www.bsc.es

A TASTE of Binary Neural Network Inference for On-Board FPGAs

Jannis Wolf, Leonidas Kosmidis

UNIVERSITAT POLITÈCNICA DE CATALUNYA

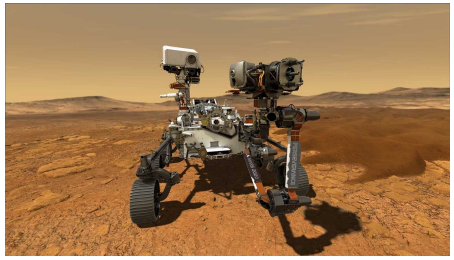
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

MBSE 2021

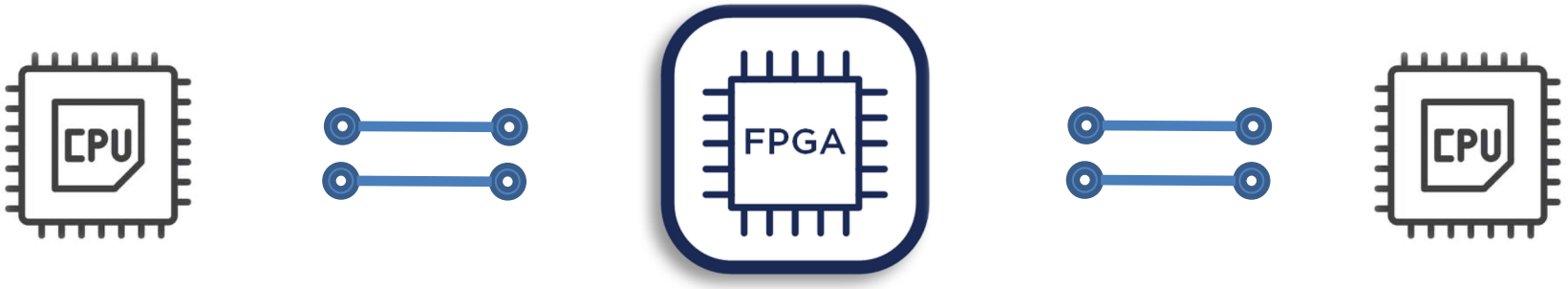
Introduction and Motivation

- Increasing interest in artificial intelligence (AI) and machine learning (ML) in space missions: e.g. Mars Perseverance, Φ-Sat-1, OPS-SAT…
- Existing space processors cannot keep up with their computational needs
- Use of COTS devices in institutional missions is challenging:
  - no radiation hardening → cannot be (safely) used beyond LEO
  - Non-space qualified software stacks, lack of RTOS support
- We present an open source hardware design to increase AI processing capabilities in space:
  - Low-cost Binarized Neural Network (BNN) accelerator based on TASTE


FPGA Binary Neural Network (BNN) Accelerator

Combination of CPU, TASTE framework and a BNN FPGA Accelerator



1. CPU loads feature vector

2. ESA's Model-Based TASTE framework handles the communication to accelerator

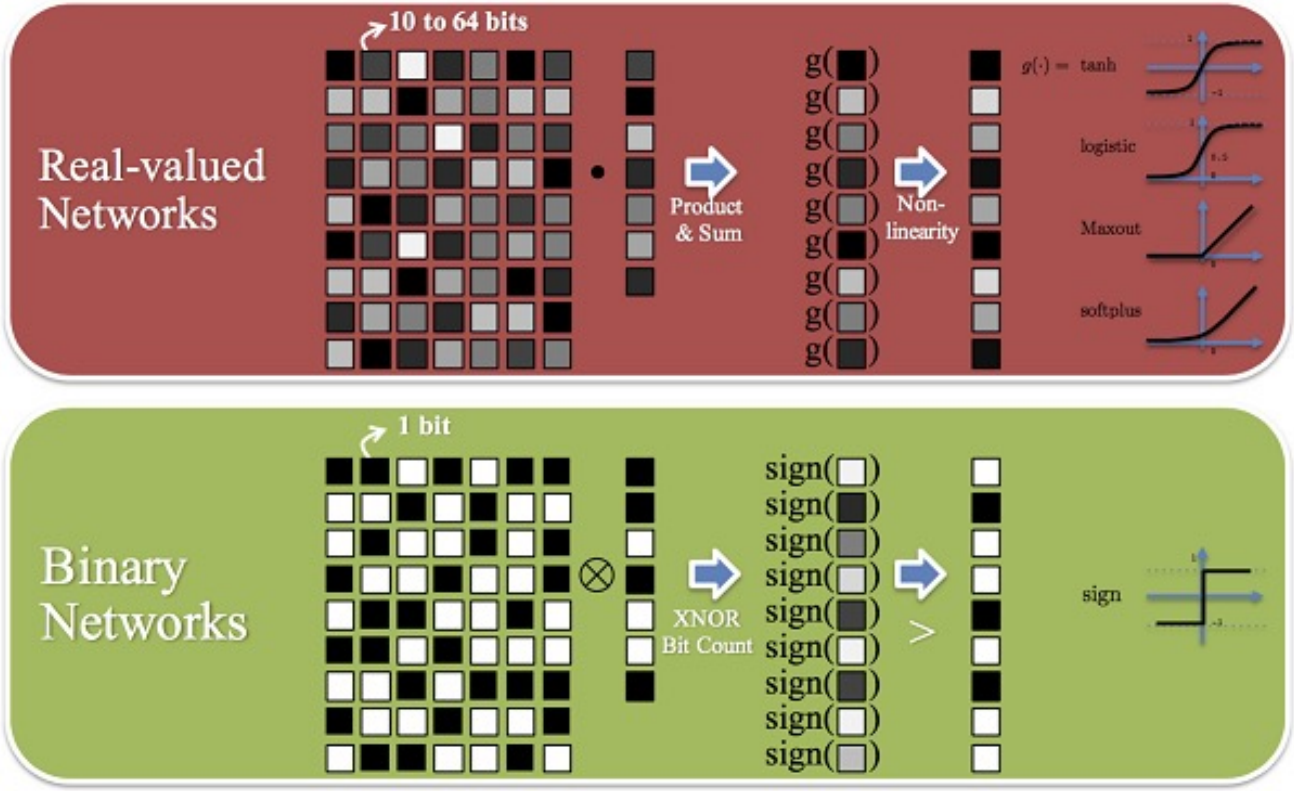
3. Custom-designed BNN Accelerator on the FPGA performs inference step

4. CPU receives prediction result

Project Properties

- **Operation principle**
  - Inference off-loading to the FPGA BNN accelerator
- **Reconfigurable design through the FPGA**
  - Flexible adaption to neural network parameters
  - Scalable parallelism
- **Reliable and Open Source from the ground up:**
  - TASTE correct-by-construction communication: software driver and hardware communication mechanism generation
  - Hand-written VHDL open source code for the accelerator assisted by a Python framework for training and model-in-the-loop verification

Binarized Neural Networks



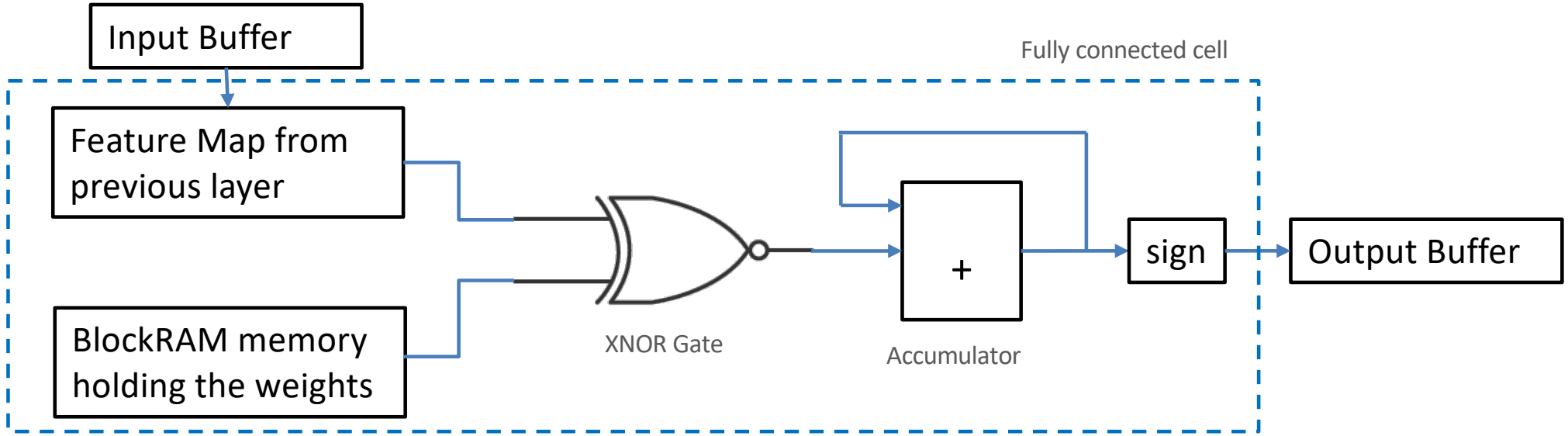
Binarization

- MAC operation is simplified to XNOR and set bit count operation
- Reduces memory usage up to 1/32
- Only marginal performance loss shown in scientific literature

Source: https://www.codeproject.com/Articles/1185278/Accelerating-Neural-Networks-with-Binary-Arithmetic

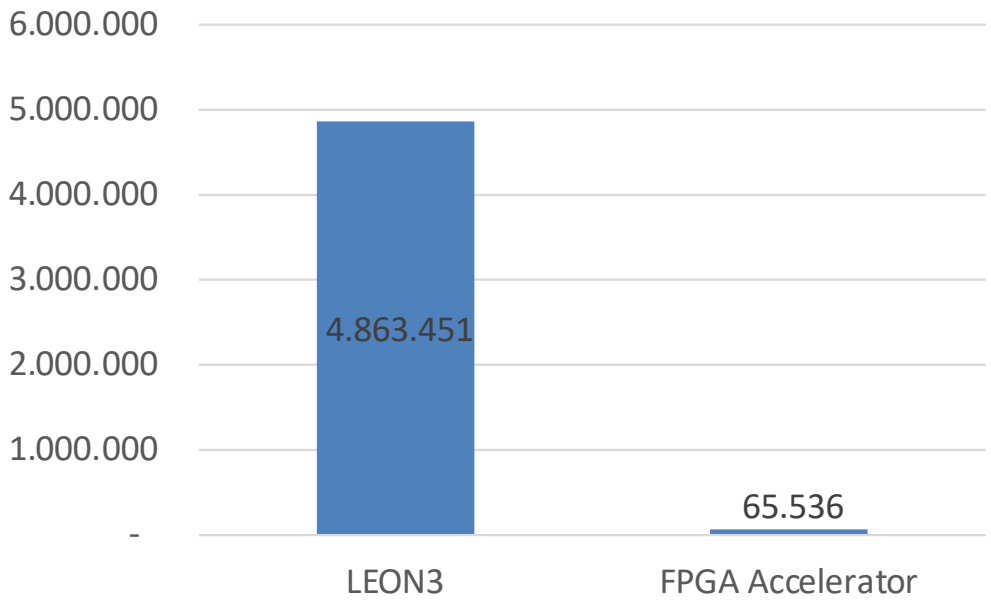
FPGA Binary Neural Network Accelerator

Basic principle: Fully connected layer cells attached through buffers



Preliminary Evaluation on Simulation

Clock cycles needed for one MNIST pass through fully connected layer with size (512, 512) on a LEON3 and on the BNN accelerator



Speed Up of about 74x. But:

- Communication overhead is not considered
- LEON3 simulation only with TSIM

→ Speed up expected to be smaller in reality

Why is this very fast?

- **Parallelization inside layer**
  - Parallel fully connected cells are only limited by available number of BRAM
- **Pipelining over layers**
  - Instead of sequential calculation on the CPU, the first layer can start with the next feature vector after completing the previous one
- **Low memory usage**
  - Effective load and store of weights

Main Components

- **The Project consists of three main components to enable MBD**

Binary PyTorch

- Python Library extending PyTorch for Binary Training functionality
- Export function of the weights to BRAM compatible files

VHDL BNN Components

- Library of Accelerator subcomponents written in VHDL with generic sizes

TASTE

- Code skeleton generation for the communication of CPU and Accelerator
- Build System for Deployment on Target Device

Workflow: Model Training

- **Training the Neural Network using Binary PyTorch**
- **Allows to experiment with various BNN models for early design space exploration**
- **Test in a model-in the loop fashion**
- **Resulting model weights can then be exported for the next steps**

Binary PyTorch Classes

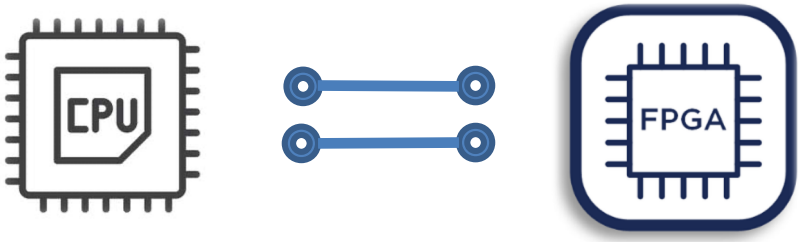
- BinaryFullyConnected()
  - export_weights()
- BinaryOptimizer()

Workflow: TASTE Code Generation

- **Communication Code Skeleton Generation**
- **Specify the data format sent and received between CPU and FPGA in ASN.1 notation**
- **TASTE returns Code skeleton in C and VHDL**

```
SRC-DATAVIEW DEFINITIONS ::=
BEGIN

  My-Integer3bit ::= INTEGER (0 .. 7)
  My-Integer2bit ::= INTEGER (0 .. 3)

  IRIS    ::= SEQUENCE {
      input-data  My-Integer3bit,
      output-data My-Integer2bit,
      validity    ENUMERATED { valid, invalid }
  }

END
```

ASN.1 communication description



C Code skeleton to send and receive data

VHDL skeleton including the Input and Output architecture declaration

TASTE‘s code generation

Workflow: Software Side

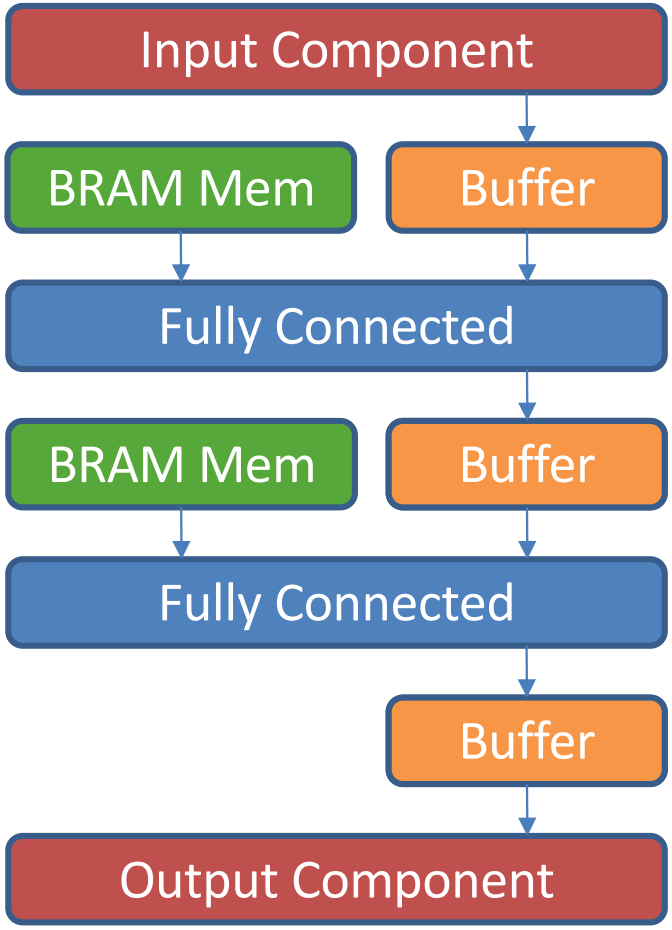
- Implement C functions
- C Code skeleton is used to implement the sending and receiving of data on the software side

```c
/* User code: This file will not be overwritten by TASTE. */

#include "speak_to_hw.h"

void speak_to_hw_startup()
{
    /* Write your initialization code here,
       but do not make any call to a required interface. */
}

void speak_to_hw_PI_pulse()
{
    /* Write your code here! */
}

```

C code skeleton generated by TASTE

Workflow: Hardware Side

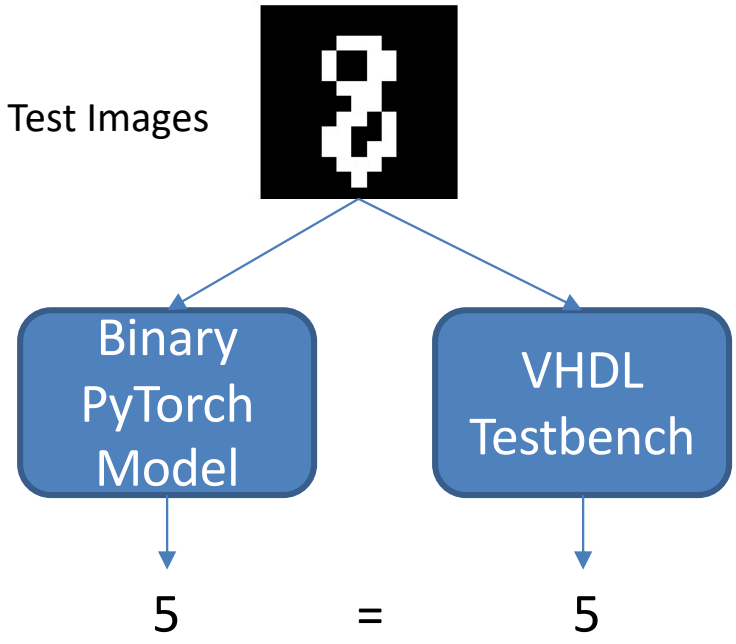
- Implementation of the Accelerator into the VHDL skeleton
- Has to be done manually by the designer (space for automation in the future)
- Facilitated through the set of VHDL BNN components that resemble the PyTorch classes



Scheme of the accelerator components

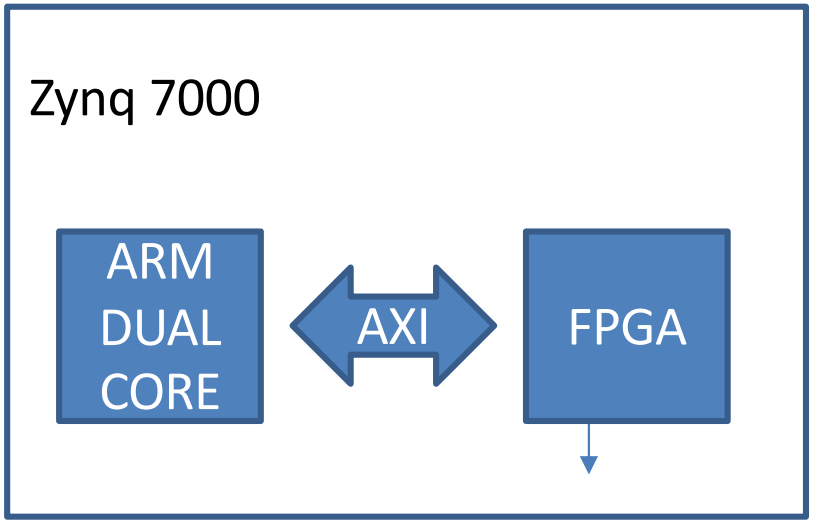
Workflow: Verification

- Verification of the inference results in simulation
- Writing testbenches and check against the PyTorch results
- Finding errors early before costly deployment
- Although not completely automated, this allows software-in the loop testing



Workflow: Deployment with TASTE's Build System

- **Deployment on target platform**
- **TASTE features different deployment nodes (LEON3/RTEMS, x86/Linux)**
- **Hardware compatibility shown for Spartan3 and is being developed for Xilinx's Zynq based SoCs [1]**
- **Hardware-in-the-loop Verification concludes the design process**



Xilinx's Zynq SoC

[1]

Advantages of Model Based Design

- **Structure of complex project through code generation**
  - TASTE correct-by-construction communication
  - Software and Hardware design moves closer together
- **Reusability of VHDL Code for custom BNN Accelerator generation**
  - Flexible adaption to neural network parameters
- **Integration into different architectures through TASTE**

Conclusions and Future Work

- **FPGA BNN neural network accelerator achieving speedups of 74x compared to a baseline LEON3 processor (tsim)**
  - Move from simulation to deployment on a FPGA
  - Evaluation with space-relevant ML benchmarks: OBPMark and MLAB presented at OBDP 2021

- **How Model Based Design supports the design process of custom accelerator**
  - Automatic code generation integrated with TASTE and PyTorch

References and Acknowledgements

- The project's source code can be found on GitHub:
  - BNN Accelerator: www.github.com/JannisWolf/fpga_bnn_accelerator

- This work is partially supported by:
  - the Xilinx University Program (XUP) and XUP Board Partner Red Pitaya
  - ESA under the GPU4S (GPU for Space) project (ITT AO/1-9010/17/NL/AF)
  - European Commission's Horizon 2020 programme under the UP2DATE project (grant agreement 871465)
  - the Spanish Ministry of Economy and Competitiveness (MINECO) under grants PID2019-107255GB and FJCI-2017-34095
  - the HiPEAC Network of Excellence

www.bsc.es

A TASTE of Binary Neural Network Inference for On-Board FPGAs

Jannis Wolf, Leonidas Kosmidis

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

MBSE 2021