UART TX RX Demo

You will learn UART and its interrupt programming by this demo. In order not to add new peripherals, use UART TX RX to control the times of LED on/off. As shown in Figure 1, short-circuit UART TX RX pins, and control the times of LED1 on/off according to the RX data.

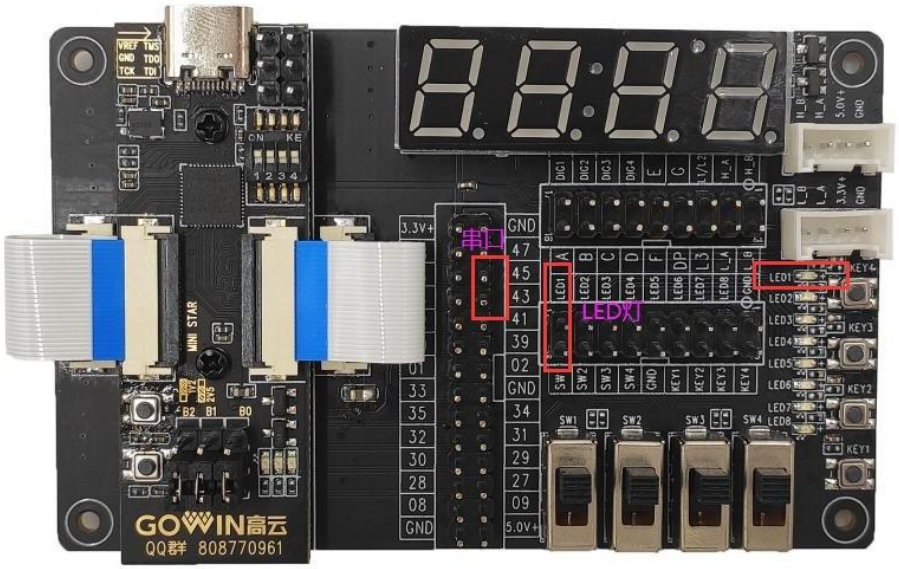


Figure 1 Mini-Star Development Board (GW1NSR-LV4CQN48P) and Extension

This demo includes four parts:

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

UART Introduction

Gowin GW1NSR-LV4CQN48P is used as the platform in this demo. From DS861, GW1NSR series of FPGA Products Datasheet, you can learn the UART module.

The SoC embedded with microprocessor system contains two UARTs: UART0 and UART1. These can be accessed and controlled through APB1 bus. The max. baud rate supported is 921.6Kbits/s. UART0 and UART1 support 8 bits communication without parity and one stop bit. APB UART Buffering diagram is as shown in Figure 2.

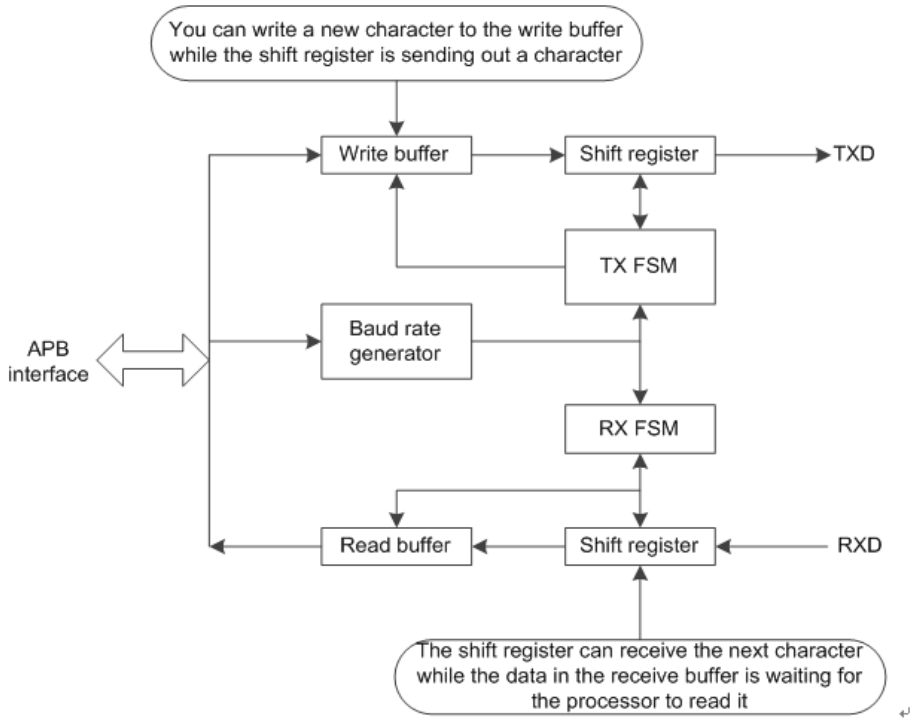


Figure 2 APB UART Buffering

As shown in Figure 2, UART transmits and receives data to/from CPU through the buffers, and the baud rate is related to the APB. Figure 3 lists UART registers.

Name	Base Offset	Type	Data Width	Reset Value	Description
DATA	0x000	Read/ Write	8	0x--	8 bits data Read: Received data. Write: Transmit data.
STATE	0x004	Read/ Write	4	0x0	[3]: RX buffer overrun, write 1 to clear. [2]: TX buffer overrun, write 1 to clear. [1]: RX buffer full, read-only. [0]: TX buffer full, read-only.
CTRL	0x008	Read/ Write	7	0x00	[6]: High-speed test mode for TX only. [5]: RX overrun interrupt enable. [4]: TX overrun interrupt enable. [3]: RX interrupt enable. [2]: TX interrupt enable. [1]: RX enable. [0]: TX enable.
INTSTATUS /INTCLEAR	0x00C	Read/ Write	4	0x0	[3]: RX overrun interrupt, write 1 to clear. [2]: TX overrun interrupt, write 1 to clear. [1]: RX interrupt, write 1 to clear. [0]: TX interrupt, write 1 to clear.
BAUDDIV	0x010	Read/ Write	20	0x00000	[19:0]: Baud rate divider. The minimum number is 16.

Figure 3 UART Registers

To program UART, the first step is to configure UART registers, and also to configure the nested vector interrupt controller (NVIC). For the NVIC details, you can see 3.11.4 section of DS861, GW1NSR series of FPGA Products Datasheet.

The register configuration is defined in gw1ns4c_uart.h in the library file directory, and the definition is as shown below:

```
typedef struct
{
uint32_t UART_BaudRate; /*UART baudrate*/
UARTMode_TypeDef UART_Mode; /*UART mode: TX RX*/
UARTInt_TypeDef UART_Int; /* UART Interrupt enable: TX, RX interrupt*/
UARTOvr_TypeDef UART_Ovr; /* UART overrun interrupt enable: TX, RX overrun interrupt
*/
FunctionalState UART_Hstm; /* UART high-speed test mode*/

}UART_InitTypeDef;
```

Note!

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

Hardware Design

This demo can be modified on the basis of the LED project. LED is on in LOW and off in HIGH. In this demo, short-circuit UART TX RX pins and use an LED as the display to test. Therefore, only **LED1 needs to be connected to the corresponding FPGA** port.

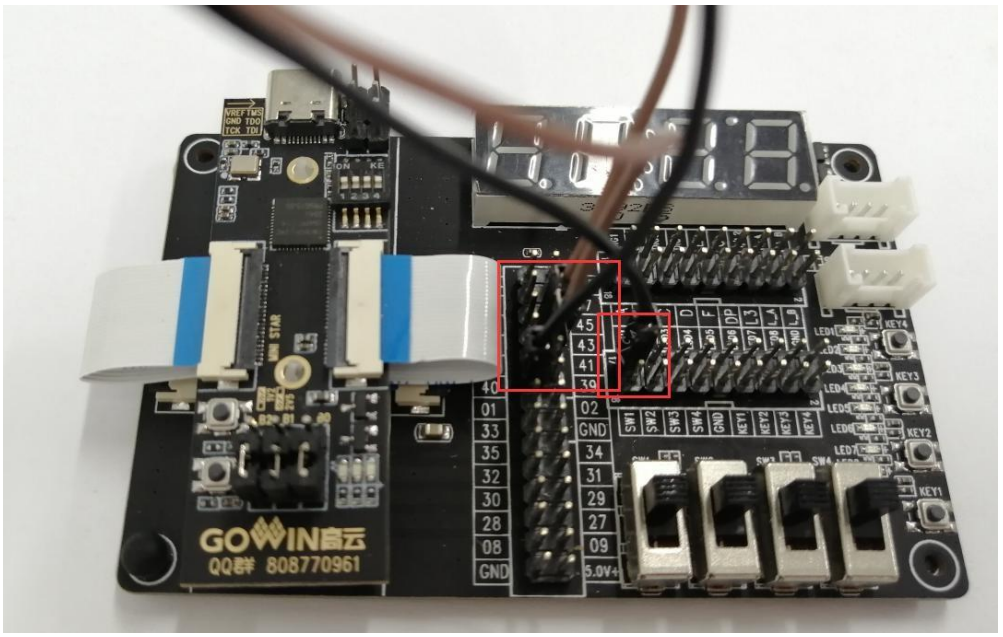


Figure 4 Connection

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

You need to modify the HDL to add UART module to the IP.

Step 1: Add UART module to the IP

Click "IP Core Generator" to open the gowin_empu.ipc file in the FPGA project folder as shown in Figure 5.

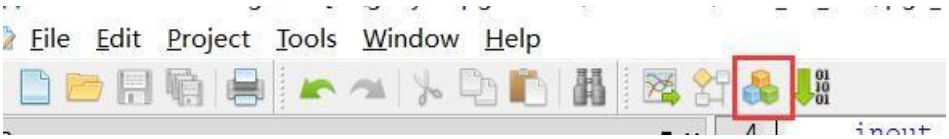


Figure 5 IP Core Generator Icon

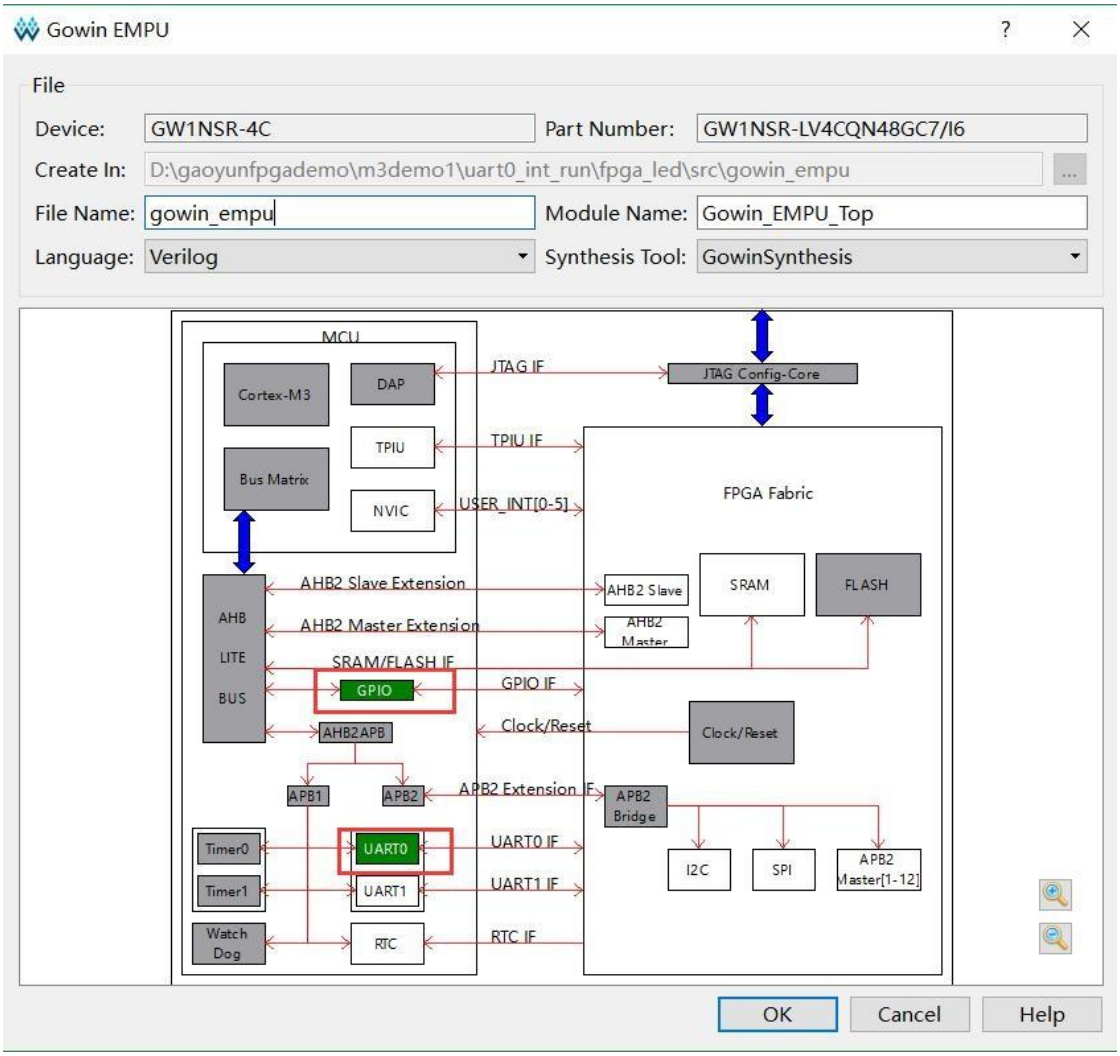


Figure 6 Gowin EMPU Configuration Interface

Double-click on the UART and select UART0 as shown in Figure 6. Click "OK" to generate a new core configuration file, and add it to the project file.

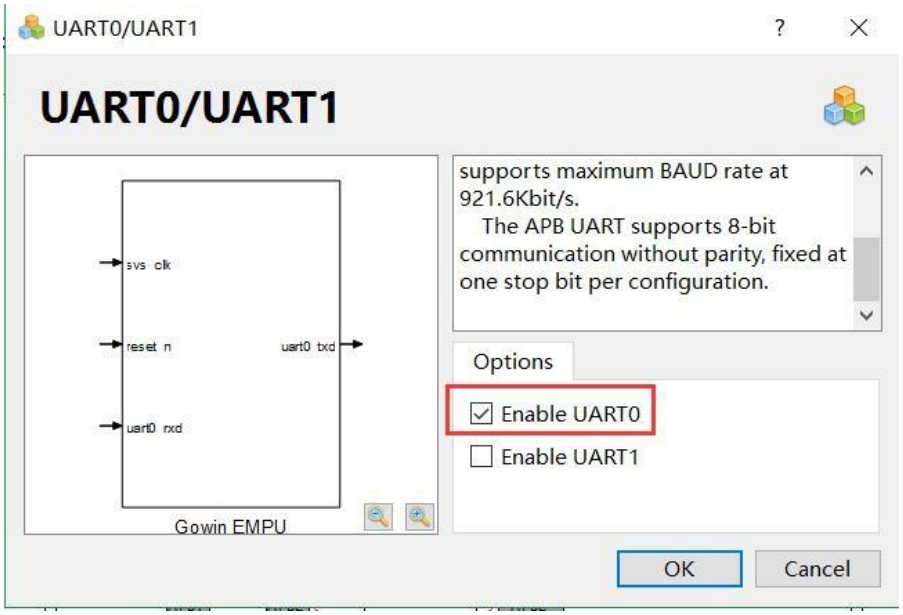


Figure 7 Select Enable UART0

Step 2: Modify top file

Add UART TX RX pins to the top, and connect UART signals generated by the core to the interface signals defined by the top, as shown in Figure 8. Click "Synthesize" to synthesize the file.

```
module top(
    inout [15:0] gpio_io,
    input reset_n,
    input  uart0_rxd,
    output  uart0_txd
);

wire m3_clk;

    Gowin_OSC U_Gowin_OSC(
        .oscout  (m3_clk),
        .oscen   (1)
    );



    Gowin_EMPU_Top your_instance_name(
        .sys_clk (m3_clk),  //input sys_clk
        .gpio    (gpio_io), //inout [15:0] gpio
        .reset_n (reset_n),  //input reset_n
        .uart0_rxd(uart0_rxd),          //input uart0 rxd
        .uart0_txd(uart0_txd)           //output  uart0 txd
    );


endmodule
```

Figure 8 Top Module

Step 3: Define UART pin

Double-click on "FloorPlanner" to define the pins. The pin definition is shown in Figure 9.

I/O Constraints

	Port	Direction	Diff Pair	Location	Bank	Exclusive	IO Type
1	gpio_io[0]	inout		40	1	False	LVCMOS33
2	gpio_io[10]	inout		drag or ty...		False	LVCMOS18
3	gpio_io[11]	inout		drag or ty...		False	LVCMOS18
4	gpio_io[12]	inout		drag or ty...		False	LVCMOS18
5	gpio_io[13]	inout		drag or ty...		False	LVCMOS18
6	gpio_io[14]	inout		drag or ty...		False	LVCMOS18
7	gpio_io[15]	inout		drag or ty...		False	LVCMOS18
8	gpio_io[1]	inout		42	1	False	LVCMOS33
9	gpio_io[2]	inout		44	1	False	LVCMOS33
10	gpio_io[3]	inout		46	1	False	LVCMOS33
11	gpio_io[4]	inout		30	2	False	LVCMOS33
12	gpio_io[5]	inout		32	2	False	LVCMOS33
13	gpio_io[6]	inout		35	2	False	LVCMOS33
14	gpio_io[7]	inout		33	2	False	LVCMOS33
15	gpio_io[8]	inout		drag or ty...		False	LVCMOS18
16	gpio_io[9]	inout		drag or ty...		False	LVCMOS18
17	reset_n	input		20	3	False	LVCMOS18
18	uart0_rxd	input		45	1	False	LVCMOS33
19	uart0_txd	output		43	1	False	LVCMOS33

Figure 9 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.

```
//Initializes GPIO
void GPIOInit(void)
{
  GPIO_InitTypeDef GPIO_InitType;

  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);

  GPIO_SetBit(GPIO0,GPIO_Pin_0);
}
```

3. Set UART registers

```
//uart0
UART_InitStruct.UART_Mode.UARTMode_Tx = ENABLE;
UART_InitStruct.UART_Mode.UARTMode_Rx = ENABLE;
UART_InitStruct.UART_Int.UARTInt_Tx = DISABLE;
UART_InitStruct.UART_Int.UARTInt_Rx = ENABLE;//Enable UART0 RX interrupt register
UART_InitStruct.UART_Ovr.UARTOvr_Tx = DISABLE;
UART_InitStruct.UART_Ovr.UARTOvr_Rx = DISABLE;
UART_InitStruct.UART_Hstm = DISABLE;
UART_InitStruct.UART_BaudRate = 115200;//Baud Rate

UART_Init(UART0,&UART_InitStruct);
```

4. Set UART0 interrupts

```
//NVIC: Enable UART0 interrupt handler
InitTypeDef_NVIC.NVIC_IRQChannel = UART0_IRQn;
InitTypeDef_NVIC.NVIC_IRQChannelPreemptionPriority = 1;
InitTypeDef_NVIC.NVIC_IRQChannelSubPriority = 1;
InitTypeDef_NVIC.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&InitTypeDef_NVIC);
```

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

```
void UART0_Handler(void)
{

    char num = '0';

  if(UART_GetRxIRQStatus(UART0) == SET)
  {
    num  = UART_ReceiveChar(UART0);

    if(num == '1')
    {
      GPIO_ResetBit(GPIO0,GPIO_Pin_0);
      delay_ms(500);
      GPIO_SetBit(GPIO0,GPIO_Pin_0);
      delay_ms(500);

    }
    else if(num == '2')
    {
      for(int i = 0;i < 2;i++)
      {
        GPIO_ResetBit(GPIO0,GPIO_Pin_0);
        delay_ms(500);
        GPIO_SetBit(GPIO0,GPIO_Pin_0);
        delay_ms(500);
      }
    }
    else if(num == '3')
    {
      for(int i = 0;i < 3;i++)
      {
        GPIO_ResetBit(GPIO0,GPIO_Pin_0);
        delay_ms(500);
        GPIO_SetBit(GPIO0,GPIO_Pin_0);
        delay_ms(500);
      }
    }
    else if(num == '4')
    {
      for(int i = 0;i < 4;i++)
      {
        GPIO_ResetBit(GPIO0,GPIO_Pin_0);
        delay_ms(500);
        GPIO_SetBit(GPIO0,GPIO_Pin_0);
        delay_ms(500);
      }
    }
    else
    {
      GPIO_ResetBit(GPIO0,GPIO_Pin_0);
    }
  }

  UART_ClearRxIRQ(UART0);
}
```

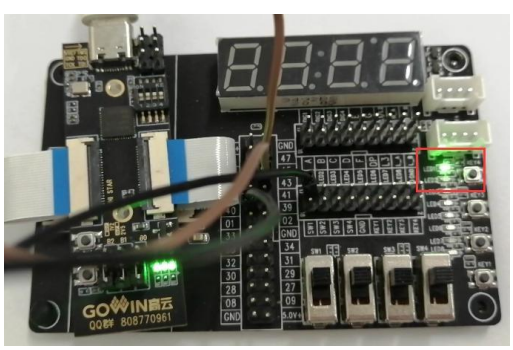
6. In the main design, add init function and set UART_sendchar to 4.

```
int main(void)
{
  //char i;

  SystemInit();
  NVIC_PriorityGroupConfig(NVIC_PriorityGroup_3);
  GPIOInit();
  Uart0Init();


  UART_SendChar(UART0, '4');

  while(1)
  ;
}
```

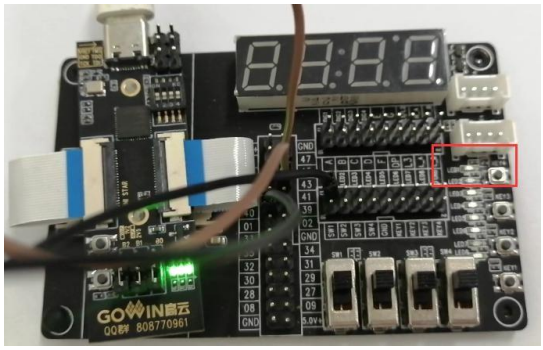
7. After bulid, the download file led.bin is generated.

Download and Verification

Use Gowin Software to download, and the demo running is as shown in Figure 10. 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 10 Demo Running