International Conference on Renewable Energies and Power Quality (ICREPQ'14)
Cordoba (Spain), 8th to 10th April, 2014
Renewable Energy and Power Quality Journal (RE&PQJ)
ISSN 2172-038 X, No.12, April 2014

Matlab/Simulink Generated FPGA Based Real-time HIL Simulator and DSP Controller: A Case Study

Z. Sütő[1], T. Debreceni[2], T. Kökényesi[1], A. Futó[1] and I. Varjasi[1]

[1] Department of Automation and Applied Informatics, Budapest University of Technology and Economics, Budapest, Hungary, E-mails: zoltan.suto@aut.bme.hu, tamas.kokenyesi@aut.bme.hu, andras.futo@aut.bme.hu, istvan.varjasi@aut.bme.hu

[2] Siemens Hungary, E-mail: tibor.debreceni@siemens.com

Abstract. The paper reports the development of a case study for creating a unified development environment for the process of code generation from Matlab/Simulink into both HDL and C languages of FPGA and DSP targets. The FPGA is used for the Hardware-In-the-Loop (HIL) simulation of the high power, main circuit of a simple solar-based battery charger while the control functions belonging to this system are implemented in a DSP. The case study is planned to be a building block of the HIL simulation of either such a sophisticated, cyber-physical system like a microgrid which connects various forms of energy produced by renewable sources.

Key words

HIL Simulation, FPGA, Rapid Prototyping, Power Electronics, Real-time Simulation

1. Introduction

Nowadays as the technology is developing, the more complex the power electronics and its controls become, the more proper hardware tool is required for HIL simulation. The general aim of the presented work is developing a fast, reliable and scalable HIL simulation framework for the rapid prototyping of complex power electronic systems. There are at least three strong reasons for using HIL simulation during the development: i) reduction of development time, ii) safety and quality requirements, and iii) prevent costly and dangerous failures [1].

The main concept of using HIL simulation in power electronic systems is that computational models replace the high-power parts of the system. The simulated parts are connected through real physical interfaces like analog and digital channels to the control boards under development, so the boards can be tested and validated in their seemingly real environment [2]. In HIL simulations the investigation of such practical issues like noise, sampling rate of the digital controllers, delay in communication channels, nonzero bias of analog inputs and outputs and so on, can be readily

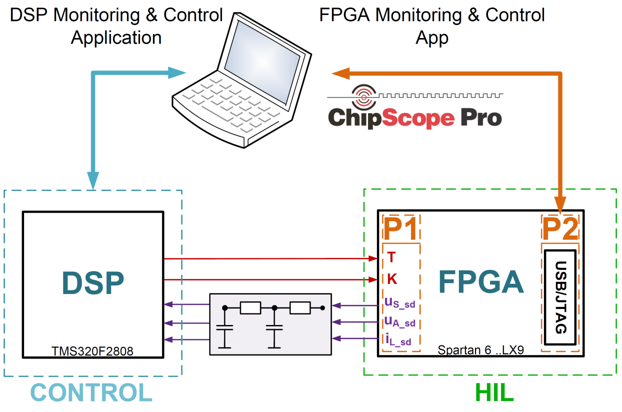


Fig. 1. The main block diagram of the HIL simulator based test environment of the DSP controller.

addressed.

Since power electronic systems exhibit extremely fast, usually quite complex and nonlinear dynamics due to the high-frequency switching action of the electronic switches, in this field nowadays the FPGA is becoming a possible candidate for HIL simulations of the main circuits and components of the system (e.g. grid, battery, 3-phase IGBT Voltage Source Inverter, Motor Model etc.) [3], [4], [5].

2. The Main Circuit Model

The block diagram of the HIL simulator based test environment of a DSP controller board can be seen in Fig. 1. Both DSP and FPGA boards are connected to a PC, and can be monitored and controlled by their HMIs. The whole case

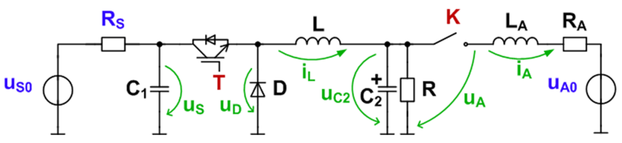


Fig. 2. Main circuit of the solar charger modeled in the HIL.

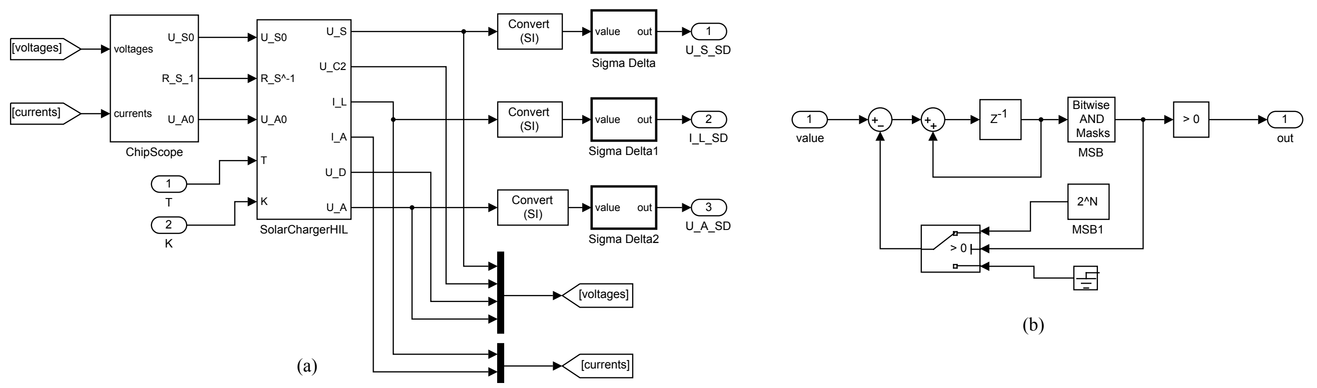


Fig. 3. (a) Subsystem of the HIL simulator. (b) Sigma-delta D/A converter.

study was designed for learning the process of generating code for both FPGA and DSP.

The modeled system is a simple solar-based battery charger. The schematic diagram of the main circuit can be seen in Fig. 2. The circuit is controlled by two logical input signals, the PWM signal T controls the IGBT and the signal K controls the DC contactor. The DSP controller measures three analog signals, the inductor current i_L, the voltage of the solar panel u_S and the battery voltage u_A, and based on these values and the charging current reference the controller fulfills the control actions. The solar panel is substituted with a simple Thevenin model of a series voltage source U_{S0} and its inner resistance R_S. U_{A0} is the no-load battery voltage which represents the actual state of charge of the battery. U_{S0}, R_S and U_{A0} are the main environmental parameters used during our real-time testing.

To create the Simulink model of this dynamic system suitable for generating the VHDL source files of the FPGA, firstly the functional description has to be done. The C_1 capacitor is charged up by the current from the Thevenin generator, thus the voltage of the C_1 will be the solar voltage u_S. Essentially the next part — with the IGBT, diode D, inductance L and the capacitor C_2 with the parallel discharging resistor R — represents a step-down (buck) converter; the voltage of the C_2 capacitor has to be held by control of the i_L current. By the control signal K of the DC contactor the battery is connected and charged by current i_A through the series inductor-resistor (L_A-R_A) of the inner impedance model of the accumulator.

The main circuit model was made by using state-space modeling [5, 6]. Let's see the dynamic equations according to the dynamic components in form of integral equations:

$$u_S = \frac{1}{C_1}\int_0^t \left(\frac{U_{S0}-u_S}{R_S} - Ti_L\right)dt \tag{1}$$

$$i_L = \frac{1}{L}\int_0^t (u_D - u_{C2})dt, \text{ where } i_L \geq 0 \tag{2}$$

$$u_{C2} = \frac{1}{C_2}\int_0^t \left(i_L - i_A - \frac{u_{C2}}{R_2}\right)dt \tag{3}$$

$$i_A = \frac{1}{L_A}\int_0^t (u_A - U_{A0} - R_A i_A)\,dt \tag{4}$$

where the diode voltage u_D can be expressed by the following piece-wise defined equation:

$$u_D = \begin{cases} u_S & \text{if} \quad T = 1 \\ 0 & \text{if} \quad T = 0 \text{ and } i_L > 0 \\ u_{C2} & \text{if} \quad T = 0 \text{ and } i_L = 0 \end{cases} \tag{5}$$

Here ideal switching behaviors are assumed for both the diode and the IGBT. Due to the DC contactor model an additional condition has to be fulfilled for the accumulator voltage u_A:

$$u_A = \begin{cases} u_{C2} & \text{if} \quad K = 1 \\ u_{C2} - U_{arc} & \text{if} \quad K = 0 \text{ and } i_A > 0 \\ U_{A0} & \text{if} \quad K = 0 \text{ and } i_A = 0 \end{cases} \tag{6}$$

U_{arc} denotes the voltage of the electric arc developing if the contactor is opened under non-zero current condition.

3. The HIL Simulator

The main block diagram of the HIL simulator can be seen in Fig. 3(a). It has logical input signals T and K coming from the DSP control board to the IGBT and the contactor. The DSP Control expects analog signals u_S, u_A and i_L from the HIL. However analog signals directly cannot be outputted by the FPGA, sigma-delta D/A converters were used [see Fig. 3(b)]. This D/A converter type is very similar to a PWM generator: the generated duty cycle at the output will be proportional to the input value. In order to reach the proportional analog value, analog filters (RC circuits) are required to connect the corresponding FPGA legs to the A/D channels of the DSP controller (Fig. 1). Note that, filters with higher cut-off frequency can be applied for sigma-delta signals than for PWM signals, so sigma-delta D/A converters are able to generate faster analog signals.

The Xilinx's ChipScope Pro is a utility tool for viewing and/or modifying inner signals in the FPGA at running time. By means of this utility HMI can be created to a design, for example to set parameters such as the actual insolation level of the solar panel by the U_{S0} voltage or the state of charge of the accumulator by its U_{A0} no-load voltage. The ChipScope subsystem placed in the model in Fig. 3(a) is an empty subsystem block, only its interface is defined by the input and output signals. The HDL architecture property of this subsystem block is defined as

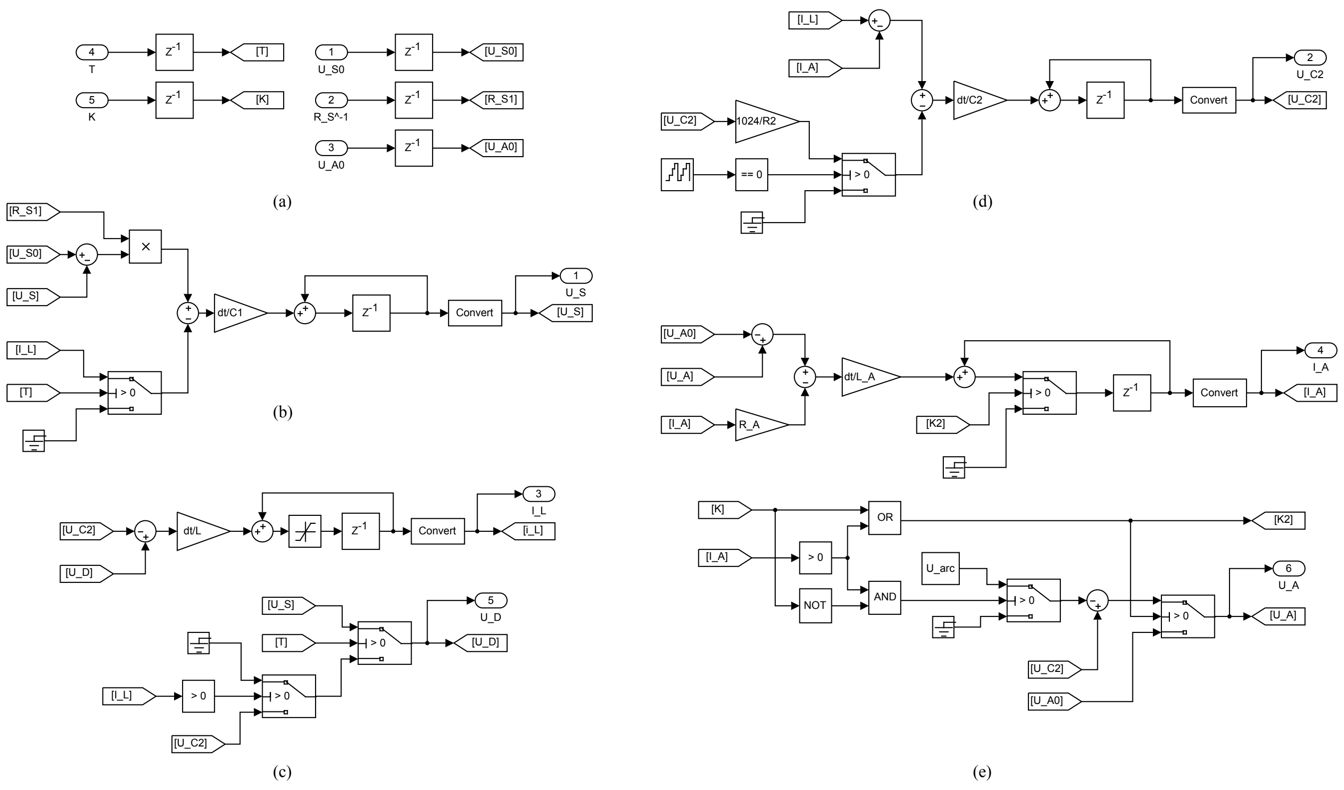


Fig. 4. (a) Initial zero-order hold of the input signals. HDL capable implementations of the integral equations of (b) the solar voltage u_S, (c) the inductor current i_L and the relating equation of the diode voltage u_D, (d) the output voltage of the buck-converter u_{C2} and (e) the accumulator current i_A and the relating equation with the contactor arc voltage U_{arc} [see eq. (1)–(6)].

BlackBox. It makes it possible to implement the block by self-written VHDL code. Using this option the ICON, VIO and/or ILA cores of ChipScope Pro can be included easily and the proper connections between them can be realized for creating the HMI. Note, that the BlackBox property cannot be set at top-level Simulink blocks, so the model in Fig. 3(a) was hidden into a top-level subsystem.

The dynamic model of the solar charger expressed by eq. (1)–(6) has been built up inside the SolarChargerHIL Simulink block. The inner structure of this block can be seen in Fig. 4. The model has to consist of only synthesizable elements, especially adder, subtractor, multiplier, delay, holder, conditional expression (switch) and general logical elements. The model was designed with fixed-point data formats. The inner values, e.g. the accumulator values of the discrete-time integrators are important to be handled in the system without data losses. There are Fixed-point tools in Simulink which supports the automatic selection of a fixed-point representation in the knowledge of the minimum and maximum values of the signals and the required precisions, but in our model the proper fixed-point representations for the signals have been set manually.

4. The DSP Controller

The targeted hardware of the DSP controller was a fixed-point Texas DSP.

A. *The control tasks*

The following main control tasks have to be solved by the controller:

Start-up: In the start up process the controller have to precharge the output capacitor of the converter to the battery voltage. The main contactor can be switched on just after this condition is fulfilled.

Charging: The controller has to ensure a controlled charging current for the battery. This is the normal operation of the system. The reference current is an input variable coming from an outer Battery Management System (BMS).

MPP tracking: If the demanded charging current of the BMS is larger than that the solar panel can provide, due to the low insolation level of the panel, than the system should be operated at a charging current with maximum allowable power. The system has to find and track this Maximum Power Point (MPP).

Standby: If the solar panel can provide so small power which is not enough to cover the losses of the charger, it has to go into a standby state. The contactor remains in close state, but the IGBT is turned off.

Of course, the DSP controller has numerous secondary tasks, e.g. ensuring fast overvoltage and overcurrent protections, providing HMI for the operator and so on.

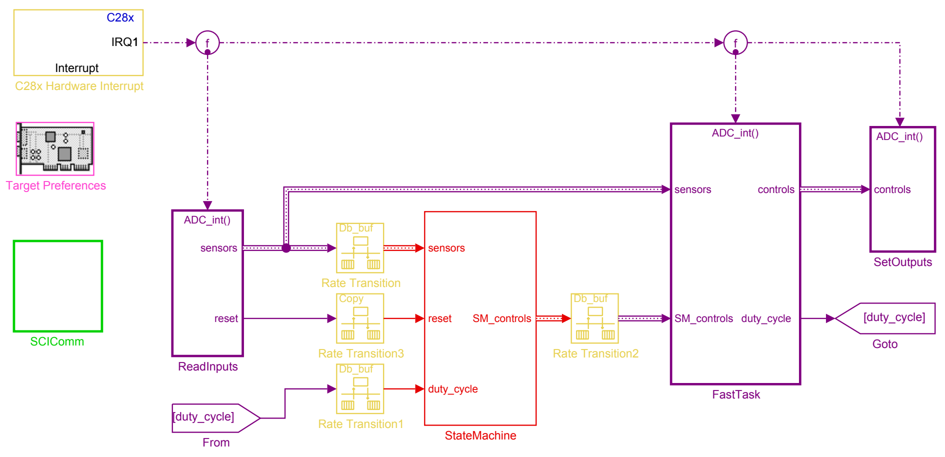


Fig. 5. Top-level model of the DSP Controller.

B. Priorities

The top-level model of the DSP controller can be seen in Fig. 5. The different colors denote tasks with different sampling rates. Faster tasks have higher priorities, and normally a slower task cannot interrupt a faster task. In our model the fastest task (purple) is assigned to the switching frequency of the PWM generator (125 μs). The PWM generator initiates the A/D conversions, and when the conversion has been finished an interrupt is generated (IRQ1). In the interrupt routine the function call subsystems: ReadInputs, FastTask and SetOutputs are executed in this order. The StateMachine (red) subsystem is running separately with 1 ms sampling time. The safe connections between tasks with different sampling rates are ensured by Rate Transition blocks.

C. Hardware dependent blocks

The hardware dependent and independent functions are well separated in the Simulink model of the DSP control. The hardware independent control logic covered by the FastTask and the StateMachines blocks can be easily moved to a test simulation environment for off-line evaluation and verification. There are three main hardware dependent block in the model:

ReadInputs block: It scales and converts the input signals obtained from the ADC outputs to fixed point representation with the proper width and fraction length. Furthermore binary signals coming from pushbuttons are handled, e.g. to reset or start the controller manually.

SetOutputs block: It has two main tasks: i) setting the required compare value (duty cycle) of the PWM block and ii) setting the binary output for controlling the main contactor. Additionally this block sets digital outputs (LEDs), e.g. to inform the operator if some overvoltage or overcurrent events happened.

SCIcomm block: The serial communication tasks with the host computer have been hidden in the SCIcomm (green) block, which is the third hardware dependent block. The tasks implemented in this block are repeated in every 100 ms. The block have no direct signal connections to other parts of the model. The parameters and variables, changed or viewed by the HMI running on the host computer, can be accessed directly writing or reading the proper memory addresses.

D. The FastTask block

One of the responsibilities of the FastTask block is the protections against overvoltages and overcurrents. If any of the measured voltages or currents reaches its maximum value the FastProtection block disables the two main outputs (the PWM and the contactor). The manual reset signal from a push button can enable the controller outputs. The reset will be valid only if all the protection events are eliminated, all the signals have already decreased below their limit values. The other responsibility of the FastTask block is the continuous current regulation. It normally ensures that the charging current or more precisely the current of the inductor follows the reference current prescribed by the outer StateMachine block. The inner structure of this current controller is shown in Fig. 6. It is a simple discrete-time PI controller with saturated output. The Mode input comes from the StateMachine block to switch between the closed loop and open loop control. In the latter case, the state machine directly defines the actual duty cycle for the PWM block, e.g. in the MPP tracking tasks.

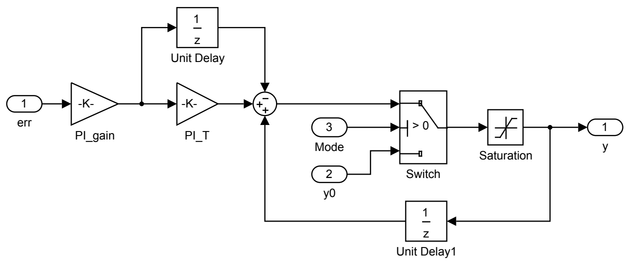


Fig. 6. The discrete-time PI current controller.

E. The state machine

The state machine summarizes well the overall operation of the DSP controller (Fig. 7). It is implemented by the StateFlow toolbox which makes it possible defining logi-

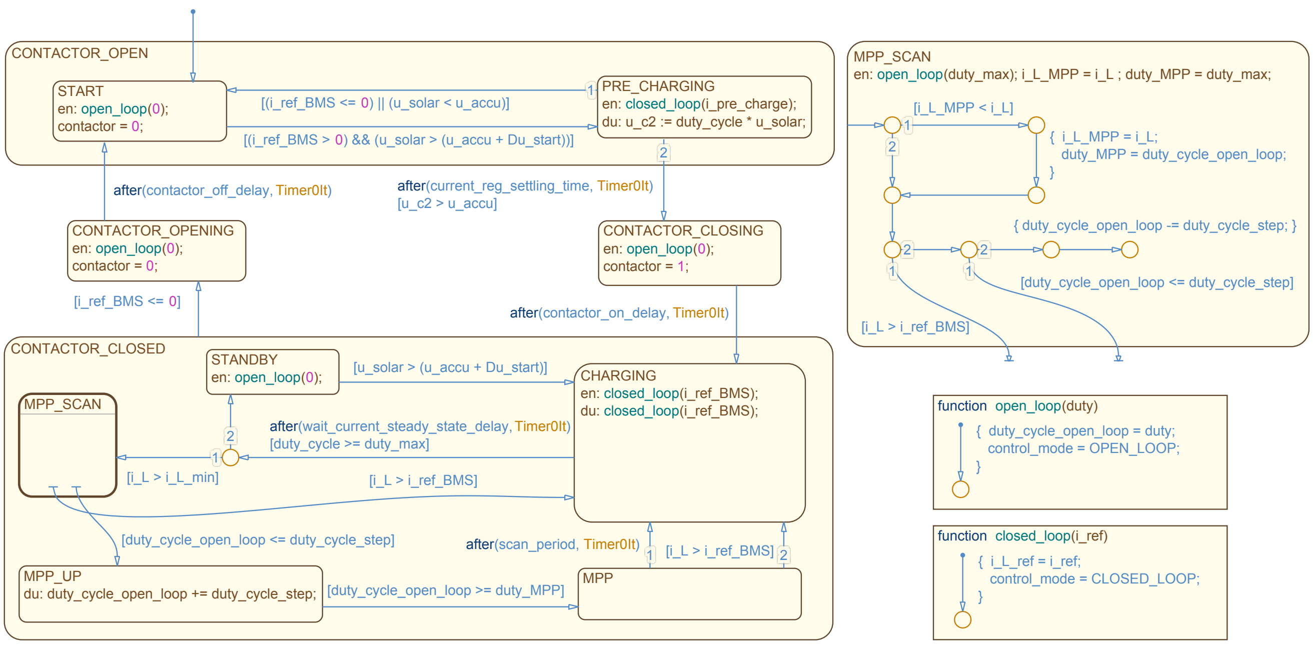

CONTACTOR_OPEN
START
en: open_loop(0);
contactor = 0;
[(i_ref_BMS <= 0) || (u_solar < u_accu)]
[(i_ref_BMS > 0) && (u_solar > (u_accu + Du_start))]
PRE_CHARGING
en: closed_loop(i_pre_charge);
du: u_c2 := duty_cycle * u_solar;
after(contactor_off_delay, Timer0It)
after(current_reg_settling_time, Timer0It)
[u_c2 > u_accu]
CONTACTOR_OPENING
en: open_loop(0);
contactor = 0;
CONTACTOR_CLOSING
en: open_loop(0);
contactor = 1;
[i_ref_BMS <= 0]
after(contactor_on_delay, Timer0It)
CONTACTOR_CLOSED
STANDBY
en: open_loop(0);
[u_solar > (u_accu + Du_start)]
CHARGING
en: closed_loop(i_ref_BMS);
du: closed_loop(i_ref_BMS);
MPP_SCAN
after(wait_current_steady_state_delay, Timer0It)
[duty_cycle >= duty_max]
[i_L > i_L_min]
[i_L > i_ref_BMS]
[duty_cycle_open_loop <= duty_cycle_step]
after(scan_period, Timer0It)
[i_L > i_ref_BMS]
MPP_UP
du: duty_cycle_open_loop += duty_cycle_step;
[duty_cycle_open_loop >= duty_MPP]
MPP
MPP_SCAN
en: open_loop(duty_max); i_L_MPP = i_L ; duty_MPP = duty_max;
[i_L_MPP < i_L]
{ i_L_MPP = i_L;
duty_MPP = duty_cycle_open_loop;
}
{ duty_cycle_open_loop -= duty_cycle_step; }
[duty_cycle_open_loop <= duty_cycle_step]
[i_L > i_ref_BMS]
function open_loop(duty)
{ duty_cycle_open_loop = duty;
control_mode = OPEN_LOOP;
}
function closed_loop(i_ref)
{ i_L_ref = i_ref;
control_mode = CLOSED_LOOP;
}

Fig. 7. The state machine Stateflow diagram.

cal type control in a more powerful way comparing to the other parts of Simulink. At power on the system enters into the Start state. When the outer BMS requests a charging current and the environment is good, that is, the insolation level of the solar panel is enough for charging, the controller pre-charges first the output capacitor of the buck converter with a small current. The output capacitor voltage is not measured, only estimated. The pre-charging current used is small, but large enough to operate the converter in continuous current conduction mode. Keeping this condition, the buck converter output voltage can be calculated as the multiplication of the duty cycle and the measured input solar voltage. When the pre-charging condition is fulfilled, the current controller is stopped, and the contactor is switched on. After a safety delay time, the state machine enters into the contactor closed state and start to manage the charging, maximum power point tracking and the standby states.

5. Real-time HIL tests of the DSP Controller

The overall case study with the FPGA and the DSP boards and with the interconnections between them is built up on an intelligent breadboard equipped with some instruments for measuring and generating signals (Fig. 8).

A. Closed-loop current regulation

The oscilloscope diagram in Fig. 9(a) shows the normal operating mode of the system, which is the closed loop current regulation. Typical signals measured between the HIL and the DSP are shown on the display in steady-state of the controller. The green curve is the PWM signal, the dark pink is the inductor current. The current ripple caused by the PWM can be observed well. The other two signals, the solar voltage (blue) and the battery voltage (orange) are practically constant at this time scale.

B. The start-up process

The pre-charging state and than the start of the current regulation is shown in Fig. 9(b). Since the time scale is large the oscilloscope averages out the PWM signal, so the green curve represents the duty cycle. The pre-charging current is just about 10 percent of the maximum current which was applied in the charging phase at the right hand side of the diagram. As the current reaches its steady-state and becomes almost constant, the duty cycle is increasing near linearly to the end of the pre-charging phase. It means that the effect of the load resistance at the output of the converter is negligible, the output capacitor of the converter is charged practically by a current source. A transient peak can be seen on the battery voltage when the contactor is closed. It happens just after the pre-charging state was finished, since the mechanical delay of the contactor is not modeled in the HIL. However the controller waits 50 ms for the contactor closing before it starts the battery charging. Entering into the charging state, the reference current jumps to a higher value and the solar voltage drops to a smaller value at the same time since the solar panel is loaded more heavily.

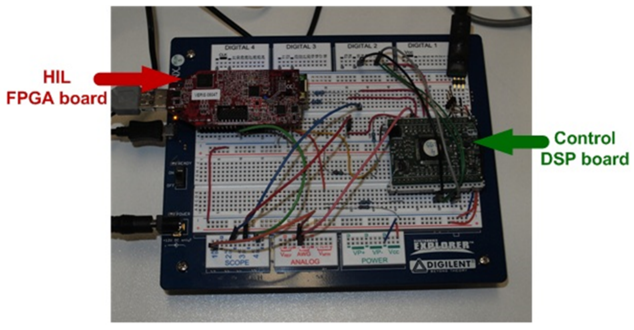
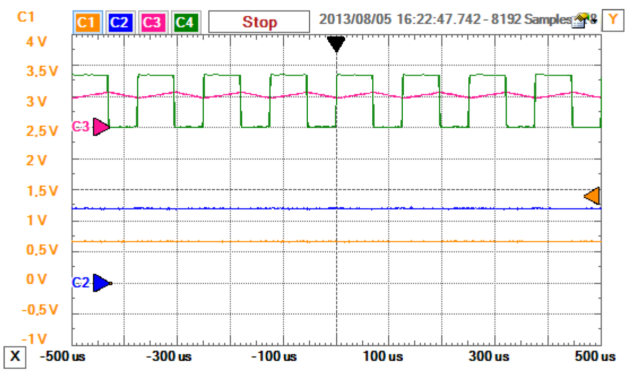
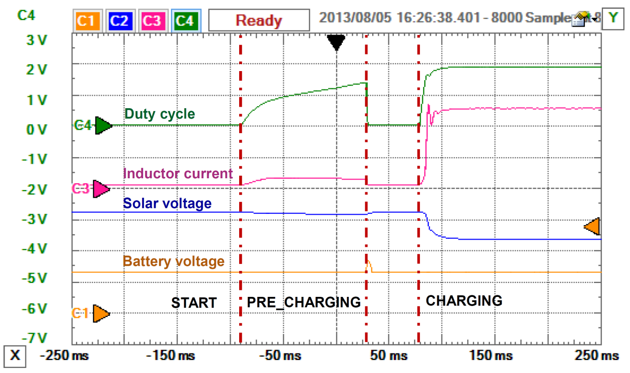


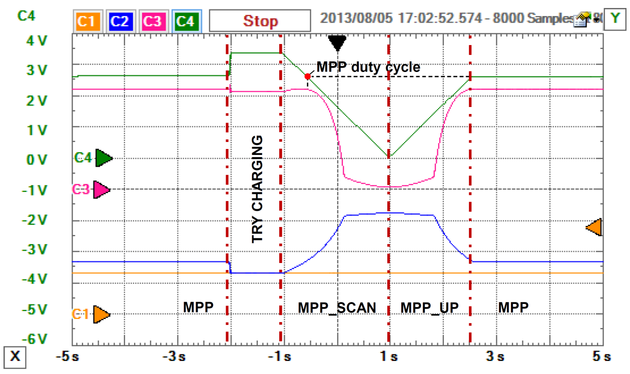
Fig. 8. Built-up on intelligent breadboard.



(a)



(b)



(c)

Fig. 9. Real-time hardware tests. (a) Typical PWM and current signals (Continuous Current Conduction). (b) The start-up process. (c) Scanning for Maximum Power Point (MPP).

C. MPP scan

The Maximum Power Point (MPP) is searched if the battery management system requires more charging current than the charger can provide. The full process of the MPP scan can be seen in Fig. 9(c). First the system tries to switch into current regulation mode. Since the reference current is larger than the current provided by the solar panel, the output of the current controller is saturated at the maximum duty cycle. It will happen almost abruptly at this time scale. Since the duty cycle is one, the solar voltage drops to the battery voltage in this phase. If the current controller cannot exit the saturation within 1 second, the controller is switched off, and the state machine starts the scan process, that is, starts decreasing the duty cycle in small steps in open loop control mode until the duty cycle reaches zero. In this scanning process the current will have a maximum value. The MPP duty cycle belonging to the maximum current is stored and after the scanning, the duty cycle is increased with the same slope to this stored MPP duty cycle. Since this is not a closed-loop operation and the environment, either the reference current or the insolation of the solar panel can change, the system tries to switch back to closed loop current regulation mode or try to find a better operating point in every 10 seconds.

6. Conclusion

The paper reported the development of a case study for creating a HIL simulation environment for power electronic systems. Generally It can be stated, that practically all components of the system can be built up using Matlab/Simulink. Programming codes for both the FPGA HIL simulator and the DSP Controller can be generated without a deep knowledge about either the FPGA development environment (Xilinx ISE) or the DSP development environment (Texas Instruments CCStudio). Both development processes from creating the models to generating the project files, building the binary codes for programming the devices and either the programming itself can be done from a unified environment in Matlab.

Acknowledgments

This work was partially supported by the Hungarian Government, managed by the National Development Agency, and financed by the Research and Technology Innovation Fund through project eAutoTech (grant no.: KMR_12-1-2012-0188). The authors wish to thank the support to the Hungarian Research Fund (OTKA K100275) and the Control Research Group of the Hungarian Academy of Sciences.

References

[1] T. Kökényesi and I. Varjasi, “FPGA-based real-time simulation of renewable energy source power converters,” *Journal of Energy and Power Engineering*, vol. 7, no. 1, pp. 168–177, Jan. 2013, ISSN 1934-8975.

[2] B. Lu, X. Wu, H. Figueroa, and A. Monti, “A low-cost real-time hardware-in-the-loop testing approach of power electronics controls,” *IEEE Transactions on Industrial Electronics*, vol. 54, no. 2, pp. 919–931, Apr. 2007.

[3] J. C. G. Pimentel and H. Le-Huy, “Hardware emulation for real-time power system simulation,” in *Proceedings of the IEEE International Symposium on Industrial Electronics (ISIE2006)*, vol. 2, Jul. 9–13, 2006, pp. 1560–1565.

[4] G. G. Parma and V. Dinavahi, “Real-time digital hardware simulation of power electronics and drives,” *IEEE Transactions on Power Delivery*, vol. 22, no. 2, pp. 1235–1246, Apr. 2007.

[5] T. O. Bachir, J.-P. David, C. Dufour, and J. Bélanger, “Effective FPGA-based electric motor modeling with floating-point cores,” in *Proceedings of the 36th Annual Conference of the IEEE Industrial Electronics Society (IECON2010)*, Glendale, Arizona, USA, Nov. 7–10, 2010, pp. 829–834.

[6] H. F. Blanchette, T. Ould-Bachir, and J. P. David, “A state-space modeling approach for the FPGA-based real-time simulation of high switching frequency power converters,” *IEEE Transactions on Industrial Electronics*, vol. 59, no. 12, pp. 4555–4567, Dec. 2012.