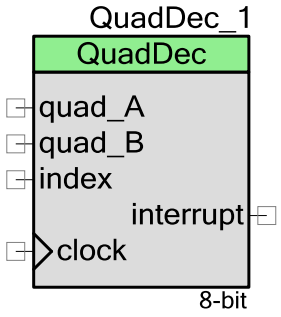
正交解码器 (QuadDec)

2.0

特性

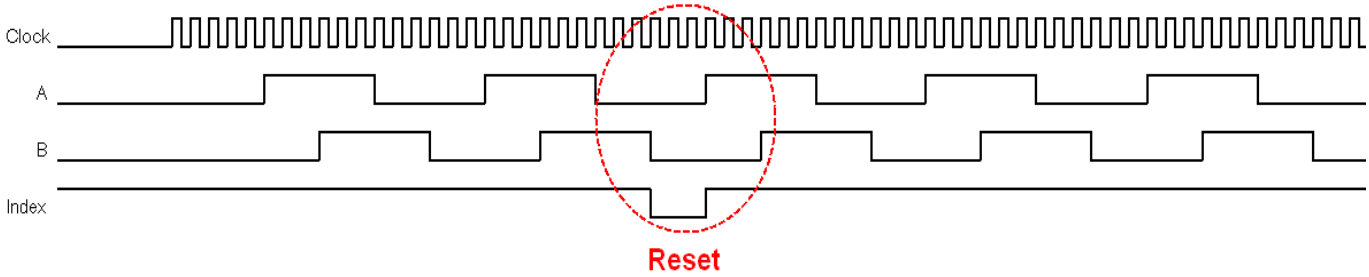


- 可调计数器大小：8、16 或 32 位
- 计数器分辨率为 A 和 B 输入频率的 1x、2x 或 4x，更精确地确定位置或速度
- 可选索引输入，用以确定绝对位置
- 可选短时脉冲过滤，用以减少输入信号上系统产生的噪音的影响

概述

正交解码器 (QuadDec) 组件提供针对成对数字信号跃变进行计数的功能。这些信号通常由电机或轨迹球上安装的速度/位置反馈系统来提供。

这两个信号通常称为 A 和 B，相位上偏离 90 度角，使得输出呈现格雷码规律。格雷码是在每次计数时，仅有一位发生变化的序列。这是避免短时脉冲发生的有效方法。此外，通过格雷码，还可能检测方向和相对位置。第 3 个可选信号命名为 Index（索引），为每转一圈产生一次绝对位置的参考。



何时使用正交解码器

正交解码器用于解码正交编码器的输出。正交解码器感应对象（例如，鼠标、轨迹球、自动控制轴等）当前位置、速度和方向。

此外，正交解码器还用于精确测量电机转子的速度、加速度和位置，并结合旋钮确定用户输入。

输入/输出连接

本节介绍正交解码器组件的各种输入和输出连接。I/O 列表中的星号 (*) 表示，在 I/O 说明中列出的条件满足的情况下，该 I/O 可能不可见。

quad_A – Input

正交解码器的“A”输入。

quad_B – Input

正交解码器的“B”输入。

索引 – 输入 *

此输入用于检测正交解码器的参考位置。使用索引输入时，如果输入 A、B，并且索引全部为 0，则计数器也要复位到 0。通常添加额外逻辑以用来选通索引肪冲。通过索引选通，计数器可以仅在多次旋转的某一次进行复位。例如，线性致动装置仅在达到较远行程限制时，才复位计数器。通过机械限位开关（其输出连接至 Index 脉冲）发送此限制信号。

默认情况下显示此输入，但取消选择 **Use index input**（使用索引输入）参数时，此输入可以被隐藏。

时钟 – 输入

时钟信号用于采样和过滤输入上的短时脉冲。如果使用短时脉冲过滤，则过滤输出不会发生变化，直到 3 个连续输入采样具有相同值时为止。进行有效短时脉冲过滤时，采样时钟周期应大于短时脉冲预期发生的最大时间周期。计数器可以按 A 和 B 输入频率的 1倍、2倍 或 4倍的分辨率进行递增或递减计数。

时钟输入频率应大于或等于 A 或 B 输入频率最大值的 10 倍。

中断 – 输出

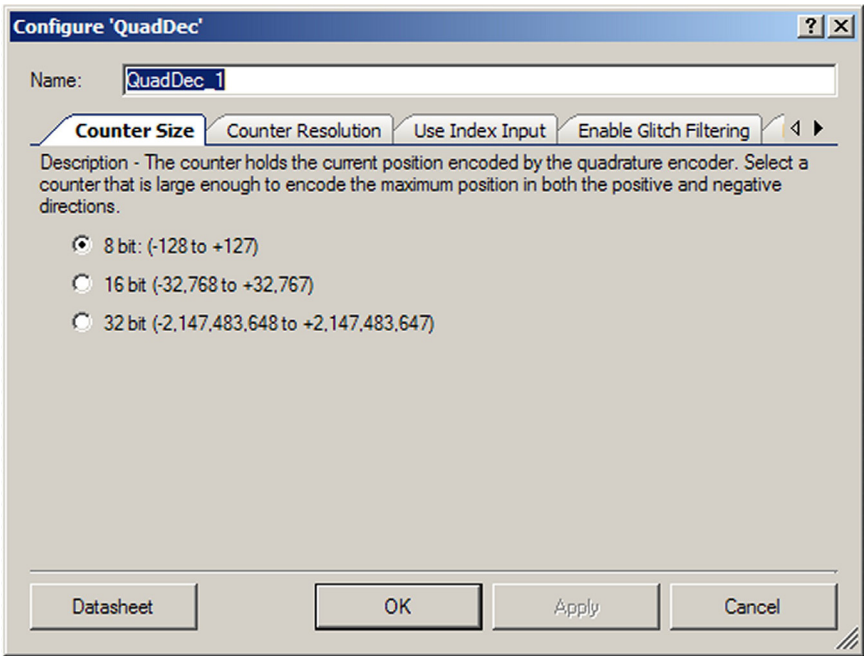
在下列一种或多种情况下中断：

- 计数器上溢出和下溢出
- 计数器因索引输入而被复位（若使用索引）
- A 和 B 输入上产生无效状态跃变

元件参数

将正交解码器组件拖入设计中，双击该组件，打开 **Configure**（配置）对话框。该对话框包含多个选项卡与分类参数。

Counter Size（计数器大小）选项卡

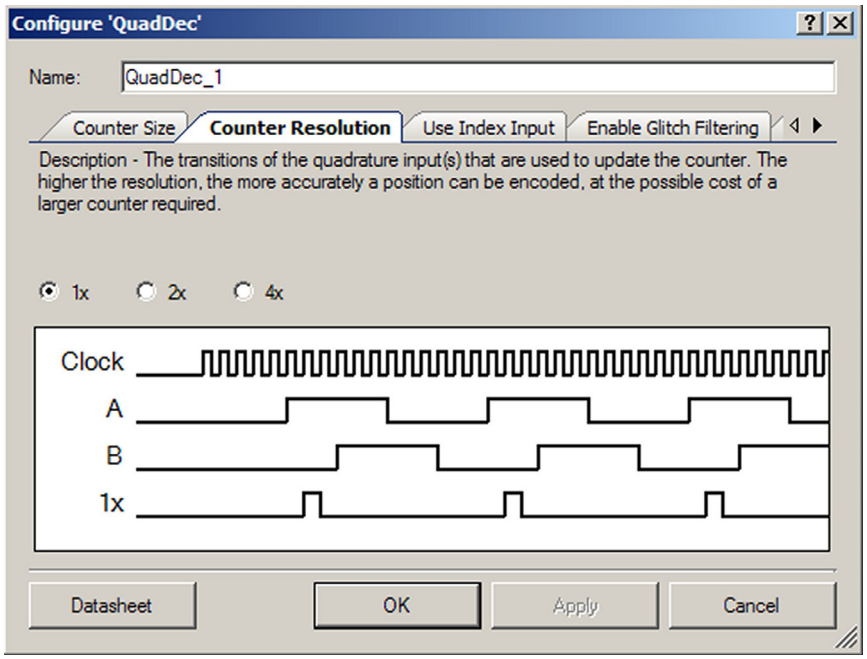


此选项卡用来定义计数器大小（以位数表示）。计数器保存的是正交编码器所编码的当前位置。

选择足够大的计数器来对正向和负向的最大位置进行编码。设置可能是：**8 位**、**16 位** 或 **32 位**。

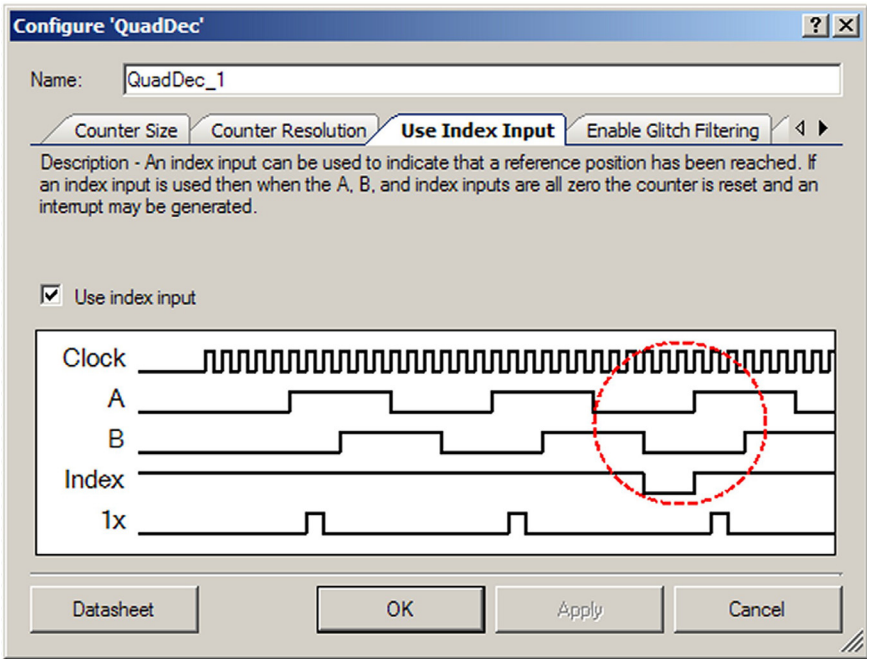
32 位计数器是在硬件计数器中实现低 16 位并在软件中实现高 16 位，从而降低硬件资源使用量。出于此目的，需要使用一个额外的 ISR。要在 32 位计数器下正常工作，必须启用中断。如有必要，可以将 ISR 代码添加至源文件；有关详细信息，请参考“中断组件”数据手册。

Counter Resolution（计数器分辨率）选项卡



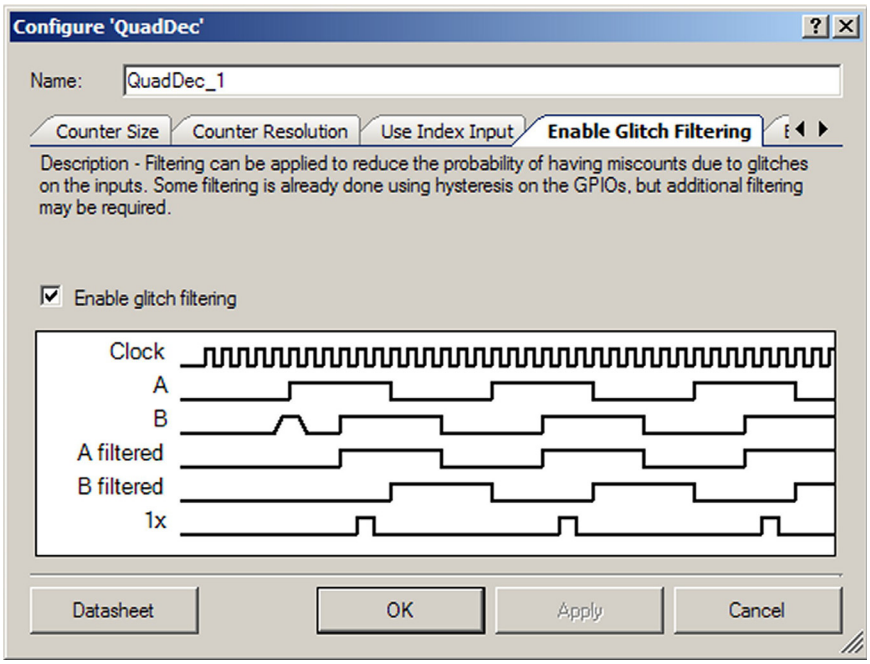
此选项卡包含在 A 和 B 输入一个周期内记数的个数。它表示了用于更新计数器的输入信号的跃变的次数。当分辨率变高时，可以更精确地解析位置，但这可能需要较大计数器。该设置可能是 **1x**、**2x** 或 **4x**。

Use Index Input（使用索引输入）选项卡



此选项卡包含用于启用或禁用索引输入的字段。索引输入用于表示已经达到参考位置。如果使用索引输入，那么当 A、B 和索引输入全部为 0 时，计数器复位，并生成中断。默认情况下，索引输入为启用状态。

Enable Glitch Filtering（启用短时脉冲过滤）选项卡



此选项卡包含用于启用或禁用数字短时脉冲过滤的字段。应用过滤，可以降低由于输入时的短时脉冲而发生错误计数的可能性。使用在 GPIO 上的输入迟滞，已经完成某些过滤，但额外过滤还是需要的。

如果启用，则过滤适用于所有输入。过滤输出不会发生变化，直到 3 个连续输入采样具有相同值时为止。为达到有效过滤，采样时钟周期应大于短时脉冲预期发生的最大时间周期。默认情况下，短时脉冲过滤为启用状态。

时钟选择

正交解码器组件时钟源必须连接。它为状态寄存器提供时钟并生成中断。

放置

正交解码器组件放置于整个 UDB 阵列中，并且所有放置信息通过 *cyfitter.h* 文件提供给 API。

资源

1x 分辨率（无短时脉冲过滤）

资源	资源类型					API Memory（API 存储器）（字节）		Pins（引脚）（每个外部 I/O）
	Datapath 单元	PLD	Status单元	Control/ Count7 单元	中断	Flash（闪存）	RAM	
8 位	1	6	2	1	0	566	7	2
8 位 *	1	6	2	1	0	566	7	3
16 位	2	6	2	1	0	652	9	2
16 位 *	2	6	2	1	0	652	9	3
32 位	2	6	2	1	1	906	14	2
32 位 *	2	9	2	1	1	906	14	3

* 使用索引输入

1x 分辨率（有短时脉冲过滤）

资源	资源类型					API Memory（API 存储器）（字节）		Pins（引脚）（每个外部 I/O）
	Datapath 单元	PLD	Status单元	Control/ Count7 单元	中断	Flash（闪存）	RAM	
8 位	1	7	2	1	0	566	7	2
8 位 *	1	9	2	1	0	566	7	3
16 位	2	7	2	1	0	652	9	2
16 位 *	2	9	2	1	0	652	9	3
32 位	2	7	2	1	1	906	14	2
32 位 *	2	9	2	1	1	906	14	3

* 使用索引输入

2x 分辨率（无短时脉冲过滤）

资源	资源类型					API Memory（API 存储器）（字节）		Pins（引脚）（每个外部 I/O）
	Datapath 单元	PLD	Status 单元	Control/ Count7 单元	中断	Flash（闪存）	RAM	
8 位	1	6	2	1	0	566	7	2
8 位 *	1	7	2	1	0	566	7	3
16 位	2	6	2	1	0	652	9	2
16 位 *	2	7	2	1	0	652	9	3
32 位	2	6	2	1	1	906	14	2
32 位 *	2	7	2	1	1	906	14	3

* 使用索引输入

2x 分辨率（有短时脉冲过滤）

资源	资源类型					API Memory（API 存储器）（字节）		Pins（引脚）（每个外部 I/O）
	Datapath 单元	PLD	Status 单元	Control/ Count7 单元	中断	Flash（闪存）	RAM	
8 位	1	8	2	1	0	566	7	2
8 位 *	1	9	2	1	0	566	7	3
16 位	2	8	2	1	0	652	9	2
16 位 *	2	9	2	1	0	652	9	3
32 位	2	8	2	1	1	906	14	2
32 位 *	2	9	2	1	1	906	14	3

* 使用索引输入

4x 分辨率（无短时脉冲过滤）

资源	资源类型					API Memory（API 存储器）（字节）		Pins（引脚）（每个外部 I/O）
	Datapath 单元	PLD	Status 单元	Control/ Count7 单元	中断	Flash（闪存）	RAM	
8 位	1	7	2	1	0	566	7	2
8 位 *	1	7	2	1	0	566	7	3
16 位	2	7	2	1	0	652	9	2
16 位 *	2	7	2	1	0	652	9	3
32 位	2	7	2	1	1	906	14	2
32 位 *	2	7	2	1	1	906	14	3

* 使用索引输入

4x 分辨率（有短时脉冲过滤）

资源	资源类型					API Memory（API 存储器）（字节）		Pins（引脚）（每个外部 I/O）
	Datapath 单元	PLD	Status 单元	Control/ Count7 单元	中断	Flash（闪存）	RAM	
8 位	1	8	2	1	0	566	7	2
8 位 *	1	9	2	1	0	566	7	3
16 位	2	8	2	1	0	652	9	2
16 位 *	2	9	2	1	0	652	9	3
32 位	2	8	2	1	1	906	14	2
32 位 *	2	9	2	1	1	906	14	3

* 使用索引输入

应用程序编程接口

应用程序编程接口 (API) 子程序允许您使用软件配置组件。下表列出了每个函数的接口，并进行了说明。以下各节将更详细地介绍每个函数。

默认情况下，PSoC Creator 将实例名称“QuadDec_1”分配给指定设计中组件的第一个实例。您可以将其重命名为遵循标识符语法规则的任何唯一值。实例名称会成为每个全局函数名称、变量和常量符号的前缀。出于可读性考虑，下表中使用的实例名称为“QuadDec”。

函数	说明
QuadDec_Start()	初始化 UDB 和其他相关硬件
QuadDec_Stop()	关闭 UDB 和其他相关硬件
QuadDec_GetCounter()	报告计数器的当前值
QuadDec_SetCounter()	设置计数器的当前值
QuadDec_GetEvents()	报告事件的当前状态
QuadDec_SetInterruptMask()	启用或禁用因事件引发的中断
QuadDec_GetInterruptMask()	报告当前中断掩码设置
QuadDec_Sleep()	准备要进入睡眠的组件
QuadDec_Wakeup()	准备要唤醒的组件
QuadDec_Init()	初始化或恢复自定义程序提供的默认配置
QuadDec_Enable()	启用正交解码器
QuadDec_SaveConfig()	保存当前用户配置
QuadDec_RestoreConfig()	恢复用户配置

全局变量

函数	说明
QuadDec_initVar	QuadDec_initVar 表示正交解码器是否完成初始化。变量将初始化为 0，并在第一次调用 QuadDec_Start() 时设置为 1。这样，第一次调用 QuadDec_Start() 子程序后，组件不用重新初始化即可重启。 如果组件需要重新初始化，则首先调用 QuadDec_Init() 函数，然后再调用 QuadDec_Start() or QuadDec_Enable() 函数。
QuadDec_count32SoftPart	在此变量中存储 32 位计数器的高16 位。
QuadDec_swStatus	在此变量中存储状态寄存器的值。

void QuadDec_Start(void)

说明： 初始化 UDB 和其他相关硬件。将计数器复位到 0，然后启用或禁用所有相关中断。启动输入监控和计数。

参数： None（无）

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

void QuadDec_Stop(void)

说明： 关闭 UDB 和其他相关硬件。

参数： None（无）

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

int8/16/32 QuadDec_GetCounter(void)

说明： 报告计数器的当前值。

参数： None（无）

Return Value（返回值）： int8/16/32: 计数器的值。返回带符号类型，这取决于计数器大小设置。正值表示顺时针移动（B 在 A 之前）。

Side Effects（副作用）： None（无）

void QuadDec_SetCounter(int8/16/32 value)

说明： 设置计数器的当前值。

参数： int8/16/32 value: 新增值。参数是有符号类型，这取决于计数器大小设置。

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

uint8 QuadDec_GetEvents(void)

说明： 报告事件的当前状态。

参数： None（无）

Return Value（返回值）： 事件，以无符号的 8 位值中的位来表示：

Bit（位）	说明
QuadDec_COUNTER_OVERFLOW	计数器上溢出
QuadDec_COUNTER_UNDERFLOW	计数器下溢出
QuadDec_COUNTER_RESET	计数器由于索引而复位，假设使用索引输入
QuadDec_INVALID_IN	无效的A 和 B 输入状态

Side Effects（副作用）： None（无）

void QuadDec_SetInterruptMask(uint8 mask)

说明： 启用或禁用因事件引发的中断。对于 32 位计数器而言，无法禁用上溢出、下溢出和复位中断；这些位被忽略。

参数： uint8 掩码：启用或禁用 8 位值中的各个位，其中值为 1 时启用中断：

Bit（位）	说明
QuadDec_COUNTER_OVERFLOW	由于计数器上溢出而启用中断
QuadDec_COUNTER_UNDERFLOW	由于计数器下溢出而启用中断
QuadDec_COUNTER_RESET	由于计数器复位而启用中断
QuadDec_INVALID_IN	由于输入状态跃变无效而启用中断

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

uint8 QuadDec_GetInterruptMask(void)

说明： 报告当前中断掩码设置。

参数： None（无）

Return Value（返回值）： 启用或禁用 8 位值中的各个位，其中值为 1 时启用中断。
对于 32 位计数器而言，上溢出、下溢出和复位使能位始终为1。

Bit（位）	说明
QuadDec_COUNTER_OVERFLOW	由于计数器上溢出而中断
QuadDec_COUNTER_UNDERFLOW	由于计数器下溢出而中断
QuadDec_COUNTER_RESET	由于计数器复位而中断
QuadDec_INVALID_IN	由于 A 和 B 输入状态跃变无效而中断

Side Effects（副作用）： None（无）

void QuadDec_Sleep(void)

说明： 这是准备组件睡眠的首选子程序。QuadDec_Sleep() 子程序保存当前组件状态。然后它调用 QuadDec_Stop() 函数和 QuadDec_SaveConfig() 保存硬件配置。

调用 QuadDec_Sleep() 函数后，调用 CyPmSleep() 或 CyPmHibernate() 函数。有关电源管理函数的更多信息，请参考 PSoC Creator *System Reference Guide*（《系统参考指南》）。

参数： None（无）

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

void QuadDec_Wakeup(void)

说明： 这是用来将组件恢复到调用 QuadDec_Sleep() 时的状态的首选子程序。QuadDec_Wakeup() 函数调用 QuadDec_RestoreConfig() 函数以用来恢复该配置。如果在调用 QuadDec_Sleep() 函数前启用该组件，则 QuadDec_Wakeup() 函数还将重新启用该组件。

参数： None（无）

Return Value（返回值）： None（无）

Side Effects（副作用）： 如果在调用 QuadDec_Wakeup() 函数之前未调用 QuadDec_Sleep() 或 QuadDec_SaveConfig() 函数，可能产生意外行为。

void QuadDec_Init(void)

说明： 根据自定义 Configure（配置）对话框设置来初始化或恢复组件。无需调用 QuadDec_Init()，因为 QuadDec_Start() 子程序调用此函数，它是开始组件操作的首选方法。

参数： None（无）

Return Value（返回值）： None（无）

Side Effects（副作用）： 所有寄存器将设置为自定义 Configure（配置）对话框中的值。

void QuadDec_Enable(void)

说明： 激活硬件并开始执行组件操作。无需调用 QuadDec_Enable()，因为 QuadDec_Start() 子程序调用此函数，它是开始组件操作的首选方法。

参数： None（无）

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

void QuadDec_SaveConfig(void)

说明： 此函数会保存组件配置和非保留寄存器。此外，此函数保存当前组件参数值，这些值是在 Configure（配置）对话框中定义或通过相应的 API 进行修改。通过 QuadDec_Sleep() 函数调用此函数。

参数： None（无）

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

void QuadDec_RestoreConfig(void)

说明：	此函数会恢复组件配置和非保留寄存器。此外，此函数还用于将组件参数值恢复至调用 QuadDec_Sleep() 函数之前的状态。
参数：	None（无）
Return Value（返回值）：	None（无）
Side Effects（副作用）：	如果在调用此函数之前未调用 QuadDec_Sleep() 或 QuadDec_SaveConfig() 函数，可能产生意外行为。

固件源代码示例

PSoC Creator 在“查找示例项目”对话框中提供了大量包括原理图和代码示例的示例项目。要获取组件特定的示例，请打开组件目录中的对话框或原理图中的组件实例。要获取通用的示例，请打开 Start Page（开始页）或 **File**（文件）菜单中的对话框。根据需要，使用对话框中的 **Filter Options**（滤波器选项）可缩小可选项目的列表。

有关更多信息，请参见 PSoC Creator 帮助中的“Find Example Project（查找示例项目）”主题。

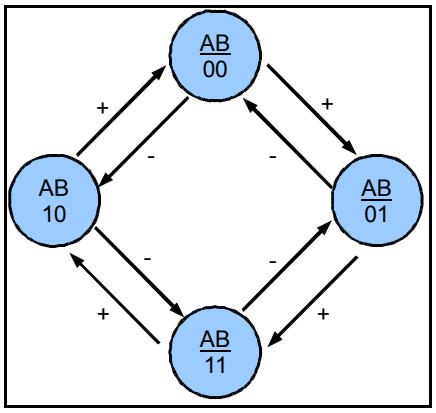
功能描述

默认配置

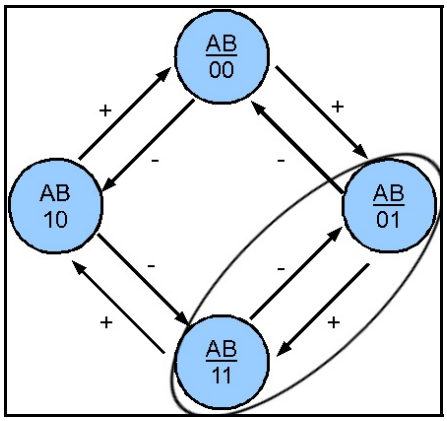
正交解码器的默认配置为 8 位递增/递减计数器，分辨率为 1x，启用索引输入和短时脉冲过滤。

状态跃变

正交相位信号通常使用状态机和递增/递减计数器进行解码。传统解码器有 4 种状态，对应于 A 和 B 输入所有可能的值。状态跃变图如下所示（未描述相同状态跃变）。标有 “+” 和 “–” 的状态跃变表示正交相位计数器递增和递减操作。



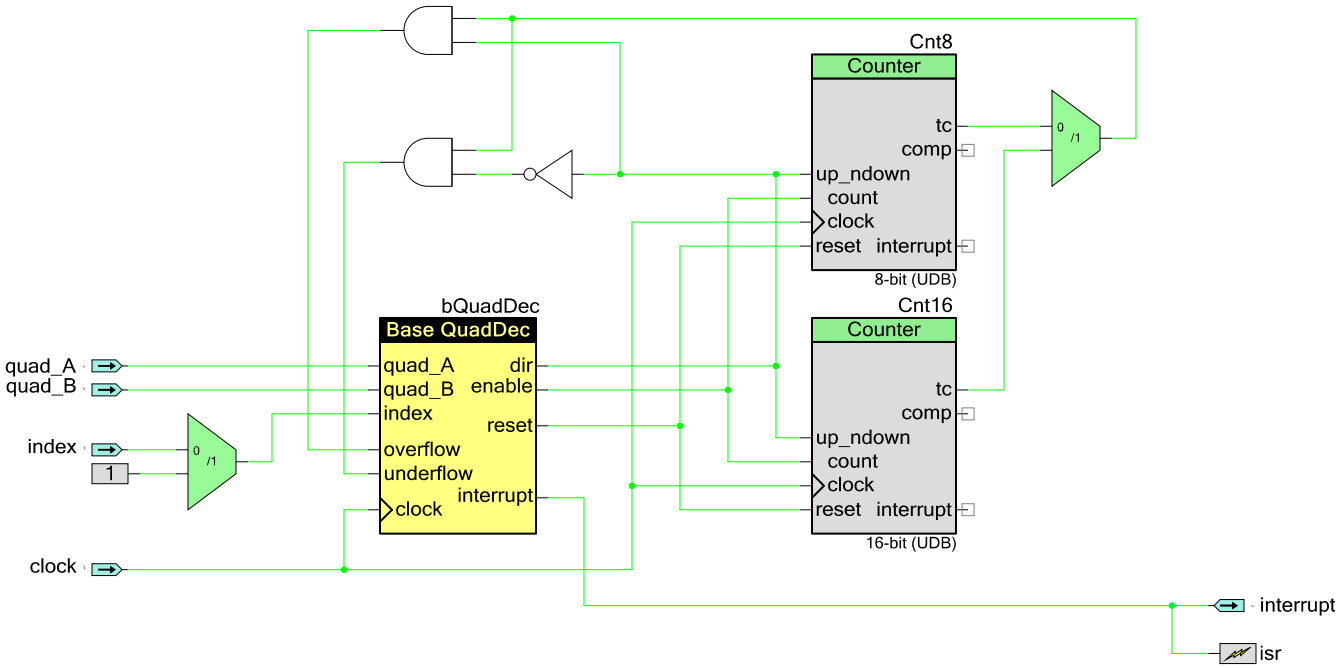
对于正交相位信号的每一个满周期，正交相位计数器发生 4 次计数变化。此外，较低分辨率计数器还可以仅用于实现状态跃变子集的递增/递减运算。四分之一分辨率解码器如下所示。



使用器件内部衍生的时钟信号来采样所有输入。

框图和配置

正交解码器仅使用 UDB 进行 配置。在此文档前面已经描述了 API，并会在下一节中描述寄存器，从而定义了组件的全部实现。



寄存器

状态

位	7	6	5	4	3	2	1	0
值	保留				无效	复位	下溢出	上溢出

状态寄存器为只读状态。它包含为正交解码器定义的各种状态位。此寄存器的值可以通过 QuadDec_GetEvents() 函数来获得。从此状态寄存器内的掩码位域的 OR 运算生成中断输出信号。

可以使用 QuadDec_SetInterruptMask() 函数设置掩码。收到中断后，可以通过 QuadDec_GetEvents() 函数读取状态寄存器，由此获取中断源。状态寄存器是透明的，因此 QuadDec_GetEvents() 不清除状态寄存器的位。状态寄存器的所有运算必须使用以下位域定义，因为这些位字段可能在编译时在状态寄存器中产生移位。

为状态寄存器定义了多个位域掩码。在这些位字段中，任何一个位字段均可能作为中断源包括在内。所有位字段在状态寄存器中均可以配置为粘连位。在生成的头 (.h) 中可以获得这些定义，如下所示：

- **QuadDec_COUNTER_OVERFLOW** – 定义为状态寄存器位“计数器上溢出”的位掩码。
- **QuadDec_COUNTER_UNDERFLOW** – 定义为状态寄存器位“计数器下溢出”的位掩码。
- **QuadDec_RESET** – 定义为状态寄存器位“因索引被复位”的位掩码。
- **QuadDec_INVALID_IN** – 定义为状态寄存器位“A 和 B 输入状态跃变无效”的位掩码。

直流和交流电气特性

下面的值表示了预计性能，它们基于初始的特性化数据。

时序特性“额定路由的最大值”

参数	说明	配置	最小值	典型值	最大值	单位
f_{CLOCK}	组件时钟频率	配置 1 [1]			34	MHz
		配置 2 [2]			29	MHz
		配置 3 [3]			16	MHz
t_{CLOCKH}	输入时钟高电平时间 [4]	不可用		0.5		$1/f_{CLOCK}$
t_{CLOCKL}	输入时钟低电平时间 [7]	不可用		0.5		$1/f_{CLOCK}$
输入						
t_{PD_ps}	输入路径延迟，要引脚相对于同步 [5]	1			STA [6]	ns

[1] Config 1（配置 1）选项：
CounterResolution： 1
CounterSize： 8
UsingGlitchFiltering： 假
UsingIndexInput： 真

[2] Config 2（配置 2）选项：
CounterResolution： 2
CounterSize： 16
UsingGlitchFiltering： 真
UsingIndexInput ： 真

[3] Config 3（配置 3）选项：
CounterResolution： 4
CounterSize： 32
UsingGlitchFiltering： 真
UsingIndexInput： 真

[4] $t_{CY_clock} = 1/f_{CLOCK}$ - 一个时钟周期的时间

[5] t_{PD_ps} 可以在静态时序结果中找到，如后面章节所述。此处列出的数字是基于许多输入的 STA 分析的额定值。

参数	说明	配置	最小值	典型值	最大值	单位
t_{PD_ps}	输入路径延迟，从引脚到同步 [7]	2			8.5	ns
t_{PD_IE}	组件时钟的输入路径延迟（边沿敏感输入）	1,2	$t_{PD_ps} + t_{SYNC} + t_{PD_si}$		$t_{PD_ps} + t_{SYNC} + t_{PD_si} + t_{I_clk}$	ns
t_{PD_si}	同步输出到输入路径延迟（路由）	1,2,3,4			STA [6]	ns
t_{I_clk}	clockX 与时钟的对齐	1,2,3,4	0		1	t_{CY_clock}
t_{IH}	输入高电平时间	1,2	t_{CY_clock}			ns
t_{IL}	输入低电平时间	1,2	t_{CY_clock}			ns
t_{PD_IE}	组件时钟的输入路径延迟（边沿敏感输入）	3,4	$t_{SYNC} + t_{PD_si}$		$t_{SYNC} + t_{PD_si} + t_{I_clk}$	ns
t_{IH}	输入高电平时间	1,2,3,4	t_{CY_clock}			ns
t_{IL}	输入低电平时间	1,2,3,4	t_{CY_clock}			ns
f_{AB}	组件 A 和 B 频率	不可用			$f_{CLOCK}/10$	MHz
t_{IND}	索引信号宽度	无短时脉冲过滤	$2 \times t_{CY_clock} + 5$			ns
		有短时脉冲过滤	$3 \times t_{CY_clock} + 5$			ns
t_{RD}	索引输入低电平到复位时间	无短时脉冲过滤		2		t_{CY_clock}
		有短时脉冲过滤		5		t_{CY_clock}
t_{GL}	短期脉冲预期发生的时间周期	有短时脉冲过滤			3	t_{CY_clock}
t_{CD}	延迟时间，从时钟上升沿到有效计数	不可用		2		t_{CY_clock}
t_E	编码脉冲宽度（低电平或高电平）	不可用	4			t_{CY_clock}
t_{ES}	编码状态周期	不可用	2			t_{CY_clock}
t_{ELP}	编码周期宽度	不可用	10			t_{CY_clock}

[6] t_{PD_ps} 和 t_{PD_si} 路由路径延迟。由于路由是动态的，这些值可以变更，并直接影响组件时钟和同步时钟频率的最大值。在静态时序分析结果中才能够找到这些值。

[7] 配置 2 中的 t_{PD_ps} 是为器件的每个引脚定义的固定值。此处列出的数字是该器件上可用的所有引脚的额定值。

时序特性“所有路由的最大值”

参数	说明	配置	最小值	典型值	最大值 [1]	单位
f_{CLOCK}	组件时钟频率	配置 1 [2]			17	MHz
		配置 2 [3]			14	MHz
		配置 3 [4]			8	MHz
t_{CLOCKH}	输入时钟高电平时间 [5]	不可用		0.5		$1/f_{CLOCK}$
t_{CLOCKL}	输入时钟低电平时间 [5]	不可用		0.5		$1/f_{CLOCK}$
输入						
t_{PD_ps}	输入路径延迟，从引脚到同步 [6]	1			STA [7]	ns
t_{PD_ps}	输入路径延迟，从引脚到同步 [8]	2			8.5	ns
t_{PD_IE}	组件时钟的输入路径延迟（边沿敏感输入）	1,2	t_{PD_ps} + t_{SYNC} + t_{PD_si}		t_{PD_ps} + t_{SYNC} + t_{PD_si} + t_{I_clk}	ns
t_{PD_si}	同步输出到输入路径延迟（路由）	1,2,3,4			STA[7]	ns

[1] “所有路由”的最大值的计算方法是：<额定值>/2，然后取整到最近的整数。此值提供了一个基础，因此用户不必担心在以该频率或低于组件频率运行该组件时遇到时序问题。

[2] Config 1（配置 1）选项：
- CounterResolution： 1
- CounterSize： 8
- UsingGlitchFiltering： 假
- UsingIndexInput： 真

[3] Config 2（配置 2）选项：
- CounterResolution： 2
- CounterSize： 16
- UsingGlitchFiltering： 真
- UsingIndexInput ： 真

[4] Config 3（配置 3）选项：
- CounterResolution： 4
- CounterSize： 32
- UsingGlitchFiltering： 真
- UsingIndexInput： 真

[5] $t_{CY_clock} = 1/f_{CLOCK}$ - 一个时钟周期的循环时间

[6] t_{PD_ps} ios 可以在静态时序结果中找到，如后面章节所述。此处列出的数字是基于许多输入的 STA 分析的额定值。

[7] t_{PD_ps} 和 t_{PD_si} 路由路径延迟。由于路由是动态的，这些值可以更改，且将直接影响最大组件时钟和同步时钟频率。在静态时序分析结果中才能够找到这些值。

[8] 配置 2 中的 t_{PD_ps} 是为器件的每个引脚定义的固定值。此处列出的数字是该器件上可用的所有引脚的额定值。

参数	说明	配置	最小值	典型值	最大值 [1]	单位
t_{I_clk}	clockX 与时钟的对齐	1,2,3,4	0		1	t_{CY_clock}
t_{IH}	输入高电平时间	1,2	t_{CY_clock}			ns
t_{IL}	输入低电平时间	1,2	t_{CY_clock}			ns
t_{PD_IE}	组件时钟的输入路径延迟（边沿敏感输入）	3,4	t_{SYNC} + t_{PD_si}		t_{SYNC} + t_{PD_si} + t_{I_clk}	ns
t_{IH}	输入高电平时间	1,2,3,4	t_{CY_clock}			ns
t_{IL}	输入低电平时间	1,2,3,4	t_{CY_clock}			ns
f_{AB}	组件 A 和 B 频率	不可用			f_{CLOCK}/10	MHz
t_{IND}	索引信号宽度	无短时脉冲过滤	2 × t_{CY_clock} + 5			ns
		有短时脉冲过滤	3 × t_{CY_clock} + 5			ns
t_{RD}	索引输入低电平到复位时间	无短时脉冲过滤		2		t_{CY_clock}
		有短时脉冲过滤		5		t_{CY_clock}
t_{GL}	短期脉冲预期发生的时间周期	有短时脉冲过滤			3	t_{CY_clock}
t_{CD}	延迟时间，从时钟上升沿到有效计数	不可用		2		t_{CY_clock}
t_E	编码脉冲宽度（低电平或高电平）	不可用	4			t_{CY_clock}
t_{ES}	编码状态周期	不可用	2			t_{CY_clock}
t_{ELP}	编码周期宽度	不可用	10			t_{CY_clock}

图 1. 不使用短时脉冲过滤的时序图

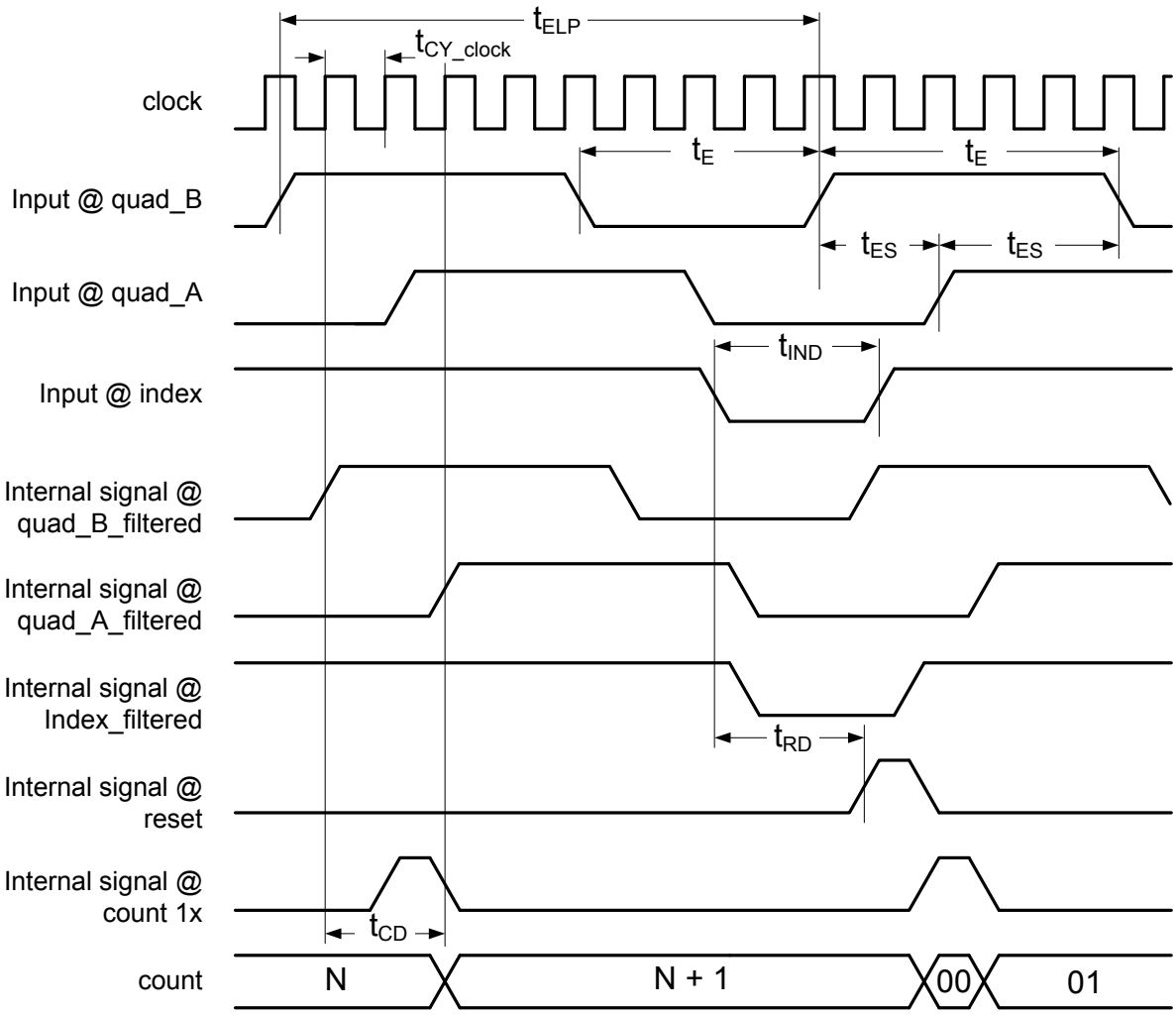
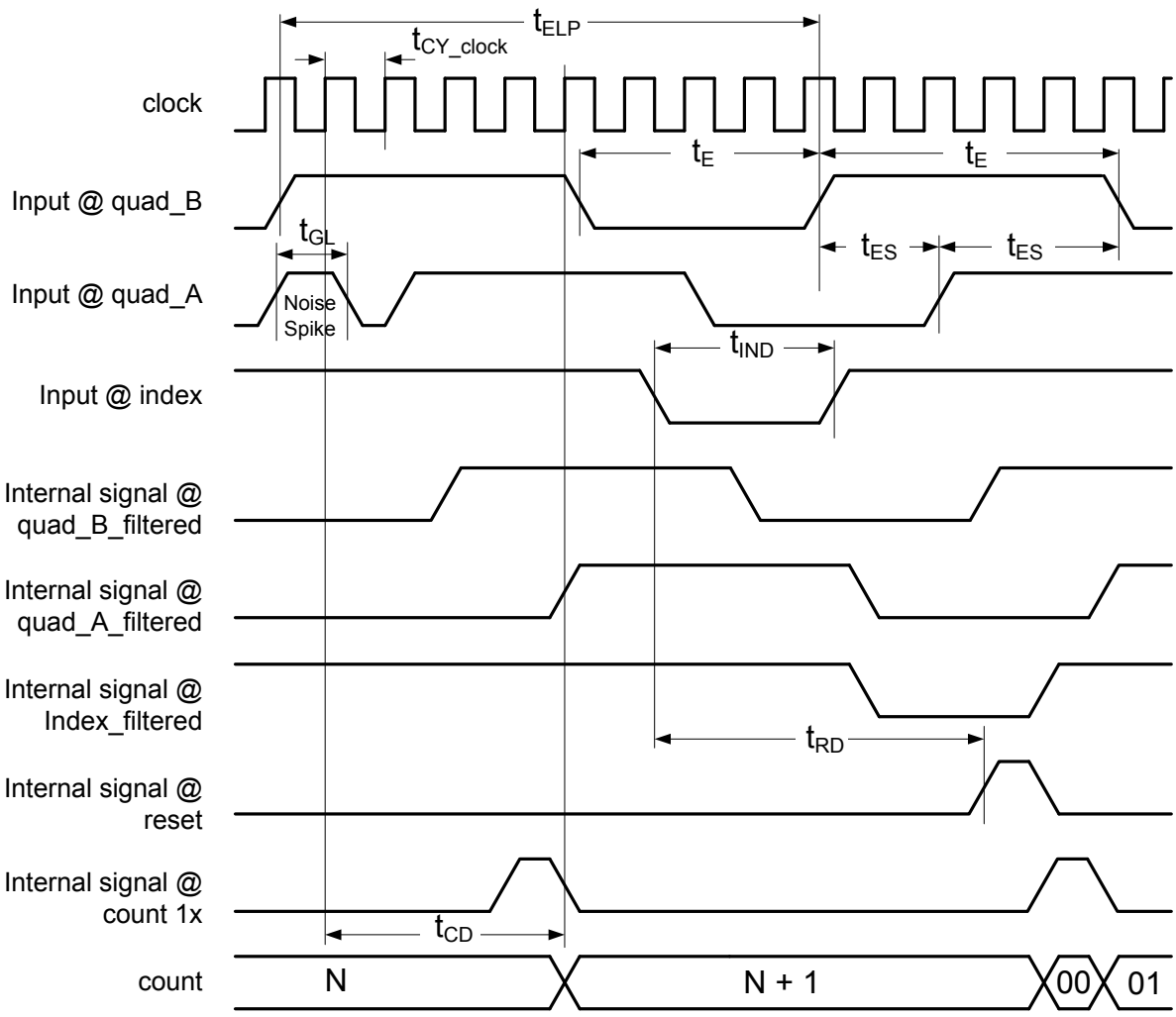


图 2. 使用短时脉冲过滤的时序图



如何将 STA 结果用于特性数据

额定路由最大值是通过使用静态时序分析 (STA) 进行多次测试而收集的。您可以使用下列方法通过 STA 结果计算设计的最大值：

f_{CLOCK} 最大组件时钟频率显示在已命名外部时钟的时钟汇总中的时序结果中。下图演示了 *_timing.html* 文件中的时钟限制示例：

-Clock Summary

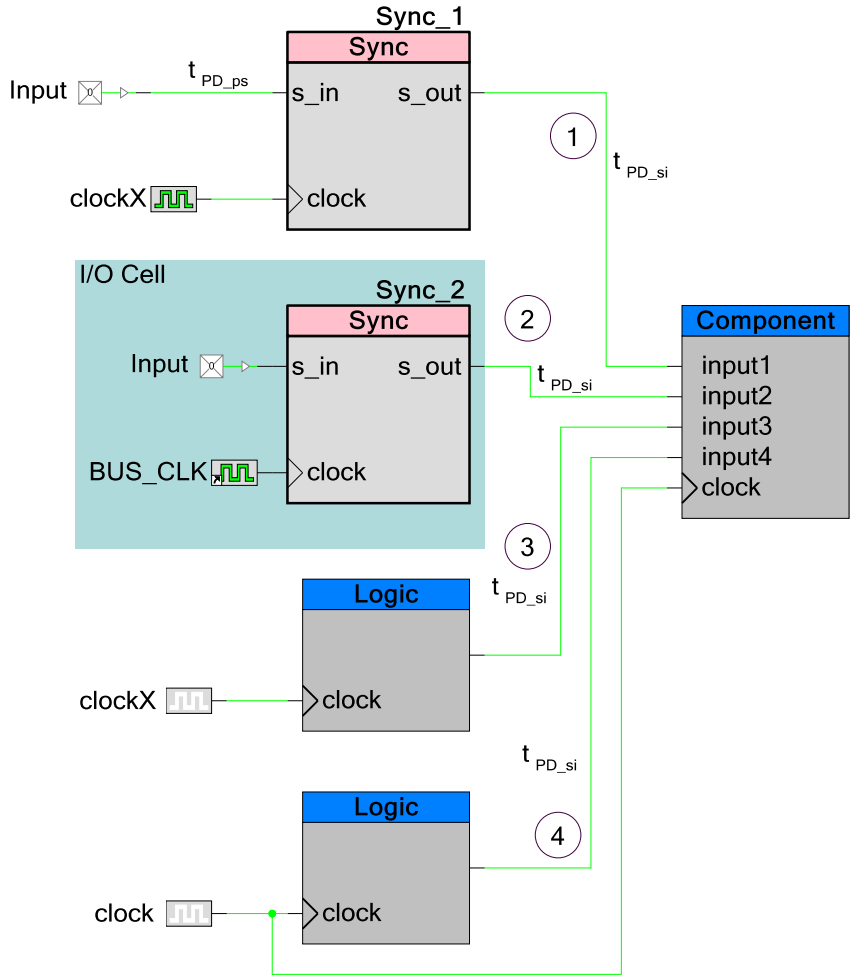
Clock	Actual Freq	Max Freq	Violation
BUS_CLK	24.000 MHz	118.683 MHz	
clock	24.000 MHz	56.967 MHz	

输入路径延迟和脉冲宽度

当表现输入功能的特征时，所有输入（无论您如何配置它们）看上去都类似于这 4 种可能的配置之一，如图 3 所示。

必须同步所有输入。同步机制取决于组件输入源。为了完全解析您的系统如何工作，您必须了解已为每个输入设置哪个输入配置以及系统的时钟配置。本节介绍如何使用静态时序分析 (STA) 结果确定系统的特性。

图 3. 组件时序规范的输入配置



配置	组件时钟	同步器时钟（频率）	图形
1	master_clock	master_clock	图 8
1	时钟	master_clock	图 6
1	时钟	clockX = 时钟 [1]	图 4
1	时钟	clockX > 时钟	图 5
1	时钟	clockX < 时钟	图 7
2	master_clock	master_clock	图 8
2	时钟	master_clock	图 6
3	master_clock	master_clock	图 13

[1] 时钟频率相等，但是不保证上升沿的对齐。

配置	组件时钟	同步器时钟（频率）	图形
3	时钟	master_clock	图 11
3	时钟	clockX = 时钟 [1]	图 9
3	时钟	clockX > 时钟	图 10
3	时钟	clockX < 时钟	图 12
4	master_clock	master_clock	图 13
4	时钟	时钟	图 9

1. 输入由器件引脚驱动，并在内部与“同步”组件同步。此组件的时钟采用与组件所使用的时钟不同的内部时钟（所有内部时钟均派生于 master_clock）。

   当表现按此方法配置的输入的特性时，clockX 可以快于、等于或慢于组件时钟。它还可以等于 master_clock，该时钟生成如图 4、图 5、图 7 和图 8 所示的特性参数。

2. 输入由器件引脚驱动，并使用 master_clock 在引脚同步。

   当表现按此方法配置的输入的特性时，master_clock 快于或等于组件时钟（从未慢于组件时钟）。这会生成如图 5 和图 8 所示的特性参数。

图 4. 输入配置 1 和 2；同步时钟频率 = 组件时钟频率（不保证时钟和 clockX 的边沿对齐）

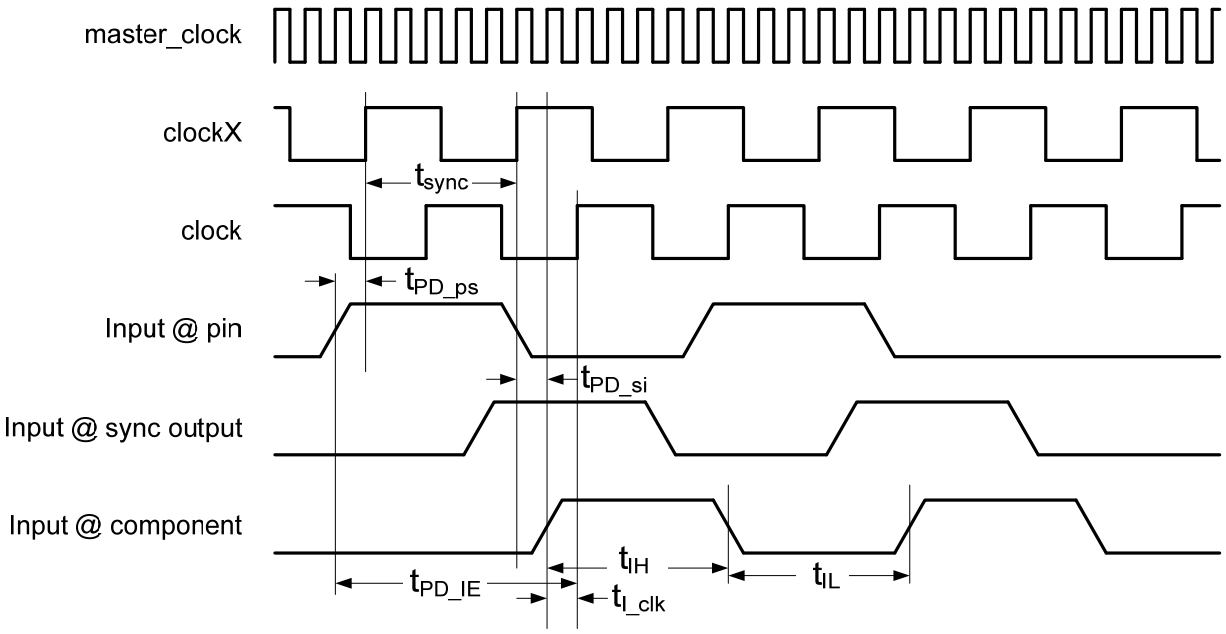


图 5. 输入配置 1 和 2：同步 时钟频率 > 组件时钟频率

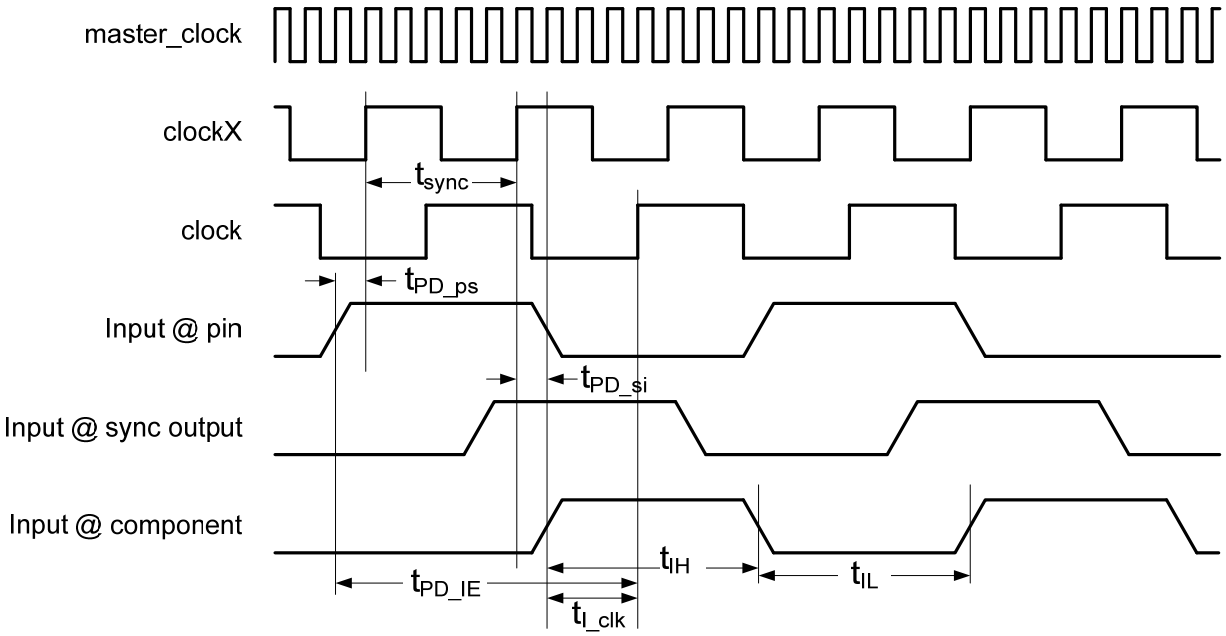


图 6. 输入配置 1 和 2：[同步 时钟频率 == master_clock] > 组件时钟频率

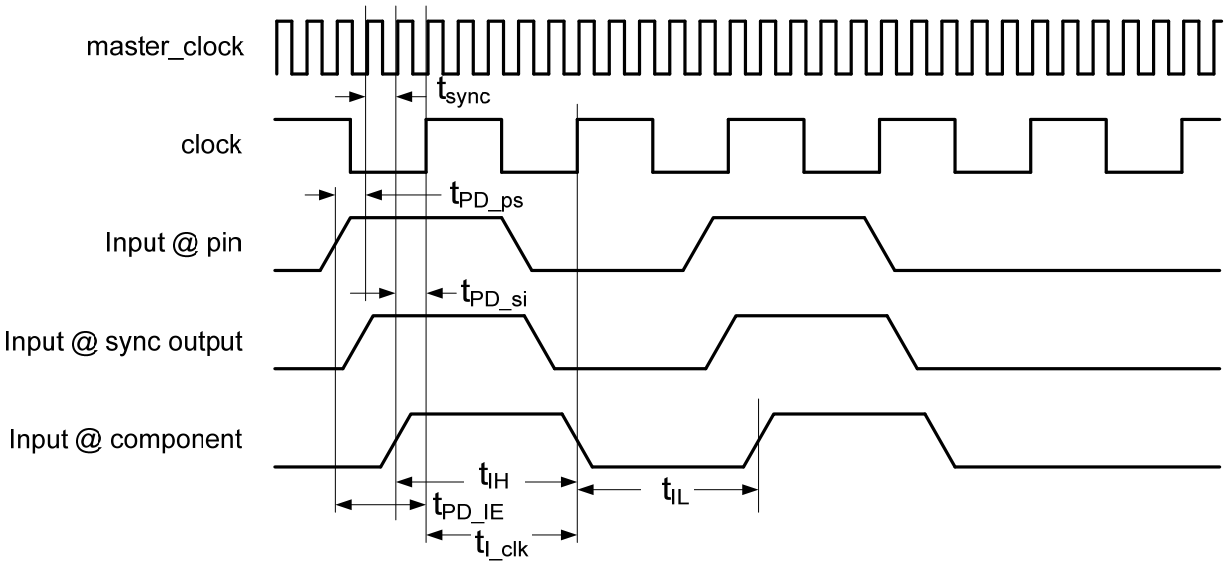


图 7. 输入配置 1：同步 时钟频率 < 组件时钟频率

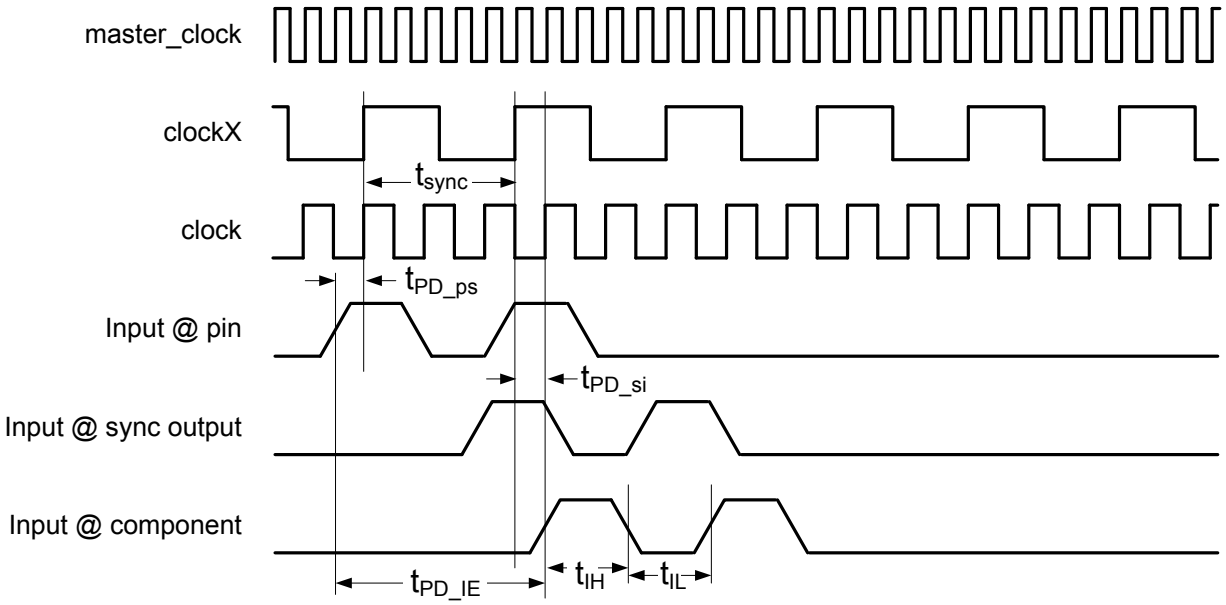
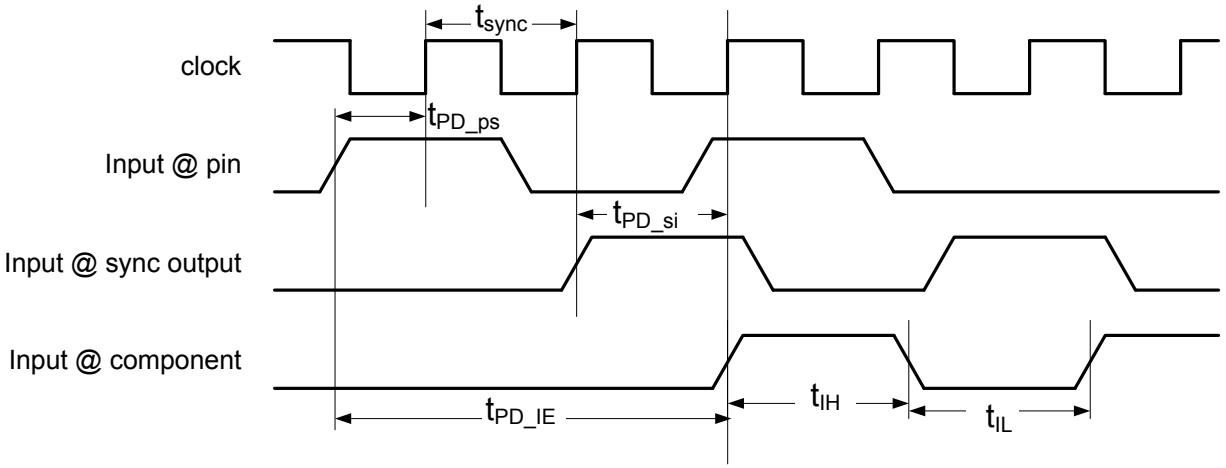


图 8. 输入配置 1 和 2：同步 时钟 = 组件时钟 = master_clock



3. 输入由 PSoC 内部逻辑驱动，它基于与组件所使用的时钟不同的时钟同步（所有内部时钟都派生自 master_clock）。

   当表现按此方法配置的输入的特性时，同步程序时钟快于、慢于或等于组件时钟，该时钟生成如图 9、图 10 和图 12 所示的特性参数。

4. 输入由 PSoC 内部逻辑驱动，它基于与组件所使用的时钟同步。

当表现按此方法配置的输入的特性时，同步程序时钟等于组件时钟，该时钟生成如图 13 所示的特性参数。

图 9. 仅输入配置 3：同步 时钟频率 = 组件时钟频率（不保证时钟和 clockX 的边沿对齐）

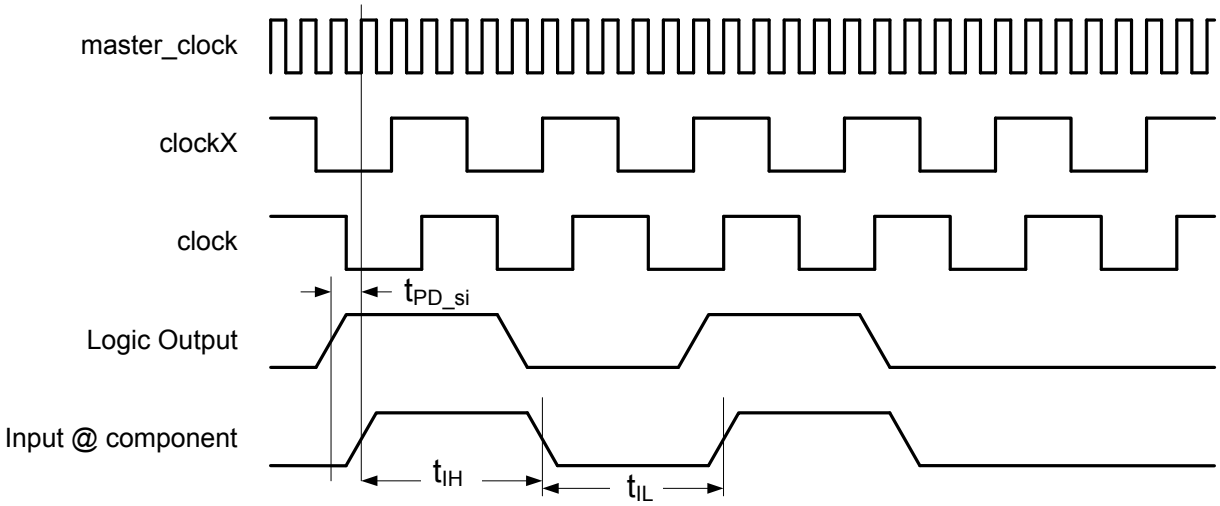
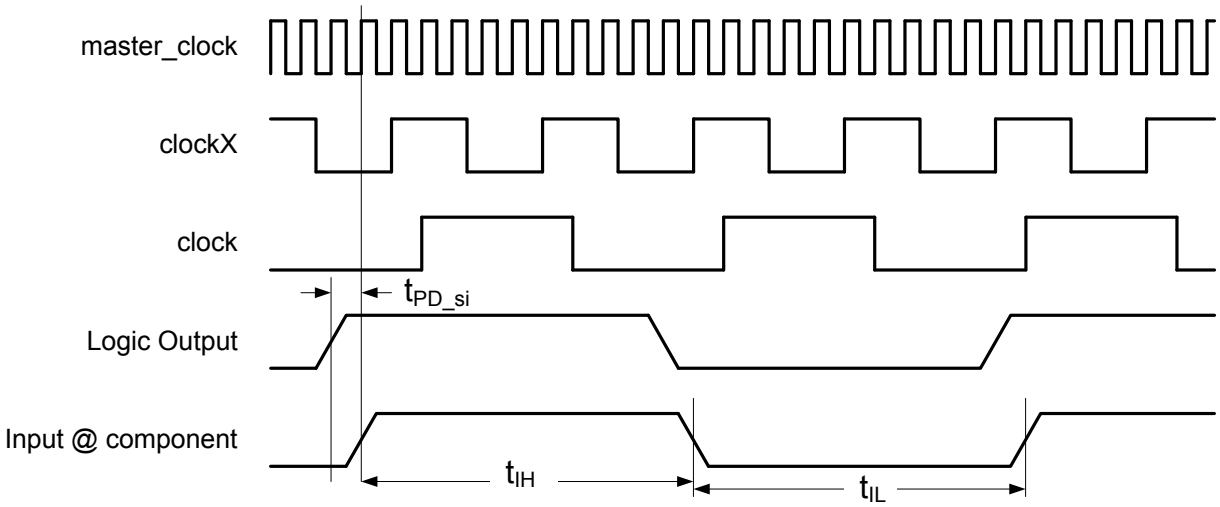


图 9 表示静态时序分析时钟的理解范围。数字时钟域中的所有时钟与 master_clock 同步。然而，可能出现两个时钟具有相同频率，而其上升沿不对齐。因此，静态时序分析工具不了解时钟同步到哪个边沿，必须假设最小值为 1 个 master_clock 周期。这意味着 t_{PD_si} 现在对系统的 master_clock 的影响有限。如果此路径延迟太长，则 master_clock 设置时间出现冲突。您必须更改系统的同步时钟，或者以较慢的频率运行 master_clock。

图 10. 输入配置 3：同步 时钟频率 > 组件时钟频率



与图 9 中的方法几乎相同，所有时钟都派生自 master_clock。STA 在此配置中指明了对一个 master_clock 周期的 master_clock 的 t_{PD_si} 限制。如果此路径延迟太长，则 master_clock 设置时间出现冲突。您必须更改系统的同步时钟，或者以较慢的频率运行 master_clock。

图 11. 输入配置 3：同步器时钟频率 = master_clock > 组件时钟频率

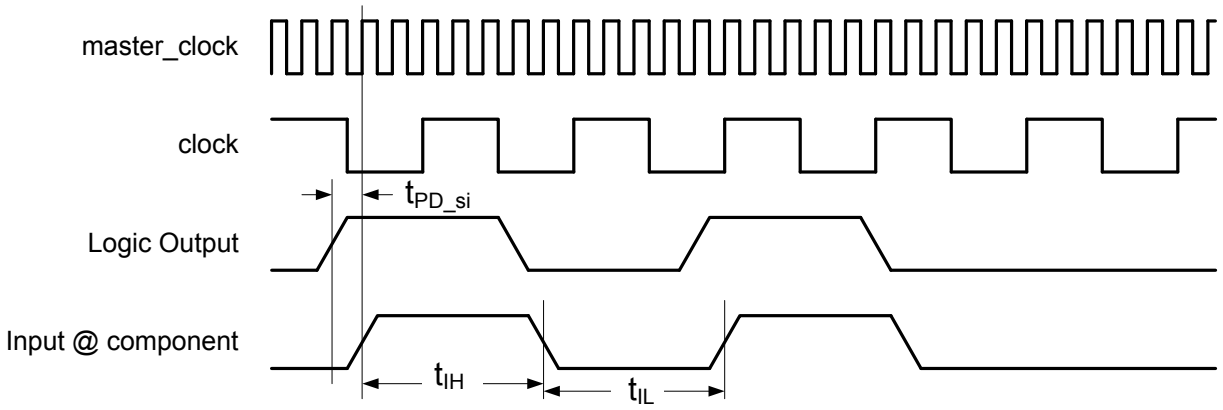
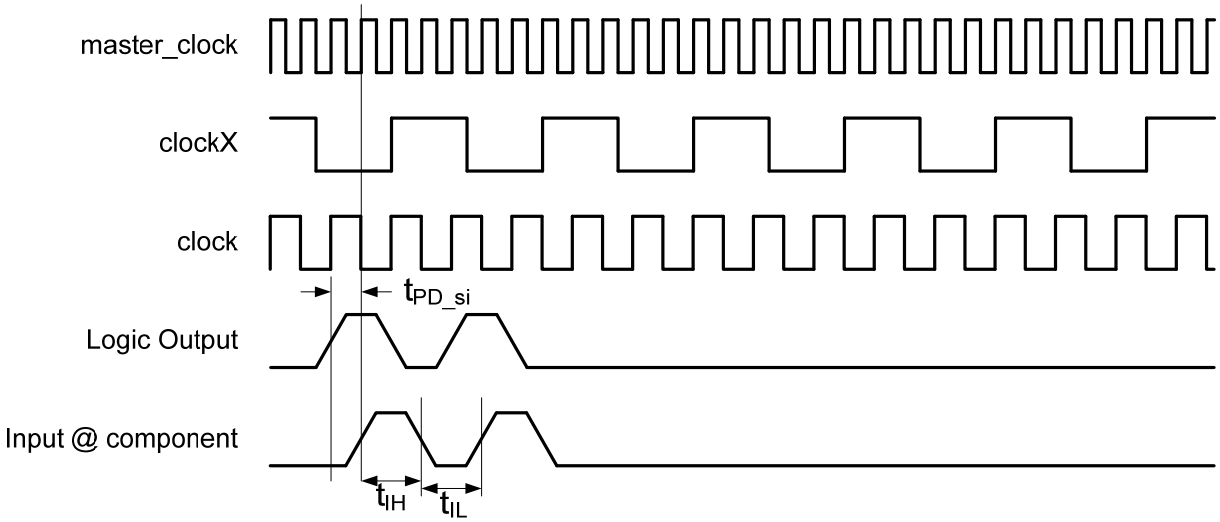
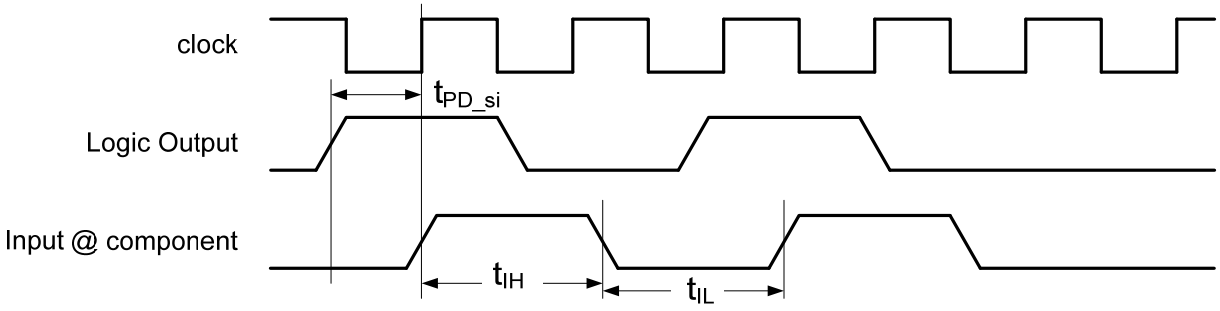


图 12. 输入配置 3：同步器时钟频率 < 组件时钟频率



与图 9 中的方法几乎相同，所有时钟都派生自 master_clock。STA 在此配置中指明了对一个 master_clock 周期的 master_clock 的 t_{PD_si} 限制。如果此路径延迟太长，则 master_clock 设置时间出现冲突。您必须更改系统的同步时钟，或者以较慢的频率运行 master_clock。

图 13. 仅输入配置 4；同步器时钟 = 组件时钟



在本节的所有上述图形中，在了解实现时使用的最关键参数是 f_{CLOCK} 和 t_{PD_IE}。t_{PD_IE} 由 t_{PD_ps} 和 t_{SYNC}（仅针对配置 1 和 2）、t_{PD_si} 和 t_{I_Clk} 定义。至关重要的是注意 t_{PD_si} 用于定义最大的组件时钟频率。t_{I_Clk} 不源自 STA 结果，而是用来表示何时寄存 t_{PD_IE}。这是在同步器与组件时钟之间路由后的余量。

t_{PD_ps} 和 t_{PD_si} 被纳入 STA 结果中。

要查找 t_{PD_ps}，请查看 *_timing.html* 文件中定义的输入设置时间。此输入的扇出可能大于 1，因此需要估算这些路径的最大值。

-Setup times

-Setup times to clock BUS_CLK

Start	Register	Clock	Delay (ns)
input1(0):iocell.pad_in	input1(0):iocell.ind	BUS_CLK	16.500

t_{PD_si} 是在“寄存器到寄存器”时间内定义的。您需要知道使用 *_timing.html* 文件的网络的名称。此路径的扇出可能大于 1，因此将需要估算这些路径的最大值。

-Register-to-register times

-Destination clock clock

Destination clock clock (Actual freq: 24.000 MHz)

+Source clock clock

-Source clock clock_1

Source clock clock_1 (Actual freq: 24.000 MHz)
Affected clock: BUS_CLK (Actual freq: 24.000 MHz)

Start	End	Period (ns)	Max Freq	Frequency	Violation
\Sync_1:genblk1[0]:INST\:synccell.syncq	\PWM_1:PWMUDB:runmode_enable\:macrocell.mc_d	7.843	127.508 MHz	24.000 MHz	

输出路径延迟

当表现输出路径延迟的特性时，必须考虑输出的去向，以了解在 STA 结果中何处可以找到数据。对于此组件，所有输出同步到组件时钟。输出可以是下列两类之一。输出到器件中的另一个组件，或输出到器件外的引脚。在第一种情况下，必须查看为上面指定的“逻辑至输入”说明显示的“寄存器至寄存器”时间（源时钟是组件时钟）。对于第二种情况，可以在 *_timing.html* STA 结果中查看“时钟至输出”时间。

组件更改

本节介绍组件与以前版本相比的主要更改。

版本	更改说明	更改/影响原因
2.0	更新了数据手册框图和配置这节中正交解码器的模块图。	与计数器组件的最新版本同时使用。
	将正交解码器组件原理图中的内部计数器组件更新到 2.0 版本。	与计数器组件的最新版本同时使用。
	删除了过期定义。	
1.50.a	向数据手册中添加了特性数据	
	对数据表进行了少量编辑和更新	
1.50	更改了 QuadDec_Start() API：删除控制寄存器的写入。	Beta5 STA 优化
	补充了 QuadDec_Sleep()/ QuadDec_Wakeup() API。	补充了API 用以支持低功耗模式。
	补充了 QuadDec_Init() API。	补充了此函数以提供用于在启动前初始化/恢复组件的 API。
1.20	更新了“配置”对话框。 删除了计数器大小低于 32 时编译之后的 *QuadDec_INT.c* 文件。 删除了计数器大小 = 32 位时*QuadDec_INT.c* 文件中的检查条件。	

© 赛普拉斯半导体公司，2012。此处所包含的信息可能会随时更改，恕不另行通知。除赛普拉斯产品的内嵌电路之外，赛普拉斯半导体公司不对任何其他电路的使用承担任何责任。也不根据专利或其他权利以明示或暗示的方式授予任何许可。除非与赛普拉斯签订明确的书面协议，否则赛普拉斯产品不保证能够用于或适用于医疗、生命支持、救生、关键控制或安全应用领域。此外，对于可能发生运转异常和故障并对用户造成严重伤害的生命支持系统，赛普拉斯不授权将其产品用作此类系统的关键组件。若将赛普拉斯产品用于生命支持系统中，则表示制造商将承担因此类使用而招致的所有风险，并确保赛普拉斯免于因此而受到任何指控。

PSoC® 是赛普拉斯半导体公司的注册商标；PSoC Creator™ 和 Programmable System-on-Chip™ 是赛普拉斯半导体公司的商标。此处引用的所有其他商标或注册商标归其各自所有者所有。

所有源代码（软件和/或固件）均归赛普拉斯半导体公司（赛普拉斯）所有，并受全球专利法规（美国和美国以外的专利法规）、美国版权法以及国际条约规定的保护和约束。赛普拉斯据此向获许可者授予适用于个人的、非独占性、不可转让的许可，用以复制、使用、修改、创建赛普拉斯源代码的派生作品、编译赛普拉斯源代码和派生作品，并且其目的只能是创建自定义软件和/或固件，以支持获许可者仅将其获得的产品依照适用协议规定的方式与赛普拉斯集成电路配合使用。除上述指定的用途之外，未经赛普拉斯的明确书面许可，不得对此类源代码进行任何复制、修改、转换、编译或演示。

免责声明：赛普拉斯不针对此材料提供任何类型的明示或暗示保证，包括（但不仅限于）针对特定用途的适销性和适用性的暗示保证。赛普拉斯保留在不做出通知的情况下对此处所述材料进行更改的权利。赛普拉斯不对此处所述之任何产品或电路的应用或使用承担任何责任。对于可能发生运转异常和故障并对用户造成严重伤害的生命支持系统，赛普拉斯不授权将其产品用作此类系统的关键组件。若将赛普拉斯产品用于生命支持系统中，则表示制造商将承担因此类使用而招致的所有风险，并确保赛普拉斯免于因此而受到任何指控。

产品使用可能受适用的赛普拉斯软件许可协议限制。