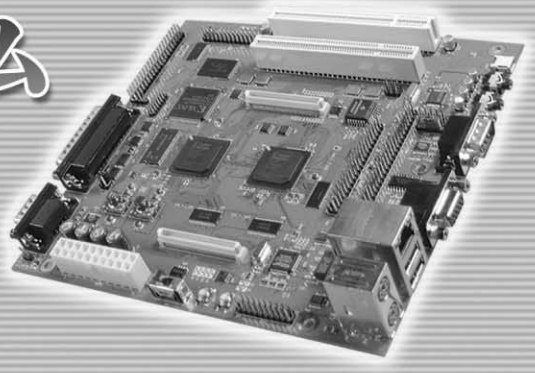
コンピュータ・システム技術学習キット活用通信



大牧 正知

第5回 MicroBlazeで割り込みを使う方法

組み込み制御では，割り込みは重要です．ここではMicroBlazeで割り込みを使う方法について解説します．

MicroBlaze自身には割り込み入力は一つしかなく，複数の割り込みを扱いたい場合は，外部に割り込みコントローラを用意します．割り込み要因が一つだけの場合は，そのままMicroBlazeに割り込み信号を接続できます．

● ハードウェアの設定

まず，前回作成したEDKデザインの，Portsの設定(**図1**)を変更します．MicroBlazeのところに，INTERRUPTという信号があるので，これをMake Externelの設定にすると，**図2**のようになります．Sensitivityの項目で，エッジ/レベル割り込み，極性などを選択できます．今回接続する割り込みは正論理で，レベル割り込みで処理することにします(**図3**)．この設定はあくまで信号に対する設定で，MicroBlazeの割り込み極性などは，MicroBlazeのパラメータで設定します．デフォルトは正論理のレベルなので，今回はデフォルトのまま使用します．

EDKの設定の変更はこれで完了です．メニューの[Hardware]→[Generate Netlist]でネットリストを再生成しますが，このままでは前回編集した`system_stub.vhd`が再生成され，上書きされてしまうので，`system_stub.vhd`はあらかじめ別の場所に保存しておくなどして残しておいてください．

Generate Netlistが完了したら，ISEで割り込みを接続します．`system_stub.vhd`が再生成されているので，前回編集したファイルに差し替えます．また，今回は，割り込み機能のある`DIPLED_SYS.VHD`を使うので，このファイルも差し替えます．

あとは，割り込み信号を追加していきます(**リスト1**)．

● ソフトウェアの設定

これでハードウェアの設定は完了です．次にソフトウェアの

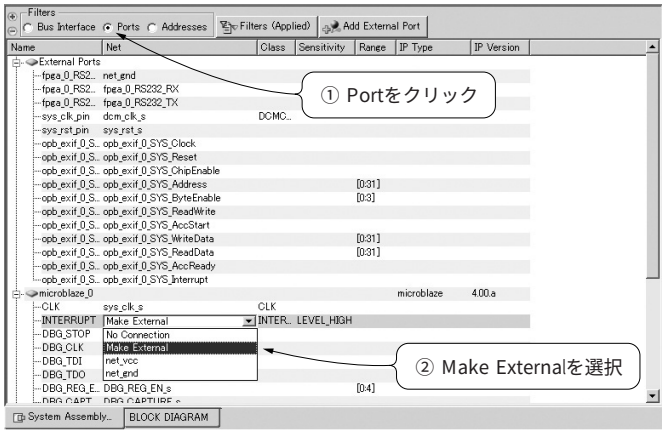


図1 前回設計したデザインの信号

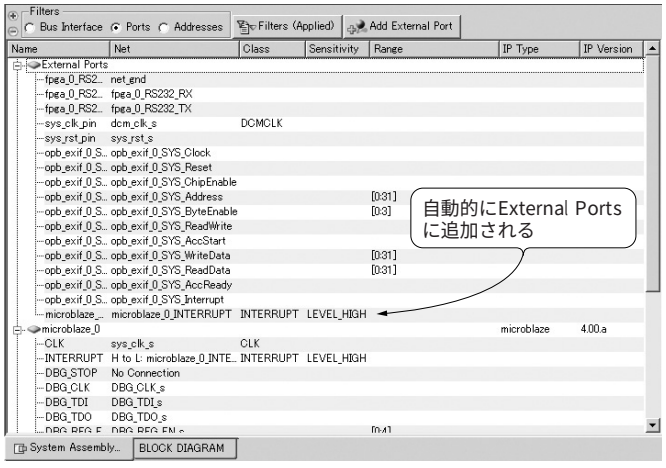


図2 割り込み信号を追加する

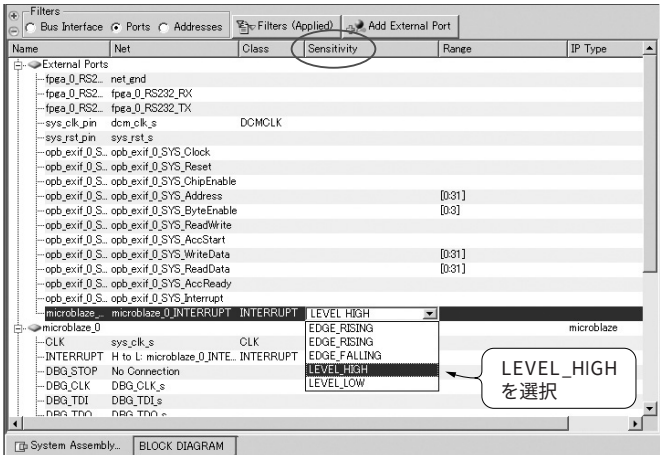


図3 割り込み信号の極性

KEYWORD——割り込み，エッジ割り込み，レベル割り込み，割り込み許可，割り込み禁止，優先順位つき割り込み，NMI，Non Maskable Interrupt

リスト1　割り込み信号を追加したトップ・モジュール（system_stub.vhd）

```
～省略～
 architecture STRUCTURE of system_stub is

  component system is
    port (
      fpga_0_RS232_req_to_send_pin : out std_logic;
      fpga_0_RS232_RX_pin : in std_logic;
      fpga_0_RS232_TX_pin : out std_logic;
      sys_clk_pin : in std_logic;
      sys_rst_pin : in std_logic;
      opb_exif_0_SYS_Reset_pin : out std_logic;
      opb_exif_0_SYS_ChipEnable_pin : out std_logic;
      opb_exif_0_SYS_Address_pin : out std_logic_vector(0 to 31);
      opb_exif_0_SYS_ByteEnable_pin : out std_logic_vector(0 to 3);
      opb_exif_0_SYS_ReadWrite_pin : out std_logic;
      opb_exif_0_SYS_AccStart_pin : out std_logic;
      opb_exif_0_SYS_WriteData_pin : out std_logic_vector(0 to 31);
      opb_exif_0_SYS_ReadData_pin : in std_logic_vector(0 to 31);
      opb_exif_0_SYS_AccReady_pin : in std_logic;
      opb_exif_0_SYS_Interrupt_pin : in std_logic;
      opb_exif_0_SYS_Clock_pin : out std_logic;
      microblaze_0_INTERRUPT_pin : in std_logic ←（割り込み入力）
    );
  end component;

 component DIPLED_SYS is
    generic (
         DEVICE_ID      : std_logic_vector(31 downto 0) := X"12345678"
                                                 -- デバイスID
    );
    port (

～省略～
    );
  end component;

～省略～
  signal fpga_0_RS232_RX_pin_IBUF : std_logic;
  signal fpga_0_RS232_TX_pin_OBUF : std_logic;
  signal fpga_0_RS232_req_to_send_pin_OBUF : std_logic;
  signal opb_exif_0_SYS_AccReady_pin_IBUF : std_logic;
  signal opb_exif_0_SYS_AccStart_pin_OBUF : std_logic;
  signal opb_exif_0_SYS_Address_pin_OBUF : std_logic_vector(0 to 31);
  signal opb_exif_0_SYS_ByteEnable_pin_OBUF : std_logic_vector(0 to 3);
  signal opb_exif_0_SYS_ChipEnable_pin_OBUF : std_logic;
  signal opb_exif_0_SYS_Interrupt_pin_IBUF : std_logic;
  signal opb_exif_0_SYS_ReadData_pin_IBUF : std_logic_vector(0 to 31);
  signal opb_exif_0_SYS_ReadWrite_pin_OBUF : std_logic;
  signal opb_exif_0_SYS_Reset_pin_OBUF : std_logic;
  signal opb_exif_0_SYS_WriteData_pin_OBUF : std_logic_vector(0 to 31);
  signal opb_exif_0_SYS_Clock_pin_OBUF : std_logic;
  signal sys_clk_pin_IBUFG : std_logic;
  signal sys_rst_pin_IBUF : std_logic;
  signal irq : std_logic; ←（追加）

begin

  system_i : system
    port map (
      fpga_0_RS232_req_to_send_pin => fpga_0_RS232_req_to_send_pin_OBUF,

～省略～
      opb_exif_0_SYS_Clock_pin => opb_exif_0_SYS_Clock_pin_OBUF,
      microblaze_0_INTERRUPT_pin => irq ←（追加）
    );

 usercirc : DIPLED_SYS
    port map(
         AVP_LED => AVP_LED,
         AVP_DIPSW => AVP_DIPSW,
         SYS_Clock => opb_exif_0_SYS_Clock_pin_OBUF,
      SYS_Reset => opb_exif_0_SYS_Reset_pin_OBUF,
      SYS_ChipEnable => opb_exif_0_SYS_ChipEnable_pin_OBUF,
      SYS_Address => opb_exif_0_SYS_Address_pin_OBUF,
      SYS_ByteEnable => opb_exif_0_SYS_ByteEnable_pin_OBUF,
      SYS_ReadWrite => opb_exif_0_SYS_ReadWrite_pin_OBUF,
      SYS_AccStart => opb_exif_0_SYS_AccStart_pin_OBUF,
      SYS_WriteData => opb_exif_0_SYS_WriteData_pin_OBUF,
      SYS_ReadData => opb_exif_0_SYS_ReadData_pin_IBUF,
      SYS_AccReady => opb_exif_0_SYS_AccReady_pin_IBUF,
      SYS_Interrupt => irq ←（追加）
    );

～省略～
end architecture STRUCTURE;
```

リスト2　割り込み制御サンプル・プログラム
（TestApp_Memory.c）

```
// Located in: microblaze_0/include/xparameters.h
#include "xparameters.h"

#include "stdio.h"

#include "xutil.h"
/*MicroBlaze割り込み関数呼び出しのため必要*/
#include "mb_interface.h"


volatile unsigned int detect_int;

user_int () {                                   ┐
    unsigned int *ptr;                          │
                                                │
    /*割り込みクリア*/                          │
    ptr = (unsigned int *)0x50000010;           │ ←（割り込み処理関数）
    *ptr = (unsigned int)0x01;                  │
                                                │
    detect_int = 0x01;                          │
                                                │
}                                               ┘


int main (void) {

    unsigned int *ptr,i,j;

   print("-- Entering main() --\r\n");

    /*DIPスイッチ回路の割り込みイネーブル*/
    ptr = (unsigned int *)0x50000014;
    *ptr = (unsigned int)0x01;

    /*MicroBlazeの割り込みイネーブル*/
    microblaze_enable_interrupts();

    j = 0x01;

    for(;;) {
         ptr = (unsigned int *)0x50000020;
         for (i=0;i<100000;i++)*ptr = j;
         if (j==0x08) j = 0x01; else j = j << 1;

         if (detect_int==0x01) {
              ptr = (unsigned int *)0x50000024;
              xil_printf("Dip value : %d%d%d%d \n\r",
                      (*ptr&0x01),(*ptr&0x02)>>1,
                      (*ptr&0x04)>>2,(*ptr&0x08)>>3);
              detect_int = 0x00;
              }
    }

   print("-- Exiting main() --\r\n");
   return 0;
}
```