Domain-Specific Architecture AI for Future Tech

Hailo's blog explores the impact of domain-specific architecture AI on computing, offering insights into its efficiency and power.

By Hailo Team

5 min. read · View original

The End of Moore's Law and Dennard Scaling

For over 30 years, the evolution of general-purpose computing was impressive and fast-paced, doubling performance every 2 years or so. But then, some would say not unexpectedly, it hit the metaphorical wall. As so many have commented, the process progress has started to slow down, and Moore's Law entered its last throws. The process technology, which did most of the heavy lifting in increasing performance from generation to generation, is slowing down.

On top of this, we hit the end of Dennard Scaling due to the decrease in our ability to reduce supply voltage when moving from one process

technology to the next.

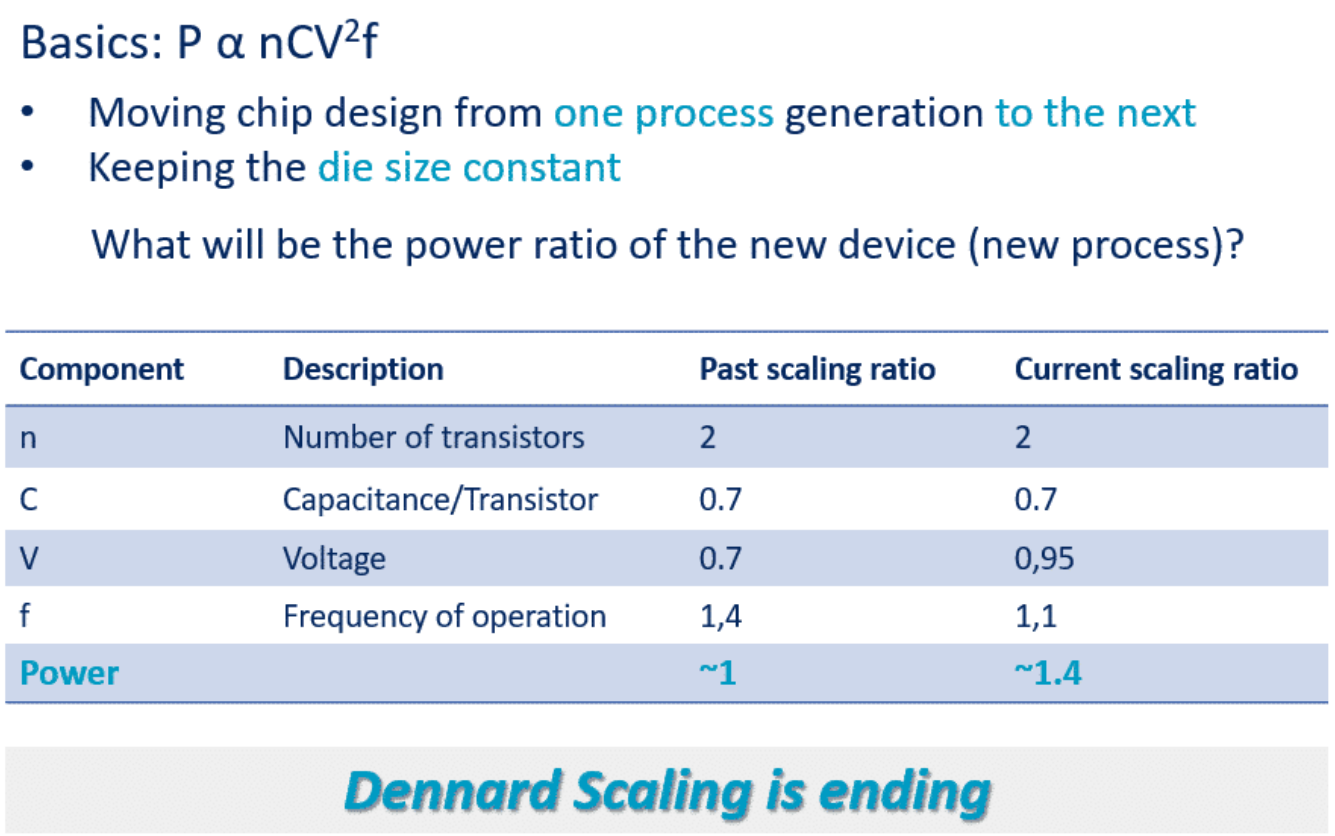

Figure 1: The end of Dennard Scaling

After years of process scaling down, we are no longer able to lower voltage (we are getting near the transistor's threshold Voltage), while frequency increase is slowing down when moving from one process to the next. When moving to the next process generation, while keeping the die area constant, the number of transistors will double, and the power will increase by 1.4X.

In addition, single thread general purpose performance increase has plateaued, while cooling devices reached “power envelop” (1). Thus, we have reached the point that not all the transistors on die can function at the same time. At any given moment only part of the on-die functions can be fully active. We call this situation “Black Silicon”. The same is true for multi-core chips, where not all cores can work simultaneously at their maximum speed. We had seen the exponential growth of usage of different applications in the cloud and on the edge. These new applications required performance and so they quickly crowned compute performance as king, his reign demanding increasingly greater power (2).

In today’s world, maximum efficiency takes precedence at both ends of the deployment spectrum: in edge devices that are limited in the energy they consume and in the mega server farms and datacenters whose skyrocketing power consumption has both financial and environmental consequences. Thus, the inevitable target of our current computing environment is to increase the possible performance per unit of power, i.e. reach the best possible power efficiency.

This motivation leads us to move away from “general purpose” computing and build “special purpose” compute engines (3). To understand why, let us look at the general-purpose Instruction Set Architecture (ISA) compute engines. In the ISA approach, we get maximum usage flexibility – you can run

any application on this kind of engine, but we “pay” for it in power efficiency. In CPUs, less than 10% of the consumed energy is expended on executing the instruction’s real function (e.g., “add”). The rest is spent reaching the instruction cache and decode/control the instruction.

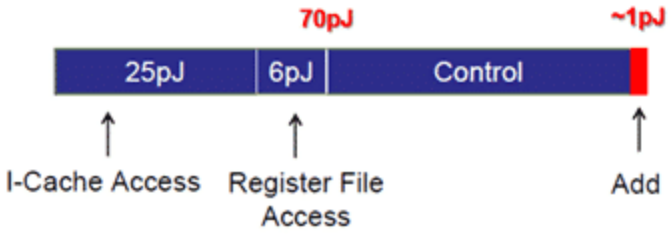

Figure 2: Instruction energy breakdown (@45nm technology) – only less than a tenth of the power consumption is dedicated to executing the instruction (2)

“But what about GPUs?” one might ask. The GPU was a development of “generalized execution” for graphic applications. It was a step towards specialization, albeit a wide one. The very repetitive nature of graphics operations allowed to “divide” the instruction overhead between large amount of data executions. It achieves greater efficiency than the CPU for a certain kind of applications, but it is still an ISA-based machine and, as such, it is hitting the same evolutionary wall.

Another implementation architecture worth mentioning in this respect is FPGA. This is where functional hardware flexibility is king and, in most

cases, where high price is paid in terms of power efficiency. The programmable fabric and layout are such that they require massive data movement across the chip, which is very costly in terms of energy.

Domain-Specific Dataflow Is a Way Around the Wall

When it looks like everything that can be done has already been done or at least considered, our next step needs to be beyond the basic premises of what we have done so far: the next natural evolution is a “sacrifice” of flexibility.

Focus and specialization create relative advantages and efficiencies, as in economics and business, so in computation. This “sacrifice” is worth it, in most cases, when the application needed is specific and its use is frequent. The high performance and power efficiency that come with specialization hold major rewards. Generally speaking, the more specific you can be with your application, the more customized, and thus more efficient, the hardware will be.

The concept of Domain-Specific Architecture (DSA) is to map the target application to the hardware without the general-purpose ISA. One such mapping is known as Dataflow architecture (4) like mapping where the application functions are mapped directly onto the hardware.

In DSA, the main question should be about the required range of the applications. For instance, if we are to build a DSA for a specific Machine Learning application, we should still keep the flexibility to use the hardware for a range of Machine Learning applications. This flexibility calls for software (e.g. deep learning compiler) and deep learning hardware co-design, a good example of which is the Hailo design path.

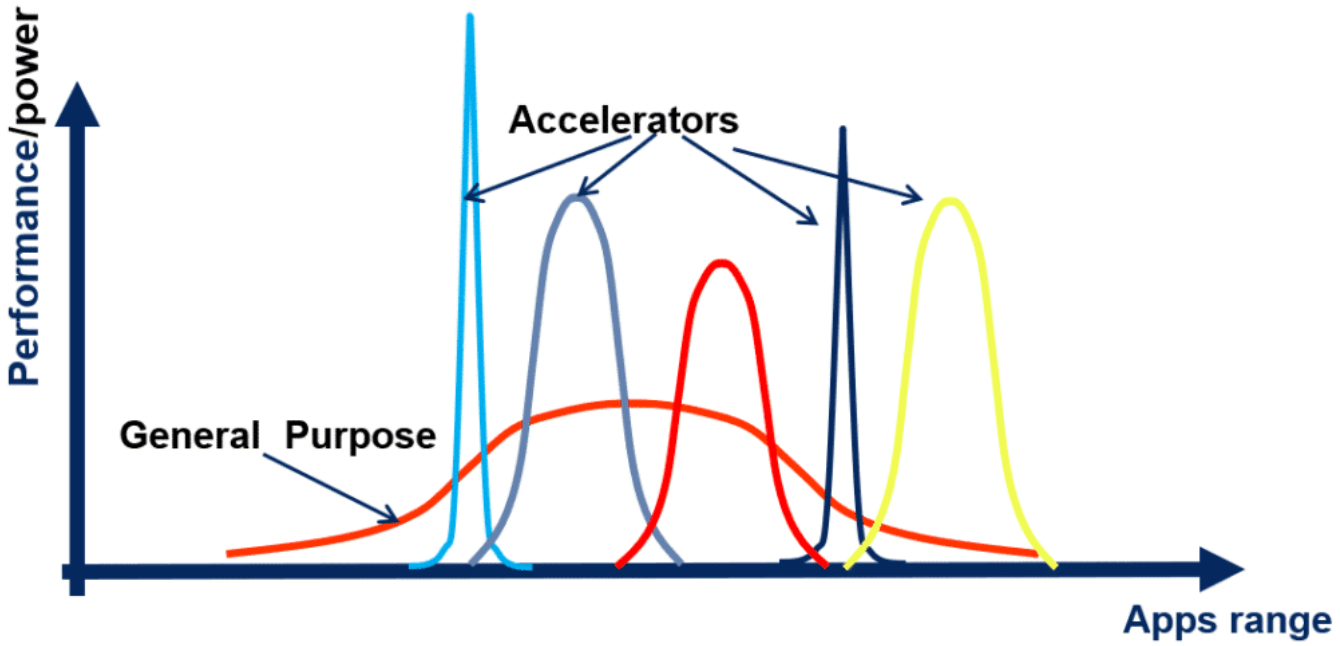

Figure 3: The tradeoff of processing efficiency and applications range flexibility – Domain-Specific architecture achieves better performance per unit of power but is designed for domain-specific workloads.

Domain-Specific and Deep Learning

Deep Learning neural networks (DNNs) require highly repetitive computing operations based on a relatively simple building block – the MAC (Multiple and Accumulate) operation. This Dataflow architecture with simple repetitive operations is highly power efficient compared to General Purpose architectures since there is no need for instructions and the compute engine is highly simplified (i.e. MAC). Other than the operation itself, the major piece in Domain-Specific Dataflow is data access, i.e. access to memory.

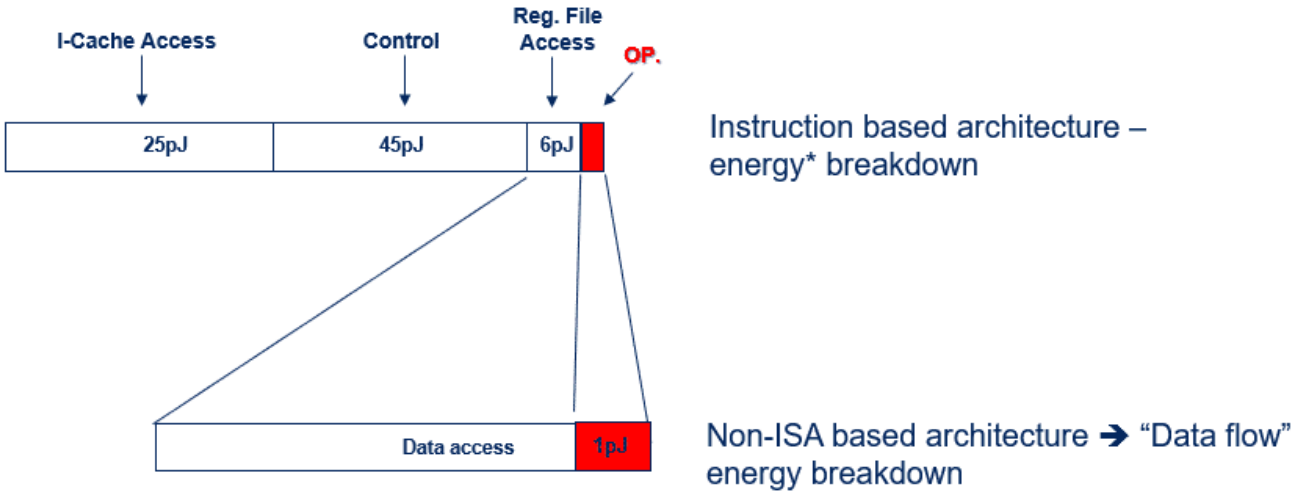

Figure 4: Dataflow consumes a fraction of the energy consumed by the instruction (energy breakdown presented in Figure 2). It expends additional energy only on data access, resulting it a much smaller per-instruction overhead.

Domain-Specific Dataflow architectures are the next stage of compute. The larger our data processing needs get (and recent history shows that this is the trend), the more previously neglected power efficiency becomes an issue and guides us away from the flexible but power-hungry and inefficient ISA. In a world of Big Data, where Deep Learning has emerged as the Killer App (5), Domain-Specific Dataflow architectures have the energy efficiency advantage. As such, they are a major enabler of mass adoption of AI technologies across domains and use cases.

To learn more watch Professor Uri Weiser and Hailo co-founder and CTO Avi Baum in this fascinating webinar: A Computer Architecture Renaissance: Energy-efficient Deep Learning Processors for Machine Vision

Sources & Notes:

1. Kozyrakis, Christos & Kansal, Aman & Sankar, Sriram & Vaid, Kushagra. (2010). Server Engineering Insights for Large-Scale Online Services. Micro, IEEE. 30. 8 – 19. 10.1109/MM.2010.73. https://www.researchgate.net/figure/Scaling-trends-for-the-transistor-count-clock-frequency-number-of-cores-and_fig2_224168227 ; new Plot and Data Collected for 2010-2017 by K. Rupp.

2. M. Horowitz, “1.1 Computing’s energy problem (and what we can do about it),” 2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014, pp. 10-14, doi: 10.1109/ISSCC.2014.6757323

3. Simone Bianco, Remi Cadene, Luigi Celona, Paolo Napoletano. (2018). Benchmark Analysis of Representative Deep Neural Network Architectures. https://arxiv.org/abs/1810.00736

4. Veen, Arthur H. (December 1986). “Dataflow Machine Architecture”. ACM Computing Surveys. 18 (4): 365–396. doi:10.1145/27633.28055

5. “Killer App” is an application family that cannot run on current implementation but can run on the new implementation. For instance, NN training of high-resolution recognition is almost impossible to run on GP-CPU but is possible on dedicated Dataflow architecture.