# Zynq-7000 AP SoC に信号処理 IP を実装して XADC サンプルを後処理する

XAPP1203 (v1.0) 2014 年 4 月 22 日

Mrinal J. Sarmah、Cathal Murphy

## 概要

このアプリケーション ノートは、『ザイリンクス All Programmable デバイスでのアナログ信号処理機能の効率的なインプリメンテーション』(WP442) [参照 1] の補足資料兼手引書となります。ホワイト ペーパーでは、All Programmable Abstraction を使用して、ザイリンクス FPGA および All Programmable SoC (AP SoC) にアナログ信号処理機能を実装するためのシンプルで簡単なデザイン フローを紹介しています。このアプリケーション ノートでは、ホワイト ペーパーのコンセプトを採用して信号処理 IP コアを作成し、Zynq-7000 AP SoC 上に完全なミックスド シグナル システムを構築する方法について詳しく説明します。

## はじめに

ほとんどのシステムは、監視および制御を目的として実世界と相互に作用できることが必要です。このために、光、熱、音などの実世界の刺激要因をアナログ電子信号に変換するセンサーがシステムには含まれます。これらのセンサーのアナログ出力は処理並びにデジタル化され、適切な情報がデジタル コントローラーやプロセッサへ伝搬されます。センサー出力のアナログ処理や調整は、精度要件および使用するセンサーの種類などの要因によって、さまざまな方法で行われます。一般的なアナログ機能には、次のものがあります。

- レベル変換 (バイポーラからシングルエンド)
- ゲイン/減衰
- 帯域幅の制限/フィルタリング/ノイズ削減
- ゲインおよびオフセット エラー キャンセレーション
- リニアライゼーション (線形化)

ほとんどのシステムはデジタル情報を処理するため、処理後のセンサー出力をデジタル化するためのアナログ/デジタルコンバーター (ADC) が必要です。ADC には多様な種類があり、一般的に分解能や処理速度で特徴付けられています。ADC はマイクロコントローラーにも組み込まれ、FPGA には XADC などの ADC が統合されています (すべての 7 シリーズ FPGA および AP SoC で)。近年のプロセス微細化に伴い、ADC 処理速度は大幅に高速化されました。また、サンプル レートが 1MSPS 以上の ADC を比較的に安価に購入できるようになりました。このように高速化したデジタル信号処理機能を最大限に活用して、ソリューションの性能を高めることができます。

ホワイト ペーパー『ザイリンクス All Programmable デバイスでのアナログ信号処理機能の効率的なインプリメンテーション』(WP442) では、ザイリンクスの All Programmable Abstraction を使用してザイリンクスの FPGA および AP SoC に一般的なアナログ信号処理機能を簡単かつ効率的に実装するコンセプトを紹介しています。この実装によって、サブシステムの設計がより正確かつ柔軟になり、安価に高速化を実現できることを説明しています。

このアプリケーション ノートでは、実際に高位合成 (HLS) ツールを使用してこれらのコンセプトをザイリンクスの FPGA または AP SoC に適用し、特定のアナログ信号処理を実現する方法について詳しく説明します。機能は次のとおりです。

- ローパス フィルタリング
- 間引き
- 線形化

このアプリケーション ノートでは、Vivado® Design Suite の IP インテグレーター (IPI) を使用してミックスド シグナル サブシステムを構築する方法について詳しく説明します。図 1 に、デジタル サブシステムのリファレンス デザインを示します。

© Copyright 2014 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.

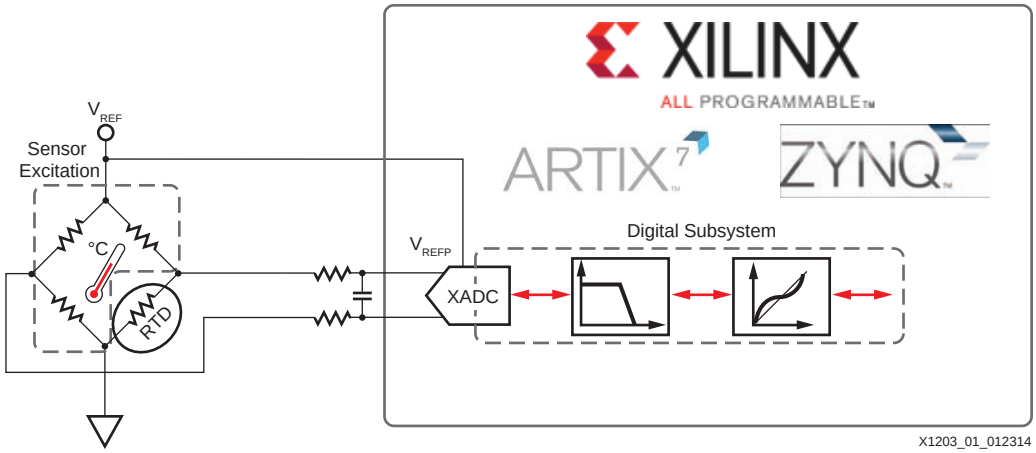



図 1：システムのブロック図

サブシステム全体または個々の IP コアを検証するには、UART ベースのアプリケーション (ハイパーターミナルまたは Tera Term) を使用できます。インプリメンテーション手順の詳細は、wiki ページ Zynq AMS Post Processing in PL を参照してください。

## ハードウェア デザインの概要

FPGA にはコンフィギュレーション可能な CLB (コンフィギャラブル ロジック ブロック) アレイがあるため、コスト効率の良い方法で特定デジタル アルゴリズムを実行できます。Zynq-7000 AP SoC のプログラマブル ロジック部分には、C/C++/RTL コードで記述された特定デジタル アルゴリズムを実行する CLB アレイがあります。Vivado Design Suite は、再利用可能で IP を中心とする方法でアルゴリズムの実装をサポートします。

Vivado 高位合成 (HLS) は、C または C++ 言語で記述されたアルゴリズムから RTL コードを生成するツールです。このツールは豊富なライブラリを使用して、C/C++ で記述された機能を RTL に変換します。ユーザーは、AXI4-Stream または AXI4-Lite インターフェイスのオプションを解釈するために適切な HLS 指示子を用いることができるため、生成された RTL を複数のシステム レベル デザインで再利用できます。

Vivado IP インテグレーター (IPI) は回路図ベースの接続を含むデザインで、IP のインスタンシエーションを可能にするツールです。IPI には、コネクション オートメーション機能とブロック オートメーション機能があり、デザインにおける適切な AXI インターコネクトのインスタンシエーションとマスター / スレーブ IP コアへの接続を自動化します。ユーザーは、Vivado HLS で生成した RTL のパッケージ化、インターフェイスの適切な定義、ユーザー ブロック デザインへの IP をインスタンシエート、コネクション オートメーション機能を使用したインターフェイスの接続が可能です。

リファレンス デザインには、センサー リニアライザー アルゴリズムが C 言語で実装されています。コードの一部を次に示します。

```
void axi_sensor_lin(ap_axiu<32,5,1,1> *input,
          ap_axiu<32,5,1,1> *output,
          unsigned int interp_matrix[1024], unsigned char gain, unsigned char offset, unsigned
char control)
 {
    AP_BUS_AXI_STREAMD(input, BUS_A);
    AP_BUS_AXI_STREAMD(output, BUS_B);

    unsigned short step_height, step_width, index, tmp_value, input_data;
    unsigned short interp_data, next_interp_data;

    input_data = (unsigned short)((input->data)&0xFFFF)>>4;
    step_width = (input_data % 4);
    index = input_data / 4;

    interp_data = (unsigned short)(interp_matrix[index]&0xFFFF);
    next_interp_data = (unsigned short)(interp_matrix[index+1]&0xFFFF);

    step_height = (-interp_data+next_interp_data);

    tmp_value = interp_data + step_height * step_width / 4;

    output->data = (unsigned int)((tmp_value+offset)*gain/128)&0xFFFF;

}
```

関数 axi_sensor_lin は、非線形 AXI4-Stream データ、リニアライザー係数、およびゲイン/オフセットを入力として使用し、線形化されたデータを出力します。リニアライザー係数やリニアライザー係数マトリクスの現在と次の値の距離測定に基づいて出力データを算出します。AP_BUS_AXI_STREAMD 指示子を使用して、入力/出力バス用に AXI4-Stream インターフェイスを推論します。Vivado HLS は、入力/出力データ用に AXI4-Stream インターフェイスを使用して RTL コードを生成します。この RTL コードは、Vivado ツールの IP パッケージャーを使用してパッケージ化できます。

図 2 に、リファレンス デザインのブロック図を示します。このデザインは、制御プロセッサと Zynq-7000 AP SoC のプログラマブル ロジック (PL) に実装された Analog Post Processing IP で構成されています。この IP コアは、PL にある DSP ブロックを使用します。

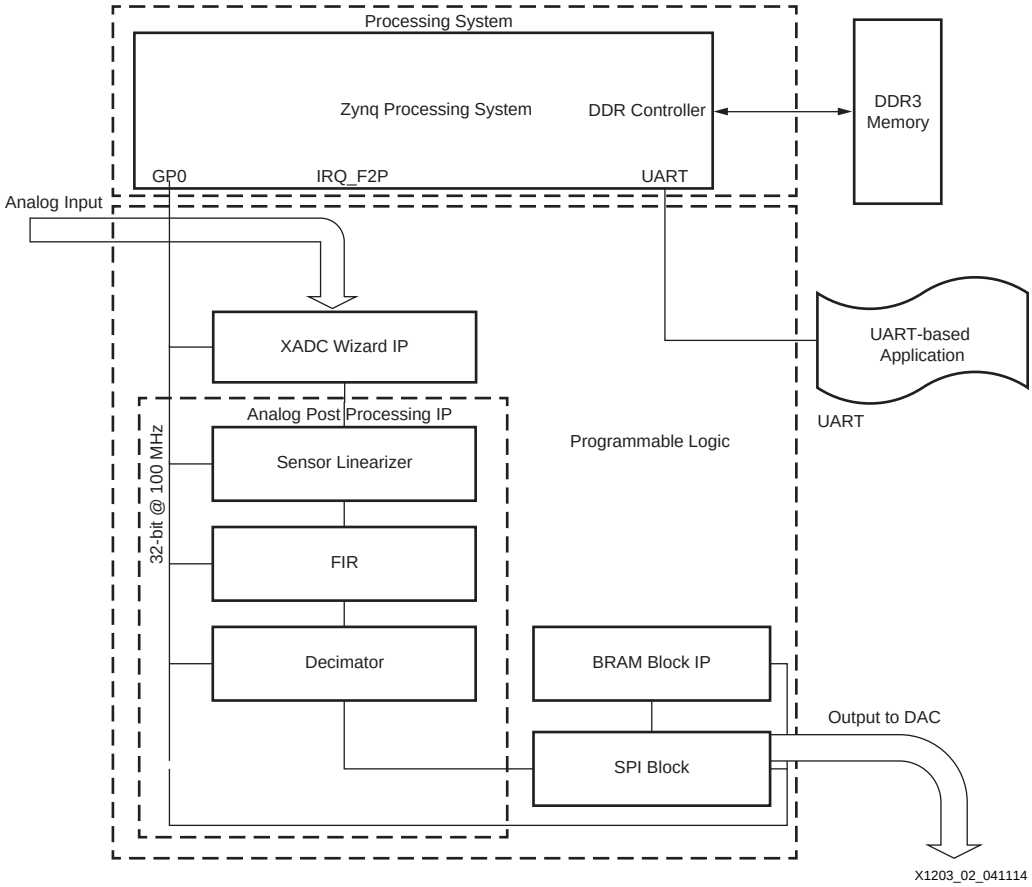



図 2 : ハードウェア ブロック図

Zynq-7000 AP SoC のデュアルコア ARM® Cortex™-A9 ベースのプロセッシング システム (PS) は、制御プロセッサとして機能し、デザインで使用されるさまざまなペリフェラル ブロックを初期化および設定します。

制御パスは、Zynq-7000 Processing System 7 (PS7) IP の汎用 (GP) ポートで始まる AXI4-Lite インターフェイス上に構築されます。デザイン内のデータ パスは、相互接続されたブロック間に AXI4-Stream インターフェイスを確立します。

データ フローのシーケンスは次のとおりです。

1. 制御ソフトウェアが XADC Wizard IP を初期化し、Sensor Linearizer (センサー リニアライザー )、FIR (無限インパルス応答 ) フィルター、および Decimator (デシメーター) ブロックを初期化します。
2. アナログ入力のスティミュラス信号が XADC $V_P/V_N$ インターフェイス ピンでサンプリングされます。
3. アナログ サンプルが XADC によって 16 ビットのデジタル コードに変換され、XADC Wizard IP の AXI4-Stream インターフェイスで有効になります。
4. リニアライザー　ブロックが制御ソフトウェアによって有効化されている場合、サンプルがこのブロックで処理されます。

5. 線形化された出力は FIR フィルター ブロックに入り、FIR フィルターが制御ソフトウェアによって有効化されている場合、このブロックでフィルタリングされた出力が生成されます。
6. デシメーター ブロックがフィルタリングされた出力を処理し、プログラム可能な間引き係数でサンプルを間引きします。
7. デシメーター出力は SPI コアへ入力され、SPI コアはサンプルを BRAM BLOCK IP へ送信します。また SPI コアは、XADC ブロックへスティミュラスを提供する AD5065 DAC にもサンプルを送信します。
8. 制御ソフトウェアは、AXI4-Lite インターフェイスを使用して BRAM Block IP からサンプルを読み出します。

## XADC Wizard IP

XADC Wizard IP は、AXI4-Lite インターフェイスを使用して完全にコンフィギュレーション可能です。すべての XADC レジスタは、同じ順序で AXI4 インターフェイスに直接マップされます。IP コアのメモリ空間へのすべての読み出し/書き込み動作は、IP コア自体へ直接実行されます。

## Sensor Linearizer (センサー リニアライザー )

この IP は、線形補間ルックアップ テーブル (LUT) を実行します。アルゴリズムを使用して、理想値と実際の結果の最大差を見つけ出します。実際の非線形応答に対して、差分を加算/減算して線形応答を得ます。非線形応答の最初の値は、理想の応答から最も大きく離れています。これは、出力が大きく変化するのに対して入力はほとんど変わらないためです。

図 3 に、このアプリケーション ノートで使用される Sensor Linearizer ブロックの図を示します。このブロックの 2 つの主なインターフェイスは、サンプルのデータ パスに使用される AXI4-Stream インターフェイスと制御インターフェイスとして使用される AXI4-Lite インターフェイスです。AXI4-Lite インターフェイスは、ユーザーが生成した係数リストを使用して BRAM をアップデートします。Sensor Linearizer ブロックの係数は、アナログ スティミュラスを生成しているセンサーの非線形関数の性質によって決定されます。AXI4-Lite インターフェイスは、AXI4-Stream インターフェイスとは異なるクロック周波数で動作できます。スレーブ AXI4-Stream インターフェイスで受信されたサンプルは、リニアライザーの係数を使用して補正され、線形ビヘイビアを示します。

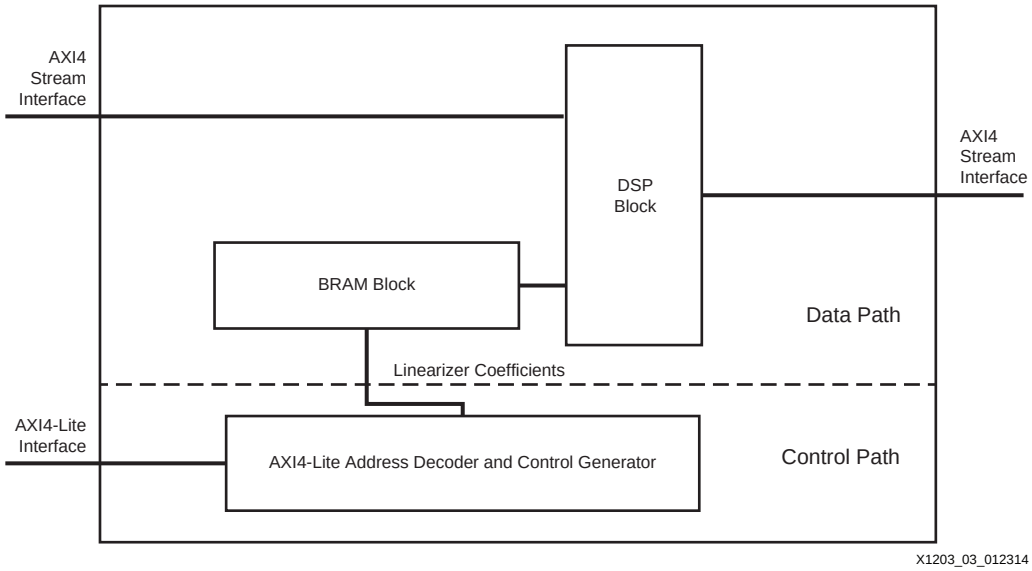



図 3 : Sensor Linearizer ブロック

## FIR フィルター

FIR フィルター IP コアは、フィルター係数で設定する必要があります。0x4 レジスタへの書き込みによって、係数のシフト レジスタへ値が書き込まれます。0x0 レジスタの下位 6 ビットには、フィルターの係数の数を書き込む必要があります。ビット位置 16 はソフトウェア リセット ビットで、17 はバイパス ビットとなります。図 4 に FIR フィルターのブロック図を示します。

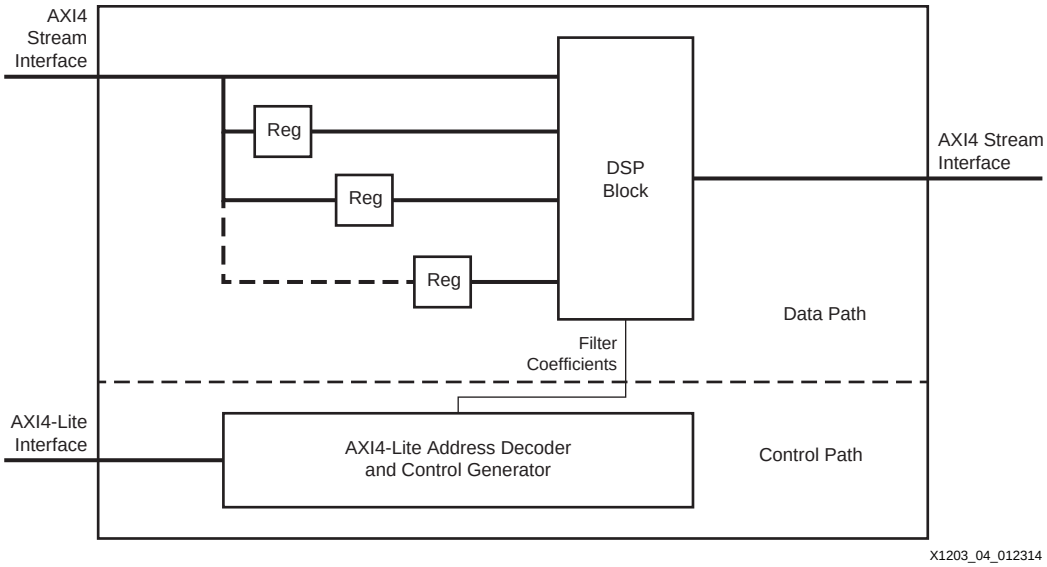



図 4： FIR フィルター ブロック

FIR フィルター ブロックの 2 つの主なインターフェイスは、データ パスに使用される AXI4-Stream インターフェイスと制御パスに使用される AXI4-Lite インターフェイスです。FIR カットオフ周波数の要件に基づいてユーザーが生成する FIR フィルター係数は、AXI4-Lite インターフェイスを使用してプログラム可能です。AXI4-Stream インターフェイスに現れるサンプルは、係数によって遅延および乗算されて出力サンプルとなります。

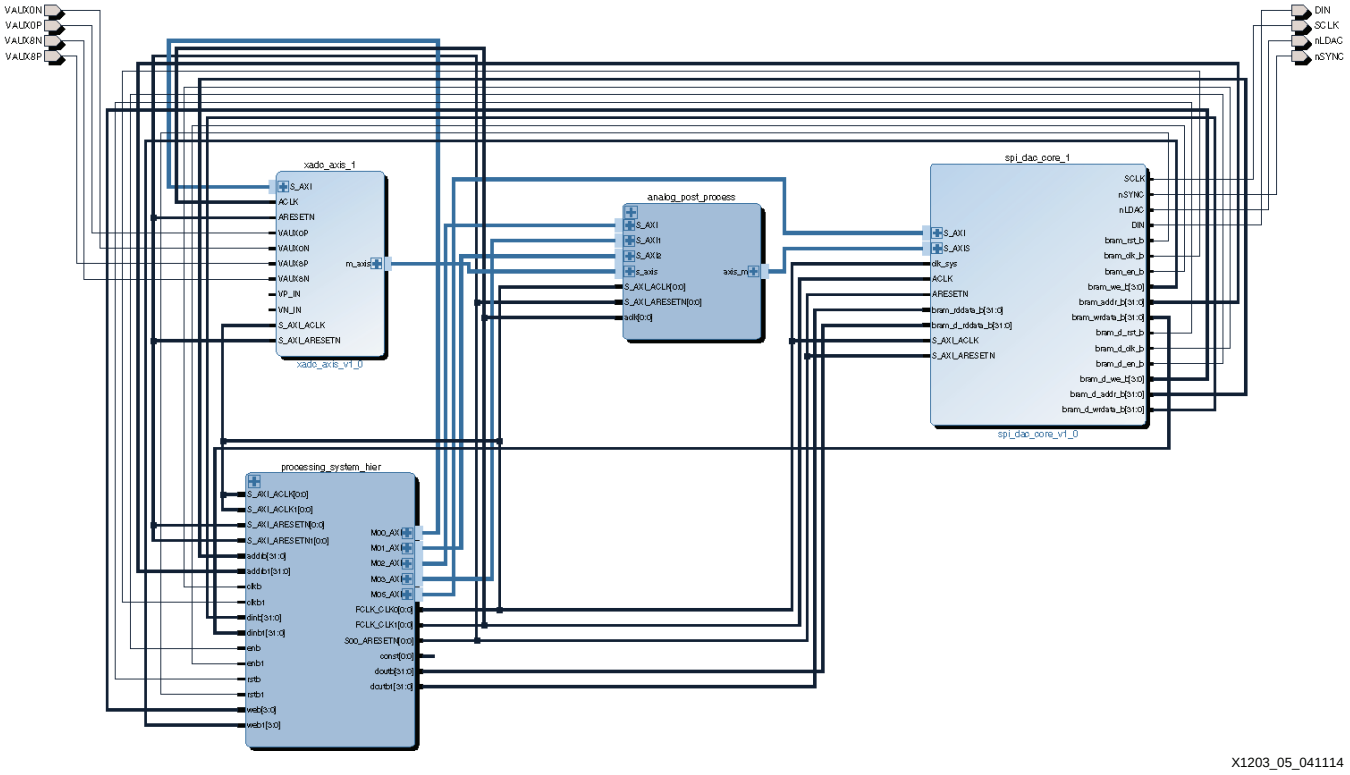
## Decimator (デシメーター ) ブロック

この IP には、すべてのチャネルに対応するカウンターが 1 つあり、カウンターが 0 に到達した場合のみ値を出力します。カウンターの初期値は、AXI-Lite インターフェイスを介してプログラム可能です。デシメーター ブロックは、FIR フィルター ブロックと連動する必要があります。出力サンプルは、プログラムされた間引き値に基づいて間引き処理されます。

上記すべてのブロックは、AXI4-Lite インターフェイスを使用してバイパスできます。各ブロックは、関連する AXI4-Lite インターフェイスを介して適用されるソフトウェア制御のリセット信号を用いてリセットできます。

リファレンス デザインは、Vivado IPI フローを使用してブロック デザインを作成しています。ブロック デザインが作成された後、IP 固有のラッパー ファイルをインスタンシエートするハードウェア ラッパーが生成されます。

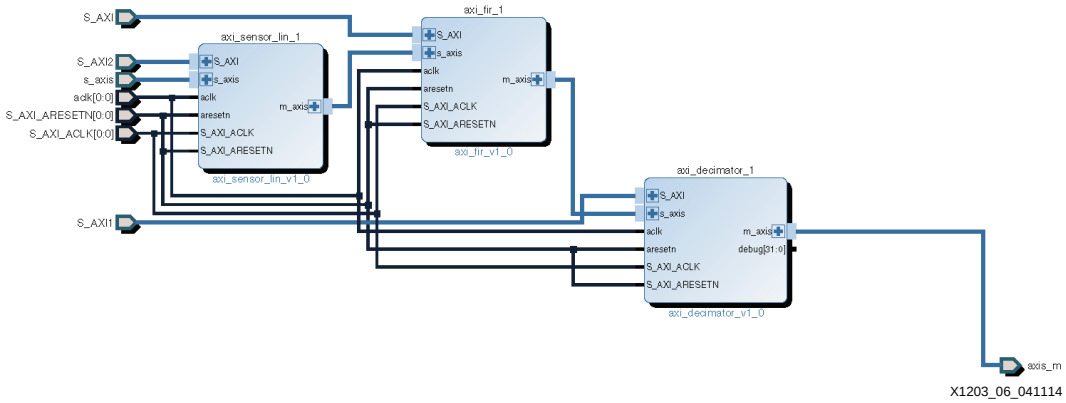
図 5 に、このアプリケーション ノートで使用される IPI デザインを示します。




X1203_05_041114

図 5 : IPI ブロック図

図 6 に、Analog Post Processing ブロックの拡大図を示します。




X1203_06_041114

図 6 : Analog Post Processing ブロックの拡大図

表 1 に、ハードウェア デザインで使用される IP コアを示します。

表 1： ハードウェア デザインで使用される IP コア

| IP 名 | 説明 | 設定 |
|---|---|---|
| Processing System-7 | Processing System-7 ブロックをインスタンシエートするラッパー ファイルを生成します。 | Processing System-7 IP ウィザードであらかじめ設定された ZC702 のデフォルト値で設定されています。 |
| AXI Interconnect IP | Processing System-7 IP からの AXI3 トランザクションを XADC Wizard IP 用の AXI4 トランザクションに変換します。 | マスター インターフェイスとスレーブ インターフェイスがそれぞれ 1 つに設定されています。 |
| Proc Sys Reset | ペリフェラルおよび AXI Interconnect IP へリセットを適用します。 | AXI Interconnect および XADC Wizard IP へリセットを適用するように設定されています。 |
| AXI BRAM Controller IP | 信号処理ブロックから後処理されたサンプルを収集し、ブロック RAM に格納します。格納されたサンプルは、Processing System-7 IP によって読み出されます。 | 完全なデュアルポート コンフィギュレーション用に設定されています。 |

表 2 に、PL に含まれるメモリ マップされたペリフェラルとアドレス マップを示します。

表 2： ペリフェラルとアドレス マップ

| PL のペリフェラル | アドレス マップ (16 進数) |
|---|---|
| XADC Wizard IP | 70000000-7000FFFF |
| FIR フィルター ブロック | 70600000-7060FFFF |
| Sensor Linearizer ブロック | 60000000-6000FFFF |
| Decimator ブロック | 70A00000-70A0FFFF |
| SPI コア | 70100000-7010FFFF |
| BRAM Control IP 1 | 40000000-4001FFFF |
| BRAM Control IP 2 | 42000000-4201FFFF |

## ソフトウェア アーキテクチャ

このアプリケーション ノートで使用するソフトウェア アプリケーションは、ザイリンクスのソフトウェア開発キットで利用できるスタンドアロンのオペレーティング システムをベースにしています。ソフトウェア アプリケーションは、デザイン ブロックのコンフィギュレーション パラメーターを制御し、周期的な間隔で AXI BRAM Controller からデータを取得します。

次の手順は、デザインのデータ フローを説明しています。ソフトウェア プログラムがこれを実行します。

1. このアプリケーション ノートで使用されるペリフェラルを初期化する
2. FIR フィルターのデフォルト係数値を設定する
3. Liniarizer ブロックのデフォルト係数値をロードする
4. Decimator ブロックをデフォルトの間引き値に設定する
5. while ループで UART ベースのアプリケーションから入力されるコマンドを待つ
6. UART アプリケーションからのコマンドをデコードして実行する

UART アプリケーションは、デザインで使用されるフィルター係数、リニアライザー係数、間引き係数値を設定します。UART アプリケーションのコマンドは、表 3 の形式を採用します。

**表 3 : UART アプリケーション コマンドの形式**

| コマンド | 説明 | 形式 |
|---|---|---|
| デザインから値を読み出す | 特定アドレスの値を読み出す | R AAAA<br>AAAA は、アドレスを 16 ビットの 16 進数形式で示します。 |
| デザインへ値を書き込む | 特定アドレスへ値を書き込む | W AAAA DDDD<br>AAAA は、アドレスを 16 ビットの 16 進数形式で示します。<br>DDDD は、書き込まれるデータを 16 ビットの 16 進数形式で示します。 |

次に、このアプリケーション ノートで使用される API 関数について説明します。すべての API は、`ams_xilinx_demo_hls.c` ファイルで定義されています。

### void config_fir (int_base_addr)

この API は、FIR フィルターの係数を設定します。

### void config_xadc (void)

この API は、XADC コア レジスタを設定します。コアは、XADC レジスタをプロセッサ メモリへ直接マップします。

### void decimation_config (XDecimator_XDecInstancePtr、int dec_value、int num_channel)

この API は、特定のデシメーター インタンスの固有チャネルを指定した間引き値に変更します。

### int decimation_initialization (void)

この API は、2 つの Decimator コアを同じ値に設定します。32 ビット ワードで 4 チャネルの間引き値を表します (MSB) 3-2-1-0 (LSB)。

### void fill_memory_ramp (void)

この API は、SPI - DAC メモリにランプを挿入します。 メモリ内のランプ数は、この API でプログラム可能です。

void fill_memory_sine (int f)
この API は、SPI - DAC メモリに周波数 *f* の正弦曲線を挿入します。データは、正の全振幅になります。

### int linearization_initialization (void)

この API は、線形化係数を設定します。LUT はシステム メモリに直接マップされます。

入力信号が x_x の場合、線形化 LUT は sqrt(y) → sqrt(x_x) = x となる必要があります。LUT には 1,024 のポジションがあり、入力データは $2^{12}$ です。つまり、LUT には 4 つのデータ空間に分けられた値があります (4096/1024 = 4)

### int main ()

これは、リファレンス デザインのメイン関数です。サンプル デザインは、XADC と FIR コアを設定し、シリアル ポートを使用して一部のパラメーターを変更するための基本的なシェルを提供します。

### void toggle_bypass_decimator ()

この API は、Decimator コアのバイパス モードをトグルします。

### void toggle_bypass_fir (unsigned int base_addr)

この API は、FIR コアのバイパス モードをトグルします。

### void toggle_bypass_lin ()

この API は、Sensor Linearization コアのバイパス モードをトグルします。

### void toggle_reset_fir (unsigned int base_addr)

この API は、FIR コアのリセット モードをトグルします。

**注記**：ユーザーは、TeraTerm アプリケーションを使用してハードウェア デザインをコンフィギュレーションできます。サポートされる UART の一覧は、ZIP ファイルの readme ファイルを参照してください。

## ハードウェア要件

このデザインは、ZC702 評価プラットフォームを使用してテストできます。ZC702 評価ボードに AMS101 カードを接続して外部データを取得する場合は、『AMS101 評価カード ユーザー ガイド』(UG886) [参照 2] の説明に従ってください。外部信号は XADC ブロックの専用チャネル ($V_P/V_N$) に接続でき、取得したサンプルは UART アプリケーションを使用して読み出すことができます。

## 比較分析

このセクションでは、FIR フィルター IP と Sensor Linearizer IP のアナログ インプリメンテーションとデジタル インプリメンテーションを比較します。この解析は、MATLAB を使用した FIR フィルターと Sensor Linearizer のアナログおよびデジタル インプリメンテーションに基づいており、結果は実際のハードウェア インプリメンテーションによって多少異なる場合があります。

### FIR フィルター IP

FIR フィルター IP のデジタル インプリメンテーションでは、アナログ コンポーネントでは実装の難しい、非常に高い次数のフィルター IP を実装できます。たとえば、FPGA には 63 次の FIR フィルターを簡単かつ柔軟に実装できますが、アナログ インプリメンテーションの場合は非常に複雑になります。デジタル フィルターは、よりよいロールオフと阻止帯域の減衰を実現します。

図 7 に、63 次のデジタル フィルターと 2 次の Sallen-Key フィルターの振幅特性を示します。図で確認できるとおり、デジタル フィルターの方がロールオフや阻止帯域の減衰が好ましいと言えます。

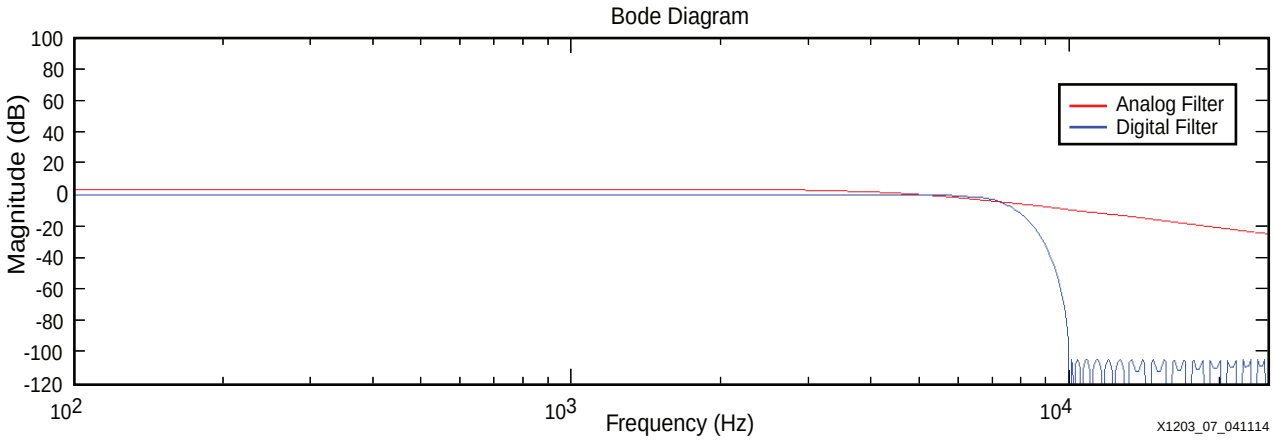



図 7：デジタルおよびアナログ フィルターの振幅特性

## Sensor Linearizer IP

デジタル ドメインでは、非常に効率的かつ正確に線形化が行われます。デジタル ドメインに Sensor Linearizer IP を実装した場合のメリットは、アルゴリズムの実装に必要なリソースが少なくて済むこと、補間誤差が少ないこと、そして精度が高いことです。非線形の補正以外にも、デジタル アプローチを取ることでゲインおよびオフセット エラーをなくすことができます。

図 8 に、線形化前のアナログ入力 (Non-Linear Sensor)、理想的な出力 (Ideal Output)、およびリニアライザー係数の生成に使用される線形化の逆関数 (Inverse Function) を示します。

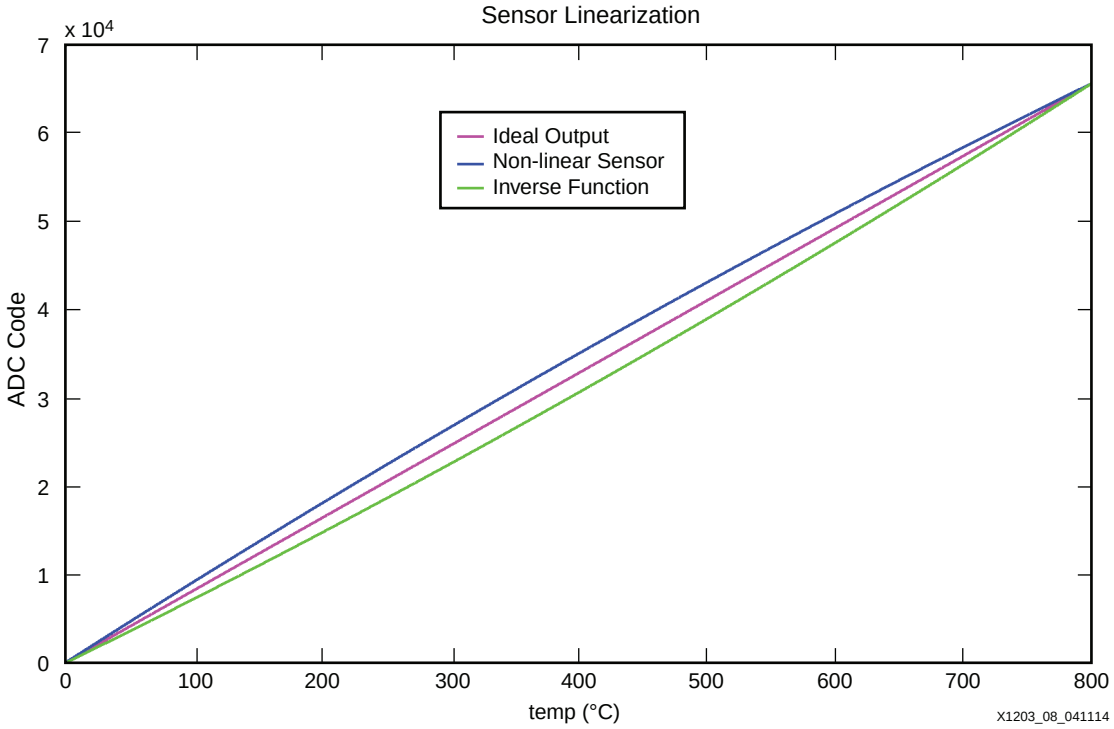



図 8 : Sensor Linearization の関数

図 9 には、理想と補間された線形関数の差の測定に基づいて計算されるデジタル線形化の実装におけるエラー プロットを示しています。

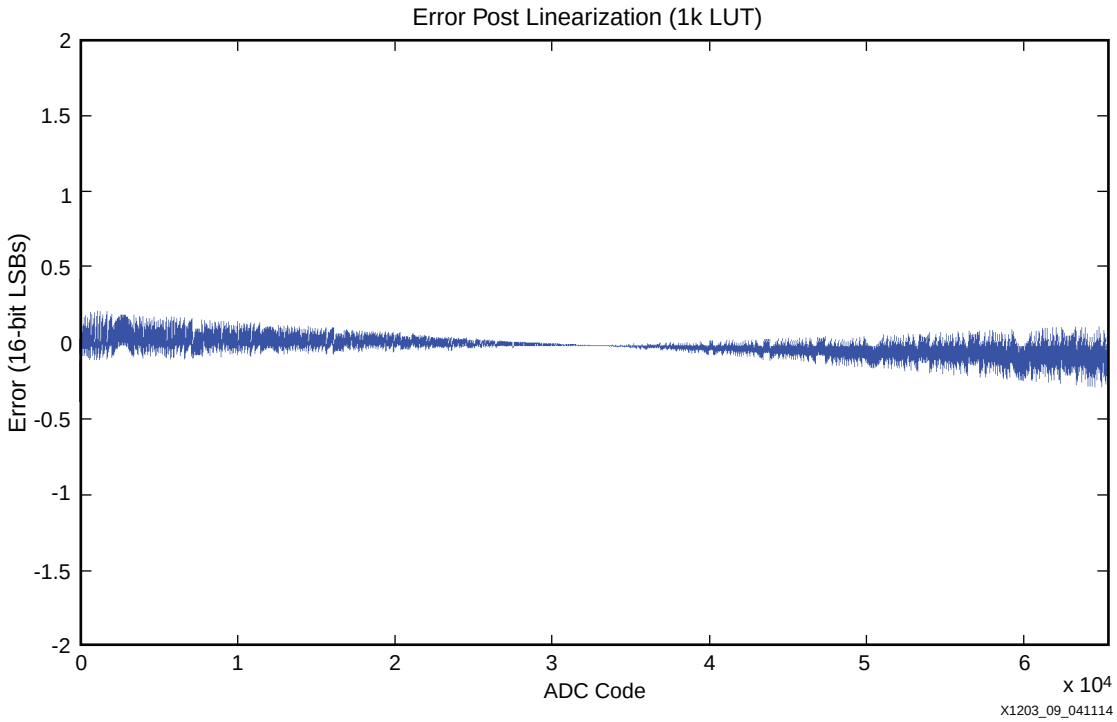



図 9 : デジタル Sensor Linearization IP のエラー測定

計算には、式 1 に示す多項式が使用されます。

$$RRTD = R0[1 + AT + BT2 + C(T - 100)T3] \quad \text{式 1}$$

説明 :

R0 は、0°C で 100Ω 抵抗 (Pt100)

$A = 3.9083 \times 10^{-3}$

$B = -5.775 \times 10^{-7}$

$C = 0$ (T > 0°の場合)、$C = -4.23225 \times 10^{-12}$ (T < 0°C の場合)

『Analog linearization of resistance temperature detectors』[参照 7] で示される 0.11% と比較して、デジタルでのエラーは 0.00003% として算出されます。

## 実験結果

XADC は、サンプリング周波数が 961.54kSPS として設定されています。つまり、可能な最大信号帯域幅は 480kHz であることを意味します。実験用セットアップでは、外部の正弦波ジェネレーターを使用して複数の周波数レベルが生成されており、結果となる信号品質の測定基準が計算されています。

FIR フィルター IP は、カットオフ周波数が 10kHz に設定された 63 次 FIR フィルター用に生成されたフィルター係数を使用して検証されました。デザインで FIR フィルターを使用した場合、周波数がカットオフ周波数内のときには信号対ノイズ比 (SNR) の向上が顕著に現れることを確認しました。Sensor Linearizer IP コアによって、AMS101 評価カードの AD5065 DAC で生成される二次非線形信号が補正されることも確認しました。

表 4 に、リファレンス デザインのリソース使用状況を示します。これらの数には、AXI4-Lite Interconnect IP を実装する際のロジックに使用されるリソースも含まれています。

表 4：リソース使用状況

| デザインの ブロック図 | フリップフロップ | LUT | DSP スライス | ブロック RAM |
|---|---|---|---|---|
| FIR フィルター | 324 (0.3%) | 523 (0.9%) | 1 (0.5%) | 0 |
| Linearizer ブロック | 442 (0.4%) | 293 (0.5%) | 1 (0.5%) | 1 (0.7%) |
| Decimator | 688 (0.6%) | 823 (1.5%) | 0 | 0 |
| XADC ブロック | 31 (0.02%) | 47 (0.08%) | 0 | 1 (0.7%) |

## リファレンス デザイン

このアプリケーション ノートのリファレンス デザインは、次のリンクからダウンロードできます。

https://secure.xilinx.com/webreg/clickthrough.do?cid=356125

readme ファイルの指示に従って、ハードウェアとソフトウェア コードを作成します。インプリメンテーション手順の詳細は、wiki ページ Zynq AMS Post Processing in PL を参照してください。

表 5 に、リファレンス デザインの詳細を示します。

表 5：リファレンス デザインの詳細

| パラメーター | 説明 |
|---|---|
| 全般 | |
| 開発元 | ザイリンクス |
| ターゲット デバイス (ステッピング レベル、ES、プロダクション、スピード グレード) | Zynq-7000 AP SoC |
| ソース コードの提供 | あり |
| ソース コードの形式 | C |
| 既存のザイリンクス アプリケーション ノート/リファレンス デザイン、CORE Generator ツール、サードパーティからデザインへのコード/IP の使用 | あり |
| シミュレーション | |
| 論理シミュレーションの実施 | なし |
| タイミング シミュレーションの実施 | なし |
| 論理シミュレーションおよびタイミング シミュレーションでのテストベンチの利用 | なし |
| テストベンチの形式 | N/A |
| 使用したシミュレータ/バージョン | N/A |
| SPICE/IBIS シミュレーションの実施 | N/A |
| インプリメンテーション | |
| 使用した合成ツール/バージョン | Vivado Design Suite 2013.4 |
| 使用したインプリメンテーション ツール/バージョン | Vivado Design Suite 2013.4 |
| スタティック タイミング解析の実施 | あり |
| ハードウェア検証 | |
| ハードウェア検証の実施 | あり |
| 使用したハードウェア プラットフォーム | ZC702 評価ボード |

## まとめ

このアプリケーション ノートでは、ADC からのサンプルを後処理して環境ノイズを除去する、コスト効率の高い方法を説明しました。ここで使用されるデザイン ブロックは、標準の AXI インターフェイスに準拠した DSP ブロック ベースの軽量ソリューションです。IP コアは、XADC サンプルを後処理する手段として、ユーザー デザインで再利用できます。Vivado IPI フローでは、回路図ベースの環境で簡単に再利用が可能になり、RTL を記述する必要がありません。

## 付録：LabView シリアル インターフェイス

表 6 では、さまざまな UART アプリケーション操作のレジスタ マップについて説明します。UART を介す送信フォーマットは、アドレスの後ろにデータ バイトが続きます。アドレス制限は 16 ビットで、データ値も 16 ビットです。

書き込み (16 ビットのアドレス、16 ビットの書き込みデータ) の場合は、OK 肯定応答信号がアサートされます。

読み出しの場合は、R AAAA の UART コマンドの後に読み出しデータが続きます。

表 6：レジスタ マップ

| UART アプリケーションのコマンド | UART の送信 | | UART の受信 |
|---|---|---|---|
| | アドレス | データ | |
| 接続の確立：デザイン バージョン レジスタ (読み出し専用) | 0x0000 | | 読み出し専用のデザイン バージョン レジスタです。16 ビットのデータを返します。デザイン バージョン (D11-8).(D7-0) となります。<br>UART コマンド：R 0000 |
| タイム ドメイン/周波数ドメイン タブ：<br>データの収集 (読み出し専用) | 0x0001 | - | UART アプリケーションでさらに処理するため、FPGA は 4,096 サンプル (8KB) のロー データを送信します。<br>同時サンプリング モードの場合、デュアル ディスプレイが必要となり、FFT は VAUX[0] チャネルに対応する偶数サンプルと VAUX[8] チャネルに対応する奇数サンプルを含む 8,192 サンプルを返します。<br>UART コマンド：R 0001 |

## 参考資料

この文書の参考資料は次のとおりです。


1. 『ザイリンクス All Programmable Device でのアナログ信号処理機能の効率的なインプリメンテーション』(WP442)
2. 『AMS101 評価カード ユーザー ガイド』(UG886)
3. AMS101 評価カードの資料 japan.xilinx.com/support/index.html/content/xilinx/en/supportNav/boards_and_kits/accessory_boards/ams101-evaluation-card.html
4. 『Zynq-7000 All Programmable SoC テクニカル リファレンス マニュアル』(UG585)
5. 『7 シリーズ FPGA および Zynq-7000 All Programmable SoC の XADC 12 ビット 1MSPS デュアル アナログ - デジタル コンバーター ユーザー ガイド』(UG480)
6. 『7 シリーズ FPGA AMS ターゲット リファレンス デザイン ユーザー ガイド』(UG960)
7. 『Analog linearization of resistance temperature detectors』 www.ti.com/lit/an/slyt442/slyt442.pdf

## 改訂履歴

次の表に、この文書の改訂履歴を示します。

| 日付 | バージョン | 内容 |
|---|---|---|
| 2014 年 4 月 22 日 | 1.0 | 初版 |

## Notice of Disclaimer

The information disclosed to you hereunder (the “Materials”) is provided solely for the selection and use of Xilinx products.To the maximum extent permitted by applicable law:(1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same.Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications.You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent.Certain products are subject to the terms and conditions of Xilinx’s limited warranty, please refer to Xilinx’s Terms of Sale which can be viewed at http://www.xilinx.com/legal.htm#tos; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx.Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx’s Terms of Sale which can be viewed at http://www.xilinx.com/legal.htm#tos.

## Automotive Applications Disclaimer

XILINX PRODUCTS ARE NOT DESIGNED OR INTENDED TO BE FAIL-SAFE, OR FOR USE IN ANY APPLICATION REQUIRING FAIL-SAFE PERFORMANCE, SUCH AS APPLICATIONS RELATED TO:(I) THE DEPLOYMENT OF AIRBAGS, (II) CONTROL OF A VEHICLE, UNLESS THERE IS A FAIL-SAFE OR REDUNDANCY FEATURE (WHICH DOES NOT INCLUDE USE OF SOFTWARE IN THE XILINX DEVICE TO IMPLEMENT THE REDUNDANCY) AND A WARNING SIGNAL UPON FAILURE TO THE OPERATOR, OR (III) USES THAT COULD LEAD TO DEATH OR PERSONAL INJURY.CUSTOMER ASSUMES THE SOLE RISK AND LIABILITY OF ANY USE OF XILINX PRODUCTS IN SUCH APPLICATIONS.

本資料は英語版 (v1.0) 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。
資料によっては英語版の更新に対応していないものがあります。
日本語版は参考用としてご使用の上、最新情報につきましては、必ず最新英語版をご参照ください。

この資料に関するフィードバックおよびリンクなどの問題につきましては、jpn_trans_feedback@xilinx.com までお知らせください。いただきましたご意見を参考に早急に対応させていただきます。なお、このメール アドレスへのお問い合わせは受け付けておりません。あらかじめご了承ください。