CHAPTER 5
FPGA SIMULATION AND IMPLEMENTATION

5.1 INTRODUCTION

Nowadays, the implementation of modern controllers for electrical drives demands a perfect satisfaction of the required performances. The most important performance criteria are as follows:

- High-level integration and density of the used target device for the control implementation
- Embedded low-cost systems based on fully integrated controllers, which ensure several control tasks by the same device
- Use of high performance algorithms
- Flexibility to modify controller strategy and parameters
- Fast implementation time by using an appropriate design methodology
- Fast real-time computation
- Reliability, accuracy, and safety in harsh environments

In order to satisfy most of these control requirements, recent research studies proved that digital hardware solutions, such as field programmable gate array (FPGA), are an appropriate alternative over software solutions (DSP and micro controllers) and analog solutions.
(Monmasson et al 2007a, Monmasson et al 2011, Rodriguez et al 2007, Monmasson et al 2011a, Dubey et al 2007). If the prototyping platform uses a FPGA to run the algorithm, a newly created simulation tool can be used not only to simulate exactly the hardware but also to automatically generate the VHDL code needed for the implementation. This software is the Xilinx System Generator (XSG), a toolbox working in MATLAB/SIMULINK environment. It provides the user with a library of blocks representing functions which can be implemented in an FPGA. It can also convert the created model into VHDL code (Mailloux et al 2007, Ricci et al 2002, Martar et al 2005, Le-Huy et al 2006, Parma et al 2007).

5.2 OVERVIEW OF FPGA

A Field Programmable Device (FPD) is a general term that refers to any type of integrated circuit used for implementing digital hardware, where the chip can be configured by the end users to realize different designs. Programming of such a device usually involves interfacing the device with specially designed programming software. Field programmable gate arrays (FPGAs) are digital integrated circuits that contain configurable (programmable) blocks of logic along with configurable interconnects between these blocks. Design engineers can configure (program) such devices to perform a tremendous variety of tasks.

The “field programmable” portion of the FPGA’s name refers to the fact that its programming takes place “in the field” (as opposed to devices whose internal functionality is hard wired by the manufacturer). A major advantage of this technology is that FPGAs can execute codes in parallel whereas Digital Signal Processors (DSPs) execute codes in series. Hence, FPGAs do not have to “store” as much data as DSP’s and the need for large amounts of Random Access Memory (RAM) degrades significantly. Another advantage of using FPGAs is their ability to work with whatever word length
the programmer chooses. Whereas DSP processors must be selected to handle the longest word length that occurs in the code (thereby reducing efficiency when processing shorter word lengths). FPGAs allow greater flexibility and efficiency by utilizing the smallest necessary word lengths (Cecati 1999, Berto et al 2003).

The historical roots of the FPGA are in the Complex Programmable Logic Devices (CPLD) of the early 1980s. The CPLD contained several thousand logic gates, while the FPGA can contain over a million. The first FPGA was introduced in 1985 by XILINX Corporation. The primary difference between the CPLD and FPGA is architecture. A CPLD is restrictive structure consisting of one or more programmable sum-of-products logic arrays feeding a clock register. The FPGA is dominated by interconnects and high level embedded functions. The process is totally automatic except the choice of the FPGA model required. Normally, the FPGA contains some other functions in addition to the algorithm. Therefore the algorithm must be connected to the remainder of the VHDL code. The last step is the synthesis and implementation of the VHDL code. This is done with Xilinx software which requires a few simple steps. After this, a configuration file is obtained permitting programming the FPGA. If some modifications of the system are required, only some steps need to be done again so the process will be much faster (Naouar 2004, Chapuis et al 2004, Riesgo 1999, Cirstea 2005, Pimentel and Le-huy 2000, Perry 2004).

5.3 ARCHITECTURAL OVERVIEW OF FPGA

The Spartan-3E FPGA family architecture consists of five fundamental programmable functional elements:

- Configurable Logic Blocks (CLBs) contain flexible Look-Up Tables (LUTs) that implement logic plus storage elements
used as flip-flops or latches. CLBs perform a wide variety of logical functions as well as store data.

- Input/Output Blocks (IOBs) control the flow of data between the I/O pins and the internal logic of the device. Each IOB supports bidirectional data flow plus 3-state operation. Supports a variety of signal standards, including four high performance differential standards.

- Block Random Access Memory (RAM) provides data storage in the form of 18-K bit dual-port blocks.

- Multiplier Blocks accept two 18-bit binary numbers as inputs and calculate the product.

- Digital Clock Manager (DCM) blocks provide self-calibrating, fully digital solutions for distributing, delaying, multiplying, dividing and phase shifting clock signals.

Figure 5.1 Structure of an FPGA
An illustration of a typical FPGA architecture appears in Figure 5.1; it consists of a sea of Logic Blocks (LBs), an interconnection network and configurable I/O blocks. As the only type of field programmable device (FPD) that supports very high logic capacity, FPGAs have been responsible for a major shift in the way digital circuits are designed. The internal structure, the operating mode and the configuration of these FPGA elements differ depending on the device family and technology.

The Xilinx Spartan 3E XCS100E reduces system cost by offering the lowest cost-per-logic of any FPGA family, supporting the lowest-cost configuration solutions including commodity serial (SPI) and parallel flash memories, and efficiently integrating the functions of many chips into a single FPGA.

- **Advanced Low-Cost Features**
  - Five devices with 100K to 1.6M system gates
  - From 66 to 376 I/Os with package and density migration
  - Up to 648 Kbits of block RAM and up to 231 Kbits of distributed RAM
  - Up to 36 embedded 18x18 multipliers for high performance applications
  - Up to eight Digital Clock Managers

- **Cost-Saving System Interfaces and Solutions**
  - Support for Xilinx Platform Flash as well as commodity serial (SPI) and byte-wide flash memory for configuration
  - Support for 18 common I/O standards, including PCI 33/66, PCI-X, Mini-LVDS and RSDS
Industry-Leading Design Tools and IP

- ISE design tools to shorten design and verification time
- Hundreds of pre-verified, pre-optimized Intellectual Property (IP) cores and property designs
- ChipScope Pro™ system-debugging environment

For this thesis, a Xilinx Spartan 3E™ development board incorporating a XCS100E FPGA was utilized and is shown in Figure 5.2.

5.4 DESCRIPTION OF XILINX SYSTEM GENERATOR 12.3

MATLAB/Simulink consists of Xilinx System Generator (XSG) Blockset, which is developed by Xilinx. Due to development of XSG Blockset the use of high level programming tool for FPGA users become very easy. The XSG Blockset consists of a library of FPGA blocks, required for
buildings of model. It also consists of a VHDL code generator which automatically generates VHDL code from the created models. The XSG Blockset can work only with Xilinx FPGAs. For implementation of VHDL code in an FPGA, additional conversions are required.

The main advantage of using the XSG Blockset to simulate the control algorithm is the possibility to generate code that can be used to program an FPGA directly from the simulation model. Furthermore, the implemented code is guaranteed to perform exactly as in the simulation because the XSG Blockset represents exactly the FPGA hardware. This reduces the implementation time because the algorithm needs to be modeled and simulated only once.

- The XSG Blockset library contains many blocks that are classified as follows:
  - Basic Elements: Contains simple signal processing functions (delay, multiplexers, registers...).
  - Math: Contains common mathematical functions (add, multiplication, relational, negate...).
  - MATLAB I/O: Contains blocks to connect the XSG section to the Simulink model and determine the quantization error created by the use of fixed-point values.
  - Memory: Contains memory elements found in FPGA (RAM, ROM, FIFO...)
  - DSP: Contains DSP like math functions (FIR, FFT...)

Many of these blocks are replicas of standard Simulink blocks. So it is easy to build a model same way than with standard Simulink blocks. To connect the XSG algorithm to the remainder of the Simulink model, two
special blocks are used. "Gateway In" transforms an incoming floating point signal to a fixed-point one with a fixed time step. "Gateway Out" is used to return a value to the normal Simulink environment. These two blocks represent the boundaries of the XSG model.

5.5 MODELING AND SIMULATION OF PROPOSED SCHEME USING MATLAB/SIMULINK AND XSG

The formation of a Simulink model for the power circuit (including the power converter and electric motor) and the control algorithm requires few steps which can be separated in two main parts.

- Design and testing of the system in MATLAB environment.
- Generation and implementation of the VHDL code in the FPGA.

The detailed steps are shown in the flow diagram in Figure 5.3. The first step is to determine the parameters of the power electronic section and the use them to create a model using the PSB blocks. Also the hardware section of the prototyping board like the ADC can be simulated as an exact replica of the real system. The next step is the building of the control algorithm with the use of only Xilinx System Generator blocks. It is then connected to the simulated power system by the "Gateway In" and "Gateway Out" blocks. Figure 5.4 shows the interconnection between the control and the power circuit. The next step is the simulation of the complete system in Simulink and the VHDL code generation which is done with the "Xilinx System Generator" blocks.
Figure 5.3 Design and implementation process

Figure 5.4 Interconnection between the control system and power circuit
The last step is the synthesis and implementation of the VHDL code. It is done with Xilinx software and needs only a few steps. After this step, a configuration file is obtained permitting programming the FPGA. In case modifications of the systems are required, not all these steps are to be done again so the process will be much faster.

5.6 MODELING OF PROPOSED SVM-DTC SCHEME USING XILINX SYSTEM GENERATOR

Xilinx System Generator Tool developed for MATLAB/SIMULINK package is widely used for algorithm development and verification purposes in DSPs and FPGAs. System Generator Tool allows an abstraction level algorithm development while keeping the traditional SIMULINK blocksets, but at the same time automatically translating designs into hardware implementations that are faithful, synthesizable, and efficient (Rodriguez-Andina et al 2007, Tzou and Hsu 1997, Castro et al 2003, Dufour et al 2006).

The design stages for required arithmetic and logical operations for the proposed are carried out in a hierarchical and modular fashion. The modules of proposed scheme can be described as follows:

- Torque and Flux Estimator with Low Pass Filter (LPF)
- PI Controller
- Space Vector Pulse Width Modulator

5.6.1 Modeling of Torque and Flux Estimator with LPF

The XSG and MATLAB/SIMULINK design of torque and flux estimator module and its submodules of proposed SVM-DTC are shown in Figure 5.5 (a)-(d).
Figure 5.5 Modeling of torque and flux estimator (a) XSG and MATLAB/SIMULINK model of torque and flux estimator (b) Modeling of current transformation subsystem with XSG (c) Modeling of voltage transformation subsystem with XSG (d) Modeling of LPF subsystem
Figure 5.5 (Continued)
5.6.2  Modeling of PI Controller Module

PI controller is modeled using Xilinx System Generator as shown in Figure 5.6. Very complex designs can be made easily with graphical algorithm approach by using the Xilinx System Generator in MATLAB/SIMULINK environment.

Figure 5.6 Modeling of PI controller

5.6.3  Modeling of SVPWM Module

The Xilinx block configuration that produced the space vector modulation for this controller is presented in Figure 5.7(a). The input magnitude and theta value that was converted into a binary value was sent into a sample and hold block as shown in Figure 5.7(b), that was then sent into a Mcode block that selects the appropriate sector of the space vector modulation hexagon.
Figure 5.7 Modeling of SVPWM  
(a) XSG model of SVPWM  
(b) Sample and hold subsystem
5.7 FPGA SIMULATION RESULTS OF PROPOSED SCHEME

The proposed SVM-DTC with LPF based torque and flux estimator is simulated using Xilinx inbuilt simulator ISim. The simulation result of LPF based torque and flux estimator is shown in Figure 5.8. The simulation result of PI controller is shown in Figure 5.9 and the switching patterns at various sectors of SVPWM are shown in Figure 5.9(a)-(c). The XSG project navigator is given in Appendix 6 and the synthesis report is given in Appendix 7.

Figure 5.8 FPGA simulation result of torque and flux estimator

Figure 5.9 FPGA simulation result of PI controller
Figure 5.10 Switching pattern of SVPWM (a) Switching pattern at sector 1 and 2 (b) Switching pattern at sector 3 and 4 (c) Switching pattern at sector 5 and 6
5.8 SYNTHESIS RESULTS OF PROPOSED SCHEME

The above designed model is synthesized using inbuilt Xilinx Synthesis Tool (XST) and the implementation process is done using Xilinx PlanAhead software. The simulation of FPGA internal structure and I/O pins of proposed SVM-DTC are as shown in Figure 5.11(a) and (b). The VHDL code for proposed scheme is given in Appendix 8.

Figure 5.11 Synthesis results of proposed SVM-DTC (a) Internal structure of FPGA (b) I/O pins of FPGA
5.9 FPGA IMPLEMENTATION RESULTS

The whole FPGA implementation is done in two phases. In the first phase, SVPWM is implemented on Spartan 3E FPGA and in the second phase, the whole proposed SVM-DTC scheme is implemented.

5.9.1 FPGA Implementation of SVPWM

Xilinx System Generator model was developed to examine the three switching patterns of SVPWM method. This model is synthesized using Xilinx ISE® and the VHDL code was developed. The switching delays and the forward drop of the power switches and the dead time of the inverter are all neglected in the models. It is assumed that the model load has a fixed $R_L = 100\Omega$, $L=1\text{mH}$ and the voltage source of the inverter is $150\text{ V}_{dc}$, inverter switching frequency $f_s = 20000\text{Hz}$. Figure 5.12 shows the experimental setup of this work.

![Experimental setup of FPGA implementation of SVPWM](image)

**Figure 5.12** Experimental setup of FPGA implementation of SVPWM
5.9.2 FPGA Implementation Results of SVPWM

The FPGA implementation results of SVPWM are shown in Figure 5.12.

Figure 5.12 Experimental results of SVPWM (a) Pulse pattern generated by FPGA at Sector 1 (S1) (b) Pulse pattern generated by FPGA at Sector 1 (S2) (c) Driving pulse for inverter
5.9.3 FPGA Implementation of Proposed SVM-DTC Scheme

The experimental setup of whole proposed SVM-DTC scheme is depicted in Figure 5.14. In order to illustrate the efficiency of the proposed SVM-DTC are carried out on a Spartan 3E FPGA. The experimental setup consists of induction motor with loading arrangement. The motor is fed by the IGBT based three phase PWM inverter controlled directly by FPGA board. The data acquisition card (DAQ) is used to collect the parameters such as current, voltage and torque. LabVIEW software is used to display the measured parameters. Details of the experimental setup are given in Appendix 9.

![Figure 5.14 Experimental setup of proposed SVM-DTC scheme](image-url)
5.9.3.1 Xilinx Spartan 3E XCS100E FPGA

For this experimental setup, a Xilinx Spartan 3E development board incorporating a XCS100E FPGA was utilized. Xilinx, a leading manufacturer of FPGA’s, primarily builds array-based circuits. These circuits incorporated chips comprised of two dimensional arrays of logic blocks that can be interconnected via horizontal and vertical routing channels. Here are the most relevant features of FPGA.

- 100K-250K gate
- 72K bits of fast dual-port block RAM
- 500 MHz operation
- USB2 full speed port for FPGA configuration and data transfers
- XCF02 platform Flash ROM that stores FPGA configurations
- User settable oscillator frequency (25, 50, and 100 MHz)
- Three on-board voltage regulators (1.2 V, 2.5 V, and 3.3 V)
- 8 LEDs
- 4-digit seven segment display
- 4 push buttons
- 8 slide switches
- 8 bit VGA port
- PS/2 port
5.9.3.2 Xilinx System Generator – ISE

The design process is summarized as follows: A Matlab-Simulink is developed to simulate power electronics elements. Elements of model internal to the FPGA are designed using the Xilinx System Generator library. The rest of the system is designed using Simulink Library blocks. Once the system is modeled, VHDL code is generated for the portion of the simulation controlled by the FPGA. After the VHDL is generated, the project is loaded into ISE Foundation software and the design is synthesized. Then the program is uploaded into the FPGA through the JTAG cable, and ChipScope Pro is used to communicate with the target hardware.

The Computer Aided Design Architecture of FPGA based design and implementation is shown in Figure 5.15.

![Figure 5.15 Computer Aided Design Architecture](image)

*Simulink simulation of power electronics and electric machines (using a Xilinx blockset inside Simulink)*

*System Generator produces VHDL code*

*Xilinx ISE Foundation generates programming file for FPGA from VHDL*

*JTAG cable*

*Xilinx XC4VLX25 FPGA is programmed through the JTAG programming cable*

*Control and data acquisition from FPGA via JTAG cable using ChipScope Pro software*

*Digital control of target hardware through custom interface card (includes A/D converters and digital I/O)*
5.9.3.3 VHDL Synthesis using ISE

The design is synthesized using ISE foundation software by opening the project file, generating the program file, and configuring the device. Once synthesized, the programming file is generated and the FPGA is programmed. The ISE foundation window is shown in Figure 5.16.
5.9.3.4 Synthesis Report

The table 5.1 describes the number of logic blocks used and device utilization of Spartan-3E FPGA when VHDL code for proposed SVM-DTC is written and synthesized using Xilinx ISE software. The synthesis report is shown in Table 5.1.

Table 5.1 Synthesis Report

<table>
<thead>
<tr>
<th>*</th>
<th>Final Report</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTL Top Level Output File Name</td>
<td>jc2_top.ngr</td>
</tr>
<tr>
<td>Top Level Output File Name</td>
<td>jc2_top</td>
</tr>
<tr>
<td>Optimization Goal</td>
<td>Speed</td>
</tr>
<tr>
<td>Keep Hierarchy</td>
<td>Yes</td>
</tr>
<tr>
<td>Target Technology</td>
<td>XC9500XL CPLDs</td>
</tr>
<tr>
<td>Macro Preserve</td>
<td>YES</td>
</tr>
<tr>
<td>XOR Preserve</td>
<td>YES</td>
</tr>
<tr>
<td>Clock Enable</td>
<td>YES</td>
</tr>
<tr>
<td>wysiwyg</td>
<td>NO</td>
</tr>
<tr>
<td>Design Statistics</td>
<td></td>
</tr>
<tr>
<td>#IOs</td>
<td>8</td>
</tr>
<tr>
<td>Cell Usage:</td>
<td></td>
</tr>
<tr>
<td>#BELS</td>
<td>24</td>
</tr>
<tr>
<td># AND2</td>
<td>8</td>
</tr>
<tr>
<td># GND</td>
<td>1</td>
</tr>
<tr>
<td># INV</td>
<td>9</td>
</tr>
<tr>
<td># OR2</td>
<td>6</td>
</tr>
<tr>
<td>#FlipFlops/Latches</td>
<td>6</td>
</tr>
<tr>
<td># FDCE</td>
<td>6</td>
</tr>
<tr>
<td>#10 Buffers</td>
<td>8</td>
</tr>
<tr>
<td># IBUF</td>
<td>4</td>
</tr>
<tr>
<td># OBUF</td>
<td>4</td>
</tr>
</tbody>
</table>

Total REAL time to Xst completion: 3.00 secs
Total CPU time to Xst completion: 2.98 secs
Total memory usage is 128884 kilobytes
Number of errors : 0 ( 0 filtered)
Number of warnings: 0 ( 0 filtered)
Number of infos : 0 ( 0 filtered)
5.9.4 Steady State Torque Response of Proposed SVM-DTC Scheme

The steady state response of proposed SVM-DTC at different speed is as shown in Figure 5.17.

Figure 5.17 Steady state torque response of proposed SVM-DTC scheme
(a) At 500 rpm (b) At 1000 rpm (c) At 1440 rpm
5.9.5 Three Phase Currents of Proposed SVM-DTC Scheme

The three phase currents of proposed SVM-DTC at different speed are as shown in Figure 5.18.

Figure 5.18 Three phase currents of proposed SVM-DTC scheme
(a) At 500 rpm (b) At 1000 rpm (c) At 1440 rpm
In order to validate the implementation of proposed SVM-DTC with LPF based torque and flux estimator, the experimental results are compared with simulation results. Both simulation and experimental results are same, thus the validity of the implementation of proposed SVM-DTC with LPF based estimator is achieved.

5.10 SUMMARY

In this chapter, FPGA simulation and implementation of the proposed SVM-DTC scheme has been presented. The proposed scheme is simulated using Xilinx System Generator in MATLAB/SIMULINK environment and implemented on a Xilinx Spartan 3E FPGA. It is found that the proposed scheme has reduced torque ripples and current distortions which is an advantage of FOC and the proposed scheme is able to maintain switching frequency at 20 kHz. It is also shown that the dynamic performance of the proposed controllers is as good as the conventional DTC scheme. The experimental results obtained have shown proof for the benefits of using FPGA solutions for the improvement of control performance of AC drives.