International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online)) Volume 4, Issue 11, November 2015)

# Acceleration Framework using MicroBlaze Soft-core Processors on FPGAs

Shrikant Jadhav[1], Christopher Doss[2], Clay Gloster[3], Youngsoo Kim[4]

[1,2]*Dept. of ECE North Carolina A&T State University, Greensboro, NC*
[3]*Dept. of CST North Carolina A&T State University, Greensboro, NC*
[4] *Dept. of EE San Jose State University, San Jose, CA*

***Abstract*— Offloading the complex computational kernel from the processor is the common way to improve performance of embedded system. In our work we are using MicroBlaze soft-core processor in design and implementation of acceleration framework. In acceleration framework MicroBlaze is coupled with co-processor with the help of communication bus. We can attach the co-processor to our design that can handle the computation part. This co-processor helps to offload the burden on the MicroBlaze and thus reduces clock cycles needed for computation.**

**In this paper we provide the acceleration framework to compute floating point natural logarithm value. The hardware implemented floating point natural logarithm unit is connected as co-processor to MicroBlaze. Xilinx provide a way to connect MicroBlaze processor and co-processor with the help of Fast Simplex Link (FSL). The FSL is used as mode of communication between floating point natural logarithm unit (co-processor) to MicroBlaze processor. We have implemented this framework on Virtex 5 and Zynq-7000. Our design consumes 28% on Zynq-7000 and 59% on Virtex 5 of the resources on FPGA. We compared the time in milliseconds required to execute different number of samples on (MicroBlaze processor + co-processor) design, on MicroBlaze soft-core processor. Our acceleration framework has achieved approximately 527x speedup on Zynq 7000 (100 MHz) over MicroBlaze soft-core design.**

***Keywords*—Field Programmable Gate Arrays (FPGA); MicroBlaze processor; soft-core; co-processor; acceleration framework; Fast Simplex Link (FSL); Floating point Natural Logarithm Unit.**

## I. Introduction

Field Programmable Gate Array (FPGA) is a semiconductor device that can be used to program desired hardware functions and applications. FPGA gives hardware designers the benefits of achieving high performance as well as re-configurability.

Hardware designers can reconfigure hardware blocks in an FPGA for specific applications or kernels in their application algorithms. Additionally, FPGA has soft-core processors such as MicroBlaze processor which can be instantiated without much of hardware resources. We use these soft-core processors as Processing Elements (PE) in link to auxiliary floating point units that we can accelerate scientific algorithms in our FPGA framework. In this paper, we demonstrated execution of logarithmic functions in FPGA with the help of software augmented with hardware floating point unit. This will ensure that HW designers will get acceleration of their algorithmic execution on a FPGA.

Floating point logarithm is used in number of real world applications. Using logarithmic scale scientific quantities are expressed as logarithm of other quantities. For example, the Decibel which is unit of measurement is a logarithmic unit expressing the ratio of physical quantity often power or intensity. In electrical signal transmission, it is used to quantify voltage loss. The earthquake intensity in Richter scale is measured by taking logarithm of the energy emitted during quake. Logarithmic scale is used in chemistry to represent pH value. Logarithm is used significantly in psychology field. Some of the human perception laws need the help of logarithm. The laws like Hick's law, Fitt's law, Weber-Fechner law uses logarithm to describe the relation between the entities the particular law is dealing with. Logarithm is used in probability theory, log normal distribution, used in maximum likelihood estimation of parametric statistical models. Logarithms are also useful in calculation of computational complexity of the algorithms. Fractals are infinitely complex geometric pattern that are self-similar across different scales. Logarithm is used to represent such Fractals. Logarithm is also used in music to describe intervals. In number theory logarithm is closely related in counting prime numbers, one of the important topics in number theory.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online)) Volume 4, Issue 11, November 2015)

Xilinx has provided a wonderful functionality in form of Fast Simplex Link (FSL). Acceleration is achieved using FSL which helps to offload computational part from the MicroBlaze processor by adding the co-processor to the design that computes the kernel part. MicroBlaze processor performing computationally intensive function like floating point natural logarithmic function will take more time or clock cycles which increases computation time. It is possible to take off the computational burden on the MicroBlaze and make it to work more efficiently.

Remainder paper is organized as follows. Section II takes a look into the other related works. Section III overviews the floating point hardware unit. Section IV describes the technical details of the framework. Section V provides the implementation details of the framework. Section VI provides results of the framework and finally Section VII concludes the paper.

## II. Related Work

Several works proposing the use of FSL to accelerate the applications have been proposed. Different kind of applications NMA, DWT, DCT, NLMS, FIR filter, biomedical application, etc. are hardware implemented and are attached to the processor using FSL in order to achieve acceleration over the software implementation. Here we have taken a look in some work which have designed hardware accelerator using FSL.

Nam Khanh Pham et al [9] introduces Nelder Mead Algorithm (NMA) solver engine based on Nelder et al. [7]. The author has provided the different ways to connect and get the results from NMA solver engine. The Hybrid solution, Embedded System (ES) and Pure HW solution are the three ways in which NMA solver engine can be used. Author has tried to prove with results that Hybrid solution involves more communication overhead than ES and Pure HW solution. In pure HW NMA solver and objective function are implemented on FPGA device so there is no communication overhead. In ES NMA solver is implemented on MicroBlaze or ARM processor and connected to objective functions with help of FSL in FPGA device. So the communication overhead involve is much less.

Matthew et al. [8] presents hardware floating point unit connected to FSL interface. The execution of hardware FSL FPU is 30% to 35% faster than software floating point library calls. In our project we have connected floating point logarithmic unit to FSL interface.

Antonino et al. [10] present design of a fast 2D-DCT hardware accelerator for a FPGA-based SoC. This accelerator makes use of a single seven stages 1D-DCT pipeline able to alternate computation for the even and odd coefficients using FSL connection to feed four 8 bit values per cycle.

Chang et al. [3] describe an embedded Normalized Least Mean Square (NLMS) adaptive filtering system, the main component of many DSP and communication systems. NLMS consist of a LMS co-processor core which implements the adaptive FIR filter using the LMS algorithm and Xilinx MicroBlaze soft processor which are connected with FSL buses.

Simone Borgio et al [1] shows how the most demanding task of the JPEG2000 compression algorithm, the two-dimensional discrete wavelet transforms can be hardware accelerated and implemented in a MultiProcessor System on-Chip prototyping platform on Field Programmable Gate Array (FPGA), CerberO. They have implemented architectures with different number of processors and hardware accelerators. Synchronization scheme adopted in CerberO is realized through the Synchronization Engine is a synchronization coprocessor and it is connected to the processors in the system through the FSL channels.

Kadlec J. et al [5] has presented two MicroBlaze designs. They have proposed single and double precision accelerator using FSL. Finite Impulse Response (FIR) is used as case study to document the acceleration results on batch computation of FIR filter. The data transfer to/from proposed floating point accelerator is done through FSL from DDR SDRAM.

El Mimouni et al [4] has used case study from biomedical field. They have designed real time cardiac monitoring system on Multiprocessor system using multiple MicroBlazes. FSL bus is used to communicate between these MicroBlazes. One MicroBlaze takes care of Human Machine Interface (HMI), other is responsible for Digital Signal Processing (DSP) and third MicroBlaze is used to control and coordinate the application software.

Calvio M.H. et al [2] has emphasized on parallelization of instruction execution. They have attached the MMX extension to MicroBlaze using FSL bus. The authors are trying to show the MMX extension capability that can execute Single Instruction Multiple Data type instructions that can help to improve overall performance of the MicroBlaze in some of the tasks.

Xu S. and Pollitt-Smith H. [11] present the MPSoC systems which consist of four MicroBlaze processors which are interconnected using FSL. The authors has presented two frameworks, one is based on SystemC modeling and simulation which allows researcher to explore MPSoC architecture and applications at high abstraction level and other framework is built around FPGA development board, related tools and system blocks allowing researcher to demonstrate system level concepts in working hardware.

Lazanyi J. [6] using Chemical Similarity Search Engine as a case study has proposed the accelerated architecture by eliminating operand and result transfer among the CPU and custom IP block. The application like auxiliary calculus where large amount of data is necessary, the commands are transmitted to the MicroBlaze through FSL and data is accessed from on-chip BRAM.

Floating point logarithm is one of the important units which is used in number of real world applications as discussed in Introduction section. As we go through the related work there are some work discussing floating point unit but we have not found floating point logarithm unit hardware accelerator. Also our framework has achieved significant acceleration over software logarithm function.

The contribution of our work is as follows

- We have implemented accelerator framework on Zynq 7000 and Virtex 5
- The 32 bits floating point logarithm unit which is hardware implemented is connected as co-processor to MicroBlaze through 32-bit FIFO called as FSL
- We compared the results from our acceleration framework with results from software function
- Using our framework we got 527x speedup on software logarithm function call

## III. Custom Hardware (Floating Point Natural Logarithm Unit)

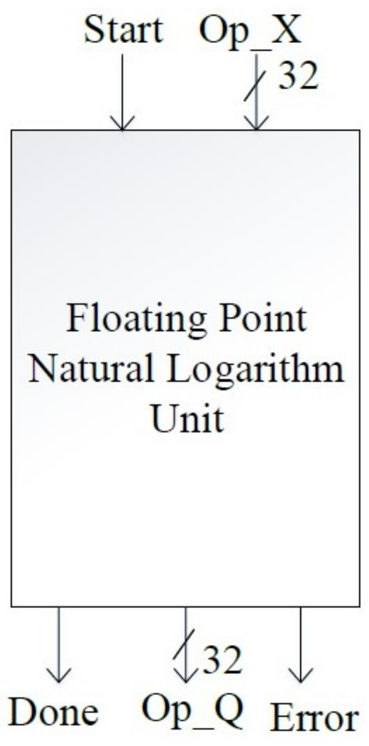


**Figure 1: Floating Point Natural Logarithm Unit**

Figure 1 shows the high level diagram of Floating Point Logarithm Unit (FPLU). The input (Op_X) and output (Op_Q) to FPLU is 32 bit. The FPLU process the data as soon as ’Start’ signal is enabled. If the data processing is successful ’Done’ signal is set high or else if data processing in FPLU fails ’Error’ signal sets high.

The single precision 32 bit floating point number representation is as shown in Figure 2. The floating point number can be positive or negative. Therefore most significant bit is sign bit. The size of sign bit is 1 bit. The 0 sign bit represents positive number and 1 sign bit represents negative number. The size of exponent field is 8 bits. Exponent is either an 8 bit signed integer from −126 to 127 or 8 bit unsigned integer from 0 to 255. The actual size of mantissa is 24 bits but leading bit is assumed as ’1’ and is therefore considered hidden, so the size of mantissa is 23 bits.

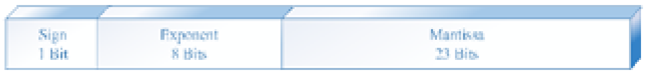


**Figure 2: Single precision 32 bit floating point number representation**

Figure 3 shows second level block diagram of floating point logarithm unit. FPLU is composed of adder, subtractor, multiplier, and divisor. The 32 bit hexadecimal value is input to the FPLU. ErrorMod unit handles error signal. Floating point logarithm calculation is performed in three stages. Total 129 cycles are required to compute the logarithm value by FPLU.

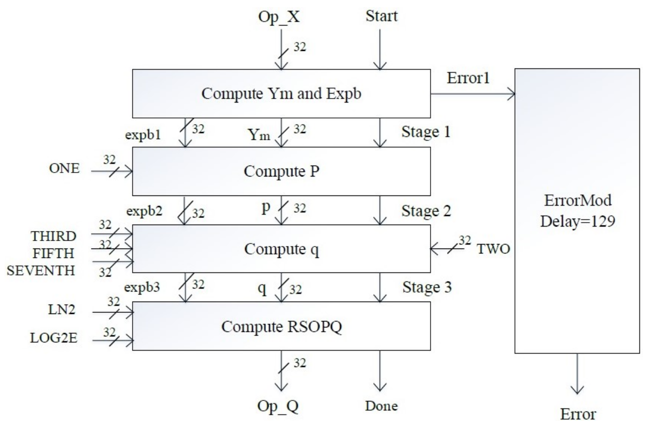


**Figure 3: Second Level Block Diagram**

## IV. Acceleration Framework

The floating point logarithm unit core is connected as co-processor to the MicroBlaze processor in order to achieve the acceleration. The FPLU core is connected to the MicroBlaze via dedicated link, using FSL as shown in Figure 4. FSL is 32 bit point to point FIFO based communication. It has dedicated and non-arbitrated architecture. In order to provide easy access FSL provides dedicated MicroBlaze processor C and ASM instruction. FSL bus interface is available in Xilinx Platform Studio (XPS) library core from Hardware → Create or Import Peripheral. FSL provides one extra control bit which is used by slave side interface to decode the word being transmitted as control word. It is also used to indicate start or end of the transmission of a frame. FSL supports both synchronous and asynchronous FIFO modes allowing the master and slave side of the FSL to clock at different rates.

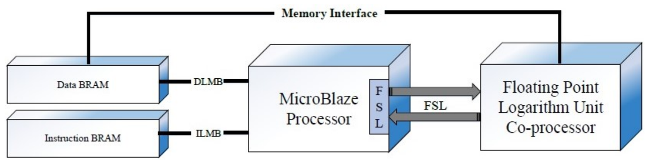


**Figure 4: Custom HW FP Natural Logarithm Unit connected as co-processor to MicroBlaze processor using FSL**

These FSL characteristics make it easy to achieve acceleration. In fact hardware core has just to communicate using two FSL ports, using data passing fashion rather than using memory mapped model. The FPLU core port is connected to MicroBlaze processor port via FSL channel. The slave port of the MicroBlaze processor is connected to the master port of the hardware core and master port of hardware core is connected to the slave port of MicroBlaze processor. Thus in acceleration framework MicroBlaze processor sends data to the FPLU core through the master port and receives through the slave port through non-blocking message passing primitives.

Using MicroBlaze soft-core processor to execute floating point logarithm is as shown in Figure 5. For the experimental purpose we executed floating point logarithm on the MicroBlaze soft-core processor. We used the results to compare and show the acceleration achieved using co-processor

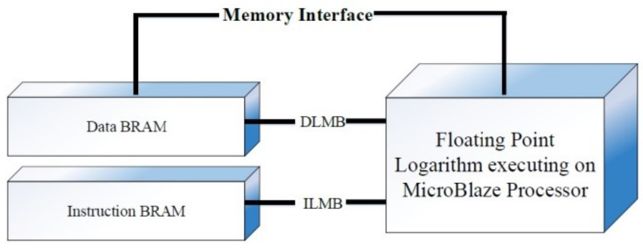


**Figure 5: Execution of FP Natural Logarithm on MicroBlaze soft-core processor**

Software application is developed in SDK in order to use the FPLU hardware accelerator on the FSL can send data to the hardware core and read their returned values by using simple non-blocking function nputfsl and ngetfsl. The nputfsl is used to feed the data to the hardware core and ngetfsl is used to receive the result from hardware core.

## V. Framework Implementation

MicroBlaze processor performing computationally intensive function (like Log function) will take more clock cycles which increases computation time. So as discussed we can offload the computation burden on the MicroBlaze and make it to work more efficiently. We have added the FPLU co-processor to our design that can handle the computation part.

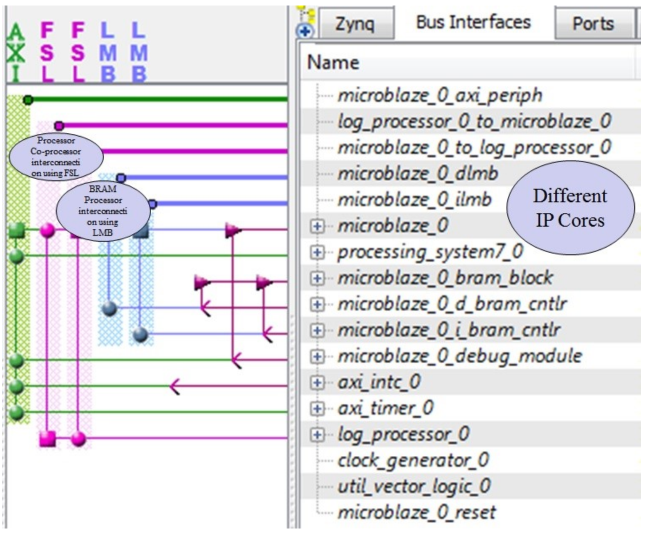
The FSL provides unidirectional dedicated FIFO based communication it helps in fast communication between MicroBlaze and co-processor. This co-processor takes over computationally intensive part of the program and helps to reduce burden on MicroBlaze processor from performing these intensive computations.

Figure 6 shows the hardware design of the acceleration framework. It is the snapshot from the XPS and shows different buses used to interconnect different design elements. The Local memory Bus (LMB) provides single-cycle access to on-chip RAM for the processor. The Advanced eXtentensible Interface (AXI) provides a connection to both on-and off-chip peripherals and external memory for processor. The Fast Simplex Link (FSL) bus acts as means for communication between the processor and co-processor in a FIFO fashion.

The LOG coprocessor is attached to MicroBlaze processor with help of two parallel FSL channels. The user will give the input through C code in Xilinx SDK. The user input is forwarded through FSL function (putfsl). One of the FSL signal (FSL_S_Data) forwards user input to LOG coprocessor for further computation. After FPLU co-processor finishes its work the result is carried back by one of the FSL signal (FSL_M_Data). The final result is received using FSL function (getfsl).

Using a separate co-processor to compute logarithm reduces number of clock cycles required as compared to computing FPLU by MicroBlaze processor itself. Thus using separate co-processor reduces burden on MicroBlaze processor and increase the efficiency and productivity of MicroBlaze processor.

The idea of multiple MicroBlaze processors with co-processor attached to it will help to speed up the process. The co-processors will be attached to the MicroBlaze with FSL and even multiple MicroBlazes will be connected to each other with help of FSL. It is quite obvious that execution time required for multiple MicroBlazes with co-processor attached will be less as compared to one MicroBlaze and one co-processor. Thus as no of MicroBlaze in the design increases the time required to complete the task will be reduced.


AXI
FSL
FSL
LMB
LMB
Processor Co-processor interconnecti on using FSL
BRAM Processor interconnecti on using LMB
Different IP Cores
Zynq
Bus Interfaces
Ports
Name

microblaze_0_axi_periph
log_processor_0_to_microblaze_0
microblaze_0_to_log_processor_0
microblaze_0_dlmb
microblaze_0_ilmb
microblaze_0
processing_system7_0
microblaze_0_bram_block
microblaze_0_d_bram_cntlr
microblaze_0_i_bram_cntlr
microblaze_0_debug_module
axi_intc_0
axi_timer_0
log_processor_0
clock_generator_0
util_vector_logic_0
microblaze_0_reset

**Figure 6: Hardware Design**

## VI. Experimental Results

The hardware design for acceleration framework is implemented on Zynq-7000 xc7z020 FPGA and is used for the experimental purpose. The Zynq-7000 runs at 100 MHz clock frequency. The results for different number samples are obtained at 100 MHz clock frequency. The 32 bits hardware timer is used in hardware design to measure the time required to execute different number of samples on FPLU co-processor and MicroBlaze soft-core processor itself. As hardware timer is 32 bits we were able to get results for 131072 samples on MicroBlaze processor. With FPLU core as co-processor we were able to get results for 16777216 samples.

Through nputfsl different numbers of samples were send to FPLU and the results for those samples were received through ngetfsl. We executed this experiment for different number of samples like 8,16,32,64....131072 number of samples. We have used hardware timer in our design to measure the time required to execute these number samples. We have compared the time required for different number of samples on FPLU core connected as co-processor against the time required executing floating point logarithm with different number of samples on MicroBlaze soft-core processor itself.

Table 1 shows acceleration achieved by FPLU core on Zynq- 7000. The measured time for different number of samples is converted into milliseconds. The table shows result for the samples with power of 8. Last column shows the acceleration achieved by FPLU co-processor on soft-core MicroBlaze processor. We have achieved the acceleration of approximately 527x.

**Table 1: Speedup of the Floating Point Logarithm Unit on Zynq 7000**

| # of Samples | Time required on Custom HW Zynq 7000 in MS | Time required on MicroBlaze Softcore in MS | Speedup on Zynq 7000 |
|---|---|---|---|
| 8 | $2.45 \times 10^{-3}$ | 1.255 | 512.45 |
| 64 | $1.91 \times 10^{-2}$ | 10.043 | 525.01 |
| 512 | $1.52 \times 10^{-1}$ | 80.348 | 527.18 |
| 4096 | 1.21 | 642.785 | 527.45 |
| 32768 | 9.74 | 5142.282 | 527.49 |
| 131072 | 38.99 | 20569.129 | 527.494 |

From figure 7 we can see the graph goes on increasing and at certain point the graph becomes linear. The number of samples are plotted on X-axis of the graph and corresponding time in milliseconds are plotted on Y-axis of the graph. The graph shows that after certain amount of samples the acceleration achieved gets stabilized. In our case for the samples 4096 and above the speedup achieved is approximately same.

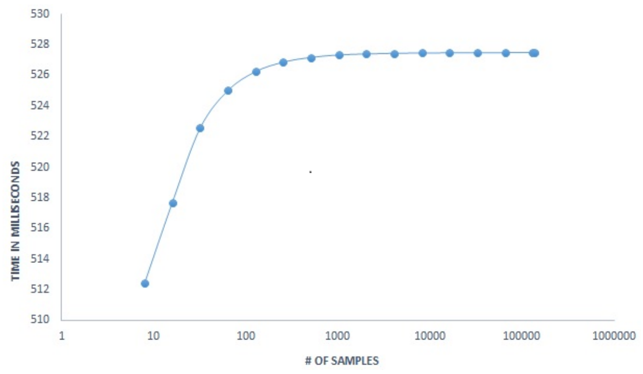


**Figure 7: Milliseconds corresponding to # of samples for FP Logarithm on Zynq 7000**

## VII. CONCLUSION

We have designed and implemented MicroBlaze-based acceleration framework using FP Natural Logarithm hardware. The acceleration framework hardware design consist of different IP cores that satisfy the application need like Timer, GPIO, interrupt controllers, etc. which are controlled by MicroBlaze processor. The hardware design is implemented with EDK suite tool and software application is developed with SDK tool with C code in the Xilkernel standalone environment. In terms of the resources utilization, our implementation occupies around 28% of the used FPGA, namely the Zynq-7000 xc7z020. We compared our results to show the speedup achieved. Our results showed the acceleration we achieved by connecting custom hardware to MicroBlaze processor with FSL.

*References*

[1] S. Borgio, D. Bosisio, F. Ferrandi, M. Monchiero,M. Santambrogio, D. Sciuto, and A. Tumeo. Hardware dwt accelerator for multiprocessor system-on-chip on fpga. In *Embedded Computer Systems: Architectures, Modeling and Simulation, 2006. IC-SAMOS 2006. International Conference on*, pages 107–114, July 2006.

[2] M. Calvio, S. Geninatti, and J. Benitez. Developing an mmx extension for the microblaze soft processor. In *Reconfigurable Computing and FPttAs, 2008. ReConFig '08. International Conference on*, pages 91–96, Dec 2008.

[3] C. Choo, P. Padmanabhan, and S. Mutsuddy. An embedded adaptive filtering.

[4] E.-H. El Mimouni and M. Karim. A microblaze-based multiprocessor system on chip for real-time cardiac monitoring. In *Multimedia Computing and Systems (ICMCS), 2014 International Conference on*, pages 331–336, April 2014.

[5] J. Kadlec, R. Bartosinski, and M. Danek. Accelerating microblaze floating point operations. In *Field Programmable Logic and Applications, 2007. FPL 2007. International Conference on*, pages 621–624, Aug 2007.

[6] J. Lazanyi. Instruction set extension using microblaze processor. In *Field Programmable Logic and Applications, 2005. International Conference on*, pages 729–730, Aug 2005.

[7] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[8] M. Ouellette and D. Connors. Analysis of hardware acceleration in reconfigurable embedded systems. In Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 3- *Volume 04*, IPDPS '05, pages 168.1–, Washington, DC, USA, 2005. IEEE Computer Society.

[9] N. K. Pham, A. K. Singh, A. Kumar, and K. M. M. Aung. Design space exploration to accelerate nelder-mead algorithm using fpga. In *Proceedings of the 2014 IEEE 22Nd International Symposium on Field-Programmable Custom Computing Machines*, FCCM '14, pages 100–, Washington, DC, USA, 2014. IEEE Computer Society.

[10] A. Tumeo, M. Monchiero, G. Palermo, F. Ferrandi, and D. Sciuto. A pipelined fast 2d-dct accelerator for fpga-based socs. In *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*, ISVLSI '07, pages 331–336, Washington, DC, USA, 2007. IEEE Computer Society.

[11] S. Xu and H. Pollitt-Smith. A multi-microblaze based soc system: From systemc modeling to fpga prototyping. In *Rapid System Prototyping, 2008. RSP '08. The 19th IEEE/IFIP International Symposium on*, pages 121–127, June 2008.