



SMARTL 用户手册

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

版本历史:

| 版本 | 日期 | 描述 | 作者 |
|-----|------------|-----------------------|-------------|
| 1.0 | 7/06/2017 | 1. 第一版 | 杭州中天微系统有限公司 |
| 1.1 | 9/08/2018 | 1. 修改 run case 脚本使用描述 | 杭州中天微系统有限公司 |
| 1.2 | 11/07/2018 | 1.增加 AXI 总线架构描述 | 杭州中天微系统有限公司 |
| 1.3 | 12/29/2018 | 1.增加 E902 产品描述 | 杭州中天微系统有限公司 |

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

目录:

| | |
|-------------------------------|----|
| 1 SmartL 平台描述 | 7 |
| 1.1 概述 | 7 |
| 1.2 SOC 架构 | 7 |
| 1.2.1 AHB 总线架构 | 7 |
| 1.2.2 AXI 总线架构 | 8 |
| 1.3 地址空间 | 9 |
| 1.4 中断号分配 | 9 |
| 2 设计方案 | 11 |
| 2.1 CPU 子系统 | 11 |
| 2.2 AHB BUS | 11 |
| 2.3 AXI BUS | 12 |
| 2.4 APB BUS | 13 |
| 2.4.1 UART | 13 |
| 2.4.2 TIMER | 19 |
| 2.4.3 STIMER | 22 |
| 2.4.4 GPIO | 23 |
| 2.4.5 Clock Gen | 27 |
| 2.4.6 System MPU (SMPU) | 28 |
| 2.4.7 PMU | 30 |
| 2.4.8 WIC | 35 |
| 3 文件结构及说明 | 36 |
| 3.1 tools/ | 36 |
| 3.1.1 CTS_MAP_TOOLS/ | 36 |
| 3.2 lib/ | 36 |
| 3.3 cpu/ | 36 |
| 3.4 soc/ | 36 |
| 3.5 tb/ | 36 |
| 3.6 synthesis/ | 36 |
| 3.7 case/ | 36 |
| 3.8 gen_case/ | 36 |
| 3.9 workdir/ | 37 |
| 3.10 CSKY_Test_Suit/ | 37 |
| 3.11 regress/ | 37 |
| 4 仿真流程 | 38 |
| 4.1 建立环境变量 | 38 |
| 4.2 执行仿真脚本 | 38 |

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

| | |
|-------------------------------|----|
| 4.2.1 准备测试程序所需文件..... | 38 |
| 4.2.2 编译测试程序 | 47 |
| 4.2.3 VCS 仿真..... | 48 |
| 4.2.4 run_case 脚本应用 | 50 |
| 4.3 测试程序调试 | 50 |
| 5 SmartL 测试程序 | 52 |
| 5.1 测试程序分类 | 52 |
| 5.2 测试程序具体描述 | 52 |
| 5.2.1 异常向量表配置演示..... | 52 |
| 5.2.2 MGU 配置演示 | 52 |
| 5.2.3 控制寄存器读写 | 52 |
| 5.2.4 Smoke 测试程序 | 52 |
| 5.2.5 Cache 使用 | 53 |
| 5.2.6 低功耗握手演示 | 53 |
| 5.2.7 动态变频演示 | 54 |
| 5.2.8 安全演示 | 54 |
| 5.2.9 世界切换 | 54 |
| 5.2.10 脉冲中断演示 | 55 |
| 5.2.11 调试 (HAD) | 55 |
| 5.2.12 性能测试测试程序 | 55 |
| 5.2.13 C 语言程序示例..... | 56 |
| 5.3 CTS 介绍..... | 56 |
| 5.4 Regress | 56 |
| 6 FPGA 流程..... | 58 |
| 6.1 采用 Xilinx ISE 器件 | 58 |
| 6.2 采用 Xilinx Vivado 器件 | 65 |
| 6.3 采用 Altera 器件 | 73 |
| 6.4 FPGA 占用资源估计 | 73 |
| 7 附件一: SDC | 75 |
| 7.1 Clock 设置 | 75 |
| 7.2 False Path 设置..... | 75 |
| 7.3 加紧 ICG path 约束 | 75 |
| 7.4 Input/output 设置 | 76 |
| 7.4.1 Delay..... | 76 |
| 7.4.2 其他 | 76 |

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

图表目录:

| | |
|---|----|
| 图表 1-1 SmartL AHB 总线平台架构..... | 7 |
| 图表 1-2 SmartL AXI 总线平台架构..... | 8 |
| 图表 1-3 系统地址分配..... | 9 |
| 图表 1-4 IP 地址映射 | 9 |
| 图表 1-5 中断向量号 | 10 |
| 图表 2-1 CPU 子系统结构图 | 11 |
| 图表 2-2 UART 结构图 | 13 |
| 图表 2-3 UART 数据流格式 | 13 |
| 图表 2-4 UART 采样时序 | 14 |
| 图表 2-5 UART 中断时序 | 14 |
| 图表 2-6 UART 寄存器 | 15 |
| 图表 2-7 UART 中断使能寄存器 | 16 |
| 图表 2-8 UART 中断标号寄存器 | 16 |
| 图表 2-9 UART 中断优先级 | 17 |
| 图表 2-10 UART 传输控制寄存器 | 18 |
| 图表 2-11 UART 传输状态寄存器 | 19 |
| 图表 2-12 UART 状态寄存器 | 19 |
| 图表 2-13 TIMER 结构图 | 19 |
| 图表 2-14 TIMER 计时器寄存器地址..... | 20 |
| 图表 2-15 Timer1 寄存器描述 | 20 |
| 图表 2-16 Timer1 回填值寄存器 | 20 |
| 图表 2-17 Timer1 当前值寄存器 | 20 |
| 图表 2-18 Timer1 控制寄存器 | 21 |
| 图表 2-19 Timer1 中断清除寄存器 | 21 |
| 图表 2-20 Timer1 中断状态寄存器 | 21 |
| 图表 2-21 TIMER 中断状态寄存器..... | 21 |
| 图表 2-22 TIMER 中断状态寄存器..... | 22 |
| 图表 2-23 TIMER 原始中断状态寄存器..... | 22 |
| 图表 2-24 GPIO 结构图 | 23 |
| 图表 2-25 GPIO 寄存器地址..... | 23 |
| 图表 2-26 Port A I/O 寄存器和 Interrupts 寄存器描述..... | 24 |
| 图表 2-27 Port A 数据寄存器 | 24 |
| 图表 2-28 Port A 数据方向寄存器 | 24 |
| 图表 2-29 Port A 源数据寄存器 | 24 |
| 图表 2-30 Port A 外部数据寄存器 | 25 |
| 图表 2-31 中断使能寄存器 | 25 |
| 图表 2-32 中断屏蔽寄存器 | 25 |
| 图表 2-33 中断类型寄存器 | 25 |
| 图表 2-34 中断极性寄存器 | 26 |
| 图表 2-35 中断状态寄存器 | 26 |

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

| | |
|---|----|
| 图表 2-36 未屏蔽中断状态寄存器 | 26 |
| 图表 2-37 中断清除寄存器 | 26 |
| 图表 2-38 中断同步寄存器 | 27 |
| 图表 2-39 clock gen 结构图 | 27 |
| 图表 2-40 生成时钟列表 | 28 |
| 图表 2-41 clock gen 寄存器地址 | 28 |
| 图表 2-42 clock gen 寄存器描述 | 28 |
| 图表 2-43 SMPU 寄存器地址 | 29 |
| 图表 2-44 SMPU 寄存器描述 | 29 |
| 图表 2-45 保护区大小及其基地址要求 | 29 |
| 图表 2-46 PMU 结构图 | 30 |
| 图表 2-47 PMU 寄存器地址 | 30 |
| 图表 2-48 PMU 唤醒源使能寄存器 | 31 |
| 图表 2-49 CK802 接口 | 31 |
| 图表 2-50 CK801/803S/CK804/CK805 信号 | 32 |
| 图表 2-51 PMU 状态机 | 33 |
| 图表 2-52 时钟分布 | 33 |
| 图表 2-53 power gating 状态机 | 34 |
| 图表 2-54 WIC 结构图 | 35 |
| 图表 4-1 建立环境变量 | 38 |
| 图表 4-2 确定编译工具 | 38 |
| 图表 4-3 编译工具安装路径没有被添加到环境变量时的报错 | 39 |
| 图表 4-4 源文件以及扩展路径 | 39 |
| 图表 4-5 设置 C 编译参数以及链接参数 | 39 |
| 图表 4-6 设置链接库文件 | 40 |
| 图表 4-7 编译流程 | 41 |
| 图表 4-8 链接描述文件 | 42 |
| 图表 4-9 初始化异常向量表 | 44 |
| 图表 4-10 初始化通用寄存器 | 45 |
| 图表 4-11 设置堆栈指针 | 46 |
| 图表 4-12 设置 VBR 以及 PSR | 46 |
| 图表 4-13 进入测试程序 | 46 |
| 图表 4-14 设置相关标志段 | 47 |
| 图表 4-15 CPU 配置及编译选项 | 47 |
| 图表 5-1 regress list | 56 |
| 图表 5-2 regress 结果 | 57 |
| 图表 6-1 新建 Project_1 | 58 |
| 图表 6-2 新建 Project_2 | 59 |
| 图表 6-3 添加设计文件_1 | 60 |
| 图表 6-4 添加设计文件_2 | 60 |
| 图表 6-5 UCF 文件示例 | 61 |
| 图表 6-6 使用网表文件综合 | 62 |

| | |
|---------------------------------|----|
| 图表 6-7 综合与布局布线..... | 63 |
| 图表 6-8 综合结果检查..... | 64 |
| 图表 6-9 ISE 使用帮助菜单..... | 64 |
| 图表 6-10 vivado 界面..... | 65 |
| 图表 6-11 vivado 新建项目..... | 66 |
| 图表 6-12 项目名称与路径..... | 66 |
| 图表 6-13 项目类型..... | 67 |
| 图表 6-14 选择 FPGA 芯片型号..... | 68 |
| 图表 6-15 确认 FPGA 型号..... | 68 |
| 图表 6-16 添加项目源文件..... | 69 |
| 图表 6-17 添加源文件界面..... | 70 |
| 图表 6-18 项目源文件层次..... | 70 |
| 图表 6-19 添加约束文件..... | 71 |
| 图表 6-20 综合项目代码..... | 71 |
| 图表 6-21 查看综合结果..... | 72 |
| 图表 6-22 布局布线..... | 72 |
| 图表 6-23 Implemented Report..... | 72 |
| 图表 6-24 FPGA 资源占用情况..... | 73 |

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

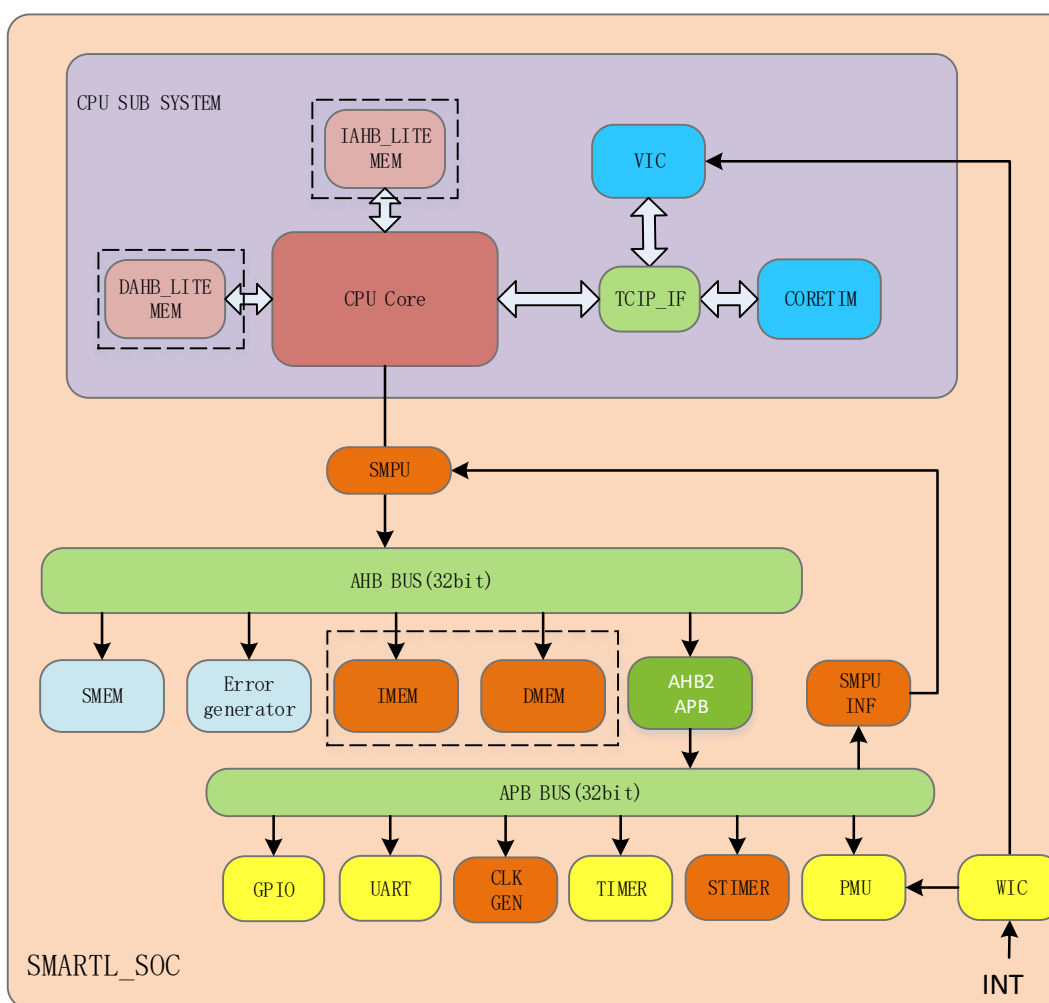
1 SmartL 平台描述

1.1 概述

SmartL 平台是用于 CK801/CK802/CK803S/CK804/CK805/E902 集成、调试仿真，以及 FPGA 应用评估的综合演示平台。用户可以在 SmartL 平台上进行 RTL 仿真，制作 FPGA 仿真以及调试，具有简单易懂，上手快等特点。该平台支持 AHB-Lite，AHB，APB，AXI 总线协议，支持 JTAG 调试。在演示平台中包含 C 语言程序实例，中断演示程序实例，低功耗演示程序实例，以及用于性能测试的（dhrystone/coremark）程序实例等。

1.2 SOC 架构

1.2.1 AHB 总线架构



图表 1-1 SmartL AHB 总线平台架构

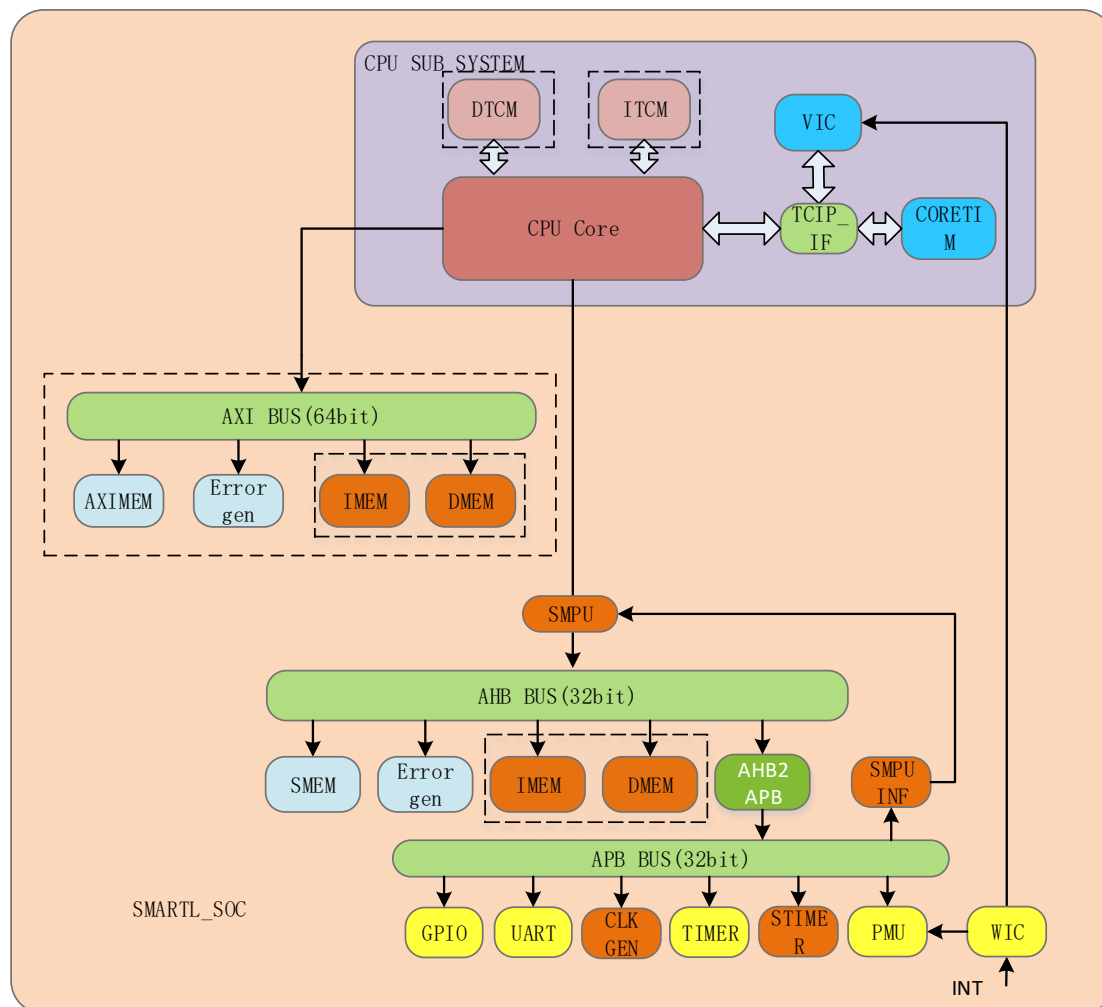
虚线框内的模块由 CPU 配置决定是否存在，当 CPU 没有配置 DLITE 时，DMEM 会被放到 AHB 总线上，ILITE 也一样，对应 IMEM。

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

SMPU、SMPUINF、STIMER 几个模块是安全演示相关的，如果客户申请的 CPU 不包含安全配置选项 TEE，可以无视这些模块，这些模块也不会起作用。

1.2.2 AXI 总线架构



图表 1-2 SmartL AXI 总线平台架构

AXI 总线架构是 CK805 集成所用，虚线框内的模块由 CPU 配置决定是否存在。当 CPU 没有配置 DTCM 时，DTCM 会根据是否配置 AXI 总线被放到 AHB 或 AXI 总线，如果配置存在 AXI 总线时，DTCM 被放置在 AXI 总线上，否则在 AHB 总线上，对应 DMEM；ILITE 也一样，对应 IMEM。

SMPU、SMPUINF、STIMER 几个模块是安全演示相关的，如果客户申请的 CPU 不包含安全配置选项 TEE，可以无视这些模块，这些模块也不会起作用。

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

1.3 地址空间

地址空间分配如下图所示：

| Address | Mapping IP |
|-----------------------|------------------------|
| 0x00000000-0x1FFFFFFF | 本地指令存储器可用地址空间 |
| 0x20000000-0x3FFFFFFF | 本地数据存储器可用地址空间 |
| 0x40000000-0x4FFFFFFF | APB 总线地址空间 |
| 0x50000000-0x507FFFFF | 片外存储可用地址空间 |
| 0x60000000-0x7FFFFFFF | 系统内存可用地址空间 |
| 0x80000000-0x8FFFFFFF | 系统内存可用地址空间(AXI 总线架构存在) |
| 0xE0000000-0xEFFFFFFF | 紧耦合 IP 空间 (TCIP) |
| 其它 | 保留，读写返回错误 |

图表 1-3 系统地址分配

各 IP 的地址映射如下所示：

| Base Address | IP |
|--------------|--------------------|
| 0x40011000 | TIMER (APB Slave) |
| 0x40015000 | UART (APB Slave) |
| 0x40016000 | PMU (APB Slave) |
| 0x40017000 | CLK_GEN(APB Slave) |
| 0x40018000 | STIMER(APB Slave) |
| 0x40019000 | GPIO(APB Slave) |
| 0x4001A000 | SMPU(APB Slave) |
| 0xE000E010 | CORETIM (TCIP) |
| 0xE000E100 | VIC (TCIP) |

图表 1-4 IP 地址映射

1.4 中断号分配

各 IP 的中断向量号如下所示：

| 中断号 | 属性 | 中断源 |
|------|-----|------------------|
| 0x20 | 非安全 | UART (APB Slave) |

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

| | | |
|-----------|-----|--------------------|
| 0x21 | 非安全 | CORETIM (TCIP) |
| 0x22-0x25 | 非安全 | TIMER (APB Slave) |
| 0x26 | 保留 | 保留 |
| 0x27-0x2e | 非安全 | GPIO(APB Slave) |
| 0x30-0x33 | 安全 | STIMER (APB Slave) |
| 0x34 | 非安全 | Pad 直接输入，演示脉冲中断 |

图表 1-5 中断向量号

如果客户申请的 CPU 不包含安全配置选项 TEE，请无视安全中断。

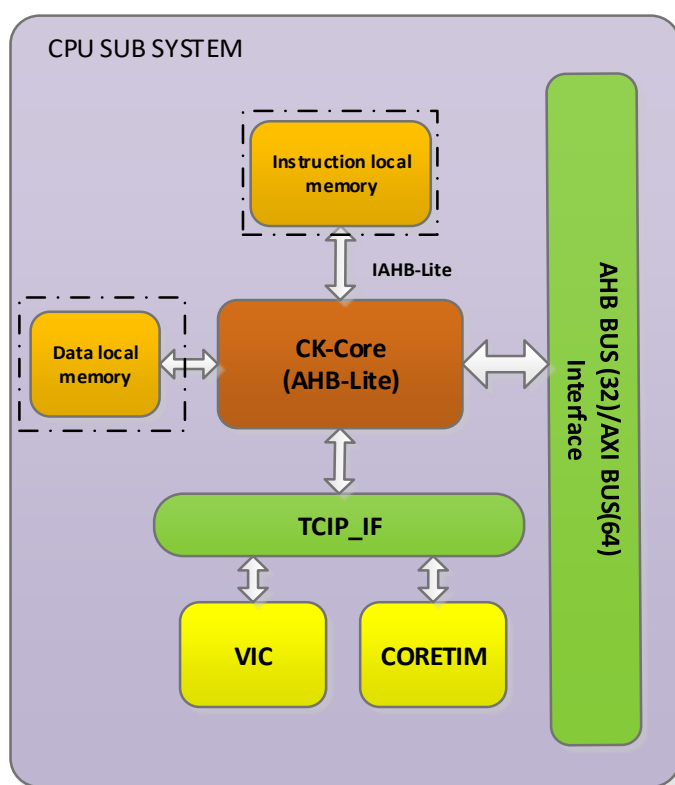
C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

2 设计方案

2.1 CPU 子系统

在 SmartL 平台中，所有配置下的 CK-CPU 都会构成一个独立的 AHB 或 AXI 子系统。在这个子系统中，CPU 的绝大部分外围信号将在子系统这个层次上被绑定成固定的值，AHB 和 AXI 接口信号会释放到系统上面和系统中的总线接口连接，调试引脚（JTAG）会连接到系统的 JTAG 引脚上面。这里特别要注意，对于 CK802/CK803S/CK804/CK805/E902，本地指令、数据存储存储器则为可配置选项。子系统集成了紧耦合 IP，包括计数器 CORETIM 和中断控制器 VIC。这些经过封装后的 CK-CPU 从系统的角度看都是统一的，如下图所示：



图表 2-1 CPU 子系统结构图

在 CPU 配置了本地数据和指令存储器之后，SmartL RTL 仿真平台默认使用本地存储器存储程序指令或数据，否则使用 AHB 或 AXI 总线对应地址上的存储器。

2.2 AHB BUS

在 SmartL 平台中，设计有一个简易的 AHB 总线仲裁模块，该模块支持 32 位总线宽度。该模块仅用于 SMARTL 平台中做简单的功能演示。AHB 按照地址空间对从 CPU 发送过来的请求进行仲裁，将其发送到对应的从设备（Slave）。

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

在 SmartL 系统中，AHB 总线上挂了一个主设备（Master），为 CKCPU；默认情况下挂了 3 个从设备，分别是系统静态随机访问存储器（SRAM）、APB 桥和错误产生器（用于返回总线上的错误情况）。当 CPU 没有定义 IAHB/DAHB_LITE 时会在 AHB 总线上相对应的指令/数据空间上例化额外的系统存储器（SRAM）以实现指令、数据的正常访问，如没有定义本地指令以及数据存储器则总线上需要例化两块新的系统存储器，SLAVE 数增加到 5 个。

系统内存是直接物理实现的 SRAM，用于存放数据，栈相关内容以及相关的全局变量等内容。该设备的地址空间为 0x60000000-0x7FFFFFFF。该存储器允许读写访问，考虑到 FPGA 成本约束和使用需求，物理上实际实现低 128K。当没有配置 IAHB_LITE 时在地址空间 0x00000000 - 0x1FFFFFFF 上增加新的存储器，物理上实际实现低 128K；当没有配置 DAHB_LITE 时在地址空间 0x20000000 -0x3FFFFFFF 上增加新的存储器，物理上实际实现低 128K。

APB 桥下实现了 APB 低速子系统，目前设计有七个 APB 设备，分别是 UART、Timer、Stimer、CLK Generator、System MPU、Power Management Unit (PMU)、GPIO。

错误产生器是为了构建总线错误访问行为设计的模块，当 CPU 访问到该外设时，它将产生一个错误的请求。该模块的地址空间为所有保留区域。

2.3 AXI BUS

在 SMARTH 平台中，设计有一个简易的 AXI 总线仲裁模块，该模块支持 64 位总线宽度。该模块仅用于 SMARTL 平台中做简单的功能演示。AXI 按照地址空间对从 CPU 发送过来的请求进行仲裁，将其发送到对应的从设备（Slave）去。在 SMARTL 系统中，AXI 总线上挂了一个主设备（Master），为 CK-CPU；挂了 4 个从设备，分别是系统指令存储器（IMEM）、系统数据存储器(DMEM)、系统随机动态存储器(SRAM)和错误产生器（用于返回总线上的错误情况）。

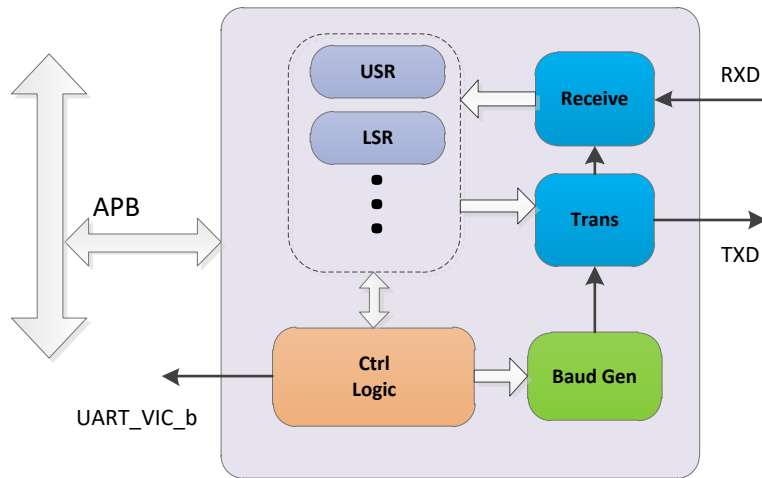
系统内存是直接物理实现的 SRAM，用于存放指令，数据，栈相关内容以及相关的全局变量等内容。其中指令和数据存储器在 CPU 配置存在 AXI 总线且没有相应的 TCM 模块时存在。其中系统指令地址空间 0x00000000 - 0x1FFFFFFF，物理上实际实现低 128K；系统数据地址空间 0x20000000 -0x3FFFFFFF，物理上实际实现低 128K，系统随机动态存储器地址空间 0x80000000 -0x8FFFFFFF，物理上实际实现低 128K。

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

2.4 APB BUS

2.4.1 UART



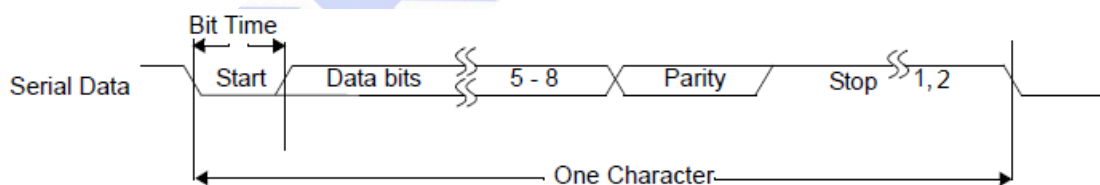
图表 2-2 UART 结构图

UART 是一种通用串行数据总线，用于异步通信。该总线支持双向通信，可以实现全双工传输和接收。

UART 的功能：

- APB 总线接口；
- 全双工独立的发送和接收通道；
- 软件检测通道状态；
- 可配置奇偶校验中断、覆盖有效数据中断、帧错误中断产生；
- 可配置传输波特率；
- 最大传输速率为系统时钟的 1/16

UART 通过串行传输总线 TXD 传送数据，RXD 接收数据，数据流格式如下图所示：



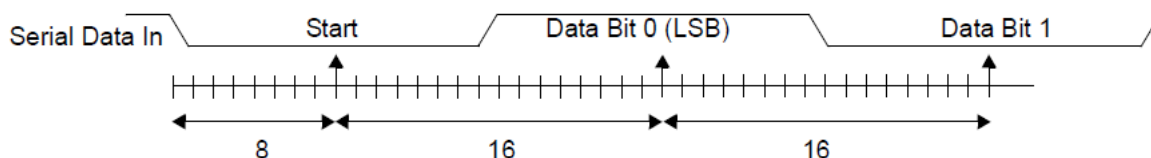
图表 2-3 UART 数据流格式

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

每帧数据有起始位、5 到 8 位的数据位和可选这得奇偶校验位以及 1~2 位的停止位，其中起始位为低电平，停止位为高电平。

数据传输通过波特率同步，软件首先确定波特率，UART 自动产生波特率时钟，在接收总线监测到低电平起始位后，每 16 个时钟周期采样一次总线，从而接收数据。采样如下图所示：



图表 2-4 UART 采样时序

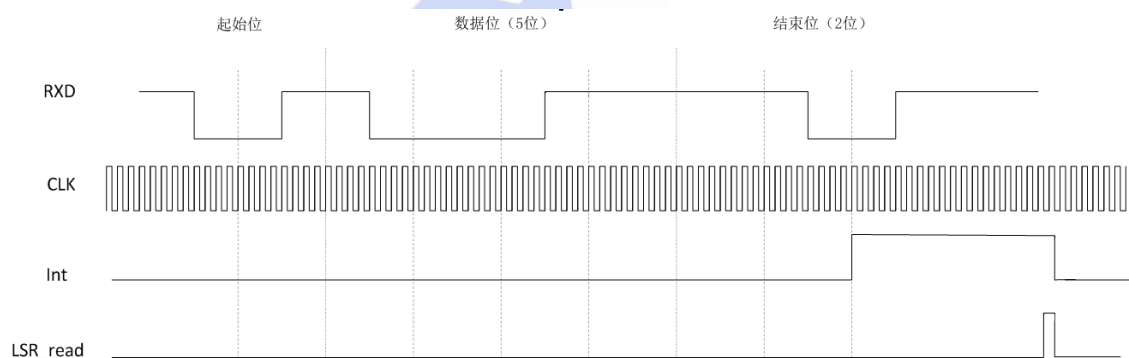
UART 波特率产生

UART 通过软件设置 DLL 和 DLH（分频时钟寄存器）来产生相应的波特率时钟分频时钟寄存器的计算公式为：系统时钟/(波特率 x16)。DLL 寄存器存储分频时钟寄存器的地位值，DLH 存储分频时钟的高位值，设置 DLL、DLH 值之前需要将 LCR 传输控制器的 DLAB 位置 1，寄存器的具体描述见下文。

UART 中断时序

UART 支持覆盖有效数据、奇偶校验错误、帧错误的传输状态中断，也支持接受数据有效中断和传输保持寄存器空中断，以及支持 UART 忙中断。软件编程者可以通过写值到 IER 寄存器中使能这些中断。具体的中断描述和产生见下文的寄存器描述表格中。

下图为一次帧错误中断的产生，如图所示：数据位的长度为 5 位，停止位的长度为 2 位，在第二个停止位时检测到了低电平，所以产生帧错误中断，此中断在软件读 LSR 寄存器后清除。



图表 2-5 UART 中断时序

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

需要注意的是：所有接收的中断，即接收状态中断和接收数据有效中断都在完整接收完一帧数据后产生，而传输保持寄存器空中断在传输保持寄存器空时就立即产生。

UART 的寄存器描述

| 地址 | 名称 | 读写 | 复位值 | 描述 |
|------|-----|-----|------|----------------------------|
| 0x00 | RBR | R | 0x0 | 接收缓冲寄存器 在LCR[7] = 0时有效 |
| | THR | W | 0x0 | 传输保持寄存器 在LCR[7] = 0时有效 |
| | DLL | R/W | 0x0 | 分频时钟寄存器低 在LCR[7] = 1时有效 |
| 0x04 | DLH | R/W | 0x0 | 分频时钟寄存器高 在LCR[7] = 1时有效 |
| | IER | R/W | 0x0 | 中断使能寄存器 在LCR[7] = 0时有效 |
| 0x08 | IIR | R | 0x1 | 中断标号寄存器 |
| 0x0c | LCR | R/W | 0x0 | 传输控制寄存器 |
| 0x14 | LSR | R | 0x60 | 传输状态寄存器 |
| 0x7c | USR | R | 0x0 | UART状态寄存器 |

图表 2-6 UART 寄存器

RBR ----- 接收缓冲寄存器

RBR 是接收缓冲寄存器，从 RXD 端口接收数据。寄存器低 8 位有效。当传输状态寄存器的 DR 位有效时，数据有效。数据必须在下一个数据接收到之前被读走，否则会出现覆盖有效数据中断。

THR ----- 传输保持寄存器

THR 是传输保持寄存器，数据从 TXD 端口传输到接收端，数据的低 8 位有效。数据只能在传输状态寄存器的 THRE 为高时写到 THR 中，否则，有效数据会被覆盖

DLL ----- 分频时钟寄存器低

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

DLL 是分频时钟寄存器的低 8 位。分频时钟寄存器控制传输波特率，寄存器只能在传输控制寄存器的 DLAB 位为高并且 UART 状态寄存器的 busy 位为 0 时写。当 DLH 和 DLL 都设为 0 时，传输不会使能。

DLH ----- 分频时钟寄存器高

DLH 是分频时钟寄存器的低 8 位。分频时钟寄存器控制传输波特率，寄存器只能在传输控制寄存器的 DLAB 位为高并且 UART 状态寄存器的 busy 位为 0 时写。当 DLH 和 DLL 都设为 0 时，传输不会使能。

IER ----- 中断使能寄存器

IER 是中断使能寄存器。使能传输和接收所产生的中断。它的定义如下：

| 位 | 名称 | 读写 | 描述 |
|------|-------|-----|--------------|
| 31:8 | - | - | 保留 |
| 7:3 | - | - | 保留 |
| 2 | ELSI | R/W | 使能接收状态中断 |
| 1 | ETBEI | R/W | 使能传输保持寄存器空中断 |
| 0 | ERBFI | R/W | 使能接收数据有效中断 |

图表 2-7 UART 中断使能寄存器

IIR ----- 中断标号寄存器

中断标号寄存器指示出当前最高优先级中断，寄存器定义和中断优先级如下面的表格所示：

| 位 | 名称 | 读写 | 描述 |
|------|-----|----|---|
| 31:8 | | | 保留 |
| 7:4 | | | 保留 |
| 3:0 | IID | R | 中断标号 0001 = 无中断 0010 = THR空中断 0100 = 接收数据有效中断 0110 = 接收状态中断 0111 = uart忙 |

图表 2-8 UART 中断标号寄存器

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

| 中断优先级分为4级，详细信息如下表所示：中断标号 | 优先级 | 中断类型 | 中断源 | 中断复位控制 |
|--------------------------|-----|----------|--------------------------|-----------------------------|
| 0001 | | 无中断 | | |
| 0110 | 最高 | 接收状态中断 | 覆盖有效数据 奇偶校验错误 帧错误 | 读LSR寄存器 |
| 0100 | 第二 | 接收数据有效中断 | 接受数据有效 | 读 RSR 寄存器 |
| 0010 | 第三 | THR空中断 | THR空 | 读 IIR 寄存器 或写数据到 THR 中 |
| 0111 | 第四 | uart忙 | 当uart忙时， 写数据到LCR 中 | 读 USR 寄存器 |

图表 2-9 UART 中断优先级

LCR ----- 传输控制寄存器

传输控制寄存器控制 UART 的传输和接收，主要控制传输的数据位长度和是否奇偶校验等，详细定义如下：

| 位 | 名称 | 读写 | 描述 |
|------|------|-----|--|
| 31:8 | | | 保留 |
| 7 | DLAB | R/W | 分频时钟寄存器访问控制位。分频时钟寄存器一直可读但只有在uart不忙时写。此位应当在初始化了分频时钟寄存器后复位，这样可以访问其他寄存器 |
| 6: 5 | | | 保留 |
| 4 | EPS | R/W | 奇偶校验选择。当uart不忙时写。 1----- 偶校验 0----- 奇校验 |
| 3 | PEN | R/W | 奇偶检验使能位 |
| 2 | STOP | R/W | 停止位个数 0 ---- 1个停止位 1 ---- 2个停止位 |

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

| | | | |
|-----|-----|-----|--|
| 1:0 | DLS | R/W | 数据长度选择 00 = 5位 01 = 6位 10 = 7位 11 = 8位 |
|-----|-----|-----|--|

图表 2-10 UART 传输控制寄存器

LSR ----- 传输状态寄存器

传输状态寄存器，标识传送和接收的状态。详细的定义如下：

| 位 | 名称 | 读写 | 描述 |
|------|------|----|--|
| 31:8 | | | 保留 |
| 7 | | | 保留 |
| 6 | TEMT | R | 传送空标识。当传输保持寄存器空并且传输移位寄存器也为空时置高 |
| 5 | THRE | R | 传送保持寄存器空标识。当传送保持寄存器为空时置高 |
| 4 | | | 保留 |
| 3 | FE | R | 帧错误位 0 ---- 没有帧错误 1 ---- 帧错误 读 LSR 寄存器将复位 FE 位 |
| 2 | PE | R | 奇偶校验错误位。当LCR的PEN位置高，此位将在发生奇偶校验错误时置高 0 ---- 没有奇偶校验错误 1 ---- 奇偶校验错误 读LSR寄存器将复位PE位 |
| 1 | OE | R | 覆盖有效数据错误 0 ---- 无错误 1 ---- 覆盖有效数据错误 读 LSR 寄存器将复位 OE 位 |
| 0 | DR | R | 接收数据有效位 0 ---- 没有接收到有效数据 |

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

| | | | |
|--|--|--|-----------------------------|
| | | | 1 ---- 接收数据有效 读 RBR 此位复位 |
|--|--|--|-----------------------------|

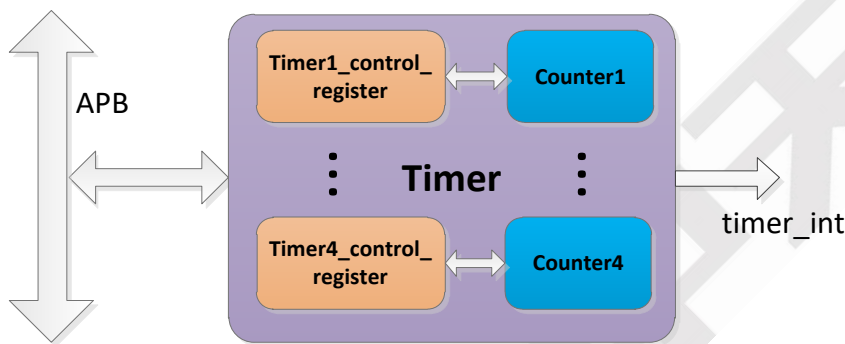
图表 2-11 UART 传输状态寄存器

UART 状态寄存器标识 UART 是否在接收或是发送数据。

| 位 | 名称 | 读写 | 描述 |
|------|------|----|--|
| 31:8 | - | - | 保留 |
| 7:1 | - | - | 保留 |
| 0 | BUSY | R | UART 忙 0 ---- UART 在空状态 1 ---- UART 忙（接收或传送数据） |

图表 2-12 UART 状态寄存器

2.4.2 TIMER



图表 2-13 TIMER 结构图

TIMER 是一个用户可编程的计时设备，具有 APB 总线接口。它内部包含 4 个独立可编程的 32 位计数器，计数器可以从一个软件写入的值开始向下计数，计数到 0 时产生独立的中断。所以 TIMER 具有 4 个中断源。每次计数到 0，计数器都会从相应的回填值寄存器读取计时初始值。

TIMER 的功能特点如下：

- 4 个独立可编程计数器
- 32 位计时位宽
- 计时时间可编程
- 支持两种运行模式：free-running 模式和 user-defined count 模式

TIMER 计时器寄存器地址：（TIMER 基址为 0x40011000）

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

| Address Range (offset) | Function |
|------------------------|------------------------|
| 0x00 to 0x10 | Timer1 Registers |
| 0x14 to 0x24 | Timer2 Registers |
| 0x28 to 0x38 | Timer3 Registers |
| 0x3c to 0x4c | Timer4 Registers |
| 0xa0 to 0xa8 | Timer System Registers |

图表 2-14 TIMER 计时器寄存器地址

下面以 TIMER 中的 Timer1 为例进行寄存器描述，Timer2~4 与其相同。

| 名称 | 地址 偏移量 | 位宽 | R/W | 复位值 | 描述 |
|------------------------|-----------|----|-----|-------|-----------------|
| Timer1LoadCount | 0x00 | 32 | R/W | 32'b0 | Timer1 回填值寄存器 |
| Timer1CurrentValue | 0x04 | 32 | R | 32'b0 | Timer1 当前值寄存器 |
| Timer1Control Reg | 0x08 | 32 | R/W | 3'b0 | Timer1 控制寄存器 |
| Timer1EOI | 0x0C | 32 | R | 1'b0 | Timer1 中断清除寄存器 |
| Timer1IntStatus | 0x10 | 32 | R | 1'b0 | Timer1 中断状态寄存器 |
| TimersInt Status | 0xa0 | 32 | R | 32b'0 | TIMER 中断状态寄存器 |
| TimersEOI | 0xa4 | 32 | R | 32'b0 | TIMER 中断清除寄存器 |
| TimersRaw IntStatus | 0xa8 | 32 | R | 32'b0 | TIMER 原始中断状态寄存器 |

图表 2-15 Timer1 寄存器描述

Timer1LoadCount

| 位 | 名称 | R/W | 描述 |
|------|-------------------|-----|--|
| 31:0 | Timer1 回填值 寄存器 | R/W | 存储 Timer1 的回填值，在每一个计数循环开始时该寄存器的值会赋给 Timer1 的计数器。 |

图表 2-16 Timer1 回填值寄存器

Timer1CurrentValue

| 位 | 名称 | R/W | 描述 |
|------|----------------------------------|-----|-------------------|
| 31:0 | Timer1 Current Value Register | R | 存储 Timer1 的当前计数值。 |

图表 2-17 Timer1 当前值寄存器

Timer1ControlReg

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

| 位 | 名称 | R/W | 描述 |
|------|---------|-----|--|
| 31:3 | 保留，读为 0 | | |
| 2 | 计时中断屏蔽 | R/W | 0: 计时中断没有被屏蔽 1: 计时中断被屏蔽 |
| 1 | 计时模式选择 | R/W | 0 : free-running 模式: 回填值为 32'bFFFFFFF 1 : user-defined running 模式: 回填值从回填值寄存器读取 |
| 0 | 计时使能 | R/W | 0 : 不使能 1 : 使能 |

图表 2-18 Timer1 控制寄存器

Timer1EOI

| 位 | 名称 | R/W | 描述 |
|------|----------------|-----|------------------|
| 31:1 | 保留，读为 0 | | |
| 0 | Timer1 中断清除寄存器 | R | 读操作将清除 Timer1 中断 |

图表 2-19 Timer1 中断清除寄存器

Timer1IntStatus

| 位 | 名称 | R/W | 描述 |
|------|----------------|-----|----------------|
| 31:1 | 保留，读为 0 | | |
| 0 | Timer1 中断状态寄存器 | R | 指示 Timer1 中断状态 |

图表 2-20 Timer1 中断状态寄存器

TimersIntStatus

| 位 | 名称 | R/W | 描述 |
|------|---------------|-----|--|
| 31:4 | 保留，读为 0 | | |
| 3:0 | TIMER 中断状态寄存器 | R | 相应比特指示各计时器的中断状态，读操作不会清除有效中断。 0: 中断无效 1: 中断有效 |

图表 2-21 TIMER 中断状态寄存器

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

TimersEOI

| 位 | 名称 | R/W | 描述 |
|------|---------------|-----|--------------------|
| 31:4 | 保留，读为 0 | | |
| 3:0 | TIMER 中断清除寄存器 | R | 读操作返回 0 并清除所有有效中断。 |

图表 2-22 TIMER 中断状态寄存器

TimersRawIntStatus

| 位 | 名称 | R/W | 描述 |
|------|-----------------|-----|--|
| 31:4 | 保留，读为 0 | | |
| 3:0 | TIMER 原始中断状态寄存器 | R | 该比特指示相应计时器中断的原始状态（不受屏蔽位影响，计时器计数到 0 时无条件置位） 0：原始中断没有发生 1：原始中断发生 |

图表 2-23 TIMER 原始中断状态寄存器

2.4.3 STIMER

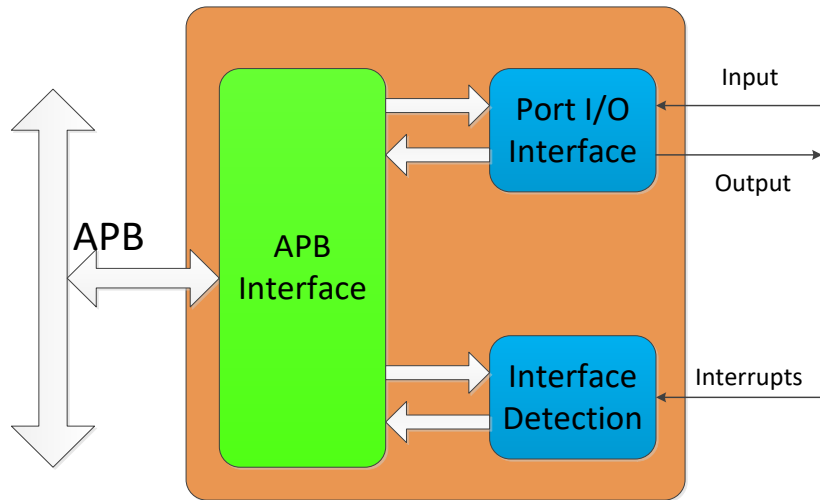
STIMER 的基地址 0x400018000，其内部结构、功能特点与控制方法和 TIMER 完全相同。

所生成的中断源需在中断控制器的对应位置根据需要配置为安全中断，以实现安全中断的产生。如需要实现安全中断的功能，应将 STIMER 所在地址写入 SPMU 的 entry 中，实现对该安全模块配置信息进行保护。同时把该中断在 VIC 内设置成安全中断，这个 IP 就变成了安全 IP。

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

2.4.4 GPIO



图表 2-24 GPIO 结构图

General Purpose Input Output(通用输入/输出), 简称 GPIO, 也称为总线扩展器。

GPIO 的功能:

- APB 总线接口
- 8 个独立可配置引脚的 I/O 端口;
- 每个 I/O 端口有独立的数据寄存器 and 数据方向寄存器;
- A 端口有其可配置的中断模式;

GPIO 寄存器地址: (GPIO 基址为 0x40019000)

| Address Range (offset) | Function |
|------------------------|----------------------------|
| 0x00 to 0x08,0x50 | PortA I/O Registers |
| 0x30 to 0x44,0x4c,0x60 | PortA Interrupts Registers |

图表 2-25 GPIO 寄存器地址

下面以 PortA 的 I/O 寄存器和 Interrupts 寄存器进行描述。

| 名称 | 地址 偏移量 | 位宽 | R/W | 复位值 | 描述 |
|-----------------------------------|-----------|----|-----|-------|----------------|
| Port A Data Register | 0x00 | 32 | R/W | 32'b0 | Port A 数据寄存器 |
| Port A Data Direction Register | 0x04 | 32 | R/W | 32'b0 | Port A 数据方向寄存器 |
| Port A Data Source | 0x08 | 32 | R | 32'b0 | Port A 源数据寄存器 |
| External Port A | 0x50 | 32 | R | 32'b0 | Port A 外部数据寄存器 |
| Interrupt enable | 0x30 | 32 | R/W | 32'b0 | 中断使能寄存器 |
| Interrupt mask | 0x34 | 32 | R/W | 32'b0 | 中断屏蔽寄存器 |
| Interrupt level | 0x38 | 32 | R/W | 32'b0 | 中断类型寄存器 |

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

| | | | | | |
|-----------------------|------|----|-----|-------|------------|
| Interrupt polarity | 0x3c | 32 | R/W | 32'b0 | 中断极性寄存器 |
| Interrupt status | 0x40 | 32 | R | 32'b0 | 中断状态寄存器 |
| Raw Interrupt status | 0x44 | 32 | R | 32'b0 | 未屏蔽中断状态寄存器 |
| Clear interrupt | 0x4c | 32 | W | 32'b0 | 中断清除寄存器 |
| Synchronization level | 0x60 | 32 | R/W | 32'b0 | 中断同步寄存器 |

图表 2-26 Port A I/O 寄存器和 Interrupts 寄存器描述

Port A Data Register

| 位 | 名称 | R/W | 描述 |
|------|--------------|-----|--|
| 31:8 | 保留，读为 0 | | |
| 7:0 | Port A 数据寄存器 | R/W | 在端口 A 为输出状态时，写该寄存器即为端口 A 的输出值。读该寄存器所得到的值即为最后一个更新这个寄存器的值。 |

图表 2-27 Port A 数据寄存器

Port A Data Direction Register

| 位 | 名称 | R/W | 描述 |
|------|----------------|-----|--|
| 31:8 | 保留，读为 0 | | |
| 7:0 | Port A 数据方向寄存器 | R/W | 该寄存器低 16 位中的每一位都控制端口 A 的每个引脚为输入或者输出。 0: 输入 1: 输出 |

图表 2-28 Port A 数据方向寄存器

Port A Data Source

| 位 | 名称 | R/W | 描述 |
|------|---------------|-----|-----------------------------|
| 31:8 | 保留，读为 0 | | |
| 7:0 | Port A 源数据寄存器 | R | 该寄存器为 0 值，即端口 A 的数据均来自软件模式。 |

图表 2-29 Port A 源数据寄存器

External Port A

| 位 | 名称 | R/W | 描述 |
|------|---------|-----|----|
| 31:8 | 保留，读为 0 | | |

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

| | | | |
|-----|----------------|---|--|
| 7:0 | Port A 外部数据寄存器 | R | 在端口 A 为输入状态时，读这个地址得到的值即为外部输入的值；在端口 A 为输出状态时，读这个地址得到的值为 Port A 数据寄存器的值。 |
|-----|----------------|---|--|

图表 2-30 Port A 外部数据寄存器

Interrupt enable

| 位 | 名称 | R/W | 描述 |
|------|---------|-----|---|
| 31:8 | 保留，读为 0 | | |
| 7:0 | 中断使能寄存器 | R/W | 该寄存器中低 8 位的每一位分别控制 Port A 的每个引脚的中断使能。 0：禁能中断 1：使能中断 |

图表 2-31 中断使能寄存器

Interrupt mask

| 位 | 名称 | R/W | 描述 |
|------|---------|-----|--|
| 31:8 | 保留，读为 0 | | |
| 7:0 | 中断屏蔽寄存器 | R/W | 该寄存器中低 8 位的每一位分别对 Port A 的每个引脚中断进行屏蔽。 0：不屏蔽中断 1：屏蔽中断 |

图表 2-32 中断屏蔽寄存器

Interrupt level

| 位 | 名称 | R/W | 描述 |
|------|---------|-----|---|
| 31:8 | 保留，读为 0 | | |
| 7:0 | 中断类型寄存器 | R/W | 该寄存器中低 8 位的每一位分别对 Port A 的每个引脚中断的类型进行设置 0：电平触发中断 1：边沿触发中断 |

图表 2-33 中断类型寄存器

Interrupt polarity

| 位 | 名称 | R/W | 描述 |
|---|----|-----|----|
|---|----|-----|----|

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

| | | | |
|------|---------|-----|---|
| 31:8 | 保留，读为 0 | | |
| 7:0 | 中断极性寄存器 | R/W | 该寄存器中低 8 位的每一位分别对 Port A 的每个引脚中断的极性进行设置 0: 低电平或下降沿触发中断 1: 高电平或上升沿触发中断 |

图表 2-34 中断极性寄存器

Interrupt status

| 位 | 名称 | R/W | 描述 |
|------|---------|-----|--|
| 31:8 | 保留，读为 0 | | |
| 7:0 | 中断状态寄存器 | R | 该寄存器中低 8 位的每一位分别描述了 Port A 每个引脚的已屏蔽中断状态。 0: 无屏蔽中断 1: 有屏蔽中断 |

图表 2-35 中断状态寄存器

Raw interrupt status

| 位 | 名称 | R/W | 描述 |
|------|------------|-----|--|
| 31:8 | 保留，读为 0 | | |
| 7:0 | 未屏蔽中断状态寄存器 | R | 该寄存器中低 8 位的每一位分别描述了 Port A 每个引脚的未屏蔽中断状态。 0: 无未屏蔽中断 1: 有未屏蔽中断 |

图表 2-36 未屏蔽中断状态寄存器

Clear interrupt

| 位 | 名称 | R/W | 描述 |
|------|---------|-----|---|
| 31:8 | 保留，读为 0 | | |
| 7:0 | 中断清除寄存器 | W | 该寄存器中低 8 位的每一位分别清除 Port A 每个引脚的边沿触发中断。 0: 不清除中断 1: 清除中断 |

图表 2-37 中断清除寄存器

C-Sky Confidential

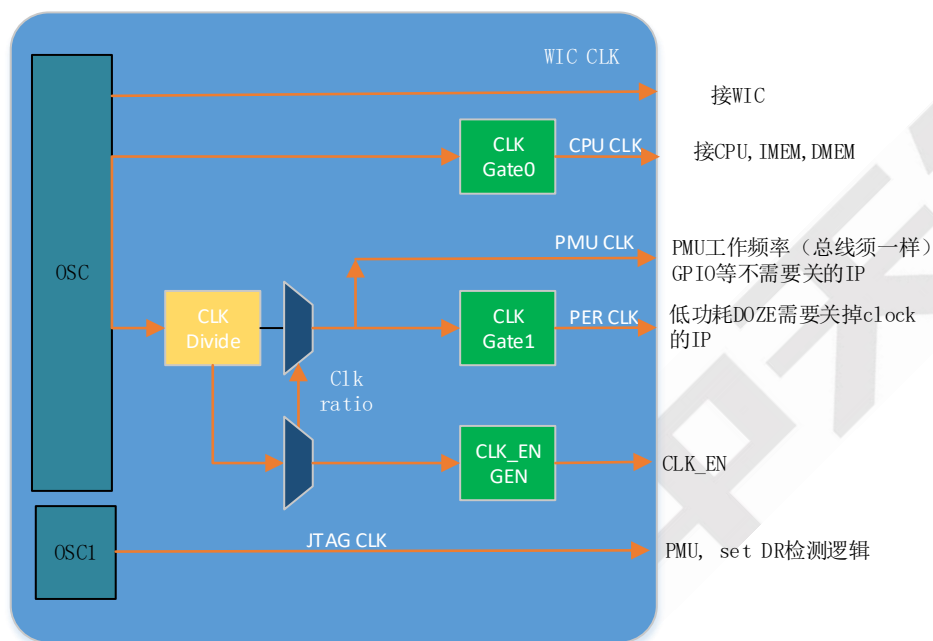
The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

Synchronization level

| 位 | 名称 | R/W | 描述 |
|------|---------|-----|--|
| 31:8 | 保留，读为 0 | | |
| 7:0 | 中断同步寄存器 | R/W | 该寄存器中低 8 位的每一位分别控制 Port A 每个引脚的电平触发中断是否同步于 pclk_intr。 0: 不同步 1: 同步 |

图表 2-38 中断同步寄存器

2.4.5 Clock Gen



图表 2-39 clock gen 结构图

Clock Gen 是一个生成并控制时钟信号的模块。主要由 Clock divider 和 Clock aligner 两部分组成。其中 Clock Divider 输入为 forever cpuclock，生成 clk ratio 0~7 的所有时钟及相应的 clk_en 信号。Clock Aligner 输入为 Divider 生成的所有时钟信号，根据配置寄存器信息进行选择，并实现动态变频的控制。同时 PMU 单元产生的时钟控制信号也会输入该单元实现对特定时钟信号的控制。

Clock Gen 的功能特点如下：

- 1 个配置寄存器，实现 clock ratio 0 -7 的动态变频
- 变频后产生对应的 clk en 信号

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

- 根据低功耗场景控制响应的时钟信号

产生的时钟包括：

| Clock 名称 | 应用模块 | 分频情况 | Clock gating |
|----------|--------------------------|------|-----------------|
| pmu_clk | GPIO TIMER STIMER PMU | 分频 | 不关闭 |
| per_clk | 系统总线及系统总线上的其他 IP | 分频 | Doze Stop 模式下关闭 |
| cpu_clk | CPU 及 MEM | 不分频 | 所有低功耗模式下关闭 |
| wic_clk | WIC | 不分频 | 不关闭 |

图表 2-40 生成时钟列表

Clock Gen 寄存器地址：

| Address | Function |
|------------|----------|
| 0x40017000 | 时钟频率控制 |

图表 2-41 clock gen 寄存器地址

寄存器设置位描述：

| 位 | 名称 | R/W | 描述 |
|------|---------|-----|-----------------------|
| 31:3 | 保留，读为 0 | | |
| 2:0 | 时钟频率 | R/W | 支持 clk ratio 0 -7 的配置 |

图表 2-42 clock gen 寄存器描述

2.4.6 System MPU (SMPU)

SMPU 是一个用于保护特定安全区域的模块。具有三个可配置的表项，可分别对应保护三段不同的地址空间。如配置有 TEE，非可信世界下通过系统总线访问被保护的区间将会返回访问错误。

SMPU 的功能特点如下：

- 实现系统总线上安全地址区间的保护；
- 三个保护区控制寄存器可配置；

SMPU 保护区控制寄存器地址：（SMPU 基址为 0x4001A000）

| Address Range (offset) | Function |
|------------------------|----------|
|------------------------|----------|

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

| | |
|-----|------------------|
| 0x0 | Entry1 Registers |
| 0x4 | Entry2 Registers |
| 0x8 | Entry3 Registers |

图表 2-43 SMPU 寄存器地址

在 SmartL 使用时三个 entry 可分别用于保护系统存储器上的自定义空间，SMPU 本身的地址空间以及 STIMER 的地址空间。保护空间大小 256B 到 1KB。

下面以 Entry1 为例进行寄存器描述，Entry2,3 的结构与功能与 1 完全相同。

| 位 | 名称 | R/W | 描述 |
|------|--------------|-----|--------|
| 31:9 | Base address | R/W | 保护区基地址 |
| 8:5 | 保留，读为 0 | | |
| 4:1 | Size | R/W | 保护区大小 |
| 0 | E | R/W | 使能位 |

图表 2-44 SMPU 寄存器描述

Base Address-保护区地址的基地址：

该寄存器指出了保护区地址的基地址，但写入的基地址必须与设置的页面大小对齐

例如设置页面大小为 1K，SPACR[10:9]必须为 0，各页面的具体要求见下表：**错误!未找到引用源。**

Size-保护区大小：

保护区大小从 256B 到 1KB，它可以通过公式：保护区大小= $2^{(Size+1)}$ 计算得到。Size 可设置的范围为 0111 到 1001，其它一些值都会造成不可预测的结果。

| Size | 保护区大小 | 对基地址的要求 |
|-----------|-------|-------------|
| 0000—0110 | 保留 | — |
| 0111 | 256B | 没有要求 |
| 1000 | 512B | Bit[9]=0 |
| 1001 | 1KB | Bit[10:9]=0 |

图表 2-45 保护区大小及其基地址要求

E-保护区有效设置：

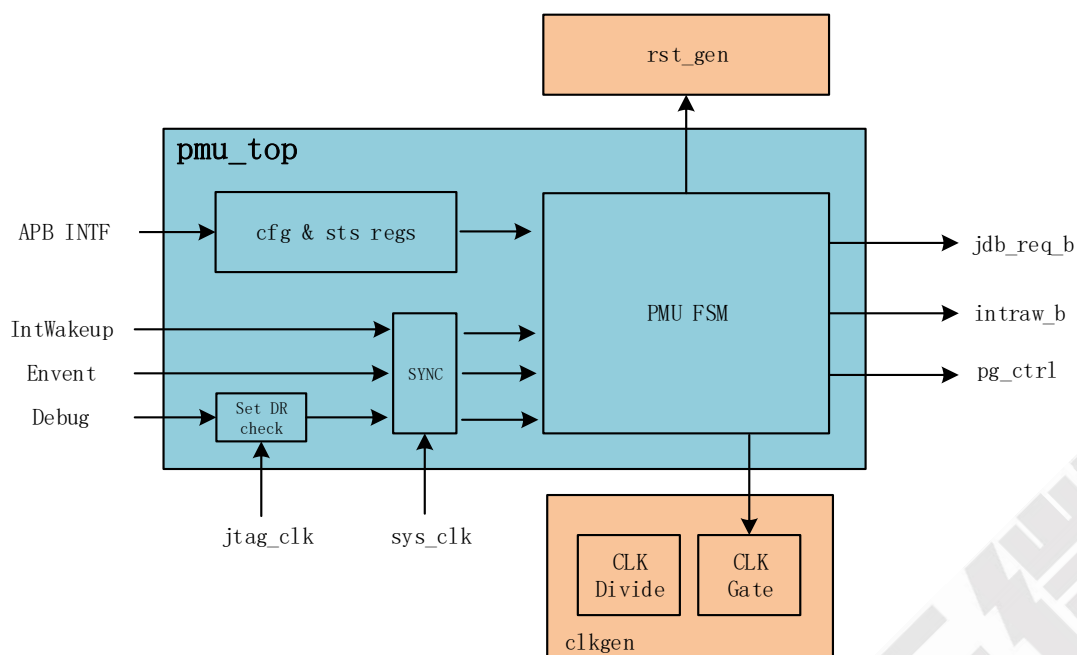
当 E 为 0 时，保护区无效；

当 E 为 1 时，保护区有效。

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

2.4.7 PMU



图表 2-46 PMU 结构图

PMU 为功耗管理单元，负责系统在各个低功耗场景下的时钟、电源及唤醒机制的管理。在任何情况下该模块均不断电，可在任何低功耗场景下采样唤醒源进行低功耗唤醒。有一个可配置的控制寄存器，对唤醒源是否有效进行控制。子模块 tap_2_sm 负责监视 jtag 上写操作，产生相应的唤醒源。

PMU 的功能特点如下：

- 在低功耗场景下可采样调试，中断及事件唤醒源，并发出唤醒请求；
- 有可配置寄存器，可配置不同类型唤醒源的使能；
- 在低功耗场景下控制对应时钟的 clock gating 控制信号；
- 在 STOP 模式下控制进入低功耗及唤醒机制，生成相应的控制及 reset 信号。

PMU 寄存器地址：

| Address | Function |
|------------|----------|
| 0x40016000 | 唤醒源使能 |
| 0x40016004 | 计数器计数值 |

图表 2-47 PMU 寄存器地址

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

寄存器设置位描述:

| 位 | 名称 | R/W | 描述 |
|------|---------|-----|---------|
| 31:4 | 写无效读为 0 | | 保留 |
| 3 | 计数器使能 | R/W | 计数器开始计数 |
| 2 | 调试唤醒使能 | R/W | 调试可唤醒 |
| 1 | 事件唤醒使能 | R/W | 事件可唤醒 |
| 0 | 中断唤醒使能 | R/W | 中断可唤醒 |

图表 2-48 PMU 唤醒源使能寄存器

2.4.7.1 CK802 与 CK801/CK803S/CK804/CK805 相关接口比较

CK802 的相关信号

| 信号名 | I/O | Reset | 定义 |
|----------------------------|-----|-------|-------------------|
| 中断相关信号: | | | |
| pad_vic_int_vld[31:0] | I | 0 | 中断有效信号: 高电平有效。 |
| vic_wic_int_exit[31:0] | O | 0 | 退出中断信息 |
| vic_wic_pending_clr[31:0] | O | 0 | 清除中断 pending |
| vic_wic_pending_set[31:0] | O | 0 | Set 中断 pending |
| 唤醒控制信号: | | | |
| wic_vic_int_awake_en[31:0] | I | 0 | 中断唤醒的使能位 |
| intraw_b | I | 1 | 唤醒请求 |
| had_pad_wakeup_req_b | O | 1 | Jtag 唤醒源 |
| vic_wic_awake_data[31:0] | O | 0 | Awake enable 信息 |
| vic_wic_awake_disable | O | 0 | 写 IWDR 操作 |
| vic_wic_awake_enable | O | 0 | 写 IWER 操作 |
| vic_wic_intraw_ack | O | 0 | 唤醒握手信号 |

图表 2-49 CK802 接口

CK801/CK803S/CK804/CK805 的相关信号:

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

| 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。 |

图表 2-50 CK801/803S/CK804/CK805 信号

由上表可见 CK801/CK803S/CK804/CK805 仅能通过 PMU 输入给 CPU 中断信号及中断配置信息。对于 E902 来说 vic 需替换为 clic。CK802 与 PMU 交互的信号在中断处理方面额外包括了 set pending 触发中断的信息，以及退出中断的信息；在唤醒方面交互了 jtag 唤醒源的部分信息，awake en 的维护信号以及唤醒的握手信号。所以在实现相关功能时, CK802 的设计与 CK801/CK803S/CK804/CK805 有所不同，具体不同点将在各个功能点进行描述。

2.4.7.2 唤醒控制

PMU 模块实现了下列三个唤醒源

- 事件唤醒

在低功耗模式下任何事件，都会造成事件唤醒源处于 pending 状态，直到低功耗模式被唤醒。只有该唤醒源 pending 且对应的控制位处于使能状态时该唤醒源能恢复时钟，产生有效的唤醒请求。

- 调试唤醒

在低功耗模式下通过观测 jtag2 状态机，任何通过 jtag 的写操作都会造成调试唤醒源处于 pending 状态，直到低功耗模式被唤醒。只有该唤醒源 pending 且对应的控制位处于使能状态时该唤醒源能恢复时钟，产生有效的唤醒请求。

- 中断唤醒

如 WIC 模块中有中断处于 pending 状态响应的中断，对应的 WIC 中存储的 awake en 处于使能状态且控制寄存器使能中断源唤醒时，该唤醒源能恢复时钟，产生有效的唤醒请求。

除 802 外其他核均不能直接向 cpu 发送唤醒请求。所以 PMU 只能通过向中断控制器发出中断请求，由中断控制器产生相应的唤醒请求。同时不支持 set pending 所引起的中断，因此只实现了部分中断唤醒的功能。

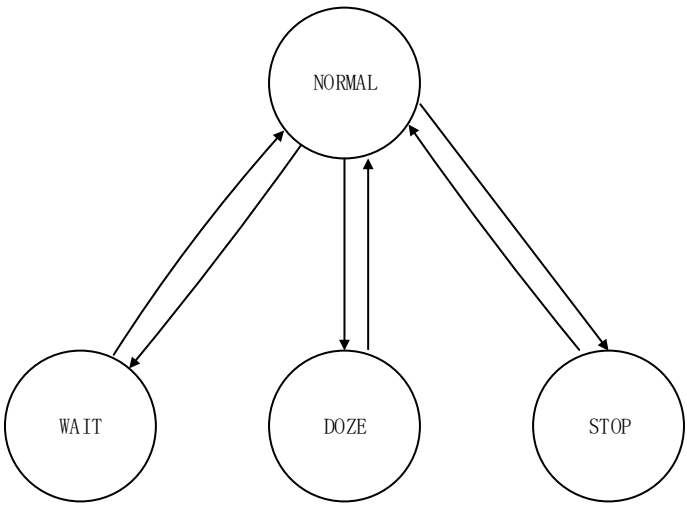
2.4.7.3 低功耗状态管理及相关控制

低功耗状态的管理主要通过主状态机和 power gating 状态机实现。

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

主状态机主要实现各个低功耗场景下的时钟控制



图表 2-51 PMU 状态机

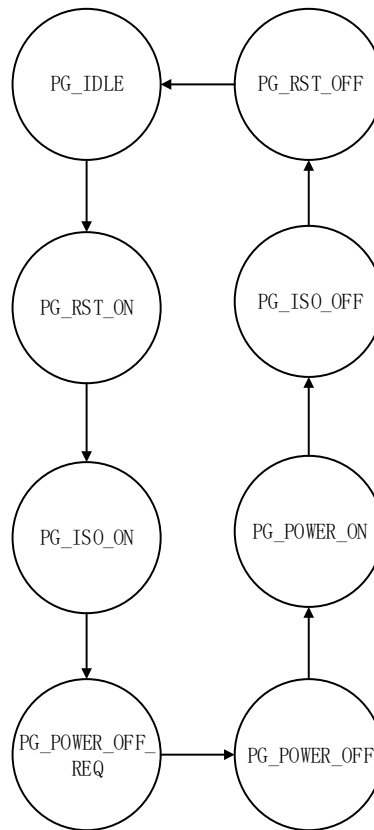
在 normal 状态下 所有时钟均不进行控制，任何 lpmd 模式的改变都会进入对应的 lpmd 状态。

- 1) 在 wait 状态下 仅关闭 CPU clk ,在 wakeup 后 返回 normal
- 2) 在 doze 状态下 关闭 CPU clk 和 per clk ,在 wakeup 后 返回 normal
- 3) 在 stop 状态下 关闭 CPU clk 和 per clk 并进入启动 power gating 状态机，只有等待 power gating 状态机完成所有处理后返回 IDLE 模式

| 时钟 | 应用模块 | Clock gating |
|---------|-----------------------|-----------------|
| pmu_clk | GPIO TIMER STIMER PMU | 不关闭 |
| wic_clk | WIC | 不关闭 |
| per_clk | 系统总线及系统总线上的其他 IP | Doze Stop 模式下关闭 |
| cpu_clk | CPU 及 I/D MEM | 所有低功耗模式下关闭 |

图表 2-52 时钟分布

Power gating 状态机主要负责 power gating 过程以及 reset 相关信号的管理。



图表 2-53 power gating 状态机

PG_IDLE 主状态机进入 STOP 状态后,状态机启动

PG_RST_ON 下个周期无条件进入 PG_ISO_ON,开始 reset 操作

PG_ISO_ON 下个周期无条件进入 PG_POWER_OFF_REQ, 开始隔离操作

PG_POWER_OFF_REQ 等待 CPU 完成断电操作后进入 PG_POWER_OFF, 否则保持状态等待

PG_POWER_OFF 等待唤醒, 任何有效唤醒, 进入 PG_POWER_ON, 否则保持状态等待

PG_POWER_ON 下个周期无条件进入 PG_ISO_OFF, 开始恢复供电操作

PG_ISO_OFF 下个周期无条件进入 PG_RST_OFF, 结束隔离操作

PG_RST_OFF 下个周期无条件返回 IDLE, 结束 reset 操作

该状态机负责管理如下信号:

PG_RESET 该信号仅在 IDLE 和 RESET_OFF 两个状态下为高。和 pad_cpu_rst_b 相与后生成 pg_reset 用于在 power gating 时给除了 mem 外所有关闭时钟的模块进行 reset

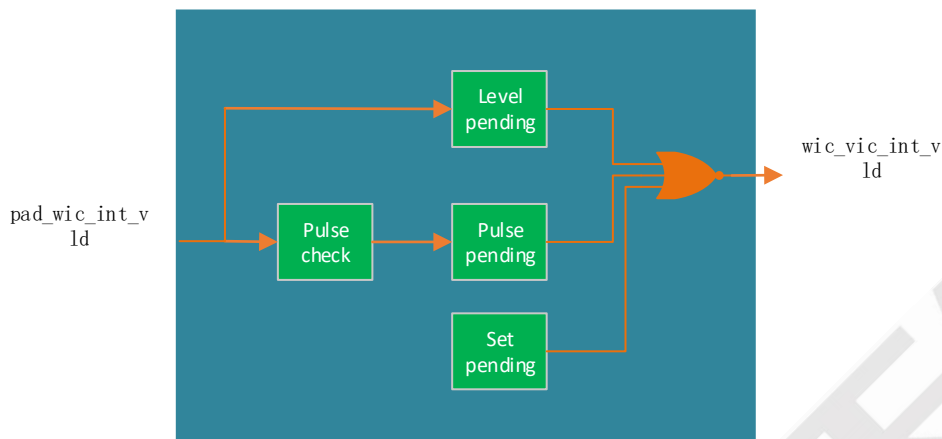
C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

PG_SLEEP_IN 表示处于 power gating 状态，仅在状态机处于 PG_POWER_OFF_REQ 和 PG_POWER_OFF 下有效

PG_ISOLATION 表示系统进行隔离处理，当状态机处于 PG_ISO_ON, PG_POWER_OFF_REQ, PG_POWER_OFF, PG_POWER_ON, PG_ISO_OFF 时有效

2.4.8 WIC



图表 2-54 WIC 结构图

WIC 模块负责在任何场景下进行中断的采样与处理。在任何场景下均不断电，clock 均不停止。最大支持 32 个中断源。对电平、脉冲及 SET pending 操作等行为均会使中断请求处于 pending 状态，并将中断请求发送至 PMU 进行唤醒控制，同时发送至中断控制器，直到中断被响应后清除。WIC 也负责 awake en 的维护，任何对 IWER 和 ISDR 的操作都将作用于 WIC 内的相关寄存器，并将该使能信息传递给 PMU。

除 CK802 外，smart 平台上的 WIC 均不支持 set pending 所引起的中断。不负责 awake en 的维护，寄存器组写无效读为 0，使能信息保存在中断控制器当中。对电平及脉冲中断源进行采样后，处于 pending 状态，拉起 int_vld 信号，并将 cfg 全部置 0，以电平中断的形式向中断控制器发出请求。当 cpu 退出中断模式后清除 pending 状态。

3 文件结构及说明

3.1 tools/

工具的存放目录，包含仿真时用到的工具、设置环境变量的源文件，将 hex 文件转换成 pattern 文件的转换工具，以及仿真时运行的脚本。

3.1.1 CTS_MAP_TOOLS/

存放着用于 remap 功能的脚本，可以根据用户设定的环境地址和配置重新生成用户所需要的 crt0.s、linkefile 等环境文件，同时筛选出所有需要的测试程序并生成相应的 pat 文件，帮助用户快速建立开发环境。

3.2 lib/

包含程序运行所需的库文件，包括初始化相关的程序、链接描述文件、异常向量表和异常服务程序。在仿真的时候会根据 CPU 型号将对应所需的文件拷贝到工作目录下，然后编译仿真。

3.3 cpu/

存放 RTL 源代码的目录，包括 CPU 的仿真模型以及相关的 Memory IP，除 CPU 模型外这些代码都是可综合的。

3.4 soc/

存放 SMART 平台的源代码目录，包括 SOC 平台的外围 IP 以及总线仲裁器等 RTL 代码。

3.5 tb/

测试平台以及相关的仿真工具模型存放目录，包含时钟、复位等信号的产生以及 pat 文件下载、仿真行为结束判定以及 CPU 程序运行轨迹的检测等文件。

3.6 synthesis/

存放测试平台的时序约束文件以及用于低功耗设计的 upf 文件。

3.7 case/

case 文件夹存放测试激励的目录，包含测试实例其形式有 C 语言和汇编两种类型，包含性能、功能、低功耗、JTAG 等测试实例。

3.8 gen_case/

gen_case 文件夹存放着使用 gen_case.pl 脚本根据配置参数生成的用于 smart 平台仿真的 case 目录。

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

3.9 workdir/

仿真运行的临时工作目录，仿真过程中的临时文件包括编译文件、verilog 可执行文件、波形文件等均在此目录中。

3.10 CSKY_Test_Suit/

此文件夹为根据用户配置 `cts_map_cfg.h` 文件而生成所有用户需要的文件，包括环境配置文件、case 文件、生成的 pat 文件等。

3.11 regress/

包含用于测试程序测试的自动化脚本，可以帮助用户自动化测试所有测试程序并生成运行结果报告。有关 regress 详细内容请参考 5.4 节。



C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

4 仿真流程

4.1 建立环境变量

首先需要 source tools/目录下面的 setup.csh 文件，将所有需要的环境变量建立起来。下图展示的是些必须的环境变量（具体设置仅做参考）：

```
source /home/cshrc/vcs2010.06.cshrc
setenv TOOL_PATH= /home/zhaok/toolchain/3.6.10
setenv SMART_PATH `pwd | perl -pe "s/smartL.*/smartL/"`
alias run_case $SMART_PATH/tools/run_case
```

图表 4-1 建立环境变量

- Source VCS 的配置文件，建议使用最新版本的 VCS，尽量在 2010.06 版本以上。由用户定义。
- 将工具链的安装目录设置为环境变量 TOOL_PATH。
- 重命名 run_case，由用户自定义。
- 上述 3.6.10 仅针对 CK801/CK802/CK803/CK804/CK805，E902 需安装 RISC-V 相应的工具链。

4.2 执行仿真脚本

整个仿真过程将围绕 run_case 脚本展开具体分为以下四部分：

4.2.1 准备测试程序所需文件

首先需要清空 workdir/目录下面的所有的文件，以保证接下来准备的所有文件都是本次仿真所必须的。同时将所需要用的文件拷贝到 workdir/目录下面准备编译。

✧ **Makefile:** 整个编译过程的主控文件。具体可以分为以下几部分内容。

- **确定编译所需要使用的工具**

使用方法参考软件工具的用户手册。具体如下图所示：

```
TOOL_EXTENSION = ${TOOL_PATH}/bin/csky-abiv2-elf-
CC = ${TOOL_EXTENSION}gcc
AS = ${TOOL_EXTENSION}as
LINK = ${TOOL_EXTENSION}ld
OBJDUMP = ${TOOL_EXTENSION}objdump
OBJCOPY = ${TOOL_EXTENSION}objcopy
CONVERT = ../tools/Srec2vmem
```

图表 4-2 确定编译工具

如上图所示使用的是 C-SKY v2 的工具，所以在前面 setup.csh 中必须引用 C-SKY v2 的安

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

装路径，否则会提示这些工具无法被找到。如下图所示：

```
csky-abiv2-elf-gcc -c -Wa,-mno-lrw -fno-inline -static -g3 -I/tools/cskytools/csky-abiv2-elf-tools-x86_64-20130312/csky-abiv2-elf/include -I. -mlittle-endian -mcpu=ck802 -Dck802 -Wa,--defsym=ck802=1 -Os -o crt0.o crt0.s
make: csky-abiv2-elf-gcc: Command not found
make: *** [crt0.o] Error 127
can't make at ../tools/run_case_line 409.
```

图表 4-3 编译工具安装路径没有被添加到环境变量时的报错

若使用的是 E902 产品，需要将 **TOOL_EXTENSION** 改为如下设置：

TOOL_EXTENSION = \${TOOL_PATH}/bin/riscv32-unknown-elf-

- 获取源文件以及扩展路径（如果需要）

```
SSRC = $(wildcard *.S)
sSRC = $(wildcard *.s)
CSRC = $(wildcard *.c)
OBJECTS = $(SSRC:%.S=%.o) $(sSRC:%.s=%.o) $(CSRC:%.c=%.o)
#INCDIR = -I/home/zhaok/toolchain/3.6.10/csky-elfabiv2/include -I.
```

图表 4-4 源文件以及扩展路径

- 设置 c 编译参数以及链接参数

根据不同的 CPU 核来选择对应的 C 编译标志以及链接编译标志，在下图中需特殊指出的是由于该 Makefile 同时支持了 CK801/CK802/CK803/CK803S/CK804/CK805/E902，所以会根据不同的 CPU 来进行相应的配置。如果用户只需要一种 CPU 核，则在该部分只需要配置该 CPU 相关的参数即可。如下图所示：

```
DEFSYM = $(shell echo ${CPU} | sed 's/\(ck[0-9]\{3\}[s]*\).*\1/')
CFLAGS = -m${ENDIAN_MODE} -mcpu=${CPU}
CFLAGS += -D${DEFSYM} -Wa,--defsym=${DEFSYM}=1
ifeq ($(findstring yes, ${SIZE}), yes)
    CFLAGS += -Os
else
    CFLAGS += -O2
endif
LINKFLAGS = -mcpu=${CPU} -mstm -m${ENDIAN_MODE} -Tlinker.lcf -nostart
files
```

图表 4-5 设置 C 编译参数以及链接参数

注意：如果用户所使用的 CPU 核包含 FPU 配置，则需要在 CFLAGS 加入 “-mhard-float” 参数，同样 LINKFLAGS 也应加上该参数。对于 E902 来说，编译选项应按如下设置：

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

CFLAGS = -march=rv32emc -mabi=ilp32e

➤ **设置链接库**

对于用户自行添加的测试程序，可能需要将某些库文件增加进入编译过程，例如 `libc`，`libgcc`。不同核的链接库文件在工具链安装路径下的不同路径下，所以需要用户按照实际的安装目录来修改该段文件。具体如下图所示：

```
LINKLIBS = -L${TOOL_PATH}/csky-elfabiv2/lib/ck803
LINKLIBS += -L${TOOL_PATH}/lib/gcc/csky-abiv2-elf/4.5.1/ck803
LINKLIBS += -lc -lgcc
```

图表 4-6 设置链接库文件

➤ **编译过程**

整个编译过程遵循从 `.c/.s->.o->.elf->.hex->.pat` 的过程。具体如下图所示：

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

```

OBJDUMPFLAGS = -S
HEXFLAGS = -O srec
%.o : %.c
    {CC} -c {CFLAGS} -o $@ $<
%.o : %.s
    {CC} -c {CFLAGS} -o $@ $<
%.o : %.S
    {CC} -c {CFLAGS} -o $@ $<
{FILE}.elf : {OBJECTS} linker.lcf
    {CC} {LINKFLAGS} {LINKLIBS} {OBJECTS} -o $@ -lm
{FILE}.obj : {FILE}.elf
    {OBJDUMP} {OBJDUMPFLAGS} $< > $@
INST_HEX = {FILE}_inst.hex
DATA_HEX = {FILE}_data.hex
{FILE}.hex : {FILE}.elf
    {OBJCOPY} {HEXFLAGS} $< {INST_HEX} -j .text*
    {OBJCOPY} {HEXFLAGS} $< {DATA_HEX} -j .data -j .rodata -j .b
ss
    {OBJCOPY} {HEXFLAGS} $< $@
INST_PAT = inst.pat
DATA_PAT = data.pat
%.pat : %.hex
    rm -f *.pat
    {CONVERT} {INST_HEX} {INST_PAT}
    {CONVERT} {DATA_HEX} {DATA_PAT}
# target setting
.PHONY :all
all : {FILE}.pat {FILE}.hex {FILE}.elf {FILE}.obj
# clean some medium code and .pat
.PHONY :clean
clean: rm -rf *.o *.pat *.elf *.obj

```

图表 4-7 编译流程

✧ **Linker.lcf:** 链接描述文件

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

在上文的 Makefile 中几次出现了 linker.lcf 文件，此文件就是在链接过程中向编译工具指示链接地址的文件，如下图所示：

```
MEMORY
{
    MEM1(RWX) :ORIGIN = 0x00000000,LENGTH = 0x20000
    MEM2(RWX) :ORIGIN = DATA_BADDR,LENGTH = 0x10000
    RAND(RW):  ORIGIN = 0x1eff0,    LENGTH = 0x1000
}

__kernel_stack = 0x6000fff8;
__user_stack = 0x6000fcf8;
__tkernel_stack = 0x6000f8f8;
__tuser_stck = 0x6000f4f8;

ENTRY(__start)
SECTIONS
{
    .rodata: {  *(.rodata*)  } >MEM2
    .text :
    {
        crt0.o (.text)
        *(.text*)
        *(.rodata)
    } >MEM1
    .data :
    {
        *(.data)
    } >MEM2
    .bss :
    {
        *(.bss)
        *(COMMON)
    } > MEM2
    .rand :
    {  *(.rand)  } >RAND
}
```

图表 4-8 链接描述文件

如上图所示，SmartL 平台上有三块内存，分别用作指令存储器、数据存储器、系统存储器，CPU 的总线条数配置不同这几块 memory 的位置也不同，请参考 1.2 节的 SmartL 架构

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

图，无论什么配置都三块存储器都存在是为了让连接环境更简单，恒定将指令连接到 0x0 开始的地址，将数据链接到 0x20000000 开始的地址。用户可以手动修改链接文件，根据自己的仿真需求进行调整。

linker.lcf 首先需要定义这两块内存，定义它们的起始地址以及长度，同时定义好堆栈指针的地址，之后则是程序在内存中的布局映射。本地指令存储器存储指令段，即.text 段。优先将 crt0.o 编译在最前面，由于 crt0.o 是初始化文件，程序的入口在这定义，所以一定要保证 crt0.o 是在指令段以及整块内存的最前面。.rodata、.data 和.bss 是数据相关的内容，填充系统内存。

✧ **crt0.s:** 初始化头文件，是整个程序的起始，主要执行以下几项任务：

➤ **初始化异常向量表**

简单起见将所有的异常入口地址均指向到程序失败结束的地址。如果程序确实需要出现异常，则需要由用户在程序中定义异常服务程序，并将该异常向量表对应的异常向量号上的异常入口地址修改成用户定义的异常服务程序的入口。否则一旦出现异常，程序会立刻结束并且显示程序执行失败。对于 E902 来说需要在由 mtvec 寄存器指向的异常入口 __trap_handler 处定义异常服务程序，并且该程序可以通过查询 mcause 中的异常编号来决定进一步跳转到更具体的异常服务程序。

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

```
.text
.export vector_table
.align 10
.export __start
.ifdef e902
__trap_handler:
    j __synchronous_exception
.align 2
j __asynchronous_int
.endif
vector_table:                //totally 256 or 128 entries
    .long    __start
    .ifdef ck801
        .rept    128
        .long    __dummy
        .endr
__start:
    .....
    .....
    .endif

    .ifdef ck802
        .rept    128
        .long    __dummy
        .endr
__start:
    .....
    .....
    .endif

    .ifdef ck803s
        .rept    255
        .long    __dummy
```

图表 4-9 初始化异常向量表

➤ 初始化寄存器

根据寄存器类型不同可以分为通用寄存器和可选择寄存器两类，下图所示的是通用寄存器的初始化。

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

```
__start:  
//initialize all registers  
.ifdef e902  
    li x1, 0  
    li x2, 0  
    ...  
    li x15, 0  
.else  
    movi r0, 0  
    movi r1, 0  
  
    ....(略)  
  
    movi r14, 0  
    movi r15, 0  
    .ifdef ck803s  
    movi r16, 0  
    movi r17, 0  
  
    ....(略)  
  
    movi r30, 0  
    movi r31, 0  
    .endif  
.endif
```

图表 4-10 初始化通用寄存器

➤ 设置并初始化堆栈指针

将堆栈指针地址赋值给堆栈指针寄存器 SP。

```
//initialize kernel stack  
lrw r14, __kernel_stack  
lrw r4, __kernel_stack  
lrw r4, 0x40  
lrw r5, 0x0  
INIT_KERLE_STACK:  
    addi r4, 0x4  
    st.w r5, (r4)  
    cmphs r14, r4  
    bt INIT_KERLE_STACK
```

图表 4-11 设置堆栈指针

➤ 设置 VBR 以及 PSR

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

将异常向量表的起始地址设置成 VBR 的值，并且将 PSR 的 EE, IE 位打开，允许发生异常。

```
//set VBR
    lrw r2, vector_table
    mtc r2, cr<1,0>
//enable EE IE bit of psr
    mfc r2, cr<0,0>
    bseti r2, r2, 6
    bseti r2, r2, 8
    mtc r2, cr<0,0>
```

图表 4-12 设置 VBR 以及 PSR

➤ 进入测试程序

```
__to_main:
    lrw r15, __exit
    lrw r0, main
    jmp r0
```

图表 4-13 进入测试程序

为了使程序能够正常返回，在进入测试程序之前需要保存退出的地址进入到链接寄存器中，尤其是使用 C 语言等高级语言作为测试语言的时候更加需要注意。

➤ 设置相关标志段



```
.export __exit
__exit:
    mfcr r1, cr<0,0>
    movih r1, 0xFFFF
    mtc r1, cr<11,0>
    movi r1, 0xFFF
    lrw r0, __kernel_stack
    st r1, (r0)
.export __fail
__fail:
    movih r1, 0xEEEE
    mtc r1, cr<11,0>
    movi r1, 0xEEE
    lwr r0, __kernel_stack
    st.w r1, (r0)
__dummy:
    br __fail
```

图表 4-14 设置相关标志段

4.2.2 编译测试程序

准备好所有的测试程序以及测试程序相关的文件之后就可以进行编译和链接了。编译的时候需要确定一下两点：当前需要仿真的 CPU 配置以及测试程序的名字。

| CPU 配置 | 编译输入 |
|-----------------|---------|
| CK801 | ck801 |
| CK802 | ck802 |
| CK803 | ck803 |
| CK803+DSP_MEDIA | ck803e |
| CK803S | ck803s |
| CK804 | ck804 |
| CK804+DSP | ck804e |
| CK804+FPU | ck804f |
| CK804+DSP+FPU | ck804ef |
| CK805 | ck805 |
| CK805+FPU | ck805f |
| E902 | e902 |

图表 4-15 CPU 配置及编译选项

以 dhrystone 程序为例，其程序名为 dhry_1_main.c，在配置 CK803s 的情况下编译整个程序的命令为：make clean；make all CPU=ck803s ENDIAN_MODE=little-endian FILE=dhry_1_main。其中的 make clean 这一步主要是为了清除前次编译产生的中间文件以及

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

目标文件，防止本次编译出现问题时该目录下仍然存在上次编译结果，给用户带来干扰。其次的 `ENDIAN_MODE` 在 SmartL 推荐使用 `little-endian`。另需要注意的是测试程序名是不需要添加后缀的。

编译完成之后会生成 `dhry_1_main.elf`，`dhry_1_main.hex`，`dhry_1_main_inst.hex`，`dhry_1_main_data.hex`，`dhry_1_main.obj`，`inst.pat` 和 `data.pat`。其中 `dhry_1.elf` 和 `dhry_1.hex` 在 SmartL 平台暂时不需要使用。用户需要使用的是 `dhry_1.obj` 文件，这个文件是所有测试程序，包含测试程序相关的文件一起编译链接之后反汇编产生的文件，其中包含了每条指令的地址信息，数据的地址信息，在调试的时候将被频繁使用。

`inst.pat` 和 `data.pat` 文件是分别将 `dhry_1_main_inst.hex` 和 `dhry_1_main_data.hex` 文件中的信息按照 `verilog` 语法中存储器文件的格式书写出来的文件。

4.2.3 VCS 仿真

使用 VCS 进行仿真，具体 VCS 的使用请参考 `synopsys` 公司的用户手册，仿真由两部分组成：`test bench` 和 `soc`。整个 `soc` 的介绍请参考第一章的平台架构，在此主要介绍 `test bench` 以及相关的内容。

➤ 时钟产生器

由于 CPU 存在两个时钟域，分别为 `clk` 和 `jclk`，分别用于驱动 CPU 和硬件辅助调试单元。分别设置 `clk` 的周期为 `10ns`，即 `100M`。`jclk` 的周期为 `33ns`，即 `30M`。时钟在仿真开始时就产生，不会为任何事件中断。

➤ 复位产生器

同样复位信号也会存在两组，分别为 `rst_b` 和 `jrst_b`。复位信号为低电平有效信号，在仿真开始是复位信号为高，`100ns` 以后会将复位信号拉低，因为 CPU 是按照同步设计，所以复位信号必须保证在一个时钟周期以上，`100ns` 以后将复位信号拉高，此时 CPU 进入正常工作模式。

➤ 测试程序下载

在编译完成之后会生成 `inst.pat` 和 `data.pat` 的文件，通过使用 `verilog` 语法中的 `$readmemh` 将这两个文件读入，然后按照存储器的格式将所有的测试程序的数据下载到存储器中。`inst.pat` 下载到本地指令存储器中，`data.pat` 下载到系统内存中。

➤ 存储器初始化

由于测试程序远远小于存储器的容量，所以在程序下载完成之后很多存储器的容量还尚未被利用，在仿真中这部分存储器的行为将会是未知的，即 `x`。为了保证这部分内存不会影响整个程序的运行，我们需要把这部分存储器进行初始化。首先需要获得已经下载的程序长度，然后将存储器总容量减去程序所需容量之后的存储器空间全部初始化成 `0`。

➤ 仿真最大时间

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

程序的长度是固定的，所以仿真需要的时间也是确定的。当程序进入不可控的时候，例如程序跑飞，或者进入死循环的情况下需要能够将程序停下来，并告诉调试人员程序出错。设置程序最大仿真时间可以在仿真到达这个时间的时候自动将仿真停止，并且报错。仿真的最大时间可以由用户自定义。

➤ 无指令退休

当程序使 CPU 锁死，指令均不能正常执行的时候，同样需要将程序停下来，并向调试人员报错。设置无指令退休时间，每隔这个时间仿真程序会检查 CPU 在这段时间内是否有指令退休，如果这段时间内没有任何一条指令退休，就认为 CPU 已经锁死，不在正常工作状态下。仿真程序将会停止并且报错。这个时间可以由用户自定义，另外需要特别注意的是如果 CPU 长时间进入低功耗状态下，CPU 将不会有任何指令退休，用户如果确实需要产生这种情况可以将这个时间调大或者将这项功能关闭。

➤ 程序测试失败

在初始化头文件中有程序测试失败的入口，所有的测试程序都会有自检测测试，如果测试失败就会进入到程序测试失败的入口，只要进入到测试失败的入口仿真就会结束并且报告调试人员程序测试失败，由调试人员来确认测试程序仿真失败的原因。常见错误我们已经列在 4.3 章节。

➤ 程序测试成功

在初始化头文件中有程序测试成功的入口，所有的测试程序都会有自检测测试，如果测试完成并且测试成功，则程序会进入到测试成功的入口，只要进入到测试成功的入口，仿真就会结束并且报告调试人员测试程序成功。

➤ 波形产生器

仿真的所有行为都会以仿真波形的形式保存下来，由用户自定义是否生成波形。波形在程序调试中起着至关重要的功能。一般建议将波形保存下来，VCS 支持 VCD 格式和 FSDB 格式的波形，可以由用户自由选择。另外如果不需要波形，可以通过 run_case 的参数将波形产生器关闭。具体用法参考 2.2.4 章节的说明。

➤ CPU 运行轨迹监视器

CPU 在运行时由 PC 来显示运行轨迹，如果打开 CPU 运行轨迹监视器，将会把 CPU 运行时的所有指令的运行轨迹保存下来，包括 CPU 的指令退休的 PC，以及每条指令退休以后的所有寄存器的状态。需要注意的是如果程序比较长，会导致生成出来的文件非常大，VCS 不支持生成文件超过 4GB，仿真会中途结束，此时需要把该监视器关闭，具体命令参考 2.2.4 章节的说明。

➤ 打印监视

在调试 C 语言等高级语言的时候会需要将一些变量信息以串口的形式打印出来，可以通过一个 UART monitor 来监视并输出打印信息，但是这样仿真速度会非常缓慢，所以 smartL

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

里开辟了一个地址，并对 `printf` 函数进行了修改，将需要打印字符的 ASCII 码存到该地址，打印 `monitor` 会监测该地址并将打印信息通过 `verilog` 打印系统函数打印到仿真窗口。

4.2.4 run_case 脚本应用

`run_case` 脚本将执行前三章节的所有内容，包含测试程序的准备，测试程序的编译以及测试程序的仿真，用户可以自定义各种需求。

- **-h** 用于打印所有的帮助信息，执行示例：`run_case -h`
- **-ma** 用于在仿真前生成 `verilog` 代码，用户一般不需要使用。
- **-sim_tool** 用于选择使用的仿真工具，默认为 `VCS`，支持 `VCS` 和 `NC-verilog`，执行示例：`run_case -sim_tool nc`
- **-nomnt** 仿真时不生成 CPU 运行轨迹监视文件，由用户自定义选择是否生成监视文件，如果有该参数则不生成监视文件。执行示例：`run_case -nomnt`
- **-nodump** 仿真时不生成任何波形文件，由用户自定义选择是否生成波形文件，如果有该参数则不生成波形。执行示例：`run_case -nodump`

所有的参数添加完成之后一定要跟上测试程序的路径以及程序名，否则将不能正常执行。执行示例：`run_case -sim_tool vcs -nodump -nomnt ../case/PVS/dhry/dhry_1_main.c` 该命令表示仿真 `dhrystone` 程序，使用仿真工具为 `VCS`，CPU 核为 `RTL` 模型，不生成波形，不生成 CPU 运行轨迹监视文件。

4.3 测试程序调试

使用前面章节的 `run_case` 脚本即可执行单个测试程序的仿真。如果发现测试程序仿真失败，则需要根据以下几点来确认问题所在。

➤ 检查信号连接是否完整

确认所有信号是否完全连接完整，尤其是 CPU 的输入，具体方式可以在 CPU 的接口上将所有的输入信号的波形检查一遍是否存在高阻态，其中需要注意的是扫描链相关的信号线如果出现高阻态可以忽视。如果其他信号存在高阻态则需要将这些输入信号连接完成。

➤ 检查时钟和复位信号

确认所有的时钟信号和复位信号是否按照时钟产生器和复位产生器的格式生成，如果时钟或者复位信号不正确会直接导致 CPU 无法正常工作。

➤ 检查总线是否发起请求

检查总线是否发起第一次请求，在复位完成以后 CPU 会进入重启异常，然后回从该异常向量表发起请求取得异常服务程序入口地址。该异常向量表地址一般为 0。

➤ 检查总线是否响应请求

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

在 CPU 发起请求之后，总线以及外围的存储器会立即响应，并将 CPU 需要的数据返回。
第一次返回的数据一般为程序的入口地址，也就是重启异常的异常服务程序起始地址。

➤ **检查 CPU 是否无故出现异常**

程序在运行过程中发生错误需要检查 CPU 是否没有按照程序设定出现异常，如果出现异常需要分析出现异常的原因，并做相应的修改。

➤ **检查是否修改指令区**

程序跑飞有可能因为程序中的指令区被程序自身修改，导致程序没有按照需求执行，修改指令区是一件非常危险的事情，所以在程序设计的时候一定要避免出现这种情况。

➤ **检查是否使用非对齐地址访问**

尽管有些 CPU 支持非对齐访问，但是在 SmartL 平台上并不支持该操作，所以如果测试程序中有相关的操作，应该使用其他方式替换掉，否则不能保证程序正常执行。

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

5 SmartL 测试程序

5.1 测试程序分类

针对在技术支持过程中客户经常产生疑问的点，我们在 SmartL 上专门开发了一些演示的测试程序，主要包含三部分内容：

1 功能性测试，覆盖所有的指令功能，编程模型结构功能以及相应的扩展指令的功能。其中编程模型结构功能包含异常处理，mgu 使用，以及各个 func 模块的使用。

2 连通性测试，覆盖集成 soc 平台时需要特别注意的信号连通性的测试。包含中断，jtag 和低功耗相关信号的连接。

3 性能测试，覆盖基本的 benchmark。包含 dhrystone, coremark, linpack 和内存拷贝的性能数据，以及打印 hello world 的简单 c 程序。

用户需要仔细检查每一根输入输出信号线的工作方式以及连接方式，为移植到用户自身平台上做准备。每一个测试程序的着重点都各有不同，所以在移植之前请务必跑通所有的测试程序，并且能够很好的理解所有测试程序想要让用户了解的容易出现问题的地方。

5.2 测试程序具体描述

5.2.1 异常向量表配置演示

名称：exception_test.s，该测试程序演示了如何编写异常服务程序以及设置异常向量表。同时演示了如何触发各个异常。

5.2.2 MGU 配置演示

名称：mgu_test.s，演示配置 MGU。

5.2.3 控制寄存器读写

名称：ck8xx_cp0_fuc.s 演示读写不同组下的控制寄存器

5.2.4 Smoke 测试程序

演示基本指令的使用，覆盖了所有的指令，并带有指令功能正确性自测试。包括基本测试、浮点指令、DSP 指令。

名称：ck8xx_smoke.s

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

名称: ck803s_dsp_media_smoke.s

名称: ck803s_fpu_smoke.s

5.2.5 Cache 使用

名称: ck8xx_cache_test.s, 演示了如何使用 cache, 覆盖从 cache 初始化, 配置, 到最后打开的整个过程。

5.2.6 低功耗握手演示

该测试程序用于演示 CPU 进入各个低功耗模式以及通过不同方式唤醒。

Case 1: Ckcpu_wait.s, Wait 模式下唤醒。Wait 仅停止 CPU 和 mem 控制时钟, 其他模块不受影响。演示步骤:

1. 设置 timer 值并使能 counter;
2. 使能 PMU 中的中断唤醒使能位;
3. 使用 Wait 指令进入低功耗模式;
4. Timer 计数到零, 产生中断唤醒 CPU。

Case 2: Ckcpu_doze.s, Doze 模式下唤醒。Doze 模式停止 CPU 和 mem 控制时钟及其他大多数模块时钟演示步骤:

1. 使能 PMU 中的事件唤醒使能位;
2. 使用 doze 指令进入低功耗模式;
3. force 产生事件, 作为唤醒源唤醒 CPU (仅适用于 802, 其他核唤醒方法参考 case1)。

Case 3: power_gating_main.c, Stop 模式下唤醒。Stop 模式 CPU power gating, mem 控制时钟及其他大多数模块时钟关闭。演示步骤:

1. 设置 timer 值并使能 counter;
2. 使能 PMU 中的中断唤醒使能位;
3. 软件保存现场, 执行 STOP 指令, 触发 power gating 状态机, power down;
4. Counter 计时结束触发中断, 触发 power gating 状态机;
5. 软件重启, 读取标志位, 恢复现场, 响应中断。

该测试程序需要修改 CPU 的端口信号, 增加 power gating 控制信号, 信号名可在配套的 upf 文件 ./tb/smartl.upf 中查询, 同时还需要设置 power gating 仿真软件 MVSIM 以及相关环境变

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

量（可参考 setup.csh 注释掉的部分），环境设置过程较为复杂，客户可参考相关内容但中天不为此过程作技术支持，只提供 upf 文件作为参考。

Case 4: ckcpu_lpmdbg.s 使用 debug 唤醒。使用 Wait 模式，仅停止 CPU 和 mem 控制时钟，其他模块不受影响。演示步骤：

1. 使能 PMU 中的 debug 唤醒使能位
2. 执行 wait 指令，进入低功耗模式
3. Force JTAG 信号 唤醒低功耗模式

5.2.7 动态变频演示

名称：Dynamic_freq_8xx.s，该测试程序用于演示 smartL 平台不同时钟的频率的设置和动态变频的过程。演示步骤：

1. 通过 store 指令，按 0-7 的顺序依次设置 clk_gen 的控制寄存器，并观测模块的时钟变化
2. 读取 clk_gen 模块寄存器看是否成功写入

5.2.8 安全演示

名称：Smpu.s，该测试程序主要演示了 SMPU 对特定地址区间的保护功能。CPU 对总线发起的请求带有安全属性位，SOC 上的鉴权模块根据该位进行访问鉴权。演示步骤：

1. 初始化配置，进入安全世界；
2. 配置 SMPU 寄存器的表项，设置保护区域并使能；
3. 切换至非可信世界，非可信世界试图篡改 SMPU 内容，被挡住返回 error
4. 非可信世界试图篡改 STIMER 内容，被挡住返回 error
5. 非可信世界试图篡改可信内存区域，被挡住返回 error；

5.2.9 世界切换

名称：tee_wsc_fuc.rs，安全示例程序用于验证，cpu 核代码是否能再安全态和非安全态下运行，具体验证如下：

1. 安全态配置 ebr，并注释说明
2. 非可信世界下读写可信、非可信的状态寄存器（T_EPSR, NT_EPSR, T_EPC, NT_EPC, 第三组控制寄存器），非可信的控制寄存器可读写，可信世界的控制寄存器不可写，

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

读为 0。

3. 世界切换指令调用
4. 可信世界读写可信世界和非可信世界控制寄存器，可读可写。

5.2.10 脉冲中断演示

名称: pulse_int_test.s, 该测试程序用于演示 smartL 平台通过脉冲触发中断和中断服务程序设置。用户配置 CPU 例化路径, 测试程序以一个 task 的形式交互, 客户的 tb 调研该 task 即可进行仿真。演示步骤:

1. 首先 reset cpu;
2. 等待汇编完成 psr 的初始化;
3. 之后后拉高 pulse_int 信号产生脉冲;
4. 检测 psr 中的 vector 号, 判断是否产生中断。

5.2.11 调试 (HAD)

名称: had_test.s, 主要向用户介绍使用 JTAG 进入调试模式的方式, 用于测试信号的连通性, 如果能够正常结束仿真则表明 JTAG 相关信号的连通性没有问题。

5.2.12 性能测试测试程序

提供性能测试集, 包括:

1) Dhrystone, 测试 CPU 整体性能, 该测试程序得出跑分需要一定的运行时间, 原测试程序里设置为 3000 遍, FPGA 上跑很快但是仿真非常慢, 若要前仿真需要将测试程序里 Number_of_Runs 改小。

2) Coremark, 测试 CPU 整体性能, 同理需要对遍数进行调整, 仿真环境只设置了两遍。

4) Memory copy, 内存拷贝性能测试

5) Memory set, 内存初始化性能测试

3) Linpack, 浮点性能 (803S), 原程序二维数组太大, 我们进行了适当缩小。

这些性能测试程序是在芯片上跑的标准程序, 前端仿真的话耗时很长, 为了适应仿真验证, 我们对这些程序遍数作了相应的修改, 同时也使用了虚拟 counter 进行统计。客户如果想要还原测试程序需要将遍数复原, 同时将文件头的 VCUNT_SIM 注释掉。

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

5.2.13 C 语言程序示例

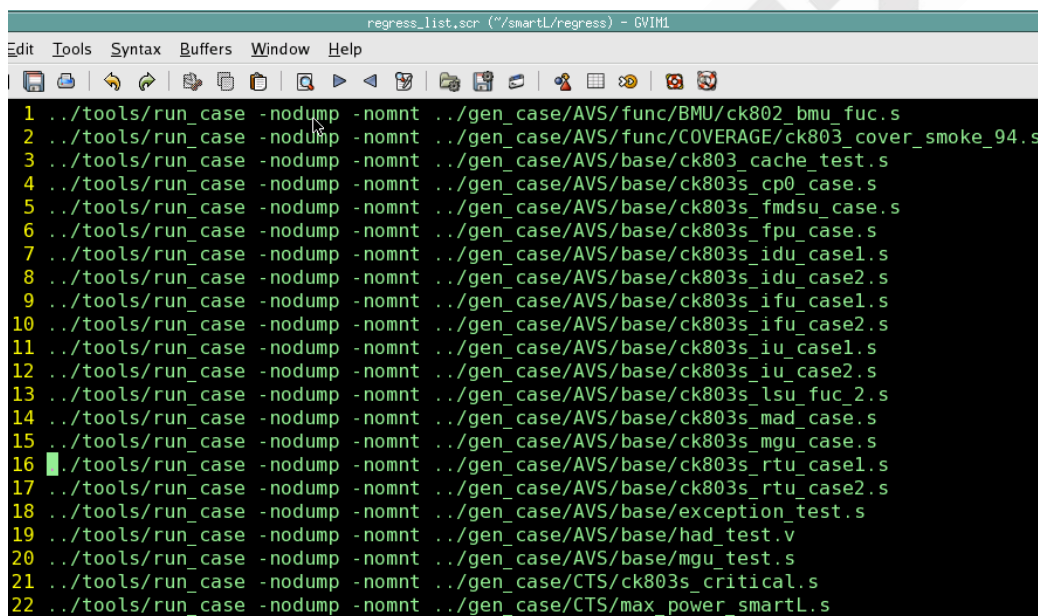
名称: hello_world_main.c, 为了方便客户快速熟悉 C-SKY 的仿真环境, 该测试程序提供了一个简单的 C 语言程序例子, 包括基本信息打印和 C 内嵌汇编等内容。

5.3 CTS 介绍

SmartL 里自带的了一些基础的测试程序, 用于演示 CPU 连接方法, 以及演示一些 CPU 基本功能使用, 但有些客户有更进一步的需求, 希望有一个更强并且易于移植到自己的 SOC 平台的测试程序集合, 针对这个需求中天开发了一套解决方案 C-SKY Test Suit (CTS), CTS 是中天为客户提供的一个强大的测试集合, 其中包含的测试程序覆盖了 CPU 集成连接验证, 功能/性能/功耗仿真以及最后的芯片测试。同时, CTS 也具有很强的可移植性, 客户简单输入 SOC 地址等信息后就可将 CTS 映射到自己的仿真平台进行仿真。更详细的内容见《CTS 用户手册》或咨询销售人员。

5.4 Regress

在 smartL 里增加了 regress 环境, 由两个脚本组成: 一个是 regress.pl 执行该文件会生出 regress_list.scr, 里面包含了所有能跑的测试程序, 如下图所示, 执行该文件即可进行所有测试程序的 regress。



```
1 ../tools/run_case -nodump -nomnt ../gen_case/AVS/func/BMU/ck802_bmu_fuc.s
2 ../tools/run_case -nodump -nomnt ../gen_case/AVS/func/COVERAGE/ck803_cover_smoke_94.s
3 ../tools/run_case -nodump -nomnt ../gen_case/AVS/base/ck803_cache_test.s
4 ../tools/run_case -nodump -nomnt ../gen_case/AVS/base/ck803s_cp0_case.s
5 ../tools/run_case -nodump -nomnt ../gen_case/AVS/base/ck803s_fmdsu_case.s
6 ../tools/run_case -nodump -nomnt ../gen_case/AVS/base/ck803s_fpu_case.s
7 ../tools/run_case -nodump -nomnt ../gen_case/AVS/base/ck803s_idu_case1.s
8 ../tools/run_case -nodump -nomnt ../gen_case/AVS/base/ck803s_idu_case2.s
9 ../tools/run_case -nodump -nomnt ../gen_case/AVS/base/ck803s_ifu_case1.s
10 ../tools/run_case -nodump -nomnt ../gen_case/AVS/base/ck803s_ifu_case2.s
11 ../tools/run_case -nodump -nomnt ../gen_case/AVS/base/ck803s_iu_case1.s
12 ../tools/run_case -nodump -nomnt ../gen_case/AVS/base/ck803s_iu_case2.s
13 ../tools/run_case -nodump -nomnt ../gen_case/AVS/base/ck803s_lsu_fuc_2.s
14 ../tools/run_case -nodump -nomnt ../gen_case/AVS/base/ck803s_mad_case.s
15 ../tools/run_case -nodump -nomnt ../gen_case/AVS/base/ck803s_mgu_case.s
16 ../tools/run_case -nodump -nomnt ../gen_case/AVS/base/ck803s_rtu_case1.s
17 ../tools/run_case -nodump -nomnt ../gen_case/AVS/base/ck803s_rtu_case2.s
18 ../tools/run_case -nodump -nomnt ../gen_case/AVS/base/exception_test.s
19 ../tools/run_case -nodump -nomnt ../gen_case/AVS/base/had_test.v
20 ../tools/run_case -nodump -nomnt ../gen_case/AVS/base/mgu_test.s
21 ../tools/run_case -nodump -nomnt ../gen_case/CTS/ck803s_critical.s
22 ../tools/run_case -nodump -nomnt ../gen_case/CTS/max_power_smartL.s
```

图表 5-1 regress list

Regress 跑完后执行另一个脚本 report_gen.pl, 则会生出 regress 的结果, 如下图所示:

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).



regress_report ("~/smartL/regress) - GVI

| 1 | 2 | 3 | 4 |
|---------|----------------------|-----------|-------|
| Block | Pattern | Result | |
| 0 | ck802_bmu_fuc | PASS | |
| 1 | ck803_cover_smoke_94 | =>NOT RUN | |
| 2 | ck803_cache_test | PASS | |
| 3 | ck803s_cp0_case | PASS | |
| 4 | ck803s_fmdu_case | PASS | |
| 5 | ck803s_fpu_case | PASS | |
| 6 | ck803s_idu_case1 | PASS | |
| 7 | ck803s_idu_case2 | PASS | |
| 8 | ck803s_ifu_case1 | PASS | |
| 9 | ck803s_ifu_case2 | PASS | |
| 10 | ck803s_iu_case1 | PASS | |
| 11 | ck803s_iu_case2 | PASS | |
| 12 | ck803s_lsu_fuc_2 | PASS | |
| 13 | ck803s_mad_case | PASS | |
| 14 | ck803s_mgu_case | PASS | |
| 15 | ck803s_rtu_case1 | PASS | |
| 16 | ck803s_rtu_case2 | PASS | |
| 17 | exception_test | PASS | |
| 18 | mgu_test | PASS | |
| 19 | ck803s_critical | PASS | |
| 20 | max_power_smartL | =>NOT RUN | |
| 21 | connect_test | =>NOT RUN | |
| 22 | pulse_int_test | =>NOT RUN | |
| 23 | ckcpu_wait | PASS | |
| 24 | power_gating | =>FAIL | |
| 25 | smpu | PASS | |
| 26 | cover_int_spec | PASS | |
| 27 | mem_set | PASS | |
| Not run | Pass | Fail | Total |

图表 5-2 regress 结果

6 FPGA 流程

6.1 采用 Xilinx ISE 器件

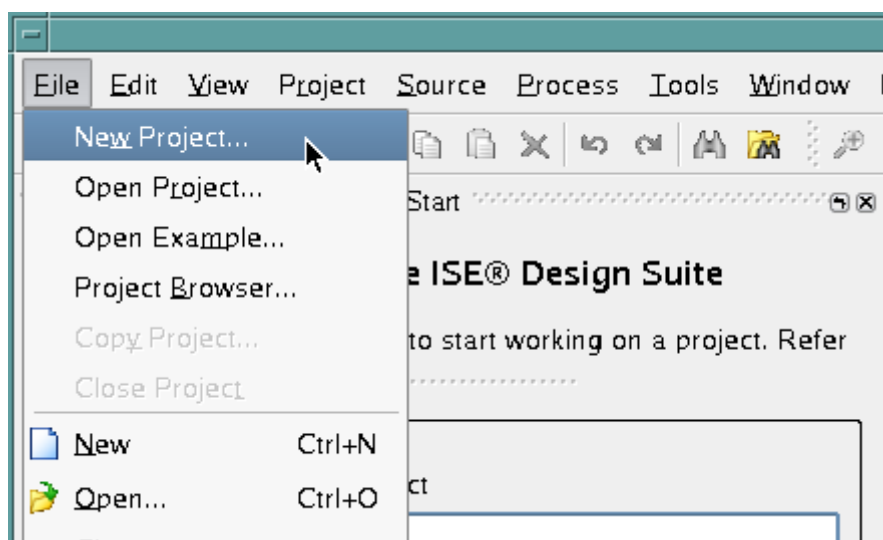
一、综合条件

1. 您已经有我公司递交的 SmartL 平台打包文件，其中包括：SoC 平台中的 verilog 源码（soc.v, ck803s.v, ahb.v, apb.v mem_ctrl.v）您所申请的特定配置的特性型号的 CPU 核的网表文件（例如：ck803s.edf）

2. 您的机器可以正常工作，并且已经有安装过 Xilinx ISE 软件，并且可以正常使用

二、建立 ISE Project

1. 打开 ISE 软件后，从菜单栏选择[File] -> [New Project...]以新建一个项目，如下图所示：

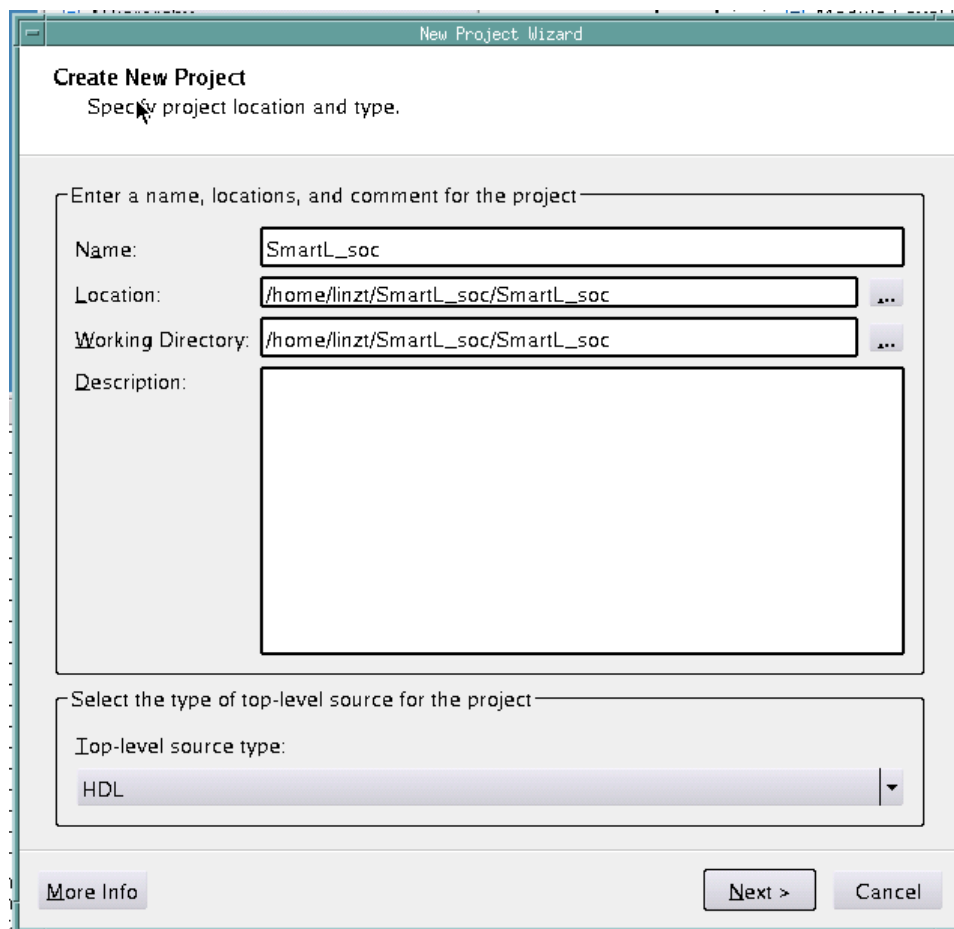


图表 6-1 新建 Project_1

2. 在弹出的[New Project Wizard] 框框中，填上项目名称（如：SmartL_soc）与路径，并且在选择[Top-level source type] 为“HDL”，然后点击[Next >]按钮，如下图所示：

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).



图表 6-2 新建 Project_2

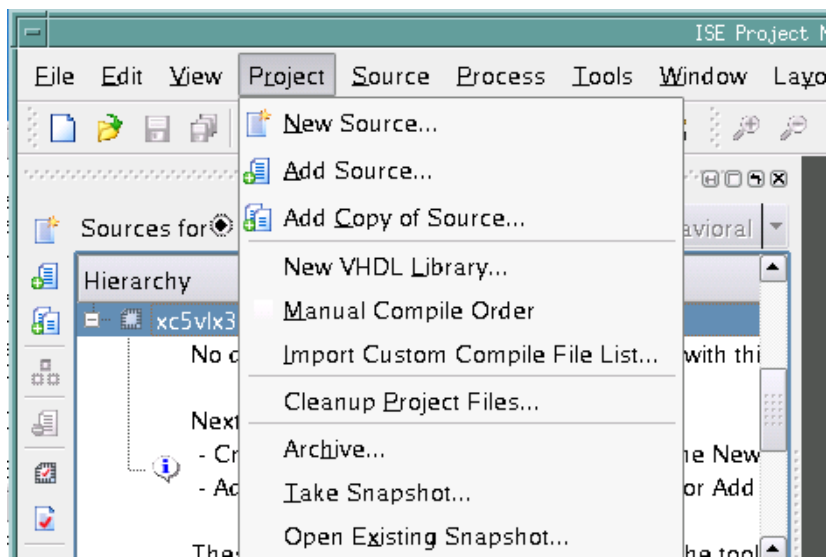
3. [Device Properties]选择 FPGA 器件型号，这一步基本没有其他选择余地，我们提供的 CPU 核网表也是根据您给我们的 FPGA 型号产生的；

4. [Create New Source]创建新的设计文件，这一步直接跳过，所有需要的文件都已经在 SmartL 平台中；

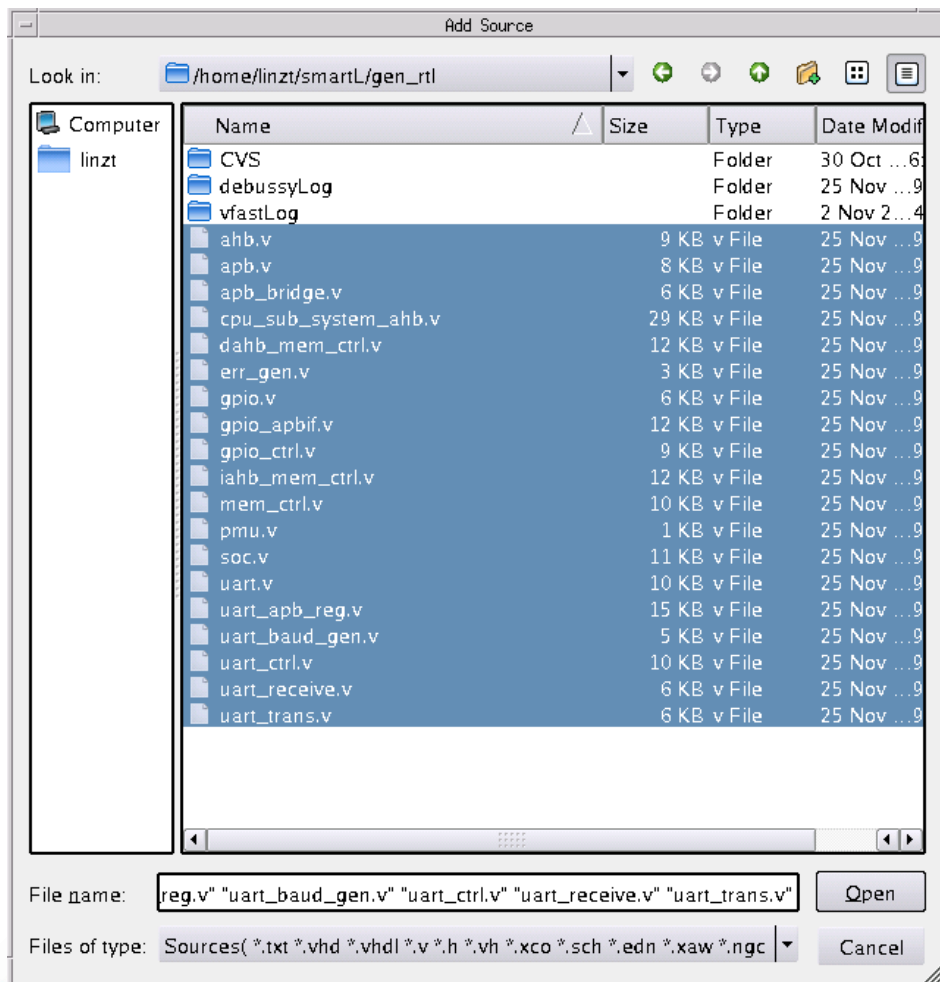
5. [Add Existing Sources]添加设计文件，这里需要把综合所需要的所有 verilog 源文件添加进来，同时还需要添加综合和布局布线的约束文件（.ucf）设计约束文件中定义了 SmartL SoC 的外部输入输出端口与 FPGA 引脚的对应关系，您可能需要参考 SmartL 中已有的.ucf 文件编写适合您自己的硬件平台的.ucf 文件，如下图所示。这一步可以直接跳过，然后在项目建立好之后再添加文件。

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).



图表 6-3 添加设计文件_1



图表 6-4 添加设计文件_2

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

选择好文件之后，点击[Open]，在出现的框框中，建议把文件后面的[Copy to Project]选项去掉，然后点击[Next]；

ucf 文件类似于如下格式：

```
#=====UART=====
NET "uart0_sout" LOC = "A9";    #TXD,Output
NET "uart0_sin"  LOC = "B9";    #RXD,Input

#=====JTAG=====
NET "jclk"       LOC = "AF7";    #TCLK
NET "jclk"       CLOCK_DEDICATED_ROUTE = FALSE;
NET "jtg_tms"    LOC = "AE9";    #TMS
NET "jtg_tdi"    LOC = "AF9";    #TDI
NET "jtg_tdo"    LOC = "AF8";    #TDO
NET "jrst_b"     LOC = "AE7";    #TRST
NET "jrst_b"     PULLUP;    #TRST signal MUST be PULLUP

#=====CLK=====
#NET "pll_core_cpucclk" LOC = "B13";    # OSC2, 25MHz
NET "pll_core_cpucclk" LOC = "C15";    # OSC1

#=====Reset=====
NET "pad_cpu_rst_b" LOC = "AF18";    #FPGA_RST1_B, K2

#Begin clock constraints

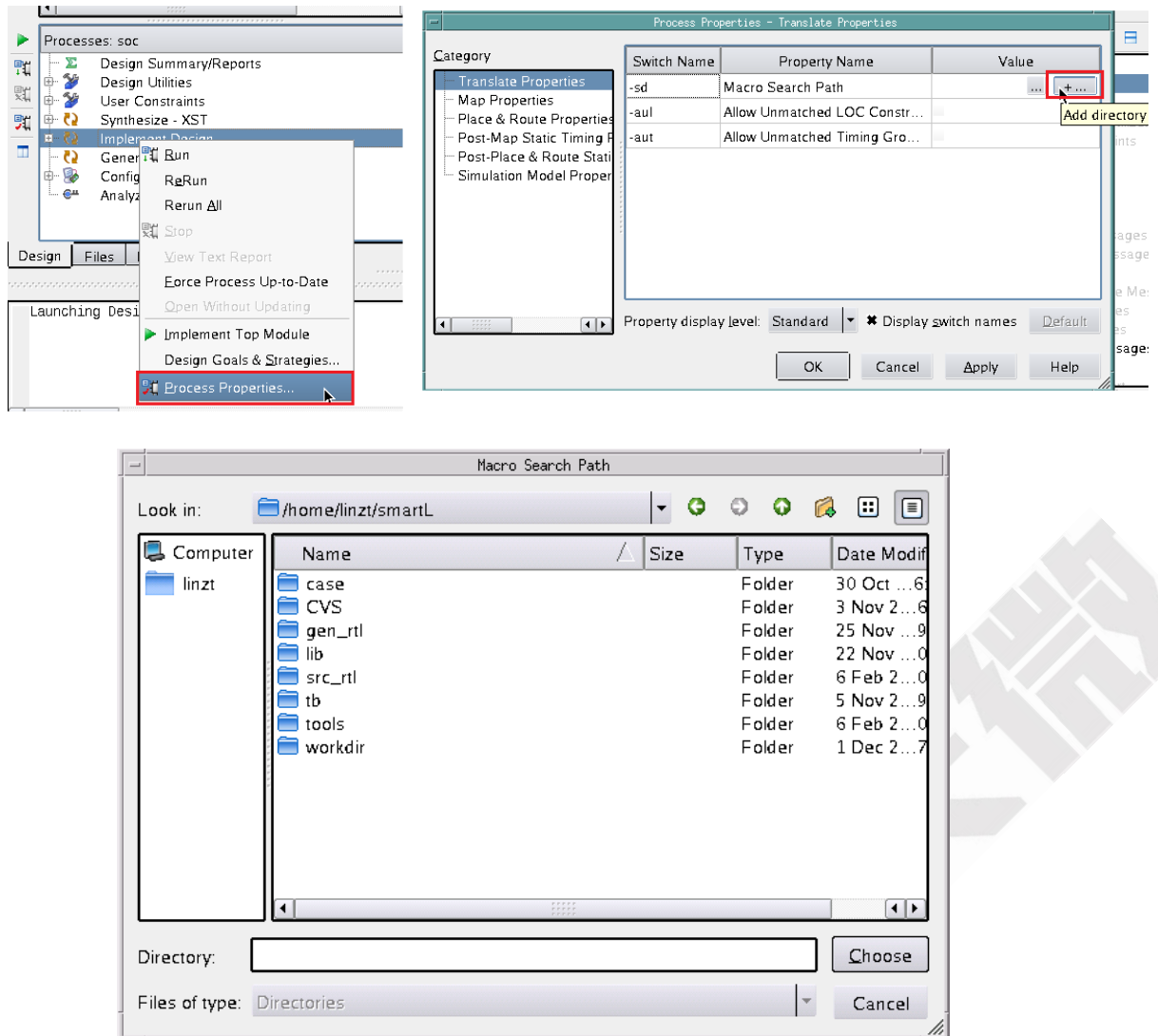
NET "jclk" TNM_NET = "jclk";
TIMESPEC "TS_jclk" = PERIOD "jclk" 16.667 ns HIGH 50.00%;

NET "pll_core_cpucclk" TNM_NET = "pll_core_cpucclk";
TIMESPEC "TS_pll_core_cpucclk" = PERIOD "pll_core_cpucclk" 20.000 ns HIGH 50.00%;
#End clock constraints
```

图表 6-5 UCF 文件示例

6. [Project Summary] 窗口给出关于此项目的一些概况，点击[Finish]完成项目的建立过程。

【说明】在第 5 步中，添加源文件时，需要把 ck803s.v 这样的文件添加进去，ck803s.v 文件其实是一个空壳，文件中只有模块的端口声明。而真实的模块实现是在 ck803.edf 网表文件中，这个网表文件不需要作为源文件添加进项目，只需要将其放在项目目录下，或者放在其他任何地方，然后通过如下图所示方法告诉 ISE，网表文件所在的位置。在 ISE 对 verilog 源文件综合过程中，将 ck803 模块作为 black box，而在真正实现的时候使用网表文件替换这个 black box。



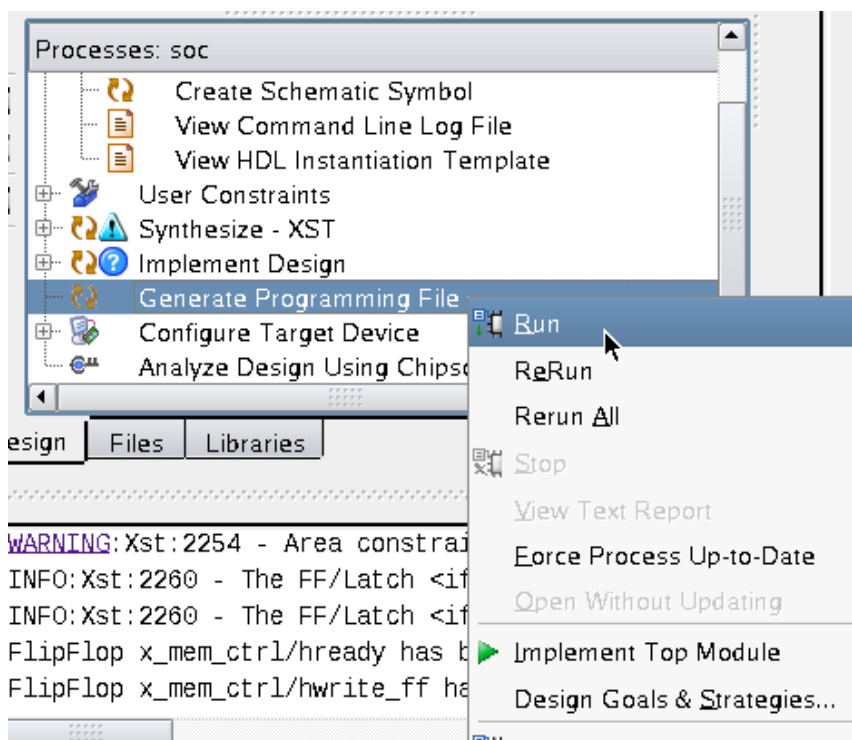
图表 6-6 使用网表文件综合

三、综合与布局布线

1. 双击[Generate Programming File], ISE 将会启动综合进程, Synthesis -> Implement Design -> Generate Programming File 逐步走下来之后, 最终会产生.bit 文件。

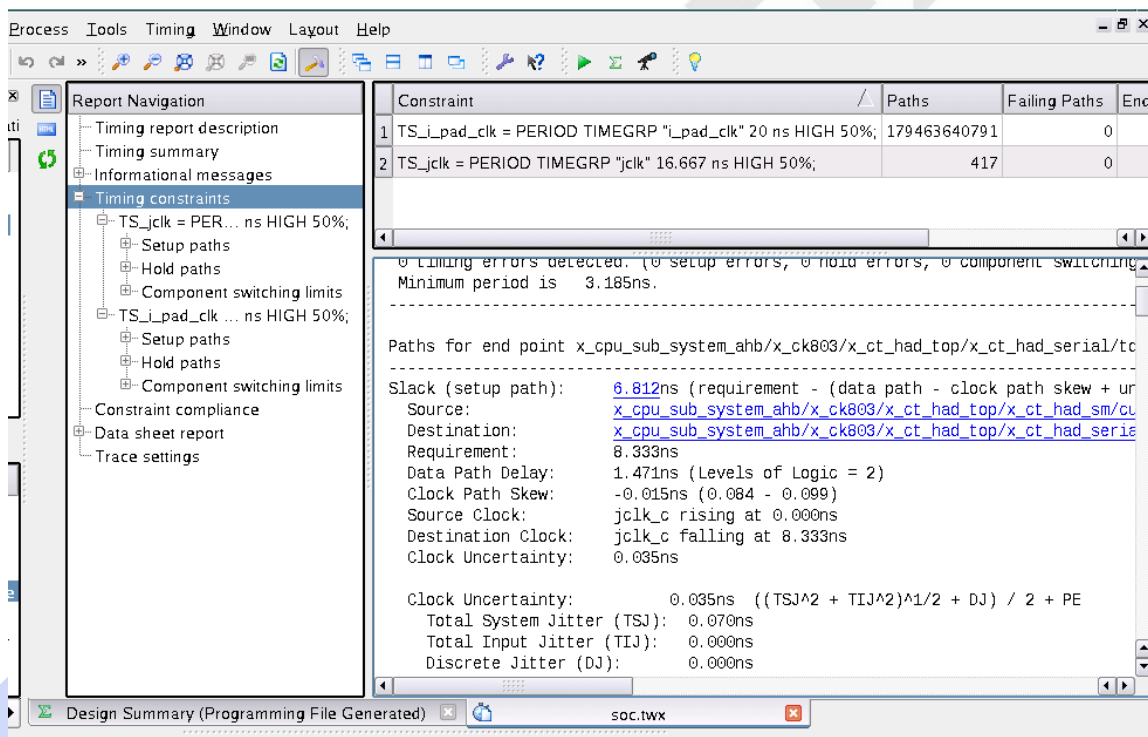
C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).



图表 6-7 综合与布局布线

2. 综合结果检查。如果 ISE 能够正常产生 .bit 文件，那么基本上可以进行下一步的调试了，但最好还是检查一下综合结果的报告，点击菜单[Project] -> [Design Summary/Reports]。最需要注意的是 Static Timing 里面的报告，以确定系统可以稳定运行的最大频率。



C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

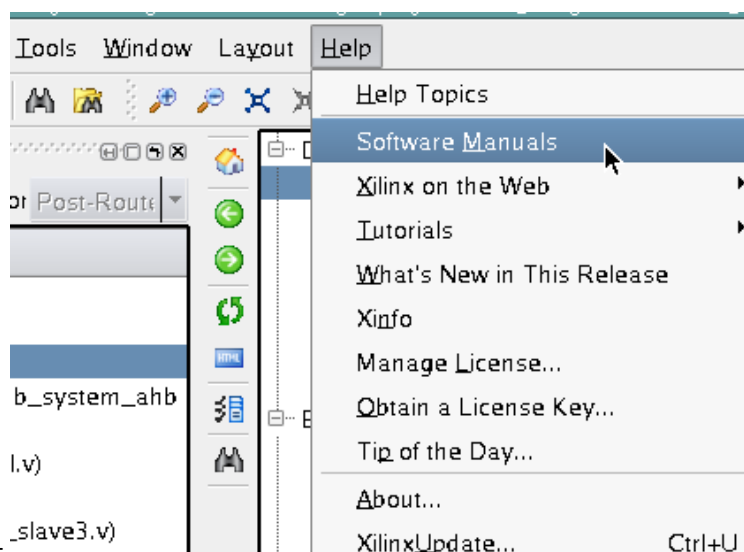
图表 6-8 综合结果检查

怎么启动 ISE 软件？

```
#Xilinx ISE 11
source /tools/fpga/xilinx_11/ChipScope/settings64.csh
source /tools/fpga/xilinx_11/ISE/settings64.csh
```

启动命令：./tools/fpga/xilinx_11/ISE/bin/lin64/ise &

哪里有 ISE 使用教程？



图表 6-9 ISE 使用帮助菜单

使用 ISE 综合 CPU 网表和 SoC 源代码的 SmartL 平台存在潜在的问题：FPGA 的内部逻辑资源不足以实现整个 SmartL 设计。

因为：SmartL 平台中使用的 mem_ctrl 是基于对 Syncore_ram 的例化实现的，而 Syncore_ram 是利用 synplify 的 SYNCore 工具产生的。在利用 ISE 对此 memory 进行综合时，ISE 不能够将此 memory 映射到 FPGA 内部的 block ram 逻辑资源，而是使用普通的 LUT 实现了一个 distributed ram，而这种形式的 memory 会消耗掉 FPGA 中大量的逻辑资源，以至于整个 FPGA 都已经实现不了了。但是，Syncore_ram 是利用 synplify 工具产生的，当利用 synplify 综合此 memory 时，synplify 能够认识是自家人，并将 Syncore_ram 映射到 Xilinx FPGA 的 ram 专用的逻辑资源而不是使用 FPGA 的 LUT。

所以，在我们提供给客户 CPU 核网表和 SOC 其他部分源代码之后，如果客户需要综合，那只有两种选择：

C-Sky Confidential

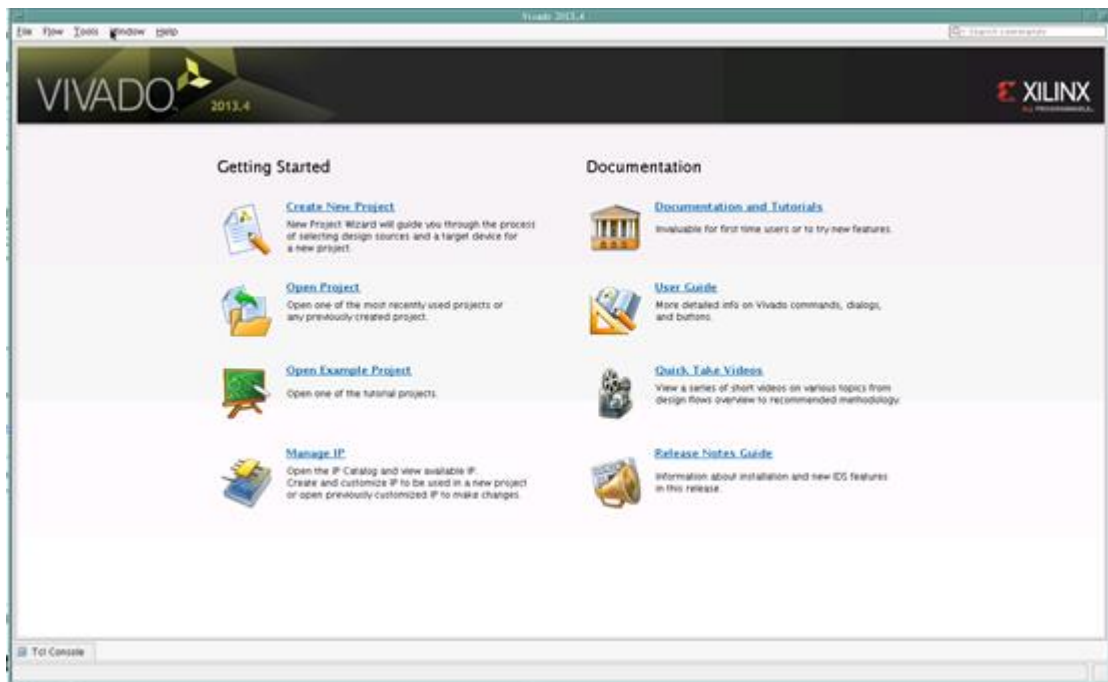
The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

1. 将整个 SoC 源代码和 CPU 网表一起扔给 synplify 综合，得到一个新的网表；然后使用 ISE 使用该新的网表进行布局布线产生 bit 文件；

2. 修改 mem_ctrl 代码，按照 Xilinx 规定的特性形式重新实现 memory，然后将整个 SoC 源代码和 CPU 核网表一起扔给 ISE 进行综合，这样 ISE 就能够识别 mem_ctrl 中的 memory，并将此映射到 block memory 逻辑资源中。

6.2 采用 Xilinx Vivado 器件

1. 执行“vivado &”启动 vivado。注意 vivado 会在你执行这条命令的目录下生成一些 log 信息，所以最好在新建的项目目录里打开 vivado：

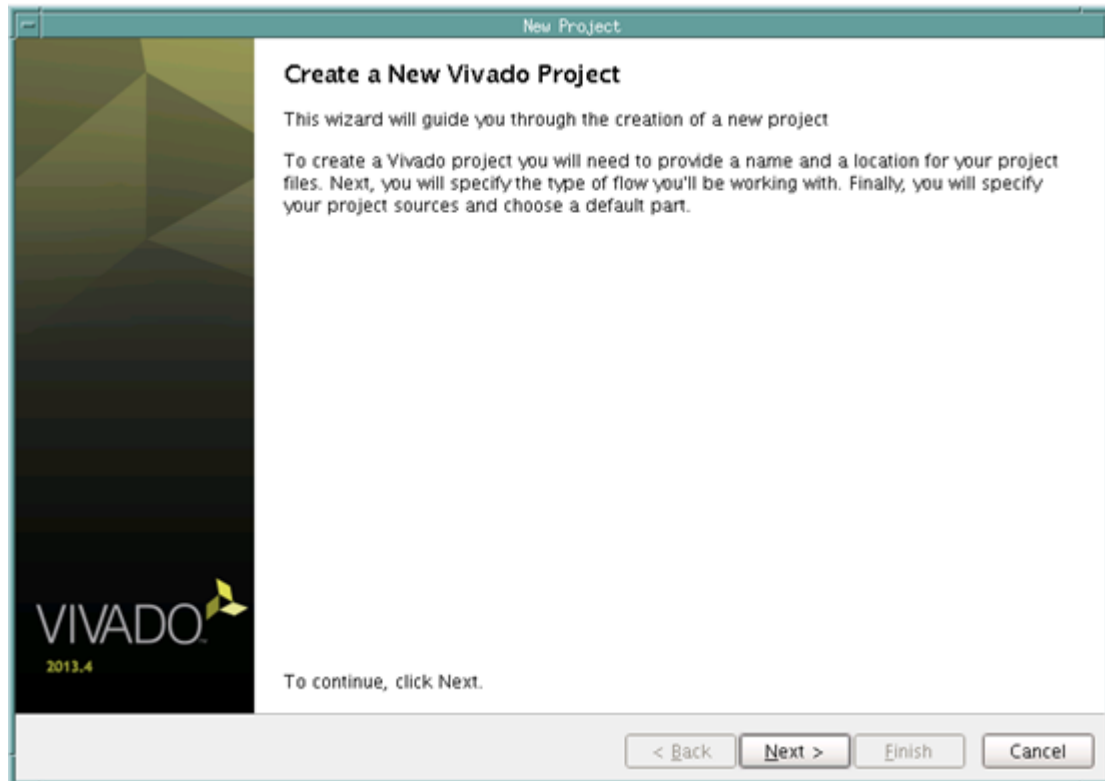


图表 6-10 vivado 界面

2. 点击 Create New Project 建立新的项目，在弹出的对话框中点 Next:

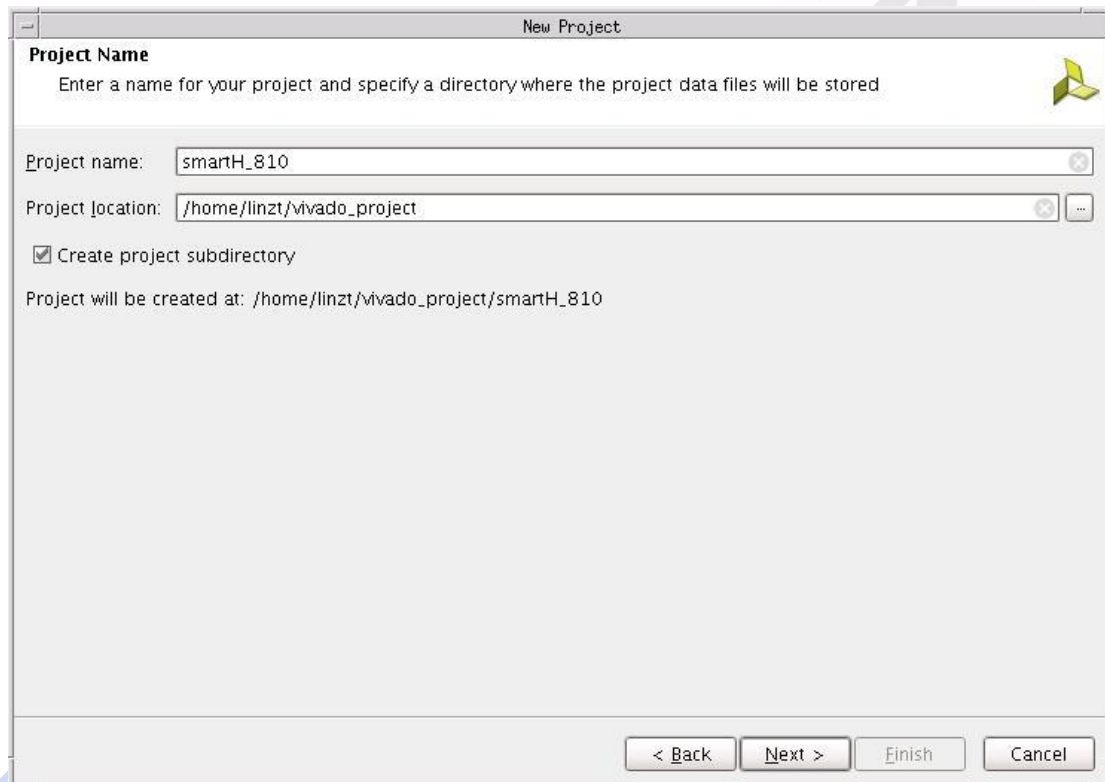
C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).



图表 6-11 vivado 新建项目

3. 输入项目名称，然后再点击 Next:

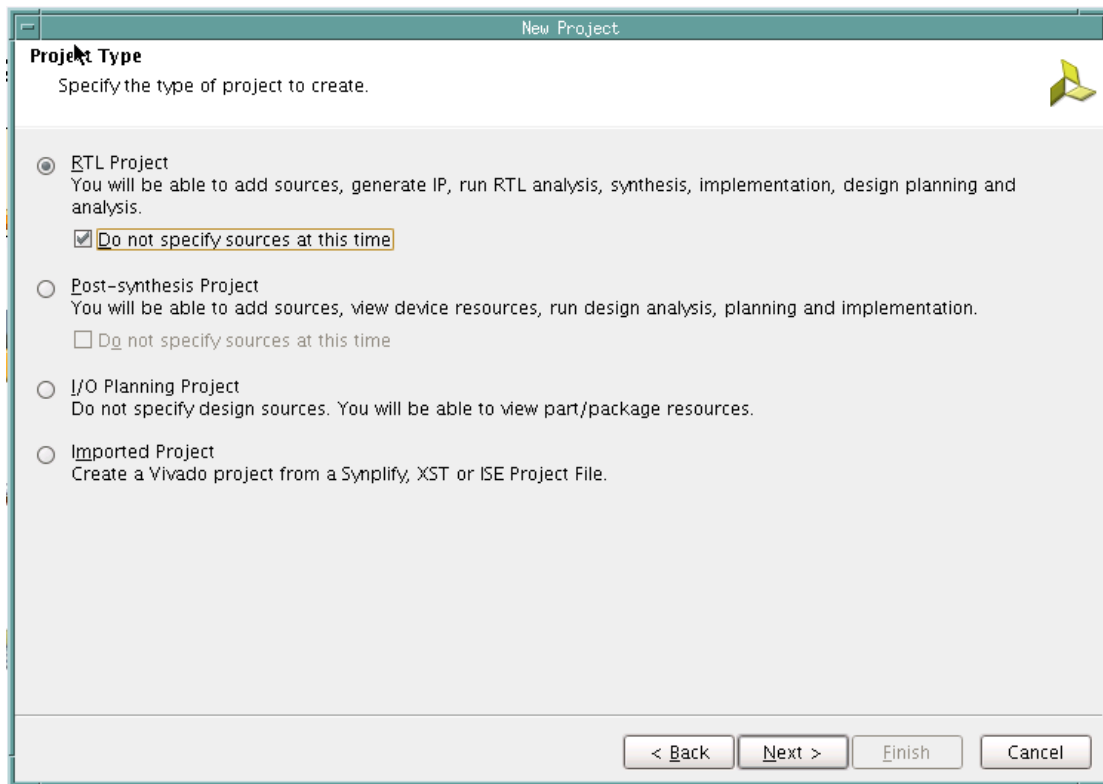


图表 6-12 项目名称与路径

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

4. 选择项目类型，因为我们要从 RTL 代码开始综合，因此选择 RTL Project。下面的 Do not specify source at this time 的勾也可以打上。如果不打上，下一步会进入添加 source file:

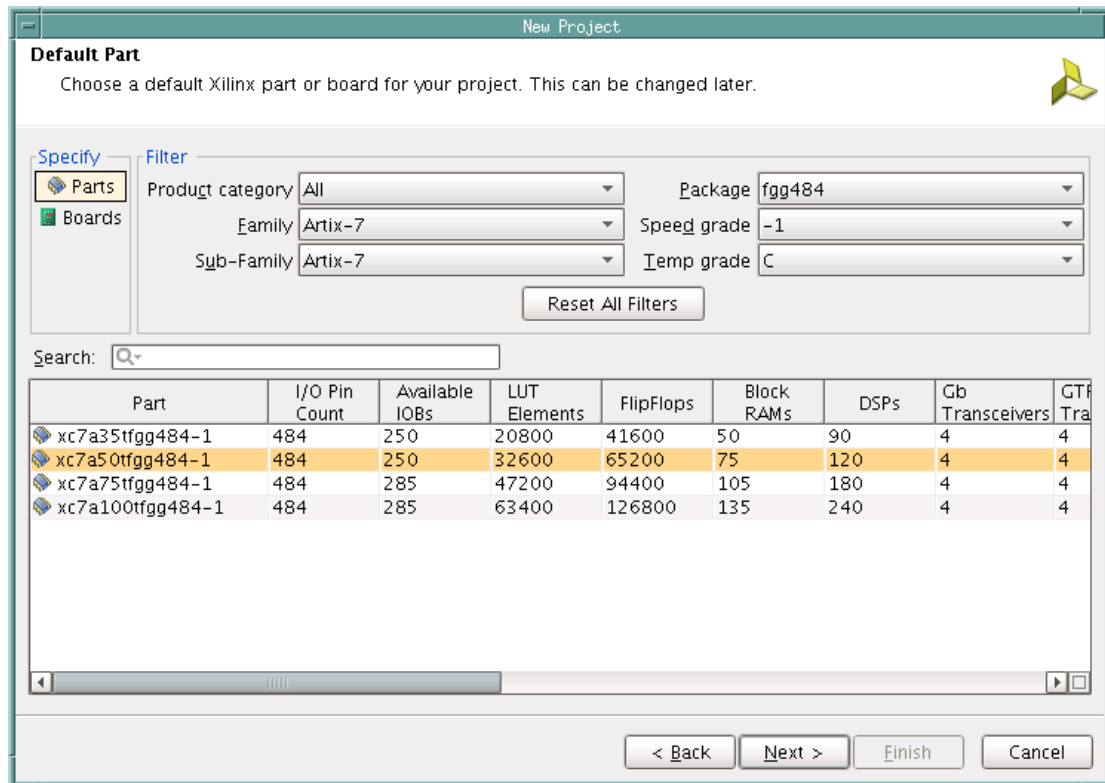


图表 6-13 项目类型

5. 选择板子的型号，然后点击 Next。我们目前 Artix-7 板子的具体型号如下：

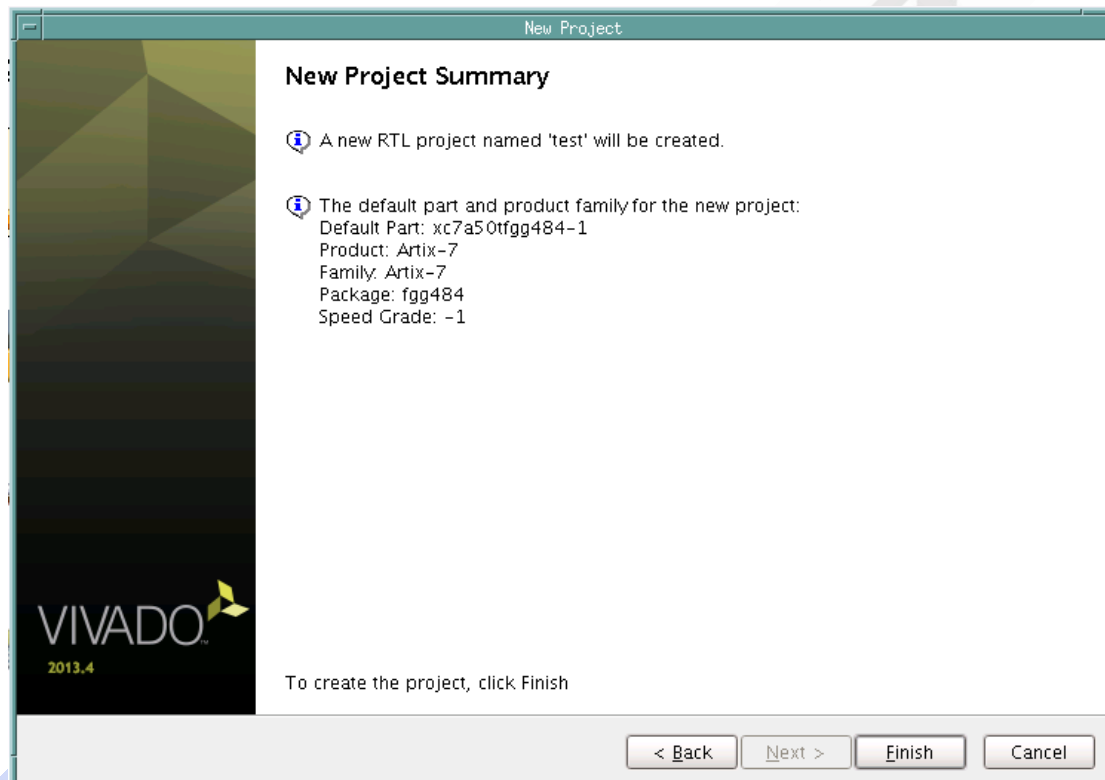
C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).



图表 6-14 选择 FPGA 芯片型号

6. 再次确认一下板子型号有没有选对，然后点击 Finish 完成项目创建工作：

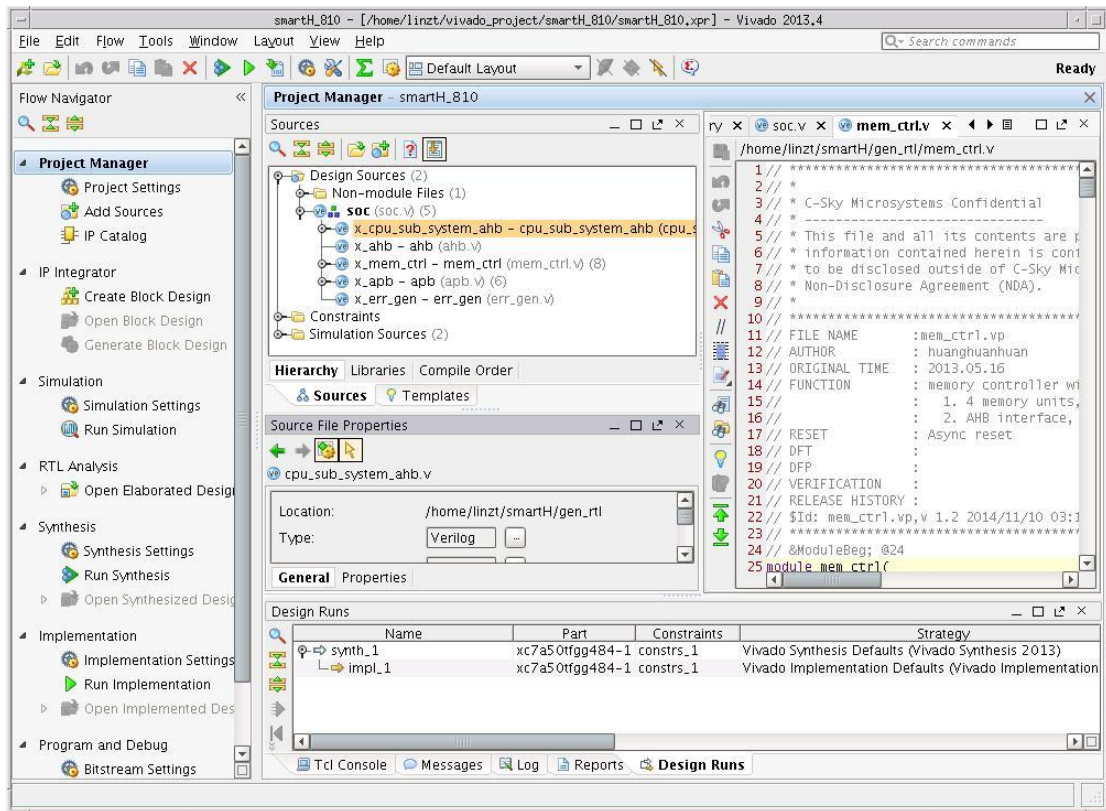


图表 6-15 确认 FPGA 型号

C-Sky Confidential

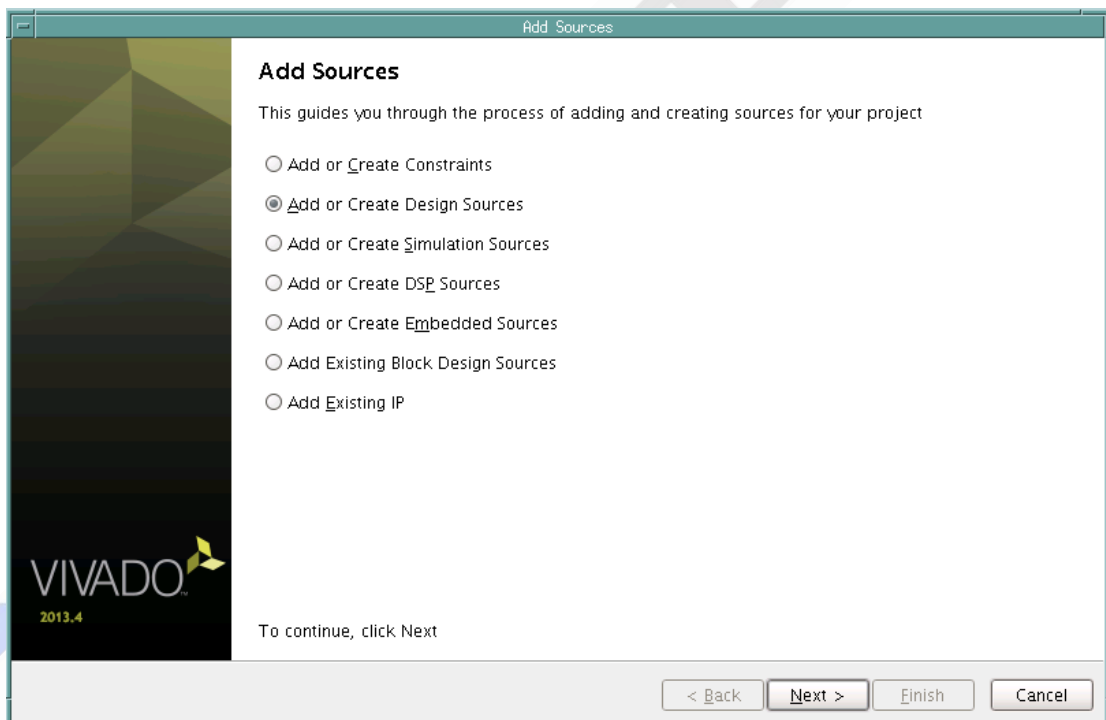
The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

7. 右键 Design Sources 或者使用快捷键 Alt+A 开始添加 Source files:



图表 6-16 添加项目源文件

8. 选择 Add or Create Design Sources, 再点击 Next:



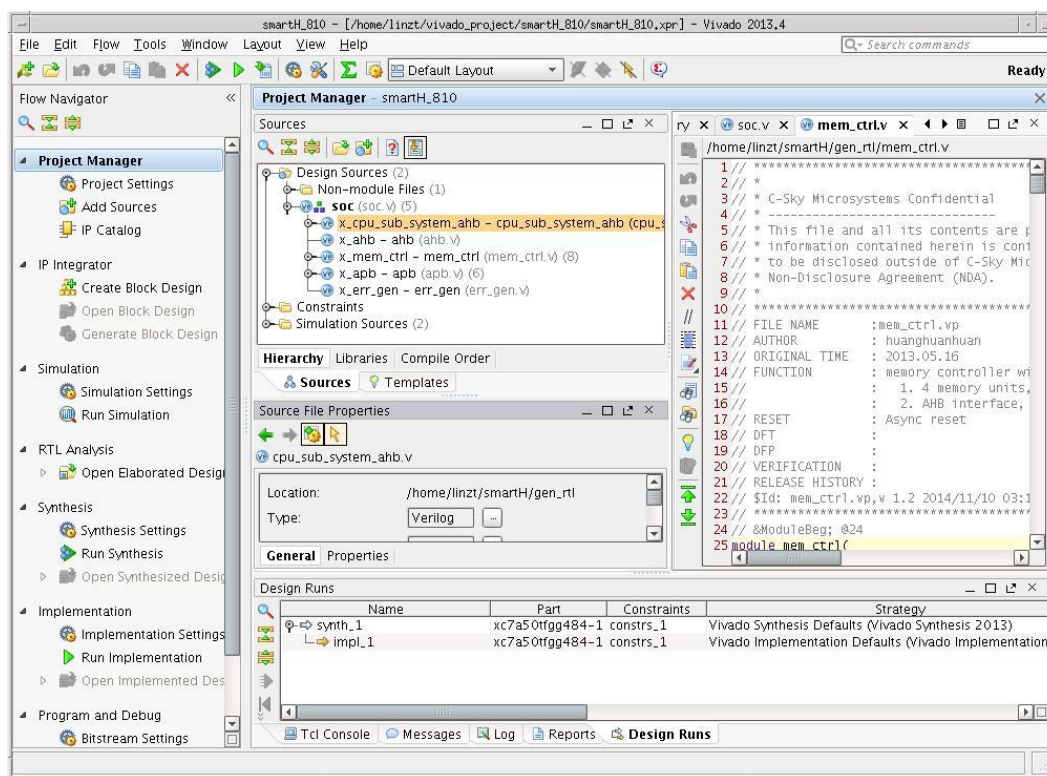
C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

图表 6-17 添加源文件界面

点击 **Add Files** 可以一个个添加源文件，点击 **Add Directories** 可以按目录添加源文件。在这里加入所有需要的.v 文件和.h 文件。并加入 CPU 核的网表文件（例如 ck803s.edf）及其对应的 ck803s.v 文件， ck803s.v 文件其实是一个空壳。完成后点击 **Finish**。

9. 如果正确添加了源文件，在这个 Sources 窗口中，Vivado 会自动加粗识别出来的 top module:

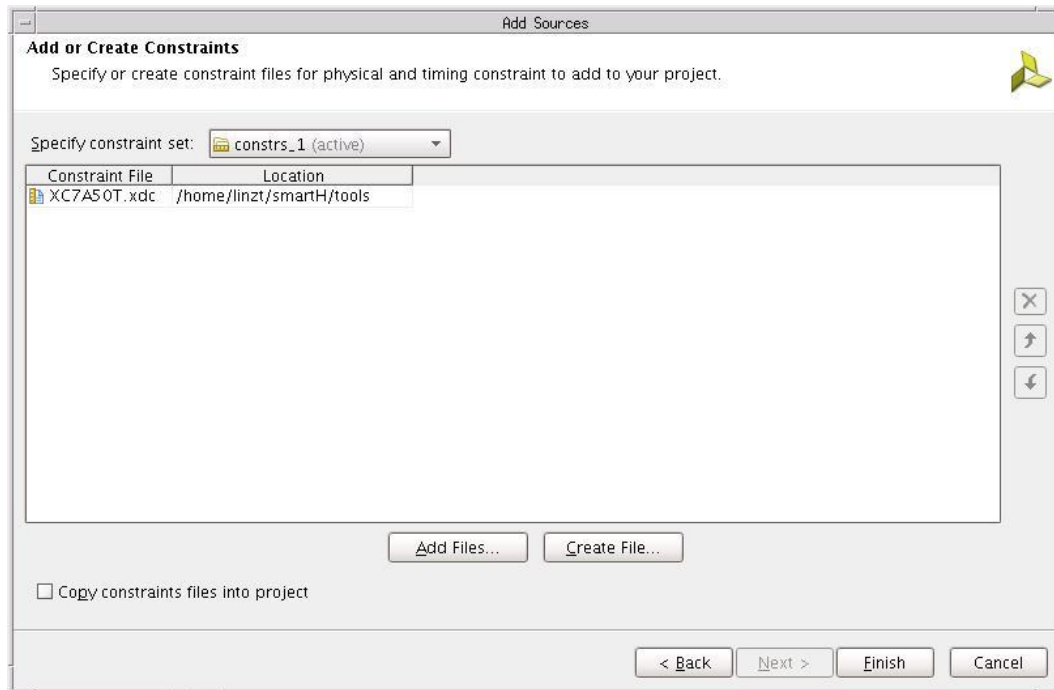


图表 6-18 项目源文件层次

10. 右键 **Constrains**, 点击 **Add Sources**, 在接下来弹出的窗口中选择 **Add or Create Constrains** 后再点击 **Finish** 来添加约束文件:

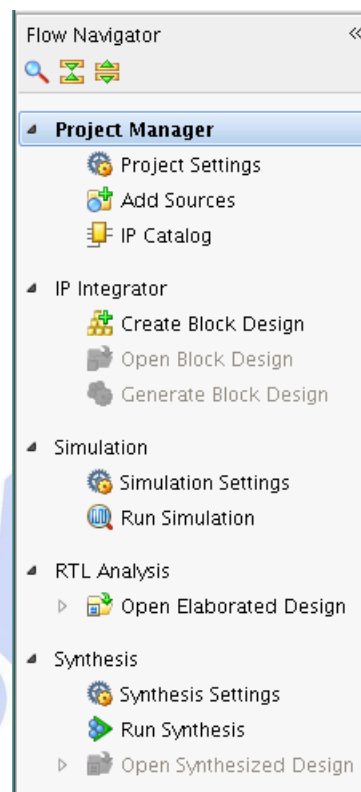
C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).



图表 6-19 添加约束文件

11. 完成后，点击 Run Synthesis，即可开始综合并生成网表文件：

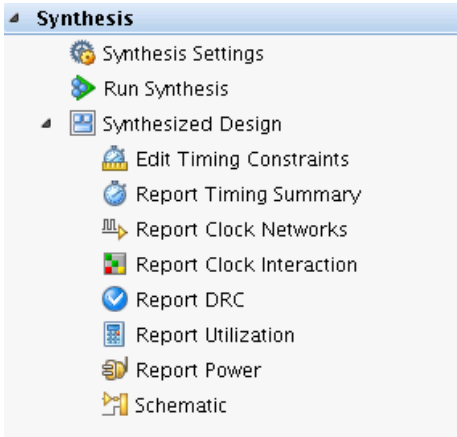


图表 6-20 综合项目代码

12. 综合完成后，可以查看一些 report:

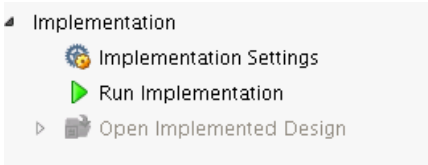
C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).



图表 6-21 查看综合结果

13. 点 Run Implementation 开始布局布线:



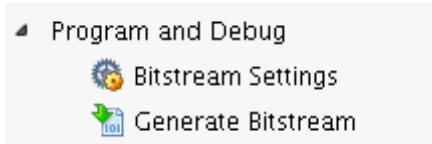
图表 6-22 布局布线

14. 完成之后，在 [project_name].runs/impl_1/ 这个目录下会生成 DCP 文件，即 [top_module_name]_routed.dcp 这个文件，该文件既可用于生成 BIT 文件。
15. 完成 Implementation 之后，可以查看 Implemented Design，在这里可以看到板上实际资源的使用：

| 1.1 On-Chip Components | | | | | |
|------------------------|-----------|-------|-----------|-----------------|--|
| ----- | | | | | |
| On-Chip | Power (W) | Used | Available | Utilization (%) | |
| Clocks | 0.010 | 3 | --- | --- | |
| Slice Logic | 0.010 | 19534 | --- | --- | |
| LUT as Logic | 0.009 | 11316 | 32600 | 34.71 | |
| CARRY4 | <0.001 | 311 | 8150 | 3.81 | |
| Register | <0.001 | 5537 | 65200 | 8.49 | |
| F7/F8 Muxes | <0.001 | 654 | 32600 | 2.00 | |
| BUFG | <0.001 | 1 | 32 | 3.12 | |
| Others | 0.000 | 250 | --- | --- | |
| Signals | 0.013 | 16470 | --- | --- | |
| Block RAM | 0.043 | 64 | 75 | 85.33 | |
| DSPs | <0.001 | 8 | 120 | 6.66 | |
| I/O | <0.001 | 17 | 250 | 6.80 | |
| Static Power | 0.073 | | | | |
| Total | 0.149 | | | | |

图表 6-23 Implemented Report

16. 点击 Generate Bitstream 生成.bit 文件:



在提供给客户 CPU 核网表和 SOC 其他部分源代码之后，如果客户需要使用 vivado 工具综合，那只有两种选择：

1. 将整个 SoC 源代码和 CPU 网表一起使用 synplify 综合，得到一个新的网表；然后使用 vivado 对该新的网表进行布局布线产生 bit 文件；

2. 修改 mem_ctrl 代码，按照 Xilinx 规定的特性形式重新实现 memory，然后将整个 SoC 源代码和 CPU 核网表一起使用 vivado 进行综合，这样 vivado 就能够识别 mem_ctrl 中的 memory，并将此映射到 block memory 逻辑资源中；

6.3 采用 Altera 器件

略。

6.4 FPGA 占用资源估计

下表列出了典型配置下，SmartL 占用 FPGA 资源的情况。

| FPGA 型号 | 资源类型 | CK801 | CK802 | CK803S |
|--------------|------|---------|---------|---------|
| Artix 7 100T | IO | 2% | 2% | - |
| | RAM | 96(135) | 96(135) | - |
| | LUTs | 8% | 11% | - |
| Artix 7 200T | - | - | - | 2% |
| | - | - | - | 133/365 |
| | - | - | - | 12% |

图表 6-24 FPGA 资源占用情况

CK801 配置:

| cpu | cfig | IMEM | DMEM | SMEM | CACHE | 板子 |
|-----|------|------|------|------|-------|------|
| 801 | | | | 256K | | 100T |

CK802 配置:

| cpu | cfig | IMEM | DMEM | SMEM | CACHE | 板子 |
|-----|------|------|------|------|-------|----|
| | | | | | | |

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

| | | | | | |
|-----|------|--|------|--|------|
| 802 | 256K | | 128K | | 100T |
|-----|------|--|------|--|------|

CK803 配置:

| cpu \ cfig | IMEM | DMEM | SMEM | CACHE | 板子 |
|------------|------|------|------|-------|------|
| 803s | 256K | 128K | | 8K | 200T |



C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

7 附件一：SDC

为了方便客户参考，SDC 文件也放在了 smartL 里，具体路径是 smartL/synthesis/CK80X_func.sdc，该 SDC 主要包含以下几个点，这里简单说明下，具体可以参考脚本里的注释。

7.1 Clock 设置

CPU 物理上有两个 clock，一个是 CPU 工作的 clock，一个 JTAG 工作的频率，一般要求 JTAG 频率低于 CPU 的二分之一，如下图所示：

```
create_clock [get_ports pad_had_jtg_tclk] -name JTG_CLOCK -period $DC_JTG_period
create_clock [get_ports pll_core_cpucclk] -name CORE_CPUCLK -period $DC_period
create_clock -name V_SYS_CLK -period $DC_period
```

V_SYS_CLK 是为 input/output port 约束而设置的。

7.2 False Path 设置

False path 主要包含两各部分，一个是两个时钟域间的同步路径，而是测试相关型号，具体如下图所列：

```
### Set falth path ###
# CPUCLK and JTG_CLOCK are asynchronou
set_false_path -from [get_clocks CORE_CPUCLK] -to [get_clocks JTG_CLOCK]
set_false_path -from [get_clocks JTG_CLOCK] -to [get_clocks CORE_CPUCLK]
set_false_path -from [get_ports pad_yy_test_mode]
```

7.3 加紧 ICG path 约束

前端综合需要对 ICG path 加紧约束，一般设置为 0.3 倍时钟频率，如下图所示：

```
### Tighten constraints on ICG path ###
foreach_in_collection cell [filter [get_cell -hier *] "ref_name =~ ${DC_clock_gate_p
refix}*"] {
    set_clock_gating_check -setup [expr $DC_period*0.3] [concat [get_object_name $ce
ll]/${DC_clock_gate_enable_pin}]
}
```

function 的综合（没插 DFT），可以将测试相关信号也设为 false path。

```
set_false_path -from [get_ports pad_yy_scan_enable]
set_false_path -from [get_ports pad_yy_test_mode]
set_false_path -from [get_ports *_si_*]
set_false_path -to [get_ports *_so_*]
```

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

7.4 Input/output 设置

7.4.1 Delay

如果是 CPU 和 SOC 一起综合，CPU port 上的 delay 并不需要特别设置，如果是单独综合的话，综合时需要根据 SOC 和 CPU 对接信号的时序来进行设置，以达到最佳效果，以下给出的是一个普通建议值，如果客户需要对某些信号精细调整，我们提供每根信号的 delay 门级数，可以在集成手册上查到。

```
### Set input & output path constraints ###  
set_input_delay -max [expr $DC_period*0.4] -clock V_SYS_CLK [filter [all_inputs] "full_name !~ *clk && full_name !~ *clock"]  
set_output_delay -max [expr $DC_period*0.4] -clock V_SYS_CLK [all_outputs]  
set_max_delay [expr $DC_period*1.2] -from [all_inputs] -to [all_outputs]
```

7.4.2 其他

function 综合可做一下设置：

```
# For func mode, test_mode = 0;  
set_case_analysis 0 [get_ports pad_yy_test_mode] ;
```

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).