The 12th IEEE International Workshop on Advanced Motion Control
March 25-27, 2012, Sarajevo, Bosnia and Herzegovina

Event-Driven Approach to Control Mechatronic System with FPGA

Robert Horvat
University of Maribor
Faculty of Electrical Engineering and Computer Science
2000 Maribor, Slovenia
robert.horvat@uni-mb.si

Karel Jezernik, Milan Čurkovič
University of Maribor
Faculty of Electrical Engineering and Computer Science
2000 Maribor, Slovenia
karel.jezernik@uni-mb.si
milan.curkovic@uni-mb.si

Abstract— **A mechatronic system consists of a mechanical system and electric actuators. The event-driven control of a mechatronic system was implemented on a field-programmable gate-array (FPGA) platform. The supervisor provides robust, safe, and transparent control, where the FSM defines all the possible directions for implementation. In order make supervisor more transparent, the FSM was divided into three main parts, with three main colours (green, yellow and red - semaphore). These colours indicate the condition of the system. The supervisor was upgraded with a Graphic user interface (GUI) with indicators that directly show the state of the FSM. The GUI includes additional logical I/O signals, in order to make system more useful. The supervisor is executed parallel with the basic motor control on the FPGA. This paper presents the robust current controller of the brushless ac (BLAC) motor, upgraded with a classical PI velocity controller.**

The application of the proposed ECA-based method is illustrated using the example of the FSM motion control of a BLAC motor with integrated I/O signals.

Keywords-component; Mechatronic system, FPGA, FSM, ECA rules, Supervisor, BLAC motor, GUI

I. INTRODUCTION

FPGA has been increasingly used in the field of mechatronic systems over the last few years, due to high dynamics and complexities of the system. FPGA brings a new approach to the control and protection of the mechatronic system by its short-time cycles and parallel implementation of processes.

This paper presents the event driven control of a mechatronic system upgraded with Graphic user interface (GUI). The main point of the mechatronic system is for it to be safe, robust, effective, and user friendly. Different applications with velocities and positionally-controlled motors are used in most mechatronic systems. Additional external signals are used according to different applications. Three phase brush less motors with alternating current (BLAC) are mostly used for high-efficiency, good dynamics, and the benefits of maintenance, and a wide-range of power. A three phase inverter allows amplitude and phase set up of the motors three phase voltage. The three parts of the system (motor, three phase inverter, and controller) are included for controlling the mechatronic system.

Management with a finite-state machine provides a simple solution for algorithm implementation on the FPGA. A finite-state machine predefined program implementation over all possible directions. The management presentation is based on three main colours (green, yellow, and red). We strive have to make this system transparent, just like the semaphore. The system's condition shows three main colours identifications.

Communication between the system and the user is realized using GUI that directly shows the actual state of the finite-state machine. The main point of GUI is to show the system condition's to the user in the most transparent way.

This system requires a short response-time of the controller, and also fast dynamic management. Digital signal-processors (DSP) or PLCs are normally used for these types of applications. The use of field-programmable gate arrays (FPGA) for the controller and supervisor is the best solution for avoiding problems with any short response times of the controller.

The FPGA's more important advantages are its parallel execution and short-time delay. The controller, supervisor and other functions are implemented simultaneously. DSP processor's process execution-time delay is about 100 μs, but on FPGA it is some ns, independent of the number of processes.

The FPGA structure requires a new approach for controller implementation. We strive have to implement most of the algorithmic instructions regarding logical operations. Those performances with transfer functions need to be replaced with logical functions.

II. MECHATRONIC SYSTEM CONFIGURATION

Mechatronic system configuration consists of three main parts, where different types of signals are connected (Fig. 1). The actuator (three phase motor) that drives the mechanical system is in the bottom-layer. The driver for the three-phase inverter is in the middle-layer, and the top-layer includes the main controller, the management, and the protection function. The system's configuration is the combination of a discrete-event system and a continuous time-system.

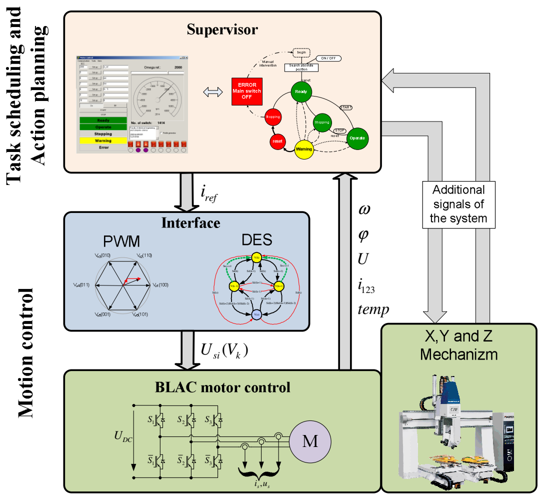


Figure 1. Mechatronic system's configuration

A. *Task scheduling and action planning – Supervisor*

The primary task of supervisor is to provide a stable, robust, and effective system. With a finite state machine are defined all possible transitions between states. When the ECA rules are included to implementing a finite-state machine, system becomes event driven system. Any transition can be done, but only when the event is recognized. FPGA implementation allows for event recognition within the range of some nano-seconds, therefore the warning and error detection is, as far as possible. The short time of calculation on the FPGA allows for an affect on the three-phase inverter, in a very short time.

The supervisor also needs to communicate with the user. An FSM of the supervisor was made using the three main colours (green, yellow and red), in order to make communication more transparent. The colours are the same as for semaphore.

The same theory is used in management, where some limits define which value means warning, and which error of the system. FSM was created with these main colours, where each color tells the user at which stage (state) the system is. When the system is in the green area, then the user knows that the system is operating normally. In the yellow area the user knows that one of the critical values exceeds the first limit. But within the yellow area system still operates normally.

The current amplitude, motor speed, DC link voltage, and limit switches are monitored by the supervisor. The supervisor can affect the parameters and the reference values when any of the measured values is critical. If the system overloads, the supervisor immediately initiates a procedure to protect the system. The protection algorithm starts by modifying the parameters and reference values. When modifying the parameters and reference values cannot solve the problem, the supervisor affects the state of the inverter directly, as shown in the state-diagram in Fig. 2.

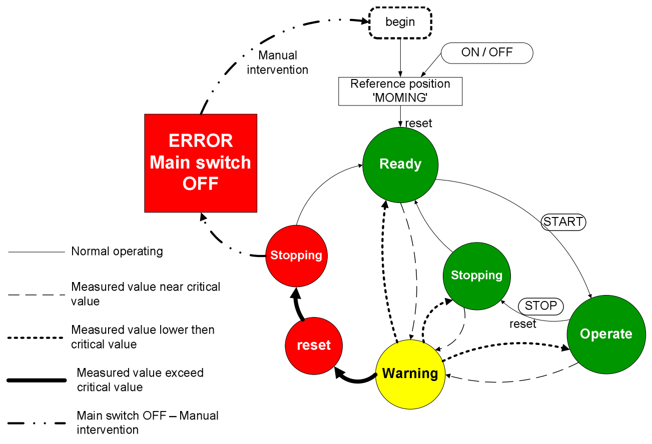

begin
Manual intervention
ON / OFF
Reference position 'MOMING'
reset
ERROR Main switch OFF
Ready
START
Stopping
STOP
reset
Operate
Warning
Stopping
reset
Normal operating
Measured value near critical value
Measured value lower then critical value
Measured value exceed critical value
Main switch OFF – Manual intervention

Figure 2. FSM of supervisor

The steering function determines the mechatronic system's operational mode. The three main possible modes are of concern: Ready, Operate, or Error. They are considered as the discrete states of the monitoring function. Initially, the Begin mode is active. Turning the main switch (ON/OFF) to ON, enables homing. That is a condition for the transition to state Ready. From the state Ready, it is possible to start the system (Operate) by pressing the START button. Operate mode allows the change of all "free" parameters and reference values. Stopping of the system is possible by pressing the STOP button. The stopping mode reset reference values. If the rotor is stopped, the system reverts to the Ready mode. In these three states, Ready, Operate, and Stopping, the system constantly monitors the values of current, the inverter voltage, and limit switches and rotor speed. If any of these values are outside the permitted quantities, the system transfers to Warning mode. The warning mode is intended to alert the user that the system is within the limit range of the safe system's activity. In the state Reset, all parameters reset to the initial values, and reset reference value of currents and velocity. This is the condition for transition to Stopping mode. If the stopping system is successful, then the system returns to the state Ready, otherwise it turns off the main switch. For the system to reboot, the user should eliminate the error that causes a critical state of the system. An example regarding the VHDL implementation of FSM for FPGA, is shown in Fig. 3.

```
process (state, in1, int2, int3, in4)
begin
   case state is
      when s0 =>           -- STATE S0
         out1 <= "00";
         if in1 = '0' then -- wait for event int1
            next_state <= s0;
         else
            next_state <= s1;
         end if;
      when s1 =>           -- STATE S1
```

Figure 3. FSM implementation on FPGA

The GUI – graphic user interface on the PC is written in Delphi programming language (Fig. 4). It allows for simple, transparent, robust, and safe communication with the user. The lights on GUI directly allow for the monitoring states of the supervisor FSM. The states of supervisor FSM are marked by the same three colours that tell the user the condition of the system. GUI allows for setup reference values, and changes the parameters.

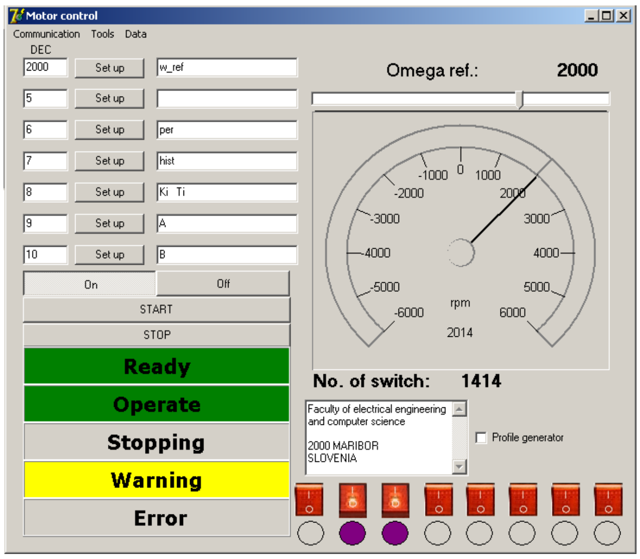


Figure 4. a) GUI - Graphic user interface

B. *Motion control*

In many cases electric drivers are used to ensure correct motion of the mechatronic system (conveyor belt). An appropriate electric drive is used Depending of maximum speed, force, and some external conditions of the mechatronic system.

The drive consists of a brushless AC motor with an attached incremental encoder and gear. Four position switches are attached to the drive, two limit position switches, the reference and home position switches in the middle.

C. *BLAC motor control*

High efficiency, small size – weight, over the last few years has required the use three-phase motors for changing the position of any CNC machine. Using the BLAC motor for the actuator requires minimal maintenance. A three-phase inverter is used to control BLAC motor. Here a combination of continuous and discontinuous signals is used. The measuring signals of currents, voltage, velocity, position, and temperature are continuous. Signals like end-switches, limit switches, and switches on three-phase inverter are discontinuous.

The BLAC motor combines many of the advantages of the permanent magnet-excited AC motor and the synchronous motor [2]. The BLAC motor needs a low-reactive current, which is very similar to the DC motor, and the current is proportional to the torque. The shaft-torque is therefore easy to estimate by detecting a three-phase current of the BLAC motor [6]. The BLAC motor is a combination of a permanent magnet synchronous motor's and a three-phase inverter [11]. The BLAC motor dynamics are governed through electrical current with

$$\frac{di_{si}}{dt} = \frac{1}{L_s}\left(u_i - R_s i_{si} - e_{si}\right); \quad \text{i=1,2,3} \qquad (1)$$

where i_{si} denotes the three-phase stator currents, u_{si} phase voltage, e_{si} EMF, R_s is the resistance and L_s is the inductance of the motor windings. The considered control problem of the BLAC motor's current control is the tracking of a three-phase current-reference signal. After the current control error is defined as $\Delta i_s = i_s - i_s^d$, (1) can be rewritten in the form of error dynamics

$$\frac{d\Delta i_{si}}{dt} + \frac{R_s}{L_s}\Delta i_{si} = \frac{1}{L_s}\left(u_{si}(V_k) - e_{si}\right) \qquad (2)$$

which contains the impacts of all the disturbances on the system. The signals of current i_{si}, speed ω and electromotive force e_{si} are continuous. The input voltage vector $u_{si}(V_k)$ is discontinuous and is a result of the switch position's combination in the VSI inverter.

The general control-scheme (Figure 5) shows the architecture of the speed / current regulations of the BLAC motor implemented into a FPGA [3]. The considered control is the tracking of a three-phase current reference signal, whose amplitude is determined by the output of the PI (proportional-integral) speed controller.

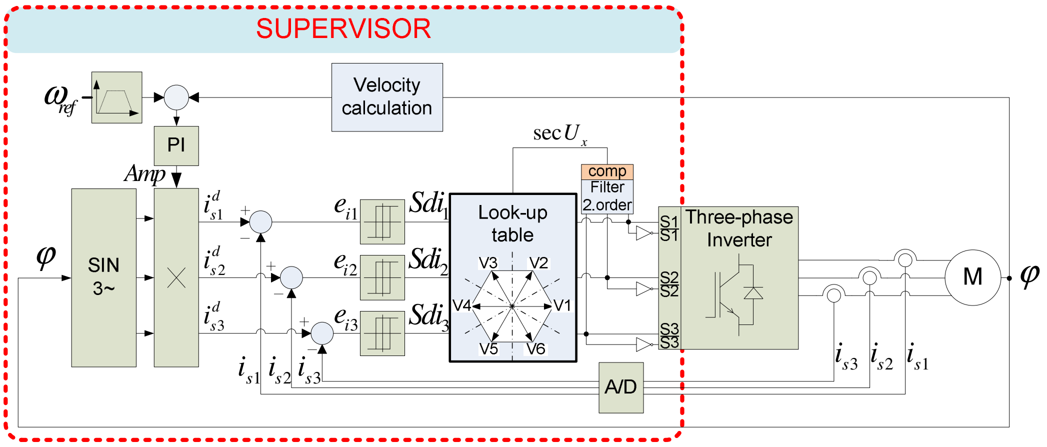


Figure 5. General control scheme of the BLAC motor

D. *Interface*

The basic principle of the current control is to manipulate the input voltage vectors $u_{si}(V_k)$ so that the desired current is produced by the inverter [4]. This is achieved by choosing an inverter-switch combination S_k that drives the stator current vector by directly applying the appropriate inverter voltages $u_{si}(V_k)$ to the BLAC motor's windings. The switch positions of the three-phase inverter are described using the logical variables V_k, depending on the states of the switches S_k which can be ON or OFF. The three-phase inverter can produce 2^3 of the voltage vector combinations; two zero vectors and 6 active vectors.

A DES system reacts only if an event is recognized [7], [9]. In order to control the current i_s, the sector of the drive voltage u_{si} is recognized first.

By considering the space-vector representation of the stator voltage $u_s(V_k)$, the voltage is presented as a vector rotating around the origin. The six active switching vectors of the three-phase transistor inverter represent the six active output-voltage vectors, denoted as $V_1 ... V_6$; V_0 and V_7 are the two zero voltage vectors. According to the signs of the phase voltages u_{s1}, u_{s2} and u_{s3}, the phase plane is divided into six sectors, denoted by Su1 … Su6.

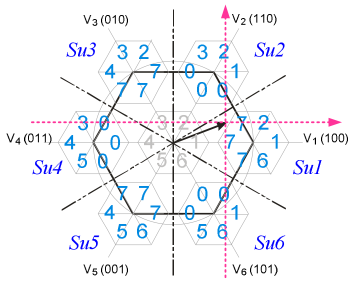
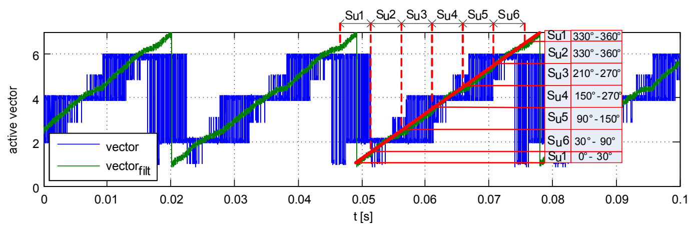


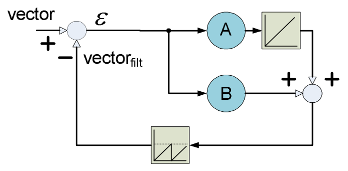
Figure 6. Stator voltage definition

In regard to the situation presented in Figure 6, the stator-voltage space vector u_s is in sector Su1. In this sector, the logical voltage vectors V_0, V_1, V_2, V_6 and V_7 are selected for the current control. V_0, V_7 are the two zero vectors, whilst V_1, V_2, V_6 are the three nearest adjacent live output voltage-vectors to this sector. Using the DES theory, the five output voltage vectors V_0, V_1, V_2, V_6 and V_7 are recognized as the discrete states of the system.

The proposed logical event-driven BLAC motor current control can be realized in the form described in Figure 5, where the signs of the current control-error are represented by sd_i, and the currently-active voltage sector by $\mathrm{sec}U_x$. The bit values of $\mathrm{sec}U_x$ represents the signs for each phase of stator voltage. The sign of the stator voltage is determined by the PLL filter that filters the active voltage vectors (Figure 7). The output filter that represents the position of the stator voltage (360°) is divided into six parts and indented for 30° (Figure 7a).



(a)



(b)

Figure 7. Determination of the stator voltage sign with second order PLL filter

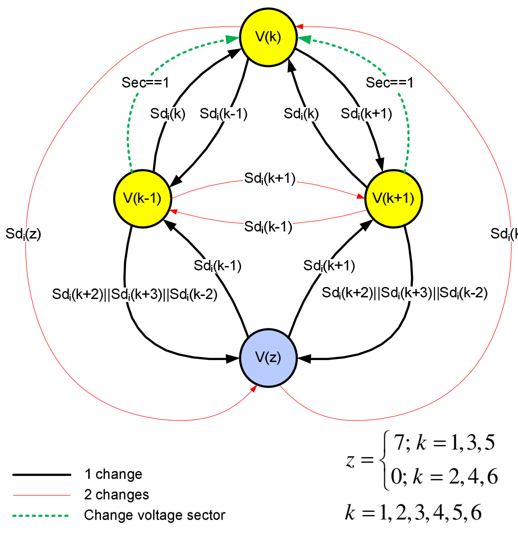
Based on the known sector, the input voltage vector $u_{si}(V_k)$ (the transistor switching pattern) for the current control is selected, by respecting the current control error Δi_{si} (2).

Finite state machine can be used to ensure a correct voltage vector $u_{si}(V_k)$ (Figure 8b). The FSM is implemented using three main states. Depending on the voltage-sector, the states of the FSM have different active voltage-vectors, and the zero vectors also change their value forms "000" to "111". The top state of the FSM represents an active voltage-vector, the states in the middle represent the adjacent active voltage vectors, and the bottom state represents a zero-voltage vector inside the voltage sector (Figure 8b). The conditions of the FSM are voltage sectors, and the current state. Any transition between states can be done when the defined event is recognized. The same can be implemented within the state-transition table (Figure 8a). The voltage vector is selected with the voltage sector, and outputs of the hysteresis current-controller. The voltage-sector is recognized (chose table column). The next step is waiting for an event that will decide which row will be used – decides the voltage vector. Active voltage vectors are marked with a blue background.

Most transitions are between the adjacent voltage-vectors and the zero voltage-vector. Consequently a zero voltage-vector inside the voltage sector is selected, according to the adjacent active voltage-vectors. For example, in voltage sector 2, the adjacent active voltage vectors are vectors V1(100) and V3(010). Both have only one active switch in the voltage inverter-leg. Therefore zero voltage vector V0(000) is selected in voltage sector 2. The transition between V1 $\leftrightarrow$ V0, and V3 $\leftrightarrow$ V0 needs to change only one switch in order to change the voltage-vector. This method provides most of transitions with only one inverter leg.

		secU$_x$ signDi	*Su1*	*Su2*	*Su3*	*Su4*	*Su5*	*Su6*
			100	**110**	**010**	**011**	**001**	**101**
events	*Sdi0*	**000**	V_0	V_0	V_0	V_0	V_0	V_0
	Sdi1	**100**	V_1	V_1	V_7	V_0	V_7	V_1
	Sdi2	**110**	V_2	V_2	V_2	V_0	V_7	V_0
	Sdi3	**010**	V_7	V_3	V_3	V_3	V_7	V_0
	Sdi4	**011**	V_7	V_0	V_4	V_4	V_4	V_0
	Sdi5	**001**	V_7	V_0	V_7	V_5	V_5	V_5
	Sdi6	**101**	V_6	V_0	V_7	V_0	V_6	V_6
	Sdi7	**111**	V_7	V_7	V_7	V_7	V_7	V_7

(a)



$$z = \begin{cases} 7; \, k = 1,3,5 \\ 0; \, k = 2,4,6 \end{cases}$$

$$k = 1,2,3,4,5,6$$

(b)

Figure 8. Interface for selecting the voltage vector implemented with: a) state transition table b) FSM

The implementation state-transition table has a simple solution for FPGA. Only one constant and two short processes are necessary for implementation (Figure 9). The first process synchronizes those input signals that define the voltage-sector. The second process makes the action of choosing the voltage-vector from the table.

```
   constant Vektor: Tabela_1:=
(-- 000,   001,  010,  011,  100,  101,  110,  111  deltaI/deltaU
   ("000","000","000","000","000","000","000","000"), -- 000
   ("000","001","000","011","000","101","000","000"), -- 001
   ("000","000","010","011","000","000","110","000"), -- 010
   ("000","001","010","011","000","000","000","000"), -- 011
   ("000","000","000","000","100","101","110","000"), -- 100
   ("000","001","000","000","100","101","000","000"), -- 101
   ("000","000","010","000","100","000","110","000"), -- 110
   ("000","000","000","000","000","000","000","000") --  111
   );

begin
   p1: process(clk)
   begin
      if clk'event and clk = '1' then
         sig_U <= sU1 & sU2 & sU3;
      end if;
   end process;

   p2:process (clk)
   begin
      if clk'event and clk = '1' then
         vektor_out <= Vektor(CONV_INTEGER(sig_U))(CONV_INTEGER(e1 & e2 & e3));
      end if;
   end process;
```

Figure 9. State transition table implementation on FPGA

I. IMPLEMENTATION

The Nexys2 circuit-board is a complete, ready-to-use low-cost development platform based on a Xilinx Spartan 3E FPGA with 1.2M logical gates. Fig. 10 presents the general structures of the different elementary modules. Implementation of the motor-controller is divided into four parts.

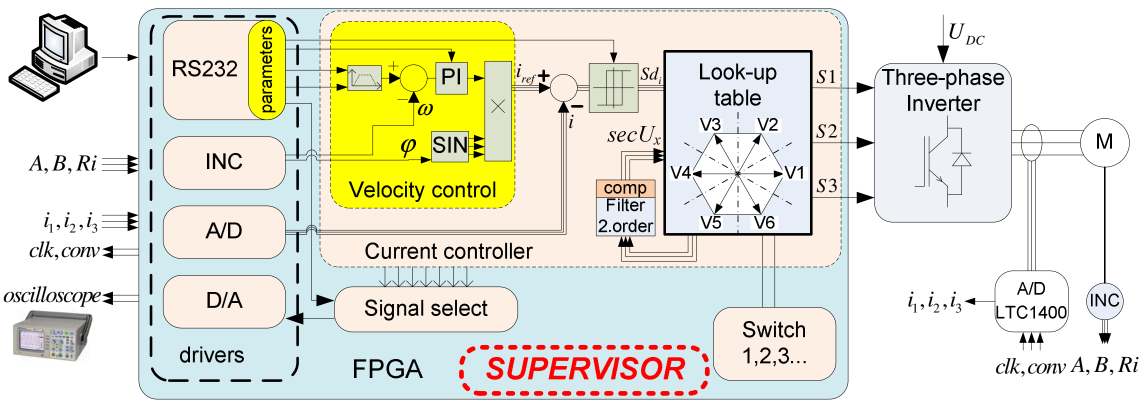


Figure 10. FPGA controller of BLAC motor

The initial part is the driver's part with ADC and DAC management modules, an incremental module for speed and position measurement, and RS 232 module for the connection of the host PC equipment to the GUI. The current controller is the second part is includes the current-controller with an additional state-transition table, and a second order PLL filter to reduce switching frequency. This part can be used for any BLAC motor. The third part with velocity-control has been made especially for the permanent-magnet synchronous motor (PMSM) [13]. The complete system control is the fourth part – the supervisor.

The VHDL code was created in Xilinx ISE software. The software scheme is divided into individual blocks for a better transparent. It is composed of a data path and a control unit coded in VHDL. The data path is composed of adders, multipliers, multiplexers, and registers. This data transfer between these operators is managed by a control unit, which is synchronized using a clock signal (CLK).

Such a scheme for the operation occupied 35% of the "number of 4 input LUTs" FPGA circuit, and 67% of the 18-bits multipliers.

II. Results

The experimental results show the velocity response of BLAC motor (Figure 11). Orderly uses of the voltage-vectors, inside the voltage sector, indirectly affect to reduce amount of switching on the inverter. Total number of switching is divided into those who used to switch one, two or all of the voltage converter legs. The results show's predominant use of switching that use only one inverter's voltage leg. Calculation of switching-number was implemented inside time interval 10 ms.

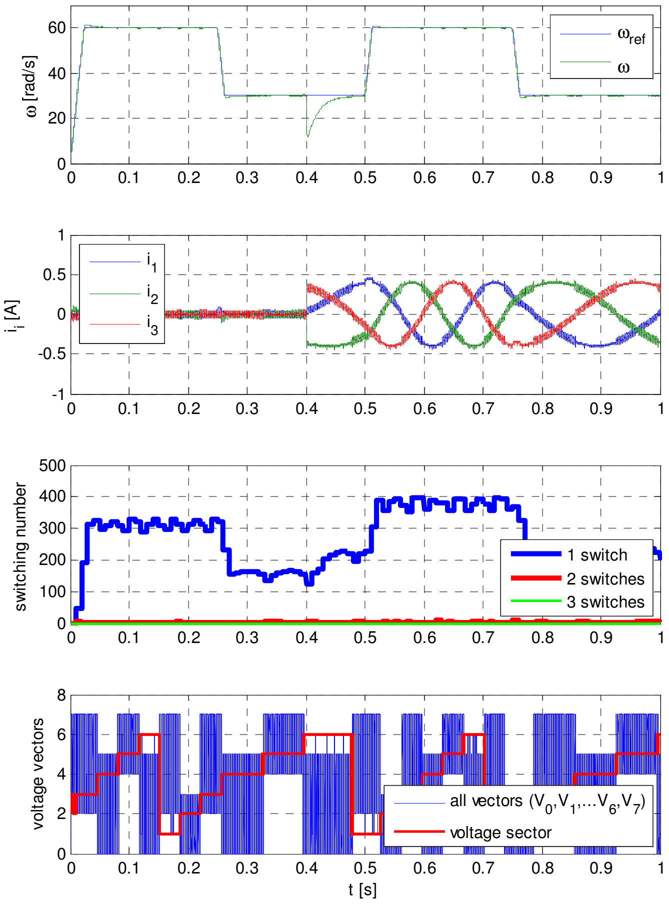


Figure 11. Experimental results: (a) Step response and (b) Total number of switches

III. Conclusion

In this study, the control of a mechatronic system has been addressed. A supervisor was implemented with a FSM that defines all possible directions and defines when some transitions can be implemented.

Special attention was paid to a transparent graphic user-interface that indirectly shows the states of the FSM supervisor. The states and indication lights are in three colours that shows a condition of the system to user. Green, yellow and red colours are used to make the GUI transparent, just like semaphore. The user immediately knows the condition of the system by recognizing the colours.

The control and supervisor of the mechatronic system are implemented on a FPGA platform. The solution is based on a parallel execution on FPGA, and by using logic-gates. The logic-gates allow reaction on event without any time delay. Therefore, most of control and supervisor functions are implemented using logic operations that allow for discrete event control. The protection functions and controller are implemented in parallel, so the control algorithm does not affect the time-delay of the protection functions. Three layers of system configuration are combined with continuous-time and discrete-event signals. Control of the BLAC motor (problems of speed and current control) are represented by a DES controller that using state transition table and a second order filter in order to reduce switching frequency. The experimental model was implemented with a low-cost FPGA platform, programmed in VHDL on freeware Xilinx ISE software.

References

[1] T. Carmely, Using Finite State Machines to Design Software, EE Times, 2009

[2] I. Boldea, N.L. Tutelea, Electric machines: steady state, transients, and design with Matlab. CRC Press, Taylor and Francis Group, Lancaster, Don: Das Aktiv-Filter-Kochbuch. Vaterstetten: IWT, 2009

[3] K. L. Short, VHDL for Engineers, Pearson education, chap. 10, 2009

[4] A. Polič, K. Jezernik, "Event-driven current control structure for a three phase inverter", Int. Rev. Electrical Eng., vol. 2, no. 1, pp. 28-35, 2007

[5] I. Boldea, S.A. Nasar, Electric Drives, CRC Press LLC, 2nd edn., 2000

[6] B. Bomar, "Implementation of a micro programmed control in FPGA", IEEE Trans. Ind. Electronics, vol. 49, no. 2, pp. 415-422, 2002

[7] C.G. Cassandras, S. Lafortune, Introduction to Discrete Event Systems, Springer Science + Business Media, 2nd edn., 2008

[8] E. Monmasson, M.N. Cirstea, "FPGA Design Methodology for Industrial Control Systems – A Review", IEEE Trans. Ind. Electronics, vol. 54, no. 4, pp. 1824-1842, 2007

[9] G.A. Wainer, Discrete-Event Modeling and Simulation: A Practitioner's Approach, CRC Press, Taylor and Francis Group, 1st edn., 2009

[10] A. Polič, K. Jezernik, "Event-driven current control structure for a three phase inverter", Int. Rev. Electrical Eng., vol. 2, no. 1, pp. 28-35, 2009

[11] R. Dubey, Introduction to Embedded System Design Using Field Programmable Gate Arrays,(Springer – Verlag London, 1st edn., 2009

[12] R. Krishnan, Permanent Magnet Synchronous and Brushless DC Motor Drives, CRC Press LLC, 1st edn., 2010