Hindawi Publishing Corporation
Journal of Control Science and Engineering
Volume 2013, Article ID 713293, 10 pages
http://dx.doi.org/10.1155/2013/713293

Research Article

FPGA Control Implementation of a Grid-Connected Current-Controlled Voltage-Source Inverter

Rickard Ekström and Mats Leijon

Division of Electricity, Swedish Centre for Renewable Electric Energy Conversion, Uppsala University, P.O. Box 534, 751 21 Uppsala, Sweden

Correspondence should be addressed to Rickard Ekström; rickard.ekstrom@angstrom.uu.se

Received 8 April 2013; Revised 20 September 2013; Accepted 25 September 2013

Academic Editor: Mohamed Zribi

The full control system of a grid-connected current-controlled voltage-source inverter (CC-VSI) has been designed and implemented on a field-programmable gate array (FPGA). Various control functions and implementation methods are described and discussed. The practical viability of the system is evaluated in an experimental setup, where a VSI supplies 30 kW into the local grid at 400 V. A phase-locked loop (PLL) is used for grid phase tracking and evaluated for simulated abnormal grid conditions. Power factor is kept at unity, and the implemented control system is stressed with step responses in the supplied active power. A moving-average filter is implemented to reduce the effects of noise and harmonics on the current control loops. A coupling between active and reactive power flow is observed for the step responses but may be ignored in this context. The proposed system is fully comparable with more conventional microprocessor-based control systems.

1. Introduction

The voltage source inverter (VSI) has become increasingly used for grid connection of renewable energy sources, such as wind power turbines or solar cells. Improved ratings in semiconductor components like the insulated-gate bipolar transistor (IGBT) are gaining market for more and more powerful applications where only the thyristor-controlled current-source converter has been used earlier. The VSI is dominating the market of distributed generation and may be controlled by either voltage or current feedback. Current-control (CC) is favoured due to its excellent dynamic characteristics and its inherent overcurrent protection. There are three major groups of current-control modulations for VSIs, namely, hysteresis control [1], predictive current control [2], and ramp comparison [3]. A survey on current-control methods is presented in [4].

Depending on the control strategy, the power converter control system has traditionally been designed with microcontrollers or complex programmable logic devices (CPLDs). However, increased complexity and more real-time data analysis have called for the development of digital signal processors (DSPs). The DSPs have very good programmability and can easily manage conditional code. Due to their ability to work with floating point arithmetics, they can handle complex mathematical functions well.

Alternatively, the application-specific integrated circuit (ASIC) may be used, where all the functions are designed into a fixed integrated circuit. The ASIC has to implement all functions in gates and very complex conditional programs may result in very poor gate utilization. Also, the ASIC is limited to fixed point arithmetic. As the gates can be programmed in parallel, the ASIC can operate simpler logic at a much faster rate than its DSP counterpart. However, once the ASIC is designed and manufactured, it cannot be modified for bugs or changed system characteristics, which makes it less useful in development projects.

The advent of the FPGA has resolved this issue and makes it superior to the ASIC for researchers and developers. Even though the FPGA requires much more area and power consumption and is about three times slower than its ASIC counterpart [5], its short time to market and field-programmable ability make it a tempting solution.

The FPGA structure is based on a matrix of controllable logic blocks (CLBs). These consist of a look-up table (LUT), which can be configured either as a RAM/ROM or as logic

functions, and a D-type flip-flop that can handle, for example, registers. The CLBs are interconnected by a programmable wire network, synchronized with a top level clock. Modern FPGAs can also have extra inbuilt features such as dedicated DSP-blocks that can handle complex computations in one clock cycle. Also, external LUT blocks may be used as memory.

The use of FPGA technique within power electronics is mostly seen where rapid input/output (I/O) is requested. Reviews of control system implementation in FPGA is found in [6–8]. Grid impedance analysis using FPGA are done in [9]. An FPGA-based grid phase tracking method is described in [10], and pulse-width modulation (PWM) implementations for inverters are found in [11, 12]. A combination of DSP and FPGA for SVM-control of a matrix converter is used in [13], and more FPGA-based SVPWM implementations can be found in [14, 15]. FPGA-implementations for fault detection in a VSI control is made in [16].

In this paper, the control system of a grid-connected CC-VSI has been designed and implemented on an FPGA. The control system includes phase tracking of the grid voltage, inverter logic output control, active and reactive power flow control, high-speed burst logging, and fault detection. Hardware implementations are discussed as well as their limitations and the number of gates required for specific functions are summarized. The control system is implemented on the FPGA-chip of a compactRIO module, and the VHDL-code is generated by LabView G-code. The system allows for integration of a RT-controller if heavier data analysis is required. The practical viability of the system is evaluated in an experimental setup with a grid-connected VSI. 30 kW at unity power factor is injected into the local electric grid, and the current-control loops are evaluated by making step responses in the active power flow.

2. FPGA Programming

The initial step in FPGA design is to define the algorithms used. An algorithm can be characterized by its complexity and time constraints. The accuracy of I/O-variables is set by their bit sizes.

Once this is done, the possibility of reusing modules in the system has to be explored. By smart time sharing, an algorithm may be put as a subroutine and routed to several different parts of the FPGA module. Finally, the same function may be expressed in different ways with the logic gates available. The most gate-efficient algorithm should be used, based on the available gate resources on the FPGA. There are various ways of implementing the VHDL-code on the FPGA. Many compilers offer the programmer to write in a more user-friendly language, which seldom is as efficient as direct VHDL-programming. The conversion into VHDL from another language will be a compromise between compilation time, space optimization, or clock cycle optimization. Below, a few key functions are summarized.

2.1. Integer Numbers. The fundamental building blocks of an FPGA are best utilized with integer maths operations.

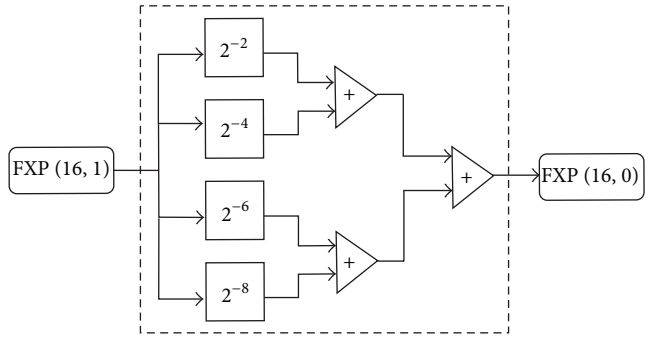


Figure 1: Multiplication by the use of adders and decimal point shifters.

Floating-point (FP) operation is possible but provides poor utilization of the logic cells [17]. If decimal points are required, fixed-point numbers (FXP) are used instead. A FXP is predefined by its bit length and the number of bits allocated for decimal accuracy and may be signed or unsigned. One drawback of FXP compared with FP is that they rapidly grow in size with the computations performed on them. Care must be taken to avoid saturation or overflow of a FXP. An evaluation of FXP precision is found in [18, 19].

2.2. Integer Multiplication. While adding and subtracting are rather straightforward operations with little space demand on the FPGA, multiplication is a big "gate-hog" that should be avoided if possible. However, if the multiplication is fixed, this is not true if the multiplying factor is a multiple of 2. Multiplication with 2^N may easily be replaced with a decimal point shift N times to the left or right on the FXP. This requires no resources, since it is only a matter of rerouting of the FXP between logic cells. If the multiplying factor cannot be approximated to a factor of 2, it may be simplified to a sum of fractions of 2 and then added. For example, the multiplier 1/3 may be approximated as

$$\frac{1}{3} \approx \frac{85}{256} = \frac{64+16+4+1}{256} = \frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \frac{1}{256} \quad (1)$$

which will give an error of about 0.4%. Depending on the accuracy required, the number of fractions may be varied. Figure 1 shows the implementation of this. The resource allocation for the proposed method compared to using a normal multiplication block is shown in Table 1 for implementations on a Xilinx-5 chip. Here, the first method can save approximately 2% of the dedicated DSP blocks on the FPGA. Implementation of division should be converted into multiplication instead, as this will save lots of space.

2.3. Integrals. On the contrary to derivatives, the integral of a signal is very straightforward on the FPGA. The integral of $V_{\text{in}}(t)$ may be approximated as a finite sum according to

$$\int_0^t V_{\text{in}}(t)\,\mathrm{d}t \approx \Delta t \cdot \sum_{i=1}^{n} V_{\text{in}}(t_i), \quad (2)$$

where Δt is the sampling time and is assumed constant here. By using shift registers, the cumulative value of the signal is

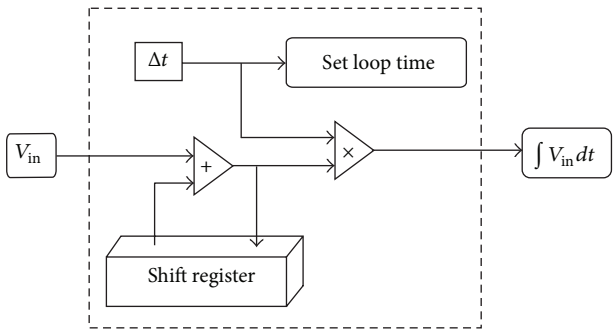


Figure 2: Implementation of the discrete integral in FPGA by the use of shift registers.

Table 1: FPGA gate utilization for two different types of multiplication.

	Multiplier	Decimal shift and adders
Total slices	4.3%	4.3%
Slice registers	2.2%	2.0%
Slice LUTs	2.0%	2.2%
DSP48s	2.1%	0%
Block RAMs	0%	0%

stored locally in the CLB and used directly for the next loop cycle as shown in Figure 2. The sampling time Δt sets the accuracy of the integral and is also used as a scaling factor for the output.

2.4. Advanced Functions. More advanced calculations, for example, trigonometric, logarithmic, or square-root functions, would require large amounts of resources if implemented as in a DSP counterpart. In the early age of FPGA, work was put into expressing these functions as combination of the simpler blocks [20]. As memory sizes grew, the use of look-up tables (LUTs) became a better option. Today, generic DSP blocks are often integrated on the FPGA semiconductor chip, and may perform these high through-put computations within one clock cycle. In Table 2, a resource comparison is done between a sine/cosine LUT-block and a trigonometric computational block producing the same output, implemented on the Xilinx-5. The speed, resolution, and functionality of the two are equivalent, but it is clear how the LUT locks up one third of the total block RAM available, making it less attractive. However, the RAM space demand of the LUT will decrease proportionally with reduced output resolution.

3. Grid Phase Tracking

There are various known methods of tracking θ, the grid phase. Zero-crossing detection is a simple method to implement but is sensitive to noise and does not give any phase information between the zero-crossings. More advanced methods include the phase-locked loop (PLL), Kalman filters, or the discrete Fourier transform (DFT). A good comparison between these may be found in [21]. In this paper, grid

Table 2: FPGA gate utilization for sine and cosine calculations using a LUT or a computational block.

	Computational block	LUT
Total slices	4.6%	5.0%
Slice registers	2.0%	2.0%
Slice LUTs	2.5%	2.7%
DSP48s	0%	0%
Block RAMs	0%	31.2%

phase tracking is performed using a three-phase synchronous reference frame PLL loop, due to its robustness and relatively simple implementation. The PLL structure is shown in Figure 3.

The closed-loop transfer function $H_{\text{PLL}}(s)$ of the PLL can be written as [21]

$$H_{\text{PLL}}(s) = \frac{K_P s + K_I}{s^3 T_S + s^2 + K_P s + K_I}, \tag{3}$$

where K_P and K_I are the gains of the PI-controller and T_s is the sampling time of the PLL. If the sampling frequency is sufficiently high, the above expression can be simplified to

$$H_{\text{PLL}}(s) = \frac{K_P s + K_I}{s^2 + K_P s + K_I}. \tag{4}$$

The sampling frequency is usually set to the switching frequency or half of it, depending on the control algorithm used. The design of the PLL is discussed in [22, 23] and is a compromise between speed and noise immunity. A discrete bit-stream PLL implementation is discussed in [24].

4. Power Flow Control

Assuming a stiff grid, the active power P and reactive power Q for a grid-connected VSI are governed by

$$\begin{aligned} P &= \frac{|V_g|\,|V_i|}{X}\sin(\delta) \\ Q &= \frac{|V_g|\,|V_i|}{X}\cos(\delta) - \frac{|V_g|^2}{X}, \end{aligned} \tag{5}$$

where V_g is the grid voltage, V_i the inverter voltage, δ the phase angle and X the reactance between the VSI and the grid.

To simplify the current-control feed-back system, all phase currents are transformed into the $dq0$-frame by:

$$I_{dq0} = T * I_{abc}, \tag{6}$$

where T is the Clarke/Park transformation matrix [25]:

$$T = \sqrt{\frac{2}{3}} \begin{pmatrix} \cos(\theta) & \cos\left(\theta - \frac{2\pi}{3}\right) & \cos\left(\theta + \frac{2\pi}{3}\right) \\ \sin(\theta) & \sin\left(\theta - \frac{2\pi}{3}\right) & \sin\left(\theta + \frac{2\pi}{3}\right) \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}. \tag{7}$$

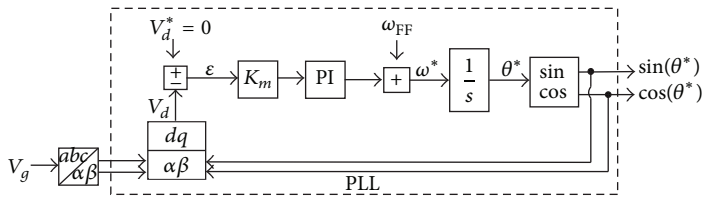


Figure 3: Schematic of the phase-locked loop for grid phase tracking.

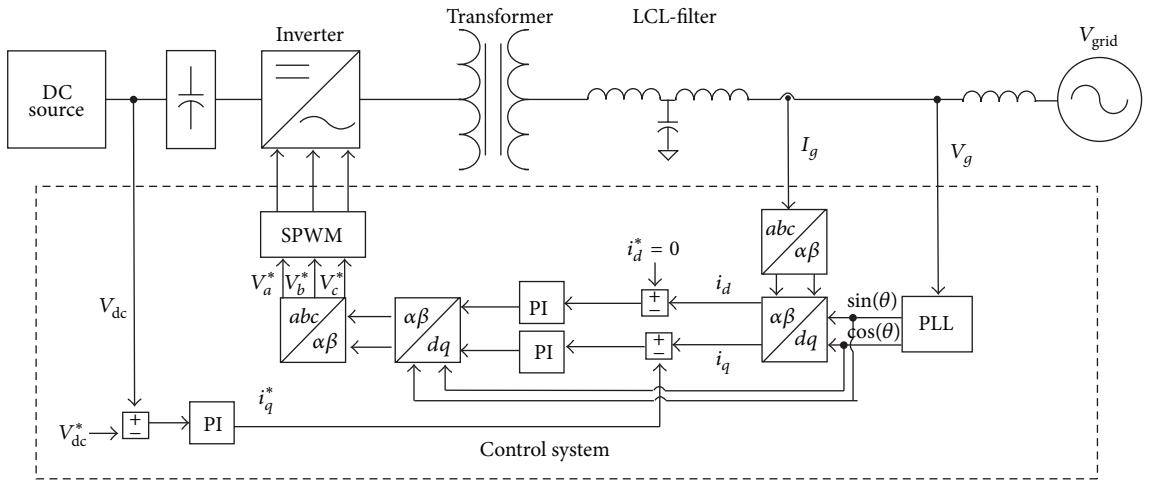


Figure 4: Experimental setup and control system overview for the grid-connected CC-VSI.

In this paper, the sinusoidal pulse-width modulation (SPWM) will be used [26], where a control signal V_c is compared with a carrier wave signal to generate the inverter output pulses. V_c is generated by

$$\begin{aligned} V_c &= V_{cd}^* \cdot \sin(\omega t) + V_{cq}^* \cdot \cos(\omega t) \\ &= m_a \sin(\omega t + \delta), \end{aligned} \tag{8}$$

where V_{cd}^* and V_{cq}^* are current-control feed-back variables used to control the active and reactive power flows. The amplitude modulation index is $m_a = \sqrt{(V_{cd}^*)^2 + (V_{cq}^*)^2}$ and the load angle is $\delta = \tan^{-1}(V_{cq}^*/V_{cd}^*)$. If $m_a < 1$, the inverter fundamental output phase voltage V_{i1} is derived as

$$V_{i1} = m_a \frac{V_{DC}}{2\sqrt{2}}. \tag{9}$$

5. Experimental Setup

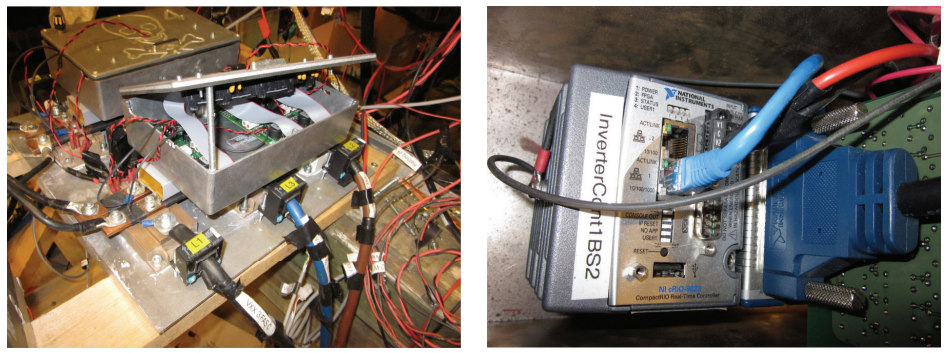
The one-line diagram of the experimental setup is shown in Figure 4. The three-phase inverter is made from 3 dual-package 400GB126D IGBT modules with 2SC0108T2Ax-17 driver boards mounted above, as shown in Figure 5(a). These have in-built short-circuit protections set to trigger 2.5 μs after fault detection. The inverter is directly connected to a three-phase 345 V/1 kV YY-connected transformer. The transformer ratings are given in Table 3.

An LCL-filter is placed on the secondary side of the transformer. The primary and secondary filter inductors are both designated the value of 1.5 mH. These are manufactured on two three-phase cores and there is an unbalance between the phases of ±10%. The shunt capacitors are connected in star configuration with 20 μF per phase. The grid main voltage at the point of common coupling (PCC) is 400 V at 50 Hz.

Table 3: Transformer rating.

Power rating	80 kVA
Voltage ratio	345 V/1 kV
Primary winding resistance	17 mΩ
Primary leakage reactance	1 mH
Secondary winding resistance	0.12 Ω
Secondary leakage reactance	22 mH
Magnetizing impedance (50 Hz)	2.3 H
Magnetizing resistance (50 Hz)	173 Ω

A compactRIO NI-9014 module with integrated RT-controller and FPGA chip from the National Instruments was used, as shown in Figure 5(b). The FPGA chip is of model Xilinx-5 [27] with an internal clock frequency of 40 MHz. The RT-controller is of the model NI-9022. Two of the four available I/O-slots in the chassis are used. In the first, an NI9205 analogue differential input module is put for data sampling. It has a total of 16 available channels when set in differential mode, with an analogue-to-digital conversion (ADC) speed of 250 kS/s for a single-ended input, and 16 bit resolution. A differential input requires three times the sampling time. In the given control strategy, seven measurements have to be made: 3 grid voltages, 3 grid



(a) (b)

Figure 5: (a) Three-phase inverter with driver circuits mounted on top. (b) The compactRIO-based control system.

Table 4: FPGA gate utilization for various implemented functions.

Loop function	Total slices	Slice registers	Slice LUTs	DSP blocks	Block RAM (kbit)	Loop time
$abc/\alpha\beta$-transform	458	880	851	0	0	9 ticks
$\alpha\beta/dq$-transform	476	1010	760	8	0	8 ticks
PLL (without transforms)	1360	2581	3342	4	0	36 μs
Control signal generation	722	1676	1479	15	0	5 ticks
SPWM	541	1,017	1,103	0	0	7 ticks
Overcurrent protection	703	1,009	1,465	1	0	84 μs
Grid monitor with burst log	—	1,375	2,621	1	0.936	84 μs
Gate usage	4,260	9,548	8,279	29	0,936	
Gates available	7,200	28,800	28,800	48	1,728	
Gate usage (%)	59.16%	33.15%	40.35%	60.42%	54.17%	

currents, and 1 DC voltage measurement. (To slim the system further, only two voltage and current measurements may be used for the grid, if the system neutral is floating.) The hardwired ADC MUX will share these equally, allowing the sampling frequency to reach 11.9 kHz. The second I/O-slot in the chassis is used for the inverter control signal output. Here, an NI9401 module is utilized, with 8 digital outputs, each with a delay time of 8 μs. For the three-phase inverter, only six of the outputs are required.

6. FPGA implementations

The control system of Figure 4 was implemented using Labview G-code. This is compiled into VHDL-code before programmed onto the FPGA chip. The advantage of using Labview is a relatively user-friendly interface, with the drawback of a not completely optimized VHDL design and some limitations in the code writing.

A general overview of the various control loops is found in Figure 6. All the necessary parts of the system are implemented in the FPGA. As can be seen, there is also provision to connect a RT-controller and a PC. These are not crucial for the control system but provide auxiliary functions. The RT-controller can be used for harmonic analysis of the grid currents, fault handling, and transfer of data from the FPGA burst log. The burst log is triggered at any fault detection like an overcurrent. The PC works as a remote interface and long-term data.

The gate utilization of all the functions implemented above is summarized in Table 4. This is not claimed to be completely optimized. Also, some auxiliary functions for initializing the system and general protections are excluded, as they are not the scope of this paper. The dominant delay time of the control system is the input sampling time of the ADCs, requiring 12 μs per sample. The data processing and internal logic have time delays in the order of ticks, which become negligible in comparison. To improve the system dynamics further, the ADCs must become faster.

Below, the major functions are described.

6.1. abc-dq0 Transformation. The $abc/dq0$ transform is used in both the PLL and the current-control loop. It is implemented as a subroutine and used for the two instances. As can be seen in Figure 6, the subroutine is implemented in series for the two computations, making it possible to reuse more efficiently. In the calculation of the SPWM control signals, V_c, the $dq0/abc$-transform is used instead, and this has to be coded separately. However, the terms $\sin(\theta)$ and $\cos(\theta)$ are still the same and may be called by local variables for a more efficient gate utilization. If we assume a balanced, steady state three-phase system, the 0-component may be omitted, and so the Clarke/Park transform is reduced to

$$T = \sqrt{\frac{2}{3}} \begin{pmatrix} \cos(\theta) & \cos\left(\theta - \frac{2\pi}{3}\right) & \cos\left(\theta + \frac{2\pi}{3}\right) \\ \sin(\theta) & \sin\left(\theta - \frac{2\pi}{3}\right) & \sin\left(\theta + \frac{2\pi}{3}\right) \end{pmatrix}. \quad (10)$$

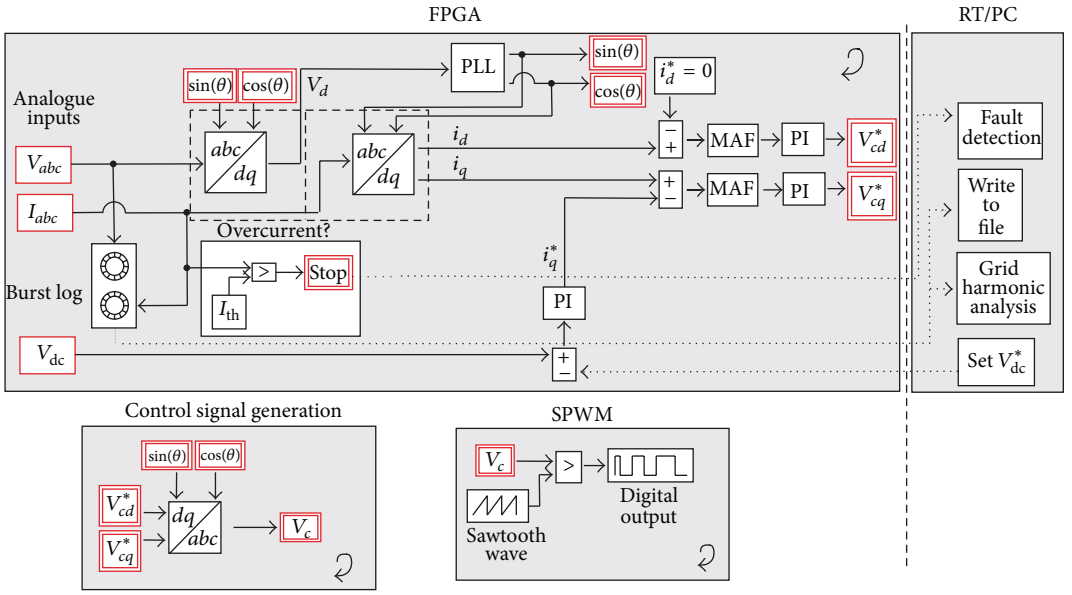


Figure 6: Overview of the FPGA implementation of the grid connection control system.

Since the FPGA used has generic DSP-blocks, some of these are used for generation of sine and cosine waveforms as discussed in Section 2.4. For any trigonometric phase input, a 16-bit output is generated within one clock cycle. However, trigonometric functions are still rather space demanding, and the matrix transformation in (10) requires six of these. Alternatively, V_{dq} can be calculated by

$$V_d = \sqrt{\frac{2}{3}}\left[V_a\cos(\theta) + V_b\left(-\frac{1}{2}\cos(\theta) + \frac{\sqrt{3}}{2}\sin(\theta)\right) + V_c\left(-\frac{1}{2}\cos(\theta) - \frac{\sqrt{3}}{2}\sin(\theta)\right)\right]$$
$$V_q = \sqrt{\frac{2}{3}}\left[V_a\sin(\theta) + V_b\left(-\frac{1}{2}\sin(\theta) - \frac{\sqrt{3}}{2}\cos(\theta)\right) + V_c\left(-\frac{1}{2}\sin(\theta) + \frac{\sqrt{3}}{2}\cos(\theta)\right)\right] \quad (11)$$

which only requires two trigonometric functions but has two extra multiplication functions instead. The two implementations of the *abc*/*dq*-transformation are illustrated graphically in Figures 7 and 8. A comparison of the gate utilization for the two types is found in Table 5. In this case, there is no difference between the two, but it illustrates how the same function can be implemented in different ways. If the trigonometric functions are more gate demanding, type 1 is more efficient, whereas if the multiplication blocks are more gate demanding, type 2 should be selected. In the experimental setup, type 1 has been implemented.

6.2. PLL. The major delay of the PLL loop is the grid voltage ADC delay times. This sets the loop frequency at 11.9 kHz, which is sufficient for the reduction of (3) to (4).

Table 5: Comparing two types of implementations of the *abc*/*dq*-transformation.

	Type 1	Type 2
Slice registers	30.8%	30.8%
Slice LUTs	31.9%	31.3%
DSP48s	12.6%	12.6%
Block RAMs	0%	0%

The integral of the frequency to get the phase is done by the procedure explained in Section 2.3. The PI-controller can be implemented by either using shift registers or using the generic DSP-blocks. Here, the latter is chosen, as it has slightly better dynamics at the expense of more gates required. The variables sin(θ) and cos(θ) are fed into local variables that may be used in parallel loops on the FPGA. Local variables are designated double frames in Figure 6.

6.3. SPWM and Current Control. Calculation of the bipolar SPWM control signals V_c for all three phases is done with the inverse Clarke/Park transformation, reusing the trigonometric results from the PLL by the local variables mentioned above. The inverter digital outputs are generated by comparison of V_c with a carrier waveform like a sawtooth wave or a triangular wave. The harmonic content of V_i does not differ much between the two [28]. Here, the sawtooth wave is selected due to its easiness to implement. To implement the sawtooth wave, a simple counter with 2N steps is used, starting at $-N$ and going up to N. If the SPWM loop time shown in Figure 6 takes $t_{\text{SPWM}}s$ to run once, the switching frequency f_s of the sawtooth wave can be calculated as

$$f_s = \frac{f_{\text{FPGA}}}{2Nt_{\text{SPWM}}}, \quad (12)$$

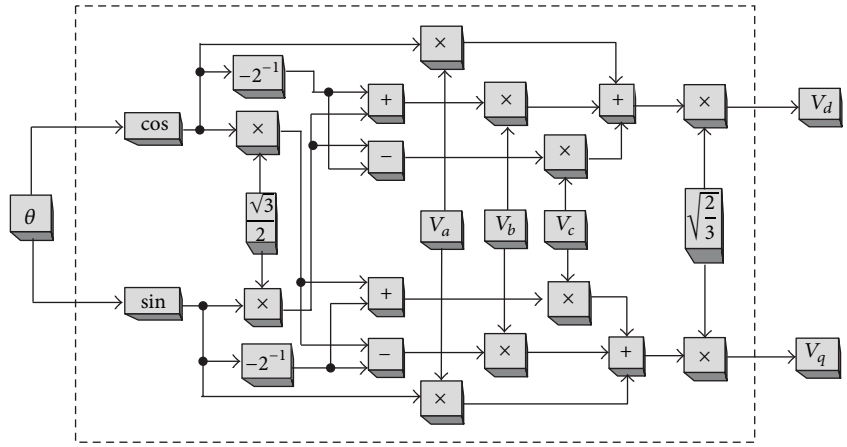


Figure 7: *abc*/*dq*-transform with type 1.

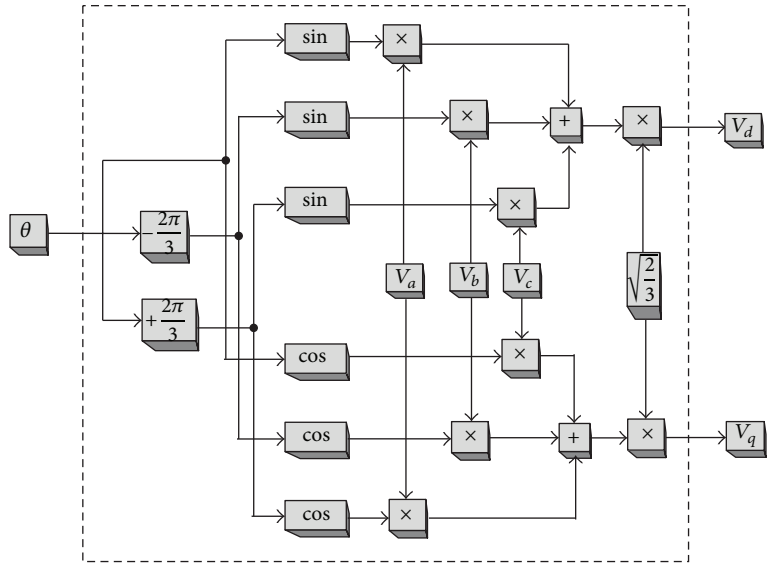


Figure 8: *abc*/*dq*-transform with type 2.

where f_{FPGA} is the clock frequency of the FPGA; here it is 40 MHz. If the SPWM-loop is put in series with the PLL, the run time becomes 84 μs. If the required f_s is 3.5 kHz, the resolution of the carrier wave becomes limited to 4 steps, making the output useless. Since V_c will not change significantly between sampling times, it is valid to put the SPWM loop as a parallel process on the FPGA. Now the SPWM loop time is only 8 ticks, and by setting, for example, $N = 512$ steps, we obtain a switching frequency of $f_s = 4.88$ kHz. This gives a good compromise between switching frequency and carrier wave resolution. The switching frequency may be altered either by changing N or introducing a delay time to prolong t_{SPWM}.

The control variables V_{cd} and V_{cq} are derived by comparing the measured i_{dq} with the set i^*_{dq} and applying two PI-control blocks on the result. i^*_d is always set to zero to get unity power factor into the grid. i^*_q is derived based on the difference of the measured V_{dc} and the set V^*_{dc}. To reduce the effects of harmonics and noise on the current signals, a moving average filter is applied on i_{dq}. This is easily implemented by calculating the mean of the last 100 measured values, stored in shift registers.

7. Results and Discussion

Initially, the PLL was tested by simulating a three-phase grid voltage internally on the FPGA, using LUTs. In Figure 9(a), the grid phase for a distortion-free 50 Hz grid voltage is tracked by the PLL with perfect results. In Figure 9(b), 5% of the 3rd, 5th, and 7th harmonics are added to the grid voltage. The phase tracking is still stable and does not completely follow the grid phase harmonic derivatives since these will be partially filtered out by the PLL transfer function.

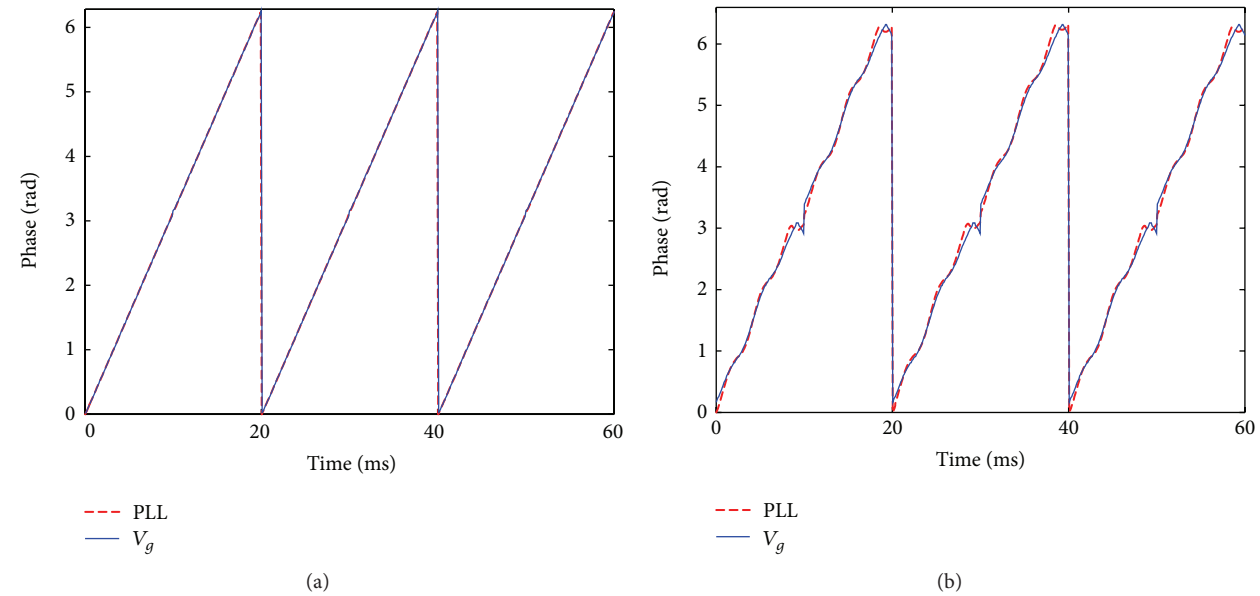


FIGURE 9: PLL phase tracking versus the grid phase for (a) a perfect grid and (b) a grid polluted with 5% of the 3rd, 5th, and 7th harmonics.

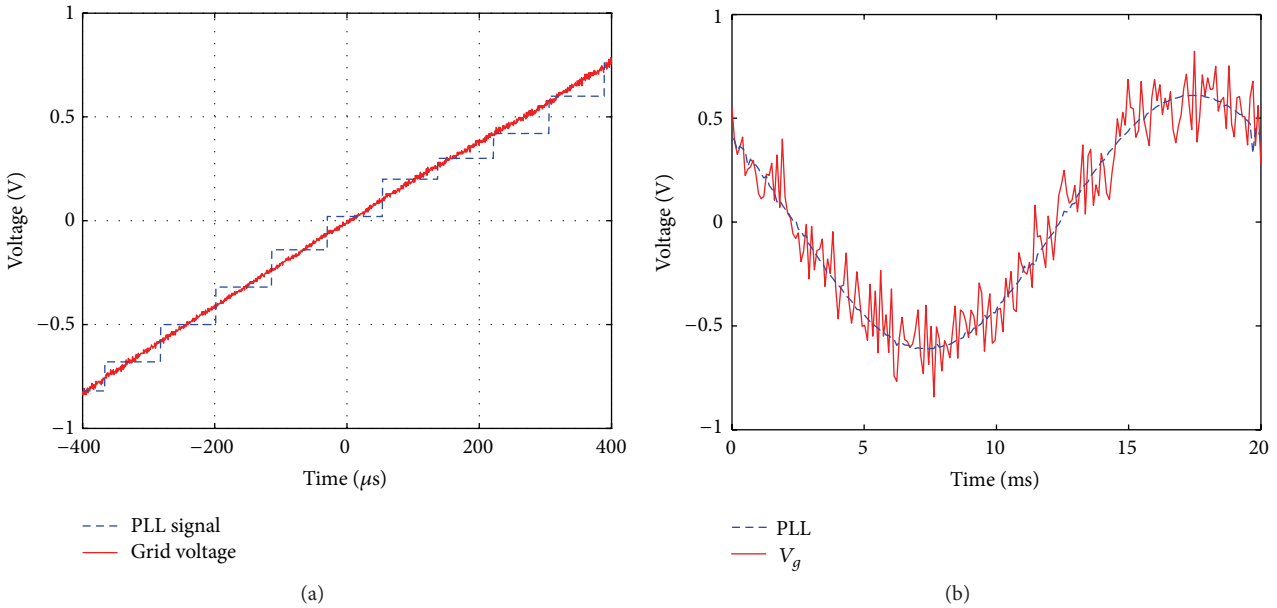


FIGURE 10: Time domain phase tracking of the grid voltage. In (a), the discrete steps of the PLL output are shown, based on a sampling time of 84 μs, around the zero-crossing of the grid voltage. In (b), 20% Gaussian noise is added to the grid voltage, and the resultant PLL output acts as a lowpass filter.

In Figure 10(a), the time-domain zero-crossings of the grid voltage and the PLL signal are compared. The PLL is updated with the analogue input sampling time of 84 μs and is set constant in between. Since the grid voltage does not change significantly between samples, the discrete PLL is a good representation of the grid voltage. In Figure 10(b), the noise immunity of the PLL is evaluated. Here, 20% Gaussian noise is added to the grid voltage, and as can be seen this is effectively filtered out by the PLL, since the integrals act as a low-pass filter.

Figure 11 shows the three phase currents at 30 kW active power injection into the grid. The reactive power flow is set to zero. The system is stable, but there is a minor unbalance between the phase currents, due to an unbalance in the harmonic LCL-filter. This can be mitigated by either compensation of the SPWM control signals, or the control system can be extended to three individual current-control loops, one per phase. However, this mitigation will not be linear due to the power transformer that will try to cancel any voltage unbalances by its internal flux. For a more stable

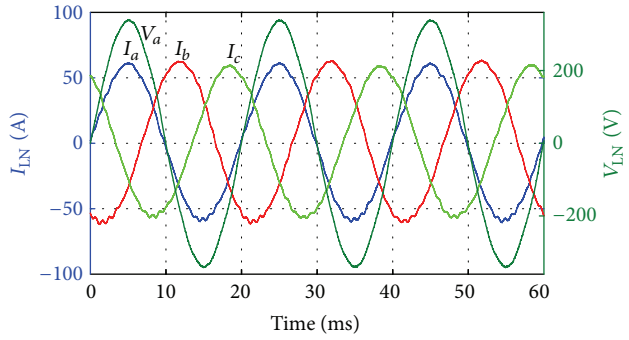


Figure 11: Grid currents and one phase voltage when the inverter supplies to the grid with 30 kW active power at unity power factor.

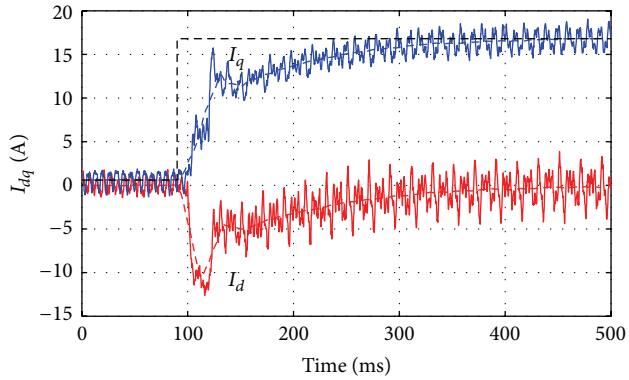


Figure 12: Step-up response in active power from 1 kW to 21 kW, $Q^* = 0$.

current-control loop, symmetrical filtering may be applied to the measured currents.

With intermittent renewable energy sources, there may be a swift change in power input to the grid, unless there is some intermediate energy storage. Thus, the current-control system has to manage step responses in active power while still keeping reactive power flows at the desired level. In Figure 12, a step-up response is made from 1 kW to 21 kW while keeping the desired reactive power flow zero, and in Figure 13 the corresponding step-down response is performed. Since the grid voltage is stable, i_q is directly proportional to the active power and i_d proportional to the reactive power.

Due to the distortion in the grid voltage, the injected current will carry lower order harmonics, which cannot be removed by the low-pass LCL-filter. These are clearly present in the ripple of i_{dq} of Figures 12 and 13. The presence of these harmonics makes it harder to tune the sensitivity of the current-control feed-back loops, and there are two ways of resolving this. Either, the 50 Hz fundamental frequency of the currents is filtered by a high-order bandpass filter. This is preferably implemented as an analogue filter, as its FPGA-implemented counterpart, though possible, will require a large number of gates. The other way is by filtering the i_{dq}-currents. In dq-frame, the DC-component corresponds to the fundamental frequency, and a simple moving average

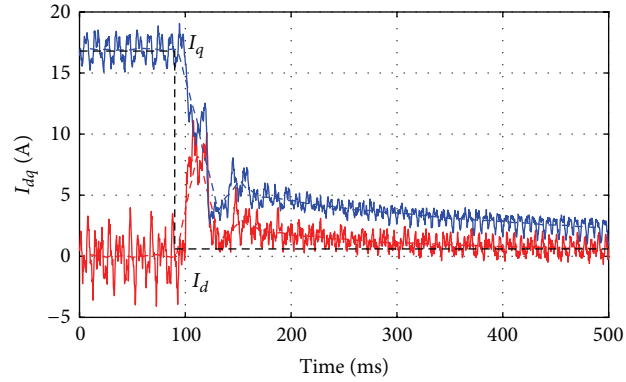


Figure 13: Step-down response in active power from 21 kW to 1 kW, $Q^* = 0$.

filter will do the trick. However, this will reduce the speed of the current-control. The method chosen depends on the required sensitivity of the current-control loop. In this paper, a smoothing moving-average filter has been applied on the FPGA. The moving average values of the i_{dq} currents can be seen as dotted lines in the figures and are used as inputs for the PI-controllers. The resultant current response takes a few cycles to settle at the new power level but is sufficiently good for these kind of applications.

It is also worth pointing out how the sudden step change in active power results in a fluctuation of reactive power, which reflects the coupling of the two in (5). A step-up change in δ results in a drop of reactive power Q. This is counteracted by the i_d feedback control loop by increasing $|V_i|$ till Q again has settled at zero. The increase in $|V_i|$ will give a further increase in active power, and finally the system settles at a new steady-state point. The inverse phenomena occur for the step-down case. For step changes within these ranges, P and Q can confidently be treated by separate control loops with relatively stable outputs. The response time can be decreased further by tuning the PI-coefficients more aggressively. If the oscillations between P and Q become an issue, feed-forward cross-coupling terms may be introduced to make the control loop more dynamic.

8. Conclusions

In this paper, the control system for a grid-connected current-control VSI has been designed and implemented on an FPGA, which differs from the more conventional microprocessor-based implementations. The system has proven viable for a grid-connected VSI supplying the local grid with 30 kW and full reactive power control. The development of FPGA technology, especially the generic DSP-blocks, makes this a competitive control hardware and is very suitable for development projects. Also, a complementary system may be used if heavier computations or data logging is required. In the article, a RT-controller is used for, for example, harmonic analysis, burst log, and fault detection.

A PLL is implemented and shows steady phase tracking of the grid voltage. Grid voltages have been simulated with

added harmonic contents and noise, still maintaining a stable phase tracking in the PLL.

There is a minor unbalance between the phase currents, due to an unbalance in the LCL-filter. This can be relieved by individual phase control of the inverter. Step responses have been performed in active power, while maintaining the reactive power at zero. Due to the lower-order harmonic content of the grid currents, a moving average filter is added to get smoother i_{dq} control variables. This results in a stable but slightly slower step response. In the power range evaluated, P- and Q-control can be treated with uncoupled current control loops, still maintaining stable step responses. To reduce the oscillations further, feed-forward cross-coupling terms have to be added between P and Q.

References

[1] K. D. Brabandere, J. V. den Keybus, B. Bolsens, J. Driesen, and R. Belmans, "Sampled-data dual-band hysteresis current control of three-phase voltage source inverters," in *Proceedings of the 15th International Conference of Electrical Machines (ICEM '02)*, Brugge, Belgium, 2002.

[2] K. D. Brabandere, J. van den Keybus, B. Bolsens, J. Driesen, and R. Belmans, "FPGA-based current control of PWM voltage source inverters," in *Proceedings of the 10th European Power Electronics and Drives Association*, Toulouse, France, 2003.

[3] D. M. Brod and D. W. Novotny, "Current control of VSI-PWM inverters," *IEEE Transactions on Industry Applications*, vol. 21, no. 3, pp. 562–569, 1985.

[4] M. P. Kazmierkowski and L. Malesani, "Current control techniques for three-phase voltage-source pwm converters: a survey," *IEEE Transactions on Industrial Electronics*, vol. 45, no. 5, pp. 691–703, 1998.

[5] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 2, pp. 203–215, 2007.

[6] E. Monmasson and M. N. Cirstea, "FPGA design methodology for industrial control systems: a review," *IEEE Transactions on Industrial Electronics*, vol. 54, no. 4, pp. 1824–1842, 2007.

[7] Z. Fang, J. E. Carletta, and R. J. Veillette, "A methodology for FPGA-based control implementation," *IEEE Transactions on Control Systems Technology*, vol. 13, no. 6, pp. 977–987, 2005.

[8] M.-W. Naouar, E. Monmasson, A. A. Naassani, I. Slama-Belkhodja, and N. Patin, "FPGA-based current controllers for AC machine drives: a review," *IEEE Transactions on Industrial Electronics*, vol. 54, no. 4, pp. 1907–1925, 2007.

[9] A. Knop and F. W. Fuchs, "High frequency grid impedance analysis with three-phase converter and FPGA based tolerance band controller," in *Proceedings of the 6th International Conference-Workshop on Compatability and Power Electronics (CPE '09)*, pp. 286–291, May 2009.

[10] T. Ostrem, W. Sulkowski, L. E. Norum, and C. Wang, "Grid connected photovoltaic (PV) inverter with robust phase-locked loop (PLL)," in *Proceedings of the IEEE PES Transmission and Distribution Conference and Exposition Latin America (TDC '06)*, Caracas, Venezuela, August 2006.

[11] Y.-Y. Tzou and H.-J. Hsu, "FPGA realization of space-vector PWM control IC for three-phase PWM inverters," *IEEE Transactions on Power Electronics*, vol. 12, no. 6, pp. 953–963, 1997.

[12] H. Abu-Rub, J. Guziński, Z. Krzeminski, and H. A. Toliyat, "Predictive current control of voltage-source inverters," *IEEE Transactions on Industrial Electronics*, vol. 51, no. 3, pp. 585–593, 2004.

[13] M. Hamouda, H. F. Blanchette, K. Al-Haddad, and F. Fnaiech, "An efficient DSP-FPGA-based real-time implementation method of SVM algorithms for an indirect matrix converter," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 11, pp. 5024–5031, 2011.

[14] A. M. Omar and N. A. Rahim, "FPGA-based ASIC design of the three-phase synchronous PWM flyback converter," *IEE Proceedings*, vol. 150, no. 3, pp. 263–268, 2003.

[15] Ó. López, J. Álvarez, J. Doval-Gandoy et al., "Comparison of the FPGA implementation of two multilevel space vector PWM algorithms," *IEEE Transactions on Industrial Electronics*, vol. 55, no. 4, pp. 1537–1547, 2008.

[16] S. Karimi, P. Poure, and S. Saadate, "Fast power switch failure detection for fault tolerant voltage source inverters using FPGA," *IET Power Electronics*, vol. 2, no. 4, pp. 346–354, 2009.

[17] Z. Salcic, J. Cao, and S. K. Nguang, "A floating-point FPGA-based self-tuning regulator," *IEEE Transactions on Industrial Electronics*, vol. 53, no. 2, pp. 693–704, 2006.

[18] C. H. Ho, C. W. Yu, P. Leong, W. Luk, and S. J. E. Wilton, "Floating-point FPGA: Architecture and modeling," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 17, no. 12, pp. 1709–1718, 2009.

[19] D. Menard and O. Sentieys, "Automatic evaluation of the accuracy of fixed-point algorithms," in *Proceedings of the IEEE/ACM Conference on Design, Automation and Test*, 2002.

[20] R. Andraka, "A survey of CORDIC algorithms for FPGA based computers," in *Proceedings of the ACM/SIGDA 6th International Symposium on Field Programmable Gate Arrays (FPGA '98)*, pp. 191–200, February 1998.

[21] M. S. Pádua, S. M. Deckmann, G. S. Sperandio, F. P. Marafão, and D. Colón, "Comparative analysis of synchronization algorithms based on PLL, RDFT and Kalman filter," in *Proceedings of the IEEE International Symposium on Industrial Electronics (ISIE '07)*, pp. 964–970, June 2007.

[22] V. Kaura and V. Blasko, "Operation of a phase locked loop system under distorted utility conditions," *IEEE Transactions on Industry Applications*, vol. 33, no. 1, pp. 58–63, 1997.

[23] S.-K. Chung, "Phase-locked loop for grid-connected three-phase power conversion systems," *IEE Proceedings*, vol. 147, no. 3, pp. 213–219, 2000.

[24] J. Bradshaw, U. Madawala, and N. Patel, "Bit-stream implementation of a phase-locked loop," *IET Power Electronics*, vol. 4, no. 1, pp. 11–20, 2011.

[25] R. Park, "Two reaction theory of synchronous machines," *AIEE Transactions*, vol. 48, no. 4, pp. 716–730, 1929.

[26] N. Mohan, T. Undeland, and W. Robbins, *Power Electronics—Converters, Applications and Design*, John Wiley & Sons, New Dehli, India, 5th edition, 2007.

[27] "Xilinx data book," 2006, http://www.xilinx.com.

[28] D. G. Holmes and T. A. Lipo, *Pulse Width Modulation for Power Converters*, Principles and Practice, IEEE Press, 10th edition, 2003.