

SPI-CHIP SPECIFICATION & REPORT

作者: Drh、Chy、Gez

时间: 2023.12

0. 简介

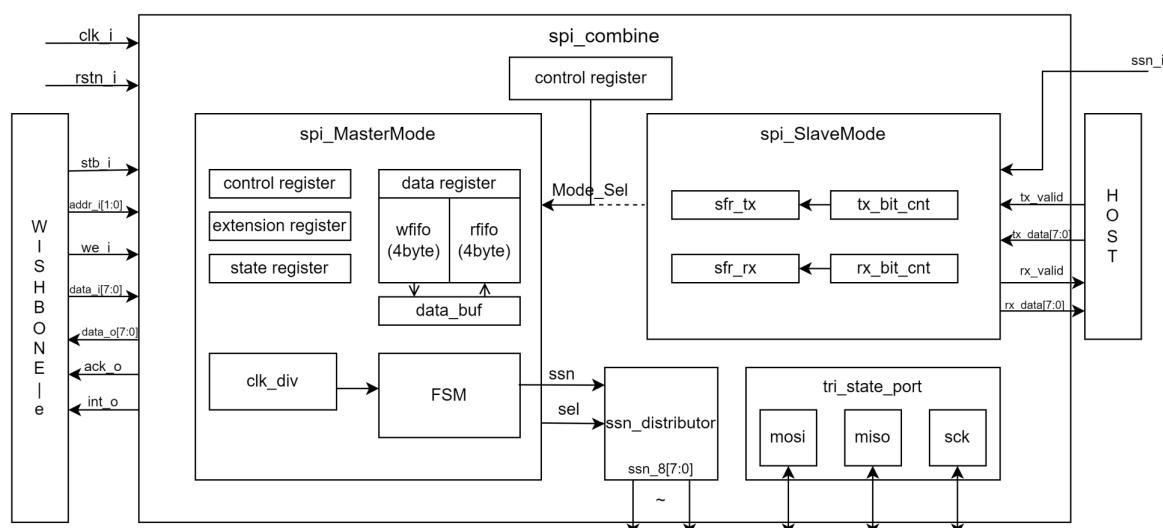
SPI (Serial Peripheral Interface) 是一种通信协议，用于在数字设备之间进行全双工通信。它通常用于连接微控制器和外围设备，如存储器芯片、传感器、显示屏等。SPI通信协议使用四条信号线进行通信：时钟线（SCLK）、主设备输出线（MOSI）、主设备输入线（MISO）和片选线（SS）。在SPI通信中，主设备控制通信的时序和传输数据，并在传输完成后，将数据传输给从设备。SPI通信协议具有简单、高速、全双工通信的特点，适合于短距离、高速传输的应用。

这是一款独立的SPI通信芯片，在Richard Herveille先生的 *Simple SPI Core* (https://opencores.org/projects/simple_spi) 项目的基础上有了很大程度的提高和完善。本芯片可以作为SPI通信的主机或从机使用，即拥有主模式和从模式的选择，并且在主模式和从模式下均有四种工作模式和多种波特率的选项。使用时，通过Wishbone总线配置芯片的工作状态。具体有如下特点：

- 主模式或从模式
- 全双工通信
- 深度为4bytes的FIFO作为数据寄存器
- 简化的Wishbone总线
- 可控的SCK时钟频率与时钟相位
- 多种传输频率可选，最高达到系统时钟的四分之一
- 多种中断模式可选
- 8位从机选择输出，主模式下可连接8个从机

1. 芯片详解

1.1 整体框图



本设计框图如上所示，顶层模块为spi_combine，分为两大模块：spi_MasterMode和spi_SlaveMode，分别为主模式和从模式。除此之外，还有数据分配器ssn_distributor模块，用于将ssn信号输出到某一个从机，实现从机8选1的操作。最后还有tri_state_port模块为三态输入输出端口：在主模式下mosi和sck作为输出，miso作为输入；在从模式下mosi和sck作为输入，miso作为输出。

1.2 各模块详解

1.2.1 顶层模块 spi_combine

I. I/O端口

Port	Width	Direction	Description
clk_i	1	input	系统时钟信号
rstn_i	1	input	系统异步复位信号（同步释放）
SPI端口：			
sck	1	inout	SPI通信时钟
ssn_i	1	input	输入片选信号（从模式）
ssn_o	8	output	输出片选信号（主模式）
mosi	1	inout	SPI数据线-主出从入
miso	1	inout	SPI数据线-主入从出
主模式：			
stb_i	1	input	类Wishbone总线对SPI芯片的片选使能信号
addr_i	2	input	地址选择对芯片内不同寄存器读写
we_i	1	input	类Wishbone总线对芯片的写使能
data_i	8	input	数据输入
data_o	8	output	数据输出
ack_o	1	output	芯片对总线的应答信号
int_o	1	output	芯片对总线的中断信号
从模式：			
tx_valid_i	1	input	发送数据有效信号
tx_data_i	8	input	发送的数据
rx_valid_o	1	output	接收到的数据有效信号
rx_data_o	8	output	接收的数据

II. 设计特点

①异步复位同步释放

系统复位信号使用 rstn_sync，具体处理如下：

```
//Rstn_sync - Asynchronous reset and synchronous release
always @(posedge clk_i or negedge rstn_i) begin
    if (!rstn_i) begin
        rstn_sync <= 1'b0;
        rstn_temp <= 1'b0;
    end
    else begin
        rstn_temp <= 1'b1;
        rstn_sync <= rstn_temp;
    end
end
```

②控制寄存器配置

芯片工作在主模式还是从模式下需要控制寄存器来控制，因此选择在顶层模块写控制寄存器spi_con：

```
//wishbone signals
assign wb_acc = stb_i ;
assign wb_wr = wb_acc & we_i ;

//-----Control register config-----//
always @(posedge clk_i or negedge rstn_sync) begin
    if (!rstn_sync) begin
        spi_con <= 8'b00100000;
    end
    else if (wb_wr && addr_i == 2'b00) begin
        spi_con <= data_i ;
    end
end

assign spi_en = spi_con[7] ; //SPI enable bit
assign mode_sel = spi_con[5] ; //Select master or slave mode
assign cpol = spi_con[4] ; //Clock polarity bit
assign cpha = spi_con[3] ; //Clock phase bit

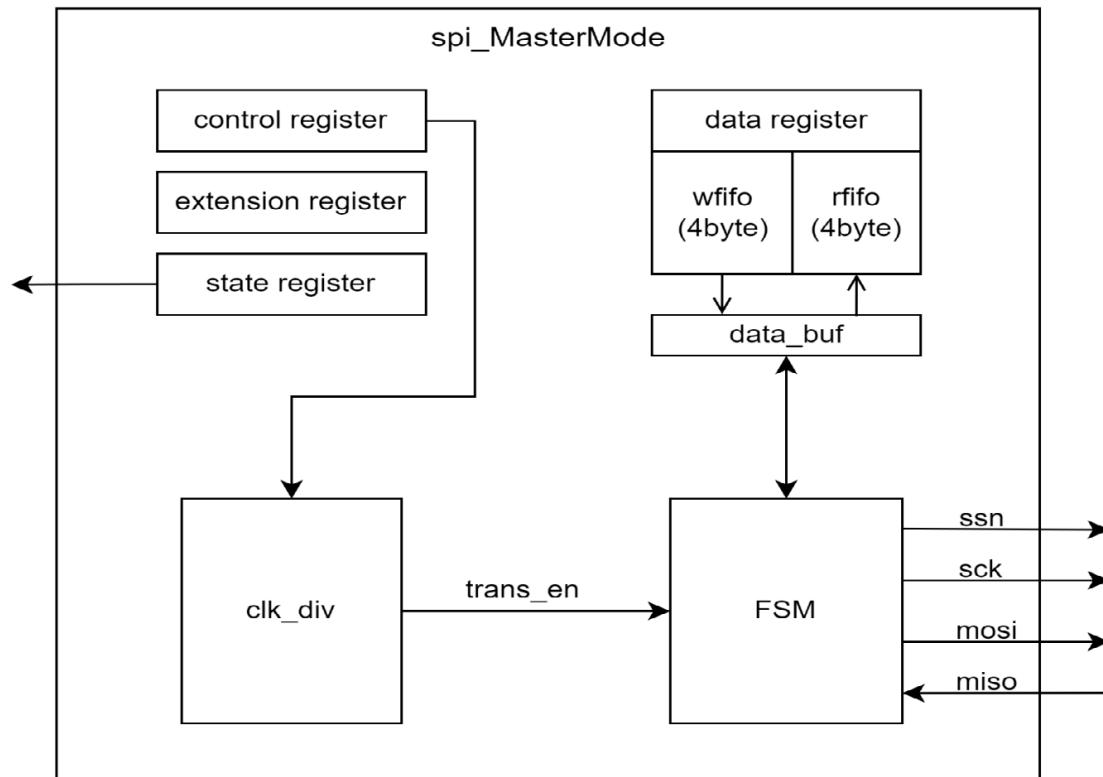
//-----Mode selection-----//
assign master_mode_en = mode_sel && spi_en ;
assign slave_mode_en = ~ mode_sel && spi_en ;
```

1.2.2 主模式模块 spi_MasterMode

I. I/O 端口

Port	Width	Direction	Description
clk_i	1	input	系统时钟信号
rstn_i	1	input	系统异步复位信号 (同步释放)
master_mode	1	input	主模式选择信号
stb_i	1	input	类Wishbone总线对SPI芯片的片选使能信号
addr_i	2	input	地址选择对芯片内不同寄存器读写
we_i	1	input	类Wishbone总线对芯片的写使能
data_i	8	input	数据输入
spi_con_i	8	input	数据寄存器
data_o	8	output	数据输出
ack_o	1	output	芯片对总线的应答信号
int_o	1	output	芯片对总线的中断信号
SPI端口:			
sck_o	1	output	SPI时钟输出
ssn_o_8	8	output	8位片选输出
mosi_o	1	output	SPI数据线-主出从入
miso_i	1	input	SPI数据线-主入从出

II. 原理框图



III. 设计特点

①控制寄存器与拓展寄存器

主模式下，复位结束后，首先配置控制寄存器，再配置拓展寄存器。每位的具体功能再稍后详细介绍

②状态寄存器

状态寄存器最高位为中断标志信号，次高位为写冲突标志信号。当标志被置“1”时，需要对状态机的对应位写入“1”，即可消除标志信号。

```
assign    wr_spi_sta = wb_wr & (addr_i == 2'b01) ;
assign    int_rq     = ! trans_cnt & rfifo_wen      ;

//Set interrupt flag
always @(posedge clk_i or negedge rstn_i) begin
    if(!rstn_i) begin
        spi_int_f <= 1'b0;
    end
    else if(master_ena) begin
        if(wr_spi_sta && data_i[7]) begin
            spi_int_f <= 1'b0;
        end
        else begin
            spi_int_f <= spi_int_f | int_rq;
        end
    end
end

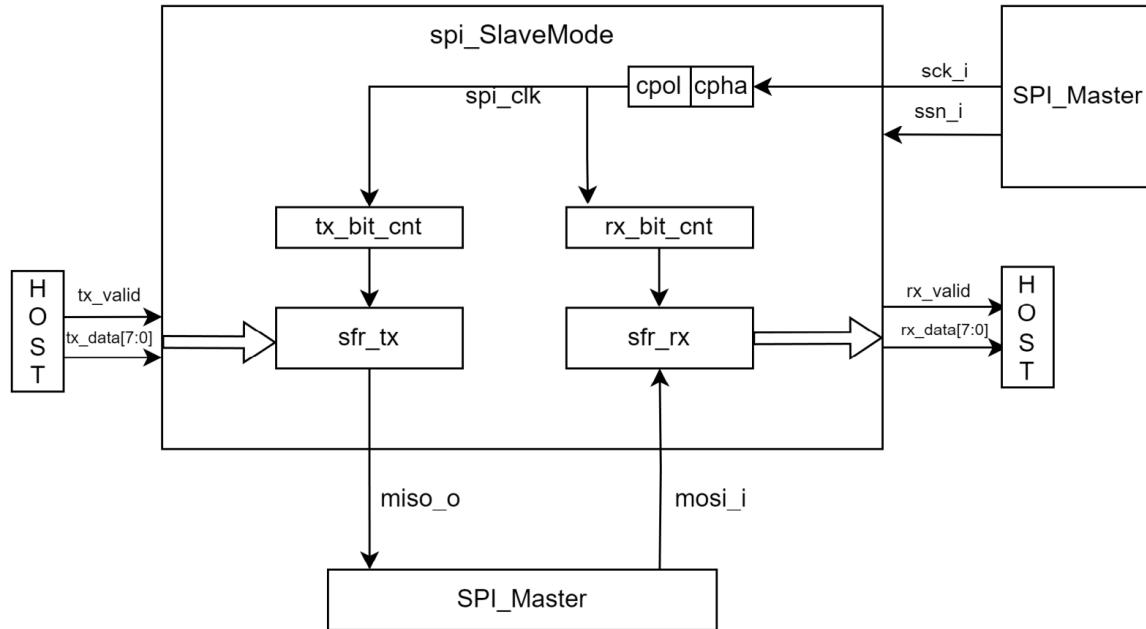
//Set write collision flag
always @(posedge clk_i or negedge rstn_i) begin
    if(!rstn_i) begin
        wr_col_f <= 1'b0;
    end
    else if(master_ena) begin
        if(wr_spi_sta && data_i[6]) begin
            wr_col_f <= 1'b0;
        end
        else begin
            wr_col_f <= wr_col_f | wfifo_ov;
        end
    end
end
end
```

1.2.3 从模式模块 spi_SlaveMode

I. I/O端口

Port	Width	Direction	Description
sys_clk_i	1	input	系统时钟信号
sys_rstn_i	1	input	系统异步复位信号 (同步释放)
slave_mode	1	input	从模式选择信号
tx_valid	1	input	发送数据有效信号
tx_data	8	input	发送数据
rx_valid_d1	1	output	接收数据有效信号
rx_data	8	output	接收数据
cpol	1	input	时钟极性
cpha	1	input	时钟相位
SPI端口：			
sck_i	1	input	SPI时钟输入
ssn_i	1	input	SPI片选输入
mosi_i	1	input	SPI数据线-主出从入
miso_o	1	output	SPI数据线-主入从出

II. 原理框图



III. 设计思路

从模式下，需要一个控制器来提供需要发送给主机的数据，并取出从主机收到的数据，在芯片框图中命名为了HOST。HOST与SPI_Master完全无关。

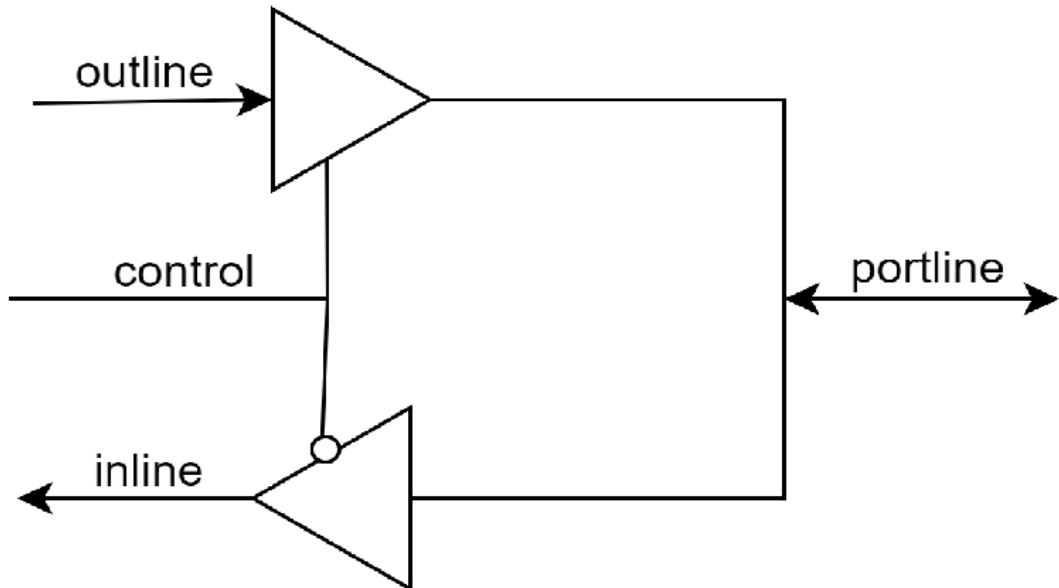
①异步时钟域信号

表示收到8位数据的标志信号rx_valid为SPI时钟域信号，在系统时钟域采样时进行打两拍操作，以避免亚稳态。

```
//Receive data config           ***sending received data to host uses
sys_clk***  
always @(posedge sys_clk_i or negedge sys_rstn_i) begin  
    if (!sys_rstn_i) begin  
        rx_valid_d0 <= 1'b0;  
        rx_valid_d1 <= 1'b0;  
        rx_data      <= 8'h00;  
    end  
    else if (slave_ena)begin  
        rx_valid_d0 <= rx_valid;  
        rx_valid_d1 <= rx_valid_d0;  
        if (rx_valid_d0 == 1'b1 && rx_valid_d1 == 1'b0) begin //catch rising  
edge of rx_valid  
            rx_data <= sfr_rx;  
        end  
    end  
end
```

1.2.4 三态输出端口 inout_port

由于本芯片即可做主机也可做从机，因此sck, mosi, miso端口均需要复用，因此设计一个三态输出端口，来切换端口为输入或输出：



1.3 寄存器简介

Name	Address	Width	Access	Description
spi_con	2'b00	8	W/R	控制寄存器
spi_sta	2'b01	8	R	状态寄存器
wfifo/rfifo	2'b10	8	W/R	数据寄存器
spi_ext	2'b11	8	W/R	拓展寄存器

1.3.1 控制寄存器 (地址为2'b00)

control_register(spi_con)

spi_en	spi_int_en	mode_sel	cpl	cpha	sck_sel[2]	sck_sel[1]	sck_sel[0]
--------	------------	----------	-----	------	------------	------------	------------

bit	function
spi_en	SPI芯片使能信号("1"有效)
spi_int_en	中断使能信号("1"有效)
mode_sel	主/从模式选择信号 ("1"主模式 / "0"从模式)
cpl	时钟极性
cpha	时钟相位
sck_sel[2:0]	SPI时钟频率选择信号

1.3.2 拓展寄存器 (地址为2'b11)

extension_register(spi_ext)

intcnt_sel[1]	intcnt_sel[0]	sck_mode[2]	sck_mode[1]	sck_mode[0]	slave_sel[2]	slave_sel[1]	slave_sel[0]
---------------	---------------	-------------	-------------	-------------	--------------	--------------	--------------

bit	function
intcnt_sel[1:0]	计数中断信号 - 决定数据块的大小，可选择在传输了多少个字节后产生中断信号 2'b00 - 每传输一个字节产生一个中断信号 2'b01 - 每传输两个字节产生一个中断信号 2'b10 - 每传输三个字节产生一个中断信号 2'b11 - 每传输四个字节产生一个中断信号
sck_mode[2:0]	系统时钟分频模式选择信号 (稍后详解)
slave_sel[2:0]	从机8选1信号 ("1"有效)

1.3.3 状态寄存器 (地址为2'b01)

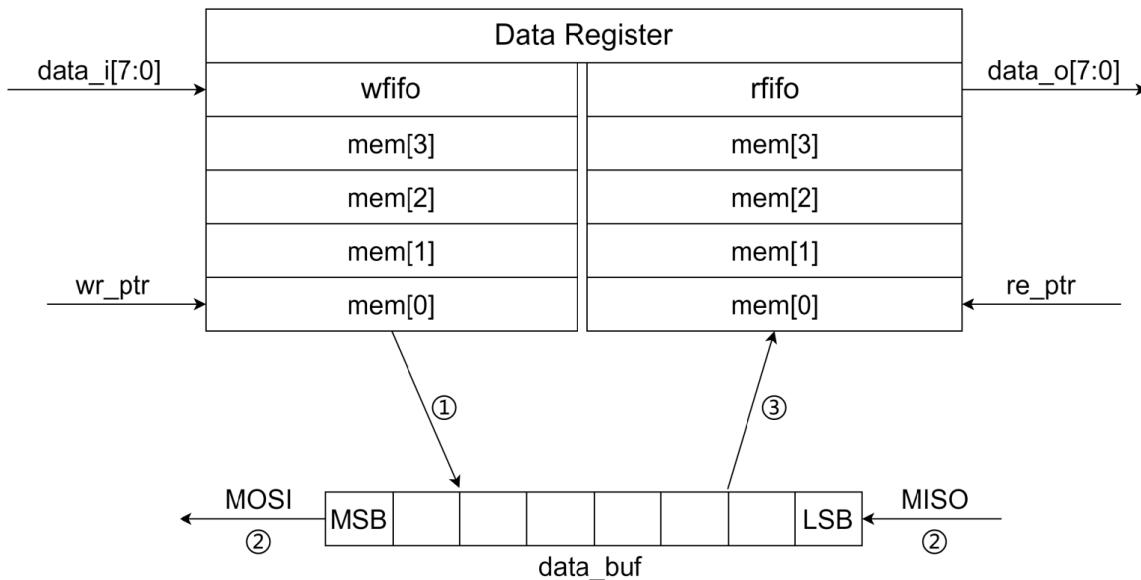
status_register(spi_sta)

spi_int_f	wr_col_f	x	x	wfifo_ful	wfifo_emp	rfifo_ful	rfifo_emp
-----------	----------	---	---	-----------	-----------	-----------	-----------

bit	function
spi_int_f	SPI中断标志信号
wr_col_f	写FIFO写冲突标志信号
wfifo_ful	写FIFO 满标志
wfifo_emp	写FIFO 空标志
rfifo_ful	读FIFO 满标志
rfifo_emp	读FIFO 空标志

1.3.4 数据寄存器 (地址为2'b10)

本设计选择使用深度为4bytes的FIFO作为数据寄存器。在主模式下，将要发出的数据存到写FIFO (wfifo) , 将收到的数据存到读FIFO (rfifo) 。具体工作流程如下图所示：



I. data_i -> wfifo

配置好控制寄存器和拓展寄存器后，地址选择数据寄存器2'10，此时类Wishbone总线的数据_i即为要传输的数据，将该数据存到wfifo，最多可存四字节数据。

II. wfifo -> data_buf -> rfifo

① wfifo不为空时，会根据FIFO的指针，将数据加载到data_buf中。

② 由状态机开始传输数据，MOSI通过MOSI线输出，整体移位后MISO线的数据存到LSB，等待8bit数据传输完成。

③ data_buf更新完成后，存入rfifo的指针位置。

III. rfifo -> data_o

类Wishbone总线可通过data_o从rfifo中指针位置读出数据。

1.4 SPI时钟频率SCK

SPI传输频率由系统时钟分频得到，具体分频由控制寄存器和拓展寄存器共同决定：

- sck_mode = spi_con [5:3]
- sck_sel = spi_ext [2:0]

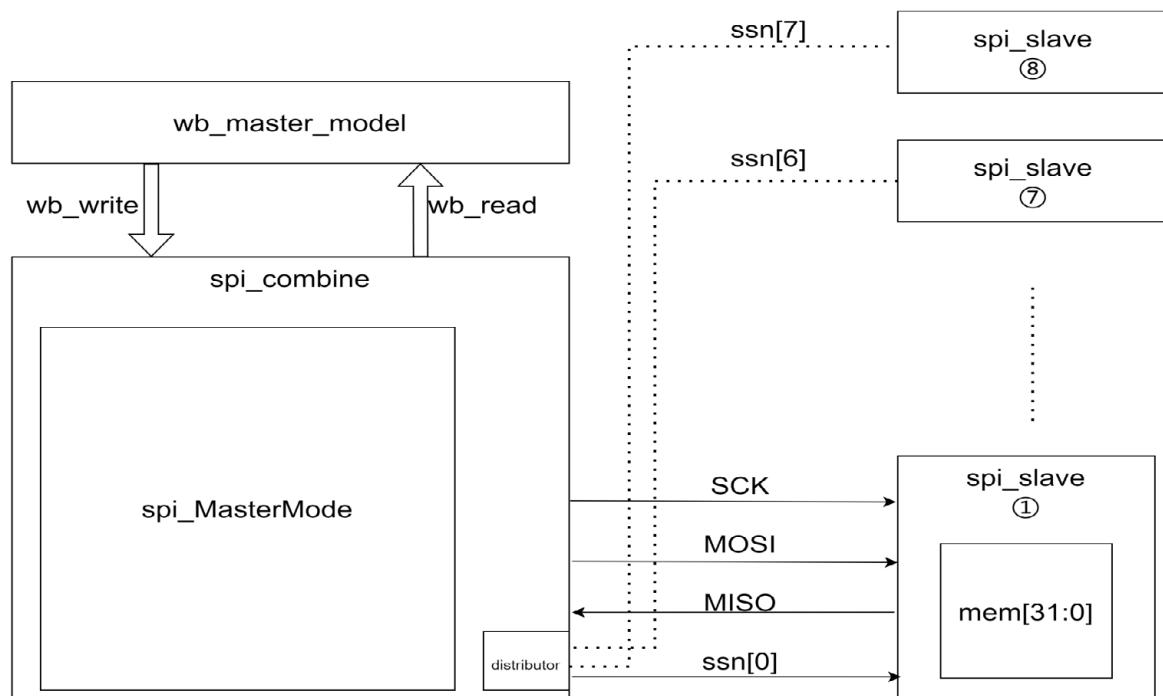
sck_mode \ sck_sel	000	001	010	011	100	101	110	111
001	4分频	8	16	64	256	1024	4096	16384
010	10	20	40	160	320	640	2560	5120
100	10	50	100	200	500	1000	5000	10000
default	4	8	16	64	256	1024	4096	16384

2. 功能仿真

2.1 主模式仿真

- (前仿时，rtl中spi_fifo.v文件中Write memory模块使用negedge触发，与后续做dc的RTL该文件该处为posedge触发有区别，原因是整理文档时跑前仿出错了，后续都改太麻烦了，综合后仿真和布局布线后仿真都没有问题，所以就没改 T T)。

2.1.1 仿真方案

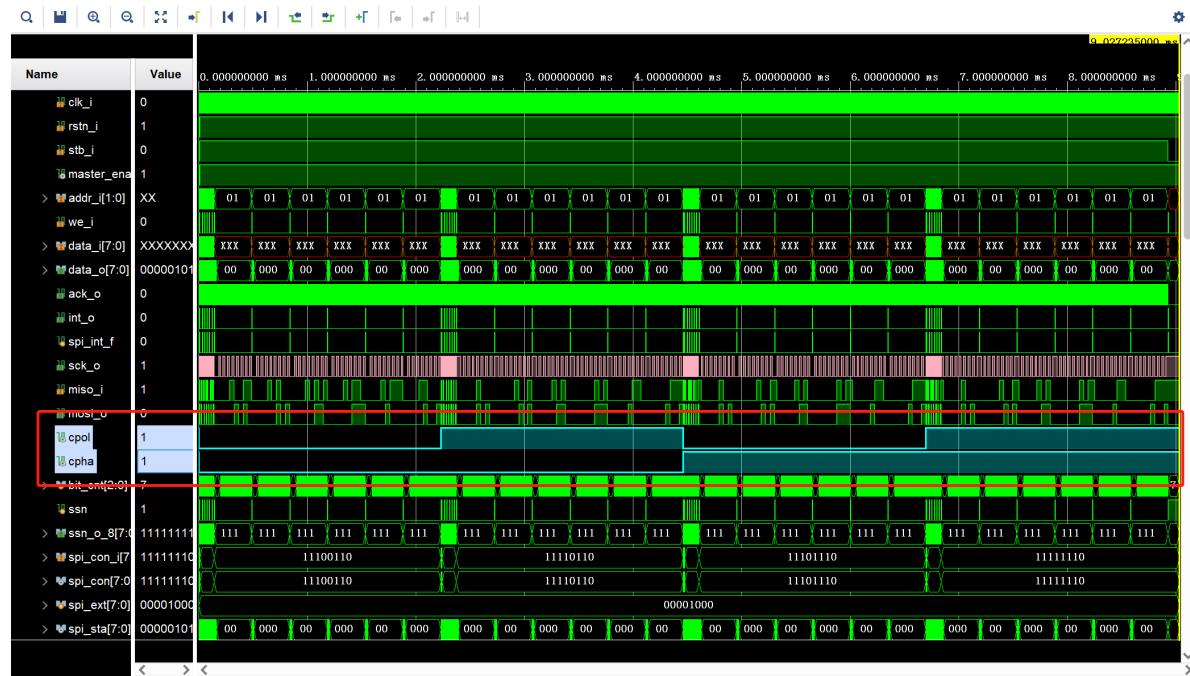


基本仿真方案如上图所示，除主模块外，编写两个仿真测试模块：wb_master_model和spi_slave。wb_master_model模块通过wb_write和wb_read两个task来模拟总线对芯片的读写操作。spi_slave为测试使用的SPI从机模块，只有复位端口及4个标准SPI端口。定义一个内部存储mem来存储从机发给主机的数据。数据传输时，每次传输将一个byte数据从mem的指定地址中取出，传输完成后，将收到的数据存到原来的地址。

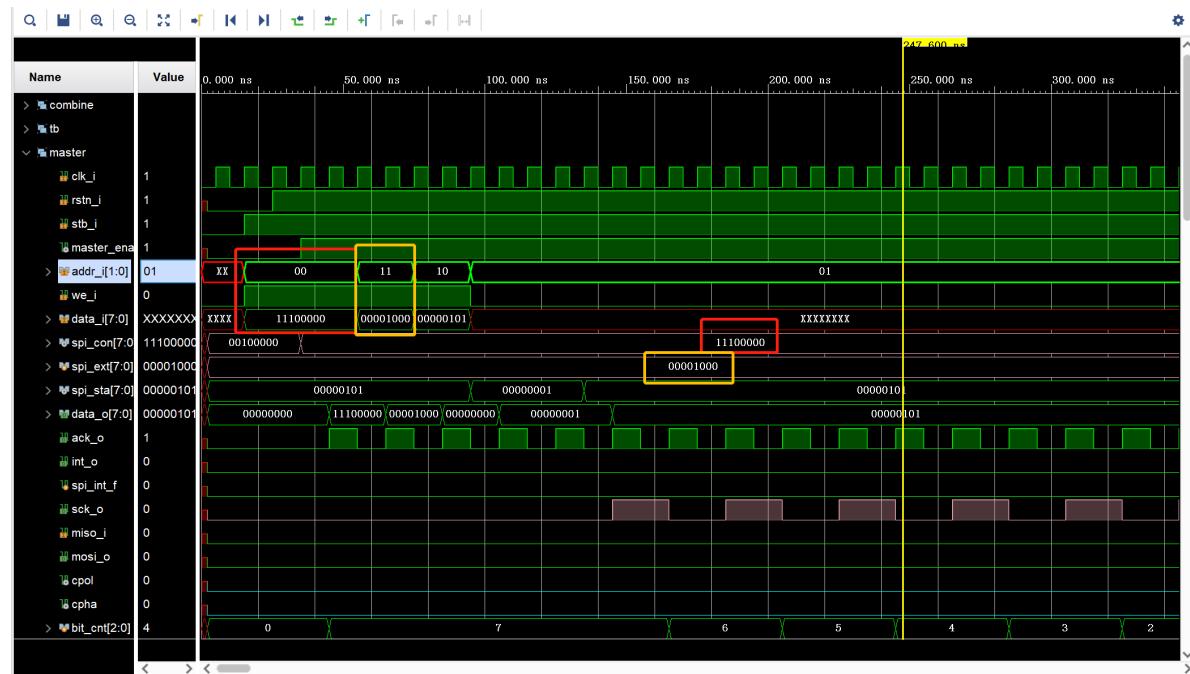
测试时，cpol和cpha不同值可切换四种工作模式，每种工作模式下选择四种SCK频率（4分频、16分频、256分频、4096分频）测试，每个频率下传输4 bytes的数据。仿真结果如下。

2.1.2 仿真结果

①整体结果

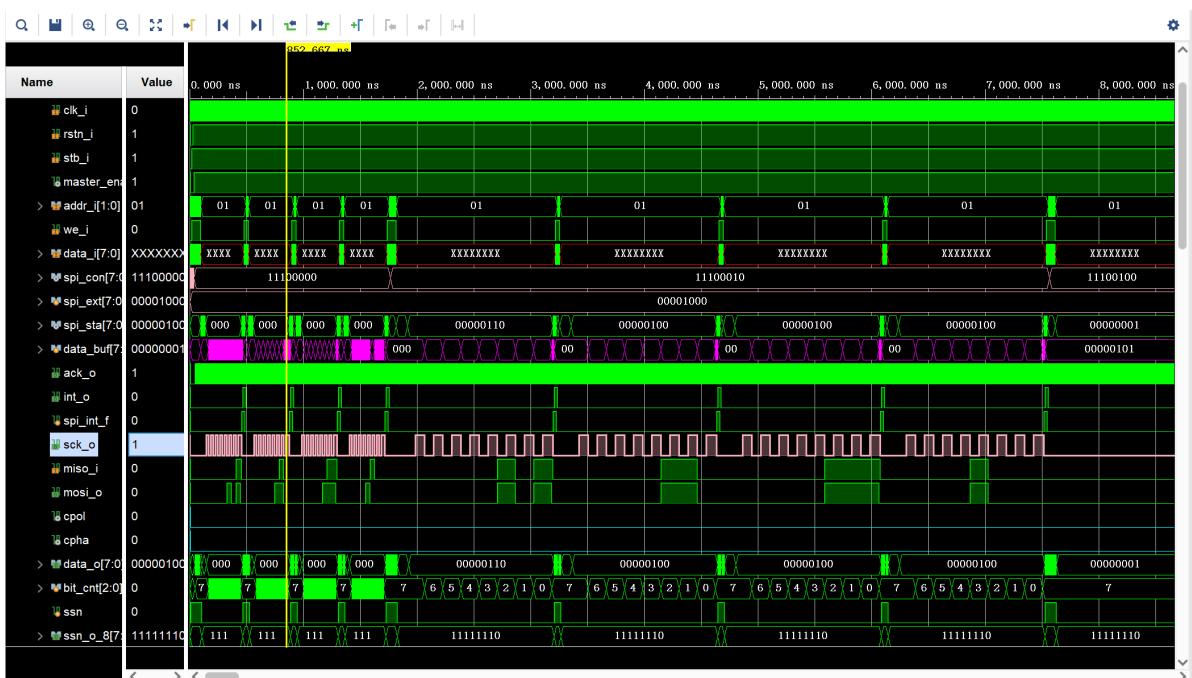
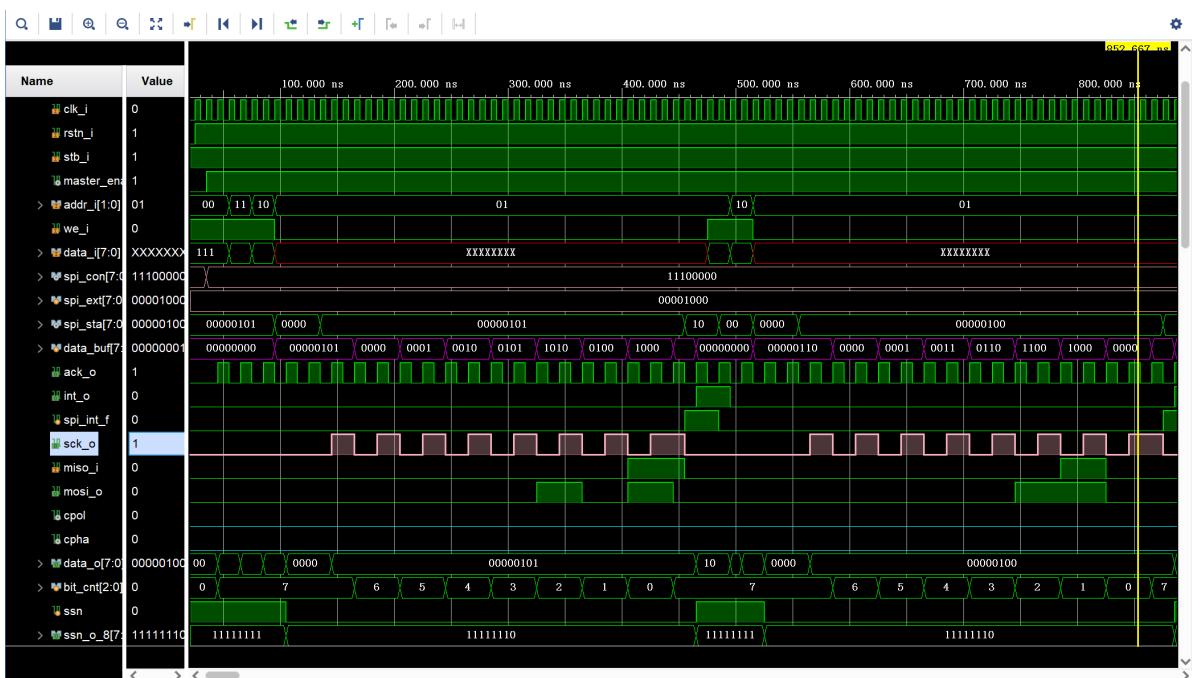


②配置控制寄存器和拓展寄存器

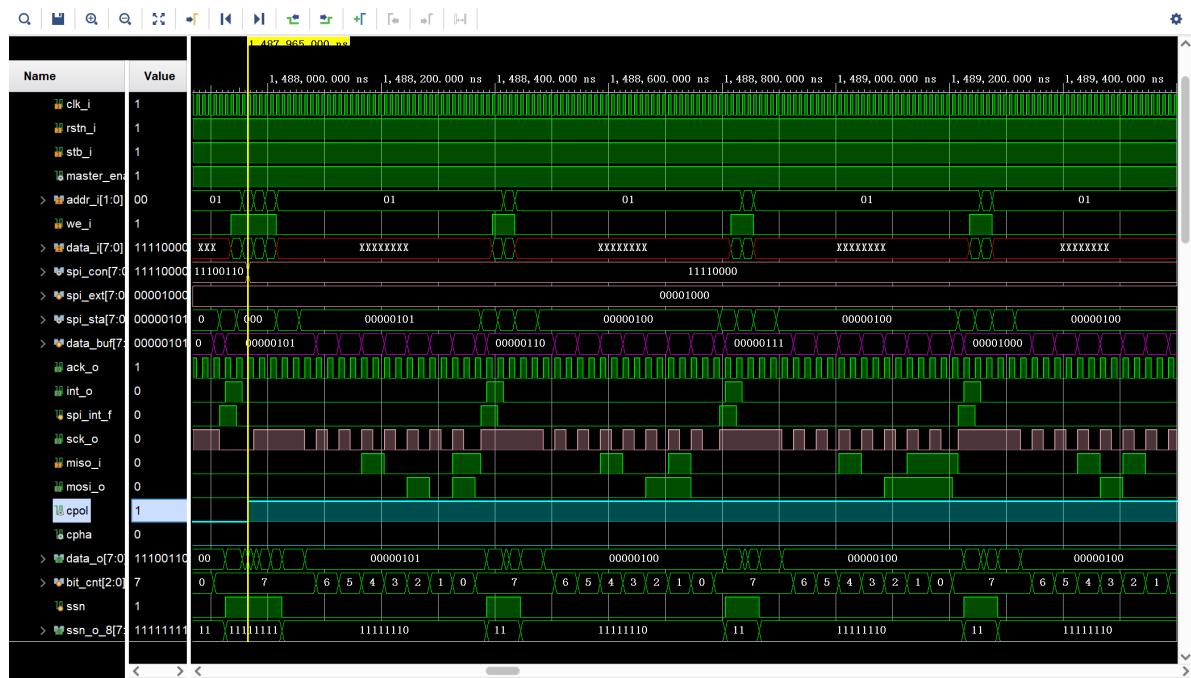


③发送并接收数据（发送的数据为5、6、7、8循环，接收的数据1-32依次递增）

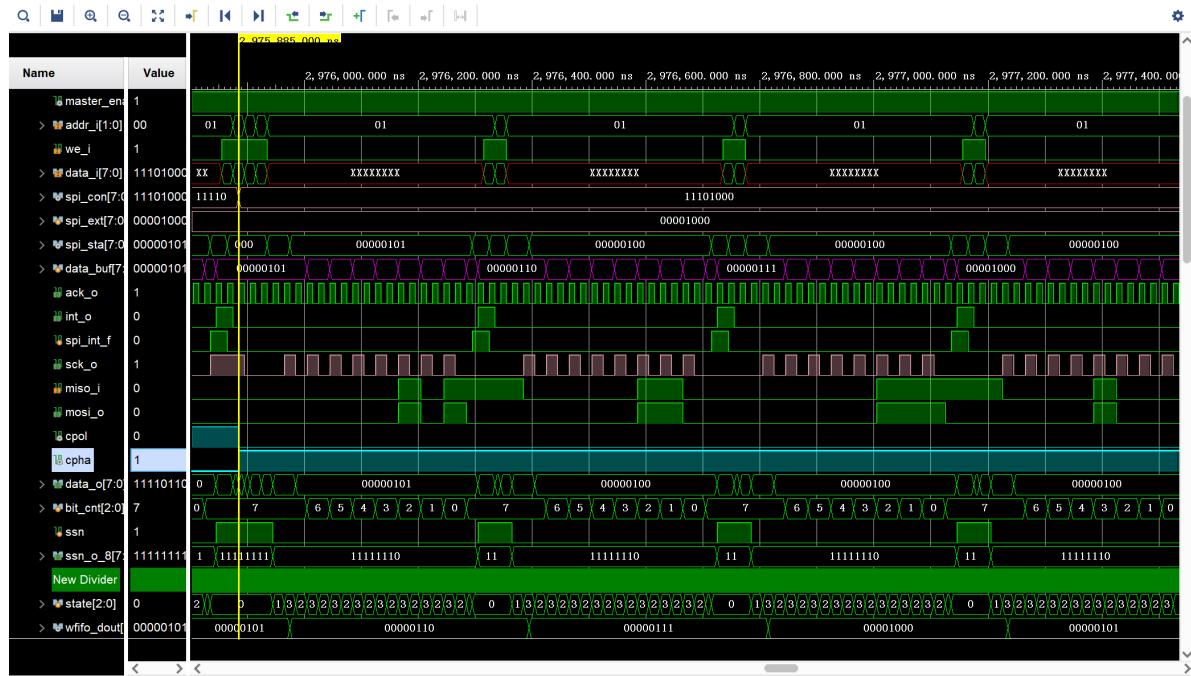
00模式：（空闲为低电平，上升沿采样数据，下降沿切换数据）



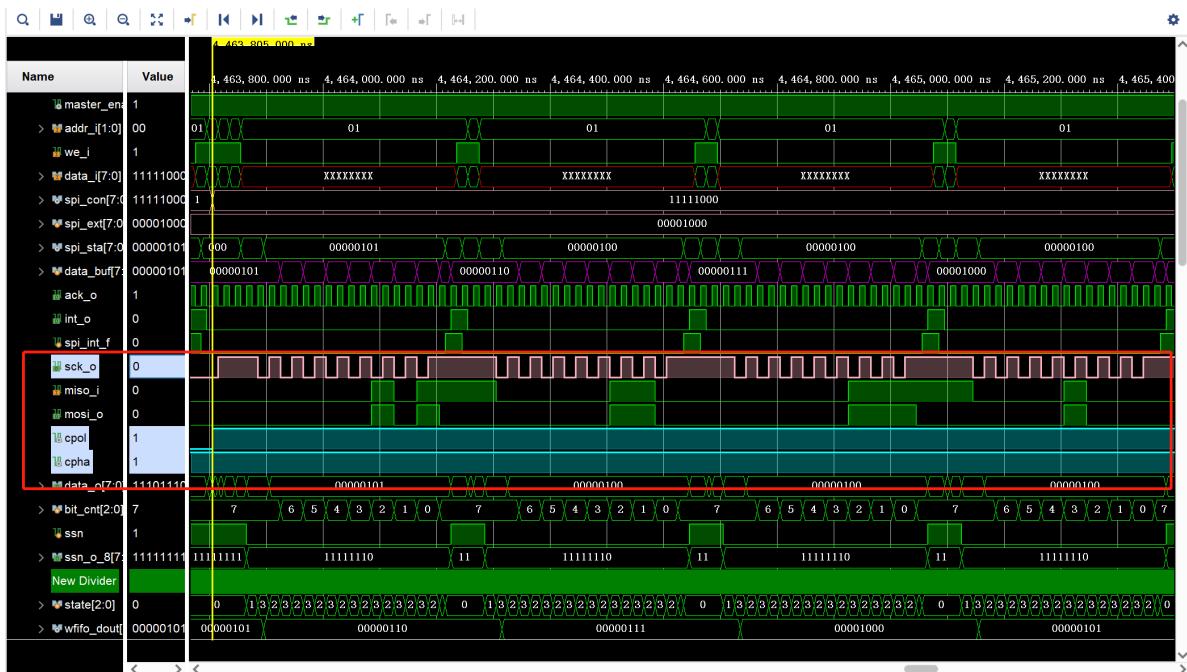
10模式：(空闲为高电平，下降沿采样数据，上升沿切换数据)



01模式：(空闲为低电平，下降沿采样数据，上升沿切换数据)

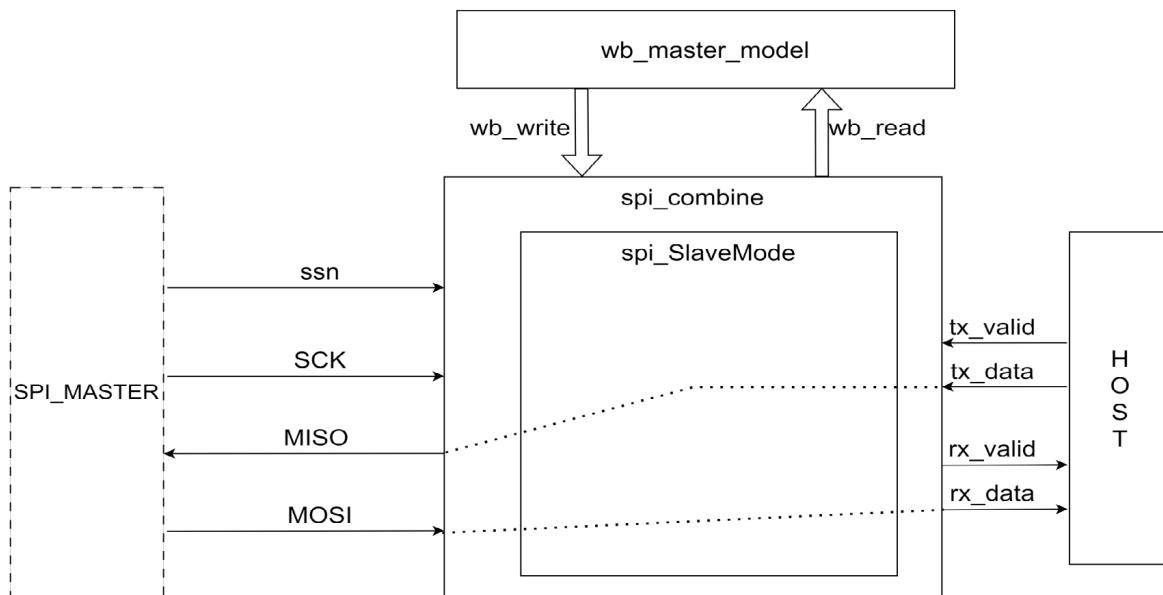


11模式：(空闲为高电平，上升沿采样数据，下降沿切换数据)



2.2 从模式仿真

2.2.1 仿真方案

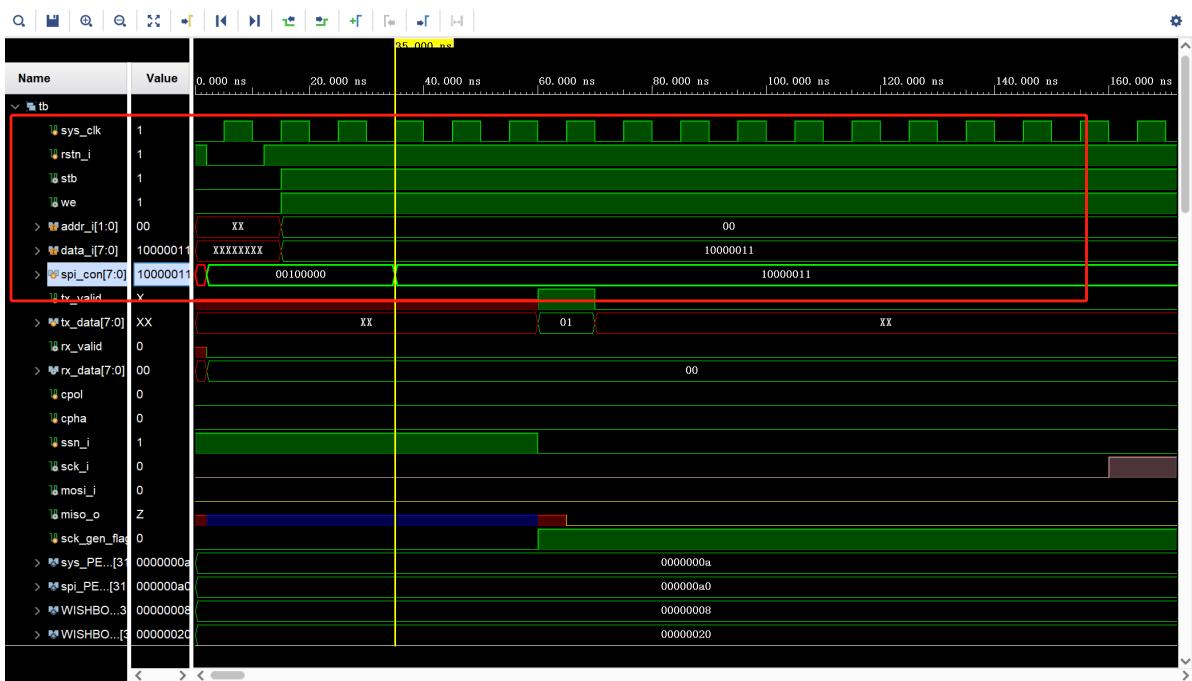


从模式仿真时，直接通过TB来输入ssn, sck和mosi信号，tx_data数据通过miso线发送出去。简单示意图如上所示。

仿真时，系统时钟周期为10ns，SPI时钟周期为160ns。从机（本芯片）要通过miso发出的数据为1~15，通过mosi接收的数据为15~1。

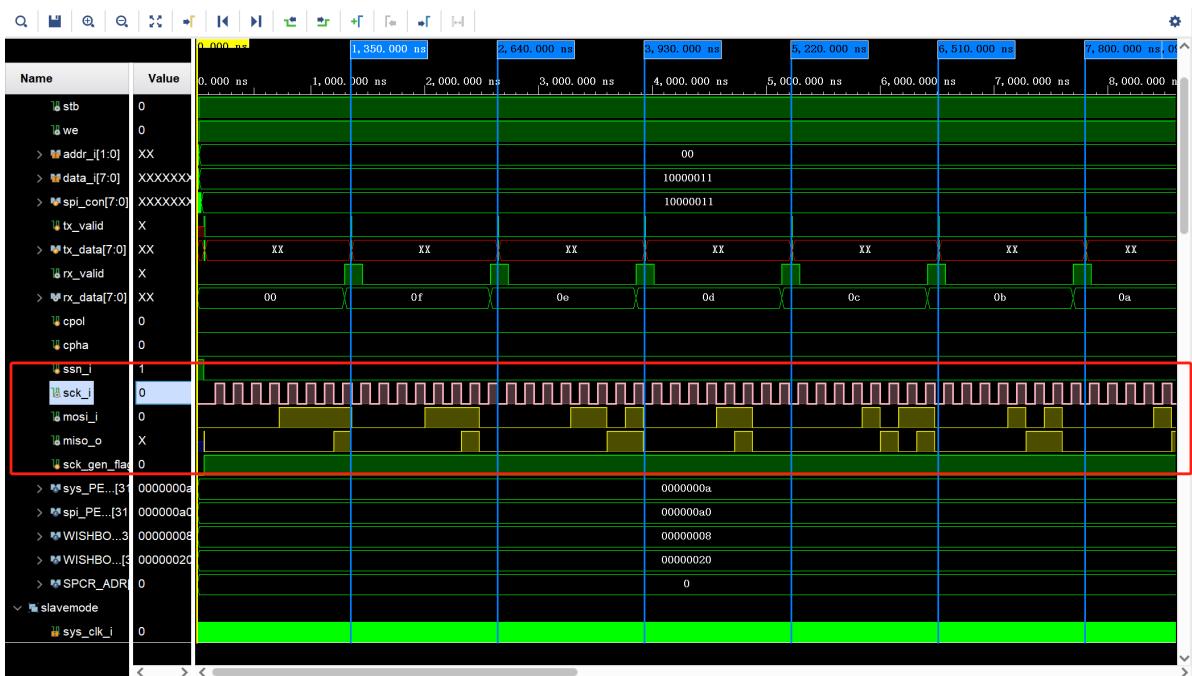
2.2.2 仿真结果

①配置控制寄存器，选择从模式 ($\text{spi_con}[5] = 0$)，分频选择16分频 ($\text{spi_con}[2:0] = 2'b011$)。



②等到sck时钟后开始传输数据

00模式: (空闲为低, 上升沿采样, 下降沿切换)



11模式: (空闲为高, 上升沿采样, 下降沿切换)

(修改tb_spi_slave.v中的第81行, 配置控制寄存器第4位为"11", 第87行cpol和cpha改为"11"。)



至此，功能仿真结束，实际上还有一些bug需要修复。

3.DC综合

综合使用的脚本为compile_dc.tcl，将时序约束和综合指令均写到脚本内，终端键入dc_shell-t -f compile_dc.tcl即可执行。

3.1 综合脚本compile_dc.tcl (部分)

①设置综合环境

```
#*****设置综合环境*****
#/* All verilog files, separated by spaces */ 
set RTL_PATH "/home/IC/SPI_CHIP/RTL"
lappend search_path $RTL_PATH
set RTL_files [list bitdistributor8.v inout_port.v spi_combine.v spi_fifo.v
spi_MasterMode.v spi_SlaveMode.v]
#/* Top-level Module */
set my_toplevel spi_combine
#/*
set file_version dc_v1
#/* set folder name
set RPT_DIR RPT
set OUT_DIR OUT
set RPT_OUT [format "%s%s" $RPT_DIR/ $file_version]
set DATA_OUT [format "%s%s" $OUT_DIR/ $file_version]
#*****设置综合环境*****
```

②设置综合工艺库文件

```

#!/*****设置综合库文件*****/
set SMIC_PDK_PATH "/home/IC/SPI_CHIP/FDRY";      #Change to your storage path
lappend search_path $SMIC_PDK_PATH
set link_library [list
$SMIC_PDK_PATH/scc018v3ebcd_uhd50_rvt_tt_v5p0_25c_basic.db] # can be changed
set target_library $link_library
define_design_lib WORK -path ./WORK
#/*****设置综合库文件*****/

```

③设置时钟约束

```

#!/*****设置约束条件*****/
#/Create clock
set sys_clk clk_i
/* Target frequency in MHz for optimization */
set sys_clk_freq_MHz 80

#/Create spi sck for slavemode
set spi_sck spi_SlaveMode_u/sck_i
/* Target frequency in MHz for optimization */
set spi_sck_freq_MHz 50

set sys_clk_period [expr 1000 / $sys_clk_freq_MHz]
set sck_period      [expr 1000 / $spi_sck_freq_MHz]

set find_clock [ find port [list $sys_clk] ]
if { $find_clock != [list] } {
    set clk_name $sys_clk
    create_clock -period $sys_clk_period $clk_name
} else {
    set clk_name vclk
    create_clock -period $sys_clk_period -name $clk_name
}

/* System clock constrain begin*/
set_dont_touch_network [get_clocks $sys_clk]
set_ideal_network [get_clocks $sys_clk]
set_clock_latency -max 0.5 [get_clocks $sys_clk]
set_clock_uncertainty -setup 0.6 [get_clocks $sys_clk]
set_clock_uncertainty -hold 0.5 [get_clocks $sys_clk]
set_clock_transition 0.4 [get_clocks $sys_clk]
/* System clock constrain end*/

set find_clock2 [ find port [list $spi_sck] ]
if { $find_clock2 != [list] } {
    set clk_name2 $spi_sck
    create_clock -period $sck_period $clk_name2
} else {
    set clk_name2 vclk
    create_clock -period $sck_period -name $clk_name2
}

/* SPI spi_sck constrain begin*/
set_dont_touch_network [get_clocks $spi_sck]
set_ideal_network [get_clocks $spi_sck]
set_clock_uncertainty -setup 0.6 [get_clocks $spi_sck]
set_clock_uncertainty -hold 0.5 [get_clocks $spi_sck]

```

```

set_clock_transition 0.4 [get_clocks $spi_sck]
/* SPI spi_sck constrain begin*/
set_clock_groups -asynchronous -group [get_clocks $clk_name] -group [get_clocks $clk_name2]
/* SPI asynchronous clks constrain end*/

/*
set driving_cell CLKBUFV2_7TV50
set loading_cell CLKBUFV8_7TV50

set_drive [drive_of scc018v3ebcd_uhd50_rvt_tt_v5p0_25c_basic/${driving_cell}/z]
[all_inputs]
set_load [load_of scc018v3ebcd_uhd50_rvt_tt_v5p0_25c_basic/${loading_cell}/I]
[all_outputs]
*/

#endif

/*add constrain
set input_delay_ns 5
set output_delay_ns 5

/* input/output delay
set_input_delay $input_delay_ns -clock $clk_name [remove_from_collection
[all_inputs] rstn_i]
set_output_delay $output_delay_ns -clock $clk_name [all_outputs]

# multicycle_path (not used)
#set_multicycle_path -setup 2 -from A -to B
#set_multicycle_path -hold 1 -from A -to B

# false path
# set asynchronous reset signal as false path
set_false_path -from [get_ports rstn_i] -to [all_registers]
*****设置约束条件*****/

```

- 将系统时钟与SPI时钟设置为异步时钟组

```

188 /* SPT asynchronous clks constrain begin*/
189 set_clock_groups -asynchronous -group [get_clocks $clk_name] -group [get_clocks $clk_name2]
190
191 /* SPI asynchronous clks constrain end*/
192
193 /*
194 set driving_cell CLKBUFV2_7TV50
195 set loading_cell CLKBUFV8_7TV50
196
197 set_drive [drive_of scc018v3ebcd_uhd50_rvt_tt_v5p0_25c_basic/${driving_cell}/Z] [all_inputs]
198 set_load [load_of scc018v3ebcd_uhd50_rvt_tt_v5p0_25c_basic/${loading_cell}/I] [all_outputs]
199 */
200
201 #/add constrain
202 set input_delay_ns 5
203 set output_delay_ns 5
204
205 /* input/output delay
206
207 set_input_delay $input_delay_ns -clock $clk_name [remove_from_collection [all_inputs] rstn_i]
208 set_output_delay $output_delay_ns -clock $clk_name [all_outputs]
~~

```

④输出时序检查报告及综合结果

```

*****检查综合结果并输出报告*****
check_design > $RPT_OUT/check_design.rpt
check_timing > $RPT_OUT/check_timing.rpt

report_design -verbose > $RPT_OUT/design.rpt
report_qor > $RPT_OUT/qor.rpt
report_area > $RPT_OUT/area.rpt

```

```

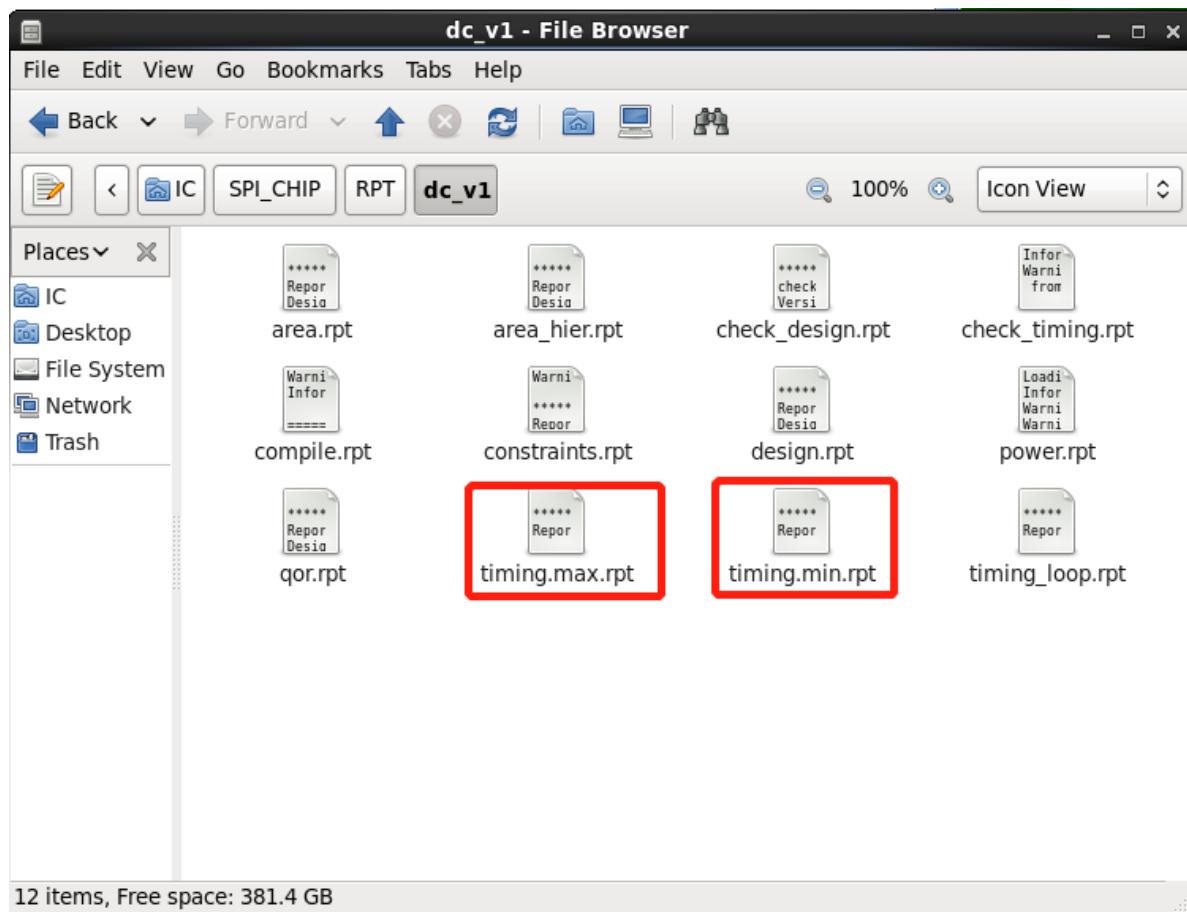
report_area -hierarchy > $RPT_OUT/area_hier.rpt
report_timing -loops > $RPT_OUT/timing_loop.rpt
report_timing -path full -net -cap -input -tran -delay min -max_paths 3 -nworst
3 > $RPT_OUT/timing.min.rpt
report_timing -path full -net -cap -input -tran -delay max -max_paths 3 -nworst
3 > $RPT_OUT/timing.max.rpt
report_constraints -all_violators -verbose > $RPT_OUT/constraints.rpt
report_power > $RPT_OUT/power.rpt
#/*****检查综合结果并输出报告*****/

#/*****输出综合结果*****/
set bus_naming_style {%-d}
write_file -f verilog -hierarchy -output $DATA_OUT/$my_toplevel.v
write_sdf -version 2.1 $DATA_OUT/$my_toplevel.sdf
write_file -f ddc -hierarchy -output $DATA_OUT/$my_toplevel.ddc
write_sdc $DATA_OUT/$my_toplevel.sdc
set_svf -off
#/*****输出综合结果*****/

```

3.2 时序检查报告

- dc_v1对应tt25工艺脚，dc_v2对应ss150工艺脚，dc_v3对应ff-40工艺脚。



① timing.max.rpt 建立时间检查报告

```

1 ****
2 **** Report : timing
3 **** -path full
4 **** -delay max
5 **** -worst 3
6 **** -input_pins
7 **** -nets
8 **** -max_paths 3
9 **** -transition_time
10 **** -capacitance
11 Design : spi_combine
12 Version: K-2015.06
13 Date : Mon Jan 1 18:08:46 2024
14 ****
15 **** Operating Conditions: tt_v5p0_25c Library: scc018v3ebcd_uhd50_rvt_tt_v5p0_25c_basic
16 **** Wire Load Model Mode: top
17 ****
18 Startpoint: ssn_i (Input port clocked by clk_i)
19 Endpoint: miso_o (output port clocked by clk_i)
20 Path Group: clk_i
21 Path Type: max
22
23 Point Fanout Cap Trans Incr Path
24
25 Point Fanout Cap Trans Incr Path
26
27 clock clk_i (rise edge) 0.00 0.00
28 clock network delay (ideal) 0.50 0.50
29 input external delay 5.00 5.50 f
30 ssn_i (in) 0.02 0.02 5.52 f
31 ssn_i (net) 1 0.00 0.00 5.52 f
32 spi_SlaveMode/u/ssn_i (spi_SlaveMode) 0.00 0.00 5.52 f
33 spi_SlaveMode/u/ssn_i (net) 0.00 0.00 5.52 f
34 spi_SlaveMode/u/U19/I (INV1_7TV50) | 0.02 0.00 5.52 f
35 spi_SlaveMode/u/U19/ZN (INV1_7TV50) 0.17 0.11 5.63 r
36 spi_SlaveMode/u/n47 (net) 2 0.01 0.00 5.63 r
37 spi_SlaveMode/u/miso_o_tri/OE (TBUFV1_7TV50) 0.17 0.00 5.63 r
38 spi_SlaveMode/u/miso_o_tri/Z (TBUFV1_7TV50) 0.12 0.23 5.86 f
39 spi_SlaveMode/u/miso_o (net) 1 0.00 0.00 5.86 f
40 spi_SlaveMode/u/miso_o (spi_SlaveMode) 0.00 0.00 5.86 f
41 miso_o (net) 0.00 0.00 5.86 f
42 miso_inout_port/inout_port (inout_port_1) 0.00 0.00 5.86 f
43 miso_inout_port/outline (net) 0.00 0.00 5.86 f
44 miso_inout_port/portline/tri1 (TBUFV1_7TV50) 0.12 0.00 5.86 f
45 miso_inout_port/portline/tri/Z (TBUFV1_7TV50) 0.24 0.41 6.27 f
46 miso_inout_port/portline (net) 2 0.01 0.00 6.27 f
47 miso_inout_port/portline (inout_port_1) 0.00 0.00 6.27 f
48 miso (net) 0.01 0.00 6.27 f
49 miso (inout) 0.24 0.00 6.27 f
50 data arrival time 6.27
51
52 clock clk_i (rise edge) 12.00 12.00
53 clock network delay (ideal) 0.50 12.50
54 clock uncertainty -0.60 11.90
55 output external delay -5.00 6.90
56 data required time 6.90
57
58 data required time 6.99
59 data arrival time 6.99
60
61 slack (MET) 0.63
62

```

②timing.min.rpt 保持时间检查报告

```

1 |
2 ****
3 Report : timing
4 -path full
5 -delay min
6 -worst 3
7 -input_pins
8 -nets
9 -max_paths 3
10 -transition_time
11 -capacitance
12 Design : spi_combine
13 Version: K-2015.06
14 Date : Mon Jan 1 18:08:46 2024
15 ****
16 **** Operating Conditions: tt_v5p0_25c Library: scc018v3ebcd_uhd50_rvt_tt_v5p0_25c_basic
17 **** Wire Load Model Mode: top
18
19 Startpoint: rstn_temp_reg
20 (rising edge-triggered flip-flop clocked by clk_i)
21 Endpoint: rstn_sync_reg
22 (rising edge-triggered flip-flop clocked by clk_i)
23
24 Path Group: clk_i
25 Path Type: min
26
27 Point Fanout Cap Trans Incr Path
28
29 clock clk_i (rise edge) 0.00 0.00
30 clock network delay (ideal) 0.00 0.00
31 rstn_temp_reg/CK (DROV1_7TV50) 0.40 0.00 0.00 r
32 rstn_temp_reg/Q (DROV1_7TV50) 0.08 0.63 0.63 f
33 rstn_temp (net) 1 0.00 0.00 0.63 f
34 rstn_sync_reg/D (DROV1_7TV50) 0.08 0.00 0.63 f
35 data arrival time 0.63
36
37 clock clk_i (rise edge) 0.00 0.00
38 clock network delay (ideal) 0.00 0.00
39 clock uncertainty 0.50 0.50
40 rstn_sync_reg/CK (DROV1_7TV50) 0.00 0.50 r
41 library hold time 0.09 0.59
42 data required time 0.59
43
44 data required time 0.59
45 data arrival time -0.63
46
47 slack (MET) 0.04
48
49

```

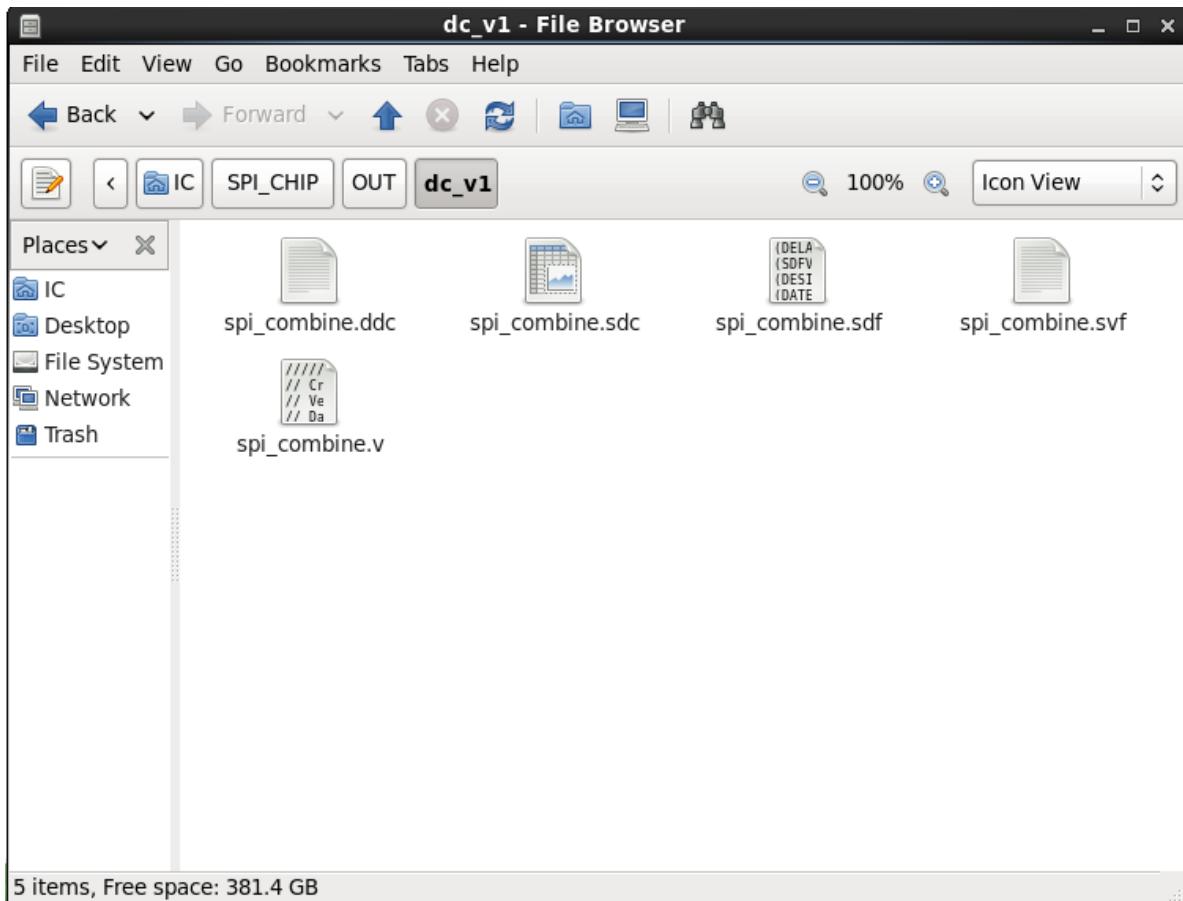
3.3功耗报告

```

7 Report : power
8 -analysis effort low
9 Design : spi_combine
10 Version: K-2015.06
11 Date : Mon Jan 1 18:08:47 2024
12 ****
13
14
15 Library(s) Used:
16
17 scc018v3ebcd_uhd50_rvt_tt_v5p0_25c_basic (File: /home/IC/SPI_CHIP/FDRY/scc018v3ebcd_uhd50_rvt_tt_v5p0_25c_basic.db)
18
19
20 Operating Conditions: tt_v5p0_25c Library: scc018v3ebcd_uhd50_rvt_tt_v5p0_25c_basic
21 Wire Load Model Mode: top
22
23
24 Global Operating Voltage = 5
25 Power-specific unit information :
26 V = Volts
27 Capacitance Units = 1.000000pf
28 Time Units = 1ns
29 Dynamic Power Units = 1mW (derived from V,C,T units)
30 Leakage Power Units = 1uW
31
32
33 Cell Internal Power = 5.4028 mW (97%)
34 Net Switching Power = 172.0538 uW (3%)
35
36 Total Dynamic Power = 5.5757 mW (100%)
37
38 Cell Leakage Power = 663.9746 pW
39
40
41 Internal Power Switching Power Leakage Total
42 Power Group Power Power Power Power (%) Attrs
43
44 io_pad 0.0000 0.0000 0.0000 0.0000 ( 0.00% )
45 vddio 0.0000 0.0000 0.0000 0.0000 ( 0.00% )
46 block_box 0.0000 0.0000 0.0000 0.0000 ( 0.00% )
47 clock_network 2.5799e-02 4.6011e-02 2.1069e-06 7.1810e-02 ( 1.29% )
48 register 5.3394 2.2374e-02 3.1898e-04 5.3618 ( 96.16% )
49 sequential 3.8538e-04 1.6153e-05 5.1045e-06 4.0153e-04 ( 0.01% )
50 combinational 3.7796e-02 0.1045 3.3869e-04 0.1417 ( 2.54% )
51
52 total 5.4028 mW 0.1729 mW 6.6397e-04 uW 5.5757 mW
53

```

3.4综合输出网表



- **.ddc 文件:** Design Description Constraints 文件，通常包含关于设计的约束信息，例如时钟频率、时序要求等。这些约束对于综合工具在生成网表时的优化和分析至关重要。
- **.sdc 文件:** Synopsys Design Constraints 文件，包含了对综合和时序分析的详细约束。这些约束包括时钟定义、时序路径、时序关系等，用于确保设计在满足时序和性能要求的情况下进行综合。
- **.sdf 文件:** Standard Delay Format 文件，包含有关每个时序元素的延迟信息，如门延迟、时钟延迟等。这些文件在后仿真时使用，确保仿真与实际硬件的延迟相符。反标在tb文件内可进行综合后仿真。
- **.v 文件:** Verilog 文件，输出网表文件，将verilog语言描述的电路转化为逻辑门的连接，可用于综合后仿真。
- **.svf 文件:** Serial Vector Format 文件，是一种用于 FPGA 配置的文件格式。它包含了配置 FPGA 的序列信息，用于在实际硬件中加载综合后的设计。

4.综合后仿真（门级网表仿真）

前仿在win下使用vivado进行，后仿时使用vcs在linux下进行。

4.1 将综合后输出的.sdf文件反标到tb中

```
initial begin
`ifdef NET_SIM
    $sdf_annotation("../SYN/DC/OUT/dc_v1/spi_combine.sdf", spi_combine);
`endif
end
```

4.2 对编译仿真的脚本文件makefile进行一些修改

```
*****Makefile for task SPI CHIP*****
.PHONY:com sim verdi cov debug clean

*****Variable Definition*****
OUTPUT = spi_master           #从模式仿真时 替换 OUTPUT
#OUTPUT = spi_slave
tb_master = tb_spi_master
tb_slave = tb_spi_slave
ALL_DEFINE = +define+DUMP_VPD \
             +define+DUMP_FSDB \
             +NET_SIM          # 1. 加入NET_SIM宏定义, 使tb中反标有效
VPD_NAME = +vpdfile+${OUTPUT}.vpd
filelist = ./netlist_filelist.f      # 2. 将source文件替换为综合后的网表文件
LIB_FILE = -v ../LIB/scc018v3ebcd_uhd50_rvt.v      # 3. 加入标准单元库的带延时的库文件
#compile command
VCS = vcs -svverilog +v2k \
      -timescale=1ns/1ps \
      -full64 \
      -fsdb \
      -debug_all \
      +notimingcheck \
      -P
/opt/Synopsys/Verdi2015/share/PLI/VCS/LINUXAMD64/novas.tab \
      /opt/Synopsys/Verdi2015/share/PLI/VCS/LINUXAMD64/pli.a \
      \
      +vcs+flush+all \
      +vcd+vcdpluson \
      -f $(filelist) \
      -o ${OUTPUT} \
      -l compile.log \
      $(ALL_DEFINE) \
      $(LIB_FILE)          # 4. 编译时读入延时库文件

#simulation command
SIM = ./${OUTPUT} \
      -l ${OUTPUT}.log \
      ${VPD_NAME} \
      -cm line+cond+fsm+tg1

#start compile
com:
${VCS}

#start simulation
sim:
${SIM}

#start debug command
debug:
dve -vpd ${OUTPUT}.vpd &

#start verdi
verdi:
```

```

# netlist simulation for master
verdi +v2k -sverilog -top ${tb_master} -nologo -f $(filelist) -ssf
tb_spi_master.fsdb &

# netlist simulation for slave
# verdi +v2k -sverilog -top ${tb_slave} -nologo -f $(filelist) -ssf
tb_spi_slave.fsdb &

#show the coverage
cov:
dve -full64 -covdir *vdb &

#clean some file
clean:
rm -rf ./verdiLog ./novas* ./*.fsdb ./csrc *.daidir ./csrc *.log *.vpd
*.vdb simv* *.key *race.out* ${OUTPUT} DVEfiles

```

4.3 netlist_filelist.f

```

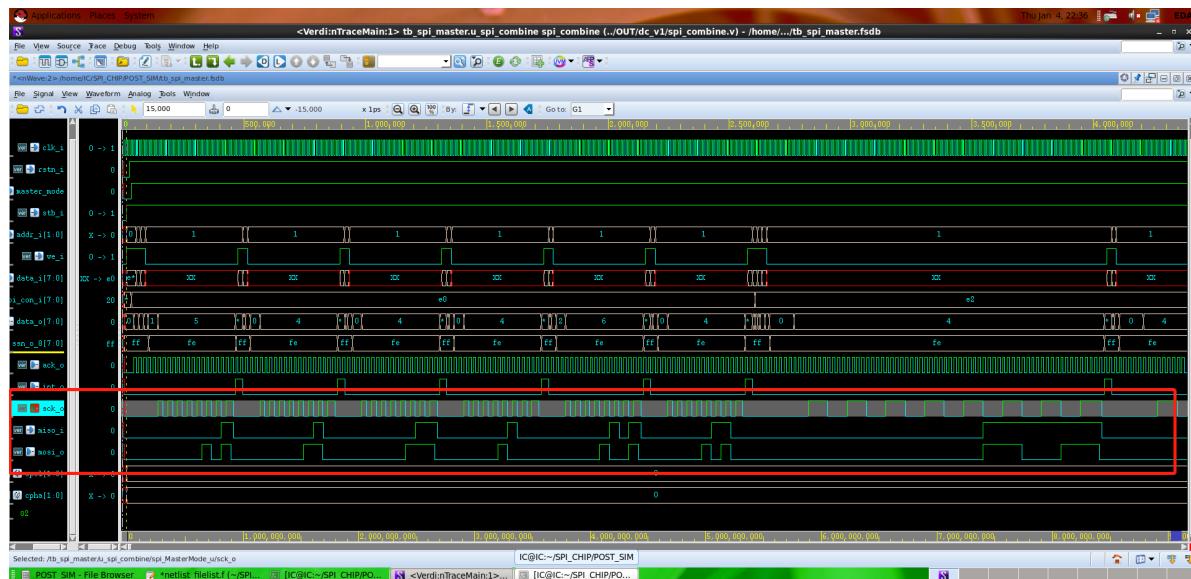
# netlist
../../SYN/DC/OUT/dc_v1/spi_combine.v

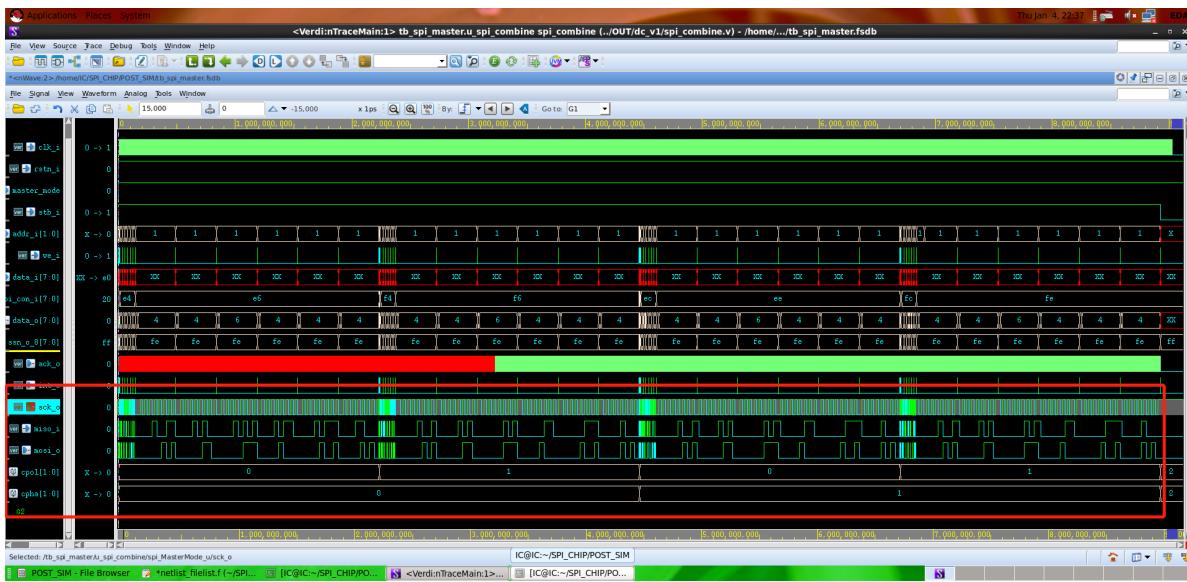
# tb
../../testbench/spi_slave.v
../../testbench/tb_spi_master.v
../../testbench/wb_master_model.v
../../testbench/tb_spi_slave.v

```

4.4 主模式仿真结果

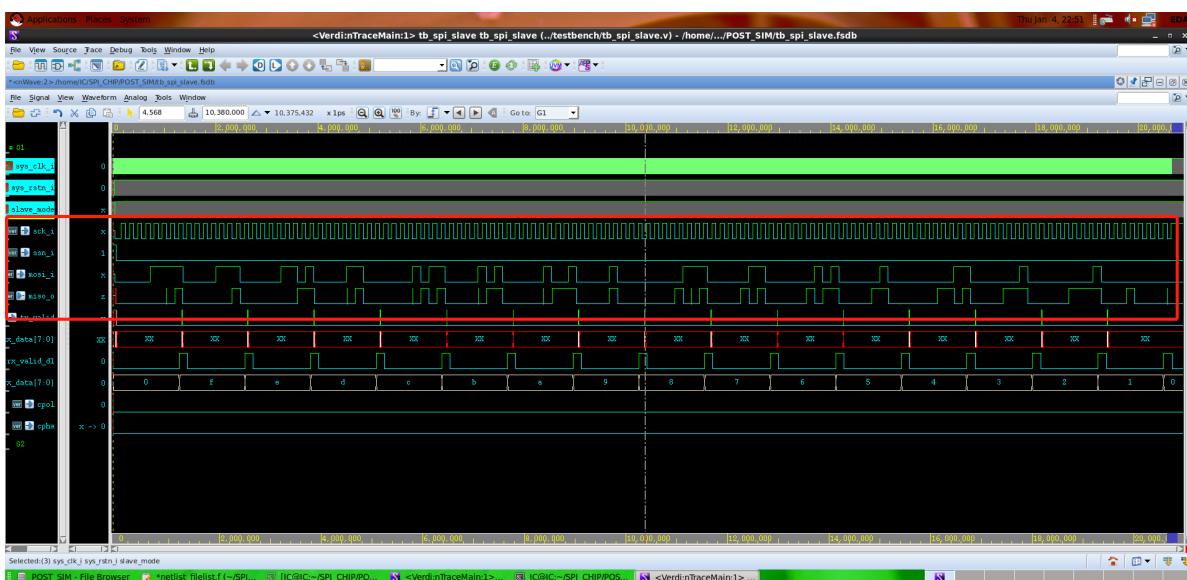
- 与前仿一致





4.5 从模式仿真结果

- 与前仿一致（不知道为啥有毛刺）

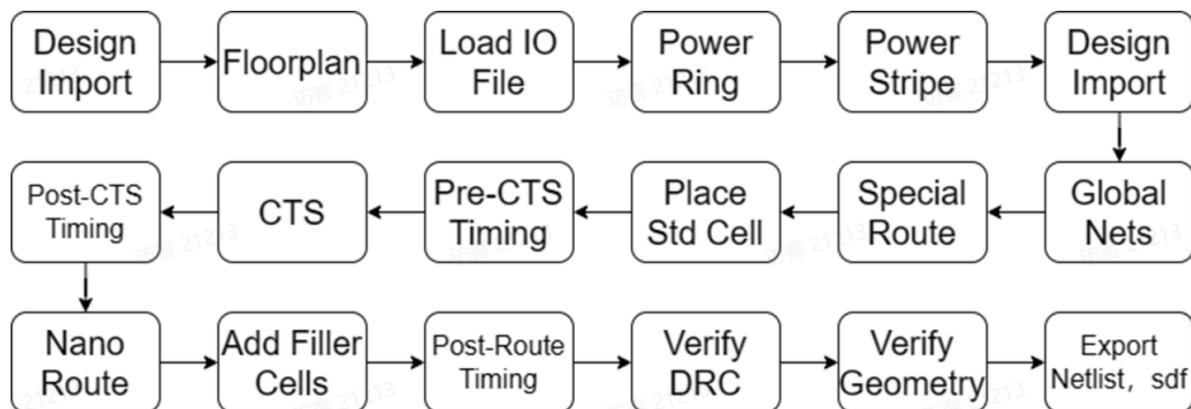


至此，DC综合后的网表仿真结束，与前仿结果基本一致，下一步进行布局布线。

5. 布局布线（版图设计）

- 后端布局布线使用cadence innovas实现。

5.1 后端整体流程



5.2 详细步骤

5.2.1 View设置

手动修改Default.view文件。

将5个lib文件和3个capTbl文件排列组合出**15个Analysis View, 并且全部进行Setup Analysis, Hold Analysis。** 得到最严格和保守的结果。

```
# Current:15 Views
# Version:1.0 MMMC View Definition File
# Do Not Remove Above Line
create_rc_corner -name TYP1 -cap_table {../TYP.capTbl} -T {-40} -preRoute_res
{1.0} -preRoute_cap {1.0} -preRoute_clkres {0.0} -preRoute_clkcap {0.0} -
postRoute_res {1.0} -postRoute_cap {1.0} -postRoute_xcap {1.0} -postRoute_clkres
{0.0} -postRoute_clkcap {0.0}
create_rc_corner -name TYP2 -cap_table {../TYP.capTbl} -T {0} -preRoute_res
{1.0} -preRoute_cap {1.0} -preRoute_clkres {0.0} -preRoute_clkcap {0.0} -
postRoute_res {1.0} -postRoute_cap {1.0} -postRoute_xcap {1.0} -postRoute_clkres
{0.0} -postRoute_clkcap {0.0}
create_rc_corner -name TYP3 -cap_table {../TYP.capTbl} -T {25} -preRoute_res
{1.0} -preRoute_cap {1.0} -preRoute_clkres {0.0} -preRoute_clkcap {0.0} -
postRoute_res {1.0} -postRoute_cap {1.0} -postRoute_xcap {1.0} -postRoute_clkres
{0.0} -postRoute_clkcap {0.0}
create_rc_corner -name TYP4 -cap_table {../TYP.capTbl} -T {85} -preRoute_res
{1.0} -preRoute_cap {1.0} -preRoute_clkres {0.0} -preRoute_clkcap {0.0} -
postRoute_res {1.0} -postRoute_cap {1.0} -postRoute_xcap {1.0} -postRoute_clkres
{0.0} -postRoute_clkcap {0.0}
create_rc_corner -name TYP5 -cap_table {../TYP.capTbl} -T {125} -preRoute_res
{1.0} -preRoute_cap {1.0} -preRoute_clkres {0.0} -preRoute_clkcap {0.0} -
postRoute_res {1.0} -postRoute_cap {1.0} -postRoute_xcap {1.0} -postRoute_clkres
{0.0} -postRoute_clkcap {0.0}
create_rc_corner -name MAX1 -cap_table {../MAX.capTbl} -T {-40} -preRoute_res
{1.0} -preRoute_cap {1.0} -preRoute_clkres {0.0} -preRoute_clkcap {0.0} -
postRoute_res {1.0} -postRoute_cap {1.0} -postRoute_xcap {1.0} -postRoute_clkres
{0.0} -postRoute_clkcap {0.0}
create_rc_corner -name MAX2 -cap_table {../MAX.capTbl} -T {0} -preRoute_res
{1.0} -preRoute_cap {1.0} -preRoute_clkres {0.0} -preRoute_clkcap {0.0} -
postRoute_res {1.0} -postRoute_cap {1.0} -postRoute_xcap {1.0} -postRoute_clkres
{0.0} -postRoute_clkcap {0.0}
create_rc_corner -name MAX3 -cap_table {../MAX.capTbl} -T {25} -preRoute_res
{1.0} -preRoute_cap {1.0} -preRoute_clkres {0.0} -preRoute_clkcap {0.0} -
postRoute_res {1.0} -postRoute_cap {1.0} -postRoute_xcap {1.0} -postRoute_clkres
{0.0} -postRoute_clkcap {0.0}
create_rc_corner -name MAX4 -cap_table {../MAX.capTbl} -T {85} -preRoute_res
{1.0} -preRoute_cap {1.0} -preRoute_clkres {0.0} -preRoute_clkcap {0.0} -
postRoute_res {1.0} -postRoute_cap {1.0} -postRoute_xcap {1.0} -postRoute_clkres
{0.0} -postRoute_clkcap {0.0}
create_rc_corner -name MAX5 -cap_table {../MAX.capTbl} -T {125} -preRoute_res
{1.0} -preRoute_cap {1.0} -preRoute_clkres {0.0} -preRoute_clkcap {0.0} -
postRoute_res {1.0} -postRoute_cap {1.0} -postRoute_xcap {1.0} -postRoute_clkres
{0.0} -postRoute_clkcap {0.0}
create_rc_corner -name MIN1 -cap_table {../MIN.capTbl} -T {-40} -preRoute_res
{1.0} -preRoute_cap {1.0} -preRoute_clkres {0.0} -preRoute_clkcap {0.0} -
postRoute_res {1.0} -postRoute_cap {1.0} -postRoute_xcap {1.0} -postRoute_clkres
{0.0} -postRoute_clkcap {0.0}
```

```

create_rc_corner -name MIN2 -cap_table {../MIN.capTbl} -T {0} -preRoute_res
{1.0} -preRoute_cap {1.0} -preRoute_clkres {0.0} -preRoute_clkcap {0.0} -
postRoute_res {1.0} -postRoute_cap {1.0} -postRoute_xcap {1.0} -postRoute_clkres
{0.0} -postRoute_clkcap {0.0}
create_rc_corner -name MIN3 -cap_table {../MIN.capTbl} -T {25} -preRoute_res
{1.0} -preRoute_cap {1.0} -preRoute_clkres {0.0} -preRoute_clkcap {0.0} -
postRoute_res {1.0} -postRoute_cap {1.0} -postRoute_xcap {1.0} -postRoute_clkres
{0.0} -postRoute_clkcap {0.0}
create_rc_corner -name MIN4 -cap_table {../MIN.capTbl} -T {85} -preRoute_res
{1.0} -preRoute_cap {1.0} -preRoute_clkres {0.0} -preRoute_clkcap {0.0} -
postRoute_res {1.0} -postRoute_cap {1.0} -postRoute_xcap {1.0} -postRoute_clkres
{0.0} -postRoute_clkcap {0.0}
create_rc_corner -name MIN5 -cap_table {../MIN.capTbl} -T {125} -preRoute_res
{1.0} -preRoute_cap {1.0} -preRoute_clkres {0.0} -preRoute_clkcap {0.0} -
postRoute_res {1.0} -postRoute_cap {1.0} -postRoute_xcap {1.0} -postRoute_clkres
{0.0} -postRoute_clkcap {0.0}

create_op_cond -name op_cond1 -library_file
{../DCLayout/prfiles/prfiles/lib/scc018v3ebcd_uhd50_rvt_ff_v5p5_-40c_basic.lib}
-P {1} -V {5.5} -T {-40.0}
create_op_cond -name op_cond2 -library_file
{../DCLayout/prfiles/prfiles/lib/scc018v3ebcd_uhd50_rvt_ff_v5p5_0c_basic.lib} -P
{1} -V {5.5} -T {0.0}
create_op_cond -name op_cond3 -library_file
{../DCLayout/prfiles/prfiles/lib/scc018v3ebcd_uhd50_rvt_tt_v5p0_25c_basic.lib} -
P {1} -V {5.5} -T {25.0}
create_op_cond -name op_cond4 -library_file
{../DCLayout/prfiles/prfiles/lib/scc018v3ebcd_uhd50_rvt_tt_v5p0_85c_basic.lib} -
P {1} -V {5.5} -T {25.0}
create_op_cond -name op_cond5 -library_file
{../DCLayout/prfiles/prfiles/lib/scc018v3ebcd_uhd50_rvt_ss_v4p5_125c_basic.lib} -
-P {1} -V {4.5} -T {125}

create_library_set -name lib1 -timing
{../DCLayout/prfiles/prfiles/lib/scc018v3ebcd_uhd50_rvt_ff_v5p5_-40c_basic.lib}
create_library_set -name lib2 -timing
{../DCLayout/prfiles/prfiles/lib/scc018v3ebcd_uhd50_rvt_ff_v5p5_0c_basic.lib}
create_library_set -name lib3 -timing
{../DCLayout/prfiles/prfiles/lib/scc018v3ebcd_uhd50_rvt_tt_v5p0_25c_basic.lib}
create_library_set -name lib4 -timing
{../DCLayout/prfiles/prfiles/lib/scc018v3ebcd_uhd50_rvt_tt_v5p0_85c_basic.lib}
create_library_set -name lib5 -timing
{../DCLayout/prfiles/prfiles/lib/scc018v3ebcd_uhd50_rvt_ss_v4p5_125c_basic.lib}

create_constraint_mode -name sdc -sdc_files {../OUT/dc_v1/spi_combine.sdc}

create_delay_corner -name lib1_corner1 -library_set {lib1} -rc_corner {TYP1}
create_delay_corner -name lib1_corner2 -library_set {lib1} -rc_corner {MAX1}
create_delay_corner -name lib1_corner3 -library_set {lib1} -rc_corner {MIN1}
create_delay_corner -name lib2_corner1 -library_set {lib2} -rc_corner {TYP1}
create_delay_corner -name lib2_corner2 -library_set {lib2} -rc_corner {MAX1}
create_delay_corner -name lib2_corner3 -library_set {lib2} -rc_corner {MIN1}
create_delay_corner -name lib3_corner1 -library_set {lib3} -rc_corner {TYP1}
create_delay_corner -name lib3_corner2 -library_set {lib3} -rc_corner {MAX1}
create_delay_corner -name lib3_corner3 -library_set {lib3} -rc_corner {MIN1}
create_delay_corner -name lib4_corner1 -library_set {lib4} -rc_corner {TYP1}
create_delay_corner -name lib4_corner2 -library_set {lib4} -rc_corner {MAX1}
create_delay_corner -name lib4_corner3 -library_set {lib4} -rc_corner {MIN1}

```

```

create_delay_corner -name lib5_corner1 -library_set {lib5} -rc_corner {TYP1}
create_delay_corner -name lib5_corner2 -library_set {lib5} -rc_corner {MAX1}
create_delay_corner -name lib5_corner3 -library_set {lib5} -rc_corner {MIN1}

create_analysis_view -name view_lib1_corner1 -constraint_mode {sdc} -
delay_corner {lib1_corner1}
create_analysis_view -name view_lib1_corner2 -constraint_mode {sdc} -
delay_corner {lib1_corner2}
create_analysis_view -name view_lib1_corner3 -constraint_mode {sdc} -
delay_corner {lib1_corner3}
create_analysis_view -name view_lib2_corner1 -constraint_mode {sdc} -
delay_corner {lib2_corner1}
create_analysis_view -name view_lib2_corner2 -constraint_mode {sdc} -
delay_corner {lib2_corner2}
create_analysis_view -name view_lib2_corner3 -constraint_mode {sdc} -
delay_corner {lib2_corner3}
create_analysis_view -name view_lib3_corner1 -constraint_mode {sdc} -
delay_corner {lib3_corner1}
create_analysis_view -name view_lib3_corner2 -constraint_mode {sdc} -
delay_corner {lib3_corner2}
create_analysis_view -name view_lib3_corner3 -constraint_mode {sdc} -
delay_corner {lib3_corner3}
create_analysis_view -name view_lib4_corner1 -constraint_mode {sdc} -
delay_corner {lib4_corner1}
create_analysis_view -name view_lib4_corner2 -constraint_mode {sdc} -
delay_corner {lib4_corner2}
create_analysis_view -name view_lib4_corner3 -constraint_mode {sdc} -
delay_corner {lib4_corner3}
create_analysis_view -name view_lib5_corner1 -constraint_mode {sdc} -
delay_corner {lib5_corner1}
create_analysis_view -name view_lib5_corner2 -constraint_mode {sdc} -
delay_corner {lib5_corner2}
create_analysis_view -name view_lib5_corner3 -constraint_mode {sdc} -
delay_corner {lib5_corner3}

set_analysis_view -setup {view_lib1_corner1 view_lib1_corner2 view_lib1_corner3
view_lib2_corner1 view_lib2_corner2 view_lib2_corner3 view_lib3_corner1
view_lib3_corner2 view_lib3_corner3 view_lib4_corner1 view_lib4_corner2
view_lib4_corner3 view_lib5_corner1 view_lib5_corner2 view_lib5_corner3} -hold
{view_lib1_corner1 view_lib1_corner2 view_lib1_corner3 view_lib2_corner1
view_lib2_corner2 view_lib2_corner3 view_lib3_corner1 view_lib3_corner2
view_lib3_corner3 view_lib4_corner1 view_lib4_corner2 view_lib4_corner3
view_lib5_corner1 view_lib5_corner2 view_lib5_corner3}

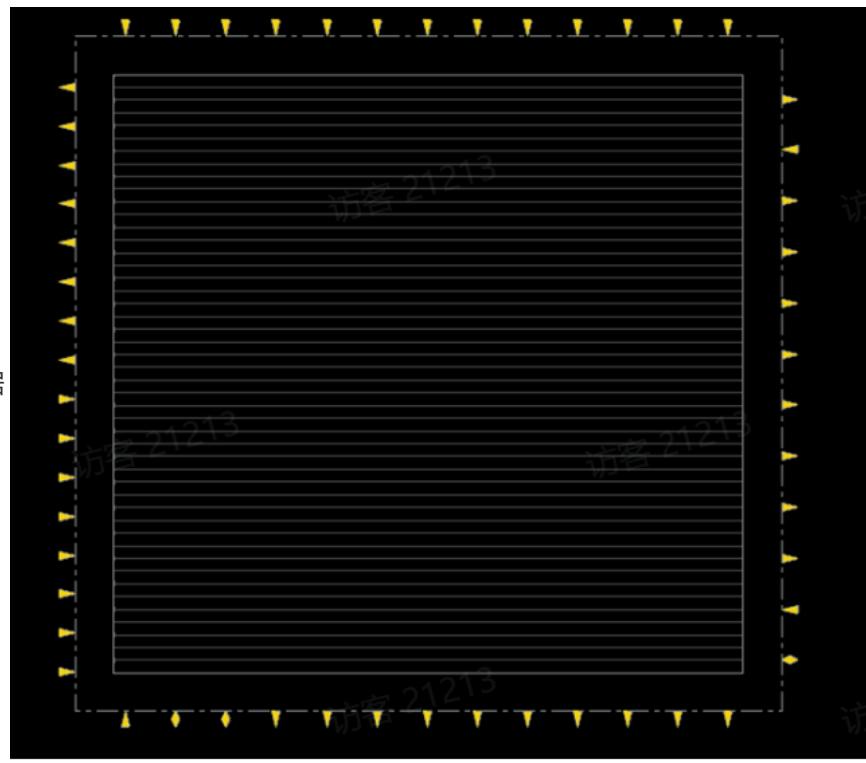
```

5.2.2 IO位置布局

编写IO Assignment File文件。

将56个IO分为13+13+12+16四组，IO Boundary: 185um×175um，通过EXCEL计算，保证同侧信号位
置分布均匀，保证总线信号分布于相同一侧。

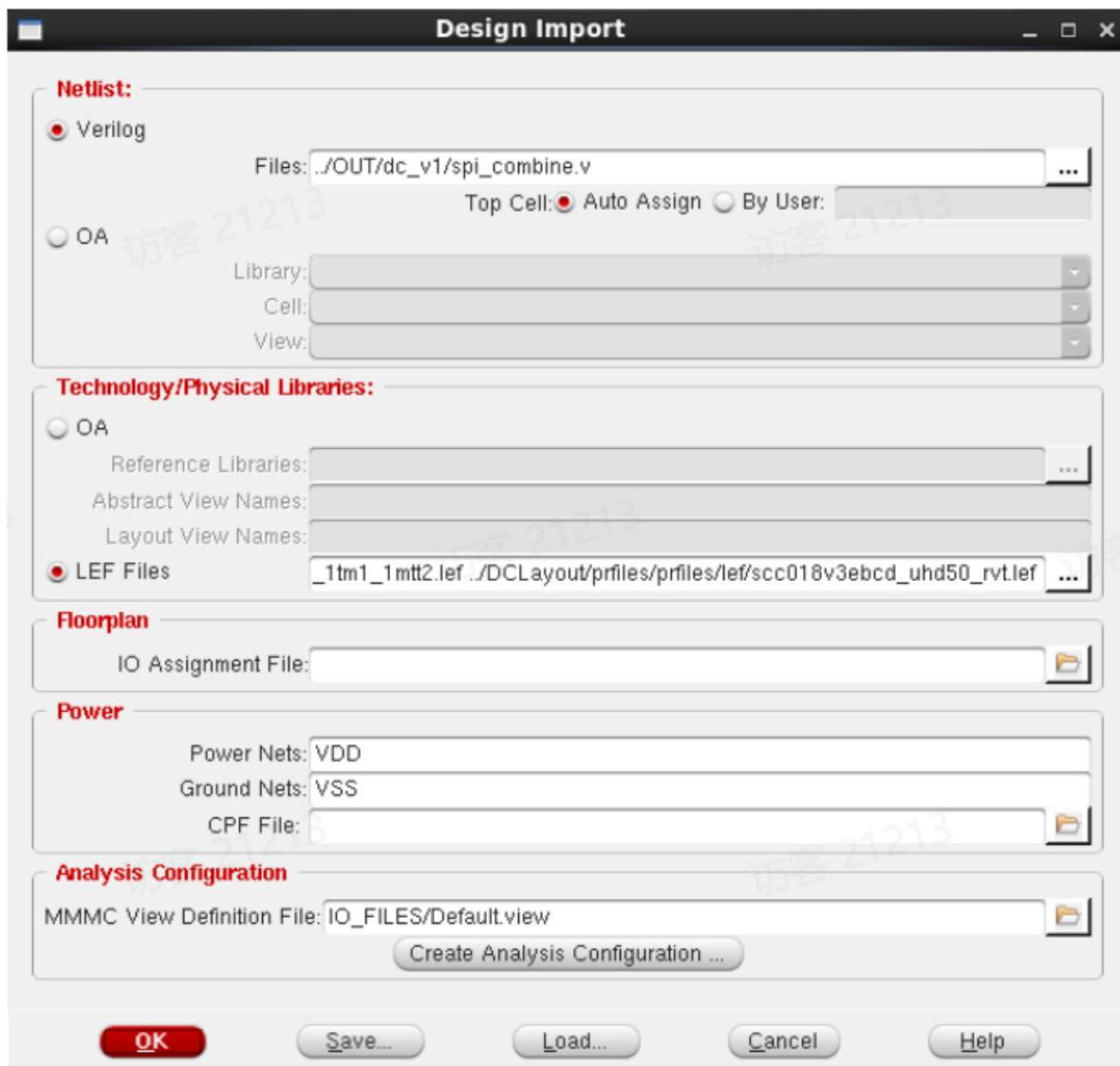
A	B	C	D	E	F	G	H	I	J	K	L	
1 宽度	185	175	IO个数	UP	BOTTOM	LEFT	RIGHT	序号	UP	BOTTOM	LEFT	RIGHT
2 间距				13	13	16	12	1	13.2143	13.2143	10.2941	13.4615
3				13.2142857	13.2142857	10.2941176	13.4615385	2	26.4286	26.4286	20.5882	26.9231
4								3	39.6429	39.6429	30.8824	40.3846
5								4	52.8571	52.8571	41.1765	53.8462
6								5	66.0714	66.0714	51.4706	67.3077
7								6	79.2857	79.2857	61.7647	80.7692
8								7	92.5000	92.5000	72.0588	94.2308
9								8	105.7143	105.7143	82.3529	107.6923
10								9	118.9286	118.9286	92.6471	121.1538
11								10	132.1429	132.1429	102.9412	134.6154
12								11	145.3571	145.3571	113.2353	148.0769
13								12	158.5714	158.5714	123.5294	161.5385
14								13	171.7857	171.7857	133.8235	
15								14			144.1176	
16								15			154.4118	
17								16			164.7059	



布局后的IO引脚位置图

5.2.3 导入设计 Design Import

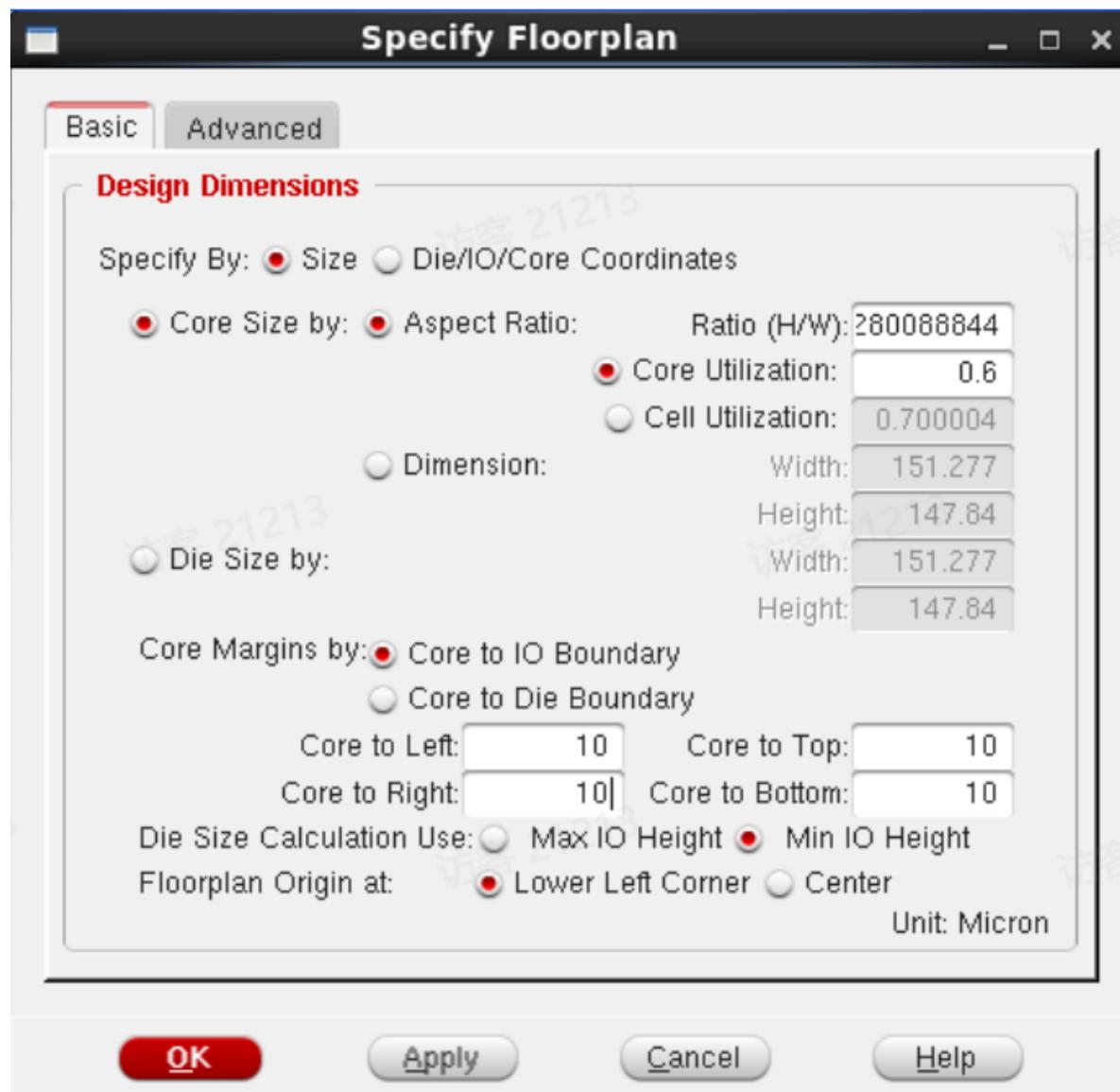
第一次导入时设置：



之后使用时，可以直接Load第一次保存的Default.globals文件。

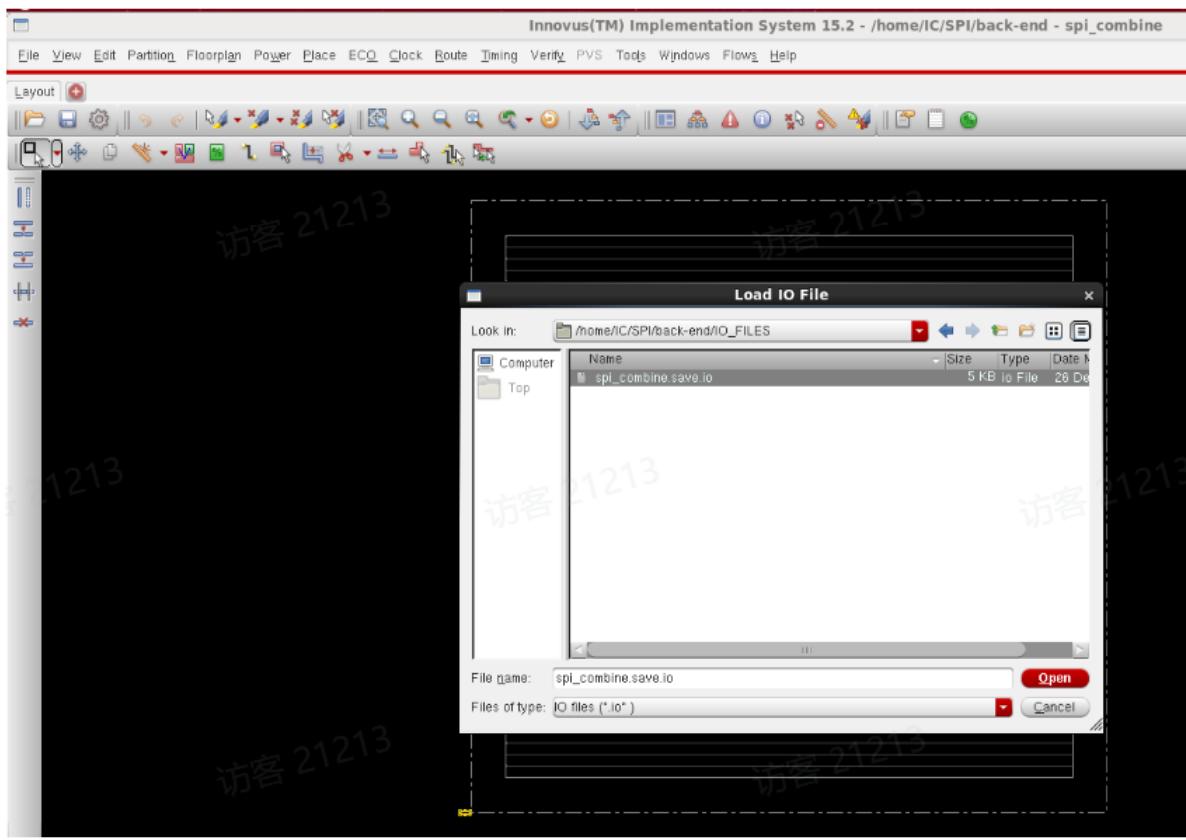
5.2.4 布局规划 FloorPlan

指对各个功能模块的排列和布局进行规划的过程。这是在芯片级别上进行的整体布局设计，旨在确定每个功能模块、逻辑单元、存储单元、输入/输出接口等元素在芯片内部的位置。floorplan的最终目标是最大程度地优化性能、功耗和面积，以及确保满足布线和物理设计方面的限制。



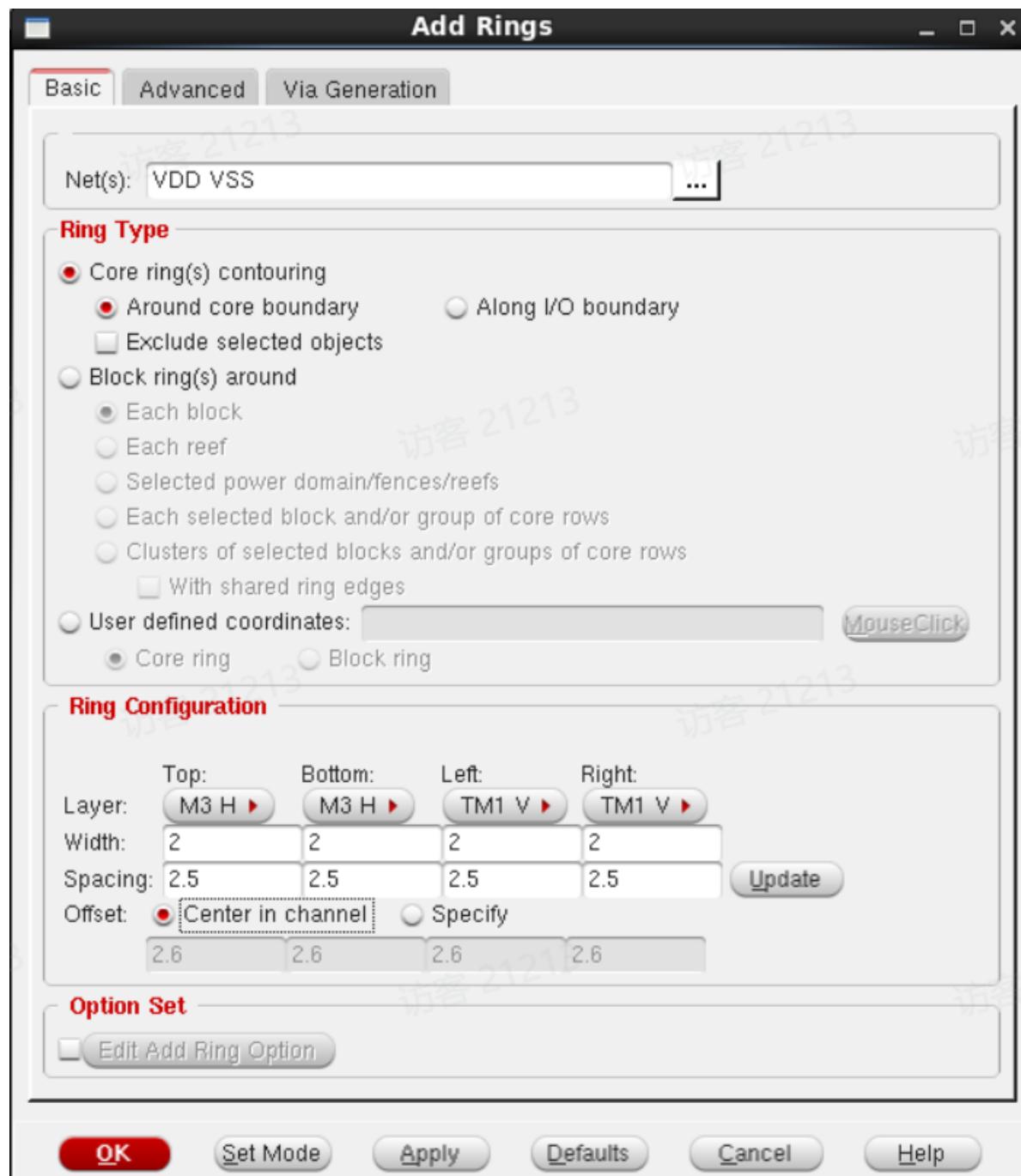
5.2.5 载入IO文件 Load IO file

包括芯片的输入/输出引脚列表、约束文件、时序信息以及其他与物理布局和布线相关的约束和要求。加载这些I/O文件是为了确保布局布线工具能够按照设计规范正确地放置和连接输入输出引脚，并满足设计的性能和功能要求。



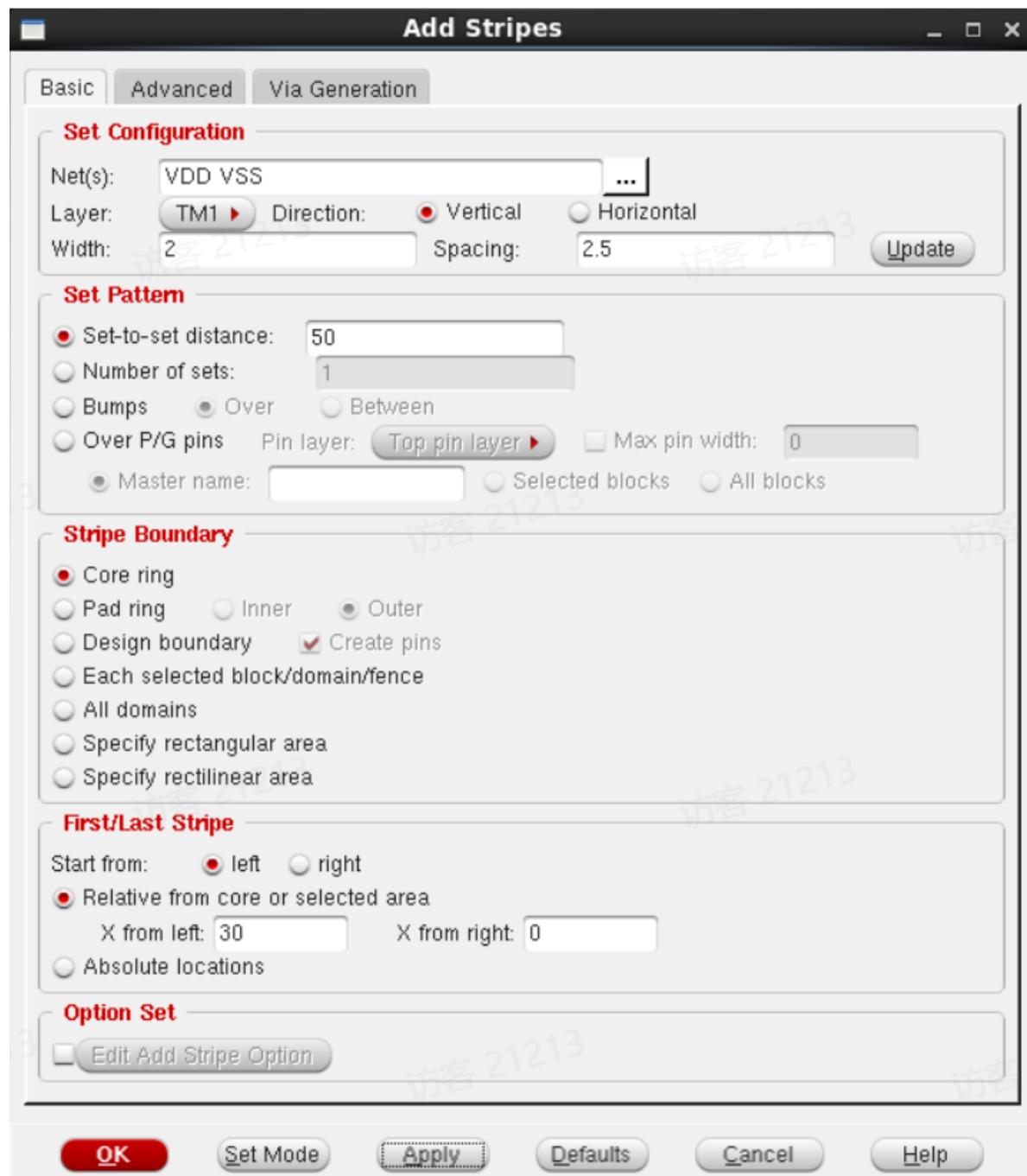
5.2.6 增加供电环 Add Power Ring

将供电网络连接在芯片的四周，以确保每个电路单元都能够得到足够的电源供应，从而提供稳定的工作环境。



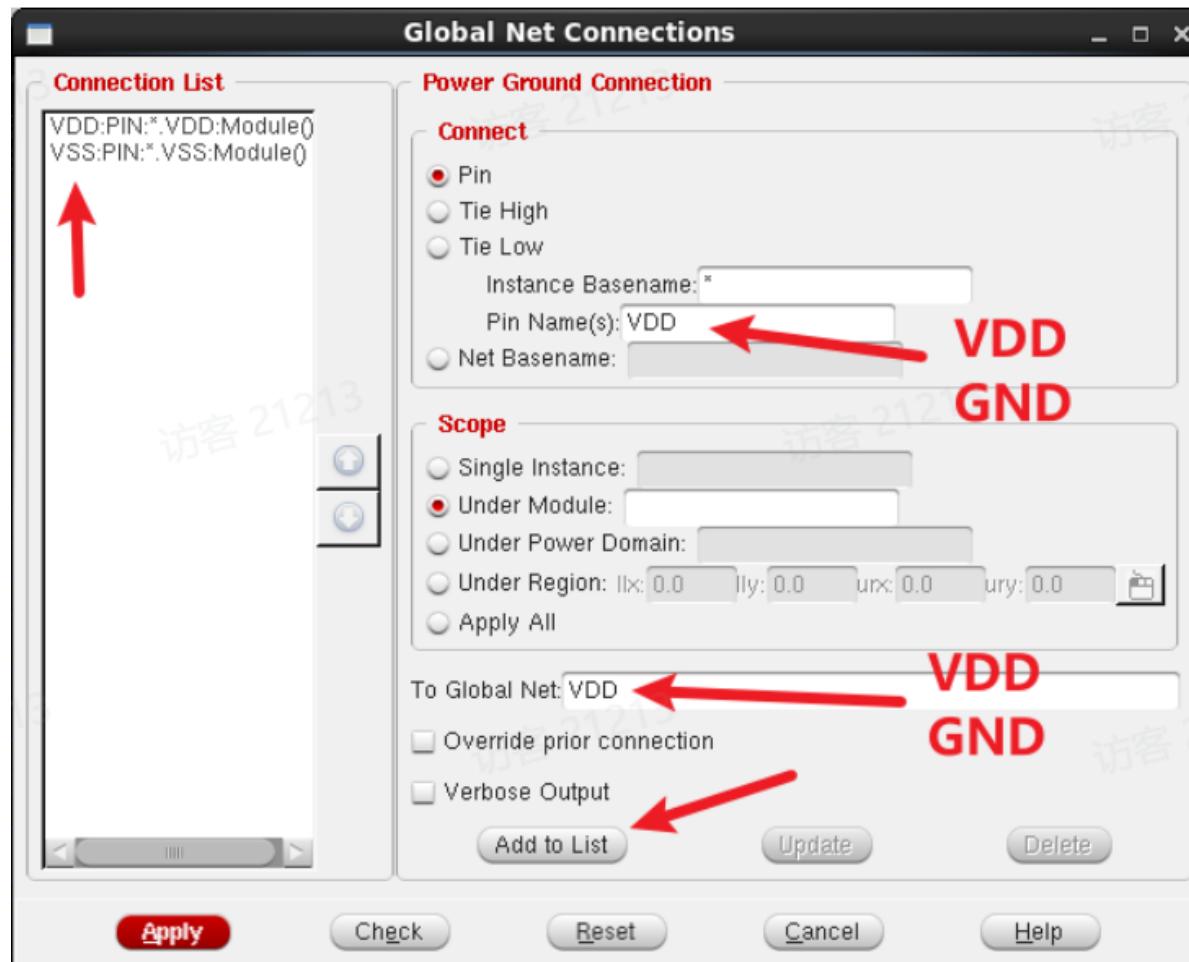
5.2.7 增加条纹 Add Stripes

在芯片的布线或者布局阶段，增加一定的布线形状以改善电气特性，减少电压下降和噪声问题。这些条纹通常被用来提高电路的传输性能和稳定性。



5.2.8 全局信号线连接 Global Net Connections

连接整个芯片的主要信号线或者电源线。这些全局连接通常包括主要的时钟信号、复位信号、供电线等，在整个芯片内部起着关键的作用。在数字芯片的后端设计阶段，需要对这些全局网连接进行布局和布线，以确保信号传输的性能和稳定性。

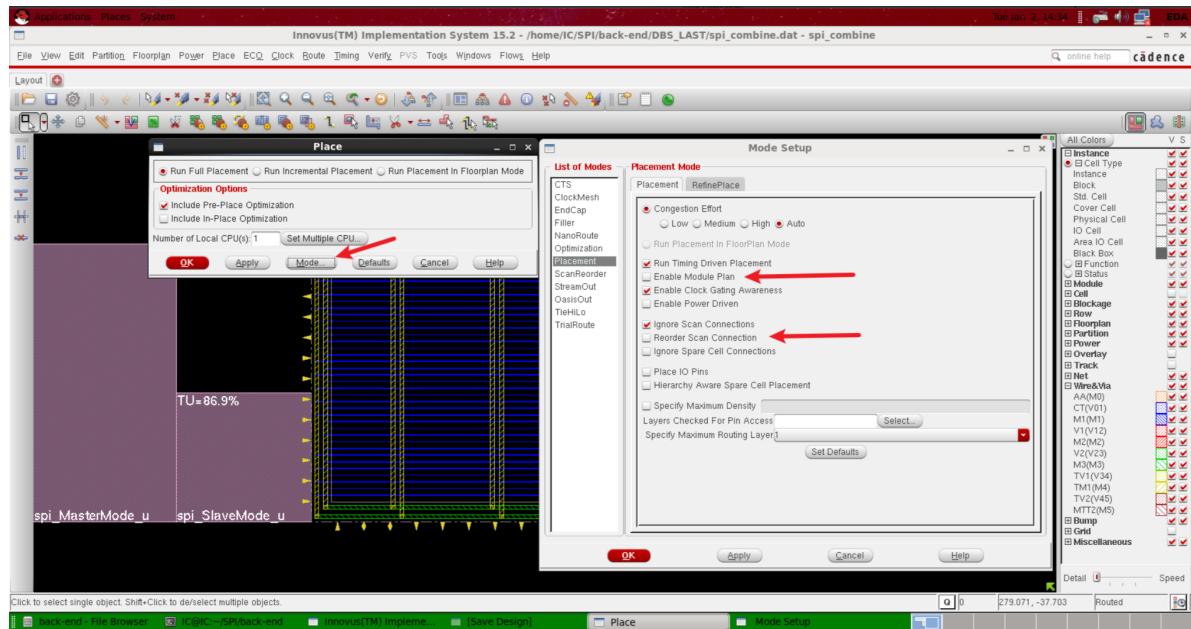


5.2.9 特殊路径 Special Route

指的是定制的电路布线路径，用于连接芯片上的特定功能模块或提供更佳的性能。这些特殊的路线可能会经过定制的布局和布线规则，以确保信号的稳定性和最佳的电路性能。



5.2.10 放置标准单元 Place Standcells



5.2.11 时钟树综合 CTS

在数字芯片设计后端的布局布线阶段，CTS (Clock Tree Synthesis) 用于设计时钟电路以确保时钟信号能够准确、稳定地传输到整个芯片各个部分。

CTS 的主要功能包括：

1. 时钟树的设计：CTS负责设计时钟树，将时钟信号从时钟源传播到整个芯片的各个触发器单元。由于时钟信号会在芯片内产生延迟和偏差，因此CTS会优化时钟树的拓扑结构和时钟信号的传输路径，以最小化时钟延迟和抖动。
2. 时钟缓冲器的插入：CTS会根据时钟信号的传输路径和延迟情况，在合适的位置插入时钟缓冲器来稳定时钟信号的传输，减少时钟抖动和延迟。
3. 时钟网络的优化：CTS会优化时钟网络的布局和布线，以减少时钟信号路径的长度和不平衡，从而降低时钟网络的波动和抖动。

因为INNOVUS不支持CTS可视化操作，在终端输入以下命令进行CTS：

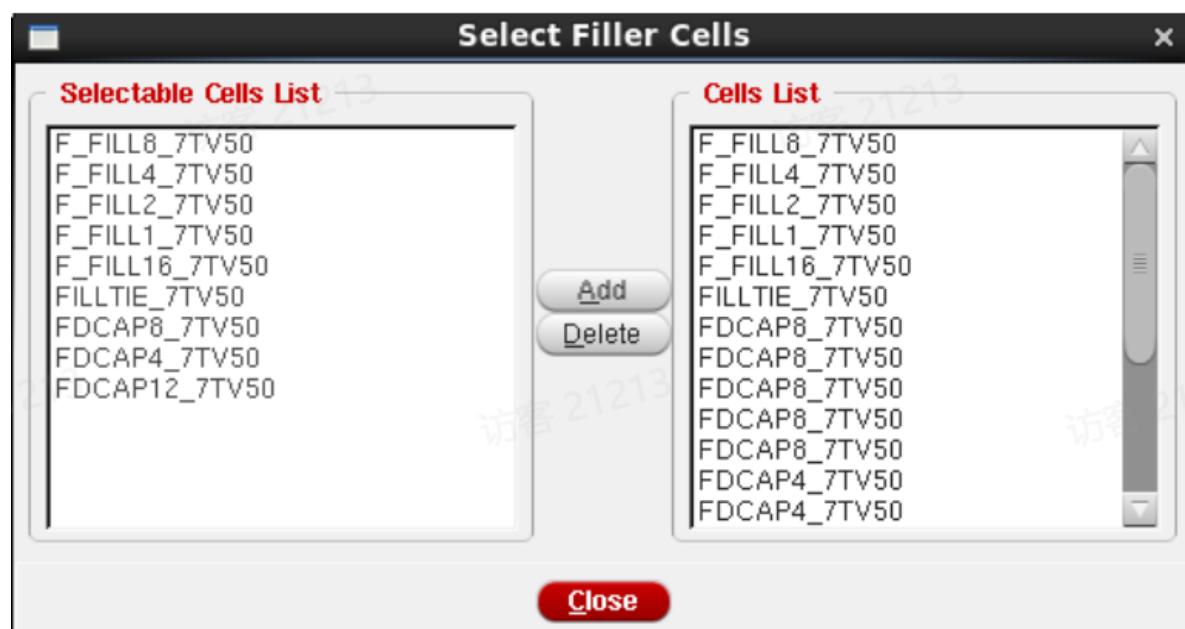
```
set BASENAME spi_combine
createClockTreeSpec -output ${BASENAME}.ctstch
setCTSMode -routeGuide true
setCTSMode -routeClkNet true
clockDesign -outDir ${BASENAME}_clock_reports
```

5.2.12 纳米布线 NanoRoute

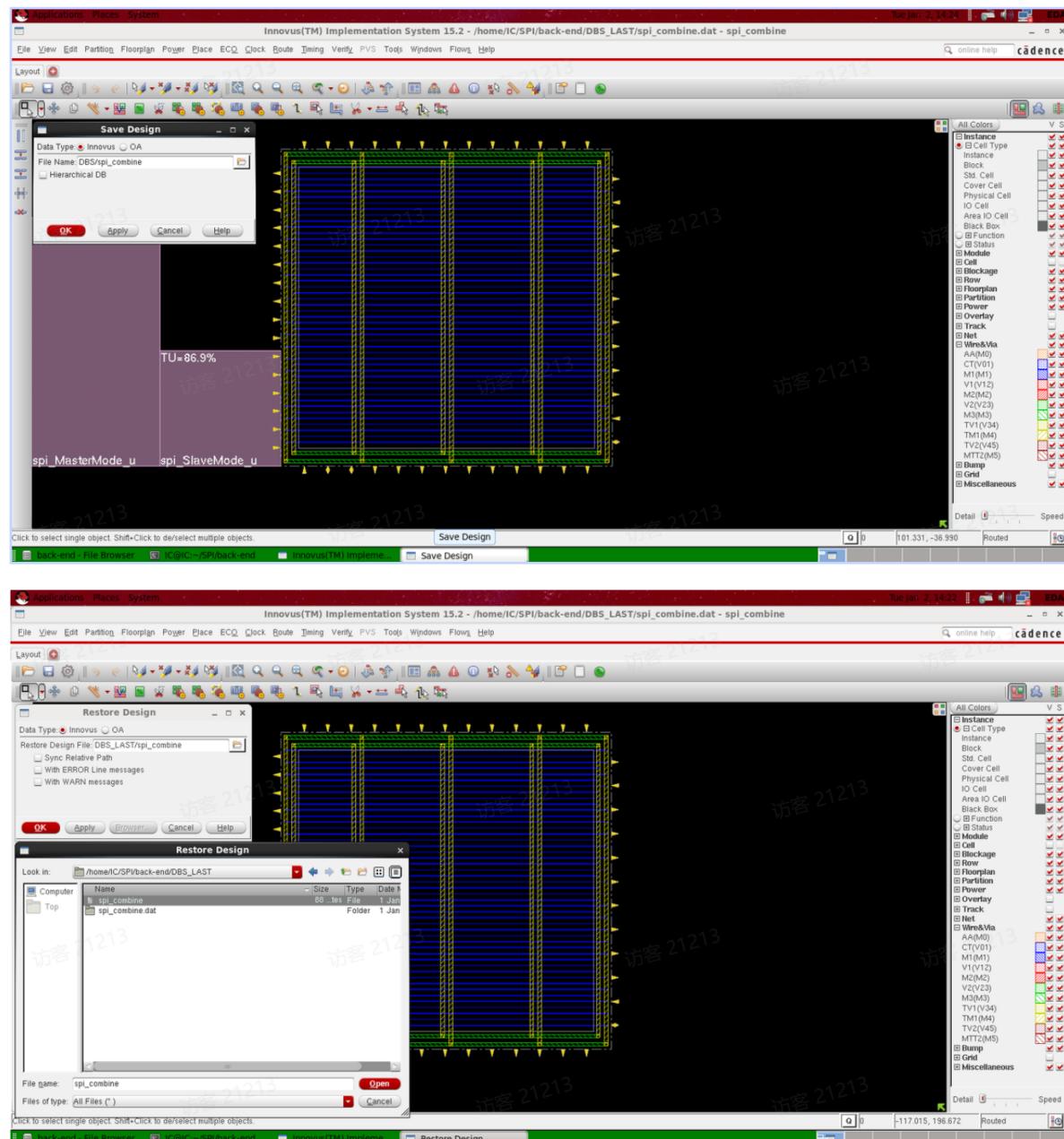


5.2.13 添加填充单元 Add Filler Cells

填补空白区域或优化布线而添加的特定类型的标准单元。这些填充单元通常用于平衡布局或解决布线规则要求。



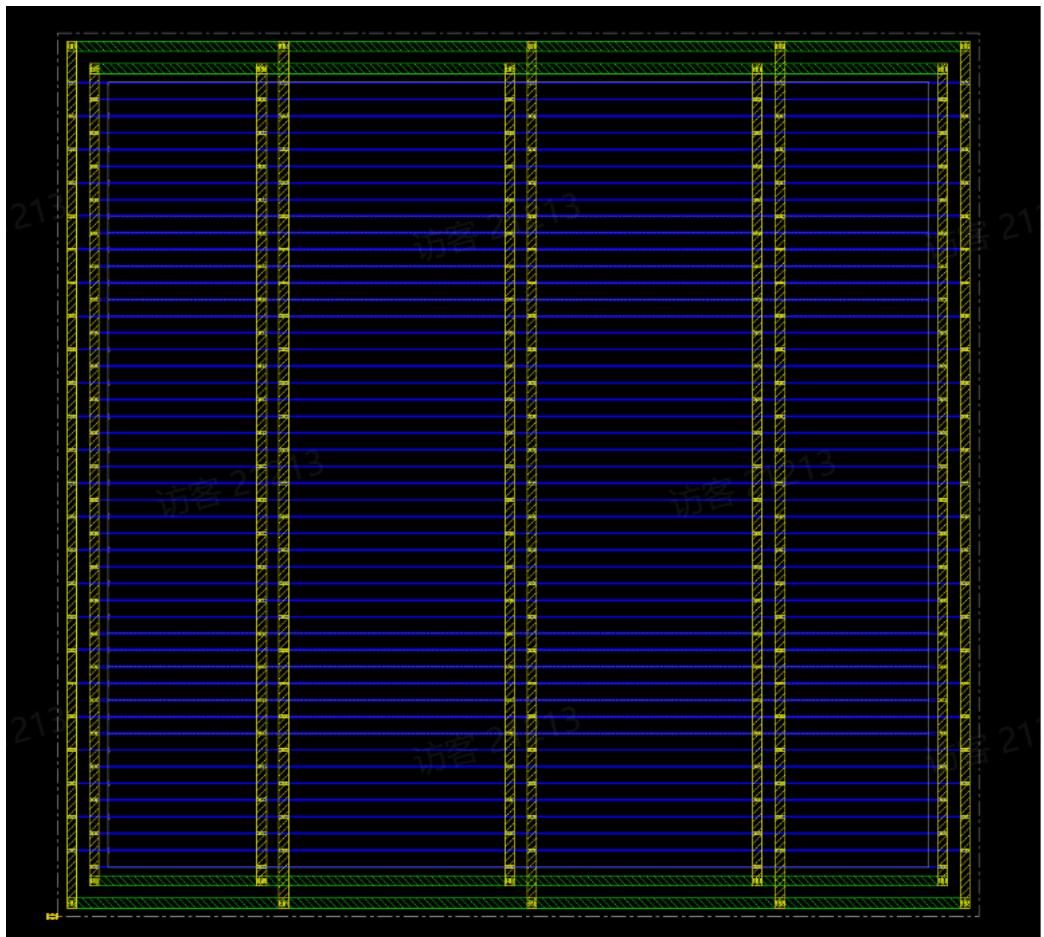
5.2.14 保存和读取设计



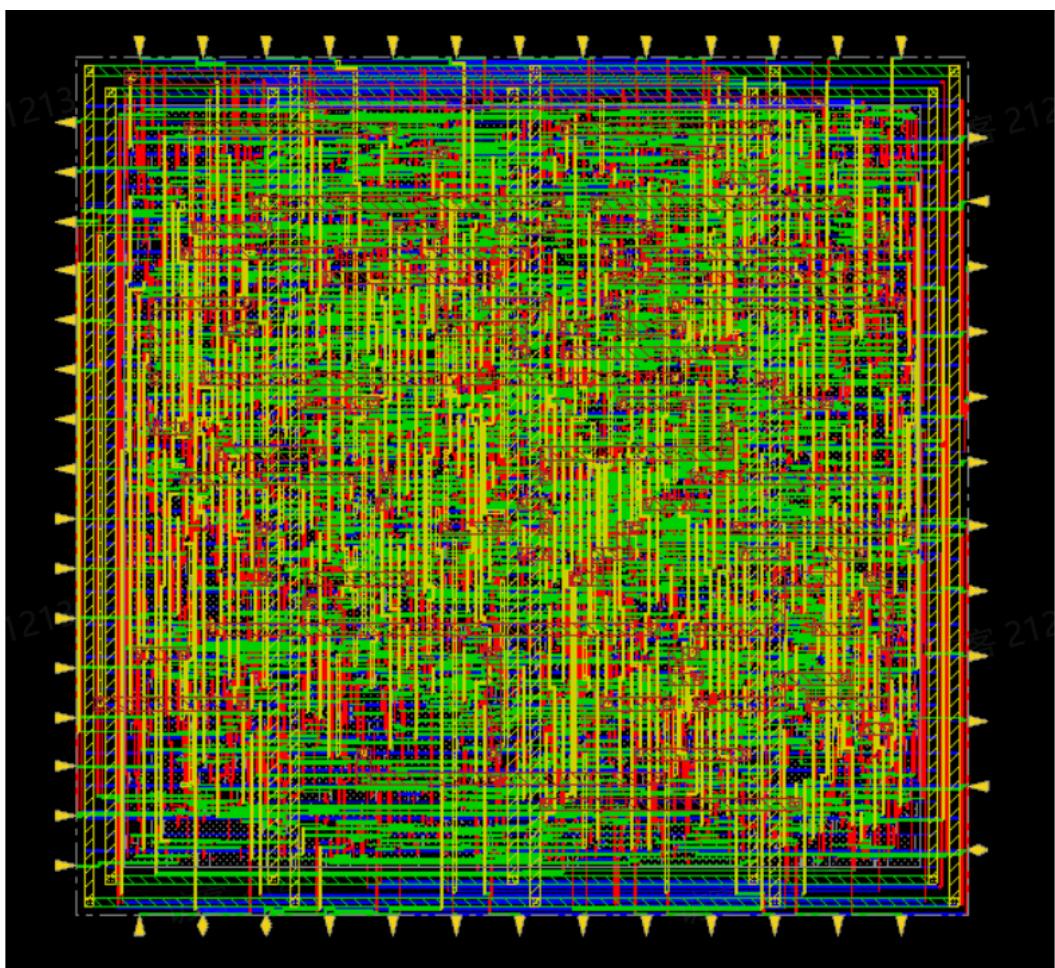
5.3 布局布线结果检查

5.3.1 版图

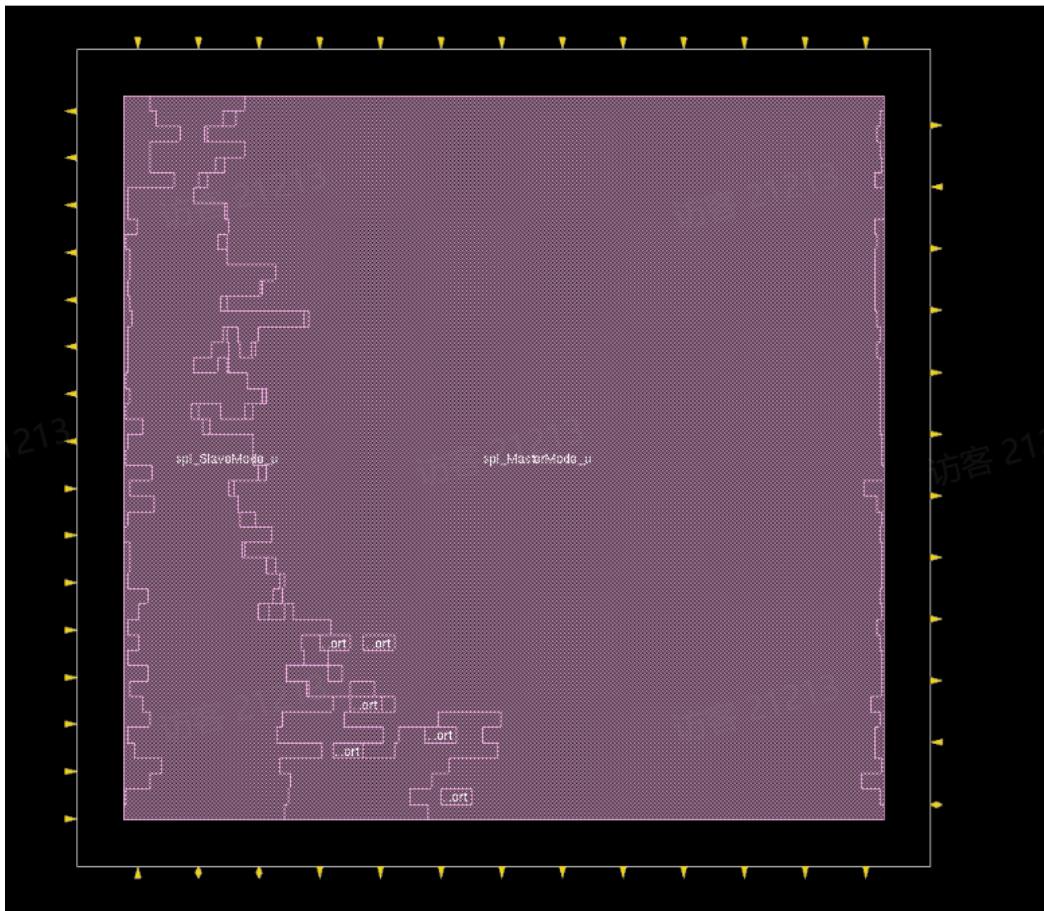
- Special Route之后版图



• 最终版图



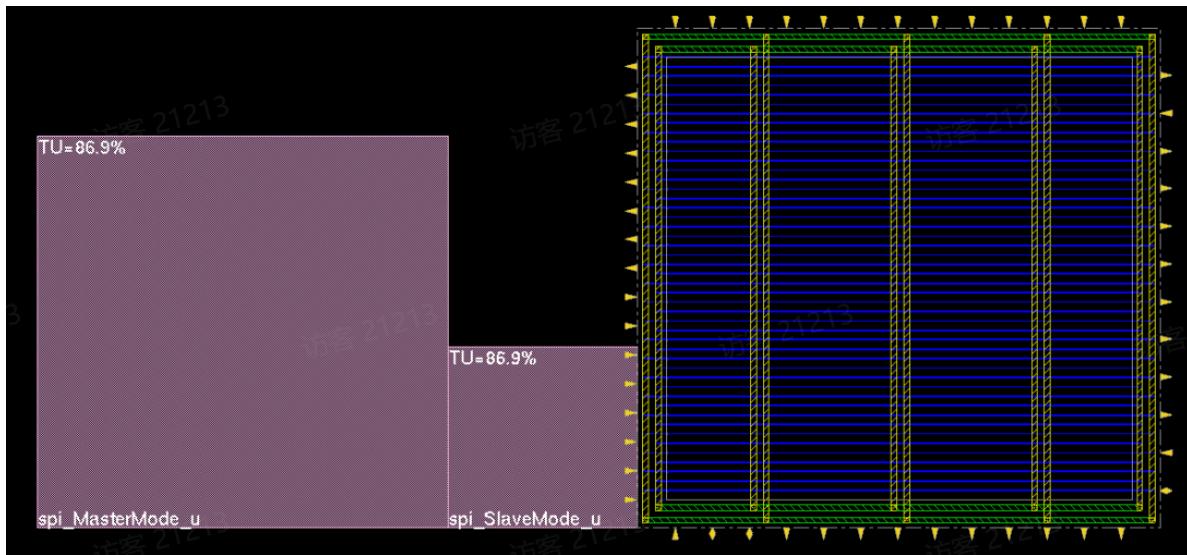
• Amoba View



- Amoba View的局部放大，可以看到三态门



- Floorplan View



5.3.2 面积利用率

面积利用率为**86.895%**。

Density: 87.376%

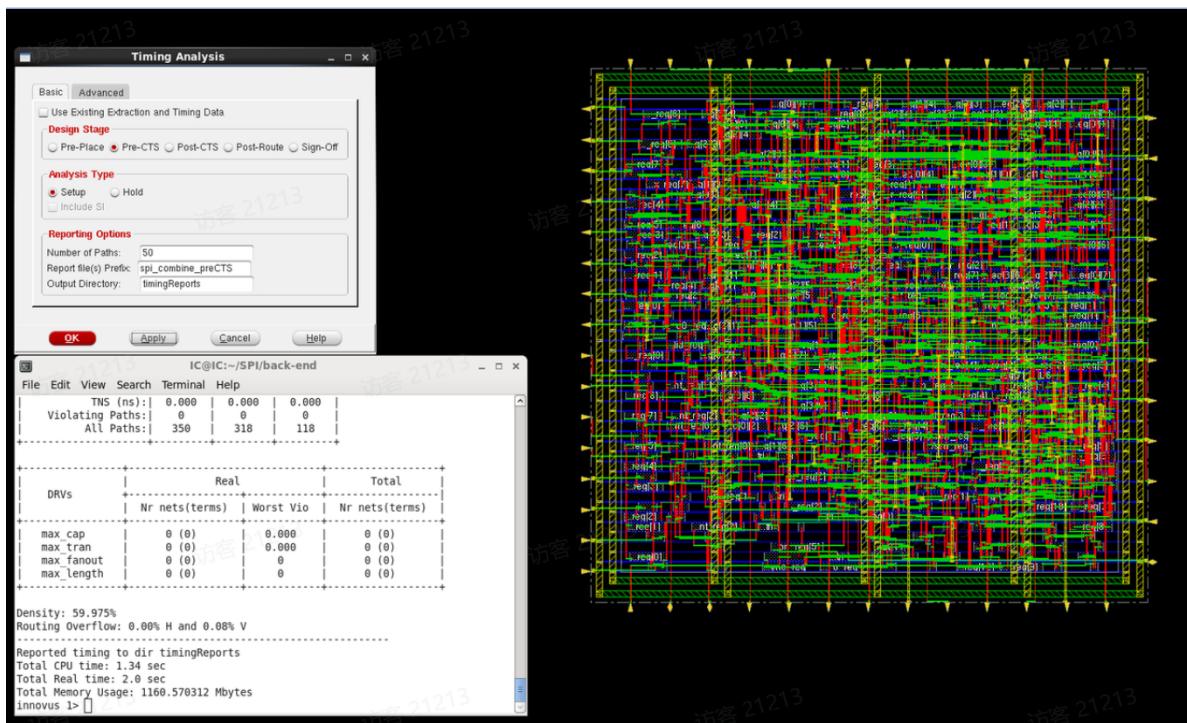
5.3.3 时序检查 Timing Check

如下表所示，Pre-CTS、Post-CTS和Post-Route三个阶段的Setup、Hold Timing检查，在做完相对应的optimize后全部通过。

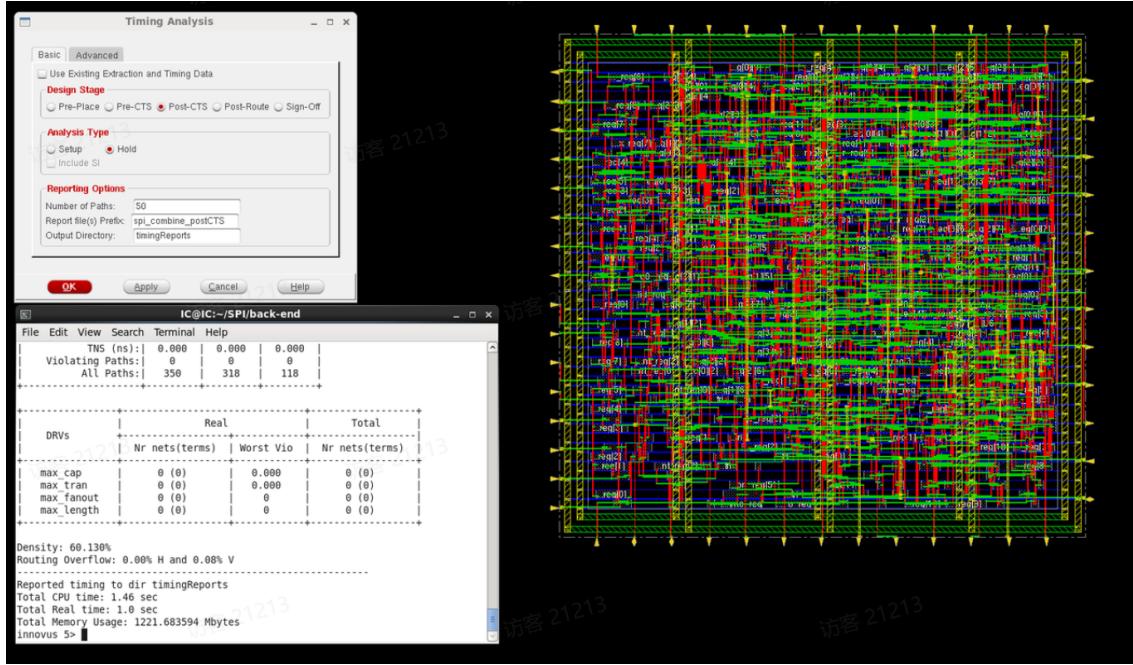
	Pre-CTS	Post-CTS	Post-Route
Setup	√	√	√
Hold	√	√	√

I. Pre-CTS Check

- optimize后的Pre-CTS-setup结果：

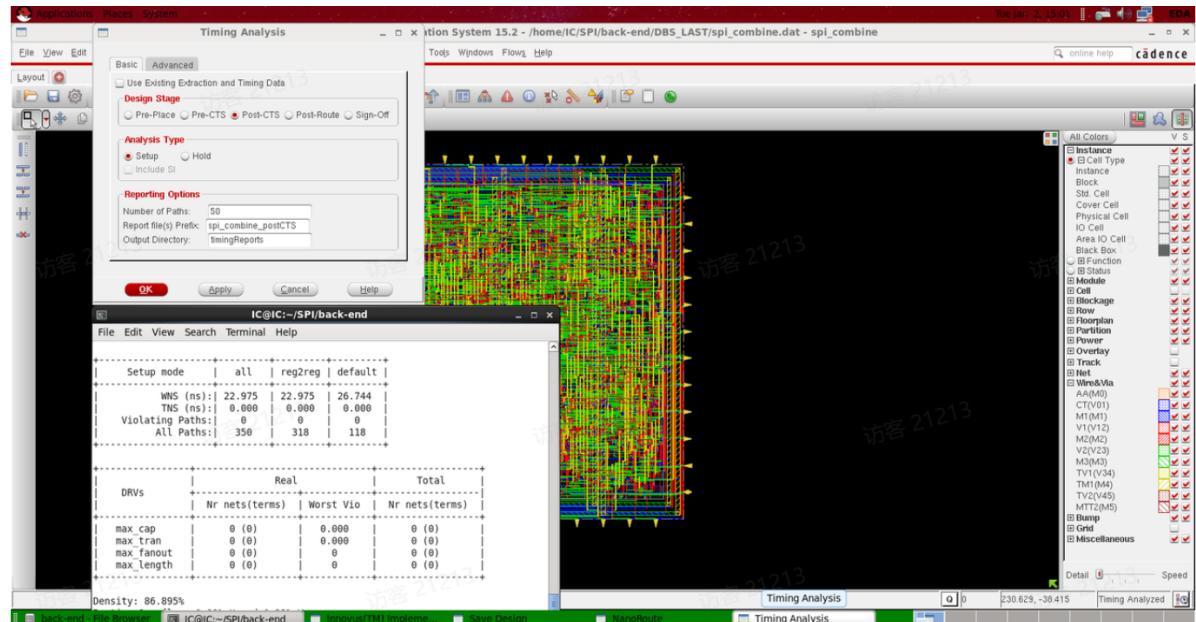


- optimize后的Pre-CTS-hold结果:

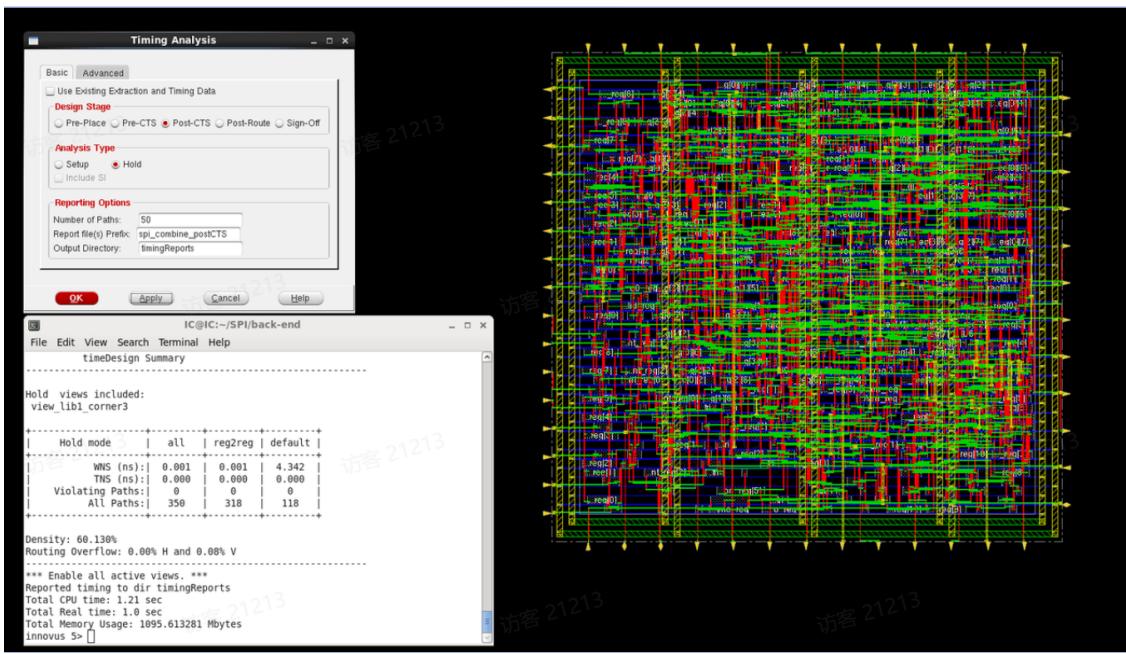


II. Post-CTS Check

- optimize后的Post-CTS-setup结果:

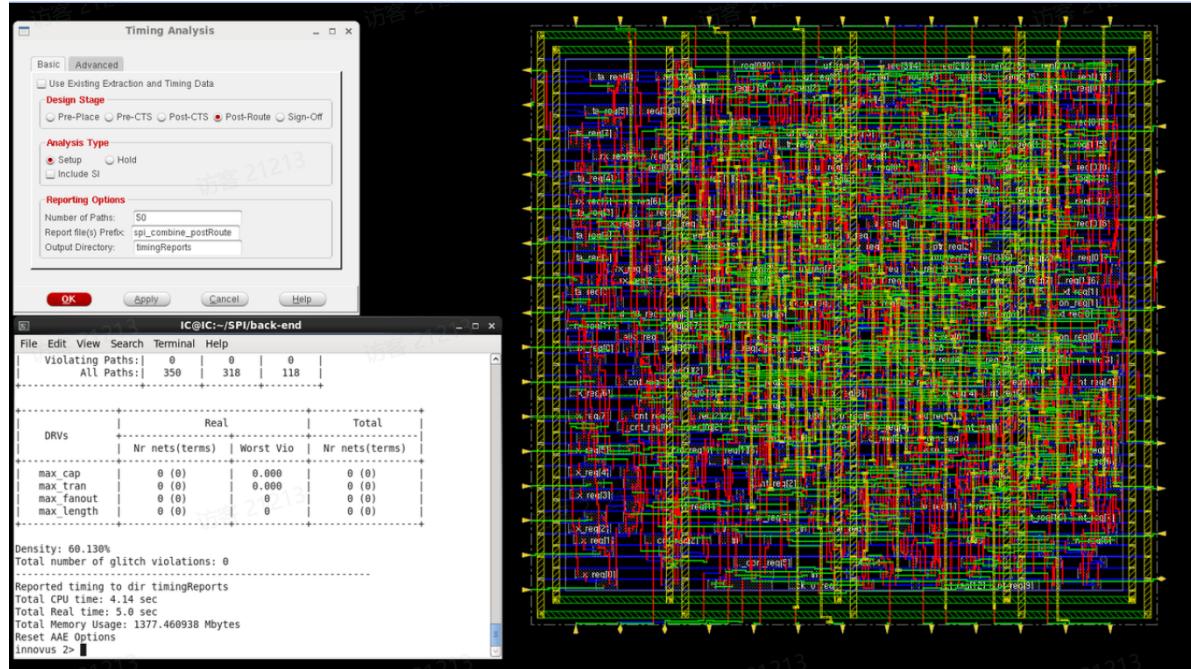


- optimize后的Post-CTS-hold结果:

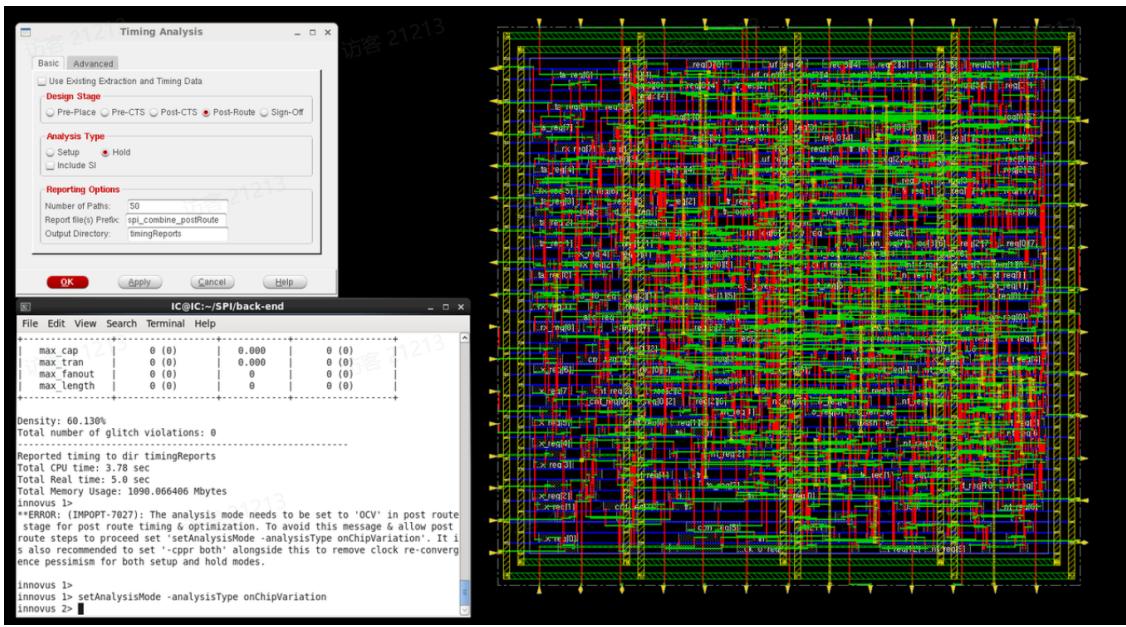


III. Post-Route Check

- optimize后的Post-Route-setup结果:



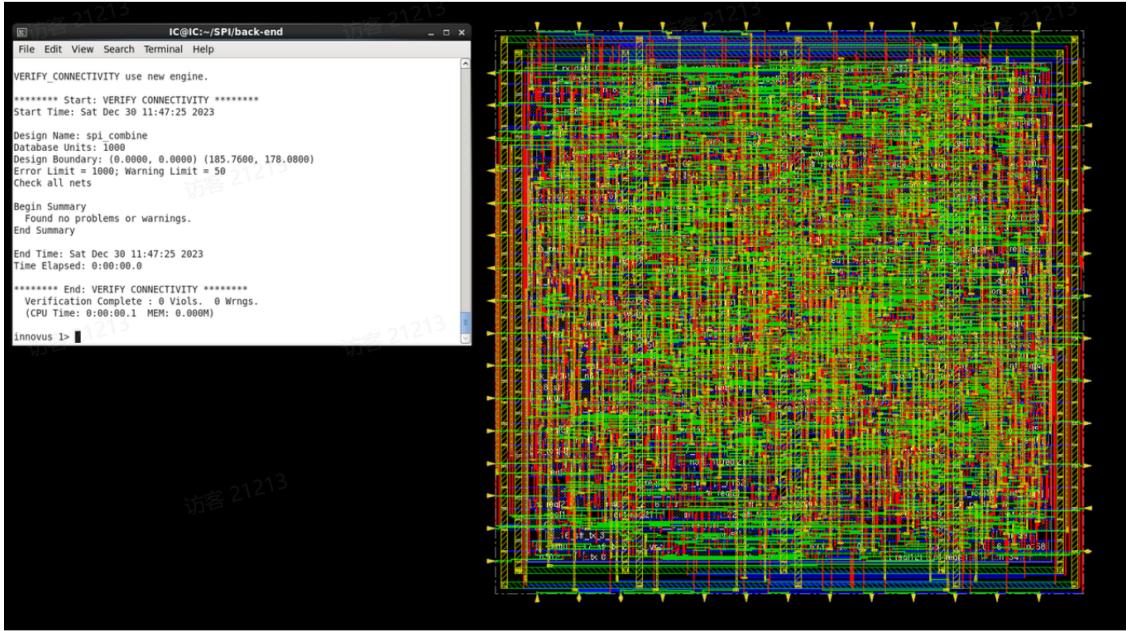
- optimize后的Post-Route-hold结果:



5.3.4 结果验证

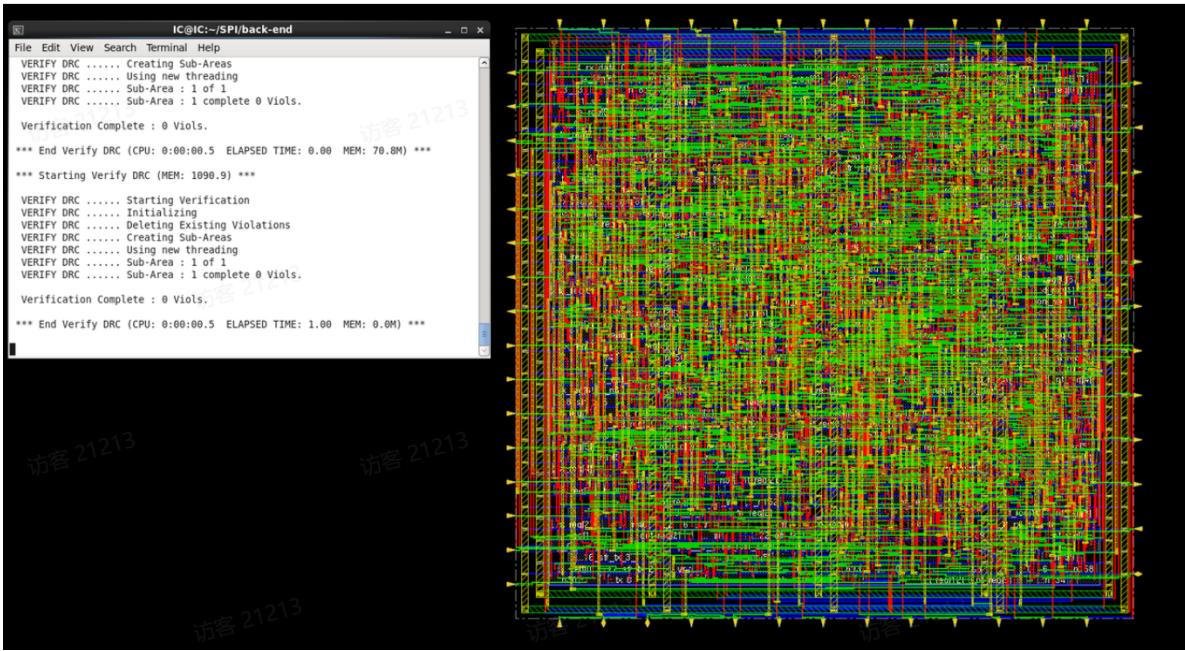
I. Connectivity

- Verify Connectivity 通过:



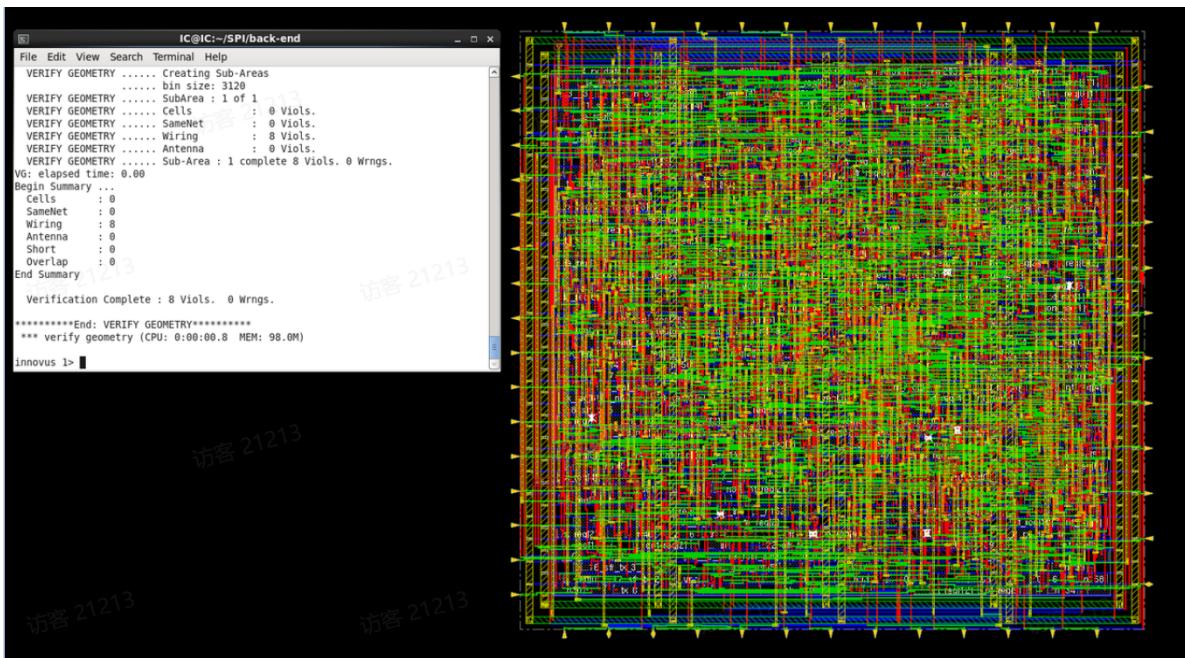
II. DRC

- Verify DRC 通过:



III. Geometry

- Verify Geometry 存在数字电路的常见报错。 (日志文件中显示为8个Wiring错误，但是实际上可能更多)



6. 布局布线后仿真

(在POST_SIM - AfterPR内)

6.1 tb中反标 布局布线输出的 .sdf文件

```
initial begin
`ifdef LAYOUT_SIM
    $sdf_annotation("../PR/Output_Files/spi_combine.sdf", spi_combine);
`endif
end
```

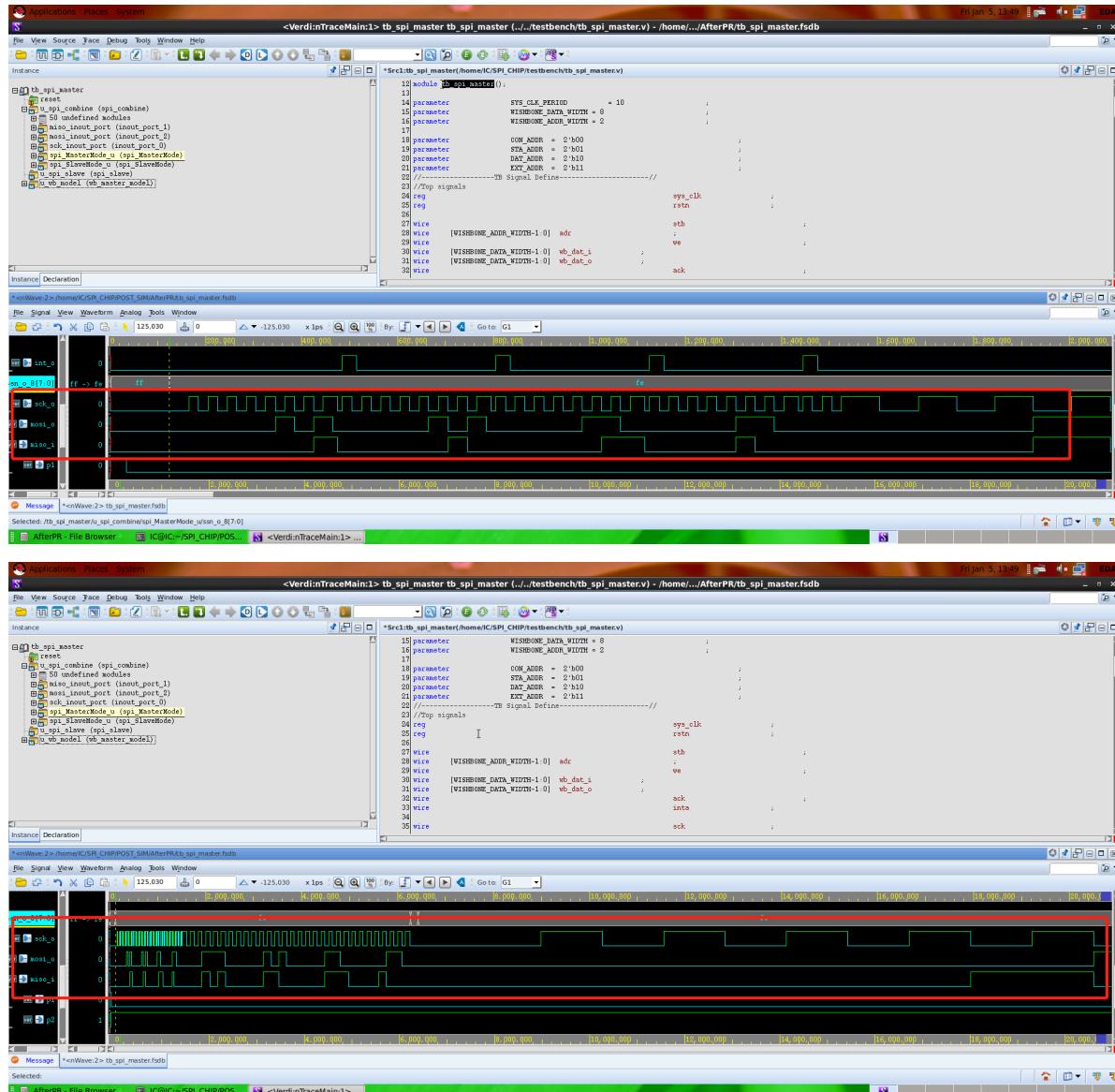
6.2 layout_filelist.f

```
# layout netlist
../../PR/Output_Files/spi_combine.v

# tb
../../testbench/spi_slave.v
../../testbench/tb_spi_master.v
../../testbench/wb_master_model.v
../../testbench/tb_spi_slave.v
```

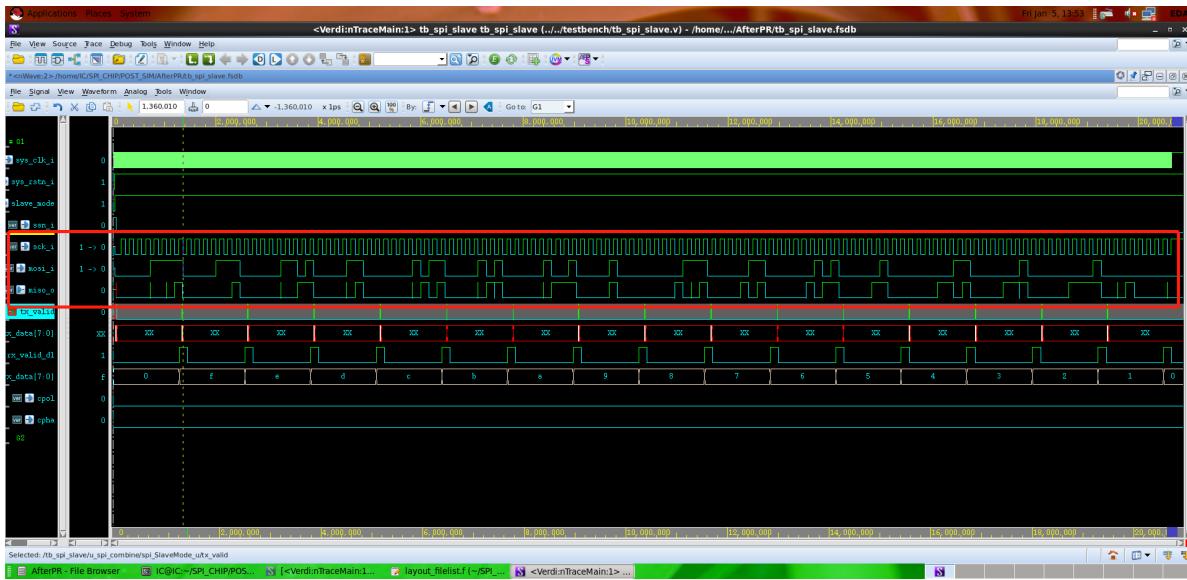
6.3 主模式仿真结果

- 与综合后网表仿真和功能仿真均一致：



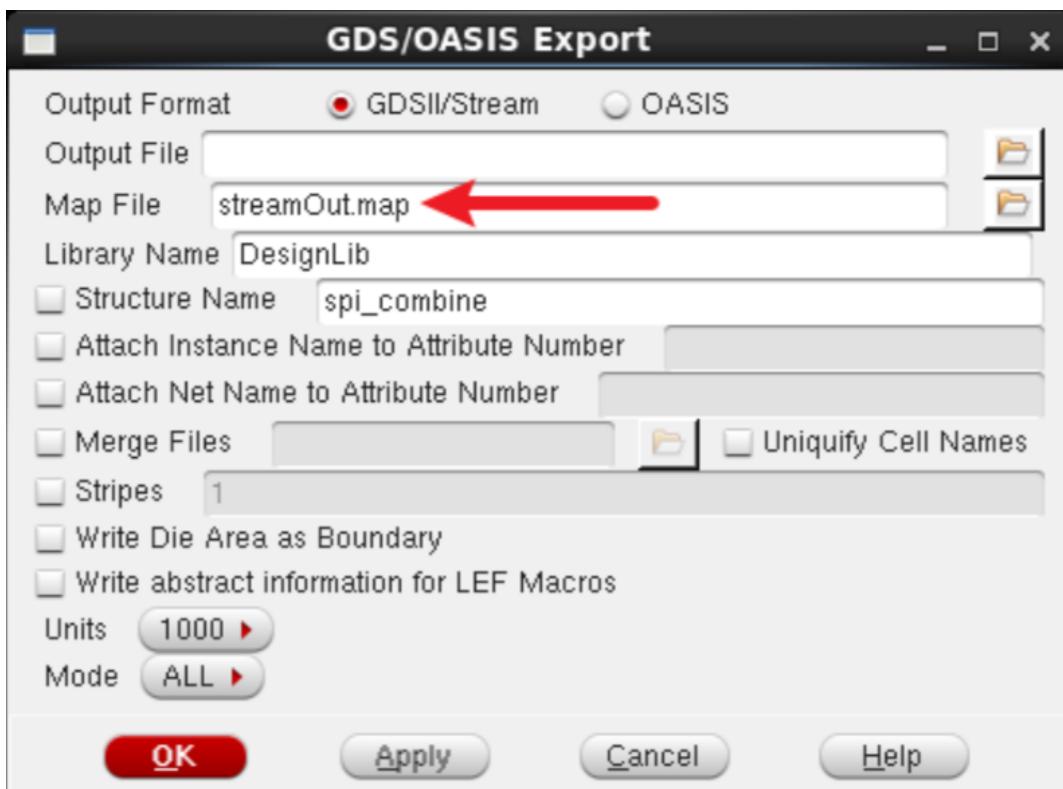
6.4 从模式仿真结果

- 与综合网表仿真及前仿一致：

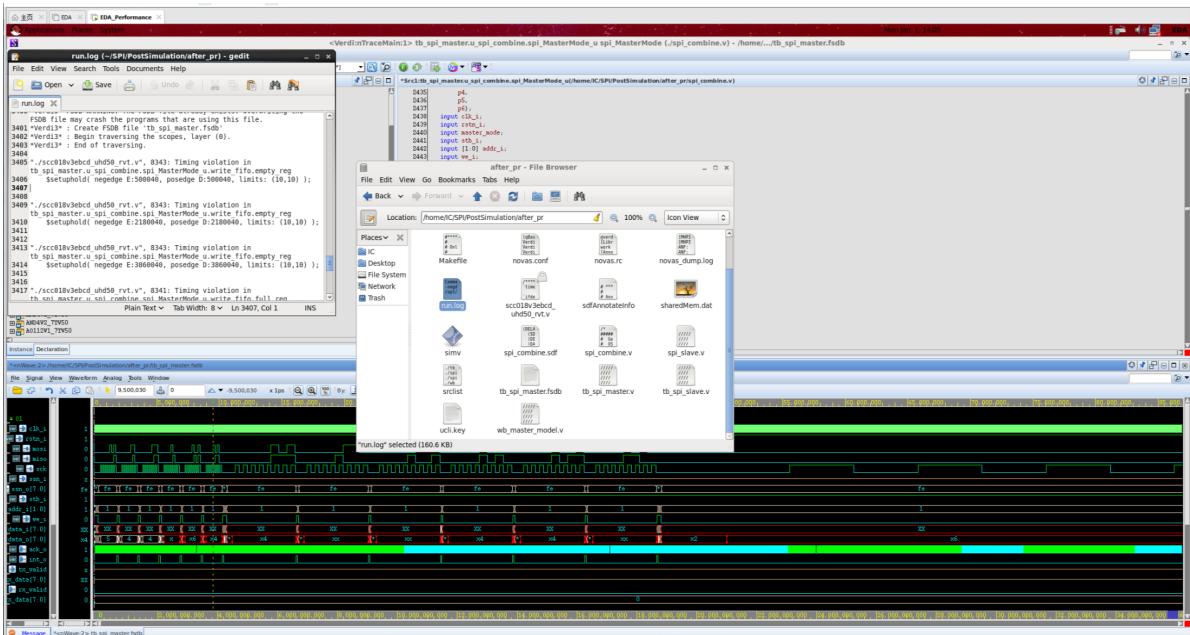


7. 目前存在的问题

① 导出GDSII文件时如何配置Map File等文件



② 布局布线后的网表进行仿真时有报时序违例



③VCS+Verdi跑不了前仿

