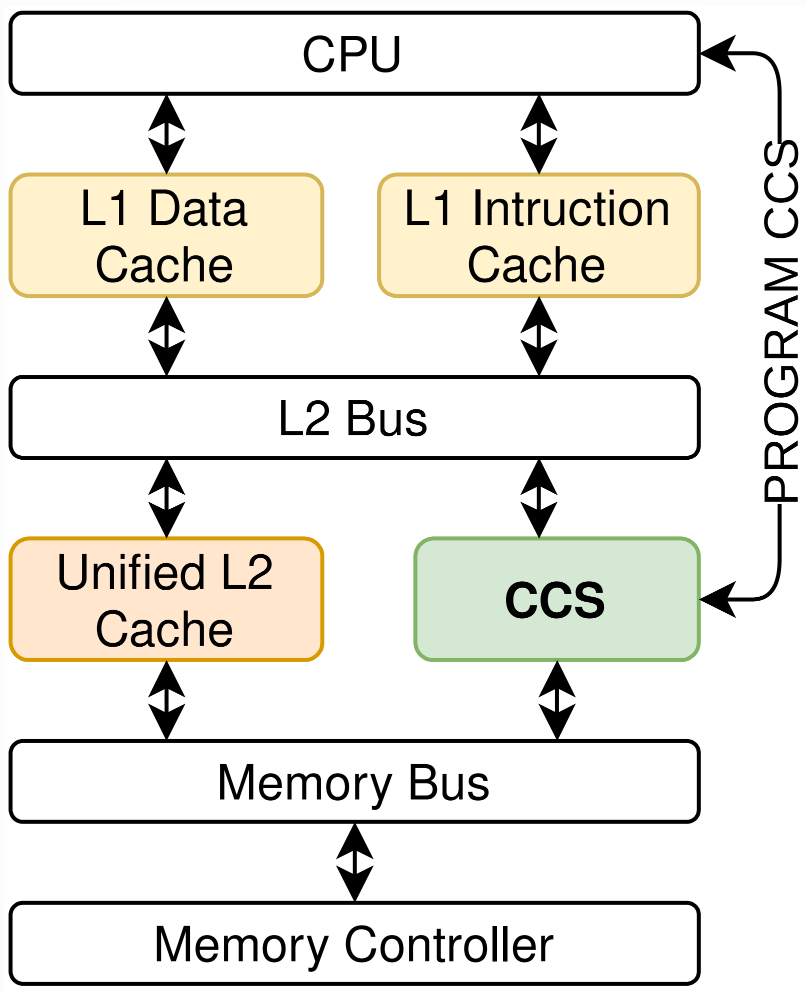


Simulation methodology

- Simulated using **gem5 architectural simulator;**
- **ARM Cortex-A53** model was enhanced with **gem5-X** for increased simulation accuracy;
- CCS was connected to the CPU through a **memory mapped interface** and **coupled to the L2 (LLC) cache;**
- A **second TLB and page-walker** was added to the CCS for address translation;
- The simulated version of the CCS is capable of processing **up to 2 16-element vectors of 32-bit fixed-points per cycle**;

L. Nathan et al., *"The gem5 simulator,"* SIGARCH Computer Architecture News, vol. 39, no 2, pp. 1-7, 2011;

Y. Qureshi et al., *"Gem5-X: a Gem5-Based System Level Simulation Framework to Optimize Many-Core Platforms,"* in SpringSim. 2019, pp. 1-12, IEEE.


CPU
L1 Data Cache
L1 Intruction Cache
L2 Bus
Unified L2 Cache
CCS
Memory Bus
Memory Controller
PROGRAM CCS

Programming the CCS

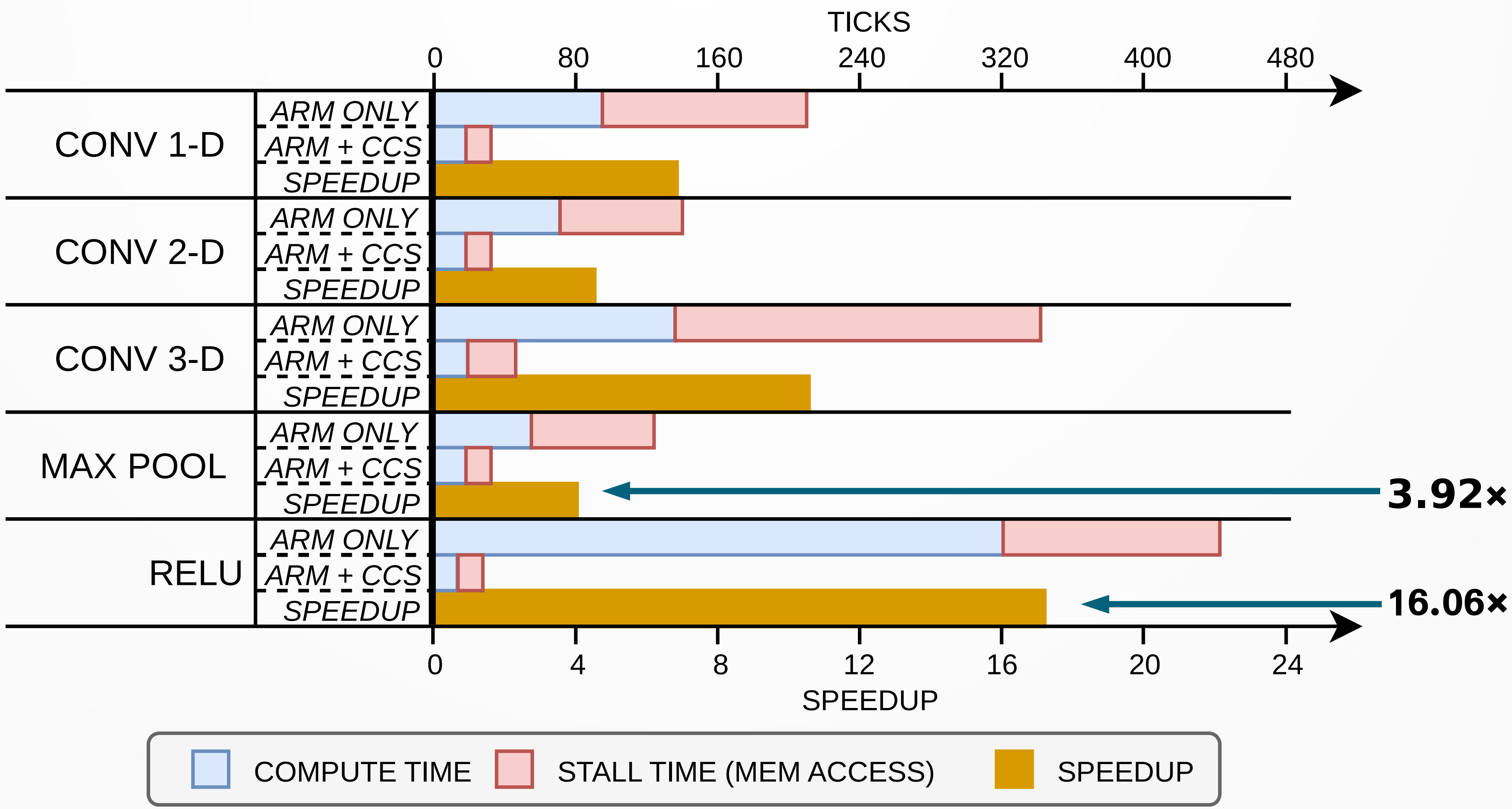
```
#define L_SIZE          // size of cache line
#define DATA_WIDTH      // width of data matrix
#define DATA_HEIGHT     // height of data matrix
#define KERNEL_LENGTH   // size of the kernel

external int **data;    // data addr
external int *kernel;   // kernel addr
external int **result;  // result addr
int a[L_SIZE];          // data buffer

// 2-D convolution
for (int i = 0; i < DATA_WIDTH; i++) {
  for (int j = 0; j < DATA_HEIGHT; j++) {
    gather_data(a, i, j);
    ccs_setup(IPVV, KERNEL_LENGTH, 0, a, kernel,
        result + i * DATA_WIDTH + j, 1);
    ccs_start();
  }
}
ccs_wait();
```

- **Library** (written in ARMv8 Assembly) to allow low-level control over the CCS;
- **Framework** (written in C) to offload convolutional and polling layers of CNNs to the CCS.

Experimental results



Summary

- The **CCS brings several advantages**:
  - Data is not transferred to the core;
  - Computation is performed in parallel;
  - Cycle control overhead is drastically reduced;
  - Pipelined data path allows to increase the performance even further.

PROCESSING CONVOLUTIONAL NEURAL NETWORKS ON CACHE

João Vieira

joaomiguelvieira@tecnico.ulisboa.pt