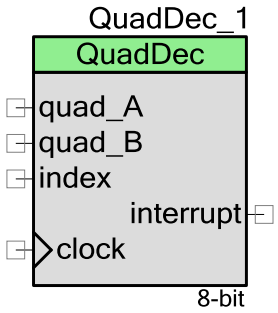
直交デコーダ (QuadDec)

2.0

特徴

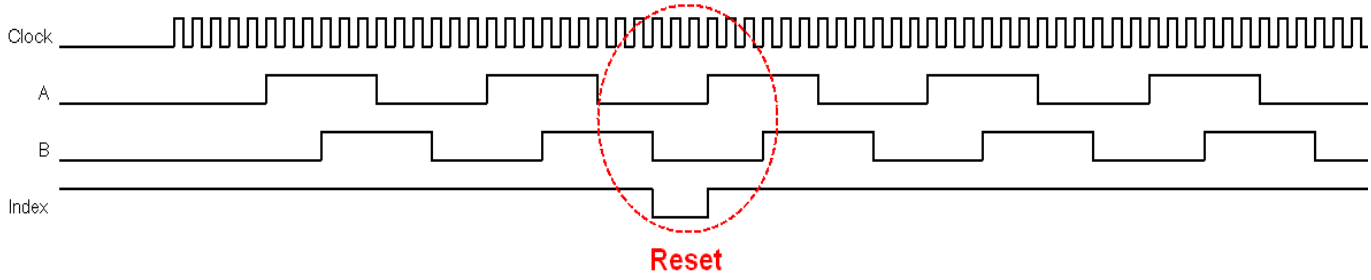


- 調整可能なカウンタ サイズ: 8、16、32 ビット
- A および B 入力での 1x、2x、または 4x 周波数のカウンタ分解能による、より正確な位置または速度の決定
- 絶対的な位置を決定するための、オプションのインデックス入力
- 入力においてシステムから発生するノイズの影響を軽減する、オプションのグリッチ フィルタリング

一般的な説明

直交デコーダ (QuadDec) コンポーネントは、デジタル信号のペアで、遷移をカウントする機能を提供します。典型的に、信号は、モーターやトラックボールにマウントされた速度/位置のフィードバック システムによって提供されます。

通常 A および B と呼ばれる信号は、グレイコード出力をもたらす、位相から 90 度外れた位置にあります。グレイ コードは、各カウントで 1 ビットだけ変わるシーケンスです。これは、グリッチを避けるために必須です。また、方向と相対位置の検出を可能にします。3 つ目のオプション信号、名前付きインデックスは、回転ごとに一度、絶対位置を確立するための基準として使用されます。



直交デコーダを使用する場合

直交デコーダは、直交エンコーダの出力をデコードするために使用されます。直交エンコーダは、物体 (例えば、マウス、トラックボール、ロボットのアクセルなど) の現在の位置、速度、方向を感知します。

直交デコーダは、モータのロータの速度、加速度、位置を正確に計測し、回転式ノブを使って、ユーザ入力を決定するためにも使用できます。

入出力接続

ここでは、直交デコーダ コンポーネントのさまざまな入出力接続について説明します。I/O リストのアスタリスク (*) は、I/O が、その I/O の説明でリストされている条件において、シンボルに隠れている可能性があることを示します。

quad_A – 入力

直交デコーダの「A」入力。

quad_B – 入力

直交デコーダの「B」入力。

index – 入力 *

この入力は、直交デコードの基準位置を検出します。インデックス入力を使用する場合、入力 A、B、インデックスがすべてゼロである場合は、カウンタもゼロにリセットされます。通常、インデックス パルスをゲートするために、その他のロジックが追加されます。インデックス ゲーティングにより、多くの可能性のある回転のうち、1 回転だけでカウンタがリセットされるようになります。1 つの例として、移動の距離制限に達した場合に、カウンタをリセットするだけのリニア アクチュエータがあります。この制限は、出力がインデックス パルスに接続されている、機械的な制限スイッチによって信号が送信されます。

この出力はデフォルトで表示されますが、**Use index input (インデックス入力の使用)** パラメータの選択を解除することで、隠すこともできます。

clock – 入力

サンプリングおよび入力のグリッチ フィルタリング用のクロック信号。グリッチ フィルタリングを使用する場合、フィルタされた出力は、続く 3 つの入力サンプルが同じ値になるまで変更されません。効果的なグリッチ フィルタリングを実行するため、サンプル クロック期間は、グリッチが予測される期間の最大時間を超えないようにします。カウンタは、A および B 入力の 1x、2x、4x 周波数の分解能で増分または減分できます。

クロック入力周波数は、A または B の最大入力周波数の 10 倍以上にします。

interrupt – 出力

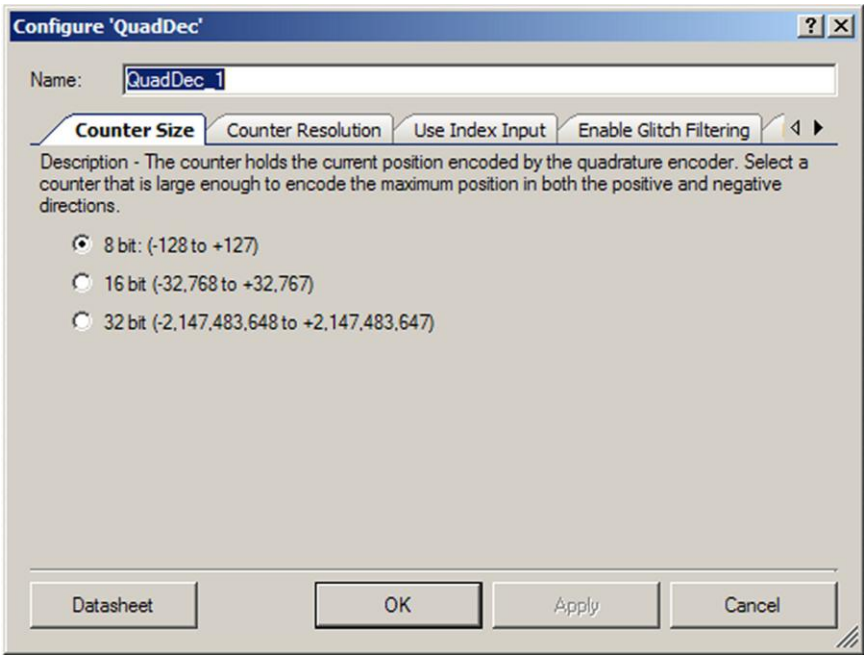
以下のいずれかのイベントでの割り込み:

- カウンタのオーバーフローおよびアンダーフロー
- インデックス入力によるカウンタ リセット (インデックスが使用されている場合)
- A および B 入力での無効な状態遷移

コンポーネント パラメータ

デザイン上に直交デコーダ コンポーネントをドラッグし、ダブルクリックして **[Configure (設定)]** ダイアログを開きます。ダイアログには、カテゴリ別のパラメータを持つ、複数のタブが含まれています。

[Counter Size (カウンタ サイズ)] タブ

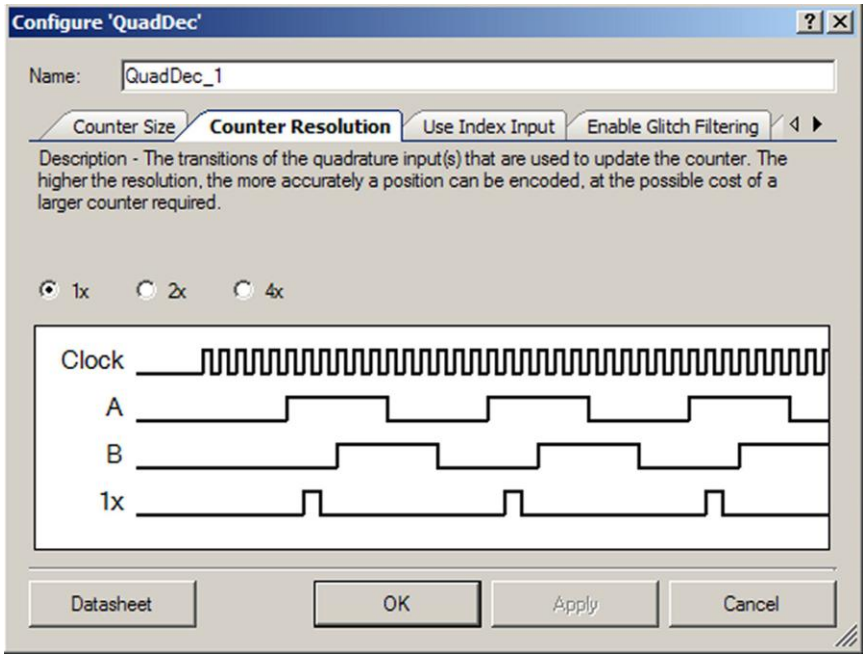


このタブは、カウンタ サイズをビットで定義するために使用されます。カウンタは、直交エンコーダによってエンコードされた現在の位置を維持します。

正と負の方向で最大位置をエンコードするのに十分なカウンタ サイズを選択してください。設定の選択肢: **8 ビット**、**16 ビット**、または **32 ビット**。

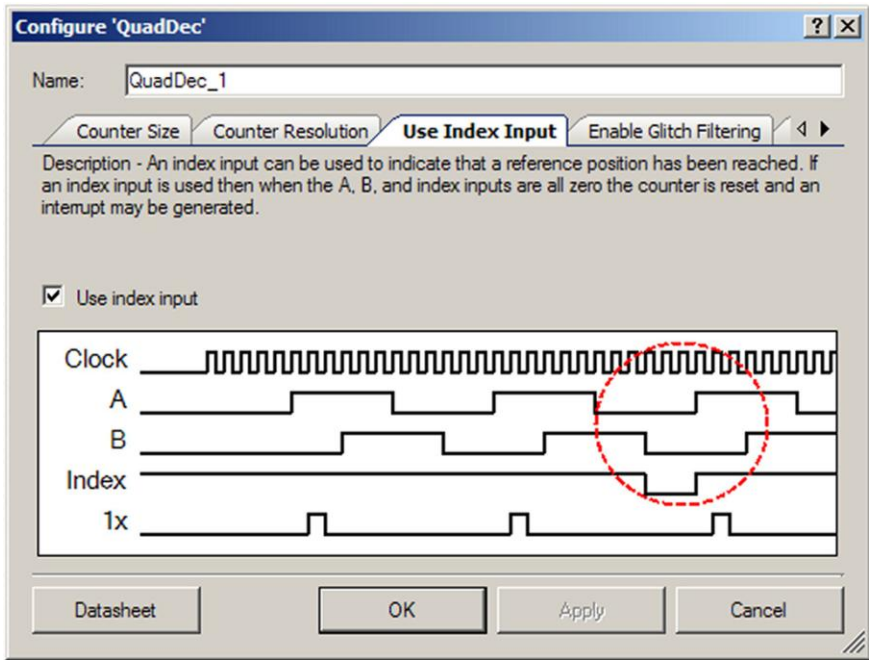
32 ビット カウンタは、ハードウェア カウンタの下位 16 ビット、ソフトウェア カウンタの上位 16 ビットを実装し、ハードウェアのリソース利用を軽減します。このターゲットでは、追加の ISR が使用されます。32 ビットのカウンタを正しく動作するよう、割り込みを有効にする必要があります。必要に応じて、ISR コードをソース ファイルに追加できます。詳しくは、割り込みコンポーネント・データシートを参照してください。

[Counter Resolution (カウンタの分解能)] タブ



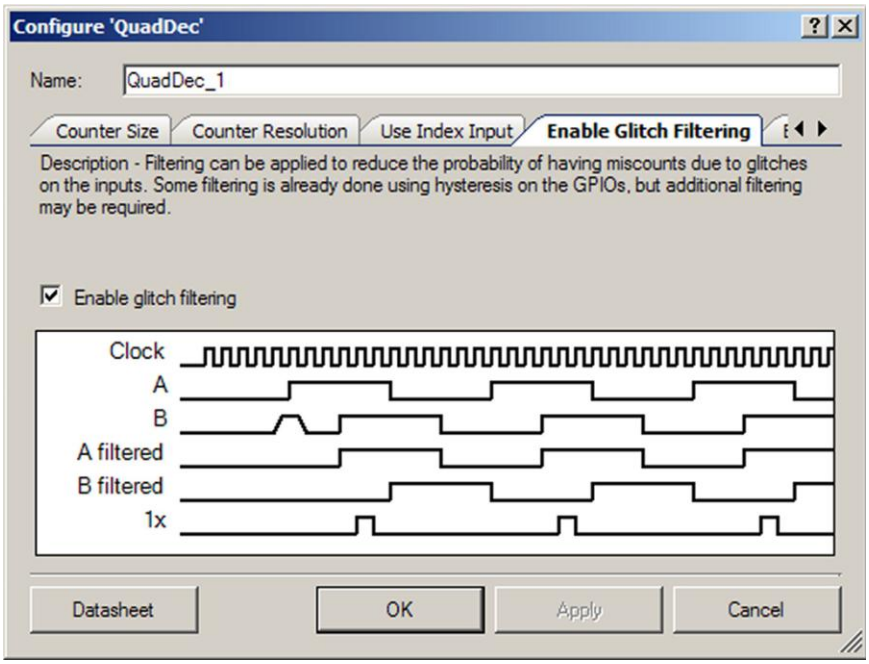
このタブには、A および B 入力の 1 周期で記録されたカウント数が含まれます。カウンタの更新に使用される入力信号の遷移を示します。コストの範囲内でのより大きなカウンタの使用では、分解能が高いほど、より結果の位置が正確になります。設定は、**1x**、**2x**、**4x** になります。

[Use Index Input (インデックス入力の使用)] タブ



このタブには、インデックス入力を有効/無効にするフィールドが含まれます。インデックス入力は、基準位置に達したことを示すために使用できます。インデックス入力が使用される場合、A、B、インデックス入力がすべてゼロになると、カウントがリセットされ、割り込みを生成できます。デフォルトでは、インデックス入力が有効になります。

[Enable Glitch Filtering (グリッチ フィルタリングの使用)] タブ



このタブには、デジタル グリッチ フィルタリングを有効または無効にするフィールドが含まれます。フィルタリングは、入力のグリッチによってのカウントミスを防ぐために適用することができます。GPIO でのヒステリシスを使用したフィルタリングがすでに適用されていますが、追加のフィルタリングが必要となる場合があります。

有効にすると、フィルタリングがすべての入力に適用されます。フィルタされた出力は、続く 3 つの入力サンプルが同じ値になるまで変更されません。効果的なフィルタリングを実施するため、サンプル クロック期間は、グリッチの発生が予測される期間の最大時間を超えないようにします。デフォルトでは、グリッチ フィルタリングが有効になります。

Clock Select（クロック選択）

直交デコーダ コンポーネントのクロッキングをするためのクロック源を接続する必要があります。ステータス レジスタの時間を記録し、割り込みを生成します。

配置

直交デコーダは、UDB アレイに配置され、すべての配置情報は、*cyfitter.h* ファイルを通して API に提供されます。

リソース

グリッチ フィルタリングを使わない 1x 分解能

リソース	リソースのタイプ					API メモリ(バイト)		ピン(外部入出力当たり)
	データパス セル	PLD	ステータス セル	Control/ Count7 セル	割り込み	フラッシュ	RAM	
8 ビット	1	6	2	1	0	566	7	2
8 ビット *	1	6	2	1	0	566	7	3
16 ビット	2	6	2	1	0	652	9	2
16 ビット *	2	6	2	1	0	652	9	3
32 ビット	2	6	2	1	1	906	14	2
32 ビット *	2	9	2	1	1	906	14	3

* インデックス入力を使用

グリッチ フィルタリングを使う 1x 分解能

リソース	リソースのタイプ					API メモリ(バイト)		ピン(外部入出力当たり)
	データパス セル	PLD	ステータス セル	Control/ Count7 セル	割り込み	フラッシュ	RAM	
8 ビット	1	7	2	1	0	566	7	2
8 ビット *	1	9	2	1	0	566	7	3
16 ビット	2	7	2	1	0	652	9	2
16 ビット *	2	9	2	1	0	652	9	3
32 ビット	2	7	2	1	1	906	14	2
32 ビット *	2	9	2	1	1	906	14	3

* インデックス入力を使用

グリッチ フィルタリングを使わない 2x 分解能

リソース	リソースのタイプ					API メモリ(バイト)		ピン(外部入出力当たり)
	データパス セル	PLD	ステータス セル	Control/ Count7 セル	割り込み	フラッシュ	RAM	
8 ビット	1	6	2	1	0	566	7	2
8 ビット *	1	7	2	1	0	566	7	3
16 ビット	2	6	2	1	0	652	9	2
16 ビット *	2	7	2	1	0	652	9	3
32 ビット	2	6	2	1	1	906	14	2
32 ビット *	2	7	2	1	1	906	14	3

* インデックス入力を使用

グリッチ フィルタリングを使う 2x 分解能

リソース	リソースのタイプ					API メモリ(バイト)		ピン(外部入出力当たり)
	データパス セル	PLD	ステータス セル	Control/ Count7 セル	割り込み	フラッシュ	RAM	
8 ビット	1	8	2	1	0	566	7	2
8 ビット *	1	9	2	1	0	566	7	3
16 ビット	2	8	2	1	0	652	9	2
16 ビット *	2	9	2	1	0	652	9	3
32 ビット	2	8	2	1	1	906	14	2
32 ビット *	2	9	2	1	1	906	14	3

* インデックス入力を使用

グリッチ フィルタリングを使わない 4x 分解能

リソース	リソースのタイプ					API メモリ（バイト）		ピン（外部入出力当たり）
	データパス セル	PLD	ステータス セル	Control/ Count7 セル	割り込み	フラッシュ	RAM	
8 ビット	1	7	2	1	0	566	7	2
8 ビット *	1	7	2	1	0	566	7	3
16 ビット	2	7	2	1	0	652	9	2
16 ビット *	2	7	2	1	0	652	9	3
32 ビット	2	7	2	1	1	906	14	2
32 ビット *	2	7	2	1	1	906	14	3

* インデックス入力を使用

グリッチ フィルタリングを使う 4x 分解能

リソース	リソースのタイプ					API メモリ（バイト）		ピン（外部入出力当たり）
	データパス セル	PLD	ステータス セル	Control/ Count7 セル	割り込み	フラッシュ	RAM	
8 ビット	1	8	2	1	0	566	7	2
8 ビット *	1	9	2	1	0	566	7	3
16 ビット	2	8	2	1	0	652	9	2
16 ビット *	2	9	2	1	0	652	9	3
32 ビット	2	8	2	1	1	906	14	2
32 ビット *	2	9	2	1	1	906	14	3

* インデックス入力を使用

アプリケーション プログラミング インタフェース

アプリケーション プログラミング インターフェース (API) ルーチンにより、ソフトウェアを使用してコンポーネントを設定できます。次の表は、各関数へのインターフェースとその説明を示しています。続くセクションでは、各関数について詳しく説明します。

デフォルトで、PSoC Creator は、インスタンス名「QuadDec_1」を、特定のデザインにおける最初のコンポーネント インスタンスに割り当てます。コンポーネントのインスタンス名称は、識別子の文法ルールに従って固有の名前に変更できます。インスタンス名は、すべてのグローバル関数名、変数名、定数名のプリフィックスになります。便宜上、以下の表では「QuadDec」というインスタンス名を使用します。

関数	説明
QuadDec_Start()	UDB とその他の関連ハードウェアを初期化します。
QuadDec_Stop()	UDB とその他の関連ハードウェアをオフにします。
QuadDec_GetCounter()	カウンタの現在値をレポートします。
QuadDec_SetCounter()	カウンタの現在値を設定します。
QuadDec_GetEvents()	イベントの現在の状態をレポートします。
QuadDec_SetInterruptMask()	イベントによる割り込みを有効または無効にします。
QuadDec_GetInterruptMask()	現在の割り込みマスク設定をレポートします。
QuadDec_Sleep()	スリープになるコンポーネントを準備します。
QuadDec_Wakeup()	ウェイクアップになるコンポーネントを準備します。
QuadDec_Init()	カスタマイザで提供されるデフォルト設定を初期化または復元します。
QuadDec_Enable()	直交デコーダをイネーブルにします。
QuadDec_SaveConfig()	現在のユーザー設定を保存します。
QuadDec_RestoreConfig()	ユーザー設定を復元します。

グローバル変数

関数	説明
QuadDec_initVar	QuadDec_initVar は直行デコーダが初期化されたかどうかを示します。変数は、0 に初期化され、最初にQuadDec _Start() が呼び出されると 1 に設定されます。これは、QuadDec_Start() ルーチンの最初の呼び出し後に再初期化しないでコンポーネントを起動します。 コンポーネントの再初期化が必要な場合は、QuadDec_Init() 関数を QuadDec_Start() または QuadDec_Enable() 関数の前に呼び出します。
QuadDec_count32SoftPart	32ビット カウンタ値の上位16 ビットは、この変数に保存されます。
QuadDec_swStatus	ステータス レジスタの値は、この変数に保存されます。

void QuadDec_Start(void)

説明: UDB とその他の関連ハードウェアを初期化します。カウンタを 0 にリセットし、すべての関連割り込みをイネーブルまたはディスエーブルにします。入力とカウントの監視を開始します。

パラメータ: なし

戻り値: なし

副作用: なし

void QuadDec_Stop(void)

説明: UDB とその他の関連ハードウェアをオフにします。

パラメータ: なし

戻り値: なし

副作用: なし

int8/16/32 QuadDec_GetCounter(void)

説明: カウンタの現在値をレポートします。

パラメータ: なし

戻り値: int8/16/32: カウンタ値。戻り値のタイプは、カウンタ サイズの設定によって割り当てられます。正の値は、反時計回りの動き (A の前に B) を示します。

副作用: なし

void QuadDec_SetCounter(int8/16/32 値)

説明: カウンタの現在値を設定します。

パラメータ: int8/16/32 value: 新しい値。パラメータのタイプは、カウンタ サイズの設定によって割り当てられます。

戻り値: なし

副作用: なし

uint8 QuadDec_GetEvents(void)

説明: イベントの現在の状態を報告します。

パラメータ: なし

戻り値: 符号なしの 8 ビット値におけるビットとしてのイベント。

ビット	説明
QuadDec_COUNTER_OVERFLOW	カウンタのオーバーフロー。
QuadDec_COUNTER_UNDERFLOW	カウンタのアンダーフロー。
QuadDec_COUNTER_RESET	インデックスが使用されている場合、インデックスによってよるカウンタがリセットされます。
QuadDec_INVALID_IN	無効な A 、B 入力状態遷移。

副作用: なし

void QuadDec_SetInterruptMask(uint8 mask)

説明: イベントによる割り込みをイネーブルまたはディスエーブルにします。32 ビット カウンタでは、オーバーフロー、アンダーフロー、リセットの割り込みは、ディスエーブルにできません。これらのビットは無視されます。

パラメータ: uint8 mask: 8 ビット値のビットをイネーブルまたはディスエーブルにします。1 は割り込みを有効にします。

ビット	説明
QuadDec_COUNTER_OVERFLOW	カウンタのオーバーフローによる割り込みをイネーブルにします。
QuadDec_COUNTER_UNDERFLOW	カウンタのアンダーフローによる割り込みをイネーブルにします。
QuadDec_COUNTER_RESET	カウンタのリセットによる割り込みをイネーブルにします。
QuadDec_INVALID_IN	無効な入力状態遷移による割り込みをイネーブルにします。

戻り値: なし

副作用: なし

uint8 QuadDec_GetInterruptMask(void)

説明: 現在の割り込みマスク設定をレポートします。

パラメータ: なし

戻り値: 8 ビット値のビットをイネーブルまたはディスエーブルにします。1 は割り込みをイネーブルにします。
32 ビット カウンタでは、常に、オーバーフローとアンダーフローのイネーブル ビットが設定されます。

ビット	説明
QuadDec_COUNTER_OVERFLOW	カウンタのオーバーフローによる割り込み。
QuadDec_COUNTER_UNDERFLOW	カウンタのアンダーフローによる割り込み。
QuadDec_COUNTER_RESET	カウンタのリセットによる割り込み。
QuadDec_INVALID_IN	無効な A、B 入力状態遷移による割り込み。

副作用: なし

void QuadDec_Sleep(void)

説明: これは、コンポーネントのスリープを準備するのに推奨されるルーチンです。QuadDec_Sleep() ルーチンは、現在のコンポーネントの状態を保存します。次に、QuadDec_Stop() 関数を呼び出し、QuadDec_SaveConfig() を呼び出してハードウェア設定を保存します。

CyPmSleep() または CyPmHibernate() 関数を呼び出す前に、QuadDec_Sleep() 関数を呼び出します。パワーマネジメント関数については、PSoC Creator *System Reference Guide (システム リファレンス ガイド)* を参照してください。

パラメータ: なし

戻り値: なし

副作用: なし

void QuadDec_Wakeup(void)

説明: これは、QuadDec_Sleep() が呼び出された際に、コンポーネントをその状態に復元するのに適したルーチンです。QuadDec_Wakeup() 関数は QuadDec_RestoreConfig() 関数を呼び出して設定を復元します。QuadDec_Sleep() 関数が呼び出される前にコンポーネントがイネーブルになっている場合は、QuadDec_Wakeup() 関数でも、コンポーネントを再度イネーブルにします。

パラメータ: なし

戻り値: なし

副作用: 最初に QuadDec_Sleep() または QuadDec_SaveConfig() 関数を呼び出さずに QuadDec_Wakeup() 関数を呼び出すと、予期せぬ動作を引き起こす場合があります。

void QuadDec_Init(void)

説明: カスタマイザの [Configure (設定)] ダイアログの設定に従って、コンポーネントを初期化または復元します。QuadDec_Start() ルーチンがこの関数を呼び出し、これがコンポーネントの動作を開始する好ましい方法であるため、QuadDec_Init() を呼び出す必要はありません。

パラメータ: なし

戻り値: なし

副作用: 全レジスタは、カスタマイザの [Configure (設定)] ダイアログの設定に従って、値が設定されます。

void QuadDec_Enable(void)

説明: ハードウェアの使用を開始し、コンポーネントの動作を開始します。QuadDec_Start() ルーチンが この関数を呼び出し、これがコンポーネントの動作を開始する好ましい方法であるため、QuadDec_Enable() を呼び出す必要はありません。

パラメータ: なし

戻り値: なし

副作用: なし

void QuadDec_SaveConfig(void)

説明: この関数は、コンポーネントの設定と保持されないレジスタを保存します。この関数は、[Configure] (設定) ダイアログで定義されている、または該当する API で変更される、現在のコンポーネント パラメータ値も保存します。この関数は、QuadDec_Sleep() 関数によって呼び出されます。

パラメータ: なし

戻り値: なし

副作用: なし

void QuadDec_RestoreConfig(void)

説明: この関数は、コンポーネントの設定とノンリテンション レジスタを復元します。また、この関数はコンポーネントのパラメータ値を QuadDec_Sleep() 関数を呼び出す前の状態に復元します。

パラメータ: なし

戻り値: なし

副作用: 最初に QuadDec_Sleep() または QuadDec_SaveConfig() 関数を呼び出さずにこの関数を呼び出すと、予期せぬ動作を引き起こす場合があります。

ファームウェア ソースコードのサンプル

PSoC Creator は、[Find Example Project] ダイアログに数多くのサンプル プロジェクトを提供しており、そこには回路図およびコード例が含まれています。コンポーネント固有の例を見るには、[Component Catalog] または回路図に置いたコンポーネント インスタンスからダイアログを開きます。一般例については、[Start Page] または **[File（ファイル）]** メニューからダイアログを開きます。必要に応じてダイアログにある **Filter Options** を使用し、選択できるプロジェクトのリストを絞り込みます。

詳しくは、PSoC Creator ヘルプの「Find Example Project（サンプルプロジェクトを検索）」を参照してください。

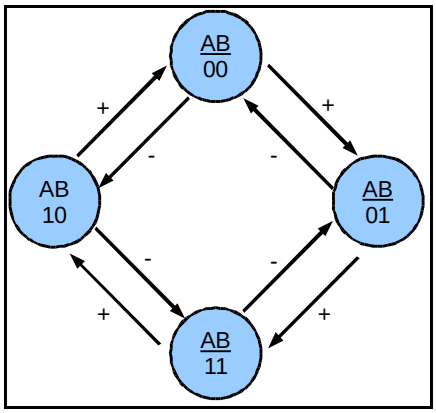
機能の説明

デフォルト設定

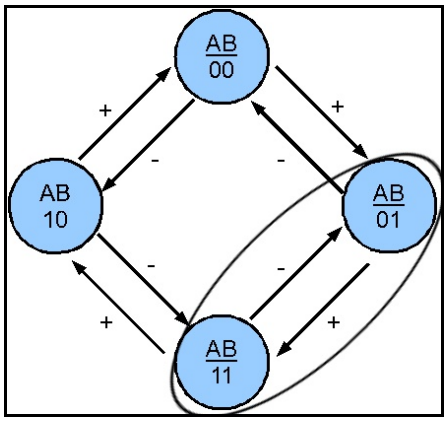
直交デコーダのデフォルト設定は、1x 分解能を持つ 8 ビットのアップおよびダウン カウンタで、インデックスが有効で、グリッチ フィルタリングが有効になっています。

状態の遷移

通常、直交位相信号は、ステートマシンとアップ/ダウン カウンタでデコードされます。従来のデコーダは、A および B 入力のすべての値に該当する、4 つの状態を持ちます。状態の遷移図を以下に示します (同じ状態の遷移は表示されていません)。「+」および「-」のついた状態の遷移は、直交位相カウンタでの増分および減分動作を示しています。



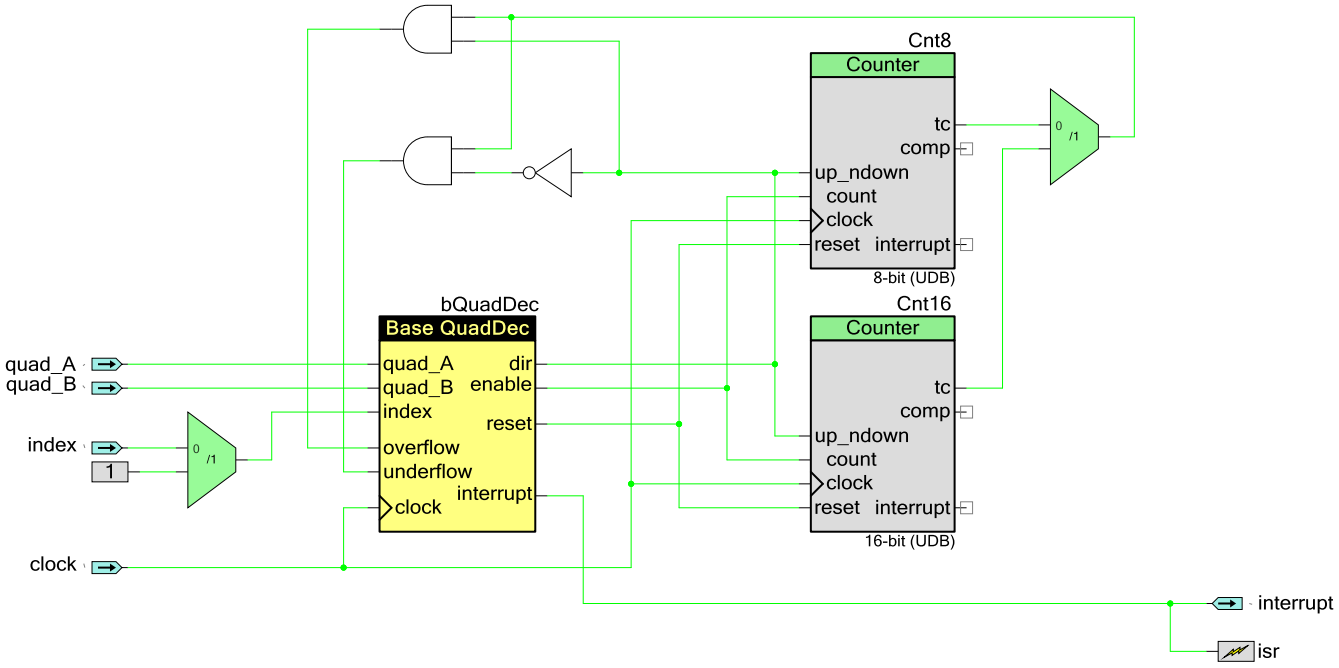
直交位相信号の各フルサイクルでは、直交位相カウンタが、4 つのカウントによって変更されます。より低い分解能カウンタは、状態遷移のサブセットのみでアップ/ダウン動作を実装することで使用可能です。4 分の 1 の分解能デコーダを以下に示します。



すべての入力は、装置で内部的に派生されるクロック信号を使ってサンプリングされます。

ブロックダイアグラムと設定

直交デコーダは、ブロックの UDB 構成としてのみ利用可能です。API はこのマニュアルの前で説明されています。レジスタについては、次のセクションのコンポーネントの総合的実装の定義の中で説明されています。



レジスタ

ステータス

ビット	7	6	5	4	3	2	1	0
値	予約済み				無効	リセット	アンダーフロー	オーバーフロー

状態レジスタは読み取り専用です。直交デコーダで定義されるさまざまなステータス ビットを含みます。このレジスタの値は、QuadDec_GetEvents() 関数で取得できます。割り込み出力信号は、状態レジスタ内でマスクされたビット フィールドの論理和 から生成されます。

QuadDec_SetInterruptMask() 関数を使って、マスクを設定できます。割り込みを受け取ると、QuadDec_GetEvents() 関数を使用して状態レジスタを読み取って、割り込みソースを取得できます。状態レジスタは透過的であるため、QuadDec_GetEvents() 関数は、状態レジスタのビットをクリアしません。状態レジスタでのすべての動作は、ビルド時に状態レジスタ内でビット フィールドが移動することがあるため、ビットフィールドの次の定義を使用する必要があります。

状態レジスタで定義されているビット フィールド マスクは複数あります。どのビット フィールドでも、割り込みソースとして含めることができます。すべてのビット フィールドは、状態レジスタのスティッキー ビットとして構成されます。定義は、以下のように生成されたヘッダー (.h) ファイルで利用できます。

- **QuadDec_COUNTER_OVERFLOW** – 状態レジスタ ビット「カウンタ オーバーフロー」のビットマスクとして定義されます。
- **QuadDec_COUNTER_UNDERFLOW** – 状態レジスタ ビット「カウンタ アンダーフロー」のビットマスクとして定義されます。
- **QuadDec_RESET** – 状態レジスタ ビット「インデックスによるリセット 」のビットマスクとして定義されます。
- **QuadDec_INVALID_IN** – 状態レジスタ ビット「A および B 入力での無効な状態遷移」のビットマスクとして定義されます。

DC 電気的特性と AC 電気的特性

以下の値は、期待される性能を示しており、初期特性データを基にしています。

「公称配線での最大」タイミング特性

パラメータ	説明	設定	Min	Typ	Max	単位
f_{CLOCK}	コンポーネント クロック周波数	構成1 [1]			34	MHz

[1]設定 1 のオプション:
CounterResolution: 1

パラメータ	説明	設定	Min	Typ	Max	単位
		構成2 [2]			29	MHz
		構成3 [3]			16	MHz
t_{CLOCKH}	入力クロック HIGH 時間 [4]	該当なし		0.5		$1/f_{CLOCK}$
t_{CLOCKL}	入力クロックLOW時 **Error! Bookmark not defined.**	該当なし		0.5		$1/f_{CLOCK}$
Inputs (入力)						
t_{PD_ps}	入力パス遅延、同期ピン [5]	1			STA [6]	ns
t_{PD_ps}	入力パス遅延、同期ピン [7]	2			8.5	ns
t_{PD_IE}	コンポーネントクロックへの入力パス遅延（エッジセンシティブ入力）	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}	入力パス遅延への同期出力 (ルート)	1,2,3,4			STA **Error! Bookmark not defined.**	ns
t_{I_clk}	clockXとクロックのアライメント	1,2,3,4	0		1	t_{CY_clock}
t_{IH}	入力 HIGH 時間	1,2	t_{CY_clock}			ns
t_{IL}	入力 LOW 時間	1,2	t_{CY_clock}			ns
t_{PD_IE}	コンポーネントクロックへの入力パス遅延（エッジセンシティブ入力）	3,4	$t_{SYNC} + t_{PD_si}$		$t_{SYNC} + t_{PD_si} + t_{I_clk}$	ns

CounterSize: 8
UsingGlitchFiltering: false
UsingIndexInput: true

[2]設定 2 のオプション:
CounterResolution: 1
CounterSize: 8
UsingGlitchFiltering: false
UsingIndexInput : true

[3]設定 3 のオプション:
CounterResolution: 2
CounterSize: 16
UsingGlitchFiltering: true
UsingIndexInput: true

[4] $t_{CY_clock} = 1/f_{clock}$ - 1 クロック周期の周期時間

[5] t_{PD_ps} は、静的タイミング解析 (STA) 結果内にあります (後述)。ここに記載されている数字は、多くの入力における STA 分析を基にした公称値です。

[6] t_{PD_ps} および t_{PD_si} はルート パスの遅延です。ルーティングは動的なためこれらの値は変化することがあり、最大コンポーネントクロックと同期クロック周波数に直接の影響を及ぼします。静的タイミング分析結果に値がある必要があります。

[7] t_{PD_ps} 構成 2 で、デバイスのピン毎に定義された固定値。ここに記載されている数字は、デバイス上で使用できるすべてのピンの公称値です。

パラメータ	説明	設定	Min	Typ	Max	単位
t_{IH}	入力 HIGH 時間	1,2,3,4	t_{CY_clock}			ns
t_{IL}	入力 LOW 時間	1,2,3,4	t_{CY_clock}			ns
f_{AB}	コンポーネント A および B 周波数	該当なし			f_{CLOCK}/10	MHz
t_{IND}	インデックス信号幅	グリッチ フィルタリングなし	$2 \times t_{CY_clock} + 5$			ns
		グリッチ フィルタリングあり	$3 \times t_{CY_clock} + 5$			ns
t_{RD}	時間をリセットするインデックス入力 LOW	グリッチ フィルタリングなし		2		t_{CY_clock}
		グリッチ フィルタリングあり		5		t_{CY_clock}
T_{GL}	グリッチの発生が予測される時間	グリッチ フィルタリングあり			3	t_{CY_clock}
t_{CD}	遅延時間、クロックの立ち上がりエッジからカウントの有効まで	該当なし		2		t_{CY_clock}
t_E	エンコーダ パルス幅 (LOW または HIGH)	該当なし	4			t_{CY_clock}
t_E	エンコーダ状態期間	該当なし	2			t_{CY_clock}
t_E	エンコーダ期間の幅	該当なし	10			t_{CY_clock}

「すべての配線での最大」タイミング特性

パラメータ	説明	設定	Min	Typ	Max [1]	単位
f_{CLOCK}	コンポーネント クロック周波数	構成1 [2]			17	MHz
		構成2 [3]			14	MHz
		構成3 [4]			8	MHz
t_{CLOCKH}	入力クロック HIGH 時間 [5]	該当なし		0.5		$1/f_{CLOCK}$
t_{CLOCKL}	入力クロックLOW時 Error! Bookmark not defined.	該当なし		0.5		$1/f_{CLOCK}$
Inputs (入力)						
t_{PD_ps}	入力パス遅延、同期ピン [6]	1			STA [7]	ns
t_{PD_ps}	入力パス遅延、同期ピン [8]	2			8.5	ns
t_{PD_IE}	コンポーネント クロックへの入力パス遅延（エッジセンシティブ入力）	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}	入力パス遅延への同期出力 (ルート)	1,2,3,4			STA Error! Bookmark not defined.	ns

[1]「すべてのルーティング」の最大値は、<公称値>/2 で最近似の整数に切り上げ/切り下げられます。この値により、このコンポーネント周波数、またはそれ以下で実行される場合に、ユーザがタイミングを合わせる必要がなくなります。

[2]設定 1 のオプション:
CounterResolution: 1
CounterSize: 8
UsingGlitchFiltering: false
UsingIndexInput: true

[3]設定 2 のオプション:
CounterResolution: 1
CounterSize: 8
UsingGlitchFiltering: false
UsingIndexInput : true

[4]設定 3 のオプション:
CounterResolution: 2
CounterSize: 16
UsingGlitchFiltering: true
UsingIndexInput: true

[5] $t_{CY_clock} = 1/f_{clock}$ - 1 クロック周期の周期時間

[6] t_{PD_ps} は、後述する静的タイミングの結果にあります。ここに記載されている数字は、多くの入力における STA 分析を基にした公称値です。

[7] t_{PD_ps} および t_{PD_si} はルート パスの遅延です。ルーティングは動的なためこれらの値は変化することがあり、最大コンポーネントクロックと同期クロック周波数に直接の影響を及ぼします。静的タイミング分析結果に値がある必要があります。

[8] t_{PD_ps} 構成 2 で、デバイスのピン毎に定義された固定値。ここに記載されている数字は、デバイス上で使用できるすべてのピンの公称値です。

パラメータ	説明	設定	Min	Typ	Max [1]	単位
t_{I_clk}	clockXとクロックのアライメント	1,2,3,4	0		1	t_{CY_clock}
t_{IH}	入力 HIGH 時間	1,2	t_{CY_clock}			ns
t_{IL}	入力 LOW 時間	1,2	t_{CY_clock}			ns
t_{PD_IE}	コンポーネントクロックへの入力パス遅延（エッジセンシティブ入力）	3,4	t_{SYNC} + t_{PD_si}		t_{SYNC} + t_{PD_si} + t_{I_clk}	ns
t_{IH}	入力 HIGH 時間	1,2,3,4	t_{CY_clock}			ns
t_{IL}	入力 LOW 時間	1,2,3,4	t_{CY_clock}			ns
f_{AB}	コンポーネント A および B 周波数	該当なし			f_{CLOCK}/10	MHz
t_{IND}	インデックス信号幅	グリッチ フィルタリングなし	2 × t_{CY_clock} + 5			ns
		グリッチ フィルタリングあり	3 × t_{CY_clock} + 5			ns
t_{RD}	時間をリセットするインデックス入力 LOW	グリッチ フィルタリングなし		2		t_{CY_clock}
		グリッチ フィルタリングあり		5		t_{CY_clock}
T_{GL}	グリッチの発生が予測される時間	グリッチ フィルタリングあり			3	t_{CY_clock}
t_{CD}	遅延時間、クロックの立ち上がりエッジからカウントの有効まで	該当なし		2		t_{CY_clock}
t_E	エンコーダ パルス幅 (LOW または HIGH)	該当なし	4			t_{CY_clock}
t_E	エンコーダ状態期間	該当なし	2			t_{CY_clock}
t_E	エンコーダ期間の幅	該当なし	10			t_{CY_clock}

図 1. グリッチ フィルタリングを使わないタイミング図

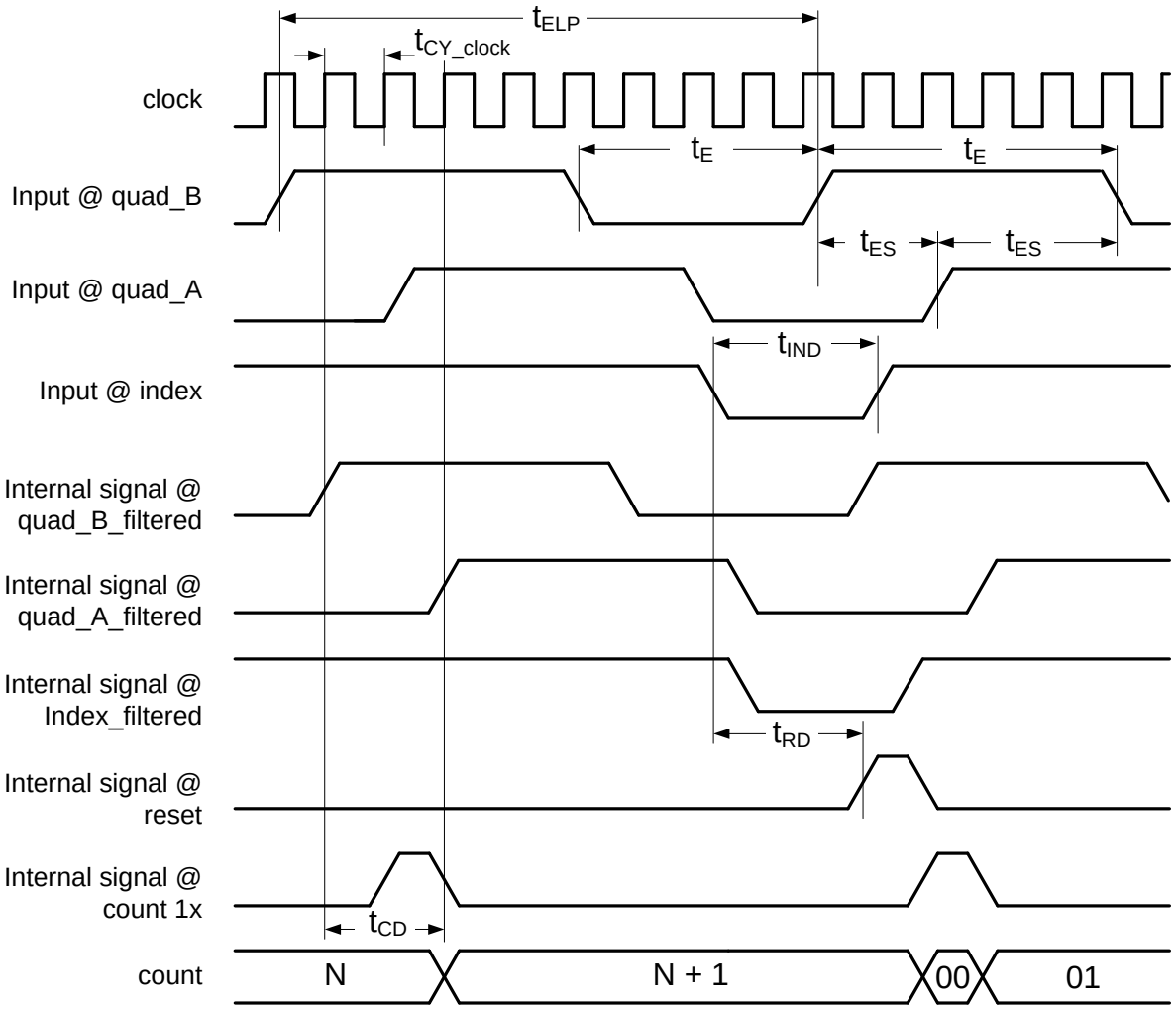
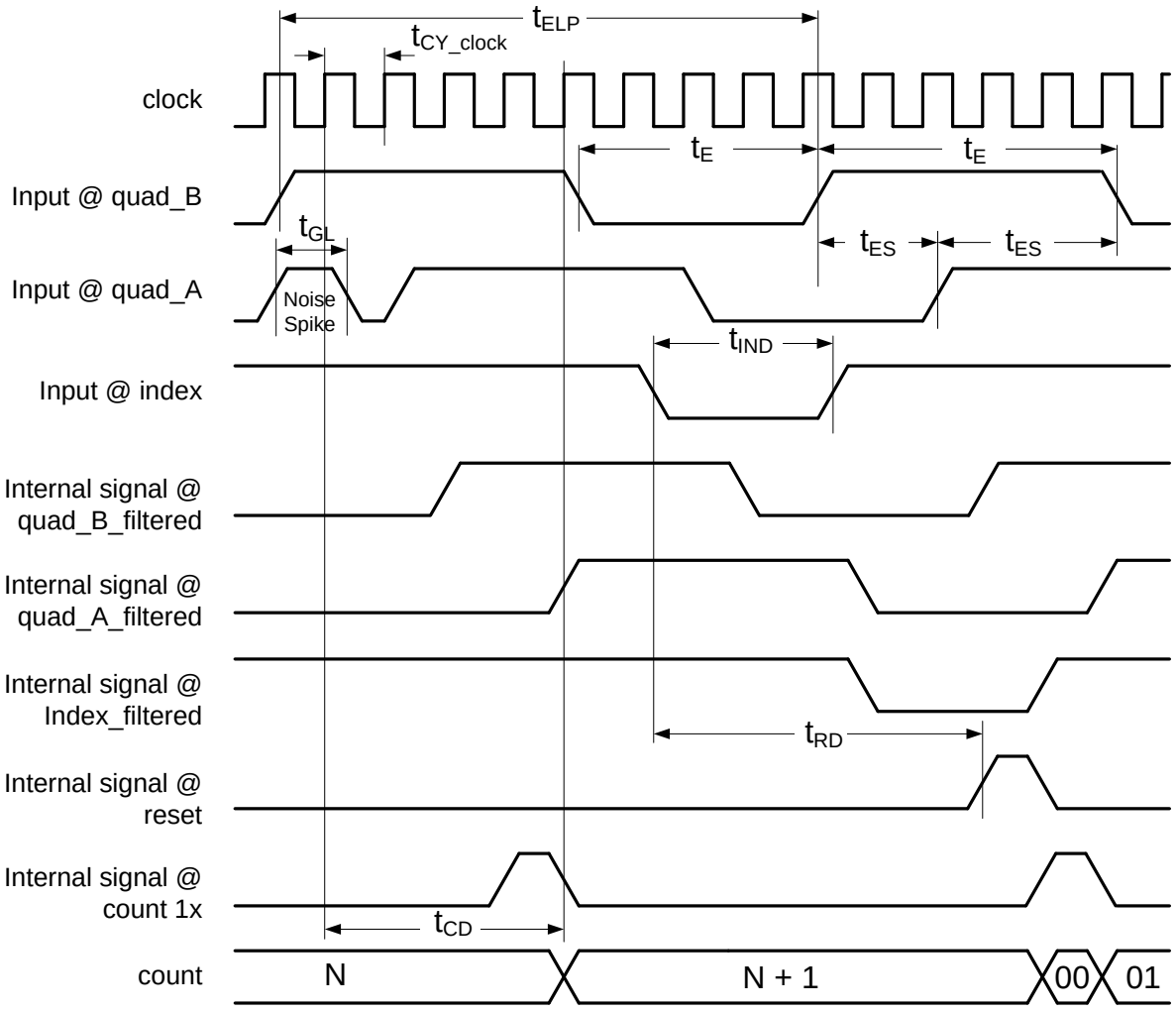


図 2. グリッチ フィルタリングを使ったタイミング図



特性データ用の STA 結果の使用方法

公称ルーティング最大値は、静的タイミング分析 (STA) を使って、複数のテスト パスから収集されます。STA 結果を用いた場合、次の手法でデザインの最大値を計算できます。

f_{CLOCK} Maximum Component Clock Frequency（最大コンポーネントクロック周波数）が、外付けクロックという名前のクロックサマリにタイミング結果として表示されます。下図は、*_timing.html* ファイルによるクロック制限の例を示しています。

-Clock Summary

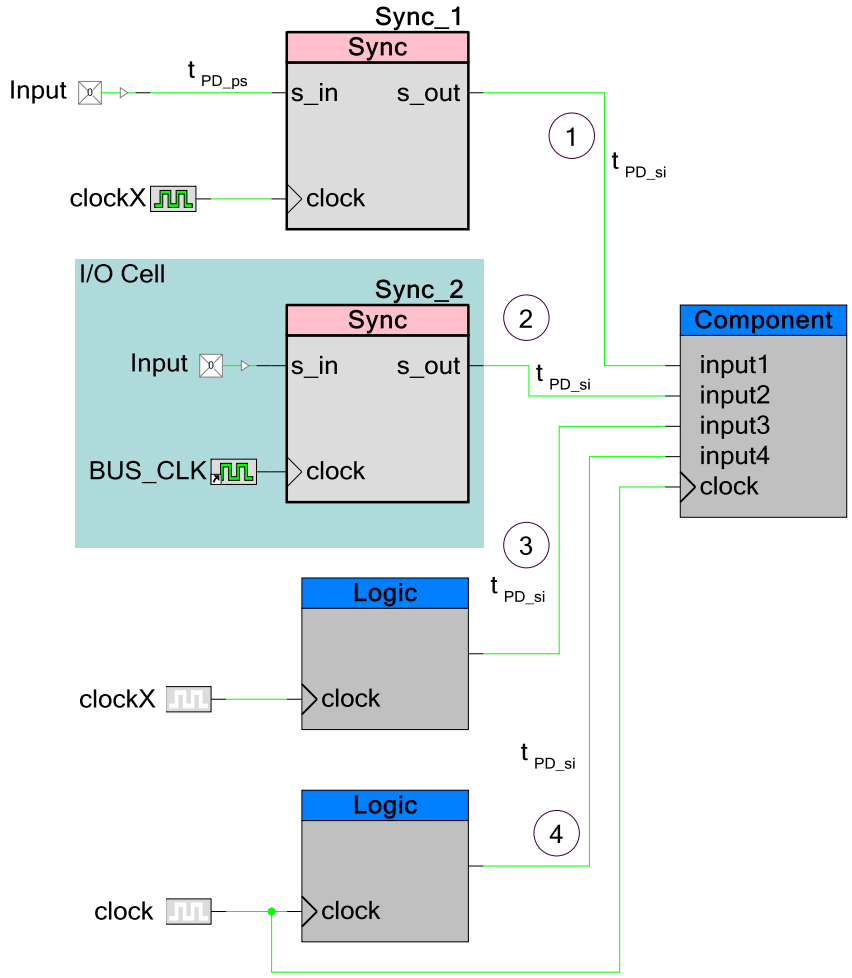
Clock	Actual Freq	Max Freq	Violation
BUS_CLK	24.000 MHz	118.683 MHz	
clock	24.000 MHz	56.967 MHz	

入力パス遅延とパルス幅

入力の機能を特性化する場合は、どのように構成しても、すべての入力は、図 3 に示されるように、4 つの可能な設定のいずれかになります。

すべての入力は同期されていなければなりません。同期のメカニズムは、コンポーネントへの入力ソースによって異なります。システムの動作を完全に解釈するには、各入力でどの入力設定を設定したか、またシステムのクロック構成を理解する必要があります。このセクションでは、Static Timing Analysis (静的タイミング分析、STA) の結果を使用して、システムの特性分析を行う方法について説明します。

図 3. コンポーネントタイミング仕様のための入力設定



構成	コンポーネント クロック	シンクロナイザ クロック(周波数)	図
1	master_clock	master_clock	図8
1	クロック	master_clock	図6
1	クロック	clockX = clock [1]	図4
1	クロック	clockX > clock	図5
1	クロック	clockX < clock	図7
2	master_clock	master_clock	図8
2	クロック	master_clock	図6
3	master_clock	master_clock	図13

[1] クロック周波数は同等ですが、立ち上がりエッジのアライメントは保証されていません。

構成	コンポーネント クロック	シンクロナイザ クロック(周波数)	図
3	クロック	master_clock	図11
3	クロック	clockX = clock [1]	図9
3	クロック	clockX > clock	図10
3	クロック	clockX < clock	図12
4	master_clock	master_clock	図13
4	クロック	クロック	図9

1. 入力は、デバイスピンによって駆動され、内部で「sync」コンポーネントと同期します。このコンポーネントは、コンポーネントが使用するクロックとは異なる内部クロックを使用します (すべての内部クロックは master_clock から派生します)。

   このような方法で構成された入力の特性を分析する際は、clockX はコンポーネントのクロックより速い、同じ、遅い場合があります。また、図 4、図 5、図 7、図 8 に示されているように、特性化パラメータを生成する、master_clock にも等しくなる場合があります。

2. この入力は、デバイスピンによって駆動され、master_clock を使用してそのピンと同期化されます。

   このような方法で構成された入力の特性を分析する際は、master_clock はコンポーネントのクロックより速いか同じになります(遅いことはありません)。これは、図 5 と 図 8 に示されているように、特性化パラメータを生成します。

図 4. 入力設定 1 および 2。同期クロック周波数 = コンポーネントクロック周波数 (clock と clockX のエッジアライメントは保証されません)

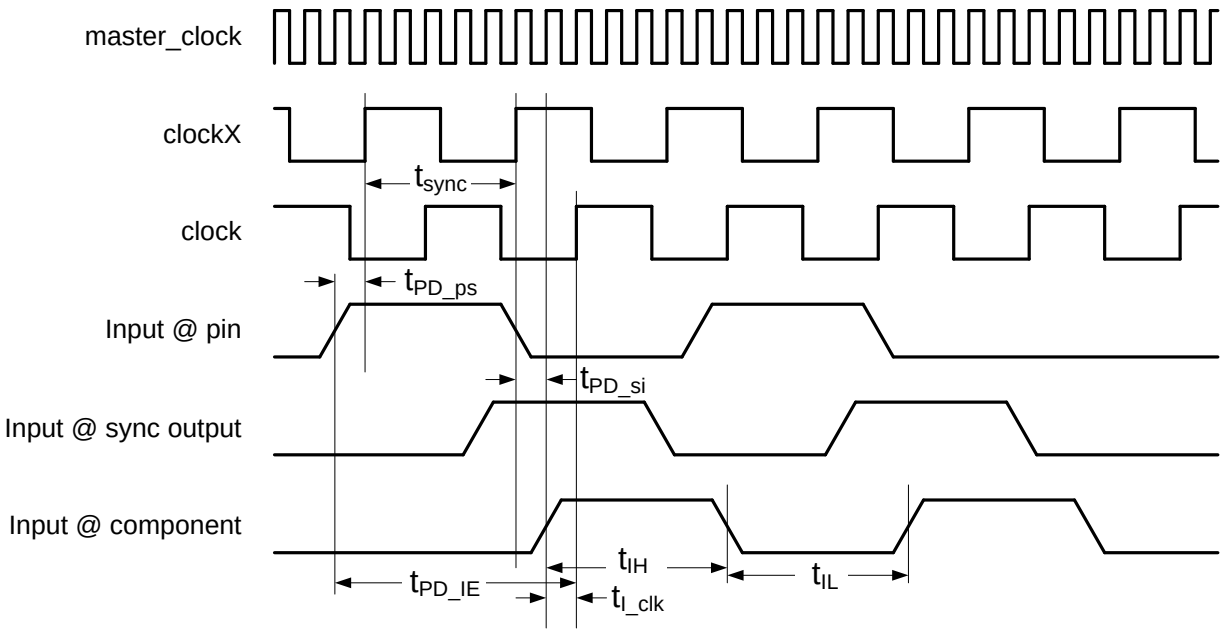


図 5. 入力設定 1 と 2。Sync。クロック周波数。> コンポーネント クロック周波数

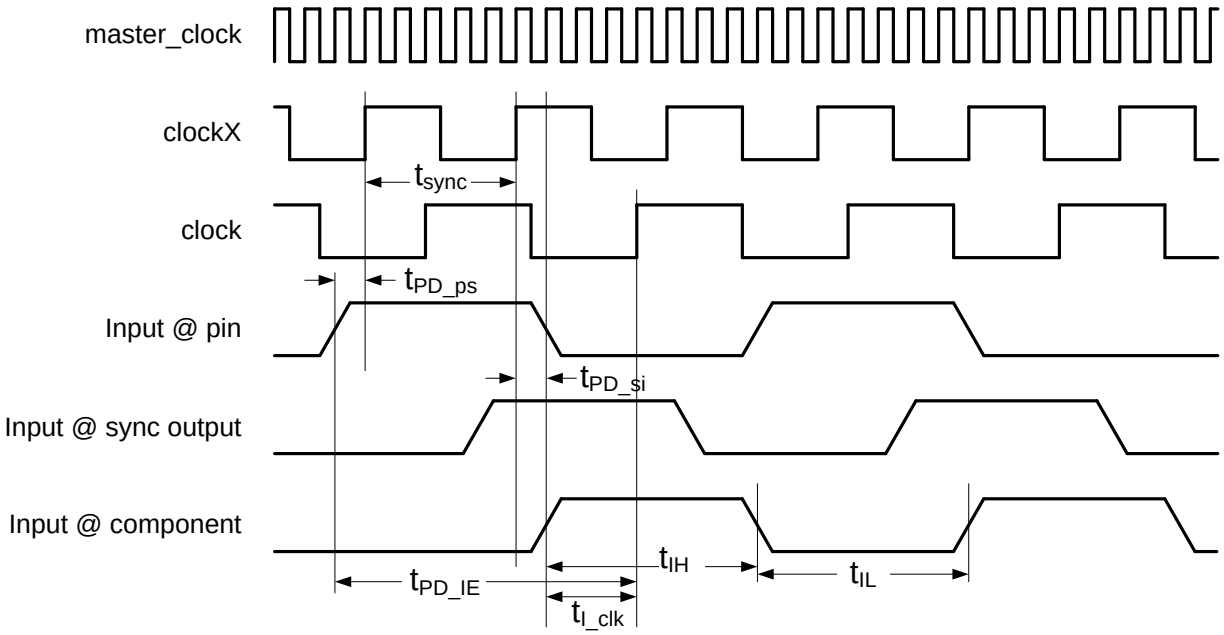


図 6. 入力設定 1 と 2。[Sync. クロック周波数。== master_clock] > コンポーネント クロック周波数

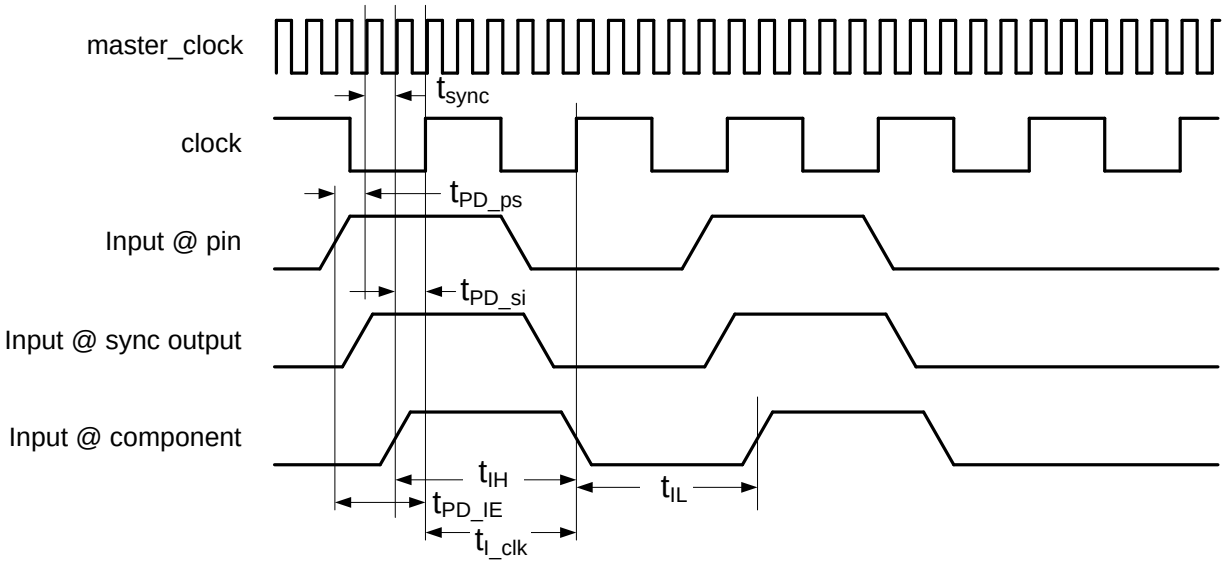


図 7. 入力設定 1。Sync。クロック周波数。< コンポーネント クロック周波数

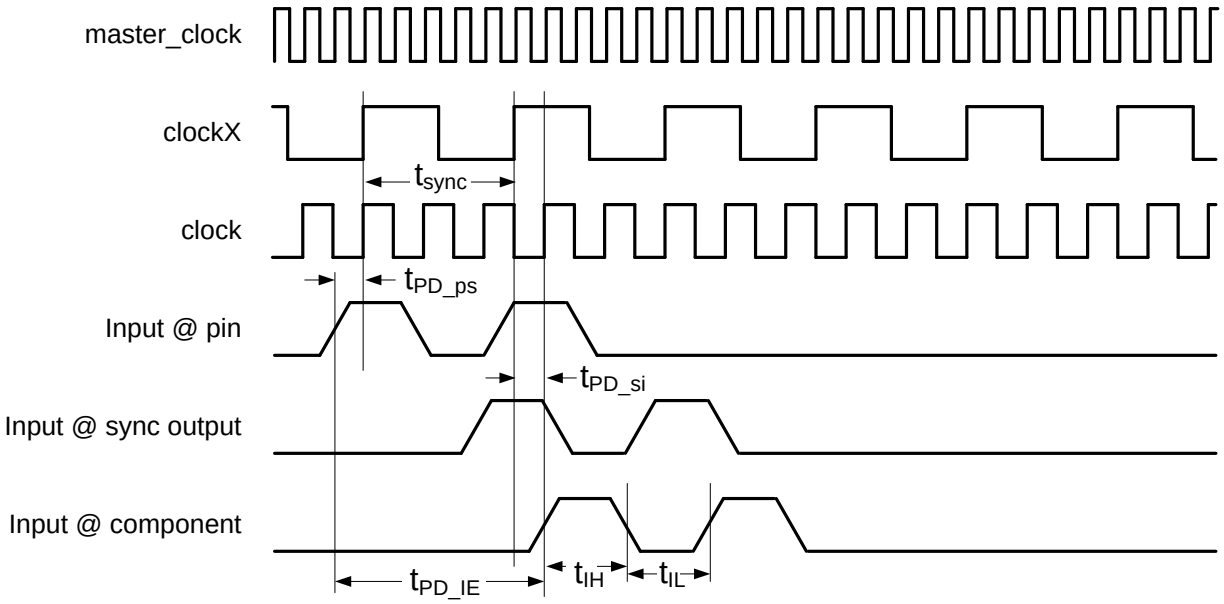
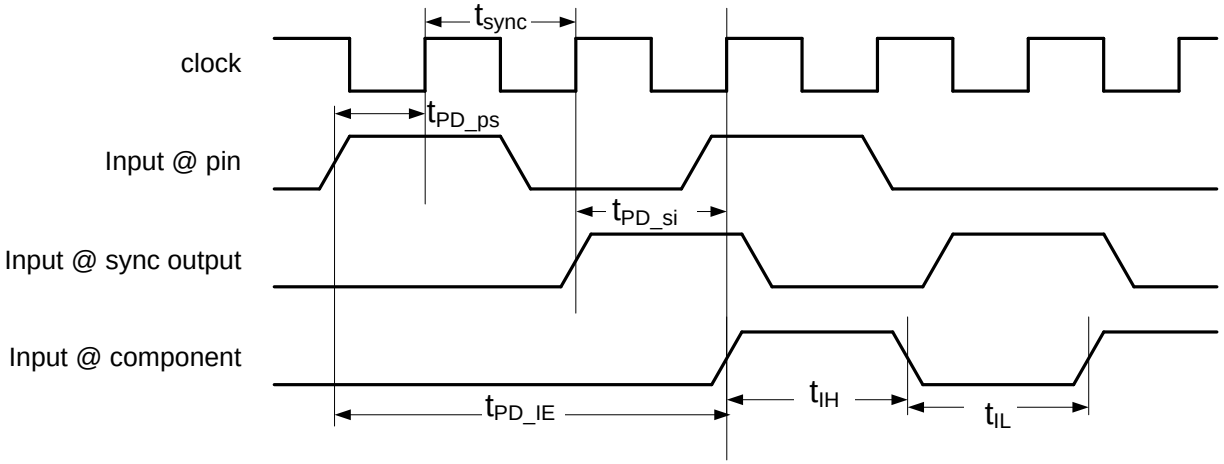


図 8. 入力設定 1 と 2。Sync。Clock = Component Clock = master_clock



3. 入力は PSoC 内部のロジックにより駆動されます。これはコンポーネントが使用するクロックとは異なるクロックをベースにして同期しています（すべての内蔵クロックは master_clock から派生しています）。

   この方法で設定された入力を特性化する場合、シンクロナイザ クロックは、図 9、図 10、図 12 に示される特性化パラメータを生成するコンポーネントのクロックより速くなるか、遅くなるか、同じになります。

4. 入力は PSoC 内部のロジックにより駆動されます。これはコンポーネントが使用するクロックと同じクロックをベースにして同期しています。

この方法で設定された入力を特性化する場合、同期クロックは、図 13 に示される特性化パラメータを生成する、コンポーネントのクロックに等しくなります。

図 9. 入力構成 3 のみ; Sync. クロック周波数。= コンポーネント クロック周波数 (clock と clockX のエッジアライメントは保証されません)

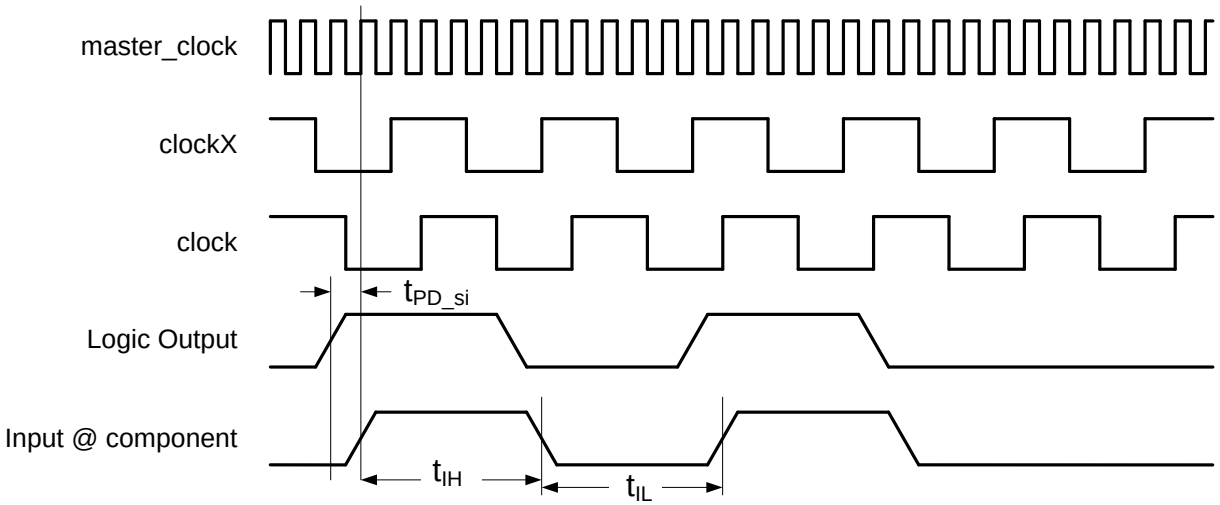


図 9 は、静的タイミング分析がクロックを解釈できる範囲を表しています。デジタルクロック領域のすべてのクロックは master_clock と同期します。但し、同じ周波数の 2 つのクロックは、立ち上がりエッジでアライメントされません。このため、静的タイミング分析ツールは、どのクロックのエッジに同期するかが特定できず、master_clock サイクルの最低値だと推定するほかありません。つまり、t_{PD_si} にシステムの master_clock のリミティング効果が含まれるようになったことを意味します。このパスの遅延が長すぎる場合は、master_clock セットアップ時間の違反が表示されます。この場合、システムの同期クロックを変更するか、master_clock を遅い周波数で実行しなければなりません。

図 10. 入力設定 3。Sync。クロック周波数。> コンポーネント クロック周波数

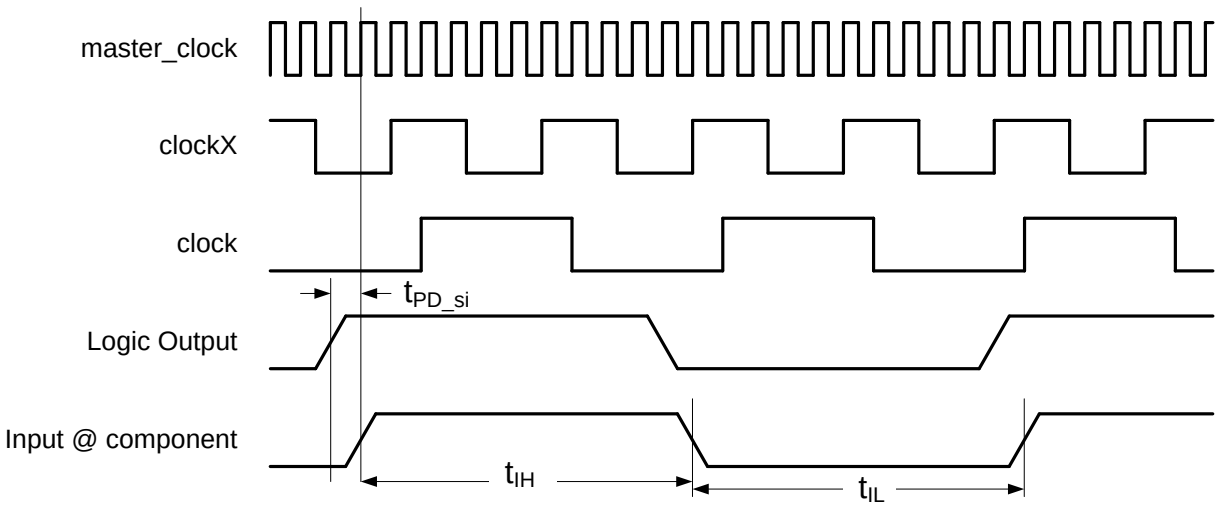


図 9 に示されているのとほぼ同じ方法で、すべてのクロックは、master_clock から生成されます。STA は、この構成で 1master_clock サイクル分、master_clock における t_{PD_si} の制限を示します。このパスの遅延が長すぎると、master_clock セットアップ時間の違反が発生します。この場合、システムの同期クロックを変更するか、master_clock を遅い周波数で実行しなければなりません。

図 11. 入力設定 3; Synchronizer Clock Frequency = master_clock > Component Clock Frequency

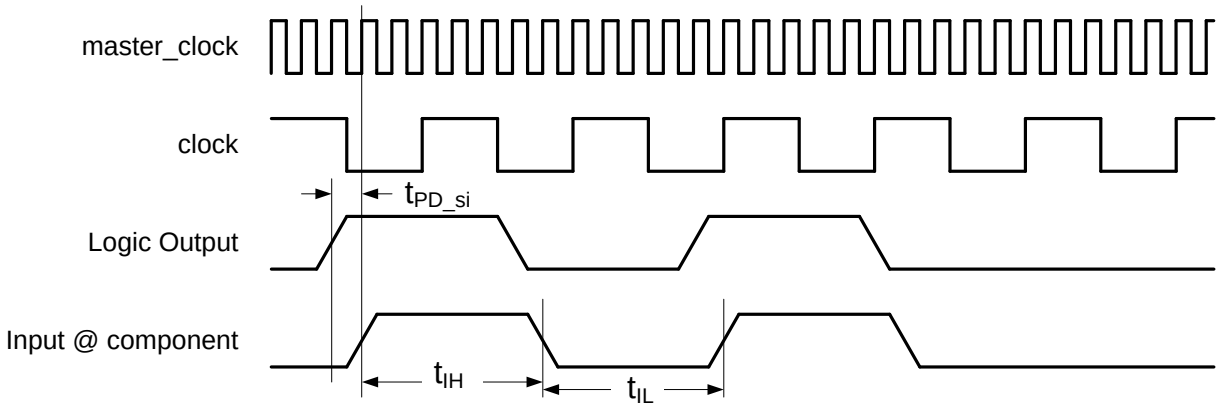


図 12. 入力設定 3; Synchronizer Clock Frequency < Component Clock Frequency

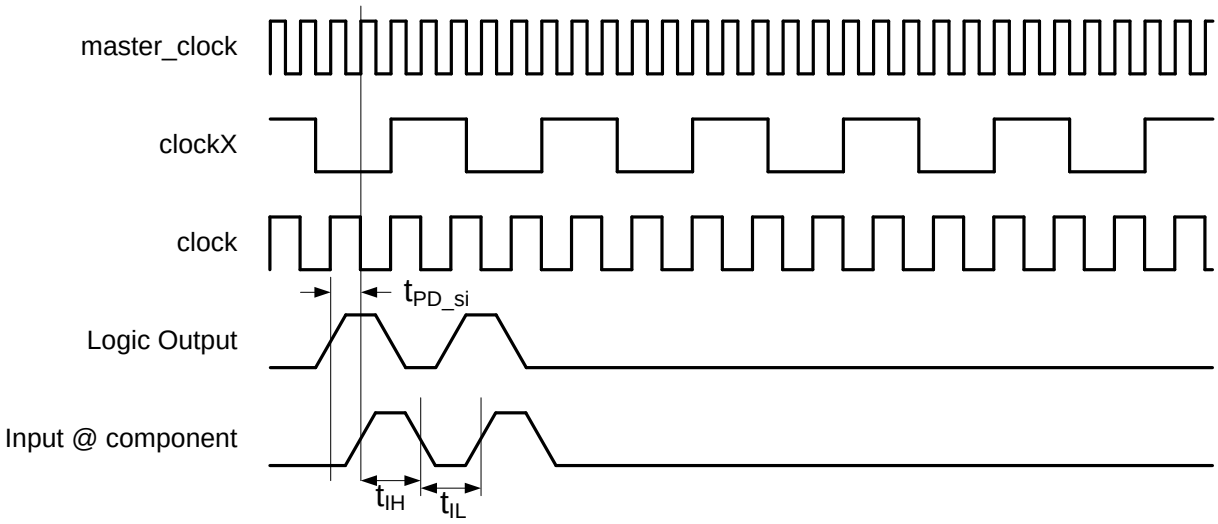
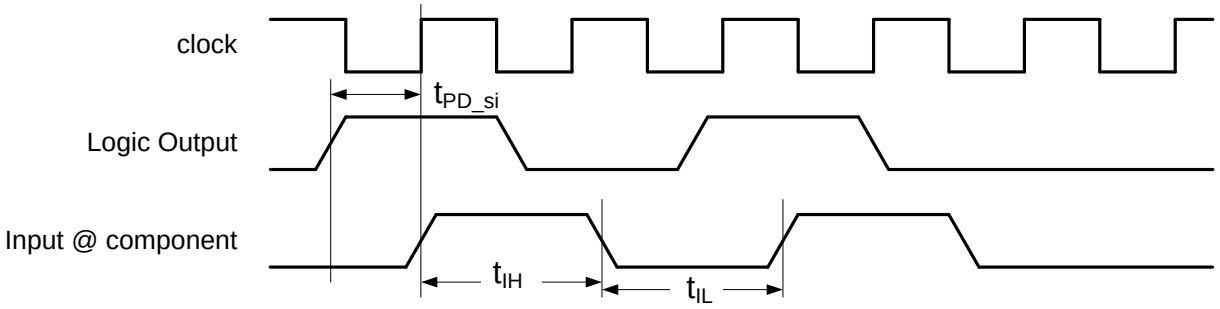


図 9 に示されているのとほぼ同じ方法で、すべてのクロックは、master_clock から派生します。STA は、この構成で 1master_clock サイクル分、master_clock における t_{PD_si} の制限を示します。このパスの遅延が長すぎると、master_clock セットアップ時間の違反が発生します。この場合、システムの同期クロックを変更するか、master_clock を遅い周波数で実行しなければなりません。

図 13. 入力設定 4 のみ; Synchronizer Clock = Component Clock



このセクションの前のすべての図で、実装を理解する上で最も重要なパラメータは、f_{CLOCK} および t_{PD_IE} です。t_{PD_IE} は、t_{PD_ps} および t_{SYNC} (設定 1 および 2 のみ)、t_{PD_si}、および t_{I_Clk} によって定義されます。非常に重要なことは、t_{PD_si} が最大コンポーネントクロック周波数を定義するということです。t_{I_Clk} は STA の結果によるものではありませんが、t_{PD_IE} が登録された場合に表示に使用されます。これは、シンクロナイザとコンポーネント クロックの間のルート後に残ったマージン。

t_{PD_ps} と t_{PD_si} は、STA 結果に含まれています。

t_{PD_ps} は、*_timing.html* ファイルで定義されている入力設定時間を参照してください。この入力のファンアウトが 1 を超える場合があるため、これらのパスの最大値を評価する必要があります。

-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} は、レジスタ間の時間に定義されています。*_timing.html* を使用するには、ネット名を知っていなければなりません。このパスのファンアウトは 1 つ以上ある可能性があり、これらのパスの最大値を評価する必要があります。

-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	

出力パス遅延

出力のパス遅延の特性分析を行う場合、STA 結果のどこでデータを見つけることができるかを知るために、出力の送信先を知らなければなりません。このコンポーネントでは、すべての出力がコンポーネントクロックに同期されています。出力は 2 つのカテゴリのうち、いずれかに該当します。出力は、デバイス内の別のコンポーネントへ送られるか、デバイス外のピンに進むかのどちらかです。前者の場合、上述のロジック・入力間の説明に記載されているレジスタ間の時間（ソースクロックはコンポーネントクロックです）を見ます。後者の場合、*_timing.html* STA 結果のクロック・出力間の時間を見ます。

コンポーネントの変更

ここでは、過去のバージョンからコンポーネントに加えられた主な変更を示します。

バージョン	変更の説明	変更の理由 / 影響
2.0	データシートのブロック図と設定ブロックダイアグラムと設定変更されています。	Counter コンポーネントの最新バージョンでの使用向けです。
	直行デコーダコンポーネント回路図の内部コンポーネントをバージョン 2.0 に更新。	Counter コンポーネントの最新バージョンでの使用向けです。
	不要定義を削除。	
1.50.a	データシートに特性データを追加	
	データシートのマイナーな編集と更新	
1.50	QuadDec_Start() API の変更: 制御レジスタへの書き込みを削除。	Beta5 STA-Based 最適化。
	QuadDec_Sleep()/ QuadDec_Wakeup() API を追加。	より低い電力モードをサポートするための API を追加。
	QuadDec_Init() API を追加。	コンポーネントを開始せずに、初期化/復元する API の提供を追加。
1.20	[Configure] ダイアログが更新されました。 カウンタ サイズは 32 以下の場合、コンパイル後 *QuadDec_INT.c* ファイルを削除。 カウンタ サイズの *QuadDec_INT.c* ファイルの状態チェックを削除。	

Copyright © 2005-2012 Cypress Semiconductor Corporation 本文書に記載される情報は、予告なく変更される場合があります。Cypress Semiconductor Corporation は、サイプレス製品に組み込まれた回路以外のいかなる回路を使用することに対しても一切の責任を負いません。特許又はその他の権限下で、ライセンスを譲渡又は暗示することもありません。サイプレス製品は、サイプレスとの書面による合意に基づくものでない限り、医療、生命維持、救命、重要な管理、又は安全の用途のために仕様することを保証するものではなく、また使用することを意図したものでもありません。さらにサイプレスは、誤動作や故障によって使用者に重大な傷害をもたらすことを合理的に予想される、生命維持システムの重要なコンポーネンツとしてサイプレス製品を使用することを許可していません。生命維持システムの用途にサイプレス製品を供することは、製造者がそのような使用におけるあらゆるリスクを負うことを意味し、その結果サイプレスはあらゆる責任を免除されることを意味します。

PSoC Designer™及び Programmable System-on-Chip™は、Cypress Semiconductor Corp.の商標、PSoC®は同社の登録商標です。本文書で言及するその他全ての商標又は登録商標は各社の所有物です。

全てのソースコード(ソフトウェア及び/又はファームウェア)は Cypress Semiconductor Corporation (以下「サイプレス」)が所有し、全世界(米国及びその他の国)の特許権保護、米国の著作権法並びに国際協定の条項により保護され、かつそれらに従います。サイプレスが本書面によるライセンシーに付与するライセンスは、個人的、非独占的かつ譲渡不能のライセンスであって、適用される契約で指定されたサイプレスの集積回路と併用されるライセンシーの製品のみをサポートするカスタムソフトウェア及び/又はカスタムファームウェアを作成する目的に限って、サイプレスのソースコードの派生著作物を複製、使用、変更、そして作成するためのライセンス、並びにサイプレスのソースコード及び派生著作物をコンパイルするためのライセンスです。上記で指定された場合を除き、サイプレスの書面による明示的な許可なくして本ソースコードを複製、変更、変換、コンパイル、又は表示することは全て禁止されます。

免責条項:サイプレスは、明示的又は黙示的を問わず、本資料に関するいかなる種類の保証も行いません。これには、商品性又は特定目的への適合性の黙示的な保証が含まれますが、これに限定されません。サイプレスは、本文書に記載される資料に対して今後予告なく変更を加える権利を留保します。サイプレスは、本文書に記載されるいかなる製品又は回路を適用又は使用したことによって生ずるいかなる責任も負いません。サイプレスは、誤動作や故障によって使用者に重大な傷害をもたらすことが合理的に予想される生命維持システムの重要なコンポーネンツとしてサイプレス製品を使用することを許可していません。生命維持システムの用途にサイプレス製品を供することは、製造者がそのような使用におけるあらゆるリスクを負うことを意味し、その結果サイプレスはあらゆる責任を免除されることを意味します。

ソフトウェアの使用は、適用されるサイプレスソフトウェアライセンス契約によって制限され、かつ制約される場合があります。