平头哥

S802 集成手册

INTEGRATION GUIDE

Number	
Revision	1.3.5
Security	Public
Date	2017-06-27

平头哥半导体有限公司 T-HEAD Semiconductor Co.,LTD

地址: 杭州市西湖区西斗门路 3 号天堂软件园 A 座 15 楼 邮编: 310012

网址: www.c-sky.com

版本历史:

版本	日期	描述	作者
1.0	2010-1-21	1. 描述包括：功能、接口、时钟、复位、同步、低功耗、中断与 DFT。	平头哥半导体有限公司
1.1	2014-3-12	1. 删除 DFT，修改中断以及接口信号	平头哥半导体有限公司
1.2	2014-8-22	1. 修改接口信号，修改低功耗唤醒机制和 VIC 的中断时序。	平头哥半导体有限公司
1.3	2014-12-17	1. 增加 scan 测试和 wrapper 测试	平头哥半导体有限公司
1.3.1	2015-06-22	1. 修改时钟接口相关说明	平头哥半导体有限公司
1.3.2	2015-07-31	1. 增加异常重定向和 tclk 唤醒相关信号	平头哥半导体有限公司
1.3.3	2016-06-30	1. 修改接口信号说明	平头哥半导体有限公司
1.3.4	2017-04-10	2. 调整文档结构 3. 增加工艺映射章节 4. 重写低功耗相关内容 5. 增加 bist 相关内容 6. 增加观测信号相关说明 7. 增加接口时序 8. 增加中断没 VIC 的说明 9. 删除 DLITE 相关内容（无该配置了）	平头哥半导体有限公司
1.3.5	2017-06-27	1. 对复位信号进行说明	平头哥半导体有限公司

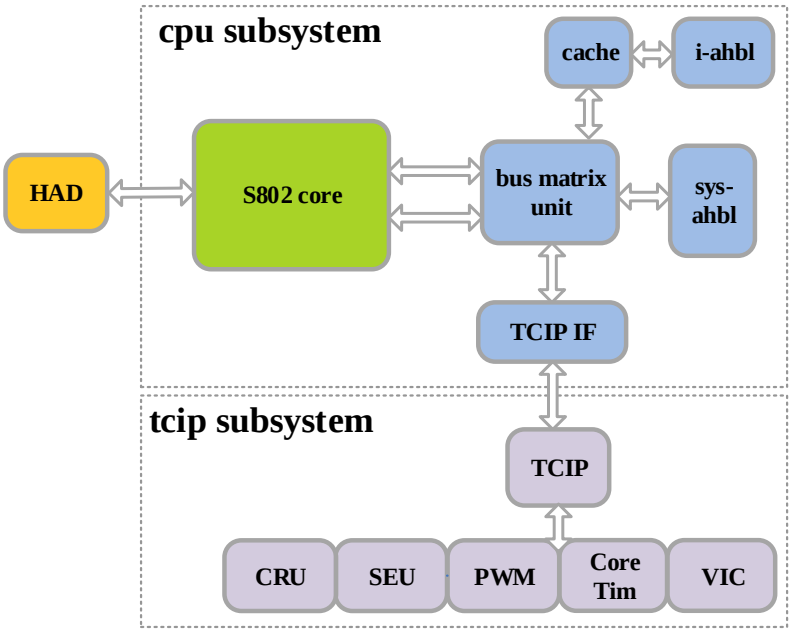
目录

图表目录

1 概述

1.1 处理器简介

S802 是平头哥半导体有限公司自主设计的面向极低功耗应用的 32 位嵌入式 CPU IP 核。S802 采用自主设计的 CSKY_V2 指令系统微体系结构，并具有可扩展指令、可配置硬件资源、可重新综合、易于集成等优点，特别针对低功耗设计和电源管理进行了优化。



图表 1-1 S802 系统框图

S802 的主要技术特征为：

- RISC 精简指令集结构；
- 32 位数据，16/32 位可变长度指令；
- 2 级流水线；
- 支持可配置的快速退休机制；
- 单周期指令和数据存储访问；
- 无延时的分支跳转；
- 支持硬件可配置内存保护区域（0-8）；
- 支持 AHB-Lite 总线协议；
- 支持硬件可配置的指令 AHB-Lite 总线接口；
- 支持总线接口多种 CPU 到总线时钟比率；
- 支持大端（big endian）和小端（little endian）；
- 支持硬件调试；
- 支持可配置的二进制转译加速机制；

- 支持可配置的高速缓存器（Cache），高速缓存容量 2KB、4KB 和 8KB 硬件可配；
- 支持单个 CPU 时钟周期访问的紧耦合 IP（TCIP）：
  - 支持可配置的信息安全增强单元（SEU），提升 CPU 在时间攻击、功耗分析攻击、错误注入和缓冲区溢出等攻击方式的防护能力；
  - 支持可配置的功耗管理单元（PWM），有效控制 CPU 的峰值功耗和平均功耗；
  - 支持可配置的 24 位循环递减的系统计时器（CoreTim）；
  - 支持可配置的矢量中断控制器（VIC），支持 1~32 个中断源硬件可配，支持中断嵌套，支持电平和脉冲中断；
  - 支持可配置的高速缓存控制寄存器单元（CRU），提供 Cache 的使能和高速缓存区的配置。
  - 支持地址观测异常相关设置

1.2 可配置选项

S802 可配置选项如下表所示：

图表 1-2 S802 可配置选项

可配置单元	配置选项	详细
硬件乘法器	无 /有	若配置则 1 个周期产生乘法结果，否则要 3-34 周期完成。
内存保护单元	0 到 8 个表项	可以配置为 0-8 个表项，其中 0 表示不实现内存保护单元。
高速缓存器	无 /2K /4K /8K	可以配置为 2KB、4KB、8KB。
可信防护技术	无 /有	配置该技术，结合中天微公司的 SoC 平台技术/系统软件，将提供系统的安全防护功能。
安全抗攻击技术	无 /有	配置该技术，将提供针对非侵入式/侵入式硬件攻击的防护
指令总线	无 /Flop-out /Non-Flop-out	在配置了指令总线的情况下，又支持寄存器输出（Flop-out）和直接输出（Non-Flop-out）两种方式。

系统总线	兼容 AHB/ 兼容 AHB Lite	可以配置为兼容 AHB 协议或者兼容 AHB Lite 协议。
矢量中断控制器	无 /INT16 /INT32	支持硬件中断的嵌套处理。支持 16 个中断源、32 个中断源。
系统计时器	无/有	用于计时。

1.3 处理器集成总览

S802 的集成按照功能可以分为以下七大系统，可以按照功能逐个系统进行集成：

1) 时钟复位信号集成

时钟复位信号集成主要介绍了 S802 处理器中 CPU 核与调试单元（HAD）的输入时钟以及复位信号；同时介绍了 CPU 核与 HAD 以及系统总线之间信号同步逻辑。

2) 总线系统集成

总线系统集成主要介绍了和配置相关的 2 个总线接口，包括：系统总线接口和指令总线接口。

3) 中断系统集成

中断系统集成主要介绍了与 S802 处理器中断处理相关的信号以及信号握手机制。

4) 调试系统集成

调试系统集成主要介绍了与调试相关的 JTAG 接口信号和其它调试辅助信号接口。

5) 低功耗系统集成

低功耗系统集成主要介绍了 S802 低功耗相关的接口信号以及在 S802 提供的三种低功耗模式(WAIT,DOZE,STOP)与正常工作模式的转换关系，具体包括了：CPU 进入和退出低功耗模式时的状态转换，进入/退出低功耗相关信号的握手时序以及不同场景下低功耗唤醒的流程。

6) DFT 系统集成

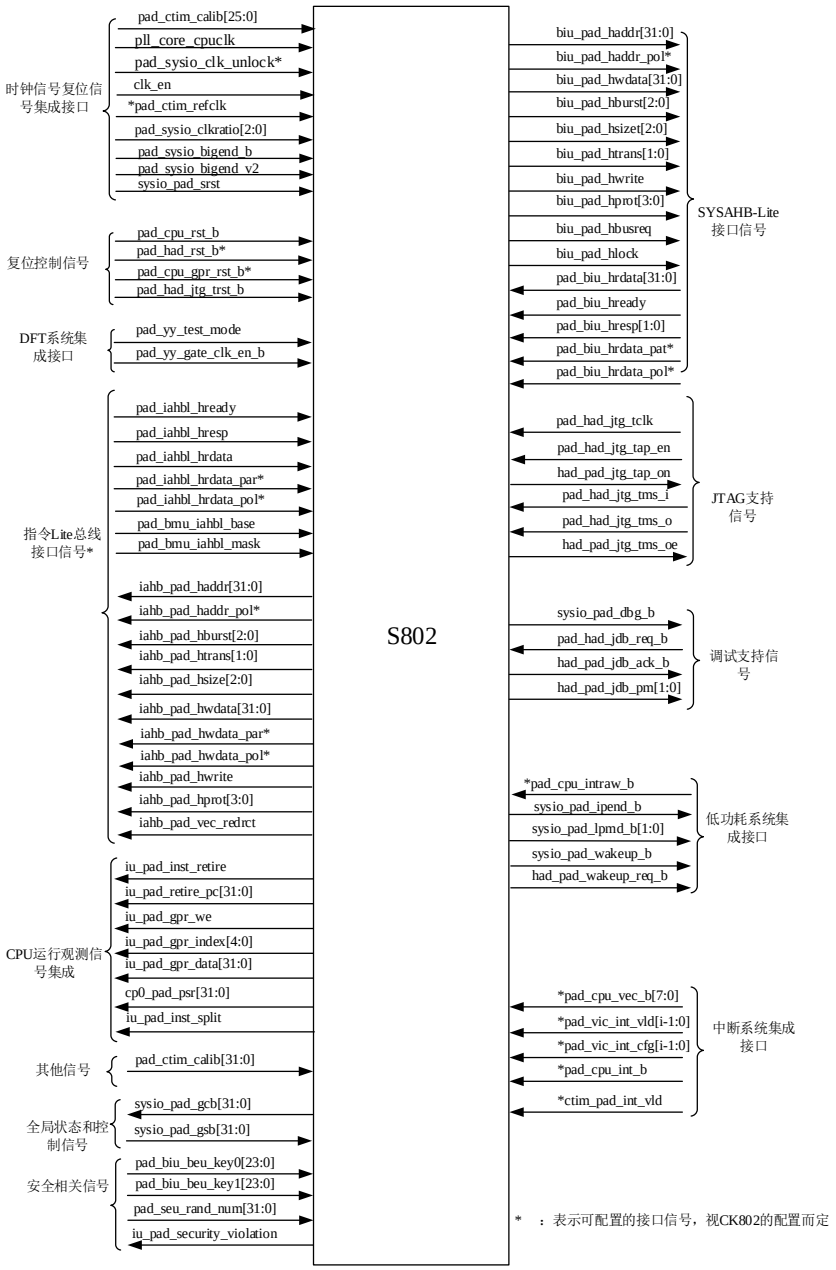
DFT 系统集成主要介绍了与测试相关的内容，包括扫描链测试和内建自测试相关的接口信号以及说明。

7) CPU 运行观测信号集成

CPU 运行观测信号集成介绍了提供给 SOC 仿真的观测信号。

2 端口信号总览

S802 处理器支持 AHB_Lite 总线接口协议。为了降低 S802 的硬件集成复杂度，S802 除了总线接口部分的信号外，其他信号在 S802 的多款处理器中，均保持了稳定的延续性，以方便 S802 的升级换代。根据 S802 顶层端口信号的特点，划分为时钟复位信号，总线系统信号，中断系统信号，调试系统信号，低功耗系统信号，DFT 系统信号，CPU 运行观测信号等几大类。



图表 2-1 S802 系列 CPU 系统接口总体描述

2.1 命名规则

pad_biu_* pad_had_* pad_iahbl_* pad_dahbl_* pad_sysio_* pad_vic_*	输入信号；
biu_pad_* had_pad_* iahbl_pad_* dahbl_pad_* iu_pad_* sysio_pad_*	输出信号；
*_b	低电平有效信号；

图表 2-2 信号命名规则

一般来说，没有特殊规定，S802 的输入输出信号均为高电平有效，但是注意，如果是以“_b”结尾的信号，则是低电平有效。

2.2 端口信号列表

信号名	I/O	Reset	时钟域	内部门级数	功能描述
时钟复位信号集成：					
状态和时钟信号： 注意：*表示当 CPU 没有配置低功耗时钟（TWO_CLK），该信号不存在。					
pll_core_cpuclk	I	-	-	< 5	CPU 工作时钟信号： 提供 CPU 工作的时钟。
pad_sysio_clk_unlock*	I	-	-	< 5	CPU 全局门控时钟使能信号： 控制 CPU 全局门控时钟单元的开启和关闭。

clk_en	I	-	CPU	< 5	CPU 同步时钟信号: 当为 1 时，CPU 时钟可以采样总线时钟的信号 注意：要求该信号不受低功耗影响，CPU 在低功耗模式下该信号仍然有效。
pad_ctim_refclk	I	-	-	< 5	定时器输入信号： 非时钟信号，定时器内部先将该信号同步到 CPU 时钟域（cpuclk），然后再以同步后的信号上升沿为计数脉冲计数。 注意：该信号的频率必须小于 cpuclk 频率的二分之一。 注：此信号仅实现于配备系统计时器的处理器中。
pad_ctim_calib[25:0]	I			< 5	core timer 校准控制信号 pad_ctim_calib[25]: NOREF pad_ctim_calib[24]: SKEW pad_ctim_calib[23:0]: TENMS 各位对应的具体功能参考紧耦合 IP 手册。 注：此信号仅实现于配备系统计时器的处理器中。

pad_sysio_clkratio[2:0]	I	-	CPU	≈10	cpu 时钟与总线时钟比例指示信号： CPU 时钟频率/总线时钟频率＝pad_sysio_clkratio + 1： 3’b000：1 比 1； 3’b001：2 比 1； 3’b010：3 比 1； 3’b011：4 比 1； 3’b100：5 比 1； 3’b101：6 比 1； 3’b110：7 比 1； 3’b111：8 比 1。 该信号没有实际功能，只供软件查询时使用
pad_sysio_bigend_b	I	-	CPU	≈15	数据格式指示信号： 0：数据或指令是 Big Endian（大端）的字节顺序； 1：数据或指令是 Little Endian（小端）的字节顺序。 该信号在 CPU 复位前被设置，不可以在 CPU 运行时动态变化。鉴于目前第三方外围 IP 在小端上面的格式基本相同，所以推荐使用小端模式。
pad_sysio_endian_v2	I	-	CPU	≈15	大小端版本指示信号： 1：v2 版本大小端； 0：v1 版本大小端。 注意：v1 版本和 v2 版本的小端完全一致。
sysio_pad_srst	O	1’b0	CPU	<5	软件复位指示信号。 表示 CPU 实现自身软复，系统可根据具体情况决定复位哪些模块。 该信号维持一个系统时钟周期。
复位控制信号： 注意：*表示当 CPU 没有配置安全扩展单元（SEU）时，该信号不存在					

pad_cpu_rst_b	I	-	CPU	≈7	处理器复位信号： 低电平时，复位 CPU；CPU 采用异步复位、同步释放的方式。 注意：要求系统将 CPU 复位信号同步到 CPU 时钟域，CPU 内部不进行同步操作。
pad_had_rst_b	I	-	CPU	<5	调试模块复位信号： 低电平时，复位调试模块，调试模块采用异步复位、同步释放的方式。 注意：要求系统将 HAD 复位信号同步到 CPU 时钟域，CPU 内部不进行同步操作。
pad_cpu_gpr_rst_b*	I	-	CPU	<5	通用寄存器复位信号： 低电平时，同步复位 S802 CPU 所有的通用寄存器。
pad_had_jtg_trst_b	I	-	JTAG	<5	JTAG 测试复位信号： JTAG 复位信号，低电平有效，复位整个 HAD 正常工作情况下该信号需置成高电平。
总线系统集成：					
pad_bmu_iahbl_base	I			<5	指令总线地址空间基地址
pad_bmu_iahbl_mask	I			<5	指令总线地址空间 mask
SYSAHB-Lite 接口信号： 注意：当 SYSAHB-Lite 配置 FLOP_OUT 时，接口信号同步到 SYS 时钟域；当 SYSAHB-Lite 为 NON FLOP_OUT 时，接口信号为 CPU 时钟域。*表示当 CPU 没有配置 SEU 时，该信号不存在。					
pad_biu_hrdata[31:0]	I	-	CPU/ SYS	≈26	读取数据信号
pad_biu_hrdata_par*	I	-	CPU/ SYS	≈15	数据奇偶校验信号
pad_biu_hrdata_pol*	I	-	CPU/ SYS	≈36	数据极性控制信号

pad_biu_hresp	I	-	CPU/SYS	≈30	传输应答信号
pad_biu_hready	I	-	CPU/SYS	≈30	外部空闲信号
biu_pad_haddr[31:0]	O	-	CPU/SYS	≈25	地址信号
biu_pad_haddr_pol*	O	-	CPU/SYS	<5	地址极性控制信号
biu_pad_htrans[1:0]	O	-	CPU/SYS	≈35	传输类型信号
biu_pad_hsize[2:0]	O	-	CPU/SYS	≈22	传输尺寸指示信号
biu_pad_hburst[2:0]	O	-	CPU/SYS	<5	突发传输指示信号
biu_pad_hwdata[31:0]	O	-	CPU/SYS	≈7	回写数据信号
biu_pad_hwdata_par*	O	-	CPU/SYS	≈18	数据奇偶校验信号
biu_pad_hwdata_pol*	O	-	CPU/SYS	<5	数据极性控制信号
biu_pad_hwrite	O	-	CPU/SYS	≈18	写传输信号
biu_pad_hprot[3:0]	O	-	CPU/SYS	≈28	保护控制信号： 指示当前总线周期进行的数据传输的特性： ***0：取指令； ***1：数据访问； **0*：普通用户访问； **1*：超级用户访问； *0**：Not secure； *1**：secure； 0***：Not cacheable； 1***：cacheable。
biu_pad_vec_redrct	O	-	CPU	<5	异常重定向信号，悬空即可

IAHB-Lite 接口信号：

注意：当 IAHB-Lite 配置 FLOP_OUT 时，接口信号同步到 SYS 时钟域；当 IAHB-Lite 为 NON FLOP_OUT 时，接口信号为 CPU 时钟域。当 CPU 没有配置 I-AHBL 时，该组信号不存在。*表示当 CPU 没有配置 SEU 时，该信号不存在。

pad_iahbl_hrdata[31:0]	I	-	CPU/ SYS	≈26	读取数据信号
pad_iahb_hrdata_par*	I	-	CPU/ SYS	≈15	数据奇偶校验信号
pad_iahb_hrdata_pol*	I	-	CPU/ SYS	≈36	数据极性控制信号
pad_iahbl_hresp	I	-	CPU/ SYS	≈30	传输应答信号
pad_iahbl_hready	I	-	CPU/ SYS	≈30	外部空闲信号
pad_bmu_iahbl_base	I	-			IAHB-Lite 基址控制信号，上电复位之后需固定
pad_bmu_iahbl_mask	I	-			IAHB-Lite 地址对齐控制信号，上电复位之后需固定
iahbl_pad_haddr[31:0]	O	-	CPU/ SYS	≈25	地址信号
iahbl_pad_haddr_pol*	O	-	CPU/ SYS	<5	地址极性控制信号
iahbl_pad_htrans[1:0]	O	-	CPU/ SYS	≈35	传输类型信号
Iabhl_pad_hsize[2:0]	O	-	CPU/ SYS	≈22	传输尺寸指示信号
Iabhl_pad_hburst[2:0]	O	-	CPU/ SYS	<5	突发传输指示信号
iahbl_pad_hwdata[31:0]	O	-	CPU/ SYS	≈7	回写数据信号
iahbl_pad_hwdata_par*	O	-	CPU/ SYS	≈18	数据奇偶校验信号
iahbl_pad_hwdata_pol*	O	-	CPU/ SYS	<5	数据极性控制信号

iahbl_pad_hwrite	O	-	CPU/ SYS	≈18	写传输信号
iahbl_pad_hprot[3:0]	O	-	CPU/ SYS	≈28	保护控制信号： 指示当前总线周期进行的数据传输的特性： ***0：取指令； ***1：数据访问； **0*：普通用户访问； **1*：超级用户访问； *0**：Not secure； *1**：secure； 0***：Not cacheable； 1***：cacheable。
iahbl_pad_vec_redrct	O	-	CPU	<5	异常重定向信号，悬空即可
中断系统集成： 注意：*表示该信号仅存在于 CPU 配置 VIC，**表示该信号仅存在于 CPU 没有配置 VIC。					
pad_vic_int_cfg[31:0]*	I	-	CPU	≈8	矢量中断控制器中断源类型配置信号： 1：脉冲中断源 0：电平中断源 注意：中断源确定后该信号为恒定值。
pad_vic_int_vld[31:0]*	I	-	CPU	≈9	矢量中断控制器中断源请求信号： 高电平时表示该中断源发起中断申请。 注意：要求系统将该信号同步到 CPU 时钟域，VIC 内部不进行同步操作。
pad_cpu_int_b**	I	-	SYS	<5	CPU 中断请求信号： 低电平时表示外部中断控制器发起中断申请。
pad_cpu_vec_b[7:0]**	I	-	SYS	<5	CPU 中断向量号： 需要在 32-255 的范围内使用。

ctim_pad_int_vld	O	1'b0	CPU	<5	中断有效指示信号： 指示系统计时器产生中断，该信号高电平有效。 注：此信号仅实现于配备系统计时器的处理器中。
调试系统集成：					
JTAG 支持信号：					
pad_had_jtg_tclk	I	-	-	<5	JTAG 测试时钟信号： JTAG 和 HAD 内部相关寄存器的时钟信号，正常工作时要求该时钟的工作频率小于 CPU 时钟频率的二分之一。
pad_had_jtg_tap_en	I	-	JTAG	<5	JTAG tap 控制器使能信号： 用于使能 JTAG tap 控制器，另外信号 pad_had_jdb_req_b 也可以达到同样的使能效果。
had_pad_jtg_tap_on	O	-	JTAG	<5	表明 tap 控制器状态指示信号： 用于指示 tap 控制器是否工作，高电平有效。
pad_had_jtg_tms_i	I	-	JTAG	<5	JTAG-2 串行数据输入信号： HAD 端在 JTAG 时钟信号（tclk）的上升沿对其采样，而外部调试器在 tclk 的下降沿设置该信号。 空闲时，最好将该信号保持为高电平，同时停止 tclk。如果 tclk 信号一直有效，用户只需保持该信号为高电平状态并维持 80 个时钟周期即可同步复位 HAD。
had_pad_jtg_tms_o	O	-	JTAG	<5	JTAG-2 串行数据输出信号： HAD 端在 tclk 的下降沿对其设置，而外部调试器在 tclk 的上升沿对其采样。

had_pad_jtg_tms_oe	O	1’b0	JTAG	<5	JTAG-2 串行数据输出有效指示信号： 要求 CPU 外部利用该信号通过一个三态门将 pad_had_jtg_tms_i 和 had_pad_jtg_tms_o 信号合为一个双向端口信号。
调试支持信号：					
sysio_pad_dbg_b	O	-	SYS	<5	调试模式指示信号： 指示 CPU 是否处于调试模式，该信号低电平有效，同步于系统时钟域（sysclk）。
pad_had_jdb_req_b	I	-	异步-	-	外部调试请求信号： 异步于 tclk，低电平有效。CPU 处于低功耗状态或者复位状态时，该信号可以快速请求 CPU 进入调试模式。 注意：802 已废弃该信号，不支持该调试方式
had_pad_jdb_ack_b	O	1’b1	JTAG	<5	CPU 响应调试模式信号： CPU 进入调试模式后，该响应信号保持两个 tclk 的低电平。
had_pad_jdb_pm[1:0]	O	2’b0	CPU	<5	处理器工作模式指示信号： 表明处理器的工作模式，异步于 tclk。 00: 普通模式； 01: 低功耗模式（STOP/DOZE/WAIT 模式）； 10：调试模式； 11：保留。
低功耗系统集成：					
sysio_pad_ipend_b	O	1’b0	SYS	<5	异步低功耗唤醒指示信号： 低电平有效，表示 CPU 接收到异步低功耗唤醒请求。

pad_cpu_intraw_b	I	-	SYS	<5	异步唤醒信号： 不请求进入中断，用于将 CPU 从低功耗模式中唤醒。这个信号会使 sysio_pad_wakeup_b 和 sysio_pad_ipend_b 有效。 注：此信号仅实现于不配备矢量中断控制器的处理器中。
sysio_pad_lpmd_b[1:0]	O	2'b11	SYS	<5	低功耗模式状态信号： 当处理器执行 doze, stop 或 wait 指令时，sysio_pad_lpmd_b[1:0]被相应的改变： 00：STOP； 01：WAIT； 10：DOZE； 11：普通模式。
sysio_pad_wakeup_b	O	1'b0	SYS	<5	低功耗唤醒指示信号： 低电平有效，表示 CPU 接收到低功耗唤醒请求，该信号将会在异步唤醒请求，或调试请求设置之后被设置。
had_pad_wakeup_req_b	O	1'b1	JTAG	<5	调试唤醒低功耗指示信号： 低电平有效，表示调试模块接收到唤醒请求，用于通知系统 PMU 进行唤醒操作
DFT 系统集成：					
pad_yy_test_mode	I	-	-	<5	测试模式信号： CPU 进入测试模式，此时 CPU 时钟和系统时钟均为测试时钟（pad_had_jtg_tclk）。在 CPU 进入测试模式，并且 pad_yy_scan_enable 有效时，才可以通过扫描链进行测试。 不用该信号时，需要接 0。

pad_yy_gate_clk_en_b	I	-	CPU	<5	门控时钟使能信号: 处理器的门控时钟使能控制信号，当该信号为低电平时，CPU 内部模块的门控时钟有效。 该信号默认接 0，不需要时该信号接 1。
CPU 运行观测信号集成：					
iu_pad_gpr_data	O	-	CPU	_	当前指令退休时 GPR 回写数据
iu_pad_gpr_we	O	-	CPU	_	当前指令退休时 GPR 回写有效信号
iu_pad_gpr_index	O	-	CPU	_	当前退休指令时 GPR 回写寄存器选择信号
iu_pad_inst_retire	O	-	CPU	_	指令退休有效信号
iu_pad_retire_pc	O	-	CPU	_	当前退休指令程序计数器的值
iu_pad_inst_split	O	-	CPU	_	当前指令为拆分指令
cp0_pad_psr	O	-	CPU	_	当前处理器状态
其它：					
pad_ctim_calib[25:0]	I	-	CPU	≈35	定时器输入信号： [25]表示是否提供参考时钟 0：提供参考时钟 1：没有提供参考时钟 [24]表示校准值是否精准 0：10ms 校准值是精准的 1：10ms 校准值是不精准的 [23:0]表示 10ms 时间校准值 注意：该信号是一个恒定值。当 CPU 没有配置系统计数器（CoreTim）时，该组信号不存在。
全局状态和控制信号： 注意：当 CPU 没有配置 GCR 和 GSR 时，改组信号不存在。					

sysio_pad_gcb[31:0]	O	32’b0	SYS	<5	全局控制信号： 全局控制总线输出，用于控制外围设备和事件，它对应于 GCR 寄存器的 32 位，通过 MTCR 指令修改 GCR 寄存器的值，达到控制外围设备的目的。当执行 MTCR 指令更新 GCR 寄存器时，sysio_pad_gcb[31:0]信号产生相应变化。
pad_sysio_gsb[31:0]	I	-	SYS	<5	全局状态信号： 全局状态总线输入，用于检测外围设备和事件，它对应于 GSR 只读寄存器的 32 位。该寄存器对外围设备和事件进行实时采样，通过 MFCR 读取 GSR 寄存器获得实时采样值。
安全相关信号： 注意：当 CPU 没有配置 SEU 时，安全相关信号不存在。					
pad_biu_beu_key0[23:0]	I	-	CPU	≈42	总线加扰或总线加密的密钥信号： 从系统中输入给 CPU 的密钥信号。
pad_biu_beu_key1[23:0]	I	-	CPU	≈46	总线加扰或总线加密的密钥信号： 从系统中输入给 CPU 的密钥信号。
pad_seu_rand_num[31:0]	I	-	CPU	<5	随机数信号： 指示真随机数。真随机数需要在该安全机制使能前开始指示，至安全机制关闭后为止。安全机制使能时，真随机数在每个 CPUCLK 周期都应当产生变化。
iu_pad_security_violation	O	1’b0	CPU	<5	安全违反信号： 指示处理器处于安全违反状态。该信号为 1 表示处理器安全违反，0 表示没有发生安全违反。

注：

1. 内部门级数是为了描述信号端口时序：如果端口是 input，该数值表示了这根信号进入处理器内部后还需要通过几级门才能到寄存器；如果端口是 output，则该数值表示该信

号从 CPU 内部寄存器出来还通过了几级门才到达端口。集成人员可通过这个数字结合 SOC 连接端口的延时，预判连接后时序是否能满足设计需求。

2. 总线信号内部门级数均参照 NON FLOP_OUT 最坏情况，如果是 FLOP_OUT 则总线相关输出都是寄存器输出，内部门级数都是 0。
3. 当总线配置为 FLOP_OUT 时，接口信号同步到 SYS 时钟域；当总线配置为 NON FLOP_OUT 时，接口信号为 CPU 时钟域。
4. CPU 内部 flop-flop 的门级数最坏情况大约为 53（有些特殊抗物理攻击配置配上后时序会更差一点）。
5. 内部门级数参照标准库为 scc55nll_hd_hvt，仅供参考，其它工艺库可能会存在一定误差。

3 时钟复位信号集成

3.1 端口列表

时钟信号：

信号名	方向	复位	时钟	功能描述
pll_core_cpuclk	I	-	-	CPU 工作时钟信号： 提供 CPU 工作的时钟。
pad_sysio_clk_unlock*	I	-	-	CPU 全局门控时钟使能信号： 控制 CPU 全局门控时钟单元的开启和关闭。
clk_en	I	-	-	CPU 同步时钟信号: 当为 1 时，CPU 时钟可以采样总线时钟的信号.
pad_sysio_clkratio[2:0]	I		CPU	cpu 时钟与总线时钟比例指示信号： CPU 时钟频率/总线时钟频率＝pad_sysio_clkratio + 1： 3’b000：1 比 1； 3’b001：2 比 1； 3’b010：3 比 1； 3’b011：4 比 1； 3’b100：5 比 1； 3’b101：6 比 1； 3’b110：7 比 1； 3’b111：8 比 1。

注意：*表示当 CPU 没有配置低功耗时钟（TWO_CLK），该信号不存在。

复位控制信号：

信号名	方向	复位	时钟	功能描述
pad_cpu_rst_b	I	-	CPU	处理器复位信号： 低电平时，复位 CPU；CPU 采用异步复位、同步释放的方式。 注意：要求系统将 CPU 复位信号同步到 CPU 时钟域，CPU 内部不进行同步操作。

信号名	方向	复位	时钟	功能描述
pad_had_rst_b	I	-	CPU	调试模块复位信号： 低电平时，复位调试模块，调试模块采用异步复位、同步释放的方式。 注意：要求系统将 HAD 复位信号同步到 CPU 时钟域，CPU 内部不进行同步操作。
pad_cpu_gpr_rst_b*	I	-	CPU	通用寄存器复位信号： 低电平时，同步复位 S802 CPU 所有的通用寄存器。

注意：*表示当 CPU 没有配置安全扩展单元（SEU）时，该信号不存在。

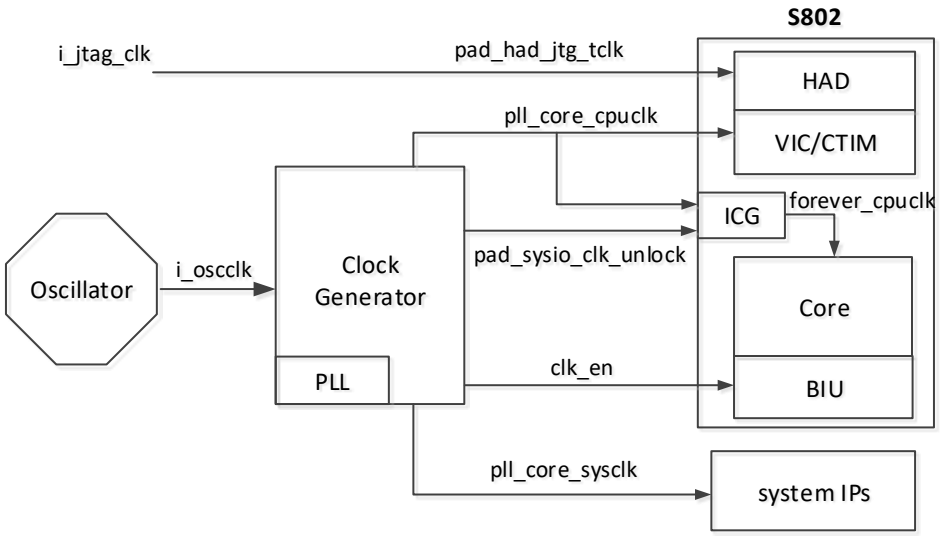
3.2 时钟信号

3.2.1 S802 时钟域

S802 CPU 有两个外部输入时钟，分别为：pll_core_cpuclk 和 pad_had_jtg_tclk。其中，pll_core_cpuclk 为 CPU 域时钟，在 CPU 处于低功耗模式时依然工作，用于中断请求的采样，系统计时器（CoreTim）计时和 HAD 调试，保证 CPU 在低功耗模式下能够采样中断请求和调试请求。另外，S802 内部维护一个全局门控时钟单元，产生一个与 pll_core_cpuclk 同频同相的 CPU 域时钟 forever_cpuclk，是 CPU 核和 BIU 的工作时钟。该时钟在 CPU 处于低功耗模式且在 pad_sysio_clk_unlock 关闭时停止工作，从而降低时钟树的翻转功耗。pad_had_jtg_clk 为 JTAG 域时钟，用于 CPU 内部调试单元（HAD）中 JTAG 状态机的运行。

另外，外围总线模块使用 pll_core_sysclk 时钟，可对 CPUCLK 进行整数比例的分频获得该系统时钟，因此 CPU 需要在 clk_en 有效时与外围 IP 进行数据交互。注意：S802 仅在配置 FLOPOUT 下支持对 CPUCLK 进行 1 到 8 分频。

注意：pad_ctim_refclk 不是时钟信号。当需要采用外部参考时钟计数时，CoreTim 先用两级工作在 pll_core_cpuclk 时钟域的寄存器将 pad_ctim_refclk 进行同步，然后再采样信号的上升沿，用采样到的上升沿为计数脉冲计数。该信号的频率必须小于 pll_core_cpuclk 频率的一半。



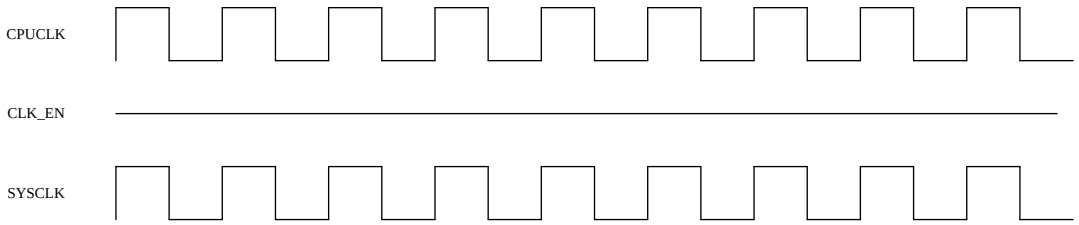
图表 3-1 S802 的时钟管理方式

图表 3-1 是一个 S802 的时钟管理示例。pad_had_jtag_tclk 是 JTAG 时钟，与 CPU 时钟异步，该时钟信号属于 SoC 系统的顶层信号，需要从芯片引脚接入。时钟产生逻辑以板上晶振时钟 i_oscclk 为输入，产生系统工作时钟 pll_core_sysclk 、CPU 工作时钟 pll_core_cpuclk 和这两个时钟的同步信号 clk_en。pll_core_cpuclk 是 VIC 中断采样电路、CoreTim 计时器电路和 HAD 调试电路的时钟，用于低功耗模式下产生中断唤醒请求或者调试请求唤醒 CPU。

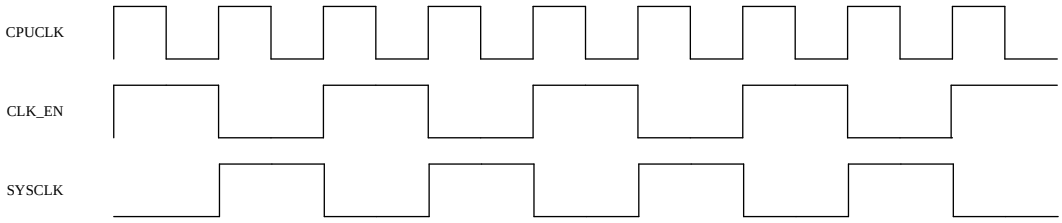
forever_cpuclk 时钟经过一个 CPU 顶层的全局门控时钟单元产生，作为 CPU 核和 BIU 的工作时钟。该全局门控时钟单元在 CPU 处于低功耗工作模式且 pad_sysio_clk_unlock 关闭时时停止 forever_cpuclk，从而减少时钟树的翻转功耗。S802 内部进一步设计有局部门控时钟单元，通过优化的控制逻辑动态控制内部时钟的开启和关闭。

3.2.2 总线分频

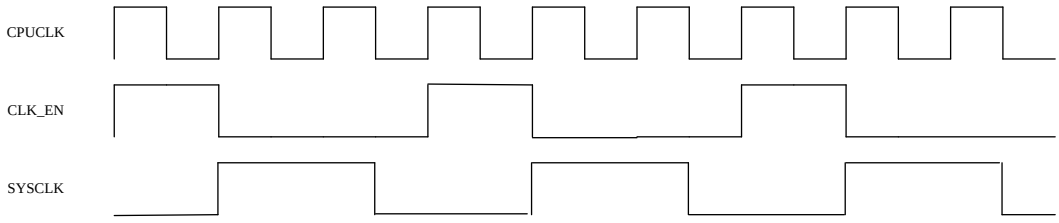
pll_core_sysclk 是外围总线模块的工作时钟，CPU 通过总线接口单元（BIU）与外围 IP 进行交互，因此 BIU 内部设计有同步控制信号，保证总线接口上的输入输出都在 pll_core_sysclk 的上升沿变化和采样，由 clk_en 信号控制。图表 3-2 至图表 3-9 是 CPU 时钟和系统时钟的关系图。



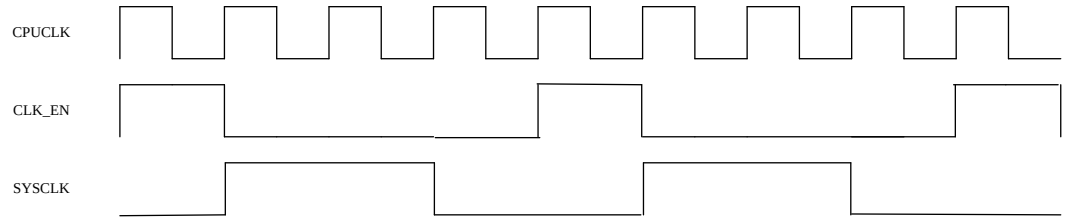
图表 3-2 CPUCLK 和 SYSCLK 1:1 波形图



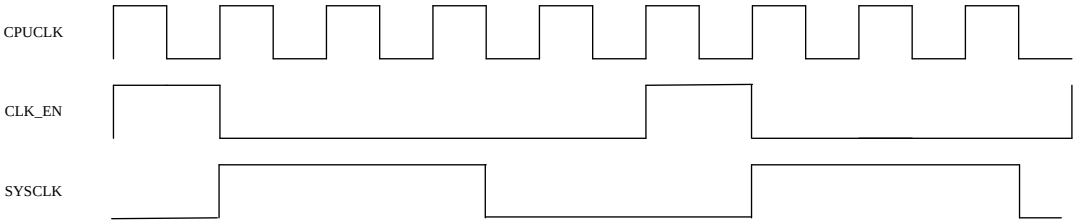
图表 3-3 CPUCLK 和 SYSCLK 2:1 波形图



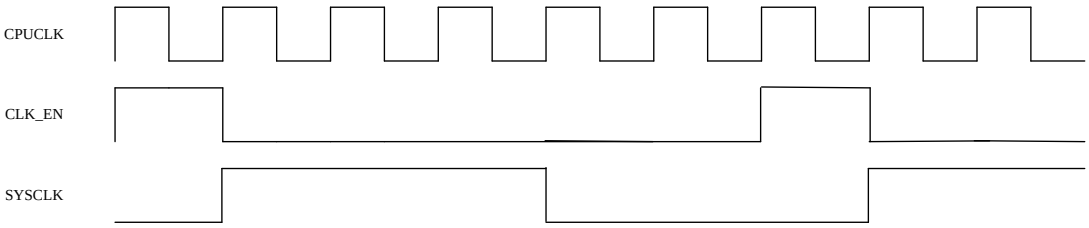
图表 3-4 CPUCLK 和 SYSCLK 3:1 波形图



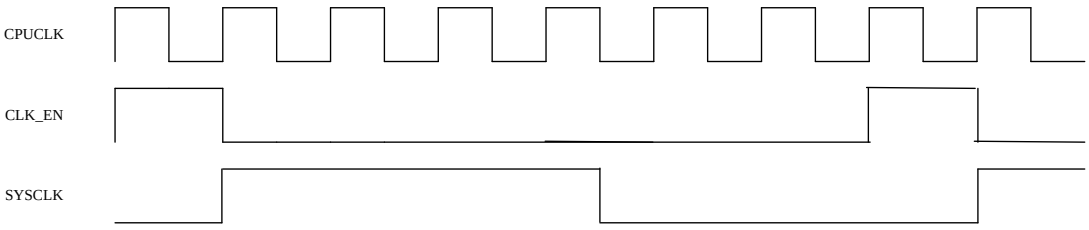
图表 3-5 CPUCLK 和 SYSCLK 4:1 波形图



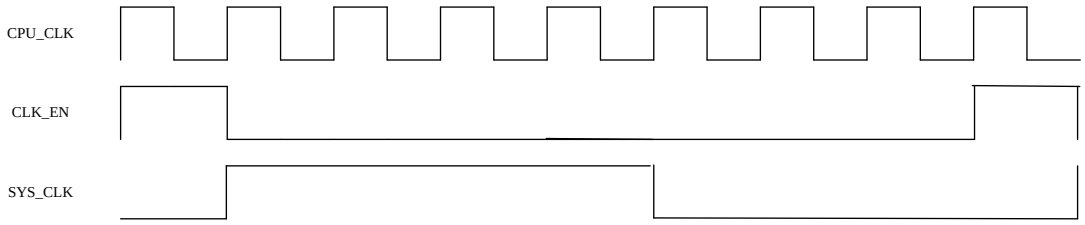
图表 3-6 CPUCLK 和 SYSCLK 5:1 波形图



图表 3-7 CPUCLK 和 SYSCLK 6:1 波形图



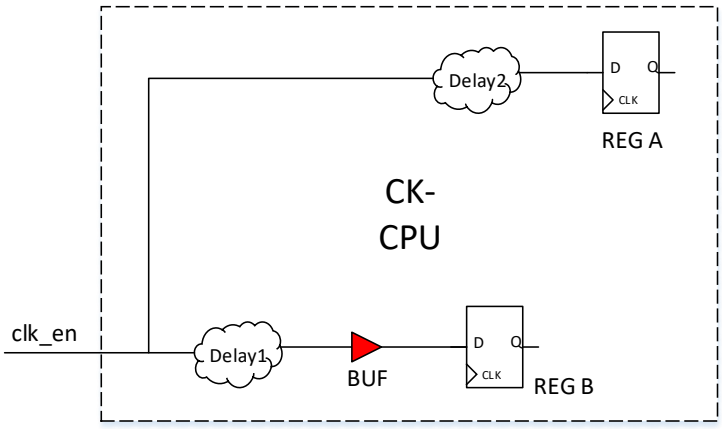
图表 3-8 CPUCLK 和 SYSCLK 7:1 波形图



图表 3-9 CPU_CLK 和 SYS_CLK 8:1 波形图

3.2.3 CLK_EN

在 CPU 单独做综合的时候需要注意如下问题，clk_en 信号会串到 CPU 里很多寄存器上，而且路径延时有长有短，比如下图中到 REG A 的延时很长，到 REG B 的延时很短，如果做 CPU 硬核时不做特别处理，当 SOC 集成 CPU 硬核做时序分析时，可能会出现 delay1 太小 REG B 出现 hold violation，需要在 CPU 硬核外面插入 BUFFER 来解决，但插入后发现 delay2 因为 BUFFER 的插入变大导致 REG A 出现了 setup violation，为了避免这种情况发生，在做 CPU 硬核的时候需要尽量把该信号所经过路径的时序做平，可以通过插 BUFFER 的方式。除此之外 SOC 时序分析时也需要对该信号进行检查。

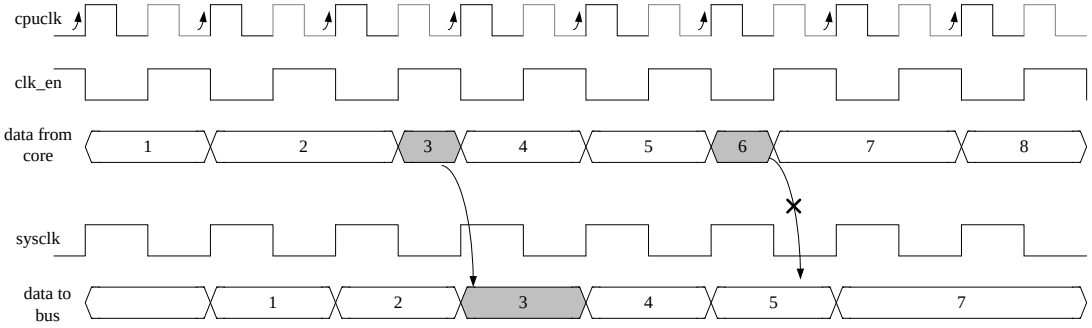


图表 3-10 CLK_EN 时序检查

3.3 多时钟域信号同步

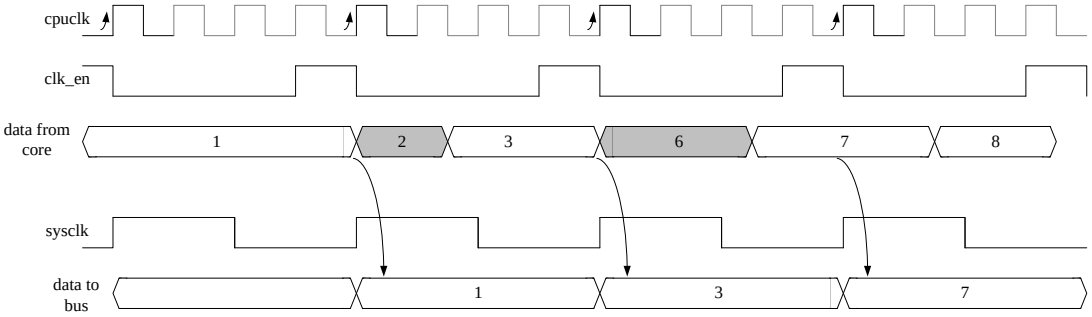
3.3.1 CPU 内核时钟域与系统总线时钟域

在 S802 的设计中，CPU 内核时钟域与系统总线时钟域在总线接口部分逻辑中有信号交互。要求这两个时钟在物理设计中保证相位对齐。这两个时钟频率为整数倍关系，仅使用 clk_en 即可完成这两个时钟之间的信号同步。clk_en 表示系统时钟的上升沿在当前 cpuclk 时钟周期有效，使用 clk_en 可以保证 CPU 内部和总线之间的信号交互按照系统时钟有效沿对齐。因此从 CPU 总线到系统的信号都具有 sysclk 的全周期时序延时。图表 3-11 和图表 3-12 分别给出了 CPU 与系统时钟比例在 2：1 与 4：1 的情况下的 CPU 内部信号变化与总线接口上的信号变化。



(a). cpuclk:sysclk = 2:1

图表 3-11 CPU 与系统时钟 2：1 情况下的接口信号



(b). cpuclk:sysclk = 4:1

图表 3-12 CPU 与系统时钟 4：1 情况下的接口信号

3.3.2 系统总线 multi cycle 的设置示例

在 S802 的总线接口单元设计时，考虑了 CPU 时钟域与系统时钟域之间信号同步的问题，S802 会保证所有的总线输出信号是在 clk_en 信号为高的那个 cpuclk 的上升沿输出，而所有的总线输入信号是在 clk_en 信号为高的那个 cpuclk 的上升沿采样。

基于这样的设计，用户在系统时钟和 CPU 时钟非 1：1 的情况下，集成 S802 时，需要

通过 set_multicycle_path 的命令来的设置 S802 与系统的交互信号。

基于 AHB 总线的设置示例如下：

set_multicycle_path -setup 4 -start -through [get_pins x_ahb_soc/x_nm_cpu_top/biu_pad_h]*

set_multicycle_path -hold 3 -start -through [get_pins x_ahb_soc/x_nm_cpu_top/biu_pad_h]*

set_multicycle_path -setup 4 -end -through [get_pins x_ahb_soc/x_nm_cpu_top/pad_biu_h]*

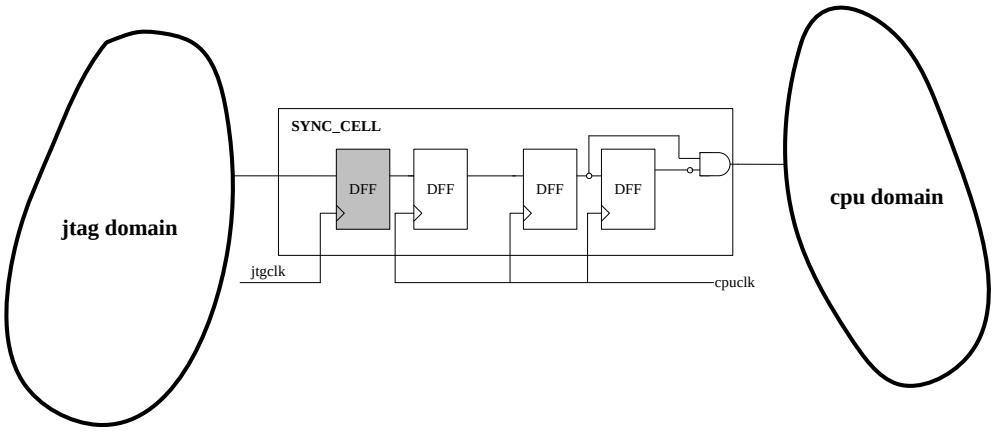
set_multicycle_path -hold 3 -end -through [get_pins x_ahb_soc/x_nm_cpu_top/pad_biu_h]*

注意：1.此示例是按照 CPU 时钟与系统时钟频率为 4：1 的假设进行设置；

2.此示例中的 *x_ahb_soc/x_nm_cpu_top/*以实际的 SOC 集成方式替换；

3.3.3 CPU 内核时钟域与 JTAG 时钟域

在 S802 的设计中，CPU 内核时钟 cpuclk 信号与 HAD 内部时钟 jtagclk 信号之间有信息交互，CPU 内部已经通过同步逻辑单元将两个时钟域之间的信号进行了有效同步，用户无需关心。其逻辑框图如图表 3-13 所示。



图表 3-13 CPU 与 JTAG 时钟域之间的同步逻辑

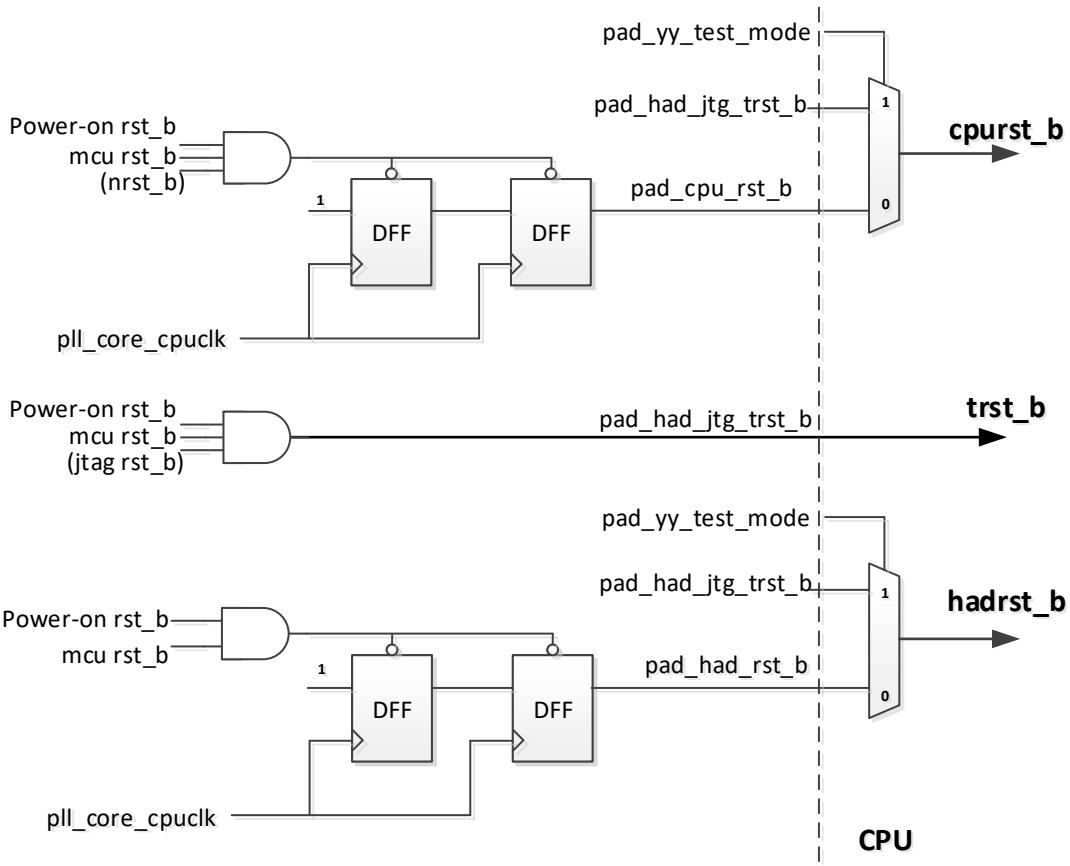
3.4 复位信号

S802 输入三根复位信号，分别是用于 CPU 核（除 HAD 之外）的复位信号 pad_cpu_rst_b、用于 HAD 中 CPU 域的复位信号 pad_had_rst_b 和用于 HAD TCLK 域的复位信号 had_pad_jtag_trst_b。

在测试模式下（pad_yy_test_mode=1），用于同步的寄存器被旁路，复位信号全部选择 pad_had_jtg_trst_b。

为了有效避免复位过程中亚稳态的出现，S802 采用异步复位、同步释放的方式进行系统的复位，因此需要各个复位信号同步到相对应的时钟域中。复位信号的同步工作由系统完成，CPU 自身仅对复位信号进行选择。下图给出了 S802 的复位信号的一个示例。

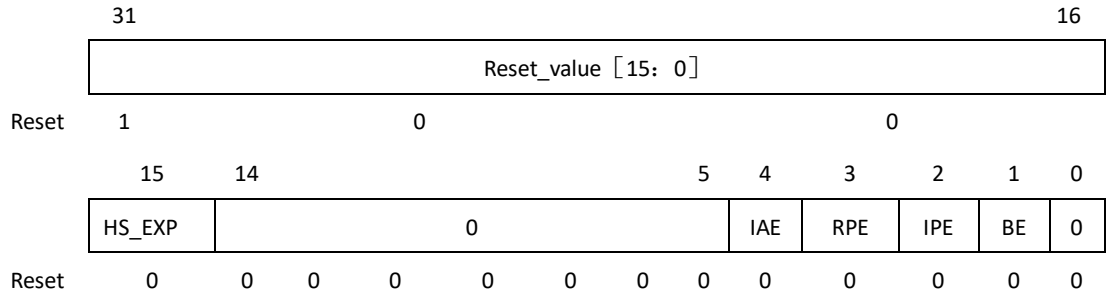
- pad_cpu_rst_b 信号由系统上电复位，mcu 复位（通常为整个 mcu 的输入 port 端口）以及 nreset（在线仿真器发出的 cpu 复位信号，用户可以根据实际情况选择是否实现 nreset，因为该功能可通过软复位实现）经过两级工作在 pll_core_cpuclk 时钟下的同步寄存器得到
- pad_had_rst_b 信号由系统上电复位和 mcu 复位经过两级工作在 pll_core_cpuclk 时钟下的同步寄存器得到
- pad_had_jtag_trst_b 由系统上电复位，mcu 复位以及 jtag 复位（在线仿真器发出的 jtag 复位信号，在 2 线 jtag 调试配置下用户可以根据实际情况选择是否实现该 jtag 复位，因为状态机可以实现自复位）得到，jtag 复位信号由在线仿真器完成时钟域同步工作；另外，在测试模式下，pad_had_jtag_trst_b 为 cpu 全局复位信号，测试机通过上述 MCU 复位端口对复位信号进行控制



图表 3-14 S802 复位信号产生机制

3.5 软复位

处理器隐式操作寄存器（CHR，CR<31,0>）是处理器一些隐式操作的集合，包括软复位操作以及处理器某些加速功能的使能。如图表 3-15 所示，当 Reset_Value 软复位使能域被写入 16'hABCD 时，将触发处理器复位操作，处理器自行复位，并通过处理器引脚 sysio_pad_srst 指示处理器发生软件复位，sysio_pad_srst 维持一个系统时钟周期。



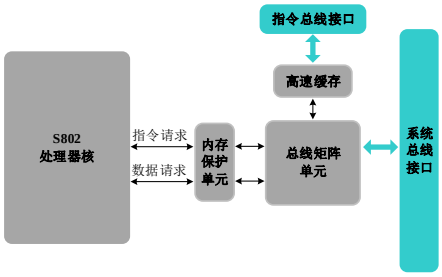
图表 3-15 隐式操作寄存器

4 总线系统集成

4.1 总线简介及配置信息

S802 实现了多总线接口，分别包括系统总线、指令总线。指令总线可由用户根据实际的系统需要进行配置。

总线矩阵为处理器内部请求访问外部总线接口提供了互联功能。总线矩阵与 CPU 内部请求及总线接口的连接关系如图表 4-1 所示。总线矩阵根据内存访问的地址仲裁总线接口类型，将处理器内部访问分发到系统总线、指令总线上。



图表 4-1 S802 总线矩阵

处理器内部的取指访问和数据访问拥有相同的总线访问权限，可以访问所有总线接口。为了解决同一时钟周期取指访问和数据访问竞争同一总线接口的问题，总线矩阵也负责请求的优先级判断。当取指请求和数据请求竞争同一总线接口时，数据请求拥有更高的优先级。

S802 多总线接口的基本信息和可配置性如图表 4-2 所示。

图表 4-2 多总线接口的基本信息和可配置性

总线接口	可配置性（有/无）	总线协议	时序方式
系统总线	可配置	AHB	寄存器输出
		AHB-Lite	直接输出（推荐）
指令总线	可配置	AHB-Lite	寄存器输出
			直接输出（推荐）

另外， S802 通过提供一组接口信号（pad_bmu_iahbl_base 和 pad_bmu_iahbl_mask），支持指令总线基地址和空间大小硬件可配置。其中，pad_bmu_iahbl_base 指定了指令总线的基地址；pad_bmu_iahbl_mask 指定了不同地址空间下对地址对齐的需求。指令总线的地址空间 1MB 到 4GB 可配置，例如设置地址空间大小为 8M，pad_bmu_iahbl_base[2:0]必须为 3’b0，pad_bmu_iahbl_mask[11:0]必须为 12’b1111 1111 1000，不同大小的地址空间具体要求见下表。

图表 4-3 指令总线对基地址和地址对齐的要求

地址空间大小	对 pad_bmu_iahbl_base 的要求	对 pad_bmu_iahbl_mask 的要求
1MB	没有要求	bit[11:0]=12'b1111 1111 1111
2MB	bit[0] =0	bit[11:0]=12'b1111 1111 1110
4MB	bit[1:0] =2'b0	bit[11:0]=12'b1111 1111 1100
8MB	bit[2:0] =3'b0	bit[11:0]=12'b1111 1111 1000
16MB	bit[3:0] =4'b0	bit[11:0]=12'b1111 1111 0000
32MB	bit[4:0] =5'b0	bit[11:0]=12'b1111 1110 0000
64MB	bit[5:0] =6'b0	bit[11:0]=12'b1111 1100 0000
128MB	bit[6:0] =7'b0	bit[11:0]=12'b1111 1000 0000
256MB	bit[7:0] =8'b0	bit[11:0]=12'b1111 0000 0000
512MB	bit[8:0] =9'b0	bit[11:0]=12'b1110 0000 0000
1GB	bit[9:0] =10'b0	bit[11:0]=12'b1100 0000 0000
2GB	bit[10:0] =11'b0	bit[11:0]=12'b1000 0000 0000
4GB	bit[11:0] =12'b0	bit[11:0]=12'b0000 0000 0000

4.2 系统总线接口

S802 的系统总线接口可以配置为支持 AMBA2.0 AHB 协议（此时请参考 AMBA 2.0 规格说明—AMBA™ Specification Rev 2.0），也可以配置为支持 AMBA3.0 AHB-Lite 协议（此时请参考 AMBA 3.0 规格说明—AMBA3 AHB-Lite Protocal Specification Rev 1.0）。

系统总线接口的基本特点包括：

支持 AMBA2.0 AHB 或 AMBA3.0 AHB-Lite 总线协议，可由用户配置；

- 在寄存器输出方式下，支持多种 CPU 和系统时钟的频率比；
- 在寄存器输出方式下，支持系统动态变频（调整 CPU 和系统时钟的频率比）。

考虑到 S802 的应用领域及成本，系统总线接口只实现了 AHB/AHB-Lite 协议中的部分内容。

在 AHB-Lite 协议下，作为主设备，总线接口支持的传输类型为：

- HBURST 只支持 SINGLE 传输，其它突发类型均不支持；
- HTRANS 只支持 IDLE 和 NONSEQ，其它传输类型均不支持；
- HSIZE 支持字、字节和半字传输，其它传输大小不支持；
- HWRITE 支持读和写操作。

在 AHB 协议下，作为主设备，总线接口支持的传输类型为：

- HBURST 支持 SINGLE 传输，其它突发类型均不支持；

- HTRANS 支持 IDLE、NONSEQ，其它传输类型均不支持；
- HSIZE 支持字、字节和半字传输，其它传输大小不支持；
- HWRITE 支持读和写操作。

在 AHB 及 AHB-Lite 协议下，总线接口接受从设备的响应类型为：

- HREADY 支持 Ready 和 Not Ready；
- HRESP 支持 OKAY 和 ERROR，其它响应类型不支持。

信号名	I/O	Reset	定义
AHB/AHB-LITE 系统总线主接口信号：			
biu_pad_haddr[31:0]	O	-	地址总线： 32 位地址总线。
biu_pad_hwdata[31:0]	O	-	写数据总线： 32 位写数据总线。
biu_pad_hburst[2:0]	O	-	突发传输指示信号： 指示传输是一次突发传输的一部分： 000：SINGLE； 001：INCR； 010：WRAP4。 S802 仅支持 SINGLE 传输。
biu_pad_hsize[2:0]	O	-	传输宽度指示信号： 指示传输的数据宽度： 000：byte； 001：halfword； 010：word。

biu_pad_htrans[1:0]	O	00	传输类型表示信号： 指示当前总线周期的传输类型： 00：IDLE； 01：BUSY； 10：NONSEQ； 11：SEQ。 S802 中仅支持 NONSEQ 和 IDLE 两种传输类型。
biu_pad_hwrite	O	0	读写表示信号： 指示当前 CPU 是读取总线数据还是写总线： 1：指示是写总线传输； 0：指示是读总线传输。
biu_pad_hprot[3:0]	O	-	保护控制信号： 指示当前总线周期进行的数据传输的特性： ***0：取指令； ***1：数据访问； **0*：用户访问； **1*：超级用户访问； *0**：Not bufferable； *1**：bufferable； 0***：Not cacheable； 1***：cacheable。
biu_pad_hbusreq	O	0	总线请求信号： 指示 CPU 请求总线的使用权。 该信号仅在配置兼容 AHB 协议时才存在

biu_pad_hlock	O	0	锁定总线信号： 当 lock 申明时，CPU 独占有总线控制权，没有其他的 bus master 可以占有总线。 该信号仅在配置兼容 AHB 协议时才存在
pad_biu_hrdata[31:0]	I	-	读数据总线： 32 位读数据总线。
pad_biu_hready	I	-	传输完成指示信号： 有效时指示当前传输已完成，CPU 将总线置于待命状态。
pad_biu_hgrant	I	-	总线占用指示信号： 有效时指示 CPU 当前已占用总线。 该信号仅在配置兼容 AHB 协议时才存在
pad_biu_hresp[1:0]	I	-	传输应答信号： 传输应答： 00：OKAY； 01：ERROR； 10：RETRY； 11：SPLIT。 S802 中仅支持 OKAY 和 ERROR 两种响应

4.3 指令总线接口

K802 的指令总线只支持 AMBA3.0 AHB-LITE 协议，可参考 AMBA 3.0 规格说明-AMBA3 AHB-Lite Protocal Specification Rev 1.0。

指令总线接口的基本特点有：

- 与 AMBA3.0 AHB_LITE 总线协议兼容；
- 支持寄存器输出（Flop-out）和直接输出（Non-Flop-out）两种方式，可由用户配置。
- 在寄存器输出方式下，支持多种 CPU 和系统时钟的频率比；

- 在寄存器输出方式下，支持系统动态变频（调整 CPU 和系统时钟的频率比）。

考虑到 S802 的应用领域及成本，指令总线接口只实现了 AHB-Lite 协议中的部分内容。在 AHB-Lite 协议下，作为主设备，总线接口支持的传输类型为：

- HBURST 只支持 SINGLE 传输，其它突发类型均不支持；
- HTRANS 只支持 IDLE 和 NONSEQ，其它传输类型均不支持；
- HSIZE 支持字、字节和半字传输，其它传输大小不支持；
- HWRITE 支持读和写操作。

在 AHB-Lite 协议下，总线接口接受从设备的响应类型为：

- HREADY 支持 Ready 和 Not Ready；
- HRESP 支持 OKAY 和 ERROR，其它响应类型不支持。

信号名	I/O	Reset	定义
指令 AHB-Lite 接口*：（视处理器配置而定）			
iahbl_pad_haddr[31:0]	O	-	地址总线： 32 位地址总线。
ilite_clk_en	I	-	总线同步时钟使能信号： iahb 与 CPU 外部系统同步时钟使能有效。
iahbl_pad_hburst[2:0]	O	-	突发传输指示信号： 指示传输是一次突发传输的一部分： 000：SINGLE； 001：INCR； 010：WRAP4。 S802 仅支持 SINGLE 传输。

iahbl_pad_hprot[3:0]	O	-	保护控制信号： 指示当前总线周期进行的数据传输的特性： ***0：取指令； ***1：数据访问； **0*：用户访问； **1*：超级用户访问； *0**：Not bufferable； *1**：bufferable； 0***：Not cacheable； 1***：cacheable。
iahbl_pad_hsize[2:0]	O	-	传输宽度指示信号： 指示传输的数据宽度： 000：byte； 001：halfword； 010：word。
iahbl_pad_htrans[1:0]	O	00	传输类型表示信号： 指示当前总线周期的传输类型： 00：IDLE； 01：BUSY； 10：NONSEQ； 11：SEQ。 S802 仅支持 IDLE 和 NONSEQ 两种传输类型；
iahbl_pad_hwdata[31:0]	O	-	写数据总线： 32 位写数据总线。

iahbl_pad_hwrite	O	0	读写表示信号： 指示当前 CPU 是读取总线数据还是写总线： 1：指示是写总线传输； 0：指示是读总线传输。
pad_iahbl_hrdata[31:0]	I	-	读数据总线： 32 位读数据总线。
pad_iahbl_hready	I	-	传输完成指示信号： 有效时指示当前传输已完成，CPU 将总线置于待命状态。
pad_iahbl_hresp	I	-	传输应答信号： 传输应答： 0：OKAY； 1：ERROR。

5 中断系统集成

5.1 中断处理过程简述

中断处理是指处理器在接受到外部中断请求后从正常的程序处理转而响应中断处理，执行特定的中断处理程序。被中断的指令将正常退休，并在退休时响应中断请求。CPU 会保存当前的指令运行状态，将下一条指令作为中断返回的指令入口，并在退出中断服务程序时恢复之前的状态。

5.2 使用 CSKY VIC

本节主要介绍了配置有 VIC 单元时，中断的响应处理及信号端口。

5.2.1 端口列表

矢量中断控制器的接口信号主要用于 VIC 接收并采样中断源的中断请求信号;

信号名	I/O	Reset	定义
中断源信号：			
pad_vic_int_cfg[31:0]	I	-	中断源类型配置信号： 0： 电平中断源 1： 脉冲中断源
pad_vic_int_vld[31:0]	I	0	中断有效信号： 高电平有效。
时钟信号：			
pll_core_cpuclk	I	-	VIC 的工作及采样时钟：
clk_en	I	-	系统时钟同步使能信号

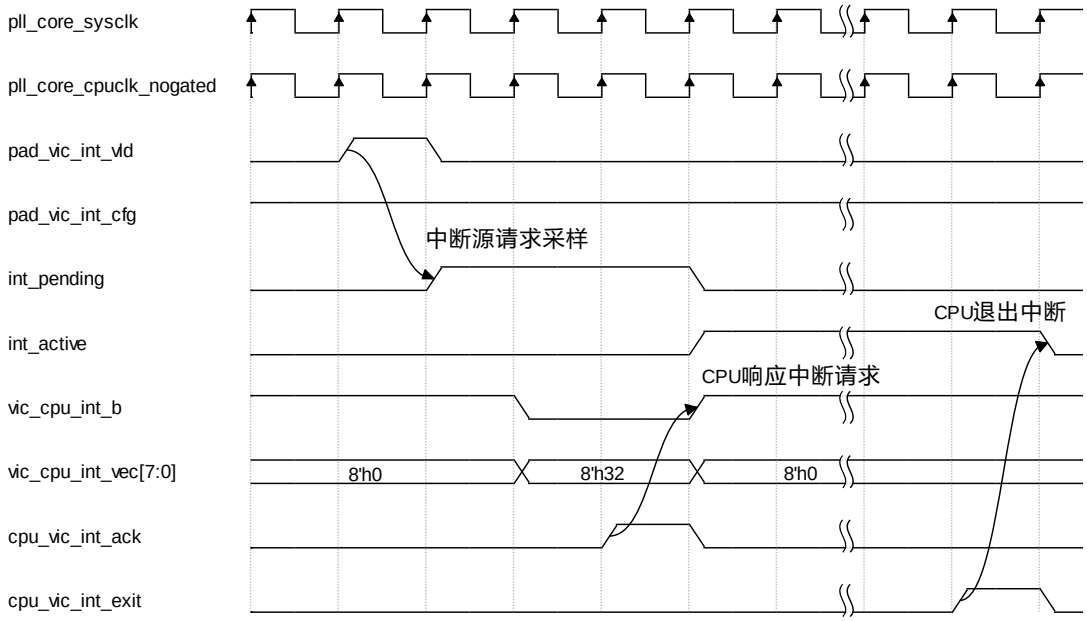
图表 5-1 矢量中断控制器接口信号

5.2.2 中断握手时序图

当 CPU 配置 VIC 时，由 VIC 负责采样外部中断源的中断请求：对于电平中断，矢量中断控制器采样到中断有效信号的高电平后设置对应中断进入等待状态，电平中断要求中断服务程序中清除外设的中断源有效信号，否则当中断退出时会重新发起中断请求。外设可以根据这一特点，常置中断信号直到不再需要中断处理程序处理；对于脉冲中断，矢量中断控制器采样中断有效信号的上升沿，然后设置对应中断进入等待状态。然后根据中断的优先级向 CPU 发送中断请求，在 CPU 响应该脉冲中断请求前，若脉冲中断源向矢量中断控制器发起多次中断请求，矢量中断控制器只会记录一次中断请求。在 CPU 响应该脉冲中断请求后，

若脉冲中断源再次向中断控制器发起请求，矢量中断控制器会再次触发对应中断进入等待状态，该处于等待状态的中断请求在中断退出后才能够再次被 CPU 响应。

图 5-2 给出了 CPU 与 VIC 中断交互时序图。当 VIC 采样到中断信号时，VIC 设置中断等待状态位 int_pending，并向 CPU 发起中断请求（vic_cpu_int_b）；当 VIC 接收到 CPU 中断响应信号（cpu_vic_int_ack）时，VIC 拉低 int_pending 并置高中断响应状态位 int_active； 当 VIC 接收到中断退出信号（cpu_vic_int_exit）时拉低 int_active，以此完成了一次中断请求的处理。中断退出信号由 CPU 在退出中断服务程序时设置。



图表 5-2 CPU 配置 VIC 时中断相关信号时序简图

5.2.3 中断嵌套

C-SKY 提供的 VIC 支持中断嵌套功能，VIC 能记录当前 CPU 正在处理哪个中断，根据设置的中断优先级判定新来的中断是否要能打断当前正在处理的中断，从而提高中断响应实时性。更多内容可以参考《802 紧耦合 IP 用户手册》。

5.3 不使用 CSKY VIC

本节主要介绍不配置 VIC 时，中断的响应处理及相关端口。

5.3.1 端口列表

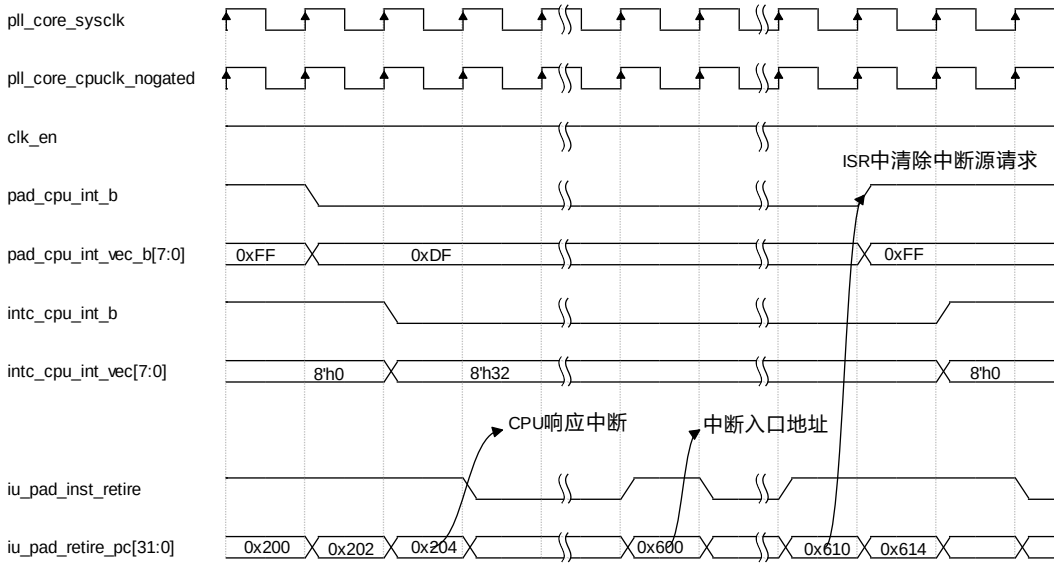
当 CPU 内部没有集成矢量中断控制器时，由外部中断控制器对中断源进行仲裁，产生中断请求发送给 CPU。CPU 对外部输入的中断请求信号进行同步并传递给核内处理。另外，CPU 也需要对外部 IP 产生低功耗唤醒信号进行同步。

信号名	I/O	Reset	定义
紧耦合 IP 总线接口信号：			
pad_cpu_int_b	I	1	中断请求信号： 低电平时表示进行普通中断申请。
pad_cpu_intraw_b	I	1	低功耗唤醒请求信号： 低电平时表示进行低功耗唤醒申请。
pad_cpu_int_vec_b[7:0]	I	-	中断向量号： 指示当前中断请求对应中断向量号.
时钟信号：			
pll_core_cpuclk	I	-	中断信号采样时钟
clk_en	I	-	系统时钟同步使能信号

图表 5-3 没有配置矢量中断控制器的接口信号

5.3.2 中断握手时序图

当没有配置 VIC 时，由外部中断控制器完成中断的采样和优先级仲裁，然后向 CPU 发起中断请求（pad_cpu_int_b 和 pad_cpu_int_vec_b）。CPU 需要将中断请求信号从 SYS 时钟域同步到 CPU 时钟域，然后响应中断进入中断服务程序。注意：CPU 在中断响应和中断退出均不会向外部中断控制器返回任何响应和退出信号，由中断服务程序和外部中断控制器负责清除中断请求。图表 5-4 CPU 没有配置 VIC 时中断相关信号时序简图给出了 CPU 响应外部中断请求的过程。



图表 5-4 CPU 没有配置 VIC 时中断相关信号时序简图

5.3.3 中断嵌套

在没有配置 C-SKY VIC 时，需要外部自行设计配置 VIC 模块，实现优先级判断及嵌套相关功能。

6 调试系统集成

6.1 端口列表

JTAG 支持信号：			
pad_had_jtg_trst_b	I	-	JTAG 测试复位信号： JTAG 复位信号，下跳变可以初始化和 JTAG 接口相关的触发器，正常工作情况下该信号需置成高电平。
pad_had_jtg_tclk	I	-	JTAG 测试时钟信号： JTAG 和 HAD 内部相关触发器的时钟信号，要求和 cpuclk 的频率比在 1：8 以上。
pad_had_jtg_tms_i	I	-	JTAG 数据输入信号： JTAG 串行输入端口，包括控制命令，数据，输入顺序由低位到高位。 注：此信号出现于实现 2 线制 HAD 的处理器中。
had_pad_jtg_tms_o	O	-	JTAG 数据输出信号： JTAG 串行数据输出端口，输出顺序由低位到高位。 注：此信号出现于实现 2 线制 HAD 的处理器中。
had_pad_jtg_tms_oe	O	-	JTAG 数据输出有效信号： 注：此信号出现于实现 2 线制 HAD 的处理器中。
pad_had_jtg_tap_en	I	-	JTAG tap 控制器使能信号： 用于使能 JTAG tap 控制器，另外信号 pad_had_jdb_req_b 也可以达到同样的使能效果。
had_pad_jtg_tap_on	O	-	表明 tap 控制器状态指示信号： 用于指示 tap 控制器是否工作，高电平有效。
调试支持信号：			

pad_sysio_dbgrq_b	I	-	外部调试请求信号： 该信号是外部调试请求信号，低电平有效，同步于 sysclk，使能处理器进入调试模式。 不用该信号时，需要接 1。
pad_had_jdb_req_b	I	-	外部调试请求信号： 该信号是外部调试请求信号，异步于 tclk，低电平有效。所以在 CPU 处于低功耗状态或者复位状态时，该信号有效可以快速请求 CPU 进入调试模式。 不用该信号时，需要接 1。 注意：802 已废弃该信号，不支持该调试方式
had_pad_jdb_ack_b	O	1	处理器响应进入调试模式： 处理器进入调试模式后，该响应信号保持两个 tclk 的低电平。
had_pad_jdb_pm[1:0]	O	00	处理器工作模式指示信号： 这些输出信号表明处理器的工作模式,异步于 pad_had_jtg_tclk。 00: normal 模式； 01: STOP/DOZE/WAIT 模式； 10：调试模式； 11：保留。

图表 6-1 debug 端口列表

6.2 JATG 接口

1. pad_had_jtg_tap_en 与 had_pad_jtg_tap_on

pad_had_jtg_tap_en 信号为调试接口中 TAP 状态机的使能信号，维持该信号为高电平至少一个 JTAG 时钟周期可以使能调试接口中的 TAP 状态机。如果该信号在 CPU 上电之后一直无效，那么使用同步方式（如设置调试接口寄存器 HCR 中的 DR 位，断点等）使 CPU 进入调试状态时可能无法调试程序。

在 TAP 状态机启动之后，had_pad_jtg_tap_on 信号将拉高。

2. pad_had_jtg_tclk

JTAG 时钟信号。该信号为外部输入信号，一般为调试器产生的时钟信号，要保证该时

钟信号的频率低于 CPU 时钟信号的频率 1/2 才能保证调试模块与 CPU 核之间的正常工作。

3. pad_had_jtg_trst_b

pad_had_jtg_trst_b 信号为 JTAG 复位信号，可以复位 TAP 状态机以及其他相关控制信号。

4. JTAG_2 相关信号

pad_had_jtg_tms_i 信号为 2 线制 JTAG 串行数据输入信号，调试接口在 JTAG 时钟信号 TCLK 的上升沿对其采样，而外部调试器在 JTAG 时钟的下降沿设置该信号；

该信号在空闲时必须保持为高电平，同时空闲时时钟信号最好停止。用户可以利用该信号同步复位 HAD 逻辑：在实现 2 线制 JTAG 接口的调试模块中，如果时钟信号一直有效，用户只需保持该信号为高电平状态并维持 80 个时钟周期即可同步复位调试模块。复位调试模块后，调试模块的 TAP 状态机回到 RESET 态，同时调试模块寄存器 HACR 复位到 0x82（指向 ID 寄存器）。

had_pad_jtg_tms_o 信号为 2 线制 JTAG 串行数据输出信号，调试接口在 JTAG 时钟信号 TCLK 的下降沿对其设置，而外部调试器在 JTAG 时钟的上升沿对其采样；

had_pad_jtg_tms_oe 信号为 had_pad_jtg_tms_o 信号有效指示信号。CPU 外部应利用该信号将 pad_had_jtg_tms_i 和 had_pad_jtg_tms_o 信号合为一个双向端口信号。

7 低功耗系统集成

7.1 端口列表

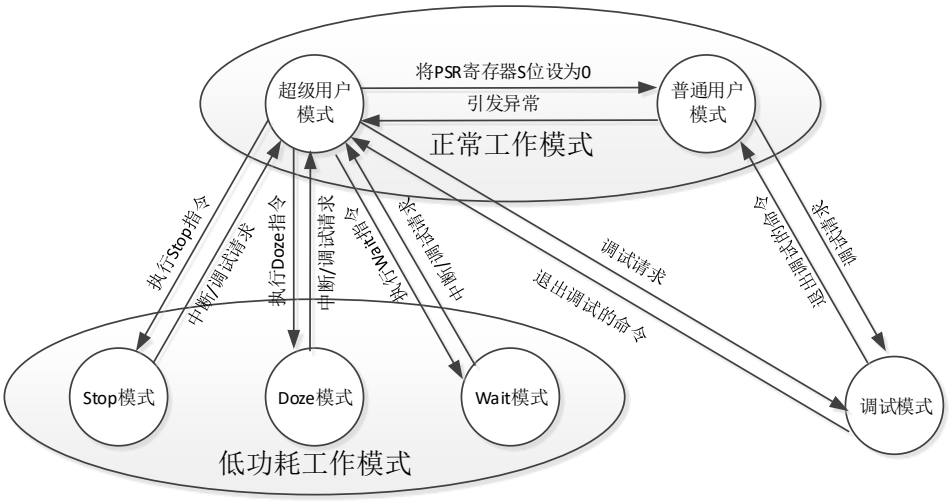
信号名	I/O	Reset	功能描述
sysio_pad_lpmd_b[1:0]	O	11	低功耗模式状态信号： 当处理器执行 doze, stop 或 wait 指令时，sysio_pad_lpmd_b[1:0]被相应的改变: 00：STOP； 01：WAIT； 10：DOZE； 11：normal。
sysio_pad_wakeup_b	O	1	处理器唤醒指示信号： 低电平表示 CPU 被唤醒且进入正常工作模式，这个信号将会在 pad_biu_intraw_b，pad_biu_fintraw_b ，pad_biu_brkrq_b[1:0]， pad_biu_dbgrq_b 或 pad_had_jdb_req_b 有效之后有效。
sysio_pad_ipend_b	O	1	异步唤醒确认信号： 指示异步唤醒请求，已经被处理器确认。
pad_cpu_intraw_b	I	-	异步唤醒信号： 不请求进入中断，用于将 CPU 从低功耗模式中唤醒。这个信号会使 sysio_pad_wakeup_b 和 sysio_pad_ipend_b 有效。 注：此信号仅实现于不配备矢量中断控制器的处理器中。

图表 7-1 低功耗端口信号列表

7.2 工作模式及其转换

CK-CPU 总共有三类工作模式：正常运行模式、低功耗工作模式和调试模式，如图表 7-2 CPU 状态转换图，其中低功耗工作模式分为三种：STOP 模式、DOZE 模式、WAIT 模式，这三种低功耗模式对于 CPU 来说都是一样的，即关闭内部 ICG，停止取指令和执行指令，唯一的区别就是向外传递的信号不一样，即 sysio_pad_lpmd_b[1:0]不一样，SOC 可以根据该信号定义不同的低功耗场景，通过 CPU 执行不同的低功耗指令并将相应的低功耗信息通过

sysio_pad_lpmd_b[1:0]传递给 SOC，实现系统的多场景低功耗设计。

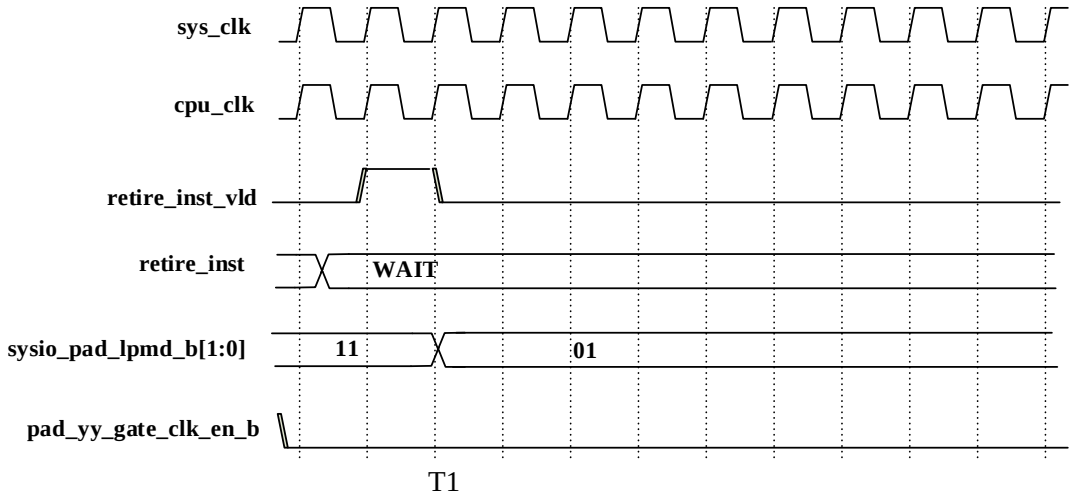


图表 7-2 CPU 状态转换图

7.3 进入低功耗模式握手

进入低功耗模式，由 CPU 执行指令发起。CPU 执行低功耗指令之后，比如下图中执行了 WAIT 之后，CPU 会关闭内部绝大部分寄存器时钟，除开少量始终采样的寄存器，CPU 内部的流水线停止运行，不再取指令和执行指令，同时还会修改 sysio_pad_lpmd_b[1:0]，通知 SOC 进行相应的低功耗操作，之后就一直处于等待状态直到被唤醒。要进入低功耗模式就需要对门控时钟进行全局使能，即 pad_yy_gate_clk_en_b 设置为 0。

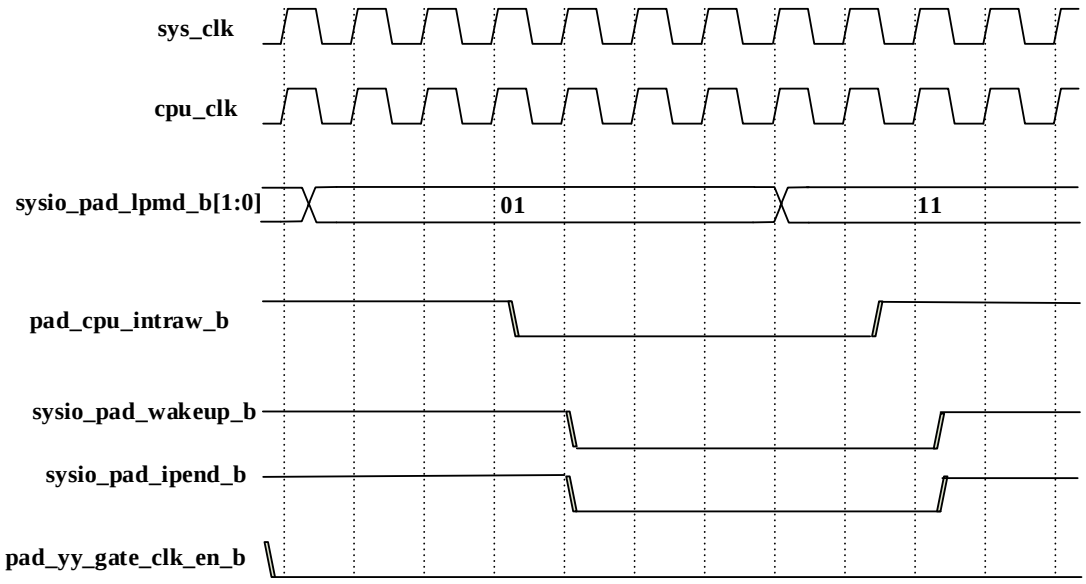
SOC 在集成时需要把 sysio_pad_lpmd_b[1:0]作为切换到低功耗模式的控制信号，由它来指示 SOC 进入不同级别的低功耗场景。



图表 7-3 进入低功耗模式握手

7.4 退出低功耗模式握手

退出低功耗模式由系统中断或者其他事件发起，SOC 通过拉低 pad_biu_intraw_b 来实现。这个信号仅实现唤醒，并不会使 CPU 进入中断，唤醒之后 CPU 继续执行低功耗之后的指令。外部模块可以通过查询 sysio_pad_lpmd_b[1:0]来查询 CPU 状态，通过 biu_pad_wakeup_b 来查询唤醒是否已被 CPU 接受，以此来作为唤醒信号 pad_biu_intraw_b 撤销的握手。



图表 7-4 退出低功耗模式握手

7.5 配合 SOC 不同低功耗场景

上面简单介绍了低功耗的握手机制，真实的 SOC 低功耗场景是比较复杂的，这里事例了三个场景供 SOC 设计者参考。该例子里为 CPU 定义了三个低功耗场景如下：

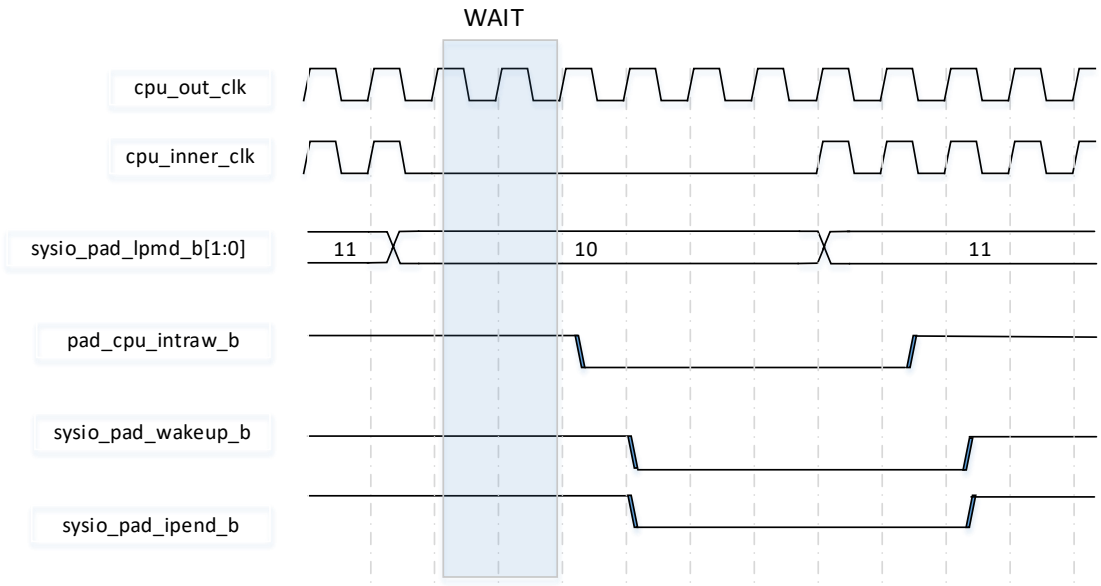
场景编号	控制标识	具体场景
1	WAIT	仅关掉 CPU 内部的 clock
2	DOZE	关掉 CPU 外部输入 clock
3	STOP	CPU 掉电

图表 7-5 低功耗三种模式

7.5.1 关内部 cpu clock

该低功耗场景下，CPU 外部的 clock 没有关掉，但是 CPU 内部的绝大多数寄存器的 clock 都被关掉了，该模式的特点就是有效减少时钟功耗，同时唤醒速度极快。只需要设置唤醒信

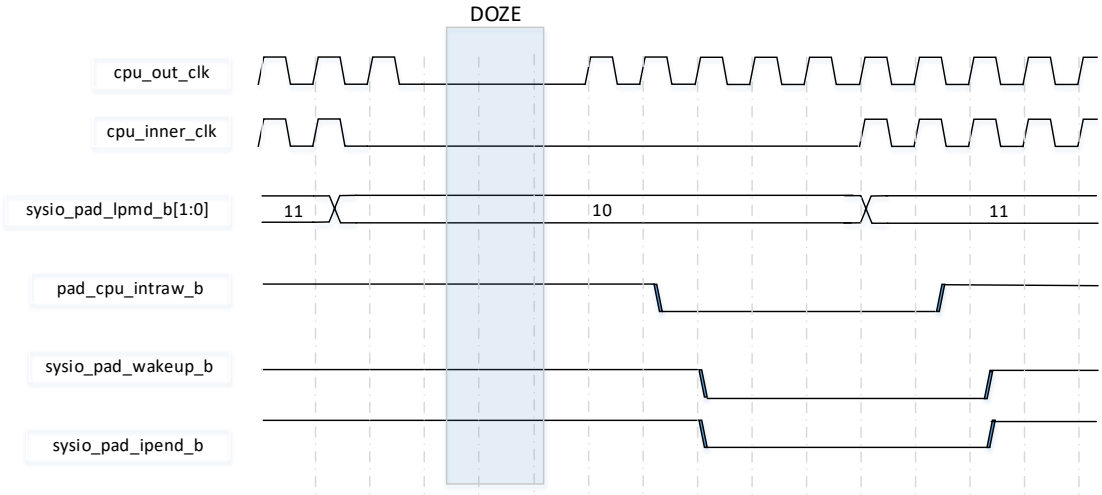
号就可以了，该模式的握手信号如下图所示：



图表 7-6 关闭 CPU 内部 clock

7.5.2 关外部 cpu_clock

该低功耗场景下，CPU 外部的时钟也被关掉，相当于整个 CPU 内部的所有寄存器的时钟都被关掉，该模式的特点就是功耗得到进一步减少，同时唤醒速度也比较快，在置起唤醒信号之前要先给 CPU 送稳定的 clock，该模式的握手信号如下图所示：



图表 7-7 关闭 CPU 外部 clock

7.5.3 Power gating

该低功耗场景下，不但整个 CPU 内部的所有寄存器的 clock 都被 gating 掉了，连 CPU 所在的 power domain 也被 power gating 掉了。该模式的特点是功耗降到最低，但是唤醒速度

非常慢，而且唤醒后无法简单的继续执行，所以在这之前需要先保存 CPU 现场，完成之后才能进入该低功耗模式。除此之外，该模式的低功耗唤醒需要走 power gating 之后恢复的流程，并且还需要软件配合恢复现场，最终才能继续执行程序。这里把该模式的流程列如下：

- CPU 首先进行现场保存，然后可能需要对 PMU 进行一些设置，比如设置 Power Down Flag，配置结束后，CPU 调用 STOP 低功耗指令，进入低功耗状态。
- PMU 会不断查询 CPU 是否已经进入低功耗状态。
- 一旦 CPU 检测到 sysio_pad_lpmd_b[1:0]为 STOP 状态，PMU 会关掉 Clock
- PMU 发起 CPU Reset。
- PMU 使能 Isolation。
- PMU 使能 Power Down Swtich。
- PMU 等待 Power Down ACK。
- 进入掉电状态，PMU 等待外部唤醒中断。
- 一旦监测到唤醒中断，PMU Power On Swtich。
- PMU 等待 Power On ACK。
- PMU 禁止 Isolation。
- PMU 打开 Clock
- PMU 释放 CPU Reset。

CPU 进入 Reset 异常，检查 Power Down Flag，发现该位是 1，发现这次启动不是重启，而是 power gating 的低功耗唤醒，CPU 就执行相应程序恢复现场，跳到相应地址继续执行程序。

总之，SOC 设计者需要在唤醒时间开销和降低功耗之间做出权衡，根据软件应用场景设计相应的低功耗模式。CSKY CPU 提供了三种低功耗模式，上面的例子中也只有三种模式，但实际上 SOC 设计者可以自己在 PMU 增加额外寄存器实现更多的低功耗场景，在执行低功耗指令之前先配置这些寄存器即可。

8 安全机制集成

8.1 端口列表

信号名	I/O	Reset	定义
处理器安全机制信号：			
iu_pad_security_violation	O	0	安全违反信号： 指示处理器处于安全违反状态。该信号为 1 表示处理器安全违反，0 表示没有发生安全违反。
pad_cpu_gpr_rst_b	I	-	通用寄存器复位信号： 用于指示将 CK802 处理器所有通用寄存器同步复位。该信号低电平有效。
pad_seu_rand_num[31:0]	I	-	随机数信号： 指示真随机数。真随机数需要在该安全机制使能前开始指示，至安全机制关闭后为止。安全机制使能时，真随机数在每个周期都应当产生变化。
总线接口安全机制信号：			
pad_biu_beu_key0[23:0]	I	-	总线加扰机制密钥信号： 从系统中输入给处理器的密钥信号，在总线加扰机制下使用的两轮 F 函数中，作为第一轮 F 函数的加密密钥。
pad_biu_beu_key1[23:0]	I	-	总线加扰机制密钥信号： 从系统中输入给处理器的密钥信号，在总线加扰机制下使用的两轮 F 函数中，作为第二轮 F 函数的加密密钥。
biu_pad_haddr_pol	O	0	总线传输地址极性翻转信号： 标记该次读写传输地址是否进行了极性翻转。（仅在配置总线极性反转位时有效）

信号名	I/O	Reset	定义
biu_pad_hwdata_pol	O	0	总线写数据极性翻转信号： 标记该次写数据是否进行了极性翻转。（仅在配置总线极性反转位时有效）
pad_biu_hrdata_pol	I	-	总线读数据极性翻转信号： 标记该次读数据是否进行了极性翻转。（仅在配置总线极性反转位时有效）
iahbl_pad_haddr_pol	O	0	总线传输地址极性翻转信号： 标记该次读写传输地址是否进行了极性翻转。（仅在配置总线极性反转位时有效）
iahbl_pad_hwdata_pol	O	0	指令总线写数据极性翻转信号： 标记该次写数据是否进行了极性翻转。（仅在配置总线极性反转位时有效）
pad_iahbl_hrdata_pol	I	-	指令总线读数据极性翻转信号： 标记该次读数据是否进行了极性翻转。（仅在配置总线极性反转位时有效）
dahbl_pad_haddr_pol	O	0	总线传输地址极性翻转信号： 标记该次读写传输地址是否进行了极性翻转。（仅在配置总线极性反转位时有效）
dahbl_pad_hwdata_pol	O	0	数据总线写数据极性翻转信号： 标记该次写数据是否进行了极性翻转。（仅在配置总线极性反转位时有效）

信号名	I/O	Reset	定义
pad_dahbl_hrdata_pol	I	-	数据总线读数据极性翻转信号： 标记该次读数据是否进行了极性翻转。（仅在配置总线极性反转位时有效）
biu_pad_hwdata_par	O	0	总线写数据奇偶校验信号： 记录当前写数据的奇偶校验位。（仅在配置总线奇偶校验机制时有效）
pad_biu_hrdata_par	I	-	总线读数据奇偶校验信号： 记录当前读数据的奇偶校验位。（仅在配置总线奇偶校验机制时有效）
iahbl_pad_hwdata_par	O	0	指令总线写数据奇偶校验信号： 记录当前写数据的奇偶校验位。（仅在配置总线奇偶校验机制时有效）
pad_iahbl_hrdata_par	I	-	指令总线读数据奇偶校验信号： 记录当前读数据的奇偶校验位。（仅在配置总线奇偶校验机制时有效）
dahbl_pad_hwdata_par	O	0	数据总线写数据奇偶校验信号： 记录当前写数据的奇偶校验位。（仅在配置总线奇偶校验机制时有效）
pad_dahbl_hrdata_par	I	-	数据总线读数据奇偶校验信号： 记录当前读数据的奇偶校验位。（仅在配置总线奇偶校验机制时有效）

图表 8-1 安全机制接口信号描述

注意：接口信号仅在其安全机制硬件配置时有效。

8.2 安全机制硬件自测试

S802 处理器安全扩展单元配有内建自测单元，用于测试安全机制的有效性。有关于时间的安全机制，可通过观测程序时间确认其有效性，该内建自测单元主要负责校验类安全机

制的有效性确认。

安全内建自测寄存器地址空间如下所示：

地址	名称	类型	初始值	描述
0xE0010000	SBER	读/写	0x00000000	安全内建自测使能制寄存器
0xE0010004	SBPR	只读	0x00000000	安全内建自测极性观测寄存器

图表 8-2 安全内建自测寄存器定义

8.2.1 安全内建自测使能寄存器（SBER）

31 6	5	4	3	2	1	0
Reserved	POL_EN	PC_EN	ICB_EN	PIPE_EN	CR_EN	GPR_EN

图表 8-3 安全内建自测使能寄存器

SBER 寄存器的位说明如下所示：

位	名称	描述
31:6	Reserved	保留
5	POL_EN	数据通路极性观测使能 0：不观测 1：观测，处理器回写 GPR 时，将带有极性的回写数据回写 SBPR（软件可通过读取 SBPR 观测数据极性）。
4	PC_EN	PC 校验攻击使能 0：不攻击。 1：攻击，执行指令时触发 PC 校验失败，处理器报错并挂起
3	ICB_EN	互补备份校验攻击使能 0：不攻击。 1：攻击，执行指令时触发互补备份校验失败，处理器报错并挂起
2	PIPE_EN	流水线校验攻击使能 0：不攻击。 1：攻击，执行指令时触发流水线校验失败，处理器报错并挂起
1	CR_EN	CR 校验攻击使能 0：不攻击。 1：攻击，执行指令时触发 CR 校验失败，处理器报错并挂起

位	名称	描述
0	GPR_EN	GPR 校验攻击使能 0：不攻击。 1：攻击，执行指令时触发 GPR 校验失败，处理器报错并挂起

图表 8-4 安全内建自测使能寄存器域描述

8.2.2 安全内建自测极性观测寄存器（SBPR）

31	1	0
Reserved		POL_VAL

图表 8-5 安全内建自测极性观测寄存器

SBPR 寄存器的位说明如下所示：

位	名称	描述
31:1	Reserved	保留
0	POL_VAL	数据通路极性观测位 若 SBER 中 POL_EN 置 1，处理器回写 GPR 时，POL_VAL 位保存带极性回写数据的最低位，通过软件读取 SBPR 时，将不做极性解扰，因此可以通过读取 SBPR 确认处理器内部数据通路极性有效性

图表 8-6 安全内建自测极性观测寄存器域描述

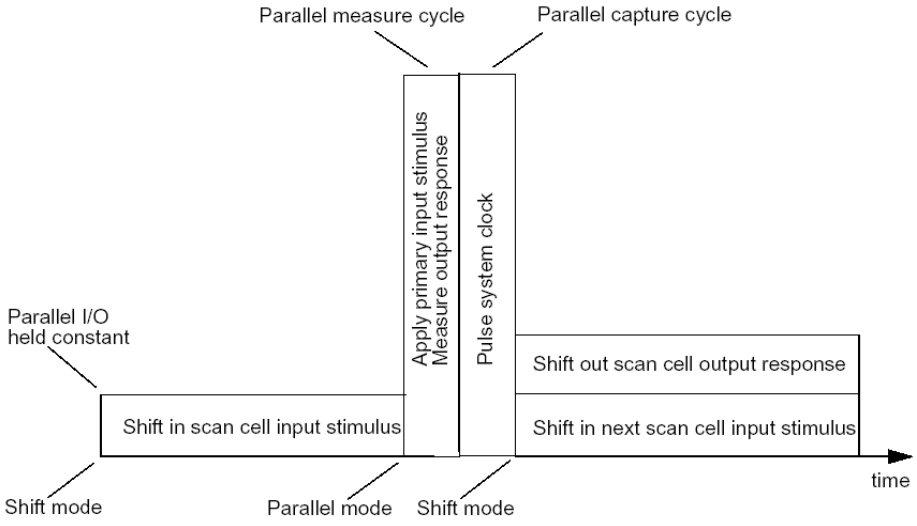
9 DFT 系统集成

根据用户的具体需求，如果是 C-SKY 硬核授权客户，通常硬核已经集成了 DFT 测试结构以及 Memory Bist 测试结构。如果是 C-SKY 软核授权客户，这部分工作需要用户自己去做，本章节将给出所需要的必要信息。图表 9-4 普通扫描链测试的信号引脚。

9.1 DFT 系统

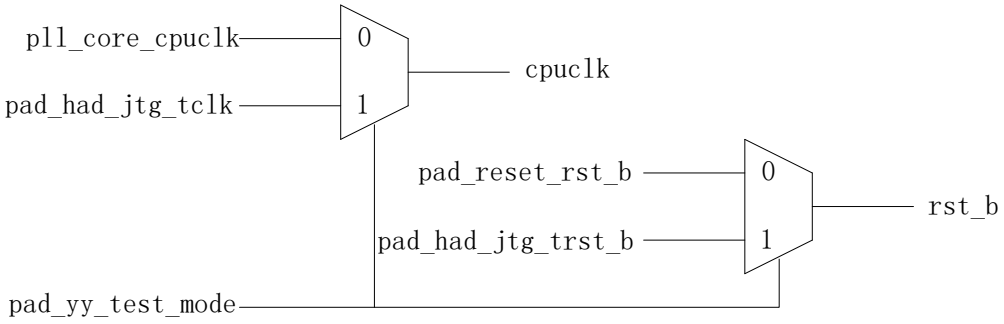
Design-For-Test (DFT) ，是在芯片中集成扫描链结构的相关设计。在芯片制造完成之后，可以在测试机上实现快速高效的芯片功能测试。DFT 方法常用在时序电路的设计和测试中，基本的方法是用可扫描的触发器替换电路中的一般触发器。然后按照一定顺序连接起来构成扫描链(Scan Chain)。在测试时，先在测试模式（test mode）把激励数据逐个按时钟送入，继而转换到电路的功能模式（function mode），且把芯片的基本输入管脚上加上激励信号，只需要施加一个时钟周期，使电路逻辑生效，然后再转入测试模式，把结果逐个通过扫描链送出，同时捕获芯片的基本输出管脚的值，将实际结果与预期结果比较，就可以达到测试的目的。这就是基于扫描链 DFT 的基本方法。

扫描链测试由移入(shift-in)、捕捉(capture)和移出(shift-out)三个阶段构成，如下图所示。**扫描链测试的阶段由 scan_en 信号控制：当 scan_en 为 1 时，它处于 shift-in 和 shift-out 的阶段；当 scan_en 为 0 时，它处于 capture 阶段**。 Shift-in 阶段，测试向量将会经由 Scan_in 端口进入芯片，并通过测试链逐拍送到所有的寄存器；Capture 阶段，芯片将返回功能模式，所有寄存器的数值，经由功能模式的路径，到达其后的寄存器，这个过程只需要一个周期；Shift-out 阶段，所有寄存器的数值，仍然通过扫描链结构，经由 Scan_out 端口移出芯片，返回测试机。通过返回的数值和测试向量中的预期结果对比，可判断芯片功能是否正确。



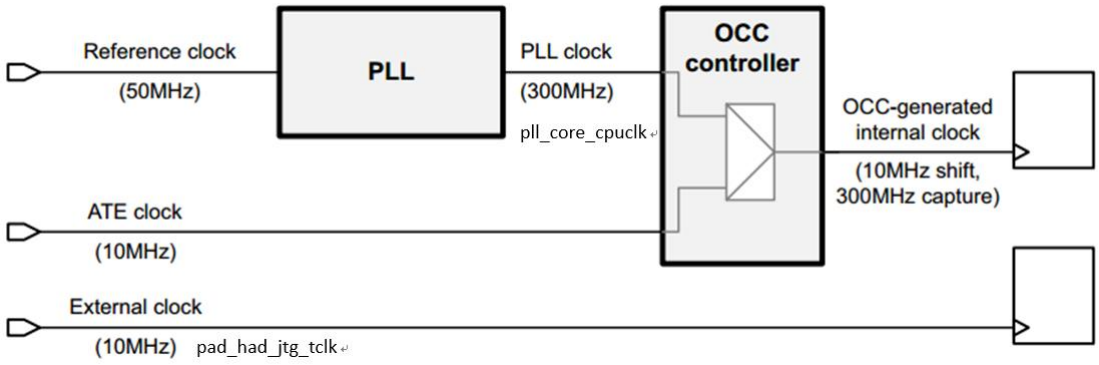
图表 9-1 扫描链测试过程

Stuck-at 测试：Stuck-at faults 指在芯片中出现的逻辑功能错误，即 0/1 的逻辑数值发生错误。这种测试的目标在于测试出芯片功能是否正确，而不在乎芯片频率是否达标。因此，在这种测试中，S802 只需要工作在一个低速时钟（Test clock）下。将来自 PLL 的高速时钟切换成低速的测试时钟，通过 pad_yy_test_mode 信号分别选择 pad_had_jtg_tclk 和 pad_had_jtg_trst_b 作为测试时的时钟信号和复位信号，如下图所示：



图表 9-2 普通扫描链测试时的时钟和复位信号选择

At-speed 测试：At-speed 测试目的是为了判断芯片频率是否达标。在高速的设计中，At-speed 测试是必需的。与 Stuck-at 测试的主要区别在于，Capture 阶段，芯片的时钟需要切换回高速的 Function clock。如此一来，达不到频率要求的 path，会出现数值错误，从而在下一个 Shift-out 阶段被检查出来。为了实现低速时钟和高速时钟切换的功能，在 S802 中，将会集成一个 OCC (On-chip-clocking) 模块，由此模块提供一个频率来回切换的时钟信号。At-speed 模式下的时钟结构示意图如下：



图表 9-3 At-speed 测试的时钟信号

Scan Compression 测试压缩：对于应用了 At-speed 测试的设计，如果设计规模比较大，测试向量将会非常冗长，造成测试时间和成本的上升。测试压缩能够大幅缩短测试向量的长度，只牺牲少量可接受的测试覆盖率。其原理为：加入一个 De-Compressor，用来将输入的向量转译为普通扫描链的向量；一个 Compressor，用来将扫描链输出的向量重新压缩。S802 在集成 DFT 时，如果加入了 At-speed 测试结构，那么也会同时加入 Scan Compression 结构。

Wrapping Cores：S802 的硬核测试，会同时以门级网表的形式提供一个 DFT wrapper，**客户需要将此结构加入 SOC 中，并进行逻辑综合，转换到 SOC 的 Target library 下。**DFT wrapper

内自建有完整的扫描链，扫描链中的寄存器会将 S802 的功能引脚包裹起来。其所提供的功能为：**在 S802 测试阶段（Wrp_if mode），wrapper 将能够向 S802 IN 管脚提供测试向量，或收集 OUT 管脚上的测试结果，从而覆盖到硬核接口上的逻辑测试；相应的，在 SOC 的测试阶段（Wrp_of mode），wrapper 也能够覆盖 SOC 到 S802 的接口逻辑。在正常的功能模式（Function mode）中，wrapper 则可完全旁路，不影响 SOC 与 S802 的正常交互**。在 S802 硬核提供的 DFT wrapper 中，不同的工作模式由 mode 信号控制，mode 信号的相应赋值可在图表 8-5 中查到。

9.1.1 Wrapped S802 测试引脚一览

下表列出了所有 DFT 相关的端口信号，以及相应的功能说明，请注意应用一栏的备注，根据客户需求的不同，相应的信号不一定出现在提供的硬核中。

信号名	功能描述	应用
pad_yy_test_mode	测试模式使能信号（Test Mode），在测试模式下均赋值为 1	Basic DFT Signal
pad_yy_scan_enable	扫描使能信号（Scan Enable），具体赋值由测试向量控制	Basic DFT Signal
pad_had_jtg_tclk	扫描测试时钟信号（Scan Clock），与测试机时钟信号相连。 Stuck-at 模式下，控制 S802 中全部寄存器； At-speed 模式下，仅控制 S802 中的低速逻辑，不穿过 OCC。	Basic DFT Signal
pad_had_jtg_trst_b	扫描测试复位信号（Reset），Active=0	Basic DFT Signal
xx_top_si_1… xx_top_si_*	扫描链输入（Scan In） S802 对应命名方式： S802：nm_	Basic DFT Signal
xx_top_so_1… xx_top_so_*	扫描链输出（Scan Out），命名方式同上	Basic DFT Signal
pll_core_cpuclk	S802 的功能时钟，来自 PLL。 Stuck-at 模式下，功能时钟完全旁路； At-speed 模式下，用作高速时钟（Fast Clock），穿过 OCC。	At-speed Signal
ate_clock	At-speed 模式下的低速时钟（Slow Clock），穿过 OCC	At-speed Signal
pll_reset	OCC pll 复位（reset）信号	At-speed Signal
pll_bypass	OCC pll 旁路（bypass）信号	At-speed Signal

scan_compression_enable	测试压缩使能信号（Active=1）。由测试向量控制。	Scan Compression
wrp_clock	Wrapper 时钟信号，控制 wrapper 内寄存器	Wrapper Signal
wrp_shift	Wrapper Shift 模式使能信号	Wrapper Signal
wrp_si_*	Wrapper Scan In	Wrapper Signal
wrp_so_*	Wrapper Scan Out	Wrapper Signal
mode1，mode2，mode3	Wrapper 工作模式控制信号	Wrapper Signal

图表 9-4 普通扫描链测试的信号引脚

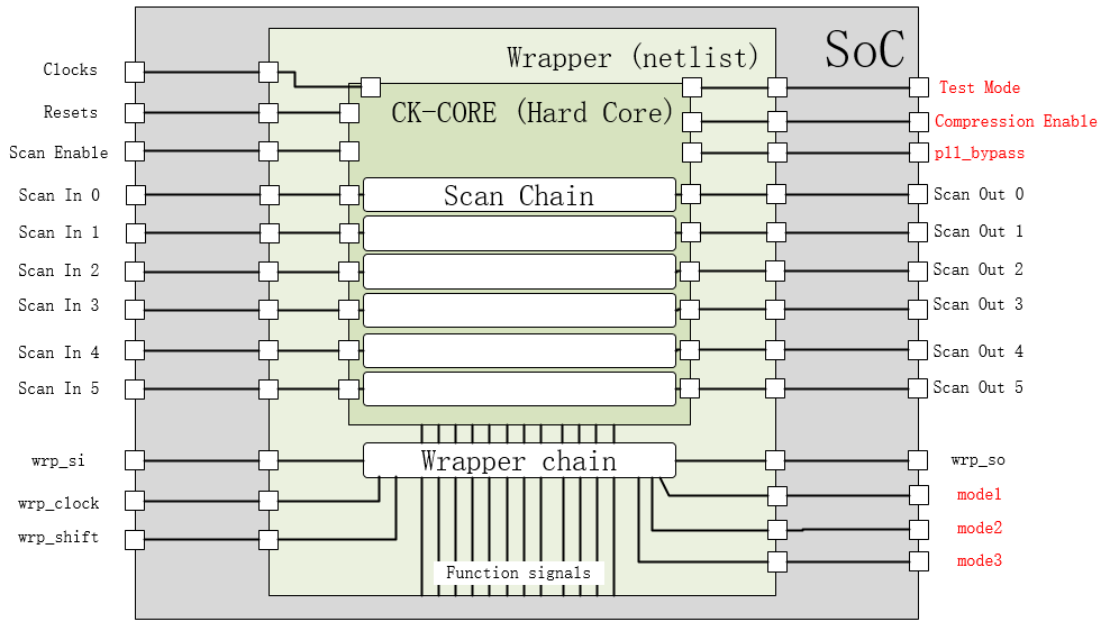
PIN NAME	wrp_of	wrp_if	Function
mode1	**0**	**1**	**0**
mode2	**1**	**1**	**0**
mode3	**0**	**0**	**0**
xx_top_si	0	scan_in	0
xx_top_so	float	scan_out	float
pad_had_jtg_tclk	0	scan_clk	jtg_tclk
pad_had_jtg_trst_b	1	scan_reset	jtag_rst_b
pad_yy_test_mode	0	1	0
pad_yy_scan_enable	0	scan_enable	0
wrp_si	wrp_si	wrp_si	0
wrp_so	wrp_so	wrp_so	float
wrp_shift	soc_scan_enable	wrp_shift	0
wrp_clock	soc_scan_clk	wrp_clock	0
pll_core_cpuclk	0	fast clock	func clock
ate_clock	0	slow clock	0
pll_reset	1	occ reset	1
pll_bypass	0	0	0
scan_compression_enable	x	1	x

图表 9-5 信号在不同模式下的赋值示例

图表 8-5 为不同模式下的 DFT 测试信号赋值示例。注意，用户可以选择将已存在的 wrapper chain 并入 SOC 的扫描链，即采用 wrp_of 模式的测试方法；也可以忽略 wrapper chain，将 wrapper 中的寄存器当作普通寄存器重新串入 SOC 的扫描链，此时 wrp_of 模式用户只需要保证 wrp_clock 和 mode 信号即可，wrp_si、wrp_shift 等信号不起作用。

9.1.2 在 SOC 中集成 Wrapped S802 硬核

在 SOC 中集成 Wrapped S802 硬核时，需要将 S802 wrapper 上的 DFT 测试信号全部连接至 SOC 的 PAD，用户必须**保证测试向量中出现的信号所连接的 IO 功能正确，确保在测试机台上测试向量全部到达 S802**，如图 8-6 所示。当 SOC 的 IO 数量受限时（IO-Limited），用户可通过 IO 复用来接出 DFT 测试所需的信号。另外，图 8-6 中标示为红色的信号，因为此类信号在测试过程中，通常保持常值不变，用户也可以通过编码的方式进行控制，来减少 IO 数量，但需保证到达 S802 相应端口的数值正确。



图表 9-6 SOC 集成示意图

9.1.3 带有 OCC 模块的 S802 在 CPU bist 模式下的赋值

Wrapper 信号连接：

信号名	功能描述
mode1	0
mode2	0
mode3	0

OCC 模块信号连接：

pad_yy_test_mode	pll_bypass	pad_yy_scan_enable	说明
0	x	0	pll_clock bist 测试模式(高速模式)
1	1	0	ate_clock bist 测试模式（低速模式）

9.1.4 后仿测试 Wrapped S802 示例

```
module ck_core_test_top;                                    //module name
// Inputs
wire core_si_x, wrp_si, test_mode, …                        //input pins for test
wire pad_biu_hresp, pad_biu_hready, pad_biu_vec_b, …        //other functional input pins
reg core_si_x_REG, wrp_si_REG, wrp_clock_REG, test_mode_REG, …
                                                            //registered input pins for test
reg pad_biu_hresp_REG, pad_biu_hready_REG, pad_biu_vec_b_REG, …
                                                            //registered functional input pins
// Outputs
wire core_so_x, wrp_so                                      //output pins for test
wire biu_pad_haddr, biu_pad_htrans, biu_pad_gcb, …          //other functional output pins
// Input Register assignments
assign core_si_x = core_si_x_REG;
assign wrp_si = wrp_si_REG;
assign test_mode = test_mode_REG;
assign pad_biu_hresp = pad_biu_hresp_REG;
assign pad_biu_hready = pad_biu_hready_REG;
assign pad_biu_vec_b = pad_biu_vec_b_REG;
…
ck_core_top U1(                                             //instance the wrapped CK-Core top module
 .core_si_x(core_si_x),   .wrp_si(wrp_si),   .test_mode(test_mode), …
 .pad_biu_hresp(pad_biu_hresp), .pad_biu_hready(pad_biu_hready), .pad_biu_vec_b(pad_biu_vec_b), …
 .core_so(core_so), .wrp_so(wrp_so),
 .biu_pad_haddr(biu_pad_haddr), .biu_pad_htrans(biu_pad_htrans), .biu_pad_gcb(biu_pad_gcb), … );

Intial begin                                                //main test procedure
      $timeformat();
      $STILDPV_setup();
      while(!$STILDPV_done()) #($STILDPV_run());
end

endmodule
```

图表 9-7 用户拿到的 Testbench 示例

用户在后仿测试 Wrapped S802 时，需要对 C-SKY 提供的测试向量做相应的修改，即将上图 8-7 中红色的部分修改为下图 8-8 中紫色的部分，注意，图中仅为示意，用户需要结合 SOC 中实际的命名与连接方式进行修改。

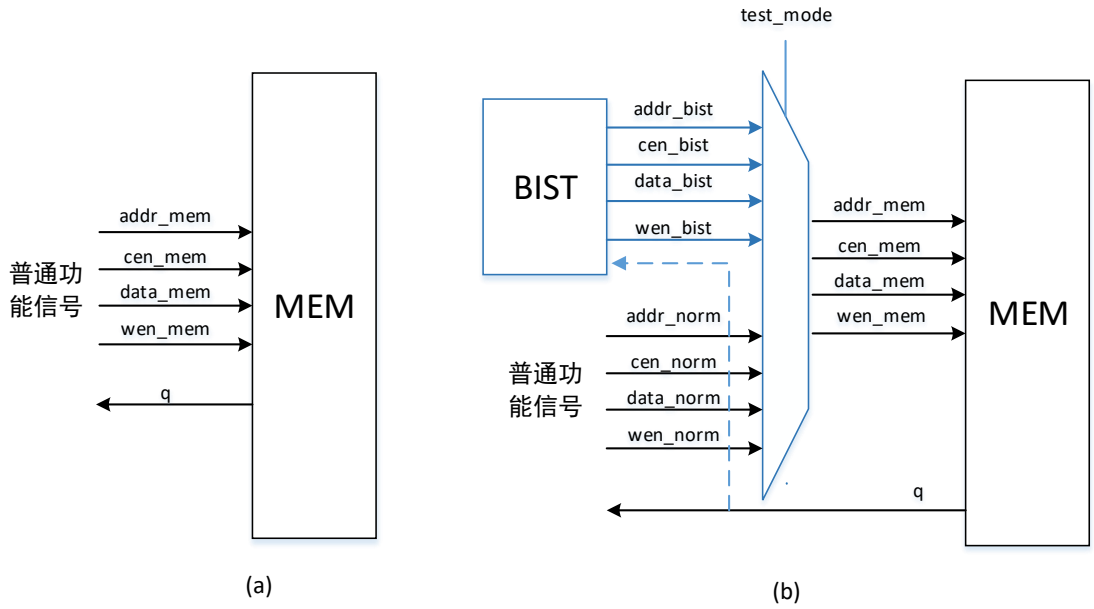
```
module ck_core_test_top;                                    //module name
// Inputs
wire core_si_x, wrp_si, test_mode, …                        //input pins for test
wire pad_biu_hresp, pad_biu_hready, pad_biu_vec_b, …        //other functional input pins
reg core_si_x_REG, wrp_si_REG, wrp_clock_REG, test_mode_REG, …
                                                            //registered input pins for test
reg pad_biu_hresp_REG, pad_biu_hready_REG, pad_biu_vec_b_REG, …
                                                            //registered functional input pins
// Outputs
wire core_so_x, wrp_so                                      //output pins for test
wire biu_pad_haddr, biu_pad_htrans, biu_pad_gcb, …          //other functional output pins
// Input Register assignments
assign core_si_x = core_si_x_REG;
assign wrp_si = wrp_si_REG;
assign test_mode = test_mode_REG;
assign pad_biu_hresp = pad_biu_hresp_REG;
assign pad_biu_hready = pad_biu_hready_REG;
assign pad_biu_vec_b = pad_biu_vec_b_REG;
…
soc_top U1(                                                 //instance the soc top module
 .i_pad_uart_sin1(core_si_x),   .i_pad_uart_sin2(wrp_si),   .i_pad_uart_sin3(test_mode), …
 .o_pad_uart_sout1(core_so), .o_pad_uart_sout2(wrp_so),
 .i_pad_ac97_bitclk(), .i_pad_ac97_sdata(), .o_pad_lcdc_hsync(), .o_pad_lcdc_vsync(),
 .b_pad_gpioa(), .b_pad_gpiob(), … );

Intial begin                                                //main test procedure
      $timeformat();
      $STILDPV_setup();
      while(!$STILDPV_done()) #($STILDPV_run());
end

endmodule
```

图表 9-8 用户修改后的 Testbench 示例

9.2 内建自测试

9.2.1 EDA 工具插入 MBIST

目前业界主流 EDA 工具都支持自动插入 MBIST，工具会自动查找到 RAM 的端口，并插入的 MBIST，如下图所示，(a)图所示为 MEM 的基本功能信号接口图，(b)图为插入 BIST 逻辑之后，工具在插入 BIST 时会 MUX 上自己的 BIST 单元，通过 test_mode 信号来选择是测试模式还是普通模式。



图表 9-9 EDA 工具自动插入 Mbist

我们**推荐**用户自己使用 EDA 工具插入 MBIST，因为这些主流工具的算法不断在更新，可以得到更高的覆盖率。如果用户选择自己插 MBIST 需要在综合时将中天的 MBIST 输入信号(pad_yy_bist_tst_en)绑死为 0，输出悬空，综合工具会将中天的 MBIST 逻辑给优化掉。

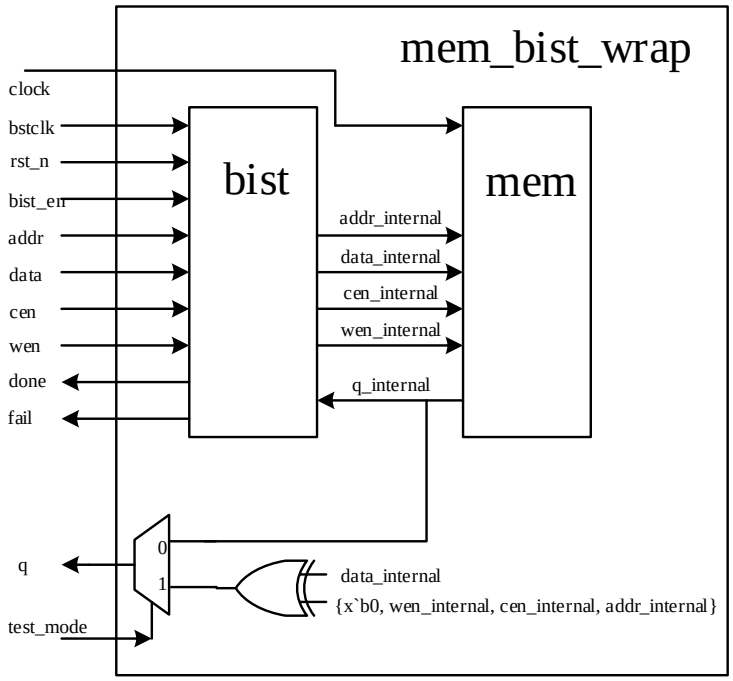
9.2.2 中天 SMBIST 单元结构图

如果用户不愿自己插 MBIST 逻辑，也可以使用中天公司提供的 SMBIST（**不推荐**）。

9.2.2.1 中天 SMbist 算法说明

SMBIST 测试输入算法采用软件模拟的 March X 算法，具体如下：

1) 从零地址开始以升序写 0 直到最高地址；
2) 从最高地址以降序读 0、写 1，直到零地址；
3) 从零地址开始以升序读 1、写 0、读 0、写 1，直到最高地址；
4) 从零地址开始以升序读 1、写 0，直到最高地址；
5) 从零地址开始以升序读 0、写 1、读 1、写 0，直到最高地址；
6) 从零地址开始以升序读 0，直到最高地址；



图表 9-10 中天 SMBIST 单元结构图

9.2.2.2 SMBIST 信号说明

No.	Signal	Description
1	CLK	Mem 模块时钟信号
2	bstclk	BIST 模块时钟信号
3	b_rst_n	复位信号（低电平有效）
4	pad_yy_bist_tst_en	BIST 测试使能信号
5	icache_data_array0_smbist_done	Array0 BIST 测试结束信号
6	icache_data_array0_smbist_fail	Array0 BIST 测试错误信号
7	icache_data_array1_smbist_done	Array1 BIST 测试结束信号
8	icache_data_array1_smbist_fail	Array1 BIST 测试错误信号
9	icache_data_array2_smbist_done	Array2 BIST 测试结束信号
10	icache_data_array2_smbist_fail	Array2 BIST 测试错误信号
11	icache_data_array3_smbist_done	Array3 BIST 测试结束信号
12	icache_data_array3_smbist_fail	Array3 BIST 测试错误信号
13	pad_yy_bist_tset_mode	测试模式使能信号

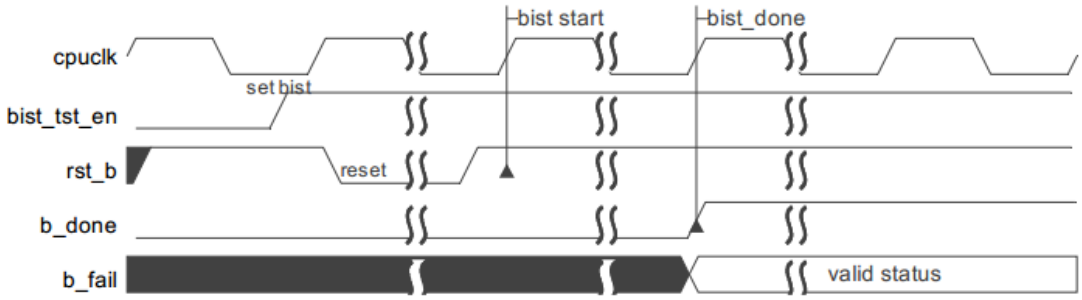
图表 9-11 Mbist 端口信号说明

9.2.2.3 SMBIST 过程说明

BIST 测试具体过程如下：

1) 启动 Mbist：置 bist 使能位（pad_yy_bist_tst_en = 1）
2) 设置 pad_yy_test_mode 为 0，选择 cpuclk 和 b_rst_n 复位状态（b_rst_n 先置 0 后置 1）
3) Mbist 开始工作

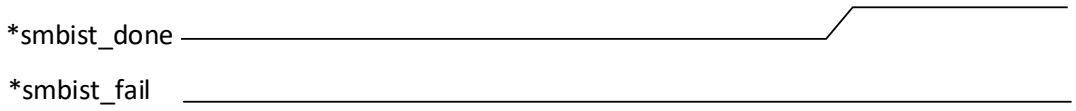
一个完整的 Mbist 过程如下图：



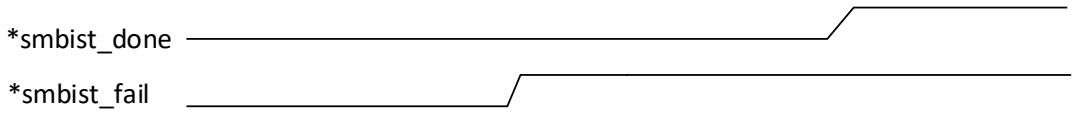
图表 9-12 中天 SMBIST 过程图

9.2.2.4 SMBIST 结果查询

当 MBIST 完成时，*smbist_done 会变成高电平，此时，*smbist_fail 的值将指示该 MBIST 的结果状态。*smbist_fail 结果状态如下:



图表 9-13 MBIST pass 状态图



图表 9-14 MBIST fail 状态图

10 CPU 运行观测信号集成

10.1 简介

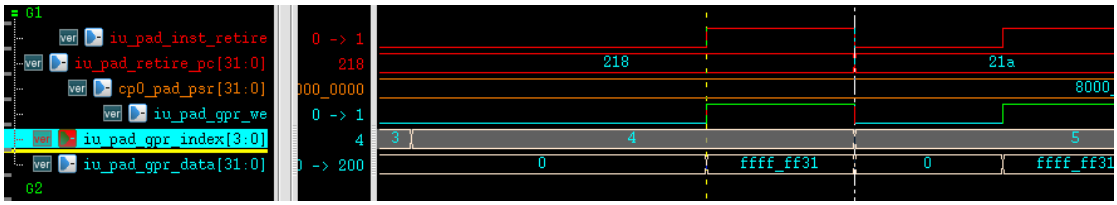
CPU 运行观测信号是提供给 SOC 集成设计人员观测所设计，不能用来集成，是悬空信号。SOC 设计人员可以通过监测这些信号来获知 CPU 的指令回写信息和寄存器的相关值。

10.2 通用观测信号

信号名	I/O	Reset	定义
iu_pad_inst_retire	O	0	指令退休指示信号： 0：当前周期没有指令退休 1：当前周期有指令退休
iu_pad_retire_pc[31:0]	O	-	退休指令的 PC： 表明当前正在退休的指令的 PC
iu_pad_inst_split	O	0	指令类型指示信号： 0：当前正在退休的指令不是拆分指令 1：当前正在退休的指令是拆分指令
iu_pad_gpr_we	O	0	通用寄存器回写指示信号 0：当前周期不回写通用寄存器 1：当前周期回写通用寄存器
iu_pad_gpr_index[4:0]	O	-	寄存器索引 表示当前回写的通用寄存器的索引
iu_pad_gpr_data[31:0]	O	-	寄存器回写总线 表示当前回写通用寄存器的值
cp0_pad_psr[31:0]	O	-	处理器状态寄存器 表示处理器当前状态的值，具体参考 S802 用户手册

图表 10-1 通用观测信号

10.3 观测信号示例波形



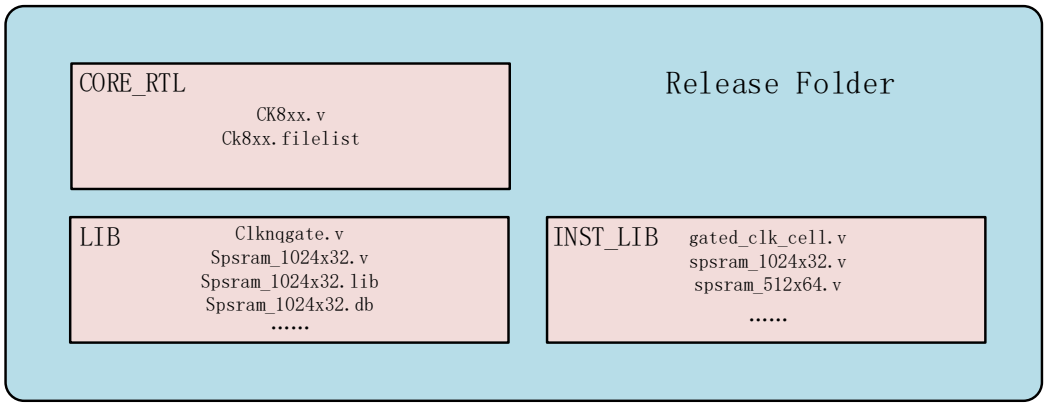
图表 10-2 观测信号示例波形

如图表 10-2 观测信号示例波形所示，光标所指区间 iu_pad_retire_pc[31:0]的 value 为 218，iu_pad_inst_retire 为高，表示 218 这条 PC 所指代的指令在该时钟周期退休，此时 iu_pad_gpr_we 为高，表示这条指令退休时，回写了通用寄存器。这时要观测回写了哪个寄存器和回写寄存器的值，于是拉出 iu_pad_gpr_index[4:0]与 iu_pad_gpr_data[31:0]两个回写通用寄存器信号，图中 iu_pad_gpr_index[4:0]的 value 为 4，iu_pad_gpr_data[31:0]的 value 为 ffffff31 表示，在此时钟周期往 4 号通用寄存器回写了数据 ffffff31。

11 工艺映射

11.1 软核授权 RTL 代码结构

C-SKY 软核授权客户都可以得到如下文件夹，内部结构如下：



图表 11-1 release 文件夹内容

CORE_RTL 目录：包含了内核代码 E8xx.v，以及一个 file list 文件 E8xx.filelist，里面包含所有的.v 文件，综合时只需要用该 filelist 中表示的文件路径，就能把所有需要的.v 文件编译进去。

INST_LIB 目录：所有 memory instrance 文件和 gated cell 的 instance 文件。

LIB 目录：调用的所有 memory 和 gated cell 仿真模型或者库。

11.2 ASIC 映射

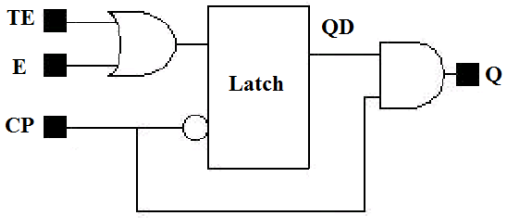
11.2.1 ICG 替换

11.2.1.1 文件所在位置

所需要修改的文件位于 INST_LIB 目录，文件名为 gated_clk_cell.v，该文件里例化了特定工艺下的 gated cell。

11.2.1.2 选择 gated cell

中天所用的 geted cell 都是上升沿有效，其逻辑图和真值表如下：



Truth Table

INPUT			OUTPUT
TE	CP	E	Q
0	⌐	0	0
0	⌐	1	1
1	⌐	x	1
x	0	x	0
x	1	x	QD

图表 11-2 正沿触发 gated cell

一般工艺库都会有两类 gated cell，一个是正沿触发而另一类是负沿触发的，注意需要选择和以上真值表一样的正沿的 gated cell。

一般工艺库里同一类有不同规格的 gated cell，这就需要用户根据自己的需求进行选择，平衡速度和功耗等因素，如下图是 TSMC28 工艺下正沿触发 gated cell 的规格：

Cell Information

(Characterization Condition:Process=Slow-Slow-Global,Voltage=0.72v,Temp=125degreeC)

PG Pin=VDD

Cell Name	Gate Count	Width(um)	Leakage(nW)		
			Min.	Ave.	Max.
CKLNQD12BWP30P140	12	3.92	204.402	245.23958	289.03
CKLNQD16BWP30P140	15	4.76	261.038	322.157	379.021
CKLNQD1BWP30P140	5.5	2.1	64.8902	69.24762	76.4891
CKLNQD20BWP30P140	18	5.6	316.508	398.60492	468.128
CKLNQD24BWP30P140	20	6.16	368.175	457.71108	542.016
CKLNQD2BWP30P140	6	2.24	74.3782	82.31652	89.418
CKLNQD3BWP30P140	6.5	2.38	84.2502	95.67109	109.96
CKLNQD4BWP30P140	7	2.52	95.644	109.48958	128.89
CKLNQD6BWP30P140	9	3.08	126.833	156.76508	178.899
CKLNQD8BWP30P140	10	3.36	152.72	186.31308	215.794
CKLNQOPTMAD16BWP30P140	18.5	6.16	342.611	380.62767	436.376
CKLNQOPTMAD4BWP30P140	10	3.78	165.473	176.1605	194.235
CKLNQOPTMAD8BWP30P140	12.5	4.48	220.022	242.23692	268.998

图表 11-3 TSMC28 工艺下正沿触发 gated cell 的规格表

11.2.1.3 端口名字

gated_clk_cell.v 文件中例化了具体工艺库的 gated cell，如下图所示：

```
CKLNQD8BWP30P140 x_gated_clk_cell (
                 .CP             (clk_in          ),
                 .TE             (SE              ),
                 .E              (clk_en_bf_latch),
                 .Q              (clk_out         )
                 );
```

图表 11-4 例化 gated cell

信号线名	功能	连接
clk_in	Clock 输入	Clock input
SE	Pad_yy_test_mode || Pad_yy_bist_tst_en || Pad_yy_gated_clk_en_b	Test clock enable
Clk_en_bf_latch	Clock function 使能信号	Function Clock enable
clk_out	Clock 输出	Clock output

图表 11-5 gated cell 信号列表

11.2.1.4 更改 gated_clk_cell.v 文件

选定 gated cell 之后修改 gated_clk_cell.v 文件，如下图所示：

```
//    CKLNQD8BWP30P140 x_gated_clk_cell (
//                  .CP           (clk_in),
//                  .TE              (SE),
//                  .E      (clk_en_bf_latch),
//                  .Q          (clk_out)
//                  );

      PREICG_X0P5B_A9TL40 x_gated_clk_cell (
        .ECK  (clk_out ),
        .E    (clk_en_bf_latch ),
        .SE   (SE ),
        .CK   (clk_in )
      );
```

图表 11-6 修改例化文件

将原 gated cell 替换掉，接上新选的 gated cell。

11.2.1.5 更新 filelist

Gated cell 替换完后需要将新的 gated cell 仿真模型文件准备好，可以放入 LIB 文件里也可以放到用户自己的文件里，重要的是需要修改 filelist，删除之前调用的 model 加入用来替换的文件路径。

11.2.1.6 不使用前端插的 gated cell

后端工具支持自动插 gated cell，如果客户不希望使用中天前端手动插的 gated cell，可以简单注释掉 gated_clk_cell.v 里的 instance gated cell 部分代码然后加上一行代码，如下图所示：

```
//    CKLNQD8BWP30P140 x_gated_clk_cell (
//                  .CP           (clk_in),
//                  .TE             (SE),
//                  .E      (clk_en_bf_latch),
//                  .Q          (clk_out)
//                  );

assign clk_out = clk_in;
```

图表 11-7 不使用前端插入的 gated cell

这样前端插的 gated cell 都没了，客户可以按照自己的意愿插入 gated cell。

11.2.2 Memory 替换

11.2.2.1 文件所在位置

所有需要修改的文件都在 INST_LIB 目录，且文件名都包含关键词 spsram_AAAxBBB, A、B 分别代表 memory 的深度和宽度。

11.2.2.2 生成 memory

所有需要的 memory 信号都可以从 INST_LIB 里看出（AAAxBBB），在生成时我们只对一个选项有要求，就是所有的 memory 都需要支持位写使能信号。其他的选项用户可以根据自己的需求选择，包括 memory 的形状、时序、面积、功耗等。如果有些 memory 过大或者过深，可以自行进行拼接（详见 10.2.2.6）。

11.2.2.3 端口名字

该文件例化了具体工艺的 memory 库 model，如下图所示：

```
sprf065ll_512x64 x_spsram_512x64(
  .A    (A),
  .D    (D),
  .BWEN ({{32{WEN[1]}}, {32{WEN[0]}}}),
  .WEN  (&WEN),
  .CEN  (CEN),
  .CLK  (CLK),
  .Q    (Q)
);
```

图表 11-8 例化 memory

信号线名	功能	连接
A	地址线	RAM 的地址线端口
D	写入数据线	RAM 的数据输入端
WEN*	低电平有效，位写使能信号	RAM 位写使能信号端 缩位与之后做 RAM 写使能信号

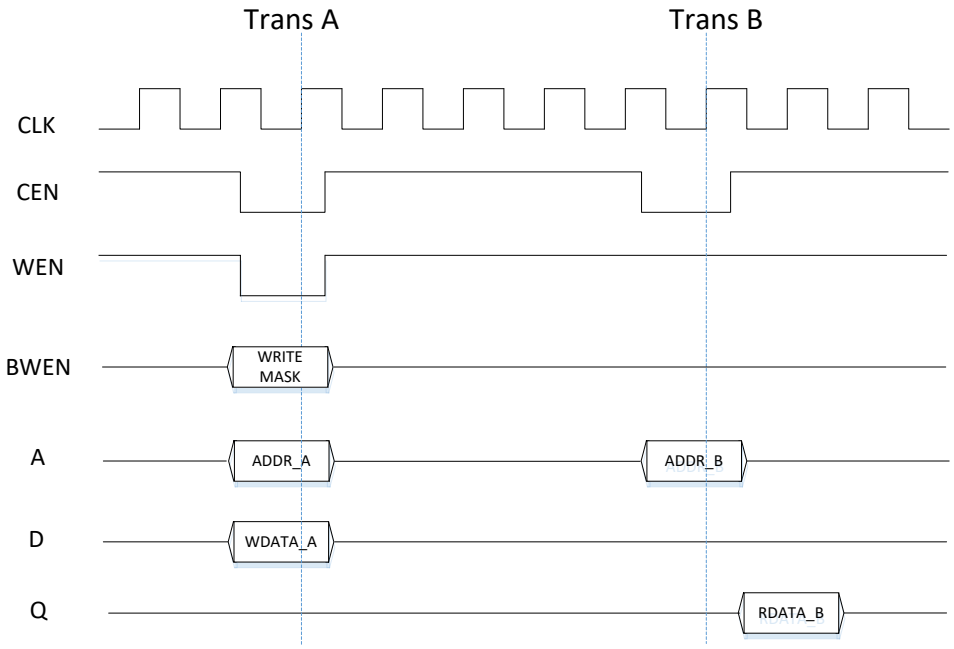
CEN*	低电平有效，位片选信号	RAM 使能信号
CLK	时钟	RAM 时钟端
Q	输出数据线	RAM 的输出

图表 11-9 memory 信号列表

*这两根信号都是低电平有效，具体要看 RAM 的相应信号是否也是低电平有效，另外，不同 vender 提供 ram 可能还有一些其他的控制 port，针对这些 port 用户需要根据自己的需要按照 RAM vender 的用户手册进行处理。

11.2.2.4 RAM 的读写时序

下图是一个 RAM 的读写时序图，Trans A 是一个写请求，Trans B 是一个读请求。写部分数据是 BWEN 控制的，参考图表 11-8 例化 memory 的具体连接方法。Trans B 时钟上升沿采到到输入读请求，下一个 cycle 将数据输出到 Q 端。



图表 11-10 RAM 的读写时序图

11.2.2.5 修改 memory instance 文件

修改 memory，将 RAM 库文件替换掉，下图就将上一节中的 memory 换成了 synopsys 提供的 memory，多出的管脚参考 vender 提供的手册后 tie 上具体的值。再将原来 instance memory 的代码注释或者删除。

```
sprf065lp_synop_512x64 x_spsram_512x64(
  .A    (A),
  .D    (D),
  .BWEB ({{32{WEN[1]}}, {32{WEN[0]}}}),
  .WEB  (&WEN),
  .CEB  (CEN),
  .CLK  (CLK),
  .TURBO(1'b1),
  .RTSEL(1'b0),
  .TSEL (2'b01),
  .Q    (Q)
);
```

图表 11-11 修改 memory 例化文件

11.2.2.6 拼接 memory

如果有些 memory 规格生成不出来，或者客户对该规格 memory 的形状或者其他一些特性不满意，用户可以用更小的 memory 拼接出需要的 memory，比如一块 2048x32 的 memory：

```
spsram040g_2048x32 x_spsram_2048x32(
  .A    (A),
  .D    (D),
  .BWEB ({{8{WEN[3]}},{8{WEN[2]}},{8{WEN[1]}},{8{WEN[0]}}}),
  .WEB  (&WEN),
  .CEB  (CEN),
  .CLK  (CLK),
  .DELAY(2'b00),
  .TEST (2'b00),
  .Q    (Q)
);
```

图表 11-12 需要拼接的 memory

某个工艺下并不支持该规格的 memory，就需要用用其他规格 memory 进行拼接，下面是用两块 1024x32 的 memory 进行拼接的例子，当然也可以 2048x16 的拼接，那是另外一种接法。

```
assign CEN0 = CEN |  A[ADDR_WIDTH-1];
assign CEN1 = CEN | ~A[ADDR_WIDTH-1];

always@(posedge CLK)
begin
  if (!CEN)
  begin
    bank_sel <= A[ADDR_WIDTH-1];
  end
  else
  begin
    bank_sel <= bank_sel;
  end
end
assign  Q[31:0] = bank_sel ? Q1[31:0] : Q0[31:0];
sprf065lp_1024x32 x_spsram_1024x32_bank0(
      .A    (A[ADDR_WIDTH-2:0]),
      .D    (D),
      .BWEB ({ {8{WEN[3]}}, {8{WEN[2]}},{8{WEN[1]}}, {8{WEN[0]}} }
            ),
      .WEB  (&WEN),
      .CEB  (CEN0),
      .CLK  (CLK),
      .TURBO(1'b1),
      .RTSEL(1'b0),
      .TSEL (2'b01),
      .Q    (Q0)
    );

  sprf065lp_1024x32 x_spsram_1024x32_bank1(
      .A    (A[ADDR_WIDTH-2:0]),
      .D    (D),
      .BWEB ({ {8{WEN[3]}}, {8{WEN[2]}},{8{WEN[1]}}, {8{WEN[0]}} }
            ),
      .WEB  (&WEN),
      .CEB  (CEN1),
      .CLK  (CLK),
      .TURBO(1'b1),
      .RTSEL(1'b0),
      .TSEL (2'b01),
      .Q    (Q1)
    );
```

图表 11-13 拼接 memory

11.2.2.7 修改 filelist

Memory 替换完后需要将，新的 memory 仿真模型文件和库文件都准备好，可以放入 LIB 文件里也可以放到用户自己的文件里，重要的是需要修改 filelist，删除以前调用的 mode 加入现在调用的文件路径。

11.2.3 DesignWare IP

11.2.3.1 简介

有些配置下的 S802 里使用了 Desingn ware IP，指代需要用到的库里的加法器乘法器或是乘法累加器，与工艺无关，用户不需要替换。

11.3 FPGA 映射

FPGA 的映射过程如下：

11.3.1 ICG 替换

若用户只用来编译，不进行综合，则 ICG 替换如 10.2.1 节

若用户要用来综合，则需要把 gated_clk_cell.v 中的输入输出相连接，如下图：

```
assign clk_out = clk_in;
```

图表 11-14 FPGA ICG 时钟输入输出连接

并且需要把 gated_clk_cell.v 文件中实例化的具体工艺库中的 cell 代码部分给注释或者删除，最后再把 filelist 中之前调用的 mode 的路径删除。

```
//          HVT_CLKLANQVHSV2 x_gated_clk_cell(
//                      .Q    (clk_out),
//                      .E    (clk_en_bf_latch),
//                      .TE   (SE),
//                      .CK   (clk_in)
//                      );
```

图表 11-15 注释 gated_clk_cell.v 中部分代码

11.3.2 Memory 替换

用户如果拿到 ASIC 版本代码需要做 FPGA 综合，需要把工艺相关库中的 memory 替换成自己的 fpga_memory。

11.3.2.1 文件所在位置

因为 FPGA 测试时已经与具体工艺无关，所以 LIB 中的文件都已经没有任何作用，只需要在 filelist 中注释掉或者删掉 LIB 的路径就可以。需要修改的文件在 INST_LIB 中的.v 文件

11.3.2.2 生成 fpga Memory

Fpga_memory 替换比较简单，用户只需要根据 CKxxx.v 中所用到的 mem 的大小，自行生成出 fpga_memory，端口信号参照 10.2.2 节中 图表 11-9 memory 信号列表。如下图所示：

```
module f_spsram_128x92(
  A,
  CEN,
  CLK,
  D,
  Q,
  BWEN,
  WEN
);
```

图表 11-16 用户根据 mem 大小自行生成的 fpga mem

11.3.2.3 替换 memory

与 ASIC 一样，需要在 instance memory 文件中替换掉之前工艺相关的例化 model，如图所示：

```
//      S5SNLLG1PH_128x92 x_tag_array(
//       .CLK  (CLK),
//       .CEN  (CEN),
//       .BWEN ({{WEN[11:8]},{21{WEN[7]}},{21{WEN[6]}},{21{WEN[5]}},{21{WEN[4]}},WEN[3:0]}),
//       .WEN  (&WEN),
//       .A    (A),
//       .D    (D),
//       .Q    (Q)
//      );
f_spsram_128x92 x_tag_array(
       .CLK  (CLK),
       .CEN  (CEN),
       .BWEN ({{WEN[11:8]},{21{WEN[7]}},{21{WEN[6]}},{21{WEN[5]}},{21{WEN[4]}},WEN[3:0]}),
       .WEN  (&WEN),
       .A    (A),
       .D    (D),
       .Q    (Q)
      );
```

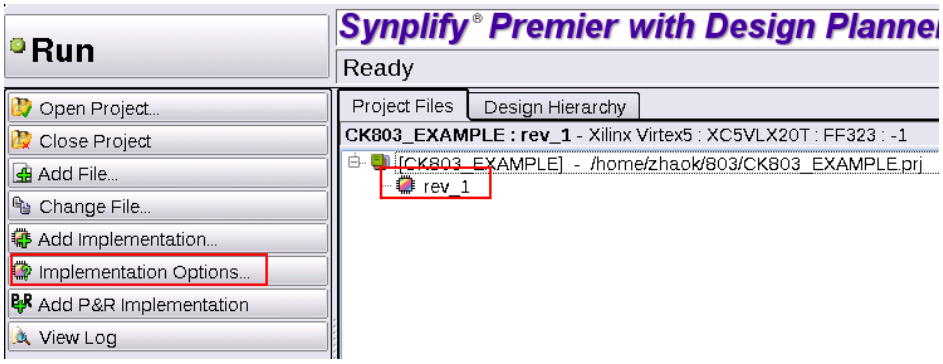
图表 11-17 替换 fpga memory

11.3.2.4 修改 filelist

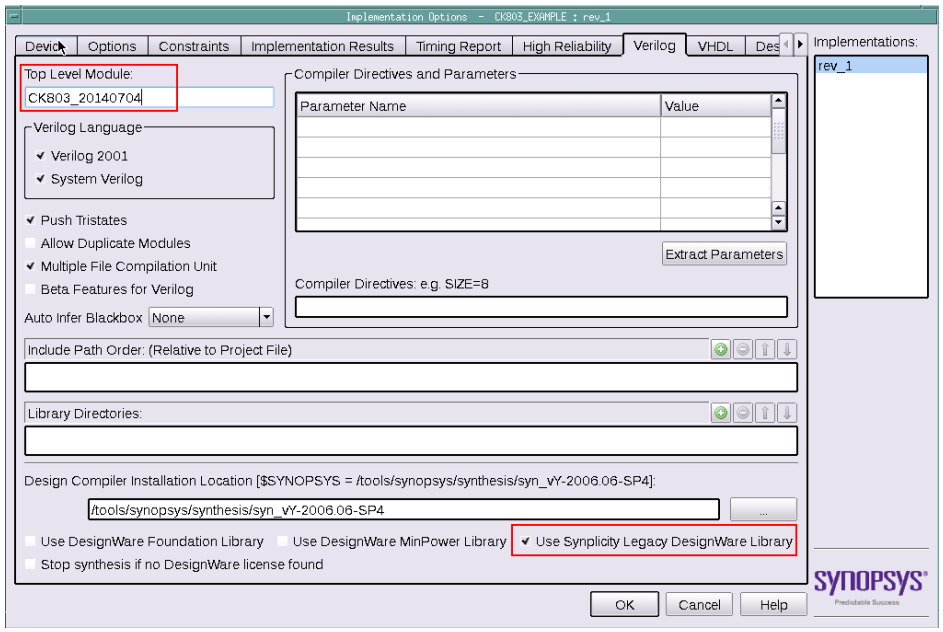
用户需要把自行生成的 fpga_memory 文件放到 LIB 中，并加入到 filelist 中，删掉之前 model 的路径。自此 fpga memory 替换工作就完成了。

11.3.3 DesignWare IP

客户在使用 FPGA 综合时，如果使用 Synplify 软件，只需要在综合选项中选择如下图所示的选项，就能把代码中实例化的 design ware 综合进去。



图表 11-18 综合选项



图表 11-19 选择 Design ware 库