Nios II - 割り込みの実現

ver.14

2020 年 6 月 Rev.3

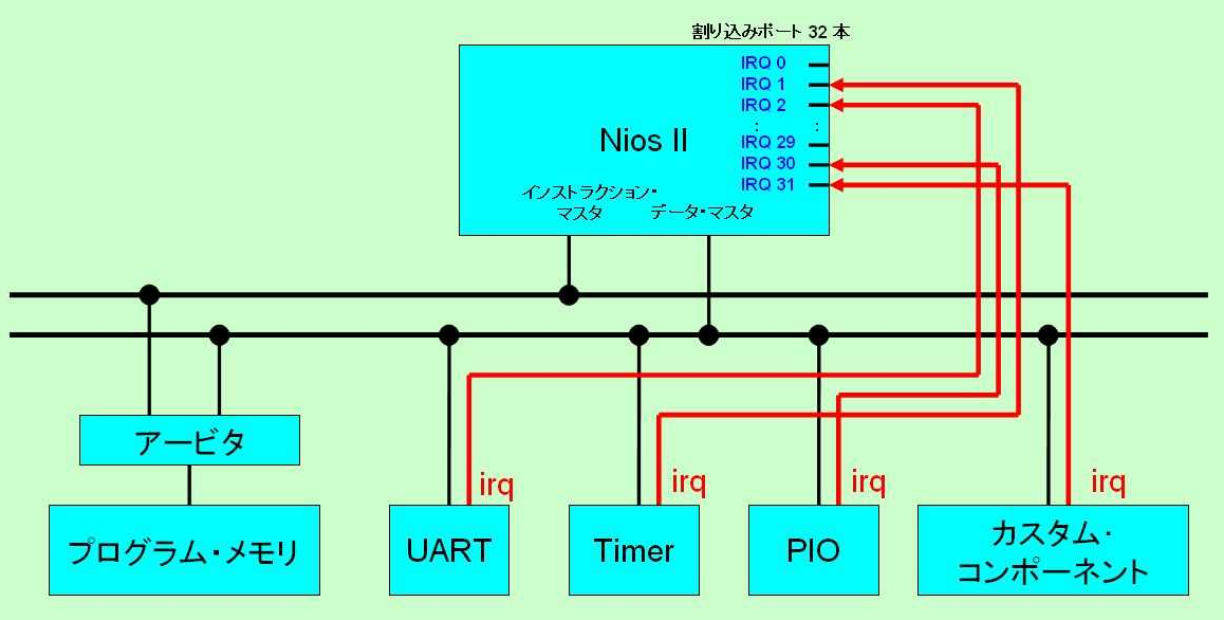
Nios II - 割り込みの実現

目次

1. はじめに

この資料では、Nios® II の割り込みの実現フローを紹介します。Nios II のシステムでは、スレーブより Nios II へ割り込みのトリガとして、リクエスト(irq)がアサートされます。そして、Nios II は各スレーブの割り込みに対応したサービス・ルーチンを実施します。下図では、UART、Timer、PIO、カスタム・コンポーネントが割り込みのリクエストを発行できるペリフェラルとして登録されています。各割り込みライン(irq)は、Nios II の割り込みポートに接続されており、Nios II は各ペリフェラルからの割り込み要求に応じて、必要な処理を実行します。各ペリフェラルは、割り込み番号として 0 から 31 までを登録でき、0 の優先順位が最も高く、31 が最も低い仕様となっています。

この資料の内容は、Quartus® II 開発ソフトウェア v14.0 と Nios II Software Build Tools(以降、Nios II SBT) v14.0 で動作確認を行っています。



2. Nios II ハードウェア割り込みの処理フロー

ハードウェア割り込みも含めて、「例外」が発生したときの処理フローについて説明します。

2-1. 「例外」の検出

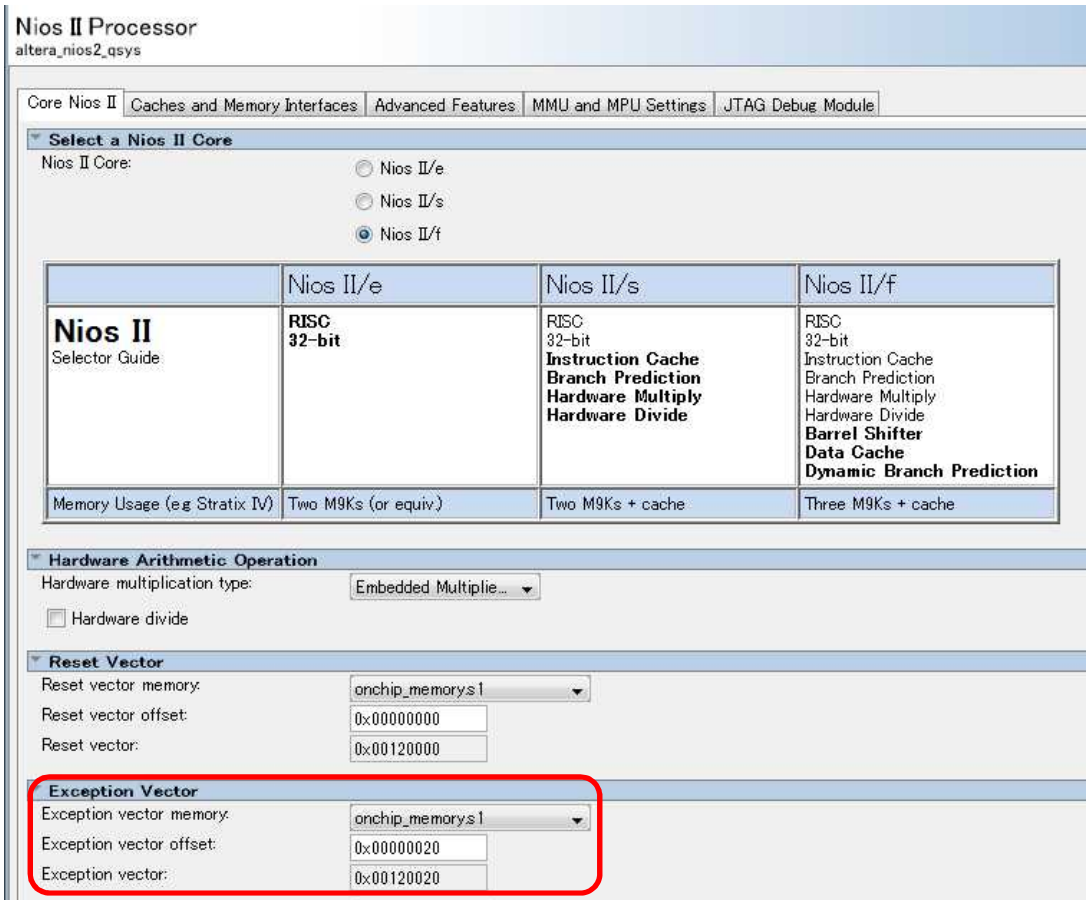
プッシュボタン等の外部割り込みに代表される外部イベントや、未実装命令検出等の内部イベントが発生した場合、Nios II プロセッサは「例外」を検出して、制御が通常の実行フローから例外処理に移行します。「例外」検出後、最初に例外発生前のプロセッサの状態を保存します。

2-2. 例外ベクタに処理を移行

例外ベクタ（例外アドレス）に実行を移します。Nios II プロセッサのアーキテクチャーでは、いかなる例外が発生しても、1 つの例外ベクタに実行を移します。ソフト・コア・プロセッサの Nios II の場合、例外ベクタはシステム生成時にユーザ側で事前に Qsys システム統合ツール（以降、Qsys）より指定する必要があります。

下図の場合、例外ベクタはアドレス・マップ上の 0x00120020 になります。このアドレスは、ベース・アドレス 0x00120000 の “onchip_memory”（FPGA 内蔵メモリ）領域のオフセット 0x20 となります。

※ オフセットは必ず 0x20 以上にしてください。



2-3. 例外ハンドラの実行

例外ベクタには例外ハンドラが配置されており、「例外」の要因を特定して適切な例外処理を実行するプログラム（例外処理ルーチン）に実行を移します。Nios II プロセッサのアーキテクチャーでは、3 種類の例外ハンドラが定義されています。

1) _irq_entry()

例外タイプを判別します。例外タイプを「ハードウェア割り込み」と判定した場合、alt_irq_handler() を呼び出します。例外タイプを「ソフトウェア割り込み」と判定した場合、software_exception() を呼び出します。

2) alt_irq_handler()

割り込み要因（割り込み要求番号）を特定して、その割り込みに対する登録済の割り込み処理ルーチンを呼び出します。割り込み処理ルーチンの登録は、alt_irq_register() という登録専用の HAL API を使用します。

3) software_exception()

ソフトウェア例外の原因を特定します。例えば、未実装例外が発生した場合、適切な命令エミュレーション・ルーチンを呼び出します。

注： 未実装例外

Nios II では、fast（速度重視）/ standard（速度・回路規模とのバランスを考慮）/ economy（回路規模重視）の 3 種類のコアを選択できますが、互いに同じソフトウェア・コードを利用できるような工夫が行われております。例えば、ハードウェア乗算器を実装している Nios II fast コアで実行される乗算命令を、ハードウェア乗算器を実装していない Nios II economy コアで実行した場合、「未実装例外」が発生して、加算命令でエミュレーションすることで乗算命令を処理します。（Nios II SBT を常時使用していれば、ユーザがこの部分を意識する必要はありません。）

2-4. 割り込み処理ルーチン（ISR）の実行

ユーザが作成した割り込み処理専用のソフトウェア・プログラムに実行を移します。

2-5. 復帰

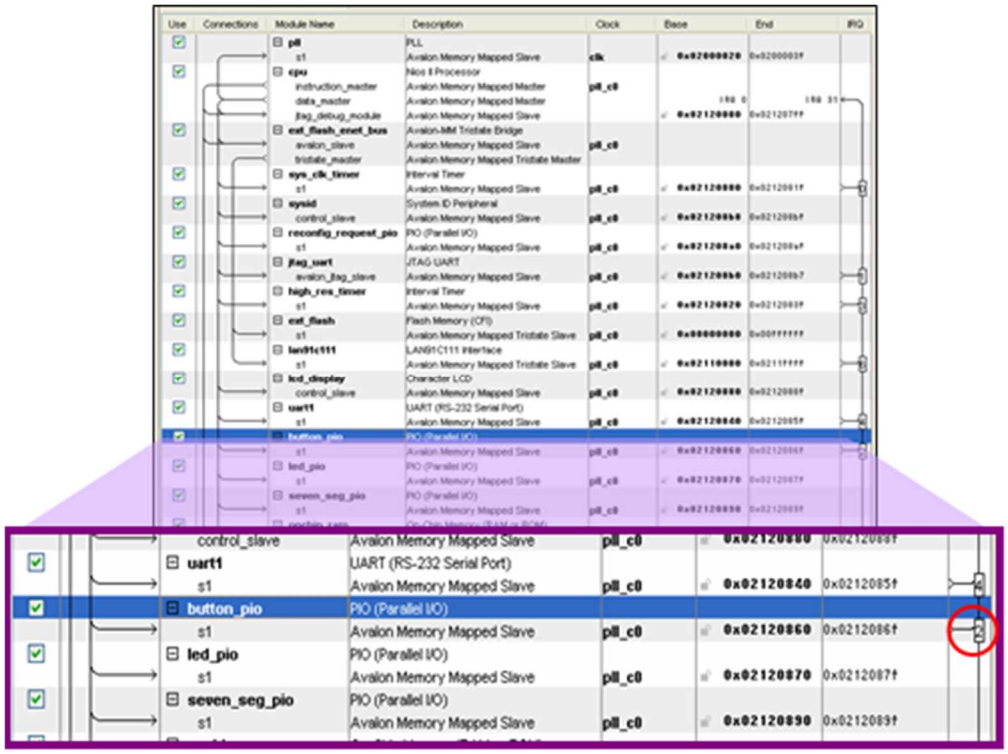
割り込み処理ルーチン（ISR）が終了すると、Nios II プロセッサは例外が発生する前の状態に復帰します。

3. 実装方法

3-1. 割り込み要求(IRQ)の設定

ペリフェラルを Qsys に追加した場合、ユーザはペリフェラルの割り込み要求(IRQ)に対してハードウェア割り込みの優先順位(IRQ 番号)を任意に設定することができます。アルテラ社の Qsys では、右側に表示される番号が IRQ 番号に該当します。この番号が若い IRQ(つまり、IRQ0)で設定したペリフェラルが優先順位の最も高い割り込みになります。

下図では、PIO ペリフェラルを "button_pio" という名称で Qsys に組み込んでいる例を示しています。ここでは、"IRQ 2" として Qsys に登録しています。



3-2. 割り込み処理ルーチンの登録方法

alt_irq_register() という登録専用の HAL API を使用します。この HAL API は、通常プログラムの main 関数に記述します。この HAL API のフォーマットは次の通りです。

```
alt_irq_register(  IRQ 番号、 ISR へのコンテキスト、 割り込み処理ルーチン名)
```

例 : alt_irq_register (BUTTON_PIO_IRQ , edge_capture_ptr , button_interrupts);

割り込み処理ルーチンは、main() の外で用意します。IRQ 番号は、Qsys で登録した割り込みをかけるコンポーネントの番号です。system.h ファイルに「ペリフェラル名_IRQ」で定義されます。

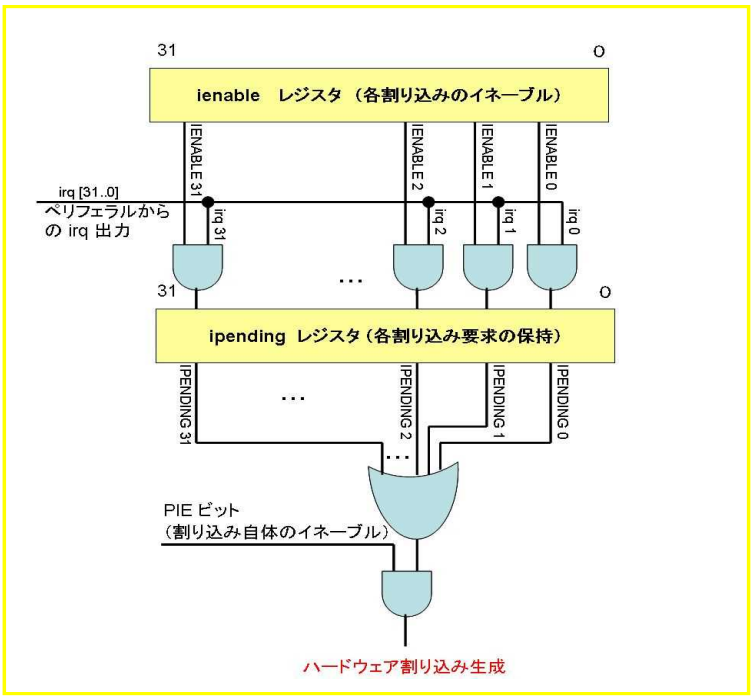
詳細は下記のドキュメントの「 9. Exception Handling 」および「 15. HAL API Reference 」を参照ください。

Nios® II Software Developer Handbook

3-3. 割り込みの有効化/無効化と多重割り込み

デフォルトでは、割り込みがかかると、alt_irq_disable_all() が実行され、今行われている割り込み以外の要求を無効にしますので多重割り込みはできません。多重割り込みを可能にする場合は、サービス・ルーチン（ISR）内で優先度の高い割り込みを許可するために、alt_irq_interruptible() の API を使用します。

下図のように、割り込み毎に割り込みのイネーブル・ビットがあり、有効になっているポートのハードウェア割り込みを認識します。



4. 参考

以下のサンプル・ソースは、ボタン割り込みを使用した場合のソフトウェア・サンプルです。このサンプルでは、ボタン 4 つを接続した PIO (BUTTON_PIO) の Edge capture register を使用して、ボタンの押された位置を検出します。同時にボタンからの割り込みを登録しているので、ボタンが押されると Nios II に対して割り込みが発生し、登録した割り込みサービス・ルーチンが実行されます。割り込みサービス・ルーチンでは、Edge capture register の値を読み込んでボタンの位置に対応した LED (LED_PIO) を点灯します。詳細は、別資料の「Nios II - ペリフェラル PIO (Parallel I/O) 簡易ユーザ・ガイド」にありますので、併せて参照ください。

```
#include <stdio.h>
#include "system.h"
#include "altera_avalon_pio_regs.h"
#include "sys/alt_irq.h"

/* ボタン割り込みのサービス・ルーチン */
static void button_interrupts(void* context, alt_u32 id)
{
    int button_position;      // ボタンが押された場所の保存に使用
    button_position = IORD_ALTERA_AVALON_PIO_EDGE_CAP(BUTTON_PIO_BASE);
                              // ボタンPIOのエッジ・キャプチャ・レジスタを読み込み
                              // どのポジションのボタンが押されたかを検出

    IOWR_ALTERA_AVALON_PIO_DATA(LED_PIO_BASE, button_position);
                              // 押されたボタンに対応したLEDのビットを点灯

    /* エッジ・キャプチャ・レジスタのリセット */
    IOWR_ALTERA_AVALON_PIO_EDGE_CAP(BUTTON_PIO_BASE, 0x0);
    button_position = 0x0;
}

/* メインルーチン */
int main(void)
{
    printf("Interrupt test¥n");

    /* PIOの初期化 */
    // 特定のボタンのみマスクをかけて有効にする（ボタンが４つの場合）
    IOWR_ALTERA_AVALON_PIO_IRQ_MASK(BUTTON_PIO_BASE, 0xF);

    // エッジ・キャプチャ・レジスタのリセット
    IOWR_ALTERA_AVALON_PIO_EDGE_CAP(BUTTON_PIO_BASE, 0x0);

    // 割り込みハンドラの登録
    alt_irq_register(BUTTON_PIO_IRQ, 0, button_interrupts);

    while(1)
    {
        // ボタンの割り込みを待つ
    }

    return 0;
}
```

改版履歴

Revision	年月	概要
1	2014 年 11 月	初版
2	2015 年 4 月	アルテラ社の Web サイトのリニューアルに伴う URL 変更 誤記訂正： P.5 誤）first ⇒ 正）fast
3	2020 年 6 月	・"3-2. 割り込み処理ルーチンの登録方法" - ドキュメントへのリンク(URL)を修正。 ・"3-3. 割り込みの有効化/無効化と多重割り込み": - alt_irq_enbale() を alt_irq_interruptible() に修正 ・"最終ページ":「免責およびご利用上の注意」内のリンクを修正