XAPP1182 (v1.0)
2013 年 11 月 18 日

Zynq-7000 AP SoC プロセッシング システムと XADC AXI インターフェイスを使用したシステム モニタリング

著者 : Mrinal J. Sarmah、Radhey S. Pandey

概要

このアプリケーション ノートは、システム モニター アプリケーションにザイリンクスのアナログ/デジタル コンバーター (XADC) を使用する方法について説明します。XADC からシステム制御情報を取得するために Zynq®-7000 All Programmable SoC (AP SoC) プロセッシング システムの AXI 汎用 (GP) ポートへ接続される AXI4-Lite(1) インターフェイスは、XADC Wizard IP が提供します。XADC ブロックは、あらかじめ設定したイベントに基づいてトリガーする専用のアラーム出力信号を提供します。このアプリケーション ノートでは、Zynq-7000 All Programmable SoC デバイス上の ARM® Cortex™-A9 CPU で実行する Linux アプリケーションを使用し、このアプリケーションが XADC のアラームしきい値を制御してアラーム出力をモニターします。

はじめに

Zynq®-7000 ファミリは、ザイリンクスの All Programmable SoC アーキテクチャで構成されています。この製品は、豊富な機能を備えたデュアルコア ARM Cortex-A9 ベースのプロセッシング システム (PS) とザイリンクスの 28nm プログラマブル ロジック (PL) を 1 つのデバイスに組み合わせたものです。PS は ARM Cortex-A9 CPU を中核として、オンチップ メモリ、外部メモリ インターフェイス、幅広い周辺接続インターフェイスを備えています。

XADC が PL にインスタンシエートされている場合、Zynq-7000 AP SoC PS と 12 ビット 17 チャネル 1MSPS のアナログ/デジタル コンバーターであるこの XADC を、AXI インターフェイスを使用して接続できます。XADC はすべての Zynq-7000 AP SoC にエンベデッド ブロックとして用意されています。

LogiCORE™ XADC Wizard IP は、AXI4-Lite 準拠のインターフェイスとオプションの AXI4-Stream インターフェイスを提供します。AXI4-Lite インターフェイスは XADC のコンフィギュレーションに使用でき、AXI4-Stream インターフェイスはデータ通信に使用できます。AXI4-Stream インターフェイスでは、XADC のデータ インターフェイスを別の信号処理 IP と接続することが可能です。このアプリケーション ノートでは、XADC を用いるシステム モニタリング アプリケーションに AXI4-Lite インターフェイスを使用する例を示します。

XADC には、システム モニタリングに利用できる電源と温度のセンサーがあります。これらのセンサーではしきい値の最小値および最大値が設定可能です。測定された物理的なパラメーター値 (電圧または温度) がしきい値を超えるとアラーム信号がアサートされます。オンチップ センサーの詳細は、『7 シリーズ FPGA および Zynq-7000 All Programmable SoC 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) [参照 1] を参照してください。

ザイリンクスは、AXI インターフェイスを使用する XADC アプリケーションでデバイス ドライバーとして利用できる Industrial Input/Output (IIO) フレームワーク ベースの Linux ドライバーを提供しています。このドライバーは XADC をさまざまな動作モード用に設定でき、XADC からデータを収集してユーザー空間レイヤーでデータを利用できるようにします。

このアプリケーション ノートでは、IIO ベースの Linux ドライバーを使用して XADC をコンフィギュレートする方法を示し、AXI GP ポート インターフェイスを用いて XADC と PS 間にデータパスを構

1. ARM® Advanced eXtensible Interface 4 (AXI4)

築する PL ハードウェア デザインを提供します。XADC のユーザー固有の設定パラメーターは、Cortex-A9 プロセッサを使用して設定します。

XADC の設定と取得したサンプル値の表示には、ウェブ サーバー ベースの GUI インターフェイスを使用します。

ハードウェア デザインの概要

ハードウェア デザインは、PL 内の XADC ブロックと Processing System-7 エンベデッド ブロック間の接続を構築することを目的としています。XADC エンベデッド ブロックには、XADC DRP のアドレス指定可能なレジスタ マップに対して読み出し/書き込みを実行するための DRP インターフェイスがあります。XADC Wizard IP が AXI4-Lite トランザクションを DRP アドレス マップに変換します。この IP は、XADC エンベデッド ブロックからのアラーム出力をプロセッサ用の割り込みイベントに変換する AXI 割り込みコントローラーをインスタンシエートします。

図 1 に、ハードウェア デザインのブロック図を示します。

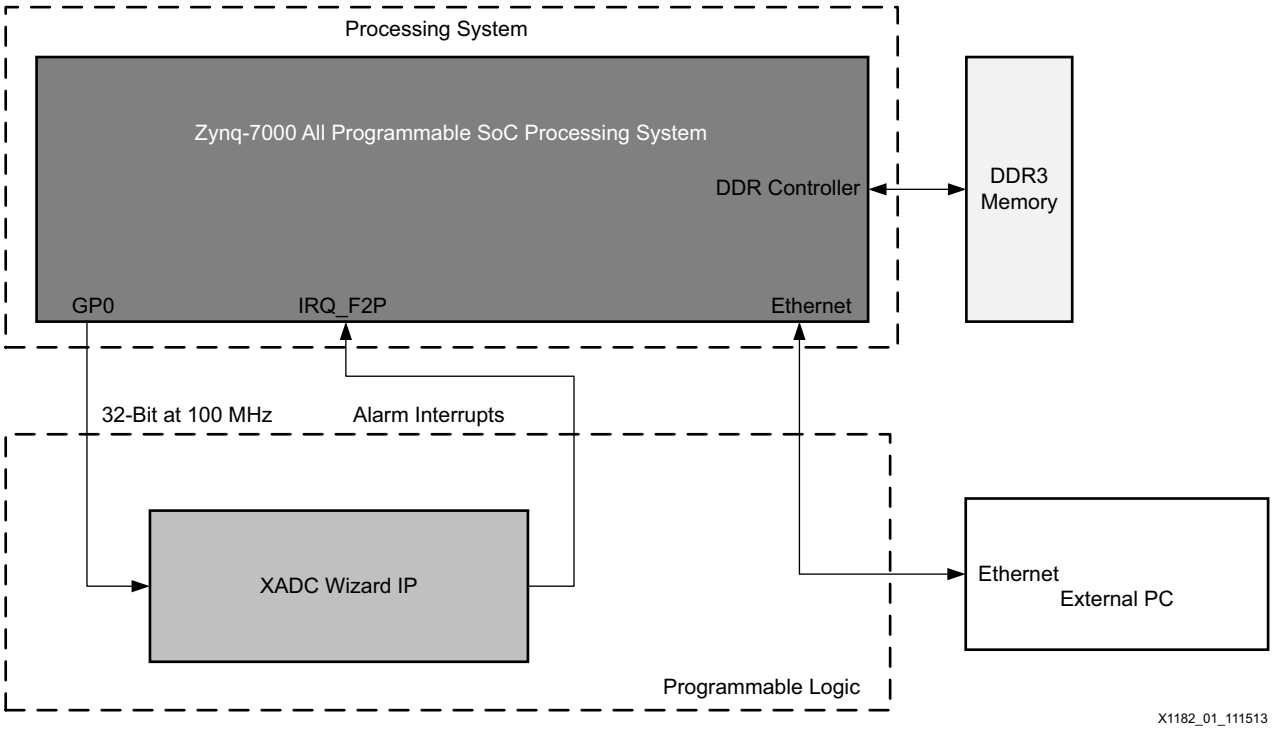

図 1 : ハードウェア ブロック図

データ フローのシーケンスは次のとおりです。

1. XADC の Linux ドライバーが XADC Wizard IP を初期化します。
2. Linux ドライバーがアラーム レジスタにユーザー指定のしきい値を設定します。
3. 電圧/温度の測定値が指定したしきい値を超えた場合、XADC からアラーム出力信号がアサートされます。
4. XADC Wizard IP が、ファブリックと PS 間の割り込みピンを介して割り込みをアサートします。
5. プロセッサが割り込みを処理し、ユーザー インターフェイスへアラーム イベントをレポートします。
6. アラーム条件が有効の間は、アラーム イベントが High を保持します。

XADC Wizard IP の割り込みポートは、IRQ_F2P 割り込みポート 0 (ID 91) へ接続されます。

このデザインは、Vivado® Design Suite IP Integrator 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 デザイン作成後、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 ラッパーを生成します。

表 1 に、ハードウェア デザインで使用したザイリンクス IP を示します。

表 1 : ハードウェア デザインで使用した IP

IP 名	説明	設定
Processing System-7	Processing System-7 エンベデッド ブロックをインスタンシエートするラッパー ファイルを生成する	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
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する	マスター インターフェイスとスレーブ インターフェイスをそれぞれ 1 つに設定
XADC Wizard IP	XADC エンベデッド ブロックをインスタンシエートし、AXI4-Lite トランザクションをエンベデッド ブロックで必要となる DRP トランザクションに変換する	連続サンプリング モードをサポートするように設定
Proc Sys Reset	ペリフェラルおよび AXI Interconnect IP へリセットを適用する	AXI Interconnect および XADC Wizard IP へリセットを適用するように設定

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 : 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

PL のペリフェラル	アドレス マップ (16 進数)
XADC Wizard IP	43C00000-43C0FFFF

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、Linux `git` ツリーに含まれる IIO (Industrial Input/Output) フレームワーク ドライバーをベースにしています。

Linux IIO サブシステムは、ADC/DAC カテゴリに分類されるデバイスをサポートするための標準フレームワークです。

このサブシステムは、ユーザー空間に次の機能を追加します。

- Sysfs インターフェイス : デバイスとの通信に使用
- キャラクター ドライバー インターフェイス : イベント情報の受信に使用

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、次の 4 つのロジック セクションで構成されます。

- XADC コア
  - ハードウェアの設定、センサー値の取得、イベント ハンドリングを目的とした IIO システムとの通信を全面的に担うライブラリです。
- ウェブ サーバー
  - ウェブ サーバーはリモート ウェブ クライアントからの接続要求に対応します。リモート コンピューター システム上で動作するウェブ クライアントは、ウェブ サーバー インターフェイスを介してセンサー データとイベント通知を取得します。サーバーは、XADC コアを介してセンサー データとイベント通知を取得します。

    - また、イベント/アラームを生成するようにハードウェアを設定する際は、ウェブ サーバーがクライアントからしきい値を取得して XADC コアに渡します。
- ウェブ インターフェイス
    - このセクションは、ウェブ サーバーと XADC コアをつなぐ役割を果たします。XADC コアが提供する API を使用して、センサー アラーム値を取得してしきい値を設定します。これらの値はウェブ サーバーを介して受け渡します。
- ウェブ クライアント
    - ウェブ クライアントは、リモート システムのウェブ ブラウザーで動作します。このクライアントは、http ポート 9090 を使用してウェブ サーバーと通信します。
    - フロント エンドとして使用できる GUI アプリケーションを実行します。この GUI には、取得したセンサー データとグラフが表示されます。また、センサー イベントのしきい値もこの GUI で設定します。

通常、このアプリケーションは Zynq-7000 All Programmable SoC デバイスのバックグラウンドでデーモン プロセスとして動作します。図 2 に、このアプリケーションと IIO フレームワークの概略図を示します。

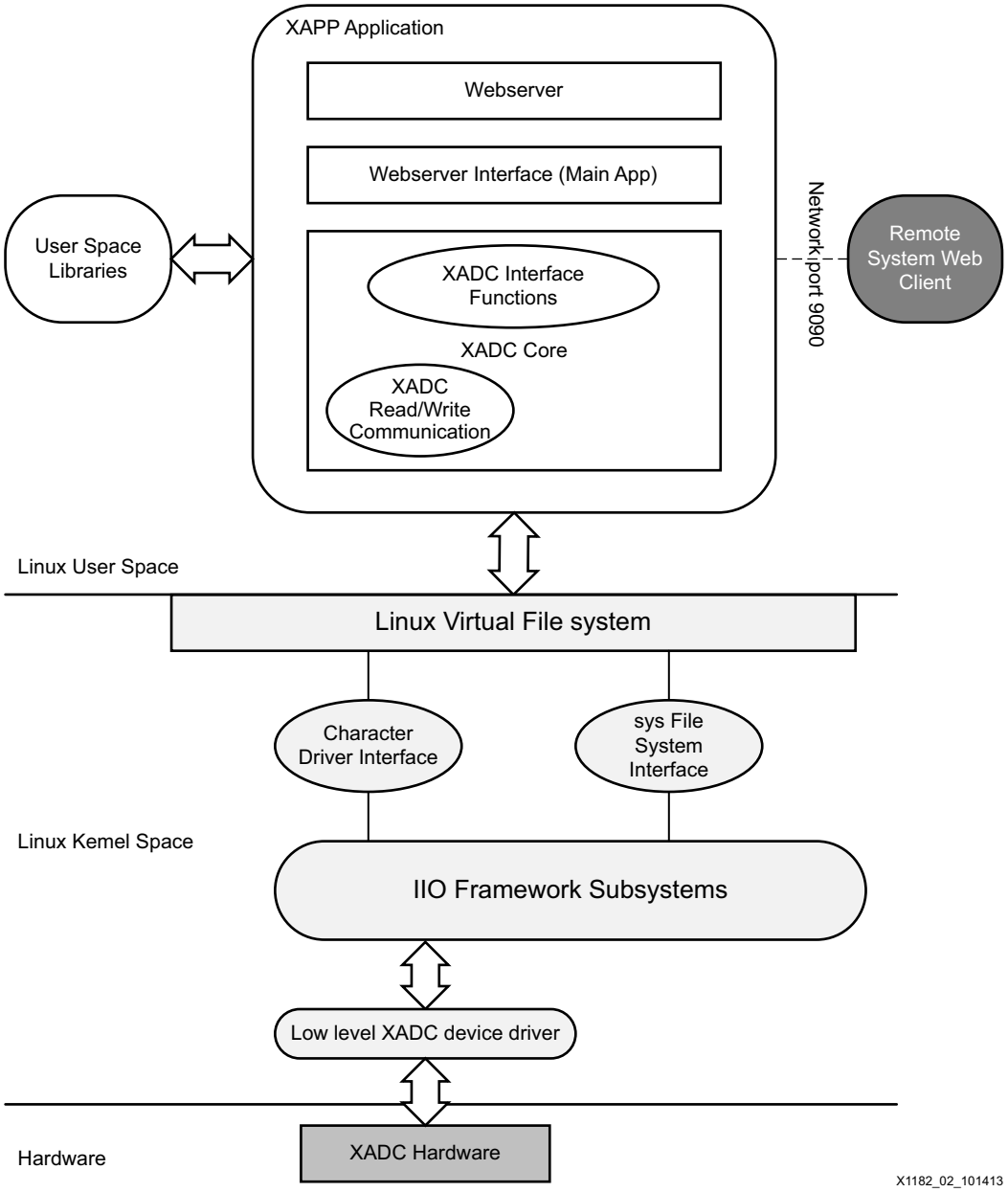

図 2 : ソフトウェアのブロック図

アプリケーションの XADC コアの API

このアプリケーションの XADC コア セクションは、IIO サブシステムを経由して行われる XADC デバイスとのすべての通信を担います。

このセクションのインプリメンテーションは、`xadc_core.h` および `xadc_core.c` の 2 つのファイルで提供されています。

ほかのアプリケーションは、`xadc_core_if.h` ファイルで XADC コアが提供する API を呼び出すことができます。

ここでは、`xadc_core_if.h` ファイルで宣言されている列挙型、構造体、API について説明します。

列挙型

- **`XADC_Parm`** — 統計情報を問い合わせることのできるパラメーターを定義します。

- **`XADC_Alarm`** — システムで利用可能なアラームを定義します。これらのアラームは、ユーザーによるプログラムとステータスの問い合わせが可能です。

構造体

- **`Xadc_callback`** — アラームに対するコールバック情報を含みます。また、関数に渡す関数ポインターと引数ポインターがあります。

関数 API

- **`int xadc_core_init(void);`**

  プログラムの最初に 1 回だけ呼び出します。この関数を実行すると、XADC デバイス ノードを検出してグローバル変数を初期化します。

  引数 : N/A
  戻り値 : [整数型] 0 : 成功、-1 : デバイス ノード検出なし

- **`int xadc_core_deinit(void);`**

  プログラムの最後で呼び出します。この関数を実行すると、リソースが解放されます。

  引数 : N/A
  戻り値 : [整数型] 0 : 成功

- **`void xadc_update_stat(void);`**

  すべての統計情報パラメーターのグローバル キャッシュを更新します。xadc_get_value() 関数を使用してキャッシュから統計情報を読み出す前に、この関数を呼び出すようにします。

  引数 : N/A
  戻り値 : N/A

- **`float xadc_get_value(enum XADC_Param parameter);`**

  指定したパラメーターのキャッシュされた統計情報を返します。戻り値の単位は、電圧の場合は mV、温度の場合は ℃ です。

  引数 : parameter : 値が必要なパラメーターの列挙型の値
  戻り値 : [浮動小数点型] 指定したパラメーターのキャッシュに格納された統計値 [(mV/ ℃)]

- **`float xadc_touch(enum XADC_Param parameter);`**

  指定したパラメーターのグローバル キャッシュ内の統計情報を更新します。xadc_get_value とは異なり、指定したパラメーターのリアルタイム値を返します。

  引数 : parameter : 値が必要なパラメーターの列挙型の値
  戻り値 : [浮動小数点型] 指定したパラメーターのリアルタイム値 [(mV/ ℃)]

- **`int xadc_set_threshold(enum XADC_Alarm alarm,`**

  `float threshold_low,`

  `float threshold_high,`

  `struct Xadc_callback *callback);`

  指定したアラームのしきい値を設定します。

  引数 :

  - alarm : しきい値が設定されている場合のアラームの列挙型の値
  - threshold_low : アラームの最小しきい値 [mV/℃]
  - threshold_high : アラームの最大しきい値 [mV/℃]

- callback : 指定したアラームに対するイベントのコールバック情報。この引数で NULL を渡すと、コールバックは登録されません。それ以外の場合は、イベントが発生すると callback > func(arg) の呼び出しが実行されます。

戻り値 : [整数型] 0 : 成功、0 以外 : エラー

- **`bool xadc_get_alarm_status(enum XADC_Alarm alarm);`**

  指定したアラームに対するイベントの現在のステータスを返します。しきい値を設定した後にのみ有効になります。イベントのステータスを知りたいだけでコールバックが不要な場合に使用します。

  引数 : alarm : イベント ステータスが必要な場合のアラームの列挙型の値
  戻り値 : [ブール型] 1 : イベントがアクティブ、0 : イベントが非アクティブ

Linux ドライバー

IIO フレームワーク (サブシステム) はカーネル空間にインプリメントされます。このフレームワークは、低レベル デバイス ドライバーを IIO デバイス ドライバーとして登録するために使用します。デバイス ドライバーは、IIO フレームワークで指定されている必須の関数をインプリメントし、これらをフレームワークの一部として登録する必要があります。ハードウェア固有の操作では、これらの関数が必ず呼び出されます。

このドライバーは、hwmon ベースの ADC ドライバーの機能を置き換えます。したがって、XADC にこのサブシステムを使用する場合は、カーネルのビルドで hwmon xadc を無効にする必要があります。

Sysfs インターフェイス

`dts` ファイルに XADC エントリがある場合、IIO サブシステムは XADC 用の sysfs インターフェイスを登録します。sysfs エントリは、`/sys/bus/iio/devices/<populated-device>` にあります。

このパス (`/sys/bus/iio/devices/<populated-device>` には、各種パラメーターの値を格納したファイル ノードが存在します。

正しいドライブを確認するには、`/sys/bus/iio/devices/<populated-device>/name` ファイルを読み出します (ここでは xadc を読み出す)。

たとえば、センサーから V_{CCINT} のロー コード値を取得するには、次のファイルを読み出します。

```
/sys/bus/iio/devices/<populated-device>/in_voltage0_vccint_raw
```

図 3 に、コマンド ラインの例を示します。

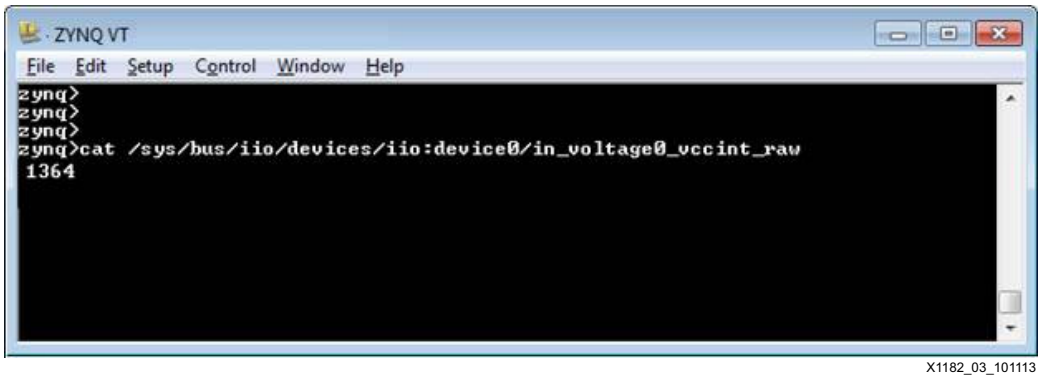

図 3 : コマンド ラインの例

コマンド ラインの実行

このアプリケーション ノートのリファレンス デザインには、内部の V_{CCINT}、V_{CCAUX}、V_{CCBRAM} チャネルを監視するためのコマンド ライン インターフェイスがあります。コマンドは、Linux OS のコマンド ライン シェルから実行できます。また、これらのコマンドは、Linux OS のブート時にシステムにインストールされます。

表 3 に、利用できるコマンドを示します。

表 3 : システム モニタリング コマンド

コマンド	説明	例
xadc_get_value_vccint	観測した V_{CCINT} の値を返します。	コンソールにコマンド **$ xadc_get_value_vccint** を入力すると、mV 単位で V_{CCINT} 値が返ります。
xadc_get_value_vccaux	観測した V_{CCAUX} の値を返します。	コンソールにコマンド **$ xadc_get_value_vccaux** を入力すると、mV 単位で V_{CCAUX} 値が返ります。
xadc_get_value_vccbram	観測した V_{CCBRAM} の値を返します。	コンソールにコマンド **$ xadc_get_value_vccbram** を入力すると、mV 単位で V_{CCBRAM} 値が返ります。
xadc_get_value_temp	観測した温度の値を返します。	コンソールにコマンド **$ xadc_get_value_temp** を入力すると、℃ 単位で温度値を返します。

イベント通知

イベント通知は、/sys/bus/iio/devices/<populated-device>/events/ ディレクトリのファイルに書き込むことによって有効にできます。

たとえば V_{CCINT} の最大および最小しきい値を設定するには、それぞれの値のロー コードを次のファイルに書き込みます。

/sys/bus/iio/devices/<populated-device>/events/in_voltage0_vccint_thresh_rising_value

/sys/bus/iio/devices/<populated-device>/events/in_voltage0_vccint_thresh_falling_value

最大しきい値と最小しきい値の両方で V_{CCINT} イベントを有効にするには、次のファイルに 1 を書き込みます。

/sys/bus/iio/devices/<populated-device>/events/in_voltage0_vccint_thresh_rising_en

/sys/bus/iio/devices/<populated-device>/events/in_voltage0_vccint_thresh_falling_en

しきい値を設定したら、キャラクター ドライバー インターフェイスを利用してイベントを取得します。

しきい値の設定後、イベントを受け取るには次の手順を実行します。

1. イベントおよび ioctl を定義した <linux/iio/events.h> をインクルードする。
   #include <linux/iio/events.h>
2. <populate-device> ディレクトリ名を使用して、デバイス ファイルを開く。
   /dev/<populate-device>
3. 手順 2 の **fd** を使用して、イベント ファイル ディスクリプターを取得する。
   ioctl(fd, IIO_GET_EVENT_FD_IOCTL, &event_fd)
4. event_fd に対する読み出しを実行する。
   read(event_fd, &event, sizeof(event));

   ここで、**event** のデータ型は struct iio_event_data。

   このイベントを events.h ファイルに記述された方法で解読し、正確なイベントを調べる。
5. この読み出しはブロッキング コールで、任意のイベントが発生すると解放されます。

GUI インターフェイス

AXI-XADC インターフェイスの GUI は、ウェブ サーバーをベースにしています。この GUI には、HTML、Javascript、TCP-HTTP プロトコル ハンドラーが使用されています。図 4 に、GUI 画面を示します。この GUI 画面は、次の 2 つのパネルで構成されます。

- XADC コントロール パネル

- XADC モニター パネル

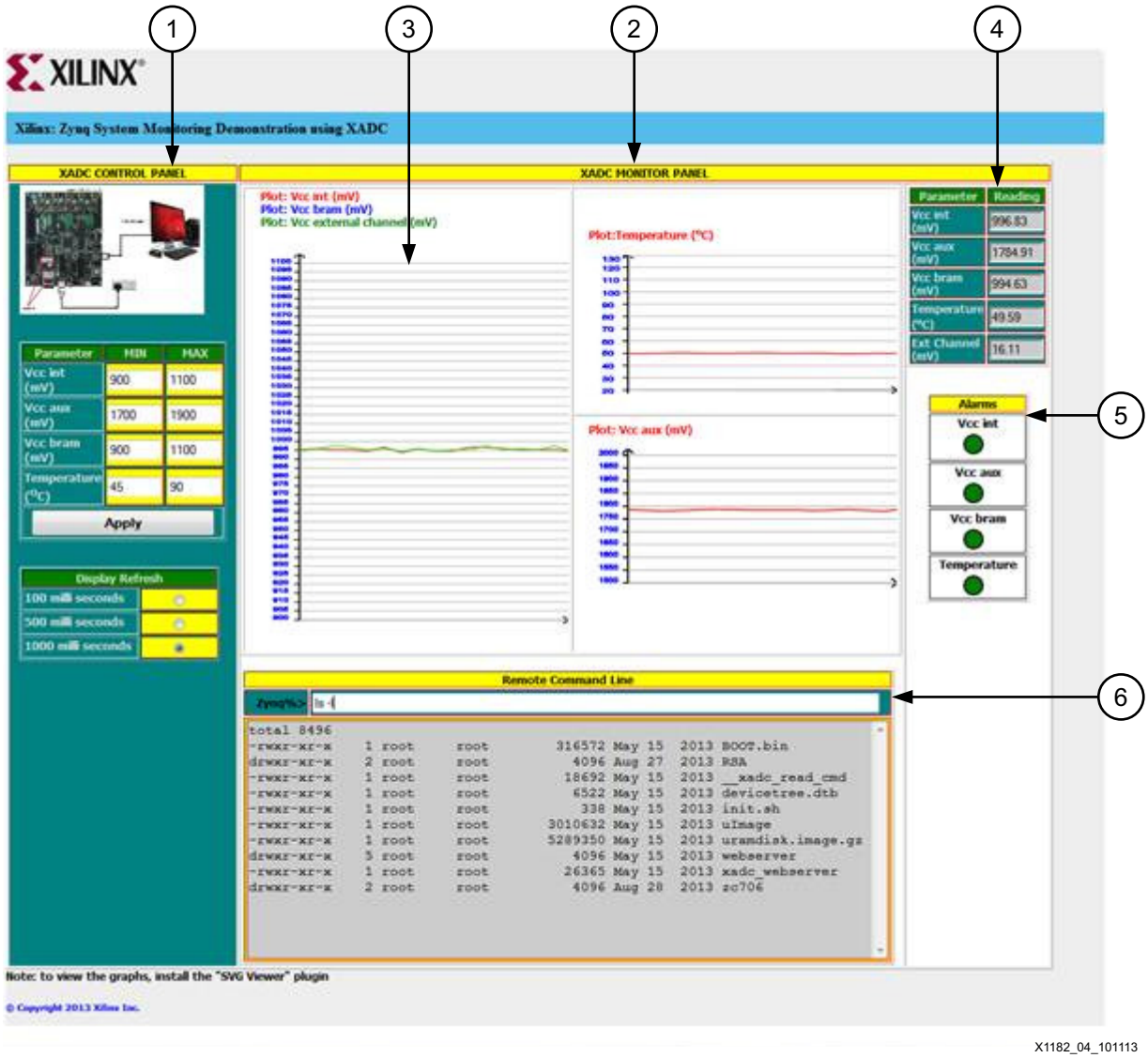

図 4 : ウェブサーバー ベースの GUI アプリケーション

図 4 について説明します。

1. XADC コントロール パネル ([XADC Control Panel])
2. XADC モニター パネル ([XADC Monitor Panel])
3. グラフ
4. 計測値
5. アラーム ([Alarms])
6. リモート コマンド ライン ([Remote Command Line])

XADC コントロール パネル

画面左側にあるパネルです (図 4 の①)。このパネルでは、V_{CCINT}、V_{CCBRAM}、V_{CCAUX}、および温度の最大および最小しきい値を設定できます。[Apply] をクリックすると、画面に表示された値が

Zynq-7000 All Programmable SoC の各レジスタに書き込まれます。これで、AXI-XADC のアラーム システムはユーザーが指定した新しいしきい値に基づいて動作します。

XADC モニター パネル

このパネルは図 4 の ② です。ウェブ クライアントが AXI XADC インターフェイスから計測値を周期的に (毎秒 1 回) 取得し、XADC モニター パネルに表示します。

XADC モニター パネルは 4 つの部分で構成されており、さまざまな計測値やグラフを表示します。

グラフ

グラフは図 4 の ③ に示されています。このウェブ ページには、4 つの折れ線グラフが表示されます。

- V_{CCINT} グラフの目盛りの範囲は 0 ～ 1,200mV です。
- V_{CCBRAM} グラフの目盛りの範囲は 0 ～ 1,200mV です。
- V_{CCAUX} グラフの目盛りの範囲は 0 ～ 1,000mV です。
- 外部電圧グラフの目盛りの範囲は 0 ～ 1,000mV です。

いずれも毎秒 1 回、最新の計測値でグラフが描かれます。これらのグラフには、直近 10 回の計測値が表示されます。また、時間の経過と共にグラフは右から左へスクロールし、右端に最新の計測値が表示されます。

計測値

測定値は図 4 の ④ に示されています。ウェブ クライアントが XADC を毎秒 1 回プローブし、最新の計測値を取得します。これらの値はグラフにも描画されます。

アラーム ([Alarms])

アラームは図 4 の ⑤ に示されています。Zynq-7000 デバイスの AXI XADC インターフェイスには、パラメーターの現在の値が最小または最大しきい値を超えているかどうかを監視する機能があります。指定した値を超えた場合には、対応するアラームが応答します。

このウェブ ページは 4 つのパラメーター (V_{CCINT}、V_{CCAUX}、V_{CCBRAM}、温度) を追跡し、アラームの状態を更新します。各アラームは、しきい値の範囲内なら緑、しきい値を超えると赤でリアルタイムに表示されます。

ユーザーは、パラメーターの最小および最大しきい値を適宜変更して、アラームの動作を確認できます。計測値が設定したしきい値を超えると、すぐにアラームが赤に変わります。

リモート コマンド ライン ([Remote Command Line])

リモート コマンド ライン画面は図 4 の ⑥ です。これは、GUI 機能に追加された Linux プロンプトです。Zynq-7000 デバイスのシェル プロンプトとして機能します。このリモート コマンド ラインは Zynq-7000 AP SoC シェルとして利用可能で、有効な Linux コマンドを任意に実行できます。入力した文字列は Zynq-7000 デバイスに伝達され、Linux コマンドとして実行されます。コマンドを実行すると、Zynq-7000 デバイスからコマンドの実行結果が GUI に送られて、リモート コマンド ライン コンソールに表示されます。

ここで入力したコマンドは一時プロセスで実行されるため、その実行が完了すると強制終了されます。つまり、コマンドの実行結果は維持されません。

たとえば **`cd /usr`** コマンドを実行した場合、その移動した新しいフォルダーには留まらず、実行結果を返した後にホーム フォルダーへ戻ります。

外部チャネルの計測

外部チャネルの電圧レベルを計測できるように、IIO フレームワークには外部チャネル計測オプションが用意されています。計測する外部チャネルは、デバイス ツリー ソース (`.dts`) ファイルで有効にして

おく必要があります。Zynq-7000 All Programmable SoC ZC702 評価キットには、XADC ヘッダー ピンに挿入する AMS101 ドーター カードが付属しています。外部信号ソースを AMS101 カードに接続する方法については、『AMS101 評価カード ユーザー ガイド』(UG886) [参照 2] 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でテストできます。AMS101 評価カードを用いて外部データを取得するには、『AMS101 評価カード ユーザー ガイド』(UG886) [参照 2] の説明に従って AMS101 カードを ZC702 評価ボードに接続する必要があります。外部信号は AMS101 カードの補助チャネルに接続でき、取得したサンプルは AXI XADC インターフェイス経由で読み出すことができます。

実験結果

計測の結果、XADC Wizard IP の AXI4-Lite インターフェイスから Zynq-7000 ファミリの AXI GP インターフェイス間でサポートされる外部信号の最大帯域幅は 2.12MHz であることが確認されています。これは CPU で XADC アプリケーションのみを実行する理想条件で計測されており、それ以外のアプリケーションは実行していません。また、アラーム イベントをドライバーで有効にして計測しています。

読み出しリクエストを発行してから完了データが受信されるまでの時間を実験的に計測しました。完了データは、EOC (End Of Conversion) 割り込み信号がアサートされると読み出されます。表 4 に複数の動作条件でのレイテンシの値を示します。

表 4 : レイテンシの値

コマンドのタイプ	ワースト ケースのレイテンシ (1)	平均レイテンシ(1)
メモリ読み出し	0,72µs	0.4771µs

1. これらの値は、AXI4-Lite クロック周波数を 100MHz に設定して計測しました。

まとめ

このデザインは、システム モニタリング用に XADC の AXI4-Lite インターフェイスを使用するためのプラットフォームを提供しています。また、XADC の AXI4-Lite インターフェイス経由で外部補助チャネルを使用して計測を実行する可能性も検討し、このインターフェイスを用いて監視できる最大信号周波数も評価しています。

このアプリケーション ノートに記載したレイテンシの値は、このインターフェイスを利用した場合の理想値です。値は、実際の CPU の負荷によって変化することがあります。

リファレンス デザイン

リファレンス デザインの ZIP ファイルは、次の URL からダウンロードできます。

https://secure.xilinx.com/webreg/clickthrough.do?cid=346167

`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。表 5 に、リファレンス デザインの詳細を示します。

デザインを再構築する場合は、Wiki ページ「Zynq AXI XADC App Note」を参照してください。

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	Zynq-7000 All Programmable SoC
ソース コードの提供	あり
ソース コードの形式	C

表 5 : リファレンス デザインの詳細 (続き)

パラメーター	説明
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	N/A
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	なし
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 2013.3
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 2013.3
スタティック タイミング解析の実施	Vivado Design Suite 2013.3
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

デバイス ツリー

ここでは、ザイリンクス Linux デバイス ツリーの XADC エントリの一部を示します。

```
xadc@43c00000 {
                compatible = "xlnx,axi-xadc-1.00.a";
                reg = <0x43c00000 0x10000>;
                interrupts = <0 59 4>;
                interrupt-parent = <&gic>;
               clocks = <&ps_clk>;
                xlnx,channels {
                        #address-cells = <1>;
                        #size-cells = <0>;
                       channel@0 {
                                reg = <0>;
                        };
                };
        };
```

可能な XADC エントリの詳細は、次の Linux カーネル ディレクトリにある資料を参照してください。

```
Documentation/devicetree/bindings/iio/xilinx-xadc.txt
```

参考資料

次の文書は、このアプリケーション ノートの補足資料です。

1. 『7 シリーズ FPGA および　Zynq-7000 All Programmable SoC 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)
1. 『AMS101 評価カード ユーザー ガイド』(UG886)
2. 『Zynq-7000 All Programmable SoC テクニカル リファレンス マニュアル』(UG585)
3. Analog Devices 社 Wiki ページ - 「Linux Industrial I/O Subsystem」の「IIO Overview」: https://wiki.analog.com/software/linux/docs/iio/iio
4. ザイリンクス Wiki ページ「Zynq AXI XADC App Note」

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
2013 年 11 月 18 日	1.0	初版

Disclaimer