

芯动力——硬件加速设计方法

第七章 基于平头哥E902处理器的SoC设计 ——(2)玄铁E902处理器与无剑100 SoC体系架构

邸志雄@西南交通大学
zxdi@home.swjtu.edu.cn

slides与源代码网址 <http://www.dizhixiong.cn/class5/>

E902架构特征

玄铁E902是平头哥半导体有限公司自主研发的**极低功耗、极低成本**嵌入式CPU核。

主要针对：

智能卡

智能电网

低成本微控制器

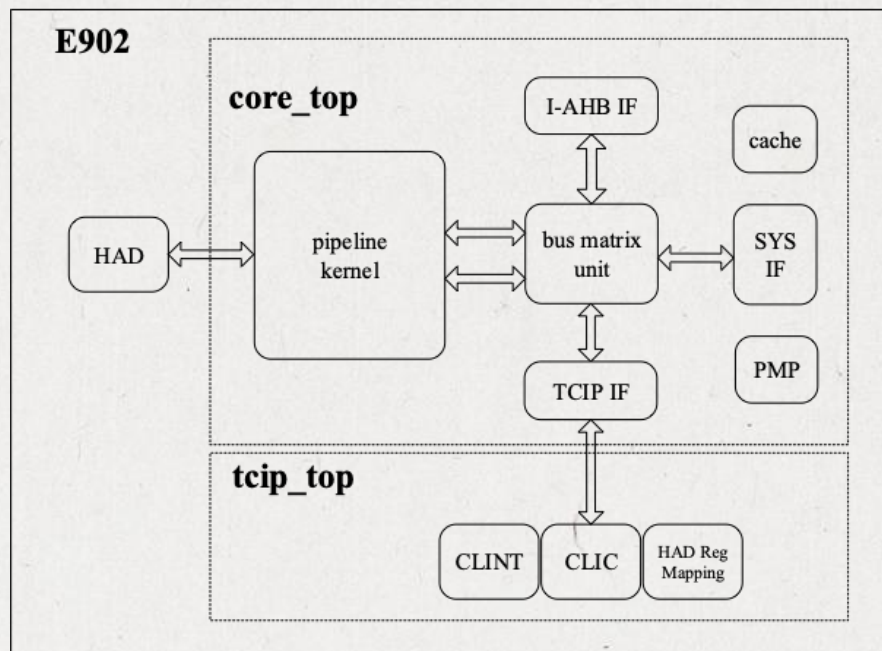
无线传感网络等嵌入式应用

适用于：

对功耗和成本极其敏感的IoT、MCU等领域。



可配置选项



硬件上“量体裁衣”

以上四个单元可以根据应用选择性地配置，如可以配置使用PMP内存保护单元应用于对安全性要求较高的IOT领域；配置是否使用硬件Cache和乘法器以提高处理器的性能；根据外设配置CLIC中断源的个数。通过选择性地配置可以更好地适配应用场景。

其中，PMP物理内存保护单元、指令Cache可以配置是否集成， wujian100 SoC中没有搭载这两个硬件单元。可以看出在最小配置下E902内核只有大约10K等效逻辑门。

可配置单元	配置选项	详细
指令cache	无 /2KiB/4KiB/8KiB	可以配置为2KiB、4KiB、8KiB
硬件乘法器	无/有	若配置硬件乘法器，仍可配置单周期快速乘法器或者多周期（3-34）慢速乘法器
内存保护单元	0/4/8/12/16	可以配置为0/4/8/12/16个表项，其中0表示不实现内存保护单元。
CLIC	中断源：1-240任意可配 中断优先级有效位：2-5位任意可配	支持1-240中断源任意可配，中断优先级有效位：2-5位任意可配，对应4-32个优先级

指令集

RISC-V指令集

基础指令集

根据处理器字长的不同划分

RV32I

RV32E

RV64I

RV128I

扩展指令集

可选的模块

“M” 乘除法指令集

“C” 压缩指令集

“F” 单精度浮点数指令集

⋮

扩展指令集均为可选的模块，通过选择不同的组合来满足不同的应用。
如选择RV32EC架构来追求小面积、低功耗的嵌入式场景。

RISC-V标准非常独特点

预留的指令编码空间

预定义的custom命令

添加自定义指令

大大增强了内核架构的灵活性。

扩展自定义指令集需要满足

- ①除定制的指令集外，必须完全符合RISC-V架构。
- ②硬件设计必须对自定义指令集进行支持
- ③需要对RISC-V的GCC编译器进行修改，保证编译器可以识别新指令。

玄铁E902支持RV32E[M]C指令集，其中M指令集可选择是否实现。当M指令集被选中时，需要硬件实现单周期或多周期乘法器。此外，还包括4条平头哥扩展指令集。

RISCV 32位机器字长处理器对应的整数指令集

RV32I

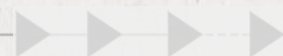


32个通用寄存器

RV32E



只实现了低16个



①RV32E整数指令集

RV32I

具有32个通用寄存器。
这些架构级的整数寄存器可以提升处理器运行的性能。

RV32E

只实现了RV32I中的低16个寄存器。
高16个寄存器消耗处理器核大约1/4的面积。
去掉可节省处理器核面积，同时减少了处理器核功耗。

②M乘法指令集

乘除法扩展指令集共包含八条指令。根据乘数和被乘数的类型如：

有符号数

无符号数

实现乘除法和取余操作

结果的截取范围

高32位

低32位

③C压缩指令集

RISC-V的压缩指令集对

32位

64位

128位

作出了扩展

压缩指令集提供了一种压缩机制可以将部分32位的指令压缩成16位。

程序中50%-60%的指令可以被RVC指令集替代，减小代码25%-30%大小，显著提升了代码密度。

④平头哥扩展指令集

E902实现了平头哥自定义扩展的cache指令集和同步指令集，其中在使用cache指令集时需要系统搭配有cache硬件。

RV32E指令集只实现了低16个。其中x0寄存器只读并且硬件设计需要满足其为常数0。每个寄存器的汇编助记符及其在函数调用中的功能如图所示。

Register	ABI Name	Description	Saver
x0	zero	Hard-wired zero	—
x1	ra	Return address	Caller
x2	sp	Stack pointer	Callee
x3	gp	Global pointer	—
x4	tp	Thread pointer	—
x5	t0	Temporary/alternate link register	Caller
x6-7	t1-2	Temporaries	Caller
x8	s0/fp	Saved register/frame pointer	Callee
x9	s1	Saved register	Callee
x10-11	a0-1	Function arguments/return values	Caller
x12-17	a2-7	Function arguments	Caller
x18-27	s2-11	Saved registers	Callee
x28-31	t3-6	Temporaries	Caller

RV32I指令集定义的32个通用寄存器

每个寄存器都有其规定的功能

x2作为堆栈指针

x1作为执行跳转指令时的返回地址

x10作为函数的返回值

x12-x17作为函数调用的参数。

这些通用寄存器在保存方式分为

caller-saved

调用者在调用函数时需要进行保存和恢复，被调用方可以任意修改或破坏该寄存器的值。

callee-saved

被调用函数负责保存和恢复的工作，或者是在被调用期间不破坏该寄存器的值。

RISC-V标准规定的三种特权级别：

机器模式 (Machine-mode)

普通用户模式 (User-mode)


超级用户模式 (Supervisor Mode)

特权层级用于在软件中提供不同的操作权限。机器模式等级最高，普通用户等级模式最低，在高特权层运行的代码比低特权层运行的代码具有更多的权限，受到的约束也越少。

Level	Encoding	Name	Abbreviation
0	00	User/Application	U
1	01	Supervisor	S
2	10	<i>Reserved</i>	
3	11	Machine	M

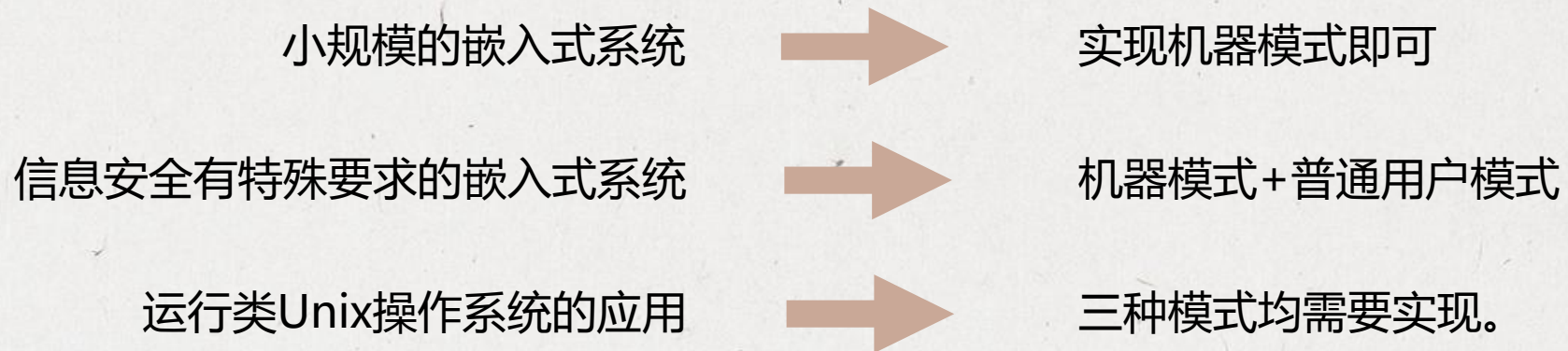
RISC-V特权层级

其中**机器模式是必须实现的**，RISC-V规定只有机器模式才具有对处理器底层的访问如内核相关控制寄存器和特权指令等。普通用户模式和超级用户模式分别用于执行常规应用程序和操作系统，可以选择性地加以实现。



Number of levels	Supported Modes	Intended Usage
1	M	Simple embedded systems
2	M, U	Secure embedded systems
3	M, S, U	Systems running Unix-like operating systems

RISC-V特权层级的结合与应用





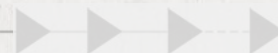
两种模式比较

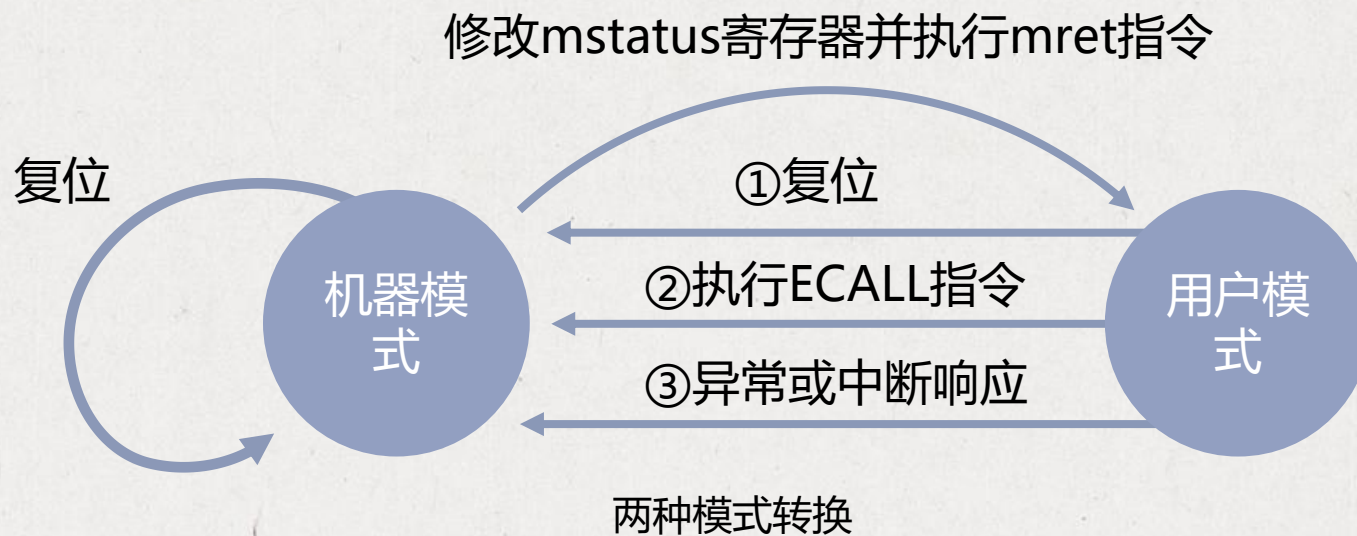
机器模式

- 可以访问所有通用寄存器、内核相关的控制寄存器。
- 可以执行所有指令。

普通用户模式

- 只可以访问通用寄存器。
- 不可执行对系统产生重大影响的**特权指令**如WFI, MRET, CSR和平头哥扩展的CACHE指令





注意：处理器在异常或中断响应时进入机器模式，执行ECALL指令属于主动触发异常从而进入机器模式。在退出异常或中断时可以选择修改mstatus寄存器进入任一模式，如不修改则恢复异常中断之前的模式。

大多数RISC处理器 采用的流水结构

- 取指
- 译码
- 执行
- 访存
- 写回



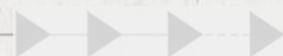
玄铁E902处理器有2级流水线:

①取指令

②指令译码、执行与回写

流水线名称	缩写	流水线作用
指令提取	IF	1、访问指令 Cache 、指令总线。 2、分支地址计算。
指令译码，执行与回写	ID	1. 指令译码。 2. 寄存器堆访问。 3. 指令执行。 4. Load/Store 指令访存地址的产生。 5. 访问数据总线。 6. 指令执行结果回写。

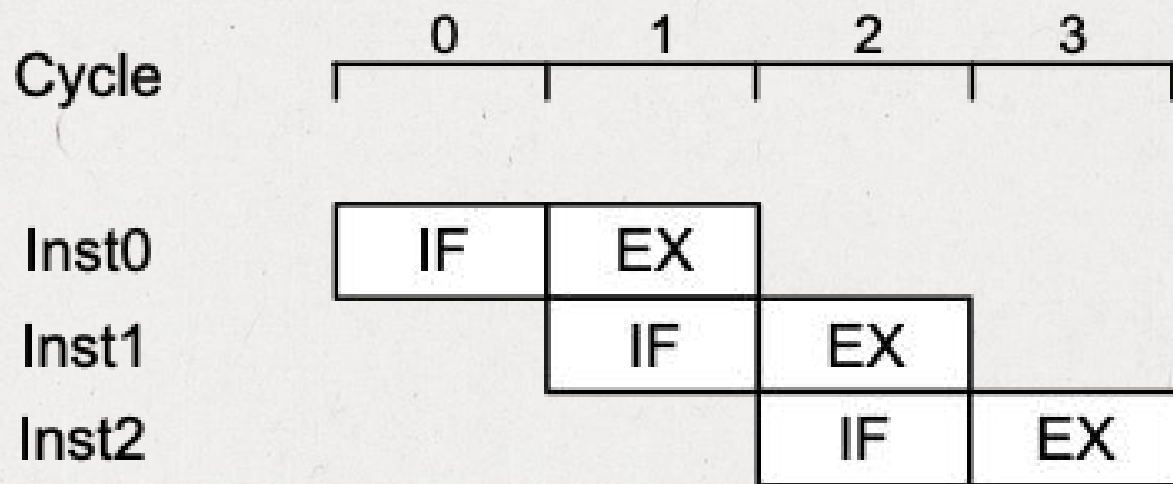
各级流水线作用



E902采用按序单发射机制，一个时钟周期发射一条指令。

①单周期指令

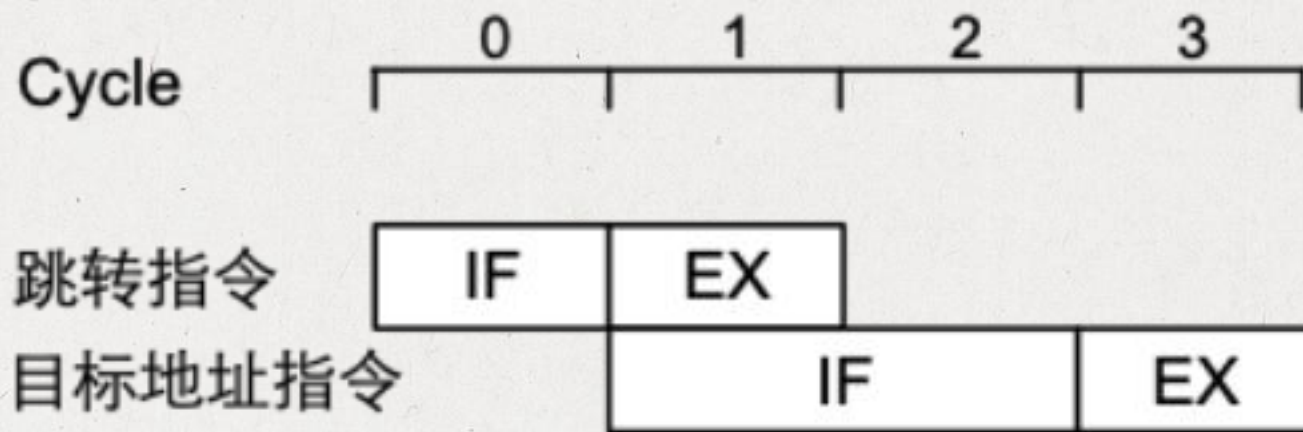
如算术和逻辑指令流水执行过程



E902采用按序单发射机制，一个时钟周期发射一条指令。

②跳转指令 (branch指令、jump指令)

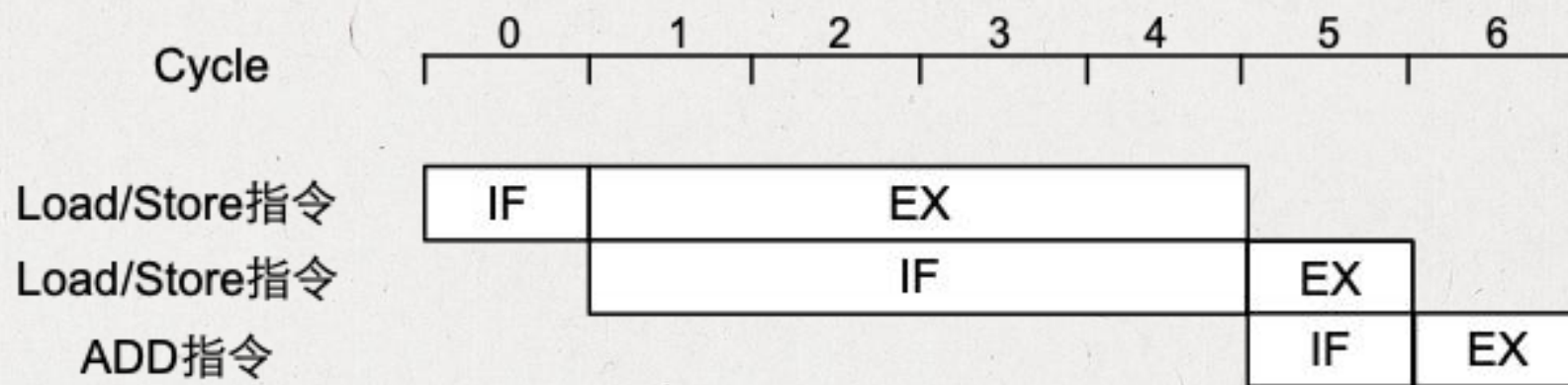
至少需要两个周期来填充流水线如果跳转指令的目标指令是字未对齐的32位指令时，取指需要两次访问指令总线，存在至少两个时钟周期的延迟。



E902采用按序单发射机制，一个时钟周期发射一条指令。

③对于内存访问指令（Load/Store）

因为采用的是**阻塞发射机制**，必须等到Load/Store指令完成之后。后续指令才能执行。当Load/Store指令等待总线返回响应时即会造成后续指令执行的停顿。



E902多总线接口包括

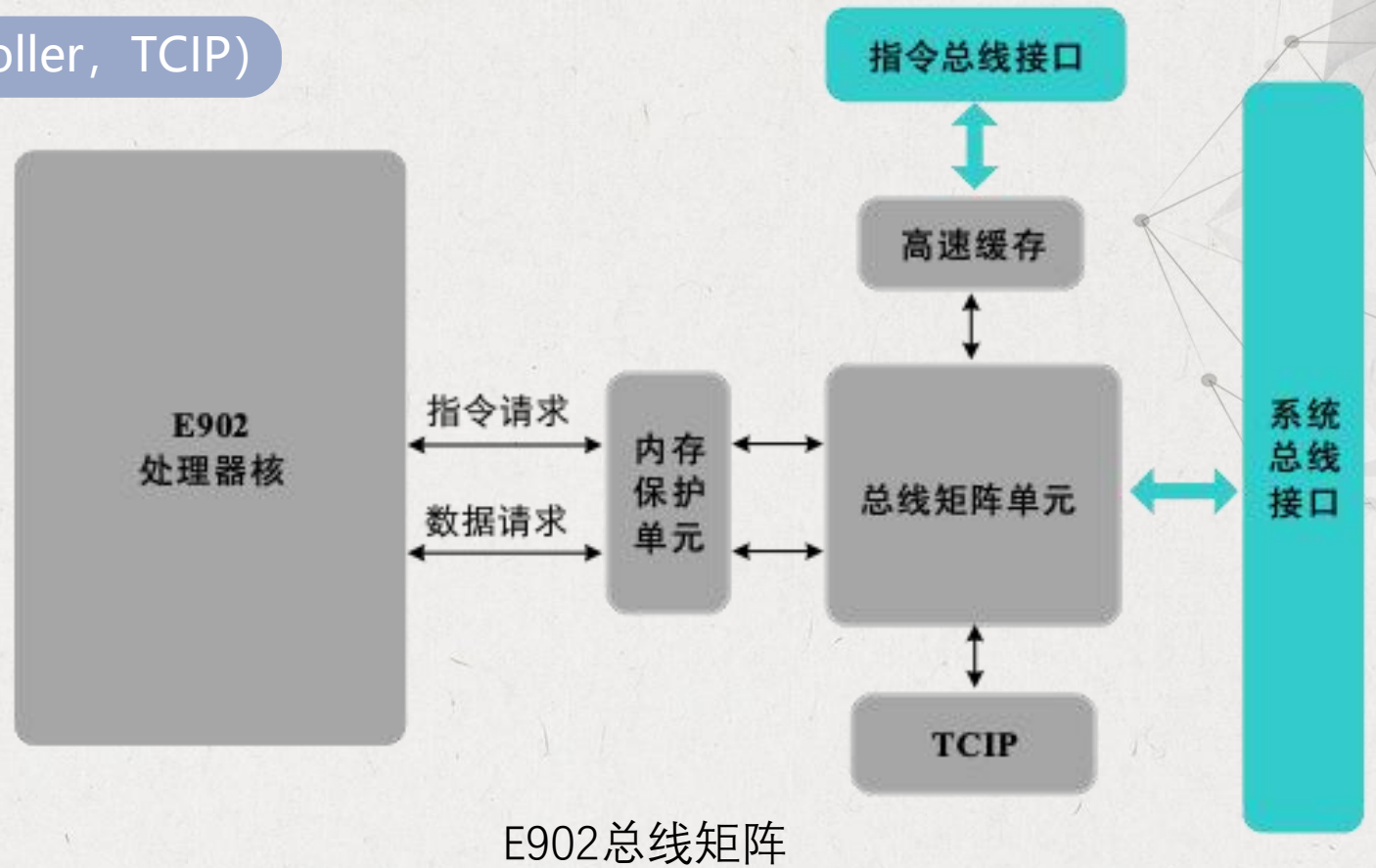
系统总线

指令总线

紧耦合IP接口 (core local interrupt controller, TCIP)

用于CLINT (Core-Local Interrupt Controller)、CLIC中断控制器和HAD硬件辅助调试单元 (Hardware Assistant Debug) 寄存器的访问。当总线矩阵接收访问紧耦合IP的地址的请求时, 该请求将直接在核内处理, 不影响指令总线和系统总线。

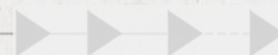
E902的系统总线与指令总线支持AMBA3.0 AHB-Lite协议, 但考虑到E902的应用领域及成本, 只实现了AHB-Lite协议中的部分内容。





总线矩阵的功能

- ①为处理器内部请求访问外部总线接口提供了**互联功能**。**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、指令总线以及TCIP接口上。
- ②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。





E902物理内存保护单元 (Physical Memory Protection, PMP) 主要保护两类系统资源:

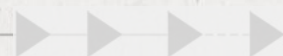
存储器


外围设备

PMP负责

对存储器和外围设备访问的合法性进行检查

判定当前工作模式下CPU是否具备对内存地址的读/写/执行访问权限。



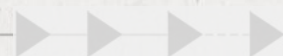


E902设计实现了RV32E的WFI指令用于使处理器从正常工作模式转入低功耗模式。

在低功耗模式下

E902的内部门控时钟管理单元会将绝大多数的寄存器时钟关闭，而跟处理器唤醒功能相关的逻辑部分的时钟不会被关闭。低功耗模式下，E902不会向总线发起数据传输请求，内部流水线停顿。

注意：wujian100 SoC中没有实现PMP单元和低功耗的相关设计。



PPA

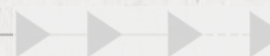
Performance	1.55 DMIPS/MHz (O2) 2.69 Coremark/MHz (O3)
Frequency	150MHz@worst case
Area	12K gates@minimal core
Power	4 uW/MHz

TSMC 40nm, 9T, RVT

Configuration is RV32EC, excluding PMP/CACHE/CLIC, etc.

平头哥官方提供的E902内核的PPA指标

可以看出，在台积电40nm的工艺下，E902内核最简版即只实现RV32EC指令集，并且除去了PMP内存保护单元、CACHE、CLIC中断管理器和其它外设的情况下，处理器最差可以达到150Mhz的频率，并且功耗只有4uW/MHz，Coremark跑分也达到了2.69/Mhz的较好结果。做到了极低功耗、集成度较高和较高的性能。



CoreMark与Dhrystone是嵌入式处理器领域最为知名和常见的跑分程序。由于Coremark和Dhrystone均只使用了**整数运算类型**，因此并不能衡量浮点运算处理性能。对于处理器的浮点性能指标，有众多专门的浮点Benchmark程序能够运行，这里不展开介绍。它们的对比如下：

	Dhrystone	CoreMark
编译器优化	很容易受到编译器优化的影响	对编译器的优化做了处理
库调用	包含库调用，很难再去比较不同的库	测试的计时部分不进行任何库调用
版本控制	没有官方来源，有几个版本在同时使用	有官方来源，只有一个版本
代码兼容性	C 代码不兼容 ANSI C	代码兼容 ANSI C



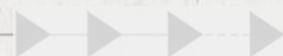
Dhrystone跑分局限性:

- ①实际上反应的更多是编译器优化的能力，而不只是MCU的处理能力。
- ②库调用在Dhrystone中消耗了基准测试大部分时间，不同的库很难比较结果。
- ③ Dhrystone本身太小，能衡量的东西太少，反应不出CPU全面的性能。

CoreMark针对Dhrystone跑分的局限性做出了设计改进，目前已迅速成为量测与比较处理器性能的业界标准基准测试。

实现原理:

CoreMark测试主循环由一个迭代循环组成，并且可以通过参数控制循环的次数。根据硬件定时器得出单位时间循环执行的次数，再除以处理器的主频，便可以得出CoreMark/MHz跑分值。



wujian100的SDK中包含了CoreMark和Dhrystone两种跑分测试工程，跑分结果最终通过串口输出，这里只对CoreMark结果进行记录。

```
2K performance run parameters for coremark.
CoreMark Size      : 666
Total ticks        : 124000000
Total time (secs): 124.000124
Iterations/Sec     : 48.387048
Iterations         : 6000
Compiler version   : GCC8.1.0
Compiler flags     :
Memory location    : STACK
seedcrc           : 0xe9f5
[0]crclist        : 0xe714
[0]crcmatrix      : 0x1fd7
[0]crcstate       : 0x8e3a
[0]crcfinal       : 0xa14c
Correct operation validated. See readme.txt for run and reporting
rules.
CoreMark 1.0 : 48.387048 / GCC8.1.0 / STACK
Score (Coremarks/MHz): 2.42
```

CoreMark跑分日志（串口输出）

Processor	Cert.	Compiler	Execution Memory	MHz	Cores	CoreMark	CoreMark / MHz↑
Espressif ESP8266		xtensa-lx106-elf-gcc...	Stack (DRAM)	80	1	191.00	2.38
Sun Microsystems SunFire V210 (2P x ...		GCC3.4.3 (csl-sol21...	DDR1-266; Heap	1503	2	3591.63	2.39
Calxeda EnergyCore ECX-1000		GCC 4.6.3	4GB DDR3 1333Mhz	1100	1	2639.22	2.40
Calxeda EnergyCore ECX-1000		GCC 4.6.3	4GB DDR3 1333Mhz	1100	1	2639.22	2.40
Intel(R) Atom(TM) CPU Z520		GCC4.4.5	1GB / Heap	1330	1	3192.17	2.40
Calxeda EnergyCore ECX-1000		GCC 4.6.3	4GB DDR3 1333Mhz	1400	1	3363.01	2.40
NXP KL25		IAR 6.60	Code in internal Flas...	48	1	115.46	2.40
Texas Instruments OMAP3530		Sourcery G++ 4.4-179	DDR 166MHz; Heap	500	1	1204.60	2.41
Texas Instruments OMAP3530		GCC4.3.3	DDR 166MHz / HEAP	500	1	1207.37	2.42
NXP KL05		IAR 6.60	Code in internal Flas...	48	1	116.18	2.42
Microchip PIC32MX360F512L (MIPS3...		GCC3.4.4 Microchip ...	Code in FLASH (0 wa...	30	1	72.78	2.43
NXP Kinetis K60 90nm	✓	Green Hills Multi v6....	Code in internal Flas...	100	1	243.40	2.43

CoreMark官网跑分排行

右图是CoreMark官网上提供的处理器跑分表，将E902内核与跑分结果在2.4左右的处理器进行对比，可以发现E902内核性能对标ARM的Cortex-m0的处理器。

Directory Structure

```
|--Project           //open source project work directory
|--riscv_toolchain   //tool chain install directory download from t-head.cn
|--wujian100_open    //wujian100_open project get from github
|--case             //test case example for simulation
|--doc              //wujian100_open user guide
|--fpga             //FPGA script
|--lib              //compile script for simulation
|--regress          //regression result
|--sdk              //software design kit
|--soc              //Soc RTL source code
|--tb               //test bench
|--tools            //simulation script and setup file
|--workdir          //simulation directory
|--LICENSE
|--README.md
```

上图是整个开源平台的目录结构，包含了FPGA实现、仿真和软件开发各个部分。

Wujian100 SoC整体可划分的四个模块:

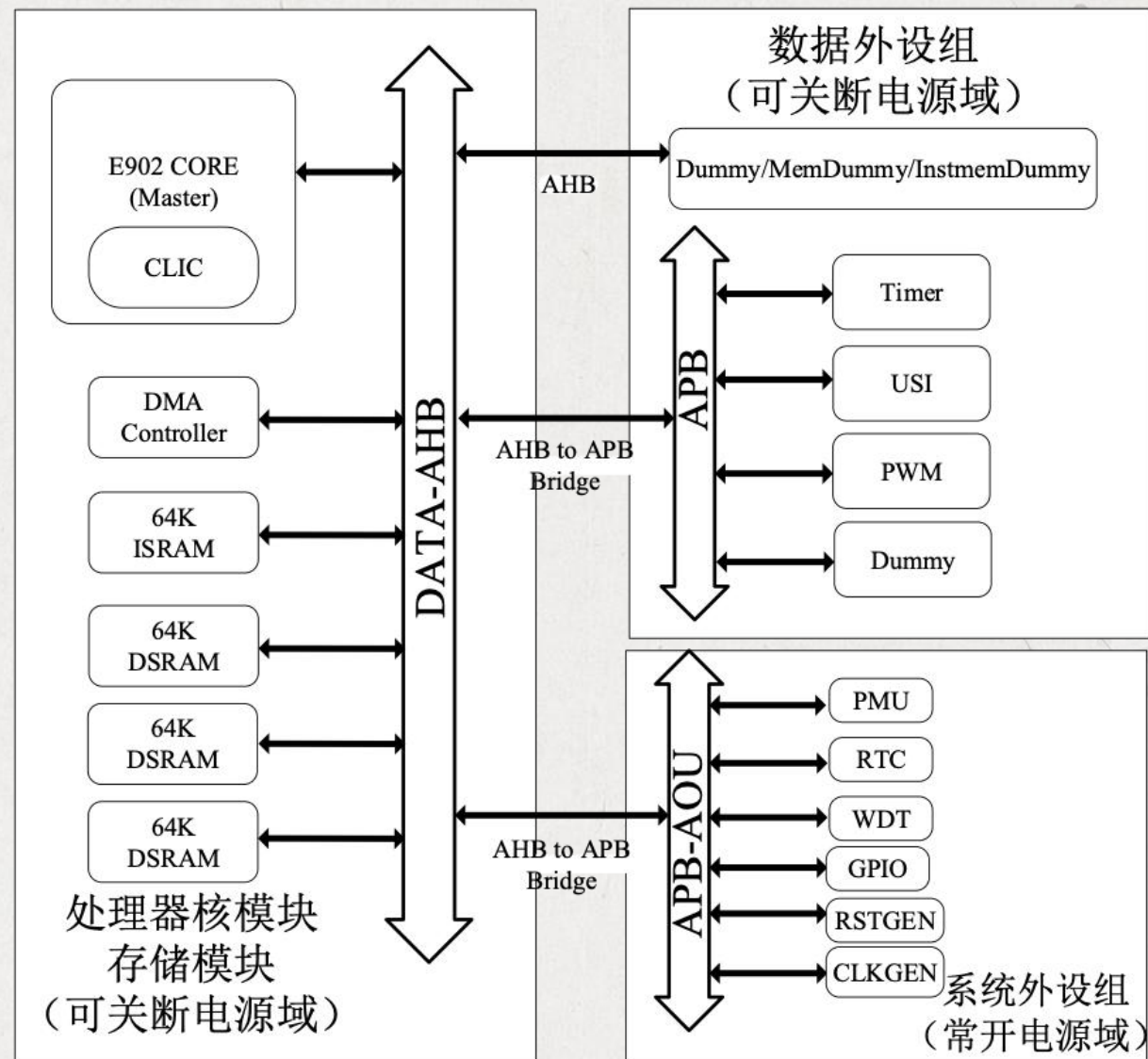
处理器核模块(cpu_top)

存储模块(retu_top)

常开电源域模块(aou_top)

可关断数据域模块(pdu_top)

数据外设组模块主要包含一些常用的外设，用于芯片和外界进行数据交换。另外，该模块还包含了Dummy模块即系统预留的用于扩展自行设计的IP接口。



维持SOC正确运转所需要的必须模块

功耗管理模块(PMU)

实时时钟模块(RTC)

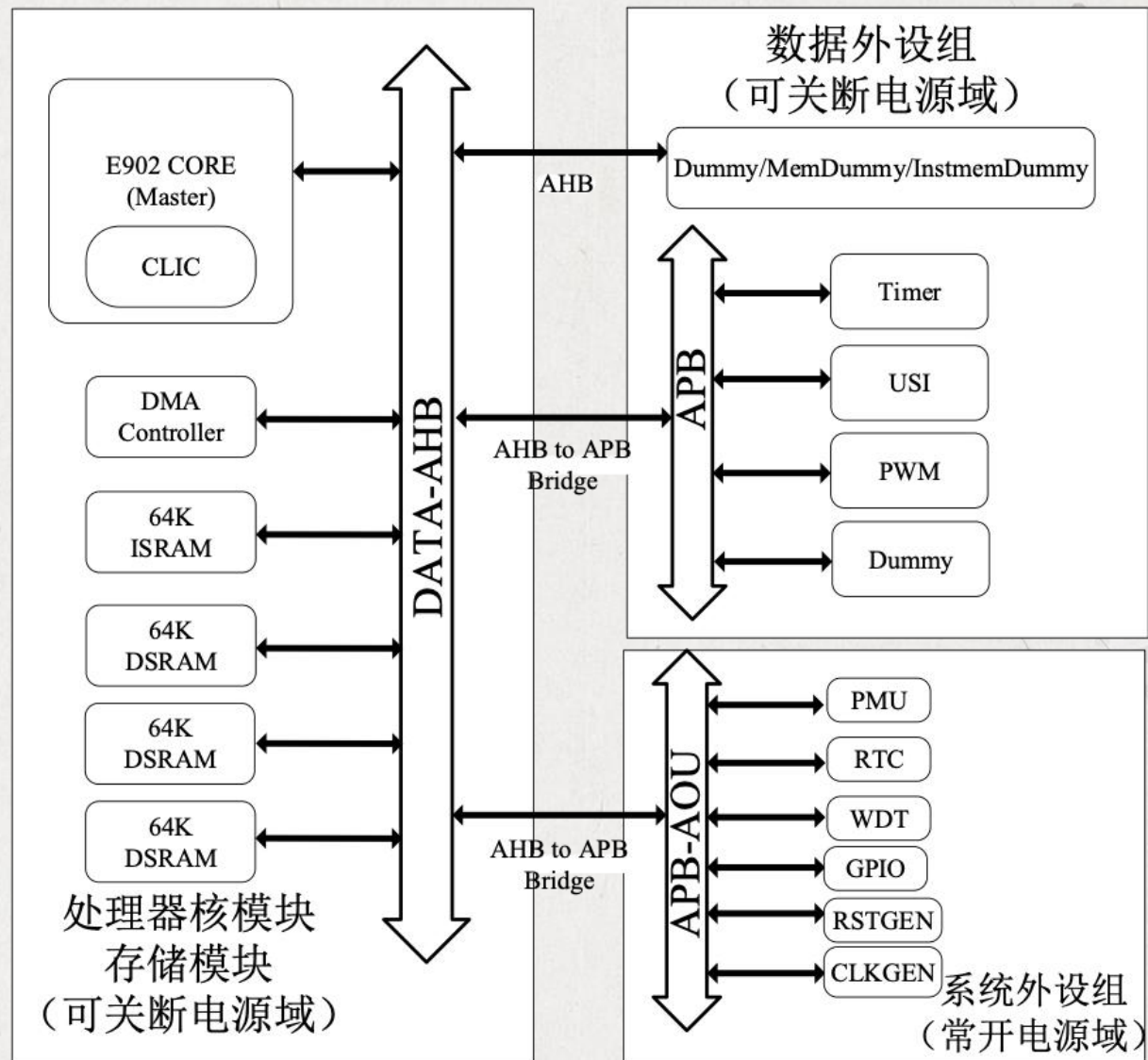
看门狗模块(WDT)

通用输入输出模块(GPIO)

复位模块(RSTGEN)

时钟产生模块(CLKGEN)

CLIC中断控制器直接通过系统紧耦合IP接口而非系统总线进行访问，因此这里将其划入E902内核部分。



wujian100 SoC为了便于用户进行扩展自设计的硬件IP和存储模块，预留了如下作为主机、从机的两种Dummy模块。Dummy模块内部将输出信号默认进行了拉低处理，在模块外部与总线矩阵相连即将挂载到AHB/APB总线上，无需用户自行实现。

```
module ahb_dummy_top(
    haddr,
    hclk,
    hprot,
    hrdata,
    hready,
    hresp,
    hrst_b,
    hsel,
    hsize,
    htrans,
    hwdata,
    hwwrite,
    intr
);
input  [31:0] haddr;
input  hclk;
input  [3 :0] hprot;
input  hrst_b;
input  hsel;
input  [2 :0] hsize;
input  [1 :0] htrans;
input  [31:0] hwdata;
input  hwwrite;
output [31:0] hrdata;
output hready;
output [1 :0] hresp;
output intr;
wire [31:0] hrdata;
wire hready;
wire [1 :0] hresp;
wire intr;
assign hrdata[31:0] = 32'b0;
assign hready = 1'b1;
assign hresp[1:0] = 2'b0;
assign intr = 1'b0;
endmodule
```

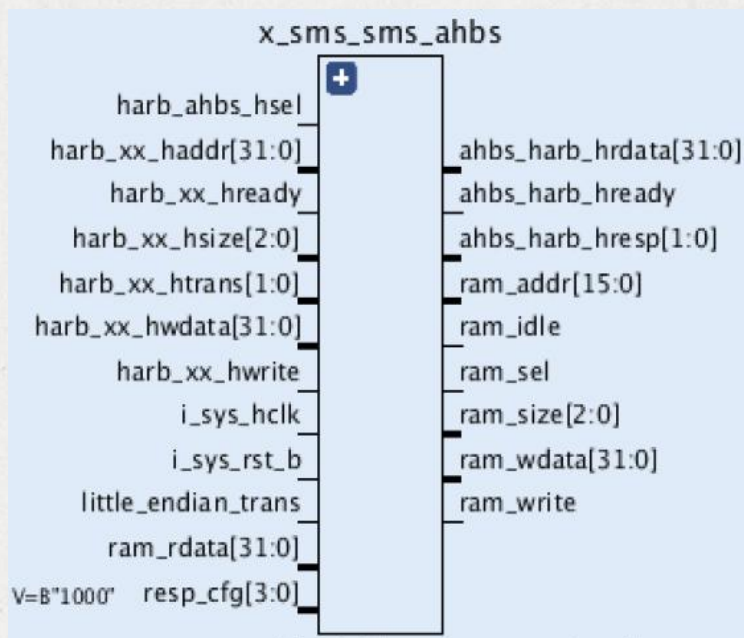
Dummy模块(Slave)

```
module ahbm_dummy_top(
    hclk,
    hrst_b,
    mhrdata,
    mhresp,
    mhready,
    mhgrant,
    mhwdata,
    mhbust,
    mhtrans,
    mhwwrite,
    mhaddr,
    mhsize,
    mhprot
);
output [31:0] mhaddr;
output [2 :0] mhbust;
output [3 :0] mhprot;
output [2 :0] mhsize;
output [1 :0] mhtrans;
output [31:0] mhwdata;
output mhwwrite;
input hclk;
input hrst_b;
input mhgrant;
input [31:0] mhrdata;
input mhready;
input [1 :0] mhresp;
wire [31:0] mhaddr;
wire [2 :0] mhbust;
wire [3 :0] mhprot;
wire [2 :0] mhsize;
wire [1 :0] mhtrans;
wire [31:0] mhwdata;
wire mhwwrite;
assign mhaddr[31:0] = 32'b0;
assign mhbust[2 :0] = 3'b0;
assign mhprot[3 :0] = 4'b0;
assign mhsize[2 :0] = 3'b0;
assign mhtrans[1 :0] = 2'b0;
assign mhwdata[31:0] = 32'b0;
assign mhwwrite = 1'b0;
endmodule
```

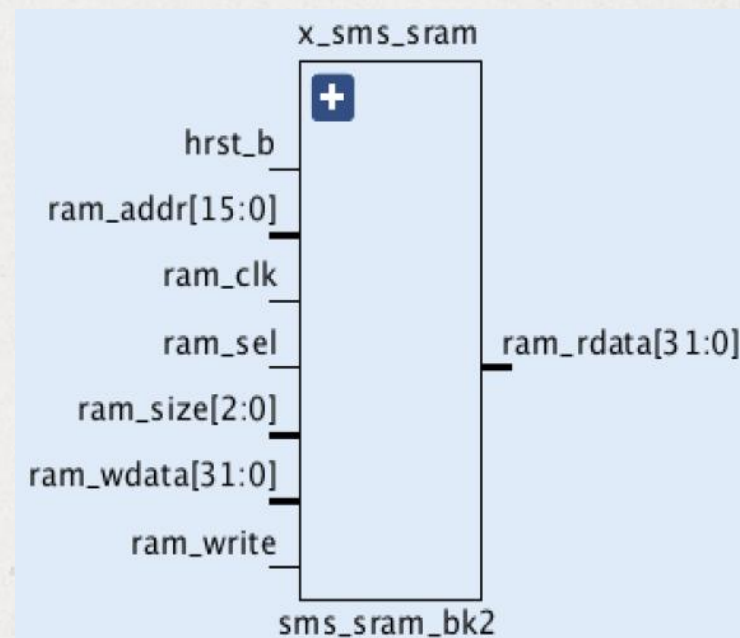
Dummy模块(Master)

这两个模块除了作为AHB主从机控制信号的差异外，从机Dummy模块还可以产生CPU中断信号，而主机Dummy模块不能。

Wujian100 SoC内部包含3块64KB的数据RAM和1块64KB的指令ROM，通过AHB总线挂载到处理器上，只支持小端模式存储数据指令。



协议转换模块



RAM模块

RAM组成部分:

连接AHB和RAM接口的协议转换模块

RAM存储模块

这里虽然称作为ROM，但实际上是通过FPGA BRAM资源实现的，当掉电之后仍需要重新为CPU加载运行程序。可以将运行完好的程序预先写入ROM空间内再进行FPGA综合实现，这样每次给FPGA加载bitstream时ROM中同时会被加载程序。

RAM存储模块由4个8位的子RAM模块组成，协议转换模块将地址信息和读写字节数等控制信号传入RAM模块后，在内部完成对4个子RAM模块的选择控制，从而根据AHB总线控制需要实现访问任意字节的存储空间。

```
module fpga_byte_sram(
    A,
    CEN,
    CLK,
    D,
    Q,
    WEN
);
parameter ADDRWIDTH = 17;
parameter DATAWIDTH = 8;
parameter MEMDEPTH = 2**(ADDRWIDTH);

input [(ADDRWIDTH-1):0] A;
input CEN;
input CLK;
input [(DATAWIDTH-1):0] D;
input WEN;
output [(DATAWIDTH-1):0] Q;

reg [(DATAWIDTH-1):0] Q;
reg [(DATAWIDTH-1):0] mem[(MEMDEPTH-1):0];

always@(posedge CLK)
begin
    if(!CEN)
    begin
        if(!WEN)
        begin
            mem[A] <= D;
            Q <= D;
        end
        else
        begin
            Q <= mem[A];
        end
    end
end
endmodule
```

```
fpga_byte_sram #(ADDRWIDTH,8,MEMDEPTH) x_fpga_byte0_sram(
    .A(A),
    .CEN(CEN),
    .CLK(CLK),
    .D(D[7:0]),
    .Q(Q[7:0]),
    .WEN(BWEN[0])
);

fpga_byte_sram #(ADDRWIDTH,8,MEMDEPTH) x_fpga_byte1_sram(
    .A(A),
    .CEN(CEN),
    .CLK(CLK),
    .D(D[15:8]),
    .Q(Q[15:8]),
    .WEN(BWEN[1])
);

fpga_byte_sram #(ADDRWIDTH,8,MEMDEPTH) x_fpga_byte2_sram(
    .A(A),
    .CEN(CEN),
    .CLK(CLK),
    .D(D[23:16]),
    .Q(Q[23:16]),
    .WEN(BWEN[2])
);

fpga_byte_sram #(ADDRWIDTH,8,MEMDEPTH) x_fpga_byte3_sram(
    .A(A),
    .CEN(CEN),
    .CLK(CLK),
    .D(D[31:24]),
    .Q(Q[31:24]),
    .WEN(BWEN[3])
);
```

子RAM模块的底层采用寄存器数组的描述方法实现，在Vivado工具会被解析并综合成BRAM资源。

注意：如果要使Vivado将寄存器描述的存储器综合为RAM单元，代码书写一定要符合综合工具规范，如果描述不规范将导致综合器无法识别解析成BRAM，从而用LUT资源实现，消耗大量资源。

每个子RAM模块采用了先写后读的策略。

Address Range	IP name	Size	Master/Slave	Description
0x0000_0000~0x0000_FFFF	ISRAM	64K	S0	ROM
0x1000_0000~0x1007_FFFF	MemDummy	512K	S1	instmem_dummy_top0
0x2000_0000~0x2000_FFFF	SRAM	64K	S2	DATA SRAM
0x2001_0000~0x2001_FFFF	SRAM	64K	S3	DATA SRAM
0x2002_0000~0x2002_FFFF	SRAM	64K	S4	DATA SRAM
0x3000_0000~0x3007_FFFF	MemDummy	512K	S5	datamem_dummy_top1
0x4001_0000~0x4001_FFFF	Dummy	64K	S7	main_dummy_top0
0x4002_0000~0x4002_FFFF	Dummy	64K	S8	main_dummy_top1
0x4010_0000~0x401F_FFFF	Dummy	1M	S9	main_dummy_top2
0x8000_0000~0x9FFF_FFFF	Dummy	512M	S11	main_dummy_top3

wujian100 SoC存储器地址映射空间

wujian100 SoC的指令RAM和数据RAM统一例化同一个64K大小的RAM单元，从地址映射空间上可以看到三个数据RAM地址是**连续的**，如果直接将例化的RAM单元增大将会造成模块间的冲突。

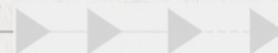


可以利用系统预留的Dummy从机模块扩展RAM/ROM。

扩展实现的两种方法：

- ①例化wujian100提供的支持AHB协议的RAM/ROM单元
- ②调用Vivado提供的BRAM controller IP，由于其只支持AXI接口，所以需要同时调用Vivado提供AHB转AXI的接口。

除了硬件上实现存储器的扩展外，软件上需要修改链接脚本以提示链接器数据和代码可以存放在新添加的存储器部分。



wujian100的链接脚本位于board/wujian100_open_evb/gcc_csky.ld目录中。

text

data

bss

链接时会将编译产生的所有目标文件中的数据段合并为一个。

text

程序代码段

data

存放所有初始值不为0的
全局变量和静态变量

bss

bss段存放未初始化或初
始值为0的全局变量。

MEMORY

```
{
    I-SRAM : ORIGIN = 0x0      , LENGTH = 0x00010000 /* I-SRAM 64KB */
    SRAM   : ORIGIN = 0x20000000, LENGTH = 0x20030000 /* on-chip SRAM 192KB */
}

REGION_ALIAS("REGION_TEXT",    I-SRAM);
REGION_ALIAS("REGION_RODATA",  I-SRAM);
REGION_ALIAS("REGION_DATA",    SRAM);
REGION_ALIAS("REGION_BSS",     SRAM);
```

wujian100链接脚本中存储空间分布

wujian100的链接脚本位于board/wujian100_open_evb/gcc_csky.ld目录中。



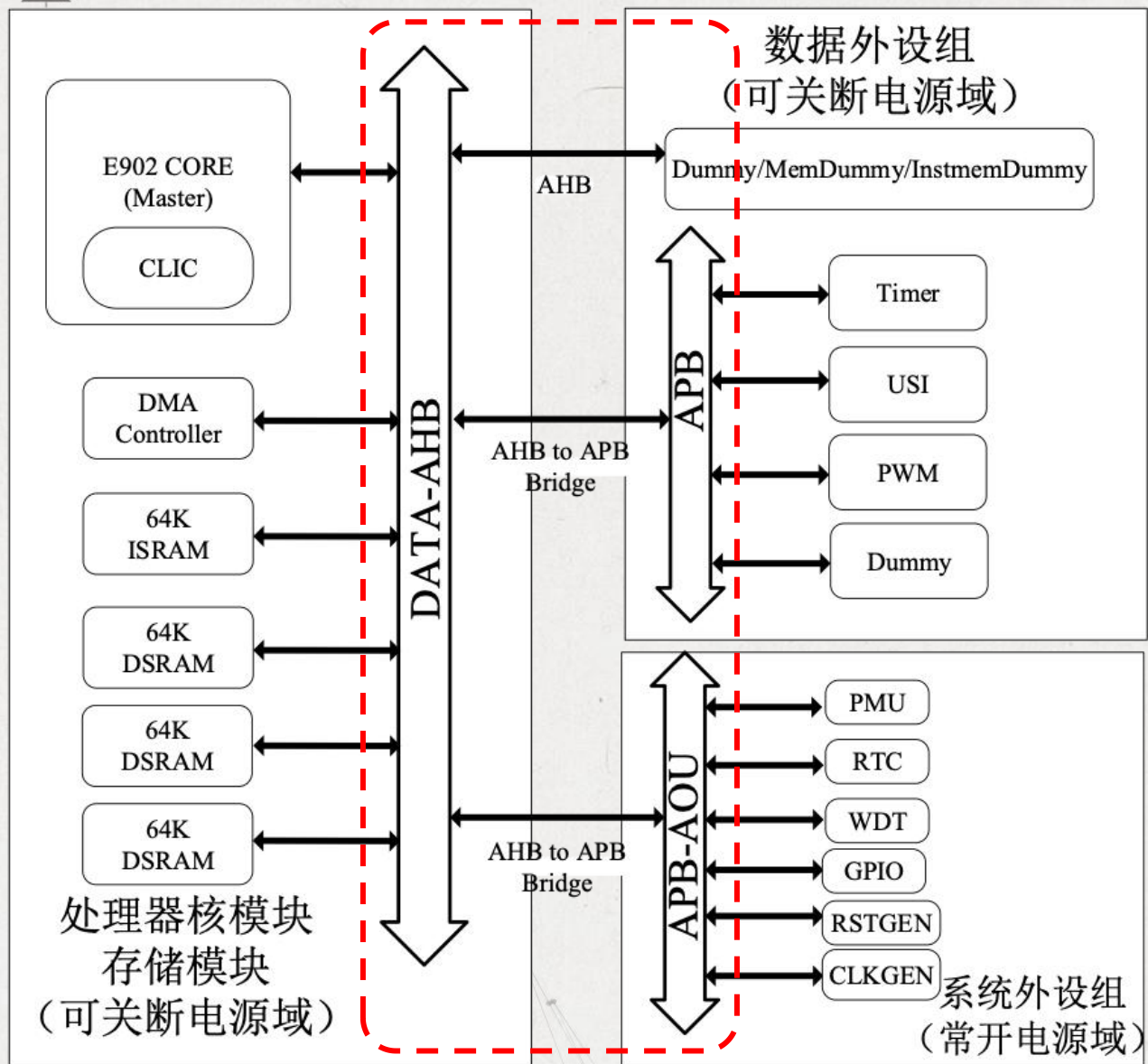
会被存放在ROM中，在启动文件中处理器会将data段数据由ROM搬运至RAM，并且将BSS段数据在数据RAM的相应位置进行初始化为0处理。

```
MEMORY
{
    I-SRAM : ORIGIN = 0x0      , LENGTH = 0x00010000 /* I-SRAM 64KB */
    SRAM   : ORIGIN = 0x20000000, LENGTH = 0x20030000 /* on-chip SRAM 192KB */
}

REGION_ALIAS("REGION_TEXT",    I-SRAM);
REGION_ALIAS("REGION_RODATA",  I-SRAM);
REGION_ALIAS("REGION_DATA",    SRAM);
REGION_ALIAS("REGION_BSS",     SRAM);
```

wujian100链接脚本中存储空间分布

当硬件上完成存储器扩展后，根据扩展存储器的地址分布需要修改链接脚本，从而指示链接器使用扩展后的存储空间。



AHB总线规范是AMBA总线规范的一部分，AMBA总线规范是ARM公司提出的总线规范，被大多数SoC设计采用。

AHB (Advanced High-performance Bus)

AHB用于高性能、高时钟频率的系统结构，典型的应用如系统内部的高速RAM、NAND FLASH、DMA、Bridge的连接。

APB (Advanced Peripheral Bus)

APB用于连接外低速部设备。

通常在Xilinx的FPGA中一般会用到AHB的AMBA总线最新版本AXI总线。而在大量嵌入式SOC里面，AHB和APB则最为常用。

AHB总线的强大之处

微控制器 (CPU)

高带宽的片上RAM

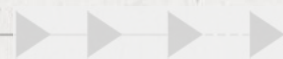
高带宽的外部存储器接口

DMA总线master

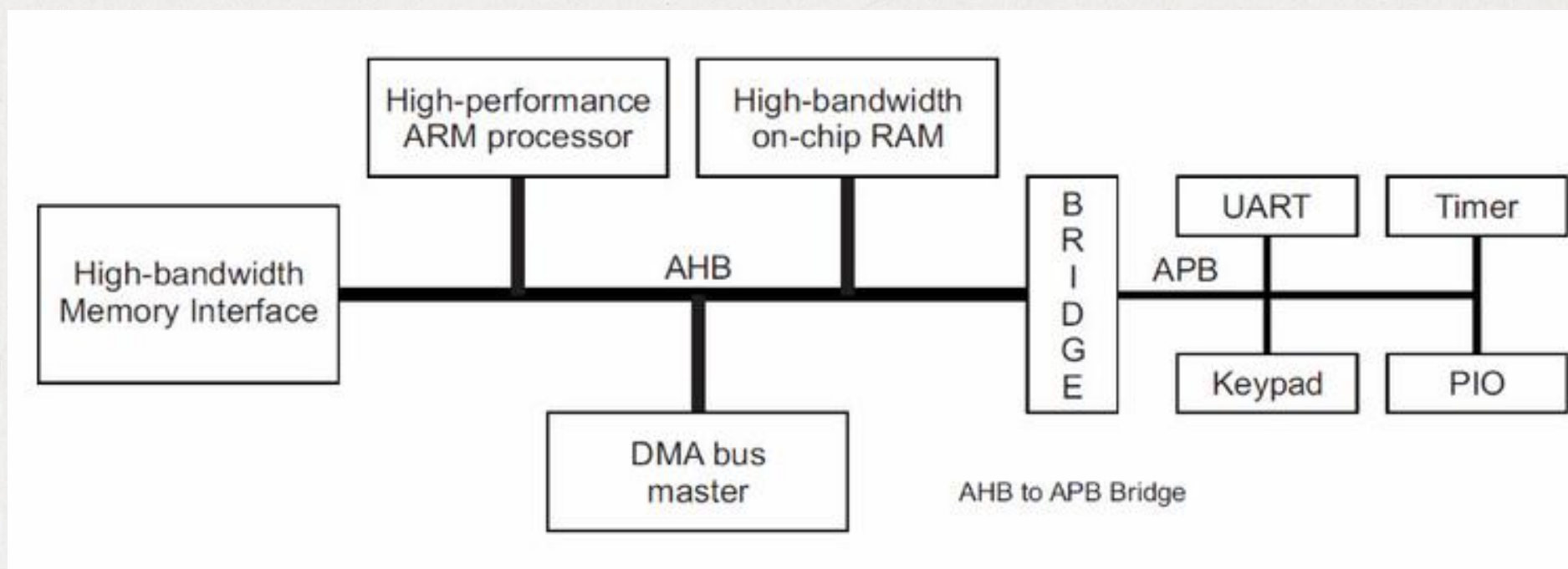
拥有AHB接口的控制器等

连接起来

构成一个独立的完整的SOC系统



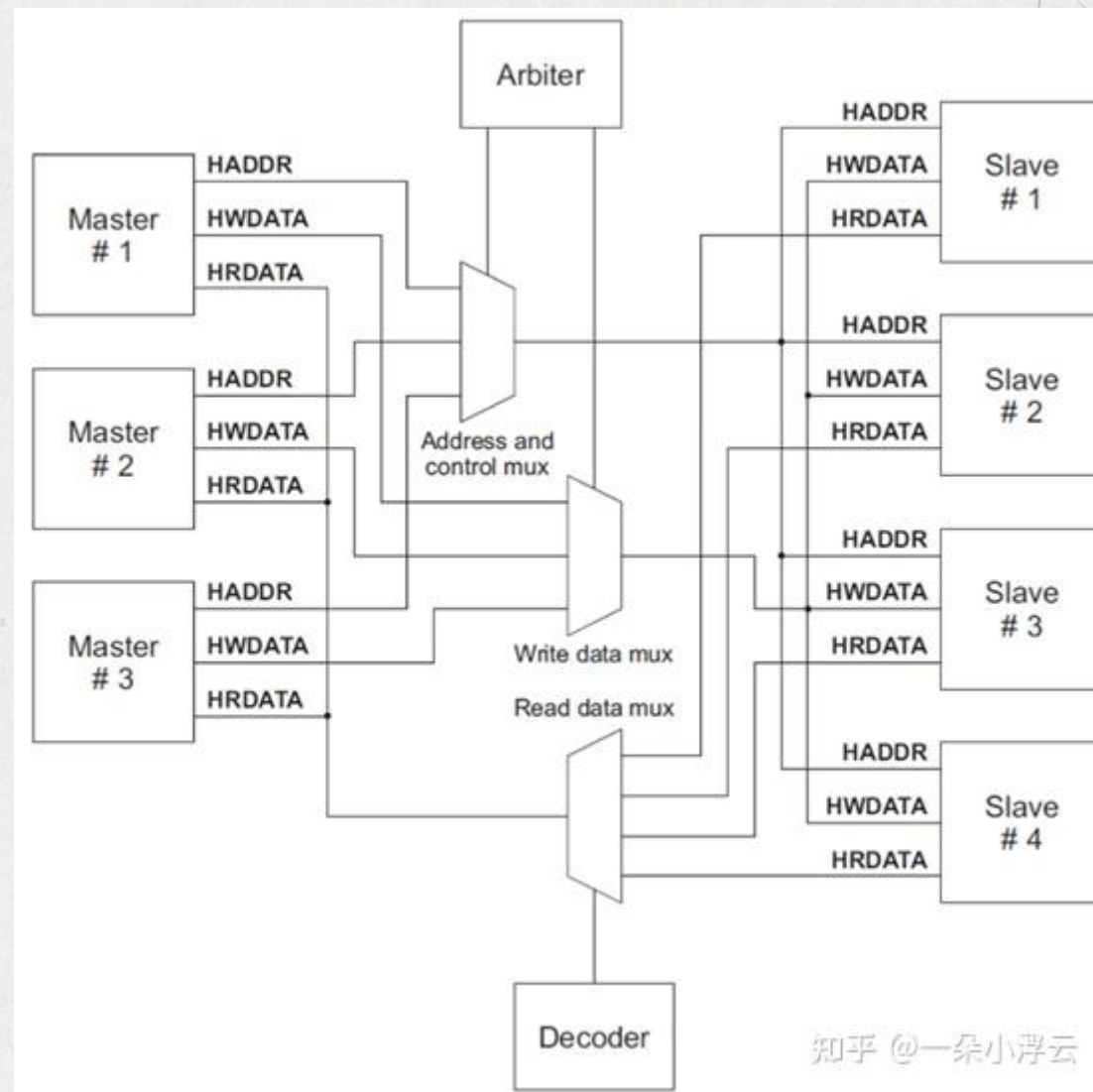
不仅如此，还可以通过AHB-APB桥来连接APB总线系统。AHB可以成为一个完整独立的SOC芯片的骨架，为系统的高性能运行起到了基石作用。



AHB总线通常设计时包含的几个设备：

AHB主控制器Master:

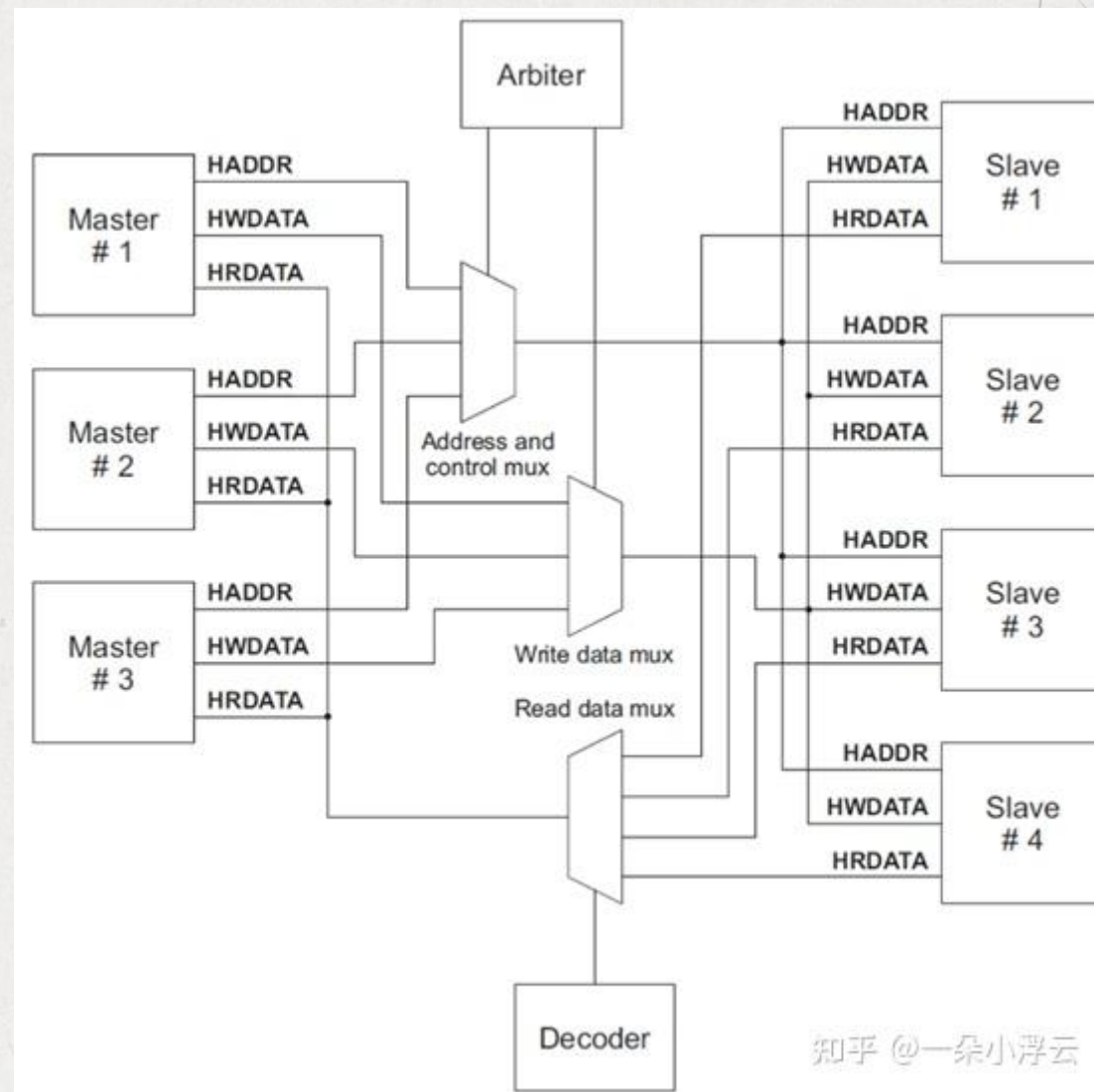
主控制器可以通过地址和控制信息，可以发起进行初始化，读，写操作。在同一时间，总线上只能有一个主控制器。



AHB总线通常设计时包含的几个设备：

AHB从设备Slave：

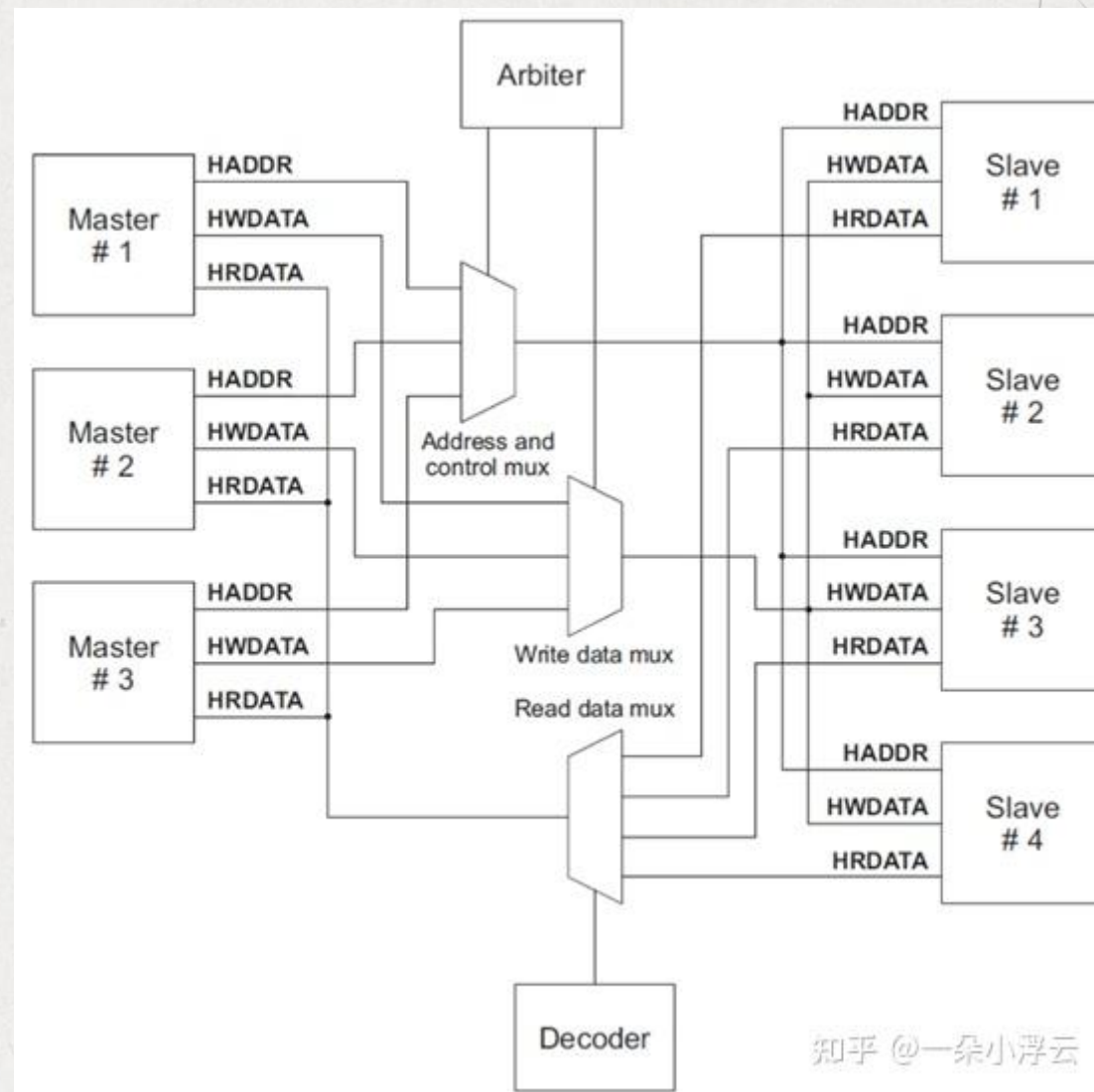
从设备通常是指在其地址空间内，通过地址映射响应主控制器发出的读写控制操作的被动设备，并对Master返回成功、失败或者等待状态。通过操作的成功与否反馈给其主控制器，完成数据的传输控制。



AHB总线通常设计时包含的几个设备：

AHB仲裁器Arbiter：

仲裁器根据用户的配置，确保在总线上同一时间只有一个主控制器拥有总线控制权限。

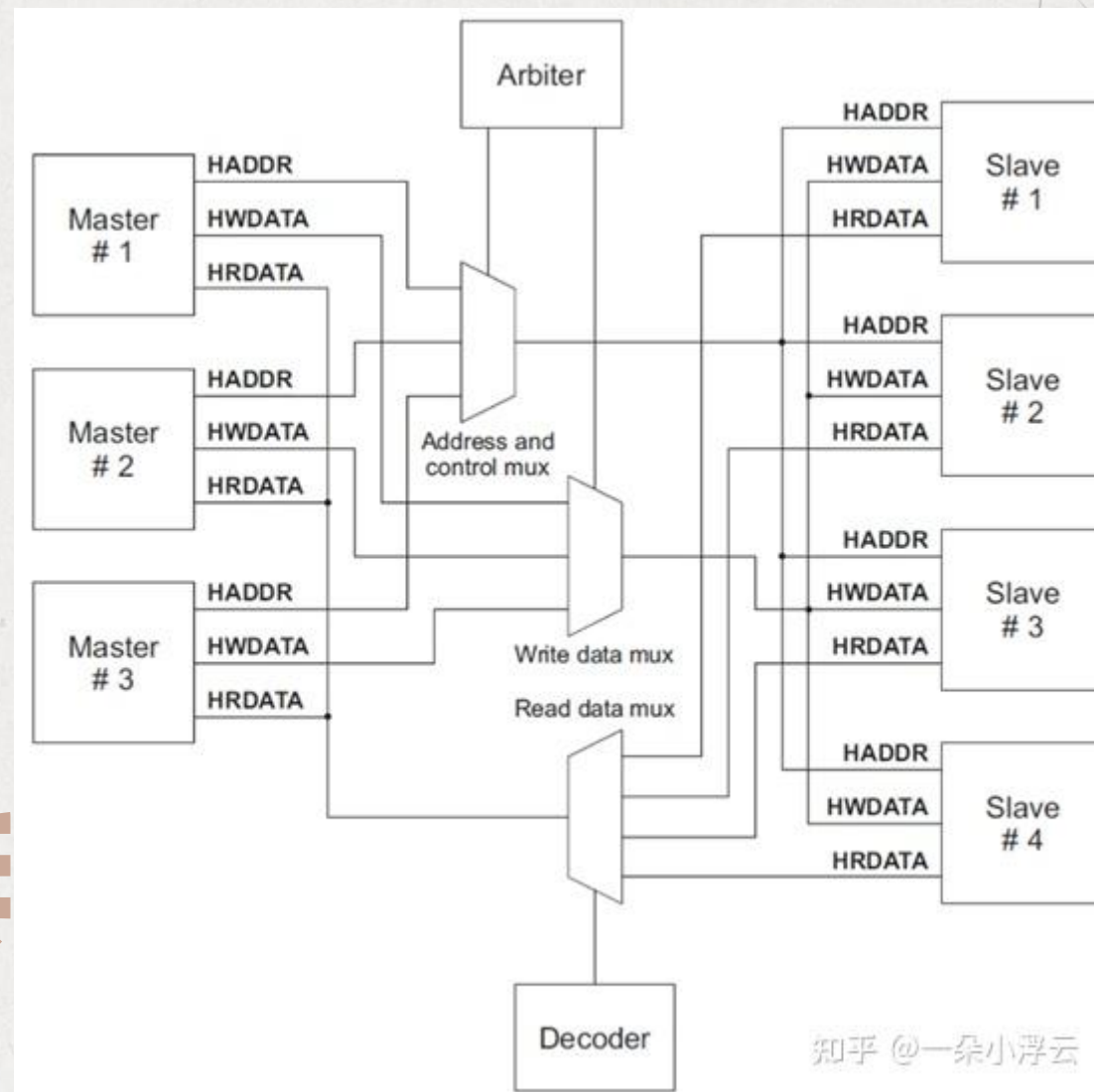


AHB总线通常设计时包含的几个设备：

AHB译码器Decoder:

译码器解析在总线上传输的地址和控制信息，负责对地址进行解码，并提供片选信号到各Slave，片选信号是通过组合逻辑对地址码译码产生的。只有当前的数据传输完成后（HREADY为高），Slave才会采样地址和控制信号以及片选信号。AHB总线上只能有一个译码器。地址译码器会带有实现默认slave的功能。

Master的addr和data都由Arbiter来作仲裁，送至Slave。而Decoder是控制Slave送回数据到哪一个Master。





参考资料

胡振波.手把手教你设计CPU——RISC-V处理器（第1版）.人民邮电出版社
出版时间：2018-06-01. 第16章

顾长怡. 基于FPGA与RISC-V的嵌入式系统设计（第1版）.清华大学出版社
出版时间：2020-02-01. 第3章

玄铁E902用户手册

