MATLAB Simulation of Scheming FPGA-Controlled Power Electronics and Drives

[1]B PATNAIK,
Gandhi Institute of Excellent Technocrats, Bhubaneswar, India
[2]TUSAR KANTI DASH,
Rayagada Institute of Technology and Management, Rayagada, Odisha, India

Abstract-We present a simple and rapid prototyping technique for Field Programmable Gate Array (FPGAs)-based digital controllers for power electronics and motor drives using MATLAB's Simulink and HDL Coder design software. The MATLAB/Simulink models are optimized and converted to target independent, specific and traceable Very High Speed Integrated Circuit Hardware Description Language (VHDL) code for FPGA programming. An example implementation of the space vector pulse width modulation (SVPWM) technique is presented, illustrating the design of a generic 3-phase voltage source inverter (VSI). Simulation and co-simulation, system level design, and verification for rapid prototyping of FPGA-based digital controllers will assist power electronics engineers and researchers to develop and prototypes in a relatively short time by eliminating tedious and time-consuming manual coding. This enables increased productivity and facilitates the development of power electronic controllers with more complex control algorithms.

Index Terms-FPGA, rapid prototyping, model based design, digital control, power electronics and drives, SVPWM.

I. INTRODUCTION

The design of modern power electronic circuits and systems requires knowledge from multiple discipline areas, including digital control, to develop innovative and custom-designed products and solutions in a short period of time [I]. MATLAB & Simulink enable an alternative way to automatically generate readable and portable IEEE standards-compliant HDL (i.e. IEEE 1076 compliant VHDL code and IEEE 1364-2001 compliant Verilog code) from MATLAB, Simulink and Stateflow models for a variety of FPGAs. Fig. 1 shows the various ways to generate HDL from MATLAB & Simulink.

Manual coding is tedious, time consuming and error prone. On the other hand, automatic code generation lets designers to make changes in the system level model, and produce an updated HDL implementation in minutes by regenerating the HDL code. Fig. 2 (from [2]) illustrates the comparison of model-based design using HDL coder and manual coding. Model-based design reduces the total project development time by 33% as compared to manual coding

. In addition, MATLAB model-based design facilitates creation of FPGA-based prototypes and automates HDL code verification by co- simulating it with simulink and optimizes the model to meet speed area power objectives for the FPGA.

The MATLAB environment provides two model-based tools for rapid system development: i) Xilinx System Generator and ii) HDL Coder. Either of these approaches provide an effective FPGA design flow when used independently. However, (as pointed out in [3]) "some projects benefit from a mixture of approaches – a workflow that combines the native Simulink workflow, device-independent or device-specific code, and code readability offered by Simulink HDL coder, with the Xilinx FPGA-specific features and optimizations offered by Xilinx System Generator." [3]

Model-based design in MATLAB & Simulink environmentfor FPGA prototyping is very flexible and makes implementation of control algorithms in FPGA for power electronics and motor drives a lot faster with no need of specialattention to internal connections in the device prototype. The prototype is used to verify various modulation strategies, control functions, and power flow regulation algorithms for various tailor made power electronic design and motor controlapplications in minimum time. Thus, by using an FPGA-based controller, the designer is able to build a fully dedicated digitalsystem that is perfectly adapted to the control algorithm beingimplemented.

Moreover, FPGA technology is now considered very useful by an increasing number of designers in myriad fields of application due to short implementation time, confidentiality of the algorithm and architecture, capable to meet many constraints for space applications, and it can be adapted to any change in design by dynamic reconfiguration [4]. Some of the benefits of using FPGA for controlling of electrical systems compared to counterpart DSPs and microcontrollers are made clear in the same paper.

Tn fact, FPGA-based digital controllers have been implemented with success in many different applications, such as power converters (e.g. PWM control of DC-DC converters) [5-6], point of load converter [7], pulse width-modulated (PWM) inverters [8-9], resonant inverters [10], power-factor correction [II], interleaved converters [12], multilevel converters [13], multilevel and matrix converters [14], fuzzy logic control of power converters and electrical drives (e.g. induction machine drives) [15-16], synchronous machine drives [17], neural network control of induction motors [16], and switched reluctance motor drives [18].

However, FPGA prototyping and design remains a mystery to many novice power electronics designers who do not have a basic knowledge of VHDL or Verilog coding. In the same context, even manual coding experts find it difficult to meet the short implementation times, due to the time required for debugging, modification of the control algorithms, and reimplementation and testing of the prototype. Model-based design on other hand has many advantages compared to manual coding and is easy even for novice designers using MATLAB and Simulink as an integral part of modern power electronic and control system design. Tn brief, using the model-based design, system architects and designers can spend more time on fme tuning the algorithms; modify models, hardware & software co-simulation for verifications, and experimentation and less time on learning and writing HDL code.

This paper aims to fill the gap between power electronics designers and FPGA-based controller implementation through system level model-based design where resource optimization has been included. Automatic generation of HDL Code is described briefly in Section II. SVPWM is implemented in Simulink using the HDL Coder toolbox and blocksets to demonstrate and design a 3-ф VST. Details of the design are presented step-by-step and finally the code is verified using co-simulation and full hardware implementation.

TT. CODE CONVERSION: MATLAB ISIMIJLINK TO VHDL CODE

HDL describes electronics circuits in terms of the circuit's operation, design, and tests to verify its operation by means of simulation. At the first step of code conversion process, the new design ideas and algorithms are represented in terms of mathematical models and are tested in MATLAB/Simulink floating point data types. However, implementation of control algorithms in FPGAs and ASICs require fixed-point data type conversion to reduce hardware resources. This conversion process often introduces quantization errors. As a consequence, a signal scaling and word-length optimization becomes a

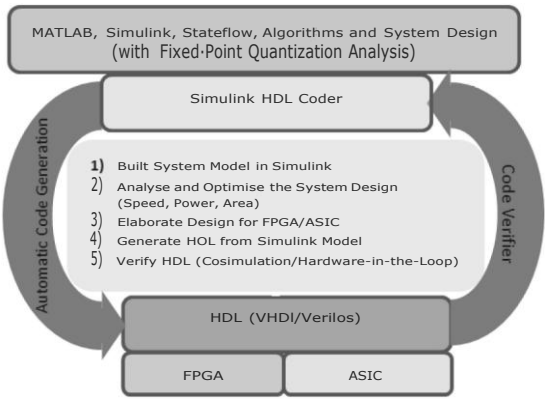


Fig. 3. Method to generate HDL Code from MATLAB and Simulink, with code verification.

difficult aspect of implementing an algorithm on an FPGA. The real HDL code generation process starts by modeling the algorithm in MATLAB Simulink using a HDL Coder library of more than 200 blocks (MATLAB keeps on updating and adding new blocksets) or Stateflow. The components and blockset supported in HDL Coder can be found by typing hdllib in the command window. Fig. 3 shows the code conversion and verification process in MATLAB Simulink HDL Coder.

Once the Simulink model is created, HDL *Workflow Advisor* guides in a step-by-step process to generate code from the model. Moreover it helps to check various other parameters and setting that is required for optimal code generation and verification.

Ill. IMPLEMENTATION OF MODEL BASED DESIGN

The implementation of HDL Coder in power electronics and drives is demonstrated by implementing a SVPWM modulation technique in a 3-ф VSI for generic applications. SVPWM refers to a special way of determining the switching sequence of the upper three power transistors of a 3-ф VST. It generates less harmonic in the output voltages and or currents in the windings of the motor load, possibility to optimize for lower switching losses, provides more efficient use of DC supply voltage as compared to direct sinusoidal modulation technique (voltage utilization of SVPWM is 2/..J3 times the sine wave) and compatible with the digital controller [19]. The circuit model of typical SVPWM modulated 3-ф VST is shown in Fig. 4. Switches SI to S6 are six power switches controlled by switching variables SWI, SW2, SW3, SW4, SW5, SW6 that shapes the 3-ф output voltage.

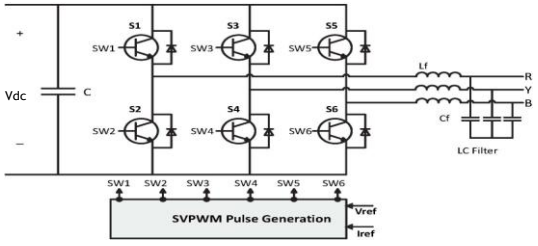


Fig. 4. SVPWM controlled VSI

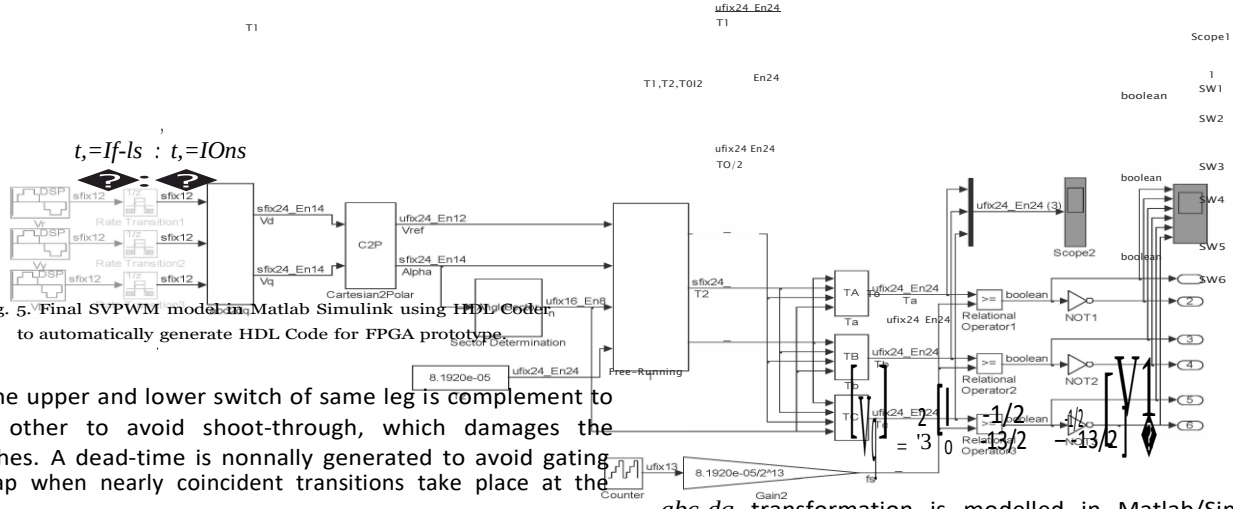


Fig. 5. Final SVPWM model in Matlab Simulink using HDL Coder to automatically generate HDL Code for FPGA prototype.

The upper and lower switch of same leg is complement to each other to avoid shoot-through, which damages the switches. A dead-time is nonnally generated to avoid gating overlap when nearly coincident transitions take place at the upper and lower switch of same leg. This can be achieved either by implementing dead-time during FPGA programming, or using a MOSFET driver with inbuilt dead-time. Nowadays, MOSFET drivers come with inbuilt dead-time and fault shutdown capability [20], so it reduces the programming complexity and resource requirements of FPGA implementations.

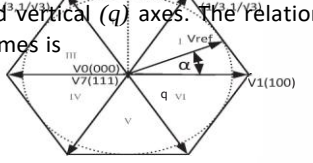
In SVPWM the reference voltage is mapped into switching space vector diagram and the duty cycles of the switches are calculated based on the mapping. There are six active states (VI, V2, V3, V4, V5 and V6) and two zero states (VO and V7) which combined in a various ways to generate the output voltage. Fig. 6 shows the basic switching vector and sector of SVPWM techniques. Fig. 5 shows the final MATLAB Simulink multirate model of SVPWM algorithm that automatically generates the HDL code for the FPGA prototype. The sampling rate of the model before the rate transition block is 1flS, and IOns after the rate transition block, for a master clock speed of 100MHz. The switching frequency of the inverter is determined by:

$$f\_ \text{Master Clock Speed} \quad \text{(I)}$$

Where, n-m-bit free counter. For example, if n=12, J;=100xIQ^6/2^{12}=24.4lkHz. The details of the each blockset is described in the following sub-sections.

A. *Transformation of abc-dq Reference Frame*

The voltage equations in the *abc* reference frame is transformed to the *dq* reference frame that consists of the horizontal (d) and vertical *(q)* axes. The relation between these two reference frames is



$$= \begin{bmatrix} 1 & -1/2 & -1/2 \\ 0 & \sqrt{3}/2 & -\sqrt{3}/2 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} \quad (2)$$

abc-dq transformation is modelled in Matlab/Simulink as shown in Fig. 7. The bloksets are HDL Coder compliant with fixed point output. The word length for each block is shown in the signal path. sfix12_En4 means the data is signed 12 bit word length with 4 bit fraction length and ufixlO_En8 means the data is unsigned 10 bit word length with 8 bit fraction length. The operators, constants and blocks support different data types and representation of data is transparent to designers in each signal path. Details on how to convert floating to fixed point will be discussed in Section TV.

B. *Determine* θ *and Angle (0.)*

For small switching time period *Ts*, V_{ref} can be considered approximately constant and can be expressed as

$$V_{ref} = \sqrt{(V_d^2 + V_q^2)} \quad (3)$$

and, angle

$$\theta = \tan^{-1}\left(\frac{V_q}{V_d}\right) \quad (4)$$

Fig. 8 shows the implementation of above mathematical expression in fixed point simulink blocks. Ready to use Cartesian2Polar block in Simulink is not supported by HDL Coder, so the above equation to calculate instantaneous atan function is implemented in floating point s-function MATLAB code and then converted to fixed point.

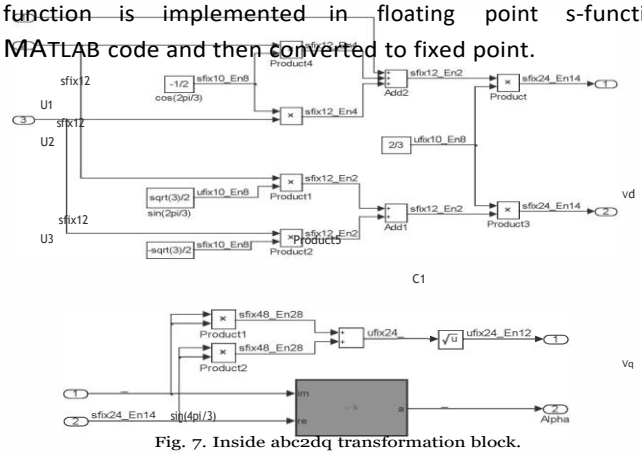


Fig. 7. Inside abc2dq transformation block.

(-2/3,0)

V5(001)/3,-1/V3

)

V6(101) (1/3,-1/V3)

C. Determine Switching Time Duration (Tib Tn To)

The instantaneous time duration of switching vector (T",Tn+1,To) for six switches are calculated in terms of a reference voltage (Vrer), angle (a), switching time period *(Tz)*, input voltage (Vdc) and sector *(n)*. For the volt-second balance in sector-I, from Fig. 6,

TI = T,V, + T,V, (5)(6)

SO,

T L I;f co: e : V,.J, ,,, i) V .T2 [cos e t)] (7)

Tz =1; +T2 +To

Solving §5, §6 and §7 we get,

7; v'3T/ Vdc |sin(ff/3 −a) (8)

And v'3T/|V,,! |sineal vd,. (9)

Hence, switching time duration in any sector *n*

fJ V"1 | ff/3 n-1 fJ V"1 I( ) (10)

T n+! = V"31 I. sln(a--;r)= n-J fJ IV I n-J n-J (-cosaSlll-JT+smaCOS — ff/3) (11)

And '0, = TI - +T,al) (12)

Where, T, =J: s-+ switching frequency of the VST

Fig. 9 shows the implementation of above mathematical expression for Tn,Tn+],To in fixed point Simulink blocks.

D. Determine the Sector (n)

It is necessary to determine the location of the reference voltage (which is rotating at co = to exactly generate the instantaneous duty cycle of each switch and switching sequence. The sector determination is implemented with compare to constant logic as shown in Fig. 10.

E. *Determine the Switching Time of Each Power Switch*

The switching pattern, sequence and time period of high side switch is determined based on the location i.e. angle (a) and magnitude of reference voltage (Vref) as shown in Fig. 11. The switching pattern and time is implemented for high side switch TA as shown in Fig. 12. Similar implementation is for TB and TC.

Fig. 10. Determination of sector (n)

Sector I: 0° Sas 600 T,
Sector II: 600 Sas 120°
Sector III: 120° as 1800
53
VO V7 V7 VO VO V2 V7 V7 VO VO V3 V4 V7 i V7 VO
TO/2 12 TI TO/2 TO/2 T2 n To/2 T2 TO/2 TO/li n n TO/2
Sector V: 2400 3000
Sector VI : 3000 3600
T, T, T, T, T,
51
S3
53
55
TO/2 V7 TI T2 TO/2 VO VS V6 V7 V7 VS VO VO VI V6 V7 i V7 VO
TO/2 TI T2 TO/2 TO/2

Fig. 9. Determination of switching time duration (Tn,Tn+"Tll)

I TO/2 TO/2 T2 TI

Fig. 11. Space Vector PWM switching patterns and corresponding switching time at each sector.

IV. FPGA PROGRAMMING

A. Floating Point to Fixed Point Conversion

Fixed point algorithm is implemented in FPGA for power, perfonnance and cost reasons. However, conversion from floating point to fixed-point is very challenging and time-consuming, typically demanding 25 to 50 % of the total design and implementation time. In a fixed point domain a pair (w, F) is considered for each of the parameter in algorithms, where *W*

is word length and F is fraction length of the parameters. Large Wand F result in better performance and lower Bit Error Rate (BER) but design consume large resources in FPGA or implementation requires expensive FPGA. On the other hand, smaller Wand F result in large BER whereas smaller footprint.

Optimization of wordlength to achieve best performance is an interactive process with the user in a MATLAB fixed point advisor which guides through the steps of converting floating point to fixed point algorithm. It also verifies the generated fixed-point code by comparing the floating and fixed point result. Multiple iterations by adjusting the word length settings, individually modifYing the data types as desired or accepting the *'proposedfixed-point types'* as recommended by fixed-point advisor are required to meet the desired accuracy (low BER) and optimum fixed point design. Fig. 13 shows the control signal of SVPWM generated by both floating and fixed point design and the corresponding error. The peak conversion error is ±1.5%. This error should be kept minimal to utilise the full modulation range or the DC link voltage.

*B. Code Conversion (from *.mdllslx and *.mfiles to VHDL)*

The HDL *Workflow Advisor* in HDL Coder automatically converts MATLAB code *(*. m files)*, Simulink *(*. mdllslx files)* from floating-point to fixed-point and generates synthesizable VHDL and Verilog code. MATLAB generates thousands of lines of VHDL codes in separate files for each blocksets. The generated code can be traced bidirectional to/from MATLAB and Simulink model. The *HDL Workflow Advisor* also highlights critical path timing in Simulink to help identifY speed bottlenecks and improve the performance of the design.

C. Verification

1) HDL *co-simulation:* The HDL Coder generates VHDL and Verilog test benches using HDL co-simulation wizard that automatically connects to the HDL simulator e.g. Cadence Incisive, Mentor Graphics ModelSim and Questa for rapid verification of generated HDL code. Fig. 14 shows the co-simulation scenario using Mentor Graphics ModelSim (student version).

This co-simulation streamlines the verification process and helps to fix the error before hardware implementation. Fig. 15 shows the co-simulation results of SVPWM MATLAB and Simulink model with ModelSim RTL-level models. The model was simulated for 3ms and it was found that the two results are exactly same.

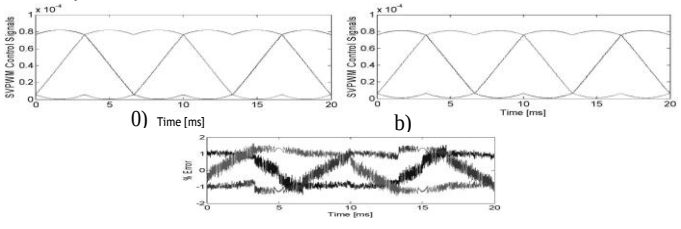


Fig. 13. SVPWM control signals generated for one cycle (50Hz) by a) Floating-Point Design; b) Fixed-Point Design and c) corresponding conversion % Error.

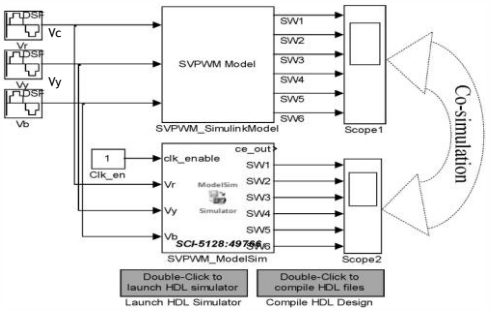
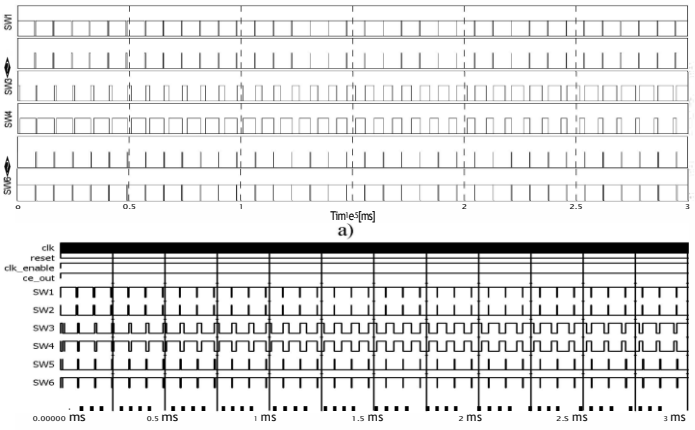


Fig. 14. Co-simulation scenario using ModelSim



b)

Fig. 15. Co-simulation results of SVPWM a) MATLAB Simulink model with b) Mentor Graphics ModelSim

2) *FPGA in-the-loop (FIL) co-simulation:* FIL test the design in real hardware for the generated HDL code. It generates a Simulink FIL block as shown in Fig. 16 that represents the HDL code. The programming file is loaded onto an FPGA with JTAG connection. TX/RX of data from Simulink to FPGA is via gigabit Ethernet crossover cable that co-

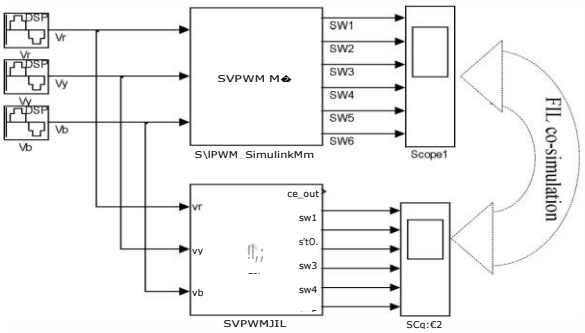


Fig. 16. FPGA in-the-loop co-simulation

D. VHDL *to bitstream and FPGA programming*

The generated HDL code is bit-true, cycle-accurate and synthesizable HDL code which is free from bugs. HDL Coder offers integration with Xilinx ISE design suite that makes it easy to implement algorithm in MATLAB and Simulink to target Xilinx FPGAs. Xilinx ISE compiles and generates the bitstream file which is then loaded into the FPGA using JTAG via the USB connection by iMPACT or Digilent Adept software. Table 1 shows the Xilinx ISE13.4 compilation report of resources utilization for SVPWM implementation in Spartan-6 XC6SLX45.

V. EXPERIMENTAL RESULTS AND DISCUSSIONS

The FPGA based approach to the automatic generation of VHDL code for digital controlled power electronics and drives using MATLAB Simulink is verified by implementing SVPWM modulated VSI in a Xilinx Spartan-6 platform.

The six driving pulses from FPGA are connected to signal amplification and isolation circuit via 8 pin PMOD connector. The power ground and logic ground are completely isolated from one another using HCPL2531 optoisolator. It protects FPGA from circulating ground current and high voltage spikes (15kV/flS). Fig. 17(a) shows the FPGA generated SVPWM control pulses for the high side switches.

Six SPW47N60C3 Cool MOS power MOSFET with best R1JS(on) are used for better efficiency design. The TTL level SVPWM signal from FPGA is amplified to meet the gate drive voltage requirement of the switch (±20V) by driver IC IR2130 from International Rectifier. It is a three phase high voltage bridge driver IC with three independent high and low side referenced output channels and has protection against fault [20]. The IR2130 also provides dead-time control to avoid any shoot-through to protect the switches. A dead-time (Td) of 2fls is introduced between high and low side switch as shown in Fig. 17(b). Too short dead time (Td<lflS) causes shoot-through current that reduces system efficiency; too long a dead time (Td>3flS) increases THD, negatively impacting the power quality.

The three phase line-to-line voltage of the VSI before filter is shown in Fig. 18(a) and Fig. 18(b) shows the filtered output voltage of the inverter. Small filter inductor (1.5mH) and capacitor (1.5flF) is required to filter out the switching harmonics. The THD of output voltage is only 3.8% and Crest Factor is 1.52, well below the IEEE allowable limits for grid connected inverter design or for motor drives application. The standby power loss is only 0.25% of rated power of the inverter (lkW).

Table I: Device utilization summary of Spartan-6 XC6SLX45
(Xilinx ISE13.4 estimated values)

Logic Utilization	Used	Available	Utilization
Number of Slice Registers	1521	54576	2%
Number of Slice LUTs	6117	27288	22%
Number of fully used LUT-FF pairs	1460	6178	23%
Number of bonded lOBs	10	218	4%
Number of BUFG/BUFGCTRLs	I	16	6%

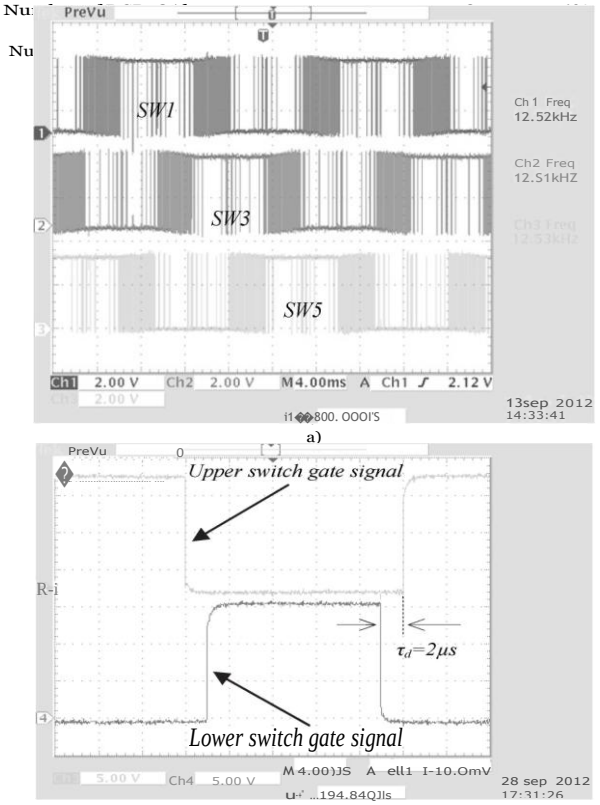


Fig. 17. SVPWM gating pulse from Spartan-6 FPGA for high side switch (Chl--SWI. Ch2--SW3. Ch3--SW5. Ch4--SW6)

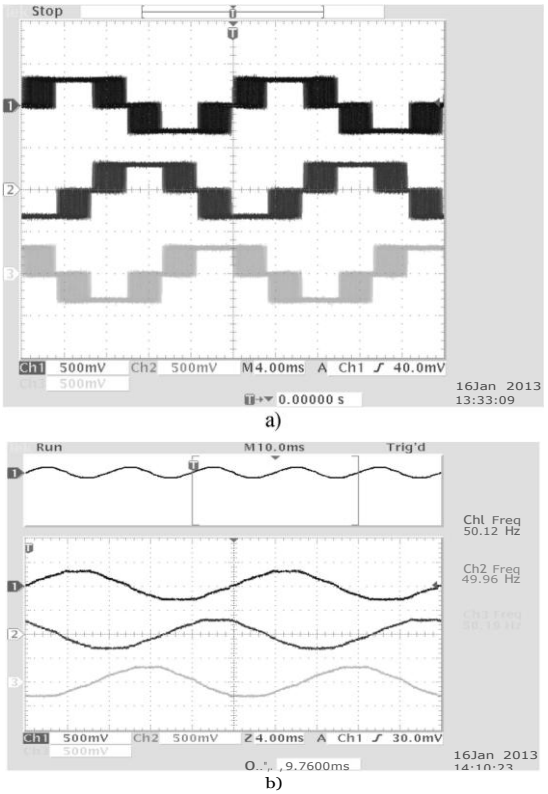


Fig. 18. Three phase voltage out from the VSI without loading a) before filter. b) after filter (Measured with high voltage differential probe GE8115 with attenuation of 1000:1. Chl-7VRy• Ch2-7VyB• Ch3-7VBR)

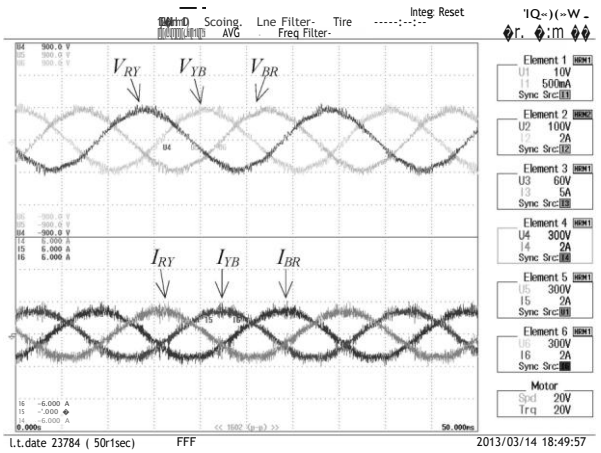


Fig. 19. Three phase voltage (300V/div) and current (2A/div) with 3300 resistive load in each phase (delta connected).

The control circuit demonstrated in this paper is an open-loop system. Complex feedback control system of power electronics and electric drives can be easily implemented using similar Matlab model based design as described in this paper. For example, power flow control, grid synchronization, variable speed drive (VSD), DC link voltage control etc.; however, it requires AID or DIA converters for proper data interface to FPGA and various peripheral sensors (e.g. voltage (V), current (I) or angular speed (m) etc.).

Fig. 19 shows the performance of the inverter during loading condition. The inverter is loaded with a delta connected resistive load of 300[2 in each phase. The load voltage and current shows a good performance of the inverter under loading conditions as well

VI. CONCLIISIONS

We have described a method to facilitate the development and implementation of FPGA-based digital controllers in power electronic converters and drives. The method is faster and provides a greater degree of confidence than traditional manual HDL coding. To illustrate the method a laboratory prototype of a 1kW FPGA-controlled Voltage Source Inverter (YSI) was described in detail in which the fixed-point control algorithm was automatically generated from Simulink models using the MATLAB HDL Coder, and verified before implementation on a Xilinx Spartan-6 XC6SLX45 board. Experimental characterization of the resulting YSI converter resulted in 3.8% total hannonic distortion and line-to-line crest factor of 1.52, well within the allowable range of IEEE standards. The very close agreement between experiment and simulation shows the efficacy of the method. The method is expected to be particularly useful for prototype development of other power electronic converters and electric drives with more complicated interfacing and control algorithms.

REFERENCES

(J':3
[I] Concettina Buccella and Carlo Cecati, "Digital control of power converters -A survey", IEEE Trans. Industrial Informatics, vol. 8, no. 3, pp. 437-447, August 2012.

[2] Stephan Van Beek and Sudhir Sharma, MathWorks, "Four Best Practices for Prototyping MATLAB and Simulink Algorithms on FPGAs", Verification Horizons, pp. 49-53, August 2011.

[3] Kiran Kintali and Yongfeng Gu, Model-based design with Simulink, HDL Coder, and Xilinx system generator for DSP," Mathworks White Paper 92077VOO, 2012.

[4] Eric Monmasson, Marcian N. Cirstea, "FPGA design methodology for industrial control systems - A review", IEEE Trans. Industrial Electronics, vol. 54, no. 4, pp. 1824-1842, August 2007.

[5] Miro Milanovic, Mi&#65533;ja Truntic and Primoz Slibar, "FPGA implementation of digital controller for DC-DC buck converter", Proceedings, Fifth IEEE International Workshop on System-on-Chip for Real Time Applications, pp. 439-443, Banff, 20-24 July, 2005.

[6] Eftichios Koutroulis, Apostolos Dollas, Kostas Kalaitzakis, "High-frequency pulse width modulation implementation using FPGA and CPLD ICs", J. Systems Architecture, vol. 52, pp. 332-344, June 2006.

[7] Lars T. Jakobsen and M. A. E. Anderson, "Digitally controlled point of load converter with very fast transient response", Digest, 12th European Conference on Power Electronics and Applications, pp. 1-10, Aalborg, 2-5 September, 2007.

[8] Diego Puyal, Luis A. Barragan, Jesus Acero, Jose M. Burdio and Ignacio Millan, "An FPGA-based digital modulator for full-or half: bridge inverter control", IEEE Trans. Power Electronics, vol. 21, no. 5, pp. 1479-1483, September 2006.

[9] Ying-Yu Tzou and Hau-Jean Hsu, "FPGA realization of space-vector PWM control IC for three-phase PWM inverters", IEEE Trans. Power Electronics, vol. 12, no. 6, pp. 953-963, November 1997.

[10] J. Tian, G. Berger, T. Reimann, M. Scherf, J. Petzoldt, "Design and implementation of a FPGA-based controller for resonant inverters", IEEE Trans. Power Electronics, vol. 12, pp. 953-963, November 1997.

[11] Angel de Castro, Pablo Zumel, Oscar Garcia, T. Riesgo and 1. Uceda, "Concurrent and simple digital control of an AC/DC converter with power factor correction based on FPGA", IEEE Trans. Power Electronics, vol. 18, no. I, pp. 334-343, January 2003.

[12] Oscar Garcia, Pablo Zumel Angel de Castro and Jose A. Cobos, "Automotive DC-DC bidirectional converter made with many interleaved buck stages", IEEE Trans. Power Electronics, vol. 21, no. 3, pp. 578-586, May 2006.

[13] Oscar Lopez, Jacobo Alvarez, Jesus Doval-Gandoy, Francisco D. Freijedo, Andres Nogueiras, Alfonso Logo and Carlos M. Penalver, "Comparison of the FPGA implementation of two multilevel space vector PWM algorithms", IEEE Trans. Industrial Electronics, vol. 55, no. 4, pp. 1537-1547, April 2008.

[14] R. Erikson, S. Angkititrakul and K. Almazeedi, "A new family of multilevel matrix converters for wind power applications: Final report", NREL Report NRELISR-500-40051, December 2006.

[15] Marcian N. Cirstea and Andrei Dinu, "A VHDL holistic modeling approach and FPGA implementation of a digital sensorless induction motor control scheme", IEEE Trans. Industrial Electronics, vol. 54, no. 4, pp. 1853-1864, August 2007.

[16] Da Zhang and Hui Li, "A stochastic-based FPGA controller for an induction motor drives with integrated neural network algorithms", IEEE Trans. Industrial Electronics, vol. 55, pp. 551-561, Feb. 2008.

[17] Mohamed Wissem Naouar, Ahmad Ammar Naassani, Eric Monmasson and IIhem Slama-Belkhodja, "FPGA-based predictive current controller for synchronous machine speed drive", IEEE Transactions on Power Electronics, vol. 23, no. 4, pp. 2115-2126, July 2008.

[18] Frede Blaabjerg, Philip C. Kjaer, Peter Omand Rasmussen and Calum Cossar, "Improved digital current control methods in switched reluctance motor drives", IEEE Transactions on Power Electronics, vol. 14, no. 3, pp. 563-572, May 1999.

[19] Richard Zhang, "High performance power converter systems for nonlinear and unbalanced load/source", PhD Thesis, Virginia Polytechnic Institute and State University, 17th Sept. 1998.

[20] International Rectifier, Data Sheet No. 60019 Rev. P, "IR21301IR2132 3-phase bridge driver", http://www.irfcom/product-info/datasheetsldatalir213O.pdf