Integrating Xilinx System Generator Simulink with ISE and HW / SW Co-synthesis using FPGA

Dr.G.L.Madhumati*, Mr.B.Muralikrishna*, Dr. Habibulla Khan**

*Professor & H.O.D, Dept. of ECE, Dhanekula Institute of Engg. and Tech, Ganguru, Vijayawada, India
+ Assistant Professor, Dept. of ECE, K L university, Vaddeswaram, Guntur, India
** Professor & H.O.D, Dept. of ECE, K L university, Vaddeswaram, Guntur, India.

*madhumati70@rediffmail.com, +boppana.muralikrishna@gmail.com, **habibulla@rediffmail.com

Abstract— System Generator is a system-level modelling tool that facilitates FPGA hardware design. It extends Simulink in many ways to provide a modelling environment that is well suited to hardware design. It provides a system integration platform for the design of DSP FPGAs that allows the RTL, Simulink, MATLAB and C/C++ components of a DSP system to come together in a single simulation and implementation environment. This paper proposes a black box model that allows RTL to be imported into Simulink and co-simulated with either ModelSim or Xilinx ISE Simulator. It can be integrated with Xilinx System Generator - Simulink platform in VLSI, based on FPGAs is presented. Hardware/Software Co-Synthesis platform with System Generator makes it possible to incorporate a design, implemented on a Xilinx Spartan3E FPGA.

Keywords—System Generator, Simulink, Hardware / Software Co-Synthesis, RTL, FPGA.

I. INTRODUCTION

Platform FPGAs have become key components for implementing high-performance digital signal processing (DSP) systems, especially in digital communications, video, and image processing applications. FPGAs have memory bandwidth that far exceeds that of microprocessors and DSP processors running at clock rates two to ten times faster, and unlike processors, Platform FPGAs possess the ability to implement highly parallel custom signal and image processing architectures [1]. One of the main impediments to wider adoption of FPGAs for image processing has been the relative unfamiliarity with FPGA technology. Converting RTL, Signal Processing, Image Processing algorithms to FPGA implementations can be tedious. The algorithm may be proven in software but with no direct link to actual implementation. Additionally, it can be difficult to subjectively verify the implementation [3]. Using a mathematical simulator to verify and create HDL implementation files bridges the gap from the algorithm architect to the FPGA engineer. Xilinx System Generator for DSP allows for high-level mathematical verification and converts the heart of the algorithm into ready-to-use HDL. Simulation inside of The Simulink tool enables to easily verify the RTL algorithm qualitatively and subjectively when used with the Toolbox. Using System Generator to develop and implement image processing algorithms allows for a thoroughly verified and easily executed design. The organization of the is as follows. Section II and III explores the features of Xilinx system generator, model based design with simulink HDL coder. Sections IV and V presents HDL code generation and design methodology. The remaining sections shows the implementation results obtained.

II. XILINX SYSTEM GENERATOR

The Xilinx System Generator allows to move an algorithm into an FPGA [2]. It extends the capabilities of the Simulink system-level simulation environment with bit and cycle-true modelling of an FPGA circuit. System Generator simultaneously provides access to key features in the FPGA fabric, including shift register logic, distributed and block memory, and embedded multipliers. System Generator includes a Simulink library of functional blocks for building DSP, arithmetic, and digital logic circuits. These polymorphic blocks compute their output types based on their inputs, although alternatively, they can specify their quantized output types explicitly [4].

Fig.1 System Generator Flow

It combines Xilinx blocks with MATLAB and Simulink blocks to create a realistic test bench and to analyse data computed by user model. The high level of abstraction provided by System Generator greatly simplifies algorithm development and verification. In addition to a system-level modelling library, System Generator includes a code generator that automatically generates a synthesizable VHDL netlist from your Simulink model [6]. This netlist includes IP (intellectual property) blocks that have been carefully
designed for high performance and density in Xilinx FPGAs. System Generator creates project and constraint files to assist implementation using the Xilinx Foundation as the major synthesis tools [5]. The System Generator Flow is shown in Fig.1.

III. MODEL BASED DESIGN WITH SIMULINK HDL CODER

HDL Coder supports code generation for Simulink models constructed with a combination of blocks from Simulink and Xilinx-specific blocksets from System Generator. The System Generator Subsystem block in HDL Coder enables you to include models built with System Generator in Simulink as subsystems. HDL Coder uses System Generator to generate code from the subsystem blocks and integrates the complete design into synthesizable HDL. This approach enables designer to:

- HDL Coder area and speed optimizations for Simulink components.
- Leverage Xilinx-optimized blocks in System Generator.
- Reuse legacy models built with Xilinx-specific blocks for simulation and code generation.

IV. HDL CODE GENERATION

Using System Generator, designer can automatically generate HDL code for FPGAs from Simulink models. User can create models of algorithms for implementation in FPGAs using high-level components from Xilinx-specific blocksets. Xilinx libraries include blocks for communication, control logic, signal processing, mathematics, and memory [7]. Using model based design, user can compile the design to an HDL netlist ready to be processed by a synthesis tool and automatically implemented with Xilinx ISE Design Suite. System Generator produces the following:

- Project files and constraint files for Xilinx ISE Design Suite
- VHDL / Verilog Test bench for functional verification

V. DESIGN METHODOLOGY

An RTL Algorithm is developed to display RGB Pixel Square Boxes in VGA Display. Later the design is invoked in Simulink through Black Box HDL in System Generator Platform. Delay and reference Blocksets are added using Simulink Tool Box. Synthesis, Bitstream generation and HW/SW Cosimulation are performed. ISE Project Navigator provides access to many advanced FPGA implementation tools, such as Constraints Editor and Timing Closure Tools that designer can use while implementing their embedded system module. Based on that application, FPGA design will fall into one of the following two situations:

1. FPGA design comprises a combination of an embedded processor sub-system and other custom logic, model based design with Simulink HDL (system Generator) in which case user must use Project Navigator to develop the other logic portion of design. Use Project Navigator to also integrate various modules and implement the top-level FPGA design.

2. In this case, designer can select to use Project Navigator for implementation, or they can implement entire design within System Generator. Use the System Generator and ISE tools to process their module design as follows:

A. Top-Down Design Methodology

Invoke ISE and create a top-level project. Then create a new Model-Based Design with System Generator Simulink source to include in the top-level design. This automatically invokes system generator, where designer develop their system module.

B. Bottom-Up Design Methodology

Invoke System Generator Simulink platform and develop their system module. Later, invoke ISE module as a source to include with ISE project. The two different hardware modules had been developed, one module for Simulink - System Generator Design Environment. Another hardware module for Custom HDL Logic designed in ISE Environment. This paper presents an evaluation of Both Top-Down and Bottom-Up Design Methodology.

Nowadays, Field Programmable Gate Arrays supporting reconfigurability and parallelism with very high performance have entered the market, making them ideal for prototyping of computationally intensive complex algorithms. They offer many advantages with respect to their non reconfigurable counterparts such as the general purpose microprocessors and DSP processors [8]. For this purpose, an FPGA based Hardware-Software (HW-SW) co-simulation methodology is adopted in this work which promises accurate and rapid prototyping along-with faster simulation times for heterogeneous systems design by introducing hardware (FPGA) within the simulation loop, as shown in Fig.2.

Fig.2 FPGA Based Hardware-Software Co-Simulation Environment

VI. IMPLEMENTATION

This Paper is implemented in Xilinx Spartan3E Starter FPGA Board (Spartan3E-FPGA, XC3S500E-Device Family, FG320-Package, and Speed Grade,-4). General Purpose Input Output
Peripheral Devices like, DIP Switches, Push Buttons, LED’s, Seven Segment Display, LCD Display, VGA, SRAM, ADC, DAC, and PS/2 etc., are controlled through Serial Communication (UART) via RS-232. It can be designed using MicroBlaze Processor in EDK and Model Based Design in System Generator Simulink Platform Design Environment.

VII. SYNTHESIS RESULTS
VIII. PHYSICAL VIEW OF IMPLEMENTATION RESULTS

IX. CONCLUSION

Modern FPGAs are capable of implementing high performance DSP functions, but the relative unfamiliarity of many DSP engineers with hardware design has slowed their wider adoption. Xilinx system generator has a unique hardware in the loop co-simulation feature that allows designers to greatly accelerate simulation while simultaneously verifying the design in hardware. The purpose of this paper is to demonstrate the flexibility of System Generator, as a multilevel platform which integrates DSP, HDL, EDK, and Image Processing to design a system.

X. REFERENCES

Xilinx, Inc., Virtex-II Pro Platform FPGA User Guide