An ASIP for Image Enhancement Applications in Spatial Domain using LISA

Anup Sarma[1], Soubhagya Sutar[2], V.K. Sharma[3] and K.K. Mahapatra[4]
Dept. of Electronics & Communication Engineering
National Institute of Technology, Rourkela, India-769008
[1]cyberanup07@gmail.com, [2]soubhagya.sutar89@gmail.com, [3]vijay4247@gmail.com, [4]kmaha2@gmail.com

***Abstract*—Application-specific instruction-set processor (ASIP) has programming flexibility and performance as compared to application specific integrated circuits (ASICs). This makes it attractive choice for the future embedded processor on system-on-chip (SOC) design. We have designed an ASIP using language for instruction-set architecture (LISA). The designed processor has optimized instructions (a total of 8) for the image enhancement applications in spatial domain. The processor architecture is tested by writing soft codes for four different image enhancement algorithms. Good qualities of enhanced images have been obtained by the simulations. Finally, the processor architecture is prototyped in FPGA and implemented using TSMC 0.18 µm CMOS standard cell technology library. The architecture uses 21.72 K gate counts and consumes total power of 2.489 mW at 50MHz clock frequency and supply voltage of 1.8 V.**

***Keywords*—Application-specific instruction-set processor (ASIP), application specific integrated circuits (ASIC), system-on-chip (SOC), language for instruction-set architecture (LISA), image enhancement.**

I. INTRODUCTION

Semiconductor technology is becoming more and more advanced every year. Continuous technology scaling has enabled us to add more functionality on a single chip. Today, we have whole system, i.e., microprocessors, memories, peripherals etc., integrated on a single low cost and high performance chip, called system-on-chip (SOC) design [1-3]. Although, general purpose microprocessors offer flexibility, they have higher hardware cost and high power consumption than the application specific integrated circuits (ASICs). An intermediate solution is to use application-specific instruction-set processor (ASIP). ASIP has low hardware cost and low power consumption (close to ASIC) because of its customized instruction-set for a particular application area [5]. The programming capability provides much lower risks and shorter time-to-market, thus increasing the design productivity [6, 7].

Image enhancement is an important application in which degraded image quality is enhanced for both human perception and machine processing [8]. Histogram equalization, image smoothing by mean and median filtering, image sharpening by high pass filtering, and many others, are the image enhancement processes. Histogram equalization algorithm is implemented in FPGA in [9, 10]. In [11], general purpose median filter chip is designed in CMOS technology. FPGA design of median filter for image processing is done in [12-15].

We have designed a single chip ASIP for image enhancement applications in spatial domain using language for instruction-set architecture (LISA), a machine description language for fast design of ASIP. Automatic software development tools (compiler, assembler, linker and simulator) and synthesizable HDL (both VHDL and Verilog) code for efficient hardware are obtained from CoWare tool using LISA [16]. The processor architecture has been tested by writing soft codes in assembly language for four different image enhancement algorithms namely, histogram equalization, mean filter, median filter and high pass filter. Good qualities of enhanced images have been obtained by the simulations in processor architecture. Finally, the processor architecture is prototyped in Xilinx FPGA XC2VP30 device on Virtex-II pro board and implemented using TSMC 0.18 µm CMOS based standard cell technology library. The architecture uses a total of 21.72 K gate counts and consumes 2.489 mW dynamic power at 50 MHz clock frequency and at the supply voltage of 1.8 V.

The remaining sections of the paper are organized as follows. Section II gives an overview processor design in CoWare using LISA. Designed ASIP for image enhancement application is presented in section III. Section IV contains FPGA and ASIC implementation results. Conclusions are drawn in section V.

978-1-4577-0792-6/11/$26.00 ©2011 IEEE

II. Overview of Processor Design Using LISA and CoWare Tools

Instruction-set processor design is highly complex process and requires a lot of effort and time. Architecture explorations and architecture implementations are the two major steps in the processor design [1]. During the architecture exploration phase, different architectures and instruction-sets are searched for to fulfill the requirements. Architecture explorations require software development tools like compiler, assembler, linker and simulator. In architectural implementation, the architecture is transformed into register transfer level (RTL) code for the hardware implementation. Designing software development tools itself is tedious and any inconsistency between simulated architecture using software and its RTL code can lead to complete failure of the design.

CoWare is a tool for fast designing of instruction-set processor. Figure 1 depicts the processor design flow using LISA and CoWare [16, 17]. Processor designing in LISA has the following advantages

- Auto generation of software development tools
- Auto generation of RTL code
- Single high level language brings consistency between software development tools and generated RTL code

Processor description in LISA mainly consists of hardware and software model of the architecture. The hardware model comprises the definition of processor resources like registers, memories, pipeline and buses for accessing memories. The software model comprises the definition of processor instruction set, their binary instruction coding, assembly syntax and response of the processor hardware to instruction.

A. Hardware Modeling

In hardware modeling, all objects that are required to build the resource model as well memory model are to be declared. From hardware point of view, resources define the processor and it is the resources that are driven to new states according to steps performed by the processor. Resources, which have to be declared globally, are visible to all operations, thereby reflecting the true hardware nature of the components. The RESOURCE section allows the declaration of objects like simple resources, memory maps and special resources. Simple resources are register and register flags, ideal memory arrays, signals and flags and other resources that are not visible in the architecture. Memory map includes, mapping of memories into processor address spaces and connectivity between memory and bus modules. Special resources comprise pipeline

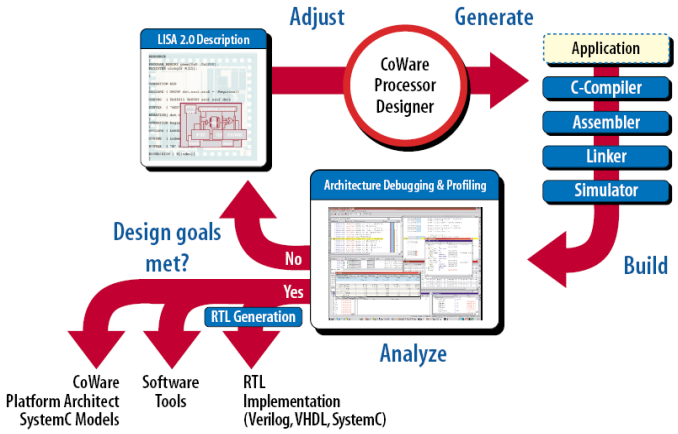


Figure 1. Complete processor design from LISA to RTL using Coware [17]

structure for instruction and data paths and pipeline registers storing data on its way from one pipeline stage to the next.

B. Software Modeling

The software model of the processor consists of processor instructions that are implemented using a basic entity in LISA 2.0 model, which is "operation", representing behavior, structure and instruction set of the programmable architecture. At the root level, the execution of an instruction is accomplished by the execution of corresponding operation. An operation possesses several attributes which are

- The CODING section describes the binary image of the instruction word.
- The SYNTAX section describes the assembly syntax of instructions and operands involved in each instruction.
- The BEHAVIOR and EXPRESSION sections enclose the behavioral attribute of the instructions.
- The ACTIVATION section describes the timing of other operations relative to the current operation (used when pipelined model is being used).

III. ASIP for Image Enhancement in Spatial Domain

Image enhancement can be done in spatial domain as well as in transformed domain. The designed ASIP instructions-set are optimized for implementing the image enhancement applications in spatial domain to reduce the hardware cost. Table 1 shows the specifications of the processor. Total 8 instruction-sets are used to process image in spatial domain. They are ADD, SUB, MUL, SHR, SHL, AND, OR and XOR.

Soft codes in assembly language have been written for four different enhancement algorithms namely histogram equalization, mean filter, median filter and high pass filter. Figure 2 shows the simulation model of the processor. Simulation in ModelSim involves creation of memory definition files using the exe2txt utility provided by CoWare. It takes the executable generated in previous step and the memory configuration file as input and as shown in the figure, the output of the 'exe2txt' application is two files, 'contents_prog_mem.mmap' and 'contents_data_mem.mmap, which are copied into the work directory of the project created by using the VHDL files of the processor model. When simulation is run, as soon as reset signal is asserted the program and data memories read these files (filenames are supplied as parameter) and loaded with values generated at the corresponding address locations. Image data is simulated in processor model and processed data is taken to reconstruct the images.

TABLE I
DESIGNED ASIP SPECIFICATIONS

Name	Bits allocated
Instruction Length Opcode Length	32-bits 8-bits
General Purpose Register	16-bits (8 nos.)
Program Counter	32-bits
Program Memory	32-bits
Data Memory	16-bits
Pipeline stages	3

A. *Histogram Equalization*

Histogram equalization is used to improve the contrast of an image. More details can be found in [8]. Figure 3 shows original and histogram equalized images using processor simulation model in ModelSim. Original histogram and histogram after processing are depicted in Figure 4.

B. *Mean Filter*

Mean filters are widely employed for image smoothing operation by reducing intensity variation among the neighboring pixels. This is achieved by replacing each pixel value in an image with the average of its neighboring pixel values, including the pixel under consideration. Here, we

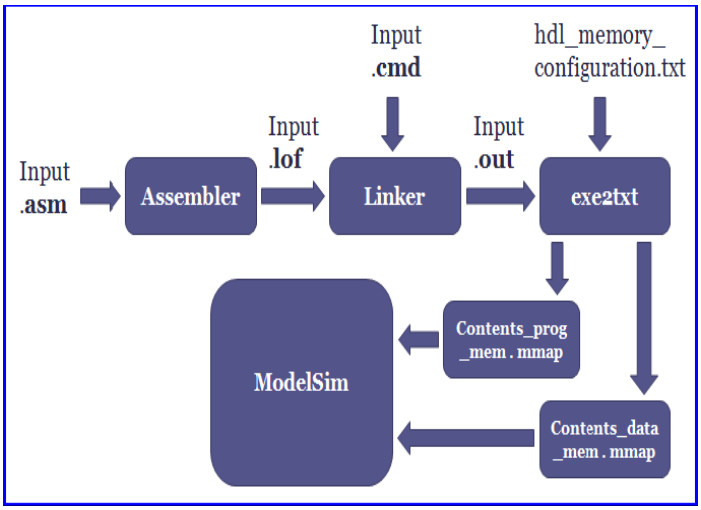


Figure 2. Processor simulation model using ModelSim

implement the following filtering kernel having form

$1/16\times$

1	2	1
2	4	2
1	2	1

Advantage of using this kernel is that different pixel values have weights that vary inversely according to their distance from central pixel. Thus, it allows more preference to be given to nearer pixel values, maximum being given to the central pixel itself, and thereby reducing effect of blurring in the process of smoothing. Another advantage is that this algorithm lends itself easily towards hardware implementation as the division factor is 16 (=sum of all filter coefficients), and is an integral power of 2. Figure 5 and Figure 6 show the original and reconstructed images after the simulation in ModelSim.

C. *Median Filter*

Median filter is a low pass filter as mean filter. It provides very good results in the presence of certain types of random noise such as salt and peppers. Pixel value under consideration is replaced by median of its neighbor and including pixel value itself. Implementation of median filtering is computationally expensive as the pixel set requires to be sorted, either in ascending or descending order. For simplicity of implementation, bubble sorting has been applied. Simulation results are depicted in Figure 5 and 6.

D. High pass filtering (Image sharpening)

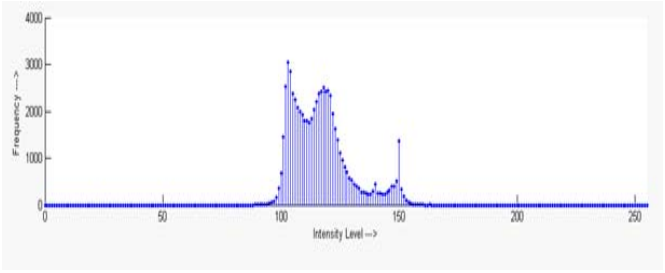
Here, high pass filtering is implemented with the help of Laplacian operator, which is a second order derivative. Filtering mask used for Laplacian is

1	1	1
1	-8	1
1	1	1

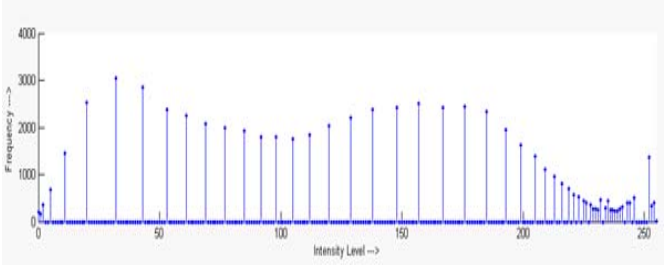
With negative center co-efficient being used, the sharpened image is obtained by subtracting the Laplacian image from the original image. The negative values that may occur in the Laplacian as well as in the final sharpened image due to subtraction are truncated to zero before the display. Results are depicted in Figure 7.

(a) (b)

Figure 3.(a) Original image, (b) histogram equalized image

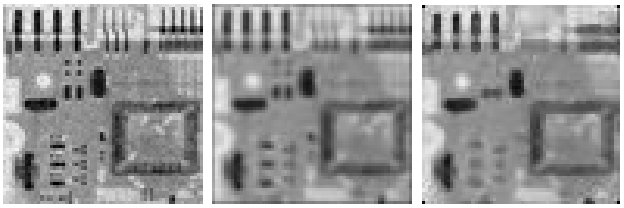


(a)



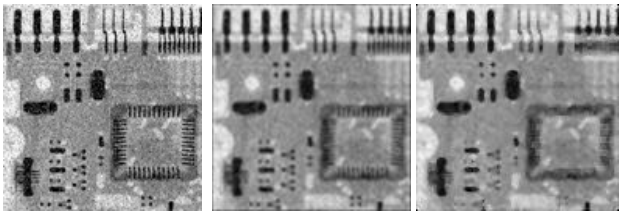
(b)

Figure 4.(a) Histogram of image in Figure 3(a), (b) histogram of image in Figure 3(b)



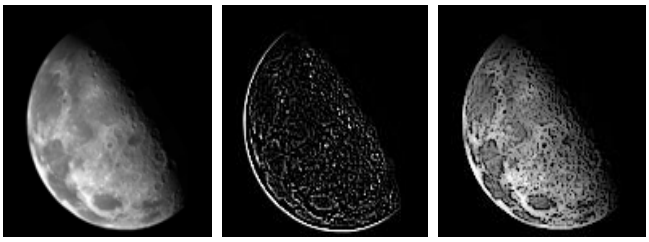
(a) (b) (c)

Figure 5.(a) Original image, free of salt and peppers noise, (b) reconstructed image by simulation for mean filtering, (c) reconstructed image by simulation for median filtering



(a) (b) (c)

Figure 6.(a) Original image containing salt and peppers noise, (b) reconstructed image by simulation for mean filtering, (c) reconstructed image by simulation for median filtering



(a) (b) (c)

Figure 7.(a) Original image, (b) laplacian filtered image, (c) enhanced image (by subtracting the Laplacian filtered from the original image)

IV. VLSI AND FPGA IMPLEMENTATION

The modeled ASIP for image enhancement is prototyped in FPGA and implemented in TSMC 0.18 µm CMOS standard cell library. RTL for the implementation is obtained by processor generator in CoWare design environment. FPGA prototyping is done in XC2VP30 device on Xilinx Virtex-II pro board. Table II shows the hardware utilization summary. One 18x18 multiplier is used as the processor instruction-sets contain one multiplication operation MUL. The generated RTL implementation results are depicted in Table III. 50 MHz operating frequency is used for the synthesis. The implemented architecture is optimum in area (217200 μm^2) as well as power consumptions (2.489 mW).

TABLE II
DEVICE UTILIZATION SUMMARY FOR ASIP IMPLEMENTED IN FPGA

FPGA-chip Xilinx XC2VP30	Used	Available	Utilization (%)
# of 4 input LUTs	6559	27392	23
# of slices	3419	13696	24
# of slice Flip Flops	532	27392	1
MULT18x18s	1	136	0

TABLE III
AREA AND POWER RESULTS FOR IMPLEMENTATION IN STANDARD CELL LIBRARY

TSMC 0.18 µm standard cell library	
Total cell area	217200 μm^2
Gate Count	21.72 K
Total Dynamic Power (global operating voltage 1.8 V)	2.489 mW
Operating frequency	50 MHz

V. CONCLUSIONS

An ASIP for image enhancement applications in spatial domain is developed using LISA processor description language. Soft codes in assembly language have been written to verify the developed architecture for four different image enhancement algorithms, histogram equalization, mean filtering, median filtering and high pass filtering. The results obtained after simulations and reconstructions have good quality of images. Finally architecture is prototyped in FPGA and implemented in 0.18 µm CMOS standard cell technology library. Total gate counts are 21.27 K and dynamic power consumption is 2.489 mW at 50 MHz clock frequency.

ACKNOWLEDGMENT

The authors acknowledge to DIT (Ministry of Information & Communication Technology) for the financial support for carrying out this research work.

REFERENCES

[1] A. Hoffmann, T. Kogel, A. Nohl, G. Braun, O. Schliebusch, A. Wieferink, and H. Meyr, "A novel methodology for the design of application specific instruction set processors (ASIP) using a machine description language," *IEEE Transactions on Computer-Aided Design*, vol. 20, pp. 1338–1354, Nov. 2001.

[2] M. Birnbaum and H. Sachs, "How VSIA answers the SOC dilemma," *IEEE Compute*r, vol. 32, pp. 42–50, June 1999.

[3] A. Chandra and K. Chakrabarty, "Low-power scan testing and test data compression for system-on-a-chip," *IEEE Transaction on Computer Aided Design of Integrated Circuits and System*, vol. 21, no. 5, pp. 597-604, 2002

[4] H.Shen and F. P´etrot, "A flexible hybrid simulation platform targeting multiple configurable processors SoC," *Proceedings of the 2010 Asia and South Pacific Design Automation Conference,*Taipei, pp.155 – 160, Jan. 2010.

[5] F. Sun, S. Ravi, A. Raghunathan, N. K. Jha. "A Scalable Synthesis Methodology for Application-Specific Processors," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol.14(11), pp 1175-1188, Nov. 2006.

[6] K. Keutzer, S. Malik, and A. R. Newton, "From ASIC to ASIP: The next design discontinuity," *Proceedings of International Conference on Computer Design*, Freiburg,Germany, pp. 84–90, Sep. 2002.

[7] O. Schliebusch, H. Meyr, and R. Leupers, *Optimized ASIP Synthesis*

from Architecture Description Language Models, Springer, 2007.

[8] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 2nd ed. Boston, MA: Addison-Wesley, 1992.

[9] A.M. Alsuwailem and S.A. Alshebeili, "A new approach for real-time histogram equalization using FPGA," *Proceedings of 2005 International Symposium on Intelligent Signal Processing and Communication Systems, ISPACS 2005*, vol., no., pp. 397- 400, 13-16 Dec. 2005.

[10] P.D. Ferguson, T. Arslan, A.T. Erdogan and A. Parmley, "Evaluation of contrast limited adaptive histogram equalization (CLAHE) enhancement on a FPGA," *2008 IEEE International SOC Conference*, pp.119-122, 17-20 Sept. 2008.

[11] M. Karaman, L. Onural, A. Atalar, "Design and implementation of a general-purpose median filter unit in CMOS VLSI," *IEEE Journal of Solid-State Circuits*, vol.25, no.2, pp.505-513, Apr 1990.

[12] R. Maheshwari, S.S.S.P. Rao, P.G. Poonacha, "FPGA implementation of median filter," *Proceedings., Tenth International Conference on VLSI Design*, 1997, pp.523-524, 4-7 Jan 1997.

[13] G. L. Bates and S. Nooshabadi, "FPGA Implementation of a Median Filter," *Proceedings of the IEEE TENCON '97, IEEE Region 10 Annual Conference*, *Speech and Image Technologies for Computing and Telecommunications,* vol. 2, pp. 437-44, 2-4 Dec. 1997.

[14] T.-W. Lee, J.-H. Lee and S.-B. Cho, "FPGA Implementation of a 3/spl times/3 window median filter based on a new efficient bit-serial sorting algorithm," *Proceedings of The 7th Korea-Russia International Symposium on Science and Technology, 2003, KORUS 2003* , vol.2, pp.237-242 vol.2, 6-6 July 2003.

[15] S.A. Fahmy, P.Y.K. Cheung, W. Luk, "High-throughput one-dimensional median and weighted median filters on FPGA," *IET Computers & Digital Techniques,* vol.3, no.4, pp.384-394, July 2009.

[16] LISA Language Reference Manual supplied by CoWare, Inc.

[17] Web,2011,http://www.europractice.rl.ac.uk/vendors/coware_ProcessorDesigner.pdf

[18] Z. Salcic and J. Sivaswamy, "IMECO: A Reconfigurable FPGA-based Image Enhancement Co-Processor Framework," *Real-Time Imaging*, Volume 5, Issue 6, pp.385-395, December 1999.

[19] Dake Liu, *Embedded DSP processor design: Application specific instruction set processors*, Morgan Kaufmann, 2008.

[20] User Reference Manual, Virtex 2 Pro supplied by Xilinx, Inc.

[21] Synopsys Reference Manual, Synopsys Inc.