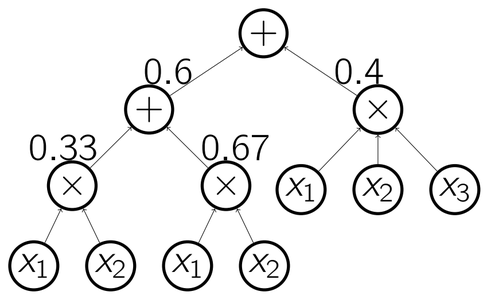
Resource-Efficient Logarithmic Number Scale Arithmetic for SPN Inference on FPGAs

Lukas Weber*, Lukas Sommer*, Julian Oppermann*, Alejandro Molina†, Kristian Kersting† and Andreas Koch*
*Embedded Systems and Applications Group, TU Darmstadt, Germany, {weber, sommer, oppermann, koch}@esa.tu-darmstadt.de
†Machine Learning Lab, TU Darmstadt, Germany, {molina, kersting}@cs.tu-darmstadt.de

TECHNISCHE UNIVERSITÄT DARMSTADT

Sum-Product-Networks

- Graphical Models used for representing **joint probabilities**
- Inferes **exact** probabilities in **linear** time w.r.t. their size
- Structure:
  - A tractable, univariate distribution is an SPN. **(Leaf-Node)**
  - A product of SPNs over different sets of variables is an SPN. **(Product-Node)**
  - A weighted sum of SPNs over the same set of variables is an SPN. **(Sum-Node)**



5pt

Fig. 1: Example of a valid SPN, capturing the joint probability distribution of the variables $\mathbf{x_1}$, $\mathbf{x_2}$ and $\mathbf{x_3}$.

Logarithmic Number Systems

- Number encoding based on logarithmic scale
- Generally: $A = (-1)^S \times 2^E$, plus additional flags for *zero, special cases*
- Inverted complexity compared to floats (addition is harder, multiplication is easier)
- Z_A and S_A also act as special-case indicators for $A = 0$ and $A = 1$ respectively.

Name	Zero	Sign	Exponent (fixed point)	
Abbr.	Z_A	S_A	E_A	
Bit-Width	1 bit	1 bit	k bits	m bits

Implementation

- Optimized towards probability values in the range $[\mathbf{0}, \mathbf{1}]$
- Addition uses **piecewise quadratic interpolation** for approximating $\mathbf{f(x) = \log_2(1 + 2^x)}$
- Parameterized for *interpolation error*, *integer-* and *fraction-*bitwidth
- Integrates with the framework by Sommer *et al.*
- Generates **fully spatial, fully pipelined** design based on textual description of SPNs

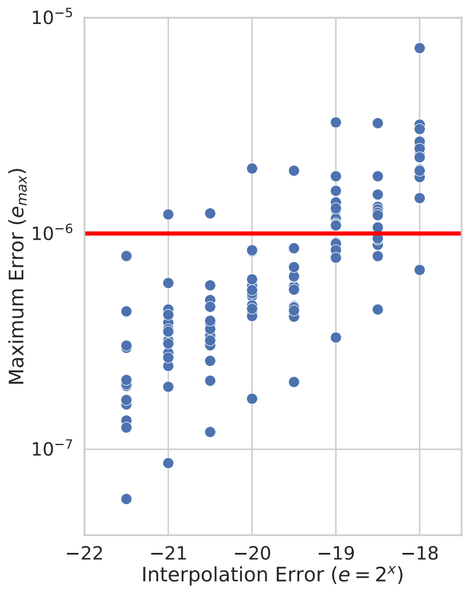


Fig. 2: Overall error in dependence of interpolation error.

Results - Utilization

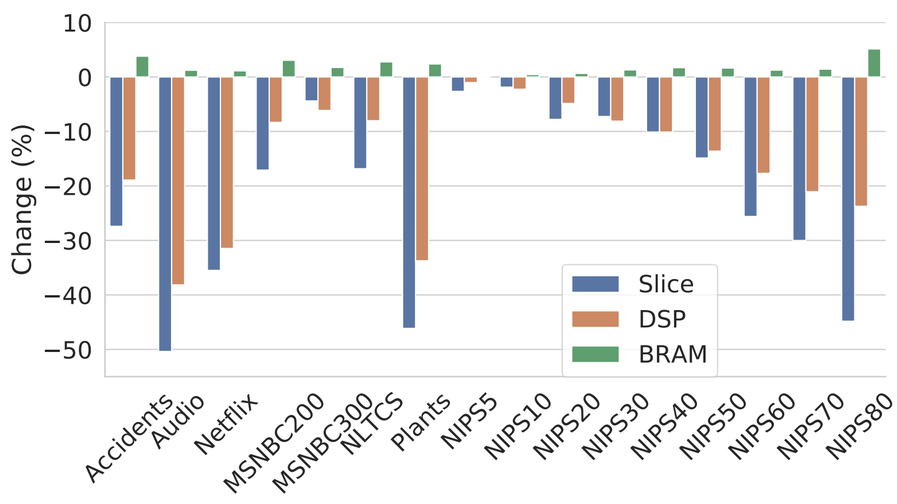


Fig. 3: Changes of percentual utilization of LNS- over FP-variant.

- Slice-utilization is **decreased by up to 50%** (Geo.-mean 14.6%)
- DSP-utilization is **decreased by up to 38%** (Geo.-mean 10.8%)
- BRAM requirement is increased, but not critical

Results - Throughput

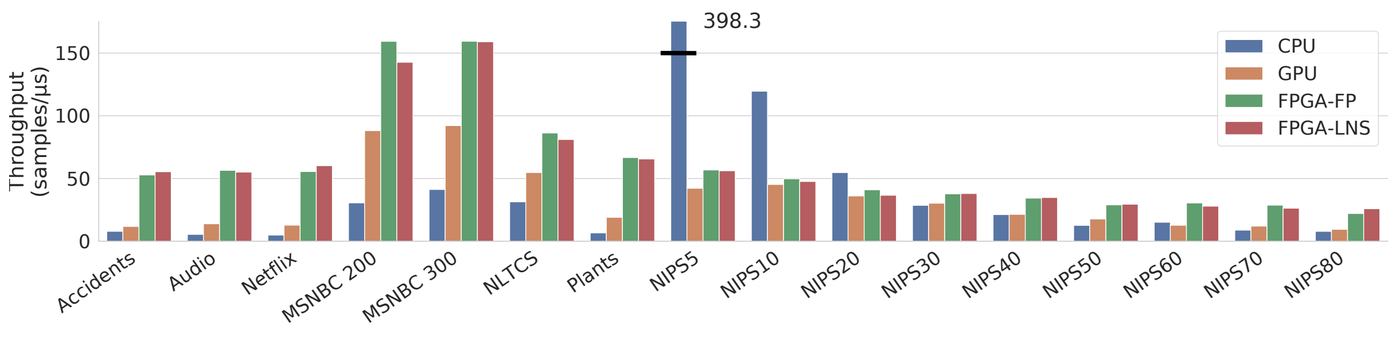


Fig. 4: Throughput including data-transport to and from the device.

- Throughput reduced by **1.1%** on average between LNS- and FP-implementations
- FPGA-implementations **outperform** CPU & GPU by **11.4x** (CPU) & **4.7x** (GPU)

Conclusion & Outlook

- Up to **4.7x** / **11.4x** faster than GPU & CPU
- Saves up to **50%** of slices, **38%** of DSPs
- Throughput is only reduced by **1.1%** on average (vs. FP-baseline)
- Reduced resource-requirement allows mapping of **bigger and more complex** SPNs
- Outlook: Extend toolflow towards other query types (e.g. marginalization) and training of SPNs

[1] L. Sommer, J. Oppermann, A. Molina, C. Binnig, K. Kersting, A. Koch, "Automated Mapping of the Sum-Product Network Inference Problem to FPGA-based Accelerators" in ICCD, 2018
[2] M. Haselman, M. Beauchamp, A. Wood, S. Hauck, K. Underwood, K. S. Hemmert, "A comparison of Floating Point and Logarithmic Number Systems for FPGAs" in FCCM, 2005

Embedded Systems & Applications