【C906-SOC最小系统搭建】(3)FPGA移植

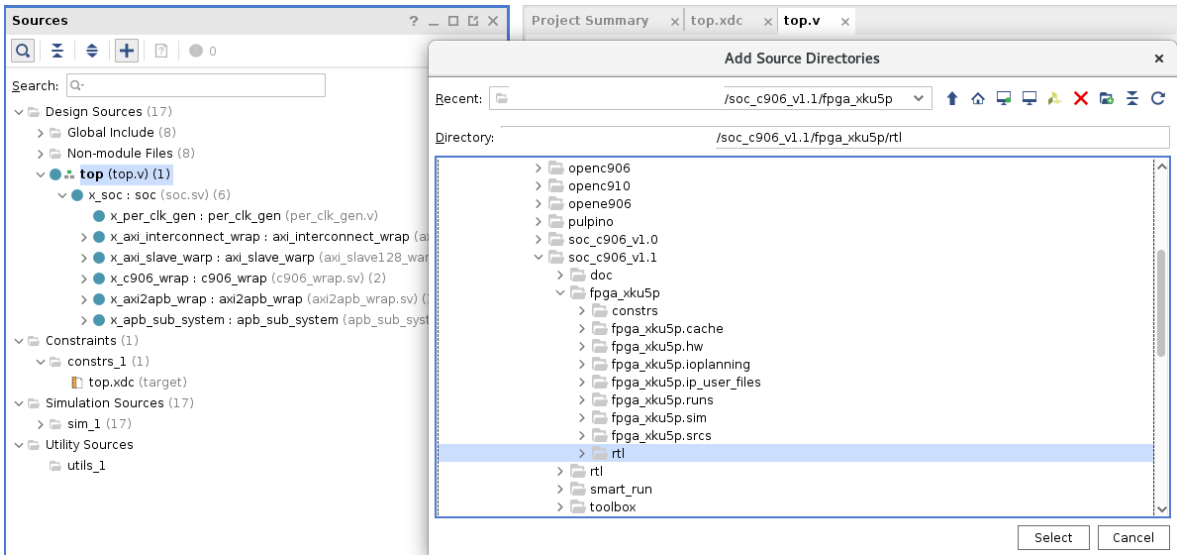
0. 开发环境

1. 本工程使用Alinx AXKU5开发板，FPGA核心为xcku5p-ffvb676-2-i；
2. 本工程使用vivado版本为2018.3；
3. 本工程使用剑池CDK作为C程序集成开发环境；
4. 本工程使用CKLink Lite下载器烧录程序到SRAM；

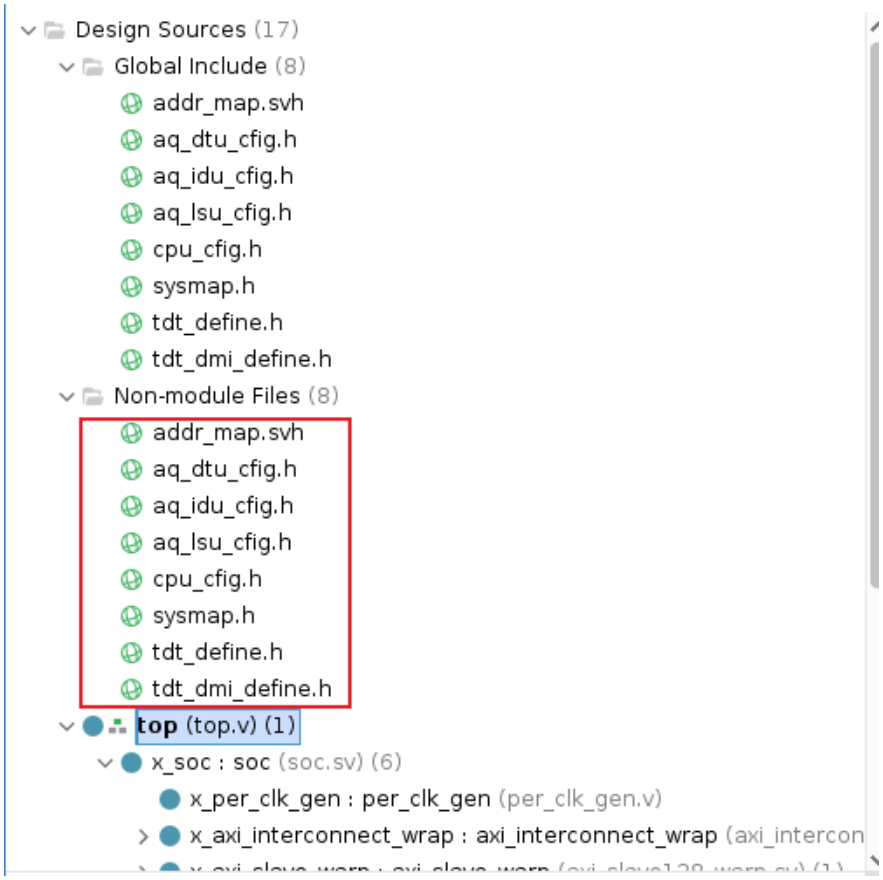
1. FPGA工程搭建

1）导入RTL代码

在fpga工程中添加代码目录



将头文件都设置为global include



2）添加顶层代码

Alinx AXKU5开发板带有一个200MHz的差分时钟输入，在顶层添加一个IBUFDS将差分时钟转为单端时钟。

此外，使用的CKLink Lite下载器只有nrst引脚，因此把soc上的i_pad_jtg_trst_b置为1，引出i_pad_jtg_nrst_b到顶层。

（top上的trst_n名字没改，实际上是nrst_n，这个信号连到i_pad_jtg_nrst_b上）

```
module top(
    input   sys_clk_p,
    input   sys_clk_n,
    input   sys_rst_n,
    // jtag
    input   tck,
    input   trst_n,
    input   tms,
    input   tdi,
    output  tdo,
    // uart
    input   rx_i,
    output  tx_o
```

```
    );

wire sys_clk;
IBUFDS X_IBUFDS(
  .O(sys_clk),   // 1-bit output: Buffer output
  .I(sys_clk_p),   // 1-bit input: Diff_p buffer input (connect directly to top-level port)
  .IB(sys_clk_n)  // 1-bit input: Diff_n buffer input (connect directly to top-level port)
);

soc x_soc(
    .i_pad_clk        (sys_clk),
    .i_pad_rst_b      (sys_rst_n),
    .i_pad_jtg_nrst_b (trst_n),
    .i_pad_jtg_tclk   (tck),
    .i_pad_jtg_tdi    (tdi),
    .i_pad_jtg_tms    (tms),
    .i_pad_jtg_trst_b (1'b1),
    .o_pad_jtg_tdo    (tdo),
    .i_pad_uart_rx    (rx_i),
    .o_pad_uart_tx    (tx_o)
);

endmodule

```

3）添加约束

给时钟、复位、jtag和串口添加时序和物理约束

```
# ------------------------------Timing Constraints ---------------------------- #
create_clock -period 5.000 -name sys_clk_pin -waveform {0.000 2.500} -add [get_ports sys_clk_p]
create_clock -period 100.000 -name tck -waveform {0.000 50.000} [get_ports tck]
set_input_jitter tck 1.000
set_input_delay -clock tck -clock_fall 5.000 [get_ports tdi]
set_input_delay -clock tck -clock_fall 5.000 [get_ports tms]
set_output_delay -clock tck 5.000 [get_ports tdo]
set_false_path -from [get_ports trst_n]
# ------------------------------PIN Constraints ------------------------------ #
set_property PACKAGE_PIN K22 [get_ports sys_clk_p]
set_property PACKAGE_PIN K23 [get_ports sys_clk_n]
```

```
set_property IOSTANDARD DIFF_SSTL12 [get_ports sys_clk_p]
set_property IOSTANDARD DIFF_SSTL12 [get_ports sys_clk_n]

# Reset Constraints
set_property PACKAGE_PIN J14 [get_ports sys_rst_n]
set_property IOSTANDARD LVCMOS33 [get_ports sys_rst_n]


# JTAG Constraints
set_property PACKAGE_PIN D14 [get_ports tck]
set_property PACKAGE_PIN A13 [get_ports tms]
set_property PACKAGE_PIN G12 [get_ports tdi]
set_property PACKAGE_PIN E13 [get_ports tdo]
set_property PACKAGE_PIN C12 [get_ports trst_n]

# Set IOSTANDARD for JTAG
set_property IOSTANDARD LVCMOS33 [get_ports tck]
set_property IOSTANDARD LVCMOS33 [get_ports trst_n]
set_property IOSTANDARD LVCMOS33 [get_ports tms]
set_property IOSTANDARD LVCMOS33 [get_ports tdi]
set_property IOSTANDARD LVCMOS33 [get_ports tdo]

set_property KEEPER true [get_ports tms]
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets tck_IBUF_inst/O]

# UART Constraints
set_property PACKAGE_PIN AE15 [get_ports rx_i]
set_property PACKAGE_PIN AD15 [get_ports tx_o]

# Set IOSTANDARD for UART
set_property IOSTANDARD LVCMOS33 [get_ports rx_i]
set_property IOSTANDARD LVCMOS33 [get_ports tx_o]

```

4）初始化SRAM（可选）

如果没有CKLink, 可以通过固化SRAM的方式，将程序烧录到SRAM。

使用gen_hex.py将上一个笔记里得到的case.pat(在./smart_run/work目录下)转成hex文件

```
def gen_hex(input_file: str):
    output_file = input_file.split('.')[0] + ".hex"
    hex_lines = []
    fp = open(input_file)
    for line in fp:
        line = line.strip()
```

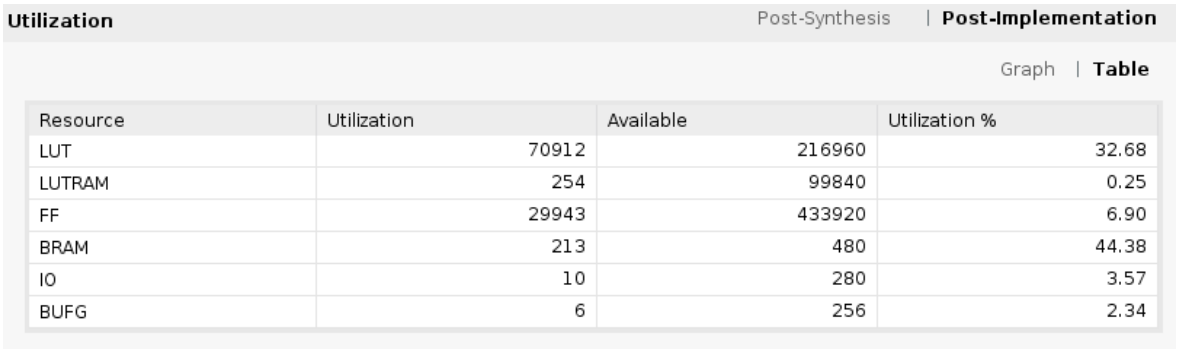
```
        line = line.split()
        addr = int(line[0][1:], 16)
        data = ''.join(line[1:])
        parts = [data[i:i + 2] for i in range(0, len(data), 2)]
        data = ''.join(parts[::-1])
        hex_lines.append(f"@{hex(addr//4)[2:].zfill(8)} {data}\n")
    fp.close()
    fp = open(output_file, 'w+')
    fp.writelines(hex_lines)
    fp.close()


if __name__ == "__main__":
    gen_hex('case.pat')
```

然后再在f_spsram_16384x128模块中添加$readmemh函数，对sram进行初始化

```
module f_spsram_16384x128 (
  A,
  CEN,
  CLK,
  D,
  Q,
  WEN
);

input                                   CLK;
input   [13:0]                          A;
)input                                  CEN;
input   [127:0]                         D;
input   [15:0]                          WEN;
output reg      [127:0]                 Q;

reg [127:0] mem [0:16384-1];
initial begin
        $readmemh("where is case.hex", mem);
end
integer i;

always @(posedge CLK) begin
        for (i=0; i<16; i=i+1) begin
                if (~CEN && ~WEN[i])
                        mem[A][i*8 +: 8] <= D[i*8 +: 8];
        end
end

```

```
30
31 always @(posedge CLK) begin
32         if (~CEN && (&WEN))
33                 Q <= mem[A];
34 end
35
   endmodule
```

5）资源消耗



Utilization Post-Synthesis | **Post-Implementation**

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	70912	216960	32.68
LUTRAM	254	99840	0.25
FF	29943	433920	6.90
BRAM	213	480	44.38
IO	10	280	3.57
BUFG	6	256	2.34

可以优化I-CACHE和D-CACHE的SRAM生成代码，把bit-mask改成byte-mask，提高bram的利用率，目前bit-mask的bram利用率只有1/64。

2. Hello World

1）方案一：固化SRAM

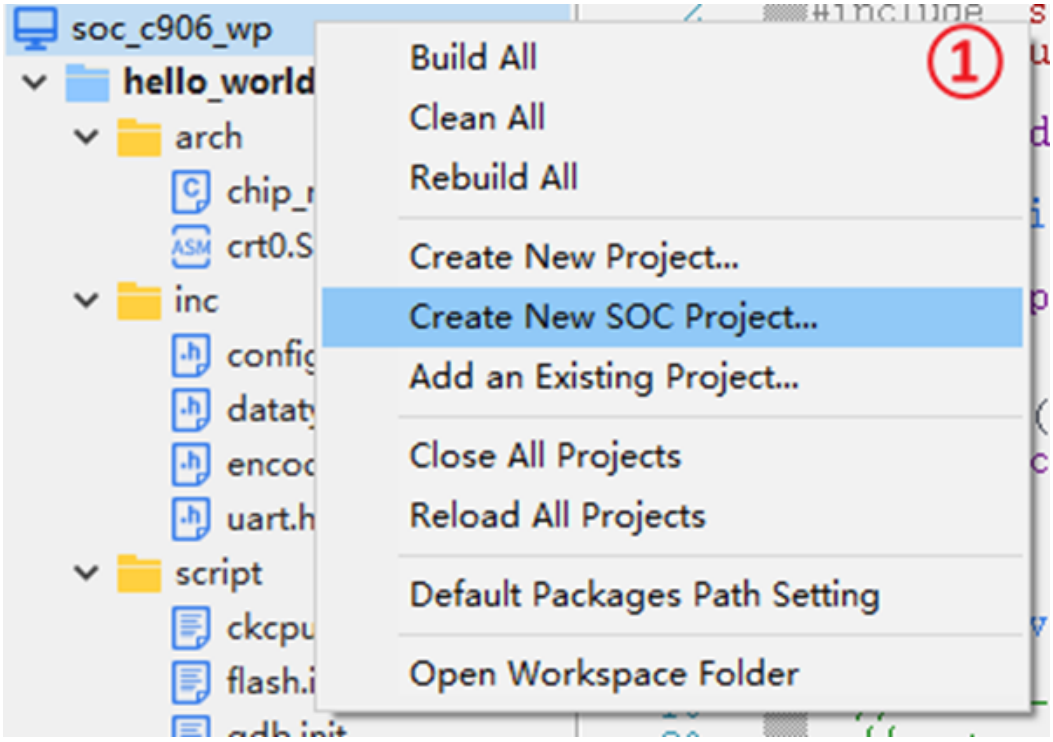
如果是使用hex文件初始化sram，那只要下载好bit流，然后连上串口，就会打印字符串了。

FPGA工程里的时钟为200MHz，之前C程序设置的时钟为100MHz，波特率为19200，所以要把串口工具的波特率设置为38400。

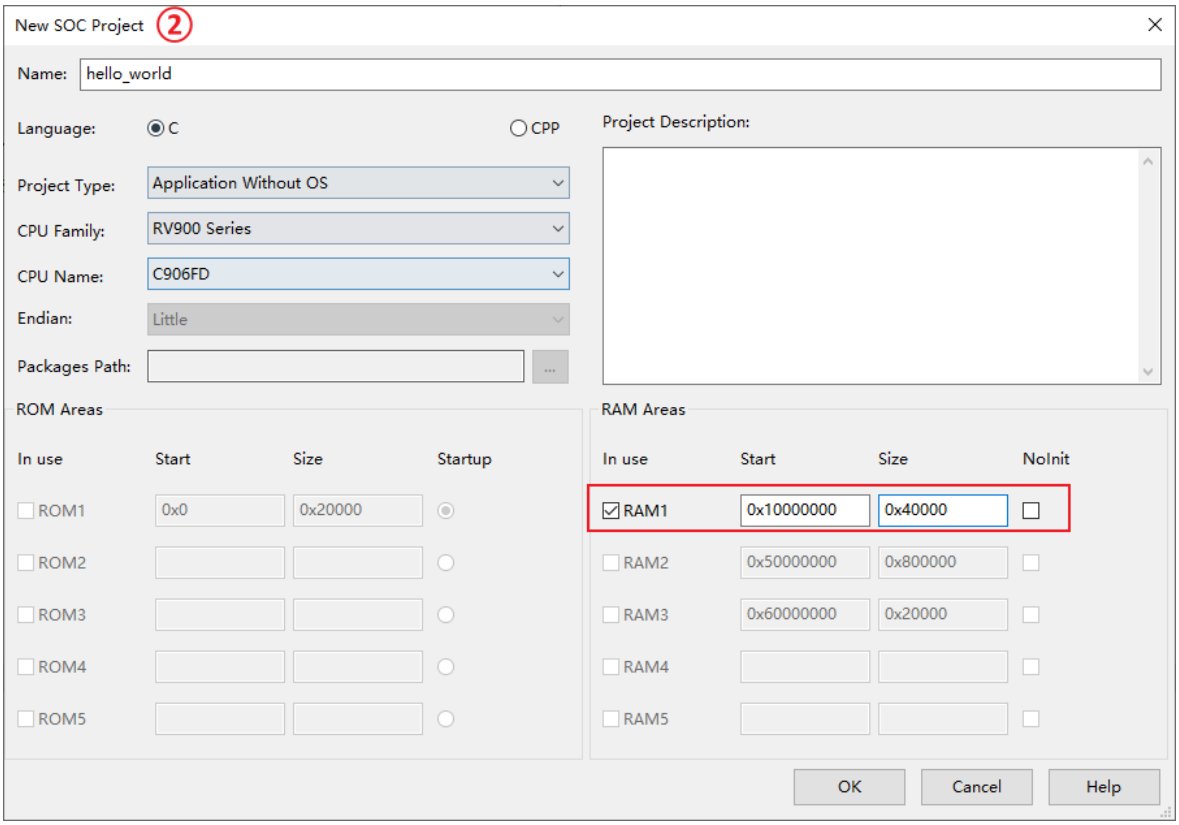
如果FPGA工程的时钟与C程序设置的时钟一致，就不需要调整串口工具的波特率。

2）方案二：CDK烧录程序

1.创建新工程

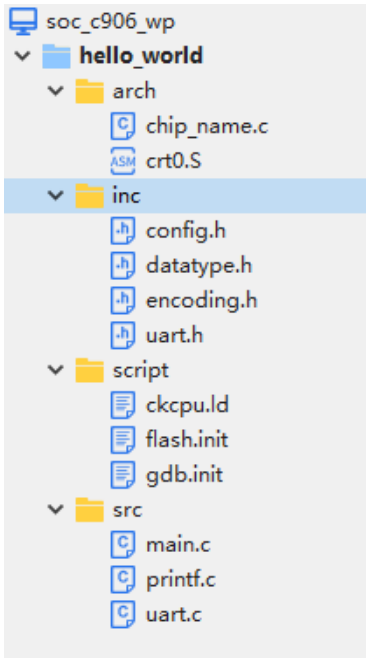


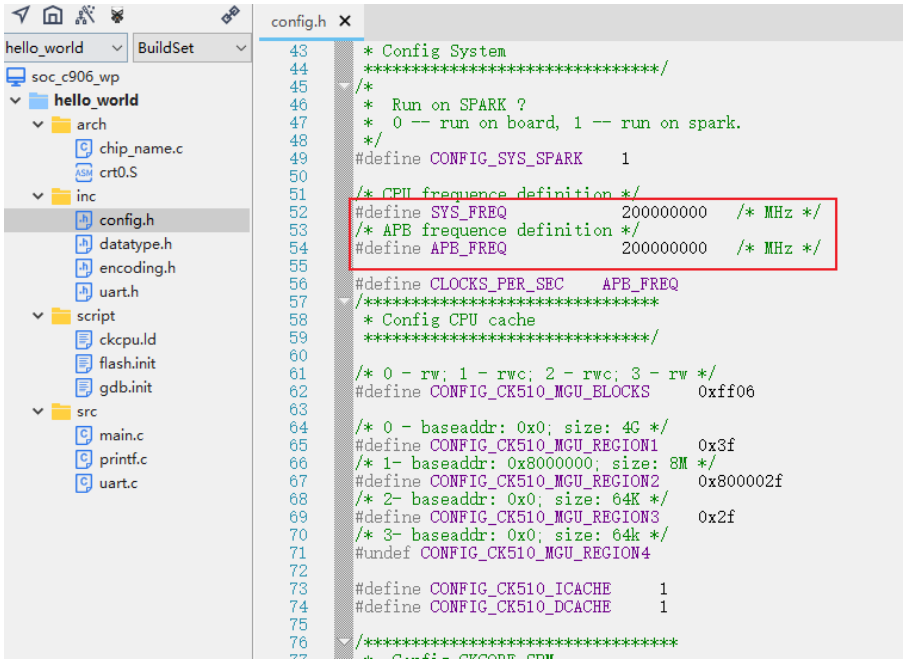
2.工程设置， RAM地址设置修改



3.添加代码

修改config.h的SYS频率和APB频率为200MHz



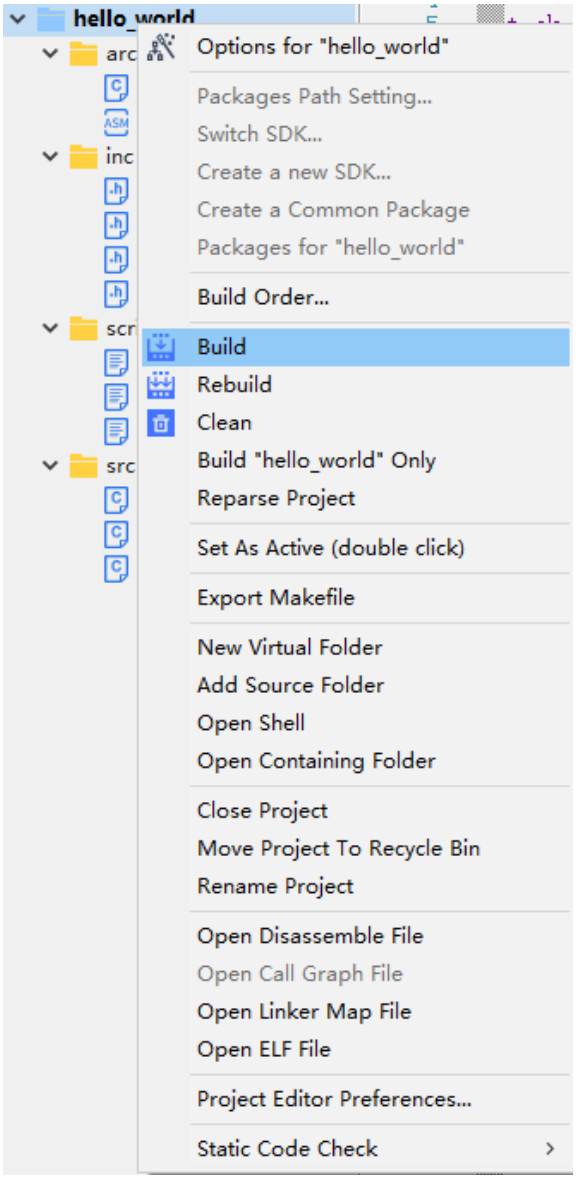


主程序main.c：

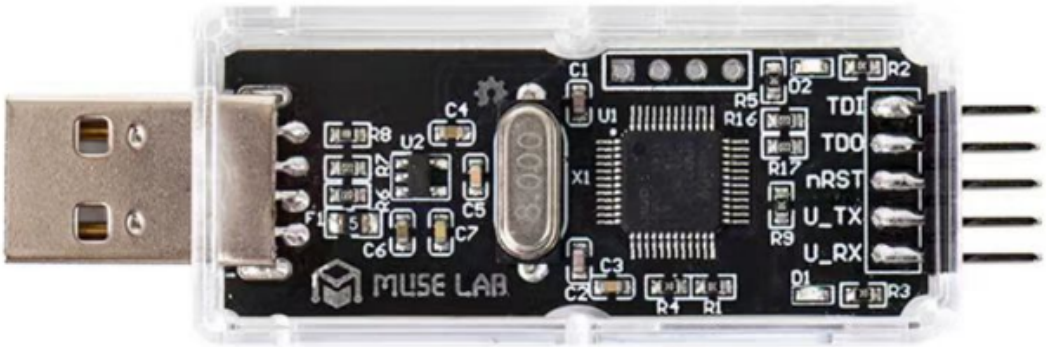
```c
#include "datatype.h"
#include "stdio.h"
#include "uart.h"

t_ck_uart_device uart0 = {0xFFFF};

int fputc(int ch, FILE *stream)
{
  ck_uart_putc(&uart0, (char)ch);
```

```
}

void delay(int count) {
    while(count--) {
    }
}

int main (void)
{
  //------------------------------------------------------------
  // setup uart
  //------------------------------------------------------------
  t_ck_uart_cfig   uart_cfig;

  uart_cfig.baudrate = BAUD;       // any integer value is allowed
  uart_cfig.parity = PARITY_NONE;     // PARITY_NONE / PARITY_ODD / PARITY_EVEN
  uart_cfig.stopbit = STOPBIT_1;      // STOPBIT_1 / STOPBIT_2
  uart_cfig.wordsize = WORDSIZE_8;    // from WORDSIZE_5 to WORDSIZE_8
  uart_cfig.txmode = ENABLE;          // ENABLE or DISABLE
  // open UART device with id = 0 (UART0)
  ck_uart_open(&uart0, 0);
  // initialize uart using uart_cfig structure
  ck_uart_init(&uart0, &uart_cfig);

  while(1) {
    printf("Hello C906!\n");
    delay(100000);
  }
  ck_uart_close(&uart0);
  return 0;
}
```

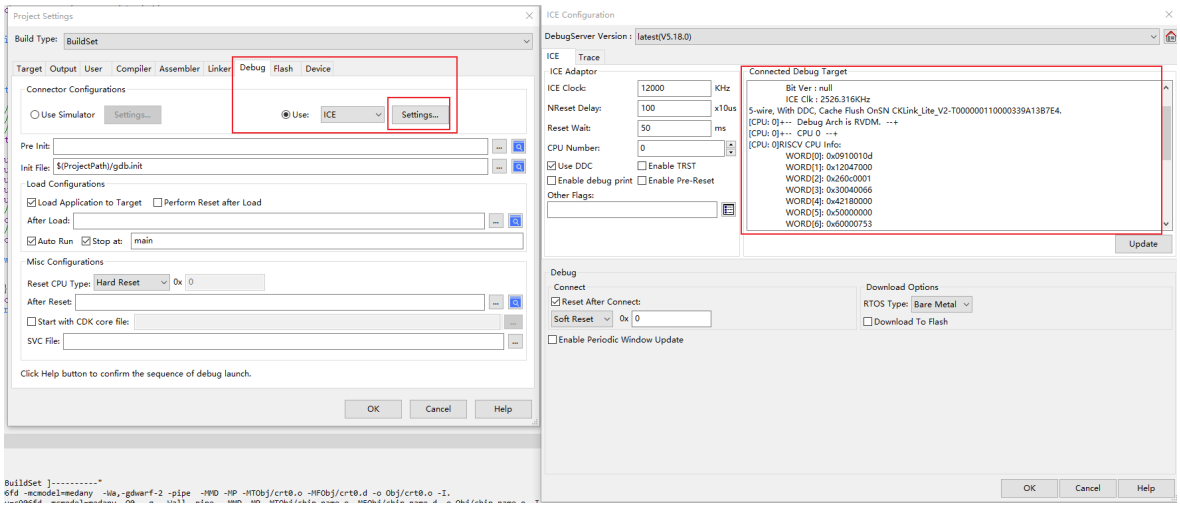
4.Build工程



5.FPGA下载好比特流，连上CKLink Lite下载器

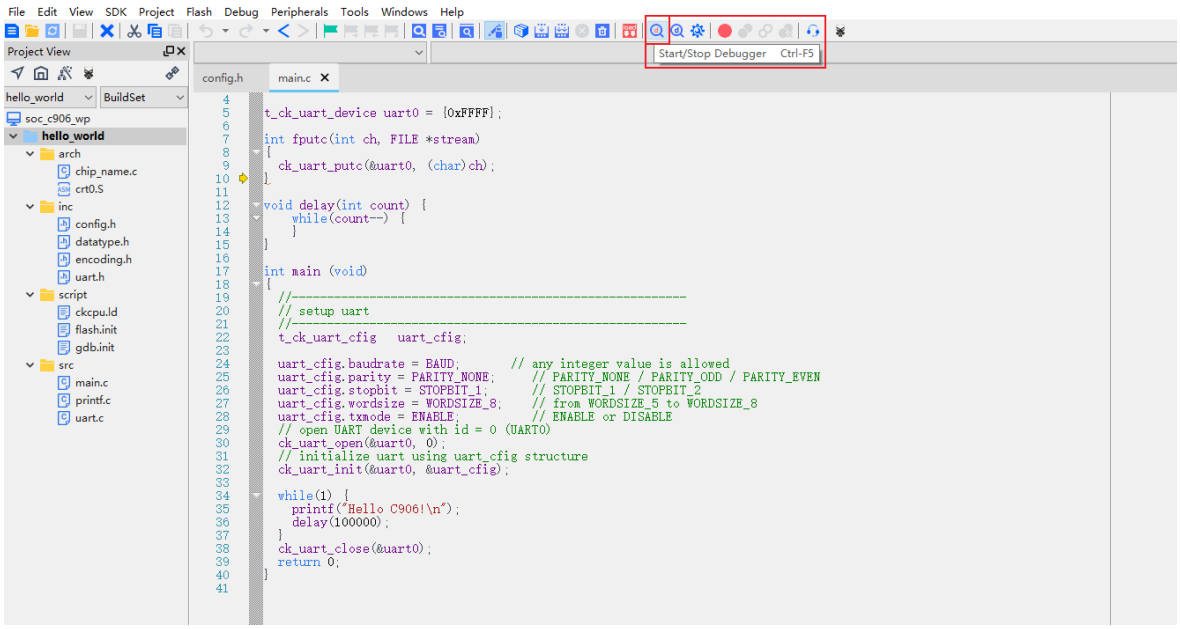


连上后Project Settings->Debug->ICE Settings->Connect Debug Target会显示CPU核信息。

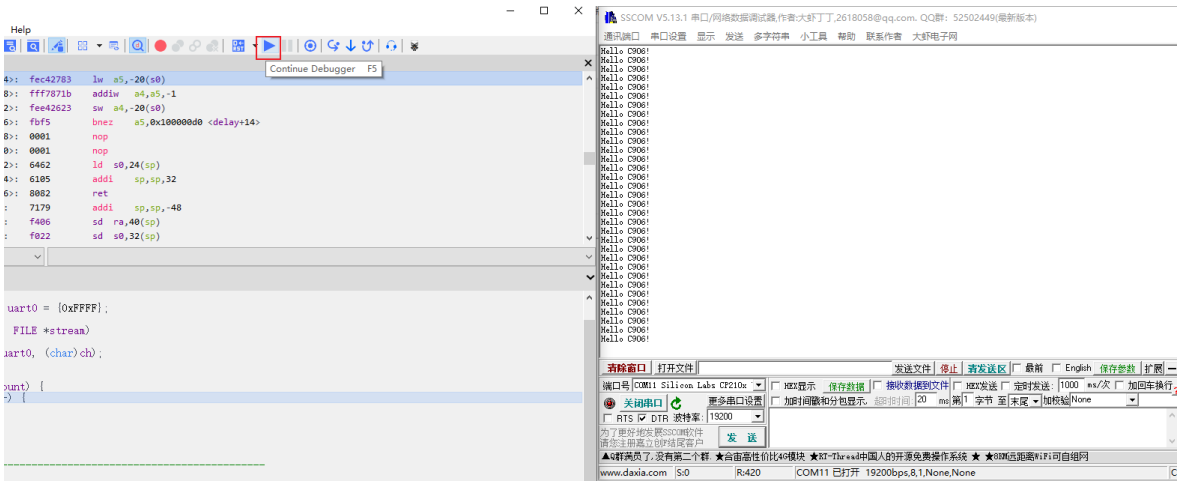


6.下载程序

点击Start/Stop Debugger按钮



点击Continue Debugger, 就能输出字符串了。


Help
Continue Debugger F5
fec42783 lw a5,-20(s0)
fff7871b addiw a4,a5,-1
fee42623 sw a4,-20(s0)
fbf5 bnez a5,0x100000d0 <delay+14>
0001 nop
0001 nop
6462 ld s0,24(sp)
6105 addi sp,sp,32
8082 ret
7179 addi sp,sp,-48
f406 sd ra,40(sp)
f022 sd s0,32(sp)
uart0 = (0xFFFF);
FILE *stream)
uart0, (char)ch);
SSCOM V5.13.1 串口/网络数据调试器,作者:大虾丁丁,2618058@qq.com, QQ群：52502449(最新版本)
通讯端口 串口设置 显示 发送 多字符串 小工具 帮助 联系作者 大虾电子网
Hello C906!
清除窗口 打开文件 发送文件 停止 清发送区 最前 English 保存参数 扩展
端口号 COM11 Silicon Labs CP210x HEX显示 保存数据 接收数据到文件 HEX发送 定时发送: 1000 ms/次 加回车换行
关闭串口 更多串口设置 加时间戳和分包显示, 超时时间 20 ms 第 1 字节 至 末尾 加校验 None
RTS DTR 波特率: 19200
为了更好地发展SSCOM软件 请您注册嘉立创P结尾客户
发 送
▲Q群满员了,没有第二个群 ★合宙高性价比4G模块 ★RT-Thread中国人的开源免费操作系统 ★ ★BDM远距离WiFi可自组网
www.daxia.com S:0 R:420 COM11 已打开 19200bps,8,1,None,None