# Using System-on-Chip Boards for the Deployment of Controller for Verification and Prototyping

Adeel Jamal, Gerd Griepentrog
Technical University of Darmstadt
Fraunhoferstrasse 4, Darmstadt, Germany
Email: adeel.jamal@tu-darmstadt.de
URL: http://www.lea.tu-darmstadt.de

12.08.2022

## Acknowledgments

This work was funded by the German Federal Ministry for Economic Affairs and Energy under the support code 0324101.

## Keywords

≪System-on-Chip Boards≫, ≪PQ Controller≫, ≪Model-based Design Workflow≫, ≪Two-level Converter≫, ≪Controller Verification & Prototyping≫

## Abstract

New control strategies for power converters are mainly tested using Hardware-in-the-Loop (HIL) based rapid control prototyping (RCP) platforms in academia and industry research. In this paper, different strategies to use the system-on-chip boards for power electronics application are presented with pros and cons of each technique. The hardware software co-design work flow utilizing Intel and Matlab's toolchain is elaborately described. The proposed approach to implement a controller for power electronics application, that is cost-effective, offers more freedom for customization and will help decrease the product's time-to-market is presented. The introduced approach will be used to experimentally verify the performance of the exemplary PQ Controller designed for Grid-tied converter with LC filter.

## Introduction

Rapid Control Prototyping (RCP) platforms are used widely for testing, validation and prototyping of different controllers used in the drives of electric traction motors and Brushless DC (BLDC) motors for electric and hybrid vehicles [1]. Similarly, they are also employed in designing controllers for new converter topologies in medium and high voltage applications [2], [3] and DC-DC converter applications [4]. Depending on the manufacturer and hardware variant, RCP platforms are usually capable of running both Hardware-in-the-Loop(HIL) [5] and Processor-in-the-Loop(PIL) simulations [6]. dSpace$^{TM}$, Speedgoat$^{TM}$ and OPAT-RT Technologies$^{TM}$ are leading vendors for providing the RCP platforms but nearly all of the RCP control boards/platforms offered by them are quite expensive with less or close to zero freedom available for any hardware customization, for instance in terms of testing new communication protocols between power electronics' board hosting power stages or from ADC adapter card to control/DSP board. Vendors require customers to buy separate licenses for additional features such as new toolsets or analog/digital interfaces. In addition to buying the hardware for upgrading the performance of the processor of RCP platforms by appending more cores for high complexity controller design implementation, e.g. multiple step prediction based model predictive controller (MPC), customers are generally also required to pay for license for usage of extra cores which makes it rather more expensive

if an upgrade is needed. Also, the need of perpetual support from the manufacturer throughout the product's life cycle should not be underestimated. The time required to transfer the controller from RCP to the ASIC/FPGA chip is relatively high as the design first needs to be converted to HDL code which could be synthesized and compiled. The availability of high performance, high resource SoC boards offer an alternative for control design deployment which is cheap and reliable. This approach will also drastically reduce time-to-market as the SoC Chips for which the entire Model-based Design workflow is employed can be directly used in the embedded SoC boards for end products with the same workflow still valid for programming it. dSpace RCP platform like MicroLabBox also utilizes the Model-based design approach to streamline the process of C and HDL (Hardware Description Language) code generation and integration but certain processes within the workflow cannot be modified or enhanced. MicroLabBox is used in the industry for automotive and control applications [5], [7]- [8]. Nowadays, SoC chips are rapidly penetrating the commercial products arena where hardware accelerators are required for fulfilling real-time processing needs. Most of the researchers working on power electronics based solutions are not catching up with the advancements in embedded technologies such as using high speed SoC/FPGA systems rather relying on RCPs for time saving in prototyping of controller designs. This paper attempts to elaborate the deployment of controller on an Intel SoC-FPGA De1-SoC board which hosts Cyclone V SoC that is powerful enough to support complex controllers. There will be some effort to master the toolchain required to deploy the design but once it is mastered, the complete step of transferring the control design in RCPs to HDL can be circumvented. If needed, the performance can be upgraded by just replacing the SoC board while the toolchain and workflow would remain the same if the chip belongs to the same family or minor changes of setting up the interface if the chip belongs to a different family.
In the first section, the model of plant and the structure of power controller [9] will be elaborated. Second section details different strategies to program the SoC Boards that can be used in power electronic applications. The Intel's toolchain and the complete workflow used to program the Cyclone V SoC chip hosted on De1-SoC board will be discussed in the third section. Paper ends with the discussion of results using the adopted toolchain and workflow with a brief conclusion.

## Model of the Plant and Controller Structure

The plant consist of an LC filter that is directly coupled to the stiff grid voltage source via grid impedance $R_g$ and $L_g$ as shown in Figure 1. In general, the impedance of the grid at the Point of Common Coupling (PCC) varies depending on the configuration and connection scheme of the distribution grid at any particular instant of time. In the design and tuning of controller, grid impedance is assumed to be constant. Three phase DC/AC converter or Inverter applies a voltage at the inductor side of the LC-filter. Filter parameters are shown in Table I.

Table I: Chosen Parameters for LC-filter and Grid

| Component | Value |
|---|---|
| Filter Inductance ($L_\mathrm{f}$) | 850 µH |
| Filter Resistance ($R_\mathrm{f}$) | 0.01 Ω |
| Filter Capacitance ($C_\mathrm{f}$) | 25 µF |
| Grid Resistance ($R_\mathrm{g}$) | 10 mΩ |
| Grid Inductance ($L_\mathrm{g}$) | 80 µH |

The voltage across the capacitor can be considered as disturbance while deriving for the gains of the controller. Therefore, the transfer function of the plant is;

$$G_{\mathrm{plant}}(s) = \frac{i_{\mathrm{L_g}}(s)}{u_{\mathrm{in}}(s)} = \frac{1}{R_\mathrm{f} + sL_\mathrm{f}} \tag{1}$$

$$G_{\mathrm{plant}}(s) = \frac{\tau_\mathrm{f}}{L_\mathrm{f}(1 + s\tau_\mathrm{f})}; \tau_\mathrm{f} = \frac{L_\mathrm{f}}{R_\mathrm{f}} \tag{2}$$

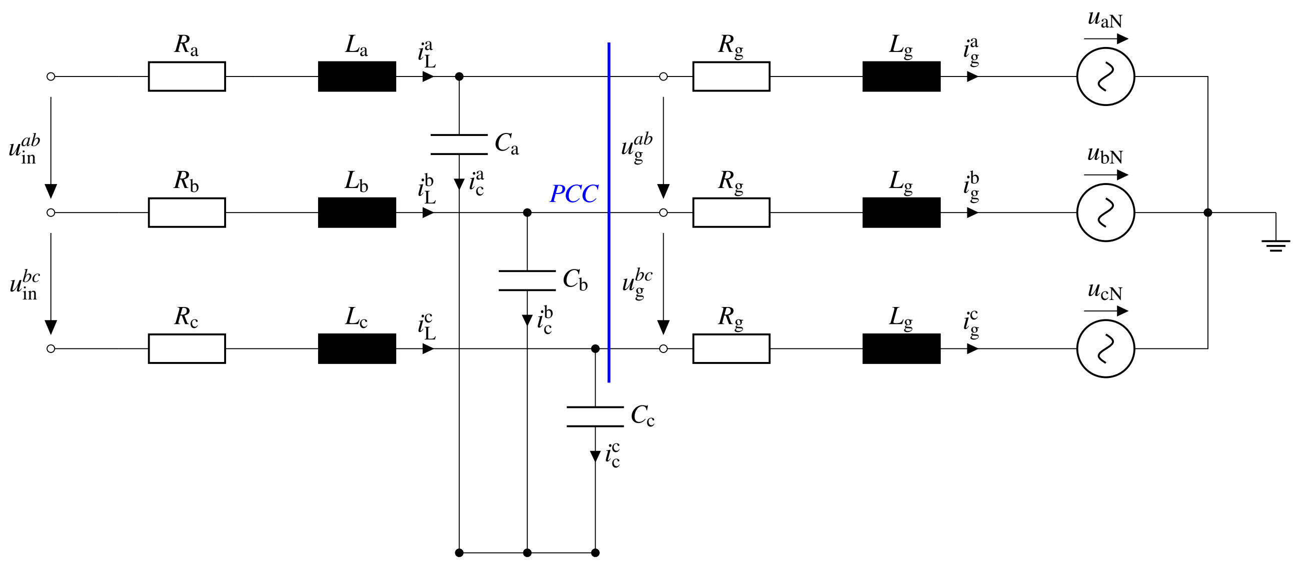

Fig. 1: Model of the plant consisting of LC filter and the Grid

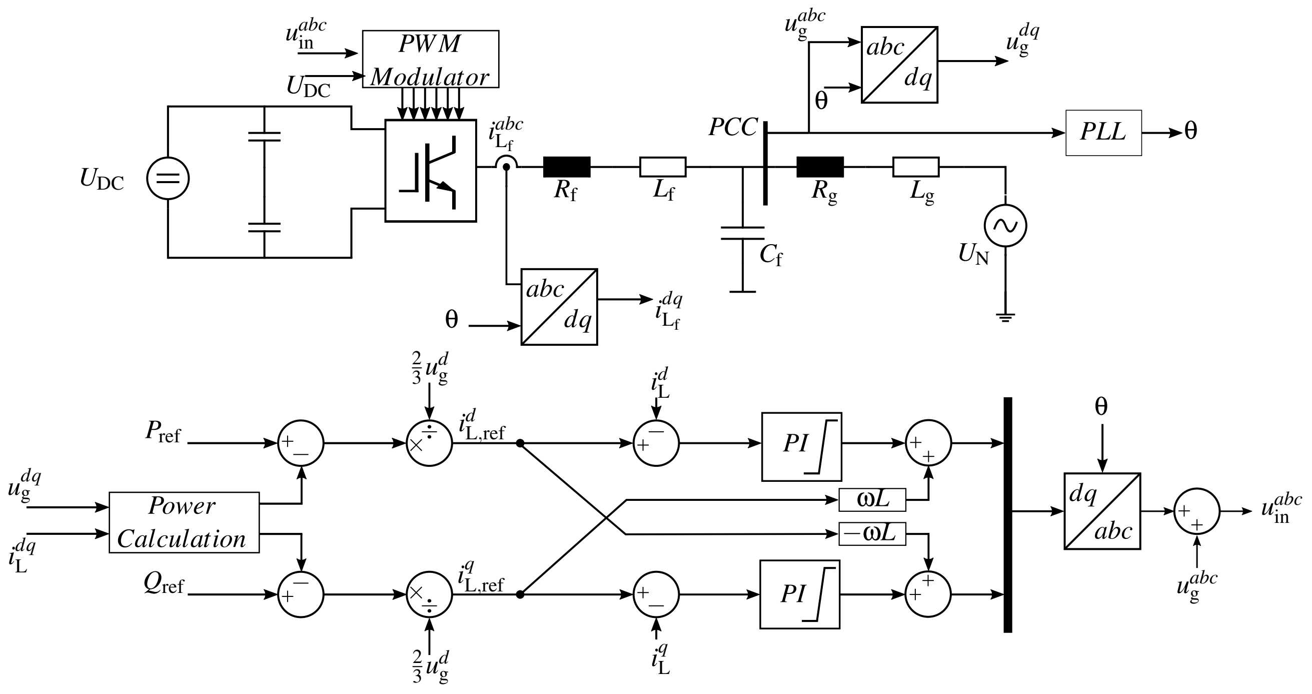

Fig. 2: PQ Controller Structure

The inverter can be modeled as a delay element;

$$G_{\text{delay}}(s) = \frac{1}{1+1.5sT_{\text{s}}} \tag{3}$$

Open-loop transfer function of the system including the PI controller with gain $k_{\text{c}}$ and time constant $\tau_0$ can be derived to get;

$$G_{\text{OL}} = \frac{k_{\text{c}}(s\tau_0+1)}{s\tau_0} \cdot \frac{1}{1+1.5sT_{\text{s}}} \cdot \frac{\tau_{\text{f}}}{L_{\text{f}}(1+s\tau_{\text{f}})} \tag{4}$$

Lastly, dominant time constant of the plant will be compensated by the controller according to the magnitude optimum criterion [10], hence time constant $\tau_0$ and gain $k_{\text{c}}$ becomes;

$$\tau_0 = \tau_f \tag{5}$$

$$k_{\text{c}} = \frac{L}{3T_{\text{s}}} \tag{6}$$

therefore the integral gain is;

$$k_{\text{c,i}} = \frac{L}{3T_{\text{s}}\tau_f} \tag{7}$$

The entire control scheme is being depicted in Figure 2. Inside the power calculation block in Figure 2, instantaneous power theory is applied to obtain the instantaneous power flows from the inverter to grid [11].

## Strategies for Programming System-on-Chip

System-on-chips available from different manufacturers offer different levels of customization options, FPGA resources, toolchains and programming workflows. Modern SoC architecture comprises of processor system and FPGA therefore, there are multiple strategies to program the SoC corresponding to the requirements of application. Following are the different strategies that have been explored, tested and implemented but the list is not exhaustive in nature:

1. Linux/RTOS or any other compatible OS runs on ARM processor and FPGA is used for controller implementation. C/C++/Python program can be written for Linux to communicate with the AXI / AVALON bus connecting processor system to FPGA. FPGA can be programmed using the VHDL/Verilog program written using open-source hardware accelerators and built-in IP cores available from the manufacturer. Lower-level hardware understanding of the address mapping of the FPGA in the Hard Processor System [12](HPS[1]) address space and knowledge of the provided APIs of the manufacturer is required to implement this approach.
2. Sometimes, the FPGA design is provided by the manufacturer for lower level control of converter such as monitoring of current, voltage and temperature but if the FPGA is completely self-programmed or the FPGA design for lower level control of converter provided by manufacturer is not chip-locked [2]; then matlab's co-design workflow can be adopted to generate the HDL code automatically from the Simulink design file through the Matlab's HDL coder library while at the same time, the ARM controller, run by Matlab Linux OS firmware, is programmed through the C code generated by embedded coder library of Matlab.
3. If the FPGA design is chip-locked then there are two possibilities;
    - Linux/RTOS running on both cores using Asymmetric Multi-processing (AMP). One core is used for monitoring and transmission of control and measurement data to the host main controller which also provides the set points. Linux is limited for the execution of different threads, and assigning their priorities to only one core and is managed by Linux kernel. The

[1]As referred in the technical reference manual of Cyclone V chip.

[2]which means lower-level control is provided as a design-partition which can be appended in the end of FPGA's final design

other core just sits idle unless it is directed by the Linux to start executing instructions at a specific address in SDRAM memory space. It must be ensured that the second core doesn't intrude into the other's memory space on SDRAM by mistake after the end of execution of its own program. Closed loop controller executes on the second core which means that, the communication between them is within the framework of Linux. Communication between the ARM processor and FPGA can be via SDRAM memory. The AMP is not straight-forward and can be quite challenging to implement on Cyclone V chip because of the following reasons:
   (a) Complexity associated with seperation of RAM, peripherals and caches between the two cores.
   (b) Difficulty in ensuring cache coherence during transfer of data between cores.
   (c) Lack of documentation or frameworks regarding AMP implementations on the Intel SoC FPGA (compared to OpenAMP for the Xilinx Zynq platform).
- One of the ARM cores is run by Linux, the other one is programmed in bare-metal[3]. Controller would be completely written in bare-metal and the delays are deterministic in nature which means the real-time execution can be guaranteed. Communication between the two cores can be on the principle of shared memory region based on interrupts. The shared memory has to be configured in Linux u-boot bootloader so that it doesn't deny the access to its memory region by default. The handling of communication and data exchange between the two cores is also not straightforward and it requires advanced considerations such as ensuring both programs running on distinct cores tend not to access the memory region at the same time hence it is recommended to be used if none of the other options is viable.

## Hardware Setup, Intel's Toolchain and Workflow

The commercial version of the Siemens' S120 converter (Two level, silicon semiconductor switch) is adapted by replacing the built-in controller hosted on factory-programmed AISC chip by De1-SoC control platform [13]. To implement the customized controller algorithm, De1-SoC control platform is connected to the power electronics of the S120 converter via an interface card used for voltage level translation and isolation between digital and power electronics circuit. De1-SoC comprises of Cyclone V SoC chip which hosts dual core ARM Cortex-A9 processor [14] alongside FPGA on a single chip. FPGA and processor system are interconnected through high-speed AXI-Avalon bus as shown in Figure 3.

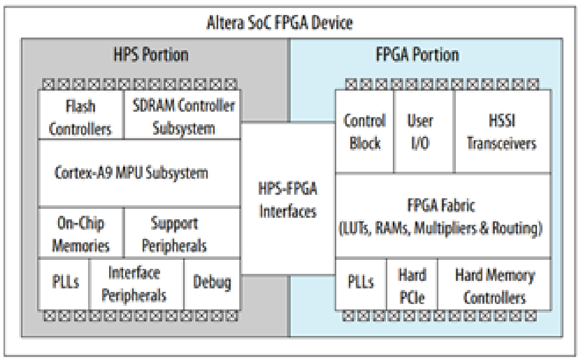

Fig. 3: Intel's SoC-FPGA Device Block Diagram [12]

The grid-connected converters demand real-time execution capability to avoid delays and other synchronization issues that could potentially lead to faults or temporary shutdown of the converter. Dual Core ARM Cortex-A9 is an application processor designed to be run using operating system (OS) by default [15]. Linux OS is configured to run on the processor in symmetric multi-processing mode i.e. the

[3]Bare-metal means that one has to program it by lower level c functions using APIs and library functions provided by the manufacturer.

Linux kernel and scheduler prioritizes the execution of threads on each core. Linux is not a Real-Time OS (RTOS) i.e. it doesn't guarantee a certain worst case maximum response time/latency to a triggering event. Similarly, Real-time patch of the Linux that is generally used to render some control on priority assignment of threads in Linux OS is also not so effective therefore the entire controller would be programmed on FPGA and all non-time critical blocks will be run on ARM processor using the MATLAB's hardware software (HW SW) co-design workflow concept.

The workflow that needs to be followed to program the FPGA and ARM processor separately is shown in Figure 4. For programming the FPGA, the HDL design must be finalized in Intel's Quartus software with the help of built-in IP Cores using Qsys/Platform designer. The generated .sof file can then be used to program the FPGA. Similarly, for programming the ARM A9 processor in bare-metal, ARM Debugger Studio 5 (DS-5) can be used. Intel's built-in hardware libraries and MATLAB's libraries can be utilized to generate the executable files for ARM processor. In Figure 4, a bird's eye view is presented, each step shown can entail further configurations and multiple sub-steps. The aforementioned entire toochain and workflow is handled by MATLAB if its HW, SW co-design workflow methodology is adapted. Following are the major steps that entails Matlab's co-design workflow;

1. Design the whole system that needs to be run on System-on-Chip in Simulink. All the blocks that are meant to be run on FPGA must be enclosed by a main subsystem block. All the blocks used must be compatible with HDL coder library for the HDL code generation. Where possible, HDL-optimized math operation blocks such as divide and square-root operation must be used as it ends up using less FPGA resources.
2. All the tasks and subsystems that are non-time critical e.g. monitoring systems must be placed outside the main subsystem block and it must be interfaced appropriately.
3. Using the HDL workflow advisor, the IP Core for the main subsystem block must be generated. The AXI bus interface code generation will be done by MATLAB if the specific pins of the SoC is defined for the main subsystem block interface.
4. Quartus project can be generated inside the HDL workflow advisor. The IP core can then be imported into the project followed by running analysis and synthesis to generate bitstream or .sof file.
5. Software interface model is to be generated with the help of HDL workflow advisor. This interface model can be used to interact with the System-on-Chip in MATLAB's real-time mode.

This is an iterative process and errors can appear at any stage of the compilation. It must be fixed step-by-step. It's possible that the design doesn't fit on the FPGA becasue of the over-usage of resources. There are different resource optimization techniques that can be employed to condense the design e.g. serialization of data. It should be kept in mind that improving the performance of the generated design is incremental and it takes time and many iterations to achieve a design that is final for deployment.

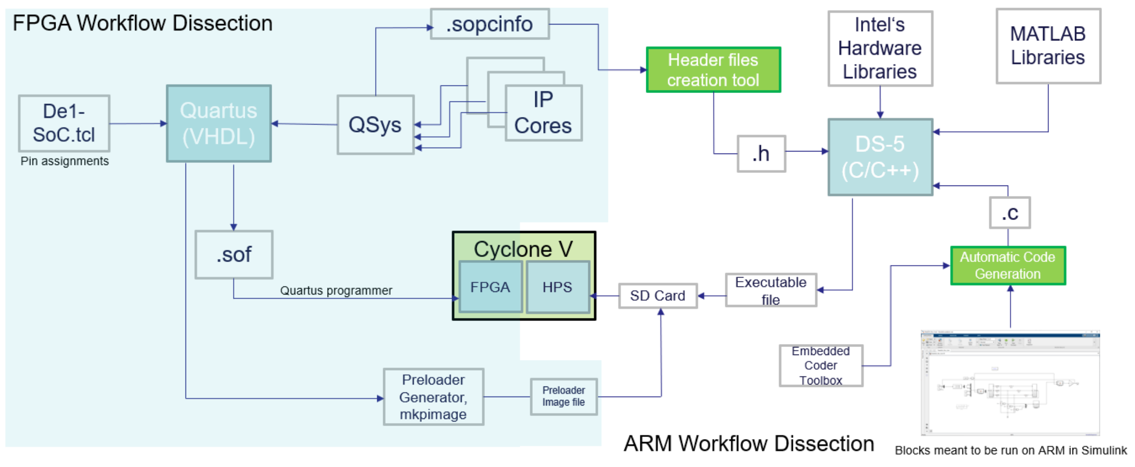

Fig. 4: Intel's SoC-FPGA Entire Toolchain and Workflow

## Results and Discussion

In this section, the result of adopting the described workflow and applying the Intel's toolchain to implement the PQ Controller structure will be presented. The PQ-controller is fully implemented inside FPGA. All the debugging and data probes transmit the data from the FPGA to the ARM processor via 120-bit wide data bus via AXI/AVALON protocol. In Figure 5, the experimental result of applying different setpoints to the controller is presented. Initially, the active (P) and reactive power (Q) setpoints are zero. Setpoints for P and Q of 10 kVA and 7 kVA are applied at t = 10 ms and t = 35 ms respectively. The measured active and reactive power follows the setpoint with some time delay dependent on the time constant of the controller. The output current though the filter's inductor and the voltage at PCC is also shown in the figure. The controller tries to regulate the active and reactive power with very little deviation and the small ripple present in the measured active and reactive power waveform is due to the switching nature of the converter. More complex controller's with extensive calculations like MPC can also be employed by using the described methodology. This approach provides a lot of freedom and flexibility to implement the controller in different ways using FPGA resources.

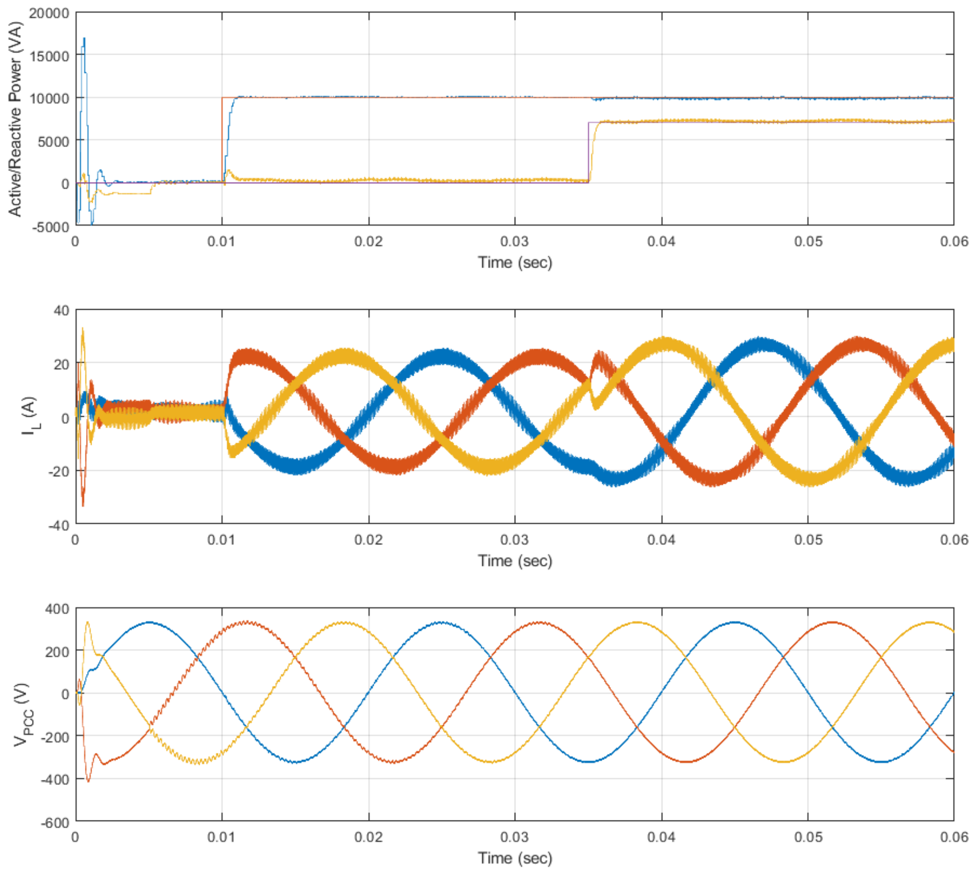

Fig. 5: Measured Active(P) and Reactive power(Q) following the reference

## Conclusion

The proposed method and workflow has several advantages. It saves a lot of time in design prototyping. When designing controllers for converters in microgrid application, it provides flexibility to change and modify the controller with very little marginal effort. The whole process of design doesn't require a deep knowledge of VHDL/Verilog or C language, only elementary knowledge is sufficient which is a huge advantage for novice. The disadvantages are following: Matlab HDL coder and embedded coder licenses are required. Built-in resource optimization techniques offered from Matlab and HDL Coder are not robust and stable in complex design e.g. feedback controller design. The complete cycle of finalization of design in Simulink, synthesizing the design files and compilation can be tedious; as in the case, the design gets large, the compilation time increases proportionally and therefore the feedback on errors made in the design will be slow. The entire design process is iterative; as it can happen that after finalizing the whole design and carrying out functional testing, the design doesn't fit on FPGA in the last stage of the compilation and fitting process.

## References

[1] H. Ross, Elektromobile Ideen verwirklichen, Published at: Elektronik automotive, 04/2012

[2] J. Dannehl, C. Wessels and F. W. Fuchs, "Limitations of Voltage-Oriented PI Current Control of Grid-Connected PWM Rectifiers With *LCL* Filters," in IEEE Transactions on Industrial Electronics, vol. 56, no. 2, pp. 380-388, Feb. 2009, doi: 10.1109/TIE.2008.2008774.

[3] Gholami-Khesht H, Davari P, Blaabjerg F. An Adaptive Model Predictive Voltage Control for LC-Filtered Voltage Source Inverters. Applied Sciences. 2021; 11(2):704. https://doi.org/10.3390/app11020704

[4] P. Karamanakos, T. Geyer and S. Manias, "Direct Voltage Control of DC–DC Boost Converters Using Enumeration-Based Model Predictive Control," in IEEE Transactions on Power Electronics, vol. 29, no. 2, pp. 968-978, Feb. 2014, doi: 10.1109/TPEL.2013.2256370.

[5] S. Khaldoune, P. -D. Maria, K. Abdelaziz and F. Maurice, "Hardware in loop methodologies for the control of dual-PMSM connected in parallel: FPGA implementation and experimentation," 2015 17th European Conference on Power Electronics and Applications (EPE'15 ECCE-Europe), 2015, pp. 1-10, doi: 10.1109/EPE.2015.7309461.

[6] Mangesh Kale, Narayani Ghatwai, Suresh Repudi, Processor-In-Loop Simulation: Embedded Software Verification & Validation In Model Based Development, Published at Design & Reuse

[7] T. Jaster, A. Rowe and Z. Dong, "Modeling and simulation of a hybrid electric propulsion system of a green ship," 2014 IEEE/ASME 10th International Conference on Mechatronic and Embedded Systems and Applications (MESA), 2014, pp. 1-6, doi: 10.1109/MESA.2014.6935601.

[8] M. Ruba, V. Chindris and D. Fodorean, "Design and experimental validation of a low voltage high current SRM for light electric vehicles," 2014 International Symposium on Power Electronics, Electrical Drives, Automation and Motion, 2014, pp. 118-123, doi: 10.1109/SPEEDAM.2014.6871910.

[9] B. Hammer, E. Lenz and U. Konigorski, "On the Design of Grid-Side Inverter Voltage Controllers," 2020 2nd IEEE International Conference on Industrial Electronics for Sustainable Energy Systems (IESES), 2020, pp. 227-232, doi: 10.1109/IESES45645.2020.9210662.

[10] D. Schröder, Elektrische Antriebe - Regelung von Antriebssystemen, 4th ed. 2015, isbn: 978-3-642-30096-7.

[11] Hirofumi Akagi, Edson Hirokazu Watanabe and Maur´ıcio Arede, "Instantaneous Power Theory and Applications to Power Conditioning", Second Edition, 2017, The Institute of Electrical and Electronics Engineers, Inc. Published 2017 by John Wiley & Sons, Inc.

[12] Cyclone V Hard Processor System, Technical Reference Manual, cv_5v4 2014.12.15, Altera 101 Innovation Drive San Jose, CA 95134

[13] https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=836

[14] https://developer.arm.com/ip-products/processors/cortex-a/cortex-a9

[15] ARM Cortex-A series Programmer's Guide, Version 4.0

[16] "Benchmark Systems for Network Integration of Renewable and Distributed Energy Resources", Final Report of Task Force C6.04.02, 2014