Reliable Machine Learning Acceleration for Future Space Processors and FPGAs: LEON, NOEL-V and TASTE

Marc Solé Bonet, Jannis Wolf, Leonidas Kosmidis

UNIVERSITAT POLITÈCNICA DE CATALUNYA

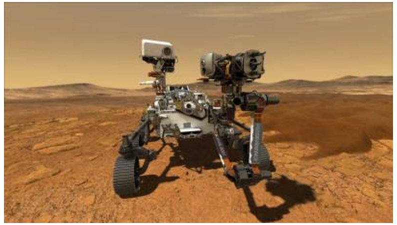
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

OBDP 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 two open source hardware designs to increase AI processing capabilities in space:
  - Low-cost short vector unit to increase AI performance in space CPUs
  - Low-cost Binarized Neural Network (BNN) accelerator based on TASTE



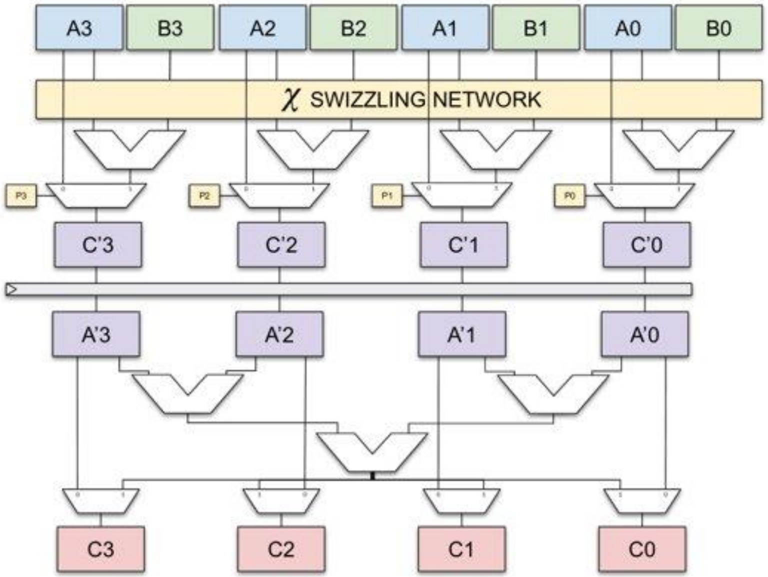
Low-Cost Short Vector Support for Space Processors

- Hardware module designed for Gaisler's LEON3 and NOEL-V processors
- Low-cost hardware unit to speed-up AI applications
  - Special instructions selected by analyzing the most common ML operations
  - 8-bit integer instructions are enough for ML [1]
- SIMD architecture such as ARM's NEON
  - But reduced hardware overhead by reusing the integer register file
  - SWAR (SIMD within a register) concept inspired by [2] but combining ideas from mobile GPUs [3] too
- Saturation option included for all instructions

[1] T. P. Jouppi et al. In-Datacenter Performance Analysis of a Tensor Processing Unit. ISCA, 2017
[2] Martin Danek. ESA IP Core Extensions for LEON2: daiFPU and SWAR. ESA TEC-ED &TEC-SW Final Presentation Days, May 2020.
[3] Trompouki, Towards General Purpose Computations on Low-end Mobile GPUs. DATE 2016.

SIMD: Architecture design

- Reusing the integer register file simplifies loading and managing the data in the registers
- Frequently found instructions in ML algorithms added: arithmetic, bitwise, min, max etc.
- Two pipeline stages:
  - Vector-Vector operations
  - Reduction operations
  - Bypasses when one is not used



SIMD: Architecture design

10	rd	001001	rs1	i	sd2	sd1	rs2

31 29 24 18 13 12 9 4 0

- Instructions encode the source and destination register and the operation code for each stage
  - Exploited unused opcodes in LEON3
  - Custom extensions for NOEL-V
- Additional embedded GPU inspired features are included:
  - Immediate instructions encode commonly used values (e.g. powers of 2)
  - Masking and Swizzling
  - Both configured using the special register %scr (SIMD control register)

SIMD: Programming

- Assembler support has been included for the new SIMD instructions
- The tests have been written in C using inline assembly
  - Currently working on compiler support

```
unsigned char weights[32*32];
unsigned char next_layer[32*32];

/* Allocate short vectors to specific registers*/
register unsigned int a asm("%g4");
register unsigned int b asm("%g5");
register unsigned int result asm("%g6");

/* initialise all a components to 0, ie a.xyzw=0 */
a = 0;
/* b.xyzw = weights[0].xyzw */
b = *((unsigned int*) &weights[0]);
/* result.xyzw = a.xyzw + b.xyzw */
asm("add %g4, %g5, %g6");
/* next_layer[0].xyzw = result.xyzw */
*((unsigned int*) &next_layer[0])=result;
```

Preliminary Evaluation: Hardware Overhead

We have implemented our design using Xilinx Vivado targeting Artix 7

- Baseline LEON3-MIN@100MHz

Resource	SIMD Cost Absolute Value	% of increase w.r.t. baseline LEON3-MIN
LUTs	1869	25%
FF	168	5.9%

- Only a fraction of the hardware cost of conventional vector implementations for embedded processors (25% vs 2X) [1]
- The integration of the SIMD reduced the core frequency to 72MHz
  - Currently working on design optimisation to achieve the original frequency

[1] M. Johns et al. A Minimal RISC-V Vector Processor for Embedded Systems. FDL 2020

Preliminary Evaluation: ML Performance

- Matrix multiplication speed-up compared to LEON3-MIN@100MHz
  - Essential building block in NN for fully connected and convolution layers

	Matrix size			
Data type	**4x4**	**8x8**	**16x16**	**32x32**
Int	2.45x	3.81x	3.08x	3.39x
Char	2.24x	3.59x	2.96x	3.31x

- Complex inference application Cifar-10 from [1][2] speed-up:

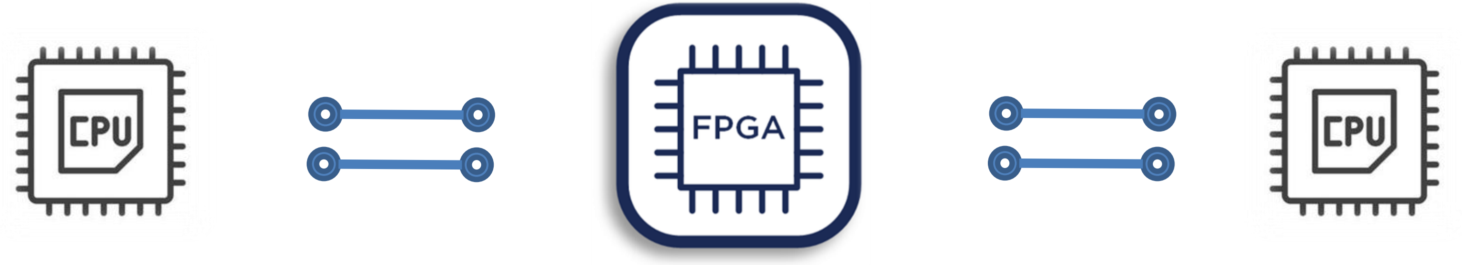
Cifar-10	4.13x

  - Performance improvements despite frequency reduction

[1] GPU4S Bench: Design and Implementation of an Open GPU Benchmarking Suite for Space On-board Processing. https://www.ac.upc.edu/app/research-reports/public/html/research_center_index-CAP-2019,en.html
[2] GPU4S (GPUs for Space): Are we there yet? & OBPMark (On-Board Processing Benchmarks) – Open Source Computational Performance Benchmarks for Space Applications, OBDP 2021

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

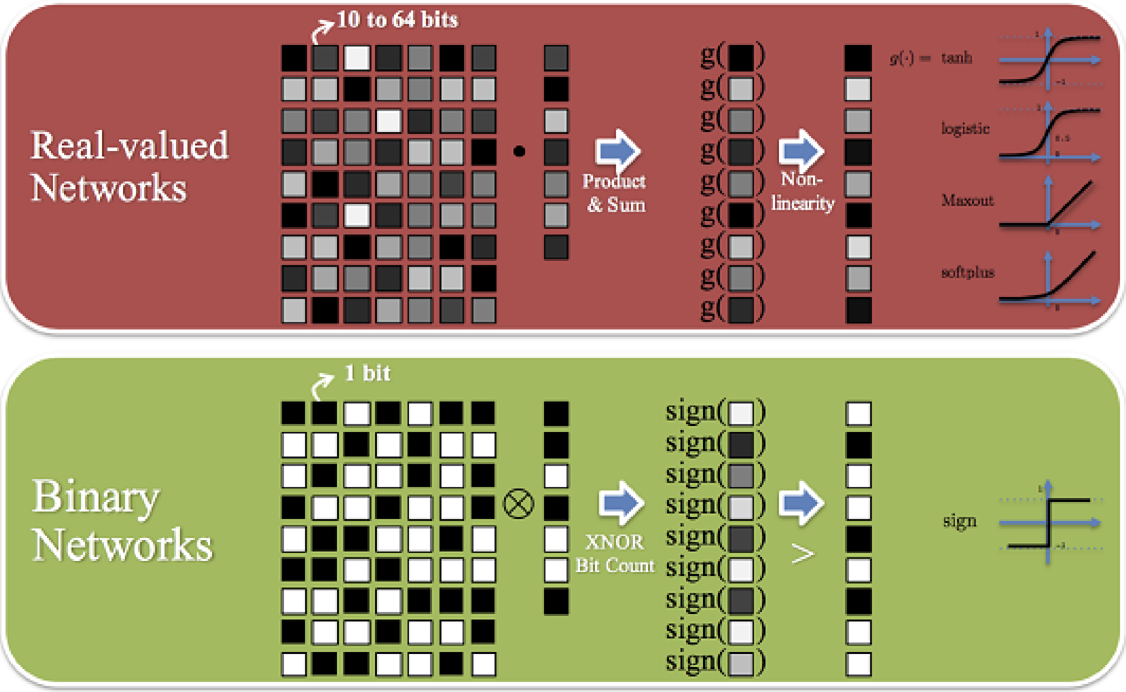
Project Properties

- **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

Binarized Neural Networks



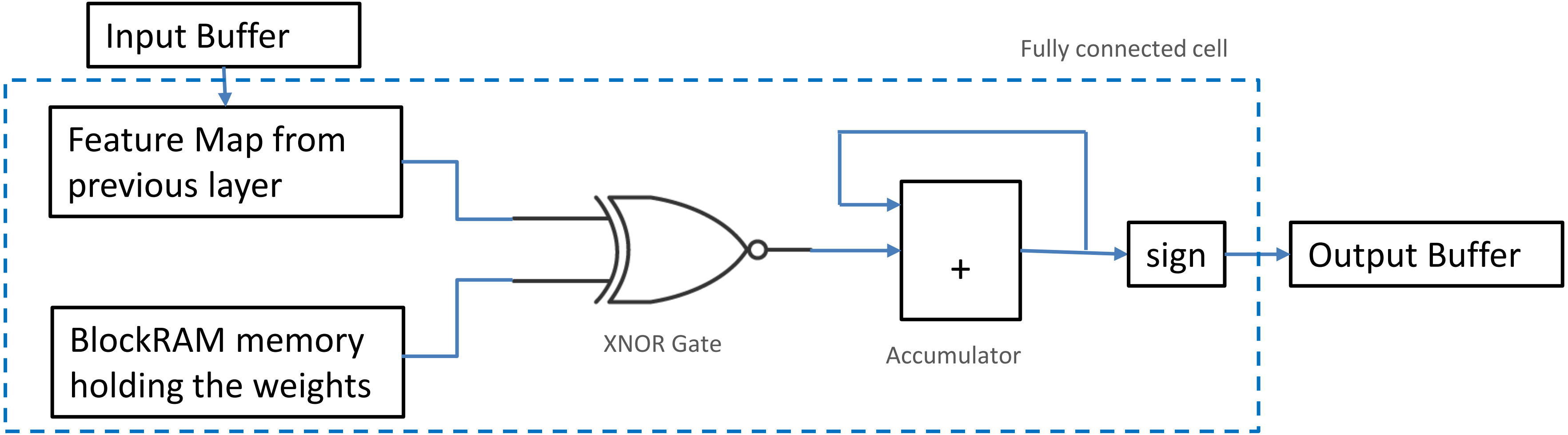
Binarization

- 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



Convolutional layer possible through parallel fully connected layers and reconnecting of the feature maps

Why is this very fast?

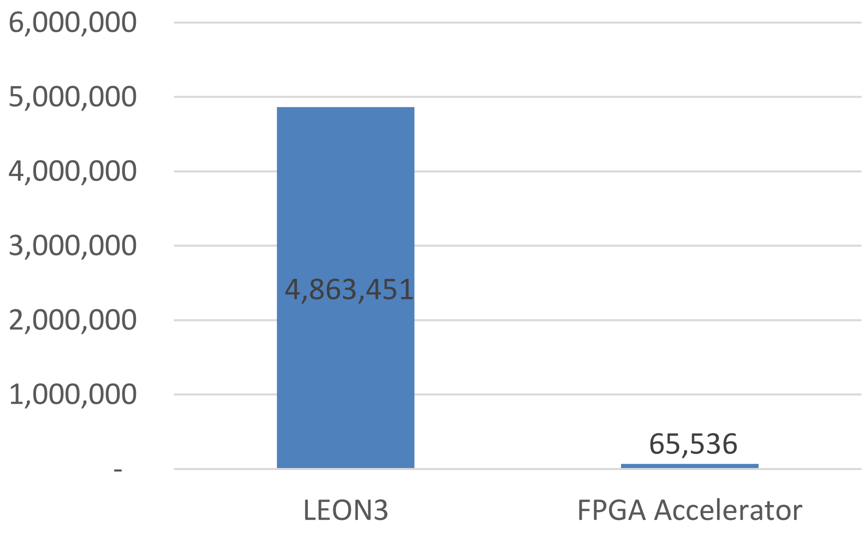
- **Parallelization inside layer**
  - Parallel bitwise execution only limited by loading weights from BRAM

- **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

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 accelerator



Speed Up of about 74x. But:

- FPGA operates at different frequency
- Communication overhead is not considered
- LEON3 simulation only with TSIM

→ Speed up expected to be smaller in reality

Future Work

- **From simulation to hardware**
  - LEON3 and BNN accelerator not connected yet
  - Simulation and verification was only performed separately

- **Python Code generator integrated with the TASTE framework**
  - Integrating binarized layer training into a big DL library like PyTorch
  - After training, get parameters and layer configurations
  - Optimize parallelization scheme and generate VHDL code for the accelerator in a complete model-based manner

Conclusions and Future Work

- **Two work-in-progress open source hardware designs for reliable machine learning acceleration of space on-board systems:**
  - A low-cost AI vector unit for LEON3 and NOEL-V, achieving speedups in matrix multiplication of up to 3.8x and 4.13x in a complex inference chain
    - Improve CPU frequency to match baseline design
    - Full compiler support
  - An FPGA BNN neural network accelerator achieving theoretical speedups of 74x compared to a baseline LEON3 processor
    - Move from simulation to FPGA
    - Automatic code generation integrated with TASTE and PyTorch
  - Both provide promising preliminary results
  - Evaluation with space-relevant ML benchmarks: OBPMark and MLAB presented at OBDP 2021

References and Acknowledgements

- Both projects participate in the Xilinx Open Hardware Design Competition 2021 (Europe):
  - Vector Unit: https://gitlab.bsc.es/msolebon/grlib-ai-extension
  - 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

Reliable Machine Learning Acceleration for Future Space Processors and FPGAs: LEON, NOEL-V and TASTE

Marc Solé Bonet, Jannis Wolf, Leonidas Kosmidis

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

OBDP 2021