Fast FPGA prototyping to explore and compare new SPWM strategies

Sergio R. Geninatti, Manuel Ortiz-López, Fco. Javier Quiles-Latorre, Tomás Morales-Leal, Andrés Gersnoviez, Antonio Moreno-Muñoz, *Senior Member, IEEE*

***Abstract*—This study presents a Flexible Test Bench (FTB) implemented with FPGA that synthesises of a large number of strategies that apply "Spread Spectrum" to reduce the energy of the fundamental harmonics present in a conventional Pulse Width Modulation (PWM). The FTB not only incorporates most of the known spread spectrum techniques but also allows to combine them and even to easily create new ones, thus providing a highly flexible test bench. The FTB has been described in standard VHDL, so it can be synthesised using any synthesis tool, and the number of logical resources used by each of them can be determined according to the spread spectrum strategy configured in its registers. Therefore, a tool is available to choose, according to the spectral response and the consumption of logic resources, the most suitable PWM spread spectrum strategy for each application. The particularity of the proposal is to have a unique hardware platform that allows the comparison of different techniques under the same physical implementation conditions since geometry and physical location are important when evaluating the conducted and radiated emission of the circuits. Another objective of our FTB is to solve in a unified way all the numerical aspects present in the FPGA-based implementation so that the numerical ranges and roundings used affect all the implemented techniques equally.**

***Index Terms*—Spread Sprectrum, Supraharmonics, PWM, FPGA.**

I. Introduction

FOR several years scientific research efforts have been focused on studying the reduction of electromagnetic emissions in power converters. The demanding applications range from small power in the LED lighting range to motor control for vehicle drives or DC/AC converters for renewable energy.

The use of self-commutated power electronic converters (PECs) in LED lighting to improve the total power factor (TPF) has increased the problem of harmonic emissions by shifting them from the hertz to the kilohertz range. Especially significant are electromagnetic emissions in the range of 2 kHz to 150 kHz, known as supra harmonics. Numerous failures and anomalous behaviours of electronic equipment have been reported, e.g. in powerline communications, household appliances, smart meters, etc. [1].

In [2] two strategies are proposed to reduce the emission of harmonics, one of which proposes to reduce emissions along the propagation path using filtering devices. New passive mitigation technologies have been proposed, such as a variable capacitance filter [3]. However, this strategy has the disadvantage that it requires more space on the Printed Circuit Board (PCB) and obviously increases the cost, so it is preferable to find another solution to reduce the emission of these interferences. The other strategy consists of several methods that try to reduce emissions at the same source. One of the most used methods to reduce the effects of radiated and conducted emission is Spread Spectrum [4], [5].

Spread Spectrum causes the energy concentrated in the clock harmonics to be distributed over a larger band of frequencies [6]. While the total average energy does not decrease [7] and its dispersion may in some cases be detrimental [8], or neutral [9], it is possible to place the harmonics where they are favoured in the filter design or where they can be absorbed by some natural condition. However, the important point is that Spread Spectrum allows unwanted ElectroMagnetic Interference (EMI) levels to be reduced so that EMI standards can be met [4]. Spread Spectrum is a subject of much scientific and technological discussion as to whether it "enhances" or "masks" the action of interference [10], so it is very important to analyse its effects on specific applications and provide selective treatment in each case. It can also be used in applications where acoustic noise and vibrations must be reduced, as in [11]. Another area where Spread Spectrum can be used to reduce EMI is in Power Line Communications (PLC). In the paper [12] it is shown that PLC is affected by EMI generated by power converters operating in close proximity and its reduction is analysed by comparing the sine, sawtooth and random modulation type. Therefore, Spread Spectrum can be used in multiple applications to reduce the interference caused by any type of DC-AC and DC-DC switched-mode power converters.

In the works referenced above and in those that will be indicated in the following section, only a single strategy has been used, or a comparison of a maximum of three basic strategies has been made in an application, indicating which would be the most appropriate type for EMI reduction, but the hardware implementation cost of each one has not been analysed. Therefore, it would be valuable to have a test bench that, with the same hardware architecture, is able to compare as many Spread Spectrum based PWM (SPWM) strategies as possible, both with respect to harmonic reduction and final implementation cost. On the other hand, it would also be desirable to be able to easily create new strategies and to be able to simulate the influence of the load, as its behaviour is normally dynamic. To this end, this paper presents a Flexible Test Bench (FTB) based on FPGA that allows testing all known Spread Spectrum methods and combining them to generate new ones using a single hardware architecture. In addition, the FTB allows rapid prototyping that resembles an industrialisable implementation as closely as possible. This

does not mean that an industrial-level reproduction using FPGAs is intended, but rather that the architecture, numerical representation, and logic solution are similar to a dedicated circuit that can be reproduced on a large scale.

The paper is organised as follows: Section II lists some of the work on SPWM with an emphasis on FPGA-based SPWM; Section III describes the parameters of a PWM signal whose variation determines the different SPWM methods, the design specifications of the FTB, its architecture, how to configure it, and the influence of the load; Section IV shows the results of the FPGA implementation of each of the SPWM methods; Section V presents some experimental results showing the validity of the FTB; finally, Section VI presents the conclusions.

II. Related works

There are two trends in the implementation of Spread Spectrum techniques. On the one hand, solutions that incorporate Spread Spectrum as an incremental innovation of conventional PWM modulation propose a modification of the clock frequency to disperse harmonics. The implementation of these techniques has been carried out in low-cost commercial integrated circuits. One of the first examples of this was offered by On Semiconductor (P6P82PS01A), which implements a variable clock generator to couple to a conventional PWM modulation chip to modify its fundamental frequency. Subsequently, many other solutions proliferated, such as Texas Instrument's LP8867 and TPSM5601, On Semiconductor's NCV890204, and Analog Devices' LT3795. All of them implemented variations in the main frequency followed by a conventional PWM.

The other, more complex, trend consists of developing and simulating Spread Spectrum with high-level tools such as MATLAB Simulink to be implemented in an embedded system based on microcontrollers or DSP. These implementations are valid for applications such as motor control, electric vehicles, or DC/AC converters for renewable energy, the cost of which justifies an embedded system, but it questions their feasibility in very low-cost, high-volume applications such as LED lighting or switch-mode power supplies. In this regard, several studies on the implementation on microcontrollers and FPGAs of SPWM techniques have been proposed in the literature. In [13] a chaotic pulse width modulation method to spread the power spectrum of an induction motor drive system is proposed. An 80C196 microcontroller is employed for the generation of chaotic numbers and a triangular carrier with a chaotic frequency is produced using an 80C196 microcontroller and a MAX038 frequency modulator. Work such as [14] showed that the control capabilities in PWM power converters were higher in FPGA-based control than in DSP. On the other hand, the need for rapid prototyping and a reprogrammable configuration has led to many FPGA-based approaches emerging [15]. Powerful co-simulation tools are now available in environments such as System Generator from Xilinx which facilitate the implementation process. In this regard, in [16] a Random Frequency PWM for an inverter of a 1-horsepower (hp) motor is implemented with an FPGA. The proposal is modeled with VHDL language and co-simulated using the HDL Cosimulation toolbox from the Matlab-ModelSim environment. [11] presents an FPGA-based random PWM control of a Buck DC/DC converter. The authors of this work propose the use of FPGAs in preference to a CPU or DSP due to their capability to execute concurrent operations. The developed design has a modular structure described in VHDL language and is successfully synthesized and tested with an embedded Altera Cyclone III FPGA. Another similar study presenting the implementation on FPGA of a random PWM is developed in [17]. A digital PWM inverter control system based on FPGA XC4010 is proposed in [18]. Several digital modulation techniques for inverters are implemented on an FPGA Spartan 3E using Matlab-Simulink environment [19], using VHDL language for producing the required switching pulses and the simulation is performed with ModelSim. The authors propose the storage by lookup tables of the sinusoidal signals generated for the modulation waveforms. [20] uses systemC and a Xilinx Spartan 3E FPGA for the design of a PWM with harmonic reduction. Following this same line of harmonic reduction, in [21] another FPGA-based random modulation technique for DC/DC converter is presented. [22] shows the implementation using an Altera FPGA of many spread-spectrum schemes for conducted-noise reduction in DC-DC converters. Finally, in [23] a strategy based on PWM frequency variation is implemented in FPGA to expand the harmonic spectrum. The work focuses on the implementation of the fixed-point operations and on solving the implementation details of the algorithm. On the other hand, it also proposes a method to generate in FPGA the pseudo-random sequence that controls the PWM frequency.

All the works shown in this section show the implementation of specific spread spectrum strategies, but there are no known works that show a test bench to test different strategies with the same prototype and to create new ones easily and to simulate the behaviour of the load with different types of PWM modulation, hence the usefulness of the FTB shown in this paper.

III. Materials and methods

A. Spread Spectrum fundamentals

In order to design a circuit capable of synthesising all possible SPWM modes, the parameters that differentiate these modes must be studied. The starting point is the four parameters defined in Fig. 1, where:

T_i = PWM full cycle time.
α_i = Power on time.
ϵ_i = Wait time before α_i.
θ_i = Remaining cycle time after α_i.

From which the frequency $f_i = 1/T_i$ and duty cycle $d_i = \alpha_i/T_i$ are derived.

The Δ prefixes in each parameter correspond to the time variation introduced to implement the SPWM. The FTB developed allows defining and relating each of these variations. Table I shows a classification of the most common randomness creation modes to produce PWMs based on Spread Spectrum.

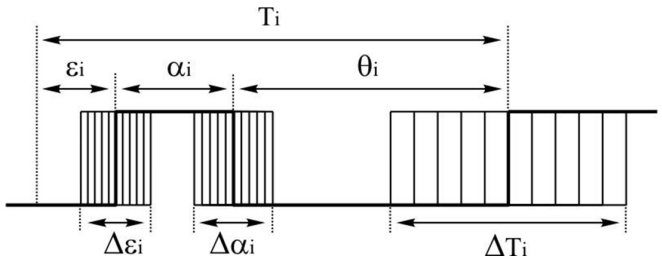

Fig. 1. PWM signal components.

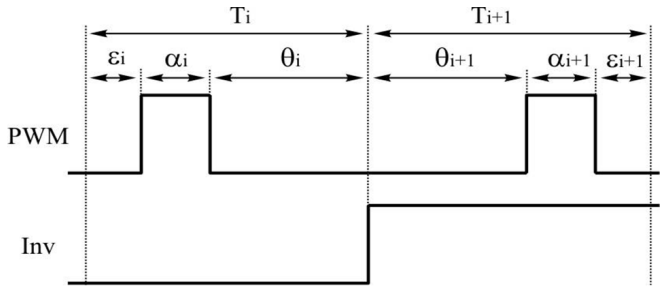

Fig. 2. Effect of the signal *Inv*.

TABLE I
RANDOM PWM MODES.

Mode	**Submode**	T_i	α_i	ϵ_i	d_i
Fixed	PWM	C	C	C	C
Random	RPPM	C	C	R	C
Random	RPWM	C	R	C	R
Random	RCFMFD	R	R	C	C
Random	RCFMVD	R	R	C	R
Random	RDRPPMFCF	C	R	R	R
Random	RCFRPPMFD	R	R	R	C
Random	RRRM	R	R	R	R

In this table, the acronyms R stands for Random; C for Constant; RPPM for Random Pulse Position Modulation; RPWM for Random Pulse Width Modulation; RCFMFD for Random Carrier Frecuency Modulation Fixed Duty; RCFMVD for Random Carrier Frecuency Modulation Variable Duty; RDRPPMFCF for Random duty ratio RPPM Fixed carrier frequency; RCFRPPMFD for Random carrier frequency RPPM Fixed duty ratio; and RRRM for Random carrier frequency RPPM Random duty ratio.

B. Design specifications

The FTB should implement most of the known Spread Spectrum methods, combine them and even add new ones easily.

In order to include the maximum number of methods in the FTB, the compilation of modes presented by [22] and extended in [4] will be used, where an exhaustive study is carried out analysing 7 random PWM modulation modes, in which 9 RPPM submodes, 3 clock variation functions (sine, triangular and exponential), 2 Chaotic PWM modes (continuous and discrete) and 2 "Programmed PWM" modes (harmonic cancellation and optimisation) are considered. In addition, [4] implements 4 of the clock variation modes, 3 periodic and one chaotic. Finally, another mode present in [24], Random PWM (RPWM), which is useful in LED lighting, is also considered.

Another feature that the FTB must have is the possibility of incorporating the PWM modulation signal in the Spread Spectrum study. Whatever the application, the load will always be a dynamic element that will cause variations in α_i, affecting the harmonic distribution. Assuming that the modulation signal introduces its own spread spectrum, a memory to store the modulation waveform is incorporated into the FTB, as solved by [16] and [19], which store sine signals in lookup tables. In our case, we extend it to other forms of modulation due to any type of load, using the concept of Direct Digital Synthesis (DDS). With this approach, the FTB also allows to study the effects of harmonic dispersion produced by the modulation superimposed on the PWM main frequency variation.

Another condition that has been imposed on the FTB is that it is possible to establish relationships between the parameters. For example, it enables defining a law of variation for T_i and set α_i to be proportional to T_i to ensure constant d_i (RCFMFD in [22]).

The FTB will also be provided with a digital modifier called *Inv*, useful to affect the symmetry of the PWM output as long as the cycle time is constant. The state of the *Inv* signal allows the ϵ_i and θ_i times to be interchanged, depending on whether it is set to '0' or '1'. As *Inv* is generated in a pseudo-random way it causes the spectrum spread in a simple way. The effect of the *Inv* signal is shown in Fig. 2 and is useful for implementing ramp inversion modes.

Finally, it is re-emphasised that, although the FTB implements all the spread spectrum strategies described by [22], [4], [24], [16] and [19], it has been designed in such a way that it is easy to combine these methods and even to create new ones.

C. Description of the FTB architecture

The FTB was conceived as an IPcore that could be implemented on any PCB with an FPGA and other resources to realise the user interface. This provides a flexible tool on which all known SPWM methods can be tested or new ones can be created. For this purpose, the IP core was divided into two distinct blocks. One block is responsible for generating the spread spectrum PWM output signal from the values contained in configuration registers that define the desired operating mode. This block is the one described in more detail in this work. The other block is the user interface, which allows the user to choose between the different possibilities offered by the FTB by entering the parameters desired by the user in the configuration registers. The user interface must be adapted to the communication possibilities of the user with the PCB (number of switches, pushbuttons, displays, etc.) on whose FPGA the FTB will be implemented. If the PCB has a simple serial interface, a user interface can be synthesised to communicate with a computer or tablet and operate the FTB with a graphical interface.

Fig. 3 shows the block diagram of the FTB. The communication between the two blocks of the IPcore is done through

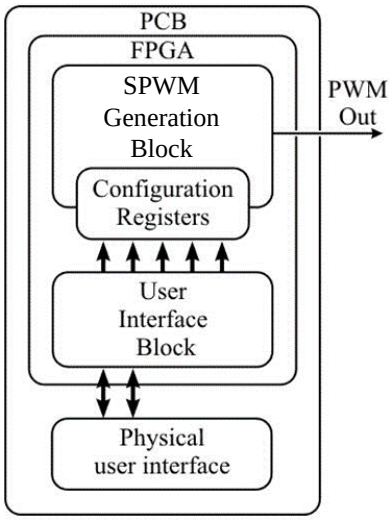

Fig. 3. FTB block diagram.

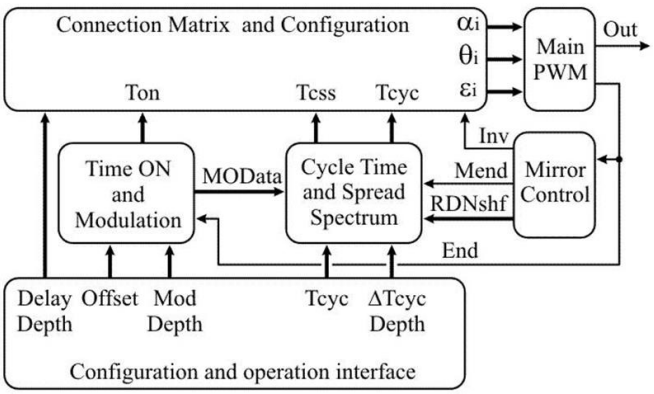

Fig. 4. Block diagram of the SPWM generation block.

the configuration registers, which are queried at the end of each PWM cycle. The values of these registers can be changed by the user at any time to determine the SPWM modulation mode.

The robustness of the FTB design lies in the definition of basic blocks that perform the functions common to all SPWM strategies. Fig. 4 shows the architecture of the SPWM generation block.

The entire system is synchronised to a fixed frequency clock of 50 MHz (*Fclk*), which can be changed to fit the resolution and frequency of a specific application.

One detail to note when describing the blocks is the importance of the numerical representation in FPGA computation. In this case, fixed point has been used, adapting the width of each bus to the numerical range of the operations to guarantee that there will be no overflow and that truncation will be controlled. Careful design of these details allows us to evaluate whether the proposed architecture is suitable for implementation in very restricted and low-cost digital systems, such as LED lighting or switched-mode power supplies. To facilitate the understanding of the internal connection buses and the range of operations, the bit width of each connection bus has been indicated.

Each of the blocks that form the PWM signal generation block is described in detail below.

1) Main PWM block: The *Main PWM* block receives 3 input signals (ϵ_i, α_i and θ_i) to implement the output signal shown in Fig. 1. It generates a pulse of width α_i which is located after a time ϵ_i from the beginning of the cycle, and at the end of the pulse it keeps the output inactive for a time θ_i to complete the period T_i. The output signal complies with:

$$T_i = \epsilon_i + \alpha_i + \theta_i \tag{1}$$

Main PWM is implemented with an up-counter synchronised by the *Fclk* signal which causes the output to switch when it reaches the values ϵ_i and $\epsilon_i + \alpha_i$ respectively. This block also provides an *End* signal that indicates to the other components of the FTB the end of a PWM cycle.

Main PWM also provides an End signal that indicates the end of a PWM cycle.

2) Configuration and Operation Interface (COI): This block allows all the options offered by the FTB to be configured and easily operated, facilitating the comparison between different Spread Spectrum strategies. The values of these registers are queried at the start of each PWM cycle. The configuration registers can be distinguished between those that hold numeric values and those that select the FTB options for implementing each of the SPWM modes.

Describing the group of registers that store the numerical parameters, this is composed of:

- *Tcyc*: Mean PWM cycle time, which defines the centre frequency around which the variations for Spread Spectrum will be applied.
- Δ*Tcyc Depth*: Expressed in percentage and represents the depth of action of the function that modifies *Tcyc* to achieve Spread Spectrum.
- *Offset*: Constant value that is part of the PWM pulse duration (*Ton*), and is defined as a percentage of the reference PWM cycle time (*Tcyc*).
- *Mod Depth*: It is expressed as a percentage and represents the modulation depth of the pulse.
- *Delay Depth*: This is expressed as a percentage and defines the pulse delay at the start of the PWM cycle as a proportion of its selected time base.

Concerning the group of registers that define the strategy to expand the spectrum, Tables II and III describe how, from their bits, the SPWM strategy that the FTB can implement is defined. Fig. 5 shows the bits of the configuration register that act on each block of the FTB. 16 copies of this configuration register have been created in order to switch quickly from one mode to another and to facilitate their comparison.

A 4-bit configuration register selects the active register out of the 16 available. The meaning of each bit (modifier) of the configuration register and its influence on the configuration of each block will be detailed in the following sections. Also, the usefulness of this register for calculating the FPGA resource usage of each of the tested Spread Spectrum strategies will be discussed in Section IV.

TABLE II
CONFIGURATION REGISTER.

Bit	Description
SW14 SW13 SW12	*Ton* modulation prescaler: Controls the increment of the modulation ROM addresses to be performed between 1 and 9 PWM cycles.
SW11	SW11 = 1 - Activates modulation of α_i SW11 = 0 - α_i fixed
SW10 SW9 SW8	Select the T_i variation function as detailed in Table III.
SW7	Applies absolute value to the source of variation selected by SW10/9/8
SW6	Controls random source update by shift register (RDNshift). SW6 = 1 - Updated at the end of each PWM cycle SW6 = 0 - Updated every two cycles of PWM
SW5	Selects the maximum duration of ϵ_i: SW5 = 1 - ϵ_i defined by SW10/9/8 (*Tcss*) SW5 = 0 - ϵ_i fixed (*Tcyc*)
SW4	Activation of ϵ_i: SW4 = 1 - ϵ_i defined by SW5 SW4 = 0 - ϵ_i
SW3	Enables random inversion of *Inv* via the BIT-random source. SW3 = 1 - Random inversion enabled. SW3 = 0 - Random inversion disabled.
SW2	Symmetrical cycle activation: SW2 = 1 - Generates symmetrical cycle consisting of two mirrored half cycles. The mirror consists of inverting the switch-off times ϵ_i and θ_i. SW2 = 0 - Symmetrical cycle deactivated.
SW1	Maximum value of α_i: SW1 = 1 - Defined by SW10/9/8 (*Tcss*) SW1 = 0 - Fixed value (*Tcyc*)
SW0	Value of T_i: SW0 = 1 - Defined by SW10/9/8 (*Tcss*) SW0 = 0 - Fixed value (*Tcyc*)

TABLE III
SELECTABLE VARIATION FUNCTIONS FROM THE CONFIGURATION REGISTER.

Conf. Reg. bits			Description
SW10	SW9	SW8	
0	0	0	Random
0	0	1	Sine
0	1	0	Triangle
0	1	1	Exponential
1	0	0	Linear chaotic
1	0	1	Random - shift
1	1	0	User defined
1	1	1	Pulse modulation

3) Cycle Time and Spread Spectrum Block (CTSSB): This block defines the PWM cycle time using *Fclk* as the time base. The internal diagram of the block is shown in Fig. 6.

CTSSB generates the two parameters *Tcyc* and Δ*Tcyc*, which affect the duration of each SPWM cycle. Internally CTSSB generates the PWM cycle variation functions (*Fssk*) using DDS. Using 3 bits of the configuration register (*SW8*, *SW9* and *SW10*) it is possible to select one of the eight waveforms stored in ROM.

The values of the selected variation function $Fssk_{(i)}$ are read consecutively by an address counter (*ADDR CNT 2*) which is incremented by the *Mend* signal at the end of each PWM cycle or alternately, depending on the configuration of the Mirror Control Block.

Each value of the selected Fssk function is multiplied by the depth factor Δ*Tcyc* (*MUL1*) and then by the mean cycle time

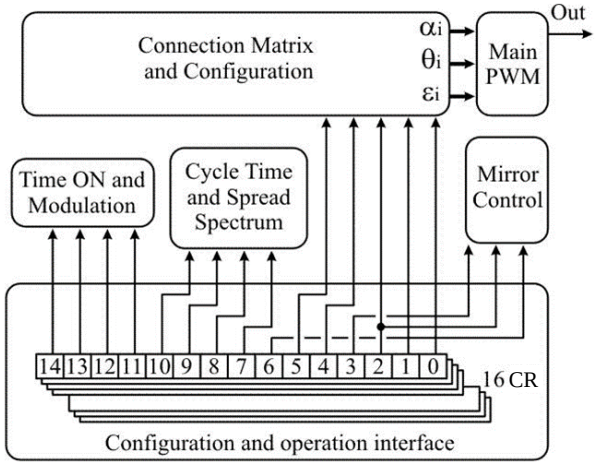

Fig. 5. Configuration register bits that act on each block of the FTB.

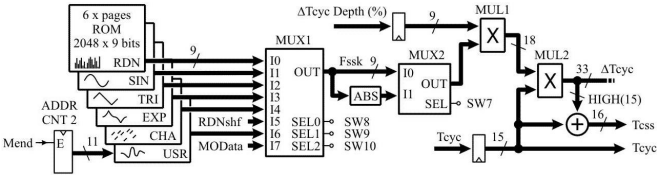

Fig. 6. Cycle Time and Spread Spectrum Block.

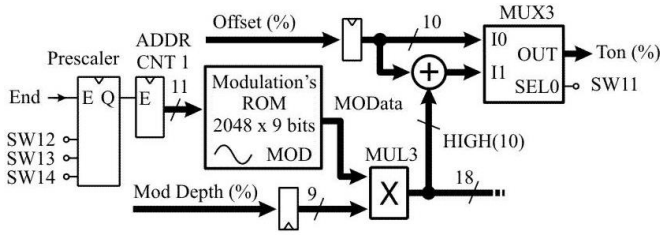

Fig. 7. Time ON and Modulation Block.

Tcyc (*MUL2*). In this way, the percentage variation $\Delta Tcyc$ is expressed in clock cycles, and the result is added to *Tcyc* to obtain the value *Tcss*.

Equation (2) shows the expression of the modified cycle value for SPWM:

$$Tcss_{(i)} = Tcyc * (1 + Fssk_{(i)} * \Delta Tcyc * 2^{-18}) \quad (2)$$

CTSSB provides two numerical values to the next stage, *Tcyc* and *Tcss*, expressed in cycles of *Fclk*.

Bit *SW7* of the configuration register applies absolute value to the variation *Fssk* function.

4) Time ON and Modulation Block (TOMB): This block defines the duration of the active PWM pulse as a percentage of the total PWM cycle. The internal diagram of the block is shown in Fig. 7.

SW11 in the configuration register defines the *Ton* output of the block, which can be either the constant numeric value *Offset* or the same *Offset* value plus a variable modulation value.

The modulation function is obtained by DDS from a table stored in a ROM. As with the previous block (CTSSB), the data is extracted cycle by cycle using an address counter (*ADDR CNT 1*) which is updated with the *End* signal.

Under the assumption that the PWM modulation can evolve more slowly than the Spread Spectrum variation, a prescaler has been added to be able to adjust the change of the modulation signal.

TOMB also incorporates a parameter to adjust the modulation depth (*Mod Depth*) using the *MUL3* multiplier.

5) Mirror Control Block (MCB): Inverting the PWM pulse from the beginning to the end of the cycle is a feature used by some SPWM strategies (Fig. 8). MCB generates the *Inv* signal that controls the inversion.

Fig. 9 shows the internal diagram of MCB. It is a circuit synchronised by *Fclk* and evolving under the control of the *End* signal. A pseudo-random sequence with a shift register has been implemented for the ramp RPPM modes, as shown in Fig. 8. A symmetry control called *Mirror* allows implementing the triangular reference RPPM modes as shown in Fig. 10.

Bit *SW3* triggers random inversion while symmetry is controlled by *SW2* and *SW6*. The *Mend* signal increments the *ADDR CNT 2* address counter alternately for the case of variable cycle ramps that must be symmetrical.

6) Connection Matrix and Configuration Block (CM&CB): As seen in the previous sections, the CTSSB block defines the time base of the SPWM, and the TOMB block the pulse

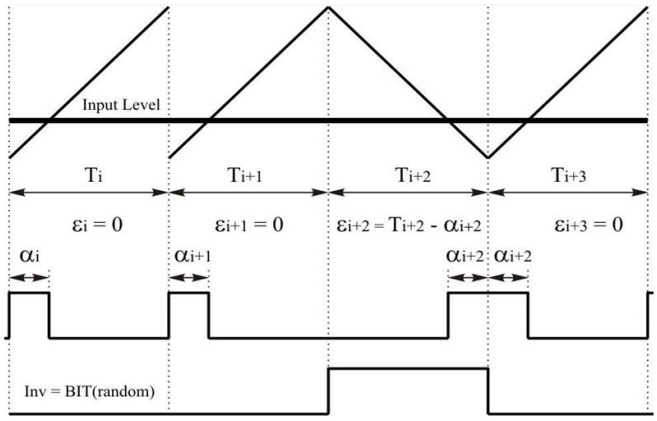

Fig. 8. Implementation of RPPM using the *Inv* signal.

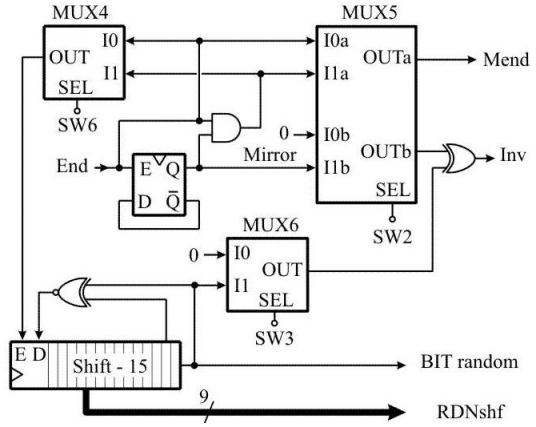

Fig. 9. Mirror Control Block.

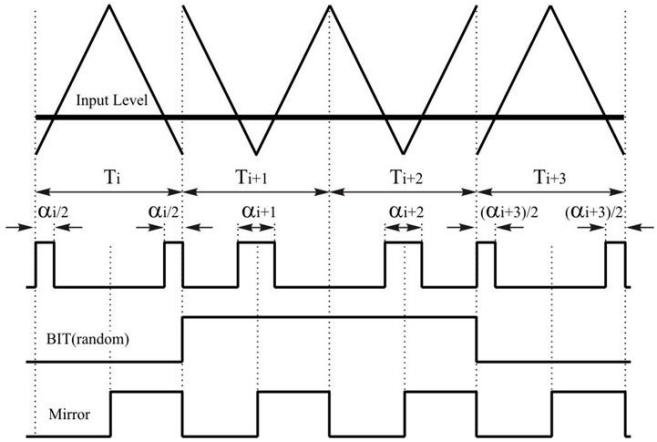

Fig. 10. Effect of the random inversion and mirror signals

width in a proportional manner. In this section it will be shown how CM&CB allows carrying out all possible combinations of the parameters generated by the SPWM and TOMB blocks, to produce different SPWM modes. Fig. 11 shows the internal organisation of CM&CB.

The inputs *Tcyc* and *Tcss* are the PWM time references, both of which can be divided by two to give the option of symmetrical modes as in Fig. 10.

TABLE IV
CONFIGURATION REGISTER FOR THE 8 MOST COMMON SPWMS.

Configuration Register bits															SPWM mode
14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
-	-	-	0	-	-	-	0	0	0	0	0	0	0	0	PWM Standard
-	-	-	0	-	-	-	0	0	0	0	1	0	0	0	RPPM1
-	-	-	0	X	X	X	0	0	1	1	0	0	0	0	RPPM2
-	-	-	0	X	X	X	0	0	0	0	0	0	1	0	RPWM
-	-	-	0	X	X	X	0	0	0	0	0	0	1	1	RCFMFD
-	-	-	0	X	X	X	0	0	0	0	0	0	0	1	RCFMVD
-	-	-	0	X	X	X	0	0	1	1	0	0	1	0	RDRPPMFCF
-	-	-	0	X	X	X	0	0	1	1	0	0	0	1	RCFRPPMFD
-	-	-	0	X	X	X	0	0	1	1	0	0	1	1	RRRM

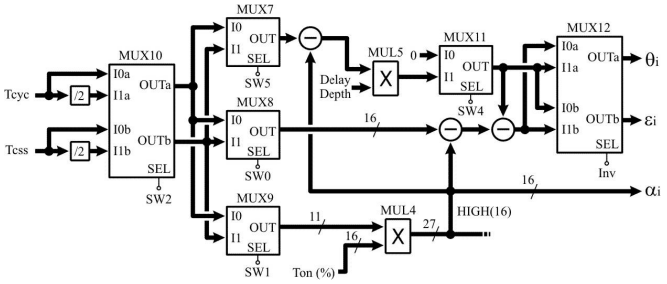

Fig. 11. Connection Matrix and Configuration Block.

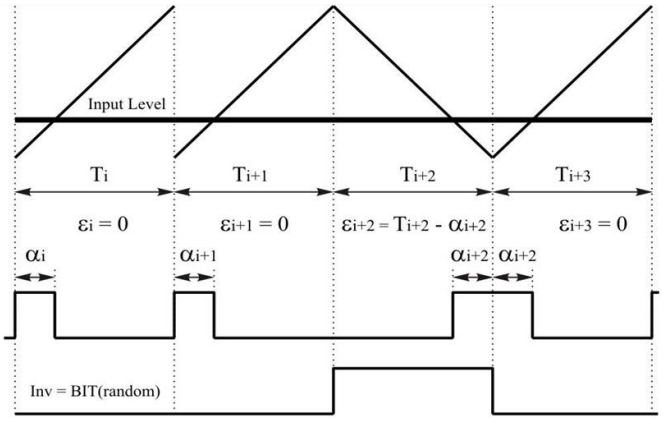

Fig. 12. RPPM1 strategy.

Using *SW0*, *SW1* and *SW5*, the time base for α_i, θ_i and ϵ_i respectively is defined individually by selecting between the two options *Tcyc* and *Tcss*.

The *MUL4* multiplier uses the time base selected by *SW1* to calculate the *Fclk* cycles corresponding to α_i. Similarly, ϵ_i is calculated using the time base selected by *SW5* from which α_i is subtracted, resulting in the total PWM OFF time (*Toff*). The *MUL5* multiplier affects *Toff* by a ratio coefficient called *Delay Depth* to obtain ϵ_i.

Finally, the time base selected by *SW0* is subtracted from α_i and ϵ_i to obtain θ_i which corresponds to the remaining time of the PWM cycle.

The *Inv* signal controls a two-way output selector to define whether ϵ_i and θ_i go before or at the end of the PWM cycle.

D. FTB Configuration

This section shows some examples of how the FTB can be configured to achieve the most common SPWM modes and how new ones can be obtained from them by modifying some of the bits in the configuration register. In addition, Section III-E describes how load effects can be simulated, which in turn will allow new SPWM modes to be generated. This is intended to demonstrate the flexibility of the FTB.

1) Configuring the most common SPWM modes: The FTB has been designed to implement the 8 most common modes studied in [22] and to introduce variants in the variation functions. Table IV shows the values that the configuration register must have to implement each of the 8 modes.

All the modes in Table IV expand the spectrum by a random variation acting on one of the PWM characteristic values (T_i, α_i, ϵ_i, θ_i). These variations are achieved by a function stored in a ROM (DDS) whose content is defined in VHDL, except for "Random - shift" which is generated by a linear feedback shift register (LFSR). Therefore, one of the 8 functions listed in Table III can be chosen via bits 8, 9, and 10 of the configuration register.

By replacing the "X" in Table IV with the values in Table III, it is possible to compare different variation laws in each of the 8 modes described in [22]. Bits defined with "-" indicate that the selected DDS function has no effect on the PWM.

2) Creating new SPWM modes: The FTB also allows other new modes to be obtained by introducing variations or even combining strategies from different modes by simply modifying some bits in the configuration register. To show the procedure, the basic PWM will be used as a starting point and, from this, six different modes will be obtained, simply by successively modifying some bits of the configuration register. To obtain the basic mode of spread spectrum RPPM1 in Fig. 12, simply set bit *SW3* to 1, as shown in Table IV. This activates the random inversion by randomly exchanging fixed values of α_i and θ_i.

As can be seen, when an inversion of α_i and θ_i occurs, one switching is lost, because the adjacent pulses are merged. To avoid this effect, a non-zero ϵ_i can be added as shown in Fig. 13, by activating *SW4* in the configuration register and setting the proportional value to *Tcyc* with *Delay Depth*.

In addition, if it is desired to make each cycle symmetrical, *SW2* can be activated in the configuration register, giving the mode shown in Fig. 14.

Another variation function can also be added by having the

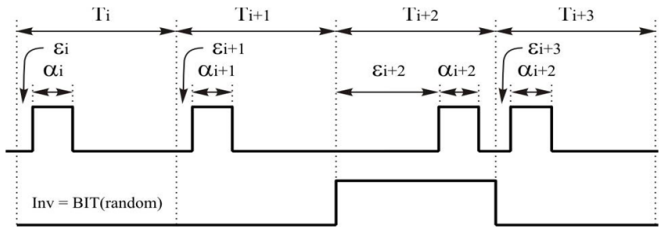

Fig. 13. RPPM1 strategy with non-zero ϵ_i.

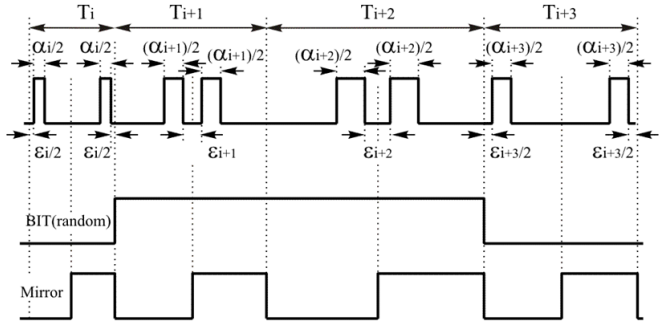

Fig. 14. RPPM1 strategy with non-zero ϵ_i and symmetry.

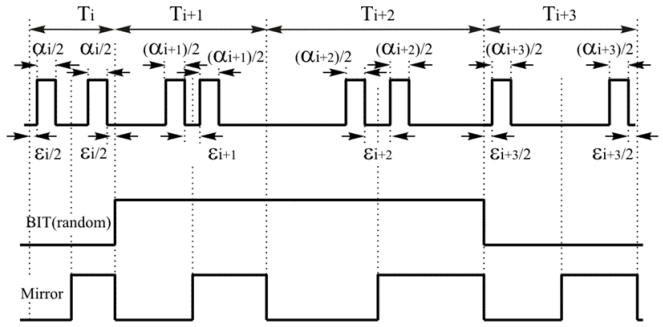

Fig. 15. RPPM1 strategy with non-zero ϵ_i, symmetry and variable T_i.

cycle time T_i modified by one of the available DDS functions. This is possible by activating SW0 in the configuration register and selecting the variation function with *SW10/9/8*. The time behaviour of the new mode is shown in Fig. 15. As can be seen, T_i varies while ϵ_i and α_i remain constant, the long-term average output value remains constant, but cycle by cycle it varies by the variation of T_i.

If the duty factor has to be respected cycle by cycle, α_i must be made proportional to T_i, which is achieved by activating *SW1* in the configuration register. This results in the mode shown in Fig. 16.

In the latter case, ϵ_i is kept constant, but if it is desired to make ϵ_i proportional to T_i, *SW5* can be activated in the configuration register and the behaviour of Fig. 17 is obtained.

This example shows how new modes have been created from the basic PWM.

It is noteworthy that the FTB allows all variants to be compared with the same electrical circuit, which validates a comparison on equal terms while allowing the hardware resources required to be estimated for each case and projected to a production-scale VLSI implementation.

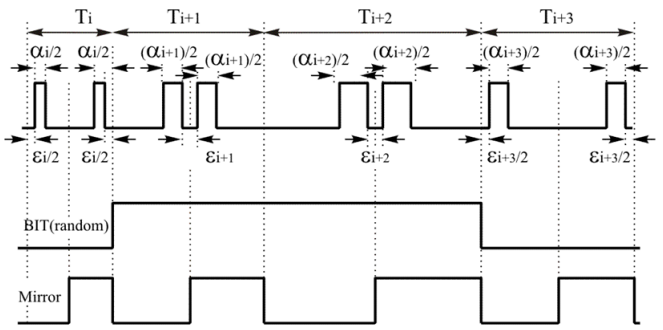

Fig. 16. RPPM1 strategy with non-zero ϵ_i, symmetry, variable T_i and proportional α_i.

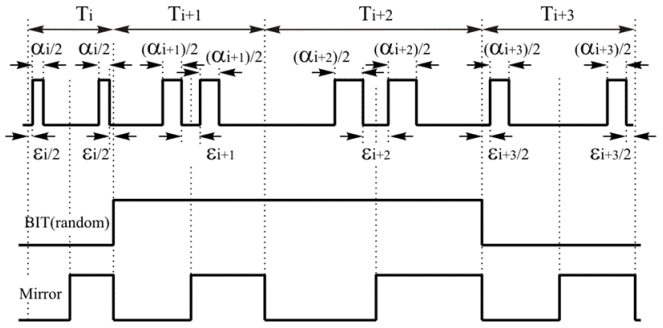

Fig. 17. RPPM1 strategy with symmetry, variable T_i, proportional α_i and ϵ_i.

Table V shows the configuration register values to obtain the described behaviours.

E. Effect of load on the Spread Spectrum

Next, the ability of the FTB to relate spread spectrum techniques to load demands will be explored, assuming that the SPWM participates in a closed-loop system where the power delivered to the load modulates the pulse width.

It is assumed that the pulse width variations (α_i) due to the response of a closed-loop regulation system produce Spread Spectrum that overlaps with the effect of the SPWM.

The block in Fig. 7, called *Modulation's ROM*, stores a DDS waveform representing the effect of an arbitrary load connected to a power control PWM. By activating *SW11* in any of the modes, the effects of the load on any of the SPWM strategies can be studied.

In applications where these variations are continuous and sufficiently intense, these can also be used to vary the PWM characteristic times (T_i, α_i, ϵ_i, θ_i) to achieve the "Spread Spectrum" effect, which is why the possibility of accessing the *Modulation's ROM* from CTSSB's MUX1 is added (Fig. 6 and Table III), which allows the load to also influence the parameters T_i, α_i, ϵ_i and θ_i.

Combining load effects with different SPWM options inspires new strategies that in this paper will be called Load Influenced SPWM (LISPWM). Table VI shows 4 variations of LISPWM compared to a standard PWM.

TABLE V
ALTERNATIVE CONFIGURATION REGISTER FOR SPWM

Configuration Register bits															Modified RPPM1 versions
14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
-	-	-	0	-	-	-	0	0	0	0	1	0	0	0	Fig. 12
-	-	-	0	-	-	-	0	0	0	1	1	0	0	0	Fig. 13
-	-	-	0	-	-	-	0	0	0	1	1	1	0	0	Fig. 14
-	-	-	0	X	X	X	0	0	0	1	1	1	0	1	Fig. 15
-	-	-	0	X	X	X	0	0	0	1	1	1	1	1	Fig. 16
-	-	-	0	X	X	X	0	0	1	1	1	1	1	1	Fig. 17

TABLE VI
CONFIGURATION TO LOAD INFLUENCED SPWM

Configuration Register bits															SPWM mode
14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	1	-	-	-	0	0	0	0	0	0	0	0	PWM standard
0	0	0	1	1	1	1	0	0	0	0	0	0	1	1	RCFMFD-M
0	0	0	1	1	1	1	1	0	0	0	0	0	1	1	RCFMFD-AM
0	0	0	1	1	1	1	1	0	0	0	0	1	0	1	RCFMFD-SAM
0	0	0	1	1	1	1	1	0	0	1	1	0	1	1	RCFMFD-DSAM

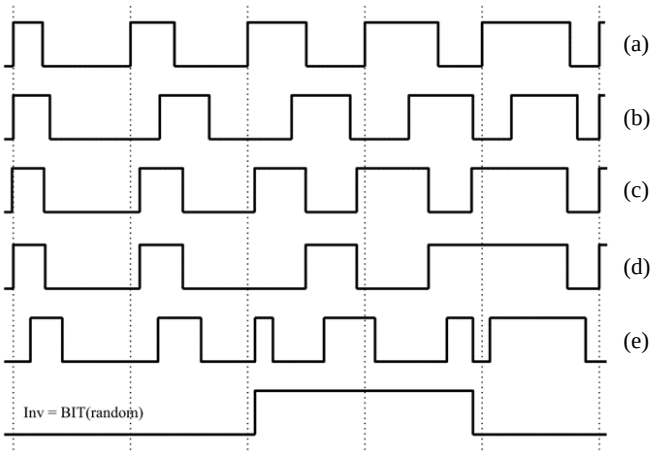

Fig. 18. LISPWM strategies derived from RCFMFD and compared to PWM.

For comparison purposes, the timing diagram of a standard PWM (Fig. 18(a)) modulating the pulse α_i with constant T_i (*SW11*=1) with the Modulation's ROM function is included.

RCFMFD-M is an option of RCFMFD for which Modulation's ROM is selected as the cycle variation function (*SW10/9/8*="111"). This makes the function modulating the PWM pulse and *Fssk* the same (Fig. 18(b)).

The next mode in Table VI, called RCFMFD-AM, applies the absolute value to Fssk (*SW7*=1) to avoid high-frequency harmonics (Fig. 18(c)).

Both of these options are useful in DC-AC converters where the pulse modulation is pronounced. However, in converters with low or no load variations, the Spread Spectrum may be insufficient. In this case, a random inversion of the PWM cycle can be applied (*SW3*=1), which ensures that, if there is no pulse modulation, the behaviour will be like RPPM1 in Table V (Fig. 12). The resulting waveform is shown in Fig. 18(d) and will be called RCFMFD-SAM in this paper.

Finally, if it is desired to prevent the cycle inversion from merging adjacent pulses, a delay ϵ_i can be added and the signal shown in Fig. 18(e) (RCFMFD-DSAM) will be obtained.

From here, it is possible to continue with the alternatives used in the previous example and summarised in Table V.

These examples demonstrate the flexibility of the FTB to introduce and test SPWM variants and show that there are still many more to be explored.

IV. TECHNOLOGICAL AND IMPLEMENTATIONS ISSUES

In the implementation of the FTB, it has been proposed to minimise the technological demands on the FPGA, precisely because it is aimed at compact, low-cost systems. Indeed, the FTB could be easily implemented in the XC3S200-4FT256C FPGA of the Spartan-3 family [25], achieving a higher performance in frequency and resolution than required by most of the targeted applications.

Once a modulation strategy has been chosen, it is necessary to know how much hardware resources will be required to implement it in isolation from the FTB, in order to assess the complexity of its large-scale, low-cost production.

The VHDL description of the FTB allows all configuration parameters defining each strategy to be set as constants and the user interface and unused configuration registers to be removed. In this way, the resulting VHDL code will result only in the circuit of the selected strategy, eliminating all the components that are not used in it.

Table VII shows the FPGA resources required for each of the 16 strategies shown in this study. The first row of Table VII also includes the summary of resources required for the complete FTB and the second row includes the resources required for the configuration and operation interface. This data is used to assess the resource consumption of each part of the FTB.

This analysis again highlights the advantages of the FTB for comparing different SPWM strategies, because the in-silicon tests have been done from a single VHDL source code, with the same FPGA technology, and at the same clock frequency.

In order to give more generality to the results, the use of functional blocks integrated in the FPGA, such as multipliers

TABLE VII
FPGA'S OCCUPANCY FACTOR AND *Fclk* OF FTB.

	Slice Flip Flops	**4 input LUTs**	**Occupied Slices**	***Fclk* [MHz]**
Full FTB	700	4797	2538	29
Interface Only	132	303	176	105
PWM Standard	138	572	331	68
RPPM (RDN)	272	2509	1342	33
RPWM (RDN)	816	2247	1426	51
RCFMFD (RDN)	816	2247	1488	51
RCFMVD (RDN)	816	2238	1446	51
RDRPPMFCF (RDN)	271	2518	1353	33
RCFRPPMFD (RDN)	272	2509	1359	33
RRRM (RDN)	269	2518	1394	33
RPPM-S	164	621	368	68
RPPM-T	164	611	358	68
RPPM-DS	208	845	475	33
RPPM-DT	213	844	480	33
RCFMFD-M (MOD)	460	1562	886	46
RCFMFD-AM (MOD)	476	1617	922	42
RPPM-SAM (MOD)	482	1642	912	42
RPPM-DSAM (MOD)	504	1843	1036	33

TABLE VIII
FPGA'S OCCUPANCY FACTOR AND *Fclk* OF RCFMFD FOR DIFFERENT DDS FUNCTIONS.

	Slice Flip Flops	**4 input LUTs**	**Occupied Slices**	***Fclk* [MHz]**
RCFMFD (RDN)	816	2247	1488	51
RCFMFD (SIN)	460	1562	886	46
RCFMFD (TRI)	250	1083	612	56
RCFMFD (EXP)	446	1527	867	50
RCFMFD (CAO)	858	2204	1271	44
RCFMFD (RDNSHF)	190	1006	576	55

or RAMBlock, typical of the Xilinx Spartan family, has been avoided. Instead, multipliers implemented with CLBs and distributed RAM have been used.

Implementation control, by means of a constant configuration register, adds a criterion for evaluating the suitability of a given strategy based on the hardware resources required and the maximum clock frequency allowed.

To emphasise the usefulness of the tool, a comparison of the logical resource occupancy of the same strategy (RCFMFD) for different digitally synthesised spread spectrum functions (DDS) is shown in Table VIII.

Table VIII also shows that, although the memory size storing the DDS functions is the same for all (2K x 9), as these are implemented with FPGA LUTs, the number of LUTs required depends on the data content. Indeed, it can be clearly seen that the Random and Chaotics functions require the most LUTs because there are no regular patterns that facilitate the simplification and thus the reduction of LUTs.

Finally, the FPGA resource occupation can be further reduced by removing the configuration and operation interface and leaving the 5 operating parameters constant: $Tcyc$, $\Delta Tcyc$, *Offset*, *Mod Depth* and *Delay Depth*. This is useful because, once an application is defined, the final circuit that will be reproduced at the industrial level will have a fixed behaviour, without the flexible resources of the FTB.

Table IX shows the considerable reduction in FPGA occupancy for each strategy in Table VIII.

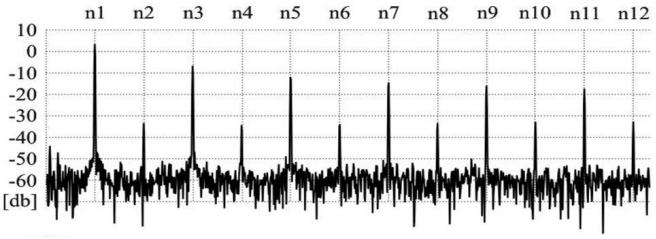

Fig. 19. Spectral diagram of standard PWM with a fixed pulse (50%).

V. SOFT AND WORKING PROTOTYPES

This section will show how the FTB highlights features of different SPWM strategies while allowing new ones to be explored. The validity of the comparisons is based on the fact that all strategies are implemented on the same hardware platform.

The FTB is a rapid prototyping tool, allowing an interactive design by observing different optimisation criteria such as:

1) Harmonic amplitude reduction.
2) Harmonic distribution.
3) Hardware resource occupation.
4) Maximum clock frequency.

To evaluate the reduction and distribution of harmonics, the spectral diagram of a standard PWM with a duty cycle of 50% shown in Fig. 19 will be taken as a reference. In the following subsections, the effect on the spectrum of the strategy used will be studied and compared.

TABLE IX
FPGA'S OCCUPANCY FACTOR AND *Fclk* OF RCFMFD WITHOUT USER INTERFACE.

	Slice Flip Flops	4 input LUTs	Occupied Slices	*Fclk* [MHz]
RCFMFD (RDN)	310	1514	804	80
RCFMFD (SIN)	302	865	485	88
RCFMFD (TRI)	170	351	210	84
RCFMFD (EXP)	157	798	451	83
RCFMFD (CAO)	287	1577	858	76
RCFMFD (RDNSHF)	196	249	153	82

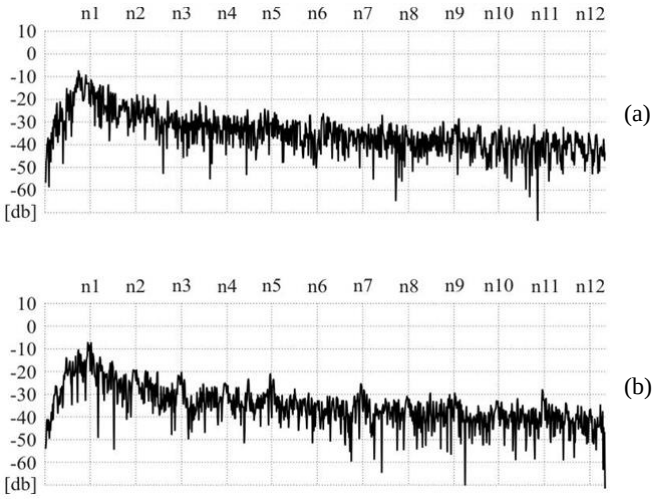

Fig. 20. RCFMFD using two different pseudo-random sources: (a) RDN; (b) RDNSHF.

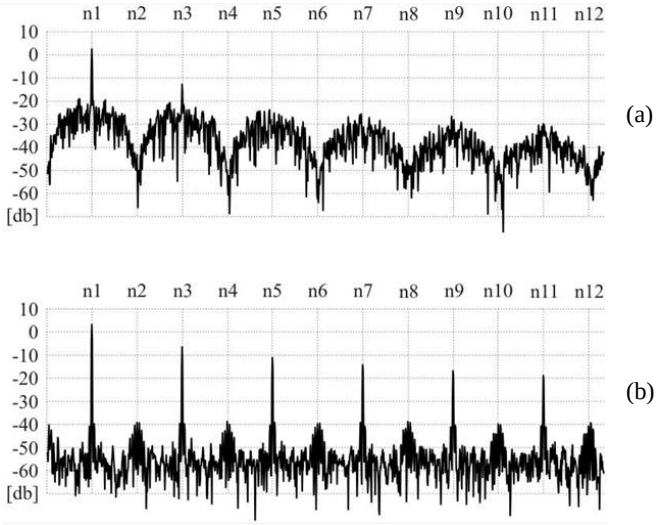

Fig. 21. RPPM using two different pseudo-random sources: (a) RDN; (b) RDNSHF.

A. *Effect of pseudorandom sequences*

It has been observed that the pseudo-random sequences used for SPWM do not have the same effect with different strategies. For example, the FTB of this work offers two types of pseudo-random sequences, one using DDS with a random function obtained with a mixed congruential generator (RDN), and the other is generated with a feedback shift register (RDNSHF).

Fig. 20 shows the spectral distribution of the RCFMFD strategy for both pseudo-random sources, (a) for RDN and (b) for RDNSHF, and both show a similar response.

As the hardware occupancy of the RCFMFD (RDN) row in Table IX (corresponding to Fig. 20(a)) is considerably higher than the RCFMFD (RDNSHF) row (corresponding to Fig. 20(b)), it justifies the adoption of RCFMFD (RDNSHF) as a better implementation of RCFMFD.

Fig. 21 shows the same for RPPM. In this case, the response is clearly different from the previous one. The RDN source (Fig. 21(a)) achieves an acceptable result, but, on the other hand, the RDNSHF source (Fig. 21(b)) shows a poor reduction for the even harmonics and practically null for the odd harmonics with respect to a standard PWM.

To reinforce the validity of the pseudo-random sources, these are compared in a third strategy (RCFMVD), giving the result shown in Fig. 22.

It can be seen that for RCFMVD, like RCFMFD, there is virtually no difference between the choice of RND or RND-SHF, which leaves open the question of why with RPPM there is such a difference (Fig. 21), especially since all strategies are implemented with the same functional blocks and with the same pseudo-random sources, thereby supporting the validity of the test.

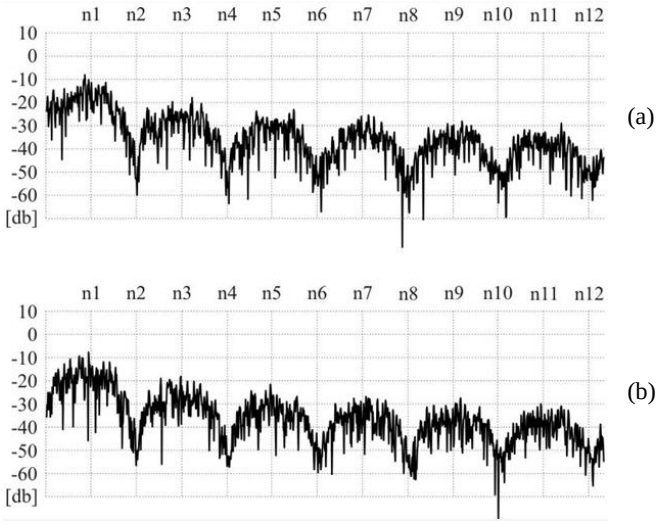

Fig. 22. RCFMVD using two different pseudo-random sources: (a) RDN; (b) RDNSHF.

The answer lies in a more exhaustive analysis of the pseudorandom sources. Fig. 23 shows the time evolution of both

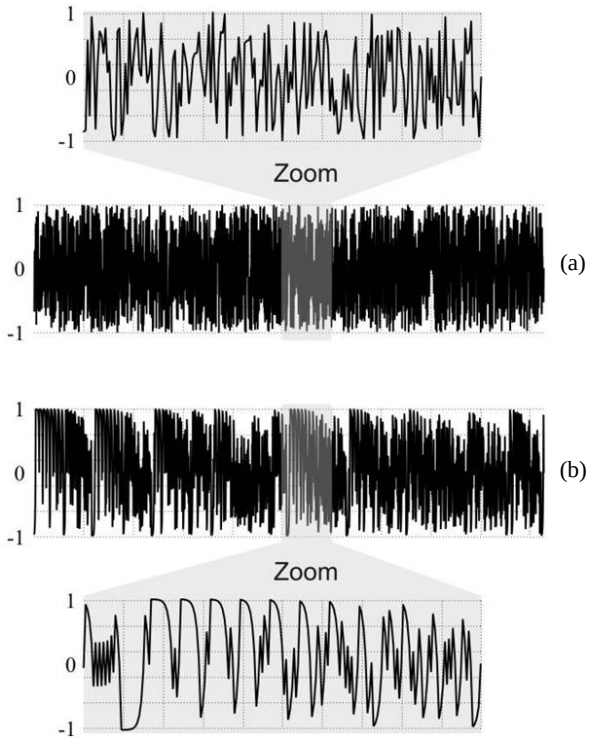

Fig. 23. Time evolution of: (a) RDN; (b) RDNSHF.

sources, with the stored RDN waveform in Fig. 23(a) and the random RDNSHF sequence in Fig. 23(b). Each figure shows a detailed zoom in which the periodicity of the random sequence can be analysed. In the figure corresponding to the RDNSHF source, a certain regular periodicity can be appreciated, unlike the RDN detail which is left with a rather irregular random appearance.

A very important aspect to take into account when comparing the spectral response is the conditions under which the Fast Fourier Transform (FFT) is applied or how the window and sampling of the spectrum analyser are adjusted. In this study, a relatively small number of PWM cycles have been taken to calculate the FFT (100 PWM cycles), which has highlighted the low quality of the RDNSHF source in Fig. 21 and has also reveals that the quality of the RDN source does not influence the different strategies in the same way.

A larger number of PWM cycles for the FFT calculation disguises the effect, as more PWM period variations are included and, when averaged, show an apparent reduction of harmonics.

B. Comparison and choice of SPWM strategies

In this section, some criteria for comparison and selection of SPWM strategies will be developed from the data obtained with the FTB described in this study. As an example, RCFMFD and RCFMVD are compared for both RDN and RDNSHF pseudo-random sources, whose spectral responses are shown in Fig. 20 and Fig. 22. At first glance, either option improves a standard PWM, since the harmonic amplitude remains at least 12 DB below the standard PWM peaks (Fig. 19).

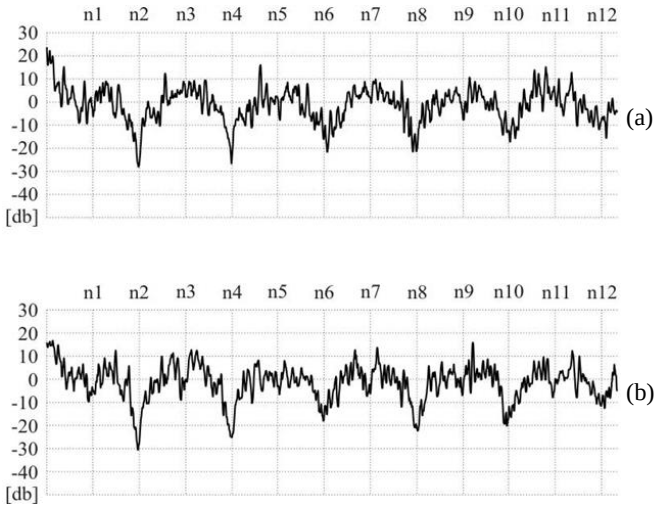

Fig. 24. Difference of RCFMFD and RCFMVD spectra using two different pseudo-random sources: (a) RDN; (b) RDNSHF.

It is also observed that the harmonic amplitude for RCFMFD is quite uniform with a progressive reduction towards the high frequencies of about 20 DB (Fig. 20). RCFMVD (Fig. 22), on the other hand, presents a similar envelope, but with very pronounced attenuation for the even harmonics, about 30 DB lower than RCFMFD.

As a criterion to objectively compare both strategies, the harmonic amplitude difference RCFMVD - RCFMFD is calculated and plotted in Fig. 24. To better appreciate the trend, a median filter is applied to smooth the variation of neighbouring harmonics.

The RCFMVD - RCFMFD difference shows that RCFMVD has a 30 DB attenuation with respect to RCFMFD in a narrow band around the even harmonics, but the rest of the frequencies remain about 6 DB above RCFMFD. This result is consistent with the idea that the total energy of the whole frequency spectrum remains constant, because SPWM only distributes the energy at different frequencies, but does not reduce the total energy. Therefore, a reduction in certain bands is accompanied by an increase in other bands, offseting the total energy.

Finally, to add another criterion for comparison, the hardware occupancy and maximum clock frequency are calculated, without the FTB user interface (Table X). Since for RCFMFD and RCFMVD, the spectral response is similar using either of the two pseudo-random sources, RDNSHF is chosen because it is much more economical in hardware resources than the RDN stored waveform.

VI. Conclusions

In this work, a Flexible Test Bench (FTB) to be implemented on any FPGA has been presented, which allows the synthesis of a large number of strategies applying "Spread Spectrum" to reduce the energy of the fundamental harmonics present in a conventional PWM. The Xilinx XC3S200-4FT256C FPGA, which has a small number of logic resources, has been used for this purpose. An implementation of each

TABLE X
FPGA'S OCCUPANCY FACTOR AND *Fclk* OF RCFMFD AND RCFMVD.

	Slice Flip Flops	**4 input LUTs**	**Occupied Slices**	***Fclk* [MHz]**
RCFMFD (RDN)	310	1514	804	80
RCFMFD (RDNSHF)	196	249	153	82
RCFMVD (RDN)	347	1348	759	106
RCFMVD (RDNSHF)	69	90	68	106

SPWM mode has been performed using only LUTs to determine the number of resources required to select the FPGA needed to implement the entire bank or a single SPWM mode.

Low-cost, low-power FPGAs are now available that could be used in both full test bench and single-mode implementations. Furthermore, it should be noted that some families of these FPGAs include a memory, called NVCM, which is non-volatile and contains the configuration data of the device. This is very important if a system to be manufactured is to be implemented since it is not necessary to add an external Flash EPROM and, moreover, it protects against copying of the design by other competitors.

The use of fixed-point numerical representation, adjusting the dynamic range of the operands, allows complex algorithms to be tested with an architecture that can then be easily implemented in VLSI without the need to incorporate processors.

In the examples developed, it has been demonstrated that the FTB is not only able to reproduce a wide range of SPWM strategies, but also to combine them to generate new strategies, with an architecture that allows a genuine harmonic reduction comparison. On the other hand, the influence of the load has also been simulated by including different types of modulation by means of DDS. This can also be used to vary the PWM timing parameters and to create new Spread Spectrum modes, which have been named LISPWM.

It also offers a simple tool to evaluate the hardware resources required for its implementation and the maximum frequency that can be achieved in each of the SPWM modes with a given technology. This feature can be very useful for testing prototype circuits that will eventually be implemented in VLSI. In this regard, a comparison has been made with respect to the pseudo-random sequence used for the SPWM with respect to both the attenuation of harmonics and the number of LUTs required for its implementation. The results obtained indicate that the generation of the pseudo-random sequence using an LFSR requires a lower number of LUTs compared to the DDS, but the attenuation of harmonics is equal or lower depending on the SPWM mode used. Therefore, the test bench can be used to determine the most suitable SPWM mode and pseudo-randomisation sequence, depending on the specifications required.

As a final conclusion, it can be stated that a low-cost platform has been obtained that allows one to reproduce, design, and compare new ElectroMagnetic Compatibility (EMC)-compliant control techniques for power electronics.

REFERENCES

[1] CENELEC, "Study report on electromagnetic interference between electrical equipment/systems in the frequency range below 150 kHz," CLC/TR 50627, Tech. Rep., 2015.

[2] K. Mainali and R. Oruganti, "Conducted EMI mitigation techniques for switch-mode power converters: A survey," *IEEE Transactions on Power Electronics*, vol. 25, no. 9, pp. 2344–2356, 2010.

[3] J. Huang and H. Shi, "Suppression of the peak harmonics from loads by using a variable capacitance filter in low-voltage DC/DC converters," *IEEE Transactions on Electromagnetic Compatibility*, vol. 58, no. 4, pp. 1217–1227, 2016.

[4] R. Gamoudi, D. E. Chariag, and L. Sbita, "A review of spread-spectrum-based PWM techniques—A novel fast digital implementation," *IEEE Transactions on Power Electronics*, vol. 33, no. 12, pp. 10 292–10 307, 2018.

[5] R. Wang, Z. Lin, J. Du, J. Wu, and X. He, "Direct sequence spread spectrum-based PWM strategy for harmonic reduction and communication," *IEEE Transactions on Power Electronics*, vol. 32, no. 6, pp. 4455–4465, 2016.

[6] M. H. Yanuar, R. Hidayat, and E. Firmansyah, "An experimental study of conducted EMI mitigation on the LED driver using spread spectrum technique," *International Journal of Electronics and Telecommunications*, vol. 62, no. 3, pp. 293–299, 2016.

[7] M. Stecher, "Possible effects of spread-spectrum-clock interference on wideband radiocommunication services," in *2005 International Symposium on Electromagnetic Compatibility (EMC'2005).*, vol. 1. IEEE, 2005, pp. 60–63.

[8] P. S. Crovetti and F. Musolino, "Interference of spread-spectrum EMI and digital data links under narrowband resonant coupling," *Electronics*, vol. 9, no. 1, p. 60, 2020.

[9] F. Musolino and P. S. Crovetti, "Interference of spread-spectrum modulated disturbances on digital communication channels," *IEEE Access*, vol. 7, pp. 158 969–158 980, 2019.

[10] H. G. Skinner and K. P. Slattery, "Why spread spectrum clocking of computing devices is not cheating," in *2001 IEEE EMC International Symposium. Symposium Record. International Symposium on Electromagnetic Compatibility (Cat. No. 01CH37161)*, vol. 1. IEEE, 2001, pp. 537–540.

[11] M. Luna, M. Pucci, G. Vitale, and G. Abbate, "FPGA-based random PWM control of a buck DC/DC converter," *Journal of Electrical Systems*, vol. 6, no. 3, pp. 323–338, 2010.

[12] S. Ustun Ercan, A. Pena-Quintal, and D. Thomas, "The effect of spread spectrum modulation on power line communications," *Energies*, vol. 16, no. 13, p. 5197, 2023.

[13] J.-H. Kim and Y.-G. Jung, "Spreading power spectrum of an induction motor drive system by chaotic pulse width modulation method," *Journal of Electrical Engineering & Technology*, vol. 16, no. 5, pp. 2685–2694, 2021.

[14] A. Fratta, G. Griffero, and S. Nieddu, "Comparative analysis among dsp and fpga-based control capabilities in pwm power converters," in *30th Annual Conference of IEEE Industrial Electronics Society*, vol. 1. IEEE, 2004, pp. 257–262.

[15] A. Al-Safi, A. Al-Khayyat, A. M. Manati, and L. Alhafadhi, "Advances in FPGA based PWM generation for power electronics applications: Literature review," in *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON'2020)*. IEEE, 2020, pp. 0252–0259.

[16] V. Stephen and L. PadmaSuresh, "Experimental evaluation of a random frequency PWM inverter scheme," in *2013 International Conference on Circuits, Power and Computing Technologies (ICCPCT'2013)*. IEEE, 2013, pp. 611–617.

[17] S. Berto, S. Bolognani, M. Ceschia, A. Paccagnella, and M. Zigliotto, "FPGA-based random PWM with real-time dead time compensation," in *IEEE 34th Annual Conference on Power Electronics Specialist (PESC'03)*, vol. 2. IEEE, 2003, pp. 513–518.

[18] S.-L. Jung, M.-Y. Chang, J.-Y. Jyang, H.-S. Huang, L.-C. Yeh, and Y.-Y. Tzou, "Design and implementation of an FPGA-based control IC for the single-phase PWM inverter used in an UPS," in *Proceedings*

of Second International Conference on Power Electronics and Drive Systems, vol. 1. IEEE, 1997, pp. 344–349.

[19] A. Rajalakshmi and A. Kavitha, "Implementation of low cost FPGA based digital modulation techniques for the suppression of EMI and improvement of power factor for three-phase grid connected PV inverters," in *2018 IEEE International Conference on Power Electronics, Drives and Energy Systems (PEDES'2018)*. IEEE, 2018, pp. 1–6.

[20] A. Rafique, A. Rehman, and R. Ahmed, "Design and implementation of a 2-level sinusoidal PWM generator with modulation based harmonic elimination," in *2008 IEEE International Multitopic Conference*. IEEE, 2008, pp. 537–541.

[21] H. Loschi, R. Smolenski, P. Lezynski, W. El Sayed, and D. Nascimento, "Reduction of conducted emissions in DC/DC converters with FPGA-based random modulation," in *2020 International Symposium on Electromagnetic Compatibility-EMC EUROPE*. IEEE, 2020, pp. 1–6.

[22] G. M. Dousoky, M. Shoyama, and T. Ninomiya, "FPGA-based design and implementation of spread-spectrum schemes for conducted-noise reduction in DC-DC converters," in *2009 IEEE International Conference on Industrial Technology*. IEEE, 2009, pp. 1–6.

[23] H. Loschi, P. Lezynski, R. Smolenski, D. Nascimento, and W. Sleszynski, "Fpga-based system for electromagnetic interference evaluation in random modulated dc/dc converters," *Energies*, vol. 13, no. 9, p. 2389, 2020.

[24] J. Garrido, A. Moreno-Muñoz, A. Gil-de Castro, V. Pallares-Lopez, and T. Morales-Leal, "Supraharmonics emission from LED lamps: A reduction proposal based on random pulse-width modulation," *Electric power systems research*, vol. 164, pp. 11–19, 2018.

[25] Xilinx/AMD, "FPGA XC3S200-4FT256C - Spartan-3 FPGA family data sheet. Available online," 2013. [Online]. Available: https://docs.xilinx.com/v/u/en-US/ds099