Application Report

如何将自定义逻辑从 FPGA/CPLD 迁移到 C2000™ 微控制器

Peter Galicki

摘要

通过将外部逻辑整合到 C2000™ 微控制器中，可配置逻辑块 (CLB) 降低了系统总成本并增强了功能。CLB 使用功能调用和基于 GUI 的编程工具 SysConfig 将外部逻辑整合到微控制器中，无需学习 VHDL 或 Verilog 等硬件描述语言。该报告向程序员、硬件工程师和系统设计人员展示如何将基于 FPGA 或 CPLD 的自定义逻辑（最初以 HDL 定义）转换为可以编程到 C2000 MCU 中的 CLB 格式。

内容

插图清单

商标

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

1 简介

就像 CPLD 或 FPGA 一样，CLB 由可以通过多种方法进行配置以实现自定义逻辑块的可编程逻辑基元组成。使用基于 GUI 的 SysConfig 工具和功能调用对 CLB 进行编程，而不是使用 VHDL 或 Verilog 来配置这些逻辑基元。由于配置方法不同，因此从技术上来说 CLB 不是 CPLD 或 FPGA，但可以用来实现相同的结果。

与外部 CPLD 和 FPGA 相比，CLB 具有一定的优势。CLB 位于 C2000 器件内部，因此它可以直接访问关键的 CPU 和外设信号，而不必考虑引脚延迟。此外，一个简单的内置 HLC 处理器可处理 CLB 与 C2000 存储器之间的数据传输，从而使 CLB 能够与 C2000 处理器上运行的软件配合工作。

借助 CLB，现在可以将外部自定义逻辑吸收到 C2000 器件中，在 C2000 内创建自定义外设，并在输入级、输出级或外设内许多预定义的位置修改现有的 C2000 控制外设。以下各节包含有关如何实现最常见用例的分步说明，以及 CLB 构建块的低级功能原理图，以帮助将 VHDL 或 Verilog 中的逻辑映射到 CLB。许多强大而灵活的 CLB 功能可提供很多好处，包括减少系统元件数量、增加区分产品的灵活性以及能够在器件出厂后通过软件在现场更新自定义逻辑。

本应用报告基于基本级别的 CLB 架构，该基本级别对于包括 F28004x、F2807x、F2837x 和 F2838x 系列在内的几种 C2000 器件是通用的。将来的版本将包括其他特性和扩展功能。

2 从硬件角度介绍 CLB

2.1 CLB 如何工作

CLB 是可由 CPU 或 CLA 通过配置寄存器进行配置的可编程逻辑基元、输入和输出多路复用器的集合。CLB 模块具有可选的输入和输出信号，这些信号连接选定的控制外设（增强型脉冲宽度调制器 (EPWM)、正交编码器脉冲 (QEP) 和增强型捕捉 (ECAP)）内部。根据逻辑基元的配置方式，自定义逻辑作用于来自控制外设内部的输入信号，产生结果送到输出信号，然后把信号向回注入到控制外设内的选定位置。

请注意，进入和离开控制外设的原有输入和输出功能不受 CLB 的影响，只有内部信号能被修改（请参阅图 2-1）。此方法可以修改所选控制外设的操作，或将其完全替换为全新的自定义外设（当在外设边界处分接输入和输出时）。请注意，即使用 CLB 逻辑完全替换了控制外设的内部，外设边界处的输入和输出也不变（相关的通用输入/输出 (GPIO) 功能也是如此）。

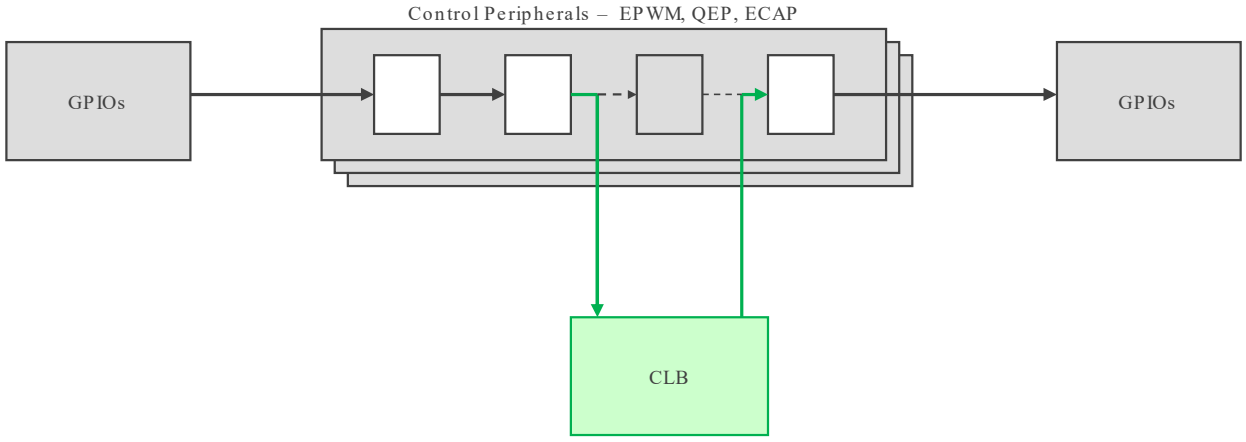

图 2-1. 在控制外设内部运行的 CLB

CLB 还可以将输出信号直接驱动到输出 XBAR（交叉开关），这种方式可以在不影响控制外设功能的情况下运行，在输出 XBar 中，可以将这些信号定向为通过选定的 GPIO 引脚输出器件。CLB 的输入不必仅来自控制外设，它们还可以源自其他外设、CPU 信号、CPU 寄存器位和 GPIO。如果 GPIO 是 CLB 的唯一输入，并且 GPIO 是 CLB 的唯一输出，则 CLB 会成为实现以往可能驻留在 FPGA 或 CPLD 中的外部固化逻辑的工具（请参阅图 2-1）。

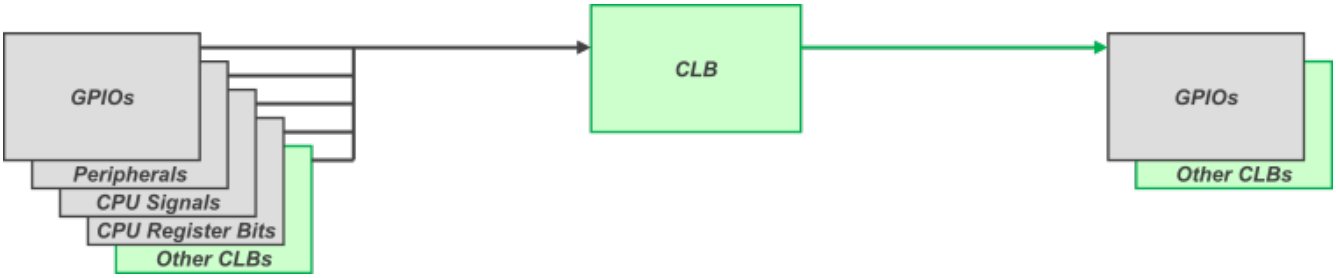

图 2-2. 在控制外设外部运行的 CLB

本文档中介绍的所有 CLB 用例本质上都是这两种运行模式的某种组合（请参阅图 2-3）。混合和匹配各种输入和输出的功能使 CLB 成为 C2000 系列的非常灵活强大的补充。以下各节提供了更多系统级信息，以说明 CLB 如何适合 C2000 芯片的其余部分。然后详细考察 CLB 构建块。接下来展示 CLB 的一些最常见示例，包括各种类型的输入和输出用法。。最后，第一个示例将会被更详细的介绍，展示如何将两个 PWM 发生器和相关的胶合逻辑从原来的外部 FPGA 移植进 C2000，以及两种实现方式如何产生相同的结果。

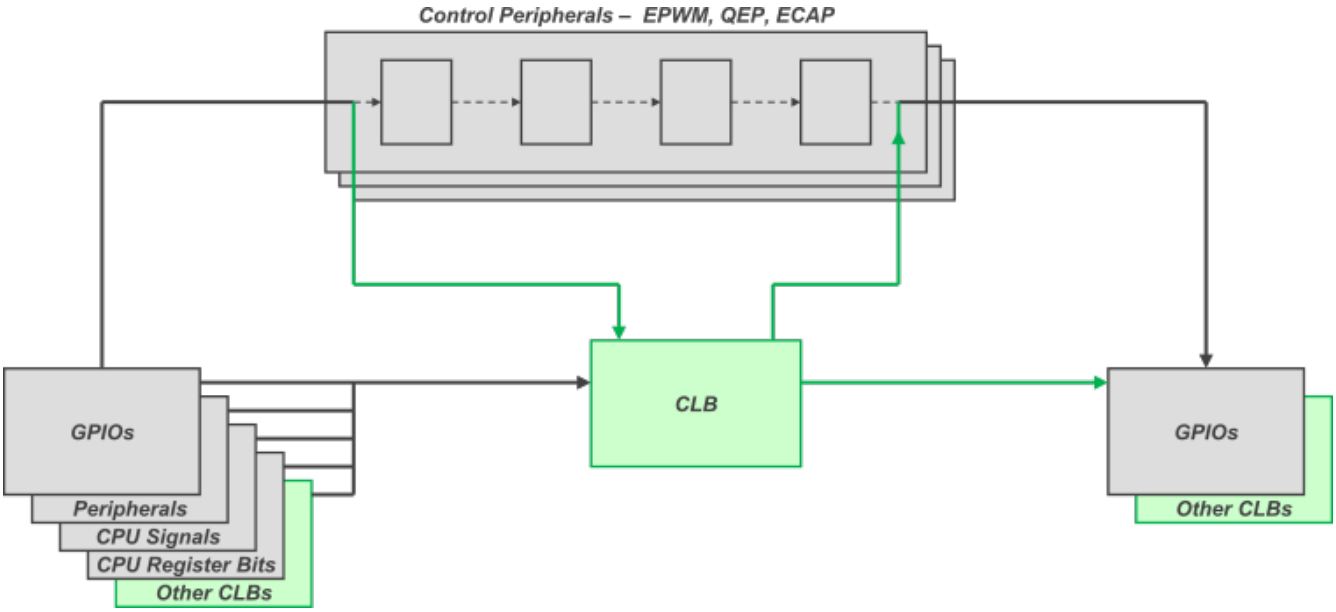

图 2-3. 在控制外设内部和外部运行的 CLB

2.2 CLB 的系统级视图

图 2-4 显示了 C2000 外设、XBar 和 GPIO 多路复用器。可以将该图与图 2-5 进行对比，后者显示了相同的外设，外加 CLB。通过比较这两个图，您可以确切地看到 CLB 触及了 C2000 的哪些部分。在其基本级别的形式中，CLB 由 CLB1、CLB2、CLB3 和 CLB4 块构成。每个 CLB 块都具有与相应控制外设的专用连接。例如，CLB1 连接到 EPWM1、QEP1 和 ECAP1。同样地，CLB2 连接到 EPWM2、QEP2 和 ECAP2 等等。此外，所有 CLB 模块都连接到称为全局信号的一组共享输入信号，这些信号来源于全部四个 EPWM 模块和 CLB XBar。每个 CLB 块还能够驱动 CLB INTR 信号来中断 CPU 或 CLA。最后，四个 CLB 块中的每个块都挂在 CPU 和 CLA 总线上，从而使 CPU 和 CLA 能够访问 CLB 配置和数据寄存器。

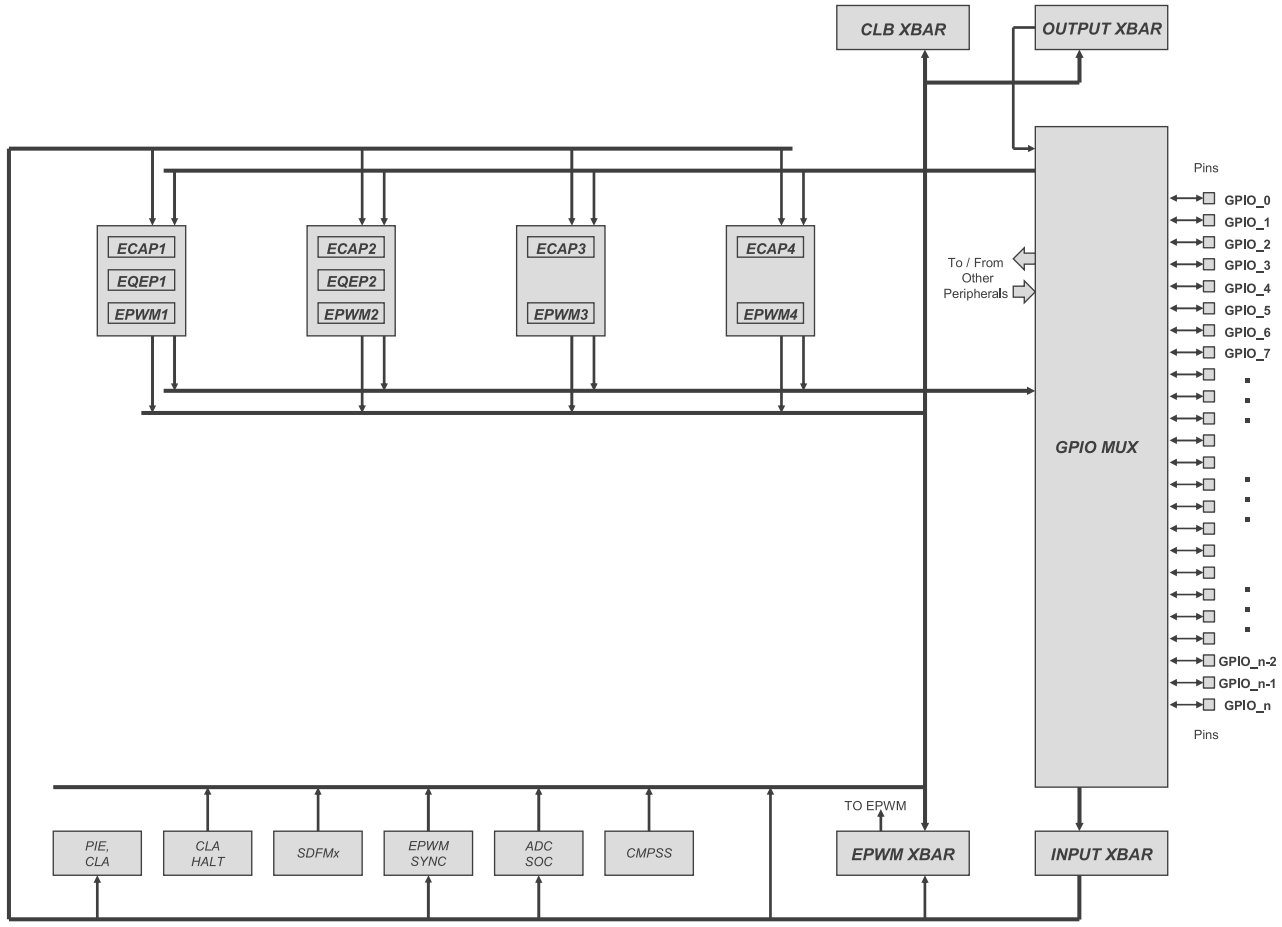

图 2-4. 显示了不带 CLB 的 C2000 外设

查看图 2-5，可以发现专用的 CLB 信号包含本地输入（黑色），CLB 输出（绿色）和外设信号多路复用器控制信号（橙色）。例如，CLB1 的本地输入包含来自 EPWM1、QEP1 和 ECAP1 的信号混合以及两个 CPU 信号 – CPU TBCLKSYNC 和 CPU HALT。CLB1 使用全局信号和本地 1 信号来生成 CLB1 输出信号。根据相应（橙色）外设信号多路复用器的设置状态，CLB1 输出信号可以注入 EPWM1、QEP1 或 ECAP1 控制外设中，或者被这些外设忽略。换句话说，（绿色）CLB 输出信号是替换信号，将替代控制外设中的原始信号注入，而（橙色）外设信号多路复用器选择使用相应的 CLB 输出信号替换控制外设的哪些内部信号（或不替换）。最后，一些（绿色）CLB 输出信号也被路由到 CLB/OUTPUT XBar，在此处它们可以通过 GPIO 多路复用器发送到选定的器件引脚，或者输入全局输入信号以成为四个 CLB 模块中任何一个的输入。

CLB2、CLB3 和 CLB4 的工作方式相同，只是 CLB3 和 CLB4 所使用的本地信号较少（缺少 QEP3 和 QEP4 外设，并且未连接到 CPU TBCLKSYNC 和 CPU HALT）。

在内部，四个 CLB 块均由三个相同的构建块组成，这些构建块是 CLB 输入选择器、CLB 逻辑块和外设信号多路复用器。CLB 输入选择器从全局和本地总线中选择八个信号。每个 CLB 逻辑块将逻辑方程应用于八个输入，以按通过配置寄存器预先配置逻辑块的方式来驱动多达八个输出信号。最后，来自 CLB 逻辑块的 8 个输出被馈送到外设信号多路复用器中，在该多路复用器中可以选择这些输出以替换（或不替换）相应控制外设内 14 个可能内部信号中的 8 个。总的来说，CLB 的三个构建块的操作由相应的 CLB 配置寄存器进行控制。CLB 输入选择器和 CLB 外设信号多路复用器的配置寄存器加载通过功能调用完成，而 CLB 逻辑块的配置寄存器通过基于 GUI 的 SysConfig 工具生成的代码进行加载。以下各节更详细地说明各个 CLB 构建块如何协同工作以修改控制外设，实现新外设，或使用选定的 GPIO 来生成胶合逻辑。

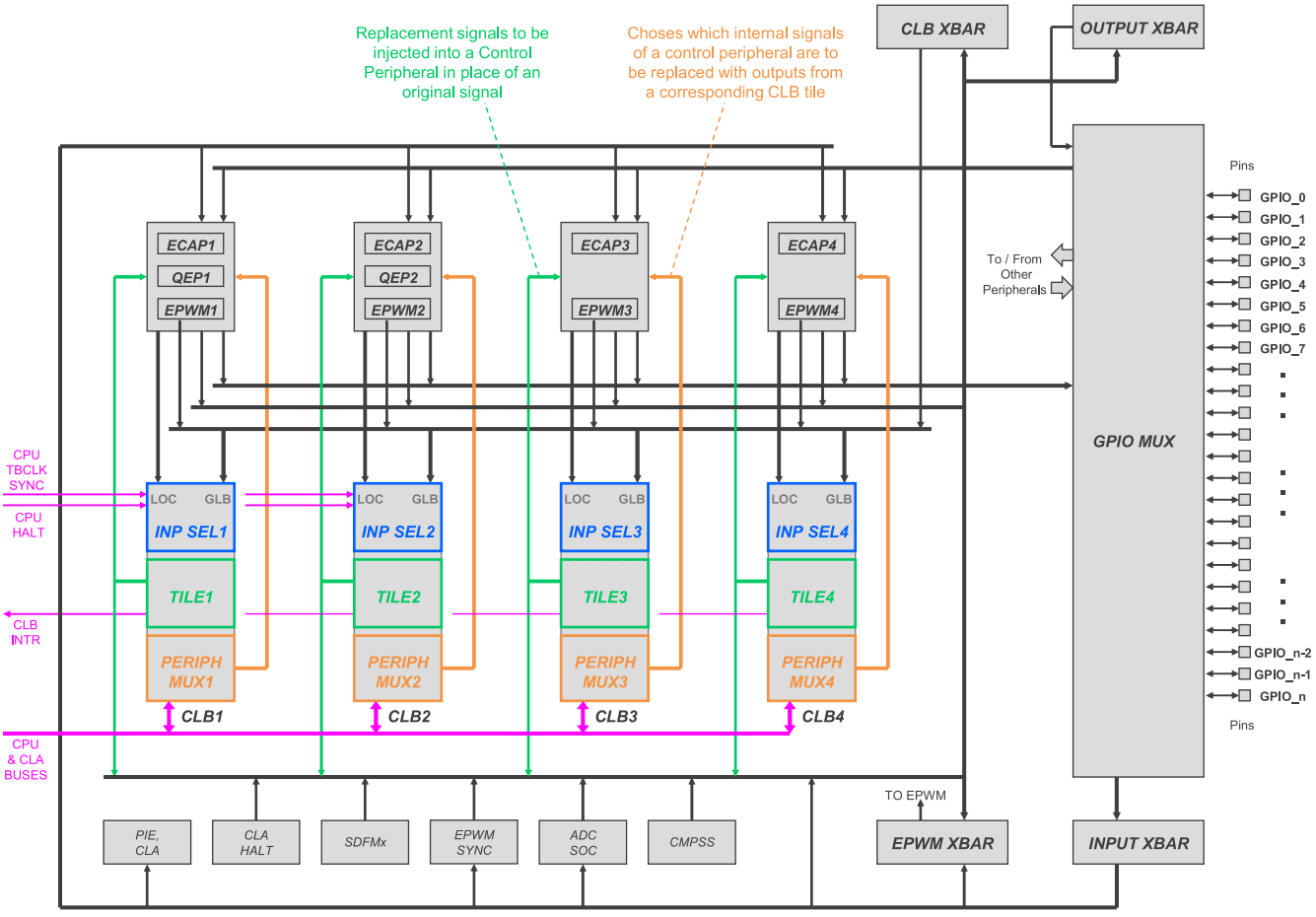

图 2-5. 显示了带 CLB 的 C2000 外设

2.3 深入探讨 CLB 架构

图 2-6 所示为四个 CLB 模块的其中一个和相关控制外设（EPWM1、QEP1 和 ECAP1）的内部细节。输入包括全局信号，本地 1 信号和 GPREG 寄存器的 8 个位。根据输入多路复用器配置寄存器预先配置的方式，输入多路复

用器选择器从这些输入中选择最多 8 个信号（蓝色），并将其发送到逻辑块 1 进行处理。根据逻辑块 1 配置寄存器的状态，将预设的逻辑操作应用于这 8 个输入，以生成 8 个 CLB1 输出信号（绿色）。这些信号都被路由到 EPWM1、QEP1 和 ECAP1 外设，其中有几个信号还到达 CLB XBar、输出 XBar 和 EPWM XBar，在此处它们被发送到 EPWM 模块、GPIO 或反馈到四个 CLB 逻辑块中的任何一个，以供进一步处理（通过全局信号总线）。

最后到达 EPWM1、QEP1 和 ECAP1 控制外设的 CLB1 输出信号可用于取代这些外设内的选定内部信号，具体取决于外设信号多路复用器寄存器的配置方式。外设信号多路复用器（橙色）控制位于控制外设内部各等级的多路复用器，用于决定将哪些信号传播到下一级 – 原始内部信号（黑色）或来自逻辑块 1 的新替换信号（绿色）。该方法提供了极大的灵活性，可替换外设的给定级或添加新的级（使用 CLB 逻辑）。例如把 ECAPI 的输入立即馈送至逻辑块 1 并且在 ECAP1 的最后一个级处注入逻辑块 1 输出的极端情况下，逻辑块 1 内部的逻辑成为新的自定义外设，完全取代原始 ECAP1 外设。请注意，尽管现在已替换了 ECPAP1 的内部，但 ECAP1 的 GPIO 多路复用器输入和输出分配保持不变，这意味着新外设必须使用与刚刚取代的 ECAP1 相同的输入和输出 GPIO。

换句话说，在对 CLB1 进行编程时，应首先选择一个要修改的控制外设，然后确定该外设的哪些内部信号将由 CLB1 逻辑块的输出进行替换（使用外设信号多路复用器）。在确定这些信号之后，应使用输入多路复用器选择器选择 CLB1 的输入，新 CLB1 功能的逻辑将需要这些输入来生成所需的输出信号。在建立 CLB1 的输入和 CLB1 的输出（使用功能调用）之后，就可以使用基于 GUI 的 SysConfig 工具对输入配置必要的逻辑以生成输出。

请注意：CLB 配置功能调用和 SysConfig 工具都通过 CPU 或 CLA 总线（品红色）将配置状态加载到 CLB1 配置寄存器中。相同的总线也可用于在逻辑块内的 HLC 处理器与 C2000 存储器之间传输数据。HLC 处理器还可以基于 CLB 内部达到的某种预定条件来驱动 CLB INTR 中断 CPU（稍后会进一步介绍）。此外，如果 CLB 未修改任何控制外设（在未使用 CLB 或 CLB 仅将信号驱动到 CLB XBar 和其他 Xbar 中时会发生该情况），则可以忽略外设信号多路复用器。以下各节提供有关输入多路复用器选择器、外设信号多路复用器和 CLB 逻辑块的更多详细信息。

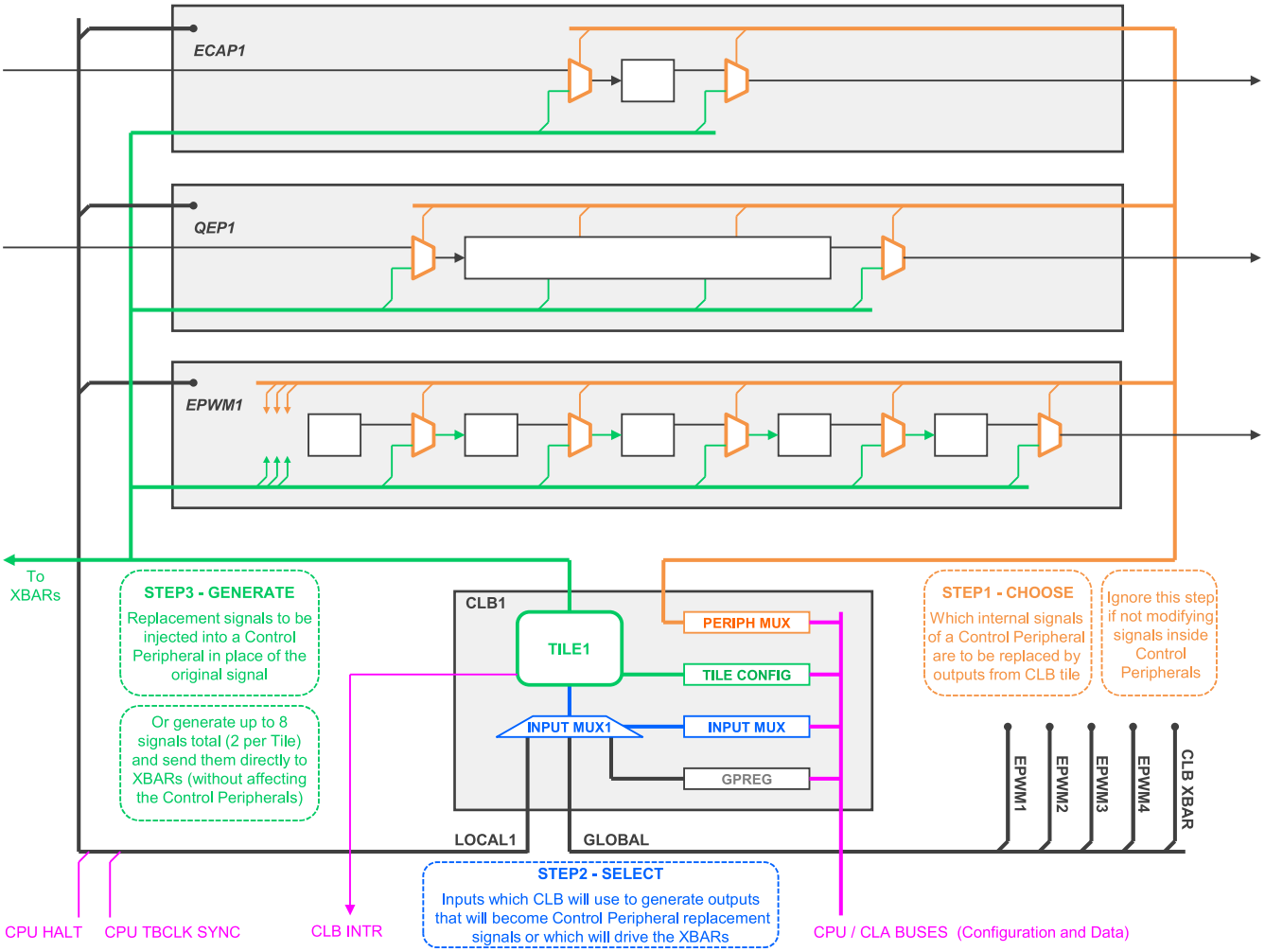

图 2-6. CLB 连接 – CLB1

2.3.1 输入多路复用器

图 2-7 所示为逻辑块 1 的输入多路复用器选择器的详细视图。其他逻辑块的输入多路复用器选择器是相同的，但有几处例外 – 逻辑块 3 和 4 的本地输入信号较少，GPREG 的 32 个位平均分布在四个逻辑块之间（每个逻辑块一个字节，低字节分配给逻辑块 1）。输入多路复用器选择器 1 具有八个相同的部分，每个部分产生一个输出，总共有八个 CLB1 输出信号。在每个部分内部，有一系列多路复用器（共 4 个），从全局多路复用器开始，该多路复用器从 72 个可用输入中选择 1 个位。该位成为下一级的位 0，本地 1 输入信号组提供第 2 级的其余 25 个位。第 2 级的一位输出保持原样地或在通过时钟同步电路之后被传递到下一级。第 3 级的输出通过一个滤波器，在该滤波器中它可能变为上升沿脉冲、下降沿脉冲或保持不变。在最后的第 4 级中，第 3 级的输出可由 GPREG 寄存器中的 8 个位之一替换。同样，输入多路复用器选择器由输入多路复用器配置寄存器通过功能调用进行控制。有关全局和本地信号分配，请参阅特定于器件的技术参考手册 (TRM) 表。

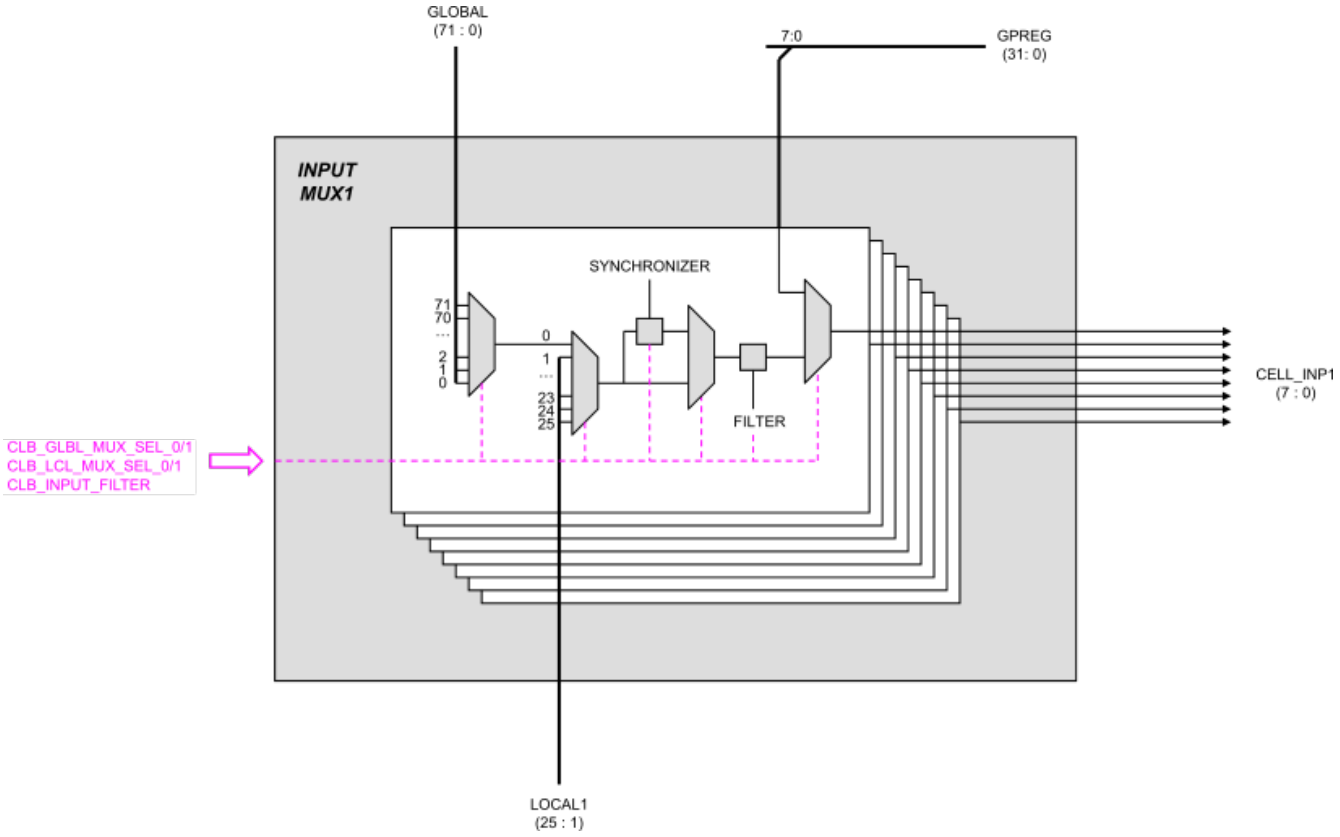

图 2-7. CLB1 的输入选择

2.3.2 （输出的）外设多路复用器

图 2-8 显示了外设信号多路复用器的功能视图，其中八个 CLB1 输出可替换所选控制外设内的多个信号。

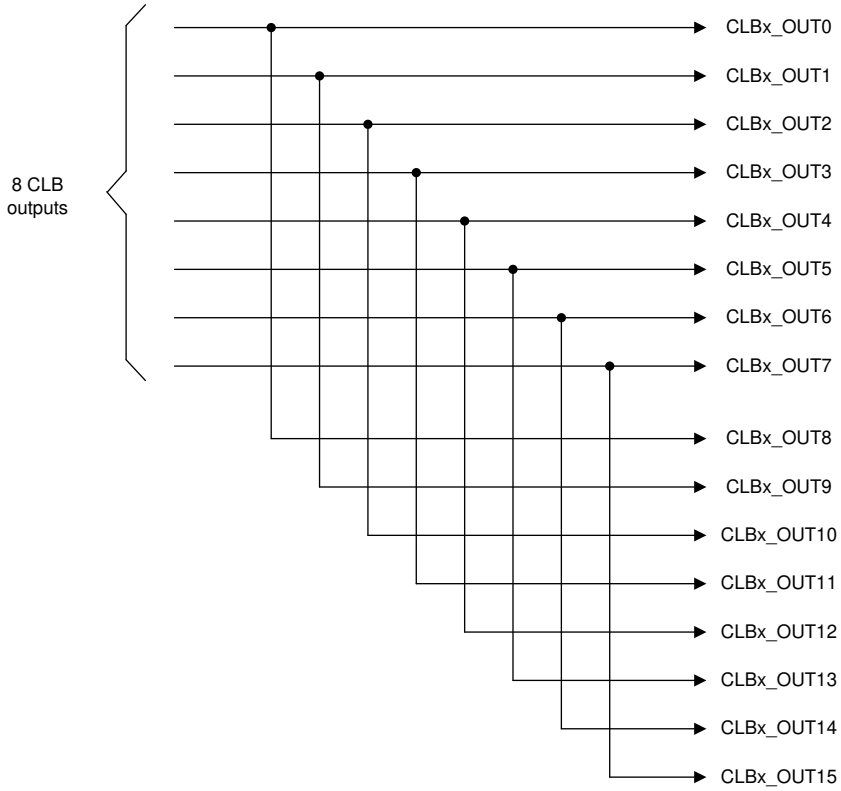

图 2-8. CLB1 输出

图 2-9 显示了 CLB1 外设信号多路复用器的另一个视图，信号中包含进行了不同颜色表示的的源和目标。CLB1 输出八个信号，其中信号 4 和 5 成为 CLB XBar、输出 XBar 和 EPWM XBar 的直接输入。利用外设信号多路复用器，全部 8 个 CLB1 输出都可用于替换控制外设组 1 的 14 个内部信号中的任何一个，包括 EPWM1、QEP1 和 ECAP1。例如，CLB 输出信号 0-7 可以替换八个 EPWM1 内部信号，CLB 输出信号 0-3 可以替换四个 QEP 内部信号，CLB 输出信号 6-7 可以替换两个 ECAP 内部信号。要确定哪个特定的 CLB 输出信号替换 EPWM1、QEP1 和 ECAP1 外设的哪个内部信号，请参阅特定于器件的技术参考手册 (TRM) 表。类似的信号分配适用于 CLB2、CLB3 和 CLB4，它们各自向控制外设组 2、3 和 4 提供八个专用信号。

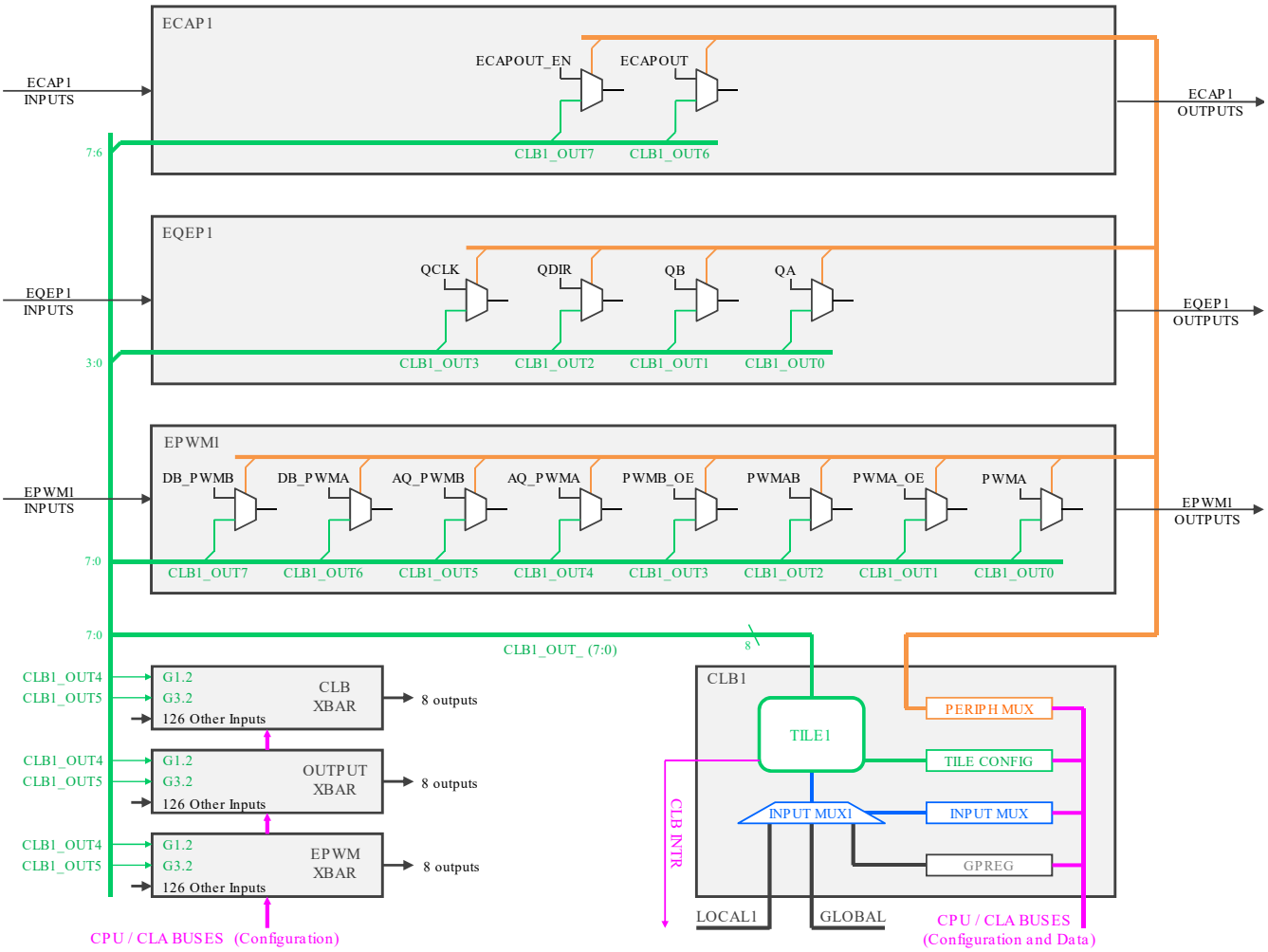

图 2-9. CLB1 的外设信号多路复用器

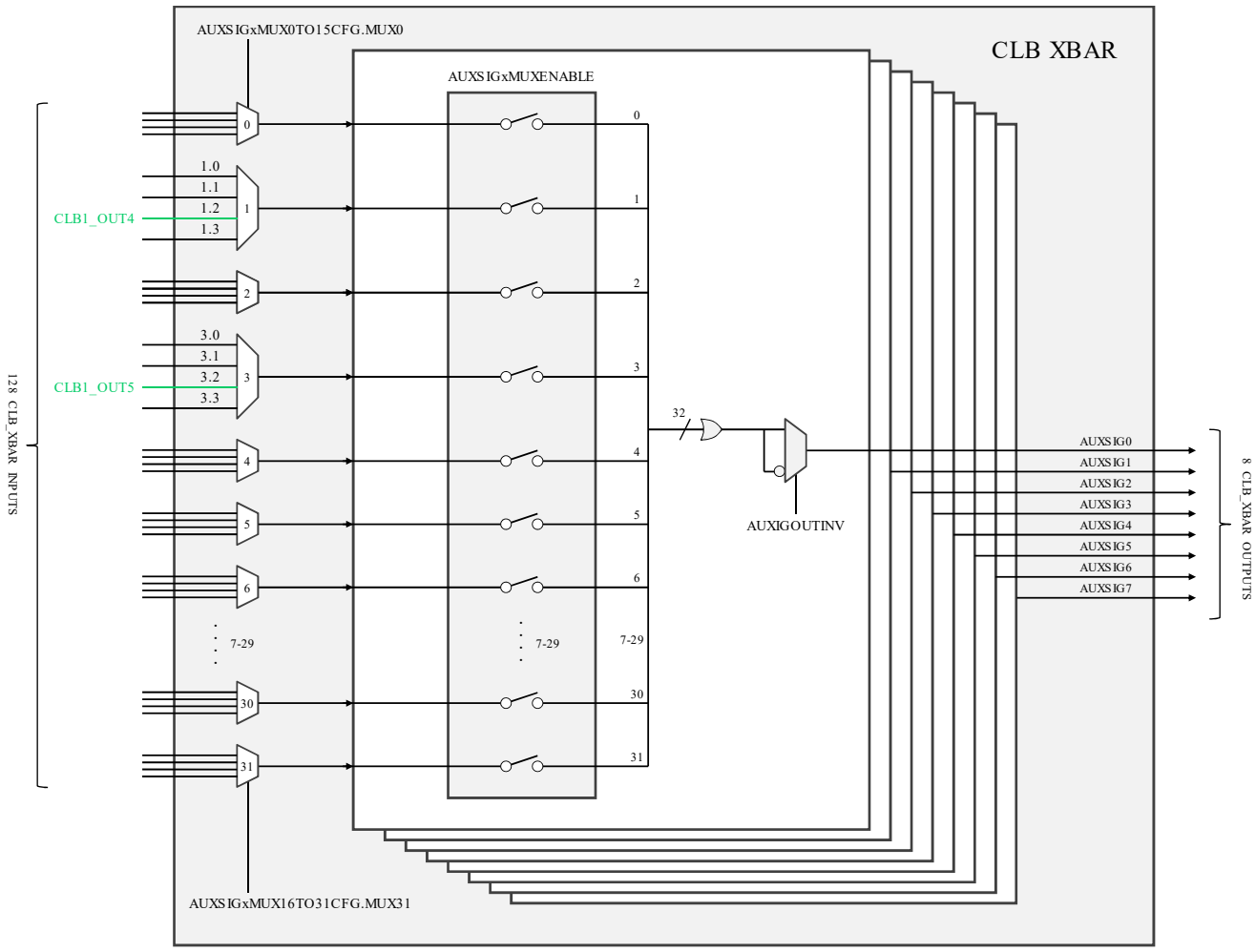

图 2-10. CLB XBar

2.3.3 CLB 逻辑块

基本级别配置中存在的四个 CLB 模块中的每一个都由一个输入信号选择器、一个 CLB 逻辑块和一个外设信号多路复用器组成。输入信号选择器选择进入 CLB 逻辑块的八个信号，外设信号多路复用器分配 CLB 逻辑块的八个输出。在逻辑块内部，逻辑运算被应用于八个输入（由配置寄存器的内容决定），以生成八个输出信号。这些逻辑运算可以由 HLC 块（高级控制器）、三个计数器、三个 LUT4 块（具有四个输入的查找表）、三个 FSM 块（有限状态机）和八个输出 LUT3 块执行（请参阅图 2-11）。这些逻辑基元中的每一个都由位于 CLB 逻辑块寄存器文件内的相应配置寄存器进行控制，该寄存器文件还包含数据交换寄存器和控制寄存器。C2000 CPU 使用数据交换寄存器来间接访问 HLC 寄存器和计数器寄存器。C2000 CPU 还使用数据交换寄存器来加载 HLC 程序存储器。CPU 使用 SysConfig 工具生成的代码来加载逻辑块配置寄存器。在配置之后，CPU 或 CLA 根据应用代码的指示访问数据交换寄存器和控制寄存器。

图 2-11 显示 CLB 逻辑块的主要特性之一是一条 32 位的逻辑总线（棕色），该总线从所有上述逻辑基元中收集结果位。该总线的最低 8 位来自 CLB 逻辑块的八个输入。随后，全部 32 个位都可用作所有逻辑基元的输入，从而可以轻松共享中间逻辑结果，同时通过八个输入构建八个输出。例如，来自计数器的输出可能在 LUT4 块内与 FSM 块的输出进行“或”运算，以生成进入输出 LUT3 的信号，从而成为 CLB 逻辑块的输出之一。HLC 还可以使用此 32 位内部逻辑总线，在最多四个选定逻辑总线信号从低到高转换时触发最多四个不同的程序。最初通过 SysConfig 工具生成的代码将这四个 HLC 程序预加载到 HLC 指令存储器中，并且每个程序最多可包含 8 条 HLC 指令。

逻辑基元主要用于处理单个信号，而 HLC 可以同时处理信号向量（信号组），例如在四个 HLC 通用寄存器（R0、R1、R2、R3）和三个 CLB 计数器内的选定寄存器（计数、匹配 1、匹配 2）之间交换数据。也可以通过 HLC ADD 和 SUB 指令来修改四个通用寄存器和计数器寄存器。其他指令执行将 HLC 寄存器或计数器寄存器中的数据推入到 4 字 FIFO 中的操作，或执行将 4 字 FIFO 中的输入拉出到 HLC 寄存器或计数器寄存器中的操作。4 字 FIFO 是与主机系统共享的 HLC 数据存储器的一种形式，有助于在 HLC 与 C2000 存储器之间移动数据。图 2-11 中以蓝色突出显示了 HLC 使用的数据移动路径。

在 C2000 侧，可通过存储器映射读取和写入来访问拉/推 FIFO。逻辑块逻辑配置寄存器、控制寄存器和数据交换寄存器，则可以直接访问进行读取和写入。在图 2-11 中，这些 CPU/CLA 传输的路径以品红色突出显示。

CPU 和 CLA 使用数据交换寄存器向 HLC 中无法通过存储器映射读取和写入直接访问的其他资源进行写入（但不读取）。这其中包括 HLC 指令存储器、四个 HLC 通用寄存器和三种类型的计数器寄存器：计数、匹配 1 和匹配 2。CPU 和 CLA 通过数据交换寄存器对 HLC 资源的这些间接访问的工作方式如下：首先，将写入地址加载到 CLB_LOAD_ADDR 寄存器中，接下来将要传输的数据写入到 CLB_LOAD_DATA 寄存器中，然后对 CLB_LOAD_EN 寄存器进行写入。最后的写操作在 HLC 侧触发写周期，通过本地接口总线完成数据传输。在图 2-11 中，该总线以绿色突出显示。

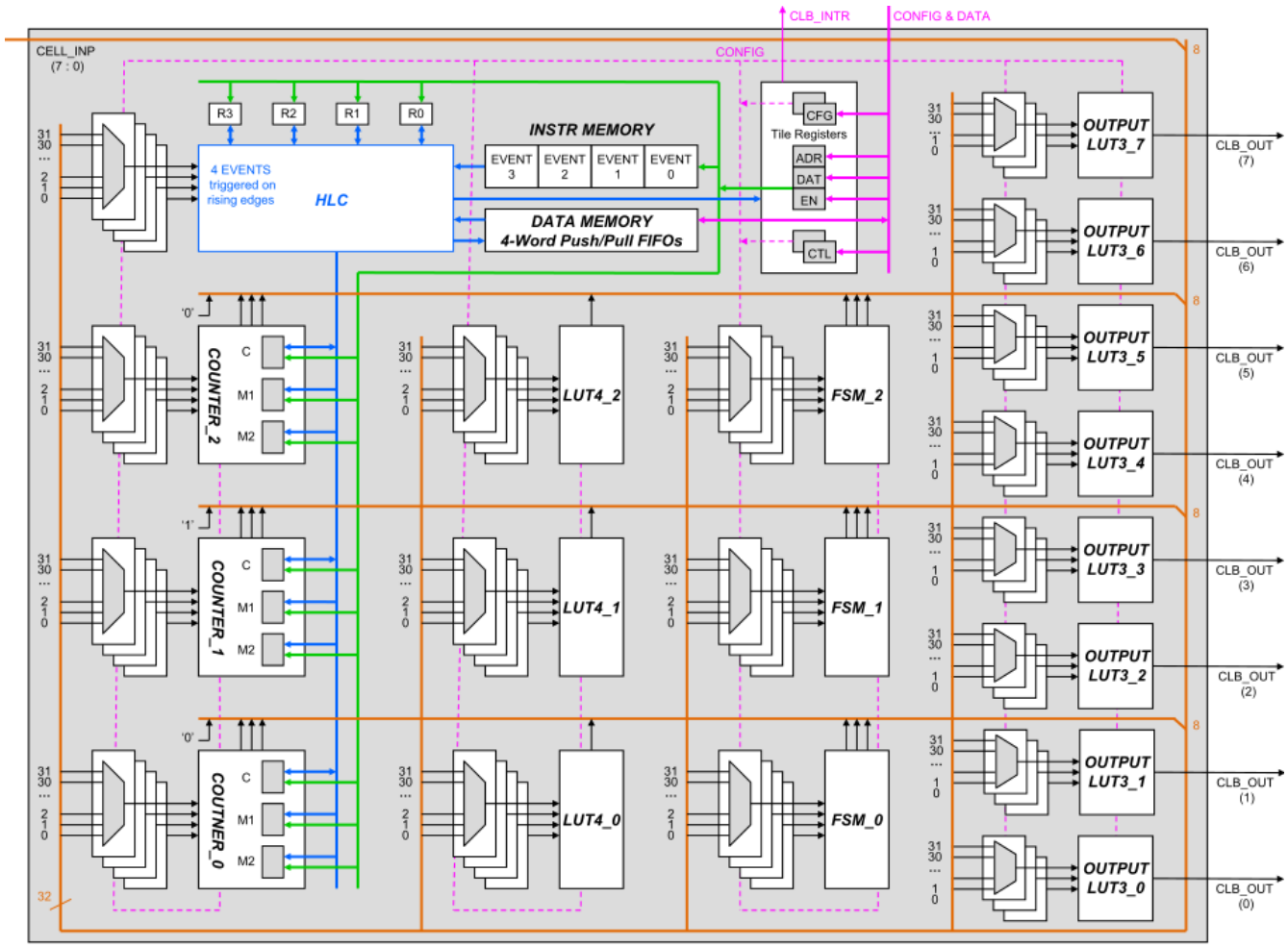

图 2-11. CLB 逻辑块

以下各节重点介绍构成 CLB 逻辑块的三种逻辑基元（查找表、有限状态机和计数器）的内部细节。

2.3.3.1 查找表 (LUT)

图 2-12 显示了 CLB 逻辑块中使用的两种 LUT：4 输入 LUT4 和 3 输入 LUT3。每个查找表都有一个输出，最后的写操作在 HLC 侧触发写周期，通过本地接口总线完成数据传输。这些配置寄存器通过 SysConfig 工具生成的代码进行加载。SysConfig 工具还选择逻辑总线的 32 个位中的哪些位用作 LUT 的输入。查找表对输入进行严格的组合逻辑，以生成输出（不存在计时寄存器）。例如，OUT 输出可以是 IN0 输入与 IN1 输入的逻辑或，然后和 IN2 输入进行逻辑与运算。

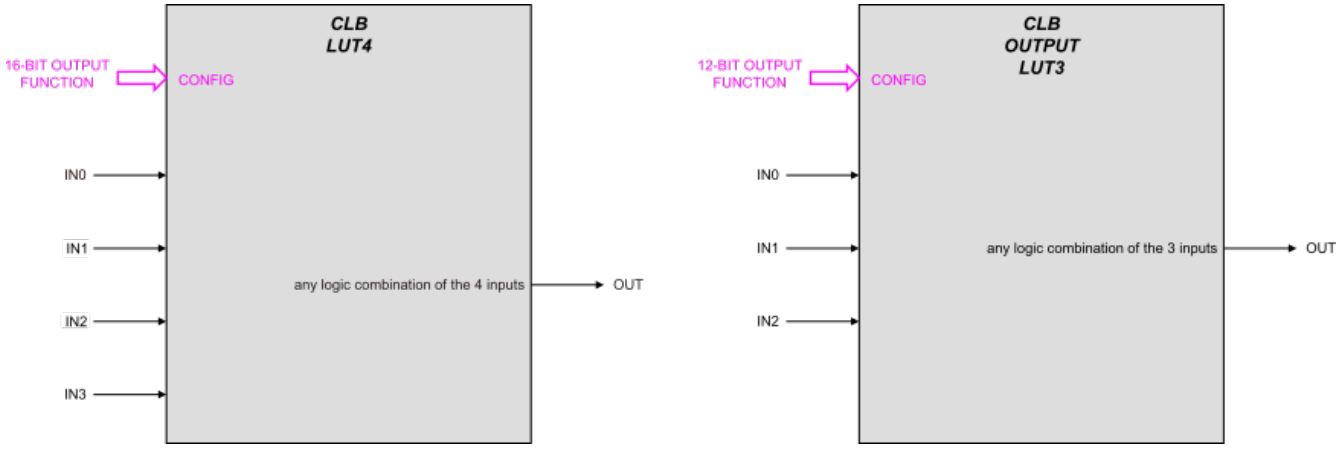

图 2-12. CLB 查找表

2.3.3.2 有限状态机 (FSM)

图 2-13 显示了 CLB 逻辑块的有限状态机块。FSM 由三个 LUT4 组合块和两个寄存器位（S0 和 S1）组成，由 CLB 时钟对其进行计时。两个 LUT4 块用于馈送 S0 和 S1 的新状态，FSM EXT_IN0 和 EXT_IN1，以及 S0 和 S1 的旧状态则作为 LUT4 的输入。在 CLB 时钟的每个上升沿，都会更新新状态（根据相应 LUT4 内部的逻辑）。S0 和 S1 状态也可用作 FSM 模块的输出，以成为共享 CLB 逻辑总线的 32 个位中的 2 位。第三个 LUT4 块以两种可编程模式运行。在一种模式下，它获取与其他两个 LUT4 块相同的输入。在另一种模式下，S0 或 S1 输入由 EXTRA_EXT_IN0 和 EXTRA_EXT_IN1 输入替换，以扩展可用于驱动 FSM_LUT_OUT 输出的逻辑组合的数量。该 FSM 输出也成为共享 CLB 逻辑总线的一部分，通过该总线可以将其馈送至其他逻辑块和 HLC。用于确定三个 LUT4 块和两个模式多路复用器内部逻辑方程的控制位由相应的 CLB 配置寄存器进行控制，这些寄存器又由 SysConfig 工具生成的代码进行配置。

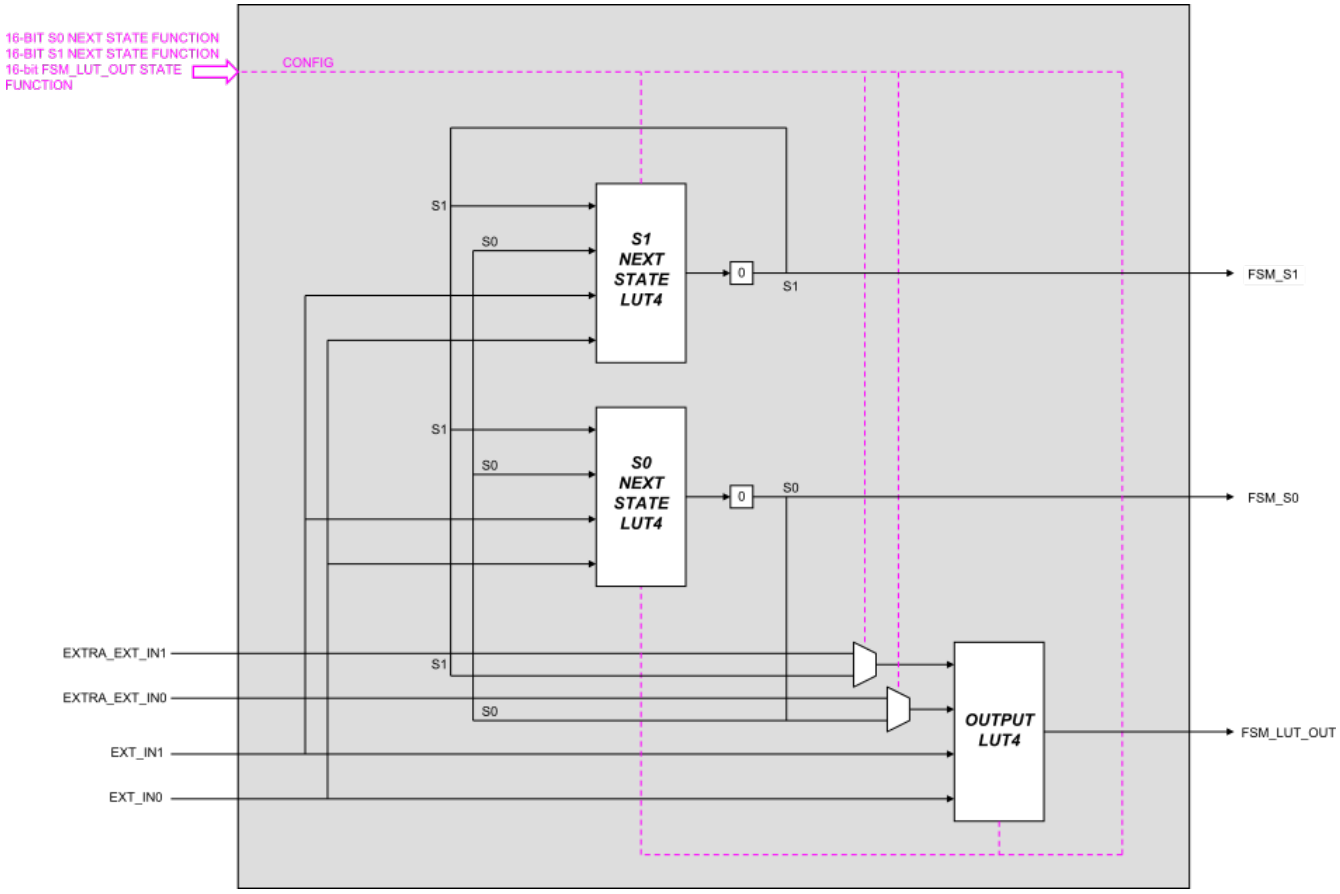

图 2-13. CLB FSM 块

2.3.3.3 计数器

图 2-14 显示了每个 CLB 实例内部可用的三个计数器之一。计数器的核心是具有计数寄存器和加法器的计数循环，该加法器根据 MODE1 输入的状态使计数寄存器递增或递减 1。计数循环还包含一个计数器多路复用器，用于使用先前计数值的递增或递减版本以外的值初始化计数寄存器。例如，可以使用新值、当前计数值的移位版本或当前计数加/减 32 位值寄存器的内容版本来初始化计数寄存器。当 MODE0 输入置位，并且 ENB 信号置位时，计数寄存器只能在 CLB 时钟的上升沿递增或递减。使用上述三个初始值之一加载计数寄存器需要 EVENT 输入从高到低转换。这会产生一个单时钟脉冲，该脉冲会立即将计数器多路复用器从递增/递减模式切换到加载模式，向计数寄存器加载新值。最后，无论其他输入的状态如何，复位输入都会将计数寄存器的内容初始化为零。计数器的全部四个输入都可以根据 SysConfig 工具先前配置的相应控制寄存器从 32 位共享逻辑总线中进行选择。

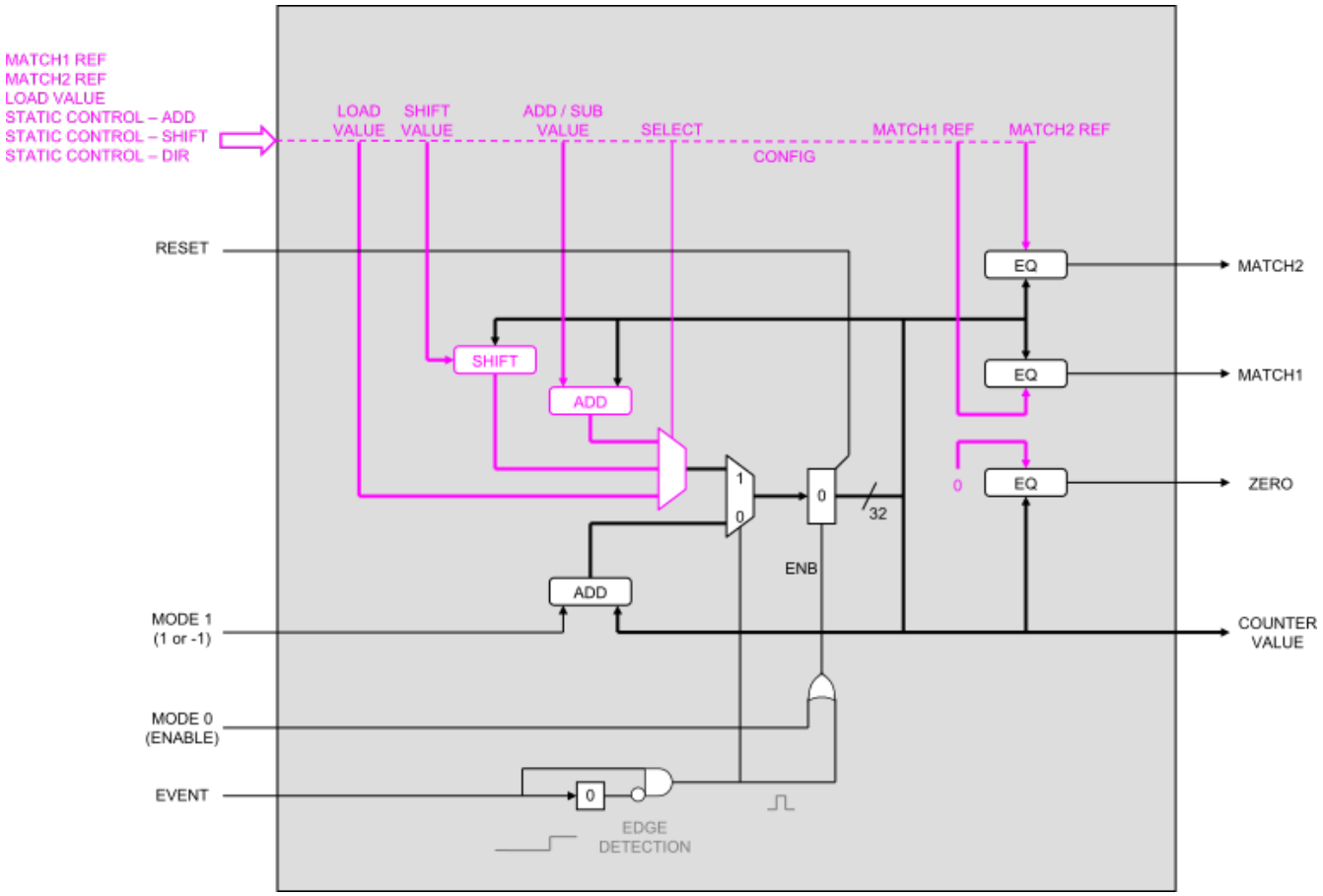

图 2-14. CLB 计数器块

计数器块的输出包括当前计数器值（32 位宽）和来自零比较器、匹配 1 比较器和匹配 2 比较器的三个 1 位状态信号。只要当前计数值与给定比较器关联的比较值相匹配，这些比较器就会发出逻辑 1 脉冲。三个 1 位计数器输出映射到 32 位共享逻辑总线的三个预定义位，以后可以用作其他计数器、LUT、FSM 和 HLC 的输入。

HLC 可以直接访问 32 位计数器值，两个匹配寄存器也是如此。C2000 CPU/CLA 也可以通过数据共享寄存器间接加载相同的寄存器。

3 CLB 用例概述

以下是四个 CLB 示例，其中展示了将自定义逻辑应用于 C2000 器件内部和外部信号的各种方法。每个示例均以三个视图显示，它们共同展示了 CLB 各个部分如何运行以及有效信号如何在 CLB 与芯片的其他部分之间流动。第一个视图是简单的原理方框图，仅限于显示给定示例的相关信号及其流经的受影响芯片组件。第二个视图提供相同传输的系统视图，其中突出显示的信号和相关的块叠加在表示整个芯片输入/输出部分的暗灰色背景上。通过该视图，可以立即了解需要通过功能调用和 SysConfig 工具配置芯片的哪些部分，以实现特定的自定义逻辑应用。最后，第三个视图更深入地介绍了 CLB 和控制外设，以准确显示正在使用 CLB 的哪些部分以及它们如何与控制外设进行交互。同样，突出显示的信号和相关的块会叠加在 CLB 和控制外设周围局部空间的暗灰色背景上，以便更好地了解底层的自定义逻辑操作。

3.1 CLB 示例 16 – 将两个 EPWM 输出与来自 CPREG 寄存器的信号组合在一起

在该示例中，自定义逻辑应用于两个 EPWM 块的输出以及 GPREG 寄存器中的一个位，产生的输出信号通过 GPIO 引脚退出器件。图 3-1 显示了该示例的功能原理图。查看信号，可以看到 EPWM1 和 EPWM2 模块的输出与 CLB1 内的 GPREG 位结合在一起，并将结果反馈到 EPWM1 中，以转发至 GPIO 多路复用器，随后再转发至标准 EPWM1 输出通常使用的器件引脚。请注意，CLB1 的输出不会直接路由到 GPIO 多路复用器，而是会在输出级返回到 EPWM1 模块，从输出级开始，芯片的其余部分像对待正常 EPWM1 输出一样处理它。另请注意，进入 CLB1 的 EPWM 信号是在输出级之前从其各自的 EPWM 模块中提取的，但它们被重定向到 CLB1，而不是被发送到 GPIO 多路复用器。该实例在 FPGA 和 C2000 MCU 都实际运行后，两者的结果是一致的，请参阅节 4。

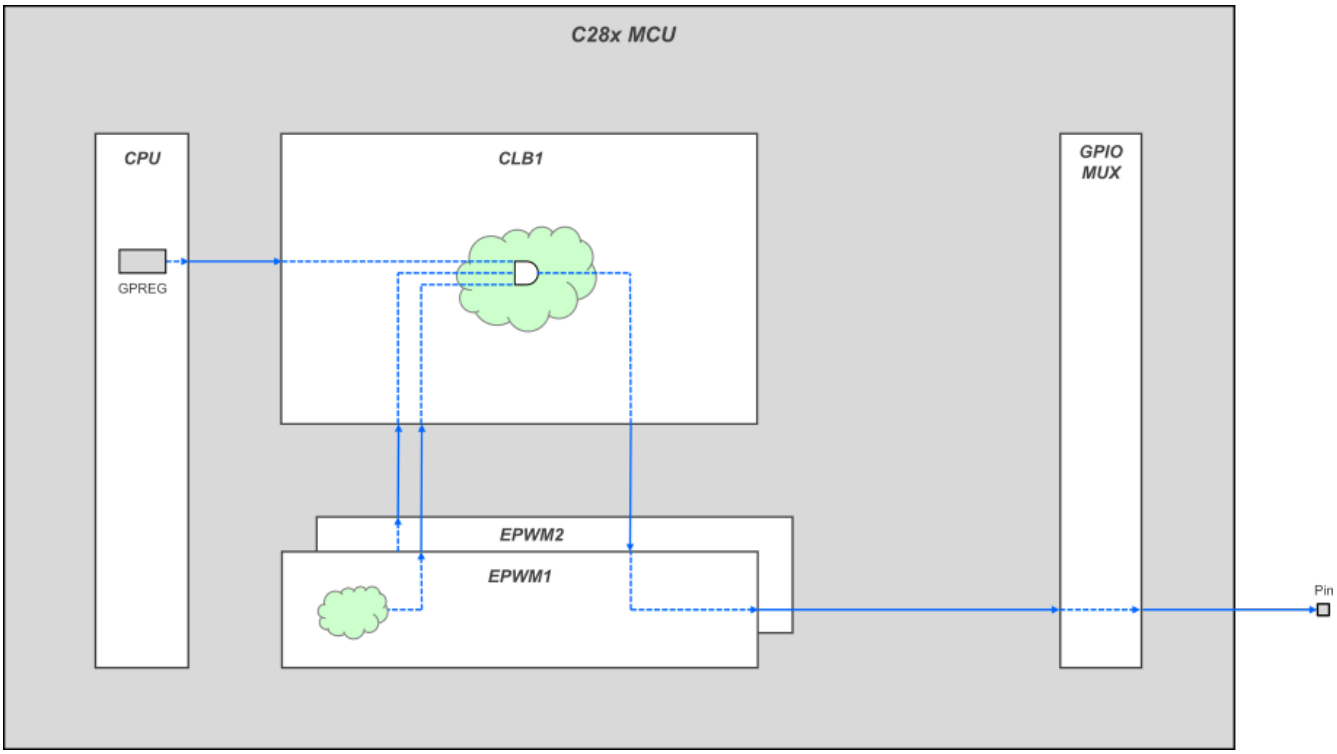

图 3-1. CLB 示例 16 – 合并两个 EPWM 输出和一个来自 GPREG 寄存器的信号

图 3-2 通过器件级视图显示了相同的示例，以在 C2000 器件 I/O 部分的暗灰色背景下突出显示活动块和产生的数据传输。此处同样显示，通过使用全局信号总线将生成的 EPWM1A 和 EPWM2A 信号转移到 CLB1 中。请注意，EPWM1B、EPWM2A 和 EPWM2B 信号从相应的 EPMW 模块直接进入 GPIO 多路复用器。在 CLB1 内部应用了自定义逻辑之后，产生的信号通过 CLB1 输出总线反馈到 EPWM1 的最后一级。在返回到 EPWM1 模块内部之后，该输出信号立即退出，与正常 EPWM1A 输出一样转发到 GPIO 多路复用器。它从此处通过正常分配给 EPWM1A 的引脚退出芯片。

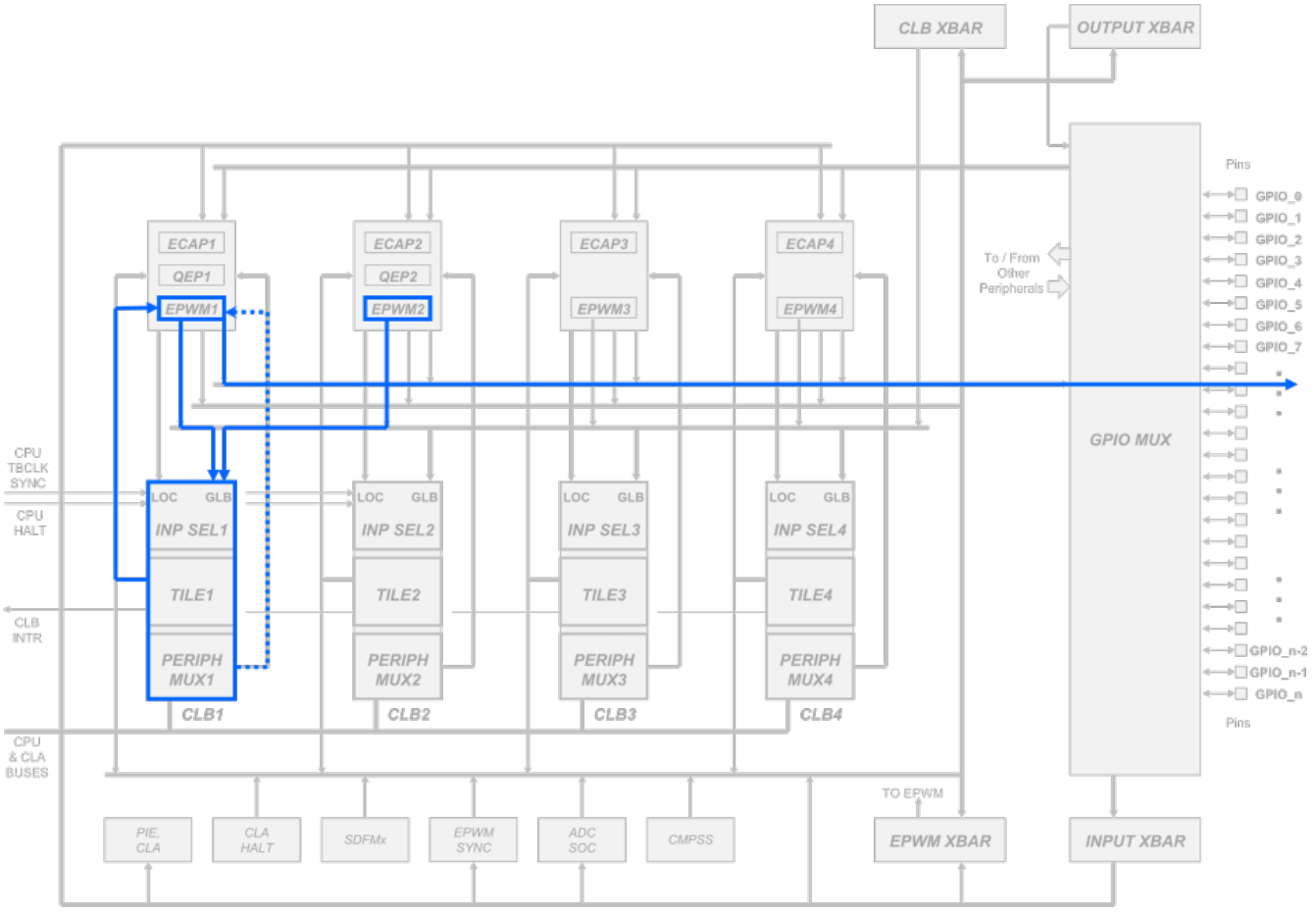

图 3-2. 示例 16 中的信号流 – 器件视图

图 3-3 通过 CLB1 连接的透镜提供了该示例的信号流详细信息。除了显示将 CLB1 与本地控制外设相连接的输入和输出之外，该视图还提供了 CLB1、EPWM1、QEP1 和 ECAP1 的内部详细信息及其内部的数据流。在进一步讨论之前，请注意，控制外设已在其内部嵌入了一组多路复用器，这些多路复用器允许来自 CLB1 本地输出总线的替代信号在预选的级注入到外设中（按照 CLB1 外设信号多路复用器控制的指示）。另请注意，可以在方便的位置从控制外设中提取信号，然后通过全局和本地 1 总线将其传送到 CLB1。

现在已做好准备，可以开始跟踪该示例的信号。首先，在 EPWM1A 信号即将离开 EPWM1 模块并进入全局信号总线之前，对该信号进行拦截。EPWM2A 信号也适用同样的操作。全局信号总线将这两个 EPWM 信号重新路由至 CLB1，其中的逻辑块 1 输入多路复用器选择这两个信号进入 TILE1。输入多路复用器还将 GPREG 寄存器中的一个位转发至逻辑块 1。这 3 个位一起进入 CLB1 的逻辑块 1，而连接到逻辑块 1 的 8 位输入总线的其余 5 个位未使用。在逻辑块 1 内部，将预定义的逻辑应用于 3 个输入，以产生一个输出信号（有八个可能的输出），并将其放置在 CLB1 输出信号总线上，该信号在此总线上返回到 EPMM1 模块。

返回到内部之后，输出信号由 EPWM1 模块最后一级后面的外设信号多路复用器进行选择，以退出 EPWM1 并开始向 GPIO 多路复用器传输，然后放置在分配给正常 EPWM1A 输出的引脚上。请注意，在此阶段，芯片的其余部分不知道也不关心从控制外设发出的信号是原始输出、CLB 修改的输出还是 CLB 产生的与原始外设无关的全新输出。这项强大的功能使 C2000 用户能够轻松地修改外设或创建新外设，而不必依赖外部 CPLD 或 FPGA，同时始终使用现有的芯片资源向/从新外设/修改的外设传递信号。

提醒一下，该示例（以及后面的示例）中的信号路径已由控制输入多路复用器选择器、逻辑块 1 配置和外设信号多路复用器的 CLB1 配置寄存器确定。输入多路复用器选择器和外设信号多路复用器控制信号由应用软件通过功能调用进行设置，而逻辑块 1 配置由 SysConfig 工具进行定义（生成代码，以响应用户与 SysConfig 图形用户界面的交互来设置逻辑块配置寄存器）。

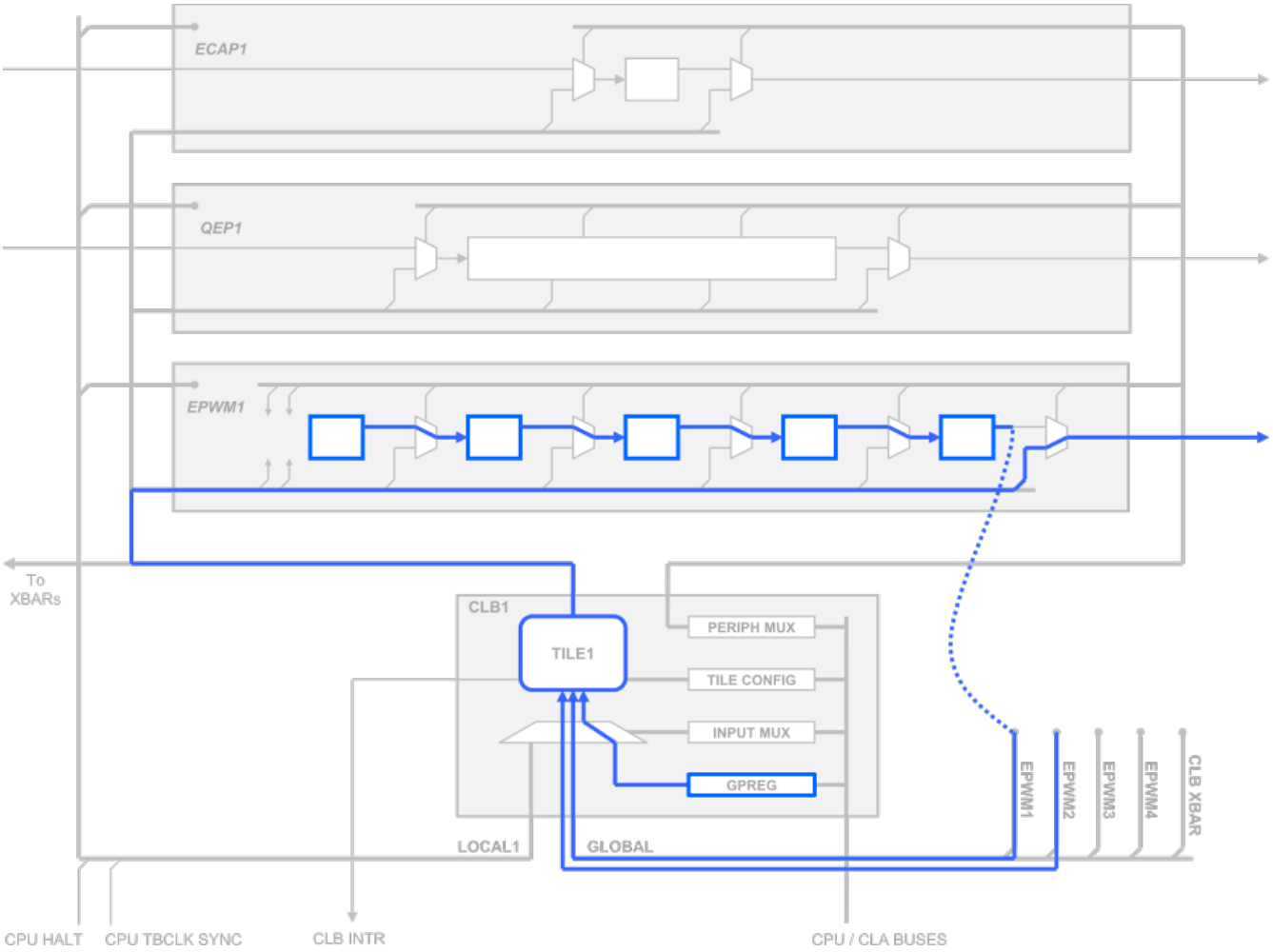

图 3-3. 示例 16 中的信号流 – CLB1 连接

3.2 CLB 示例 17 – 使用 CPU 信号修改外设输入信号

在该示例中，一些自定义逻辑应用于 QEP2 控制外设的输入。图 3-4 显示了该示例的功能原理图。查看信号，可以看到输入由 GPIO 引脚提供，然后输入从该引脚传递到 QEP2。输入信号进入 QEP2 后，立即提取该信号并将其重新路由到 CLB2，该信号在此处与 CPU HALT 信号合并。产生的信号恰好在 QEP 逻辑的第一级之前向回注入到 QEP 中，就好像它是原始输入一样。从这时起，QEP 逻辑进行接管，处理经修改的输入。

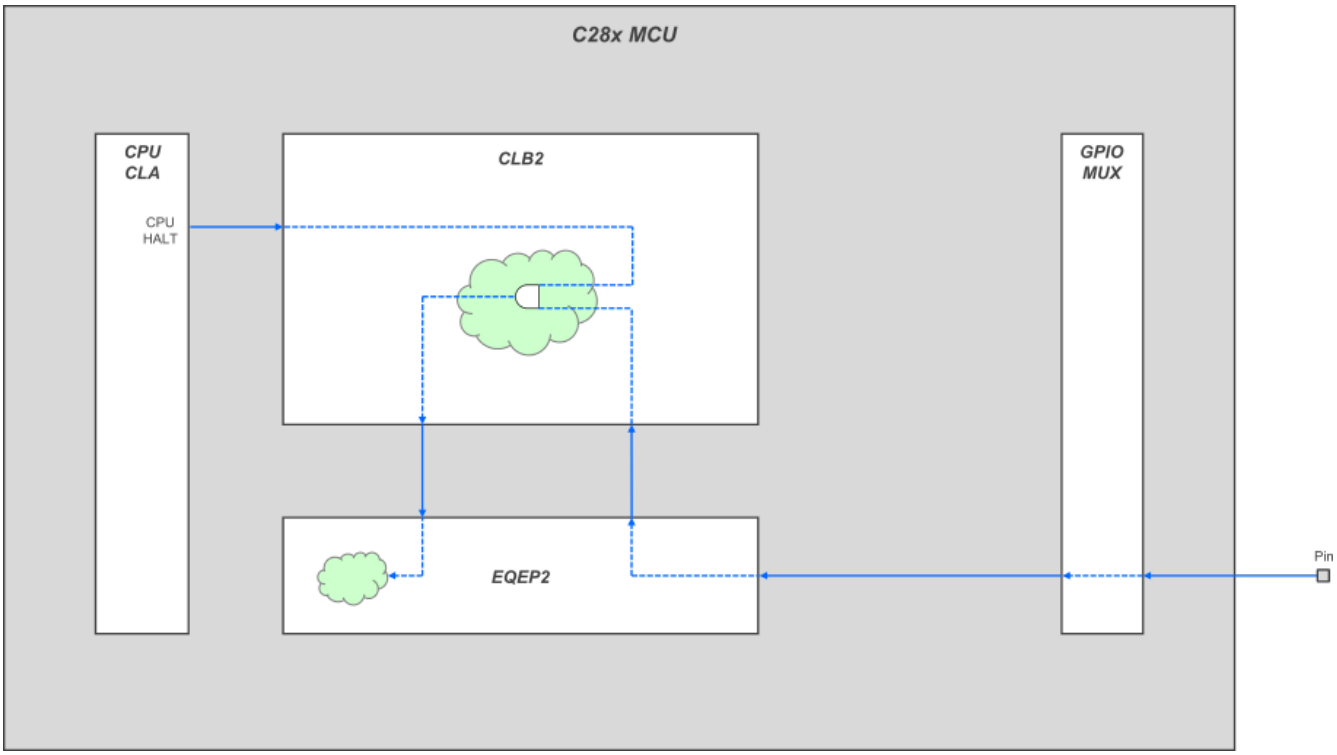

图 3-4. CLB 示例 17 – 使用 CPU 信号修改外设输入信号

图 3-5 在器件级视图中显示了相同的示例，以在 I/O 部分的暗灰色背景下突出显示活动块和产生的数据传输。此处同样显示，GPIO 输入在通过 GPIO 多路复用器之后到达 QEP2。它从此处放置在本地 2 信号总线上并转发至 CLB2。CPU HALT 信号也通过同一本地 2 总线进入 CLB2。在这两个信号进入 CLB2 内部之后，经本地 2 总线选择并传递到 CLB 逻辑块 2。它们在此处进行逻辑组合（根据逻辑块 2 配置寄存器）并放置在 CLB2 输出总线上，它们通过该总线反馈到 QEP2 外设。同时，外设信号多路复用器 2 块提供一个多路复用控制信号，该信号指示 QEP2 使用新计算的输入来代替原始输入。然后，以与原始信号相同的方式在 QEP2 内部处理该替换输入。

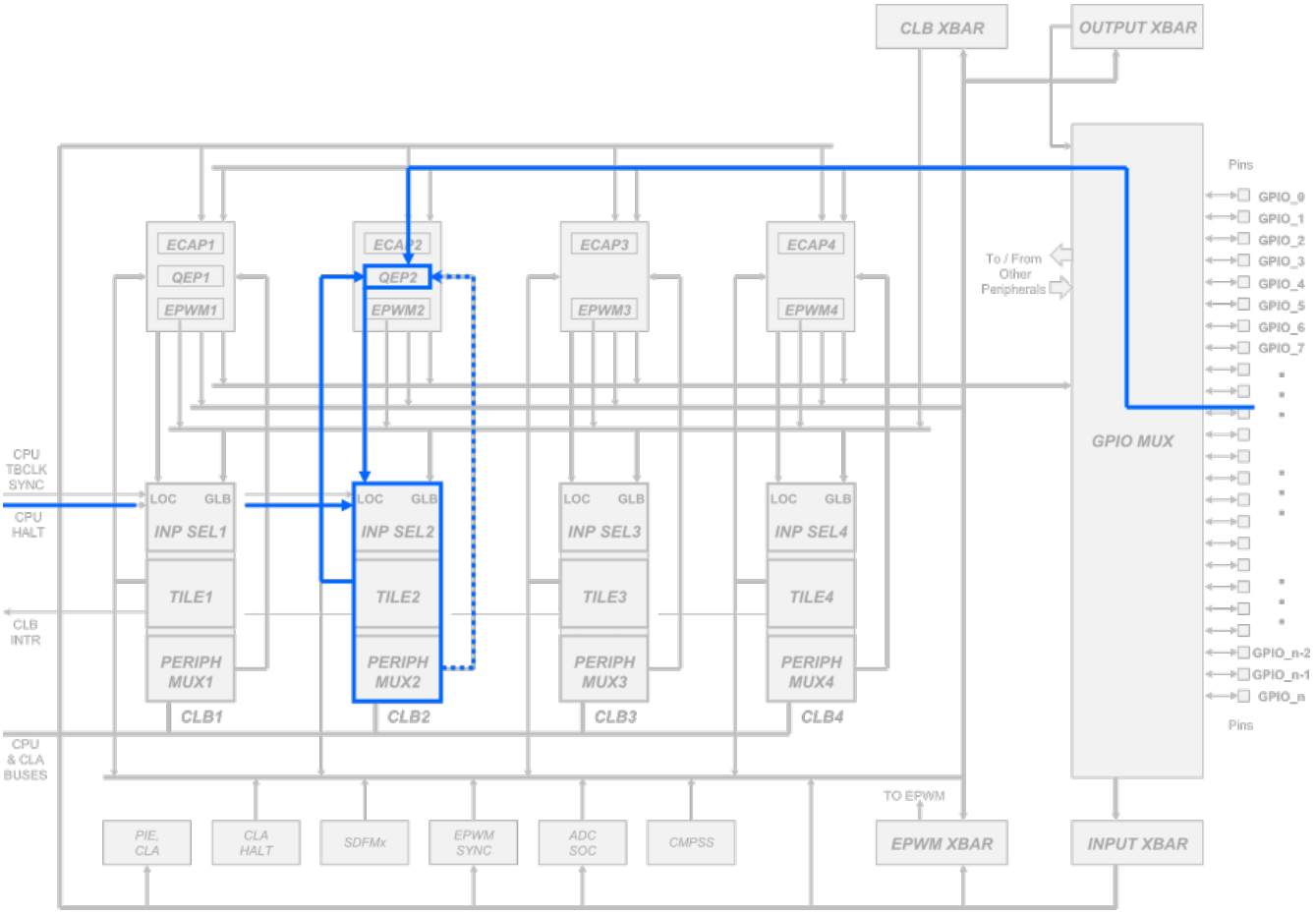

图 3-5. 示例 17 中的信号流 – 器件视图

图 3-6 在外设级别提供该示例的信号流详细信息，并提供了一定的 CLB2 和 QEP2 外设内部可见性。再次跟踪信号，从进入 QEP2 的原始输入开始立即提取出来，将其放置在本地 2 总线上，该总线还具有 CPU HALT 信号。CLB2 输入多路复用器从此处将这两个信号发送至逻辑块 2，在此处这两个信号组合在一起，并将结果放置在 CLB2 输出总线上，经修改的信号通过该总线代替原始输入信号恰好在第一级之前反馈到 QEP2 中。从此处开始，经修改的输入由 QEP2 进行处理，就好像它是原始输入一样。

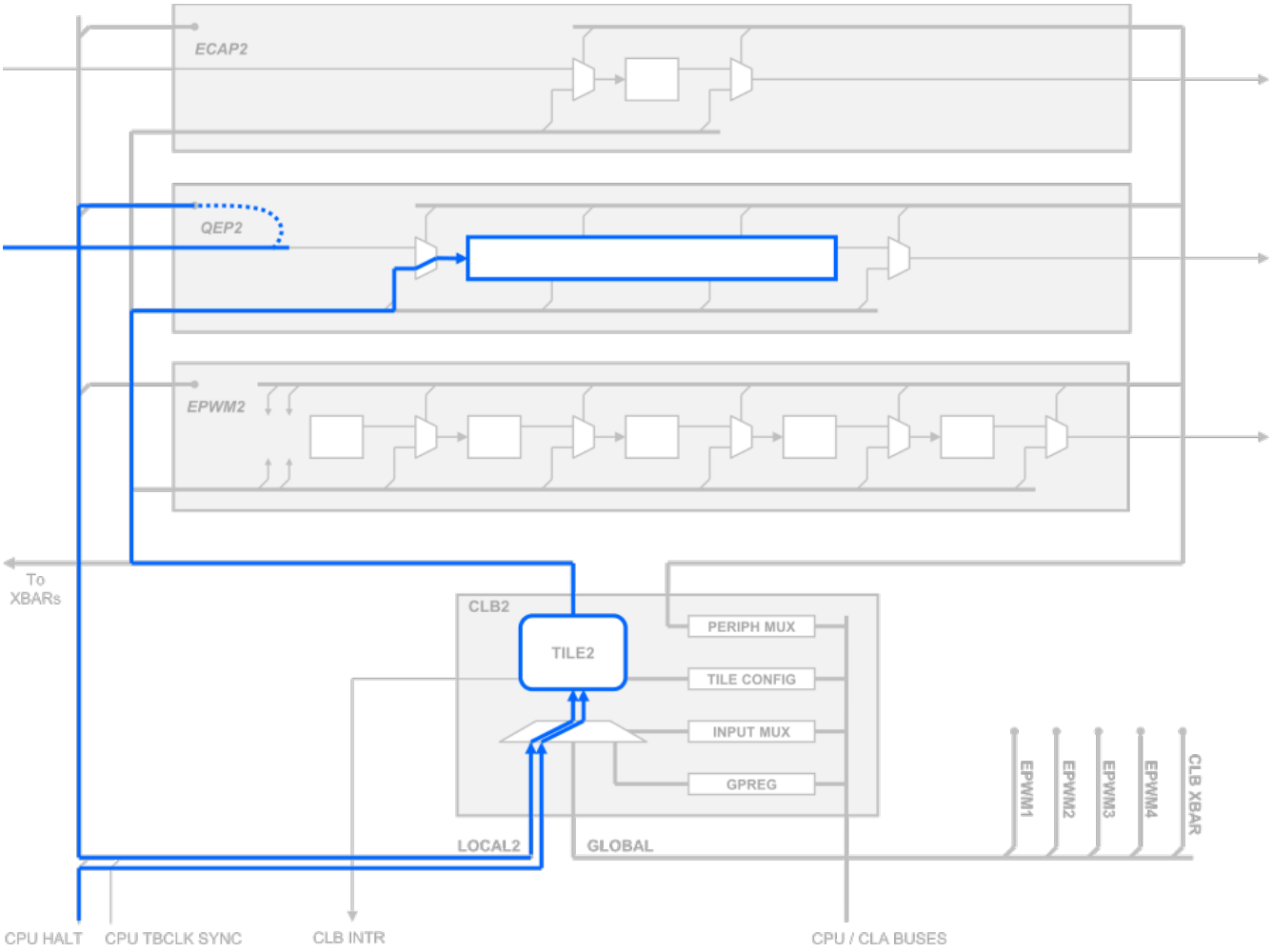

图 3-6. 示例 16 中的信号流 – CLB2 连接

3.3 CLB 示例 18 – 创建您自己的外设来替代 ECAP3

在该示例中，ECAP3 外设内部的所有逻辑都由已在 CLB3 内部编程的自定义逻辑替换。查看信号，可以看到输入由 GPIO 引脚提供，然后输入从该引脚传递到 ECAP3。输入信号进入 ECAP3 后，立即提取该信号并将其重新路由到 CLB3，在此处应用自定义逻辑以将其与 CMPSS 外设的输出（该输出必须首先通过 CLB XBar，然后再进入 CLB3）进行组合。产生的输出恰好在 ECAP 逻辑的最后一级之后向回注入到 ECAP3 中，从而有效替代原始 ECAP3 逻辑。该信号从此处开始直接传输到 GPIO 多路复用器，随后在该多路复用器中分配至输出引脚（请参阅图 3-7）。

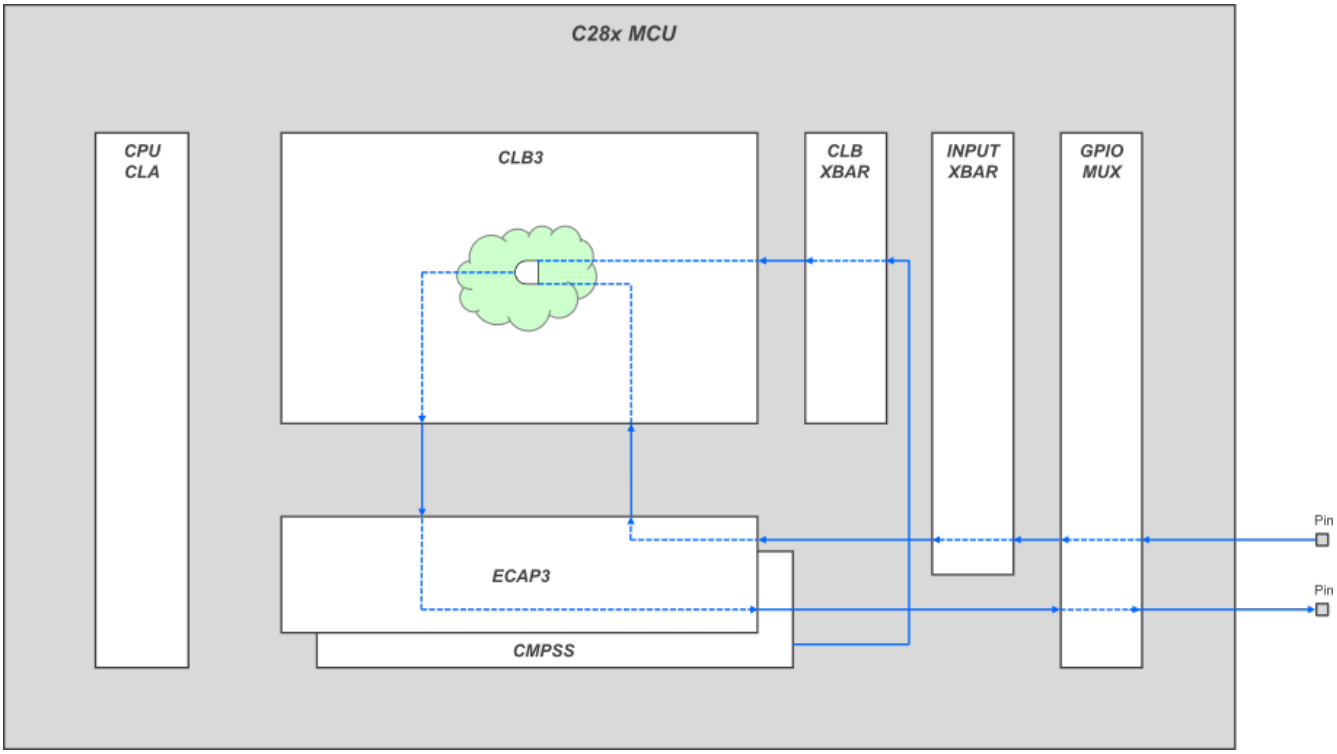

图 3-7. CLB 示例 18 – 创建您自己的外设来替代 ECAP3

图 3-8 在器件级视图中显示了相同的示例，以在 I/O 部分的暗灰色背景下突出显示活动块和产生的数据传输。此处同样显示，GPIO 输入在通过输入 XBar 之后到达 ECAP3。它从此处放置在本地 3 信号总线上并转发至 CLB3。同时，CMPSS 外设的输出在首先通过 CLB XBar 之后也通过全局信号总线到达 CLB3。在 CLB3 内部，从本地和全局总线中选择这两个信号，并根据先前定义的 CLB3 配置寄存器在逻辑块 3 内部对其进行逻辑组合。该操作的输出放置在 CLB3 输出总线上并发送回到 ECAP3。同时，外设信号多路复用器 3 块提供一个多路复用控制信号，指示 ECAP3 使用刚刚计算的信号来代替原始 ECAP3 输出。然后，以与原始 ECAP3 输出通常采取的方式相同的方式路由该替换输出 – 首先进入输出 XBar，然后进入 GPIO 多路复用器，在此处将其放置在通常分配给 ECAP3 的输出引脚上。

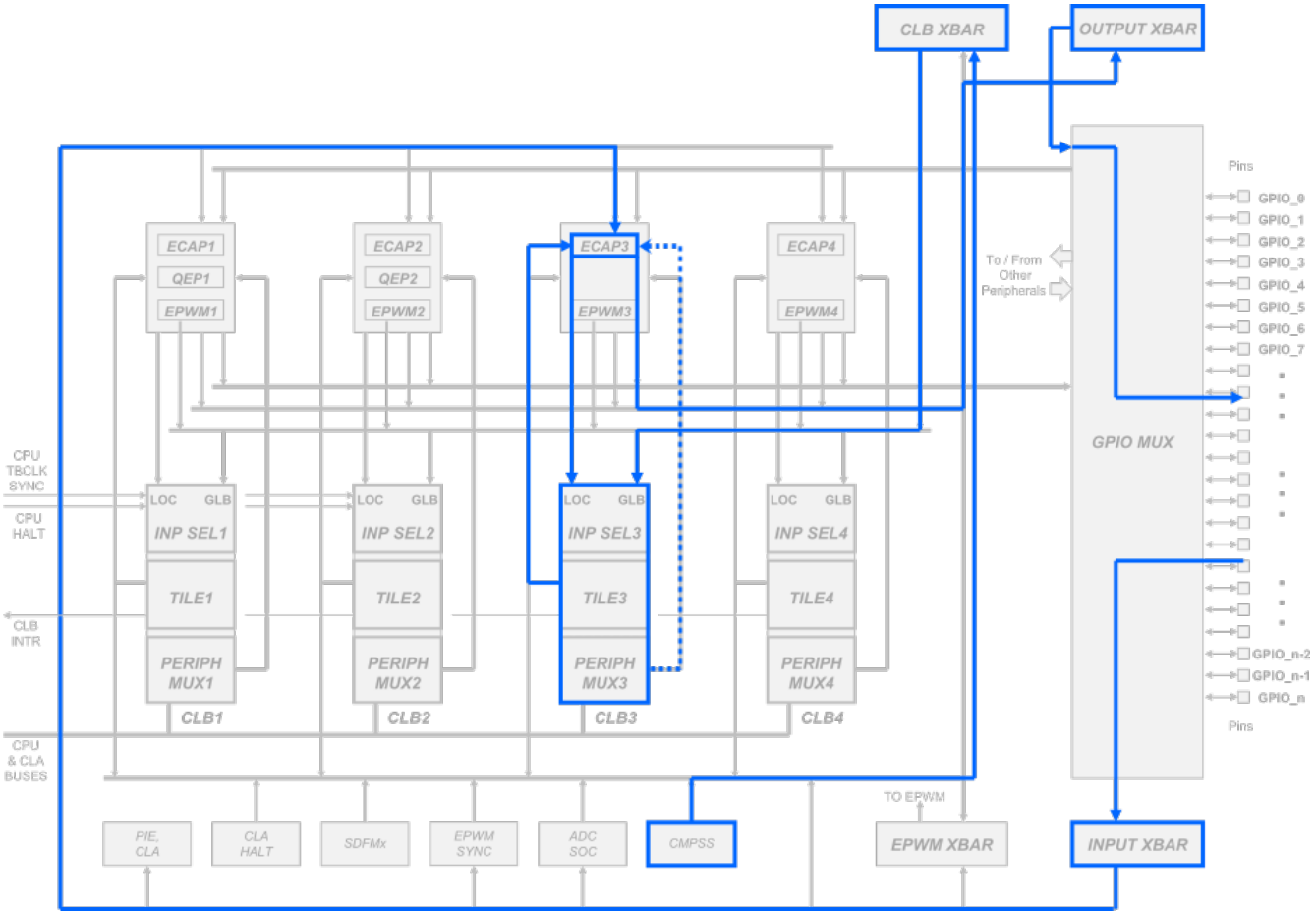

图 3-8. 示例 18 中的信号流 – 器件视图

图 3-9 在外设级别提供该示例的信号流详细信息，并提供了一定的 CLB3 和 ECAP3 外设内部可见性。该图的要点应该是 ECAP3 外设的内部内容已被 CLB3 内部的自定义逻辑完全替代。通过以下方式来实现：在原始输入进入 ECAP3 之后立即对其进行提取，并恰好在其退出 ECAP3 之前注入来自 CLB3 的替代输出，以代替原始输出。在 CLB3 内部，提取的信号通过本地 3 总线进入逻辑块 3，继而在此处与从 CLB XBar 通过全局信号总线到达逻辑块 3 的 CMPSS 输出合并。请注意，进出 ECAP3 的输入和输出没有改变 - 唯一改变的是 ECAP3 外设的内容已由逻辑块 3 内部的自定义逻辑替代。

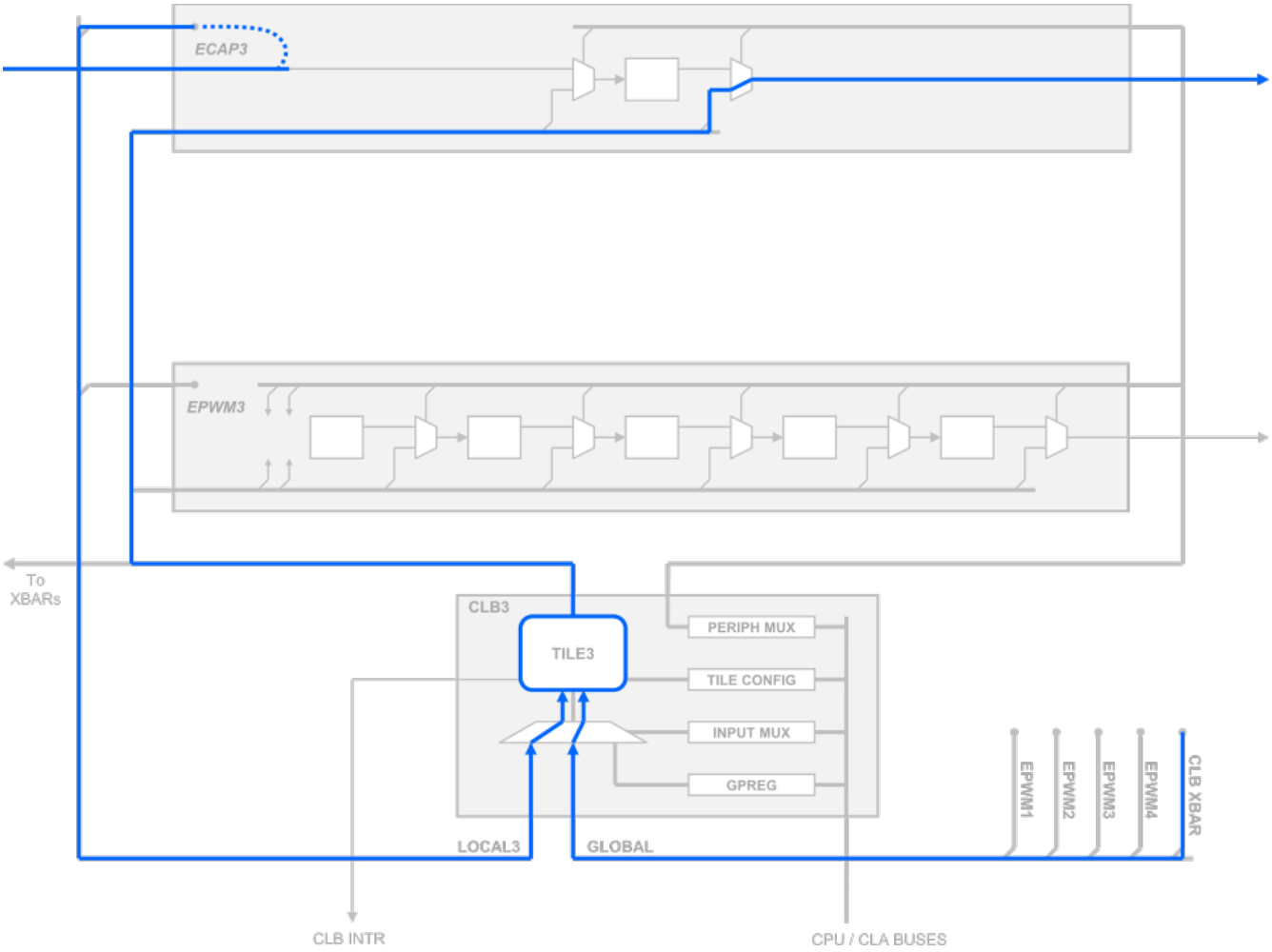

图 3-9. 示例 18 中的信号流 – CLB3 连接

3.4 CLB 示例 19 – 仅使用外部信号来创建您自己的外设

该示例与先前的三个示例的不同之处在于，此处未使用 C2000 外设。相反，CLB4 仅对外部 GPIO 输入进行操作以生成 GPIO 输出。图 3-10 显示有三个外部输入进入 GPIO 多路复用器，并在到达 CLB4 的过程中经过输入 XBar 和 CLB XBar。这三个信号进入 CLB4 之后，会根据 CLB4 配置寄存器进行逻辑组合，然后输出通过输出多路复用器返回到 GPIO 多路复用器，输出在此处放置在分配的输出引脚上。尽管修改内部外设的功能非常强大，但该示例显示，为了使用 CLB，您无需修改或放弃任何外设。可以将自定义逻辑简单地添加到所有其他 C2000 功能之上，以吸收（例如）来自外部器件（如 CPLD 或 FPGA）的胶合逻辑。

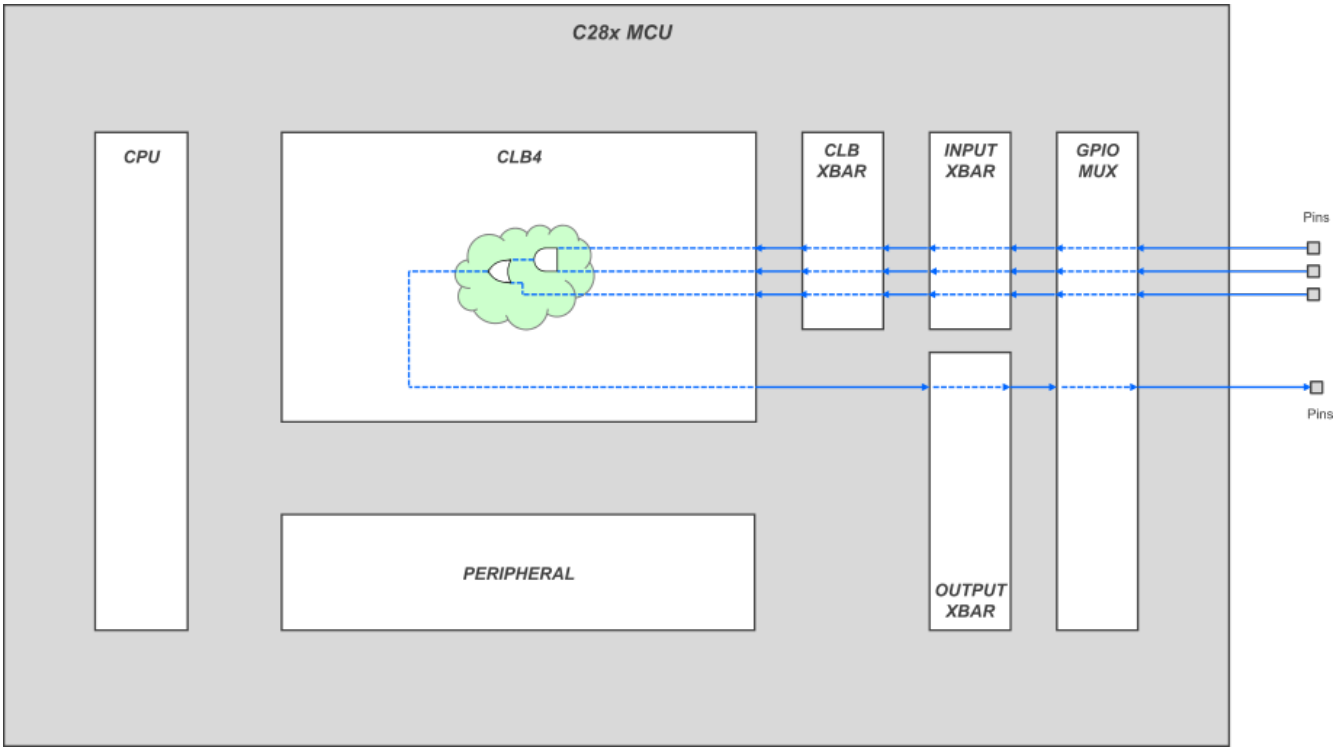

图 3-10. CLB 示例 19 – 仅使用外部信号来创建您自己的外设

图 3-11 在器件级视图中显示了相同的示例，以在整个 I/O 部分的暗灰色背景下更好地显示活动块和产生的数据传输。在此处，它显示了三个 GPIO 输入，它们通过 CLB4 的输入选择部分中的输入 XBar 和 CLB XBar 到达全局信号总线，它们在此处转发至逻辑块 4 以应用自定义逻辑。产生的输出通过输出 XBar 路由到 GPIO 多路复用器，以放置在分配的输出引脚上。在这种情况下不使用 CLB4 的外设信号多路复用器块，并可以将其忽略，因为没有要更改的外设（因此没有要替换的外设信号）。

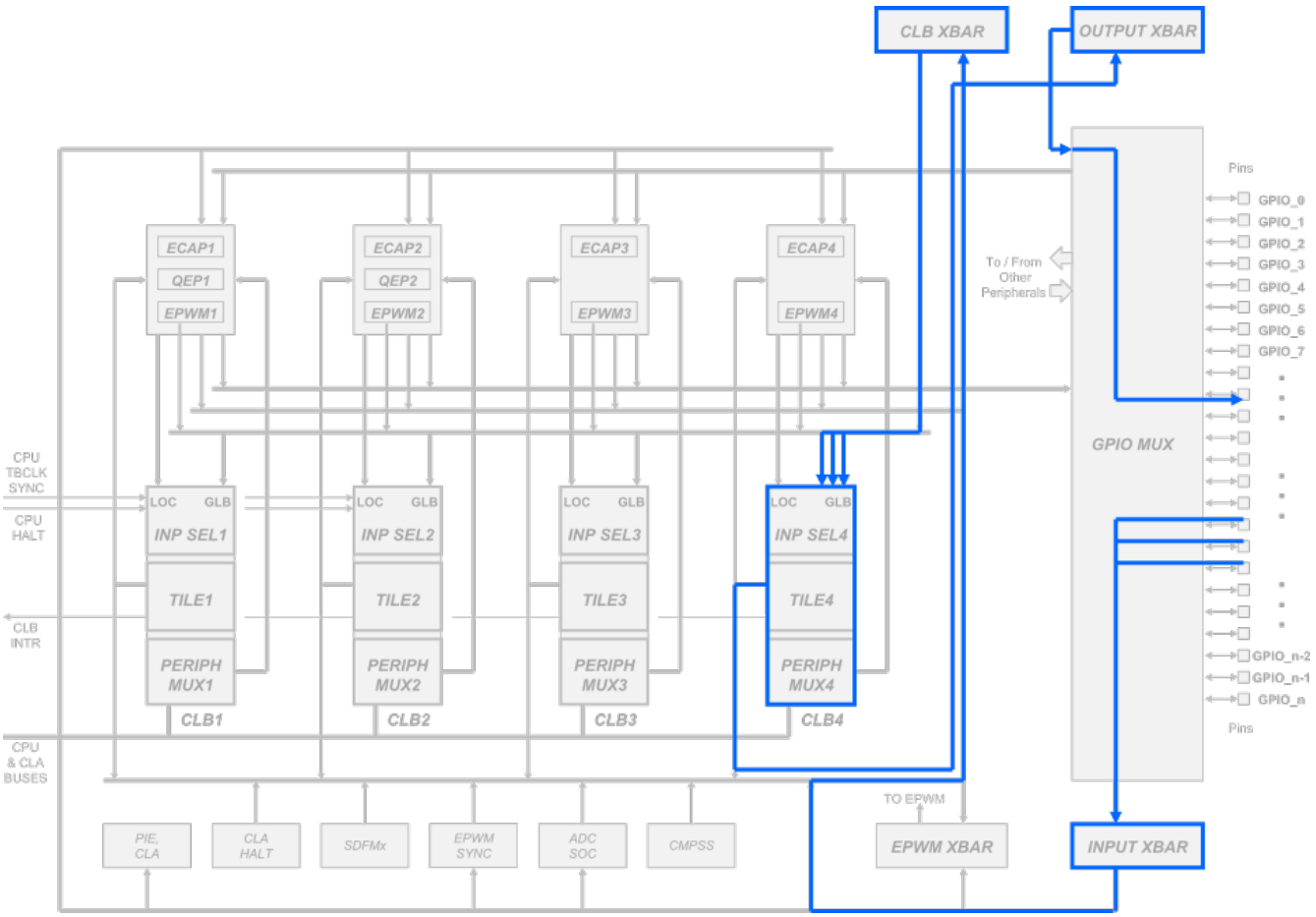

图 3-11. 示例 19 中的信号流 – 器件视图

图 3-12 提供了 CLB4 内部该示例的信号流详细信息。来自 CLB XBar（以及之前的输入 XBar）的三个输入信号使用全局信号总线到达 CLB4 的输入多路复用器部分。配置寄存器中的内容指示输入多路复用器将三个输入转发到逻辑块 4，在逻辑块 4 中应用自定义逻辑以产生输出信号。该输出通过 CLB4 输出信号总线离开逻辑块 4，向输出 XBar 和 GPIO 多路复用器传输。同样，由于在这种情况下不涉及控制外设，不需要使用外设信号多路复用器来替换任何内部外设信号，因此可以保持不变。外设信号多路复用器的默认状态是保留所有外设信号（不使用 CLB 信号替代任何外设信号）。

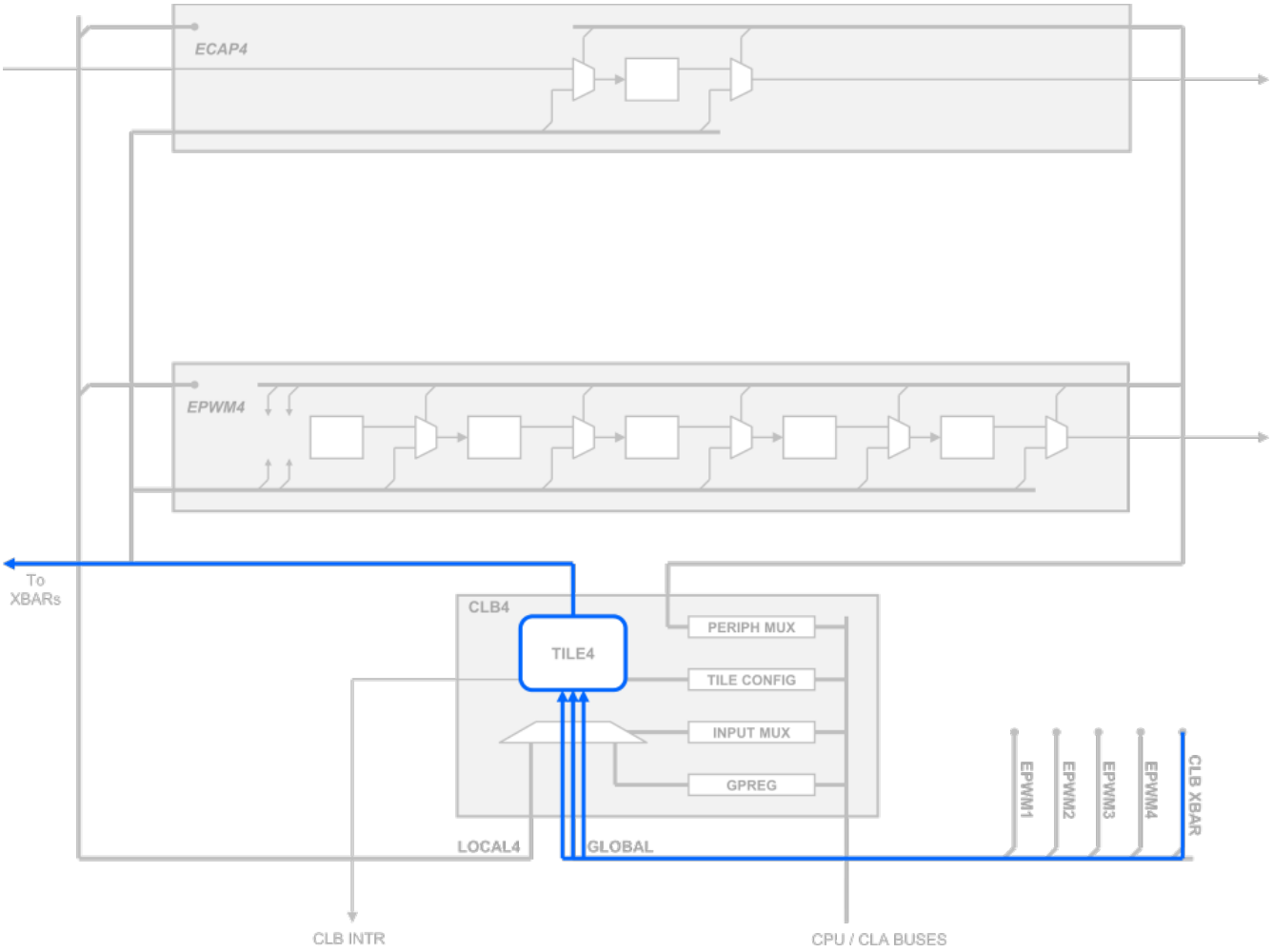

图 3-12. 示例 19 中的信号流 – CLB4 连接

4 FPGA 至 CLB 逻辑转换示例 16

该示例对节 3.1 中的示例 16 进行了扩展，以演示如何仅使用四个可用的 CLB 逻辑块之一将外部自定义逻辑整合到 C2000 微控制器中。图 4-1 显示了基于 FPGA 的印刷电路板。已使用 VHDL 对 FPGA 进行编程，包括两个 PWM 发生器模块和一个胶合逻辑块，该块将两个 PWM 波形与一个内部信号进行组合以驱动单个输出引脚。仔细观察内部 FPGA 信号，两个 PWM 信号 INPUT1 和 INPUT3 进入胶合逻辑块，在此处它们与静态 INPUT2 信号（逻辑 0）进行组合。在胶合逻辑块内编程的逻辑功能传递 7 个 INPUT1 脉冲，然后输出 INPUT2 信号并维持 3 个 INPUT1 脉冲的时间，并重复该模式，直到 INPUT3 变为逻辑 1，此时输出变为高电平（请参阅图 4-1 的波形）。

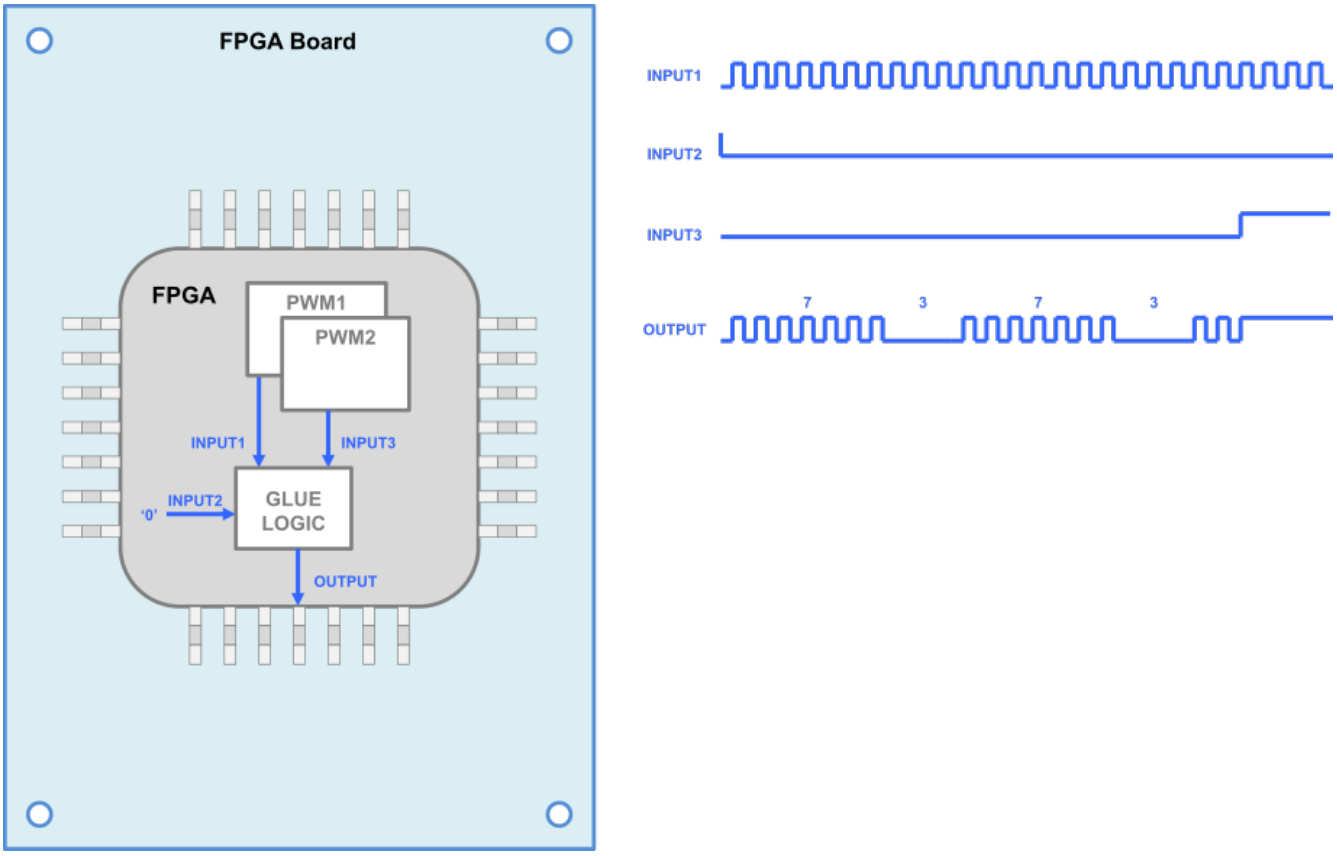

图 4-1. FPGA 内部具有 PWM 发生器和胶合逻辑的系统板

图 4-2 形象化地展示了将 FPGA 逻辑吸收到 C2000 器件中的转换过程，其中 EPWM1 和 EPWM2 控制外设提供 PWM 信号，CLB1 提供胶合逻辑。使用 VHDL 对 FPGA 内的胶合逻辑进行编程，而使用功能调用和基于 GUI 的 SysConfig 工具对 CLB1 内的胶合逻辑进行编程（无需了解 VHDL 或 Verilog）。

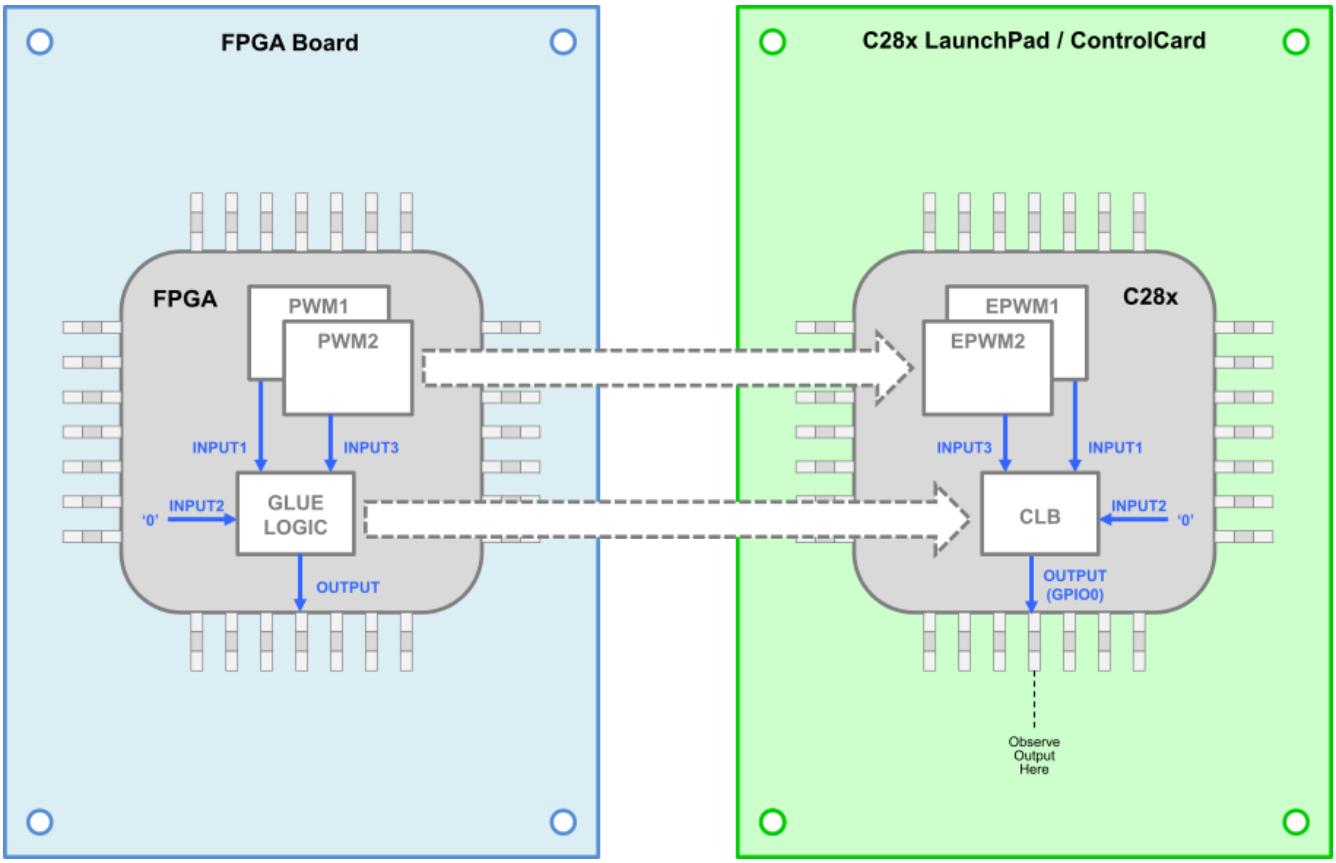

图 4-2. 将 FPGA 中的 PWM 发生器和胶合逻辑映射到 C2000

图 4-3 基于 C2000 板子产生的波形。比较这些波形，图 4-3 显示它们与 FPGA 板的波形是相同的。本节的剩余部分介绍该示例基于 FPGA 的实现和基于 C2000 的实现的详细信息。这些详细信息包括胶合逻辑的 FPGA 和 C2000 版本的原理图和逻辑波形。此外，还包括 FPGA VHDL 源代码和 C2000 项目文件。

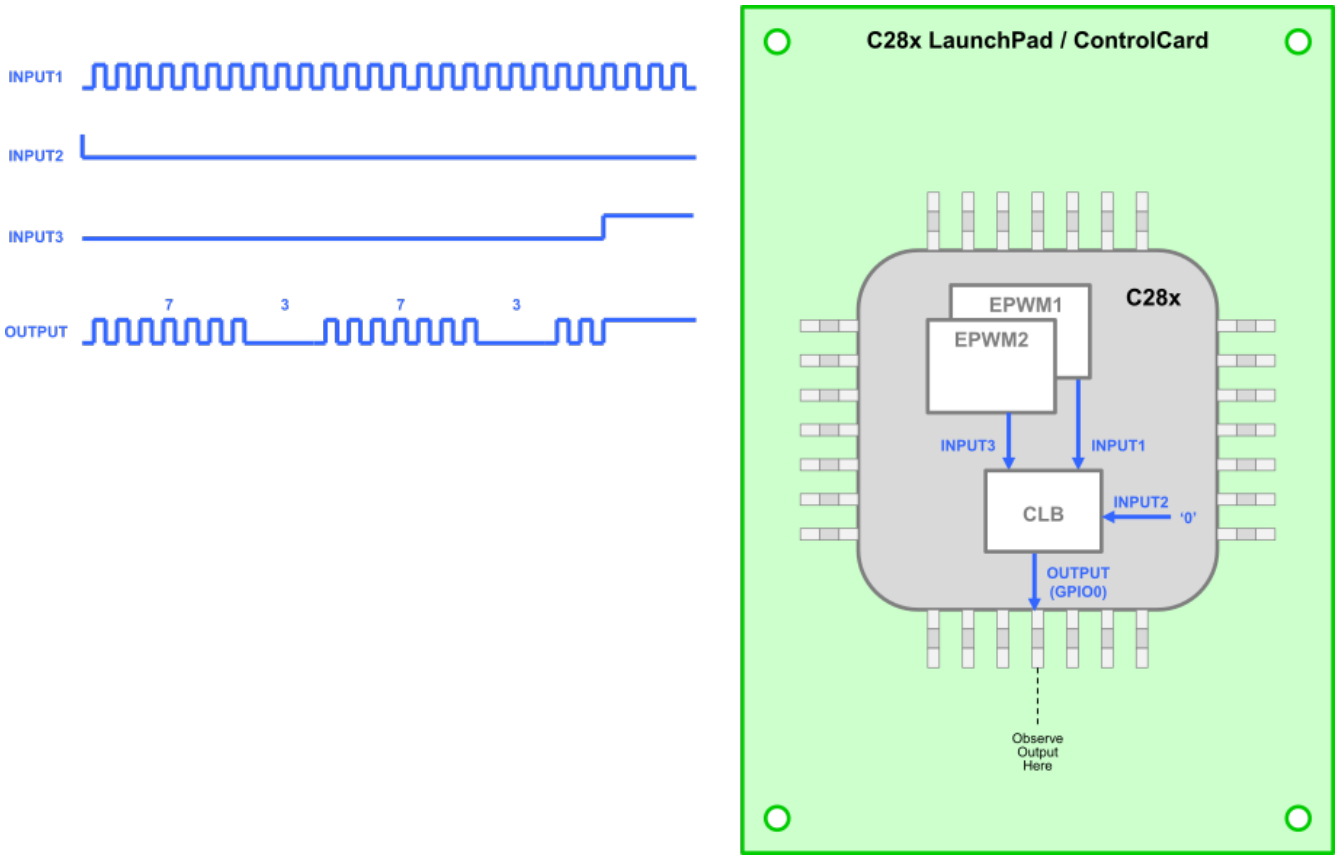

图 4-3. 使用 CLB 和两个 EPWM 外设获得的相同结果

4.1 原始 FPGA 设计

如上所述，使用两个 PWM 发生器和一个胶合逻辑块对 FPGA 进行编程，该胶合逻辑块将特定的逻辑应用于 PWM 输出和第三个信号（逻辑 0）。首先以原理图形式显示胶合逻辑块内部的逻辑，以快速形象化地展示其功能，然后提供对原理图进行镜像的 VHDL 源代码。此外，还显示了提供仿真输入的 VHDL 测试台源代码。

4.1.1 FPGA 胶合逻辑的原理图

胶合逻辑块有三个输入信号和一个输出。输入 1 是连续方波，输入 2 是逻辑 0 常数，输入 3 是另一个频率低于输入 1 的方波。OUT0 信号 反映了根据图 4-4 中所列胶合逻辑功能进行的三个输入逻辑组合。

首先，简单状态机的初始 SELECT2 输出为 0，因此输出 0 信号的值直接映射了输入 1 的值。这导致多路复用器电路选择 IN1 代替 IN2 来驱动内部信号 IN4。接下来，输入 4 与输入 3 进行逻辑或运算，结果通过带寄存器输出触发器传递至驱动输出 0。初始 IN3 为逻辑 0，因此 IN0 在延迟一个 CLB 时钟周期以后，开始输出 IN1 的值。

同时，下降沿检测电路会为输入 1 的每次从低到高转换创建一个单时钟脉冲 ENABLE。对于每个 ENABLE 脉冲，计数器将计数值递增 1。随着计数器的递增，它会根据两个匹配值（MATCH1 和 MATCH2）来检查当前计数值。达到任一匹配值后，组合逻辑就会立即向计数器和简单状态机发出单个 RESET 脉冲。这在计数值变为 7 时首次发生。产生的 RESET 脉冲将计数器重置回 0，并将简单状态机的状态（SELECT2 信号）从 0 翻转为 1。

该新状态导致多路复用器选择输入 2 而不是输入 1，因此在一个时钟周期延迟后输出变为 0。在该新状态下，达到 MATCH2 后，计数器将仅在输入 1 的三个脉冲之后发出另一个 RESET 脉冲，此后重复该模式，计数器再次开始计数到 7。这一直持续到 IN3 变为逻辑 1 为止，此时无论 IN1 或 IN2 的逻辑电平如何，OUT0 都被强制变为逻辑 1。

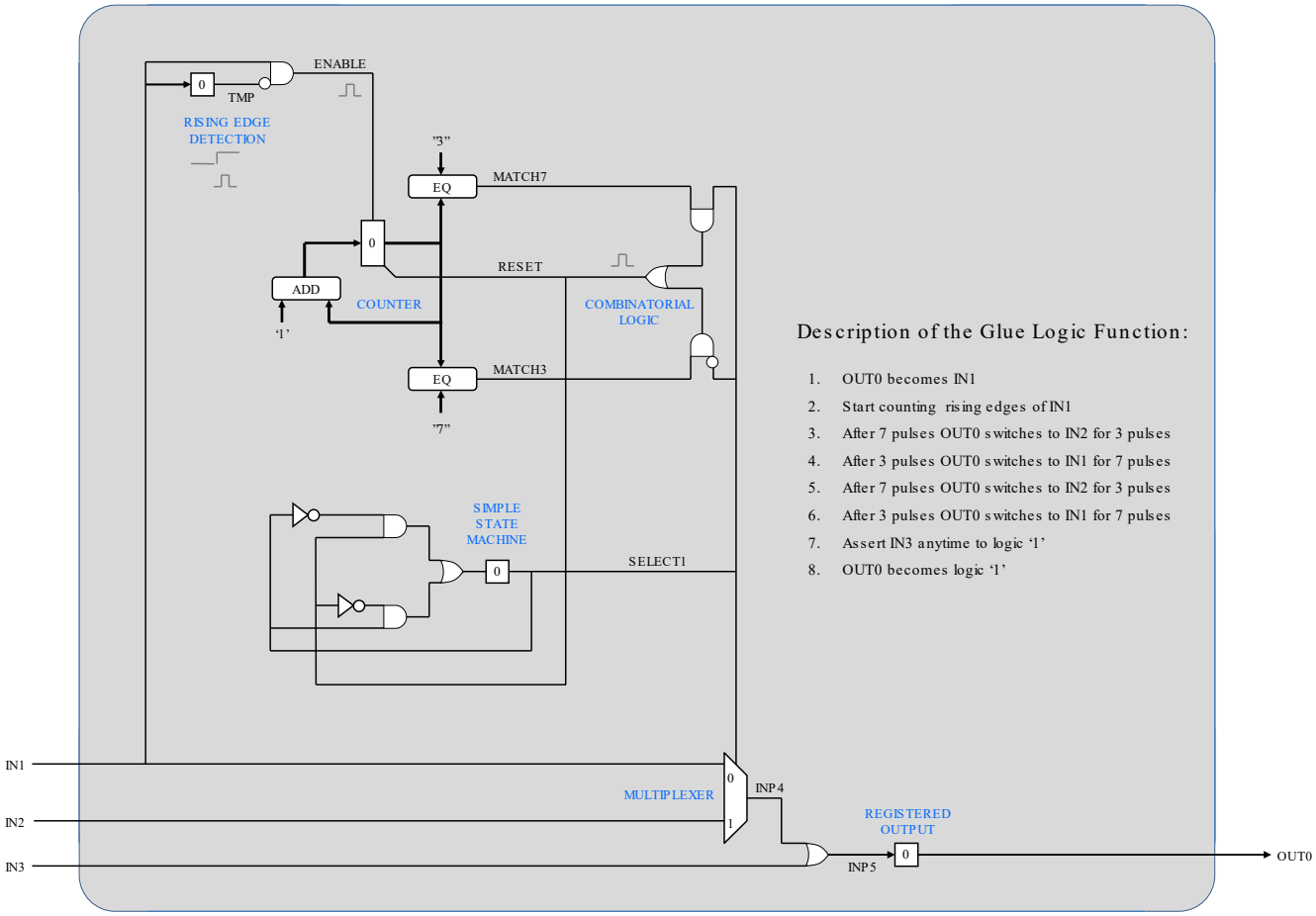

图 4-4. CLB 示例 16 – FPGA 内部的胶合逻辑实现

4.1.2 胶合逻辑的 VHDL 代码

图 4-5 显示了实现图 4-4 的原理图功能的 VHDL 源代码。首先，entity 语句标识三个输入和一个输出以及时钟源（CLB 时钟）。接下来，signal 语句声明内部信号。在信号声明之后，有 5 个信号组合分配语句（瞬时操作，无时钟延迟）。

然后，两个已注册的处理过程用来输出时钟信号。第一个已注册过程 reg1 在 ENABLE 为高电平时在 CLB 时钟的每个上升沿更新 32 位计数器，但每当 RESET 变为高电平时将其循环回至 0。第二个已注册过程 reg2 在 CLB 时钟的每个上升沿更新其余的三个寄存器。这些寄存器包括在下降沿检测电路中使用的 TEMP 寄存器、在简单状态机中使用的 SELECT2 寄存器以及用于锁存输出的 Output 0 寄存器。

```
------------------------------------------------------------
-- FILE     : clb.vhd                                       --
-- TITLE    : CLB Glue Logic Example16 Top Entity           --
-- FPGA     : xc7a200t-3fbg484                              --
-- PROJECT  : clb16.zip                                     --
-- TOOL     : ISE14.7                                       --
------------------------------------------------------------

library IEEE;
    use IEEE.STD_LOGIC_1164.all;
    use IEEE.STD_LOGIC_UNSIGNED.all;

entity CLB is
     port(
          CLK   :  in    std_logic;
          IN1   :  in    std_logic;
          IN2   :  in    std_logic;
          IN3   :  in    std_logic;
          OUT0  :  out   std_logic := '0'
          );
end CLB;


architecture galicki of CLB is

          signal    temp    :  std_logic := '0';
          signal    enable  :  std_logic;
          signal    count   :  std_logic_vector(3 downto 0
                               := (others => '0');
          signal    match1  :  std_logic;
          signal    match2  :  std_logic;
          signal    reset   :  std_logic;
          signal    select2 :  std_logic := '0';
          signal    in4     :  std_logic;

begin
          enable  <=  (not IN1) and temp;
          match1  <=  '1' when (count = X"7") else '0';
          match2  <=  '1' when (count = X"3") else '0';
          in4     <=  IN2 when (select2 = '1') else IN1;
          reset   <=  (match1 and (not select2))
                        or (match2 and select2);
```

```
   reg1: process (CLK)
                begin
                   if (CLK'event and CLK = '1') then

                      if reset = '1' then
                         count  <=  (others => '0');
                      else
                         if enable = '1' then
                            count  <=  count + 1;
                         end if;
                      end if;

                   end if;
         end process;


   reg2: process (CLK)
                begin
                   if (CLK'event and CLK = '1') then

                      select2  <=  (reset and (not select2))
                                    or ((not reset) and select2);
                      temp     <=  IN1;
                      OUT0     <=  in4 or IN3;

                   end if;
         end process;

end galicki;
```

图 4-5. CLB 示例 16 – FPGA 胶合逻辑的 VHDL 源代码

4.1.3 测试输入的 VHDL 代码

图 4-6 它显示了上述 FPGA 胶合逻辑代码的仿真测试台。该 VHDL 测试台定义了时钟频率和 VHDL 仿真器测试胶合逻辑功能所需的输入状态。查看测试台，可以看到初始语句声明了时钟、三个输入和一个输出。接下来，将其与从测试台实体内部实例化的胶合逻辑实体的相应信号进行匹配。接下来是定义时钟和三个输入的三个过程。时钟频率设置为 100MHz，输入 2 设置为常数逻辑 0。其余两个输入（IN1 和 IN3）模拟 PWM 发生器的输出，其中 IN1 具有高频方波（320ns 周期），IN3 具有低频方波（14.72µs 周期）。仅为输入 3 定义了一个脉冲。

```
--------------------------------------------------------------------
-- FILE    : tb1.vhd                                               --
-- TITLE   : CLB Glue Logic Example16 Simulation Test Bench --
-- FPGA    : xc7a200t-3fbg484                                      --
-- PROJECT : clb16.zip                                             --
-- TOOL    : ISE14.7                                               --
--------------------------------------------------------------------

library IEEE;
    use IEEE.STD_LOGIC_1164.all;
    use IEEE.NUMERIC_STD.all;

entity TB1 is
end TB1;

architecture galicki of TB1 is

            signal   clk   :  std_logic;
            signal   in1   :  std_logic;
            signal   in2   :  std_logic;
            signal   in3   :  std_logic;
            signal   out0  :  std_logic;


component   CLB
       port(
            CLK     :   in    std_logic;
            IN1     :   in    std_logic;
            IN2     :   in    std_logic;
            IN3     :   in    std_logic;
            OUT0    :   out   std_logic
            );
end component;
```

```
begin

clb16:                              CLB
    port map(
            CLK   =>   clk,
            IN1   =>   in1,
            IN2   =>   in2,
            IN3   =>   in3,
            OUT0  =>   out0
            );

clock: process
               begin
                     wait for 0 ns;
                     clk  <=  '0';
                     loop
                        wait for 5 ns; -- 100 MHz
                        clk  <=  not clk;
                     end loop;
            end process;


inp1: process
               begin
                     wait for 0 ns;
                     in1  <=  '0';
                     loop
                        wait for 160 ns;
                        in1  <=  not in1;
                     end loop;
            end process;

                     in2  <=  '0'; -- inp2

inp3:    process
               begin
                     in3  <= '0'; wait for 7360 ns;
                     in3  <= '1'; wait;
            end process;

end;
```

图 4-6. CLB 示例 16 – 胶合逻辑输入的 VHDL 源代码

4.1.4 FPGA 胶合逻辑仿真波形

图 4-7 显示了由仿真测试台执行的胶合逻辑功能的仿真结果。除时钟、输出和三个输入之外，波形还显示了关键内部信号：ENABLE、RESET 和 SELECT 2。查看 OUT0 信号，可以发现，很明显，它反映了在上述 VHDL 源代码中编程的预期功能 – 在七个 IN 1 周期中输出 IN1 信号，然后在三个 IN1 周期中变为输出 IN2 信号（逻辑 0），该模式持续发生，直到 IN1 变为逻辑 1，这时 OUT0 被强制变为逻辑 1。

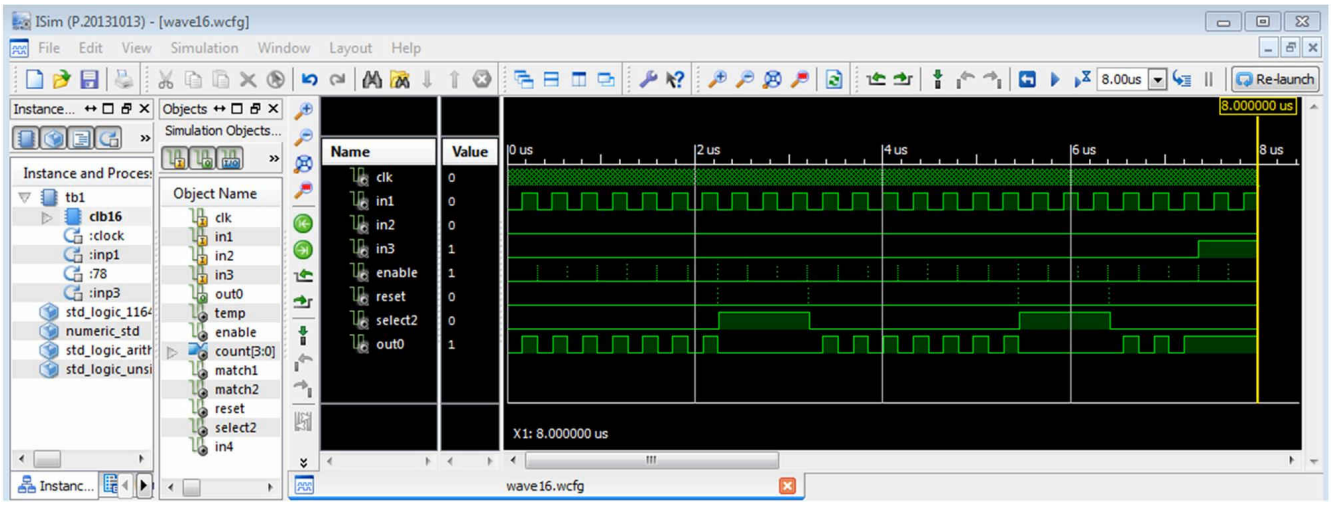

图 4-7. CLB 示例 16 – VHDL 仿真器波形

4.2 FPGA 到 CLB 的转换过程

示例 16 中的自定义逻辑具有两个主要元素：两个 PWM 发生器和一个胶合逻辑块，该块将产生的 PWM 输出与静态（逻辑 0）信号进行组合以产生单个输出。下面描述了将这些元素从 FPGA 平台转移到 C2000 微控制器的过程。

4.2.1 将 PWM 发生器映射到 EPWM 外设

C2000 微控制器具有可编程控制外设，包括增强型 PWM 外设 (EPWM) 的多个实例。选择了 EPWM1 和 EPWM2 来代替该示例的 FPGA 实例中使用的两个 PWM 发生器。有关初始化 EPWM 模块以生成与两个 FPGA PWM 发生器匹配的相应输出方波的代码，请参阅 C2000Ware 内的示例 16 项目文件。

4.2.2 将 VHDL 中的胶合逻辑映射到 CLB

图 2-11 显示形成 CLB 逻辑块的构建块包括三个计数器、三个 LUT4 块（4 输入查找表）、三个 FSM 块（有限状态机）和八个输出 LUT3 块。这些块可通过基于 GUI 的 SysConfig 工具进行编程，该工具根据用户与 GUI 的交互，生成代码来配置相应的逻辑块配置寄存器。将 HDL 代码转换为 CLB 的过程包括四个步骤：

- 使用输入选择功能调用从全局和本地信号总线（或 GPREG 寄存器）中为每个 CLB 逻辑块选择最多八个输入，并使用外设信号多路复用器功能调用指定八个可能的输出中的每一个是否替换所选控制外设内的信号
- 绘制 HDL 代码的功能方框图（如图 4-4 所示）
- 将该方框图细分为最适合映射到特定 CLB 构建块中的更小子部分集合（如图 4-8 所示）
- 使用 SysConfig GUI 分别为每个生成的子部分实现逻辑。

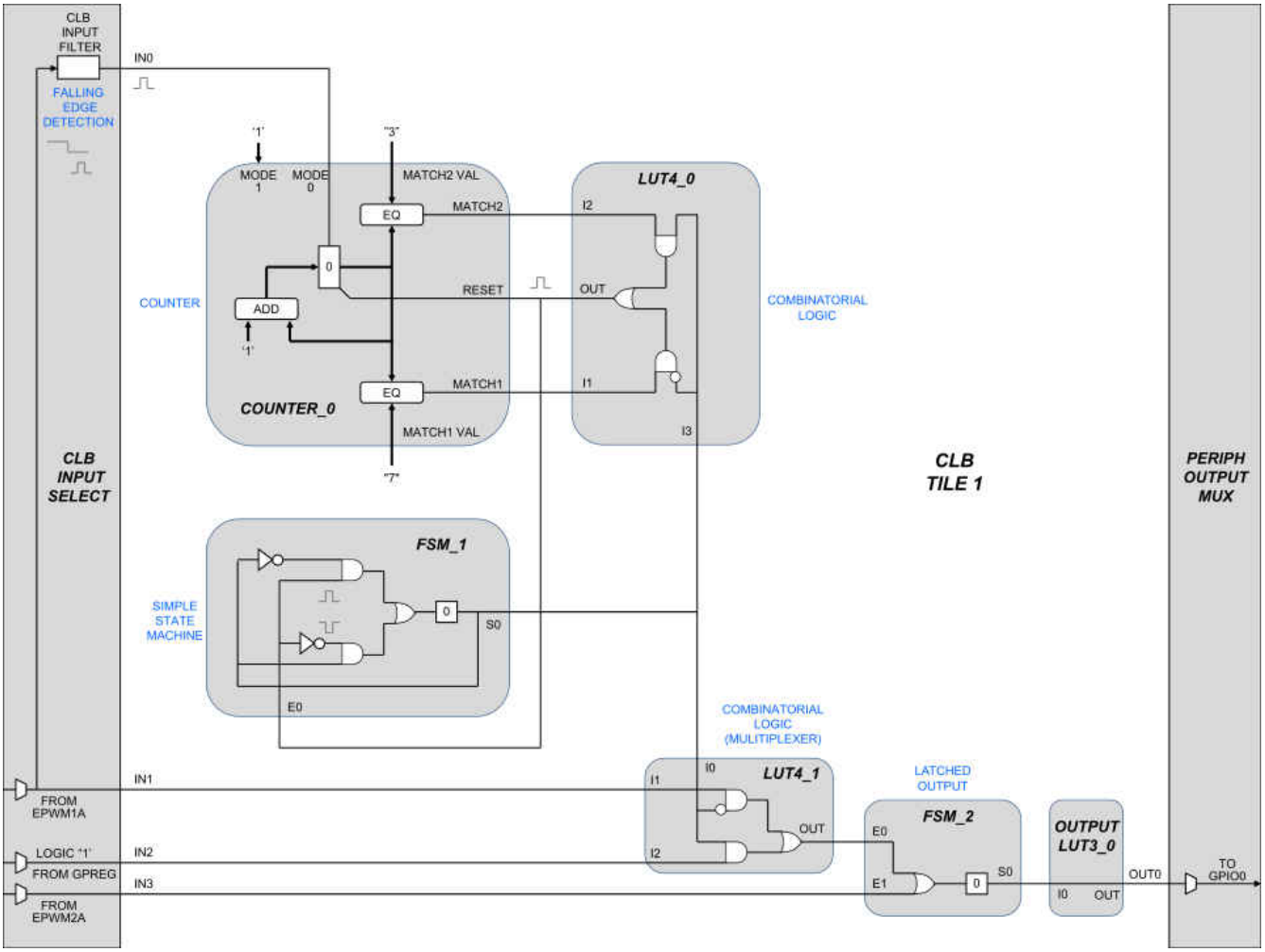

图 4-8. CLB 示例 16 – CLB1 的逻辑分配

4.2.2.1 输入

在此提醒一下，在该示例中 EPWM1 外设有两种用途：向 CLB1 提供输入 1 信号（用于计算 PWM 波形），以及在 EPWM1 的最后一级注入 CLB1 的输出 0，以使用通常为 EPWM1A 输出保留的内部信号将其转发到 GPIO 多路复用器，然后转发到 GPIO0 引脚（请参阅图 3-1）。

图 2-7 显示了输入多路复用器 1 具有嵌入式同步器和滤波器电路，分别将输入与 CLB 时钟进行同步，以及在每个高电平到低电平转换或低电平到高电平转换时将输入信号整形为脉冲。该示例使用滤波器为信号 1 的每个下降沿生成一个单时钟电平脉冲，以使计数器对信号 1 脉冲进行计数（请参阅图 4-4 和图 4-5）。

图 2-7 中多路复用器链的每个多路复用器级需要一个专用的功能调用（即使在该示例中没有位输入使用本地信号，也还是需要本地信号功能调用）。通过全局输入功能调用选择该示例中通过全局信号总线到达逻辑块 1 的两个 EPWM 输入（输入 1 和输入 3）。输入 2 来自 GPREG 寄存器，通过 GPREG 输入功能调用进行选择。

4.2.2.2 逻辑分配

在讨论完输入之后，我们现在重点讨论在为实现胶合逻辑功能而选择的逻辑块 1 内发生的情况。同样，此处的目标是将图 4-4 中的连续方框图构建成功能上等效且最适合映射到各个 CLB 逻辑基元中的更小块集合。图 4-8 显示了生成的方框图。以下是用于将图 4-4 转换为图 4-8 的步骤：

- 首先，如上所述，已在输入选择器内部实现了下降沿检测（因此不必在逻辑块内实现它）。接下来，将计数器功能分离出来，以在 COUNTER_0 块内实现。然后，将剩余的逻辑分为使用寄存器的两部分和不使用寄存器的两部分。

- 使用寄存器的两部分（表示 VHDL 注册的处理过程）已按照功能进行分离，并映射到 FSM_1 和 FSM_2 块中。这两者中只有第一个是有限状态机，其中给定时钟边沿处的输出变为下一个时钟边沿处的输入。第二个只是一个带寄存器的输出，没有任何反馈被映射到 FSM 块中，因为这是具有内置寄存器的最简单 CLB 基元。
- 两个不带寄存器的组合逻辑部分已按照功能进行分离，并映射到 LUT4_0 块和 LUT4_1 块中。LUT4_0 块生成 RESET 脉冲，该脉冲将计数器重置为零，并将 FSM_1 状态从 0 翻转为 1 或从 1 翻转为 0。LUT4_1 块实现了可以在图 4-4 底部看到的多路复用器，以根据当前状态 S0 选择输入 1 或输入 2 信号。

4.2.2.3 输出

最后，OUTPUT LUT3_0 用于将产生的输出发送至嵌入在 EPWM1 最后一级中的多路复用器，以通过 GPIO 多路复用器继续传输，驱动通常分配给 EPWM1A 的 GPIO0 引脚。请注意，每个逻辑块内部有八个 OUTPUT LUT3 基元（请参阅图 2-11）。每个基元都可以将其输出作为替换信号驱动到控制外设的 14 个预选级之一中，以替换该级中通常存在的原始信号（请参阅图 2-9）。如本节开头的步骤 1 中所述，由 PERIPH 信号多路复用器功能调用来决定给定输出信号是否替换原始信号。

4.3 生成的 C2000 设计

示例 16 的 CLB 设计现在已对所有组件进行了配置，在 C2000 微控制器内部去模仿原始 FPGA 设计。这包括两个 EPWM 外设、用于配置输入选择和外设信号多路复用器的 CLB 功能调用以及 SysConfig 生成的 CLB 逻辑块配置代码。至此，设计完成。接下来看看 CLB 逻辑块 1 内部生成的信号连接，并通过运行 CLB 仿真器和外部逻辑分析仪来验证设计输出与 FPGA 设计的输出是否匹配。

4.3.1 信号连接

C2000Ware 随附的工具之一是图形化逻辑块查看器，它可以显示给定 CLB 设计的所有 CLB 组件以及相关的信号互连。该工具可以快速检查所有预期的 CLB 组件是否已启用并正确连接。除了标识活动的组件和连接之外，该查看器还显示 FSM 和 LUT 组件内部的输入信号、输出信号和逻辑方程。请注意，该查看器的当前版本仅限于显示 CLB 逻辑块的内部信息（它当前不支持逻辑块的输入信号选择和逻辑块输出的外设信号多路复用器分配）。图 4-9 显示了该查看器的屏幕截图，该屏幕截图适用于示例 16 的 CLB 逻辑块 1。查看逻辑块 1 的组件，您可以看到预期的两个 FSM 块、两个 LUT 块和一个计数器，其中 FSM 和 LUT 块显示了输入和输出之间的内部逻辑方程。仔细查看这些逻辑方程，您将看到它们与 图 4-8 的原理图相匹配。同样明显的是，有四个输入信号进入逻辑块 1，一个输出信号从逻辑块 1 中退出。

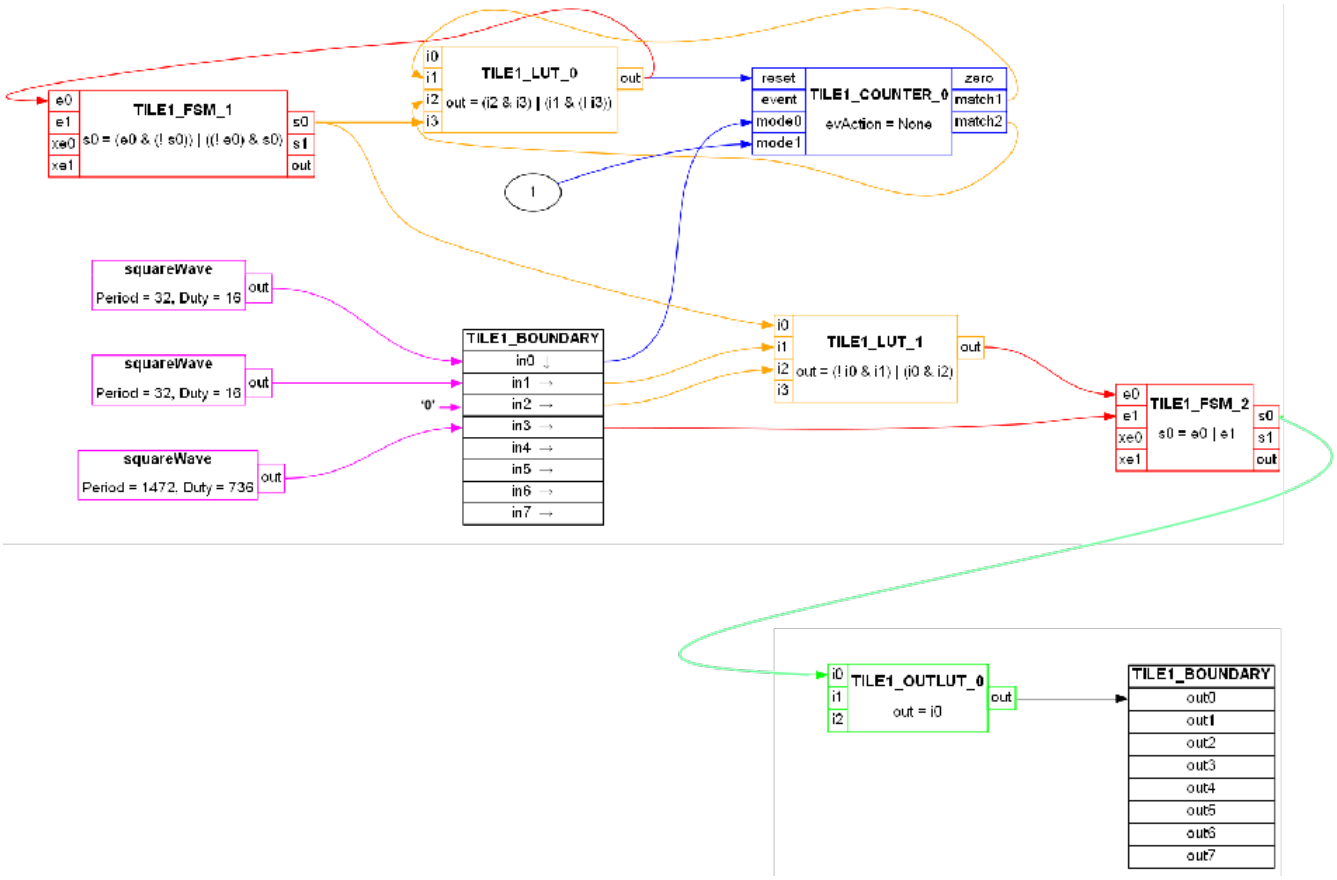

图 4-9. CLB 示例 16 – 以可视化方式显示逻辑块 1 内部的信号连接

4.3.2 仿真波形

随 C2000Ware 提供的 CLB 工具套件还包括 CLB 仿真器。与逻辑块查看器一样，该工具的当前版本模拟逻辑块，但不模拟输入选择或输出的外设信号多路复用。为了弥补这一点，模拟器可以提供常用输入，如常数、波形和脉冲。这些是使用 SysConfig 工具的 GUI 通过下拉菜单选择（用于配置逻辑块逻辑基元的相同方法）定义的。图 4-10 显示了来自 CLB 仿真器的 CLB 示例 16 的屏幕截图。检查波形，您可以看到四个输入和输出以及一些关键的内部信号。将这些波形与 FPGA 仿真器的波形（图 4-7）进行比较，您可以看到完全匹配，这表明示例 16 的 C2000 版本确实与原始 FPGA 设计相匹配。有关上面使用的功能调用和 SysConfig 代码，请参阅 C2000Ware 示例 16 项目文件。

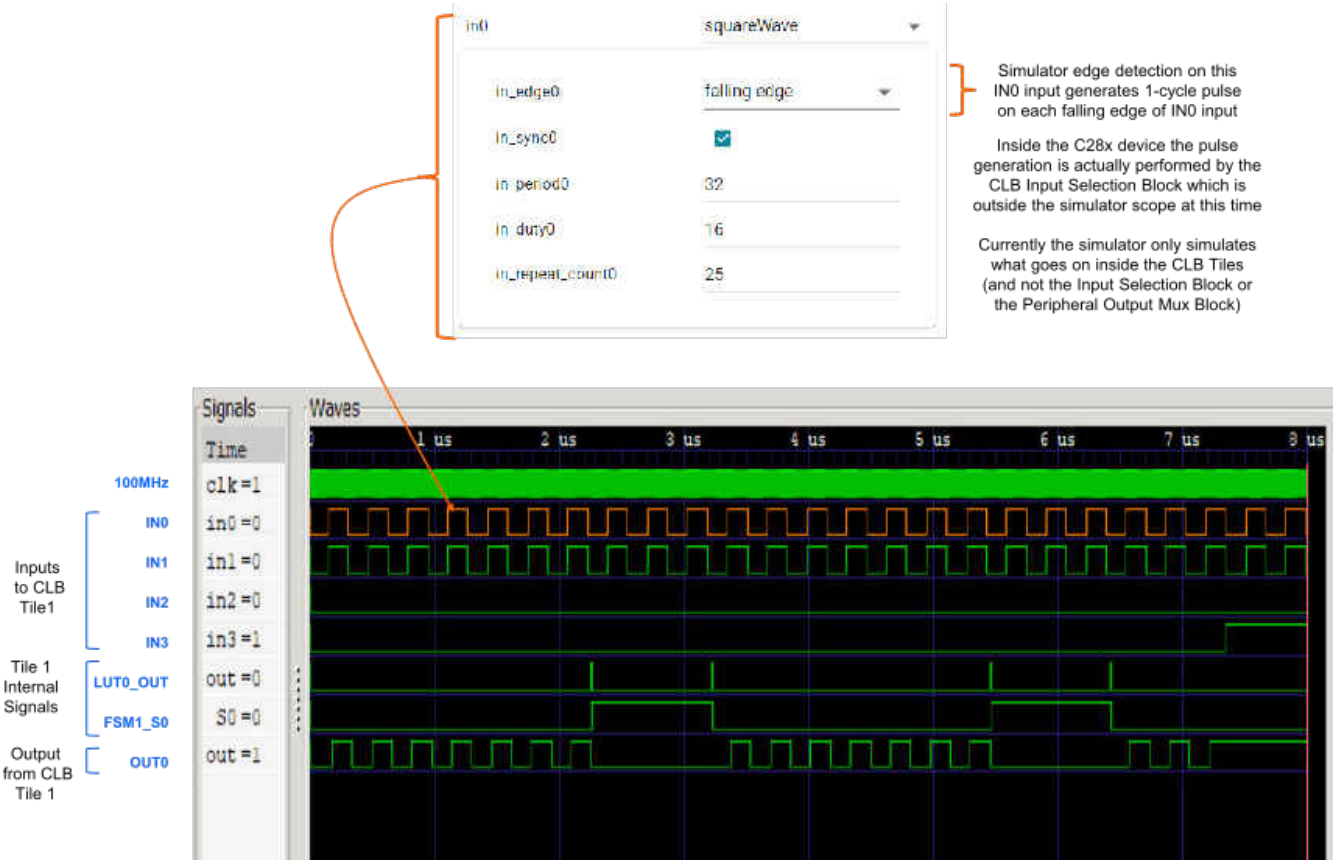

图 4-10. CLB 示例 16 – CLB 仿真器波形

4.3.3 ControlCard、LaunchPad 波形

仿真确认在 C2000 CLB 内部实现的自定义逻辑与原始 FPGA 实现的逻辑相匹配之后，就该使用实际的 C2000 硬件来验证结果了。图 4-11 包含连接到 F28004x ControlCard 的 GPIO_0、GPIO_1 和 GPIO_2 的逻辑分析仪的屏幕截图（还使用 F28004x LaunchPad 捕获了相同的波形）。GPIO_0 表示 CLB 输出，该输出已按照配置逻辑块 1 的外设信号多路复用器的功能调用的指示替换了原始 PWM1A 输出。虽然原始的 PWM1A 信号已被 CLB1 输出替换，但相同的功能调用指定 CLB1 的其他 7 个输出不替换其对应的内部控制外设信号，因此可以在 GPIO_1 引脚上看到原始 PWM1B 输出，可以在 GPIO_2 上看到原始 PWM2A 输出。

这样就完成了示例 16 中所述的自定义逻辑从外部 FPGA 到 C2000 的转移。结果证实，可以将复杂的功能（如 PWM 发生器和自定义胶合逻辑）从外部可编程逻辑器件转移到 C2000 微控制器中，并且结果完全相同。该功能通过实现自定义逻辑以外部可编程逻辑器件无法实现的方式来访问 C2000 CPU 和外设的内部信号，从而降低了总系统成本，同时增强了功能。

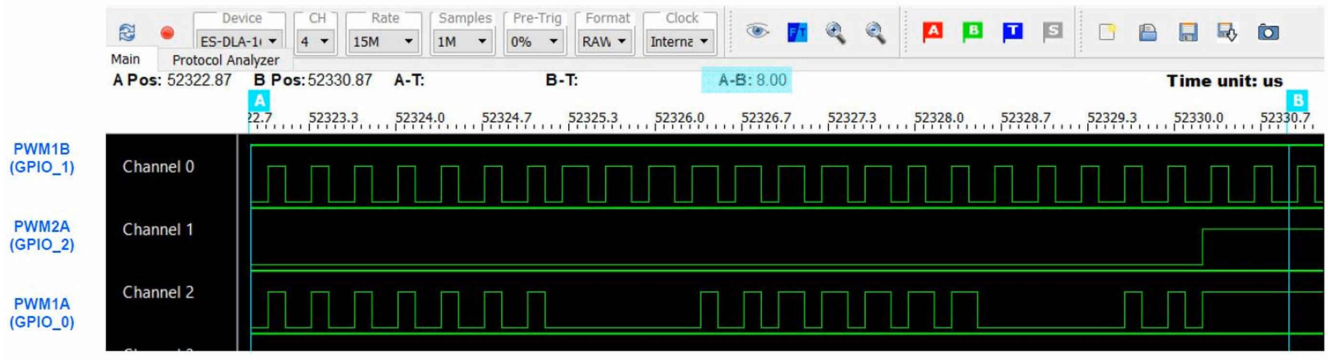

图 4-11. CLB 示例 16 – C2000 LaunchPad/ControlCard 波形

5 参考文献

有关特定 C2000 器件内的 CLB 模块的更多信息，请参阅特定于器件的 C2000 数据表和技术参考手册 (TRM)。

- 德州仪器 (TI)： *《TMS320F28004x Piccolo™ 微控制器》数据表*
- 德州仪器 (TI)： *《TMS320F28004x 技术参考手册》*
- 德州仪器 (TI)： *《TMS320F2837xD 双核 Delfino™ 微控制器数据表》*
- 德州仪器 (TI)： *《TMS320F2837xD 双核微控制器技术参考手册》*
- 德州仪器 (TI)： *《具有连接管理器的 TMS320F2838x 微控制器数据表》*
- 德州仪器 (TI)： *《TMS320F2838x 微控制器技术参考手册》*
- 德州仪器 (TI)： *《CLB 工具用户指南》*
- 德州仪器 (TI)： *《TMSF28078x 微控制器数据表》*

6 修订历史记录

注：以前版本的页码可能与当前版本的页码不同

Changes from SEPTEMBER 28, 2019 to JULY 30, 2020 (from Revision * (September 2019) to Revision A (July 2020)) **Page**

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