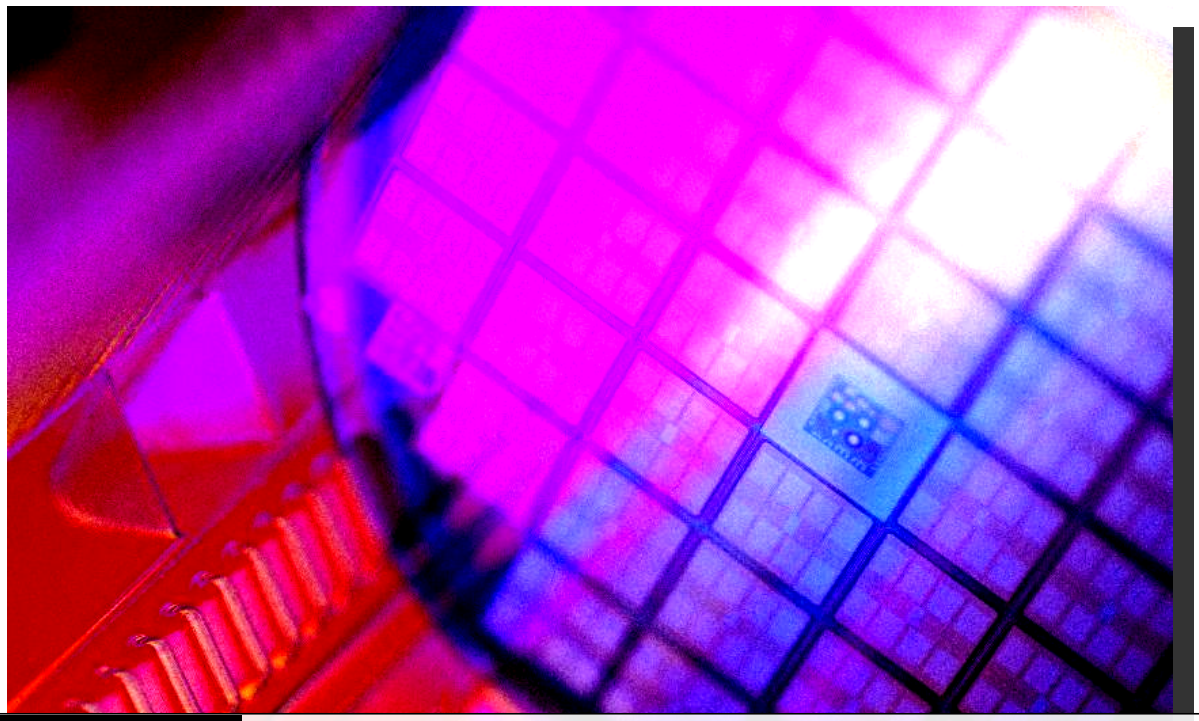
FES1301T05-V0.00



ZRtech

FPGA 开发套件 NIOS 实验教程

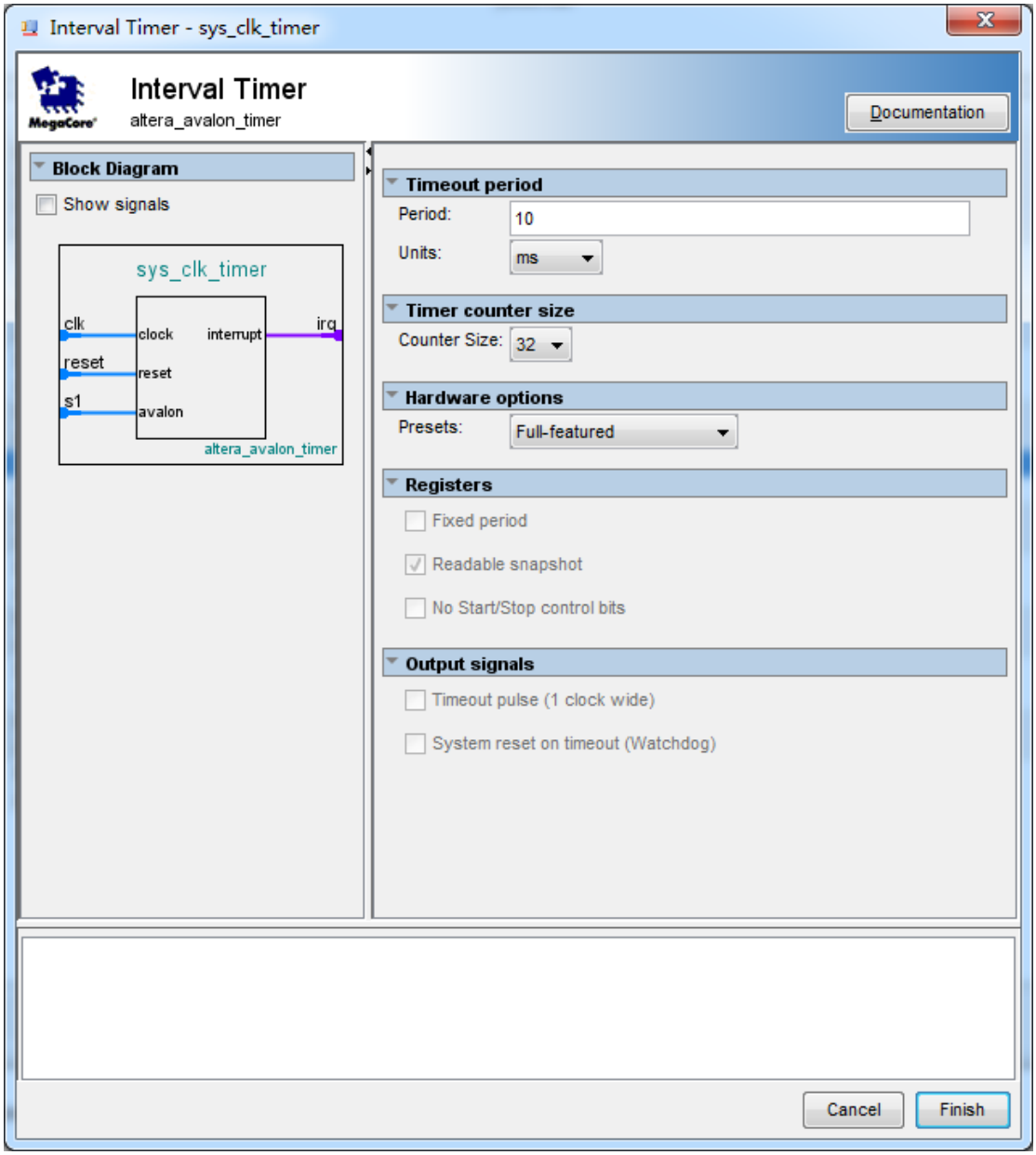
—NIOS 定时器及 PIO

www.zr-tech.com

实验五、NIOS 定时器及 PIO

众所周知，在单片机系统中，定时器是很重要的外设，同样，定时器对于 NIOS II 也是必不可少的。

1. 首先需要在 SOPC 中增加 Interval Timer，如下所示。



2. 在 Hardware Option 里可以把定时器设置为 3 种预设配置：

   a) Simple periodic interrupt：这种情况下适合系统只需要产生一个周期性的中断，其

周期值固定，并且定时器不能停止运行，不过中断可屏蔽；

b) Full-featured：这是我们在单片机系统中最常用的一种设置，定时器可启用可停止，周期可配置；

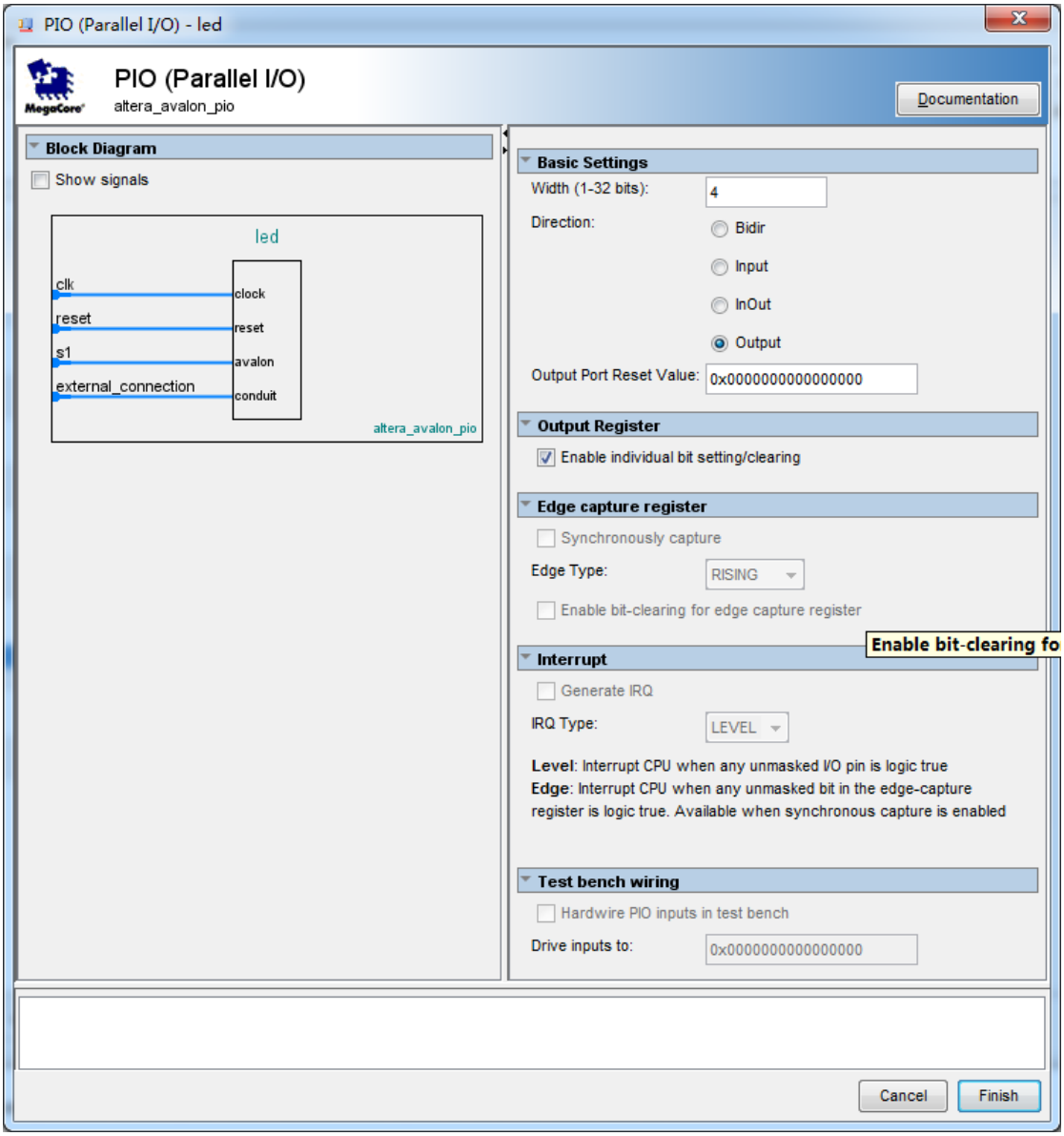
c) Watchdog：这也是一种常见设置，为了防止程序跑飞而产生系统复位。

3. Timeout period 配置为 10ms，表示每个系统 tick 是 10ms；在 system.h 中可以看到系统 tick 为 10ms，每秒有 100 个 tick。

```
#define SYS_CLK_TIMER_PERIOD 10
#define SYS_CLK_TIMER_PERIOD_UNITS "ms"
#define SYS_CLK_TIMER_TICKS_PER_SEC 100u
```

Tick 这个概念如果大家了解过 ucos 等操作系统，就会知道它是系统的基本事件刻度，所有任务的执行都与它有关。在 NIOS 中，Altera 为 tick 准备了专门的 HAL API 函数，参见 sys/alt_alarm.h，由于我们着重关注 Full-featured 定时器，这里就不详细分析了。我认为 Timeout period 只有在 Hardware Option 设置为 Simple periodic interrupt 时才有意义。

4. Espier 开发板上安装了 4 个 led，我们可以把它们设置为 PIO，宽度设为 4，方向设为输出。



5. 生成 SOPC，完成例化，编译并配置 FPGA 之后就可以开始对定时器和 PIO 进行编程了。详细步骤此处不赘述，详见《图解 NIOS 建立》篇。

6. 测试程序借助了网络的力量，如有版权问题，请联系告知

```
#include <stdio.h>

#include <string.h>

#include <system.h>
```

```
#include "altera_avalon_timer_regs.h"
#include "altera_avalon_pio_regs.h"
#include "alt_types.h"
#include "sys/alt_irq.h"



alt_u32 timer_isr_context; // 定义全局变量以储存isr_context指针
void Timer_Initial(void);
void Timer_ISR(void* isr_context);


alt_u8 irq_flag = 0;
alt_u8 led_state = 0xf;


int main()
{
    Timer_Initial(); // 初始化定时器中断


    while(1)
    {
        if(irq_flag == 1)
        {
```

```
            led_state = (led_state >> 1) | (led_state << 7);
            IOWR_ALTERA_AVALON_PIO_DATA(LED_BASE, led_state);
            irq_flag = 0;
        }
    }
    return 0;
}

// 定时器中断初始化
void Timer_Initial(void)
{
    // 改写timer_isr_context指针以匹配alt_irq_register()函数原型
    void* isr_context_ptr = (void*) &timer_isr_context;

    // 设置PERIOD寄存器
    // PERIODH << 16 | PERIODL = 计数器周期因子 * 系统时钟频率因子 - 1
    // PERIODH << 16 | PERIODL = 0.5s*48M - 1 = 23999999 = 0x16E35FF
    IOWR_ALTERA_AVALON_TIMER_PERIODH(SYS_CLK_TIMER_BASE, 0x016E);
    IOWR_ALTERA_AVALON_TIMER_PERIODL(SYS_CLK_TIMER_BASE, 0x35FF);

    // 设置CONTROL寄存器
```

```
    //       位数 |  3   |  2   |  1   |  0  |
    // CONTROL | STOP | START| CONT | ITO |
    // ITO    1，产生IRO；                    0，不产生IRQ
    // CONT   1，计数器连续运行直到STOP被置1；   0，计数到0停止
    // START 1，计数器开始运行；                0，无影响
    // STOP   1，计数器停止运行；                0，无影响
    IOWR_ALTERA_AVALON_TIMER_CONTROL(SYS_CLK_TIMER_BASE,
            ALTERA_AVALON_TIMER_CONTROL_START_MSK | // START = 1
            ALTERA_AVALON_TIMER_CONTROL_CONT_MSK  | // CONT   = 1
            ALTERA_AVALON_TIMER_CONTROL_ITO_MSK);    // ITO    = 1


    // 注册定时器中断
    alt_ic_isr_register(
            SYS_CLK_TIMER_IRQ_INTERRUPT_CONTROLLER_ID, // 中断控制器标号，从
system.h复制
            SYS_CLK_TIMER_IRQ,       // 硬件中断号，从system.h复制
            Timer_ISR,               // 中断服务子函数
            isr_context_ptr,         // 指向与设备驱动实例相关的数据结构体
            0x0);                    // flags，保留未用
}
```

```
// 定时器中断服务子函数
void Timer_ISR(void* timer_isr_context)
{
    // 应答中断，将STATUS寄存器清零
    IOWR_ALTERA_AVALON_TIMER_STATUS(SYS_CLK_TIMER_BASE,
            ~ ALTERA_AVALON_TIMER_STATUS_TO_MSK);    // TO = 0
    // 用户中断代码
    irq_flag = 1;
}
```

7. 程序必须引用的几个头文件

   a) altera_avalon_timer_regs.h 定义了定时器的各种寄存器

   b) altera_avalon_pio_regs.h 定义了 PIO 的各种寄存器

   c) system.h 定义了 SOPC 的基本信息

   d) sys/alt_irq.h 定义了中断 API 函数

文档内部编号：FES1301T05

编号说明：
首一字母：F-FPGA系列
首二字母：L-理论类 E-实验类 T-专题类
首三字母：C-普及类 Q-逻辑类 S-软核类
数字前两位：代表年度
数字后两位：同类文档顺序编号
尾字母/数字：C目录，T正文，数字表示章节号

修订记录

版本号	日期	描述	修改人
0.00	2013.9.25	FES1301T05 文档建立	kdy