External Interrupt Demo

You will learn how to use GPIO as external interrupt and output, and how to program the KEY1 as an external interrupt to control the LED on and off by this demo. As in Figure 1, the KEY1 on the extension board controls one LED on and off.

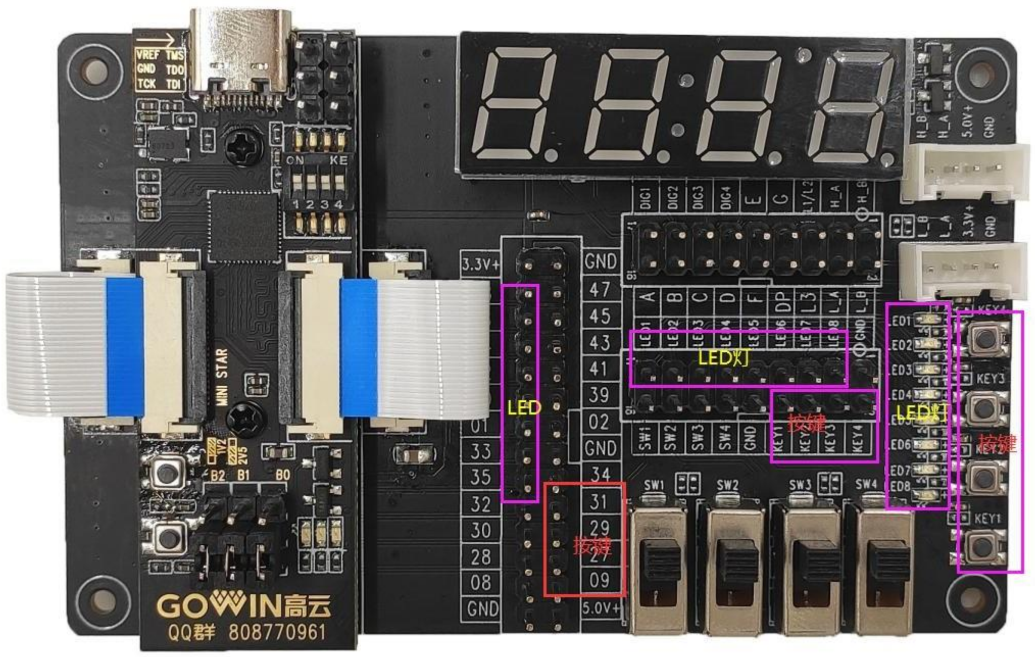


Figure 1 Mini-Star Development Board (GW1NSR-LV4CQN48P) and Extension

This demo includes four parts:

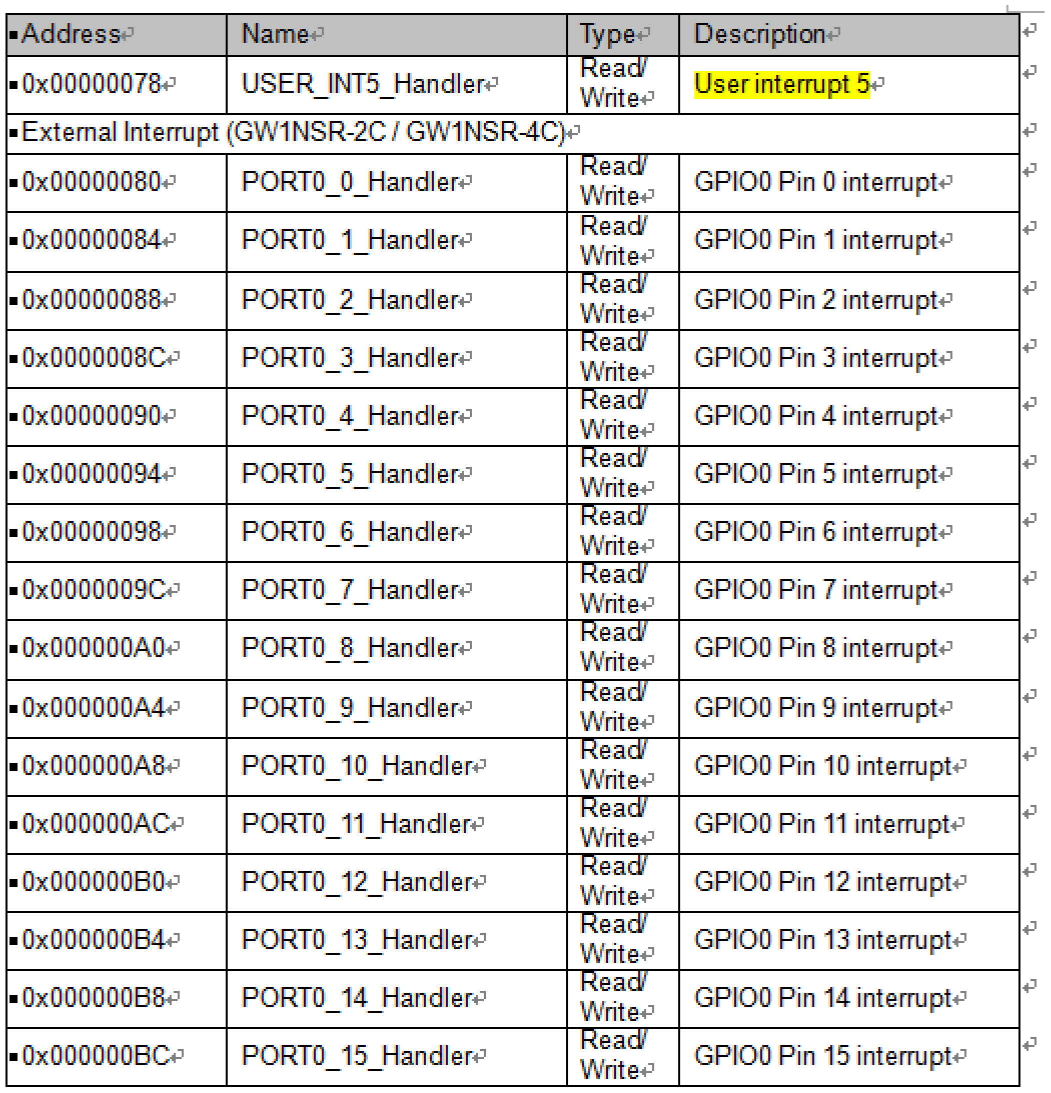
1. Interrupt Introduction
2. Hardware Design
3. Software Design
4. Download and Verification

Interrupt Introduction

Gowin GW1NSR-LV4CQN48P is used as the platform in this demo. From DS861, GW1NSR series of FPGA Products Datasheet, you can learn the NVIC module and GPIO module, and know there is a Cortex-M3 RISC embedded in this chip.

NVIC supports low-latency interrupt processing, and GW1NSR-4C supports 16 interrupts as shown in Table 1.

Table 1 NVIC External Interrupt Vector Table



Address	Name	Type	Description
0x00000078	USER_INT5_Handler	Read/ Write	User interrupt 5
External Interrupt (GW1NSR-2C / GW1NSR-4C)			
0x00000080	PORT0_0_Handler	Read/ Write	GPIO0 Pin 0 interrupt
0x00000084	PORT0_1_Handler	Read/ Write	GPIO0 Pin 1 interrupt
0x00000088	PORT0_2_Handler	Read/ Write	GPIO0 Pin 2 interrupt
0x0000008C	PORT0_3_Handler	Read/ Write	GPIO0 Pin 3 interrupt
0x00000090	PORT0_4_Handler	Read/ Write	GPIO0 Pin 4 interrupt
0x00000094	PORT0_5_Handler	Read/ Write	GPIO0 Pin 5 interrupt
0x00000098	PORT0_6_Handler	Read/ Write	GPIO0 Pin 6 interrupt
0x0000009C	PORT0_7_Handler	Read/ Write	GPIO0 Pin 7 interrupt
0x000000A0	PORT0_8_Handler	Read/ Write	GPIO0 Pin 8 interrupt
0x000000A4	PORT0_9_Handler	Read/ Write	GPIO0 Pin 9 interrupt
0x000000A8	PORT0_10_Handler	Read/ Write	GPIO0 Pin 10 interrupt
0x000000AC	PORT0_11_Handler	Read/ Write	GPIO0 Pin 11 interrupt
0x000000B0	PORT0_12_Handler	Read/ Write	GPIO0 Pin 12 interrupt
0x000000B4	PORT0_13_Handler	Read/ Write	GPIO0 Pin 13 interrupt
0x000000B8	PORT0_14_Handler	Read/ Write	GPIO0 Pin 14 interrupt
0x000000BC	PORT0_15_Handler	Read/ Write	GPIO0 Pin 15 interrupt

A programmable priority level of 0-7 is for each interrupt. A higher level corresponds to a lower priority; for example, level 0 is the highest interrupt priority and level 7 is the lowest.

Table 2 Interrupt Priority Group

Num.	Preemption Priority Width	Sub Priority Width	Parameter
0	0	3	NVIC_PriorityGroup_0
1	1	2	NVIC_PriorityGroup_1
2	2	1	NVIC_PriorityGroup_2
3	3	0	NVIC_PriorityGroup_3

You can use interrupt group library function to configure.

```
void NVIC_PriorityGroupConfig(uint32_t NVIC_PriorityGroup);
```

Such as `NVIC_PriorityGroupConfig(NVIC_PriorityGroup_3);`

The interrupt attributes can be configured through structures, which are defined as follows:

```
typedef struct
{
  uint8_t NVIC_IRQChannel; /* interrupt request*/
  uint8_t NVIC_IRQChannelPreemptionPriority; /*preemption priority */
  uint8_t NVIC_IRQChannelSubPriority; /* sub priority*/
  FunctionalState NVIC_IRQChannelCmd; /* interrupt request command*/
} NVIC_InitTypeDef;
```

The interrupt request definition in the gw1ns4c.h file is as shown below.

```
typedef enum IRQn
{
/****** Cortex-M3 Processor Exceptions Numbers ********************************/
NonMaskableInt_IRQn = -14, /*!< 2 Cortex-M3 Non Maskable Interrupt */
HardFault_IRQn = -13, /*!< 3 Cortex-M3 Hard Fault Interrupt */
MemoryManagement_IRQn = -12, /*!< 4 Cortex-M3 Memory Management Interrupt */
BusFault_IRQn = -11, /*!< 5 Cortex-M3 Bus Fault Interrupt */
UsageFault_IRQn = -10, /*!< 6 Cortex-M3 Usage Fault Interrupt */
SVCall_IRQn = -5, /*!< 11 Cortex-M3 SV Call Interrupt */
DebugMonitor_IRQn = -4, /*!< 12 Cortex-M3 Debug Monitor Interrupt */
PendSV_IRQn = -2, /*!< 14 Cortex-M3 Pend SV Interrupt */
SysTick_IRQn = -1, /*!< 15 Cortex-M3 System Tick Interrupt */
/****** GW1NS-4C Specific Interrupt Numbers ***********************************/
UART0_IRQn = 0, /* UART 0 RX and TX Combined Interrupt */
USER_INT0_IRQn = 1, /* Interrupt handler 0 to user extension */
UART1_IRQn = 2, /* UART 1 RX and TX Combined Interrupt */
USER_INT1_IRQn = 3, /* Interrupt handler 1 to user extension */
USER_INT2_IRQn = 4, /* Interrupt handler 2 to user extension */
RTC_IRQn = 5, /* RTC Interrupt */

PORT0_COMB_IRQn = 6, /* GPIO Port 0 combined Interrupt */
USER_INT3_IRQn = 7, /* Interrupt handler 3 to user extension */
TIMER0_IRQn = 8, /* TIMER 0 Interrupt */
TIMER1_IRQn = 9, /* TIMER 1 Interrupt */
I2C_IRQn = 11, /* I2C */
UART0VF_IRQn = 12, /* UART 0,1 Overflow Interrupt */
USER_INT4_IRQn = 13, /* Interrupt handler 4 to user extension */
USER_INT5_IRQn = 14, /* Interrupt handler 5 to user extension */
Spare15_IRQn = 15, /* Undefined */
PORT0_0_IRQn = 16, /*!< All P0 I/O pins can be used as interrupt source.*/
PORT0_1_IRQn = 17, /*!< There are 16 pins in total */
PORT0_2_IRQn = 18, /*! PORT0_2 Interrupt */
PORT0_3_IRQn = 19, /*! PORT0_3 Interrupt */

PORT0_4_IRQn = 20, /*! PORT0_4 Interrupt */

PORT0_5_IRQn = 21, /*! PORT0_5 Interrupt */
PORT0_6_IRQn = 22, /*! PORT0_6 Interrupt */
PORT0_7_IRQn = 23, /*! PORT0_7 Interrupt */
PORT0_8_IRQn = 24, /*! PORT0_8 Interrupt */
PORT0_9_IRQn = 25, /*! PORT0_9 Interrupt */
PORT0_10_IRQn = 26, /*! PORT0_10 Interrupt */
PORT0_11_IRQn = 27, /*! PORT0_11 Interrupt */
PORT0_12_IRQn = 28, /*! PORT0_12 Interrupt */
PORT0_13_IRQn = 29, /*! PORT0_13 Interrupt */
PORT0_14_IRQn = 30, /*! PORT0_14 Interrupt */
PORT0_15_IRQn = 31 /*! PORT0_15 Interrupt */

}IRQn_Type;
```

Interrupt signal can be detected by level and pulse. For the details, you can see GPIO register shown in Table 3. The processor is automatically saved when the interrupt is entered and automatically restored when the interrupt is exited, no additional instruction is required.

Table 3 GPIO Register

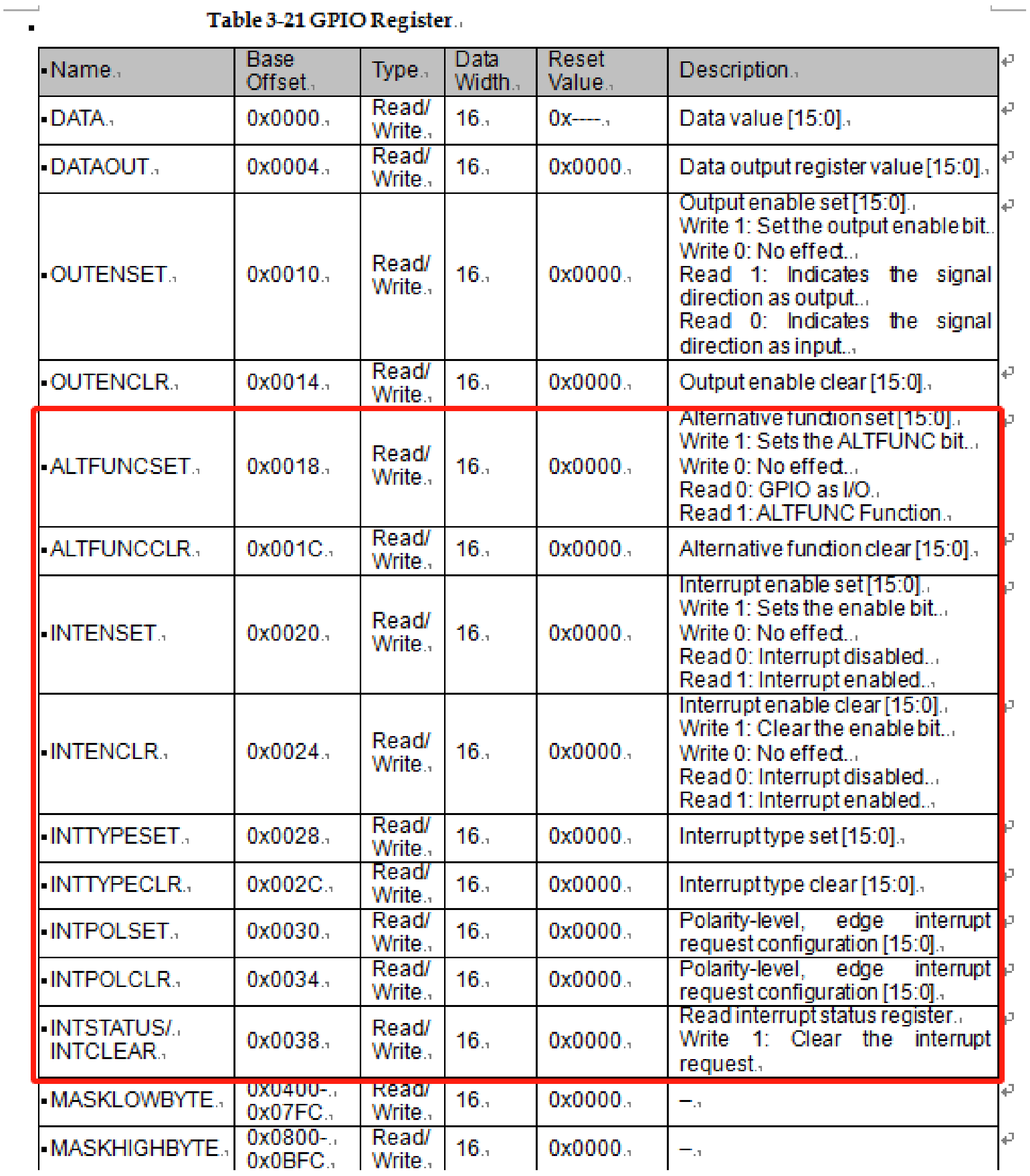


Table 3-21 GPIO Register

Name	Base Offset	Type	Data Width	Reset Value	Description
DATA	0x0000	Read/ Write	16	0x----	Data value [15:0]
DATAOUT	0x0004	Read/ Write	16	0x0000	Data output register value [15:0]
OUTENSET	0x0010	Read/ Write	16	0x0000	Output enable set [15:0] Write 1: Set the output enable bit Write 0: No effect Read 1: Indicates the signal direction as output Read 0: Indicates the signal direction as input
OUTENCLR	0x0014	Read/ Write	16	0x0000	Output enable clear [15:0]
ALTFUNCSET	0x0018	Read/ Write	16	0x0000	Alternative function set [15:0] Write 1: Sets the ALTFUNC bit Write 0: No effect Read 0: GPIO as I/O Read 1: ALTFUNC Function
ALTFUNCCLR	0x001C	Read/ Write	16	0x0000	Alternative function clear [15:0]
INTENSET	0x0020	Read/ Write	16	0x0000	Interrupt enable set [15:0] Write 1: Sets the enable bit Write 0: No effect Read 0: Interrupt disabled Read 1: Interrupt enabled
INTENCLR	0x0024	Read/ Write	16	0x0000	Interrupt enable clear [15:0] Write 1: Clear the enable bit Write 0: No effect Read 0: Interrupt disabled Read 1: Interrupt enabled
INTTYPESET	0x0028	Read/ Write	16	0x0000	Interrupt type set [15:0]
INTTYPECLR	0x002C	Read/ Write	16	0x0000	Interrupt type clear [15:0]
INTPOLSET	0x0030	Read/ Write	16	0x0000	Polarity-level, edge interrupt request configuration [15:0]
INTPOLCLR	0x0034	Read/ Write	16	0x0000	Polarity-level, edge interrupt request configuration [15:0]
INTSTATUS/ INTCLEAR	0x0038	Read/ Write	16	0x0000	Read interrupt status register Write 1: Clear the interrupt request
MASKLOWBYTE	0x0400-0x07FC	Read/ Write	16	0x0000	–
MASKHIGHBYTE	0x0800-0x0BFC	Read/ Write	16	0x0000	–

External interrupt requests are sent to the NVIC module via GPIO, and external interrupts can be programmed by configuring the GPIO and NVIC.

Note!

For the details, you can see 3.11.10 GPIO in DS861, GW1NSR series of FPGA Products Datasheet.

Hardware Design

This demo can be modified on the basis of the LED project.

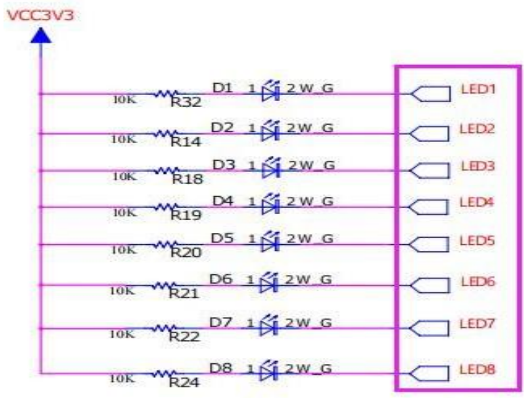


Figure 2 LED Schematic

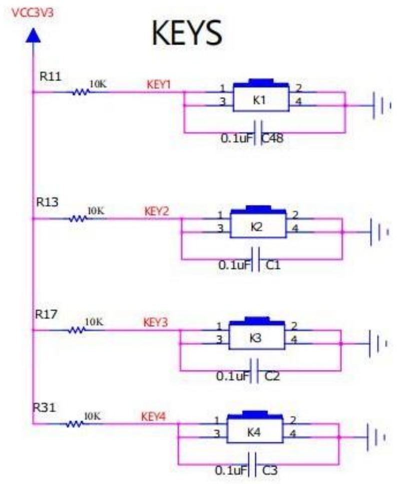


Figure 3 KEY Schematic

From the schematic, you can know LED is on in LOW and off in HIGH. When the key pressed, it is low level; when the key released, it is high level. This demo only needs one key as the external interrupt and one LED as the display. Therefore, only KEY1 and LED1 need to be connected to the corresponding FPGA ports. For example, KEY1 is connected to 45 and LED1 is connected to 40 as shown in Figure 4.

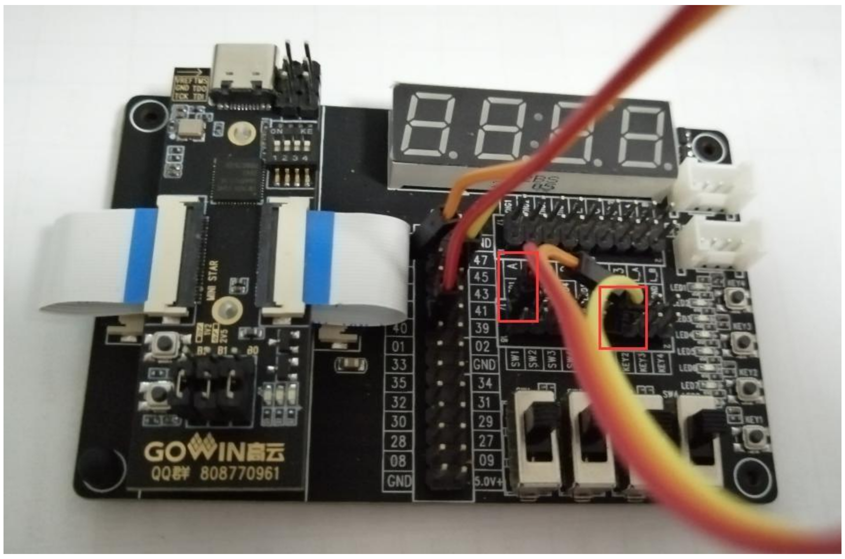


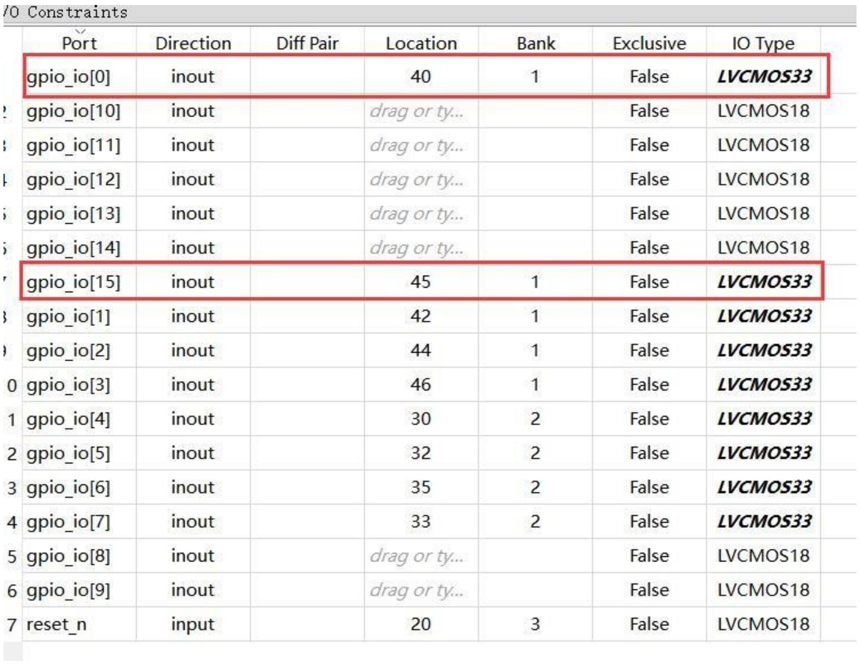
Figure 4 Connection of LED and FPGA

Software Design

The software design includes two parts: FPGA internal hardware logic and Cotex-M3 software control code, which can be modified on the basis of the led project.

FPGA Internal Logic Design

This HDL code does not need to be modified, and you only need to modify the pin connection according to the table, which has been described above. Figure 5 shows the pin definition.



/O Constraints

Port	Direction	Diff Pair	Location	Bank	Exclusive	IO Type
gpio_io[0]	inout		40	1	False	*LVCMOS33*
gpio_io[10]	inout		drag or ty...		False	LVCMOS18
gpio_io[11]	inout		drag or ty...		False	LVCMOS18
gpio_io[12]	inout		drag or ty...		False	LVCMOS18
gpio_io[13]	inout		drag or ty...		False	LVCMOS18
gpio_io[14]	inout		drag or ty...		False	LVCMOS18
gpio_io[15]	inout		45	1	False	*LVCMOS33*
gpio_io[1]	inout		42	1	False	*LVCMOS33*
gpio_io[2]	inout		44	1	False	*LVCMOS33*
gpio_io[3]	inout		46	1	False	*LVCMOS33*
gpio_io[4]	inout		30	2	False	*LVCMOS33*
gpio_io[5]	inout		32	2	False	*LVCMOS33*
gpio_io[6]	inout		35	2	False	*LVCMOS33*
gpio_io[7]	inout		33	2	False	*LVCMOS33*
gpio_io[8]	inout		drag or ty...		False	LVCMOS18
gpio_io[9]	inout		drag or ty...		False	LVCMOS18
reset_n	input		20	3	False	LVCMOS18

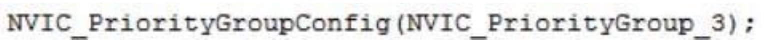
Figure 5 FPGA Pin Configuration

Then click Place & Route to generate the logic file fpga_led.fs.

Cotex-M3 Software Control Design

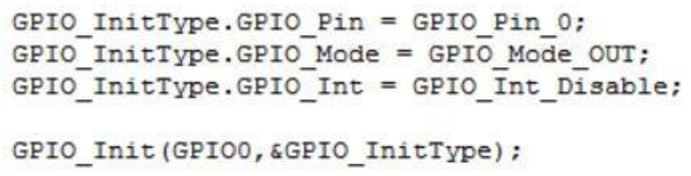
You can modify this design on the basis of the led project. Open Led.uvprojx in the Keil_led\PROJECT folder.

1. Group interrupt



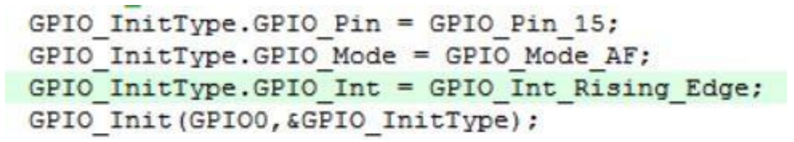
```
NVIC_PriorityGroupConfig(NVIC_PriorityGroup_3);
```

2. Set GPIO0[0] to output to drive LED.



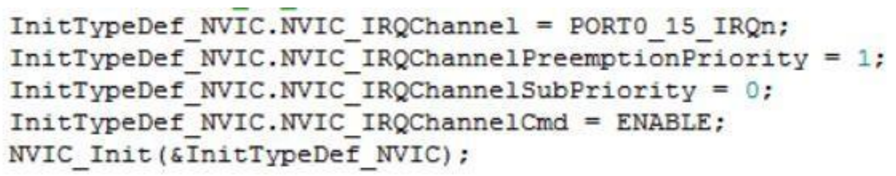
```
GPIO_InitType.GPIO_Pin = GPIO_Pin_0;
GPIO_InitType.GPIO_Mode = GPIO_Mode_OUT;
GPIO_InitType.GPIO_Int = GPIO_Int_Disable;

GPIO_Init(GPIO0,&GPIO_InitType);
```

3. Set GPIO0[15] to rising-edge external interrupt.



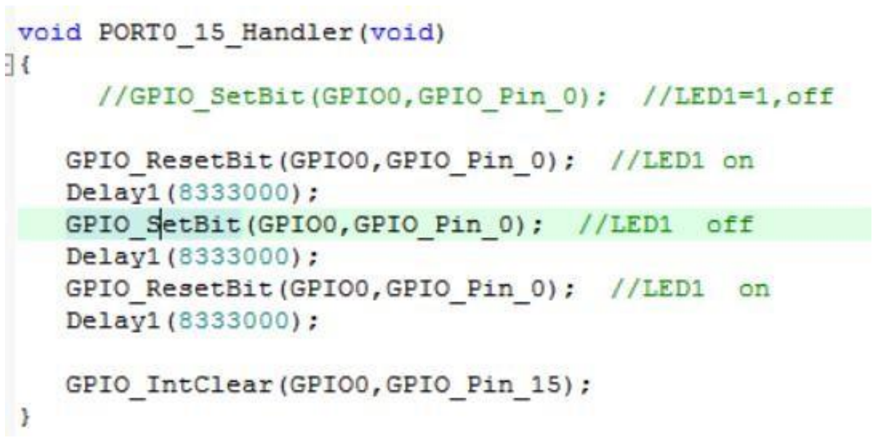
```
GPIO_InitType.GPIO_Pin = GPIO_Pin_15;
GPIO_InitType.GPIO_Mode = GPIO_Mode_AF;
GPIO_InitType.GPIO_Int = GPIO_Int_Rising_Edge;
GPIO_Init(GPIO0,&GPIO_InitType);
```

4. Set GPIO0[15] attributes.



```
InitTypeDef_NVIC.NVIC_IRQChannel = PORT0_15_IRQn;
InitTypeDef_NVIC.NVIC_IRQChannelPreemptionPriority = 1;
InitTypeDef_NVIC.NVIC_IRQChannelSubPriority = 0;
InitTypeDef_NVIC.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&InitTypeDef_NVIC);
```

5. Program interrupt, and the library function is in the gw1ns4c_it.c.



```
void PORT0_15_Handler(void)
{
     //GPIO_SetBit(GPIO0,GPIO_Pin_0);  //LED1=1,off

   GPIO_ResetBit(GPIO0,GPIO_Pin_0);  //LED1 on
   Delay1(8333000);
   GPIO_SetBit(GPIO0,GPIO_Pin_0);  //LED1  off
   Delay1(8333000);
   GPIO_ResetBit(GPIO0,GPIO_Pin_0);  //LED1  on
   Delay1(8333000);

   GPIO_IntClear(GPIO0,GPIO_Pin_15);
}
```

6. After bulid, the download file led.bin is generated.

Download and Verification

Use Gowin Software to download, and the demo running is as shown in Figure 6. The FPGA hardware platform file is fpga_led.fs, and the Cotex-M3 software file is led.bin, so be careful to choose the correct file path and bulid file.

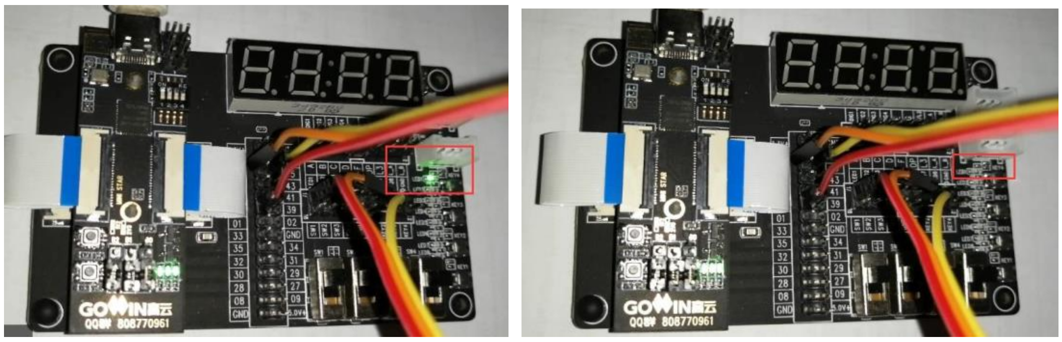

LED ON LED OFF

Figure 6 Demo Running