



---

# Viterbi コンパイラ ユーザーガイド

この資料は英語版を翻訳したもので、内容に相違が生じる場合には原文を優先します。こちらの日本語版は参考用としてご利用ください。設計の際には、最新の英語版で内容をご確認ください。



101 Innovation Drive  
San Jose, CA 95134  
[www.altera.com](http://www.altera.com)

ソフトウェア・バージョン： 11.0  
ドキュメント・デート： 2011年5月



Copyright © 2011 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.



## 第1章 このコンパイラについて

特長	1-2
リリース情報	1-2
サポートされるデバイス・ファミリ	1-3
パフォーマンスおよびリソース使用率	1-4
ハイブリッド・アーキテクチャ	1-4
パラレル・アーキテクチャ	1-6
インストールおよびライセンス	1-7
OpenCore Plus 評価機能	1-8
OpenCore Plus タイム・アウト動作	1-8

## 第2章 使用法

デザイン・フロー	2-1
DSP Builder フロー	2-1
MegaWizard Plug-In Manager フロー	2-2
MegaCore ファンクションのパラメータ化	2-5
シミュレーションの設定	2-9
MegaCore ファンクションの生成	2-10
デザインのシミュレーション	2-13
デザインのコンパイル	2-13
デバイスのプログラミング	2-13

## 第3章 機能の説明

ソフト・シンボル入力	3-1
エンコード手法	3-1
ステート・メトリック	3-2
パンクチャリング手法	3-2
トレリス符号化変調	3-3
トレリス終端	3-7
トレリスの開始	3-8
Avalon Streaming インタフェース	3-8
パラメータ	3-9
Architecture タブ	3-9
BER 見積もりツール	3-9
ノード同期化	3-10
Code Sets タブ	3-11
Parameters タブ	3-12
スループット・カリキュレータ	3-13
レイテンシ・カリキュレータ	3-13
Test Data タブ	3-14
信号	3-14
タイミング図	3-18
MegaCore 検証	3-20

## 追加情報

改訂履歴	Info-1
アルテラへの問い合わせ	Info-1
表記規則	Info-2



このドキュメントでは、アルテラの Viterbi コンパイラについて説明します。Viterbi コンパイラは、幅広い標準 Viterbi デコーダを実装する、高性能でソフト・デシジョンの Viterbi MegaCore ファンクションから構成されています。

Viterbi デコード（最尤復号法または前向き動的計画法としても知られている）は、漸近的に最適のデコード技法を使用することにより畳み込みコードをデコードする最も一般的な方法です。その基本フォームにおいて、Viterbi デコードは、最適な網羅的探索を実行する効率的な再帰的アルゴリズムです。

畳み込みエンコーダおよび Viterbi デコーダは、ノイズの多いチャネル（例えば：通信チャネル）での誤り訂正を提供するのに一緒に使用できます。畳み込みエンコーダは、送信前にデータのストリームに冗長符号（すなわち、余分なビット）を追加します。

レートおよび生成多項式は畳み込みコードを記述するため、畳み込みエンコーダを説明します。レートとは、入力ビットあたりの送信済みビットの数です。例えば、 $1/2$  のレートは、1 ビットをエンコードして、送信用の 2 ビットを生成します。同様に、 $2/3$  のレートは、2 ビットをエンコードして、送信用の 3 ビットを生成します。確定的なパターンに基づいてエンコードされるビットを一部削除することでコードをパンクして、レートを向上させることができます。

生成多項式は、エンコードされたビットを生成するために数学的に組み合わせられる畳み込みエンコーダの状態のビットを表します。エンコードされるビットあたりには、1 つの生成多項式があります。生成多項式の長さ（ビット単位）は拘束長と呼ばれます。通常、高い拘束長を持つシステムは比較的堅牢です。ただし、拘束長が Viterbi デコーダの複雑さは拘束長が高いほど高くなるため、9 以上の拘束長は珍しいです。

ノイズの高いチャネルは、レシーバでビット・エラーを発生させます。Viterbi アルゴリズムは、実際に受信されるシーケンスに一番近いビット・シーケンスを見つけます。Viterbi デコーダは、畳み込みエンコーダが付与した冗長性を使用してビットストリームをデコードし、エラーを削除します。

レシーバは、ハード・シンボルまたはソフト・シンボルのいずれかを Viterbi デコーダに送信することができます。ハード・シンボルはバイナリの  $\pm 1$  に相当します。ソフト・シンボルは、ビットが正または負の信頼性を表すようにマルチ・レベル化されています。例えば、チャネルが非衰減かつガウスである場合、特定のビットに量子化された一致するフィルタの出力は、適切なソフト入力となります。パンクしたシンボルは `eras_sym` の入力で示されています。Viterbi アルゴリズムは、ソフト入力シンボルを使用する場合により良いパフォーマンスをしています。

Viterbi デコーダはデータのブロックまたは連続ストリームで動作します。プロセス時には、N 個のシンボルが使用されています。ここで、N はエンコードされたシンボルの数です。トレースバック長とは、デコーダがビットを決定する前に処理されたトレリス状態の数です。

## 特長

Viterbi コンパイラは、2つの高性能で面積が最適化されたソフト・デシジョンの Viterbi MegaCore ファンクション（ハイブリッド・アーキテクチャおよびパラレル・アーキテクチャ）を提供しています。両方の MegaCore ファンクションでも、BER 見積りツール、ノードの同期、および複数のコード・セット（可変拘束長を含む）を指定することができます。

Vertibi コンパイラは以下の機能をサポートしています。

- 高速なパラレル・アーキテクチャ：
  - 250 Mbps を超える性能
  - 完全にパラレルで動作
  - 最適化されたブロック・デコードおよび連続デコード
- 中低速なハイブリッド・アーキテクチャ：
  - コンフィギュレーション可能な ACS ユニット数
  - メモリをベースとするアーキテクチャ
  - 幅広い性能、幅広いロジック・エリア
- 以下を含む完全にパラメータ化された Viterbi ファンクション：
  - コード・ビット数。
  - 拘束長
  - ソフト・ビット数
  - トレースバック長
  - コードされたビットあたりに多項式
- Avalon® ストリーミング（Avalon-ST）インタフェース
- 可変な拘束長
- トレリス符号化変調（TCM）のオプション
- 使いやすい IP Toolbench インタフェース
- DSP Builder レディ
- デコーダを検証するための VHDL テストベンチ
- アルテラでサポートしている VHDL、Verilog HDL シミュレータ上で使用可能な IP ファンクション・シミュレーション・モデル IP
- 柔軟なライセンス供与 — 必要な機能のみを使用
- OpenCore Plus 評価をサポート

## リリース情報

表 1-1 に、Viterbi コンパイラのリリースに関する情報を示します。

表 1-1. Viterbi コンパイラのリリース情報

項目	説明
バージョン	11.0
リリース月	2011 年 5 月
注文コード	IP-VITERBI/HS (パラレル・アーキテクチャ) IP-VITERBI/SS (ハイブリッド・アーキテクチャ)
プロダクト ID	0037 (パラレル・アーキテクチャ) 0038 (ハイブリッド・アーキテクチャ)
ベンダ ID	6AF7

 このリリース情報について詳しくは、[「MegaCore IP Library Release Notes and Errata」](#) を参照してください。

アルテラは、Quartus® II ソフトウェアの現行バージョンが、各 MegaCore ファンクションの前のバージョンをコンパイルできることを検証しています。[「MegaCore IP Library Release Notes and Errata」](#) では、この検証に例外がある場合に報告します。アルテラは、1 リリースより前の MegaCore ファンクション・バージョンのコンパイルは検証していません。

## サポートされるデバイス・ファミリ

表 1-2 では、アルテラ IP コアのデバイス・サポート・レベルを定義します。

表 1-2. アルテラ IP コアのデバイス・サポート・レベル

FPGA デバイス・ファミリ	HardCopy デバイス・ファミリ
<b>暫定サポート</b> — この IP コアは、デバイス・ファミリの暫定タイミング・モデルに対して検証済みです。このコアはすべての機能要件を満たすが、メガファンクションに対し、デバイス・ファミリについてのタイミング解析がまだ行われています。製造デザインでの使用は注意が必要です。	<b>HardCopy コンパニオン</b> — この IP コアは、HardCopy コンパニオン・デバイスの暫定タイミング・モデルに対して検証済みです。IP コアが機能要件をすべて満たしているが、タイミング要件については評価中です。生産デザインでの使用は注意が必要です。
<b>最終サポート</b> — この IP コアは、デバイス・ファミリの最終タイミング・モデルに対して検証済みです。IP コアがデバイス・ファミリの機能要件およびタイミング要求値をすべて満たしており、生産デザインで使用可能です。	<b>HardCopy コンパイル</b> — この IP コアは、HardCopy デバイス・ファミリの最終タイミング・モデルに対して検証済みです。IP コアがデバイス・ファミリの機能要件およびタイミング要求値をすべて満たしており、生産デザインで使用可能です。

表 1-3 に、Viterbi MegaCore ファンクションによる各アルテラ・デバイス・ファミリへのサポートのレベルを示します。

表 1-3. サポートされるデバイス・ファミリ ( 1 / 2 )

デバイス・ファミリ	サポート
Arria® GX	最終
Arria II GX	最終
Arria II GZ	最終
Cyclone®	最終


表 1-3. サポートされるデバイス・ファミリ ( 2 / 2 )

デバイス・ファミリ	サポート
Cyclone II	最終
Cyclone III	最終
Cyclone III LS	最終
Cyclone IV GX	最終
HardCopy® II	HardCopy コンパイル
HardCopy III	HardCopy コンパイル
HardCopy IV E	HardCopy コンパイル
HardCopy IV GX	HardCopy コンパイル
Stratix®	最終
Stratix II	最終
Stratix II GX	最終
Stratix III	最終
Stratix IV GT	最終
Stratix IV GX/E	最終
Stratix V	暫定
Stratix GX	最終
その他のデバイス・ファミリ	サポートなし

## パフォーマンスおよびリソース使用率

ここでは、以下のデバイスで Quartus II ソフトウェアを使用し、さまざまなアーキテクチャおよび拘束長 (L)、組合せおよび ACS ユニット (A) を使用する場合の標準的な期待パフォーマンスを示します。

- Cyclone III (EP3C10F256C6)
- Stratix III (EP3SE50F780C2)
- Stratix IV (EP4SGX70DF29C2X)

 性能は主に拘束長 (L) に依存します。

## ハイブリッド・アーキテクチャ

表 1-4 ~ 表 1-6 に、BER オプションおよび次のパラメータを使用したハイブリッド・アーキテクチャのパフォーマンスを示します。

$$v = 6 \times L$$

$$\text{softbits} = 3$$

$$N = 2$$

ここで、

v はトレースバック長です。  
L は拘束長です。  
N はコード化されたビットの数です。  
A は ACS ユニットの数です。

表 1-4. ハイブリッド・アーキテクチャのパフォーマンスおよびエリア利用率 — Cyclone III デバイス

パラメータ		組み合わせ LUT 数	ロジック・ レジスタ	メモリ・ ブロック (M9K)	fMAX (MHz)	スループット (Mbps)
A	L					
1	5	605	387	5	193	19
1	7	825	502	6	197	6
2	7	977	619	6	191	12
4	7	1,259	833	6	185	19
1	9	1,577	922	12	188	1
2	9	1,730	1,047	12	185	3
4	9	2,044	1,277	12	178	6
8	9	2,653	1,723	14	174	11
16	9	3,807	2,585	18	166	17

表 1-5. ハイブリッド・アーキテクチャのパフォーマンスおよびエリア利用率 — Stratix III デバイス

パラメータ		組み合わせ LUT 数	ロジック・ レジスタ	メモリ・ ブロック (M9K)	fMAX (MHz)	スループット (Mbps)
A	L					
1	5	542	387	5	327	33
1	7	730	502	6	330	10
2	7	889	620	6	341	21
4	7	1,127	833	6	323	32
1	9	1,419	922	12	312	2
2	9	1,582	1,047	12	303	5
4	9	1,896	1,277	12	318	10
8	9	2,466	1,723	14	298	19
16	9	3,487	2,587	18	297	30

表 1-6. ハイブリッド・アーキテクチャのパフォーマンスおよびエリア利用率 — Stratix IV デバイス ( 1 / 2 )

パラメータ		組み合わせ LUT 数	ロジック・ レジスタ	メモリ		fMAX (MHz)	スループット (Mbps)
A	L			ALUT	M9K		
1	5	548	407	4	4	331	33
1	7	736	526	6	5	328	10
2	7	894	643	6	5	337	21
4	7	1,134	857	6	5	319	32
1	9	1,459	978	24	10	312	2

表 1-6. ハイブリッド・アーキテクチャのパフォーマンスおよびエリア利用率—  
Stratix IV デバイス ( 2 / 2 )

パラメータ		組み合わせ LUT 数	ロジック・ レジスタ	メモリ		fMAX (MHz)	スルー プット (Mbps)
A	L			ALUT	M9K		
2	9	1,622	1,103	24	10	307	5
4	9	1,936	1,333	24	10	310	10
8	9	2,509	1,780	24	12	285	18
16	9	3,526	2,643	24	16	293	29

## パラレル・アーキテクチャ

表 1-7 ~ 表 1-9 に、BER オプションを使用することなく、次のオプションを使用したパラレル・アーキテクチャのパフォーマンスを示します。

$$v = 6 \times L$$

$$N = 2$$

ここで、

v はトレースバック長です。

L は拘束長です。

N はコード化されたビットの数です。

表 1-7. パラレル・アーキテクチャのパフォーマンスおよびエリア利用率—Cyclone III デバイス

パラメータ				組み合わせ LUT 数	ロジック・レジ スタ数	メモリ・ ブロック (M9K)	fMAX (MHz)	スルー プット (Mbps)
ソフト ビット	L	最適化	最適状態 ファイ ン ダー					
7	3	ブロック	オフ	2,218	847	5	184	184
7	2	連続的	オフ	2,048	814	5	181	181
3	3	なし	オフ	725	436	5	211	211
5	3	なし	オフ	1,117	574	5	198	198
7	1	なし	オフ	2,218	964	7	198	198
7	3	なし	オフ	2,624	1,108	7	187	187
7	4	なし	オフ	2,825	1,180	7	181	181
3	3	なし	オン	755	464	5	205	205
5	3	なし	オン	1,275	720	5	200	200
7	3	なし	オン	3,307	1,732	7	188	188

表 1-8. パラレル・アーキテクチャのパフォーマンスおよびエリア利用率 —Stratix III デバイス

パラメータ				組み合わせ LUT 数	ロジック・レジスタ数	メモリ・ブロック (M9K)	fMAX (MHz)	スループット (Mbps)
ソフトビット	L	最適化	最適状態 ファイナダー					
7	3	ブロック	オフ	2,059	848	5	281	281
7	2	連続的	オフ	2,015	816	9	279	279
3	3	なし	オフ	548	437	5	327	327
5	3	なし	オフ	918	574	5	307	307
7	1	なし	オフ	2,013	970	7	292	292
7	3	なし	オフ	2,401	1,109	7	285	285
7	4	なし	オフ	2,596	1,180	7	285	285
3	3	なし	オン	565	464	5	326	326
5	3	なし	オン	1,092	723	5	308	308
7	3	なし	オン	3,082	1,732	7	277	277

表 1-9. パラレル・アーキテクチャのパフォーマンスおよびエリア利用率 —Stratix IV デバイス

パラメータ				組み合わせ LUT 数	ロジック・レジスタ数	メモリ		fMAX (MHz)	スループット (Mbps)
ソフトビット	L	最適化	最適状態 ファイナダー			ALUT	M9K		
7	3	ブロック	オフ	2,058	847	--	5	285	285
7	2	連続的	オフ	2,015	815	--	9	292	292
3	3	なし	オフ	606	523	40	2	345	345
5	3	なし	オフ	942	608	16	4	316	316
7	1	なし	オフ	2,044	1,012	24	6	292	292
7	3	なし	オフ	2,436	1,153	24	6	289	289
7	4	なし	オフ	754	545	6	5	311	311
3	3	なし	オン	624	551	40	2	341	341
5	3	なし	オン	1,115	756	16	4	314	314
7	3	なし	オン	3,117	1,777	24	6	288	288

## インストールおよびライセンス

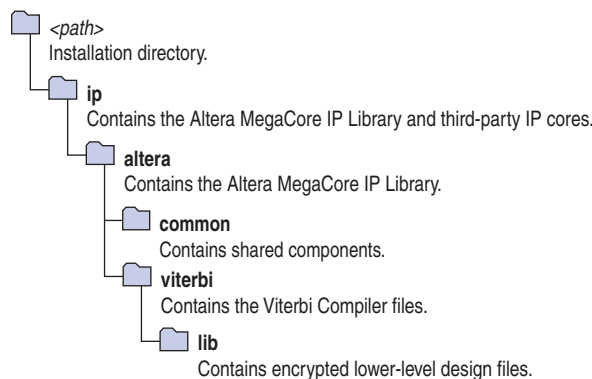
Viterbi コンパイラは、MegaCore® IP ライブラリの一部であり、Quartus® II ソフトウェアとともに配布されます。また、アルテラのウェブサイト ([www.altera.co.jp](http://www.altera.co.jp)) からダウンロードすることもできます。



システム要件とインストール手順については、『[Altera Software Installation and Licensing](#)』マニュアルを参照してください。

図 1-1 に、Viterbi コンパイラをインストールした後のディレクトリ構造を示します。この場合、`<path>` がインストール・ディレクトリです。Windows でのデフォルトのインストール・ディレクトリは、`c:\altera\<version>` です。Linux では、`/opt/altera<version>` です。

図 1-1. ディレクトリ構造



## OpenCore Plus 評価機能

アルテラの無償 OpenCore Plus 評価機能により、以下の処理を実行できます。

- 作成したシステム内のメガファンクション（アルテラ MegaCore ファンクションまたは AMPPSM メガファンクション）の動作をシミュレーションする。
- デザインの機能を検証したり、サイズやスピードを迅速かつ簡単に評価したりする。
- メガファンクションを含むデザインに対し、時間制限付きのデバイス・プログラミング・ファイルを生成する。
- デバイスをプログラムし、デザインを実機上で検証する。

Viterbi コンパイラのライセンスは、お客様が機能と性能に満足し、かつデザインを製品化する場合にのみ、ご購入いただく必要があります。ライセンス購入後は、アルテラ・ウェブサイト（[www.altera.co.jp/licensing](http://www.altera.co.jp/licensing)）からライセンス・ファイルを要求して、コンピュータにインストールできます。ライセンス・ファイルを要求すると、アルテラから電子メールで **license.dat** ファイルが送信されます。インターネットをご利用いただけないお客様は、アルテラの販売代理店にお問い合わせください。



OpenCore Plus ハードウェア評価について詳しくは、[「AN320: OpenCore Plus Evaluation of Megafunctions」](#) を参照してください。

## OpenCore Plus タイム・アウト動作

OpenCore Plus ハードウェア評価機能は、以下の動作モードでメガファンクションの実機評価をサポートします。

- *Untethered*（アンテザード） — デザインは限定時間のみ実行されます。

- **Tethered** (テザード) — ボードとホスト・コンピュータ間に接続が必要です。デザイン内のすべてのメガファンクションが **Tethered** モードをサポートしている場合、デバイスはより長時間または無制限に動作できます。

**OpenCore Plus** 評価機能モードで動作中のメガファンクションのうちの 1 つでも評価時間に達すると、デバイス内のすべてのメガファンクションが同時にタイムアウトします。デザイン内に複数のメガファンクションがある場合、特定のメガファンクションのタイムアウト動作は、他のメガファンクションのタイムアウト動作によってマスクされることがあります。

**Viterbi** コンパイラの **Untethered** タイムアウトは 1 時間、**Tethered** タイムアウト値は無制限です。

ハードウェア評価期限経過後にデザインは動作を停止し、decbit 出力が **Low** に維持します。



## デザイン・フロー

Viterbi コンパイラは次のデザイン・フローをサポートします。

- **DSP Builder** : Viterbi コンパイラ・バリエーションを含む DSP Builder モデルを作成する場合、このフローを使用します。
- **MegaWizard™ Plug-In Manager** : パラメータを設定し、手動でデザイン内にインスタンス化できる Viterbi コンパイラのバリエーションを作成する場合、このフローを使用します。

この章では、これらのフローのいずれかに Viterbi コンパイラを使用する方法を説明します。各フローのパラメータは、同じオプションを提供し、5 ページの「MegaCore ファンクションのパラメータ化」で説明されています。

これらのフローのいずれかでデザインをパラメータ化およびシミュレーションを実行した後、完成したデザインを Quartus II ソフトウェアでコンパイルすることができます。

## DSP Builder フロー

アルテラの DSP Builder は、アルゴリズム開発の環境下で行った DSP デザインをハードウェアで実現するための支援を行い、デジタル信号処理 (DSP) のデザイン・サイクルを短縮します。

DSP Builder は、The MathWorks 社の MATLAB® および Simulink® システム・レベルのデザイン・ツールのアルゴリズム開発、シミュレーション、および検証の機能と、アルテラの Quartus II 開発ソフトウェアおよびサードパーティの合成およびシミュレーション・ツールを組み合わせます。既存の Simulink ブロックをアルテラの DSP Builder ブロックおよび MegaCore ファンクション・バリエーション・ブロックと組み合わせて、システム・レベル仕様を検証し、シミュレーションを実行することができます。

DSP Builder では、MegaCore ファンクションの Simulink シンボルが、アルテラの DSP Builder ブロックセットから MegaCore ファンクション・ライブラリの Simulink ライブラリ・ブラウザに表示されます。

MATLAB/Simulink 環境で Viterbi コンパイラを使用するには、次の手順に従います。

1. 新規 Simulink モデルを作成します。
2. MegaCore ファンクション・ライブラリから viterbi\_<version> ブロックを選択し、モデルに追加し、このブロックにユニックな名前を付けます。
3. モデル内の viterbi\_<version> をダブル・クリックしてパラメータ・エディタを表示させ、MegaCore ファンクション・バリエーションをパラメータ化します。Viterbi コンパイラにパラメータを設定する例については、5 ページの「MegaCore ファンクションのパラメータ化」を参照してください。
4. パラメータ・エディタで **Finish** をクリックし、Viterbi コンパイラ MegaCore ファンクション・バリエーションを生成します。生成されたファイルについて詳しくは、12 ページの表 2-1 を参照してください。

5. Viterbi コンパイラをモデル内のほかのブロックに接続します。
6. DSP Builder モデル内の MegaCore ファンクション・バリエーションをシミュレートします。



DSP Builder フローについて詳しくは、[「DSP Builder User Guide」](#)の「Using MegaCore Functions」の章を参照してください。



DSP Builder フローを使用するとき、デバイスの選択、シミュレーション、Quartus II コンパイル、およびデバイス・プログラミングはすべて DSP Builder 環境で制御されます。

DSP Builder は、Avalon® Memory-Mapped (Avalon-MM) および Avalon ストリーミング (Avalon-ST) ソース/シンク・インタフェースによって、SOPC との統合をサポートします。



これらのインタフェースの種類について詳しくは、[「Avalon Interface Specifications」](#)を参照してください。

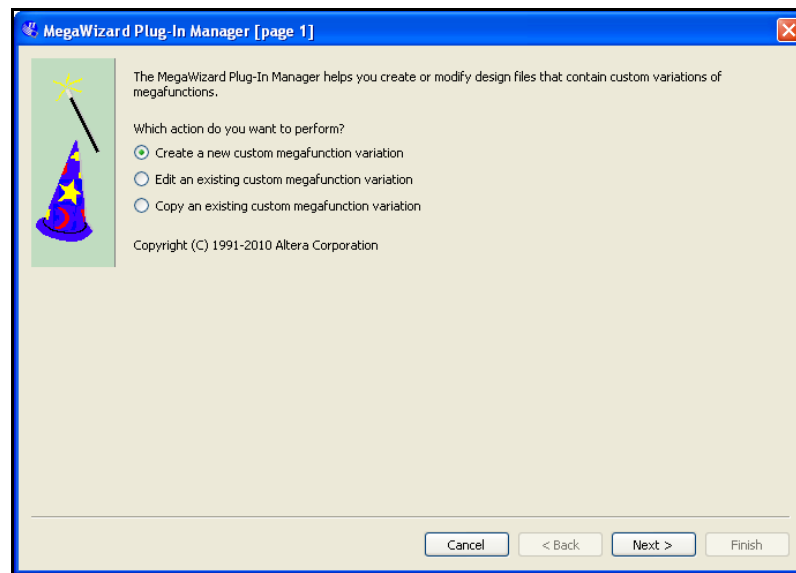
## MegaWizard Plug-In Manager フロー

MegaWizard™ Plug-In Manager フローでは、Viterbi コンパイラ MegaCore ファンクション・バリエーションをカスタマイズし、手動で Quartus II デザインに組み込むことができます。

MegaWizard Plug-in Manager フローを使用するには、以下のステップに従います。

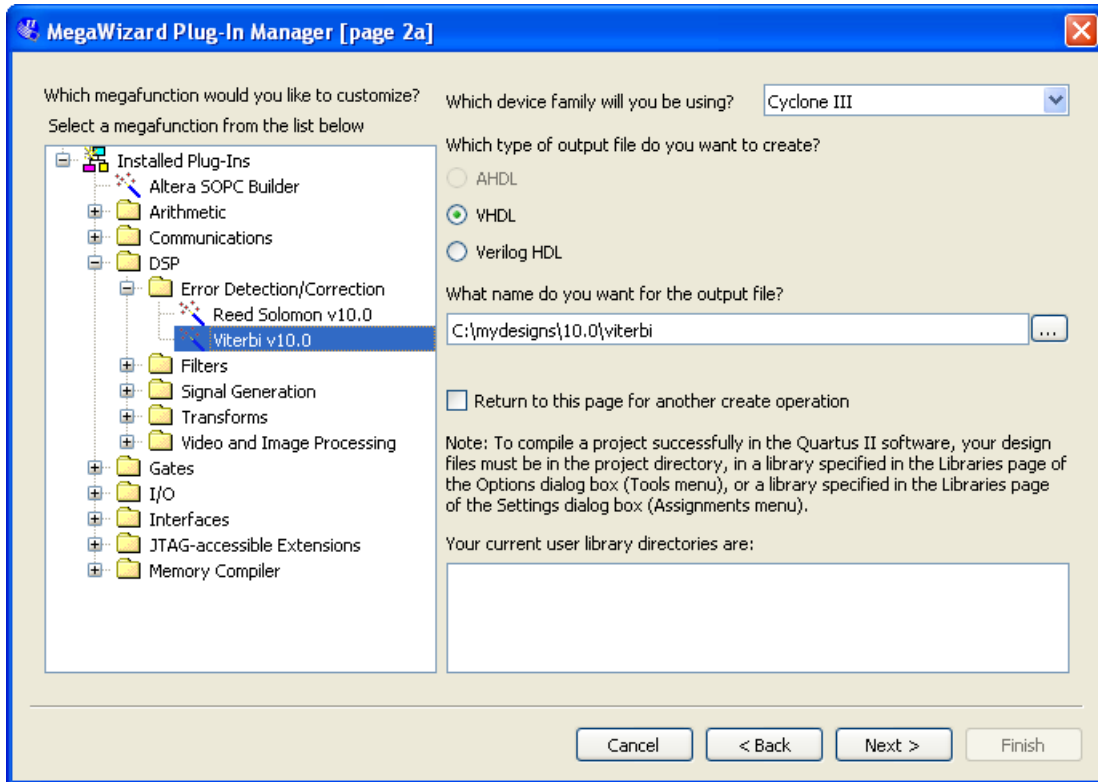
1. File メニューの **New Project Wizard** を使用して、新規のプロジェクトを作成します。
2. Tools メニューの **MegaWizard Plug-in Manager** を実行し、新規のカスタム・メガファンクション・バリエーションを作成するオプションを選択します (図 2-1)。

図 2-1. MegaWizard Plug-In Manager



3. Click **Next** をクリックし、**Installed Plug-Ins** タブの **DSP >Error Detection/Correction** セクションから **Viterbi <version>** を選択します。
4. デバイス・ファミリが、**New Project Wizard** で指定したものと同一であることを確認します。
5. デザインのトップレベル出力ファイル・タイプを選択します。ウィザードでは VHDL と Verilog HDL をサポートしています。
6. MegaWizard Plug-In Manager に、**New Project Wizard** で指定したプロジェクト・パスが表示されます。**MegaCore** ファンクション出力ファイル **<project path>\<variation name>** のバリエーション名を追加します。図 2-2 に、これらの設定を行った後のウィザードを示します。

図 2-2. メガファンクションの選択



7. **Next** をクリックして、IP Toolbench を起動します。

## MegaCore ファンクションのパラメータ化

MegaCore ファンクションをパラメータ化するには、以下のステップを実行します。

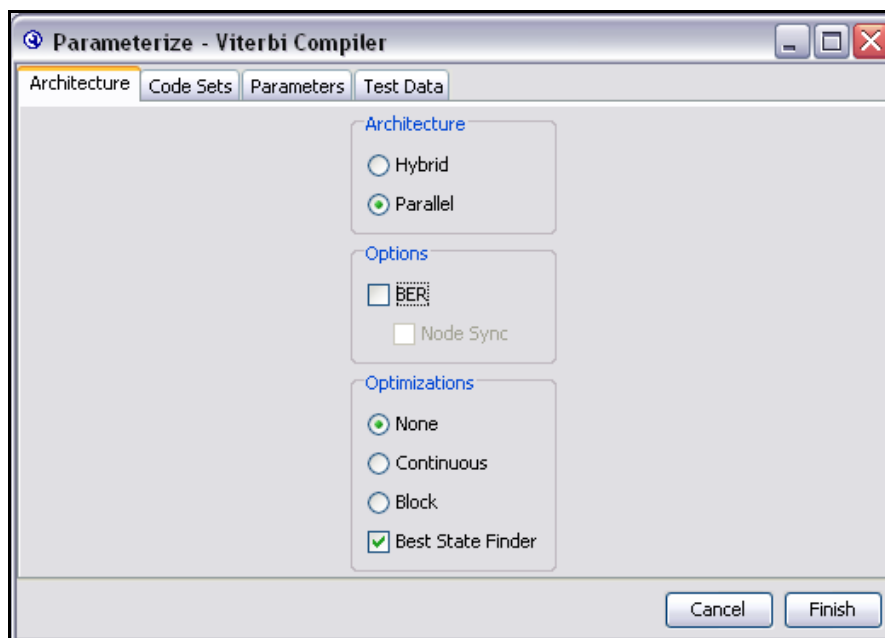
1. IP Toolbench で **Step 1: Parameterize** をクリックします (図 2-3)。

図 2-3. IP Toolbench—Parameterize




2. アーキテクチャを選択します。**Hybrid** または **Parallel** のいずれかを選択します (図 2-4)。

図 2-4. Selecting the Architecture

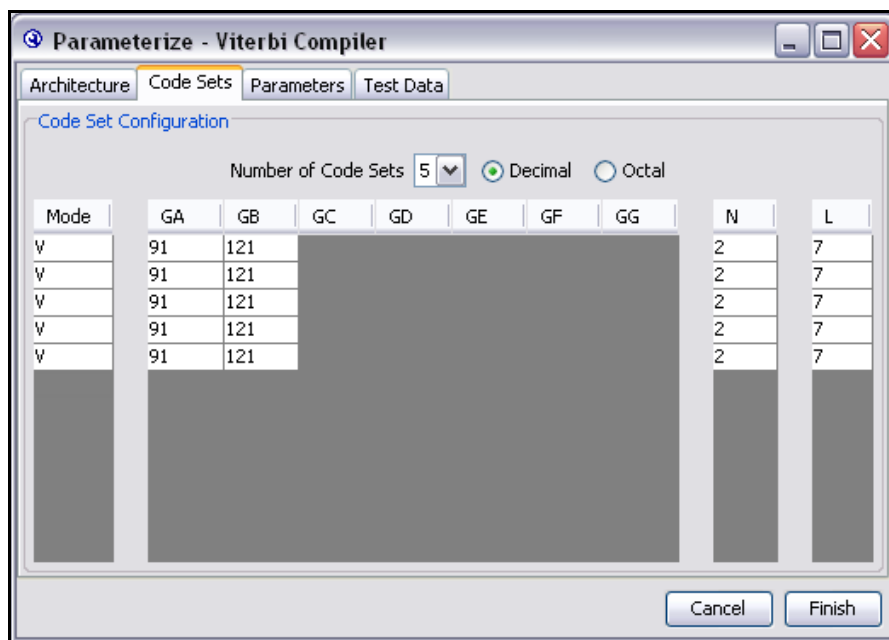


3. 必要とするオプションをオンにします。**BER** をオンにする場合、**Node Sync** をオンにすることができます (図 2-4)。


 **BER** について詳しくは、9 ページの「BER 見積もりツール」を参照してください。

4. パラレル・アーキテクチャの場合、次の最適化オプションのいずれかを選択できます。
- **None**— コアは連続デコードおよびブロック・デコードを組み合わせで使用します。このオプションでのみ、**Best State Finder** オプションをオンにすることができます。
  - **Block**— このオプションは、ブロックの全サイズを格納するメモリ付きのトレースバック・エンジンを実装します。
  - **Continuous**— このオプションは固定なトレースバック長を実装し、アーキテクチャのサイズを低減します。
5. **Code Sets** タブをクリックします (図 2-5)。

図 2-5. コード・セットのコンフィギュレーション

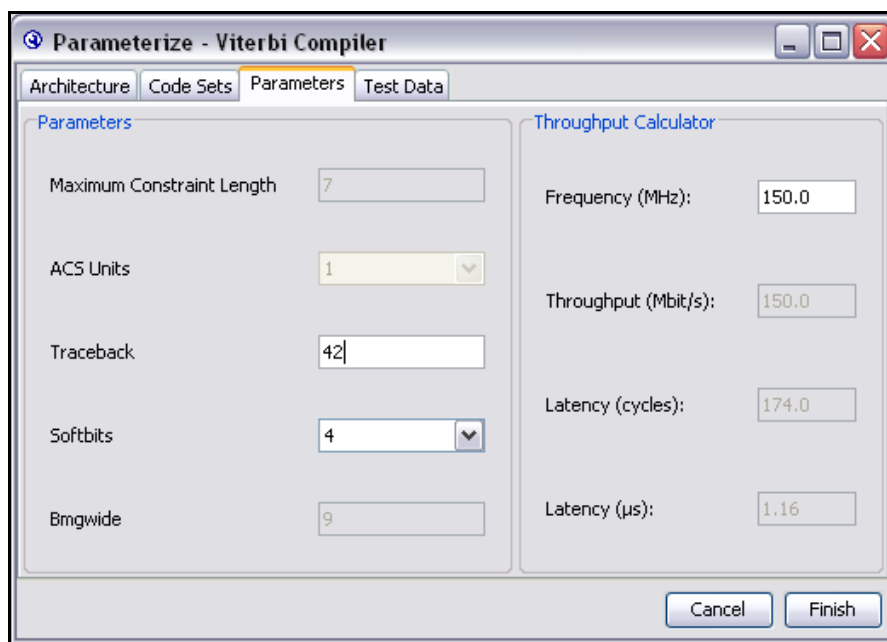


6. 以下の手順で、必要とするコード・セット情報を入力します。
  - a. **Number of Code Sets** を選択します。複数のコード・セットを使用するために、1以上の値を選択します。
  - b. **Decimal** (十進数) または **Octal** (8進数) を選択します。
  - c. **Viterbi** モード (**V**) または **TCM** (trellis coded modulation) モード (**T**) を選択します。
  - d. 必要とされる多項式 (**GA**、**GB**、**GC**、**GD**、**GE**、**GF**、および **GG**) の値を入力します。
  - e. コード化されたビットの数 (**N**) の値を入力します。
  - f. コード・セットの拘束長 (**L**) を入力します。


 複数のコード・セットについて詳しくは、11ページの「Code Sets タブ」を参照してください。

7. **Parameters** タブをクリックします (図 2-6)。


図 2-6. パラメータの選択



8. 以下の手順で、実装したい特定の **Viterbi** コードを定義するパラメータを選択します。
  - a. ハイブリッド・アーキテクチャでのみ、**ACS Units** で **ACS** ユニットの数を選択します。
  - b. **Traceback** で、トレースバック長を入力します。
  - c. **Softbits** で、ソフトビットの数を選択します。

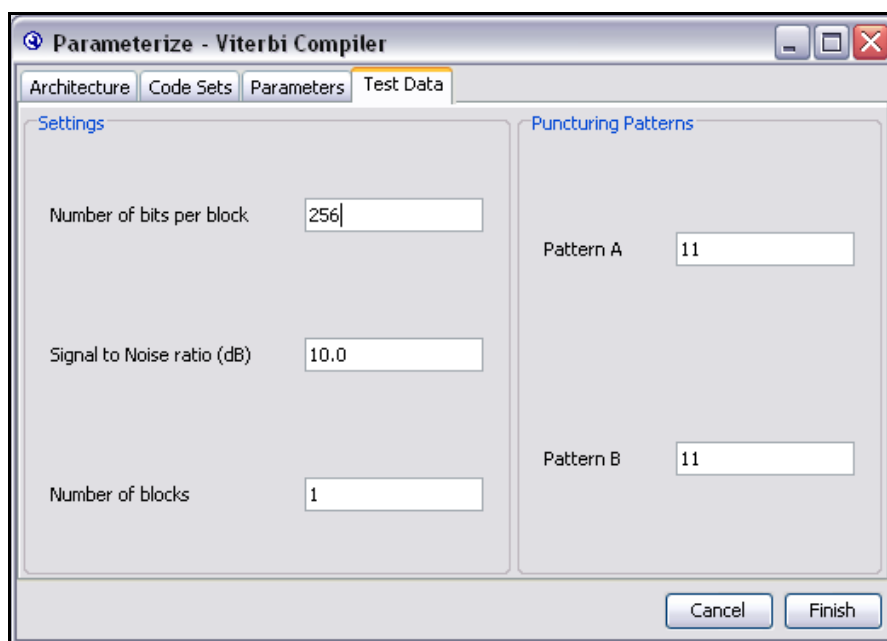
 **Code Sets** タブで指定された最大拘束長 (L) は、読み出し専用のフィールドとして表示されます。**Bmgwide** フィールドは、**N**、**L** および **Softbits** の最大値から算出されたステート・メトリック累積の精度を表示します。

9. スループット・カリキュレータに値を入力します (図 2-6)。スループット・カリキュレータは、特定の周波数に対してスループットを計算します。


 これらのパラメータについて詳しくは、12 ページの「Parameters タブ」を参照してください。スループット・カリキュレータに使用される式については、13 ページの「スループット・カリキュレータ」を参照してください。レイテンシ・カリキュレータに使用される式については、13 ページの「レイテンシ・カリキュレータ」を参照してください。

10. **Test Data** タブをクリックします (8 ページの図 2-7)。


図 2-7. テスト・データの設定



11. テストベンチのテスト・データの設定を入力します。

 IP Toolbench は VHDL テストベンチを生成します。この VHDL テストベンチは、アルテラがサポートする任意の VHDL シミュレータに使用できます。このテストベンチは、このタブで指定されたデータを使用します。

- a. **Number of bits per block** に、ブロックあたりのビット数を入力します。最小値は拘束長と同じです。
- b. **Signal to Noise ratio** に、dB 単位で信号対ノイズ比を入力します。
- c. **Number of blocks** に、ブロック数を入力します。
- d. **Puncturing Patterns** に、パンクチャリング・パターンを入力します。テスト用のデパンクチャされたデータを指定することができます。

 テスト・データのパラメータについては、14 ページの「Test Data タブ」を参照してください

12. **Finish** をクリックします。

## シミュレーションの設定

IP 機能シミュレーション・モデルは、Quartus II ソフトウェアで生成するサイクル精度の正確な VHDL または Verilog HDL モデルです。このモデルにより、業界標準の VHDL および Verilog HDL シミュレータを使用した IP の高速機能シミュレーションが可能になります。



これらのシミュレーション・モデル出力ファイルは、シミュレーション目的にのみ使用することができ、特に合成やその他の目的には使用できません。これらのモデルを合成に使用すると、機能しないデザインが作成されます。

MegaCore ファンクション用の IP 機能シミュレーション・モデル (<variation name>.vo または <variation name>.vho) を生成するには、以下のステップに従います。



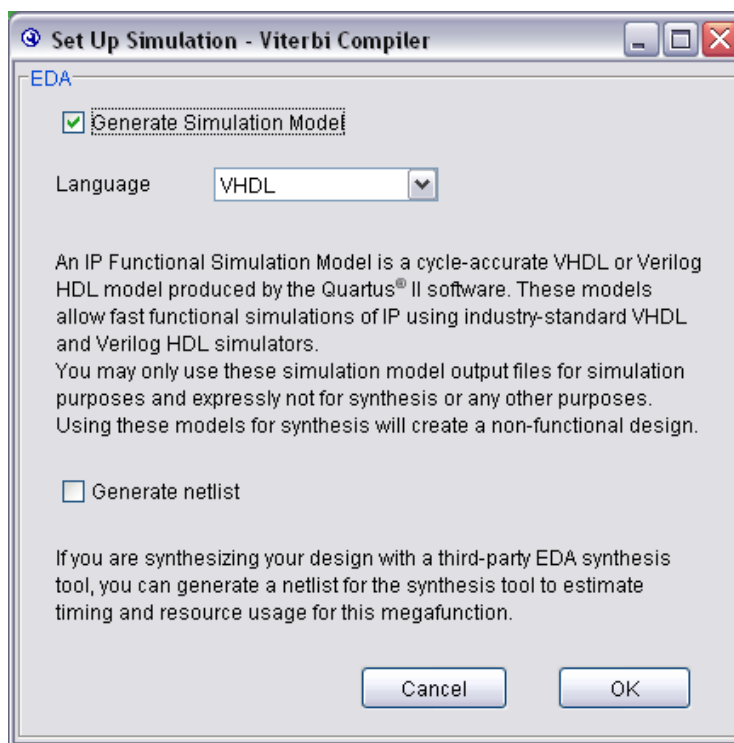
1. IP Toolbench の **Step 2: Set Up Simulation** をクリックします (5 ページの  2-3)。
2. **Generate Simulation Model** をオンにします (  2-8)。

図 2-8. シミュレーション・モデルの生成



3. **Language** ドロップダウン・ボックスで言語を指定します。

4. 一部サードパーティ合成ツールでは、詳細なロジックは含まず MegaCore ファンクションの構造のみを含むネットリストを使用して、MegaCore ファンクションを含むデザインの性能を最適化することができます。合成ツールでこの機能がサポートされている場合、**Generate netlist** をオンにします。
5. **OK** をクリックします。

## MegaCore ファンクションの生成

MegaCore ファンクションを生成するには、次のステップに従います。

1. IP Toolbench の **Step 3: Generate** をクリックします (5 ページの図 2-3)。

ファイル生成フェーズを完了するには、数分かかかる場合があります。生成の進行状況およびステータスは、レポート・ウィンドウで表示されます。

11 ページの図 2-9 に、生成レポートを示します。

図 2-9. 生成レポート (注 1)

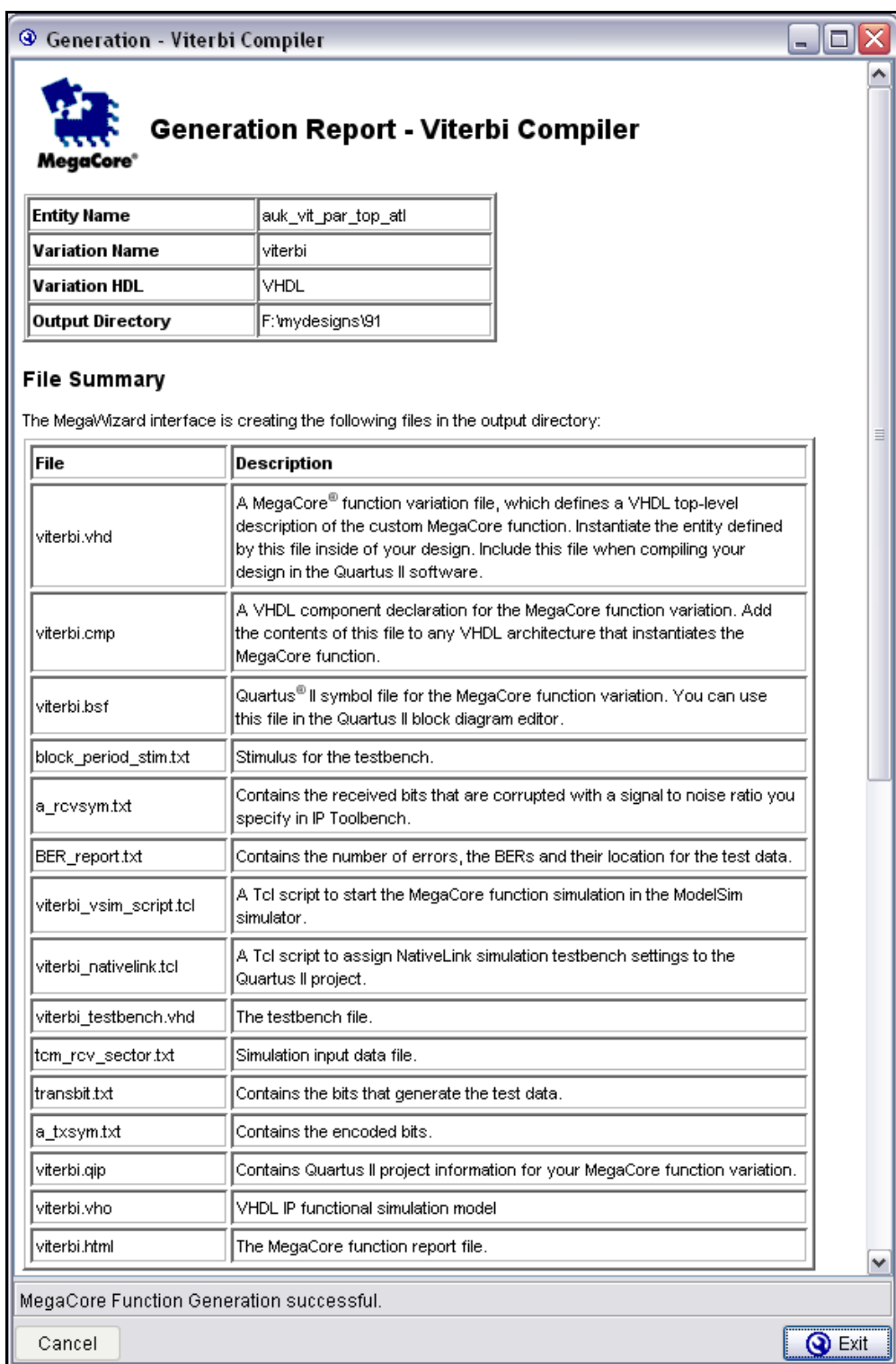


図 2-9 の注 :

- (1) **Entity Name** が自動的に追加されます。パラレル・アーキテクチャの場合、\_par が追加されます。ハイブリッド・アーキテクチャの場合、\_hyb が追加されます。

表 2-1 に、プロジェクト・ディレクトリに生成されるファイルを示します。IP Toolbench レポートに指定されるファイルの名前とタイプは、デザインを VHDL または Verilog HDL のいずれで作成したかによって異なります。


表 2-1. 生成されるファイル (注 1)

ファイル名	説明
<variation name>.bsf	MegaCore ファンクションのバリエーション用 Quartus II シンボル・ファイル。Quartus II ブロック図エディタでこのファイルを使用できます。
<variation name>.cmp	MegaCore ファンクション・バリエーション用の VHDL コンポーネント宣言ファイルです。このファイルの内容を、MegaCore ファンクションをインスタンスする VHDL アーキテクチャの 1 つに追加します。
<variation name>.html	HTML フォーマットの MegaCore ファンクション・レポート・ファイル。
<variation name>.qip	Quartus II IP ファイルが生成されます。このファイルには、Quartus II コンパイラで MegaCore ファンクション・バリエーションを処理するのに必要なすべてのアサインメントおよび他の情報が含まれています。MegaWizard を終了する時に、この .qip ファイルを現行 Quartus II プロジェクトに追加するよう促されます。
<variation name>.vho または .vo	VHDL または Verilog HDL の IP 機能シミュレーション・モデルです。
<variation name>.vhd または .v	カスタム MegaCore ファンクションの VHDL または Verilog HDL トップレベルの記述を定義する MegaCore ファンクション・バリエーション・ファイルです。デザイン内部のこのファイルによって定義されたエンティティをインスタンスします。QuartusII ソフトウェアでのデザインのコンパイル時にこのファイルがインクルードされます。
<variation name>_nativelink.tcl	Tcl スクリプトです。デザインを指定の EDA ツールでネイティブにシミュレートするように Quartus II の NativeLink を設定します。
<variation name>_syn.vhd または _syn.v (2)	一部のサードパーティ合成ツールで使用できるオプションのタイミングおよびリソース・ネットリストです。
<variation name>_testbench.vhd	テストベンチです。
<variation name>_vsim_script.tcl	ModelSim シミュレータの MegaCore ファンクションを起動します。
a_rcvsym.txt	IP Toolbench で指定された信号対ノイズ比に破損された受信ビットが含まれています。
a_txsym.txt	エンコードされたビットを含みます。
BER_report.txt	テストデータのエラー数、BER、およびエラー位置が含まれています。
block_period_stim.txt	1 ブロックごとに変化するテストベンチ・ステイミュラス。
tcm_rcv_sector.txt	テストベンチをデコードするための TCM コードのセクタ数が含まれています。TCM コードが定義されていない場合、このファイルは空になります。
transbit.txt	テスト・データを生成するビットが含まれています。

表 2-1 の注：

- (1) <variation name> プリフィックスは、IP Toolbench によって自動的に生成されます。
- (2) \_syn.vhd または \_syn.v file は、IP Toolbench インタフェースの Set Up Simulation ページでイネーブルされる場合にのみ生成されます。


2. 生成レポートを表示した後、**Exit** をクリックして IP Toolbench を閉じます。  
**Quartus II IP Files** ウィンドウで **Yes** をクリックし、カスタム MegaCore ファンクションを記述する .qip ファイルを現行 Quartus II プロジェクトに追加します。

 MegaWizard Plug-In Manager について詳しくは、Quartus II ヘルプを参照してください。

これにより、カスタム・バリエーションをシステム・デザインに組み込み、シミュレーションおよびコンパイルを実行することができます。

## デザインのシミュレーション

シミュレーションは、NativeLink を使用して Quartus II ソフトウェアからサードパーティ製シミュレーション・ツールを使用して実行できます。

 NativeLink について詳しくは、「Quartus II ハンドブック Volume 3」の「*Simulating Altera Designs*」の章を参照してください。

Tcl スクリプト・ファイル `<variation name>_nativelink.tcl` を使用して、NativeLink テストベンチのデフォルト設定を Quartus II プロジェクトに割り当てることができます。

Quartus II ソフトウェアで NativeLink を使用してシミュレーションの設定を行うには、以下のステップを実行します。

1. カスタム・バリエーションを作成します。ただし、Quartus II プロジェクト名に一致するバリエーション名を指定してください。
2. サードパーティ・シミュレータ実行ファイルへの絶対パスが設定済みかどうかチェックします。Tools メニューで、**Options** をクリックし、**EDA Tools Options** を選択します。
3. Processing メニューで、**Start** をポイントして **Start Analysis & Elaboration** をクリックします。
4. Tools メニューの **Tcl Scripts** をクリックします。  
`<variation name>_nativelink.tcl` Tcl スクリプトを選択して、**Run** をクリックします。Tcl スクリプトが正常にロードされたことを確認するメッセージをチェックします。
5. Assignments メニューの **Settings** をクリックして、**EDA Tool Settings** を展開し、**Simulation** を選択します。Tool Name でシミュレータを選択します。
6. Tools メニューで **EDA Simulation Tool** をポイントして、**Run EDA RTL Simulation** をクリックします。

## デザインのコンパイル

Quartus II ソフトウェアを使用して、デザインをコンパイルすることができます。デザインのコンパイルに関するインストラクションについては、Quartus II Help を参照してください。


## デバイスのプログラミング

デザインをコンパイルした後、ターゲットのアルテラ・デバイスをプログラムし、ハードウェアでデザインを検証します。

アルテラの無償の OpenCore Plus 評価機能では、ライセンスを購入する前に、Viterbi コンパイラを評価できます。OpenCore Plus 評価機能により、IP 機能シミュレーション・モデルを生成し、期限付きのプログラミング・ファイルを作成できます。

-  IP 機能シミュレーション・モデルについて詳しくは、「Quartus II ハンドブック Volume 3」の「*Simulating Altera Designs*」の章を参照してください。

デザインで Viterbi コンパイラをシミュレーションし、ハードウェアにより期間限定でデザインの評価を実行できます。

-  OpenCore Plus ハードウェア評価について詳しくは、8 ページの「OpenCore Plus タイム・アウト動作」および「AN320: OpenCore Plus Evaluation of Megafunctions」を参照してください。

Viterbi デコーダは、連続ストリームおよびブロック・ストリームをデコードすることができます。Viterbi デコーダは通常、連続モードで動作します。

連続モードでは、デコーダはトレースバック長以上のシンボル数を処理するまで待機します。デコーダはトレースバック長で指定されたビット数をトレースバックした後に、出力ビットの送信を開始します。この動作は、連続モードに維持する限り続けますが、パケット終了 EOP (End of Packet) がアサートされると変更されます。その後、デコーダがブロック・モードに切り換わり、最後のシンボルまたはステートからトレースバックを開始します。tr\_init\_state 信号は、トレースバック動作を開始する終了ステートを示します。ブロック・デコードの場合、最後のビット (通常は 0) を示し、tb\_type ポートを 1 に設定することが推奨されています。

## ソフト・シンボル入力

シンボルあたりの軟判定ビット (softbits) は、 $2^{\text{softbits}} - 1$  のソフト 0 および  $2^{\text{softbits}} - 1$  のソフト 1 を表します。入力値は、受信された信号振幅を表します。入力が対数尤度フォーマットである場合、変換が必要であり、信号のインテグリティを維持するために追加のソフトビットが必要となります。デバンクチャされた値は個別にマークされます。softbits = 1 時に、デコーダは硬判定入力を許容します。

表 3-1 に、softbits = 3 時の軟判定シンボル入力の表現例を示します。

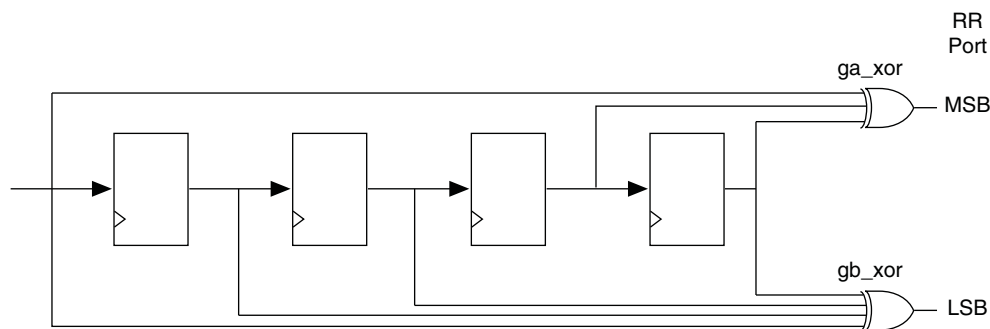
表 3-1. 軟判定入力の表現

ソフト・シンボル	意味 g
011	最も強い '0'
010	強い '0'
001	弱い '0'
000	最も強い '0'
111	最も弱い '1'
110	弱い '1'
101	強い '1'
100	最も強い '1'

## エンコード手法

2 ページの図 3-1 に、再帰的畳み込みエンコーダを示し、パラメータは  $L = 5$ 、 $N = 2$ 、そして多項式  $GA = 19$  および  $GB = 29$  です。GA は十進数で 19 であり、二進数では 10011 です。二進数表現の最上位ビットは入力データ・ビットでの接続です。最下位ビットはシフト・レジスタ・チェーンの最後の接続を表します。XOR ファンクションは、モジュロ 2 の加算を実装します。

図 3-1. エンコード手法



## ステート・メトリック

Viterbi デコーダ・ステート・メトリックは、累算メトリック (Euclidean ではない) であり、最小メトリックの代わりに最大メトリックに基づいています。メトリックは、成長につれて、オーバーフローを防止するために正規化する必要があります。正規化が発生するときに、デコーダはすべてのメトリックから  $2^{(bmgwide-1)}$  を減算し、normalization レジスタを +1 増加します。

ベスト・パスの合計メトリック値 = (正規化の数)  $\times$   $(2^{(bmgwide-1)}) + bestmet$

ベスト・パスの合計メトリック値、処理されたシンボル数、および BER ブロックの数は、チャンネルの質および softbits に適な値があるかどうかを示します。最高なメトリックをもたらすステートは bestadd で出力されます。

## パンクチャリング手法

パラレルおよびハイブリッド・アーキテクチャの両方は外部パンクチャリングをサポートします。示されたすべてのパンクチャされたコードは、レート 1/2 のマザーコードに基づいています。外部ディパンクチャリングの場合、デコーダの外部受信データ・ストリームをディパンクチャし、デコーダおよびシンボルにデータを同時に入力する必要があります。

表 3-2 に、一部の定義できる可能なパンクチャリング手法およびそのレートを示します。

表 3-2. 一部のパンクチャリング手法 ( 1 / 2 )

パンクチャ・レート	パンクチャリング手法				
	ビット (1)	乗数			
2/3	CA	1	0		
	CB	1	1		
3/4	CA	1	0	1	
	CB	1	1	0	
4/5	CA	1	0	0	0
	CB	1	1	1	1

表 3-2. 一部のパンクチャリング手法 ( 2 / 2 )

パンクチャ・レート	パンクチャリング手法							
	ビット ( <i>i</i> )	乗数						
5/6	CA	1	0	1	0	1		
	CB	1	1	0	1	0		
6/7	CA	1	0	0	1	0	1	
	CB	1	1	1	0	1	0	
7/8	CA	1	1	1	1	0	1	0
	CB	1	0	0	0	1	0	1

表 3-2 の注：

(1) CA は最上位（最初の送信ビット、最初の受信シンボル）を意味します。CB は最下位（最後の送信ビット、最後の受信シンボル）を意味します。

## トレリス符号化変調

トレリス符号化変調 (TCM) は、変調およびエンコード・プロセスを組み合わせ、帯域幅の増加なしで効率を向上させます。

帯域幅が制約されたチャンネルは、 $R/W > 1$  の領域で動作します。ここで、 $R$  はデータ・レートで、 $W$  は使用可能な帯域幅です。このようなチャンネルの場合、デジタル通信システムは、帯域幅を効率的に使用するマルチレベル位相変調を使用します。例えば、位相偏移変調 (PSK)、位相振幅変調 (PAM) または直交振幅変調 (QAM) です。

帯域幅が制約されたチャンネルに TCM を使用する場合、信号帯域幅を拡張することなく性能が向上します。信号位相の数が 4 から 8 に増加すると、同じエラー・レートを維持するためには、約 4dB の追加信号パワーが必要です。したがって、TCM がメリットを提供するために、レート 2/3 コードの性能向上はこの 4dB のペナルティより大きいでなければなりません。変調がエンコーディング・プロセスの不可欠な部分であり、符号化信号のペア間の最小 Euclidian 距離を増加させるためのコードと組み合わせで設計されている場合、信号セットの拡大による損失を容易に克服し、比較的単純なコードでコーディングを大幅に改善することができます。

帯域幅が制約されたシステムはすべてこの手法でメリットを得られます。例えば、衛星モデム・システムです。

TCM モードでのアルテラ Viterbi デコーダは、 $N = 2$  のみをサポートします (1/2 のマザー・コード・レートのみをサポート)。

1/2 レートの畳み込みコード (4 ページの図 3-2) で 1 つの情報ビットをエンコードし、第 2 の情報ビットをコードされていないままにする場合を見てみましょう。8 点の信号点 (例：8-PSK) と共に使用する場合、2 ビットが信号点内の 4 つのサブセットの 1 つを選択し、残りの情報ビットは各サブセット内の 2 ポイントの 1 つを選択します。

図 3-2. ハーフ・レートの畳み込みコード

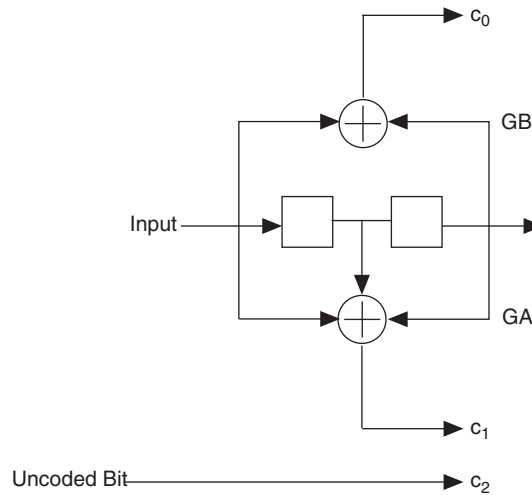


図 3-2 に、コードされたビットおよびセクタ番号のマッピングを示します。このマッピングは重要ではありません。サブセット間の最小距離の増加という主要な特性を維持するようにサブセットを並べ替えることで、ほかのマッピングを編み出すことができます。IP Toolbench およびテストベンチは図 3-2 に示すマッピング付きの TCM を作成します。ただし、8-PSK、16-PSK およびその他のためのシンボル・マッピングを含むほかのマッピングを作成することが可能です。


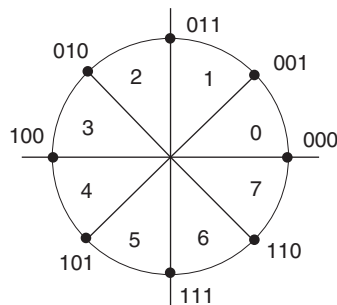
 ほかのマッピングを作成する場合、MegaCore ファンクションの外で作成されたブランチ・メトリックを入力ポートに正しく接続し、トレリスを生成するための多項式 GA および GB を正しくコンフィギュレーションする必要があります。

図 3-3. コードされたビットおよびセクタ番号のマッピング



4 ステートのトレリスは、1/2 レートの畳み込みエンコーダのトレリスです。このトレリスは、コードされていないビット  $c_2$  に対応する各伝送内に平行・パスを追加します。したがって、このデコーダはコードされたビット ( $c_1$ 、 $c_0$ ) を使用してそれぞれ 2 つの信号ポイントを含む 4 つのサブセットのいずれかを選択し、そしてコードされていないビットを使用して、各サブセット内の 2 つの信号ポイントのいずれかを選択することができます。

図 3-4. 4 ステートのトレリス

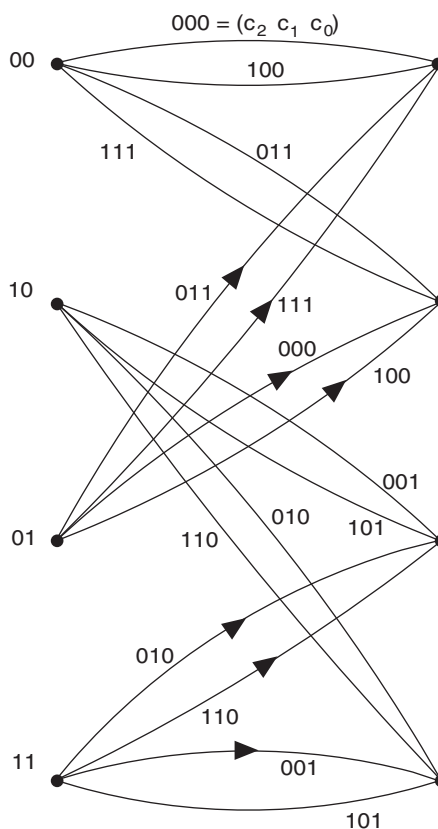


図 3-5 に、トレリス・デコーダとして実装された Viterbi デコーダを示します。デコーダは、受信されたシンボルを処理して、4つのブランチ・メトリックおよび1つのセクタ番号を取得します。ブランチ・メトリックは、トレリス・モードで Viterbi デコーダを入り、エンコードされたビットが取得されます。その後、このビット・ストリームが再エンコードされ、このエンコーダの出力がセクタ番号と共に使用され、コードされていないビットを取り出します。すべてのロジックは提供されたテストベンチで実装されます。

このブランチ・メトリック値およびセクタ番号が IP Toolbench によって生成されるため、これらの値を作成するロジックはありません。テストベンチは必要時にセクタ番号を読み出すため、遅延機能および回転がありません。IP Toolbench によって作成されるデータはエラーがないため、位相が一致します。ただし、実際のシステムでは、位相を計算する必要があります。

 TCM コードを使用する場合、BER ブロックが入力でエラーを計算しないため、BER ブロックは有意義な出力 (numerr) を生成しません。

図 3-5. トレリス・デコーダとして実装される Viterbi デコーダ

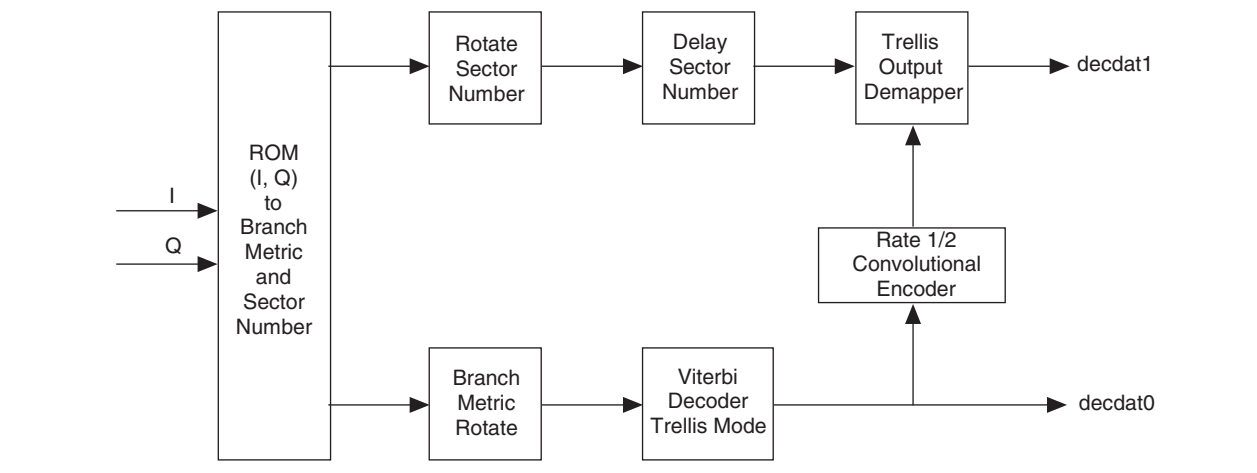
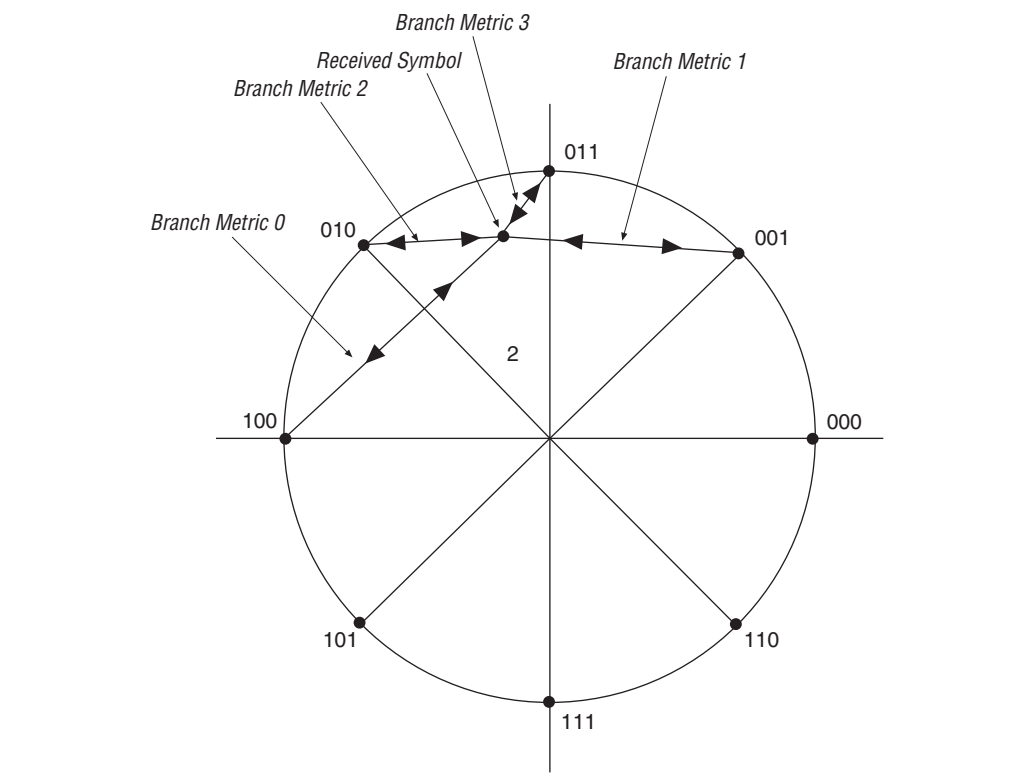


図 3-6 に、受信されたシンボルを 4 つのブランチ・メトリックおよびセクタ番号に変換されることを示します。デコーダは距離を、0...00 ~ 1...11 (ソフトビットの数) の符号なしの数として最も近い 4 つのシンボル・ポイントに計算します。ここで、範囲はシンボル・マップの範囲と同じです。デコーダが累積メトリック (Euclidean メトリックではない) を使用できるため、デコーダはこれらの距離を反転します (000 が 111 に、001 が 110 に)。

図 3-6. 受信されたシンボルを 4 つのブランチ・メトリックに変換



例えば、図 3-6 に、受信されたシンボルは 4 つの最も近いシンボル・マップ・ポイントへの距離でセクタ 2 番に着陸しました。

- 1111
- 1101
- 1011
- 0001

ここで、4 つのソフトビットの半径の距離は 1111 です。距離は次のブランチ・メトリックを取得するために反転されます。

- ブランチ・メトリック 0 = 0000
- ブランチ・メトリック 1 = 0010
- ブランチ・メトリック 2 = 0100
- ブランチ・メトリック 3 = 1110

デコーダはコードされたビット ( $c_1$ 、 $c_0$ ) によってブランチ・メトリック番号を選択します。この番号は、ブランチ・メトリックを Viterbi デコーダの `rr` 入力 of 何処に接続するかを決定します。ブランチ・メトリック 3 は `rr` の最上位ビット (MSB) になり、ブランチ・メトリック 0 は `rr` の最下位ビット (LSB) になります。

## トレリス終端

ブロック・デコードでは、ブロックの最後のビットを適切にデコードする手法を必要とします。この手法は、畳み込みエンコーダで発生するすべての状況に適合します。ここでは、2 種類の手法を説明します。

1 番目の手法では、畳み込みエンコーダは 1 つのブロックに供給され、そしてそのブロックの最後から取得される ( $L-1$ ) ビットによって終端されます。これらのビットは未知です。畳み込みエンコーダの初期状態は最後の ( $L-1$ ) 情報ビットによって設定されます。

この手法は「tail-biting」として呼ばれ、デコーダでブロックを複製するか、あるいはデコーダにブロックを 2 回供給することでデコードを行います。中間点でデコードすることで、トレリスは強制的に初期状態および最終状態の両方である状態になります。最初のデコード・ブロックから、ブロック後半のビットを取得でき、2 番目 (またはデコーダをパスする 2 番目) のデコード・ブロックから、ブロック前半のビットを取得できます。



「Tail-biting」手法では、デコーダをトレーニングするために、ブロックは十分なサイズを持つ必要があります。そうでない場合、BER の損失が発生します。

2 番目の手法では、畳み込みエンコーダは 0 に初期化されます。したがって、トレリスの初期状態が 0 であることが分かっています。畳み込みエンコーダへの ( $L-1$ ) ビットが分かっています。これで、畳み込みエンコーダは既知の終了状態に持ち込まれます。そしてデコーダはこの情報を使用して、`tr_init_state` でトレリスの終了状態を設定します。

The `tr_init_state` 信号は、ブロックの最後の ( $L-1$ ) ビットの反転から派生されます。

例えば、...000101 で終了するブロックの場合を考えてみましょう。

$L = 5$ 、最後の  $(L - 1) = 4$  ビットが分かっている場合、`tr_init_state` は 0101 に設定され、反転されると 2 進数では 1010、10 進数では 10 です。

IP Toolbench は、各ブロックの最後の  $(L - 1)$  ビットが分かっているように `tr_init_state` を生成します。

## トレリスの開始

パラレル・デコーダは、新しいブロックにステート 0 からトレリスを開始します。一方、ハイブリッドでは、`bm_init_state` で開始ステート（通常は 0）を設定することができます。この信号の範囲は  $0 \sim 2^{(L-1)-1}$  です。

`bm_init_value` 信号は、`bm_init_state` が指定したステートのステート・メトリックを開始します。ほかのすべてのステートは 0 から開始します。このポートに適切な値は約  $2^{(bmgwide-2)}$ 、または  $2^{(N+softbits)}$  から  $2^{(bmgwide-1)}$  までの任意の値です。



連続モードでは、ステート・メトリックはリセットしません。そのため、同じデータ・ブロックを複数回送信すると、差が発生する可能性があります。今回初めて、ステート・メトリックをステート 0 に設定するには 0 を入力します。これは、最初のステートが常にステート 0 であることを基づいています。将来のブロックに対しては、ステート・メトリックは前のブロックの終了時のものをすべて含みます。

## Avalon Streaming インタフェース

Avalon® Streaming (Avalon-ST) インタフェースは、ソース・インタフェースからシンク・インタフェースへのデータ転送に対して標準的な柔軟性の高いモジュラ式プロトコルを定義しており、データパスにおけるデータ・フローのコントロール・プロセスを簡略化します。Avalon-ST インタフェース信号は、チャンネルやパケット境界の概念のない従来の単一データ・ストリームをサポートします。このようなインタフェースは通常、`data`、`ready`、および `valid` 信号から構成されます。Avalon-ST インタフェースは、複数のチャンネルに渡ってインタリーブされたパケットでバーストとパケット転送のために、より複雑なプロトコルをサポートすることができます。Avalon-ST インタフェースは、マルチチャンネルのデザインを本質的に同期させます。これで、複雑な制御ロジックを実装することなく、効率的かつ時分割の実装が実現できます。

Avalon-ST インタフェースはバックプレッシャーをサポートします。バックプレッシャーはフロー制御の一種であり、シンクがソースにデータを送信することでデータ送信を停止することができます。シンクは通常、FIFO バッファがフルの場合、あるいは出力に輻輳が発生する場合にバックプレッシャーを使用してデータ・フローを停止します。Viterbi MegaCore ファンクションを含むデータ・パスの設計時に、ダウンストリーム・コンポーネントが常にデータを受信できることを知っている場合、バックプレッシャーの必要はありません。Viterbi のソース・レディ信号 `source_rdy w p High` にドライブし、シンク・レディ信号 `sink_rdy` 接続しないようにすると、より高いクロック・レートが達成可能です。



Avalon-ST インタフェース・プロトコルについて詳しくは、「[Avalon Streaming Interface Specifications](#)」を参照してください。

## パラメータ

このセクションでは、次のパラメータおよび製品オプションに関する情報について説明します。これらのオプションは **IP Toolbench** で設定できます (5 ページの「MegaCore ファンクションのパラメータ化」を参照)。

- Architecture タブ
- Parameters タブ
- Code Sets タブ
- Test Data タブ

### Architecture タブ

表 3-3 に、**Architecture** タブで設定可能なオプションを示します。

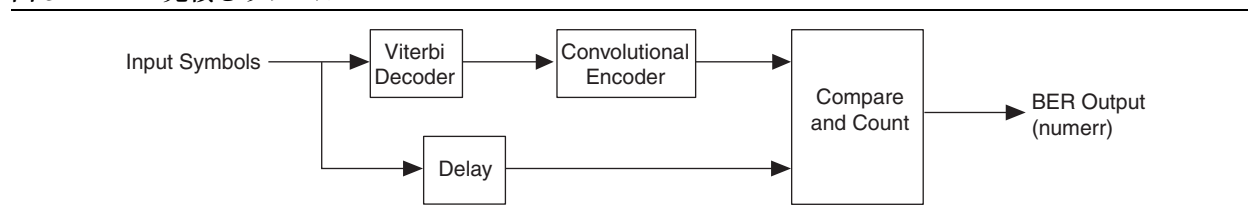
表 3-3. Architecture タブのオプション

パラメータ	値	説明
Hybrid or Parallel	-	ハイブリッドまたはパラレル・アーキテクチャの選択。
BER	On または Off	<b>BER</b> 見積もりツールのオプションを指定します (9 ページの「 <b>BER</b> 見積もりツール」を参照)。
Node Sync	On または Off	ノード同期化のオプションを指定します ( <b>BER option</b> がオンの場合にのみ使用可能)。
Optimizations	None、Continuous、または Block	パラレル・デコーダの最適化を指定します。 <b>None</b> を選択する場合、 <b>Best State Finder</b> をオンにすることができます。ただし、ロジックを節約するには、 <b>Best State Finder</b> をオフにします。

### BER 見積もりツール

図 3-7 に、BER 見積もりツールのブロック図を示します。

図 3-7. BER 見積もりツール



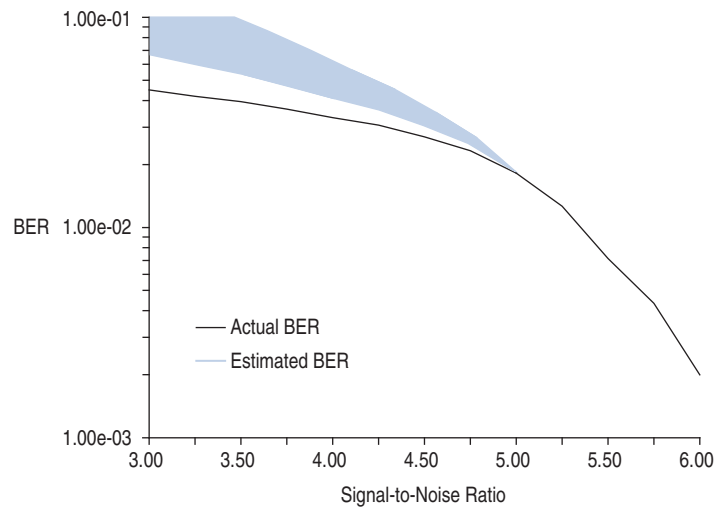
BER 見積もりツールは、再エンコード手法および比較手法を使用して、入力データでのエラー数を見積もります。信号対ノイズ比が十分に高く、デコーダがエラーなしの出力をデコードできる場合、BER の推定は実際のチャネル BER に非常に近いです。

デコーダがエラーなしの出力をデコードしていない場合、推定 BER は実際のチャネル BER より高く、ランダムです。これにより、出力エラーに正比例する不確実性が発生します (10 ページの図 3-8 を参照)。



TCM コードの場合、BER ブロックは TCM コードに対して入力のエラーを計算しないため、有意義な出力 (numerr) を生成しません。

図 3-8. 実際の BER vs 推定 BER

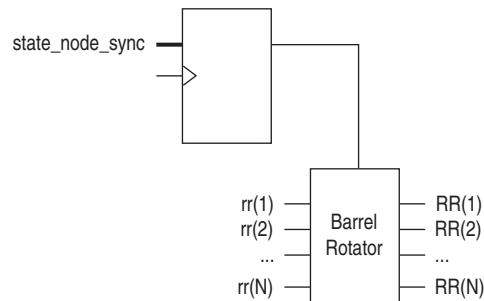


### ノード同期化

外部同期化を使用していない場合、N ビットの順序が分かっていない場合があります。ノード同期化オプションにより、デコーダが同期されているまで rr 入力を反転することができます。ノード同期化を使用するには、BER を観察して、state\_node\_sync を変更し続けて、正しい BER 値が得られるまで rr 入力を反転します。

図 3-9 に、ノード同期化のブロック図を示します。

図 3-9. ノード同期化



注：

(1) バレル・ロテータは、ノード同期化オプションを選択する場合にのみ実装されます。

次の式は、ノード同期化を表します。

$$RR[i] = rr[((state\_node\_sync + i - 1) \bmod N) + 1]$$

ここで、N に対して、i が 1 となります。

RR および rr は、N のバスおよび softbits 幅の配列として扱われます。state\_node\_sync の有効値の範囲は、0 から (N - 1) までです。

## Code Sets タブ

表 3-4 に、Code Sets タブで設定可能なオプションを示します。

表 3-4. Code Sets タブのオプション

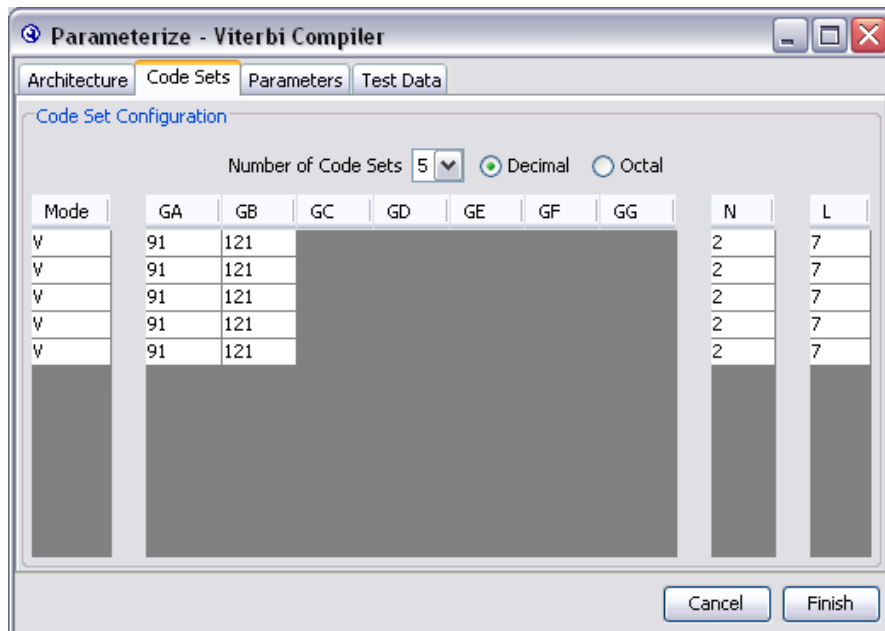
パラメータ	値	説明
Number of Code Sets	1 ~ 8	Viterbi コンパイラは複数のコード定義をサポートします。複数のコード・セット・オプションは最大 8 つのコード・セットをサポートします。コード・セットは、コード・レートおよび関連する生成多項式から構成されています。
Decimal or Octal	-	生成多項式が 10 進数または 8 進数で表示するかを選択します。デザイン・ファイルでの表示は 10 進数ですが、10 進数または 8 進数で入力することができます。
Mode	V ~ T	Viterbi モード (V) または TCM モード (T)。
GA, GB, GC, GD, GE, GF, GG (1)	-	生成多項式です。複数コード・セットのオプションが使用された場合、多項式の異なるセットがそれぞれの GI グループに入力されます。IP Toolbench は、任意の多項式で上書きできるデフォルト値を提供します (ウィザードは、入力された値が有効であるかどうかをチェックしない)。
Number of coded bits. (N)	2 ~ 7 (ハイブリッド) 2 ~ 4 (パラレル) (2)	エンコードされるビットに対して、N ビットが出力されます。複数コード・セットのオプションにより、最大 5 つの N パラメータが任意の順序で使用できます。
Constraint length (L)	3 ~ 9	拘束長です。畳み込みエンコーダのステート数を定義します。ここで、ステート数は $2^{L-1}$ です。コード・セットごとに、異なる L 値を選択することができます。

表 3-4 の注：


- (1) パラレル・アーキテクチャの場合、GA、GB、GC および GD のみを使用されます。
- (2) Viterbi モードにのみ使用されます。N = 2 は、TCM モードにのみサポートされます。


図 3-10 に、**Code Sets** タブを示します。

図 3-10. Code Sets タブ



複数のコード・セットの場合、一番目のコード定義は第 1 行に対応し、sel\_code 入力 =0 によって選択されます。第 2 行は sel\_code = 1 によって選択されます。第 3 行は sel\_code = 2 によって選択されます（以降同様）。定義ごとに、N、多項式、拘束長 (L)、およびモード (Viterbi または TCM) を選択できます。様々な拘束長とモードを組み合わせる使用することができます。IP Toolbench が作成したテスト・データは、コード定義をテストします。これらのテストは、テストベンチでのシミュレーション時に表示され、あるいは **block\_period\_stim.txt** ファイルにあります。

 ハイブリッド・モードでは、拘束長が 3 および 4 の場合、tr\_init\_state のビット幅が 4 になりますが、MegaCore ファンクションはこの余分の上位ビットを無視します。

 複数の拘束長の場合、Viterbi アルゴリズムが原因で、前にエデコードされたビットの一部は正しくない場合があります。これを回避するには、より低い BER を設定し、間違ったトレリス・パスの確率を低減し、**Optimization** を **None** に設定し、**Best State Finder** をオンにします。

## Parameters タブ

表 3-5 に、**Parameters** タブで設定可能なオプションを示します。

表 3-5. Parameters タブのオプション

パラメータ	値 e	説明
最大拘束長 ( $L_{MAX}$ )	5 ~ 9 (ハイブリッド) 3 ~ 9 (パラレル)	最大拘束長 $L_{MAX}$ です。(また、11 ページの「Code Sets タブ」を参照してください。)
ACS Units (A)	1、2、4、8、または 16	ACS ユニットの数です。これは並列度を追加します (ハイブリッド・アーキテクチャのみ)。使用可能な値の範囲は、最大拘束長 $L_{MAX}$ の値に依存します。
Traceback (v)	8 (最小)	トレースバック長です。これは、デコードされたビットを取得するのにトレース・バックされるトレリスのステージ数です。この値は、パンクチャされていないコードに対しては通常 $6 \times L$ に設定され、高度にパンクチャされているコードに対しては最大 $15 \times L$ に設定されます。
Softbits (softbits)	1 ~ 16	シンボルあたりの軟判定ビットの数です。ソフトビットが 1 に設定される場合、デコーダは硬判定デコーダとして動作します。eras_sym 入力により、削除されたシンボルを入力することができます。(1 ページの「ソフト・シンボル入力」を参照してください。)
Bmgwide	-	ステート・メトリック累積の精度です (2 ページの「ステート・メトリック」を参照)。IP Toolbench は最適な値を選択し、表示します。この最適な値は、 $N_{MAX}$ 、 $L_{MAX}$ および softbits に依存します。

### スループット・カリキュレータ

スループット・カリキュレータは、次の式を使用します。

$$\text{ハイブリッド・スループット} = f_{MAX}/Z$$


ここで、

$$\begin{aligned} \log_2 C = 3 \text{ の場合、} & Z = 10 \\ \log_2 C > 3 \text{ の場合、} & Z = 2^{\log_2 C} \\ \log_2 C = L_{MAX} - 2 - \log_2 A & \\ L_{MAX} & \text{ は最大拘束長} \\ A & \text{ は ACS ユニットの数} \end{aligned}$$

$$\text{パラレル・スループット} = f_{MAX}$$

### レイテンシ・カリキュレータ

レイテンシ・カリキュレータは、Viterbi デコーダのレイテンシの近似指標を与えます。レイテンシとは、データを処理して出力するのにかかるクロック・サイクル数です。レイテンシは、MegaCore ファンクションに入る最初のシンボル (sink\_sop) から MegaCore ファンクションを出すシンボル (source\_sop) によって測定されます。レイテンシはパラメータに依存します。

 正確なレイテンシを得るために、シミュレーションを実行します。

ハイブリッド・アーキテクチャの場合、レイテンシ・カリキュレータは次の式を使用します。

$$\text{クロック・サイクルの数} = Z \times V$$

ここで、

$V$ は、入力  $tb\_length$  のトレースバック長です。

$\log_2 C = 3$  の場合、 $Z = 10$

$\log_2 C > 3$  の場合、 $Z = 2^{\log_2 C}$

$\log_2 C = 1_{\max} - 2 - \log_2 A$ ,

$A$  は ACS ユニットです。

 パラレル・アーキテクチャの場合、クロック・サイクル数は約  $4V$  です。

## Test Data タブ

表 3-6 に、**Test Data** タブで設定可能なオプションを示します。

表 3-6. Test Data

パラメータ	説明
ブロックあたりのビット数	ブロックあたりのビット数です。 「ブロックあたりのビット数 × ブロック数」は 50,000,000 以下でなければなりません。
信号対ノイズ比 (dB)	信号対ノイズ比です。値は 1 ~ 100 でなければなりません。
ブロック数	ブロックの数です。 「ブロックあたりのビット数 × ブロック数」は 50,000,000 以下でなければなりません。
パターン A	パンチャリング・パターン A を入力します。
パターン B	パンチャリング・パターン B を入力します。

## 信号

Viterbi デコーダのデータ入力およびデータ出力には、Avalon Streaming (Avalon-ST) インタフェースが使用されています。入力は Avalon-ST シンクであり、出力は Avalon-ST ソースです。Avalon-ST インタフェースの `READY_LATENCY` パラメータは 1 に設定されます。


 Avalon-ST インタフェースについて詳しくは、「[Avalon Interface Specifications](#)」を参照してください。

図 3-11 に、Viterbi デコーダの Avalon-ST インタフェースを示します。

図 3-11. Avalon-ST インタフェース

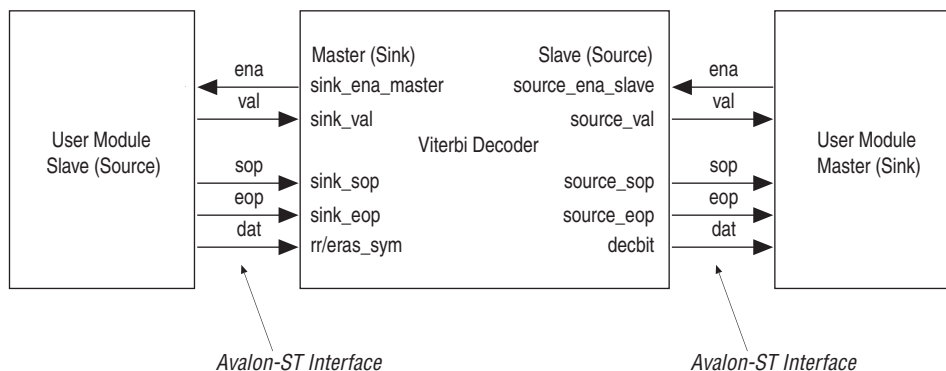


表 3-7 に、グローバル信号を示します。

表 3-7. グローバル信号

信号名	説明
clk	メインのシステム・クロックです。MegaCore ファンクション全体は clk の立ち上がりエッジで動作します。
reset	リセットです。リセットを High にアサートすると、デコーダ全体が非同期でリセットされます。reset 信号はシステム全体をリセットします。reset 信号は、clk の立ち上がり同期してディアサートする必要があります。

表 3-8 に、Avalon-ST シンク信号を示します。

表 3-8. Avalon-ST シンク信号 ( 1 / 2 )

信号名	Avalon-ST 名	方向	説明
sink_rdy	ready	出力	データ転送イネーブル信号です。sink_rdy インタフェース・シンクによってドライブされ、インタフェースにわたるデータ・フローの制御に使用されます。sink_rdy は、シンクからソースまでのリード・イネーブルとして動作します。  sink_rdy が clk の立ち上がりエッジでアサートされると、ソースが次のクロック・サイクルで Avalon-ST データ・インタフェース信号をドライブし、sink_val をアサートすることができます。ハイブリッド・アーキテクチャでは、sink_rdy は一度に 1 クロック・サイクルアサートされます。そのときにデータが使用できない場合、次の sink_rdy パルスまで待機しなければなりません。この信号は以前に、sink_ena_master と呼ばれます。
sink_val	val	入力	データ有効信号です。sink_val は、データ信号の有効性を表します。sink_rdy がサンプリングしてアサートされるときに、sink_val はクロック・エッジごとに更新されます。sink_rdy がサンプリングしてディアサートされるときに、sink_val は dat バスと共に現在の値を維持します。sink_val がアサートされるときに、Avalon-ST データ・インタフェースが有効です。sink_val がディアサートされるときに、Avalon-ST データ・インタフェースは無効になり、無視する必要があります。新しいデータを受信したかどうかを確認するために、シンクは sink_rdy 信号の以前の状態で sink_val 信号を検証します。
sink_sop	sop	入力	パケット (ブロック) 開始信号です。sink_sop は、rr バスで境界を区分します。sink_sop が High のときに、パケットの開始は rr バスで表示されます。sink_sop は、各パケットの最初の転送でアサートされます。この信号は、ブロックのデコードにのみ適用されます。
sink_eop	eop	入力	パケット (ブロック) 終了信号です。sink_eop は、rr バスで境界を区分します。sink_eop が High のときに、パケットの終了は rr バスで表示されます。sink_eop は、各パケットの最後の転送でアサートされます。この信号は、ブロックのデコードにのみ適用されます。

表 3-8. Avalon-ST シンク信号 ( 2 / 2 )

信号名	Avalon-ST 名	方向	説明
rr (1)	dat (2)	入力	データ入力です。この入力には n シンボルがあり、各シンボルはクロックあたりに softbits の幅をしています。1 ページの「エンコード手法」では、畳み込みエンコーダの出力に対応する入力シンボルについて説明します。ソフト・シンボルのマッピングについては、表 3-1 を参照してください。
eras_sym [Nmax:1]	dat (2)	入力	アサートされると、eras_sym は削除されたシンボルを表します。

表 3-8 の注：

- (1) TCM モードでは、rr 幅は  $(2^N \times \text{softbits}:1)$  です。Viterbi モードでは、rr 幅は  $(nmax \times \text{softbits}:1)$  です。  
 (2) rr および eras\_sym の両方は、Avalon-ST の dat 入力として見られています。

表 3-9 に、Avalon-ST のソース信号を示します。

表 3-9. ソース信号

信号	Avalon-ST 名	方向	説明
source_rdy	ready	入力	データ転送をイネーブルする信号です。source_rdy はシンク・インタフェースによってドライブされ、インタフェースにわたるデータ・フローの制御に使用されます。ena は、シンクからソースへのリード・イネーブルとして動作します。source_rdy がソースのドライブする clk の立ち上がりエッジでアサートされると、次の clk の立ち上がりエッジでは、Avalon-ST データ・インタフェースが信号を送信し、source_val をアサートします。シンクは次の clk の立ち上がりエッジでデータ・インタフェース信号をキャプチャします。ソースが新しいデータを供給できない場合、source_val は有効なデータ・インタフェースをドライブできるまでデアサートされます。以前に、source_ena_slave と呼ばれます。
source_val	val	出力	データ有効信号です。source_val は、decbit 信号で有効な出力がある場合に 1 クロック サイクル High にアサートされます。
source_sop	sop	出力	パケット (ブロック) 開始信号です。連続最適化を選択する場合、この信号はオープンになり、テストベンチから削除する必要があります。
source_eop	eop	出力	パケット (ブロック) 終了信号です。連続最適化を選択する場合、この信号はオープンになり、テストベンチから削除する必要があります。
decbit	dat	出力	source_val 信号がアサートされる場合、decbit 信号は出力ビットを含みます。

表 3-10 に、コンフィギュレーション信号を示します。

表 3-10. コンフィギュレーション信号

信号名	説明
ber_clear	BER カウンタをリセットします。BER ブロック・オプションにのみ使用されます。
state_node_sync[log2(Nmax):1]	rr のノード同期化回転を指定します。sink_sop がアサートされると、state_node_sync 信号がラッチされます。
sel_code[log2(Ncodes):1]	コードワードの選択です。「0」は最初のコードワードを選択し、「1」は 2 番目のコードワード (以降同様) を選択します。バス・サイズの増加は、指定されるコードの数に依存します。sink_sop がアサートされると、sel_code がラッチされます。
tb_length[]	トレースバック長です。tb_length の最大長は、パラメータ v の最大値とは同じです。sink_sop がアサートされると、tb_length がラッチされます。連続最適化を選択する場合、この信号はディセーブルされ、テストベンチから削除する必要があります。
tb_type	将来の互換性のために、アルテラは、tb_type を常に High に設定することを推奨しています。tb_type が Low 時のブロック・デコードでは、デコーダはステート 0 から開始します。tb_type が High の時、デコーダは tr_init_state[(L-1):1] で指定されたステートを使用します。ブロック・デコードの場合、tb_type を High に設定します。sink_eop がアサートされると、tb_type がラッチされます。連続最適化を選択する場合、この入力はトップ・レベル・デザインから削除され、内部コアで 0 に接続されます。
tr_init_state[(L-1):1]	tb_type が High 時に、トレースバックの開始ステートを指定します。tr_init_state は、sink_eop がアサートされるとラッチされます。連続最適化を選択する場合、この入力はトップ・レベル・デザインから削除され、内部コアで 0 に接続されます。詳細については、7 ページの「トレリス終端」を参照してください。
bm_init_state[(L-1):1] (1)	bm_init_value[] バスからの値で初期化するステートを指定します。ほかのすべてのステート・メトリックは 0 に設定されます。bm_init_state は、sink_sop がアサートされるとラッチされます。
bm_init_value[(L-1):1] (1)	開始ステートを初期化するメトリックの値を指定します。ほかのすべてのメトリックは 0 に設定されます。bm_init_value は、 $(L \times 2^{(\text{softbits} - 1)})$ 以上でなければなりません。bm_init_value は、sink_sop がアサートされるとラッチされます。
<p>表 3-10 の注:</p> <p>(1) ハイブリッド・アーキテクチャのみ。</p>	

表 3-11 に、ステータス信号を示します。

表 3-11. ステータス信号

信号	説明
normalizations [8:1]	normalizations バスは、sink_sop が最後にアクティブされてから発生した正規化の数をリアル・タイムで示します。(2 ページの「スタート・メトリック」を参照してください。)
numerr [] (1)	numerr バスは、ブロックで検出されたエラー数を含みます。このバスはエラーが検出されるたびに更新され、各エラーの位置を表示することができます。numerr は、source_sop がアサートされる時にリセットされ、source_sop の後に 2 クロック・サイクルで有効です。このバスの幅は IP Toolbench によって自動的に設定されます。BER ブロックを選択していない場合、この信号はオープンにしています。
bestmet [bmgwide:1]	ベスト・メトリックです。bestmet 信号は、ベスト・ステート・ファインダが検出するベスト・ステート・メトリックを示します。ベスト・メトリックを含むステートは bestadd に示されます。連続最適化を選択するか、または最適化をオフにして IP Toolbench でベスト・ステート・ファインダをオフにする場合、この信号はオープンにしています。
bestadd [(L-1):1]	ベスト・アドレス・ステートです。このアドレスは、ベスト・ステート・ファインダが検出するベスト・メトリックに対応します。このステートのメトリックは bestmet に示されます。連続最適化を選択するか、または最適化をオフにして IP Toolbench でベスト・ステート・ファインダをオフにする場合、この信号はオープンにしています。

表 3-11 の注：

(1) BER は、BER 見積もりツールのオプションがイネーブルされる場合にのみ使用されます。

## タイミング図

図 3-12 に、ハイブリッド Viterbi デコーダの入力タイミング図を示します。sink\_rdy 信号は、Z クロック・サイクルごとに 1 クロック・サイクルアサートされます。(Z の値については、13 ページの「レイテンシ・カリキュレータ」を参照してください。) データがソース側で受信されなくてデコーダがフルになった場合、sink\_rdy は新しいデータを受信できるまでディアサートされます。デコーダは、sink\_rdy がアサートされる場合にのみデータを受信します。

図 3-12. 入力タイミング図 — ハイブリッド

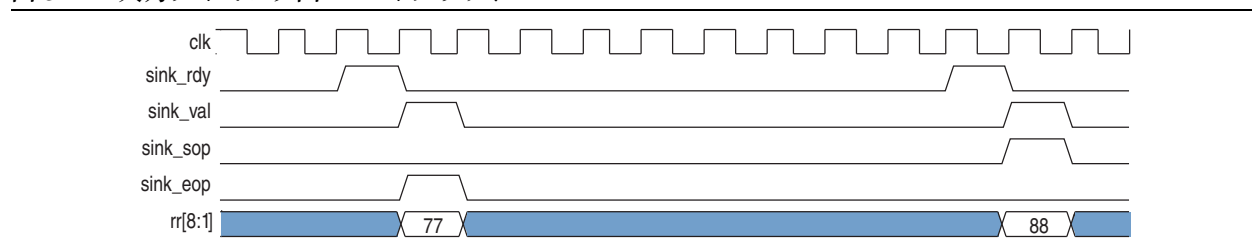


図 3-13 に、パラレル Viterbi デコーダの入力タイミング図を示します。

図 3-13. I 入カタイミング図 — パラレル

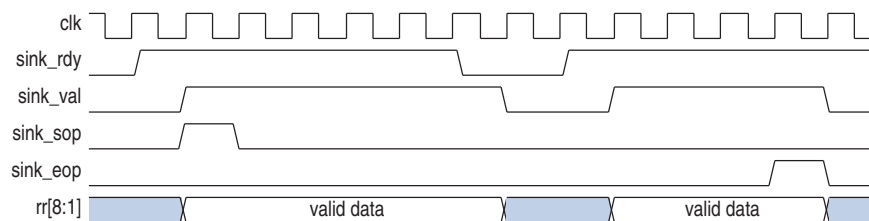
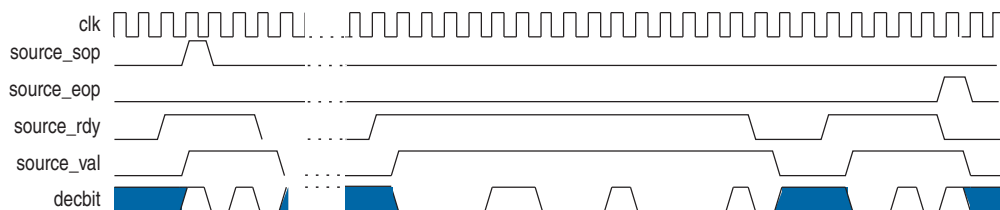


図 3-14 および 19 に、出力タイミング図を示します。図 3-14 では、source\_val 信号が最初に 8 または 16 クロック・サイクルアサートされます。その後、source\_rdy がアサートされる限り、source\_val 信号は残りのデータ量に対応するクロック・サイクル数アサートされます。

図 3-14 に、ソース (Viterbi) からシンク (ユーザー) までのサイド接続における Avalon-ST インタフェースのブロックまたはパケットの標準的な終了を示します。

図 3-14. 出力タイミング図 — 例 1



19 ページの図 3-15 に、異なる終了を示します。

図 3-15. 出力タイミング図 — 例 2

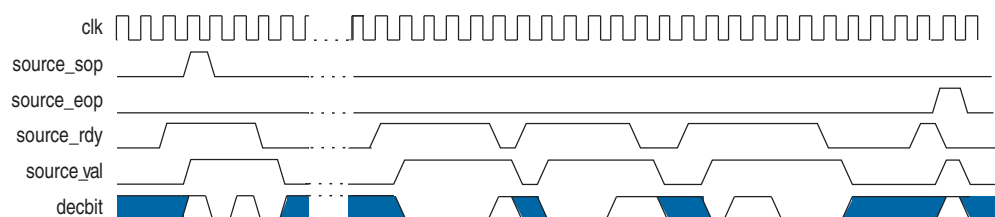
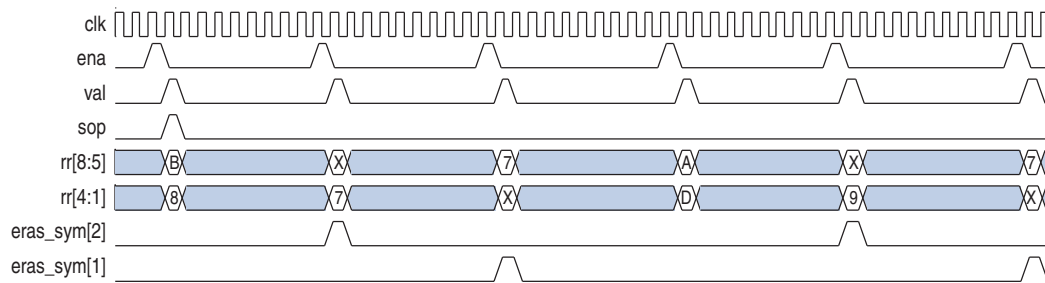


図 3-16 に、デパンクチャ・タイミング図を示し、パターン 110110 (パンクチャリング・レート 3/4) の eras\_sym を示します。eras\_sym パターンを変更することで、ほぼすべてのデパンクチャ・パターンが実装できます。

図 3-16. デパנקチャ・タイミング図



## MegaCore 検証

MegaCore 検証には自動回帰テスト・スイートが含まれており、以下で説明されます。

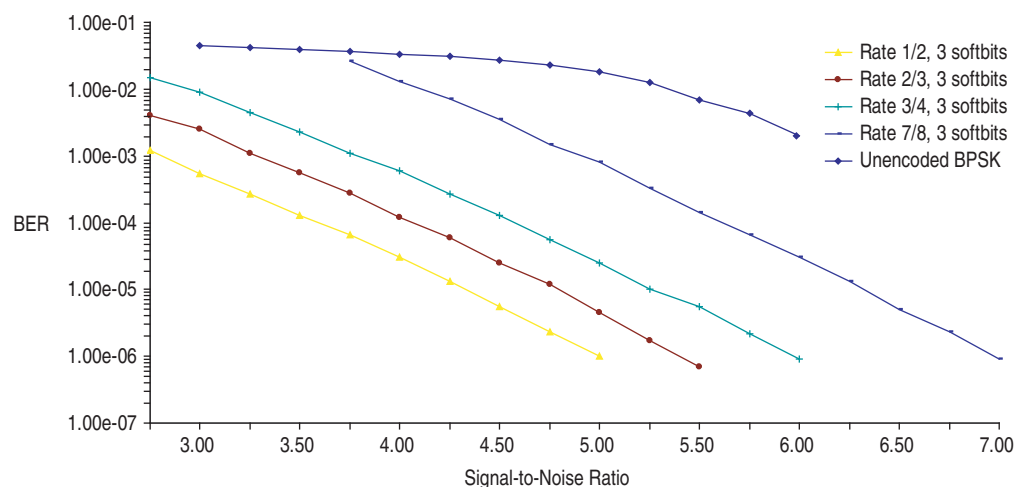
スクリプトは、RTL レベルでシミュレーションをドライブします。データは、ランダムで生成され、エンコードされます。オリジナルの送信ビットは **transbit.txt** と呼ばれるファイルに格納されます。オプションで、ガウス雑音チャンネル・モデルとして追加され、データがデコーダのテストベンチに使用されるためにフォーマットされます。テストベンチを供給するファイルは **a\_rcvsym.txt** です。テストベンチはデコーダのデコードされたビットを収集し、**decoded.txt** に格納します。それらのビットは **transbit.txt** でのオリジナル・ビットと比較されます。

スクリプトは、RTL VHDL シミュレーションの包括的なパラメータ・セットをカバーするテスト・セットを定義します。

テストベンチは、Avalon-ST インタフェース・テスト用の多くのパターン、およびすべての可能なテスト・シナリオを生成できます。

最初のテストはノイズなしのデータで行われます。そして、BER 性能を検証するために、ノイズ付きのデータを使用して様々な信号対ノイズ比で数百万ビットを実行します。BER 性能は、Viterbi デコーダの論理的な動作に一致します (20 ページの [図 3-17](#))。

図 3-17. 実際の BER vs. 様々なレートの信号対ノイズ比



もう 1 つのサブセットのパラメータは、合成後 Vital VHDL ネットリストを使用してノイズなしのデータでテストされます。

このセットのテスト・パターンおよびパラメータは包括的であり、ハイブリッドおよびパラレル・アーキテクチャのパラメータ・セットの機能またはパラメータ・セットにおける誤動作をすべて検出できる必要があります。



## 改訂履歴

次の表では、このユーザーガイドの改訂履歴を示します。

日付	バージョン	変更内容
2011年5月	<b>11.0</b>	<ul style="list-style-type: none"> <li>■ Arria® II GX、Arria II GZ、Cyclone® III LS、および Cyclone IV GX デバイスのサポート・レベルを最終サポートに更新</li> <li>■ HardCopy III、HardCopy IV E、および HardCopy IV GX デバイスのサポート・レベルを HardCopy コンパイルに更新</li> </ul>
2010年12月	<b>10.1</b>	<ul style="list-style-type: none"> <li>■ Arria II GZ デバイスの暫定サポートを追加</li> <li>■ Stratix IV GT デバイスのサポート・レベルを最終サポートに更新</li> </ul>
2010年7月	<b>10.0</b>	<ul style="list-style-type: none"> <li>■ Stratix V デバイスの暫定サポートを追加</li> </ul>
2009年11月	<b>9.1</b>	<ul style="list-style-type: none"> <li>■ メンテナンス更新</li> <li>■ Cyclone III LS、Cyclone IV、および HardCopy IV GX デバイスの暫定サポートを追加</li> </ul>
2009年3月	<b>9.0</b>	Arria II GX デバイスのサポートを追加
2008年11月	<b>8.1</b>	トレリス終端およびトレリス開始のセクションに注を追加
2008年5月	<b>8.0</b>	Stratix IV デバイスのサポートを追加
2007年10月	<b>7.2</b>	メンテナンス・リリース
2007年5月	<b>7.1</b>	<ul style="list-style-type: none"> <li>■ Arria GX デバイスのサポートを追加</li> <li>■ 信号の説明を改正</li> <li>■ ber_clear 信号を新たに追加</li> <li>■ パラレル・アーキテクチャの最適化オプションを追加</li> </ul>
2006年12月	<b>7.0</b>	Cyclone III デバイスのサポートを追加
2006年12月	<b>6.1</b>	フォーマットを更新

## アルテラへの問い合わせ

アルテラの製品に関する最新情報については、次の表を参照してください。






お問い合わせ先 (注 1)	お問い合わせ方法	アドレス
技術的なご質問	ウェブサイト	<a href="http://www.altera.com/support">www.altera.com/support</a>
技術トレーニング	ウェブサイト	<a href="http://www.altera.com/training">www.altera.com/training</a>
	電子メール	<a href="mailto:custrain@altera.com">custrain@altera.com</a>
アルテラの資料に関するお問い合わせ	電子メール	<a href="mailto:literature@altera.com">literature@altera.com</a>
一般的なお問い合わせ	電子メール	<a href="mailto:nacomp@altera.com">nacomp@altera.com</a>
ソフトウェア・ライセンスに関するお問い合わせ	電子メール	<a href="mailto:authorization@altera.com">authorization@altera.com</a>

注：

(1) 詳しくは、日本アルテラまたは販売代理店にお問い合わせください。

## 表記規則

本書では、以下の表に示す表記規則を使用しています。

書体	意味
太字かつ文頭が大文字	コマンド名、ダイアログ・ボックス・タイトル、ダイアログ・ボックス・オプション、およびその他の GUI ラベルを表します。例：Save As ダイアログ・ボックス
太字	ディレクトリ名、プロジェクト名、ディスク・ドライブ名、ファイル名、ファイルの拡張子、およびソフトウェア・ユーティリティ名を表します。例：\qdesigns ディレクトリ、d: ドライブ、および chiptrip.gdf ファイル
斜体かつ文頭が大文字	資料のタイトルを表します。例えば、「AN 519: Stratix IV Design Guidelines」
斜体	変数を表します。例：n + 1 変数名は、山括弧 ( ) で囲んでいます。例：<file name> および <project name>.pof ファイル
文頭が大文字	キーボード・キーおよびメニュー名を表します。例：Delete キー、Options メニュー
「小見出しタイトル」	かぎ括弧は、資料内の小見出しおよび Quartus II Help トピックのタイトルを表します。例：「表記規則」
Courier フォント	信号、ポート、レジスタ、ビット、ブロック、およびプリミティブ名を表します。例：data1、tdi、および input アクティブ Low 信号は、サフィックス n で表示されています (例：resetn)。 コマンドライン・コマンド、および表示されているとおりに入力する必要があるものを表します。例：c:\qdesigns\tutorial\chiptrip.gdf また、Report ファイルのような実際のファイル、ファイルの構成要素 (例：AHDL キーワードの SUBDESIGN)、ロジック・ファンクション名 (例：TRI) も表します。
1.、2.、3.、および a.、b.、c. など	手順など項目の順序が重要なものは、番号が付けられリスト形式で表記されています。
■ ■	箇条書きの黒点などは、項目の順序が重要ではないものに付いています。
	指差しマークは、要注意箇所を表しています。
	注意は、製品または作業中のデータに損傷を与えたり、破壊したりするおそれのある条件や状況に対して注意を促します。
	警告は、ユーザーに危害を与えるおそれのある条件や状況に対して注意を促します。
	矢印は、Enter キーを押すよう指示しています。
	足跡マークは、詳細情報の参照先を示しています。