Application Note

UCD3138A ：使用 PCM 实现 Inverse BUCK-BOOST 控制与常见问题解决

Peng Zhang *FAE AA2*

摘要

UCD3138A 的 PCM 模式经常被用于控制如 HSFB, PSFB, 以及 Inverse BUCK-BOOST (IBB) 电源拓扑中。对于 IBB 拓扑，由于采样电感电流可能产生负压，因此实际应用中需要增加一个偏压，将输入到 UCD3138A 的电压抬升，这可能会引起环路的失控。本文将介绍如何设置 UCD3138A 的 PCM 环路以实现两相交错的 IBB 控制。并结合实际使用中，在存在偏压的条件下，实现 IBB 系统需要的最短脉冲发波预充电和正常环起两个功能。

内容

插图清单

1 IBB 基本控制环路

UCD3138A 作为 TI 电源控制系统中被广泛使用的一种数字控制器，其具有高速高精度硬件环路。并且高灵活度让 UCD3138 适用于多种电源拓扑中。UCD3138 是一种完全可编程的解决方案，能够根据客户在不同的应用需求中进行定制化配置。而 UCD3138 的 PCM 模式经常应用在 IBB 拓扑的控制上。在一种 IBB 的应用中系统设计如下：

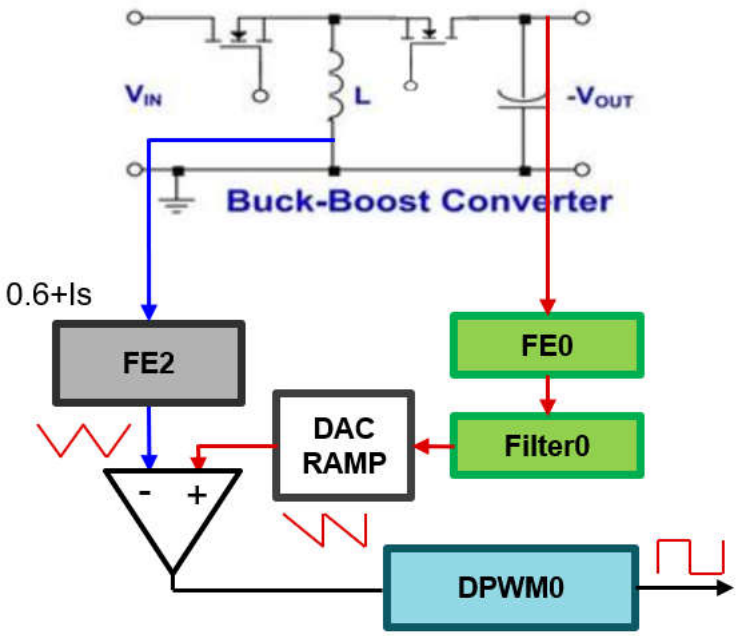


图 1-1. UCD3138A 应用于 IBB 控制

在实际使用中，由于客户需要预充电，输入偏压等需求。可能出现 PCM 控制失效的问题。因此正确理解 UCD3138A 的 PCM 环路与关键寄存器的配置与计算过程是解决此类问题的关键。

2 UCD3138A PCM 环路基本配置

UCD3138A 最为核心的特点就是它具有可编程配置的高速硬件环路。UCD3138A 在实际测试中，此环路从采样至完成调节的响应时间可在 400ns 以内，而最新一代的 UCD3138R 能够实现 156ns 以内的环路响应速度。使用 UCD3138A 进行 PCM 的控制。从硬件角度应当如图 4 理解：FE0 采样输出电压，与 FE0 中的 DAC 参考值进行比较获得误差值，进入 Filter 0 中通过 PID 运算(增益可编程)获得电压环的计算结果，即为 Fliter0 的输出 Filter_Duty。由于 UCD3138A 只有 FE2 模块支持 PCM 模式，因此 Fliter0 的输出应当 Loop 给 FE2 中作为 Ramp 模块的输入。而 FE2 的采样则作为电流环。当 FE2 采样得到的电流环输出大于经过 Ramp 补偿后的电压环输出结果 (DAC 当前值) 时，DPWM0A 发出高电平。反之 DPWM0B 发出高电平。实现 PCM 控制。

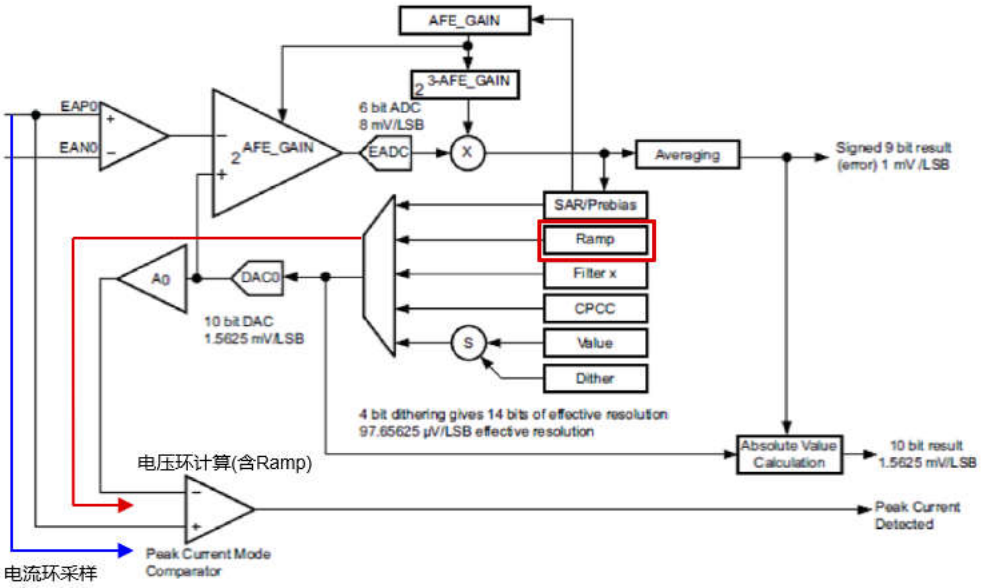


图 2-1. PCM 硬件环路

关键的寄存器配置如下：

```
'''
void init_loop_mux(void)
{
    LoopMuxRegs.DPWMMUX.bit.DPWM0_FILTER_SEL = 0;       // use filter 0 for DPWM 0
    LoopMuxRegs.SAMPTRIGCTRL.bit.FE0_TRIG_DPWM0_EN = 1; // use DPWM0 for filter0 sample trigger
//  LoopMuxRegs.SAMPTRIGCTRL.bit.FE1_TRIG_DPWM0_EN = 1; // use DPWM0 for filter1 sample trigger

    LoopMuxRegs.FECTRL2MUX.bit.DPWM0_FRAME_SYNC_EN = 1; //also need a trigger to make it work better.
    LoopMuxRegs.FECTRL0MUX.bit.DPWM0_FRAME_SYNC_EN = 1; //also need a trigger to make it work better.


    LoopMuxRegs.APCMCTRL.bit.PCM_FE_SEL    = 2;   // use FE2 for PCM
    LoopMuxRegs.APCMCTRL.bit.PCM_LATCH_EN = 1;   ///


    LoopMuxRegs.APCMCTRL.bit.PCM_EN        = 1;   // enable PCM

    LoopMuxRegs.PCMCTRL.bit.PCM_FILTER_SEL = 0; //select filter0
}
```

```
//----------------------------------------
Dpwm0Regs.DPWMCTRL0.bit.PWM_A_INTRA_MUX  = 0;   // Send PWM-C out the A output

Dpwm0Regs.DPWMCTRL0.bit.CBC_PWM_AB_EN = 1; //使能CBC
Dpwm0Regs.DPWMCTRL0.bit.CBC_ADV_CNT_EN = 1;

Dpwm0Regs.DPWMCTRL0.bit.BLANK_A_EN = 1;        //使能DPWM0A Blank time
Dpwm0Regs.DPWMBLKABEG.bit.BLANK_A_BEGIN = 0;   //Blank 起始位置
Dpwm0Regs.DPWMBLKAEND.bit.BLANK_A_END = 25; //25 x 4ns =100ns

Dpwm0Regs.DPWMCTRL0.bit.BLANK_B_EN = 1;            //使能DPWM0B Blank Time
Dpwm0Regs.DPWMBLKBBEG.bit.BLANK_B_BEGIN = EVENT3;   // Blank B 起始位置
Dpwm0Regs.DPWMBLKBEND.bit.BLANK_B_END = EVENT3 + 25; //25 x 4ns =100ns

Dpwm0Regs.DPWMCTRL0.bit.PWM_EN = 1;     // enable DPWM0 locally

FaultMuxRegs.DPWM0CLIM.bit.ANALOG_PCM_EN   = 1; //DPWM0 Enable PCM

//----------------------------------------
Dpwm0Regs.DPWMCTRL0.bit.PWM_A_INTRA_MUX  = 0;   // Send PWM-C out the A output

Dpwm0Regs.DPWMCTRL0.bit.CBC_PWM_AB_EN = 1; //使能CBC
Dpwm0Regs.DPWMCTRL0.bit.CBC_ADV_CNT_EN = 1;

Dpwm0Regs.DPWMCTRL0.bit.BLANK_A_EN = 1;        //使能DPWM0A Blank time
Dpwm0Regs.DPWMBLKABEG.bit.BLANK_A_BEGIN = 0;   //Blank 起始位置
Dpwm0Regs.DPWMBLKAEND.bit.BLANK_A_END = 25; //25 x 4ns =100ns

Dpwm0Regs.DPWMCTRL0.bit.BLANK_B_EN = 1;            //使能DPWM0B Blank Time
Dpwm0Regs.DPWMBLKBBEG.bit.BLANK_B_BEGIN = EVENT3;   // Blank B 起始位置
Dpwm0Regs.DPWMBLKBEND.bit.BLANK_B_END = EVENT3 + 25; //25 x 4ns =100ns

Dpwm0Regs.DPWMCTRL0.bit.PWM_EN = 1;     // enable DPWM0 locally

FaultMuxRegs.DPWM0CLIM.bit.ANALOG_PCM_EN   = 1; //DPWM0 Enable PCM
```

```
void init_front_end0(void)
{
    FeCtrl0Regs.EADCDAC.bit.DAC_VALUE = 160;
    FeCtrl0Regs.EADCCTRL.bit.AFE_GAIN = 2;

//    FeCtrl2Regs.RAMPCYCLE.bit.SWITCH_CYC_PER_STEP = 0;   //32ns/cycle
//
    FeCtrl2Regs.DACSTEP.bit.DAC_STEP              = 1000; //
    FeCtrl2Regs.RAMPCTRL.bit.PCM_START_SEL        = 1;    // 0 - Use DAC value from DAC; 1 - use filter out
    FeCtrl2Regs.RAMPDACEND.bit.RAMP_DAC_VALUE = 500; //may not be perfect, keep above 0 to avoid potential problems

    FeCtrl2Regs.EADCDAC.bit.DAC_VALUE  =320 * 16;   //Set DAC at 0.97V

    FeCtrl2Regs.EADCCTRL.bit.AFE_GAIN  = 2;

    FeCtrl2Regs.EADCCTRL.bit.EADC_MODE = 5;       // Peak current mode
    FeCtrl2Regs.RAMPCTRL.bit.RAMP_EN   = 1;       //use internal slope--1182012
    FeCtrl2Regs.EADCCTRL.bit.D2S_COMP_EN = 1; //vital to enable comparator - this enables APCM comparator.
}

void connect_filter_0_to_front_end2_DAC(void)
{
    //LoopMuxRegs.SAMPTRIGCTRL.bit.FE0_TRIG_DPWM0_EN = 1; //use DPWM0 for filter0 sample trigger
   // LoopMuxRegs.FILTERKCOMPA.bit.KCOMP0 = 0x3fff; //0x3FFF is full scale

    LoopMuxRegs.EXTDACCTRL.bit.DAC2_SEL = 4; // 3 for CPCC. 4 for filter 0;
    LoopMuxRegs.EXTDACCTRL.bit.EXT_DAC2_EN = 0; //enable DAC 2 to take input from selected spot.1 is driven by filter
}
```

结合波形图能够更好的理解寄存器在实际控制中的作用。FE2 中的 DAC_Step 寄存器能够控制 Ramp 的补偿斜率，DAC_Value 能够控制 Ramp 补偿的终点。这里应当注意，在 Ramp 补偿过程中，DAC 的当前值始终向 DAC_Value 设置的终点靠近，如果 Filter 的输出大于 DAC_Value, 则 DAC 当前结果持续增加，反之则减小。

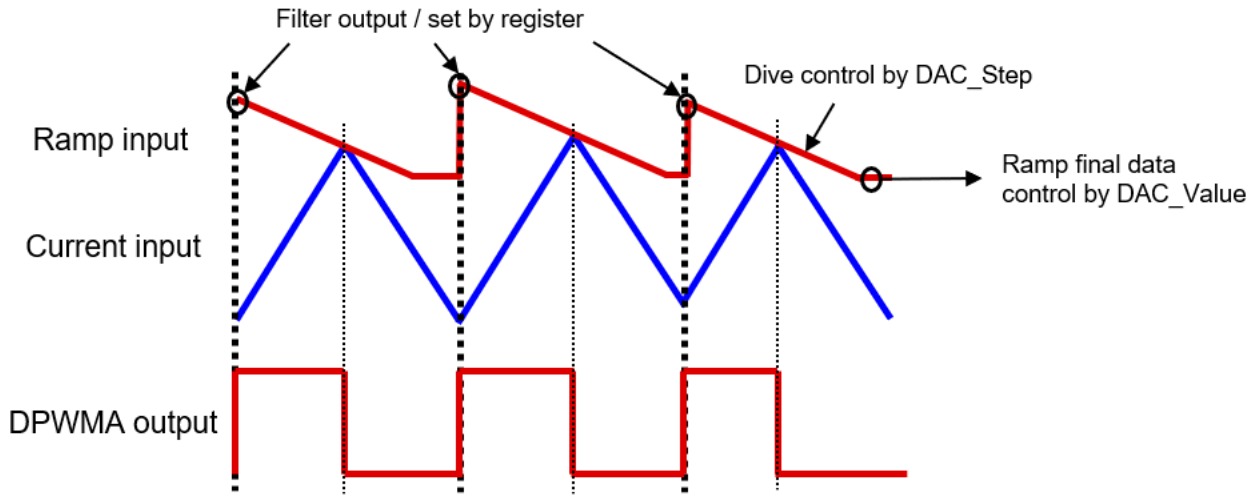


图 2-2. UCD3138A PCM 的工作原理

对于 Blank time 的意义，是在 MOSFET 导通时，电感电流不连续，此时开关节点可能产生一个较大的振铃，使输出电压被错误的采样，导致 DPWM 被误关断。针对这个问题 UCD3138 能够通过使能 Blank 寄存器，屏蔽 DPWM 的上升沿后的一段时间，使 DPWM 的上升沿不会错误的触发 PCM。使能 DPWM Blank 功能与 Blank 时间的寄存器如下：

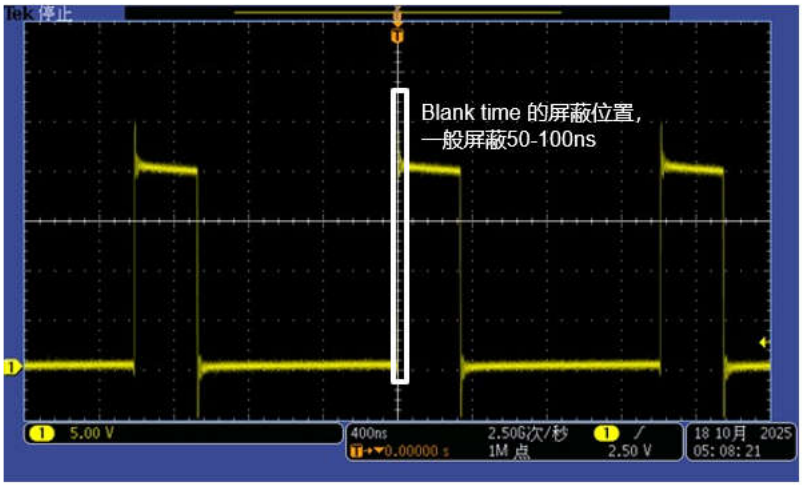


图 2-3. 开关节点的振铃与误触发保护

```
//----------------------------------------
Dpwm0Regs.DPWMCTRL0.bit.PWM_A_INTRA_MUX  = 0;   // Send PWM-C out the A output

Dpwm0Regs.DPWMCTRL0.bit.CBC_PWM_AB_EN = 1; //使能CBC
Dpwm0Regs.DPWMCTRL0.bit.CBC_ADV_CNT_EN = 1;

Dpwm0Regs.DPWMCTRL0.bit.BLANK_A_EN = 1;        //使能DPWM0A Blank time
Dpwm0Regs.DPWMBLKABEG.bit.BLANK_A_BEGIN = 0;   //Blank 起始位置
Dpwm0Regs.DPWMBLKAEND.bit.BLANK_A_END = 25; //25 x 4ns =100ns

Dpwm0Regs.DPWMCTRL0.bit.BLANK_B_EN = 1;            //使能DPWM0B Blank Time
Dpwm0Regs.DPWMBLKBBEG.bit.BLANK_B_BEGIN = EVENT3;   // Blank B 起始位置
Dpwm0Regs.DPWMBLKBEND.bit.BLANK_B_END = EVENT3 + 25; //25 x 4ns =100ns

Dpwm0Regs.DPWMCTRL0.bit.PWM_EN = 1;     // enable DPWM0 locally

FaultMuxRegs.DPWM0CLIM.bit.ANALOG_PCM_EN   = 1; //DPWM0 Enable PCM
```

图 2-4. UCD3138A Blank time 的使能与时间设置

3 引入偏压后的 PCM 失效分析

实际使用中，UCD3138 在进行 IBB 控制的过程中，会有两个调整，是在理论控制基础上需要进行附加的功能。

1. 利用 UCD3138 的 Blank 功能为环路进行预充电 (共计 100ms 左右时长)，而后再正常启动环路。
2. FE 采样需要增加一个偏置电压，使输入到 FE 的电压恒正。

对于同步 Inverse BUCK-BOOST 拓扑电源控制来说，需要进行电压环与电流环的采样。电压环一般采样输出电压，其极性并不会反转，因此对于 UCD3138 的 Front End(FE) 接口是安全的。但是轻载工作于 FPWM 模式下，IBB 同其他电源类似，电感电流将会反向 (如 图 3-1 [3])。

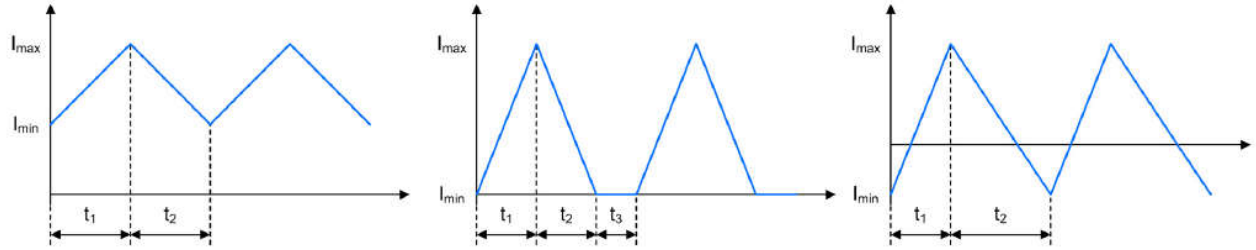


图 3-1. 轻载时电感电流反向

因此电流环采样时输出到 FE 接口的电压值可能为负值，这样对 UCD3138 来说是危险的，很可能造成 UCD3138 的损毁。

7.1 Absolute Maximum Ratings(1)

over operating free-air temperature range (unless otherwise noted)

		VALUE		UNIT
		MIN	MAX	
Voltage	V33D to DGND	–0.3	3.8	V
	V33DIO to DGND	–0.3	3.8	V
	V33A to AGND	–0.3	3.8	V
	BP18 to DGND	-0.3	2.5	V
	Ground difference, |DGND – AGND|		0.3	V
	Voltage applied to any pin, excluding AGND(2)	–0.3	3.8	V

图 3-2. UCD3138A 电压输入范围 [2]

因此，我们通常需要为 FE 的输入提供一个正偏压来解决这个问题，如 图 3-3 所示为 UCD3138 的输入提供 0.6V 的正向偏压，以实现 FE 的输入恒正。

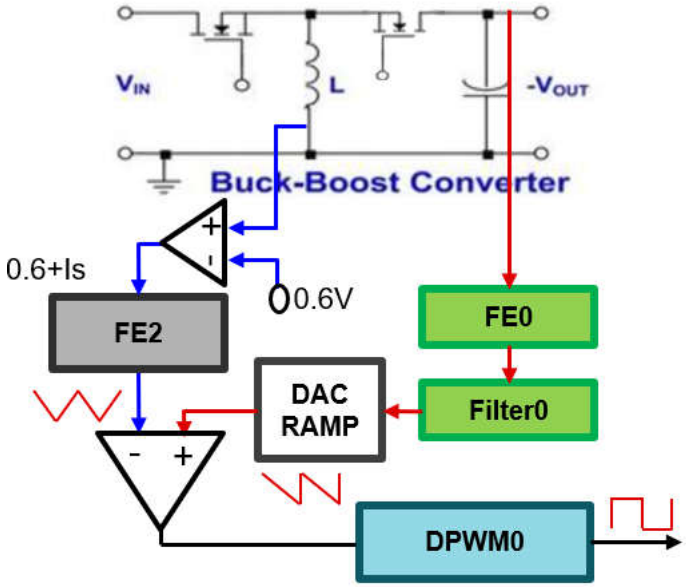


图 3-3. 为 FE2 提供 0.6V 的正向偏压

通过 图 3-4 能够更为直观的说明，在实际工作中，预充电阶段与正常工作阶段的电压环与电流环的不同计算预期。

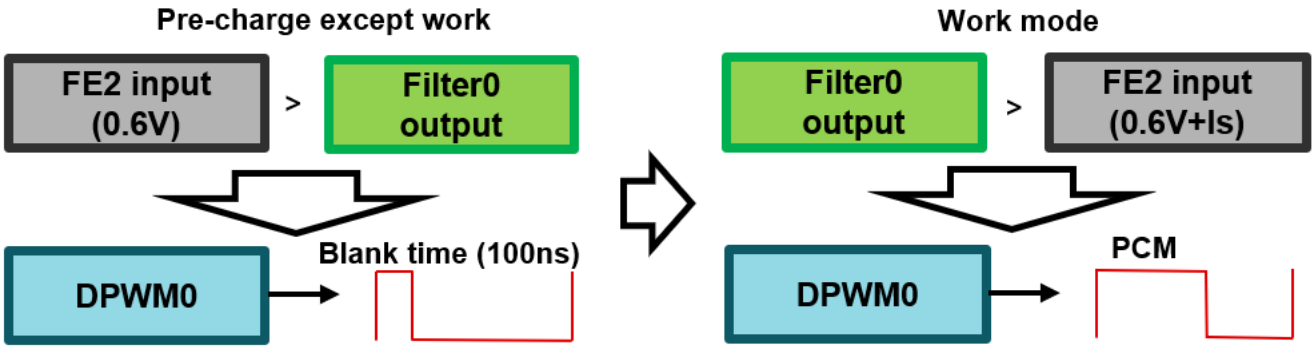


图 3-4. 预充电阶段与正常工作模式的环路预期

但是提升了这个偏压会导致一个问题。FE2 输入的电流环采样值将为 0.6V+*Isample* ，这样在启动时，即使没有电流，FE2 的采样也会有 0.6V，而启动时输出电压为 0，这样环路始终计算输出电压偏低。因此，第一阶段的利用 Blank time 进行预充电的功能，会失去控制，从波形上看，DPWM0A 会一直导通，直到达到最大占空比或是触发过流保护。

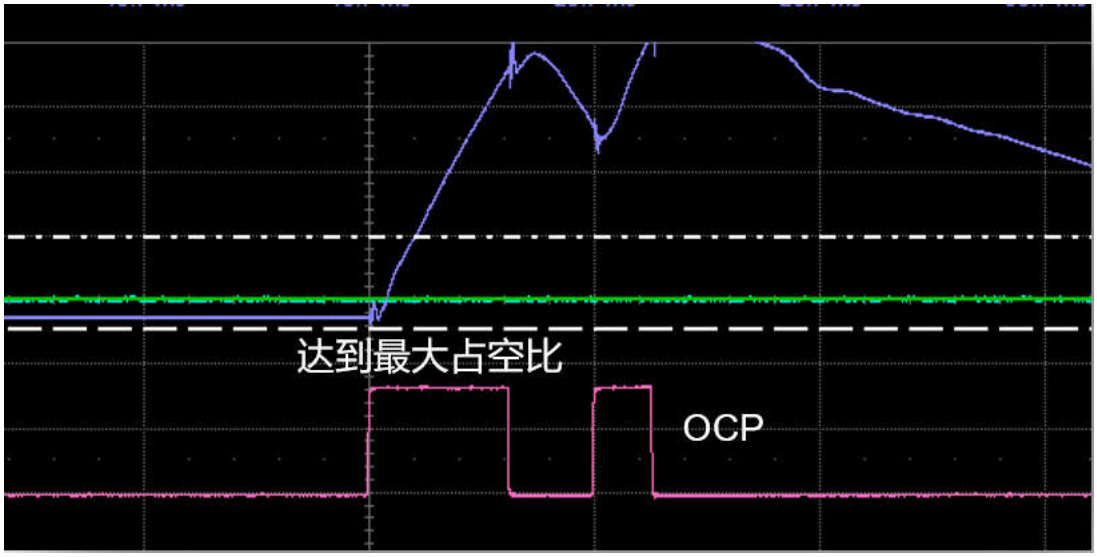


图 3-5. PCM 失效

4 环路修正与实验验证

根据问题的分析，可以得到，在预充电阶段失控的主要原因是环路的 Filter 计算结果没有适应这个存在偏压的条件。而 Filter 的计算结果由 FE0 的误差输入与 Filter 的环路增益共同控制。针对一般的环路增益不足或者基准设定问题，可以调整 FE0 的 DAC_Value 调节阈值或者调节 Filter 的 KI 系数，调节增益。但是针对 IBB 启动预充电失效的问题，DAC_Value 无法设定为负值，电压输出为 0 的时候误差输出恒正。而环路增益系数在后续正常工作模式中还要继续使用，修改起来相对麻烦。针对这个问题，我们推荐使用 Filter 中的 Clamp 寄存器，对 Filter 的输出进行钳位。

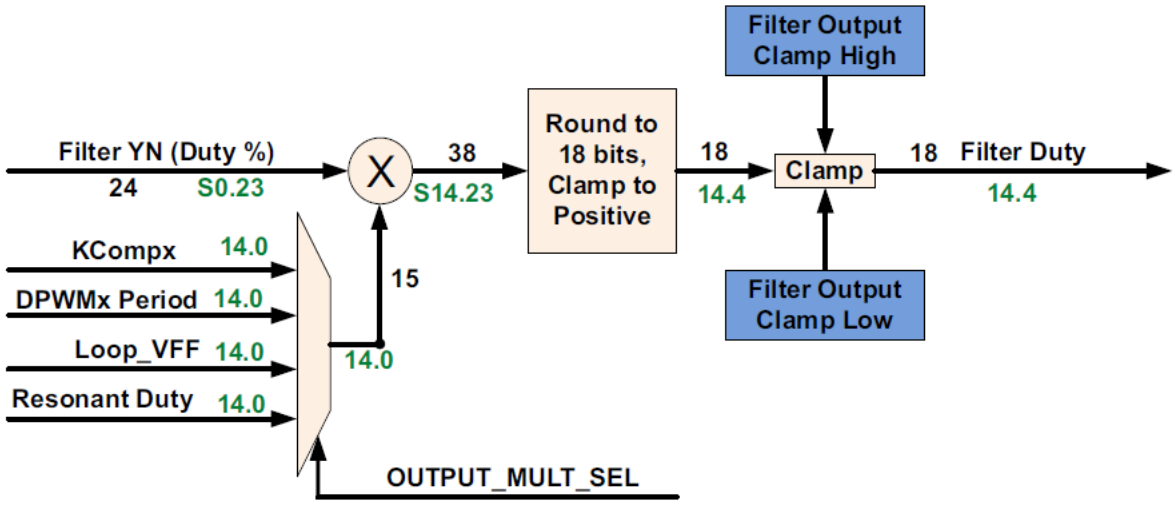


图 4-1. Filter output Clamp 寄存器 [1]

针对 IBB 的两环节。首先我们应当将 Filter Output Clamp Low 钳位至一个较低的值 (比如 0x10) 。确保电压环的输出在第一环节始终低于电流环的 0.6V 采样值。这样在开启 100ns 的 Blank time 后将会正常实现预充电的短脉冲发波。

图 4-2. 预充电发波

在 100ms 的延时过后，将钳位恢复正常范围。并将 FE0 中的 DAC_Value (FB Ref) 设置为正常工作的模式，环路将正常启动。

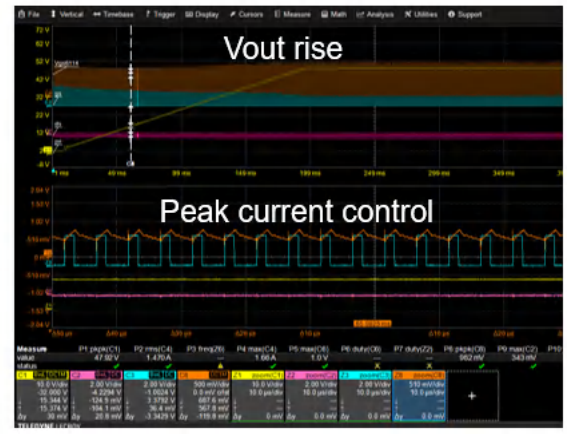


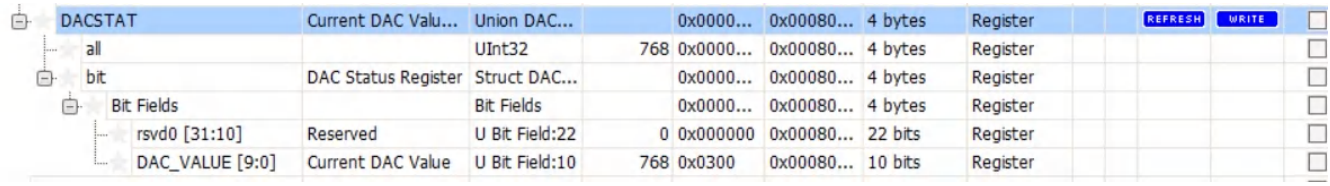
图 4-3. IBB 正常工作发波

至此我们实现了一种简单的方式，能够使采样电感电流的应用中，在具有偏压的条件下，完成预充电和正常环起，这两个功能。

5 其他调试经验与常见问题

应当注意的是，Filter 存在多级的钳位，而在实际应用中，我们应当抓住应用的需求，如果是希望检测计算过程中有溢出或者过低的情况，可以对前两级进行钳位。但是在 UCD31XX 技术文档中，说明的类似 Filter 计算结果将 loop 给某个模块。没有特殊说明的话，将统一为 Filter_Duty 的计算结果，并且在后续移动中如果需要截位，统一截低位，保留高位数据。

以 PCM 控制为例。如果我们设置 Filter Output Clamp Low & High 同时为: 0x300FF ，但是 Front End 模块的 DAC 只支持 10bit，那么实际 DAC 的赋值将会是 0x300。这里是很多开发者常见的疑问。我们可以将 PCM 模式开启后，将 DAC_Step 设置为 0，设置 Filter Output Clamp Low & High 为同一值，然后再通过 memory debugger 读取 DAC 的当前值。与计算结果相对比，来验证自己对于计算的理解过程是否正确。



DACSTAT	Current DAC Valu...	Union DAC...		0x0000...	0x00080...	4 bytes	Register	REFRESH	WRITE
all		UInt32	768	0x0000...	0x00080...	4 bytes	Register		
bit	DAC Status Register	Struct DAC...		0x0000...	0x00080...	4 bytes	Register		
Bit Fields		Bit Fields		0x0000...	0x00080...	4 bytes	Register		
rsvd0 [31:10]	Reserved	U Bit Field:22	0	0x000000	0x00080...	22 bits	Register		
DAC_VALUE [9:0]	Current DAC Value	U Bit Field:10	768	0x0300	0x00080...	10 bits	Register		

图 5-1. 设置 Clamp 寄存器并利用 Memory debugger 读取 DAC_Value 的值

6 小结

采用 UCD3138 系列数字电源控制器能够实现高速，高精度的控制且控制拓扑灵活可变的优点。同时具有丰富的应用手册及文档来帮助不同程度的开发者进行开发不同应用拓扑。而下一代 UCD3138R 将会具有更强大的主核 (100MHz M33)，更高的 DPWM 精度以及环路响应速度。综上 UCD3138 系列产品将为实现高速高精度电源控制提供一个优秀的解决方案。

7 参考文献：

1. UCD3138A datasheet
2. UCD31xx Digital Power Supply Controller 技术手册
3. Power Stage Topology Handbook

重要通知和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、与某特定用途的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他安全、安保法规或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的相关应用。严禁以其他方式对这些资源进行复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。对于因您对这些资源的使用而对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，您将全额赔偿，TI 对此概不负责。

TI 提供的产品受 TI 销售条款)、TI 通用质量指南 或 ti.com 上其他适用条款或 TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。 除非德州仪器 (TI) 明确将某产品指定为定制产品或客户特定产品，否则其产品均为按确定价格收入目录的标准通用器件。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

最后更新日期：2025 年 10 月