Application Note

如何使用可配置逻辑块 (CLB) 实施定制串行接口

Gustavo Martinez

摘要

本应用报告介绍了如何使用可配置逻辑块 (CLB) 设计定制串行接口。本文介绍了一种设计基于 CLB 的串行端口的分步方法，还说明了常见设计挑战和潜在解决方案。最后，给出了两个串行端口设计示例，包括完整的设计详细信息、仿真结果和硬件测试结果。第一个示例是基于 CLB 的串行端口设计，该端口通过音频系统中常用的时分多路复用 (TDM) 总线发送和接收数据。第二个示例使用基于 CLB 的串行端口将数据发送到 LED 矩阵显示屏。C2000ware 软件开发套件 (SDK) 中包括所有示例代码。

内容

插图清单

English Document: SPRAD62
Copyright © 2023 Texas Instruments Incorporated

表格清单

商标

C2000™ and Code Composer Studio™ are trademarks of Texas Instruments.
所有商标均为其各自所有者的财产。

English Document: SPRAD62
Copyright © 2023 Texas Instruments Incorporated

1 引言

串行接口通常用于许多工业和汽车应用。当今微控制器中提供的标准串行接口可支持访问众多外围器件。但是，在某些情况下，可能需要定制串行接口来支持特殊的串行协议。C2000™ 实时微控制器中包括的可配置逻辑块 (CLB) 可通过编程来仿真这些定制串行接口。

本应用手册介绍了一种分步设计方法，用于指导设计人员设计、仿真和测试定制串行接口。整个应用报告中介绍了一些常见的设计挑战和潜在的解决方案。最后，报告中提供了两个示例，这两个示例包含完整的设计详细信息、仿真结果和硬件测试结果。示例代码可以很容易地从 C2000ware 软件开发套件 (SDK) 中获得。

2 串行端口设计方法

以下是使用 CLB 设计定制串行接口的分步指南。此处介绍的步骤在 CLB 逻辑设计过程中从头到尾指导设计人员。针对常见的设计挑战和缺陷提供了解决方案。

本应用报告假定读者已经熟悉 CLB 的架构和 Code Composer Studio (CCS) 集成开发环境 (IDE)。以下是可用于快速概览 CLB 和 C2000 实时微控制器的培训材料列表：

- 可配置逻辑块 (CLB) 介绍（视频）
- 可配置逻辑块 (CLB) 架构（视频）
- 可配置逻辑块 (CLB) 编程工具培训（视频）
- 适用于 C2000 实时微控制器的 SysConfig 开发工具
- C2000 Academy 在线培训

请务必记住，不同 C2000 实时微控制器的 CLB 类型、CLB 逻辑块数量和交叉开关 (XBAR) 逻辑会有不同。有关与特定 C2000 实时微控制器相关的 CLB 的详细信息，请参阅器件特定的 TRM。

2.1 步骤 1：了解设计要求

*任何*成功的 CLB 逻辑设计的关键在于清楚了解设计要求。对于串行接口设计，在 CLB 逻辑设计阶段需要透彻了解总线协议。这包括基本信息（例如信号数量、输入和输出配置）以及帧编码要求（例如起始位、停止位、奇偶校验位和校验和位）。

其次，必须清楚地了解总线接口的时序要求，以确定使用 CLB 实施设计的可行性。应特别注意串行总线运行速度、信号极性以及所需的建立时间和保持时间。

最后，必须识别实施串行总线接口所需的任何支持信号。例如，这包括由片上 PWM 生成以便为移入和移出 CLB 逻辑块的数据计时的输入时钟信号，或者用于触发串行总线上的同步命令发送的周期性计时器中断。

2.2 步骤 2：识别至 CLB 逻辑块的所需输入

在大多数情况下，需要将输入信号传递到 CLB 逻辑块。通常，需要考虑三类信号：

- 通过 GPIO 引脚采样的外部串行总线信号。例如，这些信号包括串行总线时钟、帧和数据信号。
- CLB 逻辑运行中使用的片上外设信号。例如，使用 PWM*n*A 引脚或计时器中断生成的时钟信号。
- 由 CPU 直接控制的 GPREG 位。这些 GPREG 位可用于触发 CLB 逻辑块中的操作或启用/禁用特定功能。

使用 CLB 全局和本地多路复用器与器件上的不同 XBAR 相结合，可以将这些输入信号连接到八个 CLB 逻辑块输入。有关更多信息，请参阅器件专用 TRM。

通常有多条路径可用于将 CLB 逻辑块连接到输入信号。表 2-1 列出了将这些信号引入 CLB 逻辑块边界的推荐路径。

表 2-1. 输入信号的推荐路径

输入信号类型	推荐的输入路径
外部串行总线信号（例如，串行总线时钟、帧和数据信号）	GPIO 引脚 ⇨ CLB 输入 XBAR 1 ⇨ CLB 本地多路复用器 ⇨ CLB 输入
内部片上外设信号（例如，使用 PWM*n*A 生成的时钟信号）	外设信号 ⇨ CLB 全局多路复用器或 CLB 本地多路复用器 ⇨ CLB 输入
由 CPU 直接控制的自定义信号	存储器映射 GPREG 位 ⇨ CLB 输入

1. 并非所有 C2000 实时微控制器上都提供 CLB 输入 XBAR。在这些器件上，可以使用 GPIO XBAR 和 CLB XBAR 将外部信号引入逻辑块边界。有关更多信息，请参阅器件专用 TRM。

Copyright © 2023 Texas Instruments Incorporated

2.2.1 GPIO 输入限定

对外部串行总线信号进行采样时，一个重要的考虑因素是 GPIO 引脚的输入限定设置。表 2-2 总结了可用的不同输入限定设置和潜在用途。

表 2-2. GPIO 输入限定使用

输入限定设置	说明	用途
异步输入（推荐设置）	输入信号不会同步到 SYSCLKOUT。 注意：可启用 CLB 输入级同步（请参阅节 2.2.2）。	用于原封不动地向 CLB 逻辑块传递信号。
同步到 SYSCLKOUT（不推荐）	输入信号同步到 SYSCLKOUT。	在信号到达 CLB 逻辑块边界之前，输入 XBAR 添加了一个小的模拟延迟，这基本上会使信号再次异步到 SYSCLKOUT。因此，在此级别同步只会增加不必要的延迟。建议改为启用 CLB 输入级同步（请参阅节 2.2.2）。
使用采样窗口进行限定（在某些情况下推荐使用）	输入信号同步到 SYSCLKOUT 并限定在指定数量的周期后才允许输入发生变化。	用于去除输入信号中的噪声。然而，在设计 CLB 逻辑时，需要考虑采样窗口引入的额外延迟。 还应启用 CLB 输入级同步，以将信号重新同步到 SYSCLKOUT，因为在信号到达 CLB 逻辑块边界之前，输入 XBAR 会添加一个小的模拟延迟。

要考虑的其他功能：

- GPIO 内部上拉电阻器。如果电路板上没有外部上拉电阻器，则启用此功能以避免悬空 CLB 输入信号。
- GPIO 输入信号反相：启用此功能可在传递到 CLB 之前使 GPIO 引脚上的信号反相。

English Document: SPRAD62
Copyright © 2023 Texas Instruments Incorporated

2.2.2 CLB 输入设置

这八个 CLB 输入具有不同的配置和滤波器选项，必须根据串行总线的开关特性仔细考虑这些选项。表 2-3 和表 2-4 列出了不同的选项及其可能的用途。

表 2-3. CLB 输入配置选项

输入配置设置	说明	用途
异步输入	输入信号不会同步到 SYSCLKOUT。	用于通过 CLB 逻辑块原封不动地将信号直接传递到 CLB 输出。
同步到 SYSCLKOUT（推荐设置）	输入信号同步到 SYSCLKOUT。	通常，需要同步逻辑块中使用的所有 CLB 输入。

表 2-4. CLB 输入滤波器选项

输入滤波器设置	说明	用途
不滤波	输入信号直接传递到 CLB 逻辑块。	用于设计依赖于输入逻辑状态的逻辑。例如，启用和禁用信号。
上升沿检测	当在输入信号上检测到上升沿时，将生成与 CLB 时钟相等的单个脉冲。	用于在串行器模式下将数据移入或移出计数器，或用于对串行时钟沿进行计数。
下降沿检测	当在输入信号上检测到下降沿时，将生成与 CLB 时钟相等的单个脉冲。	用于在串行器模式下将数据移入或移出计数器，或用于对串行时钟沿进行计数。
任何边沿检测	当在输入信号上检测到下降沿或上升沿时，将生成与 CLB 时钟相等的单个脉冲。	用于需要作用于输入信号的两个边沿的逻辑。

可能需要使用不同的输入滤波器设置将单个外部输入映射到多个 CLB 逻辑块输入。这使得部分 CLB 逻辑块逻辑不在外部信号的上升沿运行，使单独逻辑不在外部信号的下降沿运行。本应用报告中给出的两个示例将单个外部信号映射到多个 CLB 输入。

2.3 步骤 3：识别来自 CLB 逻辑的所需输出

在大多数情况下，CLB 逻辑块需要驱动串行总线信号。例如，这些信号包括串行总线时钟、帧和数据信号。每个 CLB 逻辑块有八个输出，可通过 CLB 输出 XBAR 或 GPIO 输出 XBAR 引出至器件引脚。

2.3.1 同步输出信号

CLB 生成的输出信号很可能由 CLB 逻辑块中的不同逻辑块生成。这会导致每个输出信号中存在不同的延迟路径。为了更大限度地增加接收器件的建立时间和保持时间，可以使用总线时钟和简单的触发器使不同的 CLB 输出互相同步，请参阅图 2-1。可使用 CLB 逻辑块中的有限状态机 (FSM) 块来实施触发器。但是，由于 CLB 逻辑块资源有限，最好仅在绝对需要时才使用此选项。

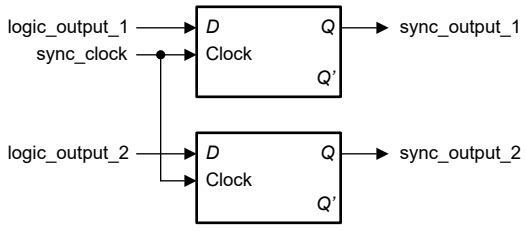

图 2-1. 使用 D 类触发器同步输出

English Document: SPRAD62
Copyright © 2023 Texas Instruments Incorporated

举个需要同步逻辑的例子，请考虑图 2-2 中的仿真，其中显示了两个输入（*in1* 和 *in2*），它们在稍微不同的时间改变状态。这两个信号使用第三个“时钟”信号彼此同步，*in0* 使用的两个边沿触发 D 类触发器通过两个 FSM 来实施。从仿真中可以看出，每个 FSM 使用“时钟”信号 *in0* 锁存并延迟其输入信号。因此，两个 FSM 输出相互同步。

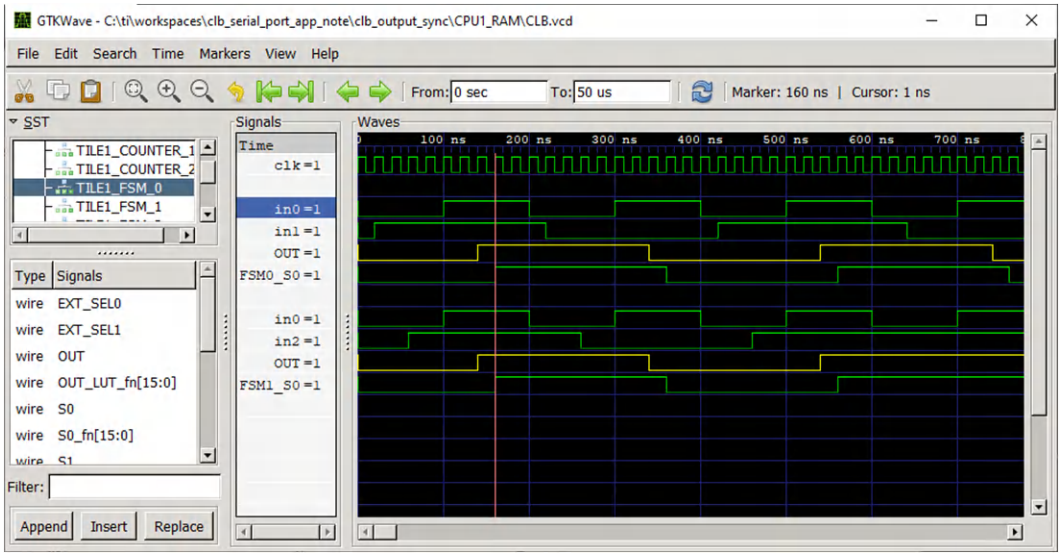

图 2-2. 使用 FSM D 类触发器同步输出

图 2-3 中显示了每个 FSM 输出和状态变量的逻辑方程。请注意，FSM *out* 和 *s0* 均设置为相同的方程，并且任一信号均可用于驱动最终输出。由于 *out* 是纯组合输出，而 *s0* 始终在下一个 CLB 时钟周期更新，因此两个信号之间存在 1 周期延迟。另请注意，在此仿真中，为同步上升沿检测设置 *in0* 输入滤波器。

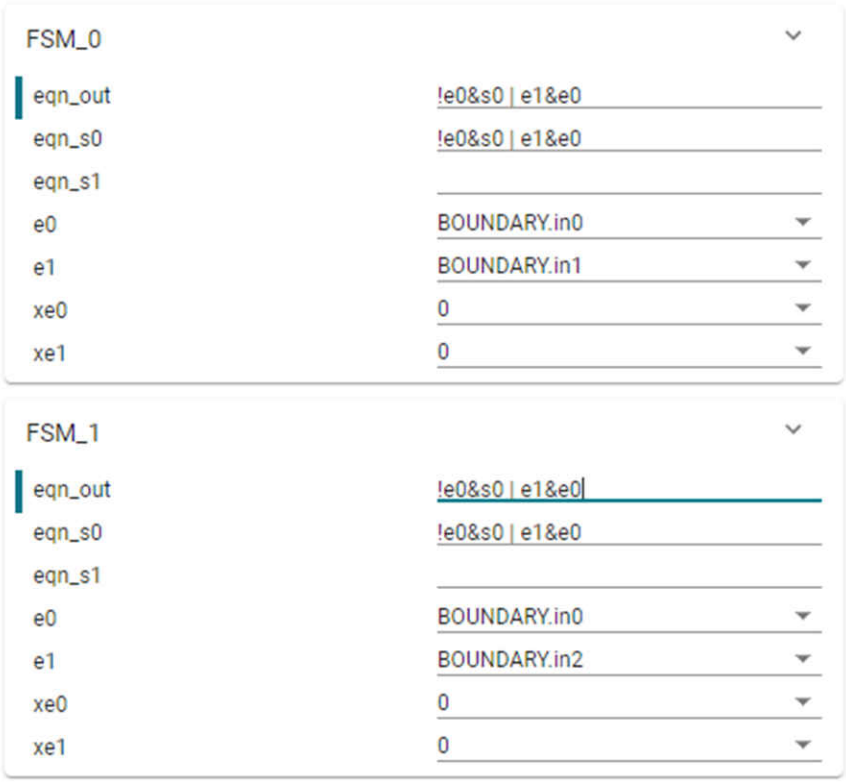

图 2-3. D 类触发器的 FSM 设置

English Document: SPRAD62
Copyright © 2023 Texas Instruments Incorporated

2.3.2 输出信号调节

每个 CLB 输出都可以通过异步输出调节 (AOC) 块进行传递，在该块中，可以对该输出进行反相和门控，然后再传递到输出 XBAR。AOC 可用于使信号反相，然后再传递到输出。

备注

如果 CLB 输入信号（例如串行时钟）需要通过 CLB 逻辑块异步传递到内部 CLB 时钟，则必须使用边界输入 4 或 5 通过 CLB 逻辑块传递输入信号。如果需要，信号必须从逻辑块输入直接传递到 AOC，在这里可以进行反相。此外，必须通过 GPIO 输出 XBAR，而不是 CLB 输出 XBAR，将 AOC 输出传递到微控制器引脚。

2.4 步骤 4：设计 CLB 逻辑

设计 CLB 逻辑是该过程中最复杂的一步。要最终确定设计，需要多次迭代逻辑设计、仿真和测试。

对 CLB 逻辑块进行编程的简单方法是利用 CLB 工具。借助 CLB 工具，用户可以为每个 CLB 逻辑块输入选择输入滤波器选项，连接每个 CLB 块中的子模块，对 FSM 块、高级控制器 (HLC) 和其他逻辑块进行编程，以及轻松地将逻辑块逻辑复制到其他逻辑块。用户还可以使用该工具来设置逻辑块逻辑的仿真。CLB 工具将实时标记逻辑设计中的任何潜在问题，例如不受支持的逻辑连接和无效的 FSM 公式。

CLB 根据使用“SysConfig”图形用户界面 (GUI)，它是 Code Composer Studio™ (CCS) 的一部分。借助 SysConfig，用户还可以配置 MCU 的输入和输出 XBAR 以及对 MCU 外设进行编程。

更多有关 CLB 工具和 SysConfig 的信息，请参阅以下资源：

- CLB 工具用户指南
- 可配置逻辑块 (CLB) 编程工具培训（视频）
- 适用于 C2000 实时 MCU 的 SysConfig 开发工具

2.4.1 资源分配

因为每个 CLB 逻辑块的资源有限，因此需要仔细考虑 CLB 逻辑块的使用。通常，对于串行接口，需要满足以下条件：

- 一个计数器用于数据接收。计数器以串行器模式运行，使 CLB 可以在每个串行时钟沿移入串行位。
- 一个计数器用于数据发送。计数器以串行器模式运行，使 CLB 可以在每个串行时钟沿移出串行位。
- 一个计数器用于串行位计数。计数器在正常计数器模式下运行。
- 需要一个 FSM 块来对接收（和/或发送）字进行计数并向 HLC 触发事件。

虽然可以在一个 CLB 逻辑块中完整实施串行接口，但在某些情况下，可能需要使用两个 CLB 逻辑块，因此建议在单独的逻辑块中实施接收和发送功能，以简化逻辑设计。

有些串行接口需要特殊的帧编码位，例如起始位、停止位和奇偶校验位。CLB 可以支持这些功能，但它需要额外的 CLB 逻辑块。如果没有所需的 CLB 块，请考虑利用 C28x CPU 在接收和发送数据中提取或添加上述位。

2.4.2 在 CLB FIFO 和 MCU RAM 之间交换数据

基于 CLB 的串行接口设计要求在（1. CLB 串行器和 CLB FIFO）以及（2. CLB FIFO 和 MCU RAM）之间移动数据。CLB 逻辑块中的 HLC 可以在串行器和 FIFO 之间移动数据。但是，HLC 无法在 FIFO 和 MCU RAM 之间直接移动数据，因为它不是 MCU 架构中的主器件。截至编写本应用手册时，C28x CPU 和 CLA 是唯一能够将数据移入和移出 CLB FIFO 的主器件，因此 DMA 不能用于此数据移动。

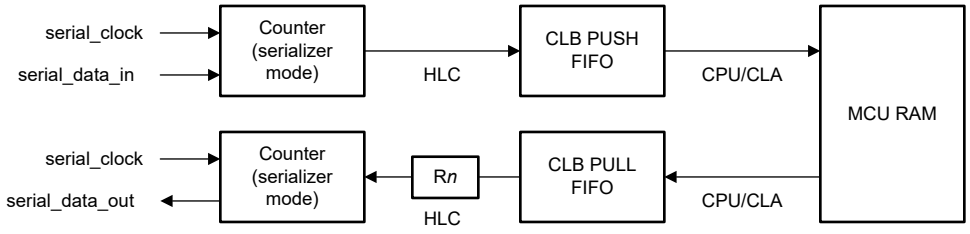

图 2-4. 数据移动流

Copyright © 2023 Texas Instruments Incorporated

备注

HLC 无法将数据直接拉入 CLB 计数器。相反，它必须使用中间步骤将数据拉入其中一个 HLC 寄存器 (R*n*)，然后将数据移动到计数器。

CPU 可以使用 CLB 本地写入接口总线直接加载计数器和 HLC 寄存器。这对于设置 CLB 逻辑块的初始状态很有用。例如，在节 4 中，使用 CLB 为照明应用中的 LED 驱动器实施定制通信总线，在启用 CLB TX 逻辑块进行发送之前，在计数器中加载发送字总数。以下代码显示了如何使用 C2000ware driverlib 函数执行此操作的示例。有关 CLB 本地写入接口总线的信息，请参阅器件特定的 TRM。

```
CLB_writeInterface(TX_TILE_BASE, CLB_ADDR_COUNTER_1_LOAD, 0xFFFFFFFF);
CLB_writeInterface(TX_TILE_BASE, CLB_ADDR_HLC_R2, 0x0);
```

2.4.3 CLB 逻辑状态和触发标志

使用 CLB 设计串行接口时，通常需要实施状态/标志位和触发位。

2.4.3.1 状态/标志位

状态/标志位让 CPU 可以轮询 CLB 逻辑的状态。例如，在 CLB 中断之后，CPU 可以轮询标志位以确定中断是由数据接收中断还是接收错误中断引起。

可以使用 HLC 寄存器来实施这些状态/标志位。但是，每个标志必须使用一个 HLC 寄存器，因为 HLC 不支持按位与和或指令。例如，HLC R0 寄存器中的非零值可用于指示接收中断，而 R1 中的非零值可用于指示接收错误。

备注

HLC “INTR <tag>” 指令和 CLB_INTR_TAG_REG 不能用于标志，因为此指令的后续使用会覆盖任何先前的标志值。

也可以通过直接读取不同 CLB 逻辑块的状态来确定 CLB 逻辑状态。例如，如果 FSM 配置为定义多个状态（例如 IDLE 和 ACTIVE），CPU 可以读取 S0 和 S1 的状态来判断 CLB 逻辑状态。

CPU 可以使用 CLB 存储器映射调试寄存器 CLB_DBG_R*n*、CLB_DBG_C*n* 和 CLB_DBG_OUT 来确定 CLB 内不同块的状态。以下代码块中显示了使用 HLC 寄存器作为状态位的示例。代码块使用 C2000ware driverlib 函数。

```
uint32_t chkIntFlag = CLB_getRegister(CCSI2_RX_TILE_BASE, CLB_REG_HLC_R1);
uint32_t rcvIntFlag = CLB_getRegister(CCSI2_RX_TILE_BASE, CLB_REG_HLC_R2);
uint32_t endIntFlag = CLB_getRegister(CCSI2_RX_TILE_BASE, CLB_REG_HLC_R3);

    ...

    // Receive interrupt
    if (rcvIntFlag & 0x1)
    {
        ...
        // Clear the receive interrupt flag register
        CLB_writeInterface(CCSI2_RX_TILE_BASE, CLB_ADDR_HLC_R2, 0x0);
    }

    // End interrupt
    if (endIntFlag & 0x1)
    {
        ...
        // Clear the end interrupt flag register
        CLB_writeInterface(CCSI2_RX_TILE_BASE, CLB_ADDR_HLC_R3, 0x0);
    }

    // Check error interrupt
    if (chkIntFlag & 0x1)
    {
        ...
        // Clear the check error interrupt flag register
        CLB_writeInterface(CCSI2_RX_TILE_BASE, CLB_ADDR_HLC_R1, 0x0);
    }
```


English Document: SPRAD62
Copyright © 2023 Texas Instruments Incorporated

2.4.3.2 触发位

触发位使 CPU 可以启用或禁用逻辑块中的逻辑或触发特定操作。例如，可以实施接收器启用位，以便在运行时选择性地启用或禁用 CLB 中的接收逻辑。这些触发位可使用 CLB_GP_REG 寄存器来实施。

CLB_GP_REG 位可以直接连接到八个 CLB 逻辑块输入。CPU 可以设置 CLB_GP_REG 中的任何位来触发 CLB 逻辑中的操作。例如，可以实施接收器启用/禁用位，以便在运行时启用或禁用 CLB 中的接收逻辑。以下代码块中显示了启用/禁用位实施的示例。代码块使用 C2000ware driverlib。

```
#define GPREG_ENABLE_RCVR           3U

void CCSI_HAL_enableClbReceiver()
{
    uint32_t gpRegVal = CLB_getGPREG(CCSI1_RX_TILE_BASE);

    // First check that receiver is in IDLE state
    while(HWREG(CCSI1_RX_TILE_BASE + CLB_LOGICCTL + CLB_O_DBG_OUT) &
            (CLB_DBG_OUT_FSM0_S1 | CLB_DBG_OUT_FSM0_S0)) {}

    // Enable receiver
    gpRegVal |= (1U << GPREG_ENABLE_RCVR);
    CLB_setGPREG(CCSI1_RX_TILE_BASE, gpRegVal);
}
void CCSI_HAL_disableClbReceiver()
{
    uint32_t gpRegVal = CLB_getGPREG(CCSI1_RX_TILE_BASE);

    // First check that receiver is in IDLE state
    while(HWREG(CCSI1_RX_TILE_BASE + CLB_LOGICCTL + CLB_O_DBG_OUT) &
            (CLB_DBG_OUT_FSM0_S1 | CLB_DBG_OUT_FSM0_S0)) {}

    // Disable receiver
    gpRegVal &= ~(1U << GPREG_ENABLE_RCVR);
    CLB_setGPREG(CCSI1_RX_TILE_BASE, gpRegVal);
}
```

备注

CLB 输入必须专用于逻辑设计中使用的任何 CLB_GP_REG 位。鉴于 CLB 输入数限制为八个，所以应审慎地使用 CLB_GP_REG 位。

Copyright © 2023 Texas Instruments Incorporated

2.5 步骤 5 ：仿真逻辑设计

由于设计中使用了不同的逻辑块和互连，串行接口设计很快会变得非常复杂。为了方便 CLB 逻辑设计的调试，CLB 工具生成了一个互连方框图 (图 2-5) 和一个仿真波形 (图 2-6)。有关如何生成这两个文件的更多详细信息，请参阅 *CLB 工具用户指南*。

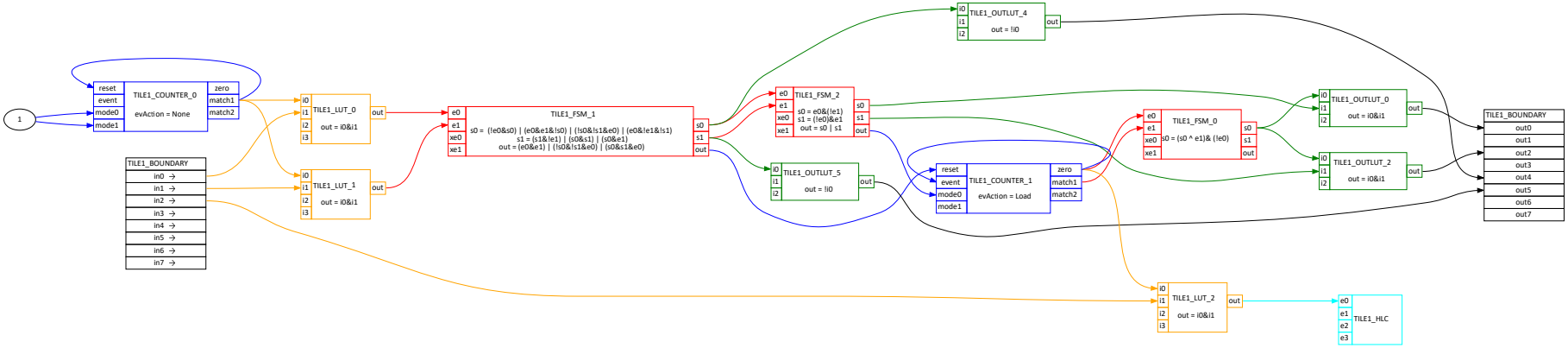

图 2-5. 示例互连方框图

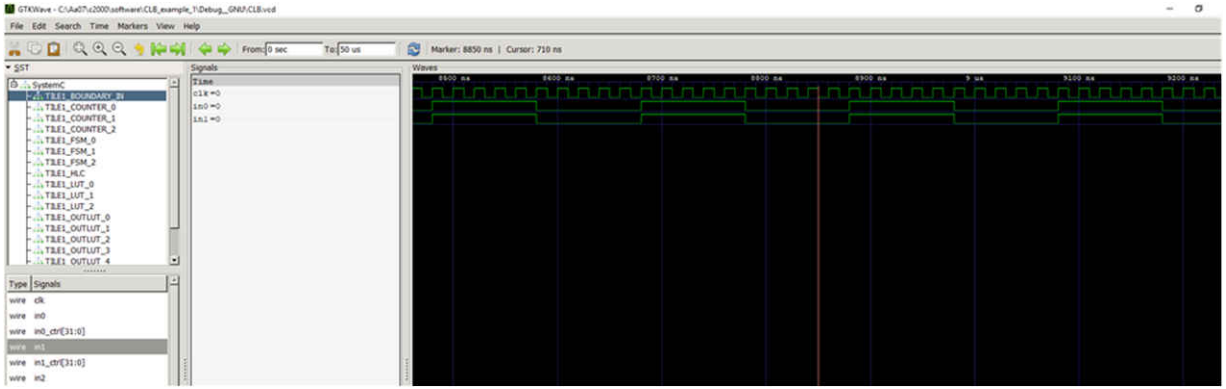

图 2-6. CLB 仿真示例

English Document: SPRAD62
Copyright © 2023 Texas Instruments Incorporated

2.6 步骤 6 ：测试 CLB 逻辑

设计过程的最后一步是通过在预期系统上运行设计来测试整个设计。

建议从可控测试设置开始，以便在移入目标应用之前对逻辑设计进行全面的特性描述。测试设置应包括 CLB 逻辑功能的完整测试覆盖。这包括不同的数据模式和时钟频率，以及引入刻意错误条件的机制，尤其是在 CLB 逻辑设计用于检测错误的情况下。

通过调整仿真器激励并查看 CLB 逻辑块内的信号，CLB 仿真器可用于调试任何观察到的测试故障。修改 CLB 逻辑后，再次编译并运行测试。

CAUTION

请记住，在室温下运行的单个系统上所获得的性能并不能保证在系统所针对的所有工作条件下也能获得同样的性能。重要的是确定所有关键设计参数并确保足够的测试覆盖范围和时序裕度，以保证逻辑设计能够满足这些参数并具有足够的裕度。

3 示例 A：在音频应用中使用 CLB 输入和输出 TDM 流

3.1 示例概述

音频应用经常使用时分多路复用法在系统中的器件之间发送和接收音频数据。在这个多路复用方案中，一个帧分成几个固定长度的时隙或通道。由于是将按时间分割的多个通道多路复用到一个发送通道中，所以该多路复用方案称为时分多路复用 (TDM)。

有些 C2000 实时微控制器上提供的多通道缓冲串行端口 (McBSP) 支持单个 TDM 流的输入和输出。但是，并非所有 C2000 实时微控制器上都提供 McBSP 外设。此外，C2000 实时微控制器中通常包含的其他串行端口外设均不支持 TDM 协议。

在此示例中，CLB 用于输入 8 通道 TDM 流 (TDM-8)，并输出相应的 TDM-8 流。C28x CPU 将接收到的数据从 CLB FIFO 移动到内部 RAM，并将发送数据从 RAM 移动到 CLB FIFO。此示例使用了单个 CLB 逻辑块来实现。输入和输出 TDM-8 流使用外部 12.288MHz 时钟来计时。

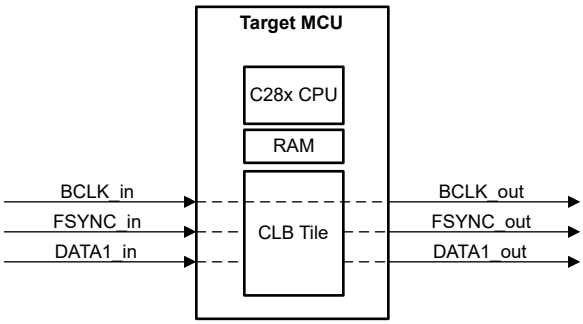

图 3-1. CLB TDM-8 示例

English Document: SPRAD62
Copyright © 2023 Texas Instruments Incorporated

3.2 步骤 1：了解设计要求

需要 CLB 逻辑块逻辑来输入和输出具有图 3-2 中所示时钟、帧同步和数据时序的 8 通道 TDM 流。

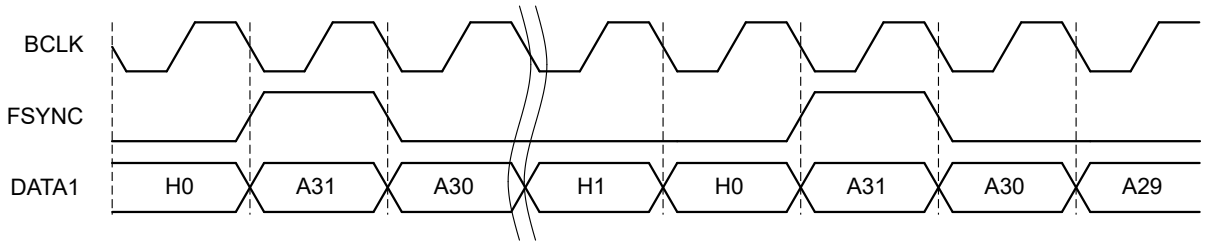

图 3-2. 所需的 TDM 流帧同步脉冲、时钟信号和数据时序

具体而言，需要满足以下条件：

- 外部固定 12.288MHz 串行位时钟（位时钟传递到输出）
- 1 个时钟帧同步脉冲宽度，无延迟
- 帧同步脉冲为高电平有效
- 发送/接收在时钟的上升沿采样的数据
- 每个 TDM 帧 8 个通道
- 每个通道 32 位字长

只需使用 GPIO 反相逻辑在信号到达 CLB 逻辑块之前使信号进行反相，即可轻松支持其他帧同步和时钟极性。

3.3 步骤 2：识别至 CLB 逻辑块的所需输入

输入 TDM-8 流信号通过 CLB 输入 XBAR 路由到 CLB 逻辑块边界输入，如图 3-3 所示。

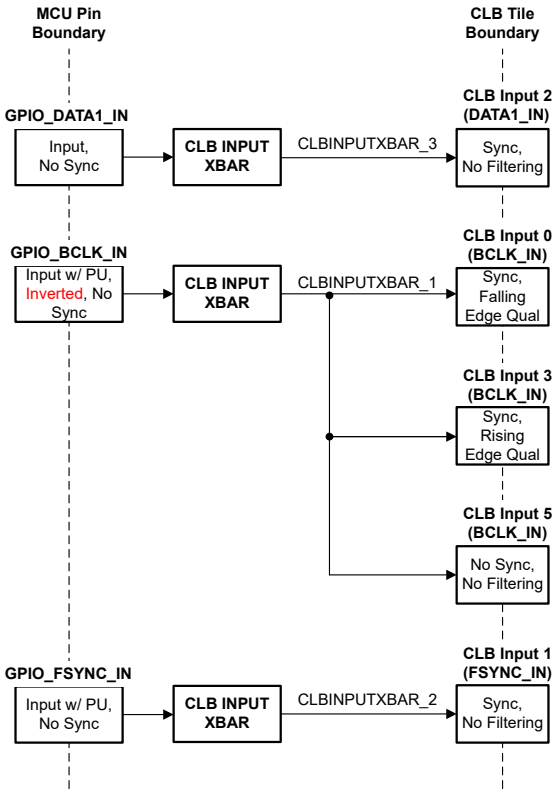

图 3-3. TDM-8 的 CLB 逻辑块输入示例

用于输入 TDM 流的所有 GPIO 均配置为异步操作。根据需要在 CLB 逻辑块输入边界启用同步。此外，在 BCLK_IN 和 FSYNC_IN GPIO 上启用内部上拉以避免输入引脚悬空。

English Document: SPRAD62
Copyright © 2023 Texas Instruments Incorporated

备注

CLB 逻辑的设计时钟极性与图 3-2 中所示的时钟极性相反，因此信号是 BCLK_IN 的反相。如需更多信息，请参阅 节 3.5。

BCLK_IN 信号路由到三个具有不同滤波配置的不同逻辑块输入。

- CLB 输入 0：此输入用于在 BCLK_IN 的下降沿触发数据接收。
- CLB 输入 3：此输入用于在 BCLK_IN 的上升沿触发数据发送。
- CLB 输入 5：此输入上禁用了滤波和同步，以便支持 BCLK 直通要求。

必须特别考虑将 BCLK_IN 信号直通到 BCLK_OUT 输出的要求。因为两个时钟不是彼此的倍数，将输入 12.288MHz BCLK 信号同步到 100MHz CLB 内部时钟会在产生的输出 BCLK 信号中引入抖动。

3.4 步骤 3：识别来自 CLB 逻辑的所需输出

输出 TDM-8 流信号使用 CLB 输出 XBAR 和 GPIO 输出 XBAR 连接到 CLB 逻辑块边界输出，如图 3-4 所示。

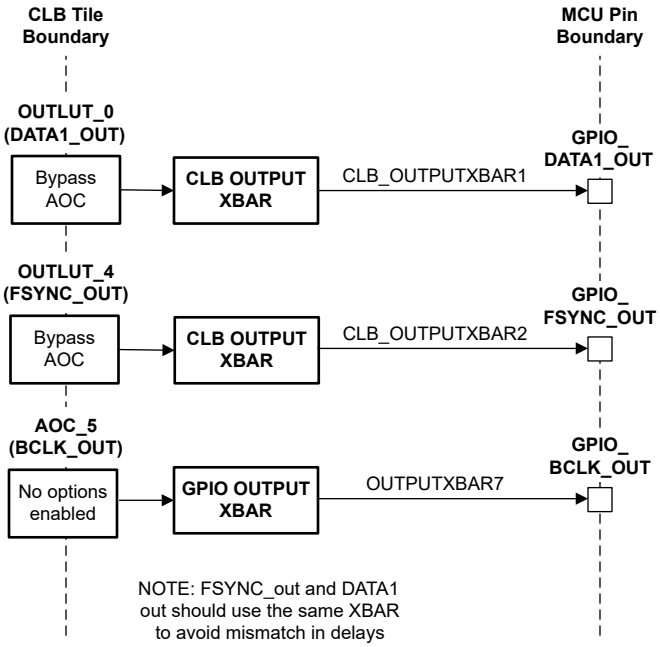

图 3-4. TDM-8 的 CLB 逻辑块输出示例

如前所述，输入 BCLK 信号必须通过 CLB 逻辑块以不寄存的方式传递（即它不同步到任何内部时钟），以避免在输出 BCLK 信号中引入抖动。只有 CLB 输出 4 和 5 可以通过 CLB 逻辑块以不寄存的方式传递。因此，CLB 输出 5 用于 BCLK_OUT。此外，BCLK 信号通过 GPIO 输出 XBAR 引出至 GPIO 引脚。

Copyright © 2023 Texas Instruments Incorporated

3.5 步骤 4：设计 CLB 逻辑

最终的 CLB 逻辑块逻辑设计如图 3-5 所示。

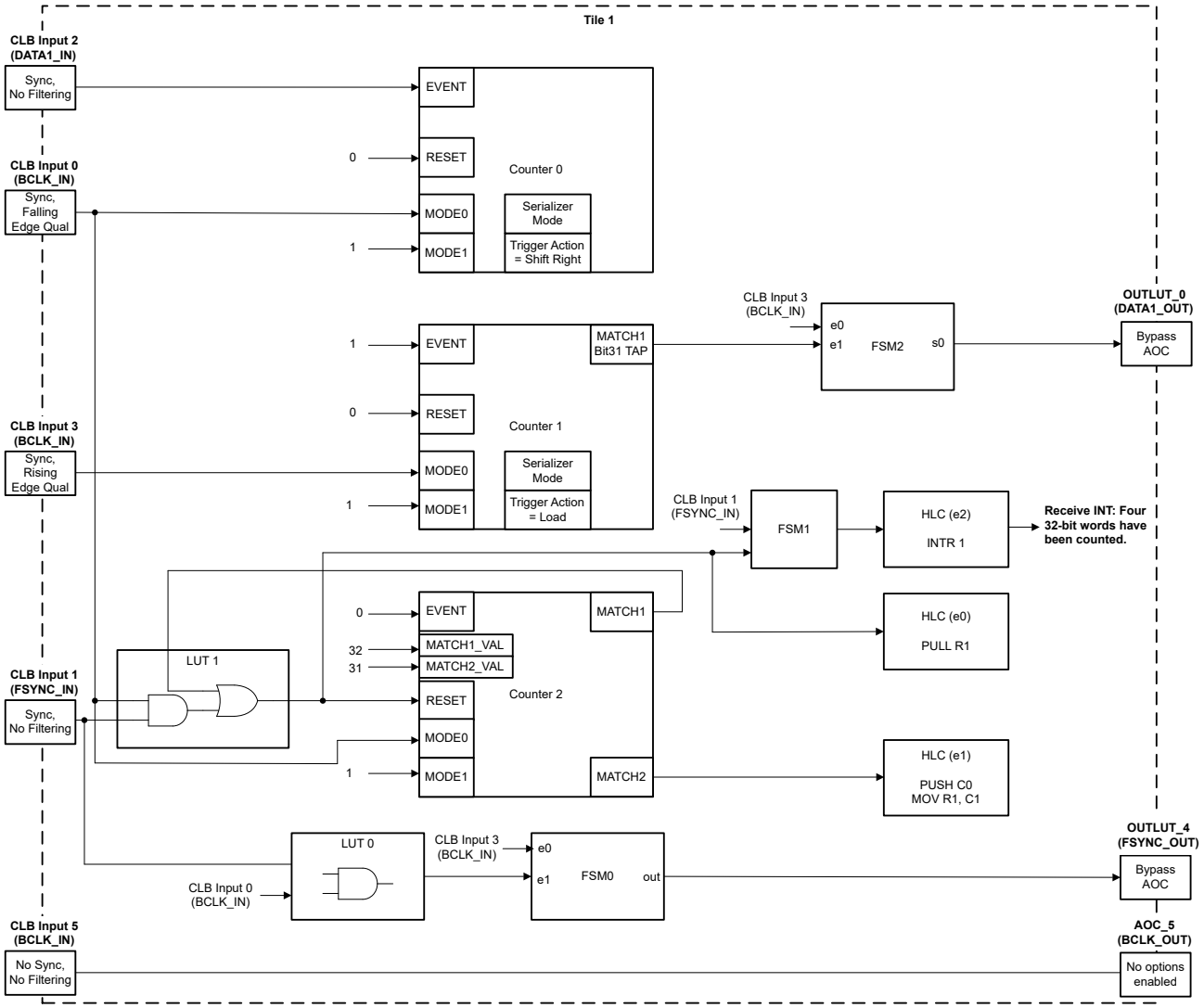

图 3-5. TDM-8 的 CLB 逻辑块逻辑示例

使用图 3-6 中所示的 TDM 时钟、帧同步和数据时序，以使 CLB 逻辑设计更简单。MCU GPIO 引脚的反相功能可用于支持相反的时钟和帧同步极性。

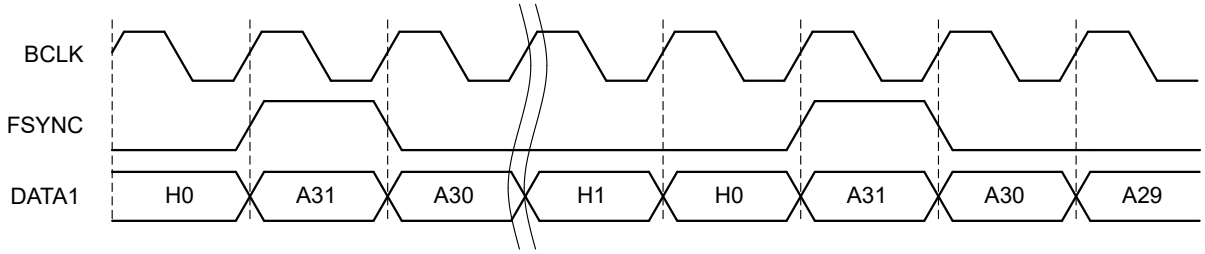

图 3-6. CLB 逻辑设计中的假设 TDM 流帧同步脉冲、时钟信号和数据时序

English Document: SPRAD62
Copyright © 2023 Texas Instruments Incorporated

3.5.1 资源分配

CLB 逻辑块资源分配如下：

- COUNTER0 用于接收输入数据。它在串行器模式下运行。DATA_IN 在 BCLK_IN 下降沿移入。
- COUNTER1 用于输出发送数据。它在串行器模式下运行。COUNTER1 在 BCLK_IN 上升沿移入。
- COUNTER2 用于对 BCLK_IN 上的下降沿进行计数。它在正常计数器模式下运行。当检测到 FSYNC + BCLK（低沿）条件或位计数达到 32 时，它会复位。
- LUT1 用于复位 FSYNC_IN 和字边界上的 COUNTER2。FSYNC_IN 与 BCLK_IN 下降沿进行逻辑与运算，确保只在 FSYNC_IN 有效时对其采样。LUT1 还会触发 HLC 从 PULL FIFO 中拉取新数据。
- FSM1 用于在接收/发送 4 个 32 位字后向 HLC 生成事件。当检测到 FSYNC_IN 时，对其计数进行复位。
- 当检测到 FSYNC_IN 时，LUT0 和 FSM0 用于生成 FSYNC_OUT 信号。FSYNC_OUT 信号与 BCLK_IN 上升沿对齐。请注意，FSYNC_OUT 不是 FSYNC_IN 的直通信号，因为这不是设计要求。
- FSM2 用于将输出 DATA1_OUT 信号与 BCLK_IN 上升沿对齐。
- HLC 用于在 CLB 推挽 FIFO 和两个串行器计数器（COUNTER0 和 COUNTER1）之间移动数据。当接收/发送 4 个 32 位字时，还会向 CPU 生成一个中断。

3.5.2 TDM 字计数器

FSM1 用于在接收/发送四个 32 位字后触发 HLC 事件。图 3-7 和表 3-1 显示了 FSM1 状态图和真值表。

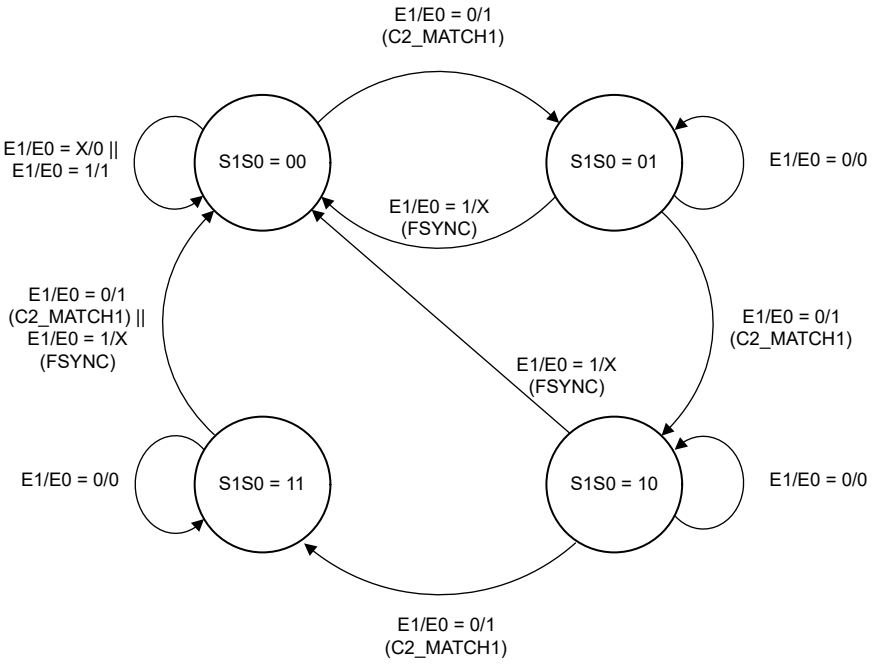

图 3-7. TDM-8 的 FSM1 状态图示例

Copyright © 2023 Texas Instruments Incorporated

表 3-1. TDM-8 的 FSM1 真值表示例

S1	S0	E1 (FSYNC)	E0 (C2_MATCH1)	S1 次态	S0 次态	输出 (HLC_INT)
0	0	0	0	**0**	**0**	**0**
0	0	0	1	**0**	**1**	**0**
0	0	1	0	**0**	**0**	**0**
0	0	1	1	**0**	**0**	**0**
0	1	0	0	**0**	**1**	**0**
0	1	0	1	**1**	**0**	**0**
0	1	1	0	**0**	**0**	**0**
0	1	1	1	**0**	**0**	**0**
1	0	0	0	**1**	**0**	**0**
1	0	0	1	**1**	**1**	**0**
1	0	1	0	**0**	**0**	**0**
1	0	1	1	**0**	**0**	**0**
1	1	0	0	**1**	**1**	**0**
1	1	0	1	**0**	**0**	**1**
1	1	1	0	**0**	**0**	**1**
1	1	1	1	**0**	**0**	**1**

简化的逻辑方程为：

- S1 (NEXT) = S1&!S0&!E1 | S1&!E1&!E0 | !S1&S0&!E1&E0
- S0 (NEXT) = !S0&!E1&E0 | S0&!E1&!E0
- OUTPUT = S1&S0&E0 | S1&S0&E1

3.5.3 FSYNC 和 DATA1 输出同步

鉴于 CLB 逻辑块中存在内部数据移动延迟，DATA1_OUT 信号与传入的 BCLK_IN 和 FSYNC_IN 信号相比，始终存在延迟（请参阅图 3-10）。为了生成彼此对齐的 FSYNC_OUT 和 DATA1_OUT 信号，使用了两个 FSM 来锁存和保持这些信号。在 BCLK_IN 的上升沿更新输出锁存。

FSM0 用于检测和延迟 FSYNC_IN。这意味着 FSYNC_IN 信号不会直接通过 FSYNC_OUT 信号传递。状态图如图 3-8 所示，相应的真值表如表 3-2 所示。

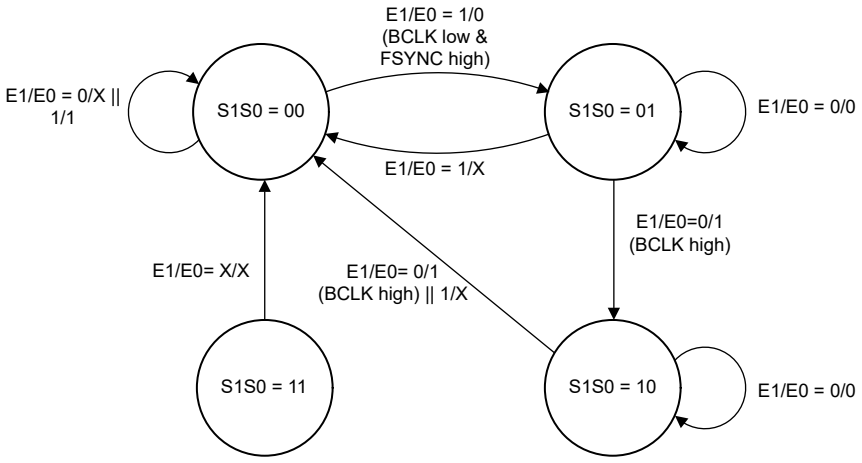

图 3-8. TDM-8 的 FSM0 状态机示例

English Document: SPRAD62
Copyright © 2023 Texas Instruments Incorporated

表 3-2. TDM-8 的 FSM0 真值表示例

S1	S0	E1 （FSYNC 高电平和 BCLK 低电平）	E0 （BCLK 上升）	S1 次态	S0 次态	OUT
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	1	0
0	0	1	1	0	0	0
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	0	0	0
0	1	1	1	0	0	0
1	0	0	0	1	0	1
1	0	0	1	0	0	1
1	0	1	0	0	0	1
1	0	1	1	0	0	0
1	1	0	0	0	0	0
1	1	0	1	0	0	0
1	1	1	0	0	0	0
1	1	1	1	0	0	0

简化的逻辑方程为：

- S1 (NEXT) = !E1&!E0&S1&!S0 | !E1&E0&!S1&S0
- S0 (NEXT) = !E1&!E0&!S1&S0 | E1&!E0&!S1&!S0
- OUT = !E1&S1&!S0 | !E0&S1&!S0

FSM2 用于实施简单的 D 类触发器，该触发器将锁存并延迟 DATA1_OUT 信号。真值表如表 3-3 所示。请注意，在表 3-3 中，S0 (next) 用于驱动最终的 DATA1_OUT 信号。

表 3-3. TDM-8 的 FSM2 真值表示例

E1 (DATA1)	E0 （BCLK 上升）	S0	S0 (next)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

简化的逻辑方程为：

- S0 (NEXT) = !E0&S0 | E1&E0

Copyright © 2023 Texas Instruments Incorporated

3.6 步骤 5 ：仿真逻辑设计

图 3-9 中仿真了数据接收操作。对于此仿真，使用 0xAAAA AAAA 的简单模式作为数据输入。仿真显示了 TDM 帧中最后一个字的最后一位与新帧开始之间的转换。

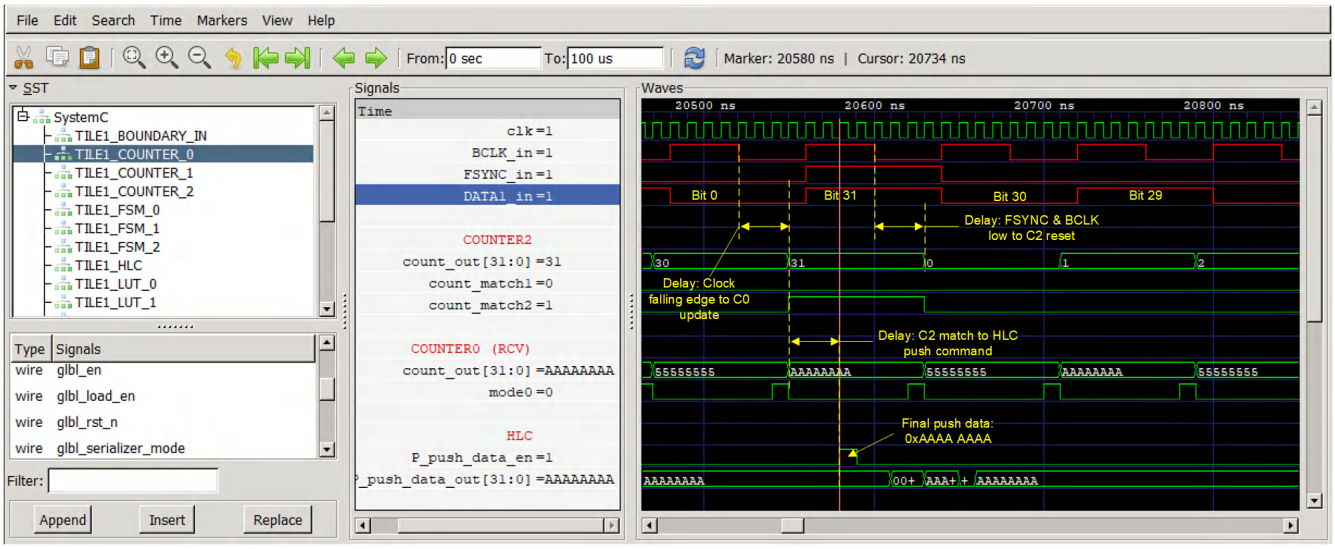

图 3-9. TDM-8 数据接收仿真

图 3-10 中仿真了数据发送操作。仿真显示了 TDM 帧中最后一个字的最后一位与新帧开始之间的转换。对于此仿真，持续发送 0x5555 5554 的模式以强调串行字之间串行器输出端可能出现短暂干扰。由于串行器必须在对其计数器值移位*后*加载，以便避免丢失发送字中的最高有效位，因此串行器输出会有一小段时间无效。

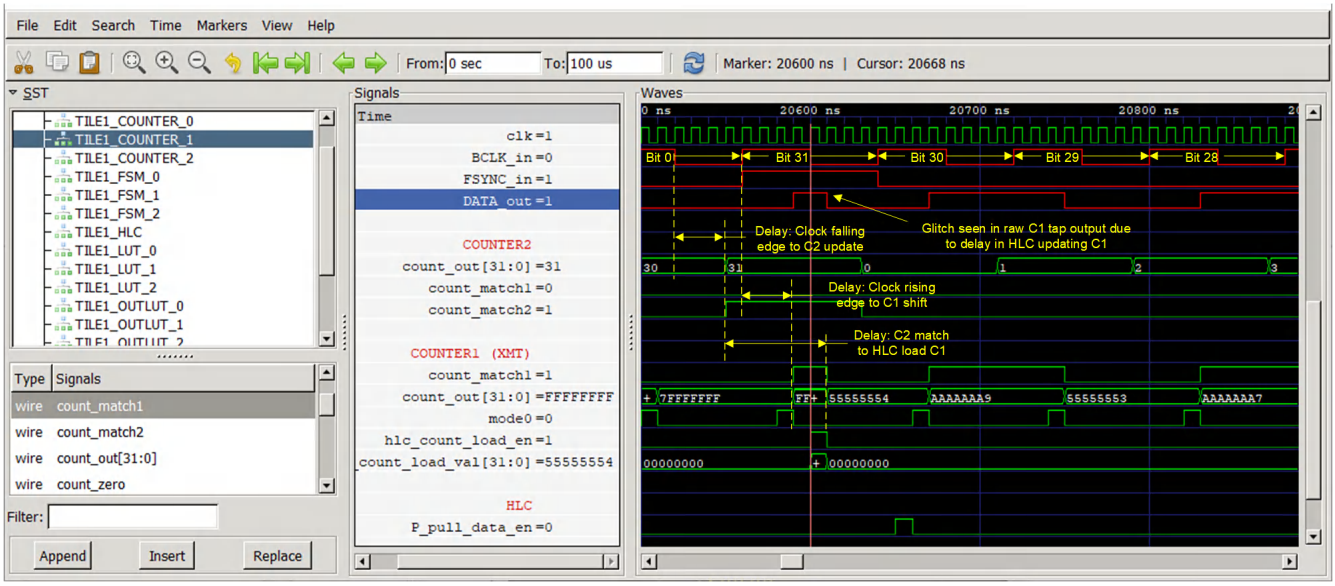

图 3-10. TDM-8 数据发送仿真

English Document: SPRAD62
Copyright © 2023 Texas Instruments Incorporated

请注意，在此 TDM 示例中，HLC 已配置为将 C0 中的值推入 FIFO，然后将新值加载到 C1（请参阅图 3-5）。这是有意为之，是为了确保在 BCLK_IN 的上升沿*之后*加载 C1。由于 HLC 操作在 BCLK 的下降沿触发，且串行器操作依赖于 BCLK 的上升沿，因此 BCLK_IN 周期存在固有依赖关系。如果延长了 BCLK_IN 周期（降低 BCLK_IN 频率或增大 CLB 时钟频率），则存在 C1 更新发生在时钟上升沿之前的风险，这会导致串行字中最高有效位丢失。

如图 3-10 所示，输出数据与输入 FSYNC 和 BCLK 之间存在延迟。为了输出彼此同步的 FSYNC 和 DATA1 信号，使用两个 FSM 来锁存和延迟这两个信号。此功能的仿真结果如图 3-11 所示。

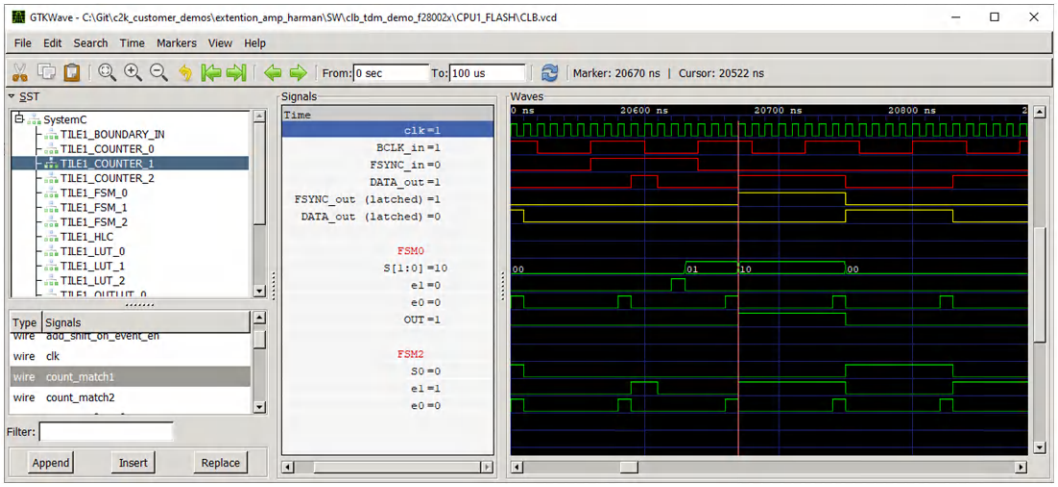

图 3-11. FSYNC 和 DATA1 同步

3.7 步骤 6 ：测试 CLB 逻辑

本节说明了如何测试为基于 CLB 的 TDM-8 示例提供的 Code Composer Studio™ 示例工程。我们在 F280025C controlCARD (TMDSCNCD280025C) 上测试了该示例。但是，代码可轻松移植到任何其他支持 CLB 类型 2 及更高版本的 MCU。

出于测试目的，使用单独的 F28388D MCU 通过 McBSP 端口生成测试 TDM-8 流。F28388D 还接收来自 CLB 的 TDM-8 流并检查错误。此测试使用 F28388D controlCARD (TMDSCNCD28388D)。

在此测试中，F28388D 配置为在 200MHz 下运行，McBSP 在 12.5MHz 下生成 TDM 波形。F280025C 配置为在最大 100MHz 下运行。

图 3-12 显示了测试 TDM-8 示例所需的设置。

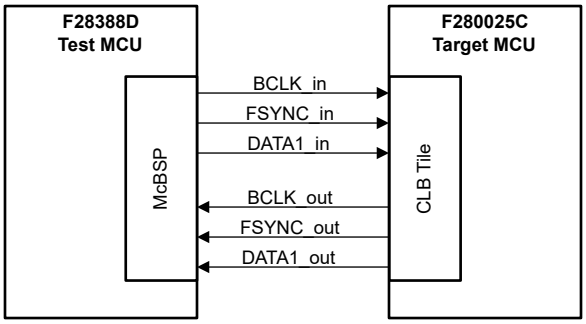

图 3-12. TDM 示例的测试设置

Copyright © 2023 Texas Instruments Incorporated

3.7.1 硬件设置和连接

运行演示需要以下硬件：

- TMDSCNCD280025C controlCARD + TMDSHSECDOCK 基板扩展坞 + 5V 电源
- TMDSCNCD28388D controlCARD + TMDSHSECDOCK 基板扩展坞+ 5V 电源

可选

- 逻辑分析仪（用于查看 TDM 总线信号）

按如下方式设置硬件：

1. 将 controlCARD 插入相应的 TMDSHSECDOCK。两个 controlCARD 都应设置为其默认设置（请参阅每个 EVM 的相应用户指南）。
2. 将 F28388D ***输出*** TDM 流连接到 F280025C ***输入*** TDM，如表 3-4 和表 3-5 所示。

表 3-4. F280025C CLB 逻辑输入 TDM 信号

TDM 输入引脚	GPIO（焊球）	DOCK 引脚
FSYNC_IN	GPIO00	49
BCLK_IN	GPIO01	51
DATA1_IN	GPIO02	53

表 3-5. F28388D McBSP 输出 TDM 信号

TDM 输入引脚	GPIO（焊球）	DOCK 引脚
MCLKX/BCLK	GPIO22	72
MFSX/FSYNC	GPIO23	74
MDX/DATA1	GPIO20	68

3. 将 F280025C ***输出*** TDM 流连接到 F28388D ***输入*** TDM，如表 3-6 和表 3-7 所示。

表 3-6. F280025C CLB 逻辑输出 TDM 信号

TDM 输出引脚	GPIO（焊球）	DOCK 引脚
FSYNC_OUT	GPIO04	50
BCLK_OUT	GPIO05	52
DATA1_OUT	GPIO06	54

表 3-7. F28388D McBSP 输入 TDM 信号

TDM 输入引脚	GPIO（焊球）	DOCK 引脚
MCLKR/BCLK	GPIO58	108
MFSR/FSYNC	GPIO59	110
MDR/DATA1	GPIO21	70

4. 在两个扩展坞之间连接几个常见的 GND。
5. 将相应的 USB 电缆连接到每个 controlCARD。
6. 将 5V 电源连接到每个 TMDSHSECDOCK（也可以使用单独的 USB 电缆为扩展坞供电）。
7. 将每个 TMDSHSECDOCK 上的 S1 设置为“EXT_ON”位置。

Copyright © 2023 Texas Instruments Incorporated

3.7.2 软件设置

C2000ware 中提供了两个 CCS 工程来测试此 CLB 示例。TDM 工程的路径为：

- <C2000WARE_INSTALL>/driverlib/f28002x/examples/clb
- <C2000WARE_INSTALL>/driverlib/f2838x/examples/c28x/mcbsp

构建和运行软件需要以下软件包：

- Code Composer Studio（CCS）版本 11.1.00 或更高版本
- C2000ware 版本 4.3.00.00 或更高版本
- GNU 编译器 (TDM-GCC) 和 GTK Wave 仿真查看器（对于运行 CLB 仿真是可选的）。有关更多信息，请参阅 *CLB 工具用户指南*。

首先，在 CCS 中按照以下步骤设置 F280025C：

1. 在 CCS 菜单中，依次点击“Project -> Import CCS Projects...”。
2. 在“Select search-directory”中输入 CLB 示例工程的路径。
   a. Path：<C2000WARE_INSTALL>/driverlib/f28002x/examples/clb
3. 单击“Refresh”。
4. 选择“**clb_ex31_tdm_serial_port**”工程。

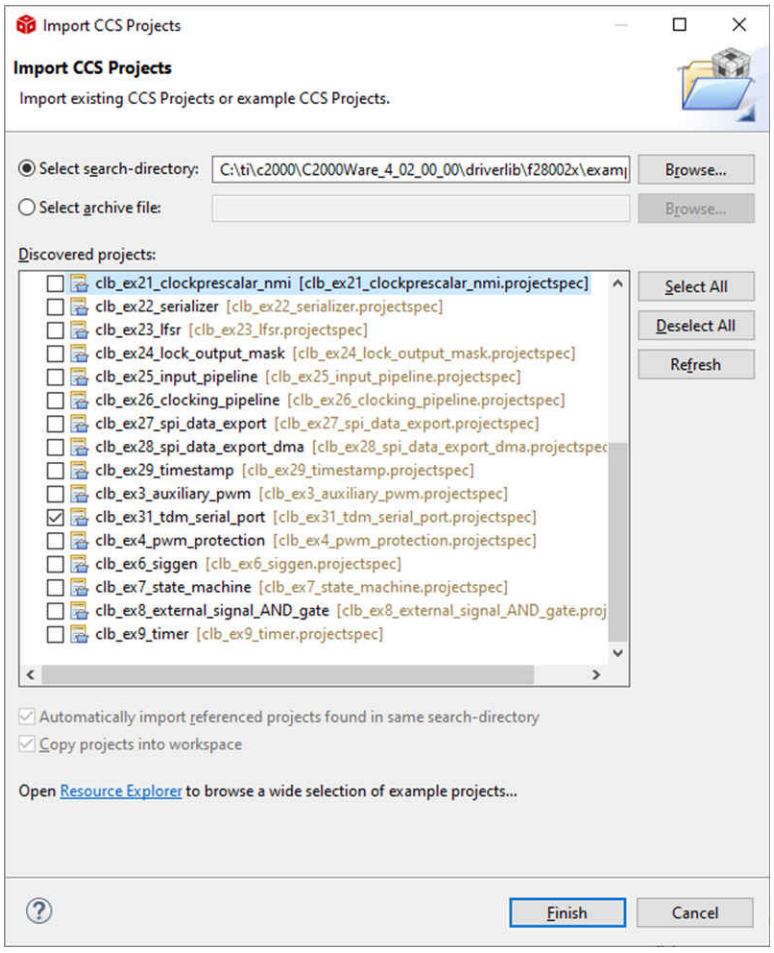

图 3-13. 导入 CLB TDM 示例工程

5. 选中“Copy projects into workspace”。
6. 点击“Finish”将工程导入工作区。

   可选：点击“Project -> Build Configurations -> Set Active -> CPU1_FLASH”构建用于闪存执行的代码。将 TMDSCNCD280025C controlCARD 设置为从闪存引导（请参阅 *TMS320F28388D controlCARD 信息指南*）。

Copyright © 2023 Texas Instruments Incorporated

7. 在 CCS 菜单中，点击“Project -> Build Project”来构建示例工程。
8. 点击“Run -> Debug”将可执行文件加载到 F280025C 目标器件。
9. 最后，在 CCS Debug 透视图中点击“Run -> Resume”以运行代码。

其次，在 CCS 中按照以下步骤操作，将 F28388D 设置为生成测试 TDM 流：

1. 在 CCS 菜单中，依次点击“Project -> Import CCS Projects...”。
2. 在“Select search-directory”中输入 McBSP 示例工程的路径。
   a. Path：<C2000WARE_INSTALL>/driverlib/f2838x/examples/c28x/mcbsp
3. 单击“Refresh”。
4. 选择“**mcbsp_ex7_tdm8_test**”工程。

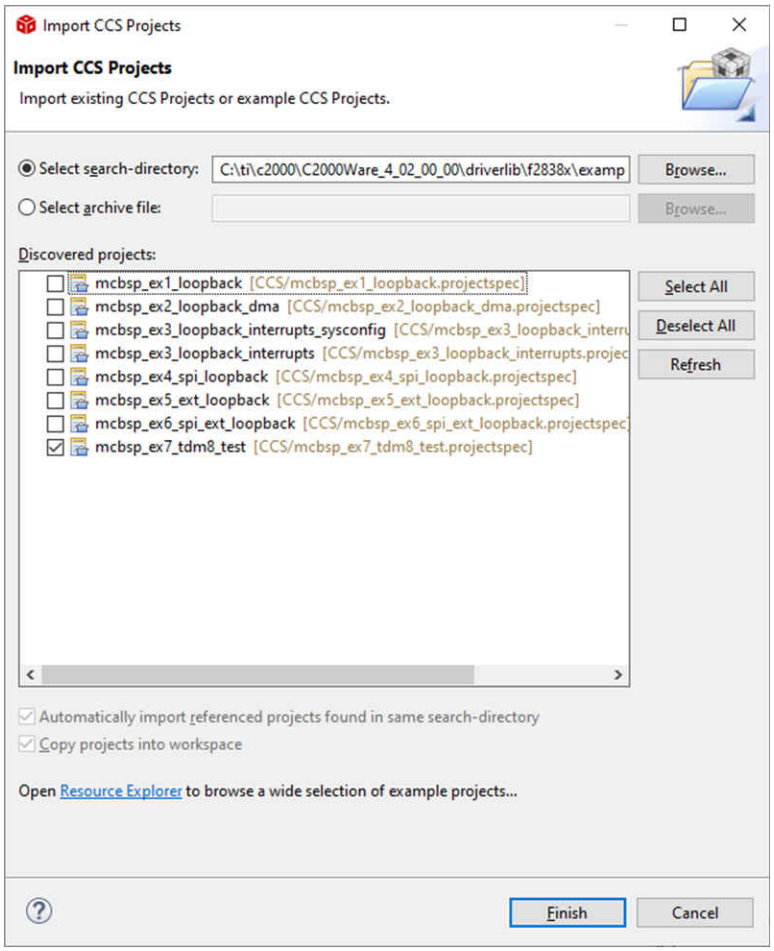

图 3-14. 导入 McBSP TDM 示例工程

5. 选中“Copy projects into workspace”。
6. 点击“Finish”将工程导入工作区。
   a. 可选：点击“Project -> Build Configurations -> Set Active -> CPU1_FLASH”构建用于闪存执行的代码。将 TMDSCNCD28388D controlCARD 设置为从闪存引导（请参阅 controlCARD 用户指南）。
7. 在 CCS 菜单中，点击“Project -> Build Project”来构建示例工程。
8. 点击“Run -> Debug”将可执行文件加载到 F28388D 目标器件。
   a. 可选：将以下全局变量添加到“Expressions”窗口中：txData、rxData、testWordDetected 和 errCountGlobal。
9. 最后，在 CCS Debug 透视图中点击“Run -> Resume”以运行代码。

3.7.3 测试输出建立时间和保持时间

来自 CLB 的输出 TDM 流中的一个关键考虑因素是接收器件看到的预期建立时间和保持时间。为了测量建立时间和保持时间，使用一个示波器来持续捕获 CLB 的输出。图 3-15 显示了未将锁存和延迟逻辑添加到 FSYNC_OUT

English Document: SPRAD62
Copyright © 2023 Texas Instruments Incorporated

和 DATA1_OUT 信号的 TDM 输出。由于 DATA1_OUT 信号中存在延迟，相对于 BCLK_OUT，DATA1_OUT 的建立时间和保持时间会减少。

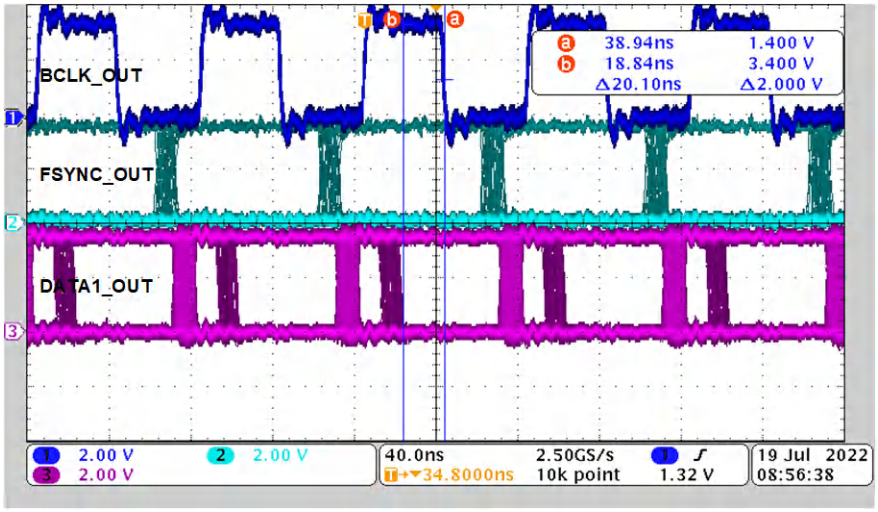

图 3-15. 无锁存和延迟逻辑的 TDM-8 输出

图 3-16 中显示的示波器图显示了最终的 TDM 输出，其中的锁存和延迟逻辑已添加到 FSYNC_OUT 和 DATA1_OUT 信号中。使用这种方法可以大大改善建立时间和保持时间。

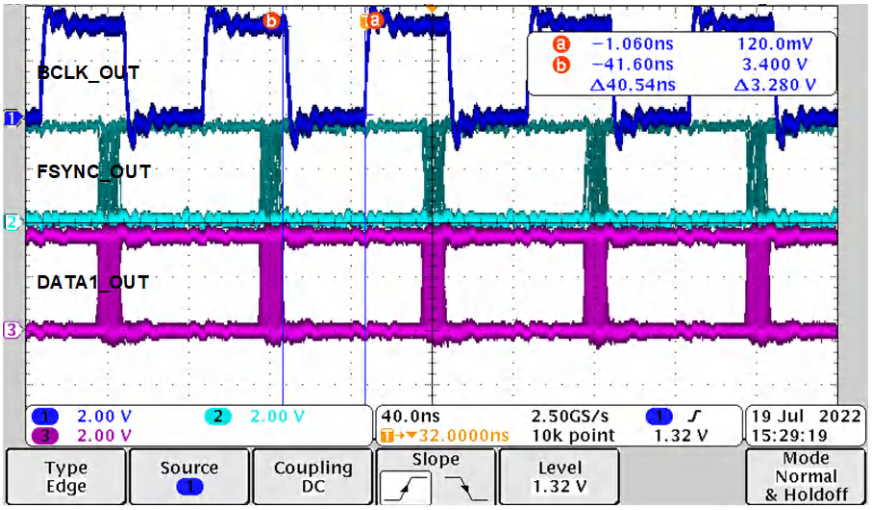

图 3-16. 最终 TDM-8 输出

由于 BCLK_OUT 必须在没有抖动的情况下通过 CLB 逻辑块传递，因此在未对 BCLK_OUT 引入抖动的情况下无法准确控制 BCLK_OUT 和其他两个信号之间的时序。但在这种情况下，建立时间和保持时间是可以接受的。如果需要，可以使用 CLB 逻辑块中的 AOC 反转 BCLK_OUT 信号，以换取建立时间和保持时间。

最后请注意 FSYNC_OUT 和 DATA1_OUT 信号上的 10ns 抖动。这是由于 CLB 逻辑块以 100MHz 的频率对传入的 BCLK_IN 信号进行采样所致。抖动进一步缩短了建立时间和保持时间。

Copyright © 2023 Texas Instruments Incorporated

3.7.4 测试数据完整性

为了测试数据完整性，F28388D 向 F280025C 发送递增的数据模式。F280025C 根据预期数据检查传入数据，如果检测到任何错误，则生成一个标志。F280025C 将所有接收到的数据回传到 F28388D。最后，F28388D 检查所有传入数据是否有错误。图 3-17 显示了正在进行的 F28388D 监测的快照。请注意，F280025C 或 F28388D 在整个 32 位范围发送过程中未检测到错误。

Variables | Expressions | Registers

Expression	Type	Value	Address
txData	unsigned long	0x843B2B6A (Hex)	0x0000A804@Data
testWordDetected	unsigned int	1	0x0000A800@Data
errCountGlobal	unsigned long	0	0x0000A802@Data
Add new expression			

图 3-17. F28388D 监测传入的 TDM 流量

4 示例 B：在照明应用中使用 CLB 为 LED 驱动器实施定制通信总线

4.1 示例概述

在此示例中，使用 C2000 实时微控制器在 16 x 16 RGB LED 矩阵显示屏上显示简单图形。LED 矩阵显示屏由一个 LP5891-Q1 LED 驱动器驱动，该驱动器使用串行总线与 C2000 实时微控制器进行通信。

两个 CLB 逻辑块用于支持与 LP5891-Q1 LED 驱动器器件进行通信所需的连续时钟串行接口 (CCSI) 总线协议。基于 CLB 的 CCSI 总线方案为总线帧编码时所需的 CPU 开销非常少，因为 CLB 会自动添加在 CCSI 总线上通信所需的 IDLE、START、END 和 CHECK 位。

可以修改示例代码以级联额外的 LED 矩阵显示屏。

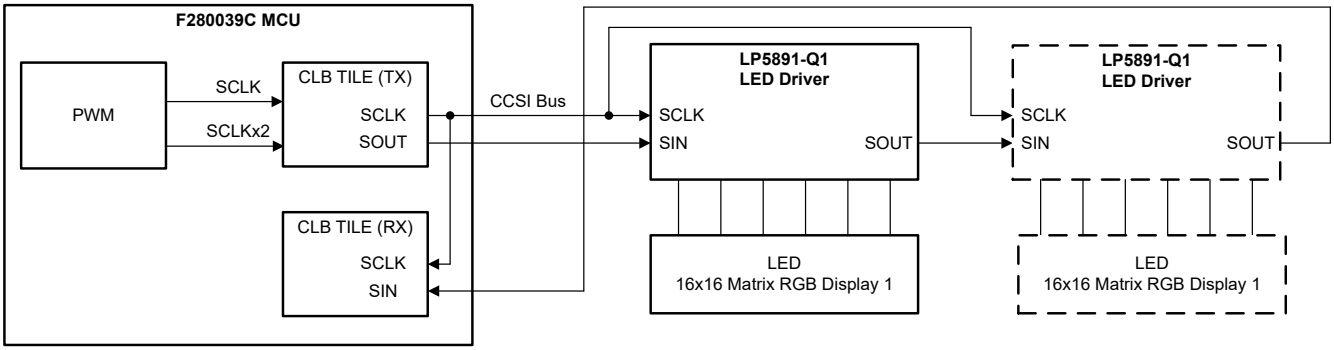

图 4-1. LED 矩阵的方框图示例

English Document: SPRAD62
Copyright © 2023 Texas Instruments Incorporated

4.2 步骤 1：了解设计要求

表 4-1 中列出了用于此演示的设计参数。

表 4-1. 设计要求

设计参数	值
显示模块大小	16 x 16 RGB LED
帧速率	60Hz
刷新率	7680 Hz
PWM 分辨率	16 位
级联器件	1（可增大至 2）
CCSI 总线计数	1
SCL 频率	5.0 MHz
GCLK 频率	70.0 MHz
T_{SW}	643ns

LP5891-Q1 LED 驱动器器件支持的 CCSI 总线协议如图 4-2 所示。CCSI 协议包括以下要求：

- 连续串行时钟
- 具有开始、空闲、数据和结束状态的可变帧长度
- 16 位标头和数据字
- 附加在所有标头和数据字上的校验位

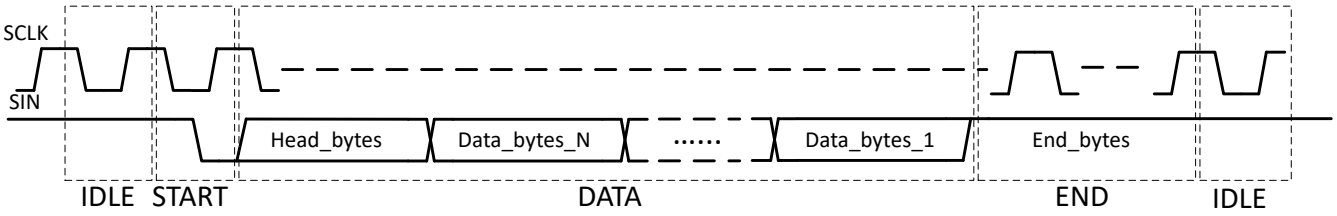

图 4-2. CCSI 帧

增加了两个额外要求。首先，需要通过可配置的单时钟和双时钟沿发送和接收来支持不同的 LED 驱动器器件。例如，TLC6983 LED 显示驱动器使用双时钟沿数据发送和接收。其次，需要能够在多条 CCSI 总线上同时生成 VSYNC 命令。VSYNC 用于同步级联链中器件的每个帧的显示。

Copyright © 2023 Texas Instruments Incorporated

4.3 步骤 2：识别至 CLB 逻辑块的所需输入

图 4-3 显示了至 TX 逻辑块的输入信号。

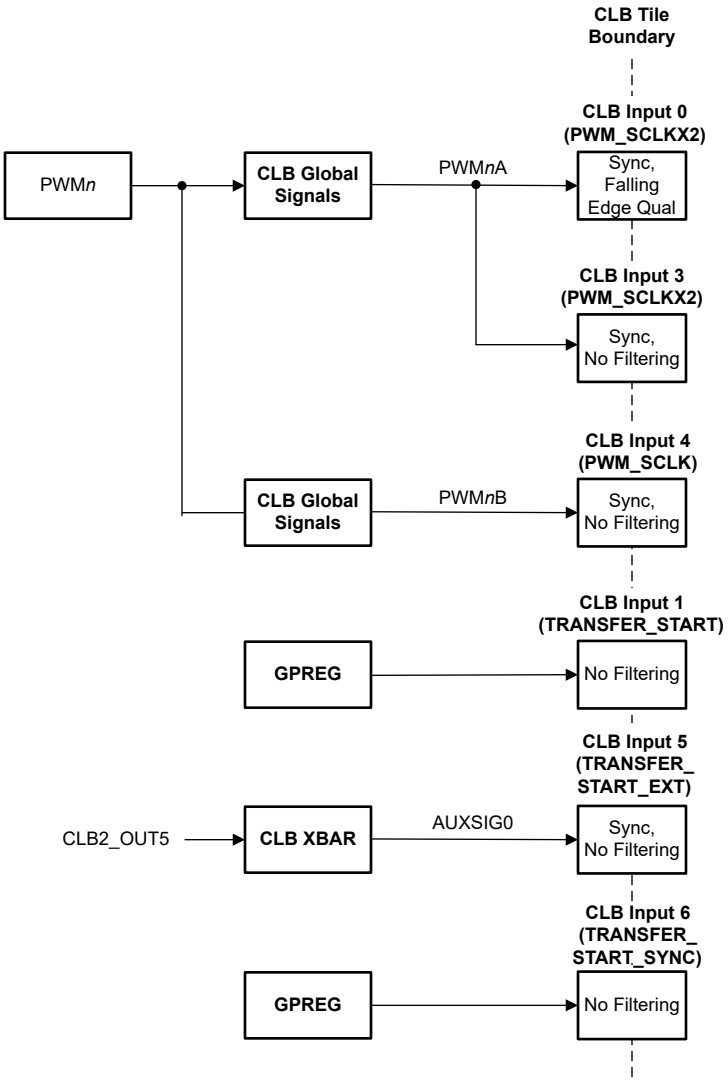

图 4-3. TX 逻辑块输入信号

TX 逻辑块使用以下输入：

- CLB 输入 0：此输入由 PWM*n*A 输出在内部驱动。PWM*n*A 输出配置为生成其频率为目标 SCLK 频率 2 倍的时钟。TX 逻辑块使用 PWM*n*A 时钟发送数据。
- CLB 输入 3：此输入由 PWM*n*A 输出在内部驱动。输入通过逻辑块传递，用于驱动 CLB_SCLKX2 输出。
- CLB 输入 4：此输入由 PWM*n*B 输出在内部驱动。PWM*n*B 输出配置为生成其频率为目标 SCLK 频率的时钟。输入通过逻辑块传递，用于驱动 CLB_SCLK 输出。

TX 逻辑块还使用三个辅助信号来同步所有 TX 逻辑块上的数据发送：

- CLB 输入 1：GPREG.1 位用于启动 TX 逻辑块上的数据传输。它由 CPU 置位/清零。
- CLB 输入 5：此输入用作到 TX 逻辑块的外部传输开始信号。此输入始终取自 CLB 逻辑块 2、输出 5。
- CLB 输入 6：在使用多个 TX 逻辑块的情况下，GPREG.6 位驱动所有 TX 逻辑块上的外部传输开始信号。只有 CLB 逻辑块 2 中的 GPREG.6 可用于在其他 TX 逻辑块中启动传输。GPREG.6 位由 CPU 置位/清零。

English Document: SPRAD62
Copyright © 2023 Texas Instruments Incorporated

图 4-4 显示了至 RX 逻辑块的输入信号。

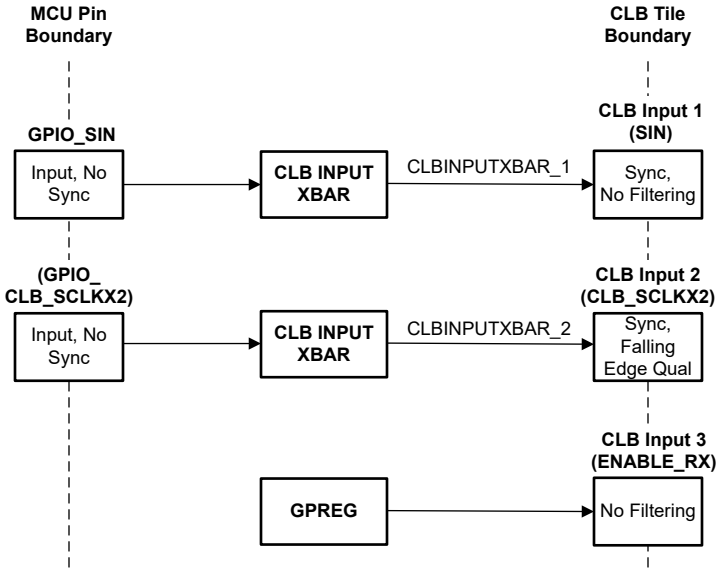

图 4-4. RX 逻辑块输入信号

RX 逻辑块使用以下输入：

- CLB 输入 1：此输入用于从 SIN 引脚接收数据。
- CLB 输入 2：此输入用于在 CLB_SCLKX2 的下降沿触发数据接收。CLB_SCLKX2 信号是由 TX CLB 逻辑块提供的外部时钟，它以 2 倍于 SCLK 的频率运行。
- CLB 输入 3：GPREG.3 位用于启用/禁用数据接收。它由 CPU 置位/清零。

4.4 步骤 3：识别来自 CLB 逻辑的所需输出

图 4-5 显示了 TX 逻辑块的输出信号。

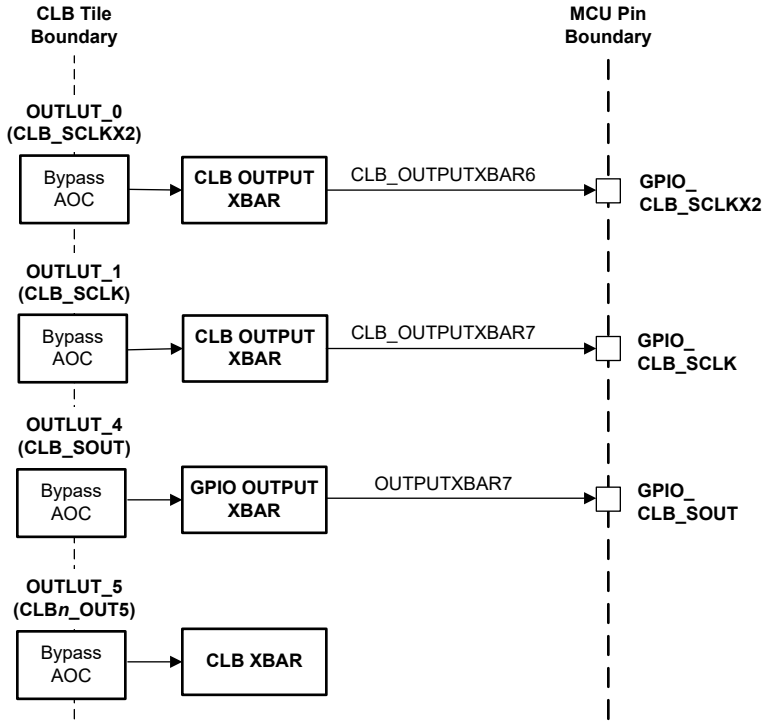

图 4-5. TX 逻辑块输出信号

Copyright © 2023 Texas Instruments Incorporated

PWM 内部生成的 SCLK 和 SCLKX2 时钟通过 TX 逻辑块传递，并在 GPIO 引脚上驱动。TX 逻辑块的输出在 CLB_Sout 引脚上驱动。最后，逻辑块的输出 5 驱动到 CLB XBAR，在这里，它可用于同步多个逻辑块的数据发送。

RX 逻辑块逻辑没有任何输出信号。

4.5 步骤 4：设计 CLB 逻辑

最终设计使用两个 CLB 逻辑块来实施单个 CCSI 串行端口。一个逻辑块用于发送操作，第二个逻辑块用于接收操作。TX 和 RX 逻辑块逻辑可复制到多个逻辑块以创建额外的 CCSI 总线。

4.5.1 TX 逻辑块逻辑

图 4-6 显示了 TX 逻辑块的逻辑。

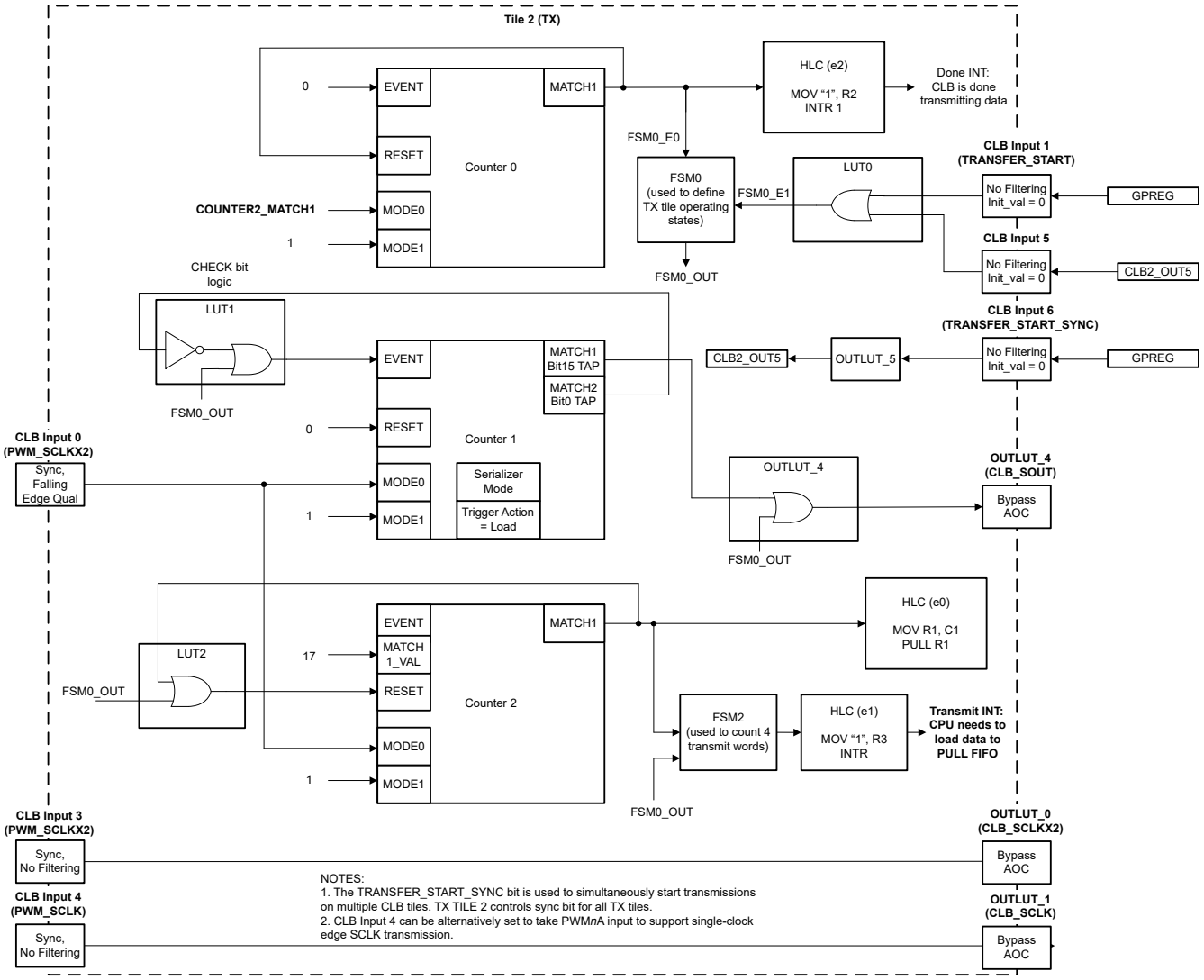

图 4-6. CLB TX 逻辑块逻辑

以下 CLB 逻辑块资源用于实施 CCSI 总线发送 (TX) 功能：

- COUNTER0 用于计算要发送的字数。它在正常计数器模式下运行。
- COUNTER1 用于在 SCLKX2 下降沿发送输出数据。它在串行器模式下运行。
- COUNTER2 用于对 SCLKX2 下降沿进行计数。它在正常计数器模式下运行。
- HLC 用于将数据从 CLB PULL FIFO 移入 COUNTER1。当发送四个 16 位字且已发送整个帧时，它也会向 CPU 生成中断。

English Document: SPRAD62
Copyright © 2023 Texas Instruments Incorporated

- FSM0 用于在 IDLE、ACTIVE 和 END 状态之间循环切换。
- 当发送了四个 16 位字时，FSM2 用于计数。当发送计数达到四时，向 HLC 生成一个事件。

为组合逻辑使用多个 LUT 和输出 LUT。

FSM0 定义了 TX 逻辑块的三种运行状态。图 4-7 和表 4-2 显示了 FSM0 状态图和真值表。

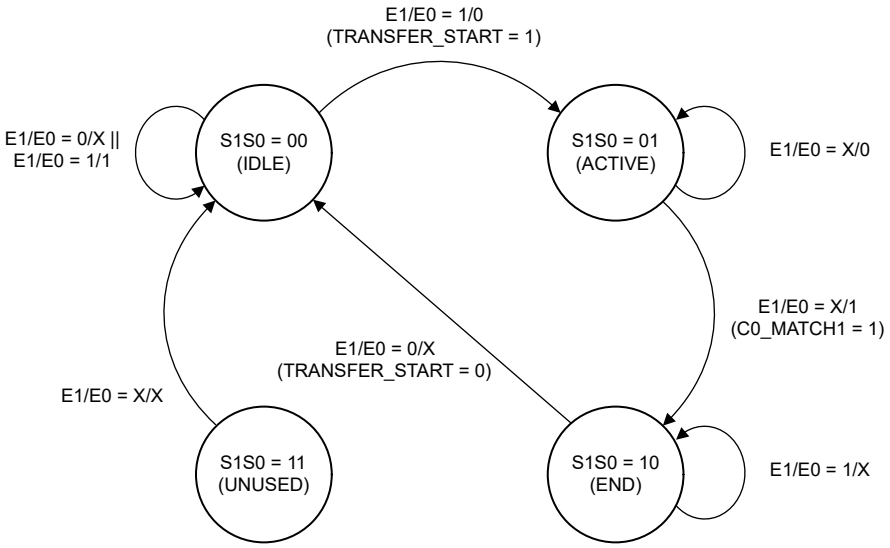

图 4-7. TX 逻辑块的 FSM0 状态图

表 4-2. FSM0 真值表

S1	S0	E1 (TRANSFER_ START)	E0 (C0_MATCH1)	S1 次态	S0 次态	输出 (FSM0_OUT)
0	0	0	0	0	0	1
0	0	0	1	0	0	1
0	0	1	0	0	1	1
0	0	1	1	0	0	1
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	0	1	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	1	0	1
1	0	1	1	1	0	1
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	0	1

简化的逻辑方程为：

- S1 = !S1&S0&E0 | S1&!S0&E1
- S0 = !S1&E1&!E0 | !S1&S0&!E0
- OUTPUT = !S0 | S1

FSM2 用于在发送四个 16 位字后触发 HLC 事件。图 4-8 和表 4-3 显示了 FSM2 状态图和真值表。

Copyright © 2023 Texas Instruments Incorporated

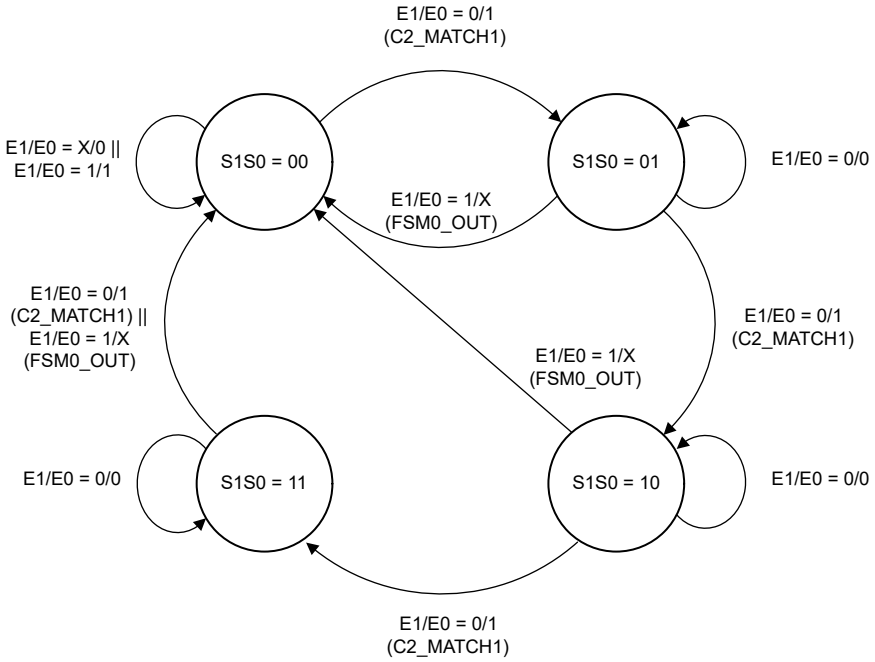

图 4-8. TX 逻辑块的 FSM2 状态图

表 4-3. FSM2 真值表

S1	S0	E1 (FSM0_OUT)	E0 (C2_MATCH1)	S1 次态	S0 次态	输出 (HLC_INT)
0	0	0	0	**0**	**0**	**0**
0	0	0	1	**0**	**1**	**0**
0	0	1	0	**0**	**0**	**0**
0	0	1	1	**0**	**0**	**0**
0	1	0	0	**0**	**1**	**0**
0	1	0	1	**1**	**0**	**0**
0	1	1	0	**0**	**0**	**0**
0	1	1	1	**0**	**0**	**0**
1	0	0	0	**1**	**0**	**0**
1	0	0	1	**1**	**1**	**0**
1	0	1	0	**0**	**0**	**0**
1	0	1	1	**0**	**0**	**0**
1	1	0	0	**1**	**1**	**0**
1	1	0	1	**0**	**0**	**1**
1	1	1	0	**0**	**0**	**0**
1	1	1	1	**0**	**0**	**0**

简化的逻辑方程为：

- S1 = S1&!S0&!E1 | S1&!E1&!E0 | !S1&S0&!E1&E0
- S0 = !S0&!E1&E0 | S0&!E1&!E0
- OUTPUT = S1&S0&!E1&E0

English Document: SPRAD62
Copyright © 2023 Texas Instruments Incorporated

4.5.2 RX 逻辑块逻辑

图 4-9 展示了接收逻辑块的逻辑。

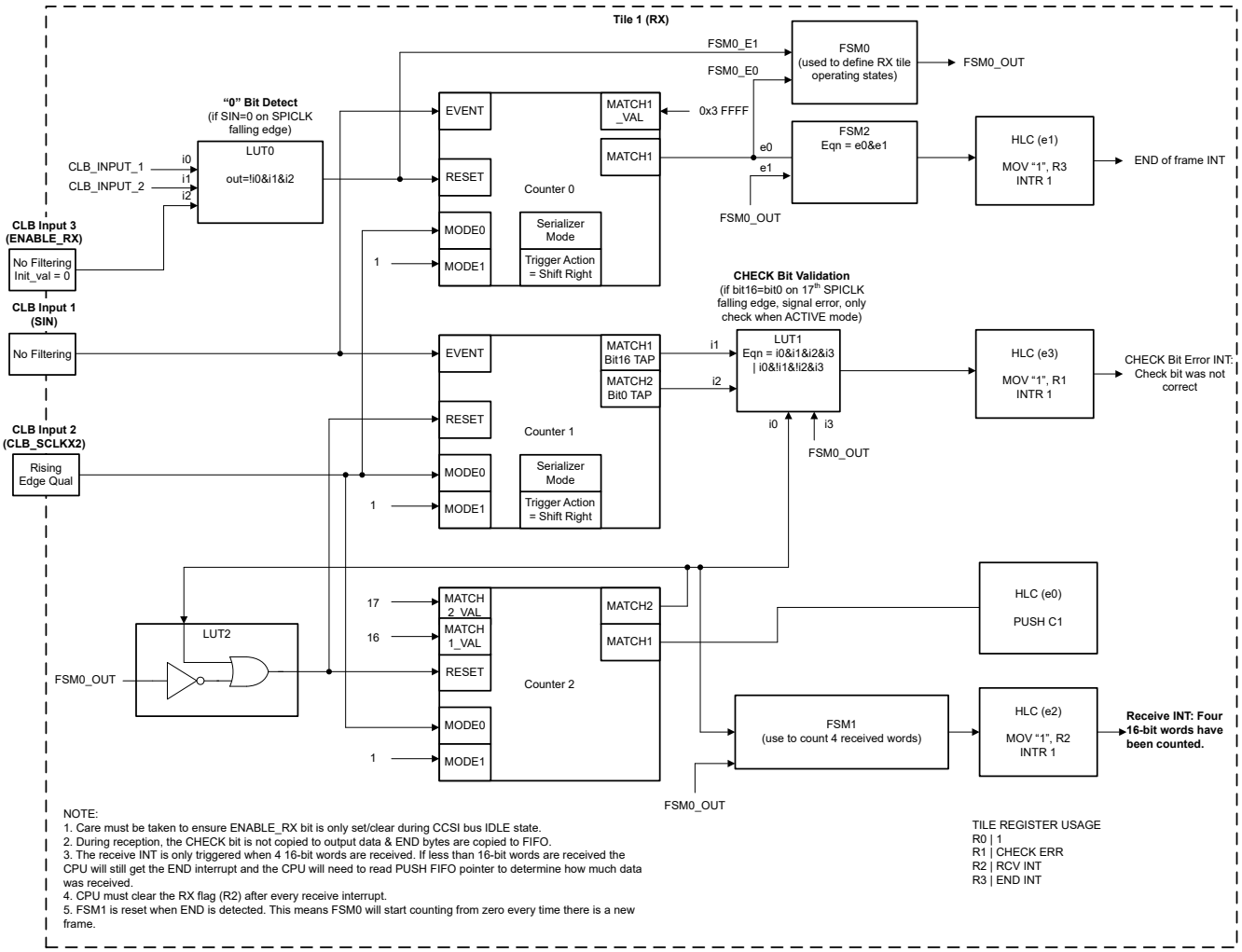

图 4-9. CLB RX 逻辑块逻辑

以下 CLB 逻辑块资源用于实施 CCSI 总线接收 (RX) 功能：

- COUNTER0 用于检测接收帧的结束。
- COUNTER1 用于在 SCLKX2 下降沿接收输入数据。它在串行器模式下运行。
- COUNTER2 用于对 SCLKX2 下降沿进行计数。它在正常计数器模式下运行。
- HLC 用于将数据从 COUNTER1 移动到 CLB PUSH FIFO。当接收到四个 16 位字以及检测到 CHECK 位错误和帧结束条件时，它还会向 CPU 生成中断。
- FSM0 用于定义和循环遍历两种状态：IDLE 和 ACTIVE。
- 当接收到四个 16 位字时，FSM1 用于计数。当接收计数达到四时，向 HLC 生成一个事件。
- 仅当处于 ACTIVE 状态时，FSM2 才用于生成帧结束中断。
- LUT0 用于检测相应的“0”位，该位用于检测 START 位和 END 字节。通过清除 ENABLE_RX 输入，可以强制将 LUT0 输出置为低电平，从而有效地禁用接收操作。
- LUT1 用于验证每个 16 位字上的 CHECK 位。如果 CHECK 位与预期状态不匹配，LUT1 输出会触发 HLC 中断。
- 当禁用接收操作（即 ENABLE_RX = 0）时，LUT2 用于使 COUNTER2 保持在复位状态。

FSM0 定义了 RX 逻辑块的两种运行状态。图 4-10 和表 4-4 显示了 FSM0 状态图和真值表。

Copyright © 2023 Texas Instruments Incorporated

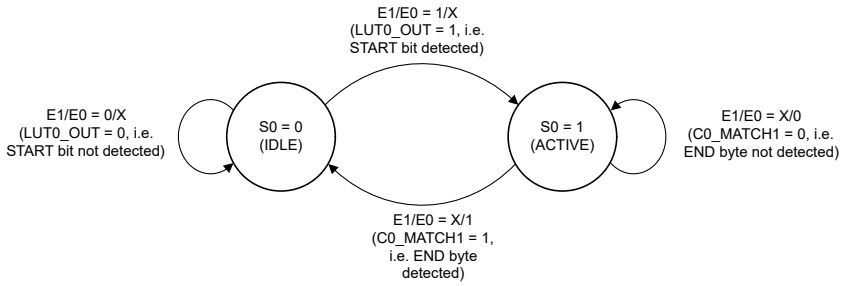

图 4-10. RX 逻辑块的 FSM0 状态图

表 4-4. FSM0 真值表

S0	E1 (LUT0_OUT)	E0 (C0_MATCH1)	S0 次态	输出 (FSM0_OUTPUT)
0	0	0	0	0
0	0	1	0	0
0	1	0	1	0
0	1	1	1	0
1	0	0	1	1
1	0	1	0	1
1	1	0	1	1
1	1	1	0	1

完整的逻辑方程为：

- S0 = !S1&!S0&E1&!E0 | !S1&!S0&E1&E0 | !S1&S0&!E1&!E0 | !S1&S0&E1&!E0
- OUTPUT = !S1&S0&!E1&!E0 | !S1&S0&!E1&E0 | !S1&S0&E1&!E0 | !S1&S0&E1&E0

FSM1 用于在接收到四个 16 位字后触发 HLC 事件。图 4-11 和表 4-5 显示了 FSM1 状态图和真值表。

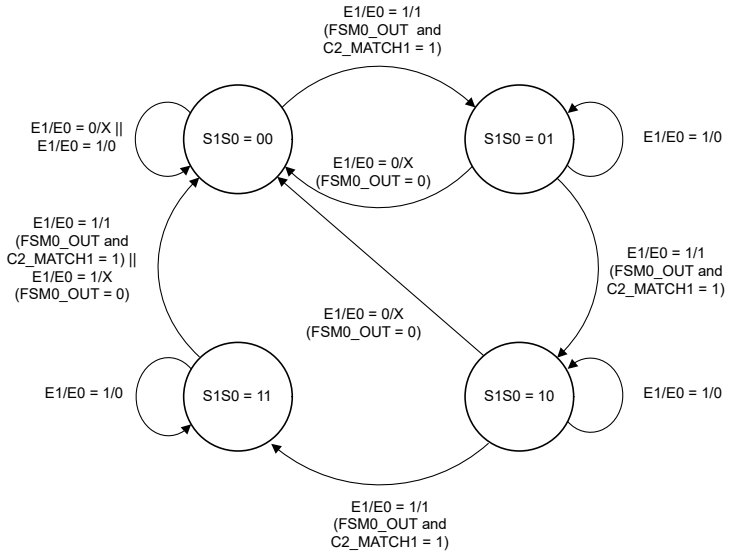

图 4-11. RX 逻辑块的 FSM1 状态图

English Document: SPRAD62
Copyright © 2023 Texas Instruments Incorporated

表 4-5. FSM1 真值表

S1	S0	E1 (FSM0_OUT)	E0 (C2_MATCH2)	S1 次态	S0 次态	输出 (HLC_INT)
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	0	1	1	0	1	0
0	1	0	0	0	0	0
0	1	0	1	0	0	0
0	1	1	0	0	1	0
0	1	1	1	1	0	0
1	0	0	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	0	1	1	1	1	0
1	1	0	0	0	0	0
1	1	0	1	0	0	0
1	1	1	0	1	1	0
1	1	1	1	0	0	1

逻辑方程是：

- S1 = !S1&S0&E1&E0 | S1&!S0&E1&!E0 | S1&!S0&E1&E0 | S1&S0&E1&!E0
- S0 = !S1&!S0&E1&E0 | !S1&S0&E1&!E0 | S1&!S0&E1&E0 | S1&S0&E1&!E0
- OUTPUT = S1&S0&E1&E0

4.5.3 数据时钟

CCSI 总线通信所需的时钟通过片上 PWM 生成。PWM*n*A 输出生成一个以双倍目标 SCLK 频率运行的 PWM_SCLKX2 时钟，而 PWM*n*B 输出生成一个以所需的 SCLK 频率运行的 PWM_SCLK 时钟。

TX 和 RX CLB 逻辑块始终使用 PWM_SCLKX2 时钟发送和接收数据。最终只有 PWM_SCLK 驱动到系统中的 LED 驱动器。

通过配置 TX 逻辑块的 PWM_SCLK 输入源，可实现双时钟或单时钟沿数据发送和接收。当需要单时钟沿发送和接收时，例如，当使用 LP5891-Q1 LED 驱动器时，可以修改 TX 输入逻辑块配置，以便将 PWM*n*A 输出驱动至 PWM_SCLK 和 PWM_SCLKX2 输入，请参阅图 4-12。PWM*n*B 输出可以不连接 CLB 逻辑块。

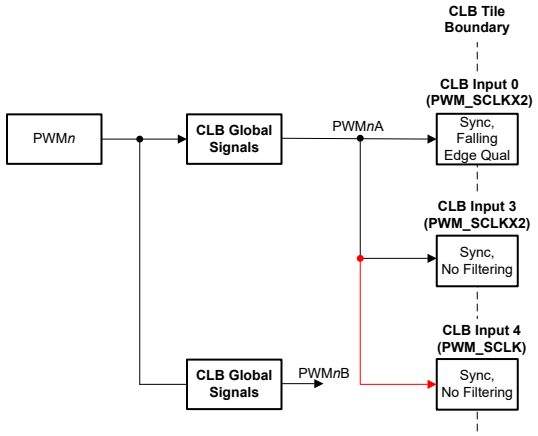

图 4-12. 单时钟沿数据发送和接收时钟配置

双时钟沿数据发送/接收可在时钟的两个边沿上实现发送和接收，请参阅图 4-13。

Copyright © 2023 Texas Instruments Incorporated

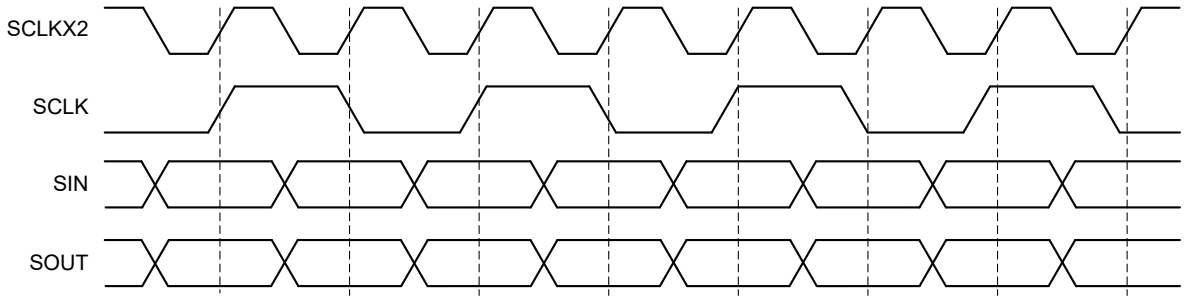

图 4-13. 使用 PWM_SCLKX2 的双时钟沿数据发送/接收

当需要双时钟沿发送和接收时，可以修改 TX 输入逻辑块配置，以便将 PWM*n*A 输出驱动至 PWM_SCLKX2 输入，并将 PWM*n*B 输出驱动至 PWM_SCLK 输入，请参阅图 4-14。

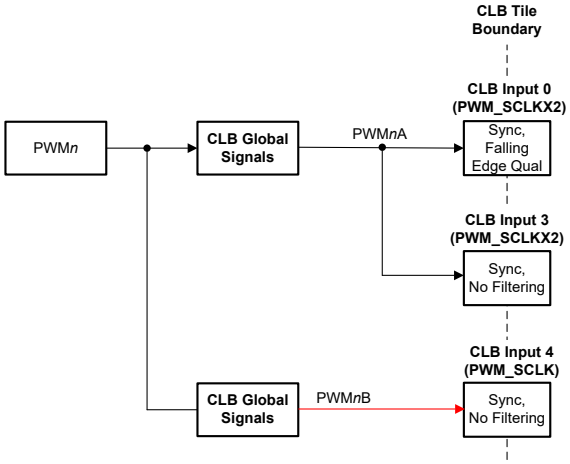

图 4-14. 双时钟沿数据发送和接收时钟配置

为了减少 CLB 逻辑引入的时序延迟，两个 PWM 时钟输入都通过 TX CLB 逻辑传递。

English Document: SPRAD62
Copyright © 2023 Texas Instruments Incorporated

4.6 步骤 5 ：仿真逻辑设计

图 4-15 中仿真了数据接收操作。对于此仿真，使用 0xAAAA 的简单模式作为数据输入。该仿真显示了由 START 位指示的帧开始、完整接收字的捕获以及 CHECK 位的验证。

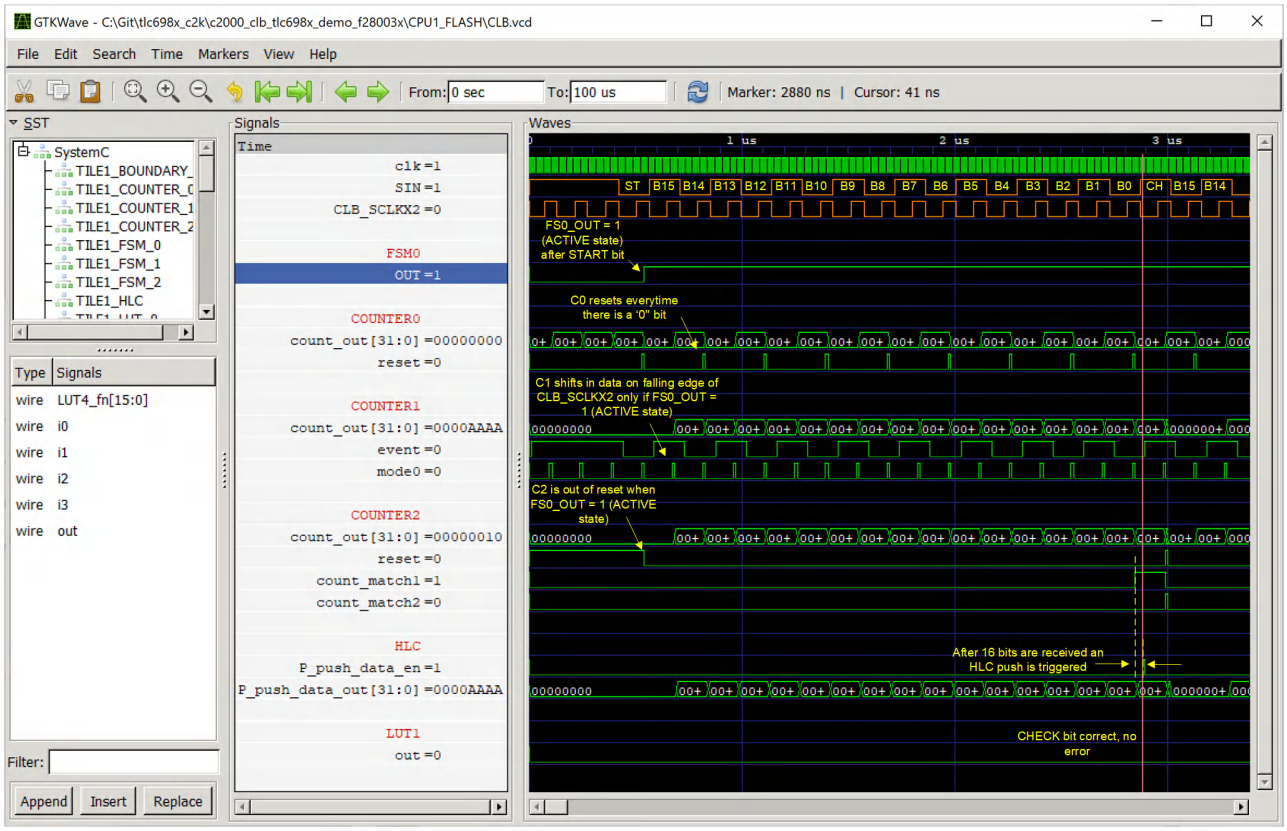

图 4-15. LED 驱动器的数据接收仿真

Copyright © 2023 Texas Instruments Incorporated

第二个数据接收仿真如图 4-16 所示。在此仿真中，向传入数据流添加了不正确的 CHECK 位，以便验证 CHECK 位逻辑的操作。该仿真显示 LUT1 块的输出变为高电平，以指示检测到 CHECK 位错误。

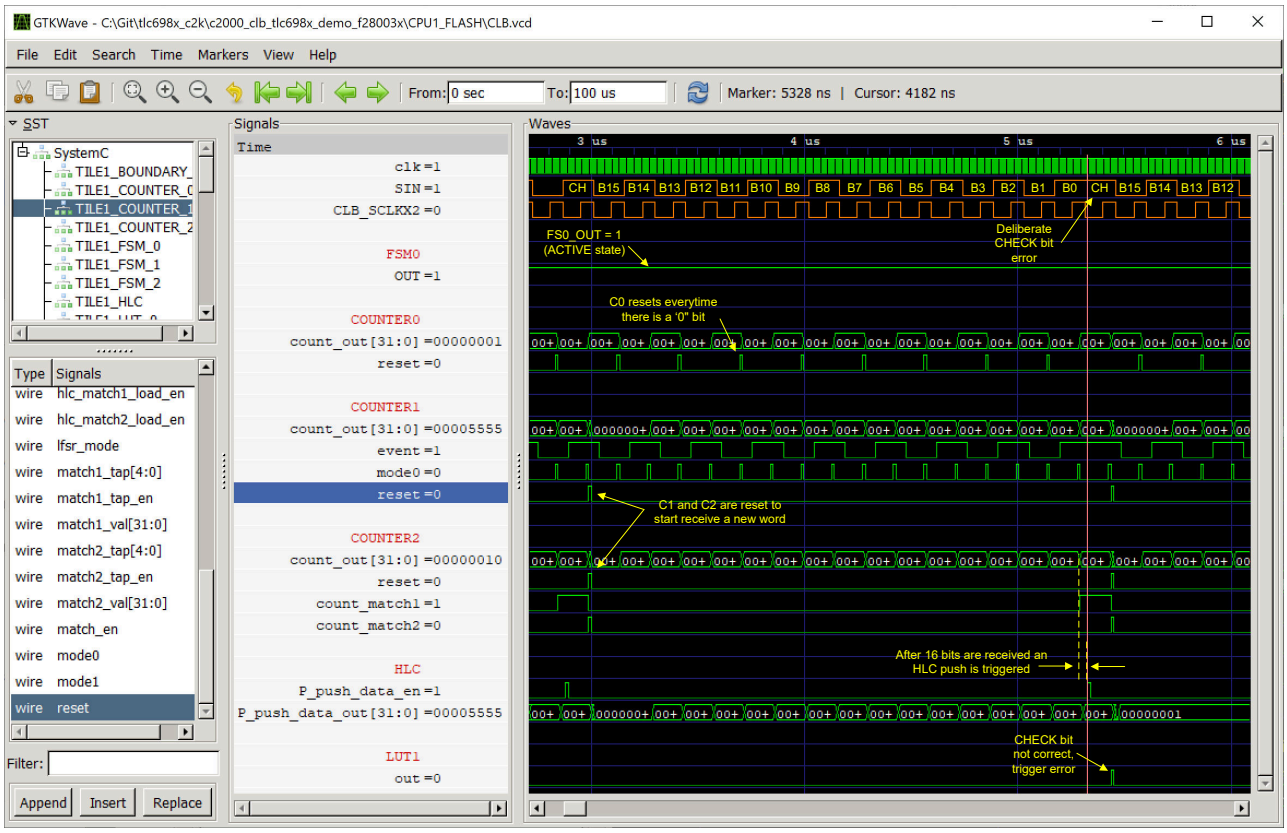

图 4-16. CHECK 位错误逻辑仿真

最后，图 4-17 中的仿真显示了 END 位的检测。FSM0 块在检测到 END 位时将 CLB 逻辑转换为 IDLE 状态。

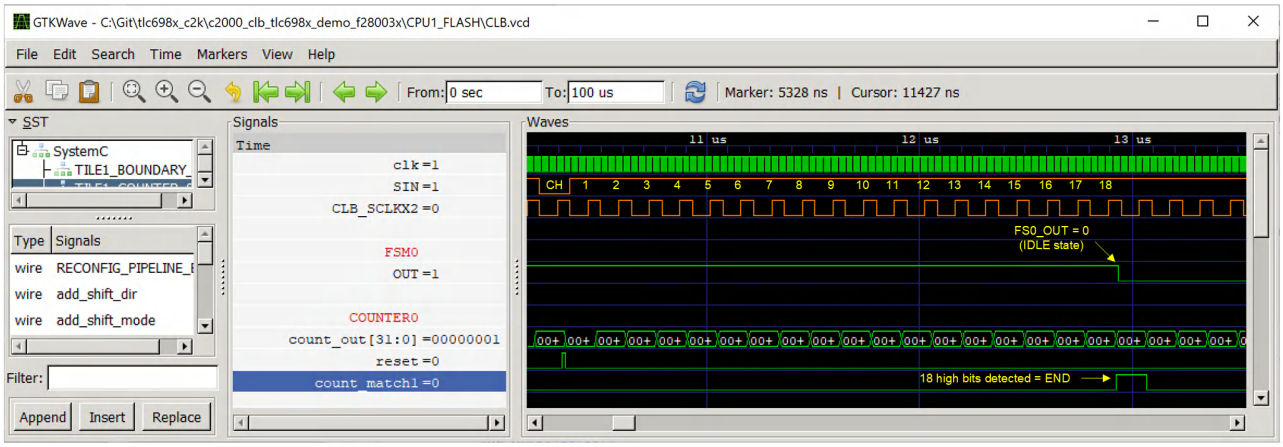

图 4-17. END 帧检测

备注

在 END 位期间会生成接收中断和 CHECK 位错误（未显示）。CPU 代码必须始终丢弃帧中的最后一个字，因为这与 END 位相对应。

English Document: SPRAD62
Copyright © 2023 Texas Instruments Incorporated

图 4-18 中仿真了数据发送操作。该仿真显示了单个 0x5555 字的发送，从 START 位开始，到 END 帧结束。为了简化 CLB 逻辑，通过允许逻辑块发送一个 0xFFFF 字来实施 START 位。使用逻辑块的 CHECK 位逻辑在第 17 个时钟周期生成 START 位。同样，要生成 END 帧，CLB 逻辑仅在最后一个字发送结束时将 CLB_SOUT 信号设置为 1。CLB 逻辑取决于 CPU 在开始新的数据发送之前至少等待 18 个 PWM_SCLKX2 周期。

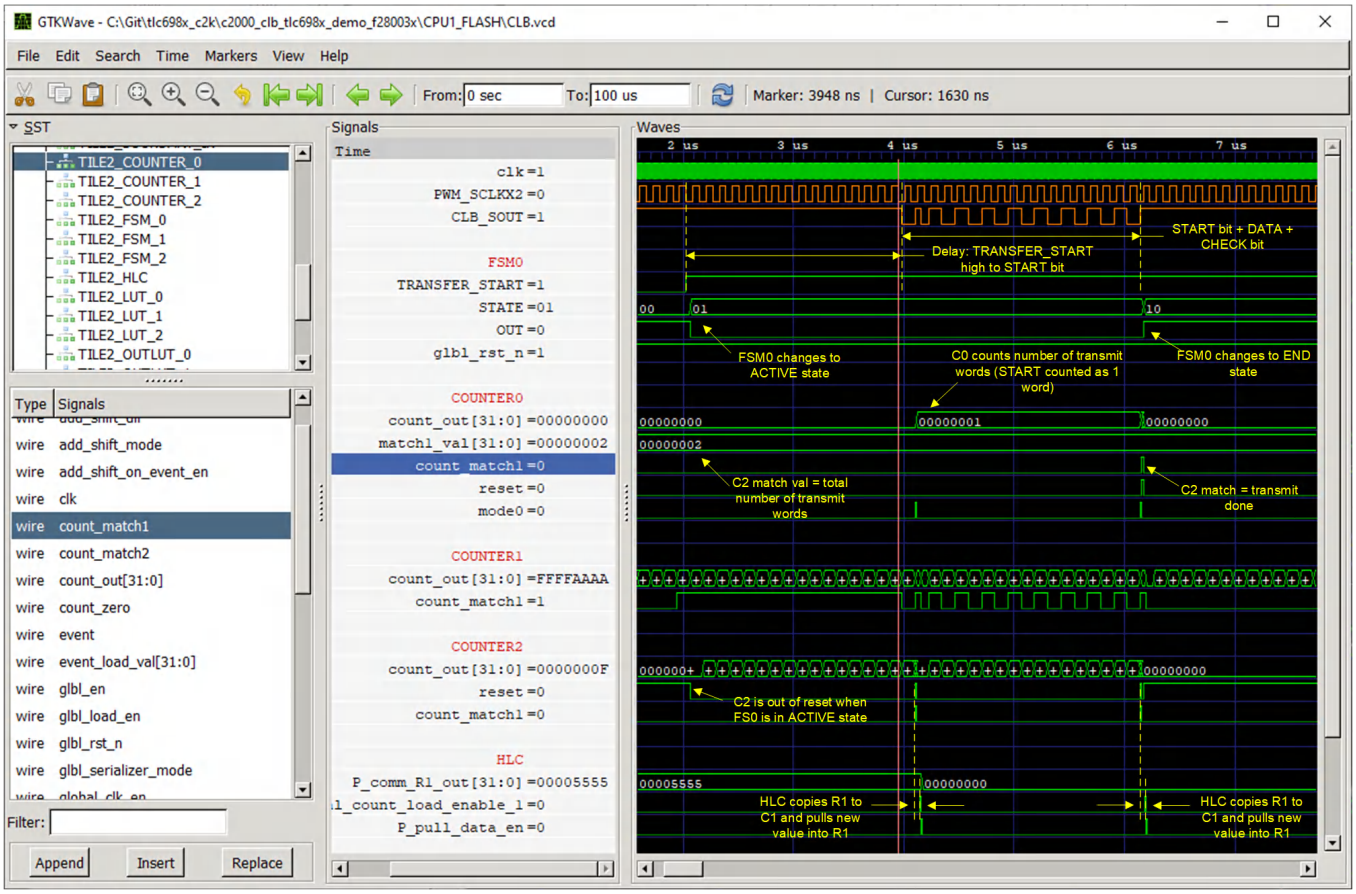

图 4-18. LED 驱动器的数据发送仿真

Copyright © 2023 Texas Instruments Incorporated

由于更新输出串行器（计数器 1）出现延迟，在数据发送操作期间会观察到输出干扰。图 4-19 显示了从 PWM_SCLKX2 低沿到输出串行器最终更新的内部延迟。干扰的结果是接收器件的建立时间缩短。建立时间不应从 PWM_SCLKX2 信号的上升沿计算，因为此时钟也会随着它通过 CLB 逻辑块而延迟。而是应根据 CLB_SCLKX2 信号计算建立时间。此信号是接收器件将观察到的 PWM_SCLKX2 的延迟版本。

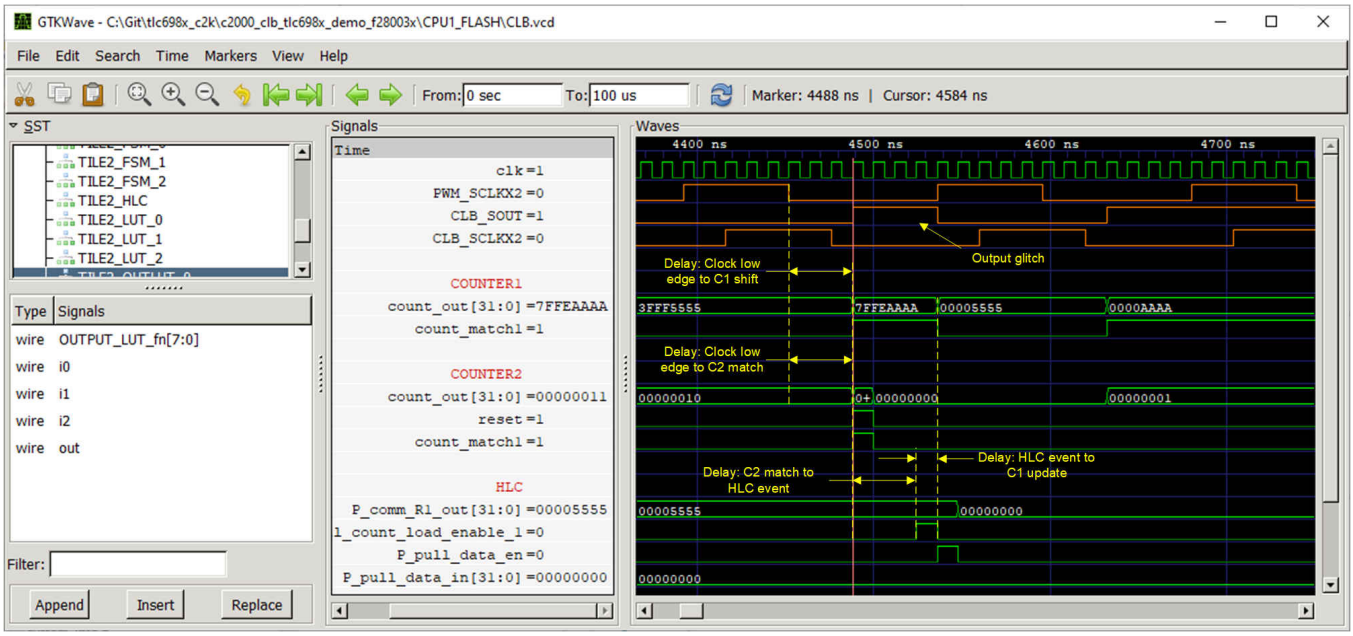

图 4-19. 输出干扰

可以通过在传递到 CLB_SOUT 之前锁存串行器输出来消除干扰。但是，由于每个发送仿真预计至少有两个 CLB 周期的建立时间，这足以满足 LP5891-Q1 器件所需的 10ns 建立时间要求，因此未使用此方法。

4.7 步骤 6 ：测试 CLB 逻辑

本节说明了如何测试为 LED 驱动器示例提供的 Code Composer Studio™ 示例工程。我们在 F280039C LaunchPad (LAUNCHXL-F280039C) 上测试了此示例。

出于测试目的，使用一个 LP5891-Q1 EVM 接收和显示来自 F280039C MCU 的图像数据。LP5891-Q1 EVM 包括一个 16x16 矩阵 LED 显示屏。

4.7.1 硬件设置和连接

运行演示需要以下硬件：

- 1 个 LAUNCHXL-F280039C LaunchPad
- 1 个 LP5891Q1EVM 矩阵 LED 显示屏

可选

- 逻辑分析仪（用于查看 CCSI 总线信号）

按如下方式设置硬件：

1. 使用默认设置来设定 LAUNCHXL-F280039C EVM：
   a. 安装 JP1 以将 5V 电源和 GND 从 USB-C 连接器连接到电路板的 XDS110 侧。
   b. 安装 JP2 以将电路板 XDS110 侧的 5V 电源连接到 LaunchPad 的其余部分。
   c. 安装 J101 上的 TCK 和 TMS 跳线以将 XDS110 连接到 F280039C 器件（如需要，也可以安装 J101 上的其他跳线）。
   d. S3 引导模式设置为闪存引导（可选）。

English Document: SPRAD62
Copyright © 2023 Texas Instruments Incorporated

2. 按如下方式设置 LP5891Q1EVM：
   a. 开关 S1 设置为 SOUTHOST，以便将 LP5891-Q1 器件的串行数据输出路由回主机控制器。
   b. 跳线 J6 设置为 MCU5V，以便选择主机控制器的 5V 电源。
   c. 跳线 J7 设置为“3V3 VR”，以便为 VLEDR 选择 3.3V 电源。
3. 将 LP5891Q1EVM 插入 LAUNCHXL-F280039C EVM 上的位置 1。使用两个电路板上的 5V、3V3 和 GND 标识作为基准，正确确定电路板的方向。
4. 将 USB-C 电缆连接到 LAUNCHXL-F280039C EVM 上的 USB 接头。当 USB 电缆插入 USB2.0/USB3.x 端口时，LaunchPad 和 LP5891Q1EVM 从 USB 端口接收电源。
5. 继续软件设置。

4.7.2 软件设置

构建和运行软件需要以下软件包：

- Code Composer Studio（CCS）版本 11.1.0 或更高版本
- C2000ware 版本 4.02.00.00 或更高版本

  （可选）用于运行仿真的 GNU 编译器 (TDM-GCC) 和 GTK Wave 仿真查看器。有关更多信息，请参阅 *CLB 工具用户指南*。

在 CCS 中按照以下步骤设置 LED 示例软件：

1. 在 CCS 菜单中，依次点击“Project -> Import CCS Projects...”。
2. 在“Select search-directory”中输入 F28003x driverlib CLB 示例工程的路径。
   a. Path：<C2000WARE_INSTALL>\driverlib\f28003x\examples\clb
3. 单击“Refresh”。
4. 选择“**clb_ex32_led_driver**”工程。

Copyright © 2023 Texas Instruments Incorporated

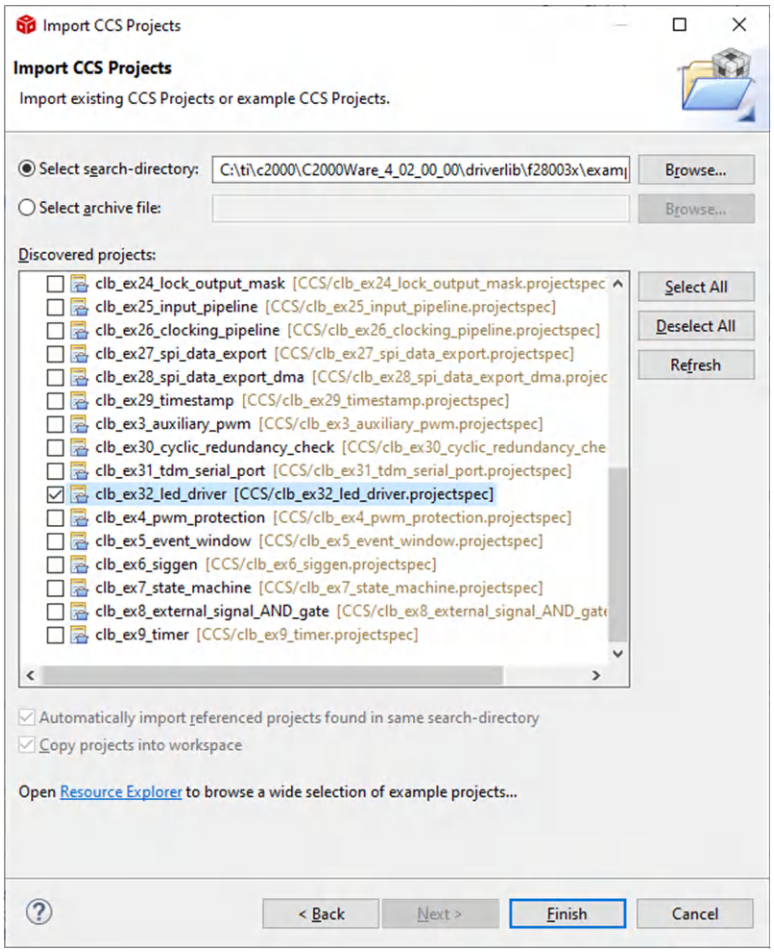

图 4-20. 导入 CLB LED 驱动器示例工程

5. 选中“Copy projects into workspace”。
6. 点击“Finish”将工程导入工作区。

   可选：点击“Project -> Build Configurations -> Set Active -> CPU1_LAUNCHXL_FLASH”构建用于闪存执行的代码。
7. 在 CCS 菜单中，点击“Project -> Build Project”来构建示例工程。
8. 右键点击工程中的 TMS320F280039C_LaunchPad.ccxml 文件，然后选择“Set as Active Target Configuration”。
9. 点击“Run -> Debug”将可执行文件加载到 F280039C 目标器件。
10. 最后，在 CCS Debug 透视图中点击“Run -> Resume”以运行代码。

4.7.3 测试输出建立时间和保持时间

来自 CLB 的输出 CCSI 总线中的一个关键考虑因素是接收器件看到的预期建立时间和保持时间。为了测量建立时间和保持时间，使用一个示波器来持续捕获 CLB 的输出。

如图 4-21 所示，CLB 输出的最短建立时间约为 55ns，足以满足 LP5891-Q1 LED 驱动器所需的 10ns 建立时间。请注意 CLB_SOUT_1 输出中的变化。如节 4.6 所述，这是由于更新串行器输出时内部延迟的变化所致。可以通过在传递到 CLB_SOUT 之前锁存串行器输出来消除干扰。但是，鉴于最终输出具有足够的建立时间来满足 LP5891-Q1 LED 驱动器要求，所以不使用此方法。

保持时间至少为 SCLK 周期的一半。

English Document: SPRAD62
Copyright © 2023 Texas Instruments Incorporated

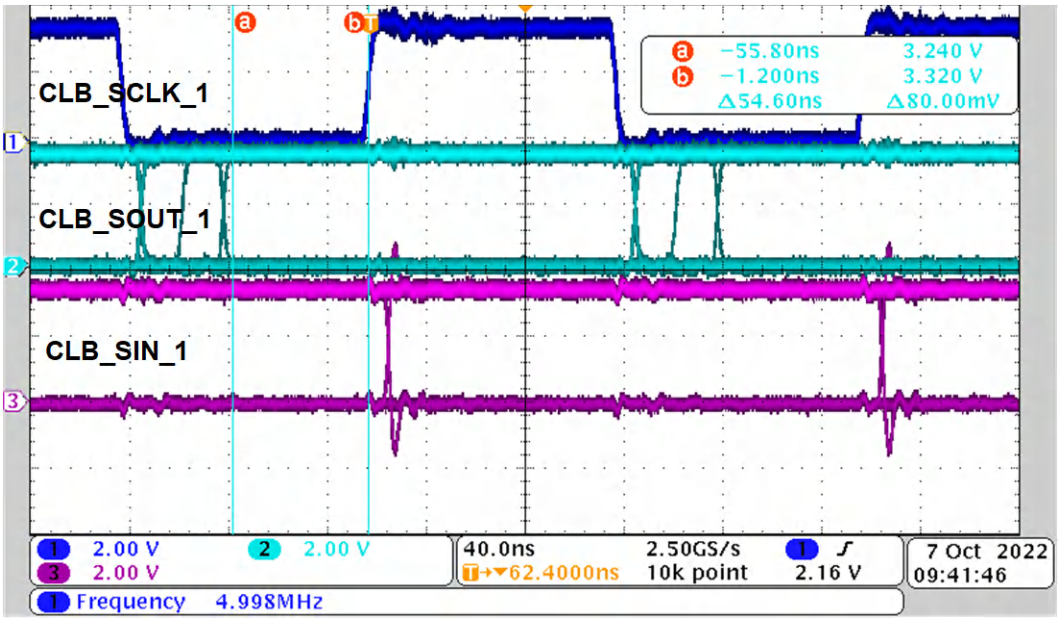

图 4-21. LED 驱动器示例建立时间和保持时间

Copyright © 2023 Texas Instruments Incorporated

5 参考文献

- 可配置逻辑块 (CLB) 介绍（视频）
- 可配置逻辑块 (CLB) 架构（视频）
- 可配置逻辑块 (CLB) 编程工具培训（视频）
- *CLB 工具用户指南*
- *TMS320F28388D controlCARD 信息指南*
- 适用于 C2000 实时 MCU 的 SysConfig 开发工具
- C2000 Academy 在线培训
- Code Composer Studio
- C2000ware
- TMDSCNCD280025C
- TMDSCNCD28388D
- TMDSHSECDOCK
- LAUNCHXL-F280039C
- LP5891Q1EVM

English Document: SPRAD62
Copyright © 2023 Texas Instruments Incorporated

重要声明和免责声明

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、某特定用途方面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功能安全、信息安全、监管或其他要求。

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 产品的应用。严禁对这些资源进行其他复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款或 ti.com 上其他适用条款/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。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2023，德州仪器 (TI) 公司