按键以及 PIO 口中断实验

1、PIO 内核简介

1)、PIO 寄存器描述

偏移量	寄存器名称		R/W	(n-1)	…..	….	2	1	0
0	数据寄存器	读	R	读入输入引脚的逻辑电平值					
		写	W	向 PIO 口输出写入值					
1	方向寄存器		R/W	控制 PIO 口为输入或者输出方向,0:输入,1:输出					
2	中断屏蔽寄存器		R/W	使能或禁止每个输入口的 PIO，1：使能,0:禁止					
3	边沿捕获寄存器		R/W	当边沿事件发生时对应位置 1					

2)、数据寄存器

读数据寄存器(Data)，返回在输人引脚上出现的值。如果 PIO 内核硬件配置为“output Ports Only”，则读数据寄存器返回未定义的值。写数据寄存器驱动输出口输出写人的值。如果 PIO 内核硬件配置为“Input Ports only "，则写数据寄存器无效．如果 PIO 内核硬件配置在双向模式下，那么方向寄存器中对应的位被设为 1（输出）时，值才输出。

3)、方向寄存器

只有 PIO 工作模式配置为“Bidirectional ports”时，方向寄存器（Direction）才存在。PIO 工作模式（输人、输出或双向）在添加 PIO 内核时指定，且在系统生成后不能改变。在仅为输入或仅为输出模式下，方向寄存器并不存在，此时，该方向寄存器返回未定义的值，写方向寄存器无效。方向寄存器控制每个 PIO 口的数据方向。当方向寄存器中的位 n 设为 1 时，端口 n 为输出模式;0 时，端口 n 为输人模式。复位后，方向寄存器的所有位设置为 0，因此所有双向 I/O 口配置为输人。

4)、中断屏蔽寄存器

当中断屏蔽寄存器（Interruptmask）的位设为 1 时，使能相对应的 PIO 输人口中断。中断操作取决于 PIO 内核的硬件配置，只有配置为输人口时才能进行中断操作。中断屏蔽寄存器只有在硬件配置为“Generate IRQ”时才存在。如果硬件配置未选中“Generate IRQ"，那么读中断屏蔽寄存器返回未定义的值，写中断屏蔽寄存器无效。复位后，中断屏蔽寄存器所有位为 0，因此禁止所有 PIO 口的中断。

5)、边沿捕获寄存器

只要在输人口上检测到边沿事件时，边沿捕获寄存器（Edgecapture）中对应位n置1。Avalon主控制器可读边沿捕获寄存器来确定边沿在哪一个PIO输人口出现。如果在PIO内核选项中未选择(Enable Bit Clearing for Edge Capture Register)则写任意值到边沿捕获寄存器将使寄存器所有位清0。要检测的边沿类型在PIO添加时指定。当硬件配置有边沿捕获功能时，边沿捕获寄存器才存在，否则，读边沿捕获寄存器返回未定义的值写边沿捕获寄存器无效。

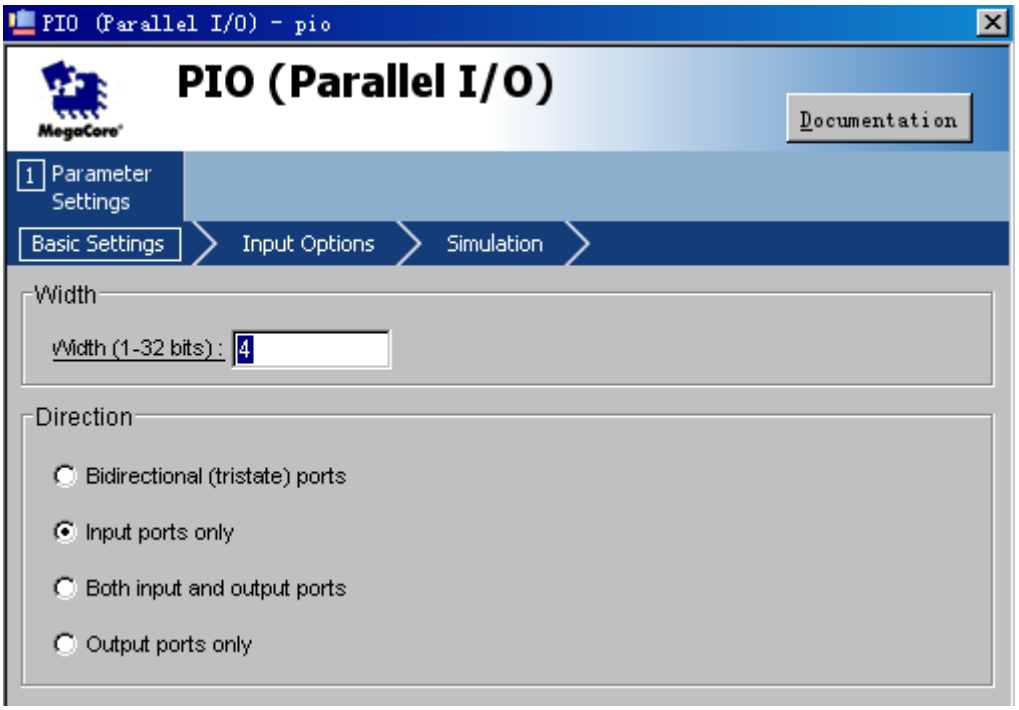
6)、中断操作

当硬件配置为电平触发方式时，只要高电平出现并且中断使能，就申请

一个中断。当硬件配置为边沿触发方式时，只要捕获到边沿事件并且中断使能时，就申请一个中断。中断（IRQ）一直保持有效直至禁止中断（中断屏蔽寄存器相应位清零）或清边沿捕获标志（向边沿捕获寄存器写一个任意值）为止。每个 PIO 核的 I/O 口共用一个中断号（系统生成时指定）、用户需要在中断服务子程序中通过中断掩码的方式来查明是哪个 I / O 产生了中断。

2、PIO 口内核配置

1)、Basic Settings 基本设置



Width 设置为 1~32 之间任意值。

Direction 设置 PIO 口的 4 种不同类型

Bidirectional(tristate)ports :双向(三态)口，该模式下 PIO 口通过控制方向寄存器来控制 PIO 作为输入或者输出。

Input ports only :PIO 仅为输入状态。

Both input and output ports:该模式下，同时生成一个输入口和一个输出口，且输入口与输出口相互独立。

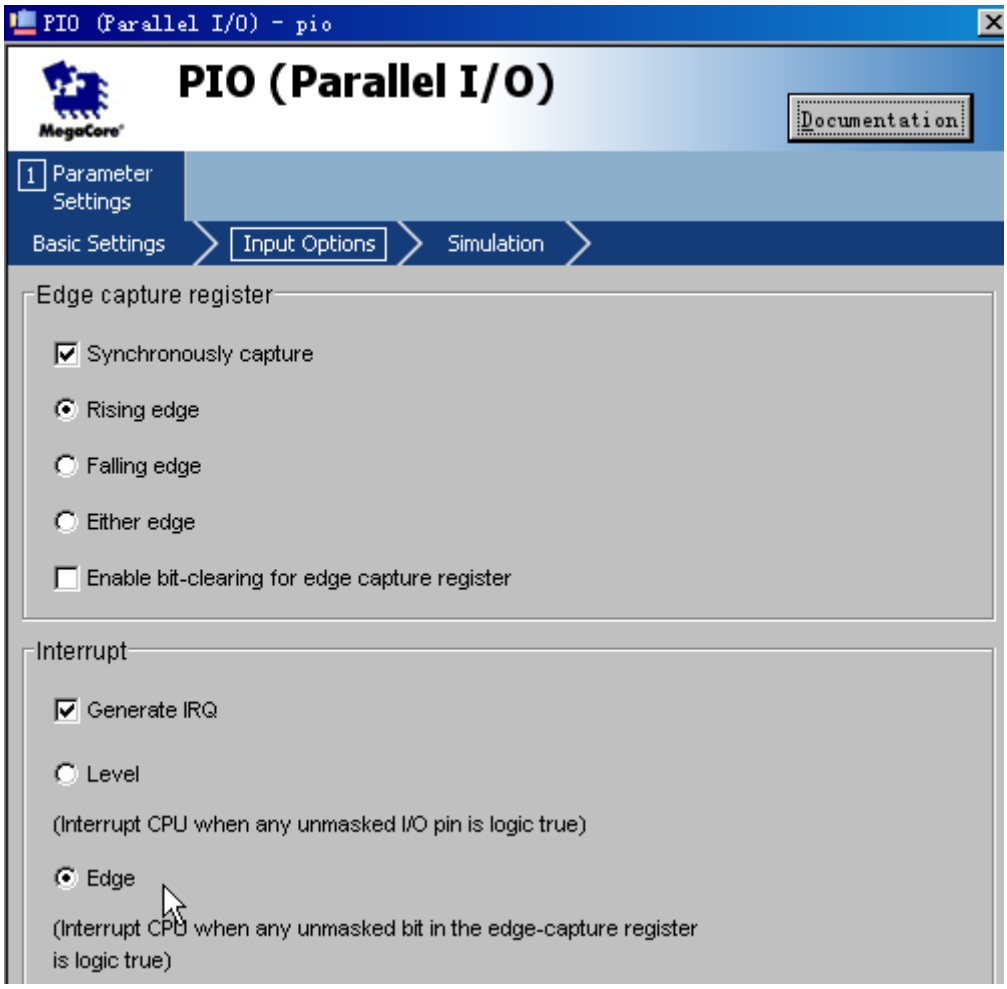
Output ports only:PIO 仅为输出状态。

2)、input Options

设置边沿捕获方式和 IRQ 中断产生方式。边沿捕获方式有上升沿、下降沿、上升或者下降沿。当指定的类型出现在输入端口时，边沿捕获寄存器应对的位置 1。当”Synchronously capture”未选中时，边沿捕获寄存器不存在。

选择 Enable Bit Clearing for Edge Capture Register 时，允许把边沿捕获寄存器某位单独清 0。

中断产生方式有电平触发和边沿触发。电平触发时只要输入口为高电平且中断使能，便产生一个中断请求。边沿触发，只要边沿捕获寄存器中相应位为 1 且中断使能，便产生一个中断请求。当”Generate IRQ”未选中时，中断屏蔽寄存器不存在。



3)、Simulation 仿真

如果需要仿真则需要进行设置。

3、软件编程举例

```
/* altera_avalon_pio_regs.h 中提供了对硬件进行寄存器级访问的宏定义*/
#include "altera_avalon_pio_regs.h"
//#include "altera_avalon_pio.h"
#include "system.h"
#include "alt_types.h"
#include "sys/alt_irq.h"
#include <stdio.h>

#define key IORD_ALTERA_AVALON_PIO_DATA(PIO_KEY_BASE)&0xf
void butto_ISR (); //按键中断服务程序

int main(void)
{
/*注册按键中断*/
 alt_irq_register(PIO_KEY_IRQ,0,butto_ISR);
 /*清pio边沿捕获寄存器*/
```

```
 IOWR_ALTERA_AVALON_PIO_EDGE_CAP(PIO_KEY_BASE, 0x0);
  /*设置pio口为输入*/
IOWR_ALTERA_AVALON_PIO_DIRECTION(PIO_KEY_BASE,0x00);
/*中断使能，即把与按键对应的pio口位置1*/
IOWR_ALTERA_AVALON_PIO_IRQ_MASK(PIO_KEY_BASE,0xf);

 while(1)//等待中断
  { ;
  }
}

void butto_ISR (void * context,alt_u32 id)//按键中断服务程序
{
  alt_u8 keytemp;
  IOWR_ALTERA_AVALON_PIO_DIRECTION(PIO_KEY_BASE,0x00);
  int si=IORD_ALTERA_AVALON_PIO_DATA(PIO_KEY_BASE)&0xf;
   if(si!=0xf)
     keytemp=si;
     data++;
    IOWR_ALTERA_AVALON_PIO_IRQ_MASK(PIO_KEY_BASE,0x0);
  switch(si)
  {
    case 0x07 :{
             printf("This is key1 IRQ\n");
               } break;
    case 0x0b:{
               printf("This is key2 IRQ\n");
              }break;
    case 0x0d:{
               printf("This is key3 IRQ\n");
               }break;
    case 0x0e:{
               printf("This is key4 IRQ\n");
               }

  }

IOWR_ALTERA_AVALON_PIO_EDGE_CAP(PIO_KEY_BASE, 0);
IOWR_ALTERA_AVALON_PIO_IRQ_MASK(PIO_KEY_BASE,0xf);
}
```

参考文档 关于 altera 提供的各种 IP CORE 文档在 SOPC BUILDER 中添加 CORE 时在设置对话框右上角的 

按钮，点击会自动下载 altera 官方的文档文件。