

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。

Altera® 3GPP LTE ターボのリファレンス・デザインはターボ・コードを使用してトレリスの終端をサポートするエンコーディングおよび早期終端をサポートする FEC (Forward Error Correction) デコーディングを表示します。リファレンス・デザインは 3GPP Long Term Evolution (LTE または LTE-A) のチャンネル・カードまたは *3GPP Technical Specification* と互換性のベースバンド・モデムのアプリケーションに適しています。

3GPP 技術仕様について詳しくは、*3GPP Technical Specification: Group Radio Access Network, Evolved Universal Terrestrial Radio Access, Multiplexing and Channel Coding (Release 8)*, TS 36.212 v8.3.0, May 2007 を参照してください。

リファレンス・ターボ・デコーダは SIC (Successive Interference Cancellation) 手法をサポートして、LTE-A 基準でスループット性能を向上させるためにチャンネル・コーディングのイコライゼーション手法としてベース・ステーション eNB のレシーバで作用されることができます。

LTE-A 基準の機能強化について詳しくは (www.3gpp.org) のウェブサイトで *LTE-Advanced Physical Layer* を参照してください。

ターボ・コードは 1993 年に Berrou (など) によって提案されました。発表されてからターボ・コードは Shannon 制限のエラー訂正能力に近いと、多くの通信とストレージ・システムにおける最適なコーディング技術となります。これらのアプリケーションは、3GPP、スペース・アプリケーション (CCSDS) テレメトリのチャンネル・コーディングの諮問委員会、WiMAX (ワイマックス: Worldwide Interoperability for Microwave Access)、および 2 ~ 数百 Mbps の範囲でスループットを必要とする 3GPP LTE を含まれています。一般的なコンフィギュレーション設定では、アルテラの 3GPP LTE ターボ・デコーダは、235Mbps のスループット・レートを提供し、3GPP LTE の対象となる高速のデータ・アップリンク・レートを満たします。

ターボ・コードについては、Proceedings of the IEEE International Conference on Communications, 1993, pp. 1064-1070 の C. Berrou, A. Glavieux, P. Thitimajshima, *Near Shannon Limit Error-Correcting, Coding, and Decoding: Turbo Codes* を参照してください。

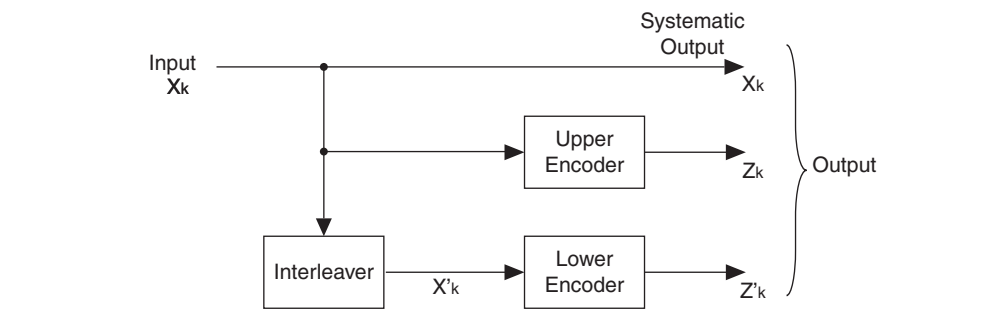
ターボ・エンコーダ

3GPP LTE 仕様で指定されている 3GPP LTE ターボ・コードは並列連結された畳み込みコードを使用しています。情報シーケンスは畳み込みコードによってエンコードされ、そして情報シーケンスのインタリーブ・バージョンは別の畳み込みコードによってエンコードされます。

ターボ・エンコーダのアーキテクチャ

ターボ・エンコーダは、2つの8状態の構成エンコーダとターボ・コードの内部インタリーバを使用して実装されます (図 1)。

図 1. ターボ・エンコーダのアーキテクチャ



ターボ・エンコーダは、以下のような機能もサポートしています。

- 3GPP LTE および LTE-A 準拠。
- すべての 3GPP LTE インタリーバ・ブロック・サイズは、実行時に選択可能。
- 1/3 のコード・レートのみである。他のは外部のレート・マッチにより実現できる。
- ダブル・バッファリングは以前のデータ・ブロックの処理中にエンコーダがデータを受け取ることができる。
- RTL テスト・ベクタの生成のための MATLAB のビット精度のモデル。
- MegaWizard Plug-In Manager を使用して VHDL または Verilog HDL テストベンチの自動生成。
- Avalon® Streaming (Avalon-ST) インタフェース。
- OpenCore Plus 評価機能ライセンス。

伝達関数

並列連結された畳み込みコードの 8 状態の構成コードの伝達関数は、次の通りです。

$$G(D) = \left[1, \frac{g_1(D)}{g_0(D)} \right]$$

ここで、 $g_0(D) = 1 + D^2 + D^3$ and $g_1(D) = 1 + D + D^3$ 。

入力ビットをエンコードする起動時に、8 状態の構成エンコーダのシフト・レジスタの初期値がすべてゼロになります。

ターボ・コードからの出力は次の通りです。

$$X_0, Z_0, Z'_0, X_1, Z_1, Z'_1, \dots, X_{K-1}, Z_{K-1}, Z'_{K-1}$$

ここで、

- X_0, X_1, \dots, X_{K-1} のビットは最初の 8 状態の構成エンコーダと内部インタリーバの両方の入力です (K はビット数である)。

- Z_0, Z_1, \dots, Z_{K-1} および $Z'_0, Z'_1, \dots, Z'_{K-1}$ のビットは最初と2番目の8状態の構成エンコーダからの出力です。
- 内部インタリーバの出力ビット (および2番目の8状態の構成エンコーダ) は $X'_0, X'_1, \dots, X'_{K-1}$ です。

トレリスの終端

図2は、トレリスの終端付きレート1/3のターボ・エンコーダの構造を示します (点線で示されます)。

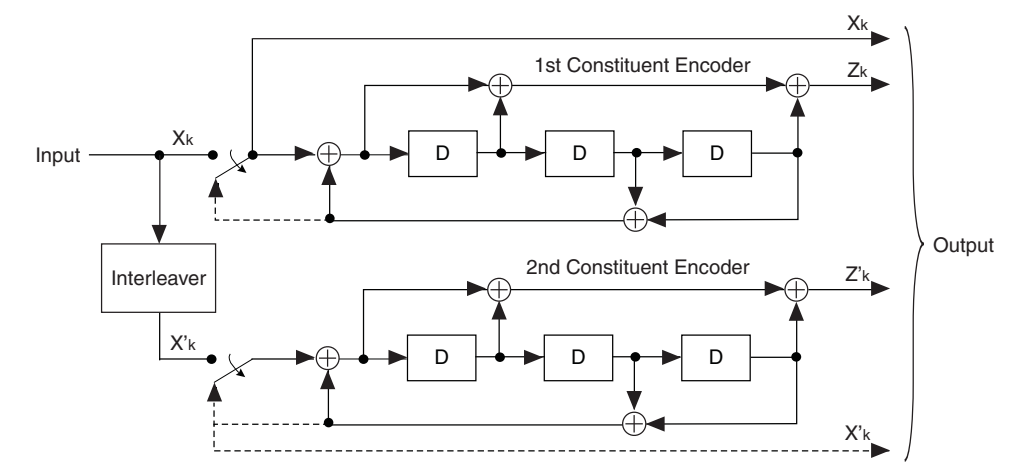
トレリスの終端は、すべての情報ビットがエンコードされた後、シフト・レジスタのフィードバックからテイル・ビットを取ることによって実行されます。テイル・ビットは情報ビットのエンコードの後にパディングされます。

第2の構成エンコーダが無効になっている場合、最初の3テイル・ビットは最初の構成エンコーダ (下の位置で図2の上側のスイッチ) を終了します。最初の構成エンコーダが無効になっている場合、最終の3テイル・ビットは第2の構成エンコーダ (下の位置で図2の下側のスイッチ) を終了します。

トレリスの終端のための送信されるビットは、次の通りです。

$$X_K, Z_K, X_{K+1}, Z_{K+1}, X_{K+2}, Z_{K+2}, X'_K, Z'_K, X'_{K+1}, Z'_{K+1}, X'_{K+2}, Z'_{K+2}$$

図2. レート1/3のターボ・エンコーダの構造



内部インタリーバ

ターボ・コードの内部インタリーバへの入力ビットは X_0, X_1, \dots, X_{K-1} で表示されています。ここで、 K は入力ビット数です。ターボ・コードの内部インタリーバへの出力ビットは $X'_0, X'_1, \dots, X'_{K-1}$ で表示されています。

入力ビットと出力ビットの関係は、次の通りです。

$$X'_i = X_{\pi(i)}, \quad i = 0, 1, \dots, K-1$$

ここで、出力のインデックス i と入力インデックス $\pi(i)$ との関係は、以下の二次形式を満たしています。

$$\pi(i) = (f_1 i + f_2 i^2) \bmod K$$

パラメータ f_1 と f_2 は、ブロック・サイズ K に依存します。表1に、3GPP Technical Specification で指定されたインタリーバのパラメータを示します。

 3GPP Technical Specification について詳しくは、3GPP Technical Specification: Group Radio Access Network, Evolved Universal Terrestrial Radio Access, Multiplexing and Channel Coding (Release 8)、TS 36.212 v8.3.0、May 2007 を参照してください。

表 1. 内部インターバ・パラメータのターボ・コード (その 1)

i	K _i	f ₁	f ₂	i	K _i	f ₁	f ₂	i	K _i	f ₁	f ₂	i	K _i	f ₁	f ₂
1	40	3	10	48	416	25	52	95	1120	67	140	142	3200	111	240
2	48	7	12	49	424	51	106	96	1152	35	72	143	3264	443	204
3	56	19	42	50	432	47	72	97	1184	19	74	144	3328	51	104
4	64	7	16	51	440	91	110	98	1216	39	76	145	3392	51	212
5	72	7	18	52	448	29	168	99	1248	19	78	146	3456	451	192
6	80	11	20	53	456	29	114	100	1280	199	240	147	3520	257	220
7	88	5	22	54	464	247	58	101	1312	21	82	148	3584	57	336
8	96	11	24	55	472	29	118	102	1344	211	252	149	3648	313	228
9	104	7	26	56	480	89	180	103	1376	21	86	150	3712	271	232
10	112	41	84	57	488	91	122	104	1408	43	88	151	3776	179	236
11	120	103	90	58	496	157	62	105	1440	149	60	152	3840	331	120
12	128	15	32	59	504	55	84	106	1472	45	92	153	3904	363	244
13	136	9	34	60	512	31	64	107	1504	49	846	154	3968	375	248
14	144	17	108	61	528	17	66	108	1536	71	48	155	4032	127	168
15	152	9	38	62	544	35	68	109	1568	13	28	156	4096	31	64
16	160	21	120	63	560	227	420	110	1600	17	80	157	4160	33	130
17	168	101	84	64	576	65	96	111	1632	25	102	158	4224	43	264
18	176	21	44	65	592	19	74	112	1664	183	104	159	4288	33	134
19	184	57	46	66	608	37	76	113	1696	55	954	160	4352	477	408
20	192	23	48	67	624	41	234	114	1728	127	96	161	4416	35	138
21	200	13	50	68	640	39	80	115	1760	27	110	162	4480	233	280
22	208	27	52	69	656	185	82	116	1792	29	112	163	4544	357	142
23	216	11	36	70	672	43	252	117	1824	29	114	164	4608	337	480
24	224	27	56	71	688	21	86	118	1856	57	116	165	4672	37	146
25	232	85	58	72	704	155	44	119	1888	45	354	166	4736	71	444
26	240	29	60	73	720	79	120	120	1920	31	120	167	4800	71	120
27	248	33	62	74	736	139	92	121	1952	59	610	168	4864	37	152
28	256	15	32	75	752	23	94	122	1984	185	124	169	4928	39	462
29	264	17	198	76	768	217	48	123	2016	113	420	170	4992	127	234
30	272	33	68	77	784	25	98	124	2048	31	64	171	5056	39	158
31	280	103	210	78	800	17	80	125	2112	17	66	172	5120	39	80
32	288	19	36	79	816	127	102	126	2176	171	136	173	5184	31	96
33	296	19	74	80	832	25	52	127	2240	209	420	174	5248	113	902
34	304	37	76	81	848	239	106	128	2304	253	216	175	5312	41	166
35	312	19	78	82	864	17	48	129	2368	367	444	176	5376	251	336
36	320	21	120	83	880	137	110	130	2432	265	456	177	5440	43	170

表1. 内部インタリバー・パラメータのターボ・コード (その2)

i	K_i	f_1	f_2	i	K_i	f_1	f_2	i	K_i	f_1	f_2	i	K_i	f_1	f_2
37	328	21	82	84	896	215	112	131	2496	181	468	178	5504	21	86
38	336	115	84	85	912	29	114	132	2560	39	80	179	5568	43	174
39	344	193	86	86	928	15	58	133	2624	27	164	180	5632	45	176
40	352	21	44	87	944	147	118	134	2688	127	504	181	5696	45	178
41	360	133	90	88	960	29	60	135	2752	143	172	182	5760	161	120
42	368	81	46	89	976	59	122	136	2816	43	88	183	5824	89	182
43	376	45	94	90	992	65	124	137	2880	29	300	184	5888	323	184
44	384	23	48	91	1008	55	84	138	2944	45	92	185	5952	47	186
45	392	243	98	92	1024	31	64	139	3008	157	188	186	6016	23	94
46	400	151	40	93	1056	17	66	140	3072	47	96	187	6080	47	190
47	408	155	102	94	1088	171	204	141	3136	13	28	188	6144	263	480

ダブル・バッファリング

データパスは、前ブロックをエンコードする際、新しいデータ・ブロックがシフトできるようにダブル・バッファされます。この手法は、I/O 操作の遅延を低減し、できるだけ多くのハードウェアを利用して、全体のスループットを向上させます。

入力データ・フォーマット

サイズ K のブロック・オーダリングに必要な入力データは、以下の通りです。

$$X_0, X_1, X_2, \dots, X_{K-1}$$

出力データ・フォーマット

出力データは 3 ビット幅です。表 2 に、サイズ K のブロック・オーダリングを示します。

表2. ターボ・エンコーダの出力データのオーダリング

出力データ	ソース・データ		
	2	1	0
0	Z'_0	Z_0	X_0
1	Z'_1	Z_1	X_1
...
$K-1$	Z'_{K-1}	Z_{K-1}	X_{K-1}
K	X_{K+1}	Z_K	X_K
$K+1$	Z_{K+2}	X_{K+2}	Z_{K+1}
$K+2$	X'_{K+1}	Z'_K	X'_K
$K+3$	Z'_{K+2}	X'_{K+2}	Z'_{K+1}

レイテンシの計算

エンコード遅延 D はデータのブロック全体をエンコードするために消費されるクロック・サイクル数です。 K はブロック・サイズの場合、 $D = K + 14$ です。

エンコード遅延は、負荷遅延を含まれないで、入力バッファに入力データをロードするためにブロック・サイズを K としてクロック・サイクルの同じ数が必要です。

例：

- $K = 6144$ のときに、 $D = 6144 + 14 = 6158$
- $K = 40$ のときに $D = 40 + 14 = 54$

レイテンシのエンコーディング（ブロックの全体をエンコードするエンコーダが要した時間）は、次の式で計算されます。

$$L = \frac{D}{f_{MAX}} \text{ s}$$

ここで、 f_{MAX} is はシステム・クロックのスピードです。

上記の例については、 $f_{MAX} = 245 \text{ MHz}$ のために、 L はそれぞれ $0.22 \mu\text{s}$ および $25.13 \mu\text{s}$ です。

スループットの計算

スループットは以下の式で計算されます。

$$T = \frac{K \times f_{MAX}}{D} \text{ bps}$$

例えば、前のセクションで、 T はそれぞれ 181.48 Mbps および 244.44 Mbps です。

テスト・ベクタの生成

シミュレーションを実行するには、以下のファイルが必要です。

- `ctc_encoder_input.txt`
- `ctc_encoder_input_info.txt`

1つのテスト・ケースは `<Turbo Install path>/turbo/lib/test_files` で提供されます。

提供されたシステムを使用してテスト・ベクトルを生成するために、次の手順を使用できます。

1. MATLAB を起動します (v 2007b 以降)。
2. 作業ディレクトリを `<Turbo Install path>/turbo/cml` に変更します。
3. 次のコマンドを入力します。

```
Cml_Startup ↵
```

4. 次のコマンドを入力して、`test` サブディレクトリを作成します。

```
mkdir ../test ↵
```

5. 次のコマンドを入力します。

```
[sim_param, sim_state] = CmlSimulate('Scenarios_LTE_ENCODER_RTL',  
[832 832 832 40 48 56 6144]); ↵
```

このコマンドは、最後のブロックのようなブロック・サイズの入力は 832 三回、その後 40、48、56 と 6144 でエンコーダに供給するためのテスト・ベクタを生成します。テストのニーズに合わせてこの行列を変更することができます。

コマンドが成功に実行されると、ファイルは `<Turbo install path>/turbo/test` のテスト・ベクタのディレクトリに表示されます。この場所を変更するには、**Scenarios_LTE_ENCODER_RTL.m** ファイルで `dump_dir` のパラメータを変更することができます。Quartus II プロジェクト・ディレクトリに、これらのファイルをコピーします。

ソフトウェアのシミュレーション・モデルは、次の 3 つのファイルを生成します。

- `ctc_encoder_input.txt`
- `ctc_encoder_input_info.txt`
- `ctc_encoder_output_gold.txt`

RTL シミュレーションの後、`ctc_encoder_output_gold.txt` の内容と一致する必要がある別のファイル `ctc_encoder_output.txt` が作成されます。

RTL シミュレーションの起動方法について詳しくは、26 ページの「[デザインのシミュレーション](#)」を参照してください。

ターボ・デコーダ

図 3 に、ターボ・デコーダの構造を示します。

図 3. ターボ・デコーダ・アーキテクチャ

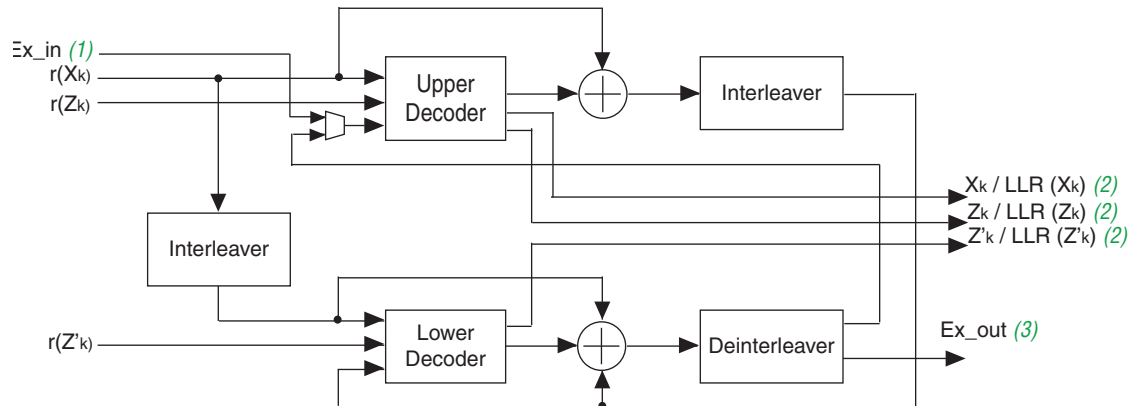


図 3 の注：

- (1) Turbo SIC and Extrinsic Input Information が有効になっている場合にのみ存在します。
- (2) LLR の出力は Turbo SIC コンフィギュレーションが有効になっている場合にのみ存在して、他の場合はハード・ビットが出力されます。
- (3) Turbo SIC and Extrinsic Output Information が有効になっている場合にのみ存在します。

ターボ・デコーダは反復作業の 2 つソフト・イン・ソフト・アウト (SISO) デコーダで構成されます。最初 (上部デコーダ) の出力は、2 番目にフィードして、ターボ・エンコーディングのイタレーションを形成します。インタリーバおよびデインタリーバは、このプロセスのリオーダ・データをブロックします。

ターボ・デコーダは、次の機能をサポートします。

- 3GPP LTE に準拠。
- LTE にわたって強化する LTE-A チャネル・コーディングのための Successive Interface Cancellation (SIC)。
- インタリーバのサイズと反復数のラン・タイム・パラメータ。
- CRC (Cyclic Redundancy Check) による早期終端のサポート。
- パラレル・エンジンの数、デコーディング・アルゴリズムの選択、入力精度、および出力サイズのコンパイル・タイム・パラメータ。
- ダブル・バッファリングは、以前のデータ・ブロックの処理中にデコーダがデータを受信できるようにすることで低レイテンシのリアル・タイム・アプリケーションをサポートしています。
- 外部メモリを必要としません。
- 性能シミュレーションまたは RTL テスト・ベクトルの生成のための C/MATLAB のビット精度のモデル。
- MegaWizard Plug-In Manager を使用する VHDL または Verilog HDL の生成。
- Avalon Streaming (Avalon-ST) インタフェース。
- OpenCore Plus 評価ライセンス。

デコーディング・アルゴリズム

次の 2 種類の MAP (Maximum A Posteriori) デコーディング・アルゴリズムをサポートされています。

- LogMAP—MAP の対数関数ドメイン上で作動し、良好なビット・エラー・レート (BER) を与えますが、より多くのロジック・リソースを消費します。このオプションは現在完全にサポートされていません。詳細はアルテラにお問い合わせください。
- MaxLogMAP—わずかに減少された BER 性能のコストで少ないロジック・リソースを使用する LogMAP の簡易型のバージョンは、LogMAP バリエーションに比例しています。このリファレンス・デザインで実装 MaxLogMAP アルゴリズムは、スケーリング係数で補正された MaxLogMAP のバージョンです。

 MaxLogMAP Turbo Decoder について詳しくは、J. Vogt, A. Finger, *Improving the Max-Log-MAP Turbo Decoder*, Electronics Letters, 36(23), 1937-1939, 2000 を参照してください。

入力データ・フォーマット

表 3 に、ターボ・デコーダに必要な入力データ・オーダリングを示します。

表 3. ターボ・デコーダの入力データ・オーダリング (その 1)

入力データ	sink_data		
	3N-1 downto 2N	2N-1 downto N	N-1 downto 0
0	Z'_0	Z_0	X_0
1	Z'_1	Z_1	X_1
...

表 3. ターボ・デコーダの入力データ・オーダリング (その 2)

入力データ	sink_data		
	3N-1 downto 2N	2N-1 downto N	N-1 downto 0
$K-1$	Z'_{K-1}	Z_{K-1}	X_{K-1}
K	X_{K+1}	Z_K	X_K
$K+1$	Z_{K+2}	X_{K+2}	Z_{K+1}
$K+2$	X'_{K+1}	Z'_K	X'_K
$K+3$	Z'_{K+2}	X'_{K+2}	Z'_{K+1}

表 3 の注:
 (1) N は入力ビット数を表します (IN_WIDTH_g)。

ターボ・デコーダは対数尤度のフォーマットになるようにすべてのデータが必要です。接続されたシステムは、次の式に従ってパリティ 1 およびパリティ 2 のビット・シーケンスを含むソフトの情報を提供する必要があります。

$$L(x) = \log \frac{P(x=1)}{P(x=0)}$$

対数尤度の値は、受信したビットが 1 である確率の対数であり、0 であるビットの確率で割って、2 の補数として表されます。ゼロの値が 1 および 0 の値は等しい確率を示し、デバଙ୍କチャリングに使用可能性があります。最も負の 2 の補数は、表現がバランスされるように未使用です。

表 4 に、4 ビットのマッピング値を示します。

表 4. 入力値

入力 (3 downto 0)	値
0111	1 に最も可能性がある
...	...
0001	1 に最低の可能性がある
0000	0 または 1 と等しい確率
1111	0 に最低の可能性がある
...	...
1001	0 に最も可能性がある
1000	未使用

出力データ・フォーマット

出力ビット数 (OUT_WIDTH_g) は 1 または 8 ビット幅を指定できます。1 ビットのオーダリングは次の通りです。

$$X_0, X_1, X_2, \dots, X_{K-1}$$

表 5 に、8 ビットの出力データ・オーダリングを示します。

表 5. 8 ビットの出力データ・オーダリング

出力 オーダー	source_data							
	7	6	5	4	3	2	1	0
1	X_7	X_6	X_5	X_4	X_3	X_2	X_1	X_0
2	X_{15}	X_{14}	X_{13}	X_{12}	X_{11}	X_{10}	X_9	X_8
...
$K/8$	X_{K-1}	X_{K-2}	X_{K-3}	X_{K-4}	X_{K-5}	X_{K-6}	X_{K-7}	X_{K-8}

表 6 は、固有出力が有効になっているとき、ターボ SIC コンフィギュレーションの出力データ・オーダリングを示します。 X_i 、 Z_i および Z'_i が意図のために LLR 出力であり、そして、それぞれ 1 および 2 のパリティ・ビットの場合、固有出力 Ex_out は E_i で表されます。これらの出力のデータ幅は可変 I コンフィギュレーションの IN_WIDTH_g+5 で指定されます。

表 6. Turbo SIC コンフィギュレーションの出力データ・オーダリング

出力 オーダー	source_data							
	7	6	5	4	3	2	1	0
1	$E_7X_7Z_7Z'_7$	$E_6X_6Z_6Z'_6$	$E_5X_5Z_5Z'_5$	$E_4X_4Z_4Z'_4$	$E_3X_3Z_3Z'_3$	$E_2X_2Z_2Z'_2$	$E_1X_1Z_1Z'_1$	$E_0X_0Z_0Z'_0$
2	$E_{15}X_{15}Z_{15}Z'_{15}$	$E_{14}X_{14}Z_{14}Z'_{14}$	$E_{13}X_{13}Z_{13}Z'_{13}$	$E_{12}X_{12}Z_{12}Z'_{12}$	$E_{11}X_{11}Z_{11}Z'_{11}$	$E_{10}X_{10}Z_{10}Z'_{10}$	$E_9X_9Z_9Z'_9$	$E_8X_8Z_8Z'_8$
...
$K/8$	$E_{K-1}X_{K-1}Z_{K-1}Z'_{K-1}$	$E_{K-2}X_{K-2}Z_{K-2}Z'_{K-2}$	$E_{K-3}X_{K-3}Z_{K-3}Z'_{K-3}$	$E_{K-4}X_{K-4}Z_{K-4}Z'_{K-4}$	$E_{K-5}X_{K-5}Z_{K-5}Z'_{K-5}$	$E_{K-6}X_{K-6}Z_{K-6}Z'_{K-6}$	$E_{K-7}X_{K-7}Z_{K-7}Z'_{K-7}$	$E_{K-8}X_{K-8}Z_{K-8}Z'_{K-8}$

レイテンシの計算

デコーディング遅延 D はデータのブロック全体をデコードするために消費されるクロック・サイクル数です。これはブロック・サイズ、実行する反復の数、およびデコーダで使用可能なエンジンの数に依存します。

次の計算は早期終端が行われていないと過程し、最悪のレイテンシです。

K はブロック・サイズ、 I はデコーディングの繰り返し数、および N はデコーダで指定されたエンジン数にします。その後、デコーディング遅延 D は、次式のいずれかを使用して計算することができます。

- $K < 264$ の場合、 $D = 26 + (2 \times f(K, N) + 14) \times 2 \times I$
- $K \geq 264$ の場合、 $D = 26 + (f(K, N) + 46) \times 2 \times I$

ここで、

$$f(K, N) = \begin{cases} K/N & K \text{ は } N \text{ に部分可能} \\ K/8 & K \text{ は } N \text{ に部分不可能} \end{cases}$$

例：


- $K = 6144$ の場合、 $N = 8$ 、 $l = 8$ 、 $D = 26 + (6144/8 + 46) \times 2 \times 8 = 13050$
- $K = 40$ の場合、 $N = 8$ 、 $l = 8$ 、 $D = 26 + (2 \times 40/8 + 14) \times 2 \times 8 = 410$

デコーディング・レイテンシ（デコーダは、デコードされたデータにブロック全体をデコードするのにかかる時間は、出力の準備ができています）は、次式を使用して計算することができます。

$$L = \frac{D}{f_{MAX}} \text{ s}$$

ここで、 f_{MAX} はシステムのクロック・スピードです。

上記の例に、 $f_{MAX} = 250 \text{ MHz}$ のために、 L はそれぞれ $52.2 \mu\text{s}$ および $1.64 \mu\text{s}$ です。


 これらの計算は、早期終端が発生していないと仮定することで、8回の繰り返し ($l = 8$) のターボ・デコーダを実行するためです。

スループットの計算

スループットは、次式を使用して計算することができます。

$$T = \frac{K \times f_{MAX}}{D} \text{ bps}$$

例えば、前のセクションで、 T はそれぞれ 117.7 Mbps および 24.38 Mbps です。

 これらの計算は、早期終端が発生していないと仮定することで、8回の繰り返し ($l = 8$) のターボ・デコーダを実行するためです。

早期終端のサポート

この LTE Turbo リファレンス・デザインのバージョンは、CRC24A または CRC24B を使用して早期終端をサポートしません（詳細は、*3GPP Technical Specification* を参照してください）。

SISO デコーダの出力（7 ページの図 3 の上下のデコーダ）で生成された CRC チェック・サムは反復ごとの後にチェックされます。入力ポートで指定した最大反復回数まで続けるよりも、ターボ・デコーダは、すぐに成功を取って CRC 結果として切斷されます。

早期終端は、消費電力および全体的なレイテンシを削減し、大幅にスループットが上記の予測を増加します。また、資料はデコーダの BER 性能を向上させることを示しています。上記の測定基準のいずれかの利益は、受信データ・ブロック、ブロック・サイズ、およびユーザーが持っている最大反復数の信号対ノイズ比（SNR）に依存しています。

テスト・ベクトルの生成

RTL シミュレーションを実行するために、次のファイルが必要です。

- **ctc_input_info.txt**
- **ctc_input_data.txt**

1 つのテスト・ケースは `<Turbo Install path>/turbo/lib/test_files` で提供されます。

提供されたシステムを使用してテスト・ベクトルを生成するために、次の手順を使用できます。

1. MATLAB を起動します (v 2007b 以降)。
2. 作業ディレクトリを `<Turbo Install path>/turbo/cml` に変更します。
3. 次のコマンドを入力します。

```
Cml_Startup ↓
```

4. 次のコマンドを入力して、**test** サブディレクトリを作成します。

```
mkdir ../test ↓
```

5. *SNR*、*CRC* タイプ、および *max_iterations* などのパラメータを変更したいときに、ファイル **Scenarios_LTE_CRC_ET_RTL.m** をチェックします。必要な変更を行って、ファイルを保存します。



これらのパラメータについて詳しくは、**readme.pdf file in <Turbo Install path>/turbo/cml/documentation** を参照してください。

6. 次のコマンドを入力します。

```
[sim_param, sim_state] = CmlSimulate('Scenarios_LTE_CRC_ET_RTL',  
                                     [832 832 832 40 48 56 6144]); ↓
```

このコマンドは、最後のブロックのようなブロック・サイズの入力は **832** 三回、その後 **40**、**48**、**56** と **6144** でエンコーダに供給するためのテスト・ベクタを生成します。テストのニーズに合わせてこの行列を変更することができます。

コマンドが正常に実行されると、ファイルは `<Turbo install path>/turbo/test` のテスト・ベクタのディレクトリに表示されます。この場所を変更するには、**Scenarios_LTE_ENCODER_RTL.m** ファイルで *dump_dir* のパラメータを変更することができます。Quartus II プロジェクト・ディレクトリ (MegaWizard Plug-In Manager で作成された) のターボ・デコーダに、これらのファイルをコピーします。

次の 4 つのファイルは、ソフトウェア・シミュレーション・モデルで生成されます。

- **ctc_input_info.txt**
- **ctc_input_data.txt**
- **ctc_decoded_output_gold.txt**
- **ctc_output_et_info_gold.txt**

RTL シミュレーションの後、**ctc_decoded_output.txt** および **ctc_output_et_info.txt** の 2 つのファイルを作成され、**ctc_decoded_output_gold.txt** および **ctc_output_et_info_gold.txt** の内容と一致させる必要があります。

RTL シミュレーションの実行方法について詳しくは、[26 ページ](#)の「[デザインのシミュレーション](#)」を参照してください。

Avalon Streaming インタフェース

Avalon Streaming (Avalon-ST) インタフェースは、Atlantic™ インタフェースが進化したものです。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、本質的に複雑な制御ロジックを実装せずに効率的な達成および時間多重実装を可能にするマルチ・チャンネル・デザインを同期させます。

Avalon-ST インタフェースでは、シンクがデータ・送信を停止するようにソースにシグナルできるフロー制御のメカニズムであるバックプレッシャーをサポートします。シンクは通常、FIFO バッファがいっぱいである場合、または、その出力で輻輳が発生しているときにデータの流れを止めるためにバックプレッシャーを使用します。

3GPP LTE ターボ・リファレンス・デザインを含むデータパスをデザインするときに、ウンストリーム・コンポーネントは常にデータを受け取ることができると知っていれば、バックプレッシャーを必要としないことがあります。source_ready 信号を High にドライブし、sink_ready を接続しないことにより、より高いクロック・レートを達成するかもしれません。

3GPP LTE Turbo リファレンス・デザインで使用される Avalon-ST インタフェースは、ゼロの READY_LATENCY の値を持っています。



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。

パケット・フォーマット・エラーの処理

ターボ・メガファンクションは、システム内のデータ・エラーを通信する 2 つのエラー・ポートがあります。

- Sink_error はアップ・フロント・エラーを受信する 2 ビットの入力ポートです。
- Source_error は、エラー状態があることを示すための 2 ビットの出力ポートです（前ブロックで、ターボ・メガファンクションまたは他の場所でキャッチされる）。

メガファンクションは、次のパケットのフォーマット関連のエラーから処理し、回復できます。

- エラー・コードは、データ・ブロックの入力中に `sink_error` ポートから受信されている場合、ターボ・メガファンクションが現在のデータ・ブロックがエラーと破棄データを含んでいると仮定します。エラー信号が `Low` にアサートされると、ターボ・メガファンクションはフレッシュ SOP (Start-Of-Packet) (`sink_sop = 1`, `sink_valid = 1`) を前提し、フレッシュ・パケットが受信されるまで、データの入力を無視します。
- SOP (`sink_sop`) または EOP (`sink_eop`) の配置に不具合が生じるときに、エラーの種類に応じてエラー・コードが発行されます。
 - 01 -> SOP (start-of-packet) の欠落
 - 10 -> EOP (end-of-packet) の欠落
 - 11 -> 予期しない SOP および予期しない EOP
- データ・ブロックは LTE 規格でサポートされていない場合、ターボ・メガファンクションは 11 の値にエラー信号を発行し、パケットの新たなスタートになるまでデータ・ブロックの残りを無視します。

入力ポートと出力ポートでのデータ・ブロックおよびダブル・バッファリングの長い処理時間のため、入力データでのエラーは、発生するとすぐに報告されます。したがって、`source_error` 信号は前のブロックの出力の間にいつでも `High` にアサートされてしまうかもしれません。

エラーが検出されると、エラー・コードは、1 クロック・サイクルだけのために表示されます。

特定のデータ・ブロックに関連する複数のエラーがある場合、ターボ・メガファンクションは、最初に検出されたエラー・コードが表示されます。

それは `source_error` ポートで検出されたエラーを報告するために数クロック・サイクルかかります。

エラー回復の例外

検出されたエラーは非常に EOP (End-of-packet) (それは EOP の欠落、予期しない SOP または予期しない EOP がある場合) の境界に近いおよび前ブロックに別の CRC タイプのエラー・ブロックの後に続くブロックがある場合、ターボ・メガファンクションは、すぐにエラーから回復しない可能性があります。しかし、エラーはすべての状況で `source_error` ポートから報告されます。

システム要件

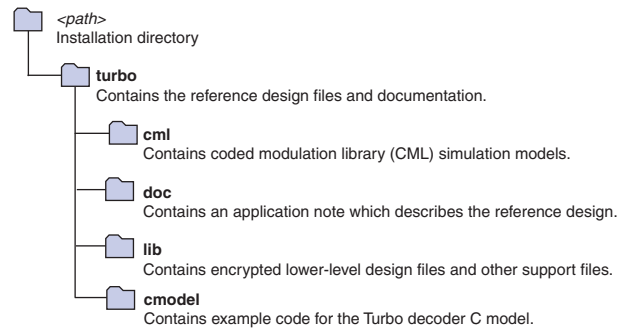
3GPP LTE ターボ・リファレンスは Windows XP および Linux Workstations をサポートし、Quartus II ソフトウェア v9.0 以降が必要です。

リファレンス・デザインのインストール

3GPP LTE ターボのリファレンス・デザインはアルテラのワイヤレス・ビジネス・ユニットから InstallShield のファイルとして入手可能です。

図 4 に、デザイン・ファイルのディレクトリ構造を示します。

図 4. リファレンス・デザインのディレクトリ構造



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 3GPP LTE ターボのリファレンス・デザイ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、実行時間に制限のあるデバイス・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。

機能および性能が十分満足できて、製品に組み込む場合にのみ、ライセンスを購入して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) 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 ハードウェアの評価機能について詳しくは、「[AN 320: OpenCore Plus 評価機能によるメガファンクションの評価](#)」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 2 種類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。

最も限定的な評価時間に達すると、デザイン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ファンクションがある場合、特定のファンクションのタイム・アウト動作は、他の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。3GPP LTE ターボ・デコーダのリファレンス・デザインの場合、アンテザード・タイム・アウトは1時間、テザード・タイム・アウト値は無制限です。

評価期限が切れると、reset_n 信号が Low になり、そのリセットに 3GPP LTE ターボ・デコーダのリファレンス・デザインが永久にを保ちます。

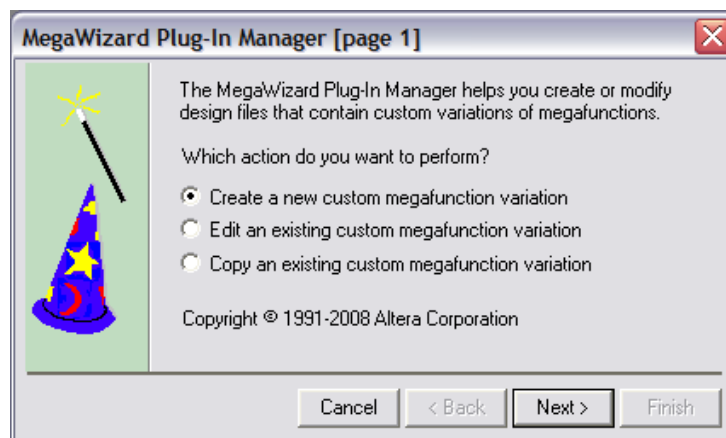
使用法

3GPP LTE ターボ・デコーダのリファレンス・デザインをインストールした後、Quartus II ソフトウェアの MegaWizard Plug-In Manager では Error Detection/Correction セクションでターボ・メガファンクションが使用可能です。

MegaWizard Plug-In Manager フローでは、ターボ・デコーダをパラメータ化し、メガファンクションを手動で Quartus II デザインに組み込むことができます。MegaWizard Plug-In Manager を使用するときは、次のステップを実行します。

1. Quartus II ソフトウェアで File メニューから **New Project Wizard** を選択して、新しいプロジェクトを作成します。
2. Tools メニューで **MegaWizard Plug-in Manager** をクリックして **Create a new custom megafunction variation** を選択します (図 5)。

図 5. MegaWizard Plug-In Manager



3. **Next** をクリックして、**DSP** セクションを展開し、**Installed Plug-Ins** リストの **Error Detection/Correction** メガファンクションから **Turbo v2.0** を選択します (図 6)。


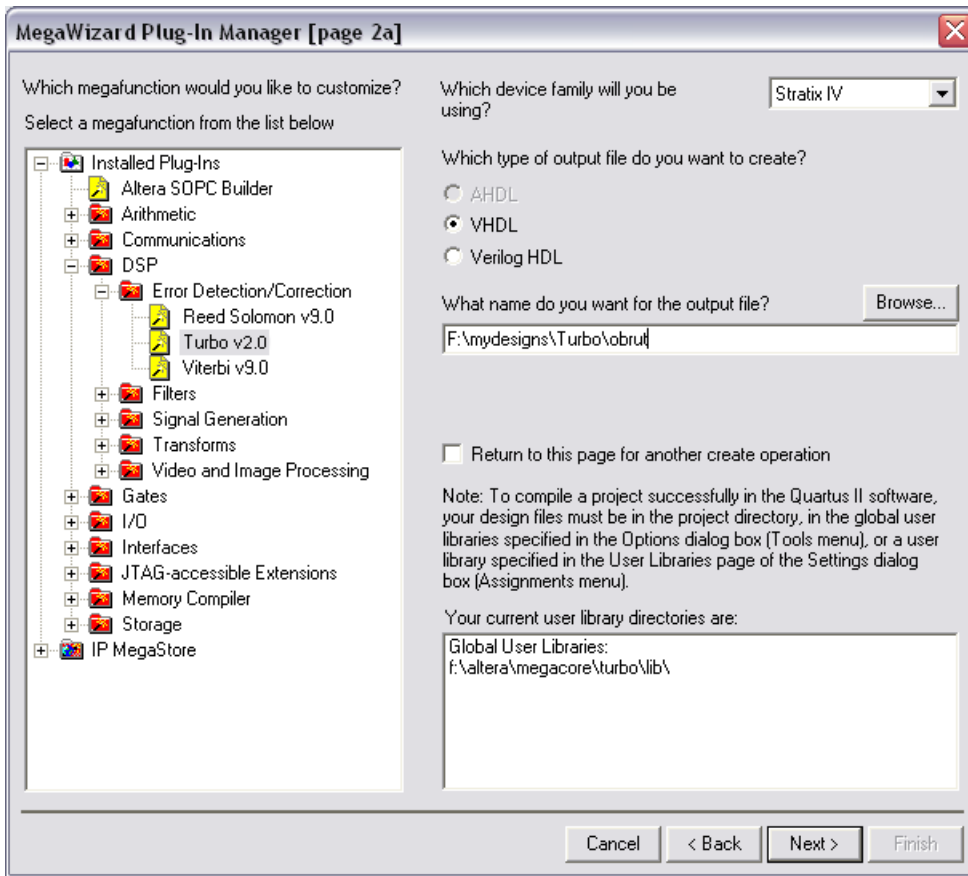
 MegaWizard Plug-In Manager に **Turbo v2.0** が表示されていない場合、Quartus II のグローバル・ユーザー・ライブラリに <Turbo Install Path>/turbo/lib を追加する必要があります (Tools メニューで **Options** をクリックする)。

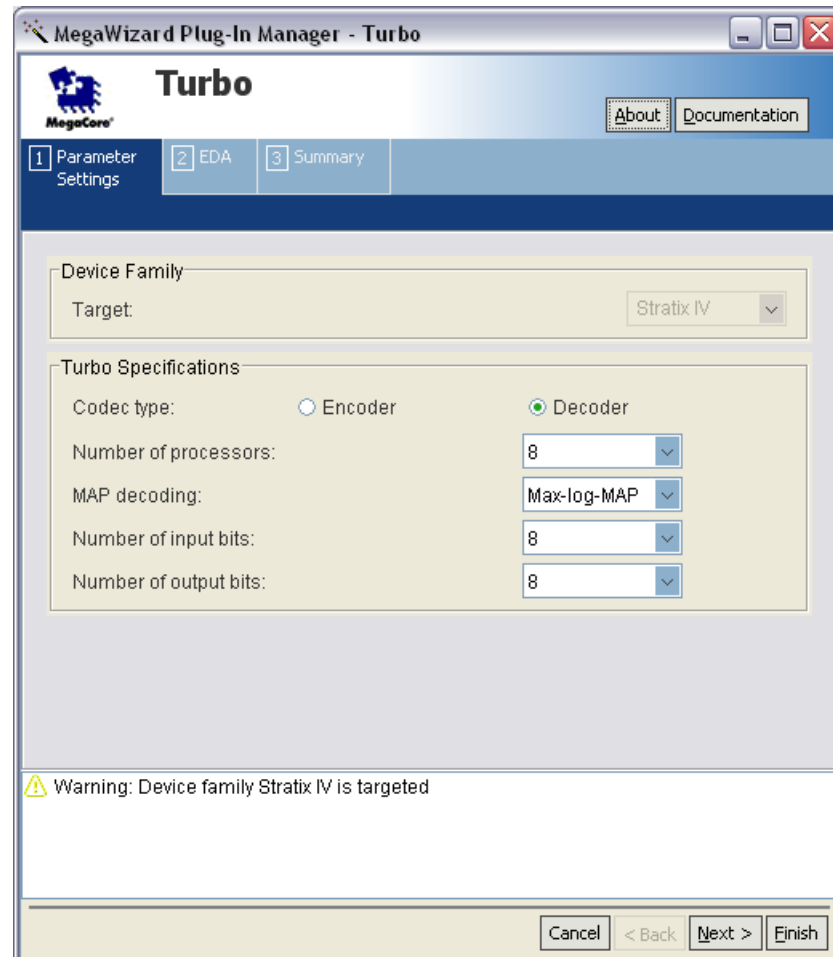
図 6. ターボ・メガファンクションの選択



4. デバイス・ファミリは、**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. メガファンクション・バリエーションのトップ・レベルのファイル名を指定し、Next をクリックして Parameter Editor の **Parameter Settings** タブを表示されます (図 7)。

図 7. Parameter Settings タブ



6. Parameter Editor を使用して、必要なパラメータを指定します。表 7 に、パラメータの説明を示します。

表 7. 3GPP LTE ターボ・パラメータ (その 1)

パラメータ	値	説明
ターゲット	Stratix IV、Stratix III、Stratix II GX、Stratix II、Cyclone III、Arria II GX、Arria GX	Quartus II プロジェクトを作成したときに指定されたターゲット・デバイス・ファミリーを表示します。
Codec タイプ	エンコーダ / デコーダ	エンコーダまたはデコーダを生成することを選択します。
エンジン数	2、4、8、16 (1)	デコーダで使用されるエンジンの数を選択します。
MAP デコーディング	MaxLogMAP、LogMAP (2)	利用可能なデコーディング・アルゴリズムのリストから選択します。
入力ビット数	4、5、6、7、8 (1)	デコーダへの入力ビット数を選択します。

表 7. 3GPP LTE ターボ・パラメータ (その 2)

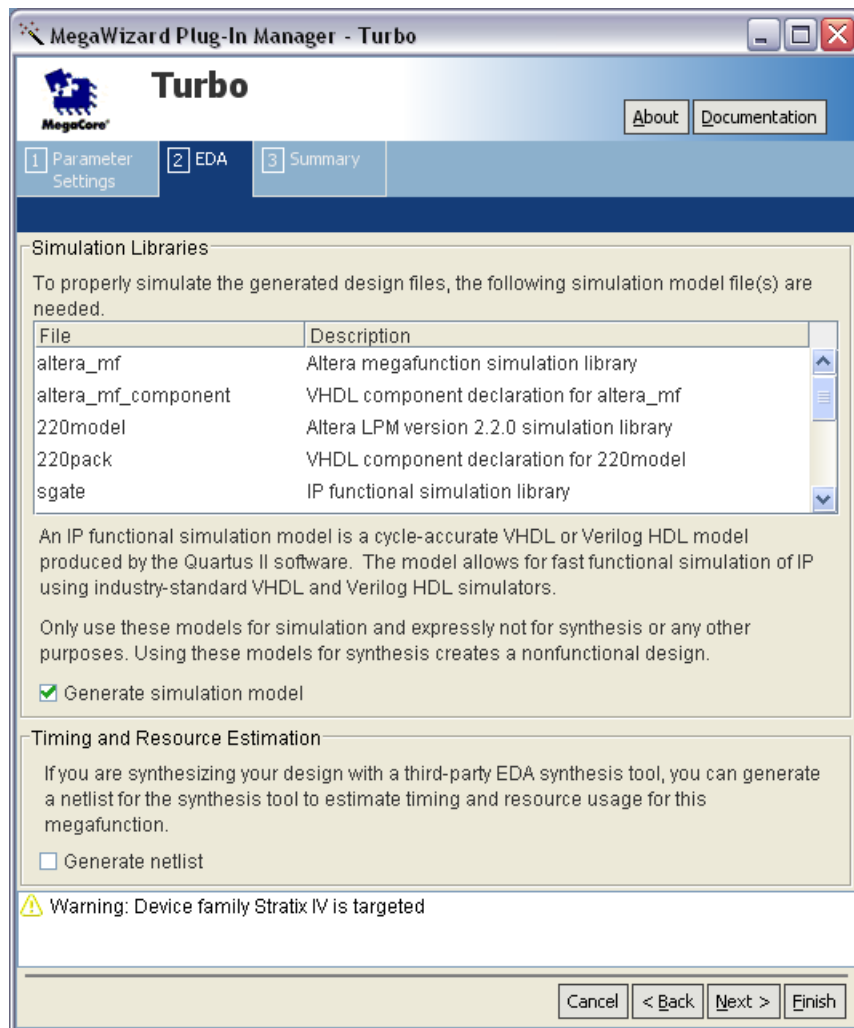
パラメータ	値	説明
出力ビット数	8 (3)	デコーダからの出力ビット数。

表 7 の注：


- (1) リファレンス・デザインは、6 または 8 の入力ビットと 2 または 8 のエンジンのためにテストされています。他のコンフィギュレーションを必要とする場合、アルテラにお問い合わせください。
- (2) LogMAP オプションは現在サポートされていません。
- (3) 8 ビット・インタフェース幅のみ現在サポートされています。

7. パラメータ化を完了するには、および **EDA** のタブを表示するために **Next** をクリックします (図 8)。

図 8. EDA タブ



8. EDA タブで、**Generate Simulation Model** をオンにします。

 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。



これらのシミュレーション・モデルは、シミュレーションの目的にのみ使用し、合成やその他の目的には使用しないでください。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。

9. 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メガ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、メガ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。合成ツールがこの機能をサポートしている場合は、**Generate netlist** をオンにします。
10. **Next** をクリックして、**Summary** タブを表示します (図 9)。

図 9. Summary タブ



11. **Summary** タブで、生成するファイルを選択します。グレイのチェック・マークは、自動的に生成されるファイルを示します。その他のファイルはすべてのオプションです。
12. **Finish** をクリックして、メガファンクションおよびサポートするファイルを生成します。ファイル生成フェーズを完了するには、数分かかる場合があります。生成の進行状況とステータスがレポート・ウィンドウに表示されます。

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

生成されるファイル

表 8 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。ファイルのタイプおよび名前は、パラメータ化時に指定するバリエーション名および HDL タイプによって異なります。例えば、デザインを Verilog HDL または VHDL のいずれで作成したかによってファイルの別のセットを作成されます。

表 8. 生成されるファイル (注 1)

ファイル名	説明
<variation name>.bsf	メガファンクション・バリエーションの Quartus II シンボル・ファイルです。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	メガファンクション・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、メガ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	メガファンクションに対して生成されるファイルとポートのリストを含む生成されるレポート・ファイルです。
<variation name>.qip	メガファンクション・バリエーション用の Quartus II プロジェクト情報が含まれています。
<variation name>.log	ログ・ファイル。
<variation name>.vhd, or .v	カスタム・メガファンクションの VHDL または Verilog HDL トップ・レベルの記述を定義するメガファンクション・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。Quartus II 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>.vho or .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>_bb.v	メガファンクション・バリエーションの Verilog HDL ブラック・ボックス・ファイルです。サードパーティ製 EDA ツールを使用してデザインを合成するときにこのファイルを使用します。
<variation name>_gb.v	一部のサードパーティ合成ツールに使用可能なタイミングおよびリソース見積りネットリストです。
<variation name>_nativelink.tcl	NativeLink シミュレーション・テストベンチ設定を Quartus II プロジェクトに割り当てる Tcl スクリプトです。
<variation name>_quartus.tcl	Quartus II ソフトウェアでコンパイルを実行できる Tcl スクリプトです。
<variation name>_tb.vhd, or .v	メガファンクション・バリエーション用の VHDL または Verilog HDL テストベンチ・ファイルです。VHDL トップ・レベルが選択されているときは VHDL ファイルが生成され、Verilog HDL トップ・レベルが選択されているときは Verilog HDL ファイルが生成されます。

表 8. 生成されるファイル (注 1)

ファイル名	説明
<variation name>_hw.tcl	簡単に SOPC Builder にターボ IP コアのバリエーションを統合するためのハードウェア Tcl ファイルです。SOPC Builder のライブラリでターボ IP コアの変動を表示するには、 Options タブの SOPC Builder の IP 検索パスに、プロジェクトのソース・ディレクトリを追加します。

表 8 の注：

- (1) <variation name> プリフィックスが MegaWizard Plug-In Manager で指定したベースの出力ファイル名を使用して自動的に追加されます。



Parameter Editor の **Summary** タブに **MegaCore function report file** のチェック・ボックスをオンにした場合、関数の変動に対して定義されているデザイン・ファイルおよびポート・リストを含む生成されるレポート・ファイルは HTML ファイルとして保存されます。

信号

生成されるレポートには、メガファンクション・バリエーションの信号も表示されます。

表 9 にターボ・エンコーダの信号を示します。

表 9. 3GPP LTE ターボ・エンコーダの信号

信号	入力/出力	説明
clk	入力	すべての内部レジスタをクロックするクロック信号です。
reset_n	入力	アクティブ Low のリセット信号です。メガファンクションは、常にデータを受信する前にリセットする必要があります。メガファンクションがリセットされていない場合、ターボ・エンコーダは、フィードバック信号に起因する予期しない結果が生じることがあります。
sink_blk_size	入力	受信ブロック・サイズを指定します。4 ページの表 1 に、パラメータ <i>Ki</i> を参照してください。
sink_sop	入力	受信パケットの開始をマークします。
sink_eop	入力	受信パケットの終了をマークします。
sink_valid	入力	sink_data でデータが有効である場合にアサートされます。sink_valid がアサートされていない場合、処理は sink_valid を再アサートされるまで停止されます。
source_ready	入力	データを受け入れることができる場合、ダウンストリーム・モジュールによってアサートされます。
sink_data	入力	入力データです。必要なデータ・オーダリングについては、5 ページの「入力データ・フォーマット」を参照してください。
sink_error	入力	入力側の Avalon ストリーミングのプロトコル違反を示すエラー信号です。sink_error ポート上の任意の非ゼロ値は、現在のデータ・ブロックを無視するようにターボ・エンコーダを引き起こします。このポートから受信した値は、数サイクル後の source_error 出力ポートに書き込まれます。
source_sop	出力	送受信パケットの開始をマークします。
source_eop	出力	送受信パケットの終了をマークします。
source_valid	出力	出力へのデータが有効である場合、メガファンクションによってアサートされます。
sink_ready	出力	メガファンクションがデータを受け入れることができることを示します。
source_error	出力	ソース側の Avalon-ST プロトコル違反を示すエラー信号： 00：エラーなし 01：SOP の欠落 10：EOP の欠落 11：予期しない EOP 他のエラーは 11 にマークします。
source_data	出力	出力データです。データ・オーダリングについては、5 ページの表 2 を参照してください。
source_blk_size	出力	送受信ブロック・サイズを指定します。このポートは、テストベンチのデバッグ・ポートであり、未接続のままにすることができます。

表 10 に、ターボ・デコーダ信号を示します。

表 10. 3GPP LTE ターボ・デコーダ信号 (その1)

信号	入力/ 出力	説明
clk	入力	すべての内部レジスタをクロックするクロック信号です。
reset_n	入力	アクティブ Low のリセット信号です。メガファンクションは、常にデータを受信する前にリセットする必要があります。メガファンクションがリセットされていない場合、ターボ・エンコーダは、フィードバック信号に起因する予期しない結果が生じることがあります。
sink_blk_size	入力	受信ブロック・サイズを指定します。4 ページの表 1 に、パラメータ <i>Ki</i> を参照してください。
sink_sop	入力	受信パケットの開始をマークします。
sink_eop	入力	受信パケットの終了をマークします。
sink_valid	入力	sink_data でデータが有効である場合にアサートされます。sink_valid がアサートされていない場合、処理は sink_valid を再アサートされるまで停止されます。
source_ready	入力	データを受け入れることができる場合、ダウンストリーム・モジュールによってアサートされます。
sink_data	入力	入力データです。必要なデータ・オーダリングについては、9 ページの表 4 を参照してください。
sink_error	入力	入力側の Avalon ストリーミングのプロトコル違反を示すエラー信号です。sink_error ポート上の任意の非ゼロ値は、現在のデータ・ブロックを無視するようにターボ・エンコーダを引き起こします。このポートから受信した値は、数サイクル後の source_error 出力ポートに書き込まれます。
sink_max_iter	入力	ハフの繰り返しの最大数を指定します。
sel_CRC24A	入力	現在のデータ・ブロックの必要な CRC タイプを指定します。 0 : CRC24A 1 : CRC24B
source_blk_id	出力	送受信のブロック ID を指定します。このポートは、テストベンチのデバッグ・ポートであり、未接続のままにすることができます。
source_sop	出力	送受信パケットの開始をマークします。
source_eop	出力	送受信パケットの終了をマークします。
source_valid	出力	出力へのデータが有効である場合、メガファンクションによってアサートされます。
sink_ready	出力	メガファンクションがデータを受け入れることができることを示します。
source_error	出力	ソース側の Avalon-ST プロトコル違反を示すエラー信号： 00 : エラーなし 01 : SOP の欠落 10 : EOP の欠落 11 : 予期しない EOP 他のエラーは 11 にマークします。
source_data	出力	出力データです。データ・オーダリングについては、18 ページの表 7 を参照してください。

表 10. 3GPP LTE ターボ・デコーダ信号 (その 2)

信号	入力/ 出力	説明
source_blk_size	出力	入力側の Avalon ストリーミングのプロトコル違反を示すエラー信号です。 sink_error ポート上の任意の非ゼロ値は、現在のデータ・ブロックを無視するようにターボ・エンコーダを引き起こします。このポートから受信した値は、数サイクル後の source_error 出力ポートに書き込まれます。
CRC_pass	出力	CRC が成功したかどうかを示します。 0 : 失敗 1 : パス
CRC_type	出力	現在のデータ・ブロックで使用された CRC タイプを示します。 0 : CRC24A 1 : CRC24B
source_iter	出力	ターボ・デコーダは、現在のデータ・ブロックの処理を停止した後に半分の反復回数を示しています。

デザインのシミュレーション

VHDL および Verilog HDL IP 機能シミュレーション・モデルとテストベンチを使用して、デザインをシミュレーションできます。

IP 機能シミュレーション・モデルは、指定した出力言語に応じて、**.vo** または **.vho** ファイルのいずれかです。シミュレーション環境で **.vo** または **.vho** ファイルをコンパイルして、メガファンクションのカスタム・バリエーションの機能シミュレーションを実行します。



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。

NativeLink を使用したサードパーティ・シミュレーション・ツールによるシミュレーション

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。

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を実行するには、以下のステップを実行します。

1. カスタム・メガファンクション・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ます。
2. Quartus II ソフトウェアの Tools メニューの **Options** タブで、サードパーティ製 EDA ツールへの絶対パスが設定されているかを確認します。
3. Processing メニューから **Start** を選択し、**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。Tcl スクリプト `<variation name>_nativelink.tcl` 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。そして **Tool Name** でシミュレータを選択し、**NativeLink Settings** で **Compile Test Bench** を選択して、**Test Benches** をクリックします。適切なテストベンチ設定が Quartus II プロジェクトに割り当てられたことを確認します。
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。



詳細は、「*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。

デザインのコンパイルおよびデバイスのプログラム

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ついて詳しくは、Quartus II Help の「*Setting up and Running a Compilation*」を参照してください。

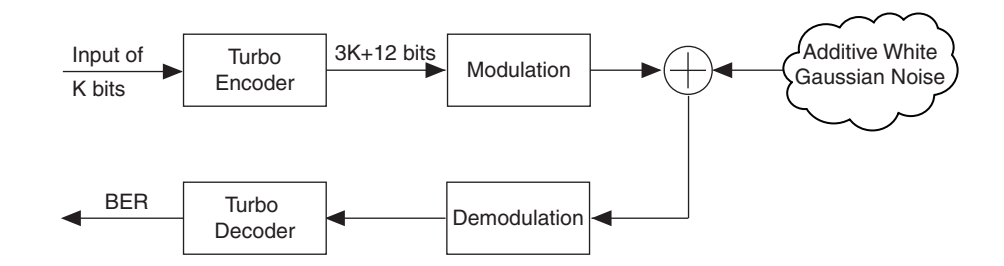
デザインをコンパイルした後、ターゲットされたアルテラ・デバイスをプログラムし、ハードウェア内でデザインを検証できます。デザインのプログラムのインストラクションについては、**Quartus II Help** を参照してください。

検証手法

次の手順では、3GPP LTE ターボのリファレンス・デザインの開発で使用する検証プロセスを説明します。

1. 浮動小数点シミュレーションモデルは、シミュレーションのフローを使用して、テスト・ベクタ生成器に接続されます (図 10)。

図 10. ターボ・デコーダの検証手法



2. 浮動小数点モデルの BER 性能は、BER の性能のリファレンスと比較されます。
3. 固定小数点モデルが開発され、その類似の BER 性能をリファレンスに達成されることをチェックするためにビット数などのパラメータを調整します。
4. RTL モデルは、VHDL で生成されます。RTL は、浮動小数点と固定小数点モデルに対して同じテスト・ベクタ生成のスイートを使用する ModelSim を使用してテストされます。
5. RTL の開発中に、結果は常に固定小数点モデルの結果で検証されます。すべての RTL ビルディング・ブロックは、RTL モデルをソフトウェア・モデルと一致していることを確認するために別々のテスト・ベンチを持っています。
6. RTL インプリメンテーションの開発中に、デザインは合成および配置配線ツールとして Quartus II ソフトウェアを使用します。リソース使用率およびパフォーマンスのために RTL を最適化されます。

参考文献

ターボ・コードおよび 3GPP 規格について詳しくは、以下の参考文献を参照してください。

1. *Avalon Interface Specifications*.
2. *AN 320: OpenCore Plus 評価機能によるメガファンクションの評価*.
3. *3GPP Technical Specification: Group Radio Access Network, Evolved Universal Terrestrial Radio Access, Multiplexing and Channel Coding (Release 8)*, TS 36.212 v8.3.0, May 2007.
4. C. Berrou, A. Glavieux、および P. Thitimajshima、*Near Shannon Limit Error-Correcting, Coding, and Decoding: Turbo Codes*, in Proceedings of the IEEE International Conference on Communications, 1993, pp. 1064-1070.

5. J. Vogt, A. Finger, *Improving the Max-Log-MAP Turbo Decode*, Electronics Letters, 36(23), 1937-1939, 2000。

Turbo Code Licensing Program

免責条項

自分自身と特定の他の当事者のための France Telecom は、Turbo Codes の技術をカバーする特定の知的財産権を主張し、およびこれらの権利をライセンスすることを決定するために、Turbo Codes Licensing Program と呼ばれるライセンス・プログラムに基づきます。この IP コアの供給は、France Telecom、TDF または GET が所有するすべての Turbo Codes でライセンスを提供せずまたは使用する権利を意味するものではありません。

Turbo Codes Licensing Program について詳しくは、以下のアドレスに France Telecom に連絡してください。

France Telecom R&D
 VAT/TURBOCODES
 38, rue du Général Leclerc
 92794 Issy Moulineaux
 Cedex 9
 France

改訂履歴

表 11 に、本資料の改訂履歴を示します。

表 11. 改訂履歴

日付	バージョン	変更内容
2011 年 1 月	2.1	<ul style="list-style-type: none"> ■ 新しいテンプレートに更新。 ■ SIC のサポートを追加。
2010 年 1 月	2.0	ターボ・デコーディングの f_{MAX} を更新。
2009 年 6 月	1.2	ターボ・デコーディングのサポートを追加。
2009 年 2 月	1.1.1	パケット・フォーマット・エラーの処理のセクションを追加。
2008 年 9 月	1.1	CRC に対する早期終端のサポートを追加。
2008 年 2 月	1.0	このアプリケーション・ノートの新版。