EXTENSION FOR XILINX SYSTEM GENERATOR – LOGARITHMIC ARITHMETIC BLOCKSET

Miroslav Líčko[1], *Bastien Métais*[2], *Milan Tichý*[1], *Rudolf Matoušek*[1]
[1]Institute of Information Theory and Automation, Czech Republic
[2]Ecole Centrale Nantes, France

Abstract: The paper introduces support of floating point (FP) data format for the Xilinx System Generator (XSG) using logarithmic arithmetic. This type of arithmetic seems to be one of the promising ways to solve FP sort of DSP problems in practice. Our 32-bit high-speed logarithmic arithmetic (HLSA) keeps the accuracy according to IEEE 754 and even speed up some kinds of FP algorithms. Promising is 19-bit equivalent utilised in this paper. It still offers reasonable precision for the practical use and has minimal HW requirements.

1 Xilinx System Generator (XSG) and Appropriate Design Flow

XSG enables to design DSP systems for Field Programmable Gate Arrays (FPGAs) using MATLAB and Simulink. The XSG comprise blockset for bit- and cycle-accurate simulation and brings the possibility to automatically generate Hardware Descriptive Language (HDL). The HDL code is generated directly from Simulink block diagrams using blockset from XSG Library. Generated HDL code can be synthetised and implemented in the FPGA.

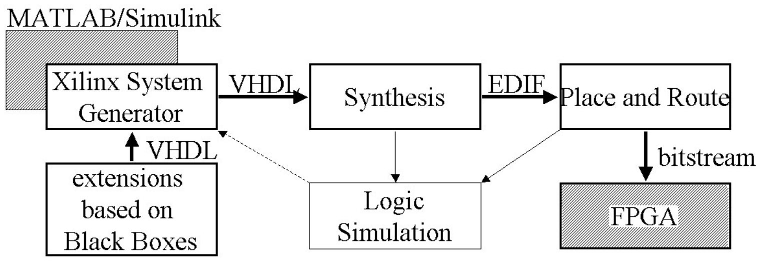

Figure 1 Design Flow using the Xilinx System Generator (XSG)

Automatically are generated (follow the *Figure 1*) not only command batches for FPGA Synthesis, Logic Simulation and implementation tools (Place and Route), but as well all necessary files for functional verification (test bench) and timing specification (constraint). In our design flow we have utilised VHDL but the XSG can deal with Verilog as well.

The automatic adaptation of FPGA technology (code generation) allowed us concentrate on the implementation problems using mostly MATLAB and Simulink environment. We have utilised XSG methodology based on Black Boxes to extend the set of functions (blockset) of the present XSG Library with our logarithmic arithmetic.

2 High Speed Logarithmic Arithmetic (HSLA)

The present XSG 2.2 offers only fixed-point arithmetic blocks and lacks so FP operations. Our HSLA based blockset extends XSG with arithmetic compatible with FP operations.
Behind the HSLA is our deep knowledge of related DSP problems. We have cooperated on the long-term research project [2] to implement arithmetic unit with the precision of FP (IEEE 754) representation. The research has resulted in the development of the logarithmic arithmetic unit and in an ASIC (European Logarithmic Microprocessor) based on it.

The FPGA circuits were used for the prototyping and verification of the ASIC. Other outputs as libraries in language-C, MATLAB/Simulink interfaces and HDL based intellectual property cores for the FPGA circuits were so programmed as side effect.

We used both high-level functions and IP cores to extend XSG using its Black Box methodology. There is an example of our 19-bit HSLA based library for the XSG in the *Figure 2*. For more information about current state of HSLA see [1] and for the base information about logarithmic arithmetic see [2] and [3].

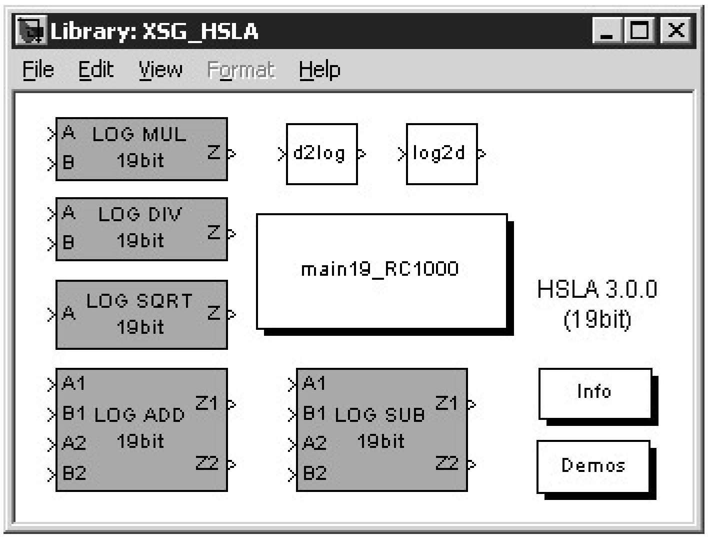

Figure 2 HSLA based library for the Xilinx System Generator (XSG)

3 Utilisation and Implementation of the HSLA based Library

There is a typical example of the utilisation in the *Figure 3*. The Gateway blocks are special XSG blocks. Their purpose is to separate FPGA algorithmic part (Subsystem in the *Figure 3*) designed using XSG blockset from the rest of (common Simulink) blocks. Block System Generator serves to automatically generate the HDL code out of the part between Gateways.

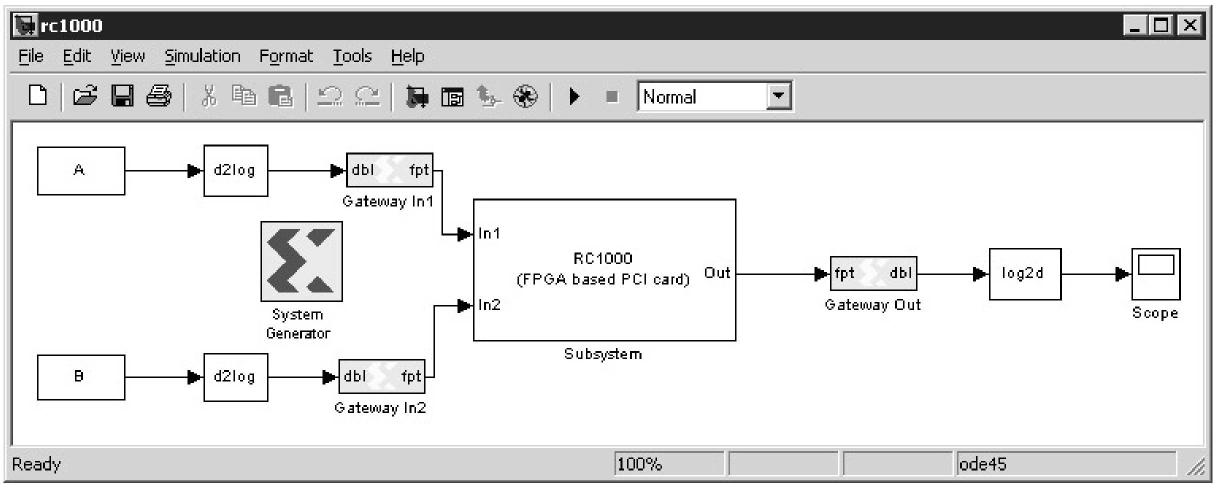

Figure 3 Support for RC1000 card

Our encapsulation of the HSLA into the XSG is based on its Black Boxes. See an example in the *Figure 4* for the logarithmic multiplication (LOG MUL) from the *Figure 2*.

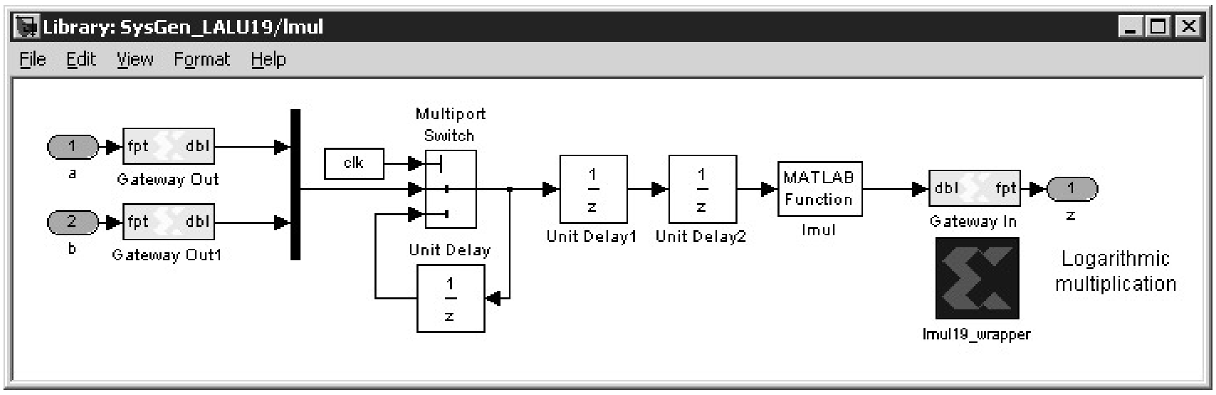

Figure 4 Implementation of the logarithmic multiplication (LOG MUL)

The in/out ports (a, b, z) of the subsystem with the Black Box are directly connected to the Gateway XSG blocks. It allows describe the functionality using common Simulink blocks for the simulation purposes. For instance, our logarithmic multiplication has latency two and is not pipelined (see parameters in the *Table 1* for all the basic operations). We have modelled this property using a combination of time-dependent Simulink blocks (see *Figure 4*).

Table 1 Implementation details of 19bit HSLA operations for Virtex2000e-6 on the card RC1000. The addition and subtraction uses 8 Virtex BRAMs.

Operation	# Slices	# Equiv. gates	Max. freq. [MHz]	Latency	Pipelined
Multiplication	218 (1%)	3 844	64	2	No
Division	235 (1%)	4 106	54	2	No
Square root	175 (1%)	3 170	59	2	No
Addition	1 478 (9%)	132 410	38	8	Yes
Substraction	1 789 (9%)	132 410	38	8	Yes

After the automatic code generation the Black Box (lmul19_wrapper in the *Figure 4*) will replace the whole part between in/out (a, b, z) ports. This file, called wrapper, is written in HDL language and provides just interface to HDL file which already describes the functionality. It allows replicate the usage of represented block many times in the same design schema.

This concept has allowed us encapsulate basic HSLA operations (see the overview in the *Table 1*) and finally add to the XSG possibility to design new class of DSP algorithms.

4 Support for the Target Hardware and Algorithm Implementations

By utilisation of XSG, we have prepared support to evaluate HSLA operations and advanced DSP algorithms based on them using MATLAB/Simulink.

We have utilised MATLAB/Simulink and XSG features to prepare support for RC1000 card [6] (see *Figure 3*) as the target HW. This wrapper can be reused for different applications running on RC1000 by changing just the part hidden under the block Subsystem.

All the basic operations form the *Table 1* were evaluated on RC1000 by this way. Parameters in the first three colums are results for the implementation on RC1000 card i.e. costs on all the necessary resources hidden behind the wrapper from *Figure 3* are counted in.

An advanced FP based DSP algorithm for the FPGA can be designed mostly under MATLAB/Simulink combining our HSLA operations and XSG. There is an illustrative example in the *Figure 5*. Different latencies and unused parts of the blocks are handled with the use of Delay XSG blocks. Latencies for all the operations are summarized in the *Table 1*.

$$\mathrm{Out1} = \frac{\mathrm{In1}}{\sqrt{|\mathrm{In1}|^2 + |\mathrm{In2}|^2}}$$

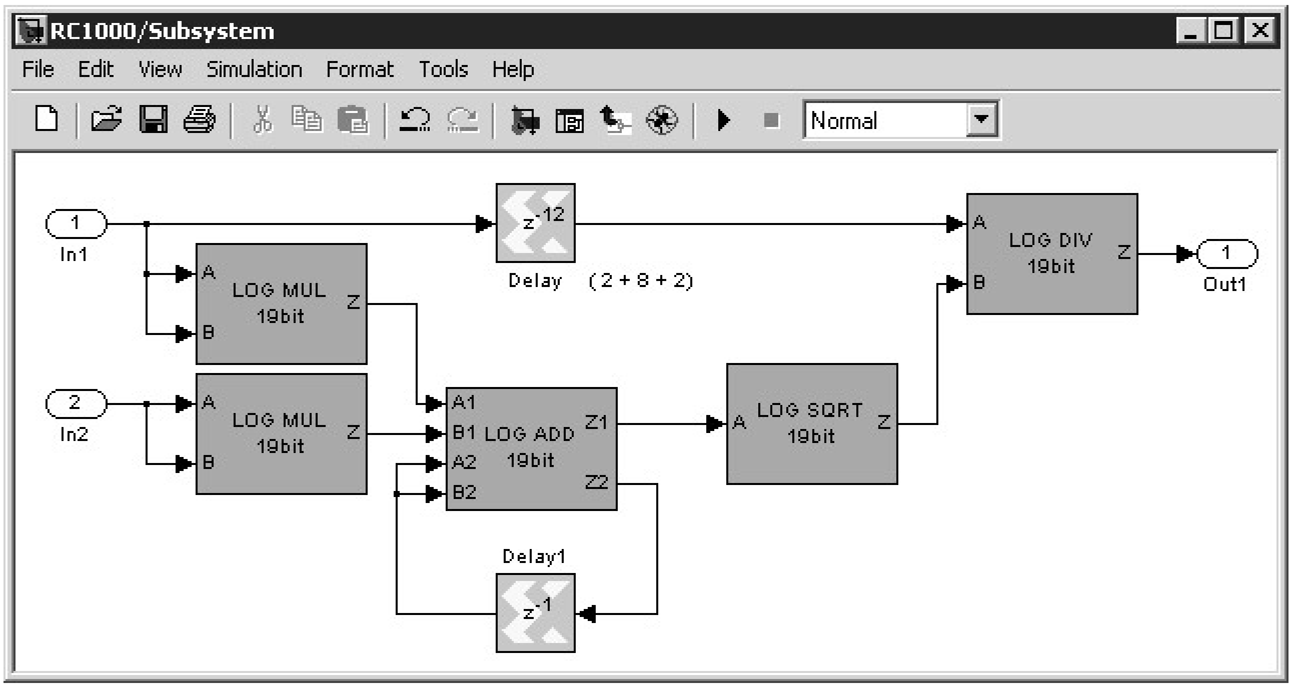

Figure 5 Utilisation of HSLA and XSG blocks

Our evaluations indicate that present realisation of the logarithmic arithmetic is unique for some kinds of DSP algorithms and that is even able to outperform, in other respects equivalent, IEEE 754 based arithmetic units.

5 Conclusion

Presented HSLA based blockset extends XSG with possibilities to solve variety of FP based DSP problems in real-time signal processing. For the latest information on HSLA see [1].

Features of the XSG we have utilised as FPGA designers:

- The automatic adaptation of FPGA technology (code generation) allowed us concentrate on implementation problems using mostly MATLAB/Simulink environment.

- We have extended set of functions (blockset) of the present XSG Library using methodology based on its Black Boxes.

In addition to the brief overview in this paper, a methodology introduced in [4] can be utilized to speed up the design process in terms of Black Boxes. At this moment we take into account partial run-time reconfiguration (PRTR) of the FPGA circuits. Under IST grant RECONF [8] we deal with PRTR methodology for dynamic re-configurable FPGA circuits.

References

[1] HSLA homepage of the Department of Signal Processing, UTIA
[available online: http://www.utia.cas.cz/ZS/home.php?ids=hsla, accessed: 28.9.2002]

[2] HSLA Project Homepage: ESPRIT programme, Long-Term Research project 33544
[available online: http://napier.ncl.ac.uk/HSLA, accessed: 28.9.2002]

[3] Coleman, J.N., Chester, E., Softley C. I. and Kadlec J. 'Arithmetic on the European Logarithmic Microprocessor', IEEE Trans. Comput. Special Edition on Computer Arithmetic, Vol. 49, No. 7, p. 702-715 and erratum vol. 49, no. 10, p. 1152 (July 2000).

[4] Licko, M., Pohl, Z., Matousek, R., Heramnek A. 'Tuning and Implementation of DSP Algorithms on FPGA' Proc. of Matlab, pp.226-230, ISBN 80-7080-446-7, (October 2001).
[available online: http://phobos.vscht.cz/matlab01/licko2.pdf, accessed: 28.9.2002]

[5] Hermanek, A., Matousek, R., Licko, M., Kadlec, J., 'FPGA implementation of logarithmic unit' Proc. of Matlab, pp.84-90, ISBN 80-7080-401-7, (2000).
[available online: http://phobos.vscht.cz/matlab00/hermanek.pdf, accessed: 28.9.2002]

[6] Alpha Data Parallel Systems Ltd. 'ADC-RC1000 card'
[available online: http://www.alpha-data.com/adc-rc1000.html, accessed: 28.9.2002]

[7] Xilinx Inc. 'Xilinx System Generator'
[available online:
http://www.xilinx.com/xlnx/xil_prodcat_product.jsp?title=system_generator, accessed: 28.9.2002]

[8] RECONF Project Homepage: IST programme 2001-34016
[available online: www.reconf.org, accessed: 28.9.2002]

Acknowledgments

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

Contact

Department of Signal Processing
Institute of Information Theory and Automation
Academy of Sciences of the Czech Republic
Pod vodarenskou vezi 4, 182 08, Prague 8, Czech Republic
Email: Licko@utia.cas.cz
www: www.utia.cas.cz/ZS