

Hardware-centric approach enables joint algorithm-architecture design.

Standard software approach limits the potential of the algorithm because it must be designed for a given predesigned architecture

The following diagram represents two algorithms that perform the same task. The one on the left is designed to run on standard Von Neumann architectures. The diagram on the right is designed to run on configurable hardware. In configurable hardware we can use programming languages such as VHDL to design jointly the algorithm and the architecture.

Note the advantages of designing the algorithm in hardware, i.e., codesigning the architecture along with the algorithm:

- The program flow is not a line, it is as many lines as you want.
- Memory I/O can be done simultaneously in the processes.
- The algorithm can be designed to have much greater performance, especially in terms of speed. Massive parallelism is possible without the need to use hundreds of computers.

In the diagram on the left, there are multiple bottlenecks: each red diamond represents a loop control. The flow goes up and down several times, and memory I/O must be done in one step. This means that every time the process needs more memory, a call to a memory device must be made.

In hardware there is no loop needed to read the memory, since the memory can be read while the process is being executed. Also, several processes can be executed at the same time, and their partial results can be computed at the same time.

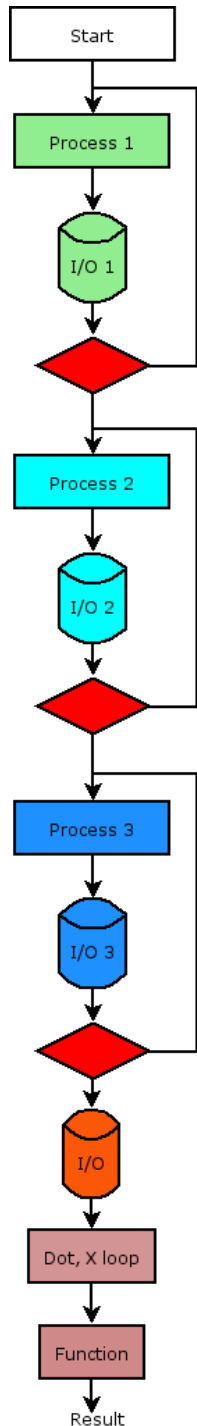
Another great advantage of designing the algorithm jointly with the architecture is that any type of instruction can be implemented. There are no limits to what we can achieve to make the algorithm work. For example, if we believe it is important, we can design a dot product between vectors of 100 components that runs in a single clock cycle. We design our own non- von Neumann architecture to do it. At execution time, this would take only one instruction, and can be done in parallel with other hardware instructions if we desire it so.

To accomplish the dot product of two 100-component vectors in a classical Von Neumann architecture, a loop is required and would take as much as 700 instructions of the form: Read the vector1 i^{th} component, read the vector2 i^{th} component, multiply them, increase the loop counter, test the ending condition, loop.

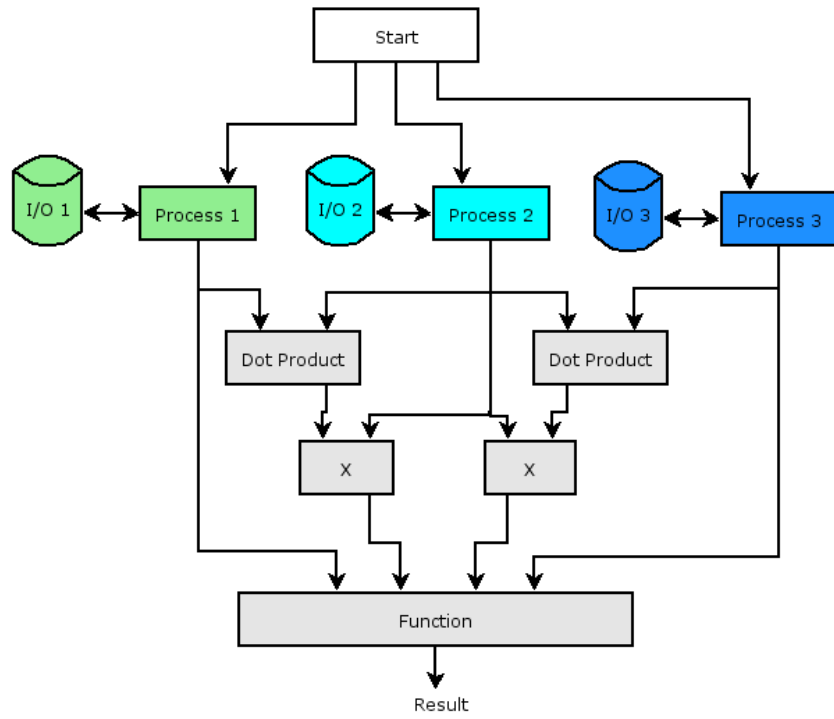
In the diagram, the red color is used to indicate where these so-called “von Neumann bottlenecks” are located.

With the implementation of custom instructions, and having several processes running in parallel, speedups of up to 100,000X can be accomplished.

Von Neumann rigid architecture



Algorithm design in a flexible architecture



Joint architecture-algorithm design: In hardware, algorithms can be redesigned to be more efficient and to overcome inherent difficulties of the Von Neumann architecture. In this example, a slow algorithm, with bottlenecks (shown red, gray and orange) is redesigned jointly with its architecture. In this case, the system avoids bottlenecks by enabling an architecture with parallel memory access, which is more natural to the mathematical problem that needs to be solved.