

Chapter 20

Using Xilinx System Generator for Real Time Hardware Co-simulation of Video Processing System

Taoufik Saidani, Mohamed Atri, Dhaha Dia, and Rached Tourki

Abstract The use of rapid prototyping tools such as MATLAB-Simulink and Xilinx System Generator becomes increasingly important because of time-to-market constraints. This paper presents a methodology for implementing real-time DSP applications on a reconfigurable logic platform using Xilinx System Generator (XSG) for Matlab. The methodology aims to improve the design verification efficiency for such complex system. It presents architecture for Color Space Conversion (CSC) RGBTOYCbCr for video processing using Xilinx System Generator. The design was implemented targeting a Spartan3 device (3S200PQ208) then a Virtex II Pro (xc2vp7-6ff672). Obtained results are discussed and compared with an other architecture. The conversion method has been verified successfully with no visually perceptual errors in the transformed images.

Keywords Video processing · codesign environment · rapid prototyping · xilinx system generator · FPGA board

20.1 Introduction

Video processing and computer vision methods become increasingly important not only in the industrial applications but also in our daily life [1]. Video processing generally exploits tasks with very high computational demands. Such tasks can be handled by the standard processors and computers or by computers connected to the computational networks [1]. However, such approach is not always suitable that's why specialized hardware solutions based on digital signal processors (DSP) or a field programmable gate arrays (FPGA) are usually used in embedded systems [2, 3]. Xilinx System Generator allows the design of hardware system

T. Saidani (✉), M. Atri, D. Dia, and R. Tourki
E μ E Laboratory, FSMonatir, 5000, Tunisia
e-mail: Saidani.taoufik@yahoo.fr; Mohamed.atri@fsm.rnu.tn;
Dhaha.Dia@issatso.rnu.tn; Rached.tourki@fsm.rnu.tn

starting from a graphical high level Simulink environment [3,4]. System Generator extends the traditional Hardware Description Language (HDL) design providing graphical modules, and thus does not require a detailed knowledge of this complex language. The Simulink graphical language allows an abstraction of the design through the use of available System Generator blocks and subsystems [3]. This reduces the time necessary between the control design derivations and hardware implementation. In addition, the software provides for the hardware simulation and hardware-in-the-loop verification, referred to as hardware co-simulation [2,4], from within this environment. This methodology provides easier hardware verification and implementation compared to HDL based approach. The Simulink simulation and hardware-in-the loop approach presents a far more cost efficient solution than other methodologies. The ability to quickly and directly realize a control system design as a real-time embedded system greatly facilitates the design process.

The remainder of this paper is divided into six sections. After introducing, a description of Design methodology for implementation on FPGA with Xilinx System Generator is presented; Section 20.3 presents a study case which is Color Space Conversion Application. In Section 20.4, experimental results and software performances are detailed. Section 20.5 shows some discussion and comparison. This paper is concluded in Section 20.6.

20.2 Design Methodology for Implementation on FPGA with Xilinx System Generator

Efficient rapid prototyping system requires a development environment targeting the hardware design platform. The used tools are MATLAB R2007a with Simulink from MathWorks [4, 5], System Generator 10.1 for DSP and ISE 10.1 from Xilinx present such capabilities (Fig. 20.1). Although the Xilinx ISE 10.1 [2, 5] foundation software is not directly utilized, it is required due to the fact that it is running in the background when the System Generator blocks are implemented. The System Generator [2] environment allows for the Xilinx line of FPGAs to be interfaced directly with Simulink. In addition there are several cost effective development boards available on the market that can be utilized for the software design development phase.

MATLAB is an interactive software for numerical computations that simplifies the implementation of linear algebra routines. Powerful operations can be performed by using the provided MATLAB commands. Simulink [2,3] is an additional MATLAB toolbox that provides for modeling, simulating and analyzing dynamic systems within a graphical environment. The software allows for both modular and hierarchical models to be developed providing the advantage of developing a complex system design that is conceptually simplified.

Xilinx System Generator is a MATLAB-Simulink based design tool for Xilinx's line of FPGAs. Complex digital circuits have been developed using multiple

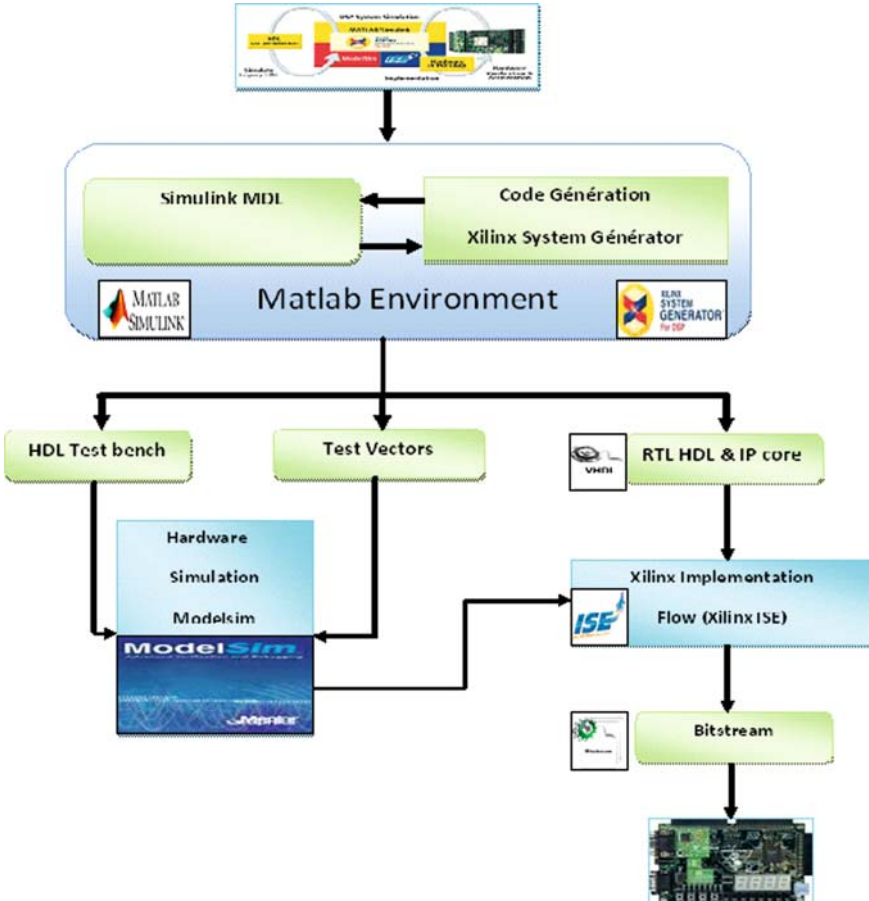


Fig. 20.1 Design methodology with Xilinx System Generator

Hardware Description Language (HDL) modules. Because of the abstraction level is very low within the HDL environment, the difficulty increases as the design becomes more complex [5].

The Xilinx Integrated Software Environment (ISE) is a powerful design environment that is working in the background when implementing System Generator blocks. The ISE environment consists of a set of program modules, written in HDL, that are utilized to create, capture, simulate and implement digital designs in a FPGA or CPLD target device [1, 2]. The synthesis of these modules creates netlist files which serve as the input to the implementation module. After generating these files, the logic design is converted into a physical file that can be downloaded on the target device.

20.3 Study Case: Color Space Conversion RGB to YCbCr

20.3.1 Overview

Color Space Conversion (CSC) [6, 7] is an important application in image and video processing systems. CSC has been implemented in software and various kinds of hardware. Hardware implementations can achieve a higher performance compared to software-only solutions. Application specific integrated circuits (ASICs) are efficient and have good performance. However, they lack the programmability of devices such as field programmable gate arrays (FPGAs) [8, 9].

Many video applications require converting video and image content from one color space to another [10–12]. Images and motion images (video) have utilized a wide variety of color spaces including: RGB, YCrCb, HSI, and other formats to represent the colors within the image [13]. Each of these color space representations has its own set of advantages and disadvantages. For example, RGB is often used for the most demanding applications where ultimate color fidelity must be maintained. Any given color that the human eye can see may be represented by a combination of the primary colors (Red – R, Blue – B, and Green – G). The human eye doesn't actually see equally well in the different color bands with our human-vision [12, 14] system optimized for the red, green bands but not quite as sensitive to changes in blues. Scientist and engineers looking for was to reduce the bandwidth and/or bit rate of a video system have created other color spaces (and sampling spaces) that reduce the amount of blue information in a system while maintaining a subjectively high picture quality. Furthermore, human vision is more highly tuned to changes in brightness (black and white or gray-scale changes) than it is to changes in hue (changes from one color or another with the same brightness). Therefore, many video systems sub-sample the color information [12] (chrominance) while transmitting the black and white (luminance) in full resolutions. This sub-sampling is often applied to luminance-chrominance color space systems such as YCrCb where Y represents the luminance information and Cr and Cb are color difference signals that represent the chrominance information. In these systems all of the Y samples are used but every other color sample is dropped. These systems are referred to as 4:2:2 sampling. The 4:2:2 nomenclatures signify that for every 4 Y samples only 2 Cr and 2 Cb samples are saved. Owing to the bandwidth saving benefits of these different image formats different video equipment will adopt different color space encodings. Interoperability between such equipment often requires a device to convert the output of one video device in a given color space to the color space needed as input for the down stream device. Some examples of color space conversion are the converting of the RGB video output from a computer VGA card to YCrCb input on a TV monitor [6, 13]. The opposite conversion path is also common where a video device such as a DVD player outputs YCrCb and the video needs to be converted to RGB to drive a monitor [13].



Fig. 20.2 Matlab implementation for `rgb2ycbcr`

20.3.2 *YCbCr Color Model*

YCbCr color model also belongs to the family of television transmission color models. In this color model, the luminance component is separated from the color components. Component (Y) represents luminance, and chrominance information is stored as two color-difference components. Color component Cb represent the difference between the blue component and a reference value and the color component Cr represents the difference between the red component and a reference value. The following conversion is used to segment the RGB image into Y, Cb and Cr components: The conversion matrix can be expressed as in Eq. (20.1) [13].

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.5 \\ 0.5 & -0.419 & -0.081 \end{bmatrix} * \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} \quad (20.1)$$

Among all the color models found, YCbCr seems to be better for skin detection since the Colors in YCbCr are specified in terms of luminance (Y channel) and chrominance (Cb and Cr channels). The main advantage of converting the image from RGB color model to the YCbCr color model is the influence of luminance can be removed during our video processing. Figure 20.2 shows the conversion of a RGB color model in to a YCbCr color model implemented with the function `rgb2ycbcr` from Matlab.

20.4 Implementation Results, Simulation and Comparisons

20.4.1 *Hardware Co-simulation*

Figure 20.3 shows the model that uses the top level HDL module and its Xilinx blokset for RGB to Y component. This model can be used for co-simulation.

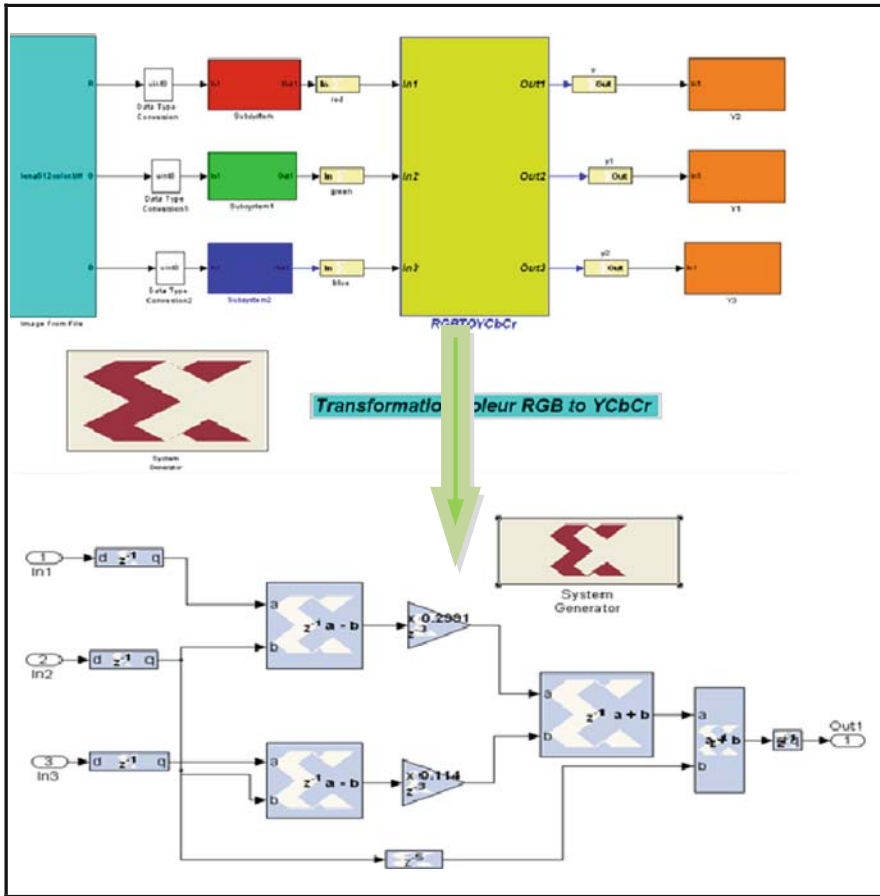


Fig. 20.3 System Generator project for simulation

Once the design is verified, a hardware co-simulation block can be generated, and then will be used to program the FPGA for the CSC design implementation. Figure 20.4 shows the model with the hardware co-simulation block. The bitstream download step is performed using a JTAG cable.

20.4.2 Simulation

After the co-simulation step the VHDL codes were automatically generated from the System Generator block sets. Behavioral and post simulation are supported by Mentor Graphics ModelSim tool (Fig. 20.5).

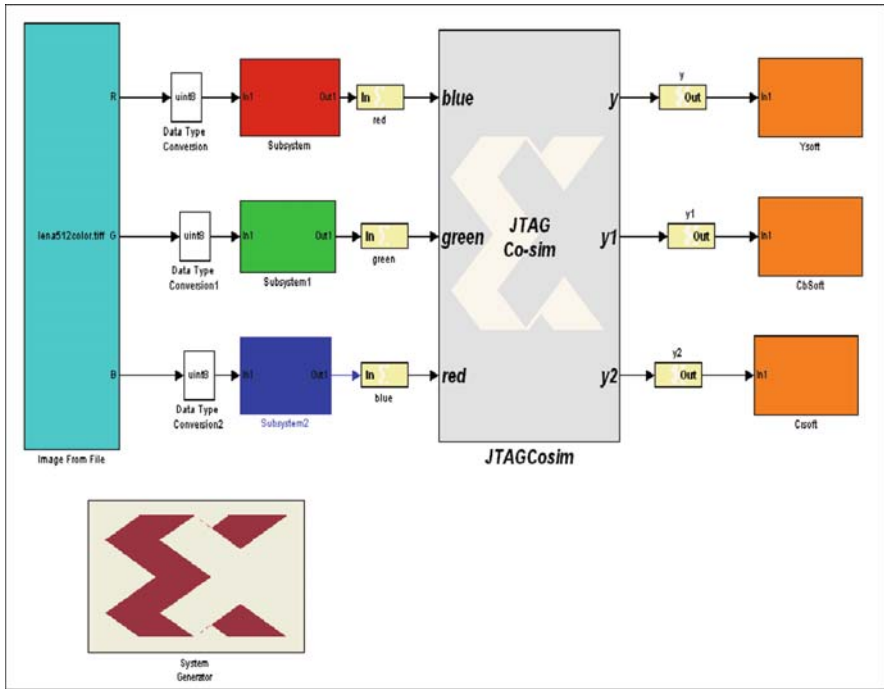


Fig. 20.4 System Generator project for hardware-in-the-loop testing

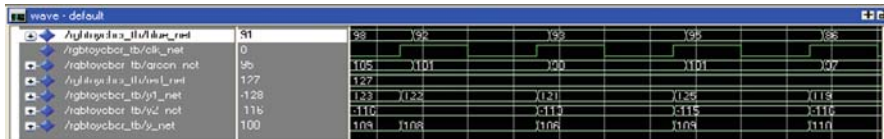


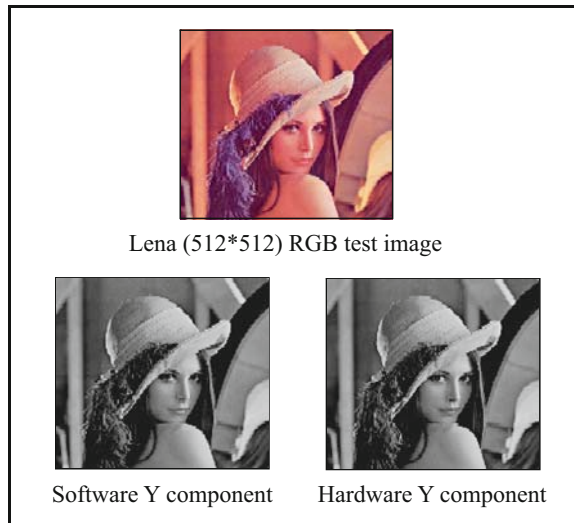
Fig. 20.5 Simulation results of the VHDL RGB to YCbCR conversion

The VHDL codes were then synthesized using Xilinx ISE 10.1i and targeted for Xilinx Spartan3 and Virtex II Pro family [2]. The optimization setting is for maximum clock speed. Table 20.1 details the resource requirements of the design. Note that in practice, additional blocks are needed for input/output interfaces, and synchronization.

The HDL-based circuit design flow is completed with the Xilinx ISE tool to perform synthesis, implementation, place & route and device programming for the whole design. For the arithmetic units, unsigned pipeline integer divider with both quotient and remainder output are parameterized and generated by Xilinx Core Generator tool [5, 9]. Multiplication uses the embedded multiplier in the hardware. The target FPGA chip is Xilinx Virtex II Pro xc2vp7-6ff672 and Spartan 3 xc3s200-5 ft256. During the Simulink-to-FPGA design flow, circuit modeling is built up

Table 20.1 FPGA resources used in the implementation for the CSC

	Spartan 3 xc3s200-5ft256		Virtex 2 Pro xc2vp7-6ff672	
	Available	Used	Available	Used
Number of slices	1,920	15%	4,928	6%
Number of slice flip flop	3,840	11%	9,856	4%
Number of 4 input LUTs	3,840	14%	9,856	5%
Number of bonded IOBs	173	43%	396	18%
Number of GCLKS	8	12%	16	6%
Maximum frequency	129.721 MHz		155.063 MHz	

**Fig. 20.6** Outputs from different implementations

with Simulink basic blocks and Xilinx specified blocks. Input and output data are combined with Matlab workspace, which is convenient to convert number format and debug. Figure 20.6 shows the software and hardware simulation for the CSC design for the input image.

20.5 Discussion

To provide a proper performance evaluation, the implemented CSC architecture using low cost available Spartan-II development system with Xilinx chip 2S200PQ208. The properties of other designs along with ours are listed in Table 20.2. As seen from this table, the design of the CSC proposed by [8] requires 380 CLB on the basis clock rate of 55.159 MHz.

Table 20.2 Performance comparison

	Our design		Design [8]	
	Available	Used	Available	Used
Number of slices	2,352	13%	2,352	16%
Number of slice flip flop	4,704	9%	4,704	7%
Number of bonded IOBs	140	53%	144	35%
Number of GCLKS	4	25%	4	25%
Maximum frequency	83.271 MHz		55.159 MHz	

On the other hand, our resulting architecture spent about 323 CLB with a working frequency up to 83.271 MHz. Obviously, our proposed architecture has lower complexity and improved efficiency in area, thus providing a good choice in terms of low-cost hardware.

From the development of FPGA technology, the methodology challenges the update of various EDA tools [11]. Based on the standard development flow, initial efforts have been transferred to high-level design and synthesis. There are many conversion tools such as C-to-FPGA, Stateflow diagram to VHDL Matlab-to-FPGA. The features of Simulink/Xilinx System Generator-to-FPGA [2, 4] flow can be discussed as follows.

References

1. Zemcik, P.: Hardware acceleration of graphics and imaging algorithms using FPGAs. SCCG'02: Proceedings of the 18th Spring Conference On Computer Graphics, pp. 25–32. ACM, New York (2002)
2. Xilinx System Generator User's Guide, www. Xilinx.com
3. Moctezuma, J.C., Sanchez, S., Alvarez, R., Sánchez, A.: "Architecture for filtering images using Xilinx system generator" World scientific advanced series in Electrical and Computer Engineering. Proceedings of the 2nd WSEAS International Conference on Computer Engineering and Applications, pp. 284–289 (2008)
4. The MathWorks Inc. Embedded Matlab Language User Guide (2007)
5. Ownby, M., Mahmoud, W.H.: A design methodology for implementing DSP with Xilin system generator for matlab. IEEE International Symposium on System Theory, pp. 404–408 (2003)
6. Han, D.: A cost effective color gamut mapping architecture for digital tv color reproduction enhancement. IEEE Trans. Consum. Electron. **51**(1), 168–174 (2005)
7. Bilal, M., Masud, S.: Efficient color space conversion using custom instruction in a risc processor. IEEE International Symposium on Circuits and Systems, pp. 1109, 1112 (2007)
8. Sapkal, A.M., Munot, M., Joshi, M.A.: R'G'B' to Y'CbCr color space conversion using FPGA. Wireless, mobile and multimedia networks, 2008. IET International Conference on Digital Object Identifier, pp. 255–258, 11–12 (Jan 2008)
9. Agostini, L.V., Silva, I.S., Bampi, S.: Parallel color space converters for JPEG image compression. Microelectron. Reliability **44**(4), 697–703 (April 2004)
10. Sima, M., Vassiliadis, S., Cotofana, S., van Eijndhoven, J.T.J.: Color space conversion for MPEG decoding on FPGA-augmented trimedia processor. Proceedings. IEEE International Conference on Application-Specific Systems, Architectures, and Processors, pp. 250–259 (June 2003)
11. Han, D.: Real-time color gamut mapping method for digital tv display quality enhancement. IEEE Trans. Cons. Electron. **50**(2):691–698 (2004)

12. A. Albiol, L. Torres, and E. J. Delp.: An unsupervised color image segmentation algorithm for face detection applications. In: Proceedings. 2001 International Conference on Image Processing, vol. 2, pp. 681–684 (2001)
13. Bensaali, F., Amira, A., Bouridane, A.: Accelerating matrix product on reconfigurable hardware for image processing applications. IEE Proceedings on Circuits Devices System **152**(3) (June 2005)
14. Kuchi, P., Gabbur, P., Bhat, S., David, S.: Human face detection and tracking using skin color modelling and connected component operators. IETE J. Res. (Special issue on Visual Media Processing) (May 2002)