ONNX-to-Hardware Design Flow for the Generation of Adaptive Neural-Network Accelerators on FPGAs

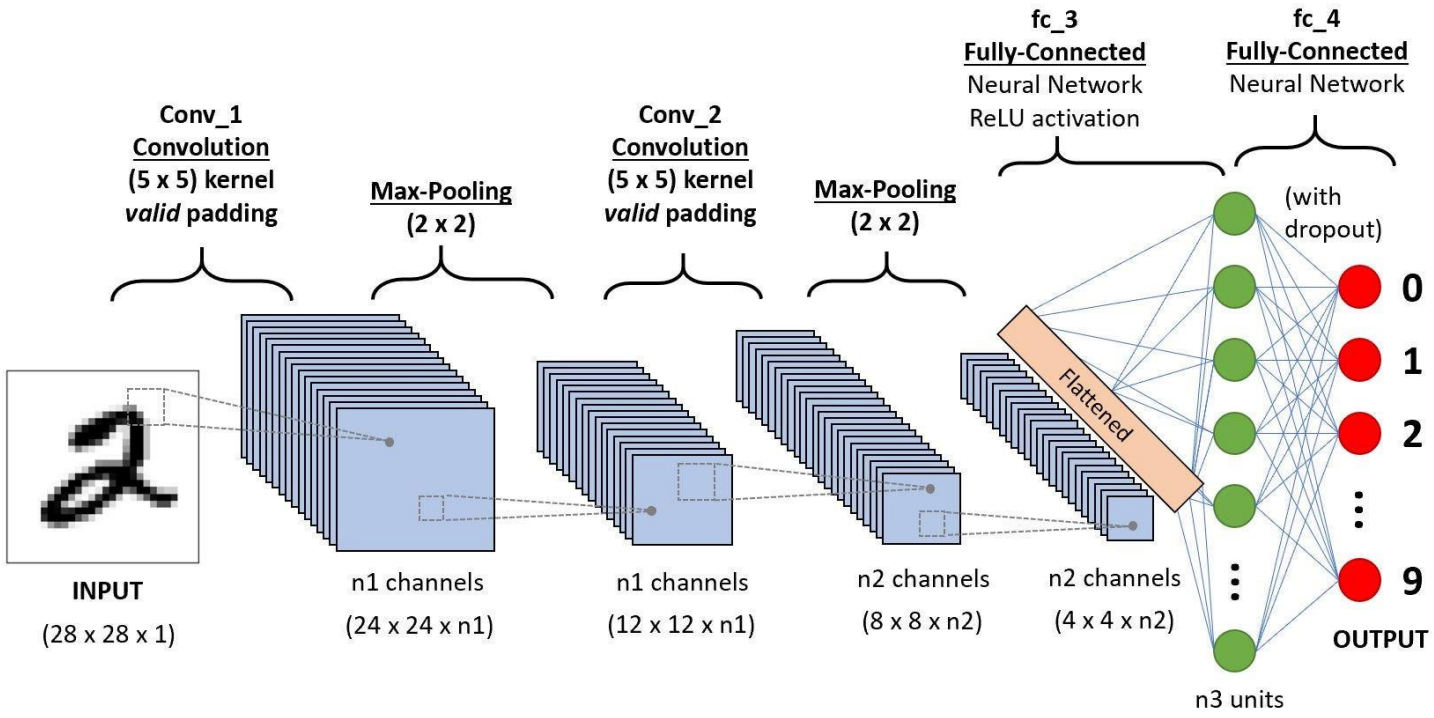
Federico Manca[2], Francesco Ratto[1]

[1]University of Cagliari (IT),
[2]University of Sassari (IT)

UNICA
UNIVERSITÀ DEGLI STUDI DI CAGLIARI

uniss
UNIVERSITÀ DEGLI STUDI DI SASSARI

Decisions at the edge



- Accelerating **Neural Networks (NN)** at the edge offer the possibility of obtaining low-latency and low-energy execution.

Accelerators

- **Specialized frameworks** with dedicated architectures offer a good solution for implementing **hardware accelerators**
- **FPGAs** offer an optimal platform to implement this kind of device

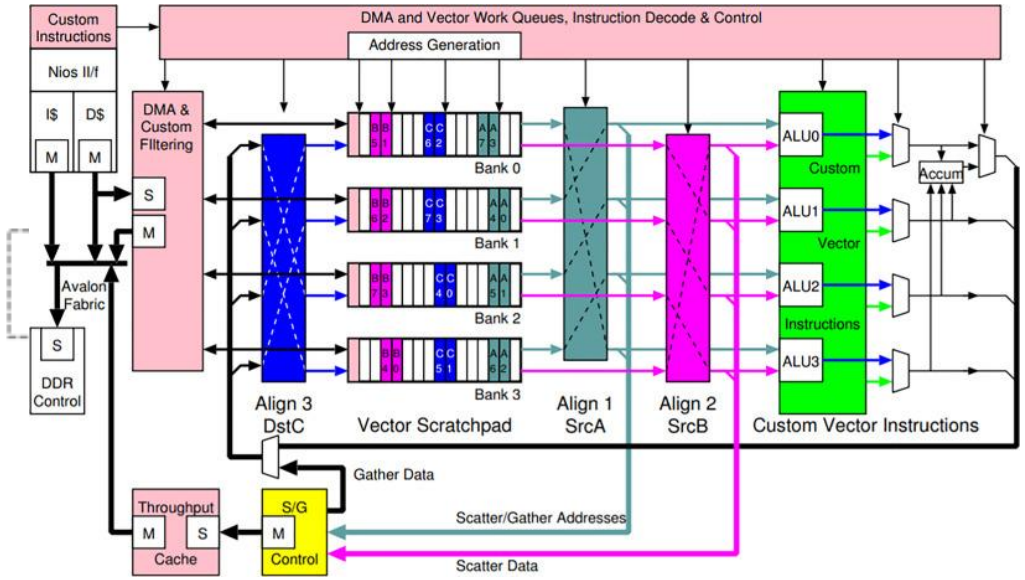
Approximate Computing

- **Approximate Computing (AC)** exploits the inherent resiliency of NN to reduce computational complexity and memory occupation, producing an acceptable error on the output.
- The technique used for reducing bit data precision is called **quantization**

Architectures

There are different types of architectures to implement a NN at the edge:

- **Streaming**: every layer has its own hardware block, enhancing parallelism;
- **Single Unit Processing**: all the computation is carried out on a single engine, favouring flexibility;
- **Vector Processor Unit**: specialized processor makes use of specialized instructions.



FINN and HLS4ML

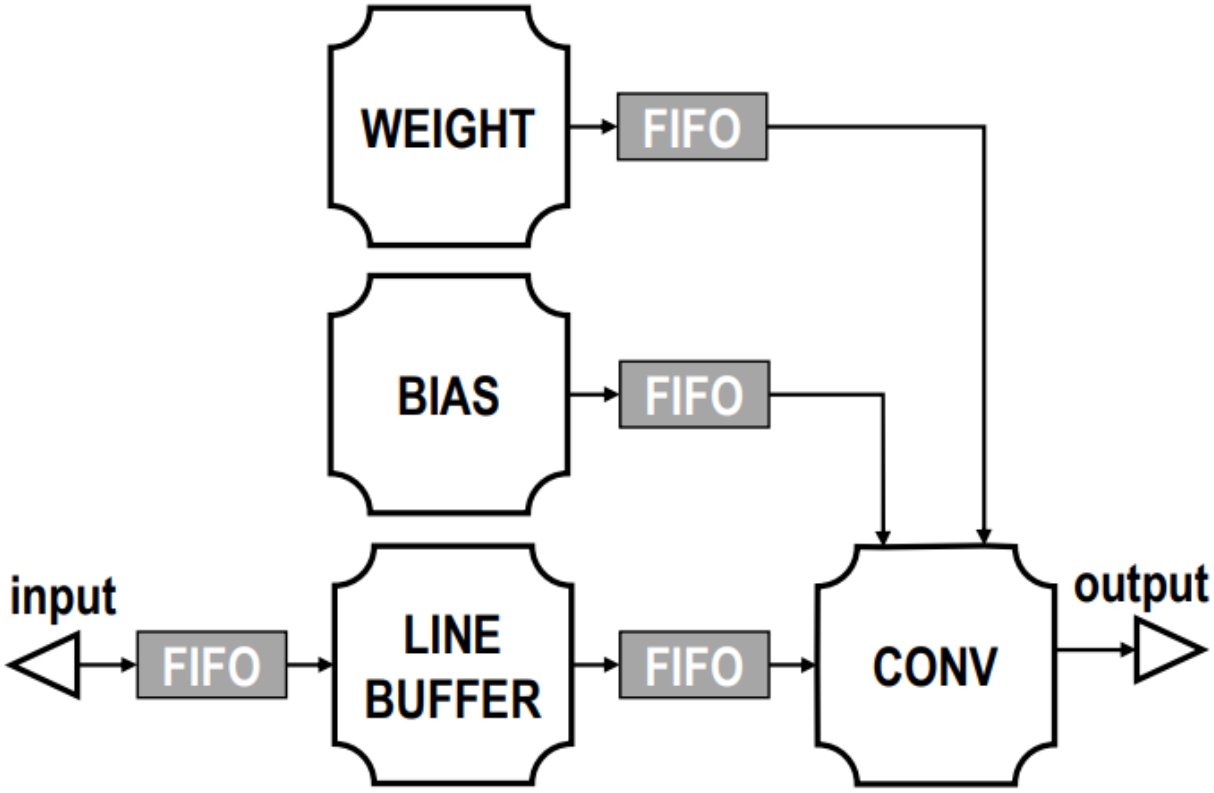
- The **dataflow** model is naturally suitable to express concurrency and parallelism
- In our previous work[1] it was chosen to use a **streaming architecture,** derivable from the dataflow model, which makes it possible to reach **high throughput** and **low power** consumption by using only **on-chip memories**.
- Other frameworks made the same choice: **FINN**, an experimental framework from AMD Research Labs based on Theano; **HLS4ML**, an open-source software designed to facilitate the deployment of machine learning models on FPGAs. Both frameworks exploit the capabilities of HLS tools.

The tools

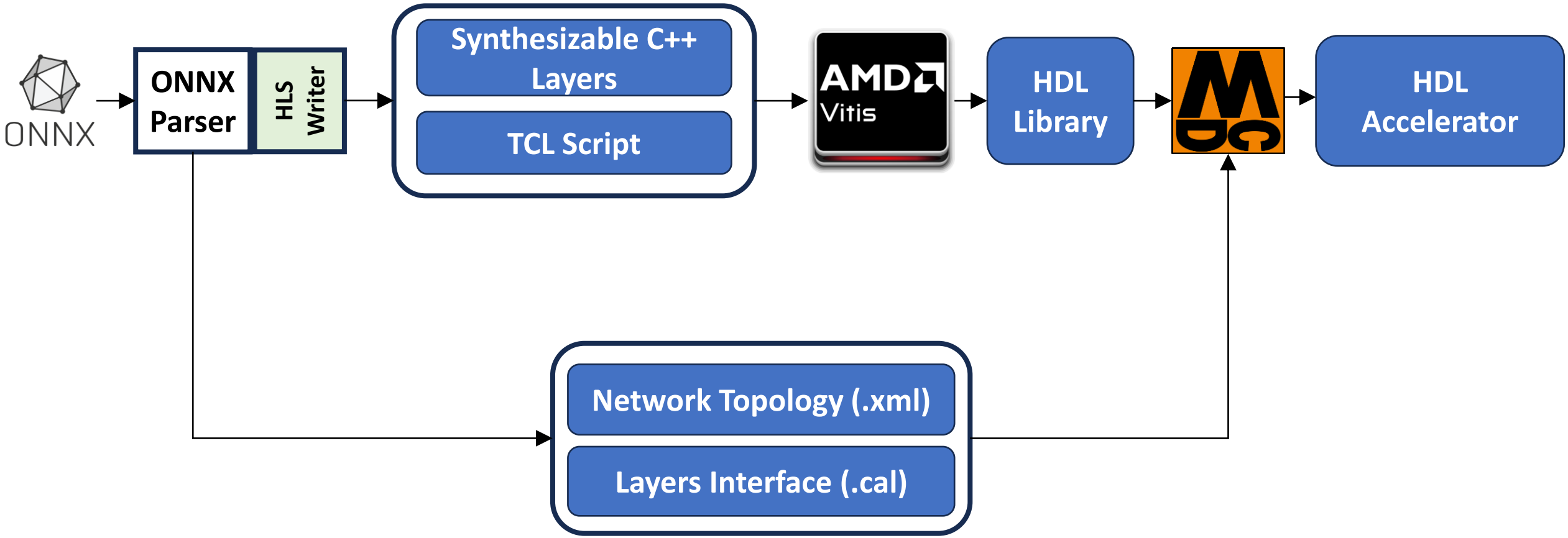
- **ONNXParser**, a Python application intended to parse the ONNX models and automatically create the **code for a target device**.
- **Vitis HLS**, which synthesizes a **C or C++** function into **RTL code**
- **Multi-Dataflow-Composer**, an **open-source tool** that can offer optional Coarse-Grained **reconfigurability** support for hardware acceleration.

Template architecture

- The C++ description of the layers is based on **ad-hoc architecture** template. For the CONV layer, core of the CNN, three actors are used: Line Buffer, Weight, Bias, and Conv actors.

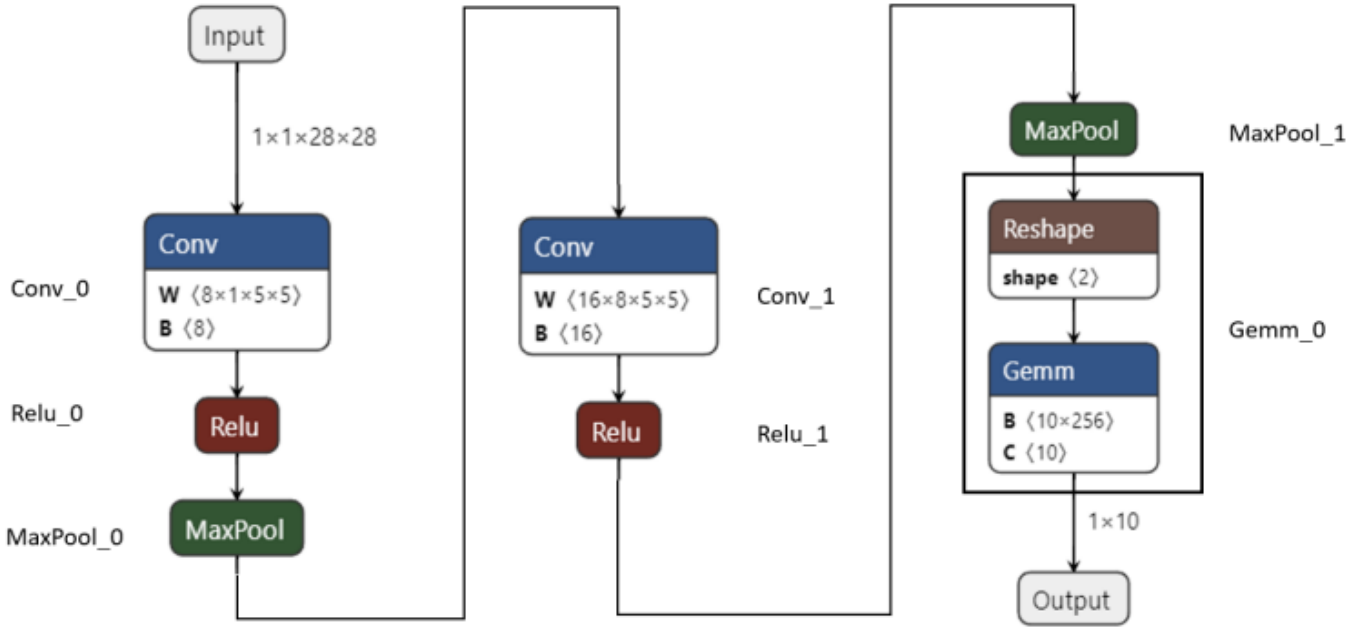


The toolflow



Preliminary assessment

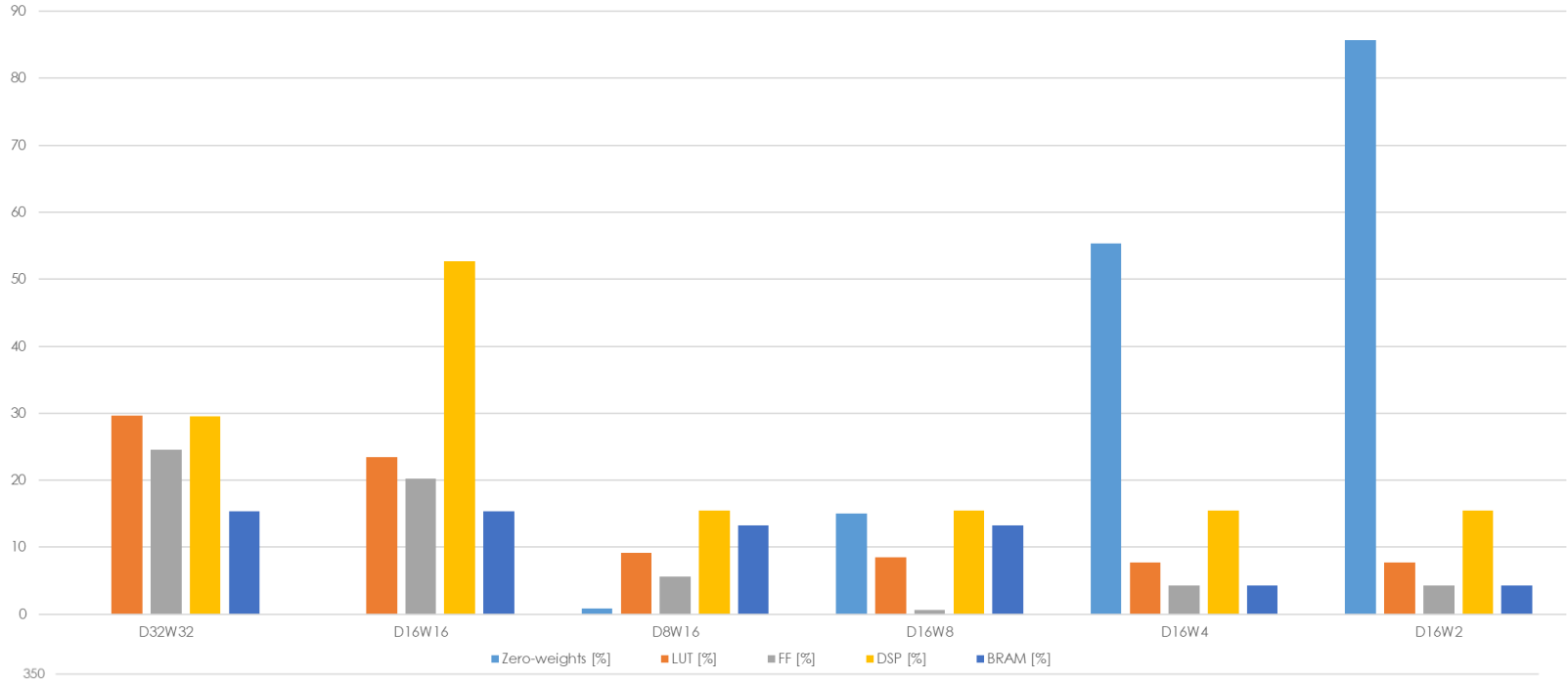
- It was carried out an exploration with **mixed precision** to test the toolflow functionality and **AC** capability on an accelerator made of **2 convolutional layers** followed by **1 fully connected layer**. The accelerator classified samples from the **MNIST** dataset.

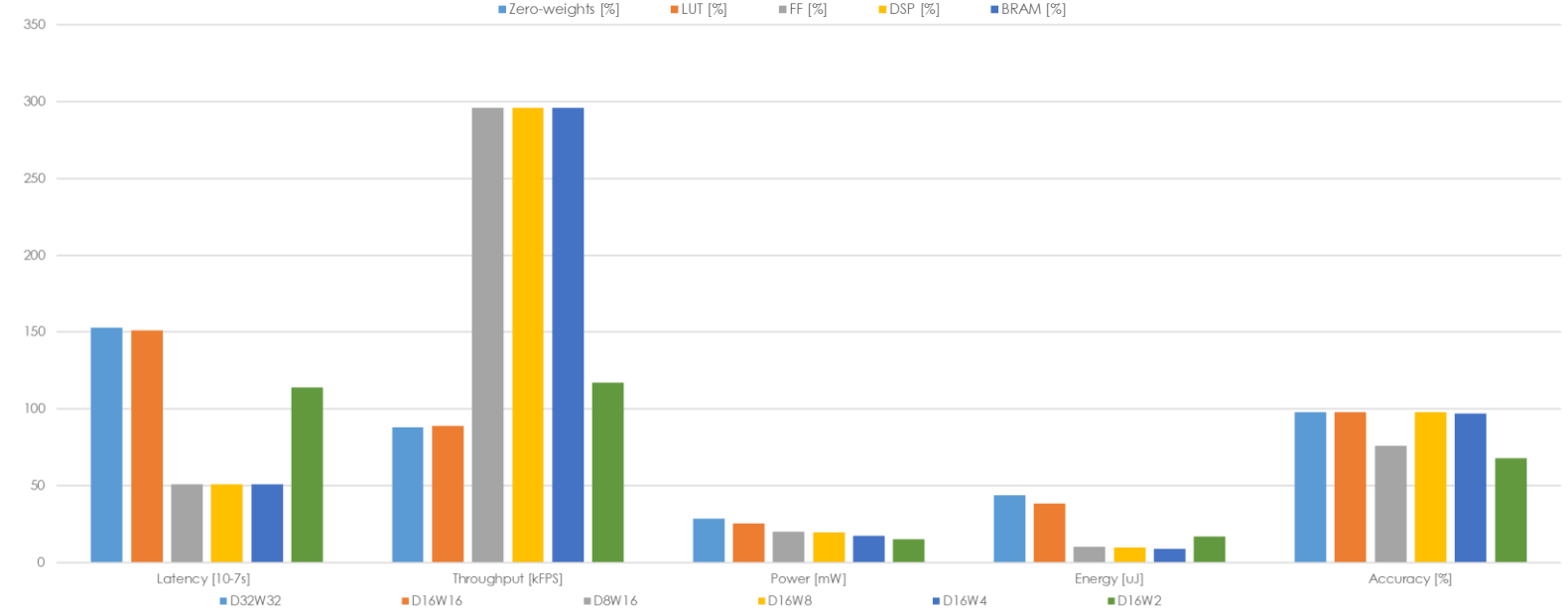


The results

Dx-Wy denotes that ***x*** *bits are used to represent* ***activations*** *and* ***y*** *bits are used to represent* ***parameters*** *in fixed-point precision.*

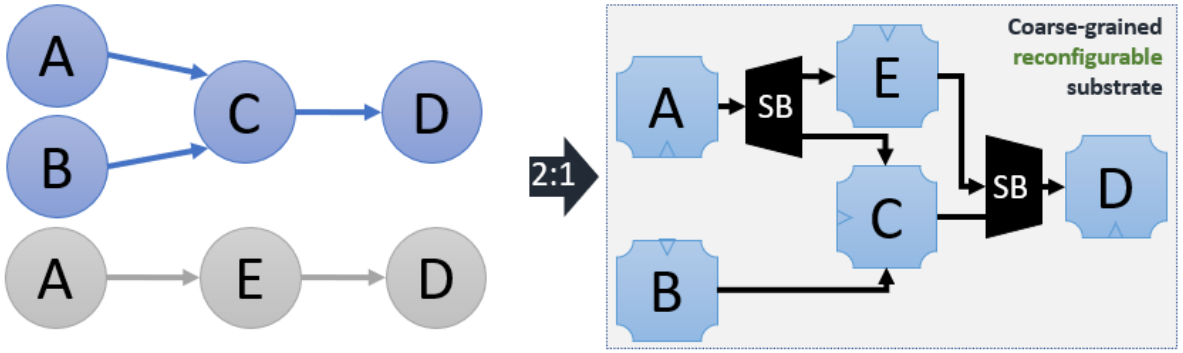
Reduced ***parameter precision*** *doesn't affect* ***accuracy*** *while reducing* ***memory footprint*** *(BRAM column) and obtaining a high percentage of* ***zero weights****.*





Future Work

- With this preliminary assessment some tradeoffs were tested for the accelerator
- The ultimate goal is to work around those tradeoffs and obtain runtime adaptivity using the power of the MDC tool to merge different dataflows.



THE END

[1] Ratto, Francesco, et al. "An Automated Design Flow for Adaptive Neural Network Hardware Accelerators." Journal of Signal Processing Systems (2023): 1-23.