处芯积律自研项目相关资料

前言:

一句话定位: 首款单核 RISC-V SoC 实践工程——Ibex + AHB/APB + ISP/DMA + 块级 UVM/FPV, 最适合 SoC 入门。
本项目包含 IP RTL、SoC 集成代码及相应验证环境。工程提供块级 UVM 基线与示例用例（ISP、QSPI）及 PINMUX 形式验证（FPV）参考流程; SoC 集成级测试用例与覆盖率需学员在基线上自行扩展。
EDA 仿真工具（VCS、Verdi 等）及 RISC-V C 编译工具链需学员自行安装或在处芯积律 VNC 实践环境中使用。线上入口: www.yueanic.com（IC 实践 → VNC 远程仿真实践 → 基础版 SoC）。

用途	路径
VNC 只读基线（学员环境）	/project/SOC_V1.1/To_Customer
建议个人工作副本	cp -a /project/SOC_V1.1/To_Customer ~/work/SOC_V1.1

RISC-V 工具链

需安装 **rv32imc** 工具链（与 Ibex 配置及 sw/common/Makefile 中 ARCH = rv32imc 一致）:

https://github.com/lowRISC/lowrisc-toolchains/releases/download/20210412-1/lowrisc-toolchain-gcc-rv32imc-20210412-1.tar.xz

常见问题: 若生成 .vmem / .srec 报错, 可安装 srecord:
sudo apt-get install srecord
并在 soc/sw/common/Makefile 中设置 RISCV_CC 指向 riscv32-unknown-elf-gcc 完整路径

SoC 架构与功能列表

本 SoC 是面向 RISC-V 爱好者的单核精简 SoC 教学工程，涵盖处理器、总线互连、存储、图像处理与外设，并配套块级 UVM 与 FPV 验证基线。

功能列表

类别	内容
CPU	lowRISC Ibex，32 位 RISC-V，RV32IMC（I + C + M），2 级流水线，M-Mode / U-Mode
总线	core2ahb 桥 + ARM CMSDK 风格 AHB 总线矩阵（BusMatrix）+ AHB→APB 桥

主存	片上 SRAM，约 512 KB（NUM_WORDS = 16384×8，32 bit 宽）
系统时钟	50 MHz（clk_cpu，testbench 中 #(5) 翻转）
外设	双路 QSPI（spi0/spi1）、UART、双路 I2C（i2c0/i2c1）
图像子系统	ISP（Bayer RAW → RGB，demosaic / dgain）+ DMA（ISP↔SRAM 数据搬运）
IO	PINMUX + PAD，顶层 pad[24:1]
验证	ISP UVM、QSPI UVM、PINMUX FPV、SoC 顶层简易 TB

一 IP 介绍

1. QSPI

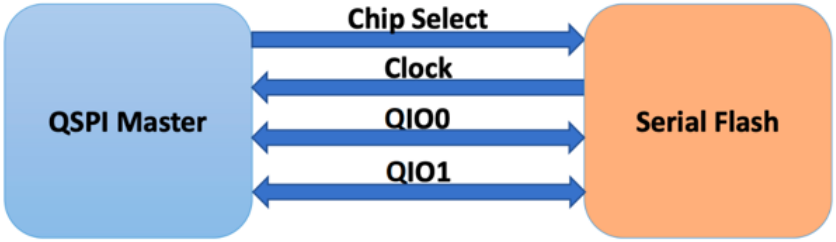
QSPI 是 Quad SPI 的简写，表示 6 线 spi，是 Motorola 公司推出的 SPI 接口的扩展，比 SPI 应用更加广泛。

Queued SPI 相对于 SPI 来说增加了两根 I/O 线（SIO2、SIO3），提供了每次传输的 bit 数。QSPI 可以的工作模式 Single-Bit SPI、Dual SPI、Quad SPI。

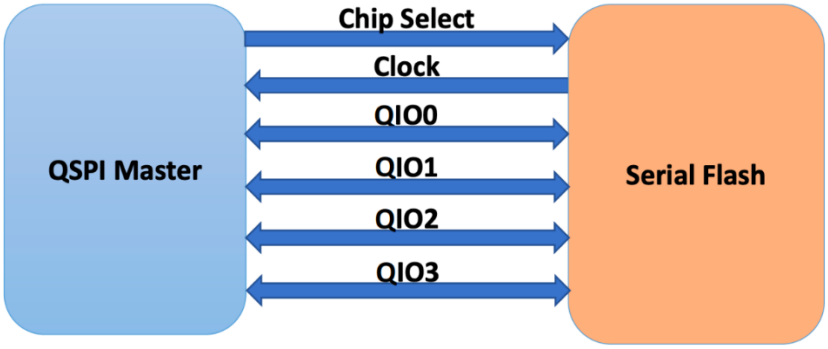
QSPI 的传输模式

QSPI 单线传输模式

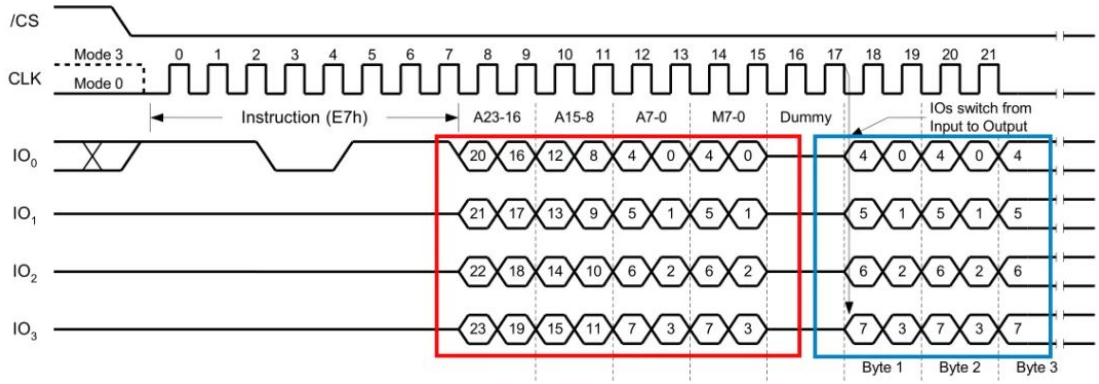
QSPI 双线传输模式



QSPI 四线传输模式



QSPI 时序图



QSPI 通过命令与 Flash 通信，每条命令包括指令、地址、交替(复用)字节、空指令和数据共五个阶段，而这五个阶段任一阶段均可跳过，但至少要包含指令、地址、交替字节或数据阶段之一。nCS 在每条指令开始前下降，在每条指令完成后再次上升。

具体的 QSPI 协议可以参考
Quad-SPI (QSPI) interface on STM32 microcontrollers

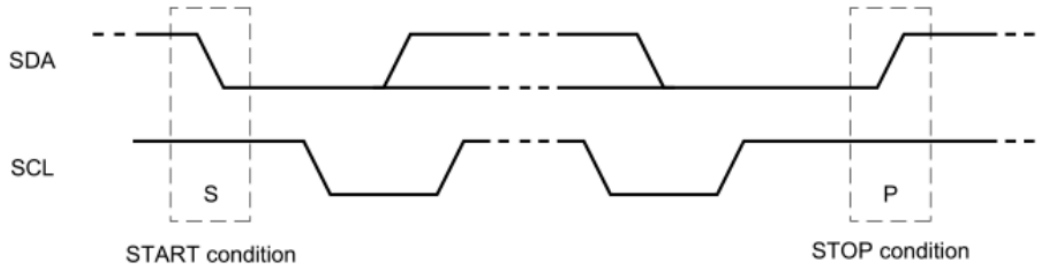
2. I2C

I2C 协议比较简单，本项目 I2C 的配置参见寄存器列表。
注意
PRE = Freq_I2C / ((5*Freq_SCL) - 1)
Freq_I2C is the clock frequency of I2C module.

START and STOP

START and STOP conditions由主机产生，具体参加下图：

START：SCL为高期间，SDA由高->低；
STOP ：SCL为高期间，SDA由低->高；



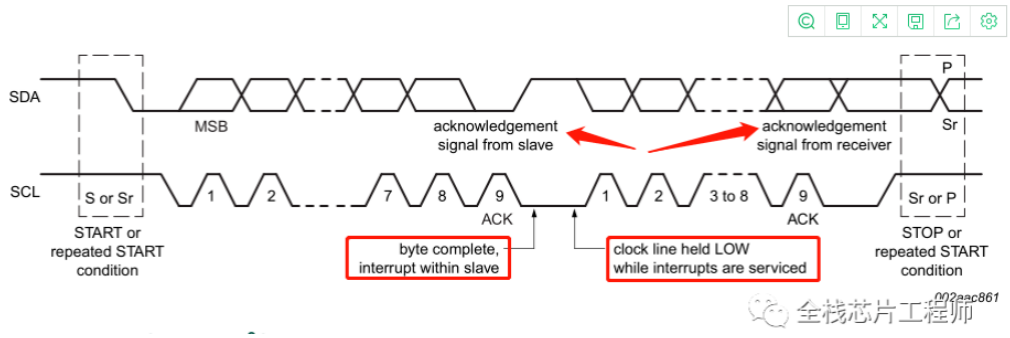
START and STOP conditions

Byte传输、ACK应答

信号传输以Byte为单位，每个字节后都由1bit的应答信号。

MSB先发，LSB最后（I2C/SPI都是MSB先发，而UART是LSB先发）。

当Slave正处理其内部中断而无法接收I2C Master数据时，Slave可拉低SCL以使得Master进入等待状态。



I2C写时序

向指定寄存器地址写入指定数据操作时序：

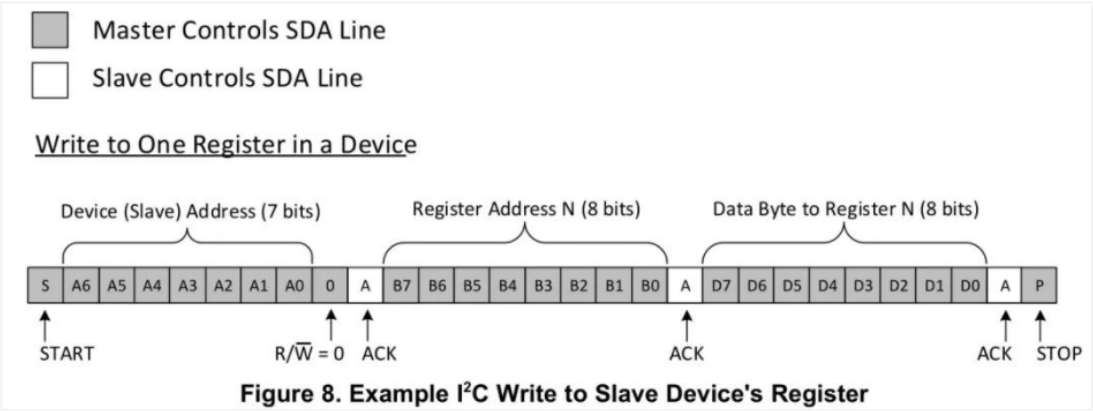


Figure 8. Example I²C Write to Slave Device's Register

3. UART

通用异步收发器（Universal Asynchronous Receiver/Transmitter），通常称作 UART，是一种串行、异步、全双工的通信协议，在嵌入式领域应用广泛。

UART 时序图如下

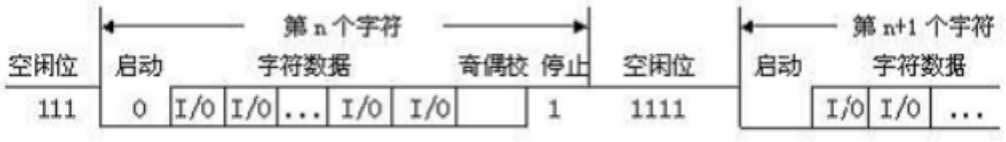


图 1 UART 工作原理

起始位：先发出一个逻辑“0”的信号，表示传输字符的开始。

数据位：紧跟起始位之后。数据位的个数可以是 4、5、6、7、8 等，构成一个字符。通常采用 ASCII 码。从最低位开始传送，靠时钟 来定位。

奇偶校验位：数据位加上这一位后（跟在数据位尾部），使得“1” 的位数应为偶数(偶校验)或奇数(奇校验)，以此来校验数据传送的正 确性。

停止位：它是一个字符数据的结束标志。可以是 1 位、1.5 位、 2 位的高电平（逻辑“1”）。

空闲位：处于逻辑“1”状态，表示当前线路上没有数据的传送。

波特率：

数据传输速率使用波特率来表示。单位 bps（bits per second），常见的波特率 9600bps、19200bps、115200bps 等等，如果串口波特率设置为 9600bps，那么传输一个比特需要的时间是 1/9600≈104.2us，即 1bit 传输时间大约为 104us，传送一个数据实际是 10 个比特（开始位-8 个数据位-停止位），传输速率实际为 9600*8/10=7680bps。

串口的具体配置参见寄存器列表，注意配置波特率时，需要 UART_LCR[7]先配置为 1 才能生效。

二 SOC 介绍

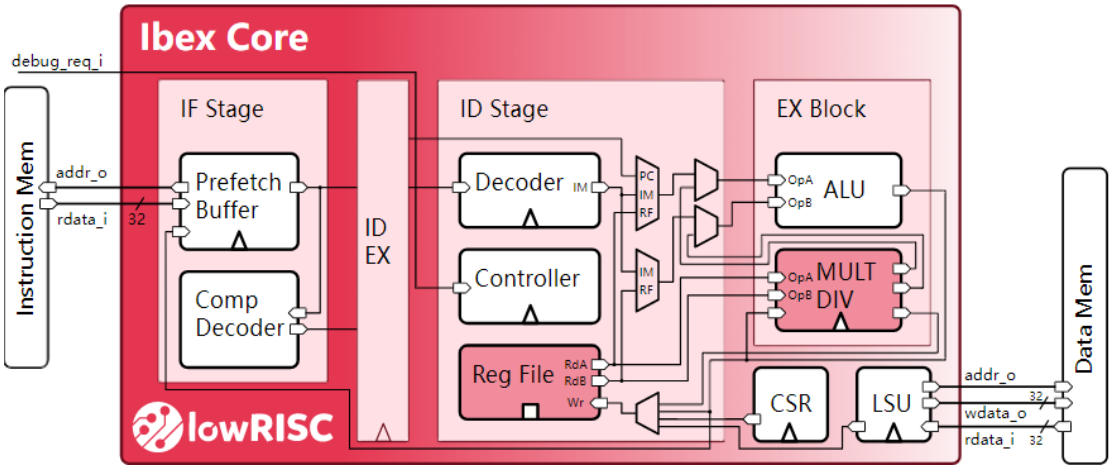
1、功能列表

本 SoC 设计是最精简的 RISC-V 处理器 SoC，包含最基本的 AMBA 总线、外设，满足 RISC-V 处理器爱好者的高效学习。

- CPU：RISC-V
- 总线：AMBA2.0,AHB/APB
- 外设：UART/I2C/QSPI
- 系统时钟：50MHz
- 主存：128KB

2. CPU 核介绍：

ibex 是一款 32 位开源 RISC-V 处理器，2 级流水，支持 RV32I、RV32C、RV32M、RV32B 等指令，支持 M-Mode 和 U-Mode。



Ibex 相关的资料介绍：

Ibex Reference Guide — Ibex Documentation 0.1.dev76+g056cb44.d20220830 documentation (ibex-core.readthedocs.io)

Ibex github 库：

ibex/vendor/lowrisc_ip/dv/sv/mem_model at master · lowRISC/ibex · GitHub

工具链信息：

https://github.com/lowRISC/lowrisc-toolchains/releases/download/20210412-1/lowrisc-toolchain-gcc-rv32imc-20210412-1.tar.xz

3、SOC 架构框图

本 SoC 设计包含 RISC-V 处理器、AHB/APB 总线、SRAM，外设包括 QSPI、UART、I2C，DMA，ISP，PINMUX，整体系统框架如下图。

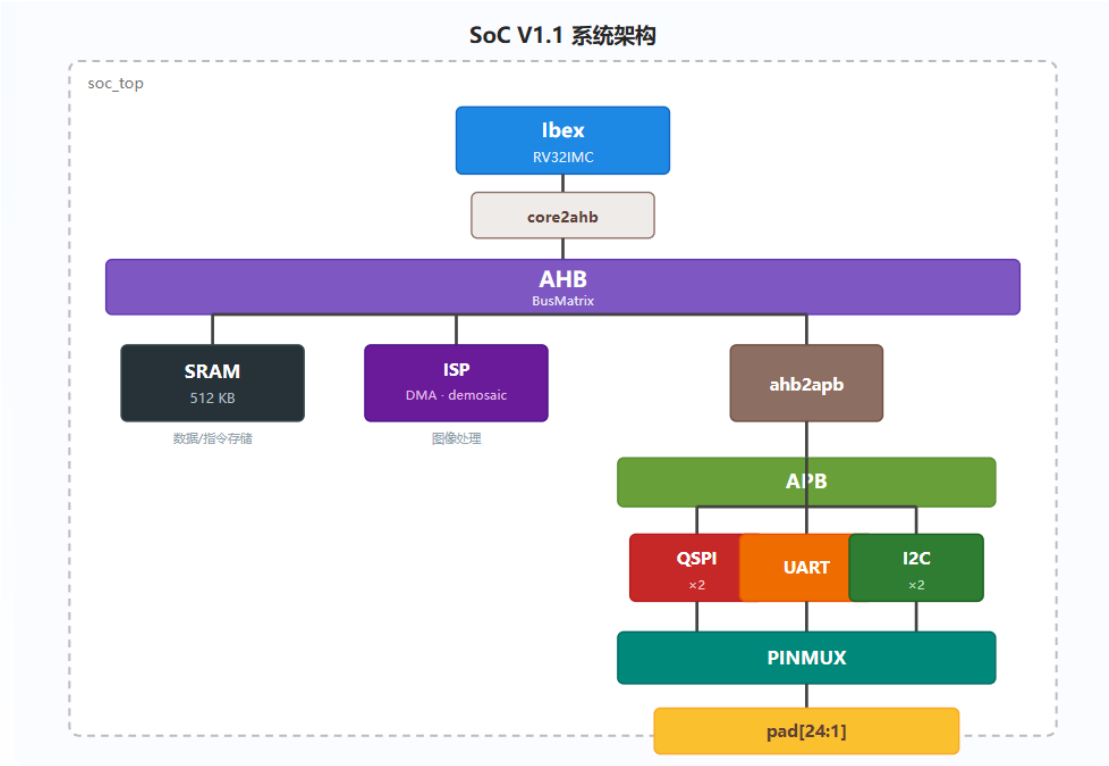


图 2 · SoC V1.1 硬件架构（层级总线视图，对齐 soc/top/top.v）

4、顶层接口信号说明

详细描述模块的顶层信号，包含信号名、位宽、输入输出属性、详细说明等。

Signal	Width	I/O	Description
SoC Interface			
hclk_100	1	I	AHB clock
hreset_n	1	I	AHB reset signal Active state:Low
UART_TX	1	O	
UART_RX	1	O	
SPI_CLK	1	O	SPI 时钟
SPI_CSN0	1	O	
SPI_CSN1	1	O	
SPI_CSN2	1	O	

SPI_CSN3	1	0	
SPI_SDO0	1	0	
SPI_SDO1	1	0	
SPI_SDO2	1	0	
SPI_SDO3	1	0	
SPI_SDI0	1	I	
SPI_SDI1	1	I	
SPI_SDI2	1	I	
SPI_SDI3	1	I	
SCL_PAD_I	1	I	
SCL_PAD_O	1	0	
SCL_PADOEN_O	1	0	
SDA_PAD_I	1	I	
SDA_PAD_O	1	0	
SDA_PADOEN_O	1	0	

5、各模块信号说明及设计细节

各个模块详细介绍见上述 IP 部分。

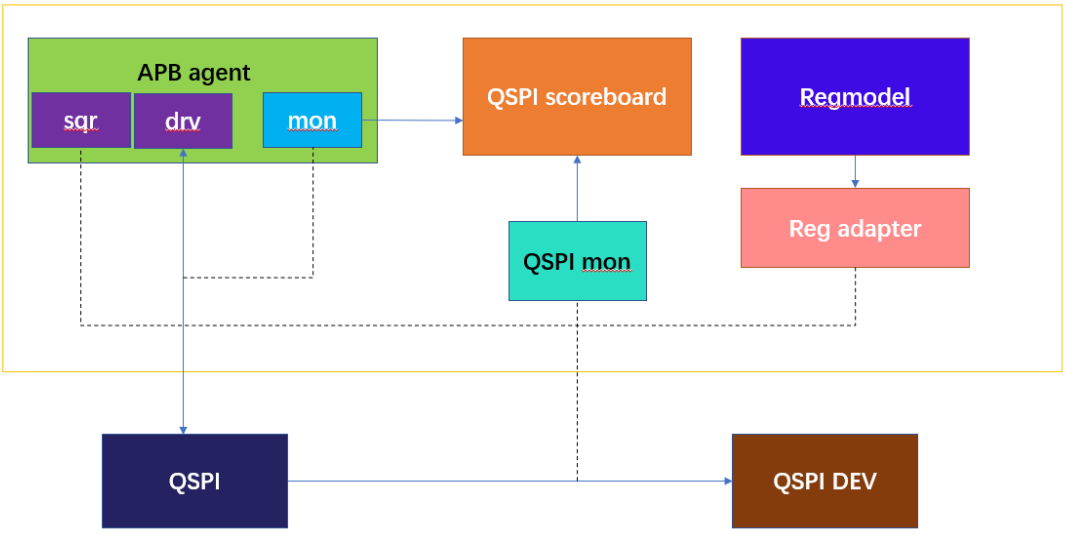
6、寄存器描述

参见 SoC 寄存器列表。

三. 验证环境

1. IP 验证环境

QSPI 的验证环境



➢ 文件夹介绍：

QSPI/tb/env 各种 QSPI testbench 的 组件

QSPI/tb/tests 测试用例

QSPI/sim 跑 simulation 的目录

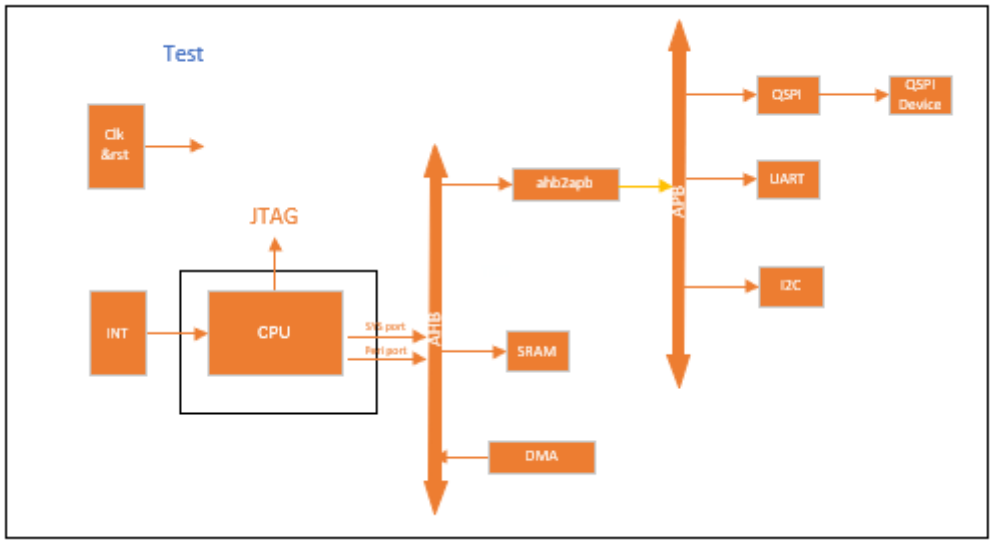
➢命令：

编译命令 make comp
运行指令 make run_{test_name}
Example : make run_qspi_read_test

大家根据自己需求更改 makefile 文件

2. SOC 验证环境

仿真环境框图



➢ 文件夹介绍：

soc/filelist ： 环境的 filelist
soc/ibex ： ibex cpu 核
soc/ip ： 外设及总线信息，其中外设全部代码可见，总线目前是加密的。
soc/model: sram model
soc/sw ： 放置各个模块测试用的 C pattern
soc/test: 挂载 qspi device 等 vip
soc/top: soc 顶层文件，例化各个模块。

➢命令：

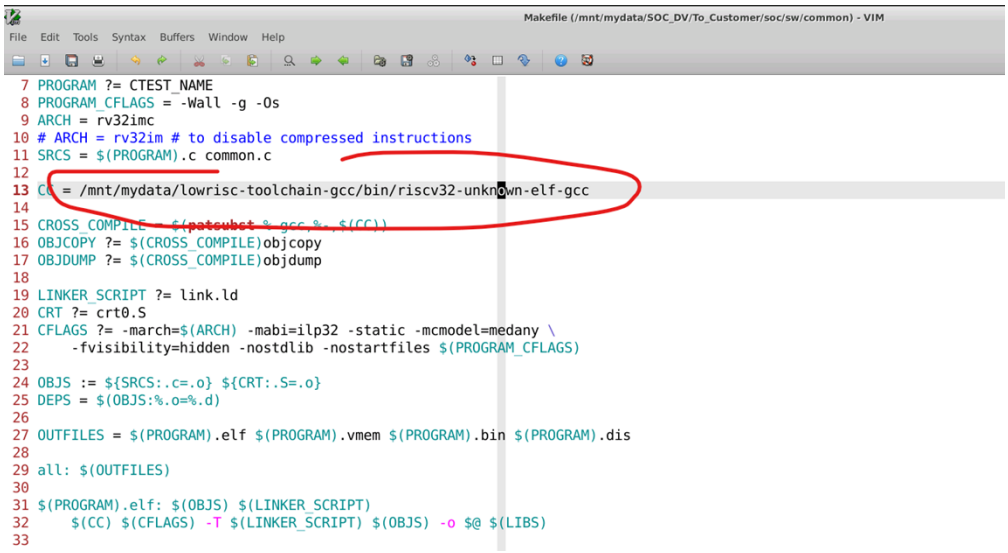
(1) 编译 test 环境 ： make comp
(2) 单独 C compile + run RTL simulation ： make run_{test_name}

注意：

1. Sw 下面的 test_name 要和 c 的 name 一致，比如 spi_test 那么 spi_test 目录下的 c 的名字叫 spi_test.c
2. 运行 soc 的 toolchain 需要自己下载。地址在

https://github.com/lowRISC/lowrisc-toolchains/releases/download/20210412-1/lowrisc-toolchain-gcc-rv32imc-20210412-1.tar.xz

安装完修改 C test 路径下的 sw/common/makefile ，如



修改图中红色圈起来的地方即可。