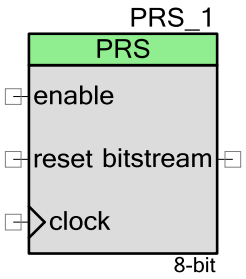
擬似乱数列(RPS)

2.0

特長



- 2~64 ビットの PRS シーケンス長
- 時間分割多重化モード
- シリアル出力ビットストリーム
- 連続またシングルステップ動作モード
- 標準またはカスタムの多項式
- 標準またはカスタムのシード値
- イネーブル入力が、その他のコンポーネントとの同期動作を提供
- 計算された疑似ランダム数はリニア フィードバック シフト レジスタ (LFSR)から直接読み出すことが可能

概要説明

The Pseudo Random Sequence (PRS) コンポーネントは LFSR を使用して、疑似ランダムビットストリームを出力する疑似欄数列を生成します。LFSR はガロア形式(モジュラー形式と呼ばれることもあります)を取り、規定された最大コード長、または期間を使用します。PRS コンポーネントは、開始後からイネーブル信号が HIGH である限り連続動作します。The PRS 数値ジェネレータは、0 以外のあらゆる有効なシード値により開始させることができます。

RPS の用途

LFSR はハードウェアに実装することができます。この方法は、直接スペクトル拡散式無線など、疑似ランダムシーケンスの非常に高速な発生が要求されるアプリケーションに便利です。

グローバル ポジショニング システム(GPS)は、LFSR を使用して高精度な相対時間オフセットを示すシーケンスを迅速に送信します。ビデオ ゲーム コンソールでもサウンド システムの一部として LSFR を使用するものがあります。

カウンタとしての使用

LFSR の状態の反復シーケンスは、分周器として使用することができ、また、非バイナリ シーケンスが使用可能な場合にはカウンタとしても使用できます。LFSR カウンタは自然バイナリ カウンタやグレイ コード カウンタより単純なフィードバック 論理であるために、より高いクロック レートで使用することができます。ただし、LFSR からすべて 0 の状態が絶対に入力されないよう注意しなければなりません。たとえば、開始時に他のシーケンスの状態にプリセットする場合などです。

入出力接続

このセクションは RPS コンポーネントのさまざまな入力や出力の接続を説明します。I/O 項目のアスタリスク (*) はその I/O が、説明に挙げられた条件において、回路シンボルに表示されない場合があることを示します。

clock – 入力 *

クロック入力は RPS を計算するための信号を定義します。この入力は API シングルステップ動作モードを選択した場合には使用できません。

reset – 入力 *

このリセット入力は RPS を同期リセットする信号を定義します。この入力はクロック制御モードを選択した場合に使用可能となります。イネーブル入力が HIGH に保持されている場合のみ RPS をリセットすることができます。

enable – 入力

RPS コンポーネントは、開始後からイネーブル信号が HIGH である限り実行されます。この入力が、その他のコンポーネントとの同期動作を提供します。

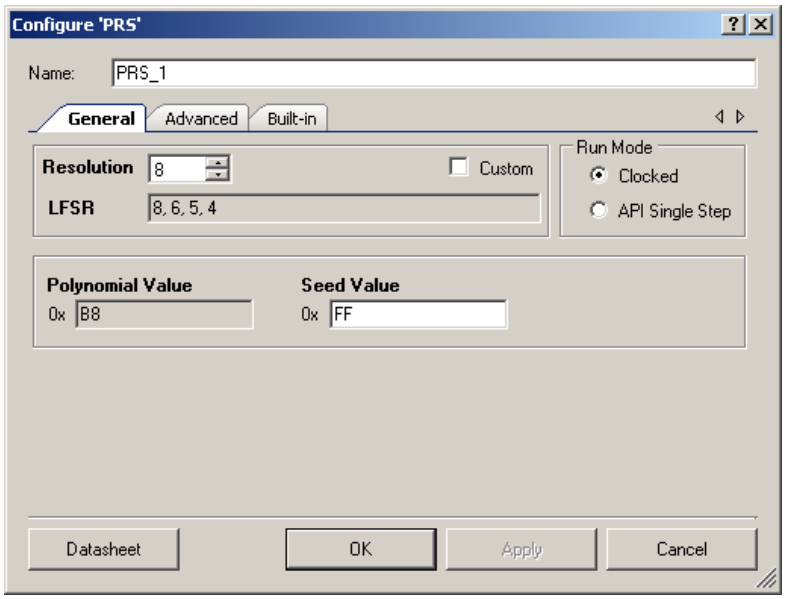
bitstream – 出力

LFSR の出力。

コンポーネント・パラメータ

PRS コンポーネントを回路図にドラッグし、ダブルクリックして **Configure** ダイアログを開きます。このダイアログには、PRS コンポーネントのセットアップをガイドする複数のタブがあります。

General タブ



分解能

PRS シーケンスの長さを定義します。このパラメータには、2～64 の値を設定できます。デフォルトは **8** です。

デフォルトでは、**Resolution** は **LFSR** 係数、および **Polynomial Value** を定義します。係数は次の表から取り込まれます。また、このパラメータは次の表で示されるとおり、最大コード長、または期間を定義します.

分解能	LFSR	Period ($2^{Resolution} - 1$)		分解能	LFSR	Period ($2^{Resolution} - 1$)
2	2, 1	3		34	34, 31, 30, 26	17179869183
3	3, 2	7		35	35, 34, 28, 27	34359738367
4	4, 3	15		36	36, 35, 29, 28	68719476735
5	5, 4, 3, 2	31		37	37, 36, 33, 31	137438953471
6	6, 5, 3, 2	63		38	38, 37, 33, 32	274877906943
7	7, 6, 5, 4	127		39	39, 38, 35, 32	549755813887
8	8, 6, 5, 4	255		40	40, 37, 36, 35	1099511627775
9	9, 8, 6, 5	511		41	41, 40, 39, 38	2199023255551
10	10, 9, 7, 6	1023		42	42, 40, 37, 35	4398046511103

分解能	LFSR	Period ($2^{Resolution} - 1$)		分解能	LFSR	Period ($2^{Resolution} - 1$)
11	11, 10, 9, 7	2047		43	43, 42, 38, 37	8796093022207
12	12, 11, 8, 6	4095		44	44, 42, 39, 38	17592186044415
13	13, 12, 10, 9	8191		45	45, 44, 42, 41	35184372088831
14	14, 13, 11, 9	16383		46	46, 40, 39, 38	70368744177663
15	15, 14, 13, 11	32767		47	47, 46, 43, 42	140737488355327
16	16, 14, 13, 11	65535		48	48, 44, 41, 39	281474976710655
17	17, 16, 15, 14	131071		49	49, 45, 44, 43	562949953421311
18	18, 17, 16, 13	262143		50	50, 48, 47, 46	1125899906842623
19	19, 18, 17, 14	524187		51	51, 50, 48, 45	2251799813685247
20	20, 19, 16, 14	1048575		52	52, 51, 49, 46	4503599627370495
21	21, 20, 19, 16	2097151		53	53, 52, 51, 47	9007199254740991
22	22, 19, 18, 17	4194303		54	54, 51, 48, 46	18014398509481983
23	23, 22, 20, 18	8388607		55	55, 54, 53, 49	36028797018963967
24	24, 23, 21, 20	16777215		56	56, 54, 52, 49	72057594037927935
25	25, 24, 23, 22	33554431		57	57, 55, 54, 52	144115188075855871
26	26, 25, 24, 20	67108863		58	58, 57, 53, 52	288230376151711743
27	27, 26, 25, 22	134217727		59	59, 57, 55, 52	576460752303423487
28	28, 27, 24, 22	268435455		60	60, 58, 56, 55	1152921504606846975
29	29, 28, 27, 25	536870911		61	61, 60, 59, 56	2305843009213693951
30	30, 29, 26, 24	1073741823		62	62, 59, 57, 56	4611686018427387903
31	31, 30, 29, 28	2147483647		63	63, 62, 59, 58	9223372036854775807
32	32, 30, 26, 25	4294967295		64	64, 63, 61, 60	18446744073709551615
33	33, 32, 29, 27	8589934591				

LFSR 係数を手動で設定する方法:

Resolution を定義します。

Custom チェック ボックスにチェックを付けます。

コンマで区切った係数を **LFSR** テキスト ボックスに入力し[**Enter**]を押します。多項式値は自動的に再計算されます。

多項式値は 16 進値で表示されます。

注 LFSR 係数値を **Resolution** 値より大きくすることはできません。

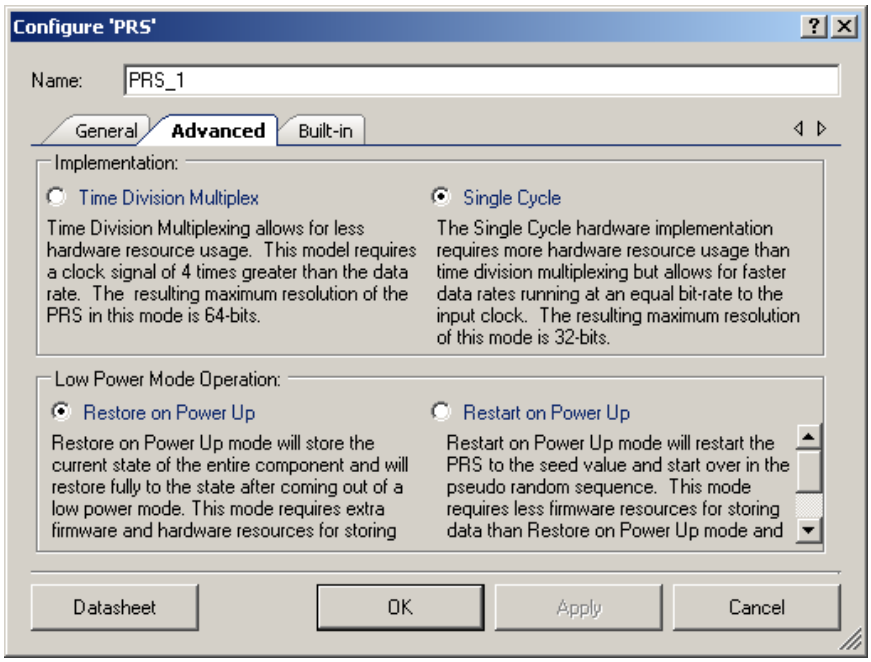
デフォルトでは、シード値は可能な限りの最大値である($2^{Resolution} - 1$) に設定されます。この値は 0 以外の任意の数値に変更することができます。シード値は 16 進値で表示されます。

注　Resolution を変更すると **Seed Value** がデフォルト値にリセットされます。

Run Mode (動作モード)

このパラメータは、コンポーネントの動作を連続、または、シングル ステップ動作に定義します。**Clocked**（デフォルト）または **API Single Step** を選択することができます。PRS 値を連続的に読み込む場合または1つの値を読み込む必要がある場合には、クロックを停止させるか、クロック制御モードでイネーブルを Low に設定しなければなりません。

Advanced タブ



PRS の **Advanced** タブには次の設定が含まれます:

Implementation (実装)

PRS コンポーネントの実装を定義します。時間多重あり、または、時間多重なし(**Single Cycle**) となります。デフォルトは **Single Cycle** です。

ローパワー モード動作

ローパワー モード後の PRS 挙動を定義します。デフォルトは **Restore on Power Up** です。

ローカルパラメータ（API での使用）

これらのパラメータは、API によって使用され、GUI には表示されません。

- **PolyValueLower(uint32)** – 16 進値による多項式値の下位半分が含まれます。デフォルトの分解能は 8 であるため、デフォルトは 0xB8h (**LFSR**= [8,6,5,4]) になります。
- **PolyValueUpper (uint32)** – 16 進値による多項式値の上位半分が含まれます。デフォルトの分解能は 8 であるため、デフォルトは 0x00h になります。
- **SeedValueLower (uint32) –** 16 進値によるシード値の下位半分が含まれます。デフォルトの分解能は 8 であるため、デフォルトは 0xFFh になります。
- **SeedValueUpper (uint32) –** 16 進値によるシード値の上位半分が含まれます。デフォルトの分解能は 8 であるため、デフォルトは 0 になります。

Clock Select（クロック選択）

Run Mode パラメータに **Clocked** オプションを選択した場合には、クロックソースを装着する必要があります。

注 Implementation (実装) パラメータで **Time Division Multiplex** を選択した場合に、8 を超える分解能で正しい PRS シーケンスを生成するには、データ レートの 4 倍のクロック信号が必要となります。

配置

RPS は、UDB アレイ全体に配置され、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

シングルサイクル、API シングルステップ

リソース	リソースのタイプ				API メモリ（バイト）		ピン（外部入出力ごと）
	データパス セル	PLD	ステータス セル	コントロール/カウント 7 セル	フラッシュ	RAM	
1～8 ビット分解能	1	1	0	1	152	3	2
9～16 ビット分解能	2	1	0	1	188	4	2
7～24 ビット分解能	3	1	0	1	244	6	2
25～32 ビット分解能	4	1	0	1	240	6	2

時分割多重、API 単一ステップ

リソース	リソースのタイプ				API メモリ（バイト）		ピン（外部入出力ごと）
	データパス セル	PLD	ステータス セル	コントロール/カウント 7 セル	フラッシュ	RAM	
9～16 ビット分解能	1	3	1	1	302	6	2
7～24 ビット分解能	2	4	1	1	548	8	2
25～32 ビット分解能	2	3	1	1	624	8	2
33～40 ビット分解能	3	4	1	1	753	12	2
41～48 ビット分解能	3	3	1	1	870	12	2
49～56 ビット分解能	4	4	1	1	956	12	2
57～64 ビット分解能	4	3	1	1	1035	12	2

シングルサイクル、クロック制御

リソース	リソースのタイプ				API メモリ（バイト）		ピン（外部入出力ごと）
	データパス セル	PLD	ステータス セル	コントロール/カウント 7 セル	フラッシュ	RAM	
1～8 ビット分解能	1	1	0	1	180	3	4
9～16 ビット分解能	2	1	0	1	244	4	4
7～24 ビット分解能	3	1	0	1	319	6	4
25～32 ビット分解能	4	1	0	1	302	6	4

時分割多重、クロック制御

リソース	リソースのタイプ				API メモリ（バイト）		ピン（外部入出力ごと）
	データパス セル	PLD	ステータス セル	コントロール/カウント 7 セル	フラッシュ	RAM	
9～16 ビット分解能	1	3	1	1	318	6	4
7～24 ビット分解能	2	4	1	1	512	8	4
25～32 ビット分解能	2	3	1	1	686	8	4
33～40 ビット分解能	3	4	1	1	855	12	4
41～48 ビット分解能	3	3	1	1	990	12	4

リソース	リソースのタイプ				API メモリ（バイト）		ピン（外部入出力ごと）
	データパス セル	PLD	ステータス セル	コントロール/カウント 7 セル	フラッシュ	RAM	
49～56 ビット分解能	**4**	4	1	1	1096	12	4
57～64 ビット分解能	**4**	3	1	1	1175	12	4

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。

デフォルトでは、PSoC Creator は、規定された設計のコンポーネントの最初のインスタンスに「PRS_1」のインスタンス名を割り当てます。インスタンス名は、識別子の構文ルールに従った独自の値に変更できます。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。読みやすいように、下表では「PRS」というインスタンス名を使用しています。

関数	説明
PRS_Start()	カスタマイザにより提供された値にシードと多項式レジスタを初期化します。PRS の計算は、入力クロックの立ち上がりエッジで開始されます。
PRS_Stop()	PRS 計算を停止させます。
PRS_Sleep()	PRS 計算を停止させ、PRS 設定を保存します。
PRS_Wakeup()	PRS 設定を復元し、入力クロックの立ち上がりエッジで PRS の計算を開始します。
PRS_Init()	シード値および多項式レジスタを初期値に初期化します。
PRS_Enable()	入力クロックの立ち上がりエッジで PRS の計算を開始します。
PRS_SaveConfig()	シード値および多項式レジスタを保存します。
PRS_RestoreConfig()	シード値および多項式レジスタを復元します。
PRS_Step()	API シングルステップ モードを使用する場合は、PRS を 1 インクリメントします。
PRS_WriteSeed()	シード値を書き込みます。
PRS_WriteSeedUpper()	シード値の上位半分を書き込みます。これは33～64 ビットPRSの時だけ生成されます。
PRS_WriteSeedLower()	シード値の下位半分を書き込みます。これは33～64 ビットPRSの時だけ生成されます。
PRS_Read()	PRS 値を読み込みます。
PRS_ReadUpper()	PRS 値の上位半分を読み込みます。これは33～64 ビットPRSの時だけ生成されます。

関数	説明
PRS_ReadLower()	PRS 値の下位半分を読み込みます。これは33～64 ビットPRSの時だけ生成されます。
PRS_WritePolynomial()	PRS 多項式値を書き込みます。
PRS_WritePolynomialUpper()	PRS 多項式値の上位半分を書き込みます。これは33～64 ビットPRSの時だけ生成されます。
PRS_WritePolynomialLower()	PRS 多項式値の下位半分を書き込みます。これは33～64 ビットPRSの時だけ生成されます。
PRS_ReadPolynomial()	PRS 多項式値を読み取ります。
PRS_ReadPolynomialUpper()	PRS 多項式値の上位半分を読み取ります。これは33～64 ビットPRSの時だけ生成されます。
PRS_ReadPolynomialLower()	PRS 多項式値の下位半分を読み取ります。これは33～64 ビットPRSの時だけ生成されます。

グローバル変数

変数	説明
PRS_initVar	PRS が初期化されたかどうかを表します。変数は 0 に初期化され、PRS_Start() が初めて呼び出されたときに 1 に設定されます。これにより、コンポーネントは PRS_Start() ルーチンへの最初の呼び出し後、再初期化なしに再起動できます。 コンポーネントの再初期化が必要な場合、PRS_Start() や PRS_Enable() 関数の前に、PRS_Init() 関数が呼び出されます。

void PRS_Start(void)

説明: シード値および多項式レジスタを初期化します。入力クロックの立ち上がりエッジで PRS 計算が開始されます。

パラメータ: なし

返り値: なし

副作用: なし

void PRS_Stop(void)

説明: PRS 計算を停止させます。

パラメータ: なし

返り値: なし

副作用: なし

void PRS_Sleep(void)

説明: PRS 計算を停止させ、PRS 設定を保存します。

パラメータ: なし

返り値: なし

副作用: なし

void PRS_Wakeup(void)

説明: PRS 設定を復元し、入力クロックの立ち上がりエッジで PRS の計算を開始します。

パラメータ: なし

返り値: なし

副作用: なし

void PRS_Init(void)

説明: 初期値で、シード値および多項式レジスタを初期化します。

パラメータ: なし

返り値: なし

副作用: なし

void PRS_Enable(void)

説明: 入力クロックの立ち上がりエッジで PRS 計算を開始します。

パラメータ: なし

返り値: なし

副作用: なし

void PRS_SaveConfig(void)

説明: シード値および多項式レジスタを保存します。

パラメータ: なし

返り値: なし

副作用: なし

void PRS_RestoreConfig(void)

説明: シード値および多項式レジスタを復元します。

パラメータ: なし

返り値: なし

副作用: なし

void PRS_Step(void)

説明: API シングルステップ モードを使用する場合は、PRS を 1 インクリメントします。

パラメータ: なし

返り値: なし

副作用: なし

void PRS_WriteSeed(uint8/16/32 seed)

説明: シード値を書き込みます。

パラメータ: uint8/16/32 seed: シード値

返り値: なし

副作用: シード値は、マスク = $2^{Resolution} - 1$ によってカットされます。

例えば、PRS 分解能が 14 ビットの場合は、マスク値が次のようになります。マスク = $2^{14} – 1$ = 0x3FFFu。

シード値 = 0xFFFFu はカットされ、シード AND マスク = 0xFFFFu AND 0x3FFFu = 0x3FFFu となります。

void PRS_WriteSeedUpper(uint32 seed)

説明: シード値の上位半分を書き込みます。これは33～64 ビットPRSの時だけ生成されます。

パラメータ: uint32 シード: シード値の上位半分

返り値: なし

副作用: シード値の上位半分は、マスク = $2^{(Resolution – 32)} - 1$ に準じてカットされます。

例えば、PRC 分解能が 35 ビットの場合は、マスク値が次のようになります:

$2^{(35 – 32)} – 1 = 2^{\wedge 3} – 1$ = 0x0000 0007u。

シード値の上位半分 = 0x0000 00FFu はカットされ、

シードの上位半分 AND マスク = 0x0000 00FFu AND 0x0000 0007u = 0x0000 0007u。

void PRS_WriteSeedLower(uint32 seed)

説明: シード値の下位半分を書き込みます。これは33～64 ビットPRSの時だけ生成されます。

パラメータ: uint32 シード: シード値の下位半分

返り値: なし

副作用: なし

uint8/16/32 PRS_Read(void)

説明: PRS 値を読み込みます。

パラメータ: なし

返り値: uint8/16/32: PRS 値を返します。

副作用: なし

uint32 PRS_ReadUpper(void)

説明: PRS 値の上位半分を読み込みます。これは33～64 ビットPRSの時だけ生成されます。

パラメータ: なし

返り値: uint32: PRS 値の上位半分を返します。

副作用: なし

uint32 PRS_ReadLower(void)

説明: PRS 値の下位半分を読み込みます。これは33～64 ビットPRSの時だけ生成されます。

パラメータ: なし

返り値: uint32: PRS 値の下位半分を返します。

副作用: なし

void PRS_WritePolynomial(uint8/16/32 polynomial)

説明: PRS 多項値を書き込みます。

パラメータ: uint8/16/32 polynomial: PRS Polynomial (PRS 多項)

返り値: なし

副作用: 多項式値は、マスク = $2^{Resolution}$ - 1 に準じてカットされます。
例えば、PRS 分解能が 14 ビットの場合は、マスク値が次のようになります。マスク = $2^{14} – 1$ = 0x3FFFu。
多項式値 = 0xFFFFu はカットされ、
多項 AND マスク = 0xFFFFu AND 0x3FFFu = 0x3FFFu。

void PRS_WritePolynomialUpper(uint32 polynomial)

説明: PRS 多項式値の上位半分を書き込みます。これは33～64 ビットPRSの時だけ生成されます。

パラメータ: uint32 多項式値 PRS 多項式値の上位半分。

返り値: なし

副作用: 多項式値の上位半分は、マスク = $2^{(Resolution – 32)}$ - 1 に準じてカットされます。
例えば、PRC 分解能が 35 ビットの場合は、マスク値が次のようになります。
$2^{(35 – 32)} – 1 = 2^3 – 1$ = 0x0000 0007u。
多項式値の上位半分 = 0x0000 00FFu はカットされ、
多項式値の上位半分 AND マスク = 0x0000 00FFu AND 0x0000 0007u = 0x0000 0007u。

void PRS_WritePolynomialLower(uint32 polynomial)

説明: PRS多項式値の下位半分を書き込みます。これは33～64 ビットPRSの時だけ生成されます。

パラメータ: uint32多項式値 PRS 多項式値の下位半分

返り値: なし

副作用: なし

uint8/16/32 PRS_ReadPolynomial(void)

説明: PRS 多項式値を読み取ります。

パラメータ: なし

返り値: uint8/16/32: PRS 多項式値を返します。

副作用: なし

uint32 PRS_ReadPolynomialUpper(void)

説明: PRS 多項式値の上位半分を読み取ります。これは33～64 ビットPRSの時だけ生成されます。

パラメータ: なし

返り値: uint32: PRS 多項式値の上位半分を返します。

副作用: なし

uint32 PRS_ReadPolynomialLower(void)

説明: PRS 多項式値の下位半分を読み取ります。これは33～64 ビットPRSの時だけ生成されます。

パラメータ: なし

返り値: uint32: PRS 多項式値の下位半分を返します。

副作用: なし

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

PRS ラン モード: クロック制御

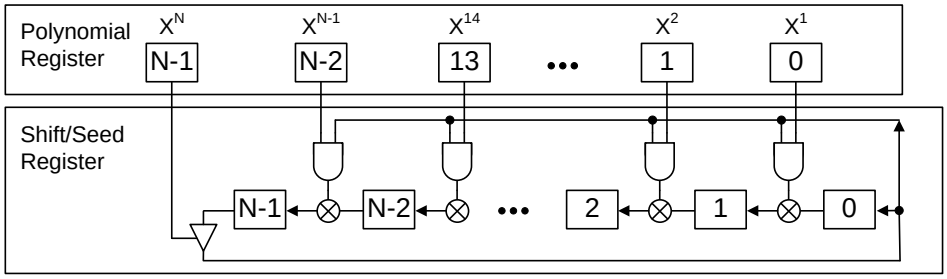
このモードでは、PRS コンポーネントは、開始後からイネーブル信号が HIGH である限り連続的に実行されます。

PRS ラン モード: API シングルステップ

このモードでは、PRS は API コールによりインクリメントされます。

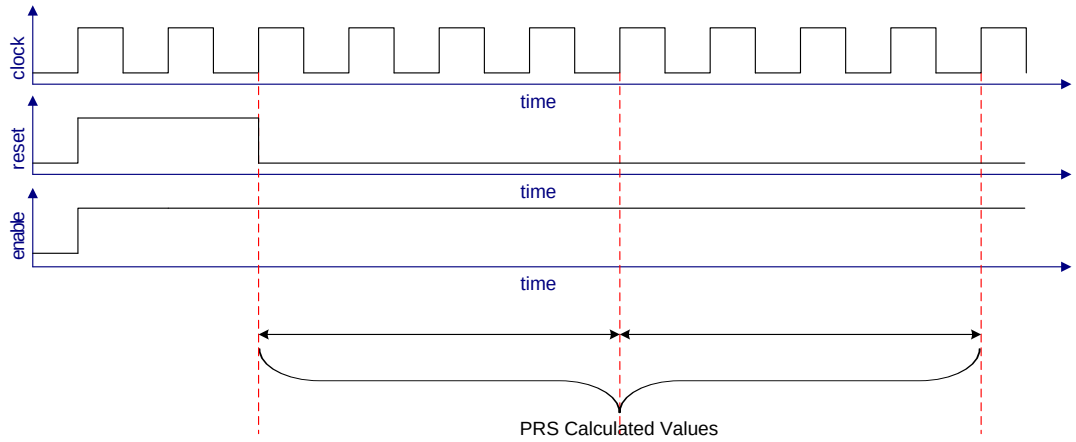
ブロックダイアグラムと設定

PRS は構成された UDB のセットとして実装されます。実装は以下のブロック図のように行われます。

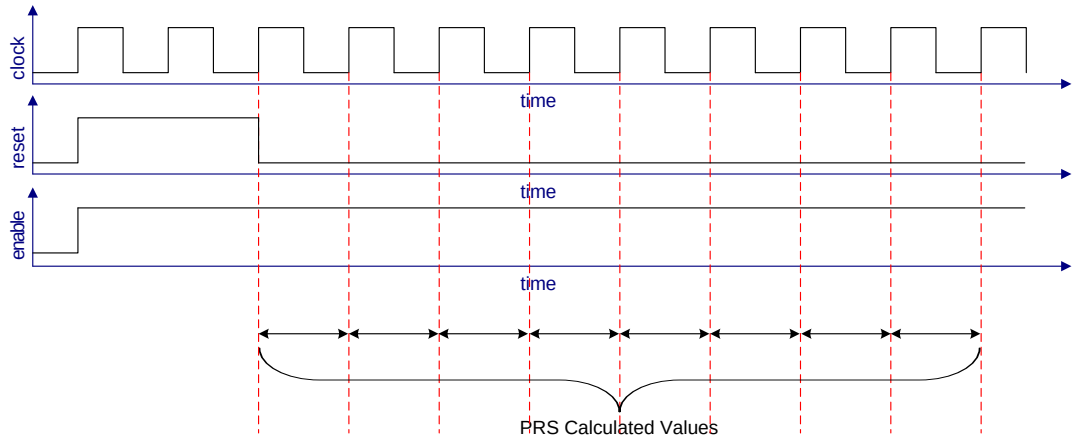


タイミング図

時間分割多重化実装モード



シングル サイクル実装モード



レジスタ

多項式レジスタ (解像度に基づき 2~64 ビット)

多項式レジスタには多項式値が含まれます。PRS_WritePolynomial()、PRS_WritePolynomialUpper()、または、PRS_WritePolynomialLower() 関数によって変更できます。PRS_ReadPolynomial()、PRS_ReadPolynomialUpper()、または、PRS_ReadPolynomialLower() を使用して現在の多項式値を読み取ることもできます。

シフト/シード レジスタ (解像度に基づき 2~64 ビット)

シフト/シード レジスタにはシード値が含まれます。PRS_WriteSeed()、PRS_WriteSeedUpper()、または、PRS_WriteSeedLower() 関数によって変更できます。PRS_ReadSeed()、PRS_ReadSeedUpper()、または、PRS_ReadSeedLower() を使用して現在のシード値を読み取ることもできます。

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。

「公称ルーティングでの最大」タイミング特性

パラメータ	説明	構成 [1]	Min	Typ	Max	単位
f_{CLOCK}	コンポーネント クロック周波数 [2]	構成1	–	–	57	MHz
		構成2	–	–	57	MHz

[1] 設定:

設定 1:
Resolution (分解能): 8 ビット
ラン モード: API シングルステップ
ローパワー モード動作: 電源立ち上げ時に復元
Implementation (実装): シングル サイクル

設定 2:
Resolution (分解能): 8 ビット
ラン モード: クロック制御
ローパワー モード動作: 電源立ち上げ時に復元
Implementation (実装): シングル サイクル

設定 3:
Resolution (分解能): 16 ビット
ラン モード: クロック制御
ローパワー モード動作: 電源立ち上げ時に復元
Implementation (実装): 時間分割多重化

構成 4:
Resolution (分解能): 16 ビット
ラン モード: API シングルステップ
ローパワー モード動作: 電源立ち上げ時に復元
Implementation (実装): 時間分割多重化

構成 5:
Resolution (分解能): 32 ビット
ラン モード: API シングルステップ
ローパワー モード動作: 電源立ち上げ時に復元
Implementation (実装): シングル サイクル

構成 6:
Resolution (分解能): 32 ビット
ラン モード: クロック制御
ローパワー モード動作: 電源立ち上げ時に復元
Implementation (実装): シングル サイクル

構成 7:
Resolution (分解能): 64 ビット
ラン モード: API シングルステップ
ローパワー モード動作: 電源立ち上げ時に復元
Implementation (実装): 時間分割多重化

構成 8:
Resolution (分解能): 64 ビット
ラン モード: クロック制御
ローパワー モード動作: 電源立ち上げ時に復元
Implementation (実装): 時間分割多重化

[2] 時分割多重実装を選択する場合には、コンポーネント クロックの周波数はデータ レートの 4 倍の大きさでなければなりません。

パラメータ	説明	構成 [1]	Min	Typ	Max	単位
		構成3	–	–	35	MHz
		構成4	–	–	30	MHz
		構成5	–	–	43	MHz
		構成6	–	–	40	MHz
		構成7	–	–	25	MHz
		構成8	–	–	30	MHz
t_{clockH}	入力クロック HIGH 時 [3]	該当せず	–	0.5	–	$1/f_{clock}$
t_{clockL}	入力クロック LOW 時 [3]	該当せず	–	0.5	–	$1/f_{clock}$
Inputs (入力)						
t_{PD_ps}	入力パス遅延、同期ピン [4]	1	–	–	STA [5]	ns
t_{PD_ps}	入力パス遅延、同期ピン [6]	2	–	–	8.5	ns
t_{PD_si}	入力パス遅延への同期出力（ルート）	1,2,3,4	–	–	STA [5]	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_{PD_IE}	コンポーネントクロックへの入力パス遅延（エッジ依存入力）	1,2	$t_{PD_ps} + t_{SYNC} + t_{PD_si}$	–	$t_{PD_ps} + t_{SYNC} + t_{PD_si} + t_{I_clk}$	ns
t_{PD_IE}	コンポーネントクロックへの入力パス遅延（エッジ依存入力）	3,4	$t_{SYNC} + t_{PD_si}$	–	$t_{SYNC} + t_{PD_si} + t_{I_clk}$	ns
t_{IH}	入力HIGH期間	1,2,3,4	t_{CY_clock} [7]	–	–	ns
t_{IL}	入力LOW期間	1,2,3,4	t_{CY_clock} [7]	–	–	ns
Outputs (出力)						
	出力レジスターレジスタ時間		–	–	–	
	ピン パス遅延への出力		–	–	–	

[3] $t_{CY_clock} = 1/f_{CLOCK.}$ これは 1 クロック周期のサイクルタイムです。

[4] t_{PD_ps} 後述の Static Timing Results（静的タイミング結果）を参照。ここにリストされた数値は、多数の入力の STA 分析に基づく呼び値です。

[5] t_{PD_ps} と t_{PD_si} はルートパスの遅延。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。これらの値は、静的タイミング分析の結果に記載されています。

[6] t_{PD_ps} 構成 2 で、デバイスのピン毎に定義された固定値。ここに挙げた数字は、デバイスで利用可能なすべてのピンの公称値です。

[7] $t_{CY_clock} = 4\times [1/f_{CLOCK}]$ 時分割多重実装が選択された場合。

「すべてのルーティングでの最大」タイミング特性

パラメータ	説明	構成[1]	Min	Typ	Max[2]	単位
f_{CLOCK}	コンポーネント クロック周波数 [3]	構成1	–	–	29	MHz

[1] 設定:

設定 1:
Resolution (分解能): 8 ビット
ラン モード: API シングルステップ
ローパワー モード動作: 電源立ち上げ時に復元
Implementation (実装): シングル サイクル

設定 2:
Resolution (分解能): 8 ビット
ラン モード: クロック制御
ローパワー モード動作: 電源立ち上げ時に復元
Implementation (実装): シングル サイクル

設定 3:
Resolution (分解能): 16 ビット
ラン モード: クロック制御
ローパワー モード動作: 電源立ち上げ時に復元
Implementation (実装): 時間分割多重化

構成 4:
Resolution (分解能): 16 ビット
ラン モード: API シングルステップ
ローパワー モード動作: 電源立ち上げ時に復元
Implementation (実装): 時間分割多重化

構成 5:
Resolution (分解能): 32 ビット
ラン モード: API シングルステップ
ローパワー モード動作: 電源立ち上げ時に復元
Implementation (実装): シングル サイクル

構成 6:
Resolution (分解能): 32 ビット
ラン モード: クロック制御
ローパワー モード動作: 電源立ち上げ時に復元
Implementation (実装): シングル サイクル

構成 7:
Resolution (分解能): 64 ビット
ラン モード: API シングルステップ
ローパワー モード動作: 電源立ち上げ時に復元
Implementation (実装): 時間分割多重化

構成 8:
Resolution (分解能): 64 ビット
ラン モード: クロック制御
ローパワー モード動作: 電源立ち上げ時に復元
Implementation (実装): 時間分割多重化

[2]「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以下で実行される場合に、ユーザがタイミング条件を気にする必要がなくなります。

[3] 時分割多重実装を選択する場合には、コンポーネント クロックの周波数はデータ レートの 4 倍の大きさでなければなりません。

パラメータ	説明	構成[1]	Min	Typ	Max[2]	単位
		構成2	–	–	29	MHz
		構成3	–	–	18	MHz
		構成4	–	–	15	MHz
		構成5	–	–	22	MHz
		構成6	–	–	20	MHz
		構成7	–	–	13	MHz
		構成8	–	–	15	MHz
t_{clockH}	入力クロック HIGH 期間 [4]	該当せず	–	0.5	–	$1/f_{clock}$
t_{clockL}	入力クロック LOW 期間 [4]	該当せず	–	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_si}	入力パス遅延への同期出力（ルート）	1,2,3,4	–	–	STA [6]	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_{PD_IE}	コンポーネントクロックへの入力パス遅延（エッジ依存入力）	1,2	$t_{PD_ps} + t_{SYNC} + t_{PD_si}$	–	$t_{PD_ps} + t_{SYNC} + t_{PD_si} + t_{I_clk}$	ns
t_{PD_IE}	コンポーネントクロックへの入力パス遅延（エッジ依存入力）	3,4	$t_{SYNC} + t_{PD_si}$	–	$t_{SYNC} + t_{PD_si} + t_{I_clk}$	ns
t_{IH}	入力HIGH期間	1,2,3,4	t_{CY_clock} [8]	–	–	ns
t_{IL}	入力LOW期間	1,2,3,4	t_{CY_clock} [8]	–	–	ns
Outputs (出力)						
	出力レジスターレジスタ時間		–	–	–	
	ピン パス遅延への出力		–	–	–	

[4] $t_{CY_clock} = 1/f_{CLOCK}$. これは 1 クロック周期のサイクルタイムです。

[5] t_{PD_ps} 後述の Static Timing Results（静的タイミング結果）を参照。ここにリストされた数値は、多数の入力の STA 分析に基づく呼び値です。

[6] t_{PD_ps} と t_{PD_si} はルートパスの遅延。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。これらの値は、静的タイミング分析の結果に記載されています。

[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。ここに挙げた数字は、デバイスで利用可能なすべてのピンの公称値です。

[8] $t_{CY_clock} = 4\times [1/f_{CLOCK}]$ 時分割多重実装が選択された場合。

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、次の手法で設計の最大値を計算できます:

f_{CLOCK} 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* によるクロック制限の例を示しています。

+Clock Summary

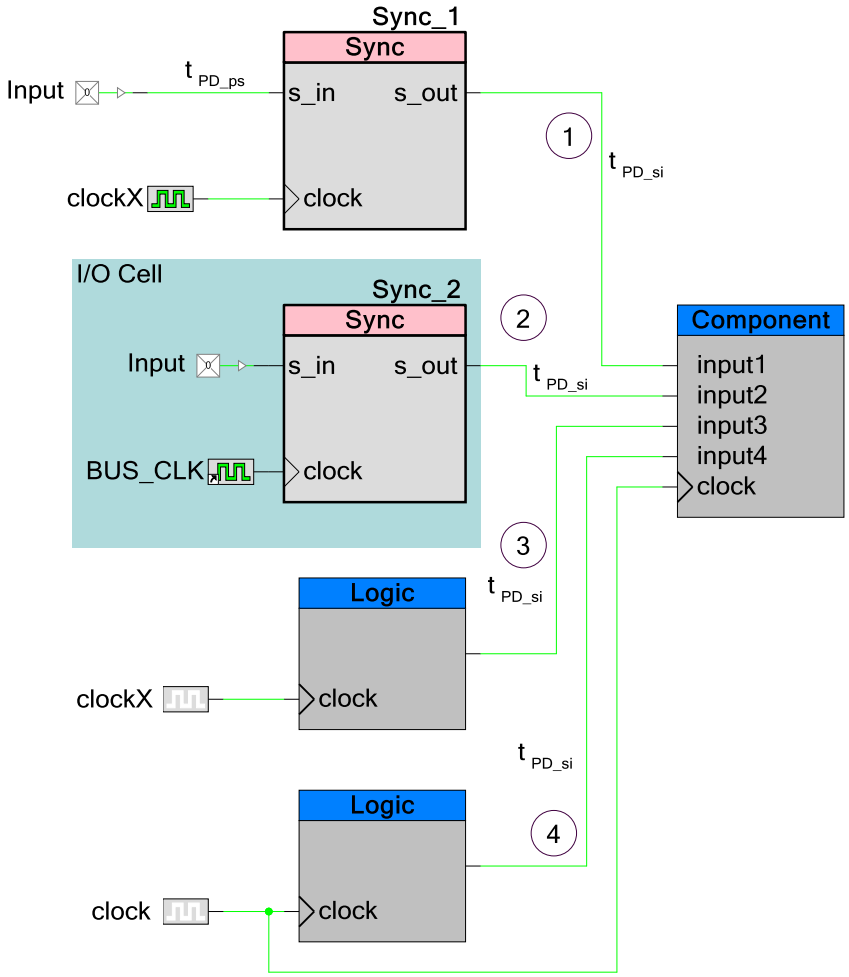
Clock	Actual Freq	Max Freq	Violation
BUS_CLK	66.000 MHz	63.295 MHz	VIOLATION
CharComp_clock	33.000 MHz	57.594 MHz	

入力パス遅延とパルス幅

入力の機能分析を行う場合、すべての入力は、どのような構成になっているかに関わらず、図 1 に示す 4 つの可能な構成の 1 つに該当します。

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、各入力でどの入力構成を設定したか、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 1. コンポーネントタイミング仕様のための入力構成



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6
1	クロック	master_clock	図 4
1	クロック	clockX = clock [1]	図2
1	クロック	clockX > clock	図3
1	クロック	clockX < clock	図 5
2	master_clock	master_clock	図6
2	クロック	master_clock	図 4

[1] クロック周波数は同じですが、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	master_clock	master_clock	図11
3	クロック	master_clock	図 9
3	クロック	clockX = clock [1]	図7
3	クロック	clockX > clock	図8
3	クロック	clockX < clock	図 10
4	master_clock	master_clock	図11
4	クロック	クロック	図7

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

   この方法で設定される入力とした場合、clockX は、コンポーネント クロックより高速、低速、または同じ場合があります。master_clock と同じにすることもできます。これにより、図 2、図 3、図 5、そして、図 6 に示す特性分析パラメータが生成されます。

2. この入力は、デバイスピンから駆動され、master_clock を使用してピンの部分で同期されます。

   この方法で構成された入力とした場合、master_clock はコンポーネントのクロックより速い場合と同じ場合があります（遅いことはありません）。これにより、図 3 と 図 6 に示す特性分析パラメータが生成されます。

図 2. 入力構成 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 および clockX のエッジアライメントは保証されません。）

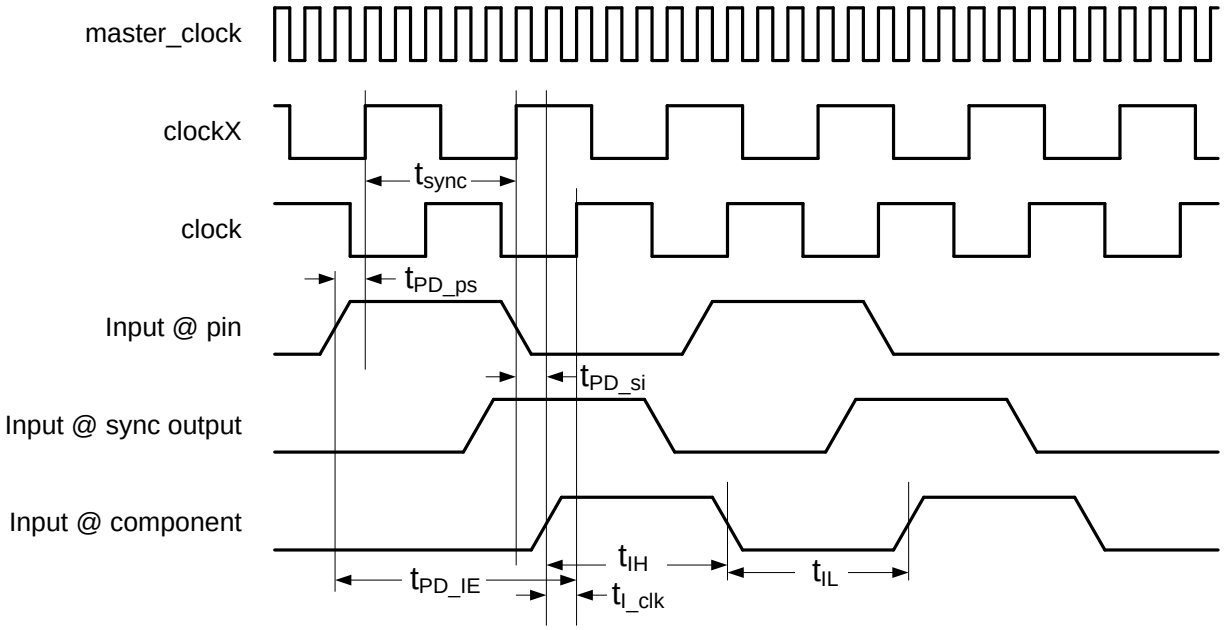


図 3. 入力構成 1 と 2; Sync. クロック周波数 > コンポーネント クロック 周波数

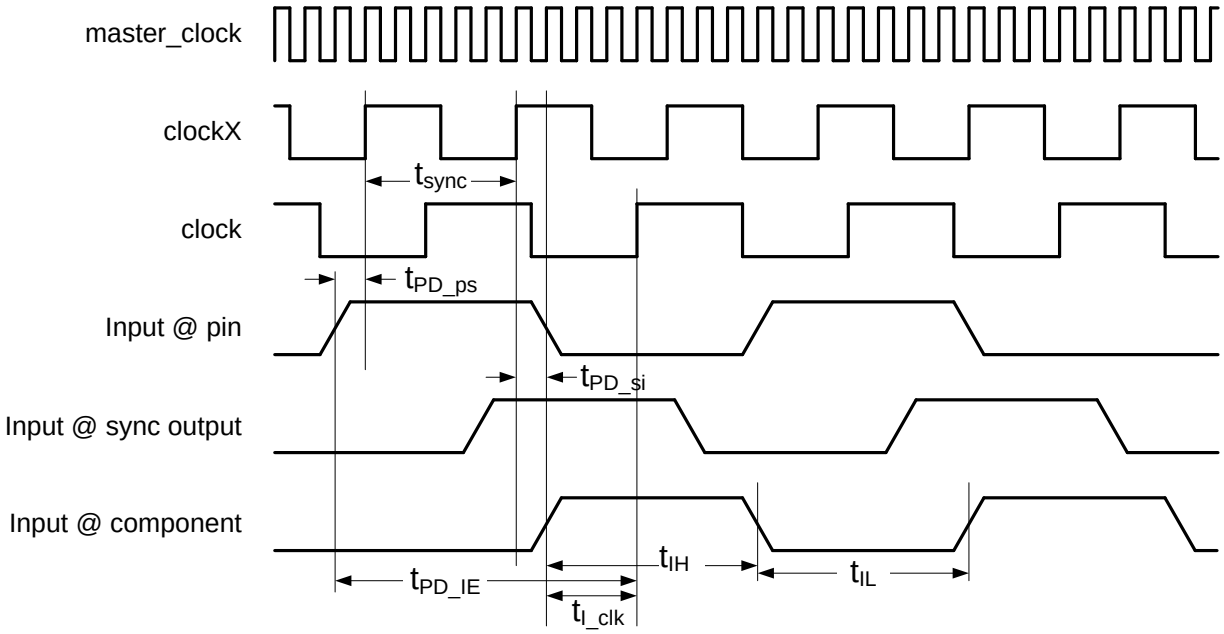


図 4. 入力構成 1 と 2; [Sync. クロック周波数 == master_clock] > コンポーネント クロック周波数

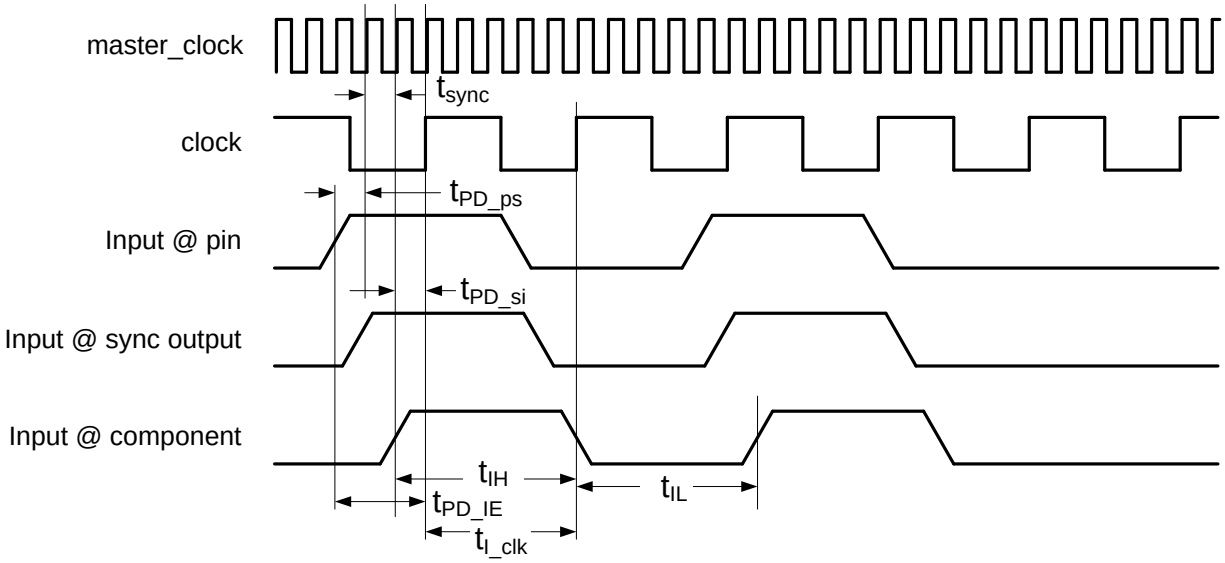


図 5. 入力構成 1; Sync. クロック周波数 < コンポーネント クロック周波数

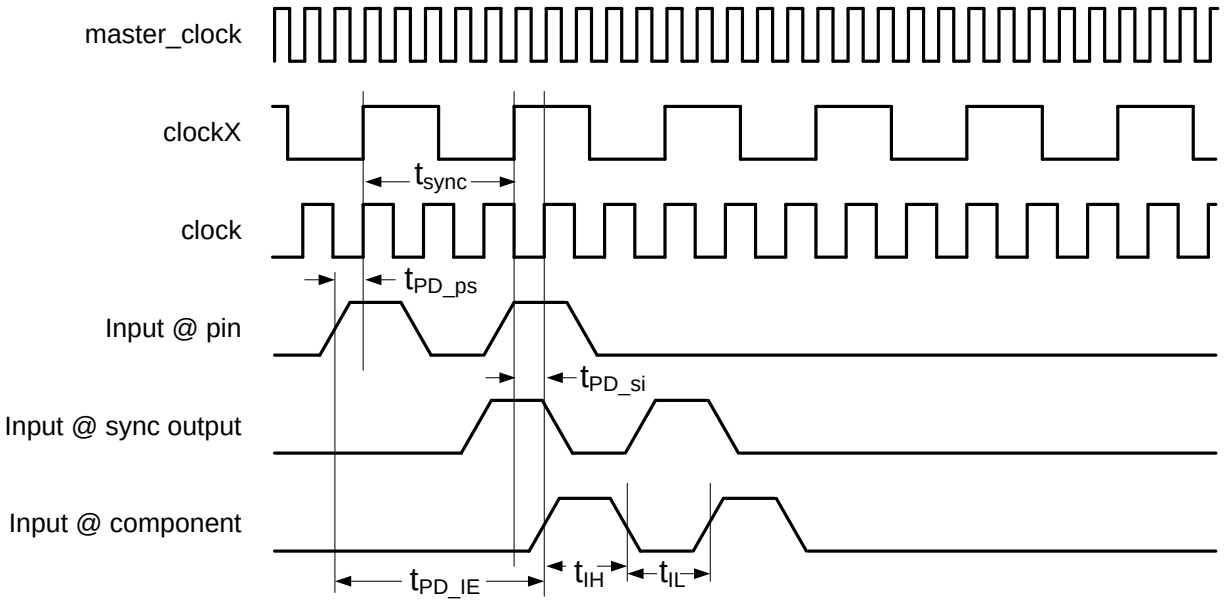
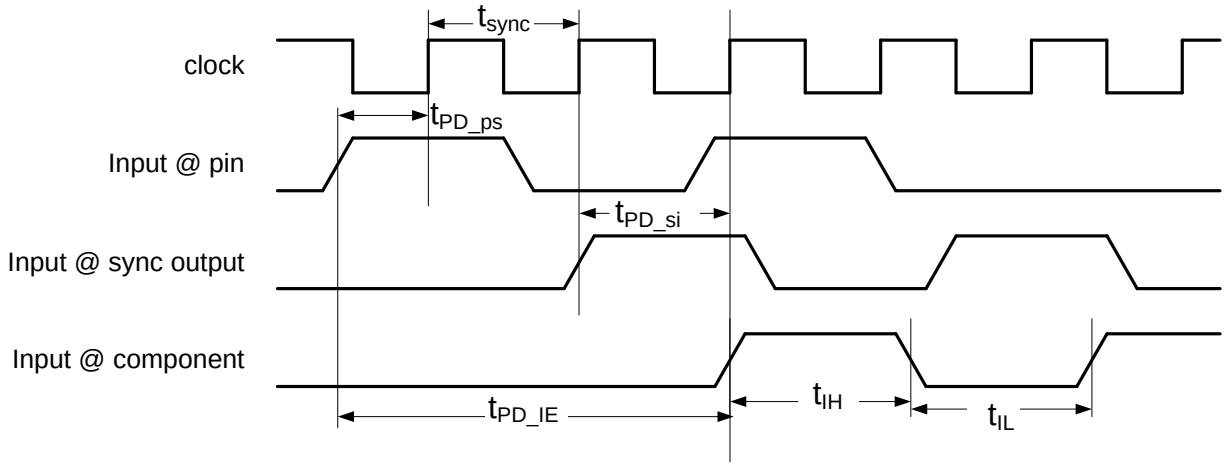


図 6. 入力構成 1 と 2; Sync.クロック = コンポーネントクロック= master_clock



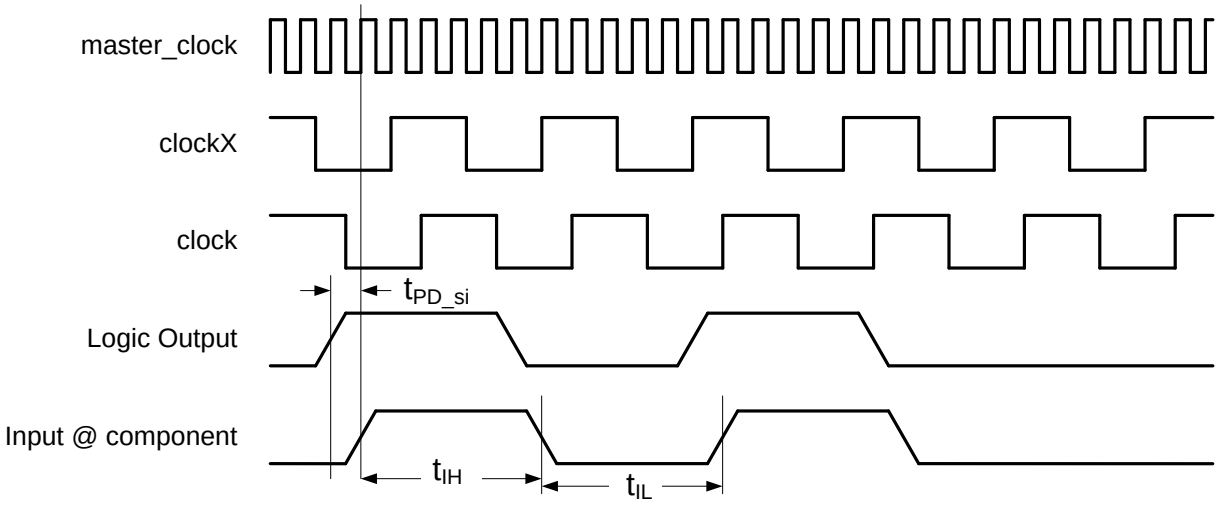
3. 入力は PSoC 内部のロジックによって駆動されます。これはコンポーネントが使用するクロックとは異なるクロックをベースにして同期しています（すべての内蔵クロックは master_clock から生成されます）。

   この方法で設定される入力を特性化する場合、シンクロナイザ クロック は、コンポーネント クロックより高速、低速、または同じになります。これにより、図 7、 図 8、および図 10 に示す特性分析パラメータが生成されます。

4. 入力は PSoC 内部のロジックに駆動されます。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。

   入力をこの方法で設定する場合には、シンクロナイザ クロックはコンポーネント クロックと同じになります。これにより、図 11 に示す特性分析パラメータが生成されます。

図 7. 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 および clockX のエッジアライメントは保証されません。）



この図は、静的タイミング分析 (STA) によるクロックの情報を示します。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。ただし、同じ周波数の 2 つのクロックは立ち上がりエッジが揃っていない場合があります。そのため、静的タイミング分析ツールは、クロックが同期しているエッジがどちらか判別できず、最低 1 つの master_clock サイクルを想定します。すなわち、t_{PD_si} は今、システム master_clock 上で限定的な効果しかありません。パス遅延が長すぎる場合には、master_clock セットアップ時間違反が発生します。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 8. 入力構成 3; Sync. クロック周波数 > コンポーネント クロック 周波数

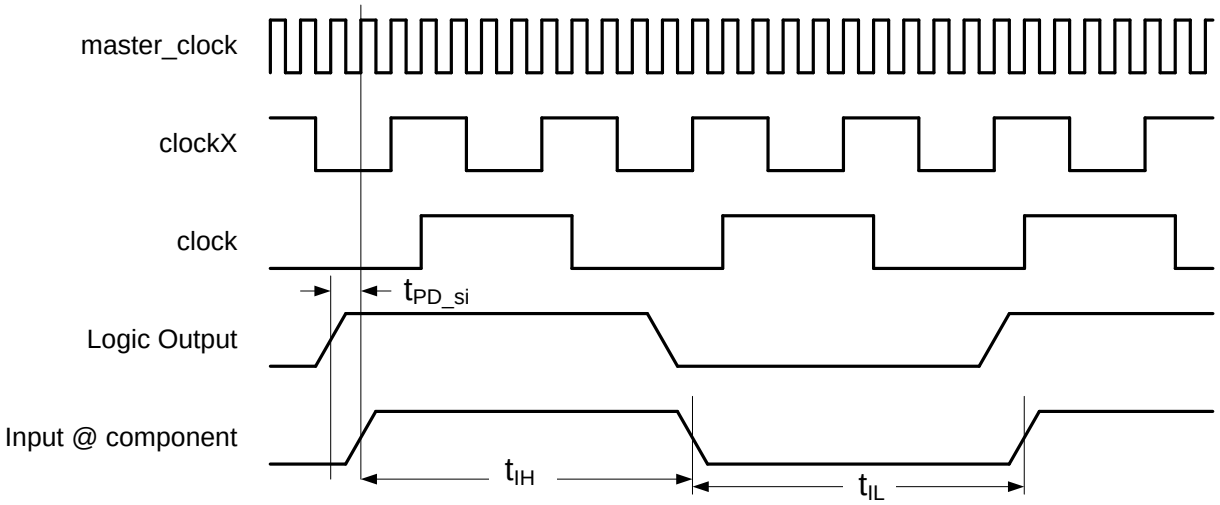


図 7 に示すのと類似した方法で、すべてのクロックは master_clock から派生します。STA は、この構成で 1master_clock サイクル分、master_clock における t_{PD_si} の制限を示します。このパスの遅延が長すぎると、master_clock セットアップ時間の違反が発生します。この場合、システムの同期クロックを変更するか、master_clock を遅い周波数で実行しなければなりません。

図 9. 入力構成 3; シンクロナイザクロック周波数 = master_clock > コンポーネントクロック周波数

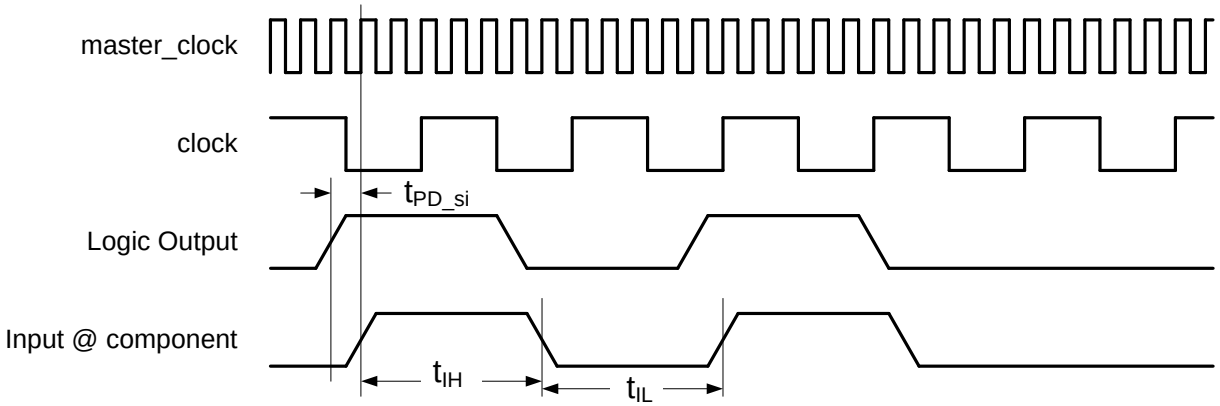


図 10. 入力構成 3; シンクロナイザクロック周波数 < コンポーネントクロック周波数

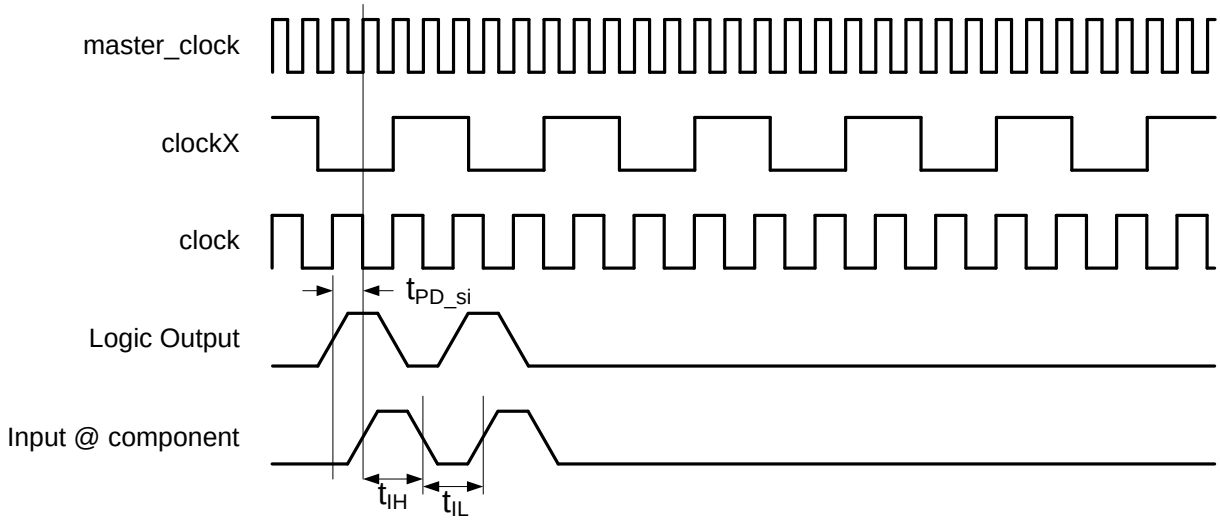
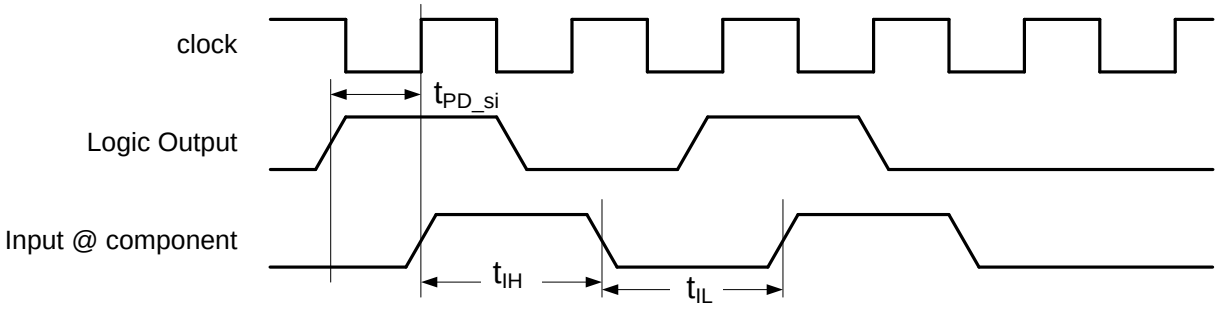


図 7 に示すのと類似した方法で、すべてのクロックは master_clock から派生します。STA は、この構成で 1master_clock サイクル分、master_clock における t_{PD_si} の制限を示します。このパスの遅延が長すぎると、master_clock セットアップ時間の違反が発生します。この場合、システムの同期クロックを変更するか、master_clock を遅い周波数で実行しなければなりません。

図 11. 入力構成 4 のみ; シンクロナイザクロック = コンポーネントクロック



このセクションのこれまでのすべての図の中で、使用中の実装を理解するために最も重要なパラメータは f_{CLOCK} と t_{PD_IE} です。t_{PD_IE} は t_{PD_ps} および t_{SYNC}（構成 1 と 2 のみ）、t_{PD_si}, および t_{I_Clk} によって定義されます。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ことに注意することが重要です。t_{I_Clk} は STA の結果によるものではありませんが、t_{PD_IE} が登録された場合は重要となります。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にあるマージンです。

t_{PD_ps} と t_{PD_si} は STA の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 Fan-out 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。

+Setup times

+Setup times to clock BUS_CLK

Start	Register	Clock	Delay (ns)
input1(0):iocell.pad_in	input1(0)_SYNC:synccell.syncd	BUS_CLK	13.246

t_{PD_si} は、レジスタ～レジスタ時間に定義されています。*_timing.html* ファイルを使用するには、ネット名を知っていなければなりません。このパスの Fan-out 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 CharComp_clock

Destination clock CharComp_clock (Actual freq: 33.000 MHz)

+Source clock BUS_CLK

Source clock BUS_CLK (Actual freq: 66.000 MHz)

Start	End	Period (ns)	Max Freq	Frequency	Violation
input1(0)_SYNC:synccell.syncq	\CharComp:sC8:PRSdp:u0\:datapathcell.sync_ov_msb	15.799	63.295 MHz	66.000 MHz	SETUP

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、デバイス内の他のコンポーネント、または、デバイス外のピンのどちらかに送られます。前者の場合、ロジック～入力の説明に記載されているレジスタ～レジスタ時間を見ます（ソースクロックはコンポーネントクロックです）。後者の場合、*_timing.html* STA 結果のクロック～出力時間を見ます。

+Clock to output times

+Clock to output times from clock CharComp_clock

Start	Register	End	Delay (ns)
CharComp_clock	\CharComp:sC8:PRSdp:u0\:datapathcell.regq_a0	CharComp_bitstream(0):iocell.pad_out	27.253

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.b	最新のリソース情報はデータシートにあります	
2.0.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	
2.0	PSoC 3 製品版シリコンのために追加されたサポート。変更は以下のとおりです: ■ 時間分割多重化の実装モードに 4x クロックを追加 ■ 1x クロックでのシングル サイクル実装が 1～32 ビットで利用可能になりました。 ■ 4x クロックでの時間分割多重化の実装モードが 9～64 ビットで利用可能になりました。 ■ 同期入力信号のリセットが追加されました。 ■ 同期入力信号のイネーブルが追加されました。 ■ 実装と低電力モードパラメータのための Configure ダイアログに、「Advanced（詳細設定）」ページが新たに追加されました。	PSoC 3 製品版デバイスをサポートする新しい要求により、PRS コンポーネントの新しい 2.0 バージョンが作成されました。
	PRS_Sleep()/PRS_Wakeup() および PRS_Init()/PRS_Enable() API が追加されました。	低消費電力モードをサポートし、ほとんどのコンポーネントの初期化と有効化の制御を分離する共通インターフェースを提供するため。

バージョン	変更の説明	変更の理由 / 影響
	PRS_WriteSeed() と PRS_WriteSeedUpper() の機能をアップデートしました。	マスク パラメータは、シード値をカットして、書き込み中に分解能を定義するために使用されました。
	リセット DFF のトリガが、次の多項式書き込み関数に追加されました。PRS_WritePolynomial()、PRS_WritePolynomialUpper()、PRS_WritePolynomialLower()。	計算を開始する前に、DFF トリガを正しい状態に設定する必要があります (多項式の最上位ビットは常に 1)。この条件を満たすため、シードまたは多項式レジスタへの書き込みは、DFF トリガをリセットします。
	一部のパラメータで、Expression View が見れるよう、Configureダイアログが更新されました。	Expression View は、記号パラメータに直接アクセスするために使用されます。このビューによって、必要に応じて、コンポーネント パラメータを外部パラメータに接続することができます。
	[Configure (設定)] ダイアログが更新され、さまざまなパラメータにエラー アイコンが追加されました。	テキスト ボックスに不正な値を入力すると、問題を説明するツールのヒントとともに、エラー アイコンが表示されます。このことによって、別々のエラー メッセージより使いやすくなります。

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。