*Application Report*

# *Achieve Delayed Protection for Three-Level Inverter With CLB*

*Aki Li, Ricky Zhang, and Ozino Odharo*

**ABSTRACT**

Three-level inverter topologies have been commonly used in high power applications, while a special protection control scheme is required, and many users tried to implement it with costly circuits externally. This application report discusses a more straightforward method using the configurable logic block (CLB) of the latest C2000™ devices.

## Table of Contents



## Trademarks

C2000™ is a trademark of Texas Instruments.

# 1 Introduction

Nowadays, three-level inverter topology has become increasingly popular in high power applications, like UPS and solar inverter. By increasing the bus voltage to 1000-V or 1500-V DC, the current can be reduced to maintain the same power levels, which can significantly reduce the power loss with less copper design. Besides, three-level inverter topology makes it possible to use the same switching device to support this much higher voltage stress than before. However, compared with traditional two-level inverter, in addition to more complicated power conversion control, a different fault protection scheme is required for three-level inverter.

Figure 1-1 shows a typical single phase three-level I-Type inverter, named neutral point clamped (NPC) inverter. The single phase NPC inverter includes 4 FETs, like IGBT, in series, where S1 and S4 are called outer switches, with S2 and S3 called inner switches.

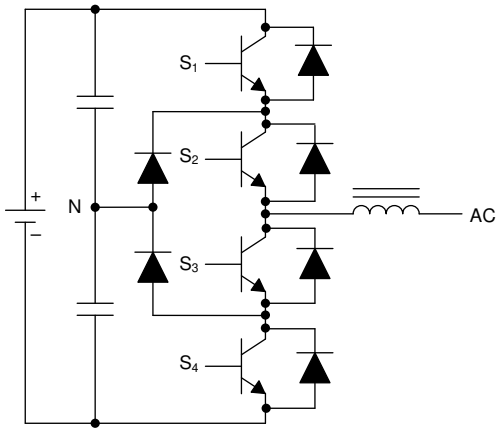



**Figure 1-1. Single Phase Three-Level I-Type Inverter**

The based control of the four series FETs are as follows.

- S1 always switches opposite of S3, and S4 always switches opposite of S2. There is a dead time between S1 and S3, as well as between S4 and S2, which always prevents a shoot-through condition.
- S1 and S4 cannot be on simultaneously.

Considering the difference between positive cycle and negative cycle when tied to the grid, the general switching states of 4 FETs in normal operation are shown in Table 1-1.

**Table 1-1. General Switching States in Normal Operation**

| Symbol | Switching States | | | |
|---|---|---|---|---|
| | S1 | S2 | S3 | S4 |
| Positive | Alternate switch | Remaining ON | Alternate switch | Remaining OFF |
| Negative | Remaining OFF | Alternate switch | Remaining ON | Alternate switch |

There are several events which lead to quick shut-down to protect the semiconductors and the system, like over current, thermal overload, and so forth. Unlike immediately switching off all the FETs simultaneously in two level inverter, for three-level inverter, it must be made sure that the correct switch-off sequence is maintained: outer switches (S1 or S4) off first, inner switches (S2 or S3) off after a specific delay, while the inner one must be switched on firstly during the recover process. This delayed protection requirement has been a challenge for lots of UPS or solar inverter customers for a long time. Since using software algorithm will cause too much delay to provide in-time protection, some customers have to use external hardware circuits, like FPGA or CPLD, to achieve such protection logic, which increases the system cost and also the development effort.

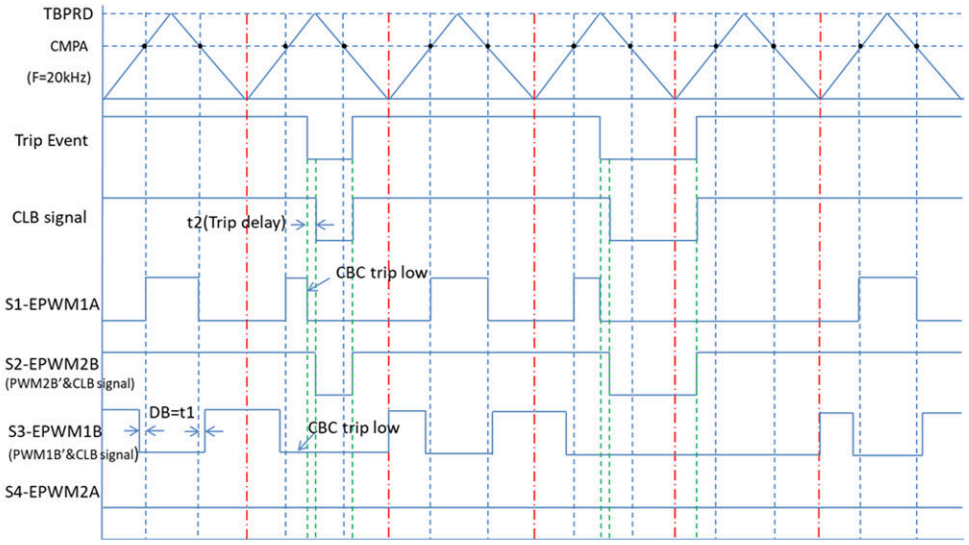
The C2000 configurable logic block (CLB) is a collection of configurable blocks that interconnect through software to implement custom digital logic functions, which can be an option to replace the function achieved by external CPLD or FPGA. This application report has designed and validated a simple protection scheme with CLB, and also discussed several extra requirements in three-level inverter. The example code is based on F280049C, and it can be easily migrated to any other C2000 devices with CLB, including F28388D/S, F28386D/S, F28379D/S, F28076 and F28004xC.

## 2 Design Overview

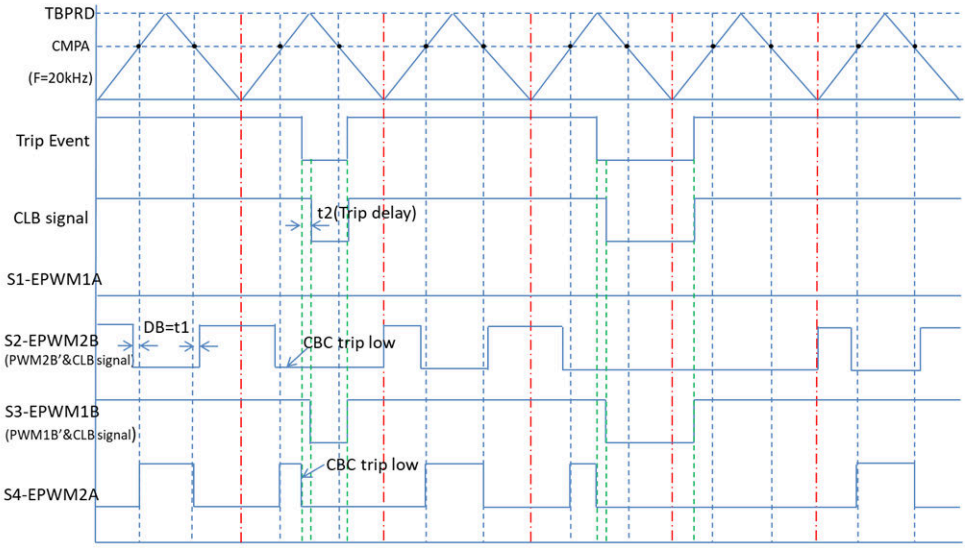
Figure 2-1 shows the expected EPWM protection results with CLB during positive cycle and negative operation.

In the given example, EPWM1A/2B/1B/2A are used for S1/2/3/4 control, respectively. The trip signal is simulated and generated with an EPWM output, EPWM8A here, for the validation of different trip conditions, including trip sustaining within or across one EPWM cycle.

Taking positive cycle operation as example, shown in 'A' of Figure 2-1, whenever the trip event occurs, EPWM1A and EPWM1B, which are configured with cycle-by-cycle tripping (CBC) action, are shut down immediately, while EPWM2B will be forced low after a delay t2. Besides, when the trip event disappears, EPWM2B will turn high immediately, which is also aligned with the basic recover requirement. The CLB signal is the key internal signal for the delayed trip feature, and EPWM1B and EPWM2B are actually the result of the AND logic of the initial EPWM output (EPWM1B' and EPWM2B') and CLB signal.




**A. Positive Cycle Operation**




**B. Negative Cycle Operation**

**Figure 2-1. Expected EPWM Protection Logic With CLB**

# 3 CLB Implementation

## 3.1 CLB Input Selection

EPWM1B and EPWM2B are routed to CLB TILE1 and TILE2, respectively, before outputting as external signals. Taking EPWM1B as example, it is routed to CLB1 by Global Signal Bus via Input Mux of Tile 1. The trip signal, sourced from the specific GPIO, needs to take 2 input bits, to form the predefined logic later.

To import these three signals into the CLB (BOUNDARY INz, where z is any number between 0–7) from the GPIOs, INPUT X-BAR and CLB X-BAR configurations are needed.

```
    // Configure CLB-XBAR AUXSIG0 as INPUTXBAR1
    //
    XBAR_setCLBMuxConfig(XBAR_AUXSIG0, XBAR_CLB_MUX01_INPUTXBAR1);
    XBAR_enableCLBMux(XBAR_AUXSIG0, XBAR_MUX01);

// CLB tile 1 configuration
    CLB_configLocalInputMux(CLB1_BASE, CLB_IN0, CLB_LOCAL_IN_MUX_GLOBAL_IN);
    CLB_configLocalInputMux(CLB1_BASE, CLB_IN1, CLB_LOCAL_IN_MUX_GLOBAL_IN);
    CLB_configLocalInputMux(CLB1_BASE, CLB_IN2, CLB_LOCAL_IN_MUX_GLOBAL_IN);

    CLB_configGlobalInputMux(CLB1_BASE, CLB_IN0, CLB_GLOBAL_IN_MUX_CLB_AUXSIG0);
    CLB_configGlobalInputMux(CLB1_BASE, CLB_IN1, CLB_GLOBAL_IN_MUX_CLB_AUXSIG0);
    CLB_configGlobalInputMux(CLB1_BASE, CLB_IN2, CLB_GLOBAL_IN_MUX_EPWM1B);

    CLB_configGPInputMux(CLB1_BASE, CLB_IN0, CLB_GP_IN_MUX_EXTERNAL);
    CLB_configGPInputMux(CLB1_BASE, CLB_IN1, CLB_GP_IN_MUX_EXTERNAL);
    CLB_configGPInputMux(CLB1_BASE, CLB_IN2, CLB_GP_IN_MUX_EXTERNAL);
```

The above code snippet shows the input configuration for EPWM1B. In this example, INPUT X-BAR 1 is used to route the trip signal from GPIO, and CLB X-BAR enables INPUT X-BAR 1 as the source for AUXSIG0, according to the CLB X-BAR Mux Configuration Table in the TRM. Here the trip signal is assigned for both CLB Input 0 and CLB Input 1. That is because the rising edge of the trip signal is needed to determine the rise edge of the expected CLB signal, which will be discussed later, so the input filter selection feature is used to detect the rising edge of the trip signal.

```
    CLB_selectInputFilter(CLB1_BASE, CLB_IN1, CLB_FILTER_RISING_EDGE);
```

## 3.2 Counter and FSM Configuration

The counter block is used to achieve the customized delay. Since the counter will always reset to 0 if the “Reset” input remains high, the trip signal, which is active low, can be used to trigger the counter to start counting. Thus, both MODE_0 and MODE_1 can be set to 1, which enables the counter increment as soon as the trip signal goes low. MATCH1 is set with the expected delay value, like 100, which means 1 µs, with the time base of 100 MHz for CLB module. Figure 3-1 shows how the counter is configured in the GUI-based SysConfig tool.

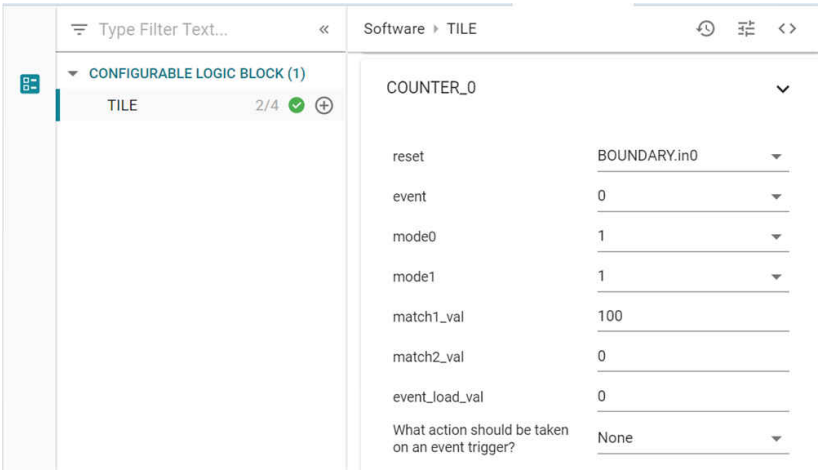



**Figure 3-1. Counter Settings in the SysConfig**

The state machine has been implemented with the FSW block as shown in Figure 3-2. Two inputs are used to identify the state of S0. E0 is referred to the counter match event, while E1 is the rising edge of the trip signal. Thus, S0 will go down at E0 and rise at E1. Thus, the Karnaugh map can be created for S0 state, as given in Table 3-1.

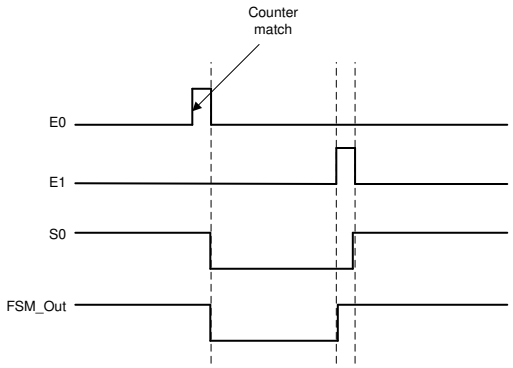



**Figure 3-2. State Machine in the FSW Block**

**Table 3-1. FSM S0 K-Map**

| S0 | E0E1 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|---|
| 0 | | 0 | 1 | 1 | 0 |
| 1 | | 1 | 1 | 0 | 0 |

Based on the Karnaugh map, the FSM equations for S0 can be deduced as shown in Equation 1.

$$S0 = (S0 \,\&\, \sim E0) \mid (\sim S0 \,\&\, E1) \tag{1}$$

Since the state transitions will always take 1 clock cycle to take effect, in order to reduce the undesirable delay, the FSM output takes the combinational output of S0 and E1, that is shown in Equation 2.

$$FSM\_OUT = S0 \mid E1 \tag{2}$$

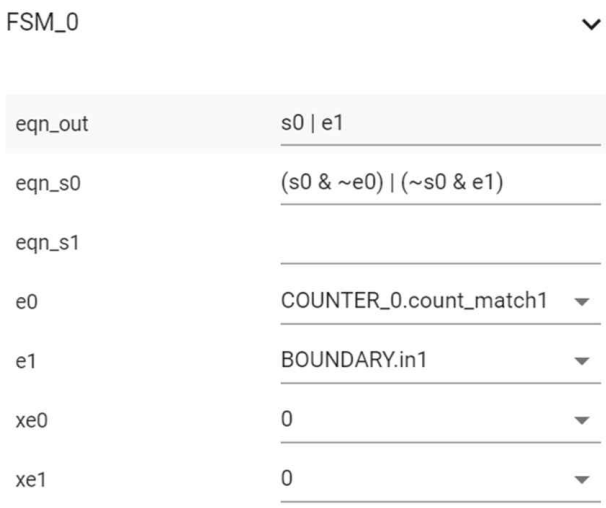



**Figure 3-3. FSM Configuration**

## 3.3 CLB Output

The final step is to use the output LUT block to combine the FSM_OUT and original EPWM1B' with AND logic. According to the Table 3-2, OUTPUT LUT2_0 of CLB TILE1 is used to route the resulting output (FSM_OUT&EPWM1B') as the new EPWM1B, as shown in Figure 3-4. Likewise, OUTPUT LUT2_0 of CLB TILE2 is used for EPWM2B. That is why two CLB TILEs are required in this application.

**Table 3-2. Peripheral Signal Multiplexer Table**

| CLB Instance | CLB Output Signal | Peripheral Signal | Peripheral Name |
|---|---|---|---|
| CBL1 | CLB1_OUT0_0 | PWMA | EPWM1 |
| CBL1 | CLB1_OUT1_0 | PWMA_OE | EPWM1 |
| CBL1 | CLB1_OUT2_0 | PWMB | EPWM1 |
| CBL1 | CLB1_OUT3_0 | PWMB_OE | EPWM1 |
| CBL1 | CLB1_OUT4_0 | AQ_PWMA | EPWM1 |
| CBL1 | CLB1_OUT5_0 | AQ_PWMB | EPWM1 |
| CBL1 | CLB1_OUT6_0 | DB_PWMA | EPWM1 |
| CBL1 | CLB1_OUT7_0 | DB_PWMB | EPWM1 |
| CBL1 | CLB1_OUT0_1 | QA | EQEP1 |
| CBL1 | CLB1_OUT1_1 | QB | EQEP1 |
| CBL1 | CLB1_OUT2_1 | DDIR | EQEP1 |
| CBL1 | CLB1_OUT3_1 | QCLK | EQEP1 |
| CBL1 | CLB1_OUT4_1 | G1.2 | CLB X BAR |
| CBL1 | CLB1_OUT5_1 | G3.2 | CLB X BAR |

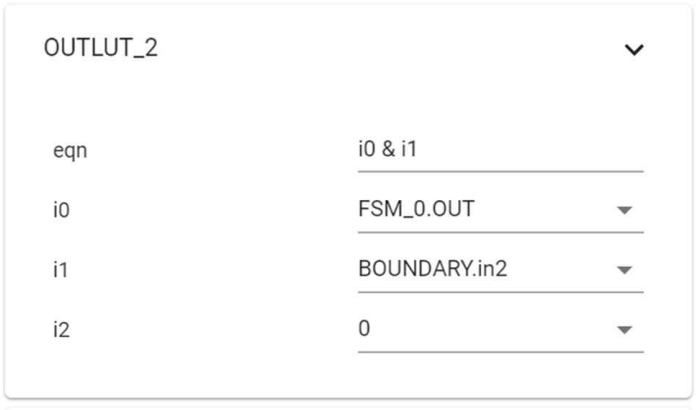



**Figure 3-4. OUTPUT LUT2_0 Configuration**

As shown in Table 3-2, each CLB output signal passes through an external multiplexer that intersects a specific peripheral signal. Thus, to export the CLB output to the original EPWM1B pin (GPIO1 in the example), it is needed to enable the OUTPUT LUT2_0 using the OUT_EN register, as shown in the below code snippets.

```
CLB_setOutputMask(CLB1_BASE, 0x4, true);   //1<<2 out2
CLB_setOutputMask(CLB2_BASE, 0x4, true);  //1<<2 out2
```

## 3.4 Completed Design

One of the tools provided with CCS is the graphical Tile Viewer showing all CLB components and the associated signal interconnections for a given CLB design. Figure 3-5 shows the completed block diagram of the design, including the logic connections among FSM, LUT and the Counter, with logic equations.

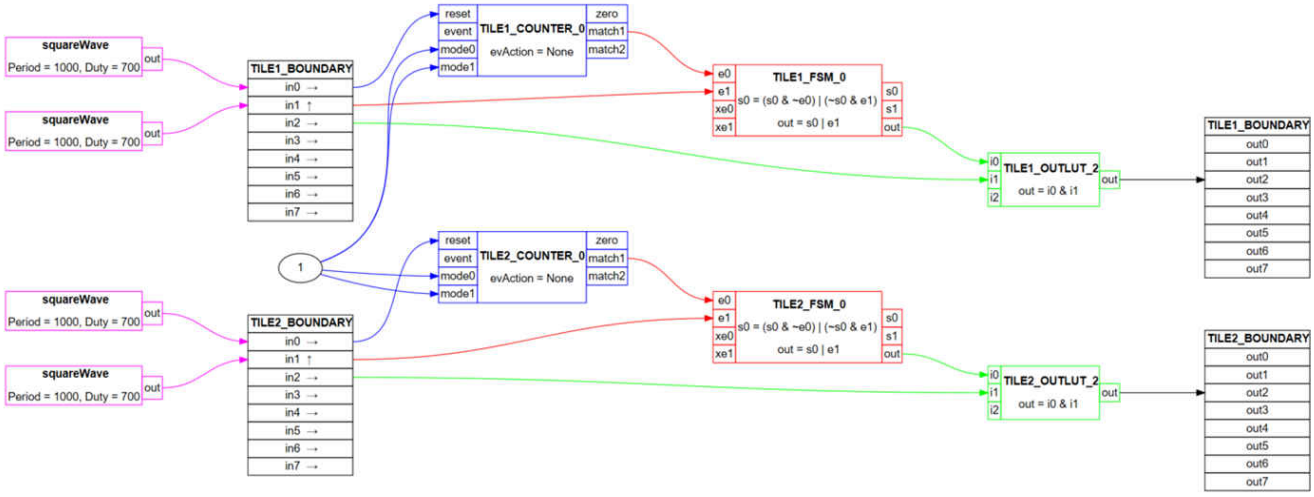



**Figure 3-5. Completed Design Block Diagrams**

In addition, as shown in the block diagram, the “BOUNDARY” items are added as the tile inputs to simulate the CLB function before the test. As shown in Figure 3-5, both “in0” and “in1” are defined as a same periodic PWM referring to the trip signal, where the rising edge is selected for the “in-edge” option of “in1”. “in2” is set as 1, which represents normally high EPWM1B or EPWM2B.

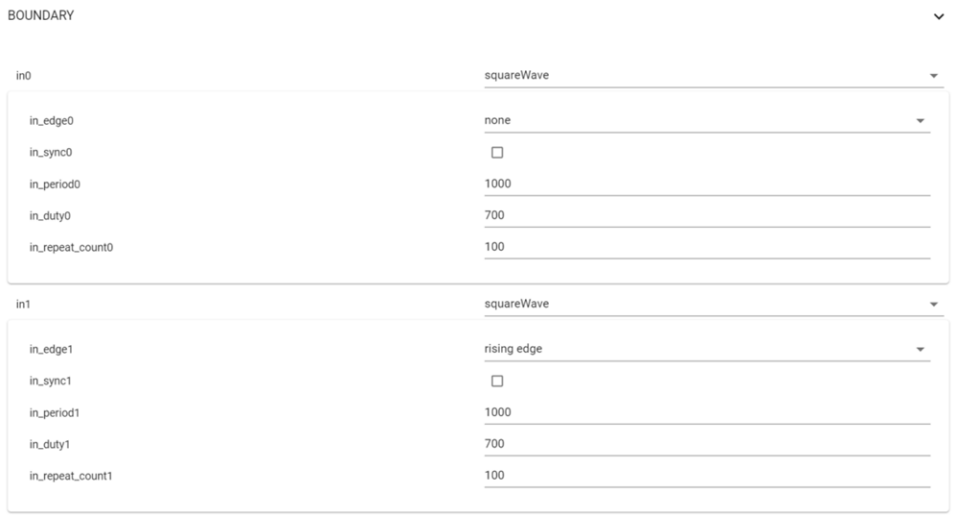



**Figure 3-6. Boundary Input Setting**

Once the CLB configuration and input stimuli have been defined, the simulation results can be obtained with the GTKwave viewer, as shown in Figure 3-7. And the expected delayed action is achieved at the CLB output signal. Note that the “clock_period” of the Global Parameters in the .syscfg tool should be changed to 10 ns, in order to match the time base of 100 MHz for CLB module, as shown in Figure 3-8. For more details of the CLB simulator, see the *CLB Tool User's Guide*.

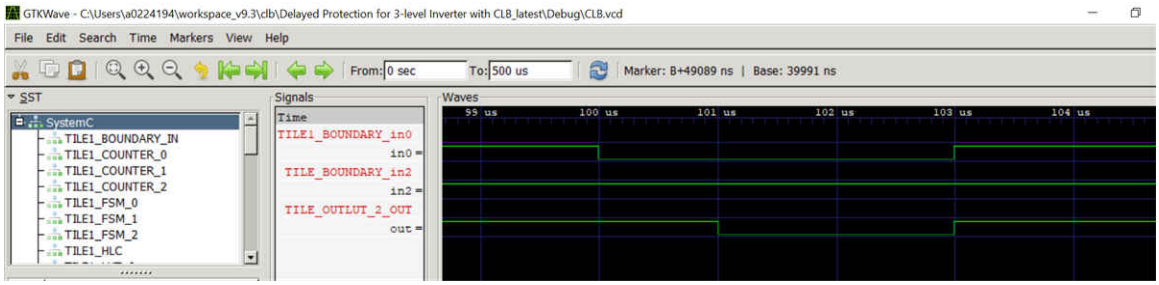



**Figure 3-7. CLB Simulation Result**

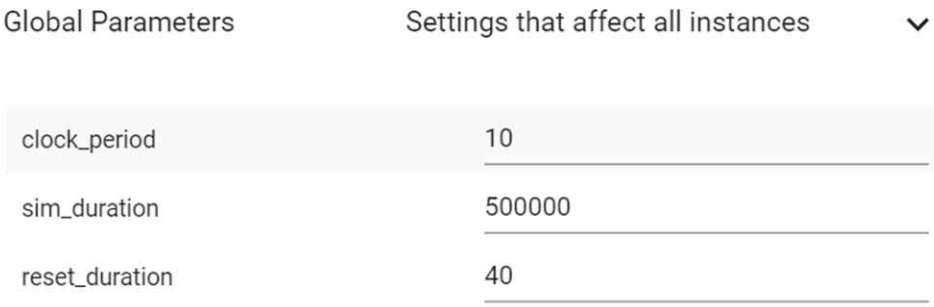



**Figure 3-8. Global Parameters Configuration**

## 4 Normal Operation With CBC Protection Configuration

As for the basic configurations of EPWM, the dead-band submodule is used for both EPWM1 and EPWM2, with active high complementary mode. The rising edges and falling edges delay can be customized using the DBRED and DBFED register.

### 4.1 CBC Protection Configuration

The trip signal (active low) is required to set as CBC trip event, which forces EPWM1A/B or EPWM2A/B low.

The following code shows that GPIO14, that is EPWM8A, is set as the source for INPUT X-BAR 1, and then fed to ePWM X-BAR TRIP4.

```
XBAR_setInputPin(XBAR_INPUT1,14);        // INPUT X-BAR 1 = GPIO14 = EPWM8
// Configure INPUT X-BAR 1 as EPWM X-BAR TRIP 4 source
XBAR_setEPWMMuxConfig(XBAR_TRIP4,XBAR_EPWM_MUX01_INPUTXBAR1);
XBAR_enableEPWMMux(XBAR_TRIP4, XBAR_MUX01);
```

The trip signal (EPWM8A) is configured as Digital Compare A High (DCAH) in both EPWM1 and EPWM2 modules, and DCAEVT2 event takes effect when DCAH becomes low. The following code snippet shows the configuration for EPWM1, which is achieved in the function “EPWM_CBC_CONFIG(uint32_t base1, uint32_t base2)”, and base1 and base2 are the base addresses of the 2 EPWM modules.

```
    //EPWM1 CBC
    // DCAH = TRIPIN4 = INPUT X-BAR 1 = EPWM8A
    EPWM_selectDigitalCompareTripInput(base1,EPWM_DC_TRIP_TRIPIN4,EPWM_DC_TYPE_DCAH);

    // DCAH = Low and DCAL = Don't care,Trigger DCAEVT2 when EPWM8A goes low

EPWM_setTripZoneDigitalCompareEventCondition(base1,EPWM_TZ_DC_OUTPUT_A2,EPWM_TZ_EVENT_DCXH_LOW);

EPWM_setTripZoneDigitalCompareEventCondition(base1,EPWM_TZ_DC_OUTPUT_B2,EPWM_TZ_EVENT_DCXL_LOW);
    // Enable CBC Trip in  ePWM1
    EPWM_setTripZoneAction(base1,EPWM_TZ_ACTION_EVENT_TZA,EPWM_TZ_ACTION_LOW);
    EPWM_setTripZoneAction(base1,EPWM_TZ_ACTION_EVENT_TZB,EPWM_TZ_ACTION_LOW);
```

## 4.2 Swapping EPWM Configurations During Zero Cross Point

Since grid-tied inverter needs to take care of the control during both positive and negative cycle, the EPWM signals should be swapped, together with related protection logic, at the zero cross point, which is handled in the PWM1_ISR as below. “negative_flag” and “positive_flag” can make sure the EPWM reconfiguration is implemented one time during the switching.

```
    if(positive_cycle)
    {
        EPWM_setCounterCompareValue(EPWM1_BASE,EPWM_COUNTER_COMPARE_A,EPwm1_Cmpa);
        if(negative_flag)
        {
          EPWM_setCounterCompareValue(EPWM2_BASE,EPWM_COUNTER_COMPARE_A,EPWM1_TBPRD);
          EPWM_enableTripZoneSignals(EPWM1_BASE,EPWM_TZ_SIGNAL_DCAEVT2);
          EPWM_disableTripZoneSignals(EPWM2_BASE,EPWM_TZ_SIGNAL_DCAEVT2);

          negative_flag=0;
          positive_flag=1;
        }
    }
    else
    {
        EPWM_setCounterCompareValue(EPWM2_BASE,EPWM_COUNTER_COMPARE_A,EPwm1_Cmpa);
        if(positive_flag)
        {
          EPWM_setCounterCompareValue(EPWM1_BASE,EPWM_COUNTER_COMPARE_A,EPWM1_TBPRD);
          EPWM_disableTripZoneSignals(EPWM1_BASE,EPWM_TZ_SIGNAL_DCAEVT2);
          EPWM_enableTripZoneSignals(EPWM2_BASE,EPWM_TZ_SIGNAL_DCAEVT2);

          positive_flag =0;
          negative_flag = 1;
        }
}
```

Table 4-1 has summarized the different EPWM settings for positive and negative cycles. During the zero cross point, the switching for EPwm1_Cmpa ↔ TBPRD for CMPA should be taken care, where CMPA is directly related to the duty cycle provided by the control loop, that is CMPA=(1- duty cycle)* TBPRD. Besides, during positive cycle, the CBC protection should be disabled for EPWM2 and enabled for EPWM1, and vice versa for negative cycle operation.

**Table 4-1. Different EPWM Settings for Positive and Negative Cycles**

| PWM Signals | Basic Setting | Positive Cycle | Negative Cycle |
|---|---|---|---|
| **EPWM1A-S1** | ↑ CAU<br>↓ ZRO, CAD | 1_CMPA=EPwm1_Cmpa<br>Enable CBC | 1_CMPA= EPWM1_TBPRD<br>Disable CBC |
| **EPWM2B-S2** | Active High Complementary to EPWM2A, with dead time=t1 | Disable CBC | Enable CBC |
| **EPWM1B-S3** | Active High Complementary to EPWM1A, with dead time=t1 | Enable CBC | Disable CBC |
| **EPWM2A-S4** | ↑ CAU<br>↓ ZRO, CAD | 2_CMPA=EPWM1_TBPRD<br>Disable CBC | 2_CMPA=EPwm1_Cmpa<br>Enable CBC |

# 5 Other Considerations

## 5.1 Trip Sourced From CMPSS

The example shown in Section 4 is the showcase for the trip signal from the GPIO, which refers to the output of the external comparator. Surely, the internal CMPSS can be used to trigger the trip signal. As for the normal operation for CBC protection, only the source of EPWM TRIP4 needs to be changed. For example, if using CMPSS1 high side comparator output for the trip source, the configurations of EPWM X-Bar should be changed as below.

```
// Configure CMPSS1_CTRIPH as EPWM X-BAR TRIP 4 source
XBAR_setEPWMMuxConfig(XBAR_TRIP4,XBAR_EPWM_MUX00_CMPSS1_CTRIPH);
XBAR_enableEPWMMux(XBAR_TRIP4, XBAR_MUX00);
```

Note that active low is needed for the CMPSS1_CTRIPH, so the output of comparator CMP1H is inverted during the initialization.

```
CMPSS_configHighComparator(CMPSS1_BASE, CMPSS_INSRC_DAC | CMPSS_INV_INVERTED);
```

Besides, the CLB X-BAR should reassign the output of CMP1H to AUXSIG0.

```
XBAR_setCLBMuxConfig(XBAR_AUXSIG0, XBAR_CLB_MUX00_CMPSS1_CTRIPH);

XBAR_enableCLBMux(XBAR_AUXSIG0, XBAR_MUX00);
```

## 5.2 Extend to 3 Phase Inverter

As for 3 phase NPC inverter, extra 4 EPWM modules are needed, for example, EPWM3&4 for the second phase, and EPWM5&6 for the third phase. According to the peripheral signal multiplexer for CLB, EPWM1~EPWM4 output can be replaced with the corresponding CLB output, which means EPWM3&4 need CLB Tile 3 and Tile 4 for the delayed protection using CLB. However, there are only four Tiles within the CLB module for most C2000 devices. As for the third phase, EPWM5B and EPWM6B can not be directly selected with the global input mux and peripheral signal multiplexer. The steps in the following sections are needed for implementing the delayed protection with the CLB.

### 5.2.1 Input Selection

At the beginning, GPIO8 and GPIO9 are configured as EPWM5A and EPWM5B, while GPIO10 and GPIO11 are configured as EPWM6A and EPWM6B, respectively. Secondly, use Input X-BAR to select the related GPIO pins for EPWM5B and EPWM6B.

```
XBAR_setInputPin(XBAR_INPUT3,9);        // INPUT X-BAR 3 = GPIO9 = EPWM5B
XBAR_setInputPin(XBAR_INPUT4,11);       // INPUT X-BAR 4 = GPIO11 = EPWM6B
```

Then, use CLB X-BAR to select the INPUT X-BAR INPUT 3 and INPUT4 as AUXSIG1 and AUXSIG2. In this example, CLB Tile 1 can be also used to handle the delayed protection for EPWM5B and EPWM6B, together with EPWM1B.

```
//for EPWM5B/EPWM6B input to CLB
XBAR_setCLBMuxConfig(XBAR_AUXSIG1, XBAR_CLB_MUX05_INPUTXBAR3);
XBAR_enableCLBMux(XBAR_AUXSIG1, XBAR_MUX05);
XBAR_setCLBMuxConfig(XBAR_AUXSIG2, XBAR_CLB_MUX07_INPUTXBAR4);
XBAR_enableCLBMux(XBAR_AUXSIG2, XBAR_MUX07);

CLB_configLocalInputMux(CLB1_BASE, CLB_IN3, CLB_LOCAL_IN_MUX_GLOBAL_IN);
CLB_configLocalInputMux(CLB1_BASE, CLB_IN4, CLB_LOCAL_IN_MUX_GLOBAL_IN);

CLB_configGlobalInputMux(CLB1_BASE, CLB_IN3, CLB_GLOBAL_IN_MUX_CLB_AUXSIG1);
CLB_configGlobalInputMux(CLB1_BASE, CLB_IN4, CLB_GLOBAL_IN_MUX_CLB_AUXSIG2);

CLB_configGPInputMux(CLB1_BASE, CLB_IN3, CLB_GP_IN_MUX_EXTERNAL);
CLB_configGPInputMux(CLB1_BASE, CLB_IN4, CLB_GP_IN_MUX_EXTERNAL);
```

### 5.2.2 Output Selection

Two extra GPIO pins are needed to export the CLB output signals, with OUTPUT X-BAR and CLB X-BAR. For example, configure GPIO16 and GPIO17 as OUTPUT X-BAR with the PINMUX settings.

```
GPIO_setPadConfig(16, GPIO_PIN_TYPE_STD);
GPIO_setPinConfig(GPIO_16_OUTPUTXBAR7);
GPIO_setDirectionMode(16, GPIO_DIR_MODE_OUT);
GPIO_setPadConfig(17, GPIO_PIN_TYPE_STD);
GPIO_setPinConfig(GPIO_17_OUTPUTXBAR8);
GPIO_setDirectionMode(17, GPIO_DIR_MODE_OUT);
```

And then, configure OUTPUT-XBAR OUTPUT7 and OUTPUT8 as CLB OUT4 and OUT5, which are enabled for EPWM5B and EPWM6B, respectively.

```
// Configure OUTPUT-XBAR OUTPUT7 as CLB1_OUT4
XBAR_setOutputMuxConfig(XBAR_OUTPUT7, XBAR_OUT_MUX01_CLB1_OUT4);
XBAR_enableOutputMux(XBAR_OUTPUT7, XBAR_MUX01);
// Configure OUTPUT-XBAR OUTPUT8 as CLB1_OUT5
XBAR_setOutputMuxConfig(XBAR_OUTPUT8, XBAR_OUT_MUX03_CLB1_OUT5);
XBAR_enableOutputMux(XBAR_OUTPUT8, XBAR_MUX03);
```

Since EPWM5B and EPWM6B share the same delayed protection logic with EPWM1 and EPWM2, Figure 5-1 shows the inner signal connectivity for CLB Tile 1.

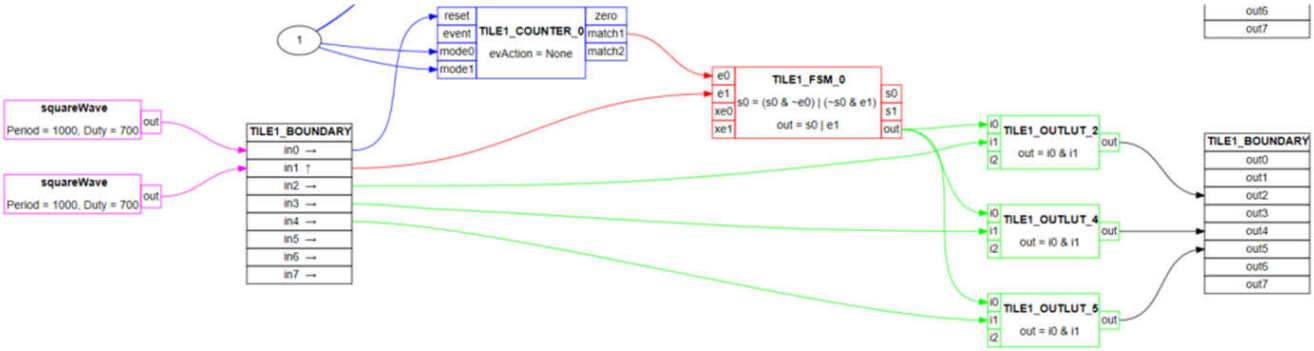



**Figure 5-1. CLB Tile 1 Inner Signal Connectivity**

## 5.3 Achieve 2 Level Protection Scheme

In actual applications, there are some non-serious events that require only the outer switches to turn off and keep the normal operation for inner switches. Thus, the two level protection scheme is needed:

- 1st level protection: only require the outer switch (S1/S4) to shut down, S2/S3 keep normal operation
- 2nd level protection: outer switch (S1/S4) to be switched off immediately, inner switch (S2/S3) should be turned off after a defined delay

From the above point of view, 1st level protection will normally occurs before 2nd level one, with the trip trigger level lower for 1st level protection, but during the sudden over current or short circuit conditions, these two protection events might occur almost at the same time. Therefore, in either protection trigger event, the EPWM for the outer switch is required to achieve CBC protection. To simply the configuration, the two trip events can be logically OR to serve as the same source for the CBC protection. According to the ePWM X-BAR architecture, the muxes that are enabled will be intrinsically OR'd before being passed on to the respective TRIPx signal on the EPWM. If using two GPIO pins for the external protection signals, two Input X-BAR INPUTs are selected for the same TRIP source.

```
// Configure INPUT X-BAR 1 as EPWM X-BAR TRIP 4 source
XBAR_setEPWMMuxConfig(XBAR_TRIP4,XBAR_EPWM_MUX01_INPUTXBAR1);
XBAR_enableEPWMMux(XBAR_TRIP4, XBAR_MUX01);
// Configure INPUT X-BAR 2 as EPWM X-BAR TRIP 4 source
XBAR_setEPWMMuxConfig(XBAR_TRIP4,XBAR_EPWM_MUX03_INPUTXBAR2);
XBAR_enableEPWMMux(XBAR_TRIP4, XBAR_MUX03);
```

Note that the trip signals should change to active high with the OR logic implemented. Actually, there can be no change for the external protection circuits if designed with active low initially. Using GPxINV in GPIO register configurations can inverse the IO pin logic.

Besides, the method discussed in Section 3 is still applied for 2nd level protection, where the 2nd level trip event is selected as the CLB input for the delayed protection. However, different from Section 3.1, it is the falling edge of the trip signal that used to determine the rising moment of the expected CLB signal. Thus the completed design block diagrams should be modified as Figure 5-2.

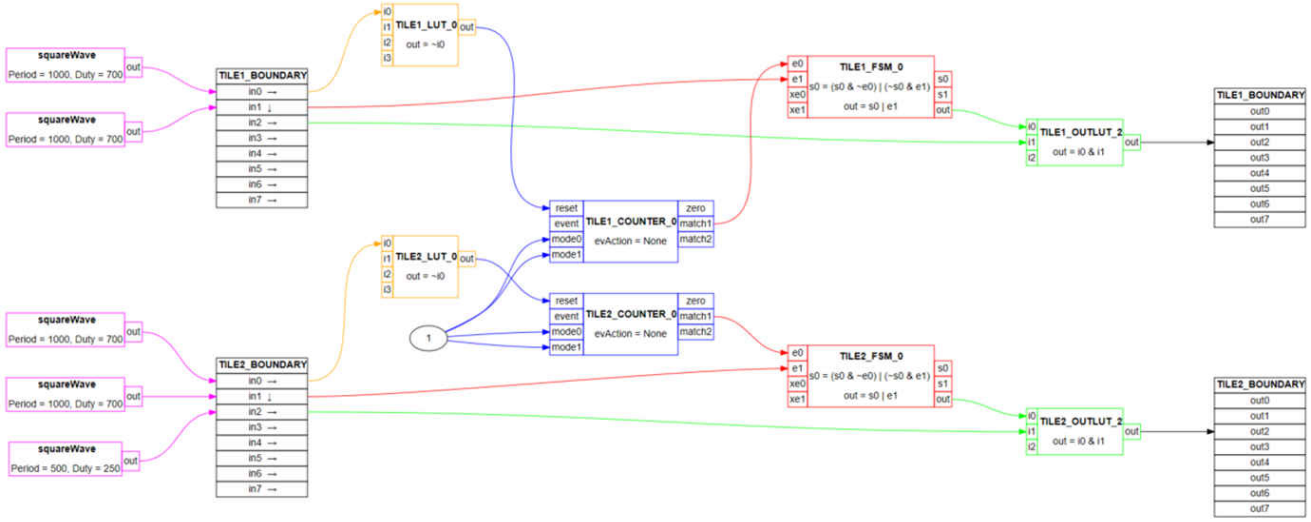



**Figure 5-2. Completed Design Block Diagrams for 2 Level Protection Scheme**

Therefore, the expected EPWM protection logic with CLB in Figure 5-2 should be updated as shown in Figure 5-3.

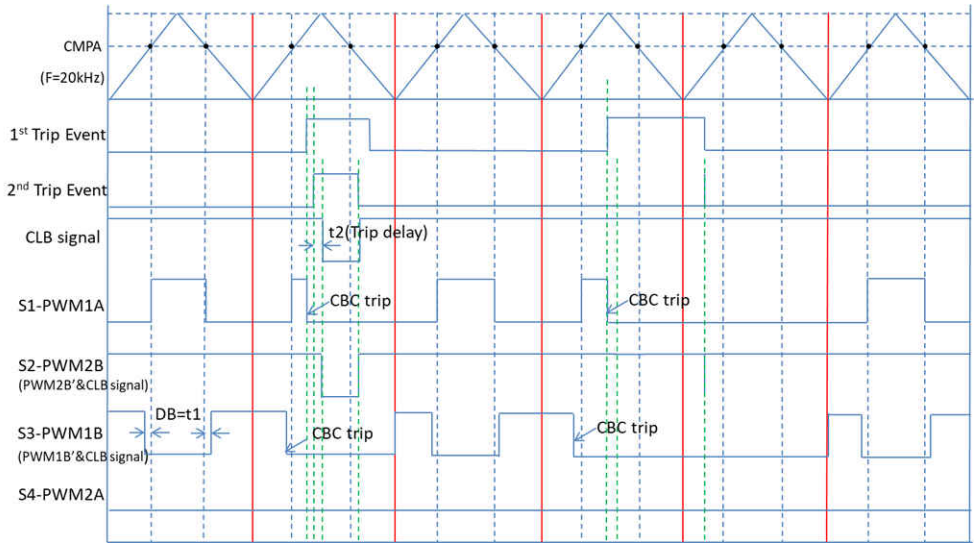



**Figure 5-3. Updated EPWM Protection Logic With CLB for 2 Level Protection Scheme**

As mentioned in Section 5.1, it is also possible to use internal CMPSS for the two level protection scheme, with the same sensing signal. Since there are two comparators with one CMPSS, including CMPH and CMPL, which can provide trigger signals for the two trip events, with two different references.

```
CMPSS_setDACValueHigh(CMPSS1_BASE, trip_point_2);
  CMPSS_setDACValueLow(CMPSS1_BASE, trip_point_1);

 // Configure CMPSS1_CTRIPH as EPWM X-BAR TRIP 4 source
 XBAR_setEPWMMuxConfig(XBAR_TRIP4,XBAR_EPWM_MUX00_CMPSS1_CTRIPH);
 XBAR_enableEPWMMux(XBAR_TRIP4, XBAR_MUX00);
 // Configure CMPSS1_CTRIPL as EPWM X-BAR TRIP 4 source
 XBAR_setEPWMMuxConfig(XBAR_TRIP4,XBAR_EPWM_MUX01_CMPSS1_CTRIPL);
 XBAR_enableEPWMMux(XBAR_TRIP4, XBAR_MUX01);
```

## 6 Test Results

The delayed protection scheme with CLB has been validated with the LaunchPad LAUNCHXL-F280049C. In the given example code, changing positive_cycle to 1 or 0 to decide positive cycle or negative cycle operation; changing EPWM8_Cmpa value to simulate different time scale for the trip signal; changing trip_within_one_cycle to simulate different types of trip signals, including sustaining within or across one EPWM cycle. It has been validated for single phase NPC inverter and extending to 3 phase NPC inverter.

The test results were measured with Kingst Logic Analyzer, as shown in Figure 6-1. And the related pin mapping information in this example is shown in Table 6-1.

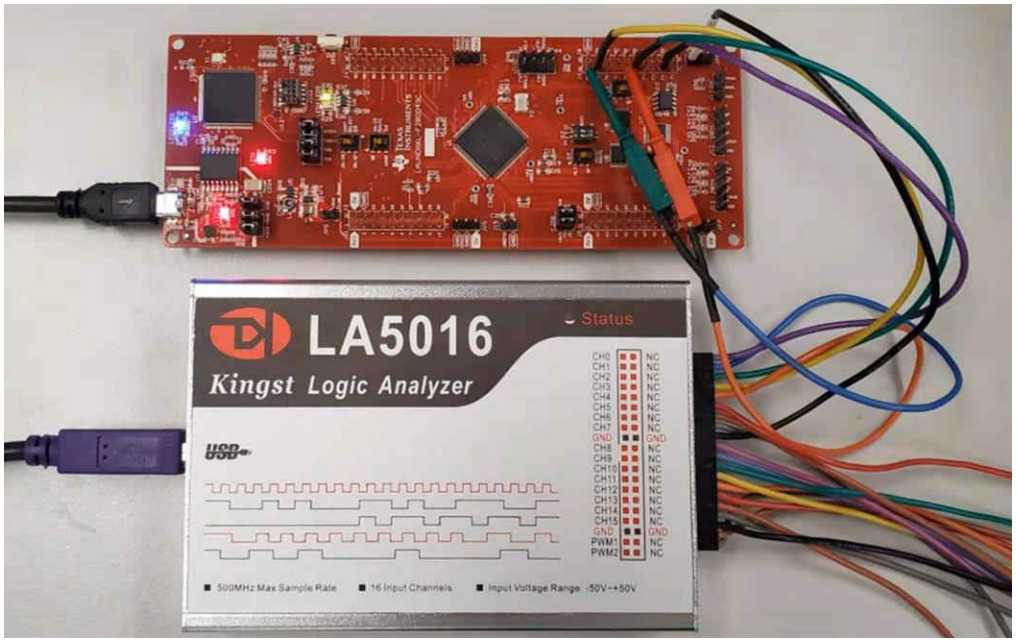



**Figure 6-1. Test Platform Set Up**

**Table 6-1. Pin Mapping in the Example**

| Signal Name | Description | Connection to the LaunchPad |
|---|---|---|
| EPWM8A | Trip Event | GPIO-14 |
| EPWM1A | Control signal for S1 | GPIO-00 |
| EPWM2B | Control signal for S2 | GPIO-03 |
| EPWM1B | Control signal for S3 | GPIO-01 |
| EPWM2A | Control signal for S4 | GPIO-02 |

**Table 6-1. Pin Mapping in the Example (continued)**

| Signal Name | Description | | Connection to the LaunchPad |
|---|---|---|---|
| EPWM5A | Control signal for S1 | See the 3rd phase for 3 phase NPC inverter | GPIO-08 |
| EPWM6B' | Control signal for S2, CLB output | | GPIO-17 |
| EPWM5B' | Control signal for S3, CLB output | | GPIO-16 |
| EPWM6A | Control signal for S4 | | GPIO-10 |
| EPWM6B | Original EPWM6B signal | | GPIO-11 |
| EPWM5B | Original EPWM5B signal | | GPIO-09 |

Figure 6-2 shows the condition of trip during positive cycle, where EPWM1A starts CBC protection right at the trip low moment, while EPWM1B turns low after a delay of 1.035 µs. It was well matched with the defined 1 µs, considering the extra timing required by the internal hardware logic. Figure 6-3 shows the trip signal during negative cycle, where the delayed protection logic works as expected.


**Figure 6-2. Trip Signal During Positive Cycle**


**Figure 6-3. Trip Signal During Negative Cycle**

Figure 6-4 shows the waveforms of EPWM5 and EPWM6 in normal operation during positive cycle, with both rising edge delay and falling edge delay of 1 µs defined. And EPWM5B and EPWM6B measured are not the initial EPWMxB, but the signals after the CLB module. It shows that the dead time between EPWM5A and EPWM5B is 1.025 µs and 0.975 µs, thus the delay induced by the CLB module should be acceptable.


**Figure 6-4. PWM5 and PWM6 in Normal Operation During Positive Cycle**

Figure 6-5 and Figure 6-6 show the waveform of EPWM5 and EPWM6 with trip signal during positive cycle and negative cycle, respectively.


**Figure 6-5. PWM5 and PWM6 With Trip Signal During Positive Cycle**


**Figure 6-6. Trip Across One PWM Period During Negative Cycle**

## 7 References


- Texas Instruments: *TMS320F28004x Real-Time Microcontrollers Technical Reference Manual*
- Texas Instruments: *TMS320F28004x Microcontrollers Data Manual*
- Rodriguez, Jose, et al. "A survey on neutral-point-clamped inverters." IEEE transactions on Industrial Electronics 57.7 (2009): 2219-2230.
- Kim, Youngroc, et al. "Design and control of a grid-connected three-phase 3-level NPC inverter for building integrated photovoltaic systems." 2012 IEEE PES Innovative Smart Grid Technologies (ISGT). IEEE, 2012.

# IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), 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, regulatory 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 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.

TI objects to and rejects any additional or different terms you may have proposed.

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