FPGA implementation of Logarithmic Unit

Heřmánek A.(1,2), Matoušek, R.(2), Ličko, M.(2), Kadlec, J.(2)

(1) Katedra telekomunikací FEL ČVUT, Praha; (2) Centrum aplikované kybernetiky, ÚTIA AV ČR Praha

Introduction

Implementation of floating point in FPGA (Field Programmable Gate Arrays) is not easy. Paper presents FPGA core implementing these operations by representation of floating point numbers as 32-bit integer (fixed point) logarithm [1]. Basic arithmetical operations are performed in the logarithm numbering system (LNS) suitable for FPGA. Implemented intellectual property core takes just 18% of the XILINX Virtex XCV1000-6 FPGA device and operates at 17MHz. We have implemented MEX library emulating bit-exactly the properties of the final hardware. Therefore, Matlab serves for as simulation and visualization tool for prediction.

32bit logarithmic arithmetic by correction of interpolation error

In the presented Logarithmic Numbering System (LNS), is real number *x* is represented as the 32bit fixed point value. LNS multiplication, division and square roots can be implemented fast as the integer plus, minus and shift operations.

The conceptual solution to the LNS addition (subtraction) is based on these formulas:

$$
\begin{aligned}
C &= A + B \\
\log(C) &= \log(A + B) \\
&= \log\left[A \cdot \left(1 + \frac{B}{A}\right)\right] \\
&= \log(A) + \log\left(1 + \frac{B}{A}\right) \\
&= \log(A) + \log\left(1 + 2^{\log(B) - \log(A)}\right) \\
L_C &= L_A + \log\left(1 + 2^{L_B - L_A}\right)
\end{aligned}
$$

$$
\begin{aligned}
C &= A - B \\
\log(C) &= \log(A - B) \\
&= \log\left[A \cdot \left(1 - \frac{B}{A}\right)\right] \\
&= \log(A) + \log\left(1 - \frac{B}{A}\right) \\
&= \log(A) + \log\left(1 - 2^{\log(B) - \log(A)}\right) \\
L_C &= L_A + \log\left(1 - 2^{L_B - L_A}\right)
\end{aligned}
$$

$$or \quad L_C = L_A + f_a(L_B - L_A) \quad where \quad f_a(r) = \log\left(1 + 2^r\right) \qquad or \quad L_C = L_A + f_s(L_B - L_A) \quad where \quad f_s(r) = \log\left(1 - 2^r\right)$$

The f_a and f_s terms have to be approximated by look-up tables. This presented for the long time serious problem. The f_a and f_s terms have to be approximated by the look-up tables. Dr. Coleman has proposed innovative, patented approach [1],[10], leading to the drastic reduction of the size of look-up tables related to the linear interpolation of f_a and f_s by parallel evaluation of partial tables and joint correction term.

Coleman's approach [1], [10] leads to a solution, which is suitable for the FPGA implementation. It avoids the need of barrel shift (used in ASIC floating-point units for shifts of the mantissa). Implementation of the hardware equivalent of barrel shifts is area-costly an ineffective in FPGA. That is why FPGA design does not use floating point. LNS ALU provides one of the first hardware solutions for this problem. See Fig. 1.

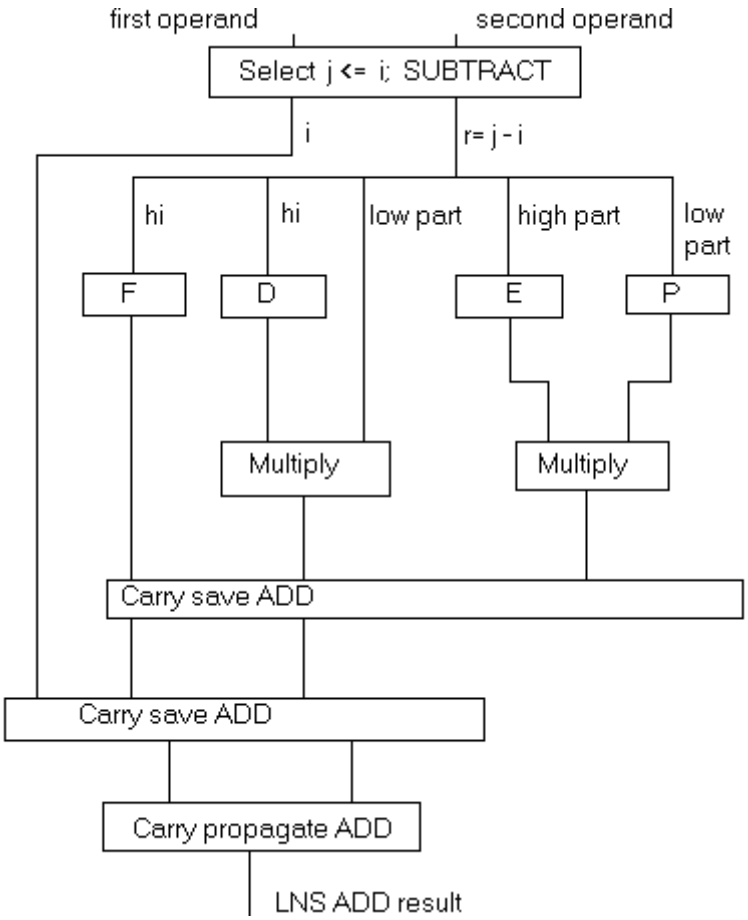

Fig. 1: Coleman's approximation method based on parallel access to 2 sets of look-up tables. The tables E and P serve for the linear interpolation. The idea behind the Coleman's method is the special (patented) selection of the approximation intervals. It results in an identical correction term for each of the approximation sub-intervals. This enables parallel computation of the correction term from a single set of F and D tables and drastic reduction of the look-up table size for the 32-bit precision.

The 32 bit LNS addition can be performed in time comparable to floating-point 32bit addition without loss of the precision. It needs approximately 32k byte of look-up tables with the 32bit word-length. Internally the LNS ALU Operates on 35bit wide data representation. Therefore, the LNS add ALU is a valid candidate for the development of the IP Cores for the FPGA based designs, which need to operate with 32-bit precision range of floating point numbers. This research is performed under the EU ESPRIT 33544 HSLA Long-term research project, coordinated by the University of Newcastle, UK [1].

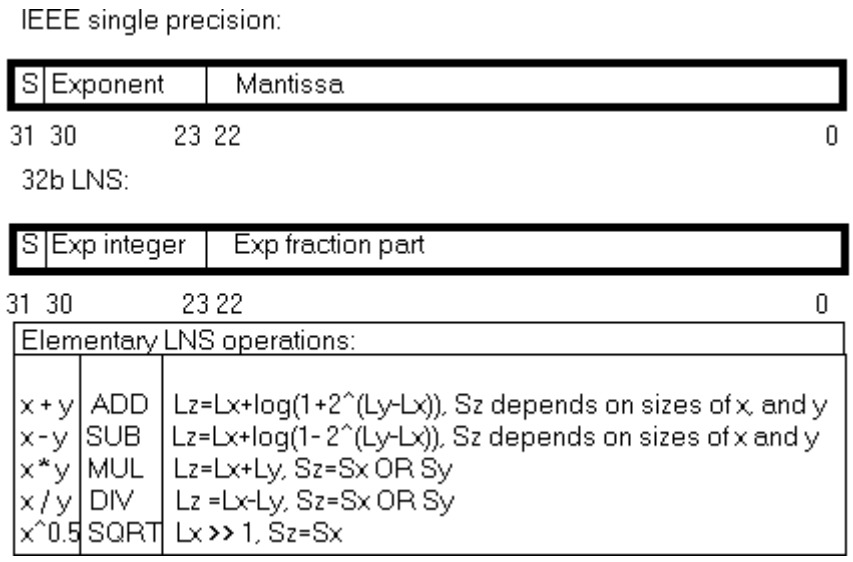

Fig. 2: IEEE 32-bit floating-point format, LNS 32-bit format and the internal implementation of the elementary LNS operations applied to the 32-bit integer representation of the logarithm.

LNS ALU C-code Implementation for Matlab/Simulink and RTW Compiler.

Port of the algorithm to FPGA has been started from C code coded originally for the 64-bit integer variables for:

- SIMULINK as the DLL S-function and Matlab 5.3 MEX-function
- W95/NT as the EXE code created by the RTW (Real Time Workshop) from Simulink for the generic target compiled by (Microsoft Visual C, version 6)
- 3L Parallel C on 64-bit Alpha AXP platform as APP application created by RTW, Alpha AXP target [2],[8].

The complete suite of bit-exact emulation of the elementary LNS operations has been created into the HSLA1v2 library. The library and works with the 64-bit integer data type **__int64** and the corresponding 64-bit integer arithmetic of modern C compilers. It is supported by Microsoft MSVC ver. 6.00 as well as the Alpha AXP compiler.

The initial section of original Pascal simulator code created the 35-bite wide tables and operates in the extended floating-point format. We have left this original table-generation in Pascal, and exported the data-tables into C headers. These headers declare tables as **static __int64[]** data types for the Matlab MEX functions. Any change in the table-definition (It can be made in the original Borland Pascal code owned by Uni. Newcastle) is reflected in the new C headers.

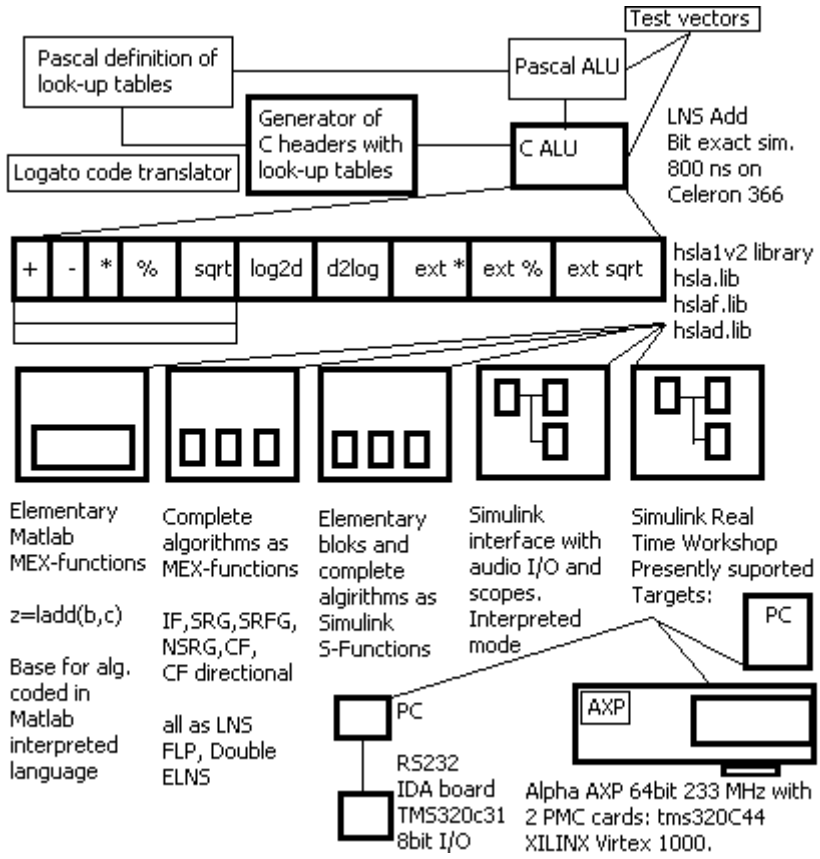

Fig. 3: presents complete structure of the final HSLA1v2 library.

FPGA implementation

The LNS ALU intellectual property core has been implemented in FPGA by the Handel-C tools and the XILINX Alliance 2.1i tools to place and route from the EDIF netlist for the FPGAs. Handel-C is C-like programming language designed to enable the compilation of programs into synchronous hardware. Handel-C enables to express algorithms at a high level. It includes constructs for the definition exact widths of data and for declaration of statements executed in parallel. In principle, each line of C- code is translated in the corresponding synchronous hardware realisation. Handel-C compiler is complemented with the DOS based simulator of created code. The language is described in [12].

In the context of Matlab, the basic design flow can be described as follows:

1. Port algorithm to Handel-C.
2. Compile program to .net file for Handel-C simulator (this can be arranged from Matlab command window in form comparable to creation of MEX-functions)
3. Use Matlab scripts to create test vectors (mat-files) for the simulator to evaluate and debug design. The simulator is able to step through all internal states of the hardware and generate results back to Matlab in the form of. The printed state values are combined with the related source code of Handel-C. Repeat steps 1.-3. To optimise the performance (latency) of the hardware in hand.

4. Use Handel-C compiler to target hardware netlist. This is in our case the EDIF 2 format for the XILINX Alliance 2.1i place and route software.
5. Use XILINX Alliance 2.1i tools to place and route netlist for the Virtex XCV1000-6 part in our case.
6. Download FPGA with result of place and route and execute the design in the target platform. The setup of constrains in the step 5 and in general all steps 2-5 have to be iterated to get fastest clock for the design for the given latency. RC1000 board from ESL [11], (now Celoxica) has been used as the target platform in our

case. The board is booted and interfaced to the PC by the application dependent C or C++ code (MS VC 6.00).

In our case, the emphasis has been again on the compatibility with Matlab.

- Matlab script is creating a set of test vectors corresponding to 1000 "single-step" iterations.
- Application specific program has been executed from Matlab as DOS executable.
- It boots the RC1000 board first.
- After the boot it opens the mat file with input vectors from Matlab and takes the responsibility for communication with the FPGA.
- step 6 has been executed from Matlab and the outputs from the FPGA have been communicated to the application specific program and converted to mat file for the Matlab post-processing

Fig. 4 presents the snapshot of the final implementation of the ALU on the FPGA and the interface to Matlab.

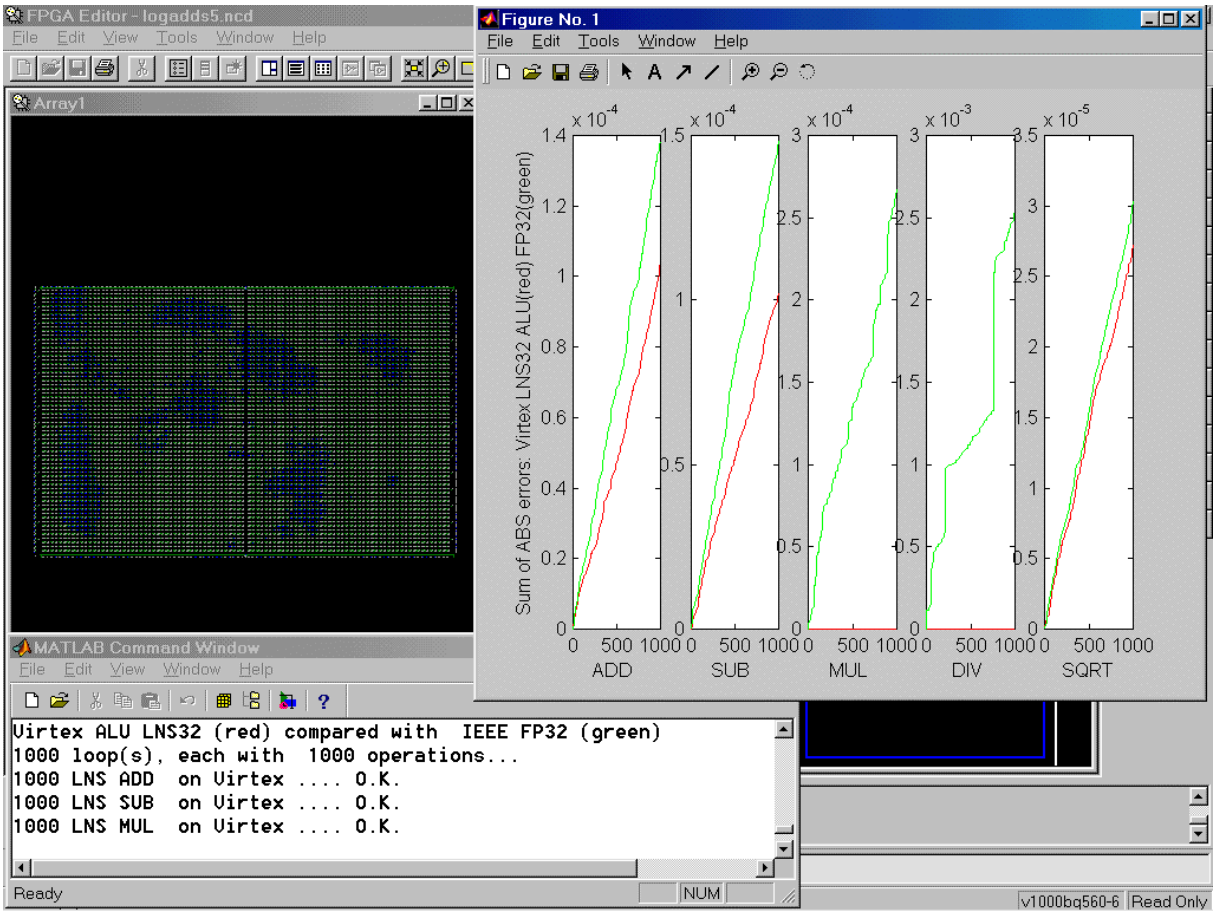

Fig. 4: presents the snapshot of the final implementation of the ALU on the FPGA and the interface to Matlab. Matlab is generating MAT file with 2x1000 random operands for each type tested of elementary operation. The ALU on the RC1000 FPGA reads the operands from Mat file (via support application communicating with the ALU at 8-bit parallel line. ALU computes the ADD, SUB, MUL, DIV and SQRT operations and the results are communicated back to Matlab. Matlab computes the HSLA1v2 version of the same computations and both results are compared. Any difference (single bit) would trigger the debugging mode with reporting of the differences. Matlab computes the IEEE single precision version of computations and the double-precision version of computations. Graphs present the absolute value of the arithmetic error is accumulated for each 1000-operation set. The larger error corresponds to the IEEE 32 bit float. The lower arithmetic error corresponds to the 32 bit LOG LNS. The 32 bit LNS outperforms 32 bit floating point.
The area of the XCV1000-6 FPGA occupied by the LNS hardware is displayed by the FPGA editor in the background.

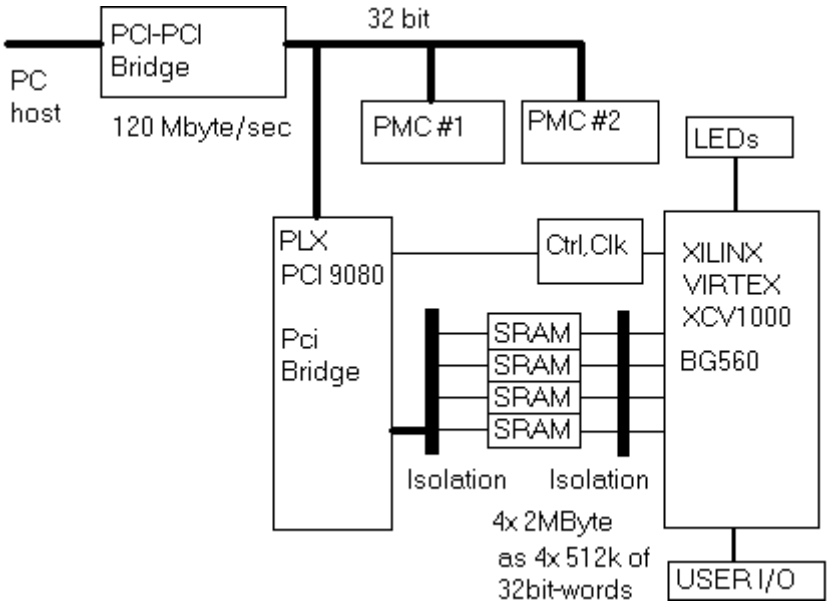

Fig. 5: block diagram of the RC1000 board [11].

Parameters of the FPGA implementation.

The implemented LNS IP-core operates at 17 MHz clocks. It consumes 18% of XCV1000-6 slices.
It is exactly 2325 slices out of 12288 available slices. None of the block-RAM areas is used.
The look-up tables are located in the four external banks. 4 Kwords in each of the 4 32bit-wide SRAMs of the RC1000 board are occupied by the tables. Each SRAM has 512Kwords of SRAM (508Kwords remain free).
The tables are pre-booted by the DMA via the PCI interface of the board.

- The operation MUL, DIV, SQRT are executed in 2 clock cycles.
- The ADD operation needs up to 8 cycles.
- The SUB executes in 8-10 cycles. 10 cycles are needed in special case of range-shift if both operands for the SUB are close.

The data range of the ALU is approximately (+/-) 3.4 e^ (+/-) 38.
The reported latency includes the complete handling of the overflows underflows and NaNs.

Conclusion

Presented FPGA design of the logarithmic ALU provides all elementary operations (minus, multiply, divide and square-root) and can be interfaced with Matlab. The LNS IP core is the basic foundation for creation advanced algorithmic IP cores based on the LNS arithmetic for the FPGA application in the hardware.

Contact

ÚTIA AV CR, (2) Prague 8, Pod vodárenskou věží 4, 182 08,
e-mail: kadlec@utia.cas.cz

This work was supported by the Ministry of Education of the Czech Republic under Project LN00B096.

References

[1] J.N. Coleman, “A high Speed Logarithmic Arithmetic Unit. ESPRIT Long-term research project No. 33544 “HSLA”.

[2] Kadlec J.: Acceleration of computation-intensive algorithms on parallel Alpha AXP processors. In: Preprints of the 3rd European IEEE Workshop on Computer-Intensive Methods in Control and Data Processing. (Rojicek J., Valeckova M., Karny M., Warwick K. eds.). UTIA AV CR, Praha 1998, pp. 89-98.

[3] Kadlec J., Schier J.: HSLA 3D Monitor Package. (Research Report No. 1925). UTIA AV CR, Praha 1998, 51 pp.

[4] Kadlec J., Schier J.: HSLA DSP Package. (Research Report No. 1924). UTIA AV CR, Praha 1998, 12 pp .

[5] Kadlec J., Schier J.: Numerical Analysis of a Normalized QR Filter Using Probability Description of Propagated Data. (Research Report No. 1923). UTIA AV CR, Praha 1998, 23 pp.

[6] Kadlec J., Schier J.: Rapid prototyping of adaptive control algorithms on parallel multiprocessors. In: Signal Processing Symposium. IEEE, Leuven 1998, pp. 115-118.

[7] Kadlec J., Schier J.: Results of the Global Probability Analysis Approach. (Research Report No. 1926). UTIA AV CR, Praha 1998, 89 pp.

[8] Kadlec J., Vialatte C.: Rapid prototyping and parallel processing under MATLAB 5. In: MATLAB Conference 1997. Kimhua Technology, Seoul 1997, pp. 120-125.

[9] Kadlec J., Matousek R., Vialatte C., Coleman N.: Port of Pascal FPGA-logarithmic-unit simulator to Simulink/RTW. In: Sbornik prispevku 7. rocniku konference MATLAB '99. VSCHT, Praha 1999, pp. 84-90.

[10] J.N.Coleman, E.Chester, C.Softley and J.Kadlec,"Arithmetic on the European Logarithmic Microprocessor", IEEE Trans. Comput. Special Edition on Computer Arithmetic, Vol. 49. No. 7. July 2000, pp. 702-715.

[11] RC1000-PP Hardware Reference Manual. Embedded Solution Limited, UK. The company have changed name to Celoxica http://www.celoxica.com/

[12] Handel-C, Language Reference Manual. http://www.celoxica.com/