7 Implementation of Pulse Width Modulation Algorithms

7.1 ANALOG PULSE WIDTH MODULATION CONTROLLERS

The pulse width modulation (PWM) controller represents the final module within the feedback control path (Figure 7.1). It is therefore suitable to implement this module in the same technology as the control module. If the feedback control is achieved with analog circuits for high-speed, closed-loop control, the PWM controller should also be analog.

The basic role of the PWM module is to convert a reference signal to a train of pulses with a duty cycle variable upon the reference. In a three-phase system, the reference is represented by a set of three-phase variables, normally symmetrical, and the PWM pattern is delivered for the six switches of the three-phase inverter.

The simplest implementation of the PWM controller separates PWM generation for each inverter leg. The conversion from reference to the upper switch control signals is achieved by comparing the reference with a triangular signal that has constant magnitude and frequency. This has been shown in Chapter 3. Figure 7.2 illustrates the simplest possible PWM generator with a simple operational amplifier. This operational amplifier provides rail-to-rail output.

The negative input of the operational amplifier sees a triangular waveform generated by charging and discharging the capacitor C from the output voltage. The positive input of the operational amplifier sees a voltage derived from the positive feedback through R4 and R3 and the input voltage VREF. Basically this side operates as a *Schmitt* trigger comparator, and the input voltage VREF controls the output pulse duty cycle. The pulse width is accordingly modified around half of the switching period depending on the polarity and value of the input voltage. Finally, the low-side switch control is achieved by reversing the control signal for the high side.

This solution does not provide a synchronization of the generated PWM with an external signal, and the period of the generated signal has its own variations due to supply voltage and temperature. An improved solution uses the integrated circuit (IC) timer 555 with input for synchronization and analog reference. The command pulses can be achieved with the same circuit timer 555, or both the command pulses and PWM can be generated within the same device, the dual timer 556. The 555 used to generate the command pulses can be applied as input to all three phases. The PWM is generated on each phase with an additional 555 circuits (Figure 7.3).

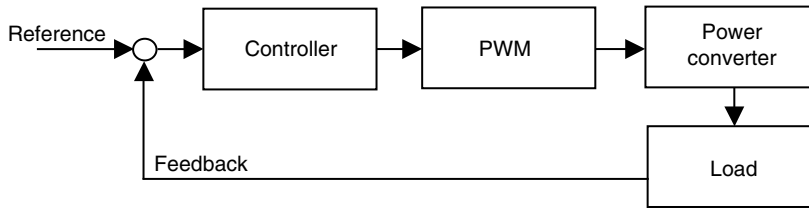

FIGURE 7.1 Schematics of the feedback control loop including a power converter.

Alternative solutions use the same triangular signal for all three channels and they are based on accurate voltage oscillators (Figure 7.4).

There are many similar solutions for analog implementation of the PWM controller; all of them work on the principles explained here. The most advanced solutions also include *Shutdown* pins for each channel in order to discontinue the PWM generation when a fault occurs. Moreover, modern requirements may differ with respect to the power delivered within the PWM signals.

Designed primarily for power supply control, TL5001 from Texas Instruments follows this type of structure, with a deadtime generator and an open collector output transistor able to control the final power-driver stage. Additional features such as under-voltage lockout and short-circuit protection are included.

Given the limited flexibility in changing PWM parameters, these analog-based solutions are hardly used. They still remain a valuable choice in high-frequency servo-drives where the whole controller needs to be implemented in analog due to the extremely high bandwidth required for the control loops. Modern alternatives include high-speed field programmed gate arrays (FPGA) or application-specific integrated circuit (ASIC) devices with predefined control loops and a PWM generator.

7.2 MIXED-MODE MOTOR CONTROLLER ICs

Motor drives in the horsepower range can be controlled with ICs without too many external components. This became a standard requirement for appliances or servo-drives when costs had to be reduced for commercial purposes.

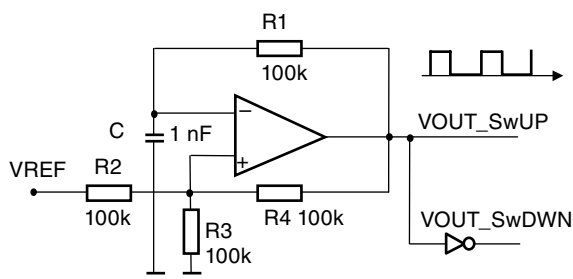

FIGURE 7.2 Simple PWM generator based on operational amplifier.

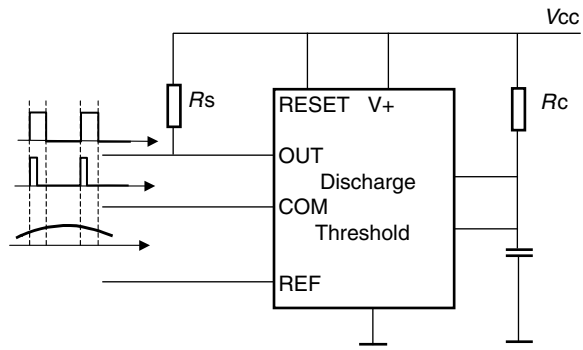

FIGURE 7.3 PWM generator with 555 oscillator.

There are many excellent mixed-mode IC-technology solutions in the market. They are designed to control low-voltage motor drives with applications in the automotive and consumer sectors. These controllers incorporate analog controllers, PWM generators with protection and deadtime circuitry, and gate drivers. The most advanced solutions also include power supply for the high-side MOSFET transistor instead of a conventional bootstrap external supply. In several automotive applications, the gate driver needs a separate power supply as the DC bus voltage may decrease below the limit required for proper gate control. For instance, A3948 [1] from Allegro Microsystems includes a boost inductor with three pairs of drivers to maintain gate control voltage.

Control systems are grouped around two application classes: the general brushless DC (BLDC) motor able to provide continuous rotation and the stepper motor

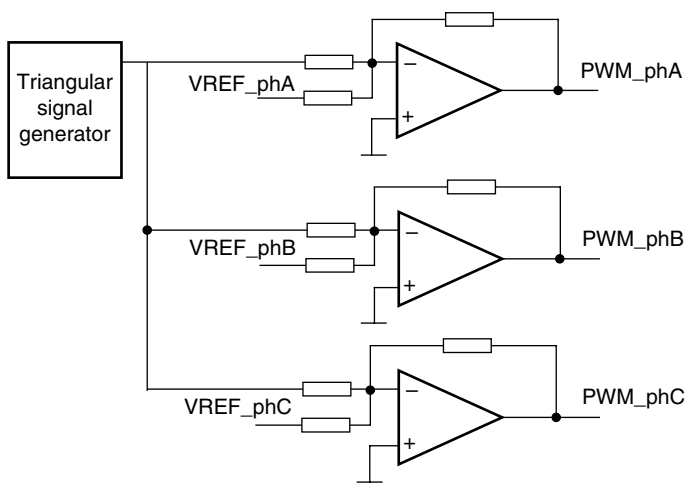

FIGURE 7.4 Schematics of a three-phase PWM generator with unique triangular generator.

able to provide start/stop operation or great positioning. Both provide integrated control solutions in mixed-mode IC technology.

The first generation of step-by-step motor incorporated the PWM current controller and the power stage within the same design, as in the Allegro A3966, in which the required phase reference currents are set by an external reference. Further developments in technology pushed for a complete solution, including a digital-to-analog controller to set current levels (A3967–A3977). The digital interface in these models allows communication with higher hierarchical levels. Further R&D is dedicated to specific applications and it includes transmission of fault conditions over the serial or parallel digital communication interface.

All these solutions are at the edge of IC technology and the future may see new products dedicated to higher power levels or with more digital and analog functions for better protection and control. For the moment, these solutions are limited to the low-voltage converter bus and small power motors.

7.3 DIGITAL STRUCTURES WITH COUNTERS: FPGA IMPLEMENTATION

7.3.1 Principle of Digital PWM Controllers

The same implementation of a timer followed up by a comparison with the reference signal can be implemented in digital with counters and compare units. Modern microcontrollers have incorporated compare units along their timers/counters that makes straightforward the PWM implementation. A single-channel PWM can be implemented with a counter, as in Figure 7.5.

PWM signals for the six switches within a three-phase inverter can be generated in several ways with counters. Generally, one signal is generated for each phase and a complementary pair of signals for the low side or the high side is achieved by a logical inversion. The designer must pay attention to the polarity within the gate driver in order to send the proper control signal to the controlled insulated gate bipolar transistor (IGBT)/MOSFET. As a rule of thumb, the gate drivers used or built in the U.S. generally do not invert the control signal polarity, whereas those made in Europe or Japan do change the control signal polarity. This is why

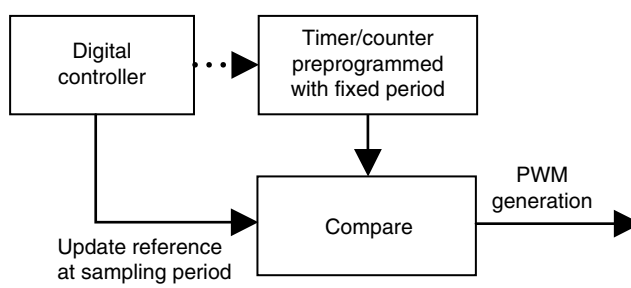

FIGURE 7.5 Principle of counter/timer use in single-channel PWM generation.

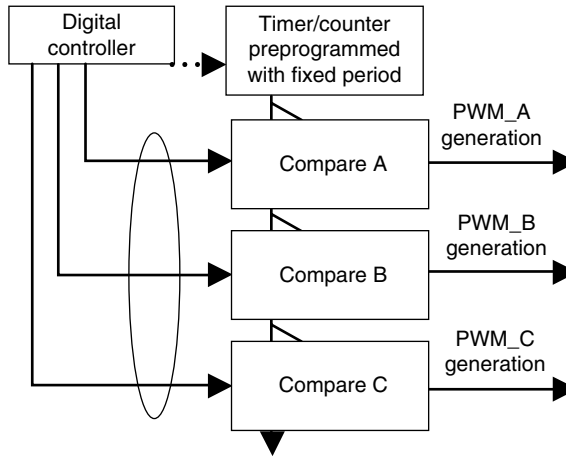

FIGURE 7.6 Three-phase PWM generation with a single counter.

all PWM from microcontrollers initially designed for the Japanese or European market have negative outputs, expecting the gate driver to reverse the signal polarity.

Finally, the design engineer needs to verify if the digital system can build the deadtime by itself or whether an external deadtime generator is necessary. Sometimes, the gate driver itself is able to generate fixed values of deadtime.

PWM can be generated with counters for a three-phase inverter in one of the following topologies:

- A single common counter and three compare units, one for each phase, using the same counting device (Figure 7.6)
- A counter and a compare unit for each individual phase (Figure 7.7)

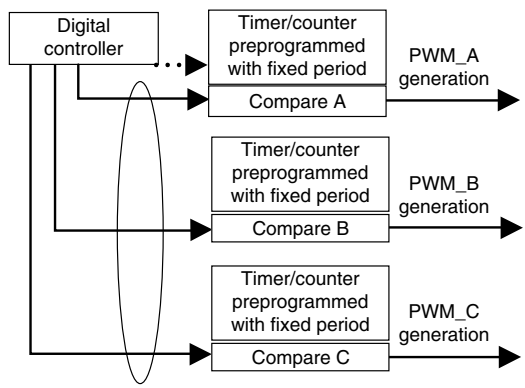

FIGURE 7.7 Three-phase PWM generation with three counters and three compares.

Depending on the internal structure of each timer, generation of the pulse width can be different. Figure 7.8a shows a left-aligned variable pulse width control, Figure 7.8b a right-aligned PWM, and Figure 7.8c a center-aligned PWM. The last one is a logical result of using the first two solutions back-to-back. Commercially available PWM generator circuits usually implement the center-aligned approach.

Some of the digital PWM ICs have been available in the market since the late 1980s. Due to simplicity in designing with FPGA and ASIC, these PWM circuits no longer represent an appealing solution, but their principle of implementation can be used as a starting point for modern FPGA or ASIC solutions.

7.3.2 Bus Compatible Digital PWM Interfaces

Figure 7.9 shows the block diagram of the Siemens SLE4520 circuit. A similar solution has been implemented by Dynex.

The SLE4520 circuit is designed as an external interface for any 8-bit microcontroller and its operation can be programmed from microcontroller or microprocessor through a data bus and a control signal bus. The time constant or pulse width associated to each phase can be stored within an 8-bit registry and loaded into counters at each SYNC signal. The programmable counters count down to zero when the state of the switching signals for the inverter control change. The deadtime constant can be programmed within a 4-bit registry. When a fault occurs in the power stage, the emergency shutdown pin is activated and it cancels out the PWM signals through the RS flip-flop.

Similar solutions for digital devices able to generate PWM with or without deadtime are developed to directly interface on the data and address bus of modern microprocessors or digital signal processors (DSPs). The same idea is actually used in custom-made FPGA or ASIC devices, but it is worthwhile quoting more examples of digital devices available for PWM generation from a processor bus.

The IXDP610 from *IXYS* can provide a single-channel pair PWM control for a switching power converter bridge. It has a complete digitally programmable interface from an 8-bit microprocessor. This has a digital comparator, comparing a counting timer with a preprogrammed constant followed by a deadtime generator. The IXDP610 is able to control power PWM devices that have switching frequencies between zero and 390 kHz, 7-bit or 8-bit resolution, and up to 11% deadtime. The output is able to drive directly 20 mA and it is suitable for opto-couplers, gate driver circuits, or low-power power modules. It also features a pulse-by-pulse shutdown protection against over-current, over-voltage or over-temperature, controlled by a logic signal from protection sensors.

7.3.3 FPGA Implementation of Space Vector Modulation Controllers

Space vector modulation (SVM) represents a special case in which a digital method is required to calculate time intervals to be loaded in counters/timers, followed up by another method able to define the switching sequence. There are numerous solutions possible for digital hardware implementation. One of them is next presented

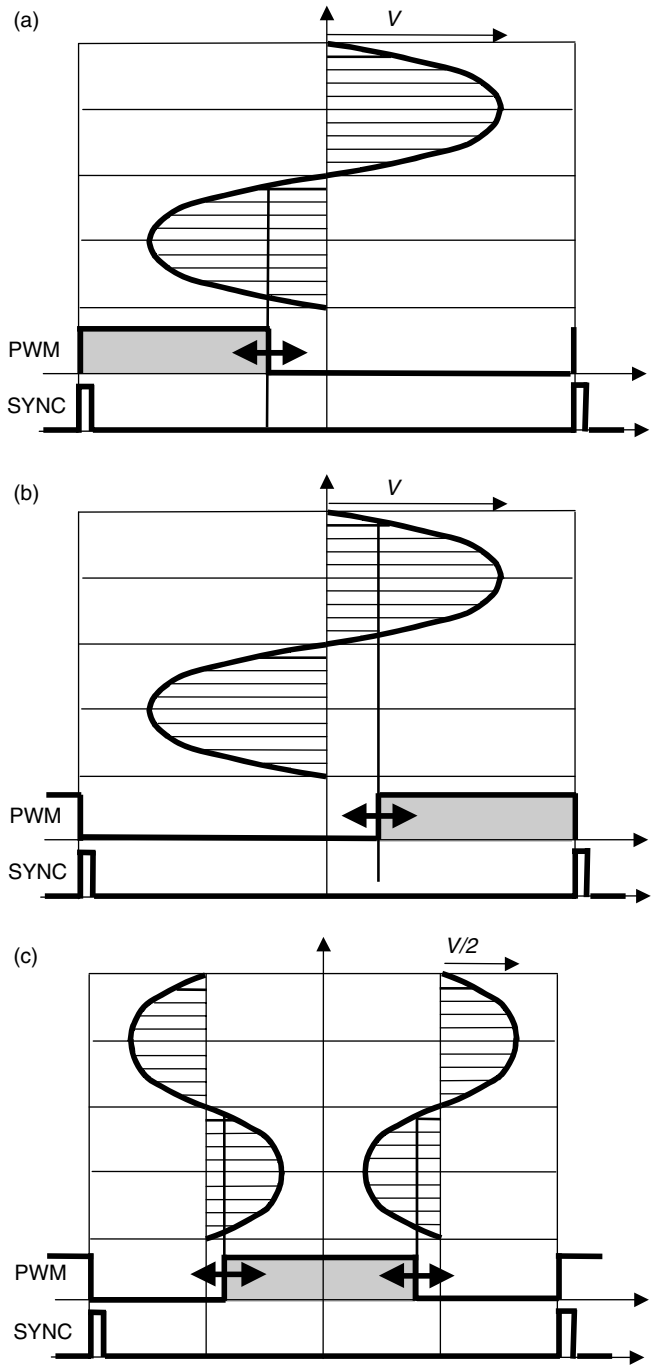

FIGURE 7.8 (a) Left-aligned, (b) right-aligned, and (c) center-aligned PWM.

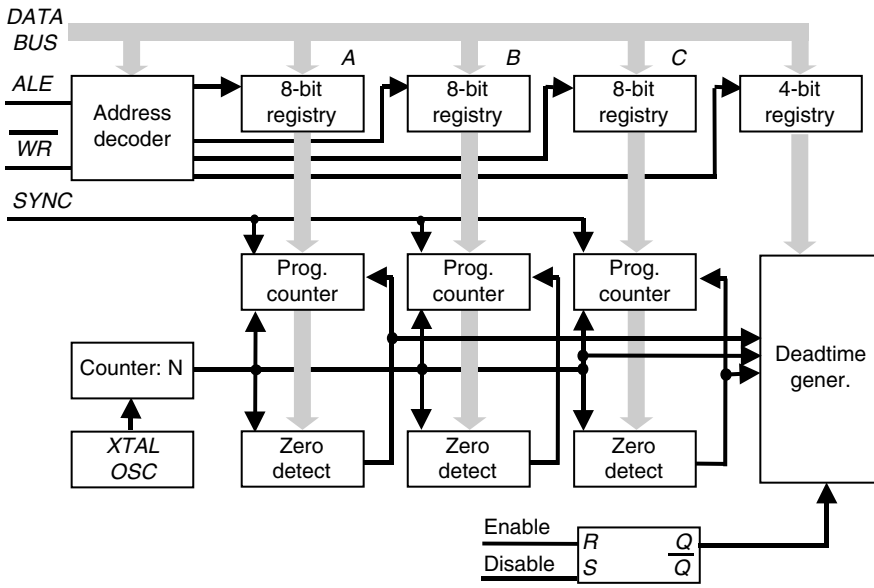

FIGURE 7.9 Block diagram of SLE4520.

[2] and it considers an SVM generation with only three states during each sampling period (*active1 — active2 — zero state*). The following interval will have a reversed sequence (*active2 — active1 — zero state*), in which the zero state is always selected with only one switching difference from the last active state.

The memory look-up table can be reduced if we take into consideration the symmetries of the three-phase system as well as the symmetries around the bisectrix of each generalized 60° sector. The most significant bits of the angular coordinate reflect the sector number and they are used to select the proper switching sequence in the second memory look-up table. The fourth most significant bit (MSB) is used to define the position within each sector and also to establish the sequence of the active states. The last four bits of the angular coordinate are used to read the t_a, t_b time constants needed for SVM generation. This memory look-up table stores these constants for an interval of 30° only (2^4 increments within 30°). Each time constant is defined on eight bits and this resolution is considered as enough for this PWM application.

An external periodic signal is used as the system clock and a frequency divider counts the sampling interval period. The definition of the pulse width is limited by having only $2^8 = 256$ points for a sampling interval, as shown in the definition of the memory look-up table. The overflow of the sampling period counter changes the state of a flip-flop D. The outputs of this flip-flop control the sequence of the time intervals t_a and t_b through two OR gates. Finally, a "switch" module is used to designate the meaning of two signals A1 and A2 corresponding to the different meanings of t_a and t_b in the first half of the 60° sector and in the second half of the 60° sector. For instance, generation of a position at 23° and a modulation

index of 0.6 leads to $t_a^1 = t_a(\text{V1}) = 0.36$ Ts and $t_b^1 = t_b(\text{V2}) = 0.23$ Ts. By symmetry, generation of a vector at 37° with the same modulation index leads to $t_b^1 = t_a(\text{V1}) = 0.23$ Ts and $t_a^1 = t_b(\text{V2}) = 0.36\ T_s$. The control switching sequence is therefore decoded with A1 and A2 and the fourth MSB with another memory look-up table.

Using two memory look-up tables is not convenient, especially because they are of different sizes. The switching sequence can be defined with a combinatorial circuit and a series of multiplex circuits (Figure 7.10). Observing all the possible switching sequences, the synthesis can be developed as shown in Table 7.1.

Figure 7.11 shows the logic decoder used instead of the memory look-up table and is built up of a divider of six Johnson counters [*D0-D1-D2*]. The outputs of these counters can be used as addresses for the multiplex circuits having *P*, *A*1, and *A*2 as inputs. Table 7.2 illustrates this decoding of the end of counting signals.

Other digital or FPGA syntheses of the SVM algorithm can be found in [3–7]. Many of them implement the SVM algorithm in a straightforward manner with calculation of Equation (5.30) followed by switching sequence decoding. The latter module can be changed from one SVM algorithm to another, for instance, in order to reduce losses (see Chapter 5).

The final stage before firing the gates of the power semiconductor devices is represented by the deadtime generator. The role and calculation of the deadtime interval have been explained in Chapter 3. The deadtime generator is usually implemented together with the PWM algorithm, but special circuits for deadtime generation also exist.

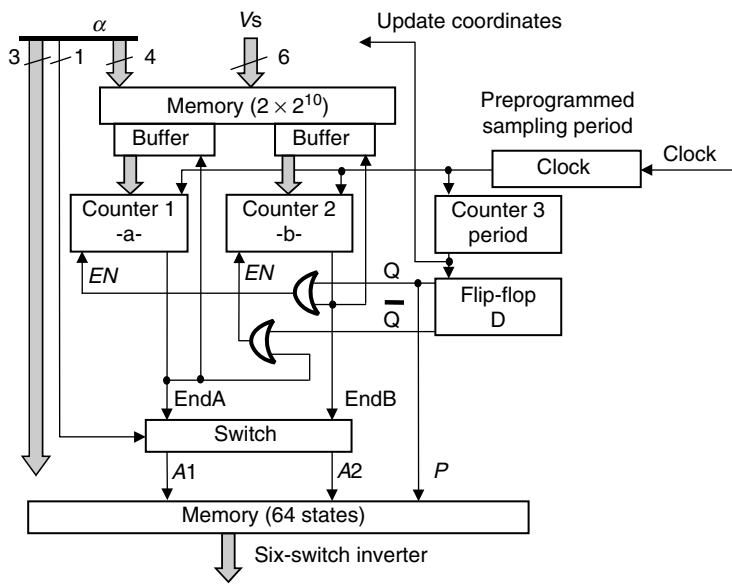

FIGURE 7.10 Principle of pure digital implementation of an SVM algorithm.

TABLE 7.1
Defining the Switching Sequence from the End of Counting Signals

Initial Zero Vector 000				Initial Zero Vector 111			
Sector	Signal B	Signal A1	Signal A2	Sector	Signal B	Signal A1	Signal A2
1	S1	S3	S5	1	S6	S4	S2
2	S3	S1	S5	2	S6	S2	S4
3	S3	S5	S1	3	S2	S6	S4
4	S5	S3	S1	4	S2	S4	S6
5	S5	S1	S3	5	S4	S2	S6
6	S1	S5	S3	6	S4	S6	S2

7.3.4 Deadtime Digital Controllers

A solution for deadtime implementation with counters is shown in Figure 7.12 and it corresponds to *IXYS* circuits IXDP630/631PI. The deadtime is always eight clock periods and the clock can be an external crystal or an RC oscillator circuit.

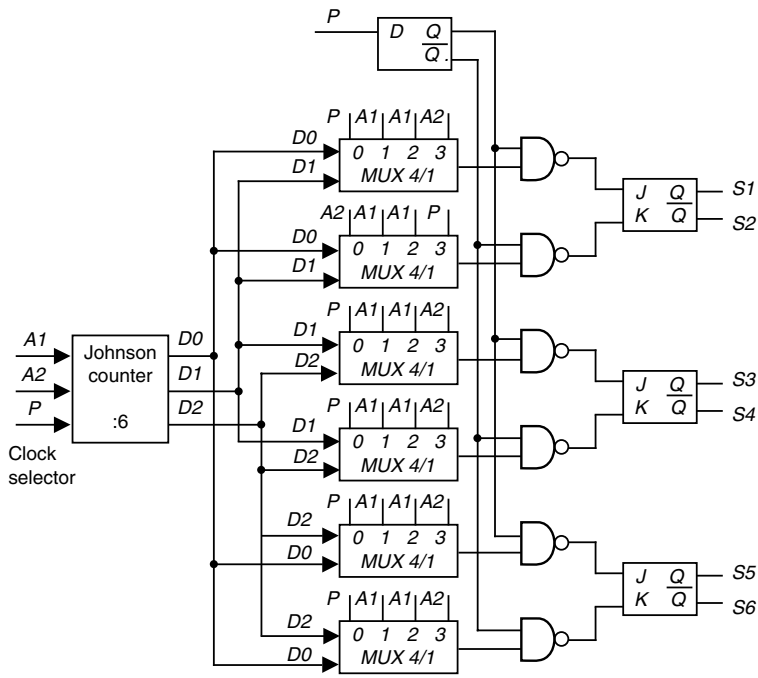

FIGURE 7.11 Example of SVM digital synthesis with multiplex circuits.

TABLE 7.2
Proposed Implementation

s	Address D1-D0	S1 Turn-On	S2 Turn-On	Address D0-D2	S3 Turn-On	S4 Turn-On	Address D2-D1	S5 Turn-On	S6 Turn-On
1	00	*P*	*A2*	00	*A1*	*A1*	00	*A2*	*P*
2	01	*A1*	*A1*	00	*P*	*A2*	10	*A2*	*P*
3	11	*A2*	*B*	01	*P*	*A2*	10	*A1*	*A1*
4	11	*A2*	*B*	11	*A1*	*A1*	11	*P*	*A2*
5	10	*A1*	*A1*	11	*A2*	*P*	01	*P*	*A2*
6	11	*B*	*A2*	10	*A2*	*P*	01	*A1*	*A1*

Controlling the clock period, one can adjust the deadtime interval. The same IC is able to generate deadtime on all three phases separately. An additional "output enable" signal is able to shutdown all or each output. The operation of this circuit can be understood from Figure 7.12. Each positive edge of the control signal SIN delays the positive edge of the gate signal *S_High* and each negative edge of the control signal SIN delays the positive edge of the gate signal *S_Low*. Negative edges of *S_High* and *S_Low* directly follow the appropriate edges of the control signals.

7.4 MARKETS FOR GENERAL-PURPOSE AND DEDICATED DIGITAL PROCESSORS

7.4.1 History of Using Microprocessors/Microcontrollers in Power Converter Control

Chapter 1 presented the main control functions required for implementation on the digital control system platform. There are two directions of development in the digital world able to implement all these functions.

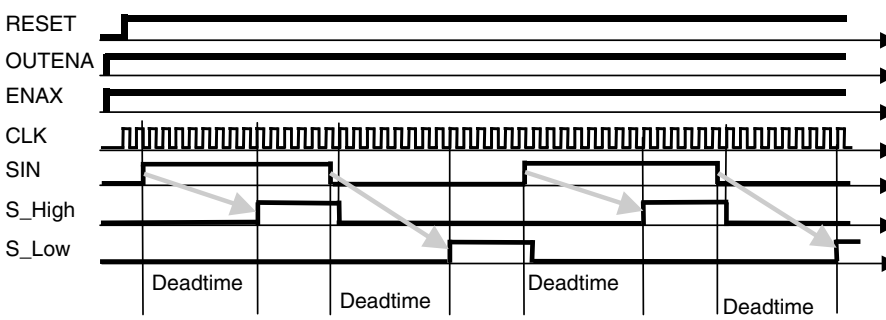

FIGURE 7.12 Time diagram of the deadtime generator circuit IXDP630/631PI.

The first direction focuses on architectures of general-use microprocessors. These devices achieve incredibly fast running of the control code, and include many functions implemented in the software. The evolution of microprocessors in the last 25 years started with families of *bit-slice* devices, such as INTEL3000 or AMD2900; 8-bit microprocessors such as INTEL I8080, ZILOG Z80, and I8085; 16-bit microprocessors such as I8086, Z8000, Motorola 68000, and Texas Instruments 16008/16032; and 32-bit microprocessors scuh as 68020 or I80385. They were upgraded in the late 1980s with arithmetic co-processors used in tandem, such as INTEL 80826/80287, National Semiconductor NS 32016/32081. This evolutionary step also included other LSI circuits:

- High-speed hardware multipliers able to calculate 24 × 24 bits in fixed point in 200 nsec, and 16 × 16 in 34 nsec (for instance, MPY016K-TRW)
- Arithmetic modules for calculation in variable points up to 80 bits (e.g., AMD 511, 9512), which work as slave co-processors to the main microprocessor device
- Multi-channel acquisition systems (e.g., AD162, AD364, DAS1150) or dedicated interfaces to existing microcomputers in digital systems, such as INTEL, PROLOG, MOSTEK, TI, or DEC, and so on
- Fast RAM memories, with double port access and huge capacity
- Special digital circuits used to fast interface to microprocessor buses
- Circuits dedicated to generation of the multi-channel PWM for control of power converters
- Complex digital circuits containing ROM-I/O-Timer peripherals dedicated to extension of existing microprocessors (e.g., MC6846 used in conjunction with Motorola 6800 device)

In order to fully understand this evolutionary step, let us take an example. *Zilog* Z80 was a quite widely used 8-bit microprocessor family in the 1980s. The clock speed was limited to 4 MHz and the peripheral functions were achieved on separate chips from the main device. A special device called PIO-Z80 was dedicated to the parallel I/O, another one to the serial communication SIO-Z80; a special memory access unit DMAC-Z80 and a timer/counter circuit CTC-Z80 completed this family. A fully operational microsystem was required to contain all these ICs on a common printed-circuit board with all the data, control, and address buses accessible from outside. Once this board was realized and tested, coding was done directly on assembly language without any real-time debugger or the possibility to visualize the operation through a monitor program.

All these digital solutions of the late 1980s or early 1990s had, however, a quite limited processing speed. Some of them are still in use due to their generality and simplicity. An alternative able to increase the general operation speed considers off-line processing and storage of the control results in a large memory look-up table. The best example is offered by implementation of different PWM algorithms through memory look-up tables. Initially, there was a substantial limitation due to the inherent limited speed of access to these memory look-up tables. Modern

ROM devices can achieve access times sufficient for the switching of modern power converters. We will come back to these solutions few years later with the advent of FPGA devices.

The second major direction of development was in modern microcontrollers designed for industrial applications and including dedicated peripherals within the same silicon die. These microcontrollers incorporated on the same chip more memory and I/O interfaces along with their MUX, A/D converters, timers, PWM channels, and so on. Several examples that met with success in the beginning of the 1990s are still in use today:

- INTEL 8051 (with its versions 8031, 8751, 8052, 8032, and different manufacturers like *Siemens* or *Philips*) contains:
  - An 8-bit microcomputer with its own clock generator
  - A dedicated module for Boolean processing
  - 4 kB of ROM (8031 without memory, 8751 equipped with EPROM, and 8052 with 8 kB and a BASIC interpreter)
  - 128 bytes RAM (with 256 at 8052)
  - Two 16-bit timer/counter circuits (8052 equipped with three such counters) used especially for PWM generation
  - Four programmable I/O, bidirectional, with option for each bit to read, write and program
  - Serial I/O channels
  - Two external interrupt inputs, that can be enabled or disabled by software (8052 equipped with six interrupts including two external inputs)
  - Dedicated registers for arithmetic operations, stack management, I/O latches
  - A strong assembly language with 111 instructions, enhanced with interpreter for BASIC or PL/M
- INTEL 80186 microcontroller created after the 80196 microprocessor with an architecture similar to the 8086-2 but also including:
  - A clock generator at 8 MHz
  - Two high-speed Direct Memory Access (DMA) channels
  - Programmable interrupt controller with five levels of priorities
  - Wait state generator
  - Dedicated bus controller
  - Possible co-processor interface
  - Software compatible with 8086 and option for programming in ASM86, PL/M86, PASCAL 86, FORTRAN86, LINK86, C, so on
- INTEL 8096 16-bits microcontroller with enhanced interfaces to analog and digital systems. Different versions include:
  - 8 kB or no memory
  - Four or eight 10-bits A/D channels with a conversion time of 42 μsec
  - Five I/O ports
  - Serial output port
  - One external interrupt

  - PWM output
  - Four channels for fast capture and event timing with the existing counter
  - Six fast outputs programmable to generate pulses at fixed time intervals with a resolution of 2 μsec
  - Two timers of 16 bits each, one for real-time synchronization and the second for countdown of external events
  - Watchdog circuit
  - Assembly language with interpreter for PL/M
- Additional microcontroller examples from the same historical class are Motorola MK 68200, MK3870, NEC μPD 78312, INTEL 8748, Philips PCB8049, MAB 4048, and so on.

7.4.2 DSPs Used in Power Converter Control

All these efforts to add more functions to microcontroller chips have not been enough for modern applications, and semiconductor engineers have moved forward to DSP devices. DSPs have emerged in the communications business for fast processing of information from data and signals. They are characterized by very quick clock rates and additional processing circuitry within the same chip. It is normal to expect a 16 × 16 multiplication calculated within 100 nsec on any of these devices.

Lately, the use of DSPs in power conversion control has been widened by dedicated motor control DSPs equipped with special peripherals designed for multiphase converter control. These devices are generic, are called *Motor Control DSPs*, and they feature three-phase power conversion PWM control useable in both machine drive and three-phase grid-tied applications. The main DSP producers in the world are — in an arbitrary order — Texas Instruments, Motorola, and Analog Devices. A brief description of the features of each type of DSP and its specifics in controlling three-phase power conversion follows.

The largest DSP market share belongs to Texas Instruments. Their DSPs have passed through a series of iterations and versions in the last 10 years. The TMS320 family of DSPs is made on a Harvard architecture that allows execution of several operations simultaneously for increased running speed.

The first use of the DSP to control power converters seems to be based on TMS32010/32020, a general use DSP. Its features include:

- Working on 16 bits with an instruction cycle of 160 nsec
- A small RAM memory of 144 × 16 bits
- A parallel 16-bit module for a 160 nsec multiplication
- Eight I/O ports of 16 bits
- An external interrupt
- Special registers for data processing
- A 16-bit timer
- Interface for memory access in multi-processor mode

In the early 1990s, Texas Instruments had developed a set of peripherals for motor drive control under the name *Event Manager*. This was their first generation motor control DSPs, TMS320F(C)24x, and it was optimized in the second generation TMS320F(C)240x that was released in the late 1990s. In 2002, TI released the third generation motor control DSPs, TMS320F(C)280x, which had a very high-speed Central Processing Unit (CPU), able to run programs at a clock frequency of 150 MHz. Section 7.5 will present in detail all hardware and software possibilities for PWM implementation on a TI DSP using the Event Manager.

Analog Devices has another good DSP program with motor control features. Several architecture solutions have been tried and developed at *Analog Devices* and the most remarkable probably refers to the development of a motor controlled co-processor. The first solution, called ADMC201, was subsequently developed into ADMC330, ADMC401, and finally incorporated as a peripheral of a DSP circuit within the same package. Section 7.7 will reiterate the historical importance of this architecture development.

The DSP family ADSP-219x represents another solution from *Analog Devices* with many applications in power electronics. This 16-bit fixed-point DSP core is able to perform 160 MIPS in its modern versions. Other power converter control-related peripherals include:

- On-chip RAM 8 k Words shared between program RAM and data RAM
- External Memory Interface with dedicated Memory DMA Controller for data and instruction transfer
- Eight-channel, 14-bit analog-to-digital converter system with up to 20 MSPS sampling rate
- Three-phase 16-bit center-based PWM generation unit with 12.5 nsec resolution at 160 MHz core clock
- Dedicated 32-bit encoder interface with companion encoder event timer
- Dual 16-bit auxiliary PWM outputs
- Three programmable 32-bit timers
- Sixteen general-purpose I/O pins
- SPI and synchronous serial interface

Other DSPs of the same generation are NEC7720, Fujitsu MB8784, and STC-DSP128.

The large number of customers and the extensive use of these DSPs in the world have encouraged development of software libraries dedicated to DSP applications. Programmable in both assembly language and C language, these devices helped develop a software culture and a style for programming proper to motor-drive applications. Recent efforts will probably provide code modularity and automatic software validation. Conventional development tools have been extended *in the loop viewers* and automatic programming tools. Companies like MathWorks have produced systems able to include the DSP system in a PC-based simulation program for quick development and debugging of new codes dedicated to power converter control.

7.4.3 Parallel Processing in Multi-Processor Structures

In parallel with the tremendous development in DSP devices, engineers have tried to use digital systems with parallel processing for control of power converters. These systems allow more processing power even if they do not directly benefit special peripherals. The whole control algorithm is therefore brought down to the time scale of the PWM algorithm.

The software operation in a multi-processor system is based on a waiting list for processes, special set of instructions for creation of parallel tasks, a minimal context for fast communication between processes, several timers, and an evolved interrupt system. Selection of the communication protocol between parallel microsystems is very important and there are several options available: series transfer, parallel transfer through handshaking, DMA transfer, FIFO transfer, double-port RAM memory transfer, and multiple bus architecture.

A device in this category, INMOS Transputer 414, uses for execution reduced instruction set controller processors able to run with 32 bits at 50 nsec and to communicate internally at high speed.

The same tendency to use parallel hardware is seen in the development of software platforms. The simplest microprocessors need to model numerous concurrent events sequentially. Assembly language is the quickest way to implement these models, though programs written in assembly language often have sequences that are redundant and too detailed. To simplify this process, many engineers use high-level programming languages like C, PL/M, PASCAL, Visual BASIC, and so on. Programming is therefore reduced to establishing clear tasks and writing special modules for each task. The main program is then grouping these tasks based on priority levels.

The sequential processing of tasks does not extract all benefits from the parallel processor hardware. The first programming language that consistently expressed parallelism in execution was APL [8]. However, the task was computed through computing power distribution, since, at that time, real parallel hardware was not available. The programming language ADA represents an important evolutionary step, especially in conjunction with the specialized 32-bits microprocessor iAPX432. The first truly parallel evaluation of tasks was carried out with the programming language OCCAM, which was able to achieve synchronization and communication between tasks on different hardware. OCCAM is the programming language for *transputer*, but it can be adapted to any other computer with minimal modifications.

OCCAM shares tasks in individual processes that run separately by communicating between them. The whole program can be seen as a network of interconnected processes. The synchronization between processes requires that all of them run in the time required by the slowest process and that results are reported at each step. *Transputer* devices simulate this concurrency through software *time-slicing*, followed by communication through the four serial communication channels. This structure can be extended further, if necessary.

Despite the computing power of these parallel architectures, they are not used in large series production of power converters. Manufacturing operations require a

reduced number of terminals to be soldered and a cheap semiconductor solution. Furthermore, firmware considerations also encourage the use of a simpler software based solution.

7.5 SOFTWARE IMPLEMENTATION IN LOW-COST MICROCONTROLLERS

7.5.1 Software Manipulation of Counter Timing

Many microcontrollers do not benefit from a dedicated motor control peripheral and they can implement PWM algorithms using software. A minimal timer is still required for real-time accounting. The software has to calculate the time intervals necessary for the next sampling interval and to sequentially program the timer for each state. Obviously, the resulting PWM cannot have a very large PWM frequency, given the real-time requirements, but it may satisfy many low-cost applications.

Consider an implementation of the conventional SVM algorithm using a single timer circuit and a parallel interface for inverter control. This digital structure imposes some requirements in designing the software controller:

- Each time constant must be calculated by the software on the basis of general SVM relationships. This will limit the maximum frequency of the fundamental waveform able to be generated by this system.
- The timer channel should produce an interrupt at the beginning of each time interval followed by a software compensation for all delays produced by this approach.
- A parallel interface is used to generate the PWM output signals on each interrupt produced by the counter.

A time diagram for this approach is presented in Figure 7.13. Due to the pure software implementation, the switching frequency is limited in the support processor and not too many other features can be run on the same platform. The advantage,

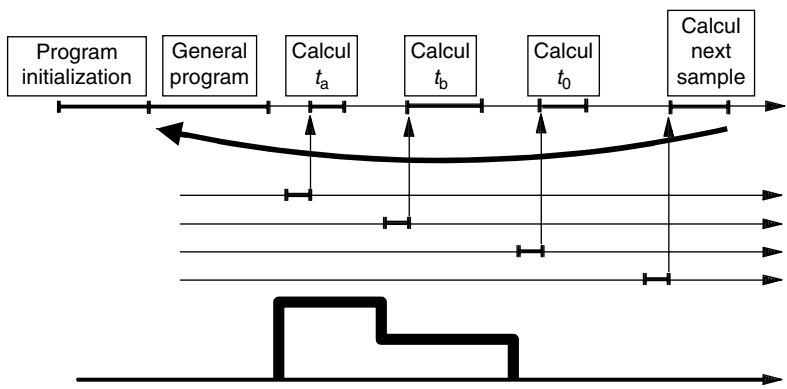

FIGURE 7.13 Time diagram of a pure software implementation of SVM.

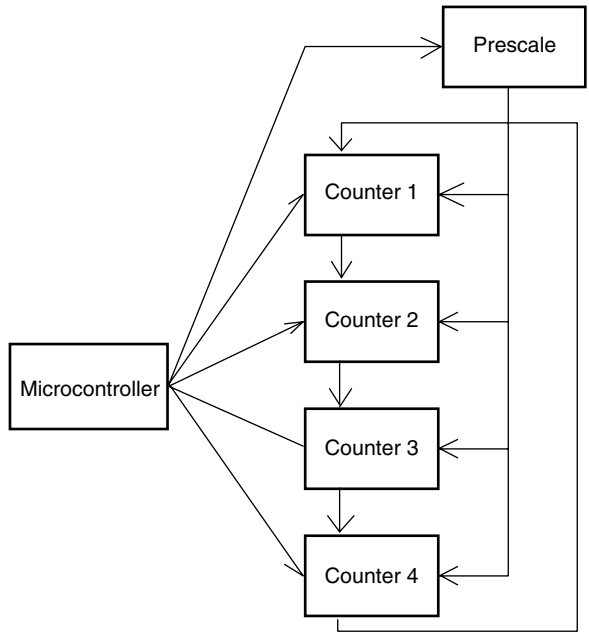

FIGURE 7.14 SVM software implementation based on four counters.

however, is an extremely reduced list of hardware requirements: a simple timer and software processing of the comparison result is enough.

This solution is not very practical, though, and another solution is presented in Figure 7.14, in which four timer channels are used for synthesis of each sampling interval. The end of each timing interval produces the start of the following counter and the timing of the next inverter state. The same software structure is required to calculate the PWM.

7.5.2 Calculation of Time Interval Constants

The second major requirement to implement a PWM algorithm is that the constants to be loaded within timers for counting the different state intervals must be determined very quickly.

Chapter 5 showed the symmetries in the generation of an SVM algorithm, symmetries after each 60° interval and around the bisectrix of each interval. This can help in reducing the size of the look-up table required to implement the SIN/COS function. Many DSP or microcontroller devices already have in their ROM memory a brief SIN look-up table, usually with 256 or 512 points over a complete cycle. If a higher resolution is required, a new look-up table can be generated by a computational software package like MATLAB or MATHCAD and incorporated with the control program. The number of points within the SIN look-up table determines the resolution in defining each pulse width. This is important for

high-switching frequency applications with advanced controlllers. Because memory is not generally a serious constraint, incorporation of a high-resolution SIN look-up table is not a problem in modern microcontrollers.

A totally different approach that is able to take advantage of the computing speed of modern controllers while saving memory requirements is to approximate the reference signals by interpolation. Different mathematical approaches for interpolation are available and they can be used to extend the range of a SIN look-up table from what is already installed in the microcontroller or DSP ROM memory to what the current requirements are to generate the PWM.

An alternative solution to achieving interpolation is fuzzy logic. The theory of fuzzy logic was first proposed in a paper of Professor Lotfi Zadeh in 1965 [9]. For many years this theory was not of interest until, in the 1970s, a series of books extensively presented the mathematical aspects of fuzzy logic [10,11]. In the early 1990s, many papers tried to implement concepts of fuzzy logic in the control of power converters, especially in replacing conventional PI controllers with fuzzy logic controllers. But it did not really succeed in power converter applications, mainly due to existence of other conventional nonlinear control methods and some difficulties in implementation.

A special feature of the fuzzy logic theory was outlined at the 1992 IEEE conference for fuzzy logic systems, where it was repositioned as a logic of interpolate thinking.

This section presents an example of the use of fuzzy logic to implement the interpolation concept, which generates an SVM from a SIN look-up table with a reduced number of points [12,13].

Consider a PWM model with 24 points over the fundamental cycle. The zero states of the model are shared equally during each sampling interval. To simplify the explanation the demonstration is limited to a generalized 60° sector. The fuzzy logic-based control relations can then be expressed as:

$$\text{If } \alpha = i^* \frac{\pi}{12}, \quad \text{then } K_a = K_i \text{ with } i = 0, \ldots, 4. \tag{7.1}$$

This says that for a reduced number of five angular coordinates over the 60° sector, we know precisely what the time constants for each active vector are. The whole set of angular coordinates is therefore described by five fuzzy subsets. Figure 7.15 presents the membership functions for each variable. This choice of the membership functions along with a linear defuzzification method presents less distortions when equivalence with an identical crisp system is sought. Besides, this is very simple to implement. The effect of different approximation approaches in the output phase voltage is shown in Figure 7.16.

The linear defuzzification approach produces a very good interpolative approximation, each harmonic of the output phase voltage having an error below 10^{-5} of the precise calculation method. This is also reflected in the total harmonic distortion (THD) calculation for the output phase voltage. Figure 7.17a shows the THD comparison in absolute values and Figure 7.17b in relative values.

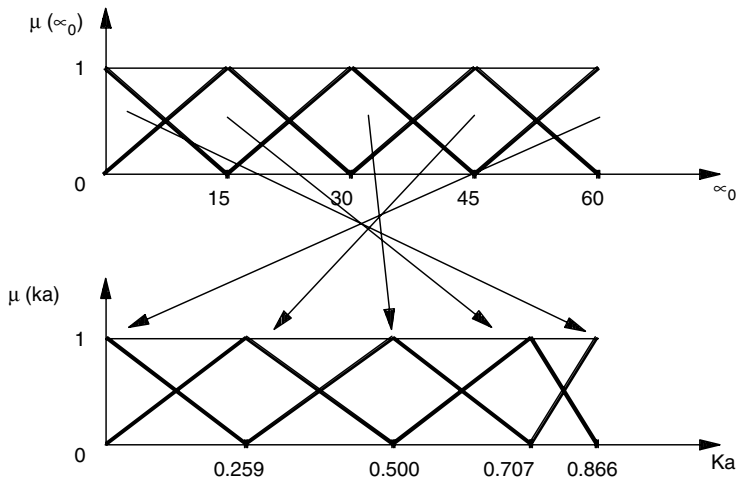

FIGURE 7.15 Fuzzy subsets for the proposed method.

A software implementation algorithm for this method is presented next along with a numerical example:

1. Calculate the polar coordinate of the next desired vector position (V_s, α)

   Example: $(V_s, \alpha) = (0.45, 212^\circ)$

2. Define α_0 as the angular coordinate within the generalized sector:

$$\alpha_0 = 212^\circ - (\text{INT}(212/60))60 = 32^\circ$$

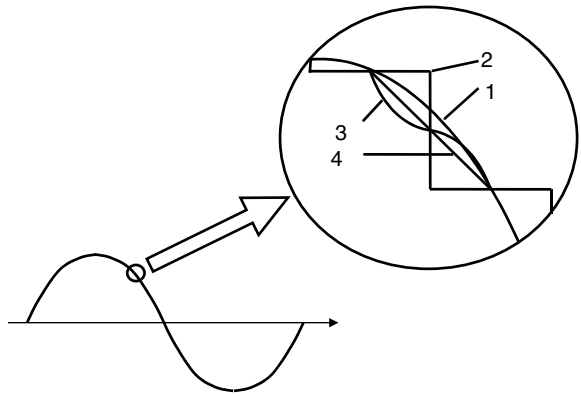

FIGURE 7.16 Effect of different approximation methods in the output phase voltages: (1) sinusoidal PWM; (2) staircase PWM; (3) center-of-gravity defuzzification; (4) linear defuzzification.

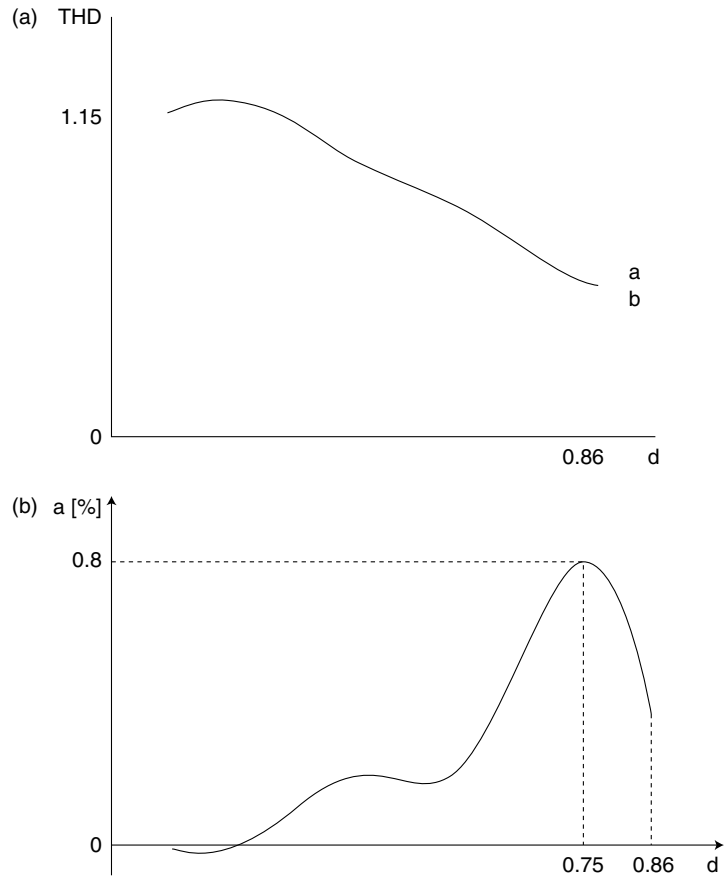

FIGURE 7.17 THD calculation for the proposed (a) interpolative approximation and (b) the precise calculation case.

3. Fuzzy estimation of the time intervals allocated to each active vector K_a, K_b based on α_0
   Define the sector number

$$I = \text{INT}(\alpha/15) + 1 = 3$$

   Calculate the membership degree of α_0 to the left subset

$$\mu_1(\alpha_0) = (15 * 1 - \alpha_0)/15 = 0.86$$

   Calculate the membership degree of α_0 to the right subset

$$\mu_2(\alpha_0) = 1 - \mu_1(\alpha_0) = 0.14$$

Calculate the time interval by linear defuzzification

$$K_a = (\mu_1(\alpha_0) * K_1 + \mu_2(\alpha_0) * K_{1+1}) = 0.86 * 0.5 + 0.14 * 0.707 = 0.528$$

Calculate the second time interval by linear defuzzification

$$K_b = (\mu_1(\alpha_0) * K_{6-1} + \mu_2(\alpha_0) * K_{5-1}) = 0.86 * 0.5 + 0.14 * 0.259 = 0.467$$

4. Determine the actual time constants by multiplication with the sampling interval period
5. Calculate $t_{01} = 0.5 * (T - t_a - t_b)$
6. Select the appropriate switching pattern ($t_{01} \rightarrow t_a \rightarrow t_b$ or $t_{01} \rightarrow t_b \rightarrow t_a$)
7. Timer programming

This fuzzy logic-based approach can be extended to V/Hz control of three-phase induction motor drives. The V/Hz characteristic is not linear for the whole range, but requires some nonlinearity in the low-frequency range in order to compensate for the voltage drop on the stator resistances. This nonlinearity can also be mapped with a fuzzy logic-based variable. The resulting controller has two inputs and can be implemented based on the rule table shown in Table 7.3. The linquistic degrees are general without any relationship to their language sense. Each rule can be interpreted as:

IF α is NS (negative small) AND f is PS (positive small), THEN t_a is NS (negative small).

The membership functions are considered triangular, symmetrical, with prototypes provided by the time constants calculated for the 24-pulse PWM case. The resulting control surface is shown in Figure 7.18, comparing the ideal and approximative cases.

TABLE 7.3
Rule Table for the Fuzzy Logic Controller

f\α	NB	NS	PS	PB
NB	PB	PB	PM	NB
NM	PB	PM	Z	NB
NS	PS	Z	NS	NB
Z	PS	Z	NM	NB
PS	Z	NS	NM	NB
PM	Z	NS	NM	NB
PB	Z	NS	NM	NB

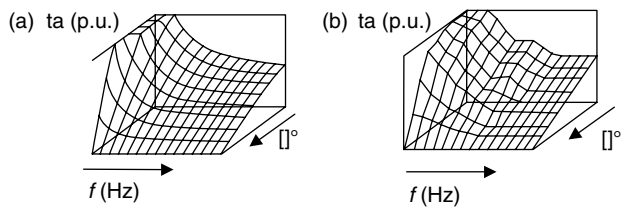

FIGURE 7.18 Controller surfaces: (a) ideal and (b) approximative.

7.6 MICROCONTROLLERS WITH POWER CONVERTER INTERFACES

Given the large market for low-power motor drives with applications in appliances and servo-drives, some manufacturers have developed microcontrollers dedicated to motor control. The pressure to make these devices low cost is extremely high and giving up features not necessary in basic inverter control applications could make them very competitive. Along with a simple, low-cost 8-bit central processing unit, these devices benefit from dedicated hardware interfaces for PWM generation and A/D conversion and data acquisition. Different communication interfaces complete the internal structure, which is optimized for motor drive applications.

A good example within this category is the NEC μPD78098x [14]. This class of microcontrollers has seven channels of programmable timer/counters for event management along with a low-cost central processing unit running at 8 MHz. Using a 10-bit timer for PWM generation, it has dedicated circuitry able to generate three pairs of output signals (inverter control) and a programmable 8-bit deadtime generator. Eight 10-bit A/D conversion channels are also available for power converter feedback control. A UART communication interface may place this device as a lower level controller in a hierarchical structure.

Because of the limited clock frequency (8 MHz), the PWM cannot be generated with a carrier frequency of 15.6 kHz (64 μsec) or higher when the timer is programmed to an 8-bit resolution or on 3.9 kHz (256 μsec) for the 10-bit resolution. The clock frequency is scaled for different switching frequencies below these values. For higher switching frequencies, the 8-bit or 10-bit resolution cannot be achieved, and the timer should work with multiple interrupts from the same timer and should divide the frequency by an appropriate value.

Motorola provides a similar device that can run at up to 32 MHz, and that has a fault-tolerant PWM controller for motor drive applications. The Motorola 68HC08 family also includes a deadtime generator and an interesting deadtime compensation capability. These are created around the so-called *Timer Interface Module*, which also provides capture features. Finally, this device includes USB, CAN, and SPI interfaces with plans for a local interconnect network. As in many modern microcontrollers or DSPs, it has an in-circuit flash option.

7.7 MOTOR CONTROL CO-PROCESSORS

Any digital feedback control solution can be implemented into a general-purpose processing unit, but the particular control and measurement of a power electronics system requires special peripherals. Current research is aimed at improving the performance of power-dedicated peripherals, and placing it along with the main computing unit.

Another direction of research is toward application-specific peripherals outside of the main processing unit in the form of a digital co-processor [15].

Analog Devices' ADMC201/·/331 is the most well-known commercial solution using this approach. Developed for motor control applications, it has modular blocks that implement the vector control method within the same programmable digital system. Developed initially as a partner for ADSP2105, it is used in conjunction with other DSPs or microcontrollers as well.

The latest descendent of this family has been incorporated along a 26MIPS fixed-point central processing unit within the ADMC401. It includes:

- Eight-channel simultaneous sampling A/D converters
- Three-phase 16-bit PWM generator
- Two 8-bit auxiliary PWM outputs
- Park direct and reverse digital transform based on angular coordinates

ADMC331 includes predefined hardware mathematical functions used in motor control, such as vector transformations based on angular coordinates. Hardware-based vector transformations provide substantial time saving in a three-phase system control. The feedback control loops can run at higher rates implementing control loops with larger bandwidth.

Very recently, International Rectifier developed a similar device (Accelerator TM) able to provide a feedback control loop with a bandwidth of 5 kHz by using FPGA support [15]. Again, all elements of a vector control algorithm were included in VHDL models and implemented within a flexible FPGA structure, providing fast parallel processing of the field orientation algorithm. The system makes possible code modularity and portability for specific applications. This digital system is designed to sit on the DC power bus and to directly control the power device's gate circuitry. Isolation of the communication interface with the higher hierarchical level is provided.

7.8 USING THE EVENT MANAGER WITHIN TEXAS INSTRUMENT'S DSPs

7.8.1 Event Manager Structure

Texas Instrument's has manufactured a set of DSP devices dedicated to motor control applications. The particular peripheral module of these DSPs is called *Event Manager* and it has all functions necessary for motor control. There are slight

differences between the *Event Manager* included in the '24x device, '24xx device, and '280x device, but all are meant to help power converter control. We refer to the *Event Manager* from the '24xx devices only noting slight differences where possible. The '24xx series is the most used at the time of writing this book.

There are two *Event Managers* within the '24xx device, each including:

- Two *general-purpose timers* that are used to implement different PWM algorithms, quadrature encoder or generation of the sampling period for different control systems. They can work on the DSP clock or on an external clock and can have programmable periods and counting directions. There are local programmable compare modules able to detect a time moment and to release an interrupt or to set an output. An interesting feature allows starting A/D conversion synchronized with a timer operation.
- Three *compare units* that direct PWM generation by comparing timer values and predefined time constants. Each compare unit has two associated PWM outputs, working on a time base provided by one of the timers. Outputs of the compare units can be programmed easily for different purposes, including carrier-based PWM generation or hardware SVM generation.
- PWM circuitry that include:
  - A hardware SVM state machine used for hardware implementation of one version of the SVM algorithm
  - Symmetrical or asymmetrical PWM generators
  - Programmable deadtime generator
  - Programmable output logic
- Three *capture units* that have two-level deep memory FIFO stacks, able to provide a time stamp for external events, useful in synchronization of events.
- *Quadrature encoder pulse circuit* used to interface with encoder sensors. It is able to decode and count the quadrature encoder pulses from an optical encoder.
- Interrupt logic and interface with the central processing module. A *special power drive protection interrupt* logic is included for fast shutdown of the PWM and power stage.

The *Event Manager* included in the '24x device was more complex and had additional functions. However, these functions were not used much and were replaced by other implementation alternatives.

7.8.2 Software Implementation of Carrier-Based PWM

Chapter 3 presented different carrier-based PWM algorithms. Their implementation is straightforward with a comparison of the reference signal with a triangular high-frequency waveform. The triangular waveform can be generated with a *General-Purpose Timer* programmed with the sampling interval period. Each of the continuous counting up, counting down, or up- and down-counting modes

may be used resulting in asymmetrical or symmetrical PWM generation. The comparison with the reference is ensured by the three available compare units, each one already tied within the Event Manager to a pair of PWM outputs. The compare units are updated at each sampling interval interrupt. The time intervals to be uploaded into the compare units of the Event Manager are centered around the constant corresponding to half of the sampling interval.

For instance, in order to implement a sinusoidal PWM with sub-unitary modulation index m and a sampling interval Ts, the period register of the general-purpose timer should be programmed with the integer N_Ts corresponding to the sampling interval, and each compare unit should be updated with:

$$\begin{cases} N\_Ta = \frac{1}{2} N\_T_s [1 + m \sin \alpha] \\ N\_Tb = \frac{1}{2} N\_T_s \left[1 + m \sin\left(\alpha + \frac{2\pi}{3}\right)\right] \\ N\_Tc = \frac{1}{2} N\_T_s \left[1 + m \sin\left(\alpha + \frac{4\pi}{3}\right)\right] \end{cases} \tag{7.2}$$

where α is calculated in increments depending on the sampling period and the fundamental period.

7.8.3 Software Implementation of SVM

The simplest solution for software-based implementation of the SVM algorithm on the Event Manager of Texas Instruments' DSPs is to use the symmetrical PWM-generation feature previously described in Section 7.7.2. The ON-time intervals for each switch can be calculated based on Equations (5.41) through Equation (5.46) and on the proper definition of the switching reference function. Each sector has the option to generate a switching pattern starting from the zero vector 000 or from the zero vector 111. This can be programmed by selecting the polarity of the PWM output signals. Figure 7.3 is an example of SVM generation of a vector position within the first 60° sector, using active vectors 100 and 110.

The time constants used within the PWM generation on the first 60° sector with compare modules are:

- When starting from 000:

$$\begin{cases} N\_Ta = \frac{T_0}{4} \\ N\_Tb = \frac{T_0}{4} + \frac{T_1}{2} \\ N\_Tc = \frac{T_0}{4} + \frac{T_1}{2} + \frac{T_2}{2} \end{cases} \tag{7.3}$$

  where T_1 and T_2 are calculated using Equation (5.41) through Equation (5.46) and T_0 using Equation (5.47)

- When starting from 111:

$$\begin{cases} N\_Ta = \frac{T_0}{4} + \frac{T_1}{2} + \frac{T_2}{2} \\ N\_Tb = \frac{T_0}{4} + \frac{T_1}{2} \\ N\_Tc = \frac{T_0}{4} \end{cases} \tag{7.4}$$

  where T_1 and T_2 are calculated with Equation (5.41) to Equation (5.46) and T_0 with Equation (5.47)

A different set of equations should be calculated for each 60° sector. The software routine for SVM generation also requires the proper selection and programming of the first zero vector used for each sector. Because this implementation uses both zero vectors on each sampling interval, all the sampling intervals over one fundamental frequency cycle can be programmed to start from the same zero vector, either 000 or 111. No additional programming is required on each sampling interval but for the change of the time constants within the compare units.

Similar implementations based on the reference function can be conceived with a timer that counts up or down.

7.8.4 Hardware Implementation of SVM

The Event Manager provides a hardware solution to implement the SVM. The user has to program the required polarity of the output PWM signals to enable the hardware implementation of the SVM and to start a general-purpose timer in the continuous up- and down-counting modes.

At each sampling interval, the appropriate interrupt routine has to receive or calculate the vector coordinates of the desired position of the voltage vector (V_d, V_q) to define the sector the vector belongs to. Each sector is characterized by two adjacent vectors V_x and V_{x+60} and they are programmed with the calculated vector codes in the Event Manager. Note that *Texas Instruments*' code programming definition of the sectors is the reverse of the convention used in this book. The hardware SVM generator also provides an option to select the vector to be used first in the sampling interval. The Event Manager will next use the switching pattern corresponding to these vectors for specified amounts of time.

The user software has to calculate the time intervals allocated to the first (T_1) and the second (T_2) active vectors on the sector as well as the time interval required for the zero state (T_0). Section 7.5 gives some hints to implement this calculation. The resulting constants are loaded as $0.5 * T_1$ in the first compare unit and $0.5 * (T_1 + T_2)$ in the second compare unit.

At the beginning of each sampling period the outputs are compared to the switching pattern representing the first preprogrammed active vector. We get the first compare match at $0.5 * \mathrm{T}_1$ from the beginning of the timer up counting. The outputs are switched to the pattern corresponding to either V_x, if this is selected as the first vector on the sector, or V_{x+60}, if this is selected as the second vector on the sector.

At the second compare match, at $0.5 * (\mathrm{T}_1 + \mathrm{T}_2)$, the pattern is switched to a zero vector (000 or 111), whichever is closer to the last active vector (or switching

pattern). The timer continues to count up to the programmed period, then slopes downwards, counting down. The next match occurs for the compare register loaded with $0.5 * (T_1 + T_2)$. This is used to enable the switching pattern for the "second" active vector. At the second match of the slope counting down, the switching pattern of the "first" active vector is transferred to the outputs.

The resulting waveforms are symmetric with respect to the middle of the sampling period. Figure 7.19 shows the two possible ways to generate a vector within the first sector (active vectors 100 and 110) while using the hardware generator

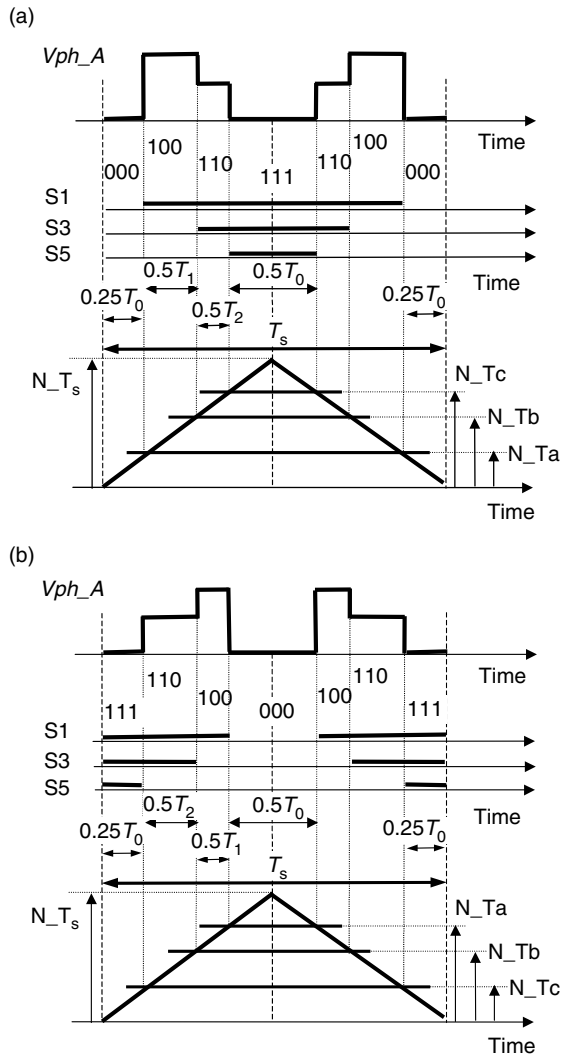

FIGURE 7.19 Example of vector synthesis using the hardware SVM generator, (a) starting the sampling interval with 000 and (b) starting the sampling interval with 111.

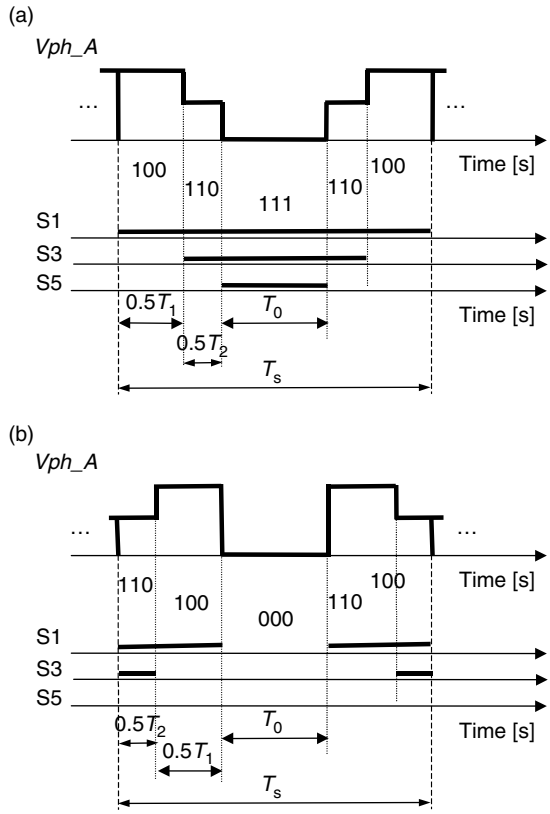

FIGURE 7.20 Example of vector synthesis using the hardware SVM generator, (a) starting the sampling interval with 100 and (b) starting the sampling interval with 110.

within the Event Manager (Figure 7.20). Switching patterns only for the high-side IGBTs are shown. Low-side IGBTs are controlled in a complementary mode.

Notice that only a zero vector is used on each sampling interval, allowing the use of same zero vector for a whole 60° sector. The first consequence relates to lack of switching on one of the inverter legs for 60° and this reduces switching losses. In other words, the hardware implementation of the SVM method within the Event Manager represents a reduced-loss PWM algorithm, suitable for motor control, in which current usually lags behind the voltage. This method (*Method DD1*) has been already analyzed in Chapter 5, Figure 5.26. At the price of some extra switchings per fundamental period, other reduced-loss SVM algorithms may be implemented on the same hardware.

7.8.5 Deadtime

A deadtime generator is also included within the Event Manager. A single value should be programmed for all six outputs and if the deadtime generator is

enabled, the rise-up (turn-ON) of each of the output signals is delayed by a preprogrammed time interval. The deadtime interval depends on clock period, clock prescaling, and the 4-bits programming constant covering all time intervals required by modern power semiconductors.

7.8.6 Individual PWM Channels

Texas Instruments' DSPs also include several PWM channels able to work individually directly from a timer. The procedure for programming each individual PWM channel is very easy, starting from setting up a timer as the period counter and loading a compare constant in a special register. When the timer reaches the value preloaded within the compare register, an interrupt can be generated and an output toggles.

7.9 CONCLUSION

This chapter presented the trends in the control of medium and high-power converters. Different theoretical principles for defining a PWM channel by analog or digital means were introduced along with various implementation solutions using modern microcontrollers and DSPs. This chapter is rich in presenting novel solutions to a state-of-the-art power converter control structure.

REFERENCES

1. Bob C, Getting More Out of Your Motor — Motor Driver ICs Integrate More Functions. Allegro Microsystems, Application Note, 2003.
2. Neacsu DO, A new digital controller for SVM algorithms, *IEEE SCS*, vol 1, pp. 101–104, 1993.
3. Tzou YY and Hsu HJ, FPGA realization of space-vector PWM control IC for three-phase PWM inverters. *IEEE Trans. PE*, 953–963, 1997.
4. Sangchai W, Wiangtong T, and Lumyong P, FPGA-Based IC design for 3-phase PWM inverter with optimized space vector modulation schemes, IEEE Midwest Symposium on CAS2000, 2000, 106–109.
5. Deng D, Chen S, and Joos G, FPGA implementation of PWM pattern generators for PWM invertors, Canadian Conference on Electrical and Computer Engineering, 2001, pp. 225–230.
6. Tonelli M, Battaiotto P, and Valla MI, FPGA implementation of an universal space vector modulator. IEEE IECON, 2001, pp. 1172–1177.
7. Cirstea M, Aounis A, McCormick M, and Urwin P, Vector control system design and analysis using VHDL (for induction motors). IEEE PESC, 2001, pp. 81–84.
8. Iverson, 1962.
9. Zadeh LA, Fuzzy sets. *Inf. Control*, 8: 338–353, 1965.
10. Negoita CV and Ralescu DA, *Applications of Fuzzy Sets to Systems Analysis*. Birkhauser Verlag and New York Halsted Press, Basel, 1975.
11. Mamdani EH and Assilian. An experiment in linguistic synthesis with a fuzzy logic controller, *Int. J. Man Mach. Stud.* 7: 1–13, 1975.
12. Neacsu D, Stincescu R, Raducanu I, and Donescu V, Fuzzy Logic Control of an PWM V/f Inverter-Fed Drive. ICEM'94, Paris, France, 1994, vol III, pp. 12–17.

13. Saetieo S and Torrey D, Fuzzy logic control of a space vector PWM current regulator for three-phase power converter, *IEEE Trans. PE*, 13: 419–426, 1998.
14. Anon, NEC 78098x MCU, NEC, Datasheet.
15. Moynihan F, Fundamentals of DSP-Based Control for AC Machines, Analog Dialogue, 33–4, 2000.
16. Takahashi T, FPGA-based High Performance AC Servo Motor Drive — Accelerator TM Configurable Servo Drive Design Platform, International Rectifier, 2001.
17. Mohan T, Undeland T, and Robbins, W, *Power Electronics — Converters, Applications and Design*, Wiley, 1995.
18. Thornborg K, *Power Electronics*, Prentice Hall, 1988.
19. Holtz J, Pulsewidth modulation — a survey, *IEEE Trans. IE*, 39: 410–420, 1992.
20. Anon., Mixed Signal DSP Controller ADSP-21990, Analog Devices, Datasheet, 2003.
21. Anon., IXYS deadtime.
22. Anon, Bus Compatible Digital PWM Controller, IXYS IXDP610 Documentation, 2001.
23. Anon., TMS320LF/LC240xA DSP Controllers Reference Guide — System and Peripherals, TI Literature no. SPRU357B, 2001.
24. Yu Z, Space Vector PWM with TMS320C24x/F24x using Hardware and Software Determined Switching Patterns. TI Literature no. SPRA524.
25. Doval-Gandoy J, Iglesias A, Castro C, and Penalver CM, Three Alternatives for Implementing Space Vector Modulation with the DSP TMS320F240, IEEE PESC 1999.
26. Trzynadlowski AM, *The Field Orientation Principle in Control of Induction Motors*, Kluwer Academic, 1994.
27. Anon., DSP 56301, Motorola, Datasheet.
28. Anon., MC68HC908MR32, Motorola, Datasheet.
29. Neacsu D, Implementation of PWM Algorithms, IEEE PESC 2005.