Application Report
SPRACH7–November 2018

Light Load THD and Efficiency Optimization of Digitally Controlled PFC Converter With Integrated Valley Switching Control

Chen Jiang, Hrishikesh Nene, Shamim Choudhury

ABSTRACT

This application report provides a guide for the use of the type-4 ePWM (Devices Covered: 2807x, 2837xD, 2837xS, 28004x) module to implement valley switching. It presents the implementation details of controlling a boost PFC with valley switching using TI's C2000™ microcontrollers. A modified 2-phase interleaved power factor correction (ILPFC) kit is used to validate the software and system operation. Detailed waveforms and test results along with a description of the software are provided. This document provides a further extension of TI design TIDM-1022 valley switching boost PFC.

Contents

List of Figures

List of Tables

Trademarks

1 Introduction

As the EV charging market boosts rapidly, increasing charging efficiency becomes a hot topic for researchers as it can be applied in the trickle charging and fast charging. Compared to the heavy load performance, light load efficiency and THD are more difficult to improve. Valley switching is a soft-switching technique that improves system efficiency of AC-DC and DC-DC converters. Significant efficiency improvements can be achieved with valley switching mainly under low load conditions where efficiency standards are difficult to meet.

A digitally controlled PFC converter for such applications is shown in Figure 1. This interleaved boost PFC converter operates mostly in continuous-conduction mode (CCM) for certain heavy load conditions. At light loads, the converter enters discontinuous mode (DCM) operation and phase shedding mode. Under this condition when the MOSFET turns off and the boost inductor (L) current decreases to zero, the inductor and the PFC MOSFET parasitic capacitance (C) form a resonant tank circuit. The resonant circuit energy then alternates between these two resonant elements until the start of next PWM cycle. Under light load conditions, this resonant current can be significant compared to the converter switching current controlled by PFC current control loop. As a result PFC input current suffers significant distortion, consequently increasing the input current THD.

Valley switching helps simultaneously improve the input current THD and the light load efficiency. The inductor current flows in reverse direction and allows MOSFET voltage to oscillate to a low voltage at the resonant frequency creating ideal condition for valley voltage switching. This valley voltage turn on of the MOSFET improves light load efficiency of the PFC converter. Also, at the valley of the voltage, the inductor current returns back from negative to zero. If the MOSFET is turned on exactly at this point, the effect of resonant current on the inductor switching current is minimized. This improves the light load THD of the input current.

The switching frequency can become very high at light loads, and therefore, it is desirable to limit it in order to decrease the switching losses. The valley skipping is one way that can limit the switching frequency to maintain high efficiency and keep low THD at the same time since the MOSFET is turn on at the next few valleys utilizing the programmable valley technique which does not need complex external logic and sensing circuit.

2 Valley switching

A simplified circuit diagram of a flyback converter is shown in Figure 1. Figure 2 depicts valley switching waveforms for flyback converter. The primary current ramps up while the switch is turned ON. With flyback operation, the stored energy is delivered to the load when the switch is turned OFF. During this time the current starts decreasing. In discontinuous mode of operation this current ramps down all the way to 0. At this time, resonant oscillations can be seen on the switch voltage and current as a result of the circuit inductance and capacitance.

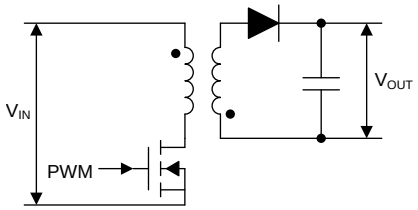

Figure 1. Flyback Circuit

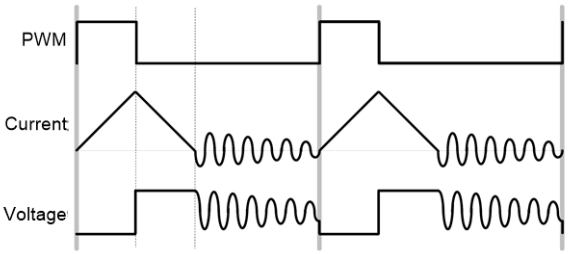

Figure 2. Valley Switching Waveforms

To achieve valley switching, the switch should be turned ON at the valley point of the voltage across it. Under low load operation it may be more efficient to wait a number of oscillation cycles which is also called valley skipping before the switch is turned ON. These resonant oscillations are dependent on circuit parasitic capacitors and are typically of a frequency in the megahertz. This is a great challenge for the controller and/or the control logic. Furthermore, there is a need for programmable number of valleys before which the switch is turned ON. Type-4 PWM modules on C2000 devices provide a way to achieve programmable valley switching without additional external circuit.

3 Type-4 PWM Based Valley Switching Implementation

Figure 3 shows how to detect the valley point based on type-4 PWM. The threshold can be set through a compare (CMPSS) module and the target period will be captured once the designed # of edge is detected. Then, a 1/4 of period delay following the final edge in this situation is added to find the valley point. A software delay can be further added to tune the turn on point under different conditions. It is obvious that the threshold is updated based on the output voltage and the dynamic calculation is implemented to achieve best performance.

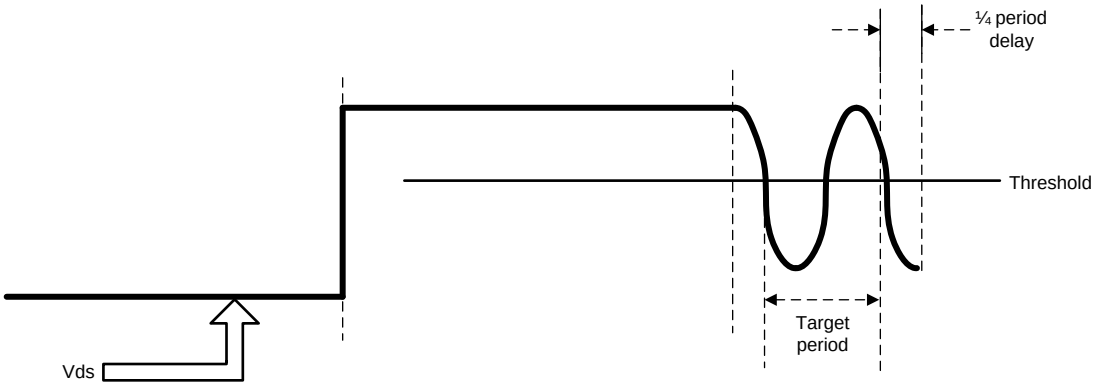

Figure 3. Valley Point Detection

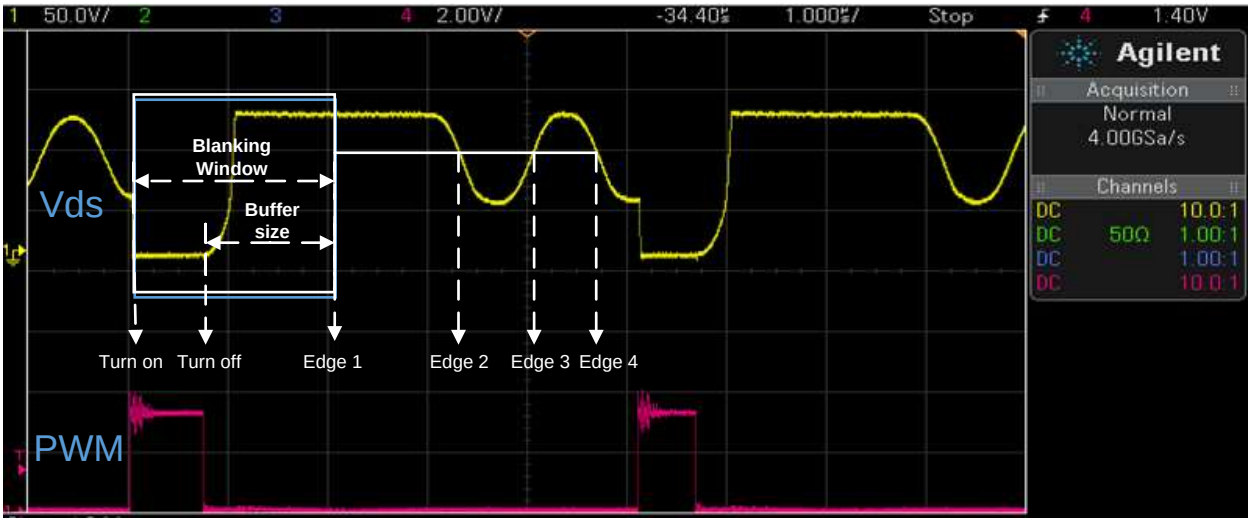

Figure 4. Blanking Window

The drain to source voltage (Vds) and PWM waveform is shown in Figure 4 and a blanking window is adapted to provide more stability and possibility for valley switching. All edges will be ignored inside the blanking window. The blanking window is dynamically updated based on the MOSFET turn on time and the buffer size. It is also worth noting that the first edge will be the end edge of the blanking window.

The most essential objective of using blanking window is to set a highest clamping frequency so that the switching frequency will not go too high causing some damage and decreasing the PFC performance. In addition, it also helps in case there are some noise in the sensed Vds signal and the number of high to low or low to high edges are reached unexpectedly. The blanking window can also prevent some switch on by mistake when there are some distortion in the Vds sensing signal especially during the period when the Vds increases from zero to Vout.

4 PWM Module Configuration

This section focuses on how to utilize the features in type 4 PWM to configure the PWM module to realize valley switching. Valley switching function along with the event filtering logic are described in Figure 5. This function can be used to achieve programmable valley switching without any additional external circuitry. This module provides an on-chip hardware mechanism that can:

- Capture the oscillation period
- Accurately delay the PWM switching instant
- Allow a programmable number of edges before the delay takes effect
- Provide multiple choices of triggers and events
- Allow easy adaptability for optimum performance under changing system/operating conditions

The DCxEVTy signal needs further processing to support valley switching. Here is a brief description of how valley switching function is enabled:

- Select one of the DCxEVTy events as input to the valley switching block (DCFCTL[SRCSEL]) with an option to add the blanking window (Blank Control Logic). This is where the comparator output (or external input) above is selected as an input to the valley switching block.
- Configure the edge filter to capture ‘n’ rising, falling or both edges through the edge selection logic (DCFCTL[EDGEMODE, EDGECOUNT]).
- Select the correct event to reset and restart the edge filter (VCAPCTL[TRIGSEL]). Edge capturing event is triggered or armed by this selected edge.
- Enable valley capture logic (VCAPCTL[VCAPE]).
- Select the start edge that will indicate the start of capture for oscillation period measurement (VCNTCFG[STARTEDGE]). This is where the 16-bit counter starts counting.
- Select the stop edge (VCNTCFG[STOPEDGE]) that will indicate the edge at which the 16-bit counter stops counting. The captured counter value (CNTVAL) provides oscillation period information.
  - The STOPEDGE value must always be greater than STARTEDGE value.
- Configure and apply the captured delay (CNTVAL) to the edge filtered DCxEVTy signal. The CNTVAL value may be applied as is or applied in conjunction with a software programmed value (useful for offset adjustment) (SWVDELVAL) or only a fraction of the delay may be applied with or without SWVDELVAL. This is useful to correctly apply a delay corresponding to the valley point. (VCAPCTL[VDELAYDIV])
- Configure VCAPCTL[EDGEFILTDLYSEL] to apply hardware delay based on the captured value above.

Once the counter is stopped, counter value is copied into CNTVAL register and counter is reset to zero. No further captures are done until the logic is triggered again by occurrence of event selected by VCAPCTL[TRIGSEL]. In this implementation, the software trigger is used as the source for VCAPCTL[TRIGSEL]. Upon occurrence of the trigger event, irrespective of the current status of the counter, the counter is reset and starts counting from zero upon occurrence of the STARTEDGE. Similarly, upon occurrence of the trigger event, the edge filter is reset and starts counting from zero upon occurrence of the STARTEDGE.

Output from the valley switching block (DCEVTFILT) is then used to synchronize the PWM time-base. The process is shown in Figure 5.

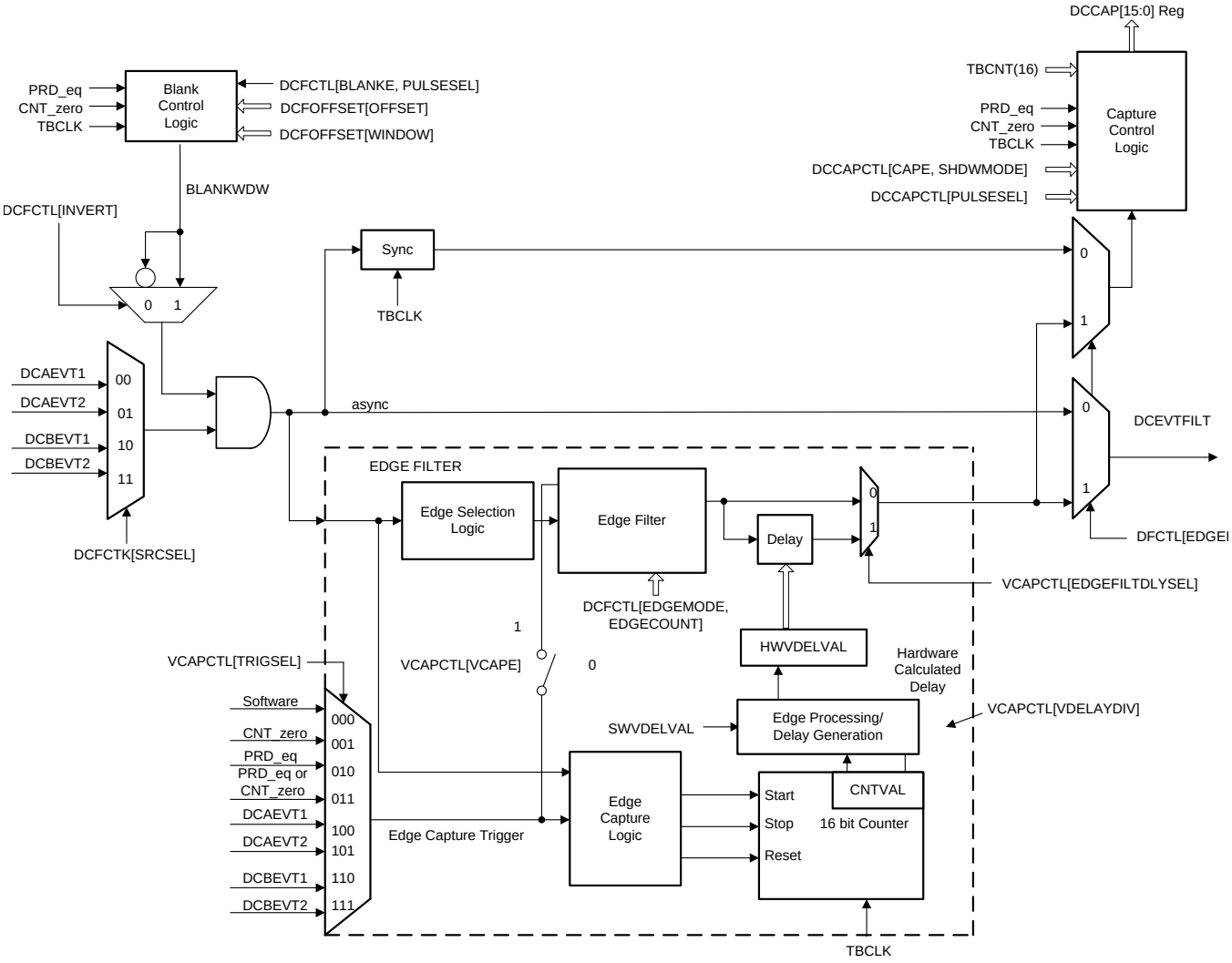

Figure 5. Valley Switching Function Along With the Event Filtering Logic

5 Digital Control Algorithm for Boost PFC

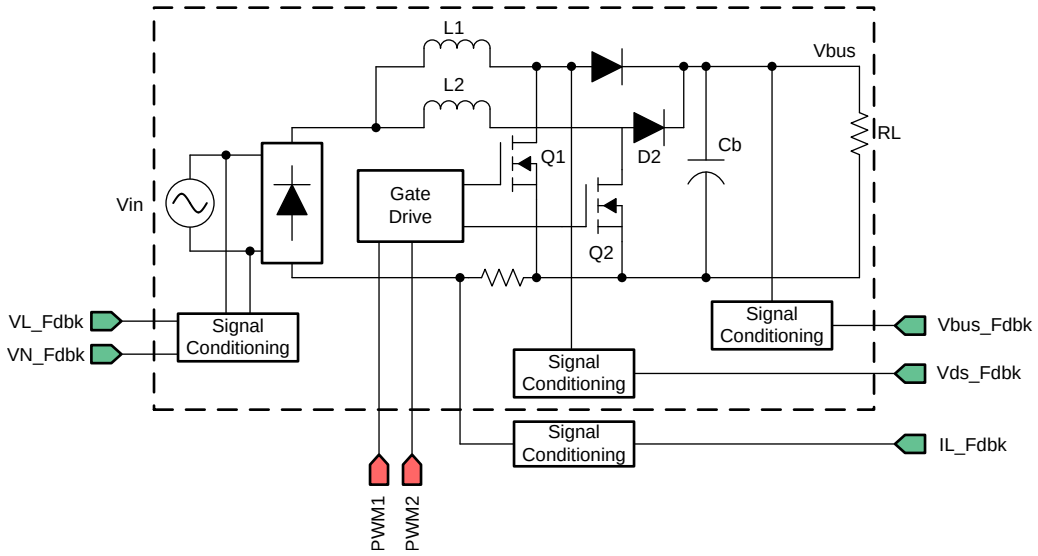

Figure 6. Digital Interleaved PFC Converter

In this part, a detailed digital control algorithm is given that covers from very light load to the heavy load condition for interleaved boost PFC converter as shown in Figure 6. This interleaved boost PFC converter runs with two phases if the load is higher than 10% (75W based on the ILPFC board for our test). Once the load falls below 10%, the phase shedding is triggered and only single phase is working. According to the rectified input voltage waveform in Figure 7, three different control methods are implemented based on the instantaneous input voltage under light load condition to avoid current distortion and increase system efficiency.

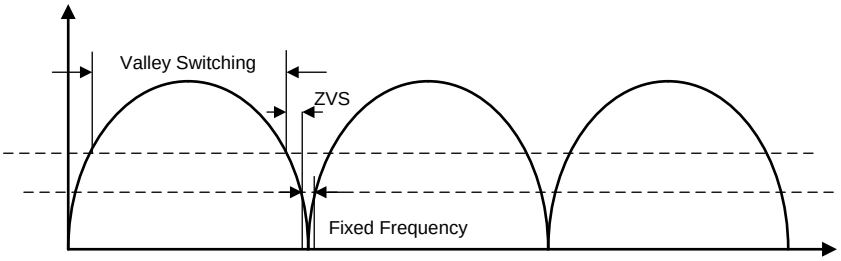

Figure 7. Three Control Methods Based on the Instantaneous Voltage and Loads

5.1 Valley switching

When the instantaneous input voltage is higher than half of the output voltage, the valley switching is enabled. The correct valley position is determined using an on-chip Valley Switching module on the controller. This module captures the oscillation frequency in order to accurately delay the PWM switching instant. It also allows programmable number of edges before the delay takes effect. The MOSFET drain to source voltage is fed to a comparator input on the microcontroller. The other comparator input is driven by an on-chip DAC which is set to a value derived from the converter operating point and represents the voltage level around which resonant oscillations occur. Comparator output changes represent the number of oscillation cycles and their period. This information is used to correctly delay the start of the PWM signal (MOSFET turn ON instant) in the next switching cycle after a programmable number of these oscillation cycles have passed. The blanking window is proposed to provide a more reliable control by eliminating the noise that might accidently trigger the valley switching.

5.2 ZVS

When the instantaneous input voltage is less than half of the output voltage, the ZVS is enabled. Considering the vol-sec balance of the boost inductor under steady state operation, the time TDb that boost inductor current return to zero for the first time can be expressed by the turn-on time TDa , input voltage and output voltage as Equation 1:

$$TDb = TDa \cdot V_{in} / (V_o - V_{in}) \quad (1)$$

If the instantaneous a.c. voltage is lower than than half of the output voltage, Vds of MOSFET's can resonate to zero volts and be clamped by the MOSFET body diode, as shown in Figure 8. During this resonation period when the boost inductor current resonates to a negative value and then second time returns to zero at time tx, the V·S applied to the boost inductor maintains balanced as well. By using V·S balancing(SA = SB + SC) and the simplification method, Tx can be described as Equation 2,

$$T_x = V_o \cdot T_r / (8 \cdot V_{in}) \quad (2)$$

This simplified equation can reduce processor computing time. Based on the Equation 1, Equation 2 and Figure 8, a PWM switching period is predicted as Equation 3,

$$T_s = TDa + TDb + (T_r / 4 + T_x) \cdot 3 \quad (3)$$

However, once the equation is implemented to update the period each cycle inside the digital controller, the impact of the drain to source voltage on the output capacitance cannot be neglected. The output capacitance is actually non-linear vs its drain to source voltage. This problem becomes even worse if the Vds is very small. The output capacitance of a typical MOSFET, such as SPP20N60C3, maintains relatively constant from 600 V down to 50 V, but the value increases by about 10 times at Vds = 25 V and nearly 100 times at Vds = 0 V. The nonlinearity introduces some error to the calculation in terms of the turn on point. However, the Vds is clamped to zero near the calculated switching point that makes the error insignificant.

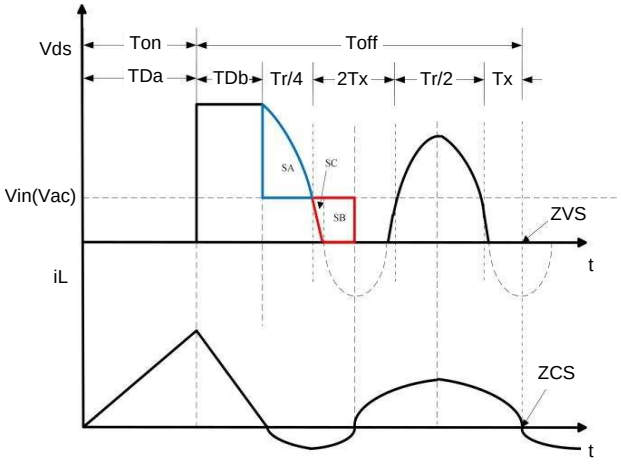

Figure 8. ZVS and Valley Skipping Calculation

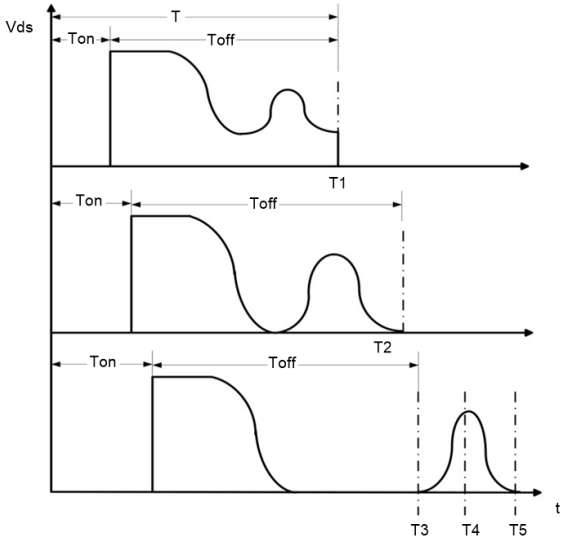

Figure 9. Valley Switching, Threshold Between ZVS and Valley Switching, Fixed Frequency Control (top to bottom)

5.3 Fixed frequency control

When the instantaneous input voltage is close to zero, the calculated switching frequency (turn on time T5) decreases significantly if the same ZVS calculation is used. In addition, considering the calculation error caused by the MOSFET output capacitance as mentioned in Section 5 (ZVS part), the real calculation based turn on point can occur once Vds increases from zero as (point T4) shown in Figure 9. In order to overcome this drawback, the fixed frequency control is selected close to the zero crossing point of instantaneous input voltage. The ideal turn on point T3 skips the peak of the Vds. The minor frequency jitter can be largely eliminated if the fixed frequency is well selected to make a seamless transition. So the current distortion near the zero crossing point of the instantaneous input voltage can be decreased.

6 Sample Code

The sample code below shows how to configure the EPWM module for valley switching, ZVS and fixed frequency control in the ISR code.

```
//valley switching code
if (valley_switching_MODE == valley_switching_enable)
{
    // Select Valley switching or ZVS based on the EPwm7Regs.TZFLG
    // valley switching mode
    if (EPWM_getTripZoneFlagStatus(EPWM7_BASE) == 0x0000 ||
        forceValleyFlag == 1)
    {
        GPIO_setPortPins(GPIO_PORT_A, 0x00008000);
        //need to re-initialize TBPRD = max_period
        EPWM_setTimeBasePeriod(EPWM1_BASE, max_period);
        EPWM_setCounterCompareValue(EPWM1_BASE, EPWM_COUNTER_COMPARE_C,
                                    max_period);

        //Enable valley switching again if previous cycle is based on ZVS
        EPWM_enablePhaseShiftLoad(EPWM1_BASE);
        EPWM_enableValleyCapture(EPWM1_BASE);
        EPWM_startValleyCapture(EPWM1_BASE);

        //DCCAP
        capturedPeriod = EPWM_getDigitalCompareCaptureCount(EPWM1_BASE);
        if (capturedPeriod < min_period)
        {
            EPWM_setCounterCompareValue(EPWM1_BASE, EPWM_COUNTER_COMPARE_C,
                                        min_period);
        }
        else if (capturedPeriod > max_period)
        {
            EPWM_setCounterCompareValue(EPWM1_BASE, EPWM_COUNTER_COMPARE_C,
                                        max_period);
        }
        else if (capturedPeriod < max_period && capturedPeriod > min_period)
        {
            EPWM_setCounterCompareValue(EPWM1_BASE, EPWM_COUNTER_COMPARE_C,
                                        capturedPeriod);
        }

        //set flag signals for mode selection debugging
        modeFlag = 1;

        //update CMPSS threshold
        if (guiVrectRMS >= 160)
        {
            // Update Vds threshold
            vds_Th_Factor = 0.77;
            // Update DAC output
            CMPSS_setDACValueLow(CMPSS5_BASE, (int16_t)(threshold1Highline));
        }
        else
        {
```

```
            // Update Vds threshold
            vds_Th_Factor = 0.73;
            // Update DAC output
            CMPSS_setDACValueLow(CMPSS5_BASE, (int16_t)(threshold1Lowline));
        }

        //add condition code to do hysteresis between ZVS and valley switching
        //WAIT_FOR_ZVS
        if (quadrant_flag == 1)
        {
            forceValleyFlag = 0;
            XBAR_enableEPWMMux(XBAR_TRIP8, XBAR_MUX09);
        }
        //FORCE_VALLEY
        else
        {
            forceValleyFlag = 1;
            XBAR_disableEPWMMux(XBAR_TRIP8, XBAR_MUX09);

            //clear trip zone flag
            EPWM_clearTripZoneFlag(EPWM7_BASE, EPWM_TZ_FLAG_DCAEVT2);
        }

    }
    //ZVS mode
    else
    {
        GPIO_clearPortPins (GPIO_PORT_A, 0x00008000);

        //disable valley switching, using TBPRD to update period
        EPWM_disablePhaseShiftLoad(EPWM1_BASE);
        EPWM_disableValleyCapture(EPWM1_BASE);

        //set flag signals for mode selection debugging
        modeFlag = 2;

        //update CMPSS threshold
        if (guiVrectRMS >= 160)
        {
            // Update DAC output
            CMPSS_setDACValueLow(CMPSS5_BASE, (int16_t)(threshold2Highline));
        }
        else
        {
            // Update DAC output
            CMPSS_setDACValueLow(CMPSS5_BASE, (int16_t)(threshold2Lowline));
        }

        //high line condition
        if (guiVrectRMS >= 160)
        {
            fix_freq = 400;

            EPWM_setTimeBasePeriod(EPWM1_BASE, fix_freq);

            if (quadrant_flag == 2)
            {
                //clear trip zone flag
                EPWM_clearTripZoneFlag(EPWM7_BASE, EPWM_TZ_FLAG_DCAEVT2);
            }
        }
        //low line condition
        else
        {
            fix_freq = 550;
```

```
            if (ac_vol_sensed < fix_freq_th)
            {
                EPWM_setTimeBasePeriod(EPWM1_BASE, fix_freq);
            }
            else
            {
                ePwm1Cmpc = EPWM_getCounterCompareValue(EPWM1_BASE,
                                                        EPWM_COUNTER_COMPARE_C);

                //TDa = dutyPU*ePwm1Cmpc
                zvs_T1 = dutyPU*ePwm1Cmpc*zvs_T_coeff1;
                zvs_Ttotal = zvs_T1 + zvs_T2;

                EPWM_setCounterCompareValue(EPWM1_BASE, EPWM_COUNTER_COMPARE_C,
                                            zvs_Ttotal);

                //below is to clamp PWM period
                if (zvs_Ttotal < min_period)
                {
                    EPWM_setCounterCompareValue(EPWM1_BASE,
                                                EPWM_COUNTER_COMPARE_C, min_period);
                }
                else if (zvs_Ttotal > max_period)
                {
                     EPWM_setCounterCompareValue(EPWM1_BASE,
                                                EPWM_COUNTER_COMPARE_C, max_period);
                }
                else if (zvs_Ttotal < max_period && zvs_Ttotal > min_period)
                {
                     EPWM_setCounterCompareValue(EPWM1_BASE,
                                                EPWM_COUNTER_COMPARE_C, zvs_Ttotal);
                }

                ePwm1Cmpc = EPWM_getCounterCompareValue(EPWM1_BASE,
                                                        EPWM_COUNTER_COMPARE_C);
                EPWM_setTimeBasePeriod(EPWM1_BASE, ePwm1Cmpc);

                //add EPWM7 for ZVS and valley switching selection
                EPWM_setTimeBasePeriod(EPWM7_BASE, ePwm1Cmpc);

                //FORCE_ZVS
                if (quadrant_flag == 1)
                {
                    zvs_Resonant_T = 130;
                }
                //CLEAR_FLAG
                else
                {
                    zvs_Resonant_T = 115;

                    //clear trip zone flag
                    EPWM_clearTripZoneFlag(EPWM7_BASE, EPWM_TZ_FLAG_DCAEVT2);
                }
            }
        }
    }

    ePwm1Cmpc = EPWM_getCounterCompareValue(EPWM1_BASE, EPWM_COUNTER_COMPARE_C);

    //EPWM3, EPWM4 sinc with EPWM1 for oversampling
    //EPWM3/4 period = 0.5 * EPWM1 period
    EPWM_setTimeBasePeriod (EPWM3_BASE,  ePwm1Cmpc/2);
    EPWM_setTimeBasePeriod (EPWM4_BASE,  ePwm1Cmpc/2);
    EPWM_setCounterCompareValue(EPWM3_BASE, EPWM_COUNTER_COMPARE_A, ePwm1Cmpc/4);
    EPWM_setCounterCompareValue(EPWM4_BASE, EPWM_COUNTER_COMPARE_A, ePwm1Cmpc/4);
```

```
        // update channel A
        // CMPC is used to load DCCAP or period register value.
        // This allows the clamped freq value to be used for duty calculations.
        // The DCCAP register value represents the resulting PWM period
        // during valley switching
        ePwm1Cmpa = ePwm1Cmpc - ePwm1Cmpc*dutyPU;
        EPWM_setCounterCompareValue(EPWM1_BASE, EPWM_COUNTER_COMPARE_A, ePwm1Cmpa);

        // update channel B
        ePwm1Cmpb = ePwm1Cmpc*dutyPU;
        EPWM_setCounterCompareValue(EPWM1_BASE, EPWM_COUNTER_COMPARE_B, ePwm1Cmpb);

        // update blanking window length
        blankingWindowLength = blankingBuffer+ePwm1Cmpb;
        EPWM_setDigitalCompareWindowLength(EPWM1_BASE, blankingWindowLength);

        // update blanking window length
        EPWM_setDigitalCompareWindowLength(EPWM7_BASE, blankingWindowLength);
    }
```

7 Experimental Results

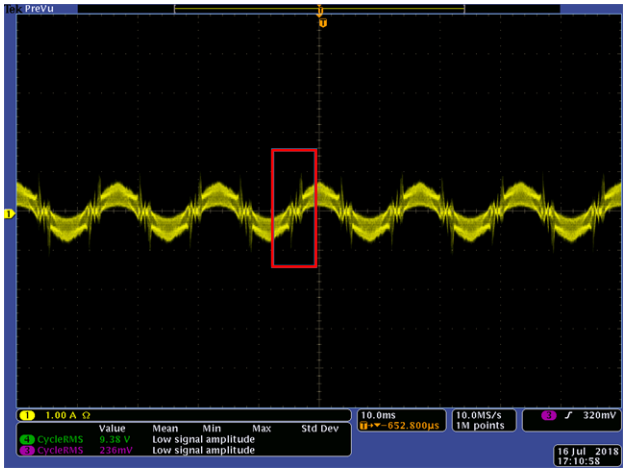

Figure 10. 120 V Input, 380 V Output, 40W Output: High Current Distortion Near Zero Crossing Point Without Optimized ZVS (with valley skipping) and Fixed Frequency Control

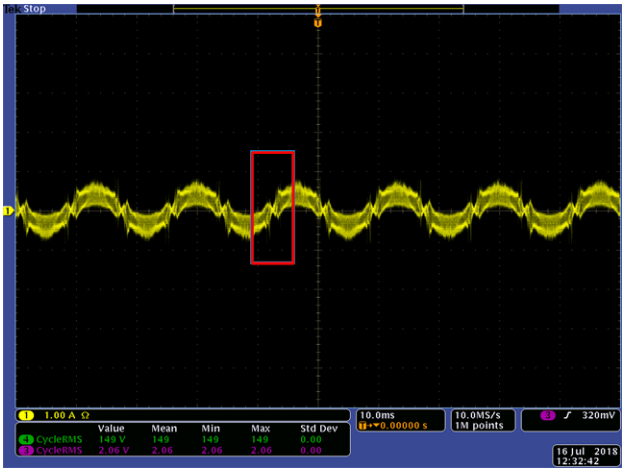

Figure 11. 120 V Input, 380 V Output, 40W Output: Low Current Distortion Near Zero Crossing Point With Optimized ZVS (with valley skipping) and Fixed Frequency Control

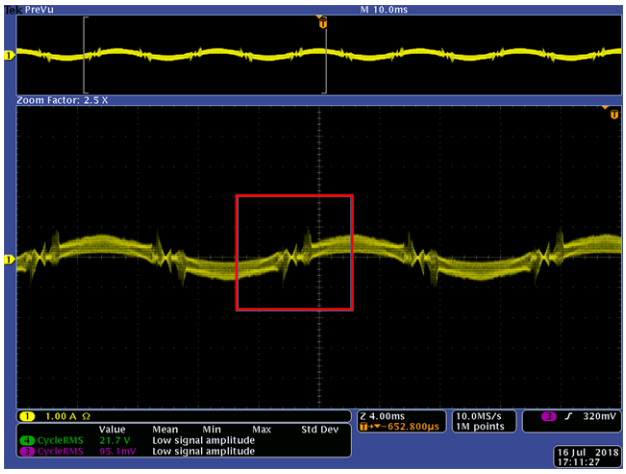

Figure 12. 120 V Input, 380 V Output, 40W Output: Zoom in the Current Distortion of (a)

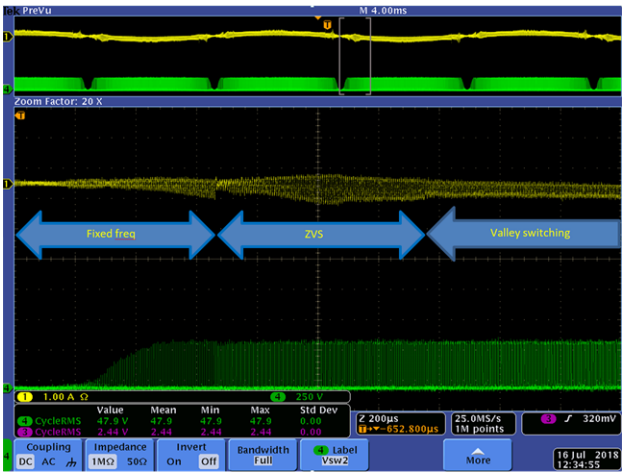

Figure 13. 120 V Input, 380 V Output, 40W Output: Zoom in Input Current Near Zero Crossing to Show the Control Logic

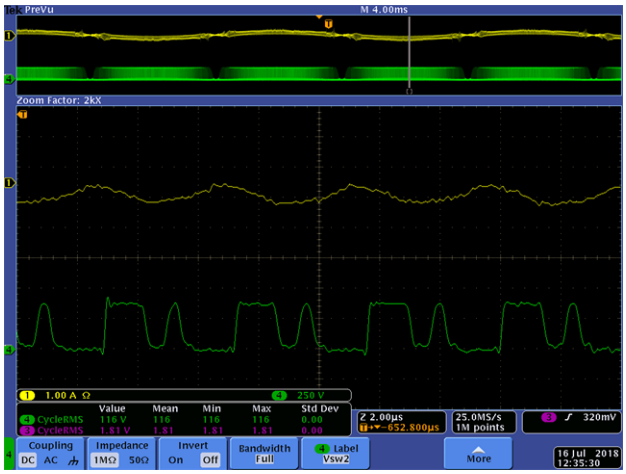

Figure 14. 120 V Input, 380 V Output, 40W Output: Vds Waveform With Valley Switching and Valley Skipping

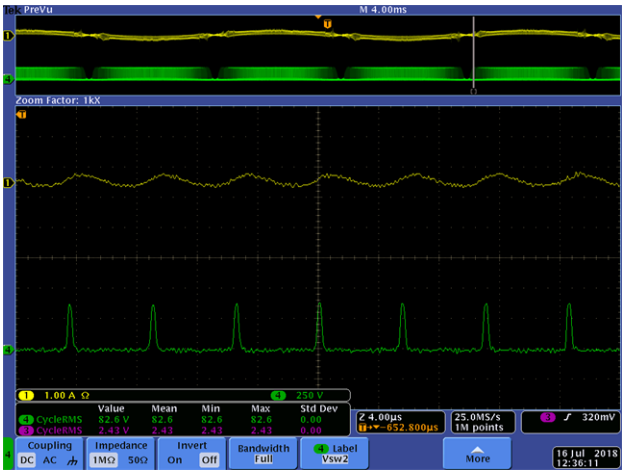

Figure 15. 120 V Input, 380 V Output, 40W Output: Vds Waveform With Fixed Frequency Control

In Figure 10, Figure 11 and Figure 12, some test results are shown to verify the performance improvement after the enhanced control solution is implemented. Figure 16 and Figure 17 show the THD and efficiency under low line condition with traditional constant frequency control (150 kHz), valley switching and presented enhanced control solution. Based on the results, the THD at some point is 15% lower compared to the constant frequency control and is further decreased after adding optimized ZVS and fixed freq control. The THD is close to 5 under 5% load (37.5W/750W). The efficiency and THD are all improved for both high line and low line conditions based on the Figure 18 through Figure 19.

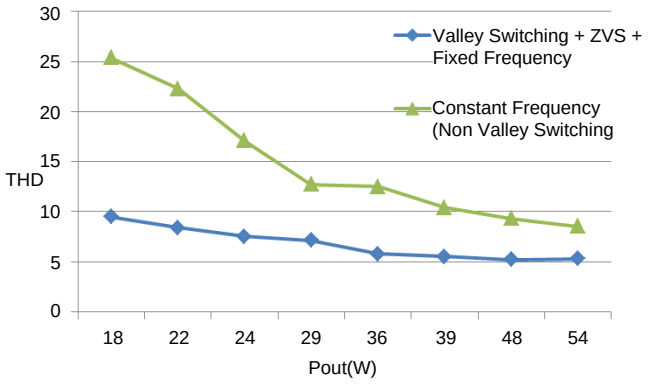

Figure 16. 120 Vin, 380 Vout: THD

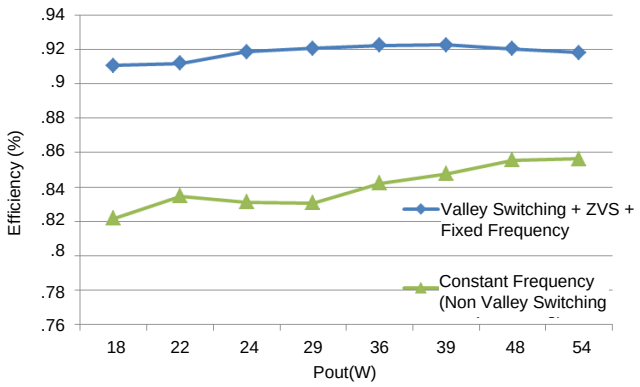

Figure 17. 120 Vin, 380 Vout:Efficiency

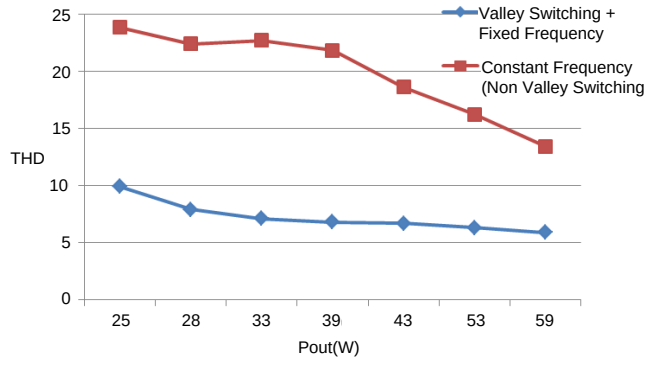

Figure 18. 220 Vin, 380 Vout: THD

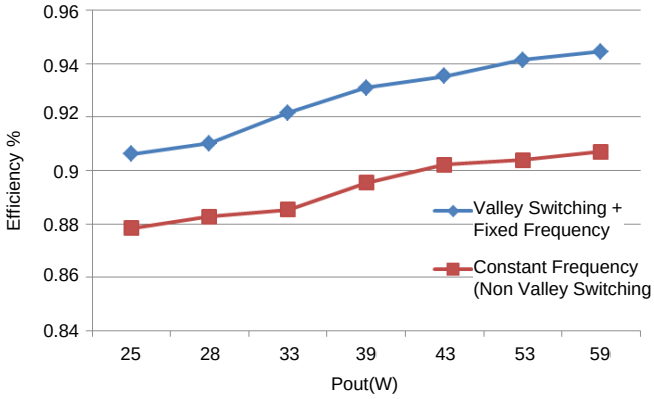

Figure 19. 220 Vin, 380 Vout: Efficiency

8 Summary

This document introduces the valley switching technique and how to configure the PWM module to realize it under light load condition. It presented a digital control optimization solution that can improve light load performance especially the efficiency and THD for a boost PFC converter without using any complex external logic circuits. The proposed optimization method utilizes the integrated programmable valley switching feature of the digital controller and the mathematical model of boost PFC converter to easily implement the valley switching and ZVS. The valley skipping is selected to maintain a relatively low switching frequency to get better light load efficiency. When the instantaneous input voltage is very low, an enhanced fixed frequency control method is adopted to eliminate the current distortion. The experimental results show that both light load THD of the input current and efficiency are improved. This method was validated by using a 750W interleaved boost PFC board that was controlled by a Texas Instruments C2000 microcontroller.

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265