Control system design methodology for SoC FPGA devices

Gabriel Santana Quintana
Institute for Applied Microelectronics (IUMA)
University of Las Palmas de Gran Canaria
Las Palmas de Gran Canaria, España
gsquintana@iuma.ulpgc.es

Pedro Pérez Carballo
Institute for Applied Microelectronics (IUMA)
University of Las Palmas de Gran Canaria
Las Palmas de Gran Canaria, España
carballo@iuma.ulpgc.es

Carlos Betancor Martín
Institute for Applied Microelectronics (IUMA)
University of Las Palmas de Gran Canaria
Las Palmas de Gran Canaria, España
betancor@iuma.ulpgc.es

***Abstract*— This paper proposes a methodology to design control system for SoC FPGA devices. The design flow is done with the MATLAB/Simulink and Vivado Design Suite tools. An implementation of a Dynamic Photovoltaic Array Reconfiguration (DPVAR) algorithm, together with a Maximum Power Point Tracker (MPPT) controller can validate the proposed design methodology. The controller will maximise the power output of a PV array under partial shading conditions. The designed control system is prototyped on the ZedBoard development board.**

Index Terms - SoC FPGA; FPGA; MPPT; DPVAR; Design Methodology

I. INTRODUCTION

The use of both FPGAs and SoC FPGAs in control applications is of great interest, as it is a flexible technology that allows the implementation of control algorithms for different industrial processes. Among the most prominent fields of application are robotics ([1], [2]), embedded controllers for industrial applications ([3], [4]), and controllers for power electronics and power generation ([5], [6]).

The use of FPGAs in robotics is increasing as robotic systems usually integrate a large variety and number of sensors, generating a large amount of data to be processed in real time. Within the field of embedded controllers, the most interesting applications are related to the automotive and aerospace industries. The controllers developed for these applications must follow real-time specifications, and therefore often require a high computational capacity.

The use of FPGAs in power electronics systems is also of special interest when a high degree of parallelism is required for the application.

II. DESIGN METHODOLOGY

The proposed design methodology is presented in Figure 1. It is based on three key verification techniques: Model-in-the-Loop (MIL), Software-in-the-Loop (SIL) and SoC-in-the-Loop (SCIL). SoC-in-the-Loop is a combination of Processor-in-the-Loop with FPGA-in-the-Loop to complete the verification of the system with a hardware/software codesign approach.

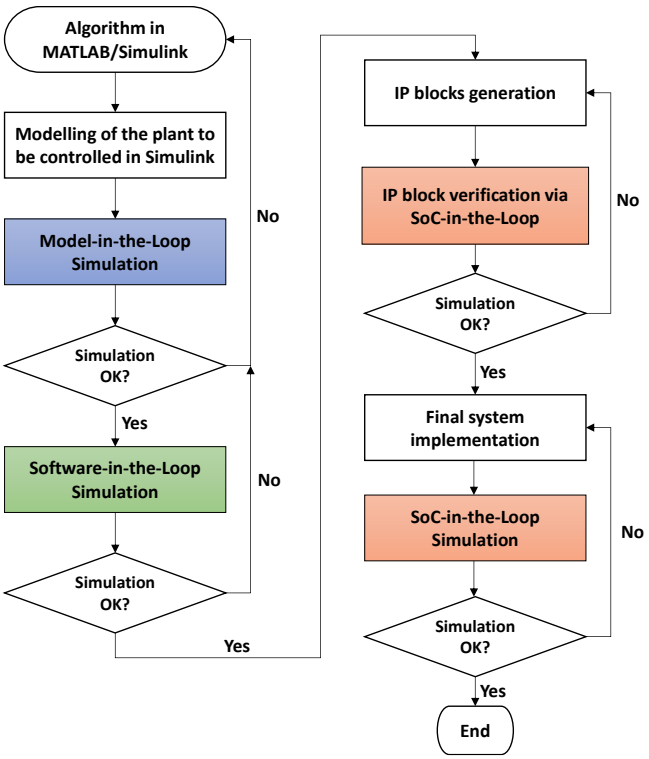

Figure 1. Design Methodology.

III. IMPLEMENTATION CASE

The implementation case chosen to verify the functionality of the design methodology is a Dynamic Photovoltaic Array Reconfiguration (DPVAR) algorithm together with a Maximum Power Point Tracker (MPPT) controller to improve the power output of a photovoltaic installation affected by partial shading. The DPVAR algorithm to be implemented in a SoC FPGA device is the MMTES as described in [7]. As for the MPPT controller, the incremental conductance [8] variant is chosen to be implemented.

A. MMTES Algorithm

Significative parts of this algorithm have been identified as suitable to be implemented on hardware and take advantage of its parallel computing capacity.

The MMTES algorithm can be applied to any size of PV array, with symmetrical or asymmetrical configurations. To be able to modify the connection of the PV panels, a switch matrix is required.

To implement this algorithm in MATLAB, it is divided into three main functions: **MMTESSort**, **PairForSwapping** and **SwapPair**. The performance of the MATLAB implementation is analysed to choose a suitable hardware/software partition for an implementation on SoC FPGA technology.

The measured execution time of the implementation of the MMTES algorithm in MATLAB for the four cases proposed in the MMTES algorithm paper, are shown in the Table 1.

Table 1. Execution time of the MMTES algorithm in MATLAB.

Case	Execution Time (milliseconds)			
	MMTES	***MMTESSort***	***PairForSwapping***	***SwapPair***
1	26	7	6	1
2	37	6	5	4
3	33	11	7	5
4	36	9	5	1

From this data, it is decided to make a hardware implementation for the **MMTESSort** and **PairForSwapping** functions and a software implementation for the **SwapPair** function and the rest of the MMTES algorithm. For the hardware implementation of the MMTESSort function, it has been implemented a Bitonic Sort algorithm, which is a hardware-oriented sorting algorithm, and for the hardware implementation of the **PairForSwapping** function, the original function has been modified to a parallelized version.

B. MPPT Controller

The incremental conductance method is based on comparing the incremental conductance `dI/dV` with the instantaneous conductance `I/V` and changing the output duty cycle according to the result obtained. The implementation of the incremental conductance MPPT controller is done in MATLAB based on the flowchart illustrated in Figure 2.

IV. SYSTEM MODELLING

The system is modelled in Simulink. The complete system consists of a total of eight modules. It is divided into two main subsystems: the hardware subsystem model and SoC FPGA subsystem, to implement the algorithm (Figure 3). The hardware part includes the PV array affected by partial shading, the switch matrix that allows altering the connection of the PV array, a Buck Converter that allows the MPPT controller to adjust the operating point of the PV system and a DC load. The MMTES algorithm, the incremental conductance MPPT controller, the auxCtrl function and the PWM generator are implemented in the SoC FPGA.

A. MIL/SIL Simulation

Two different loads have been used for simulations, including resistive and inductive ones, with different values. The MIL simulation calculates the operation of the algorithms implemented in MATLAB/Simulink with the PV system model.

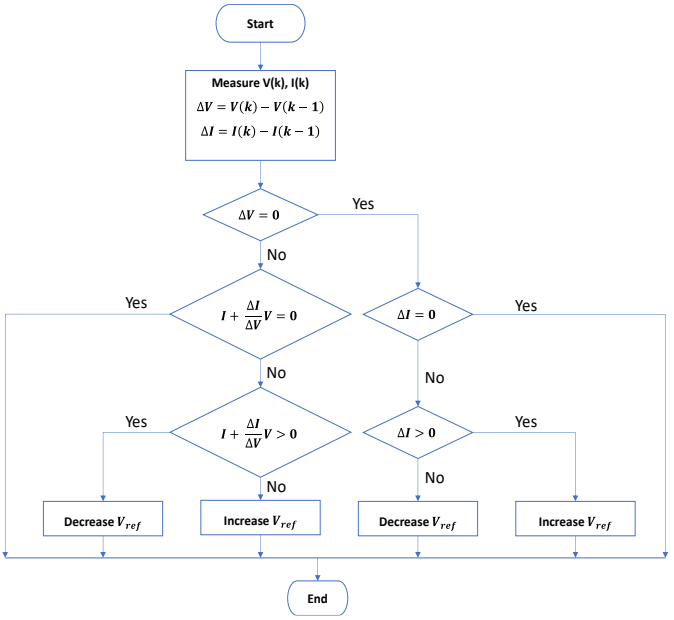

Figure 2. MPPT controller flowchart.

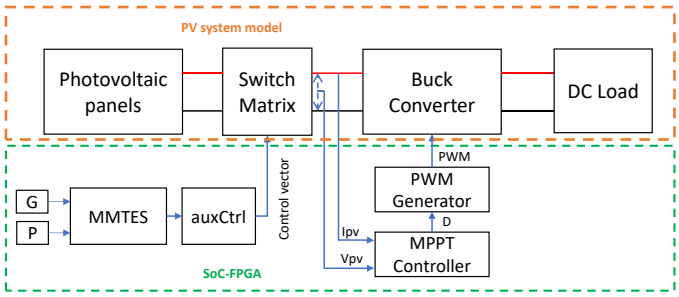

Figure 3. System block diagram.

In the SIL simulation, Simulink generates code from the implementation of the MMTES algorithm, the MPPT controller, the auxCtrl function and the PWM generator and simulates their operation together with the PV system model.

The Table 2 shows the output power values obtained with a resistive load in the MIL and SIL simulations. There is no difference in the output power obtained with both simulations.

Table 2. Output power obtained with resistive loads in MIL and SIL simulations.

Case	Pout (W)					
	MIL			***SIL***		
	R_1	***R_2***	***R_3***	***R_1***	***R_2***	***R_3***
1	823.5	829.1	828.9	823.5	829.1	828.9
2	789.7	801.1	801.3	789.7	801.1	801.3
4	793.6	794.7	796.8	793.6	794.7	796.8

V. DEVELOPMENT AND VERIFICATION OF IP BLOCKS

Using the HDL Workflow Advisor tool, IP blocks have been created from Simulink models. These IP blocks are designed to communicate with the SoC FPGA processor through an AXI interface. Once the IP blocks are created, a new model is automatically generated in Simulink called software interface in which the algorithm, for which the IP

block was generated, is changed to an AXI interface to write and read data to/from the blocks.

To verify the operation of the created IP block, a bitstream is loaded to program the FPGA. After this, a simulation is realized in external mode from Simulink (Figure 4). This simulation connects the development computer running MATLAB/Simulink with the prototyping board via Ethernet, generates an executable from the software interface model and launches it in the processor of the Zynq device, monitoring its operation. This allows to observe and verify the results from the IP block.

VI. IMPLEMENTATION AND VERIFICATION

A project is generated in Vivado Design Suite in which the generated IP blocks are integrated together with the PS (Processing System) of the Zynq mounted on the ZedBoard and other blocks necessary for the interconnection. The synthesis and implementation of the design is realized, obtaining the resource consumption illustrated in Table 3. After this, the bitstream is generated to program the PL (Programmable Logic) of the Zynq with the designed system.

The software for the designed system is generated from a Simulink model called ZedBoardModel. The MPPT controller and the MMTES algorithm are implemented in this model. The implemented algorithms are divided into two distinct parts: the software part and the hardware part. The software part implements the partitioning of the algorithms that are executed in the PS of the Zynq, i.e., in the ARM processor. The hardware part implements the partitioning of the algorithms that are executed on the PL of the Zynq, i.e., on the FPGA. In addition to the algorithms mentioned above, the designed system is equipped with UDP communication over Ethernet. This is necessary to complete the verification of the final system.

To verify the designed system, the PS of the ZedBoard is first programmed with the software generated from the ZedBoardModel.

After this, its operation is simulated together with SimscapeModel. This Simulink model contains the model of the system to be controlled and exchanges data during the simulation via UDP over Ethernet. This communication channel allows the voltage and current values of the PV array, the irradiance matrix and the position matrix to be sent to the ZedBoard during the SimscapeModel simulation. In the ZedBoard this data is used as input values for the MPPT controller and the MMTES algorithm. The duty cycle calculated by the MPPT, and the control vector calculated by the MMTES algorithm are transmitted from the ZedBoard back to SimscapeModel. In this way, by simulating SimscapeModel, the implemented system on the ZedBoard is verified.

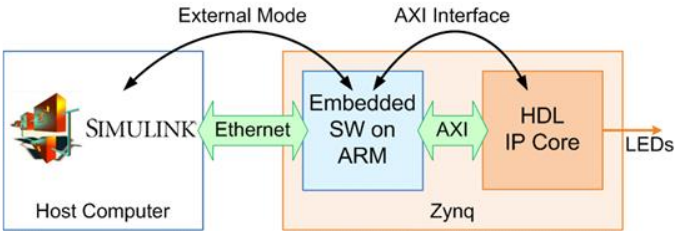

Figure 4. External mode simulation [9].

Table 3. Resource utilization.

	LUT	FF	DSP
BitonicSort	2975 (5.59 %)	606 (0.57 %)	0
MPPT_Inc	1743 (3.31 %)	244 (0.25 %)	6 (2.73 %)
PairForSwappingPll	733 (1.38 %)	496 (0.47 %)	0

Figure 5 shows the result of performing the SimscapeModel simulation together with the system implemented on the ZedBoard. This simulation corresponds to case study 1 with R_2 load. Figure 5 is divided into two distinct graphs. The upper graph compares the output power obtained during the SoC in the Loop simulation with that obtained in the MIL and SIL simulations. The lower graph shows the value of the difference between the output power of the SoC in the Loop and MIL/SIL simulations. The output power obtained with the SoC in the Loop simulation is almost identical to that obtained in the MIL and SIL simulations. The difference between the output powers varies between 0 and 10^{-4}.

During the verification process, measurements are taken of the runtime of the hardware/software implementation of the MMTES algorithm on the ZedBoard. To compare the runtimes, a new model is created in Simulink. This new model changes the hardware/software implementation of the MMTES algorithm to a purely software implementation. By simulating SimscapeModel with this new model loaded on the ZedBoard, the execution time of the pure software implementation of the MMTES algorithm running on the ARM processor of the ZedBoard can be measured. A comparison of the measured execution times is presented in Table 4.

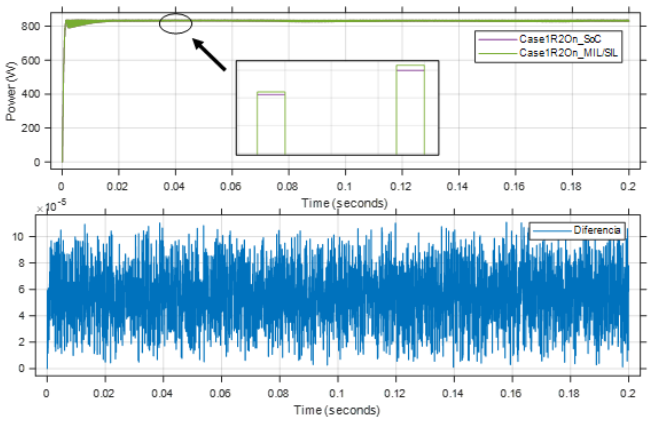

Figure 5. Simulation of SimscapeModel in conjunction with ZedBoardModel for case 1 with load R_2.

Table 4. Comparison of execution times for software and hardware/software implementation.

	Software (µs)	Hardware Software (µs)	Difference
PairForSwapping	3.59	7.60	+ 4.01
Sort	20.95	2.22	+ 0.27
Rest of the functions	29.30	9.49	-19.81
Total	53.84	38.31	- 15.53

The average execution time for hardware/software implementation is shorter than for pure software implementation, at around 29%. However, the software runtime of the PairForSwapping and Sort functions is shorter than that of the hardware/software implementations of PairForSwappingPll and BitonicSort. This is due to the Data Movement, as it takes time to transmit the data to the IP blocks in the PL of the ZedBoard.

The average execution time of the hardware/software implementation is shorter than that of the software implementation, even though the PairForSwappingPll and BitonicSort functions are slower in the studied cases, due to the execution times are more stable. The smallest runtime measured for the software implementation is shorter than the shorter runtime of the hardware/software implementation. On the other hand, the longer runtime of the software implementation is much longer than that of the hardware/software implementation. The runtime value of the software implementation varies, in contrast to the hardware/software implementation where it stays stable.

VII. CONCLUSION

The functionality of the developed implementation has been verified with a SoC in the Loop simulation with a development board. In addition to verifying the functionality, it has been checked that the hardware/software implementation of the MMTES algorithm shows improvements in terms of execution time and temporal stability compared to a purely software implementation. The average execution time of the MMTES algorithm has been reduced by 29%.

The functionality of the proposed design methodology is demonstrated through its implementation. Among the improvements it offers is the facility to perform the hardware/software partitioning of a design from a high-level and, based on this, to automatically generate code for the embedded software of the software partition and HDL code for the hardware partition.

Performing the design flow steps included in the presented co-design methodology from a single tool facilitates and speeds up the design process. This is a great advantage especially in cases where a short development time or the evaluation of different algorithm alternatives is required. In addition, the possibility offered by this methodology of using models created in Simulink during the verification process is an attractive feature, especially in the industrial environment where it can be complicated and costly to verify an implementation with the real system.

REFERENCES

[1] Z. Wan *et al.*, “A Survey of FPGA-Based Robotic Computing,” *IEEE Circuits Syst. Mag.*, vol. 21, no. 2, pp. 48–74, 2021, doi: 10.1109/MCAS.2021.3071609.

[2] G. Divya Vani, K. S. Rao, and M. C. Chinnaiah, “Self-Automated Parking with FPGA-Based Robot,” in *Micro and Nanoelectronics Devices, Circuits and Systems: Select Proceedings of MNDCS 2021*, T. R. Lenka, D. Misra, and A. Biswas, Eds. Singapore: Springer Singapore, 2022, pp. 459–470 [Online]. Available: https://doi.org/10.1007/978-981-16-3767-4_45

[3] M. Vyas, “Trends of FPGA use in Automotive Engineering,” in *2018 3rd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*, 2018, pp. 580–591, doi: 10.1109/RTEICT42901.2018.9012495.

[4] S. Di Cairano and I. V Kolmanovsky, “Real-time optimization and model predictive control for aerospace and automotive applications,” in *2018 Annual American Control Conference (ACC)*, 2018, pp. 2392–2409, doi: 10.23919/ACC.2018.8431585.

[5] Ó. Lopez, J. Alvarez, J. Doval-Gandoy, and F. D. Freijedo, “Multilevel Multiphase Space Vector PWM Algorithm,” *IEEE Trans. Ind. Electron.*, vol. 55, no. 5, pp. 1933–1942, May 2008, doi: 10.1109/TIE.2008.918466.

[6] H. Hatas, N. Genc, and A. Mamizadeh, “FPGA Implementation of SPWM for Cascaded Multilevel Inverter by Using XSG,” in *2019 4th International Conference on Power Electronics and their Applications (ICPEA)*, 2019, pp. 1–6, doi: 10.1109/ICPEA1.2019.8911189.

[7] G. H. K. Varma, V. R. Barry, R. K. Jain, and D. Kumar, “An MMTES algorithm for dynamic photovoltaic array reconfiguration to enhance power output under partial shading conditions,” *IET Renew. Power Gener.*, vol. 15, no. 4, pp. 809–820, Mar. 2021, doi: 10.1049/RPG2.12070. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1049/rpg2.12070. [Accessed: 05-Oct-2021]

[8] H. Deopare and A. Deshpande, “Modeling and simulation of Incremental conductance Maximum Power Point tracking,” in *2015 International Conference on Energy Systems and Applications*, 2015, pp. 501–505, doi: 10.1109/ICESA.2015.7503400.

[9] MathWorks, “Getting Started with Targeting Xilinx Zynq Platform - MATLAB & Simulink - MathWorks España.” [Online]. Available: https://es.mathworks.com/help/hdlcoder/ug/getting-started-with-hardware-software-codesign-workflow-for-xilinx-zynq-platform.html. [Accessed: 02-Nov-2021]