4.6 A 65nm CMOS 6.4-to-29.2pJ/FLOP@0.8V Shared Logarithmic Floating Point Unit for Acceleration of Nonlinear Function Kernels in a Tightly Coupled Processor Cluster

Michael Gautschi[1], Michael Schaffner[1], Frank K. Gürkaynak[1], Luca Benini[1,2]

[1]ETH Zurich, Zurich, Switzerland, [2]University of Bologna, Bologna, Italy

Energy efficient computing and ultra-low power operation are strong requirements for a vast number of application areas such as IoT and wearables. While for some applications integer and fixed-point processor instructions suffice, others (eg. SLAM, Stereo Vision, nonlinear regression and classification) require a larger dynamic range, typically obtained using single/double precision floating point (FP) instructions. Logarithmic Number Systems (LNS) have been proposed [1,2] as an energy-efficient alternative to conventional FP, as several complex operations such as MUL, DIV, and EXP translate into simpler arithmetic operations in the logarithmic space and can be very efficiently calculated using integer arithmetic units. However, ADD and SUB become nonlinear and have to be approximated by look-up tables (LUTs) and interpolation, which is typically implemented in a dedicated LNS unit (LNU) [1,2]. The area of LNUs grows exponentially with the desired precision, and an LNU with accuracy comparable to IEEE single precision format is larger than a traditional floating-point unit (FPU). However, we show that in multi-core systems optimized for ultra-low-power operation such as the PULP system [3], one LNU can be efficiently shared in a cluster as indicated in Fig. 1. This arrangement not only reduces the per-core area overhead, but more importantly allows several costly operations such as FP MUL/DIV to be processed without contention within the integer cores without additional overhead. We show that for typical nonlinear processing tasks, our LNU design can be up to 4.2x more energy efficient than a private-FP design.

For an accurate comparison, we have manufactured and measured two separate chips with a quad-core cluster system using the UMC65nm LL technology. Each chip contains an identical cluster with four 32b OpenRISC cores, a 16kB shared tightly coupled data memory (TCDM) and 1kB instruction caches. In the 1st chip, one LNU using an 8 bit integer, 23 bit fractional and a sign bit LNS format is shared by four cores, and in the 2nd chip all four cores have their own private FPU. A third shared FPU design has also been evaluated, but not included in this comparison as it was much slower (up to 46%) due to contentions (up to 96%). LNU and FPUs have been tightly integrated into the integer ALUs of the processors, as illustrated in Fig. 1 and can be accessed using standard OpenRISC FP instructions. The FPU used for comparisons in this work is IEEE single-precision compliant and supports FP ADD/SUB/MULT and casts and is based on a multiply-add architecture with a shared normalizer. The measured FPU energy efficiency @0.8V of 15.3-18.4 pJ/op is compatible with other state-of-the art designs [5,6].

The main challenge of the LNU circuit shown in Fig. 2 is to efficiently implement the two nonlinear functions (f+ and f- as shown in Fig. 3) that are needed to calculate ADD/SUB in LNS. These two functions are approximated using a combination of LUTs and interpolators. To achieve IEEE single precision accuracy with reasonable area overhead, two separate blocks are used. The main 2nd order polynomial interpolator handles the regions where f+ and f- are almost linear and the corresponding LUTs have been partitioned into logarithmically spaced segments, using only 1k and 2.1k entries for f+ and f- respectively. Each set of coefficients has been optimized using a finite-precision-aware minimax fitting algorithm to minimize the bit widths (from 3 to 32 bits) and the amount of required coefficient samples at the given precision requirement. For results that fall in the critical region (Fig.3), a novel co-transformation block has been designed, which employs a mathematical transformation to circumvent precision problems [1,2]. First it evaluates $(1-2^z)/(1+2^z)$ which can be efficiently approximated using a first order polynomial. The result is then fed to a log2 block, which has been implemented with the HOTBM method from [4]. In order to minimize LUT size, the log2 domain has been reduced to [1,2), and the output of $(1-2^z)/(1+2^z)$ is calculated in shifted format by storing the 0th order coefficients in pre-shifted format. The coefficients are then properly aligned and processed. This arrangement allows the employed zero counter, shifter and log2 block to be used to calculate typecasts and native log2 functions as well. The pre-processing block of the LNU decodes the command, chooses the appropriate interpolator block and gates the input operators. Evaluations showed that silencing is critical for low-energy operation and reduces energy consumption for LNU additions from 107.7 pJ/op to only 28.7 pJ/op in our design. Finally, the post-processing block combines all intermediate results and performs special case handling such as over-/underflows. Our LNU implements more functions (casts, exp, log), uses far smaller LUTs (14.1 kB instead of 22.9 kB) and reduces the area overhead by 35% when compared to the most advanced design in literature [2].

Both units are tightly integrated in the processor's data path as shown in Fig. 1 and share a write back port of the register file with the load-store unit. The FPU requires only one pipeline stage, while the LNU requires three stages. The shared LNU is managed by a fair round-robin arbiter. The shared LNU is only needed to process LNS ADD, SUB, LOG, EXP and casts. Many LNS operations such as MUL, DIV, SQRT and comparisons can be directly computed in the integer ALU of the cores in a single cycle, which is energy efficient, reduces LNU contentions, and makes it more attractive for sharing in a multi-core setting. The efficiency of several LNS and FP-instructions is compared in Fig. 4. While LNS ADD/SUB are less energy efficient than the FP equivalents, the LNS MUL requires 36% less energy than in FP. Apart from these basic instructions, LNS supports extremely energy efficient, single cycle square-roots (6.4 pJ/op) and divisions (12.1 pJ/op) utilizing the shifter and adder of the ALU with dedicated special case handling. Also, complex functions such as 2^x and $\log_2(x)$ can be computed in the LNU in four cycles for 13.3 and 22.6 pJ/op, respectively.

Both architectures have been benchmarked using a suite consisting of linear algebra kernels, matrix decompositions and more complex, nonlinear functions involving projective transforms, radial basis functions, trigonometric functions and distance computations. The benchmark set has been generated using MATLAB and its C++ embedded coder to ensure that competitive kernel implementations are used. The LLVM compiler has been adapted to support the LNS format for OpenRISC, letting the compiler handle automatically low-level details, such as LNU latency. The upper part of Fig. 5 shows kernel characteristics such as FP instruction ratios, codesize and IPC where the lower part shows kernel execution time with its power consumption and energy savings. For complex kernels, such as 3D-distance computations and Cholesky decompositions the LNU can make use of its extended ISA (DIV, SQRT) allowing to run those applications up to 4.2x more efficiently as illustrated in Fig. 6. A big difference is seen where EXP and LOG functions are frequently used because they can be performed natively on the LNU, while the FPU has to use software emulations consuming 51(exp) to 85(log) cycles.

For pure linear algebra kernels AXPY, GEMM, and GEMV, the cluster with private FPUs is 5-13% more energy efficient than shared LNU due to LNU's longer latency for FP add/sub. This relatively small gap even for ADD-SUB intensive benchmarks is easily amortized for complex algorithms with many multiplications, divisions and nonlinear functions. The utilization of the shared LNU on our benchmarks was 0.37 on average with a maximum of 0.61 where the high utilization led to 10% stalls due to access contentions. On average, such contentions only occurred in 4% of all FP operations. The manufactured chips with private FPU and shared LNU integrated in a multi-core cluster are shown in Fig. 7. The two chips are comparable in size, but the shared LNU, with a top energy efficiency of 6.4 pJ/FLOP @0.8V, allows to compute complex nonlinear kernels up to 4.2x more energy efficiently than the FPU cluster.

Acknowledgments

We thank M. Burger, T. Gautschi, L. Müller, Y. Popoff, F. Scheidegger, and F. Schuiki for their valuable work and commitment during their semester projects. This research was supported by the IcySoC project, evaluated by the Swiss NSF and funded by Nano-Tera.ch with Swiss Confederation financing.

References

[1] J.N. Coleman et al. "The European Logarithmic Microprocessor," in IEEE TC, vol.57, no.4, pp.532-546, April 2008

[2] R.C. Ismail et al. "ROM-less LNS," in IEEE ARITH, pp.43-51, July 2011

[3] D. Rossi et al. "A -1.8V to 0.9V Body Bias, 60 GOPS/W 4-core Cluster in low-power 28nm UTBB FD-SOI technology," S3S, 2015

[4] J. Detrey et al. "Table-based polynomials for fast hardware function evaluation." IEEE ASAP, pp. 328-333, 2005

[5] S. Galal et al. "Energy-Efficient Floating-Point Unit Design," in IEEE TC , vol.60, no.7, pp.913-922, July 2011

[6] H. Kaul et al. "A 1.45 GHz 52-to-162GFLOPS/W variable-precision floating-point fused multiply-add unit with certainty tracking in 32nm CMOS." ISSCC'12

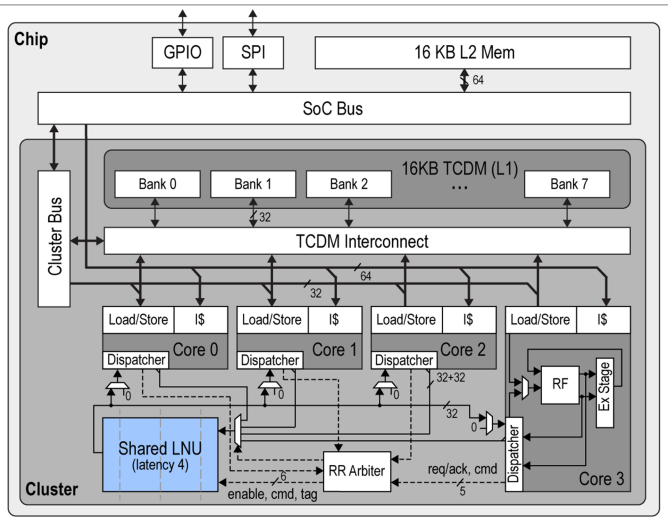


Figure 4.6.1: Four-core cluster architecture with one shared LNU directly integrated in the pipeline of the processors.

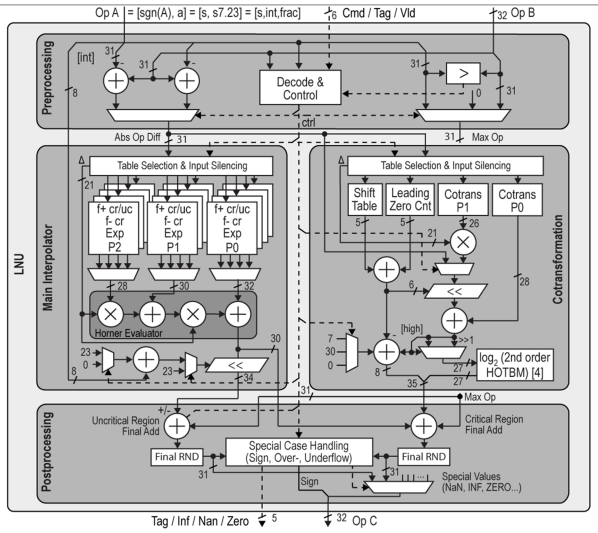
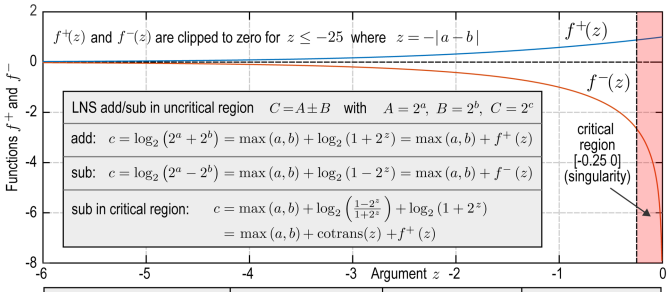


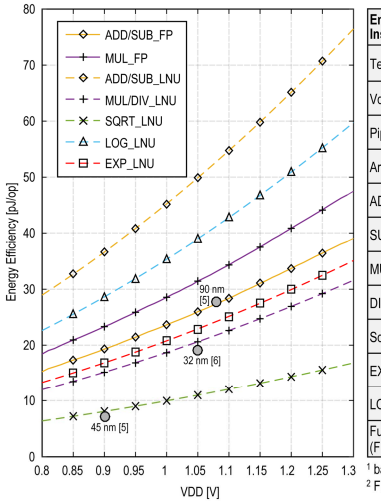
Figure 4.6.2: LNU architecture with main interpolator and cotransformation



LNU Comparison	LNU	ELM [1,2]	ROM-less [2]
functionality	ftoi, itof, add, sub, exp, log	add, sub	add, sub
precision (max/avg err) [ulp] [1]	0.479 / 0.177	0.454 / 0.178	0.498 / 0.174
LUT size [entries] / [Kbit]	5.1k / 113.1	8.2k / 256.4	4k / 183.3
technology	65 nm	180 nm	180 nm
area [squm]	57'264	904'943	589'357
Total area [kGE]	62	-	-
Post-synthesis (w/o registers) [kGE]	40	97	63
min delay [ns]	6	11.74	7.10
max delay [ns] (critical region)	6	13.15	14.79

[1] precision compared to standard floating point as calculated in [3]

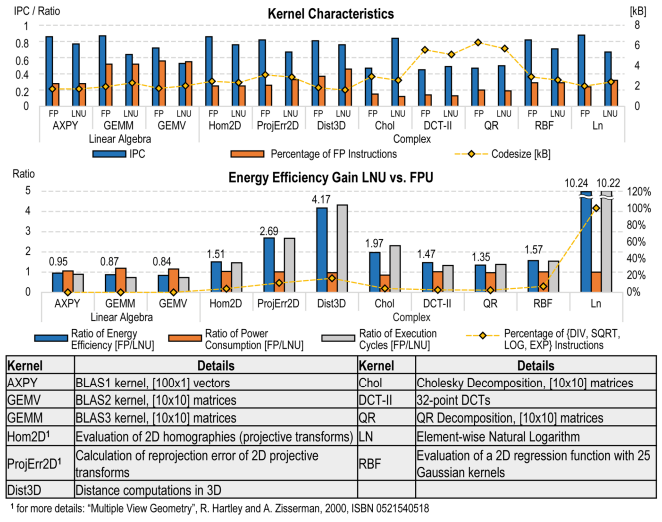
Figure 4.6.3: LNS f^+ / f^- functions with singularity in the critical region where the cotransformation applies. Comparison with related work.



Energy / Instruction	FPU [1]	LNU [1]	Horowitz [5]		Intel [6]
Technology	65nm LVT	65nm LVT	90nm	45nm	32nm
Vdd [V]	0.8	0.8	1.08	0.9	1.05
Pipeline Depth	2	4	10	6	3
Area [mm^2]	0.015	0.089	0.113	0.016	0.045
ADD [pJ/FLOP]	15.3	28.7	-	-	-
SUB [pJ/FLOP]	15.2	29.2	-	-	-
MUL [pJ/FLOP]	18.4	11.8	-	-	-
DIV [pJ/FLOP]	-	12.1	-	-	-
Sqrt [pJ/FLOP]	-	6.4	-	-	-
EXP [pJ/FLOP]	-	13.3	-	-	-
LOG [pJ/FLOP]	-	22.6	-	-	-
Fused MUL/ADD (FMA) [pJ/FLOP]	-	-	27.7 [2]	7.2 [2]	19.2

[1] background power (NOP) subtracted, includes core activity
[2] FPU datapath only

Figure 4.6.4: Energy efficiency comparison of different FP- and LNU-instructions.



Kernel	Details	Kernel	Details
AXPY	BLAS1 kernel, [100x1] vectors	Chol	Cholesky Decomposition, [10x10] matrices
GEMV	BLAS2 kernel, [10x10] matrices	DCT-II	32-point DCTs
GEMM	BLAS3 kernel, [10x10] matrices	QR	QR Decomposition, [10x10] matrices
Hom2D[1]	Evaluation of 2D homographies (projective transforms)	LN	Element-wise Natural Logarithm
ProjErr2D[1]	Calculation of reprojection error of 2D projective transforms	RBF	Evaluation of a 2D regression function with 25 Gaussian kernels
Dist3D	Distance computations in 3D		

[1] for more details: "Multiple View Geometry", R. Hartley and A. Zisserman, 2000, ISBN 0521540518

Figure 4.6.5: Energy, speedup, and power comparison of shared LNU vs. private FPU.

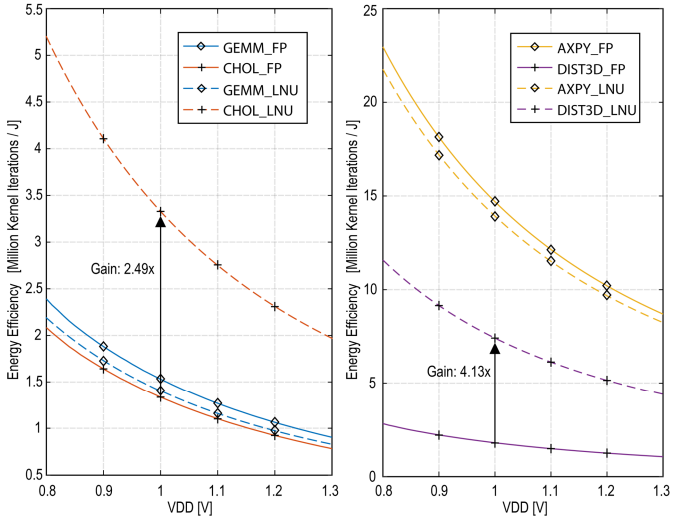
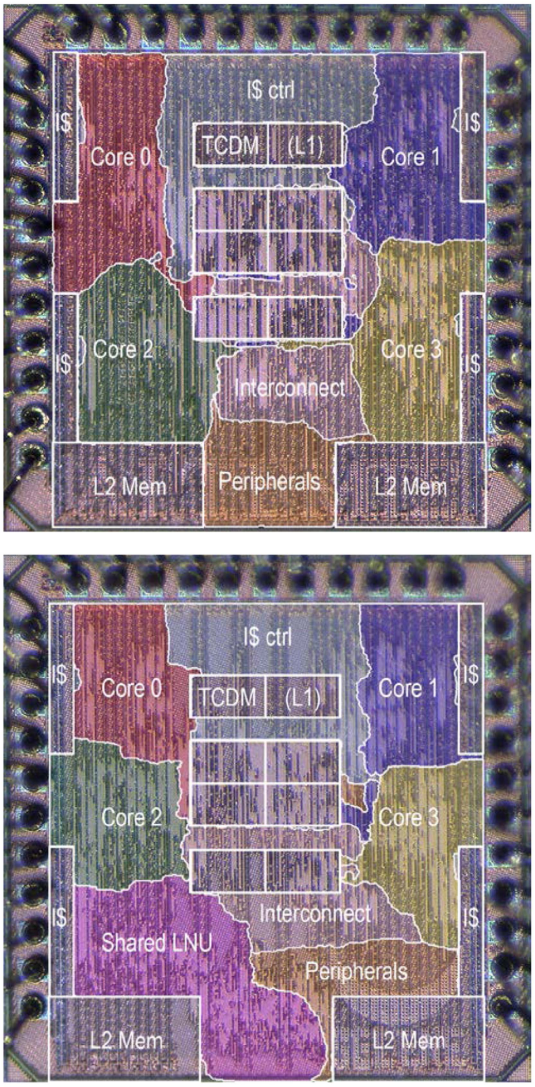


Figure 4.6.6: Energy efficiency in kernel iterations per J at different VDD levels.



Implementation Details		Private FPU	Shared LNU	ELM [1,2]
Technology		65nm LVT	65nm LVT	180nm
max speed [MHz]		374	337	125
max. Throughput [GFLOPS]		1.1	0.9	0.084
Power @100MHz, 1.2V, 25°C [mW]		41.84	44.0	-
Leakage @ 1.2V, 25°C [mW]		2.823	3.019	-
Best FP-operator efficiency @0.8V [pJ/FLOP]		15.2	**6.4**	-
Precision (max err) [ulp]		0.5	0.478[1]	0.454[1]
avg. lnu/fpu utilization		0.21	0.37	-
Total area [kGE]		719	749	-
Single core area [kGE]		51.1[2]	44.5	-
Instruction support		**Private FPU**	**Shared LNU**	**ELM [1,2]**
Latency add/sub/casts	hw	2/2/2	4/4/4	3/3(4)/ -
Latency mul/div/sqrt [3]	hw	2/-/-	1/1/1	1/1/1
	sw	-/62/56	-/-/-	-/-/-
Latency exp/log [3]	hw	-/-	4/4	- / -
	sw	51/85	-/-	-/-

[1] precision compared to FP [3] [2] including FPU [3] sw emulations: *div* (range reduction, linear initial estimate and 3 Newton iterations), *sqrt* (fast-inverse sqrt and 3 Newton iterations), *log* and *exp* (range reduction, polynomials from “Computer Approximations”, J. Hart, 1978, ISBN 0882756427)

Figure 4.6.7: Chip photos and datasheet.