**目录**

[AMBA 2](#_Toc426023210)

[什么是总线 2](#_Toc426023211)

[AMBA 2](#_Toc426023212)

[APB（ Advanced Peripheral Bus ）高级外设总线 3](#_Toc426023213)

[特性 3](#_Toc426023214)

[数据传输 3](#_Toc426023215)

[信号 5](#_Toc426023216)

[接口框图 5](#_Toc426023217)

[AHB（ Advanced High-performance Bus ）高级高性能总线 7](#_Toc426023218)

[特性 7](#_Toc426023219)

[简单传输 7](#_Toc426023220)

[信号 8](#_Toc426023221)

[接口框图 10](#_Toc426023222)

[AXI（Advanced eXtensible Interface）高级可扩展接口 13](#_Toc426023223)

[特性 13](#_Toc426023224)

[通道 13](#_Toc426023225)

[数据传输 13](#_Toc426023226)

[信号 14](#_Toc426023227)

[乱序传输 17](#_Toc426023228)

[SharkL的总线： 19](#_Toc426023229)

[Bus Monitor 20](#_Toc426023230)

[简介 20](#_Toc426023231)

[AXI Bus Monitor 21](#_Toc426023232)

[寄存器 21](#_Toc426023233)

[AHB Bus Monitor 26](#_Toc426023234)

[寄存器 26](#_Toc426023235)

[Bus Monitor功能 32](#_Toc426023236)

[带宽统计 32](#_Toc426023237)

[监控数据 34](#_Toc426023238)

[文件节点 36](#_Toc426023239)

[参考文献 40](#_Toc426023240)

# AMBA

## 什么是总线

总线（Bus）是计算机各种功能部件之间传送信息的公共通信干线，它是由导线组成的传输线束，按照计算机所传输的信息种类，计算机的总线可以划分为数据总线、地址总线和控制总线，分别用来传输数据、数据地址和控制信号。

总线是一种内部结构，它是cpu、内存、输入、输出设备传递信息的公用通道，主机的各个部件通过总线相连接，外部设备通过相应的接口电路再与总线相连接，从而形成了计算机硬件系统。

在计算机系统中，各个部件之间传送信息的公共通路叫总线，微型计算机是以总线结构来连接各个功能部件的。

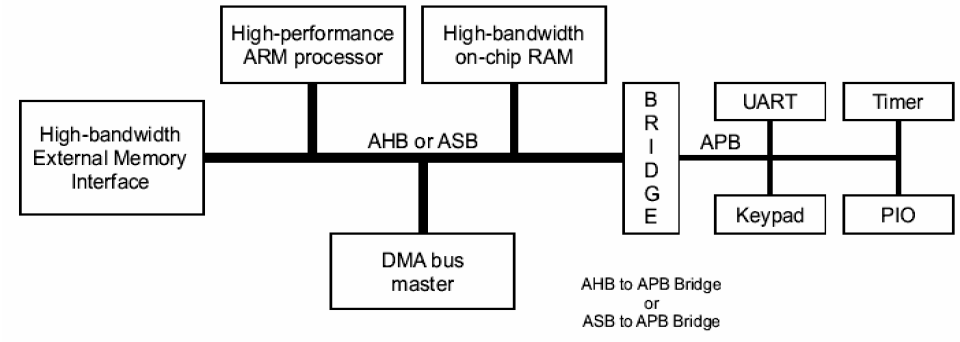
## AMBA

高级微控制器总线结构(Advanced Microcontroller Bus Architecture)

AMBA总线是由ARM公司提出的一种开放性的片上总线标准，规范定义了在设计高性能嵌入式微控制器时的一种片上通信标准，它独立于处理器和工艺技术，具有高速度低功耗等特点，受到集成电路设计行业的广泛欢迎。

主要的总线规范有：

* 高级外设总线（Advanced Peripheral Bus, APB）
* 高级系统总线（Advanced System Bus, ASB）
* 高级高性能总线（Advanced High-performance Bus, AHB）
* 高级可扩展接口（Advanced eXtensible Interface, AXI）



一个以AMBA架构的SoC，一般包含高性能的系统总线AHB和低功耗的外设总线APB。

系统总线（AHB）负责连接如嵌入式处理器、DMA控制器、片上存储器和其他外设接口，或者其他需求高带宽的元件。

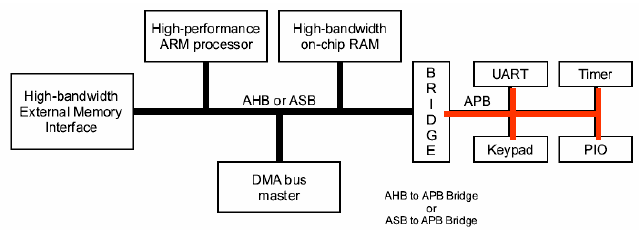
而外设总线（APB）则用来连接系统周边的外部设备，其协议相对AHB较为简单。

AHB与APB之间通过桥接器（Bridge）互联。

## APB（ Advanced Peripheral Bus ）高级外设总线

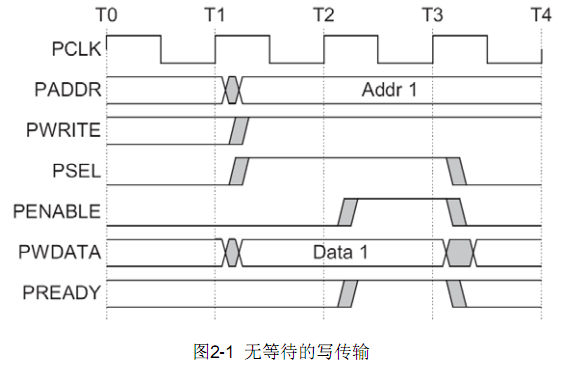
### 特性

* 用于连接低带宽的周边外设（UART、Keypad、Timer、PIO等）
* APB中只有一个Master—APB Bridge（与AHB Bus连接）
* 不需要Arbiter
* 非流水线操作
* 双周期传输
* 无等待周期和响应信号



### 数据传输

* **写数据**

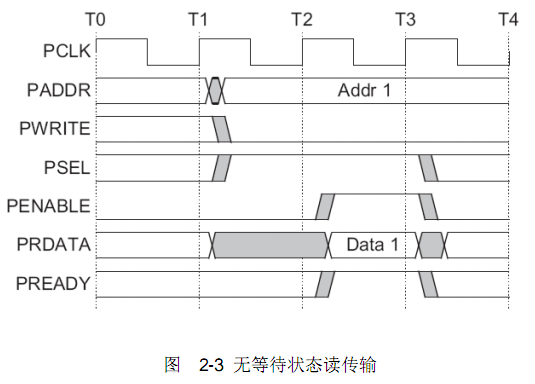


地址、写入数据、写入信号和选择信号都在时钟上升沿后改变。第一个时钟周期叫做Setup phase。

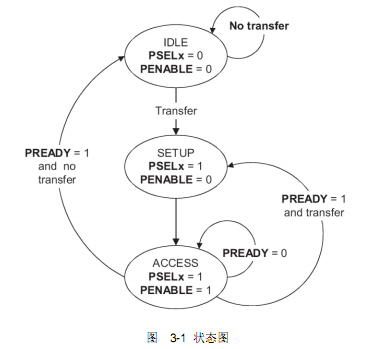
下一个时钟沿后使能信号PENABLE被置位，表示Access phase就位。地址、数据和控制信号在Access phase期间有效。传输在该周期后结束。

使能信号PENABLE, 在传输结束后清空。选择信号PSELx同样被置低，除非紧接着下一传输开始。

* **读数据**



* **状态机**

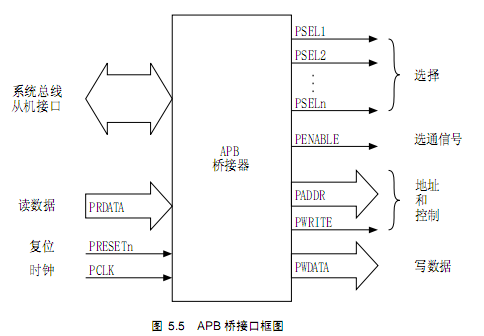


### 信号

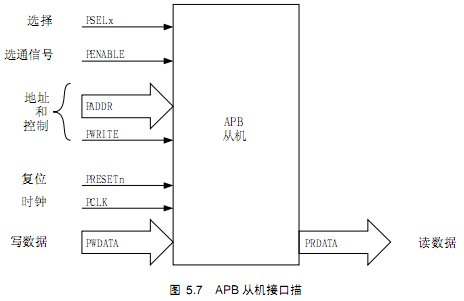


### 接口框图

* **Bridge**



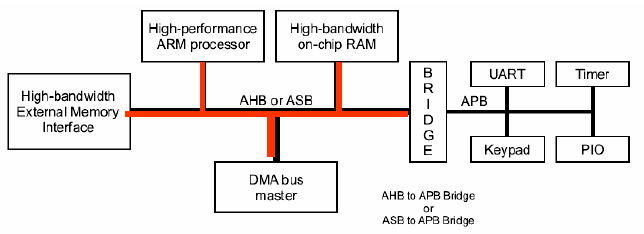
* **Slave**



## AHB（ Advanced High-performance Bus ）高级高性能总线

### 特性

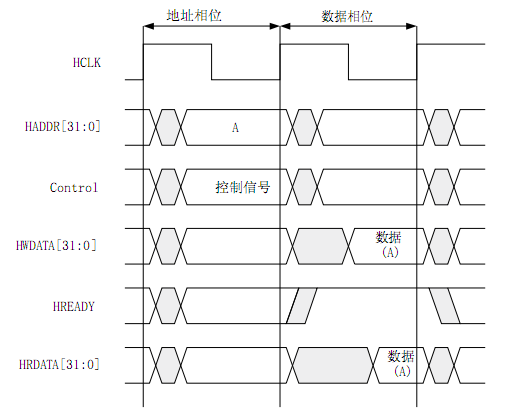
* 高速、高性能总线
* Pipeline操作
* 支持多个Master（最多16个）
* 支持Burst传输
* 支持Split传输
* Non-tristate Bus
* 单时钟边沿操作----上升沿触发操作



### 简单传输

AHB传输包含两个截然不同的部分：

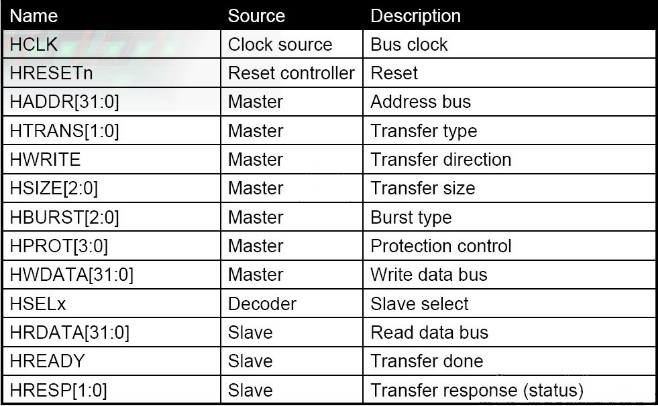
* 地址相位，只持续单个周期；
* 数据相位，可能需要多个周期。这通过使用HREADY信号实现。



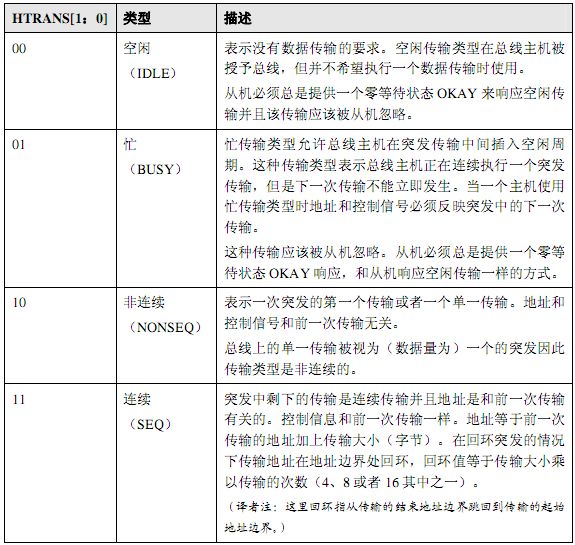
在这个没有等待状态的简单传输中：

* 主机在HCLK的上升沿之后将地址和控制信号驱动到总线上；
* 然后在时钟的下一个上升沿从机采样地址和控制信息；
* 在从机采样了地址和控制信号后能够开始驱动适当的响应并且该响应被总线主机在第三个时钟的上升沿采样。

### 信号



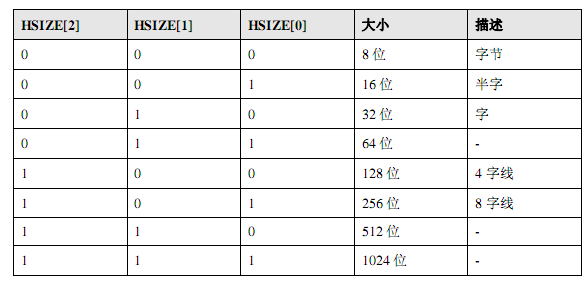
* **HTRANS**



* **HBURST**



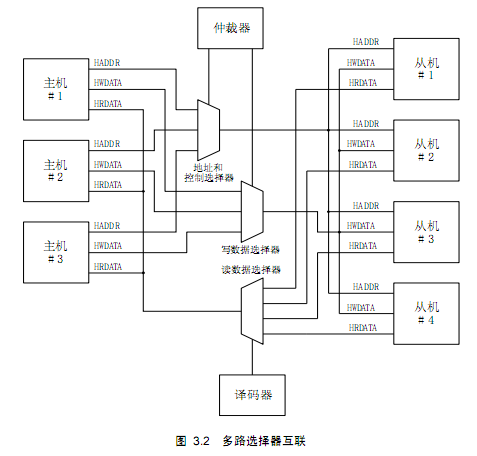
* **HSIZE**



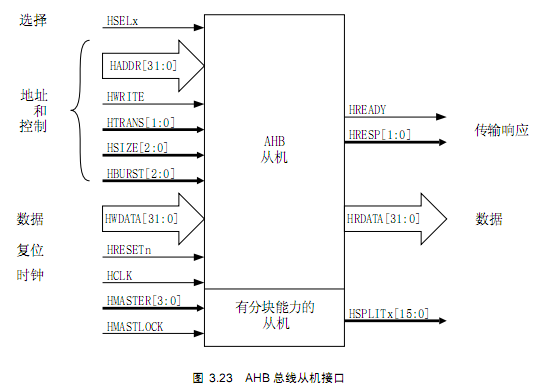
### 接口框图

* **总线互联**

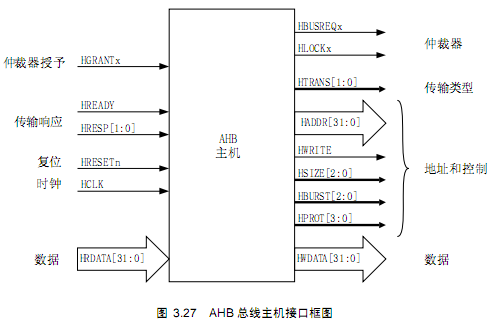
AMBA AHB总线协议设计使用一个中央多路选择器互联方案。使用该方案所有总线主机设备输出地址和控制信号来指示它们相执行的传输同时仲裁器决定哪一个主机能够将它的地址和控制信号连通到所有的从机。当然也需要一个译码器来控制读数据和响应多路信号选择器，多路信号选择器选中来自传输中所包含从机的适当信号。



* **AHB总线从机**

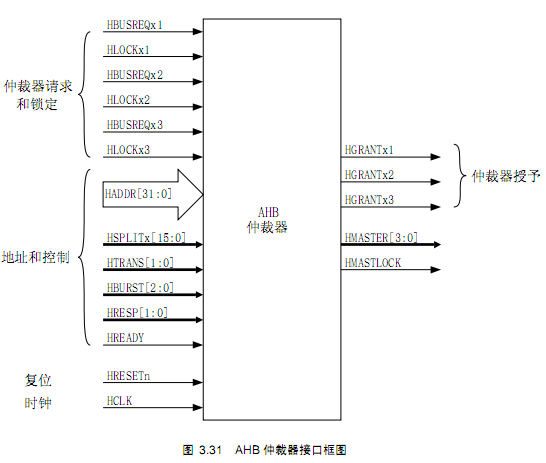


* **AHB总线主机**



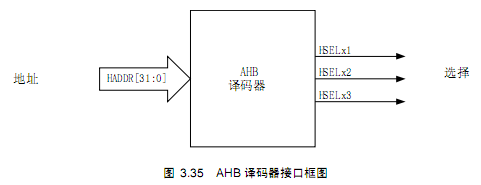
* **AHB仲裁器**

用于决定哪一个主机能够将它的地址和控制信号连通到所有的从机



* **AHB译码器**

用于决定哪个从机来响应这次信号



## AXI（Advanced eXtensible Interface）高级可扩展接口

### 特性

* 高性能、高带宽、低延迟的片内总线
* 单向通道体系结构
* 支持多项数据交换
* 独立的地址和数据通道

### 通道

AXI总线共有5个通道：

* 读地址通道（Read address channel, AR）
* 写地址通道（Write address Channel, AW）
* 读数据通道（Read data channel, R）
* 写数据通道（Write data channel, W）
* 写响应通道（Write response channel, B）

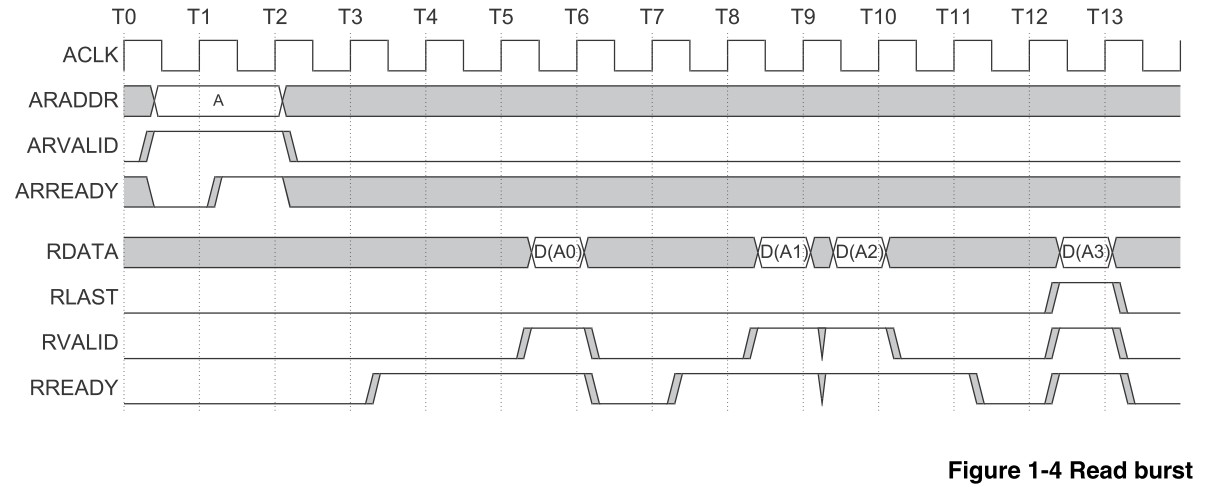
这5条独立的通道都包含一个信息信号和一个双路的VALD、READY握手机制。

信息源通过VALID信号来指示通道中的数据和控制信息什么时候有效。目地源用READY信号来表示何时能够接收数据。读数据和写数据通道都包括一个LAST信号，用来指明一个事物传输的最后一个数据。

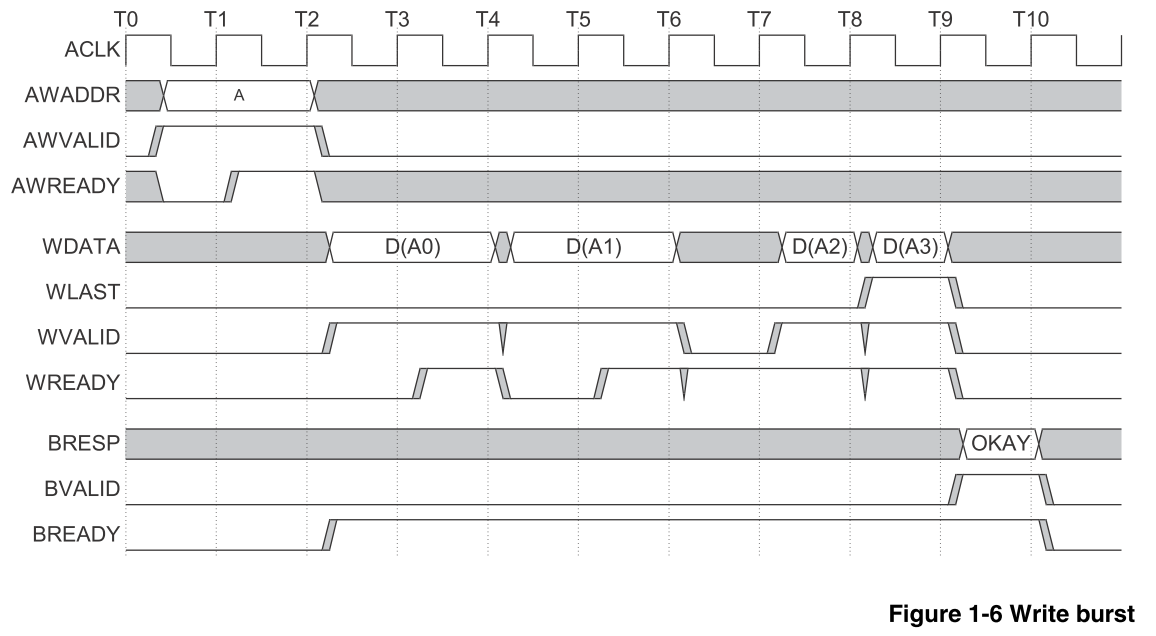
传输地址信息和数据都是在VALID和READY同时为高时有效。

### 数据传输

* **读事务时序**

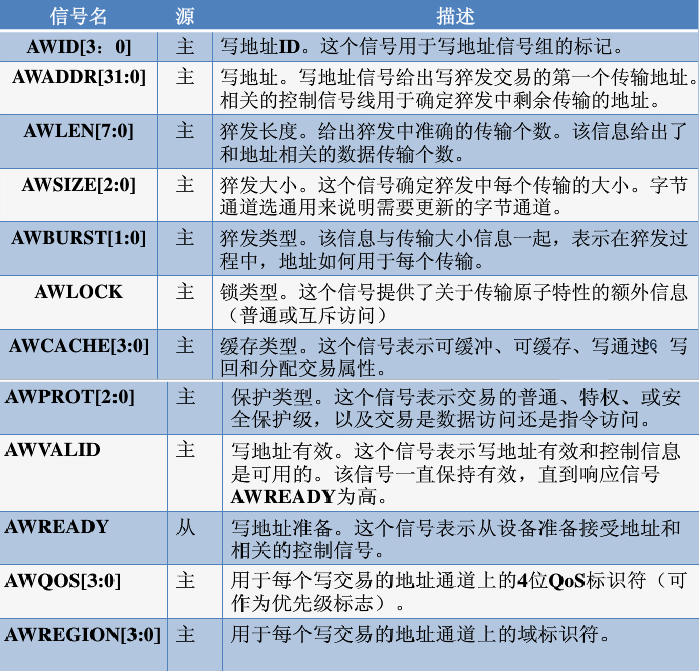


* **写事件时序**



### 信号

* **写地址通知信号**



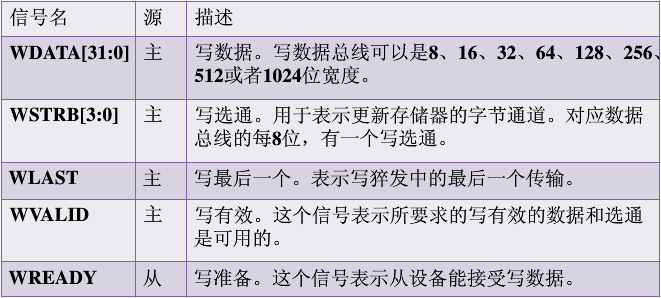
* **读地址通道信号**



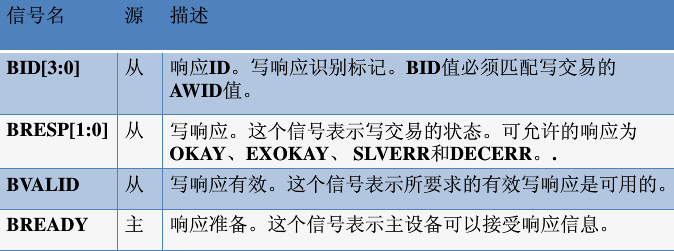
* **读数据通道信号**



* **写数据通道信号**

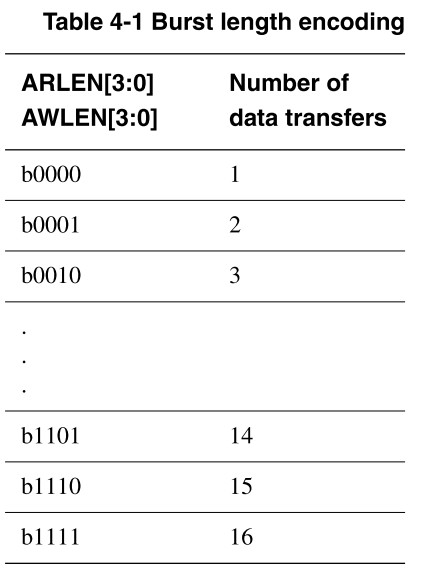


* **写响应通道信号**



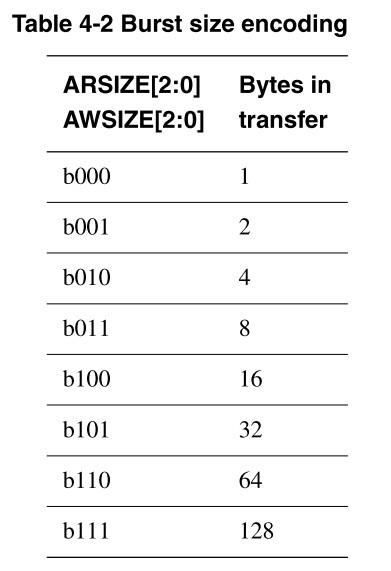
* **AWLEN/ARLEN**

指定每一次突发式读写所传输的数据的个数。

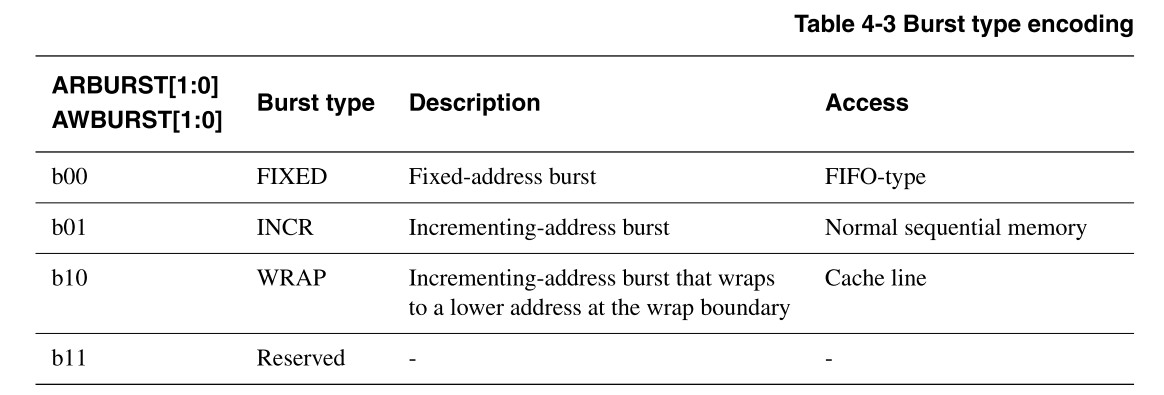


* **AWSIZE/ARSIZE**

指定每一个时钟节拍所传输的数据的最大位数。



* **AWBURST/ARBURST**



### 乱序传输

AXI协议支持乱序传输。他给每一个通过接口的事务一个IDtag。协议要求相同ID tag的事务必须有序完成，而不同ID tag可以乱序完成。

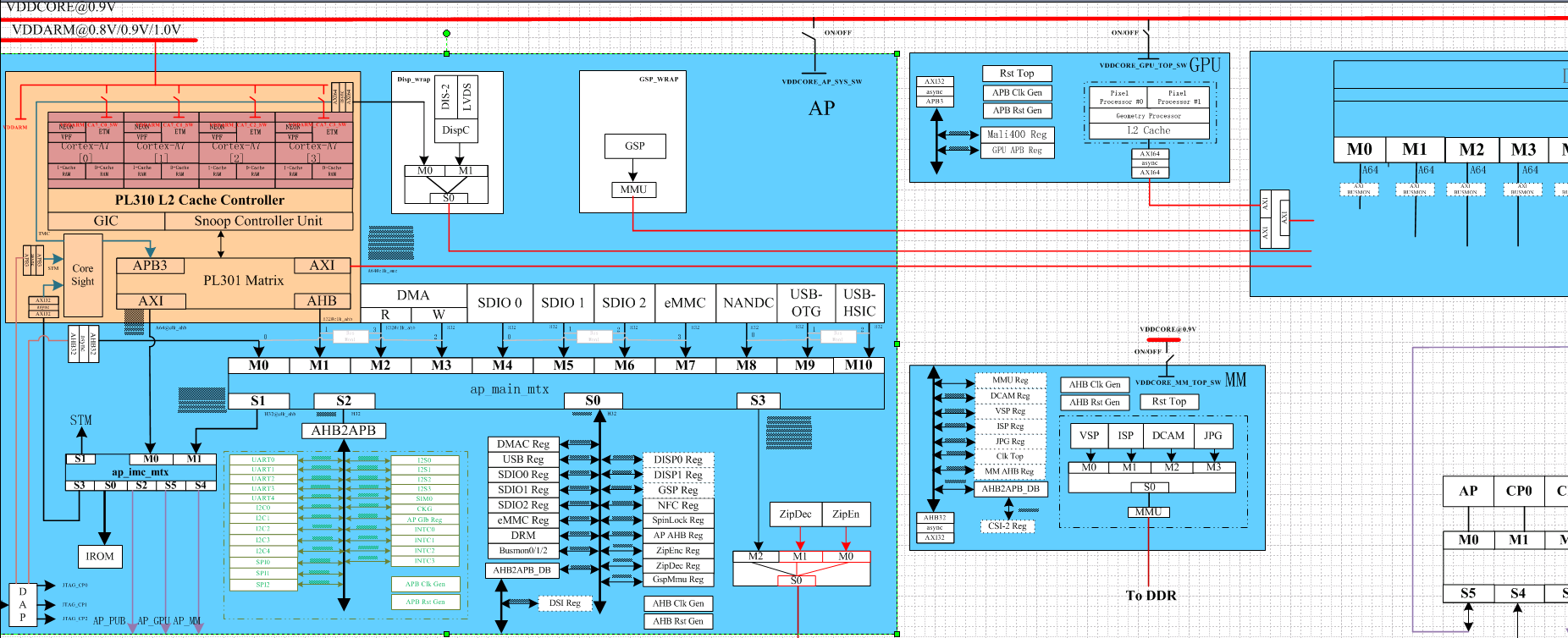
1. 下面介绍5中事务IDs：
2. AWID 这个ID tag是写地址群组信号。
3. WID 这个是写ID tag在写事务中，与写数据在一起，主机传送一个WID去匹配与地址相一致的AWID。
4. BID 这个ID tag是写响应事务中。设备会传送BID去匹配与AWID和WID相一致的事务。
5. ARID 这个ID tag是读地址群组信号。
6. RID 这个ID tag是在读事务中。设备传送RID去匹配与ARID相一致的事务。
7. 主机可以使用一个事务的ARID或者AWID段提供的附加信息排序主机的需要。事务序列规则如下：
8. 从不同主机传输的事务没有先后顺序限制。他们可以以任意顺序完成。
9. 从同一个主机传输的不同ID事务，也没有先后顺序限制。他们可以以任意顺序完成。
10. 相同数值的AWID写事务数据序列必须按照顺序依次写入主机发送的地址内。
11. 相同数值的ARID读事务数据序列必须遵循下面的顺序：

当从相同设备读相同的ARID时，设备必须确保读数据按照相同的地址顺序接受。

当从不同的设备读相同的ARID时，接口处必须确保读数据按照主机发送的相同的地址顺序。

1. 在相同的AWID和ARID的读事务和写事务之间没有先后顺序限制。如果主机要求有顺序限制，那么必须确保第一次事务完全完成后才开始执行第二个事务。

## SharkL的总线：



*（具体详见内网shicts1上各芯片的架构文档）*

# Bus Monitor

## 简介

Bus Monitor是一种监控设备，用于监控处理器与外部模块之间的总线操作(包括数据和地址的访问)，添加这个设备的主要目的主要是为了方便调试，其在系统中的位置可以用下面的简图说明：

总线Master

Memory/IO

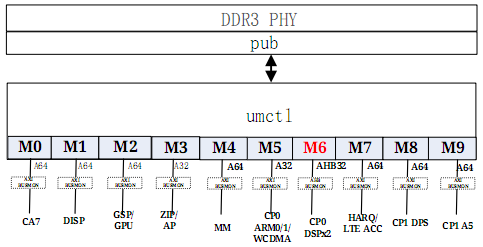
Bus Monitor

展讯芯片的Bus Monitor分为两类：AXI Bus Mon和AHB Bus Mon，顾名思义，是分别监测AXI和AHB总线的，但其实，并不是所有AXI和AHB总线上的活动都能被捕捉到，它有自己的局限性，AXI只是监测pub\_emc\_mem这个matrix上的AXI总线活动，AHB监测ap\_main\_mtx这个matrix上的AHB总线活动。

*注：以下内容，将以SharkLT8芯片，Adnroid5.1平台为例*

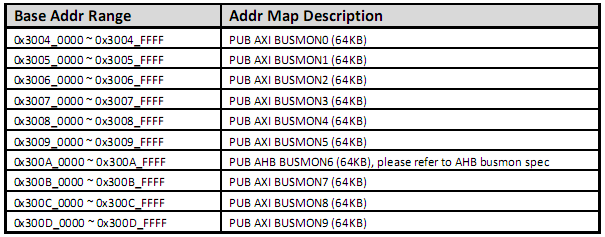
## AXI Bus Monitor

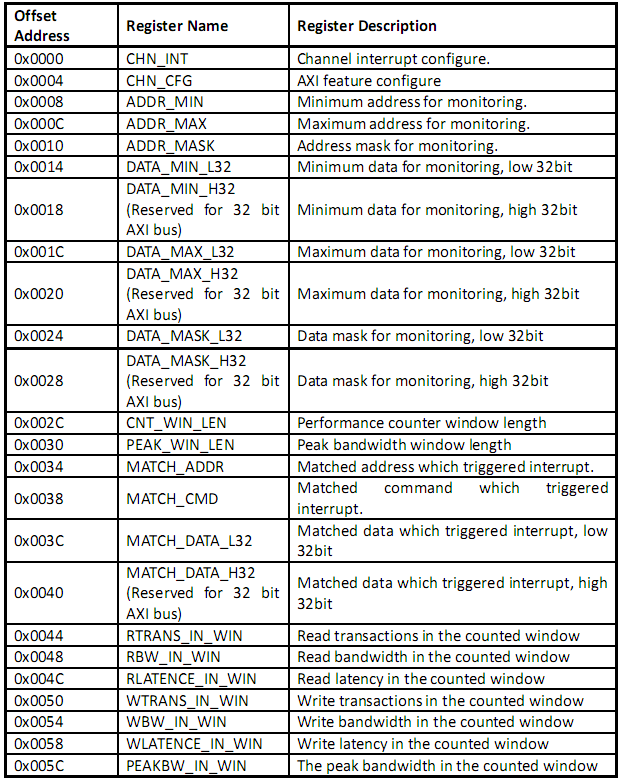
BUSMON\_AXI是对DDR访问的监控，可以对10个通道做分别配置



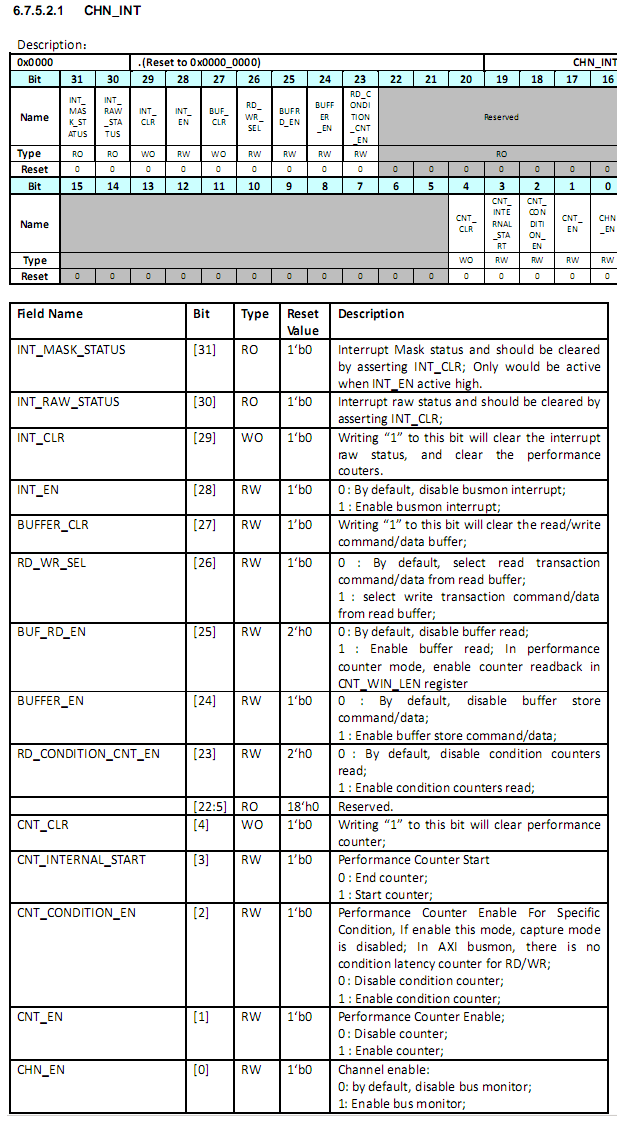
### 寄存器

* **寄存器 Address Map**

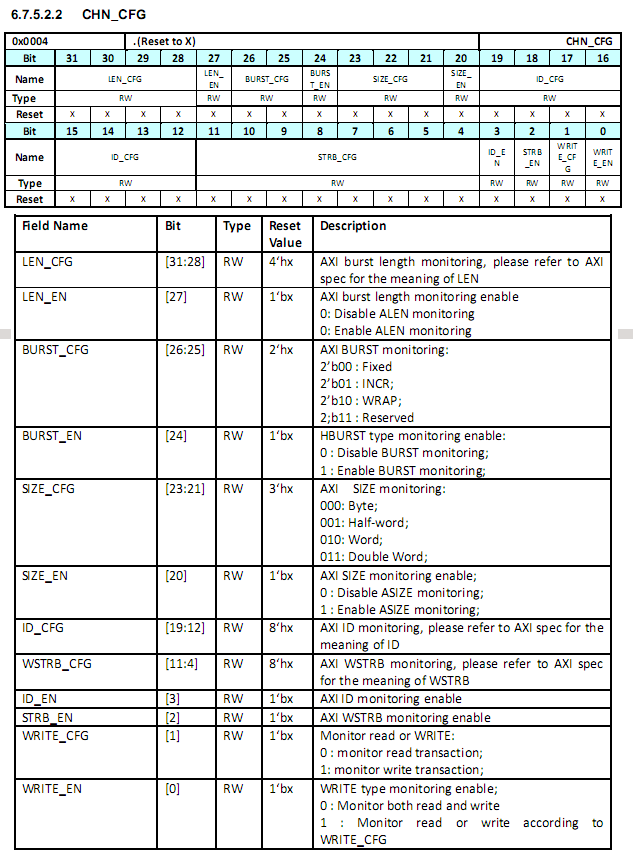




1. **CHN\_INT**

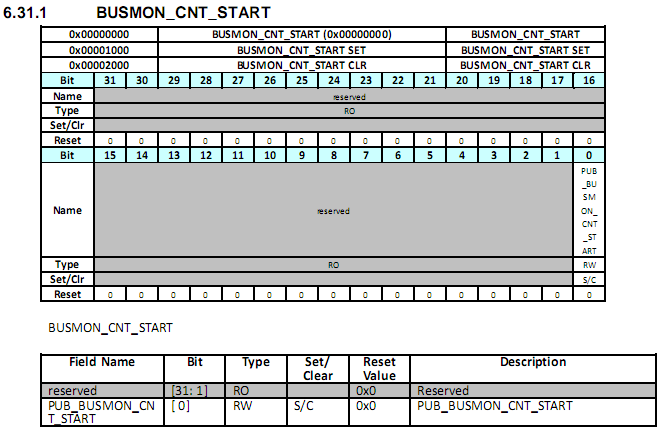


1. **CHN\_CFG**



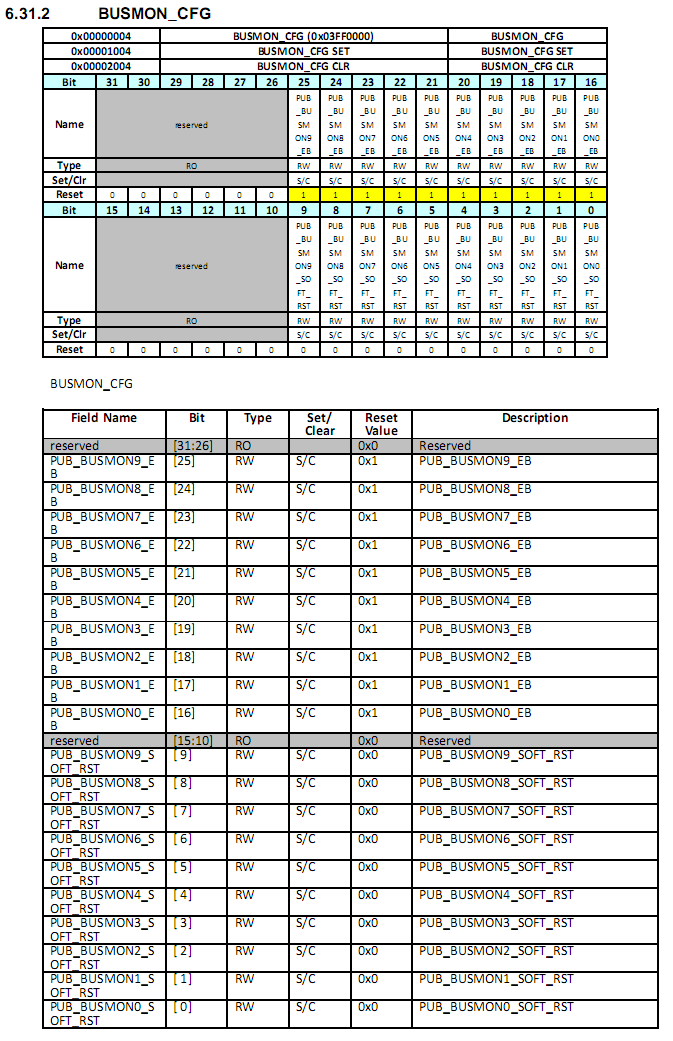
* **PUB Count Start**

寄存器：0x30020000

****

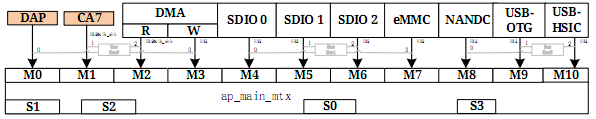
* **使能**

寄存器：0x30020004



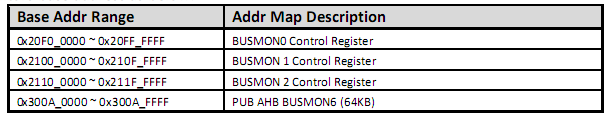
## AHB Bus Monitor

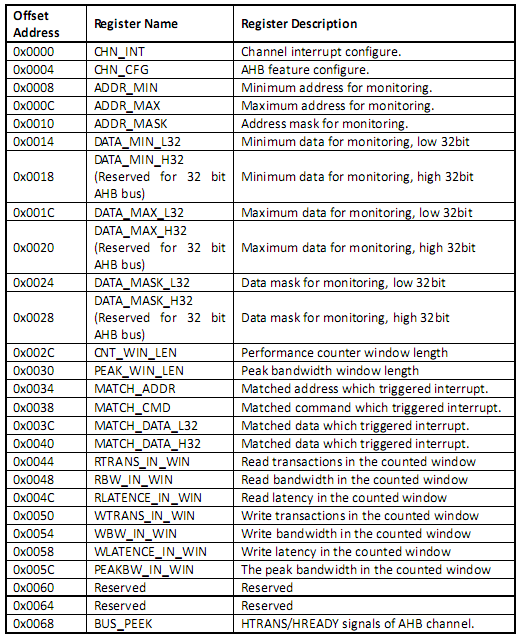
BUSMON\_AHB是对main matrix访问的监控，它将各个master分为三组，每个组监控其中一个通道



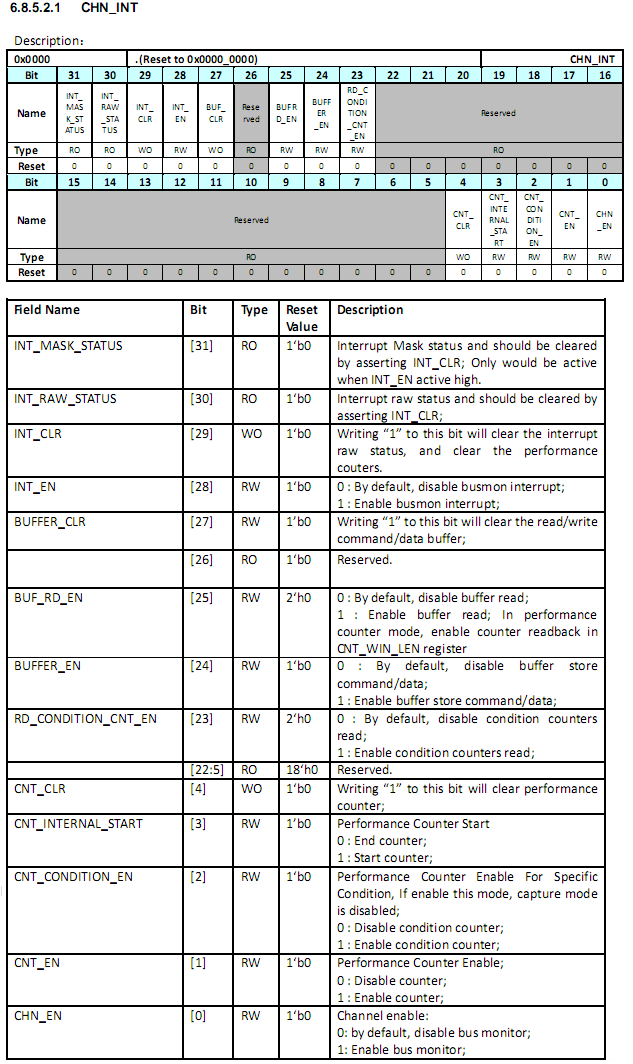
### 寄存器

* **寄存器 Address Map**

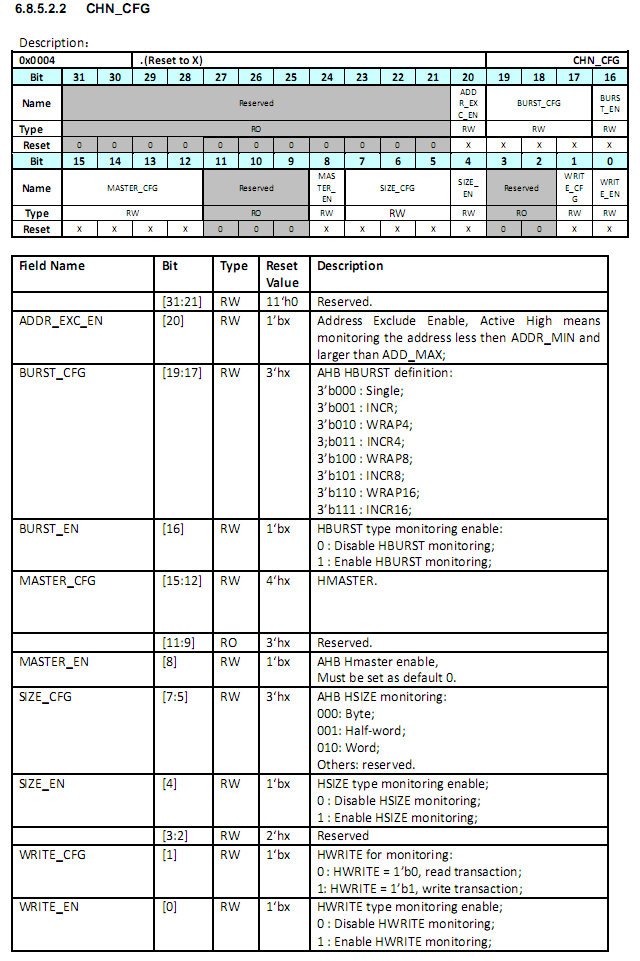




1. **CHN\_INT**

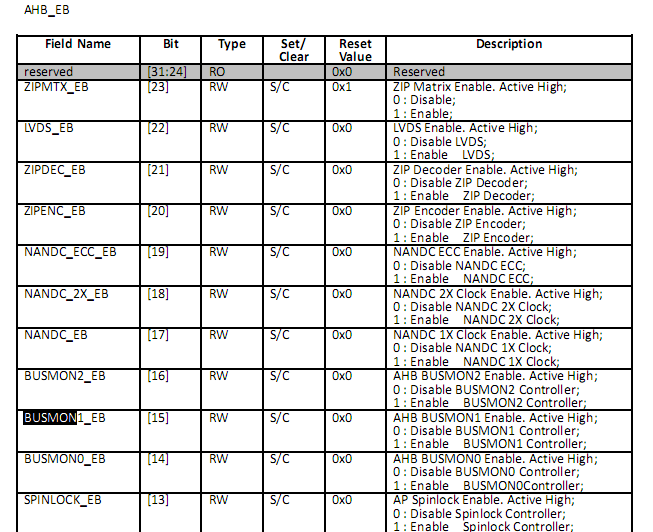


1. **CHN\_CFG**



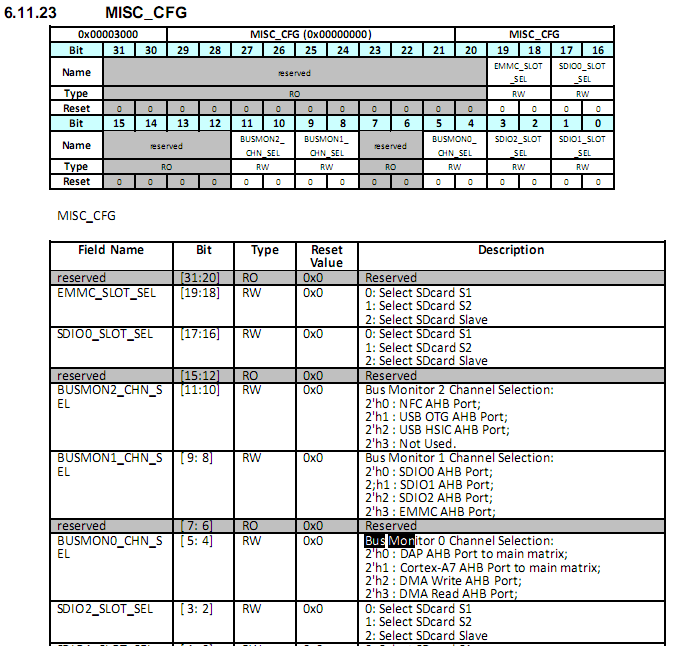
* **使能**

寄存器地址：0x20e00000



* **master通道选择**

寄存器：0x20e03000



## Bus Monitor功能

Bus Monitor的功能主要有：

* 带宽统计
* 监控数据

### 带宽统计

带宽数据导出功能主要用于DDR的数据分析，是DDR 优先级配置的重要依据。

#### 相关寄存器

设置：

CHN\_INT /CNT\_WIN\_LEN /PEAK\_WIN\_LEN

统计数据：

RTRANS\_IN\_WIN /RBW\_IN\_WIN /RLATENCE\_IN\_WIN /WTRANS\_IN\_WIN /WBW\_IN\_WIN /WLATENCE\_IN\_WIN /PEAKBW\_IN\_WIN

#### Bus Monitor统计带宽原理

用户通设置一个统计带宽时间S，Bus Monitor将这个参数设置至其WIN\_LEN reg中，每个S 时间单位Bus Monitor会触发中断，此时暂停Bus Monitor计数，读取当前的统计值，可获取当前S时间内的DDR访问量。如此，循环设置通过中断来获取每个S时间单位的访问量。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| t\_start | t\_stop | CA7 rt | CA7 rb | CA7 rl | CA7 wt | CA7 wb | CA7 wl |
| 83620 | 83630 | 8684 | 555776 | 53 | 3865 | 225536 | 9 |
| 83630 | 83640 | 1071 | 68544 | 56 | 645 | 31568 | 8 |
| 83640 | 83650 | 226 | 14464 | 58 | 515 | 23984 | 8 |
| 83650 | 83660 | 92 | 5888 | 57 | 496 | 22320 | 8 |
| 83660 | 83670 | 375 | 24000 | 54 | 563 | 26208 | 8 |
| 83670 | 83680 | 79 | 5056 | 60 | 494 | 22640 | 8 |

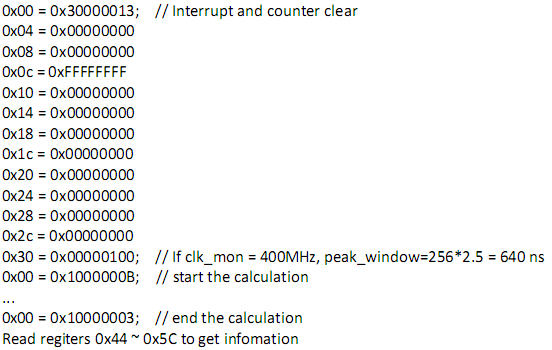
#### 使用方法

通过设置CNT\_INT.CNT\_EN=1来进入统计模式

1、手动启停时间窗

CNT\_WIN\_LEN = 0

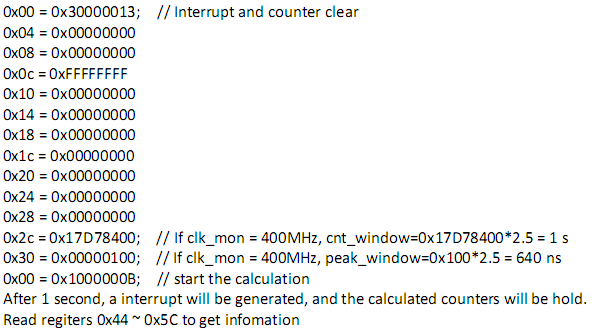
CHN\_INT.CNT\_INTERNAL\_START = 1开始统计，CHN\_INT.CNT\_INTERNAL\_START = 0结束统计



2、固定时间窗

CNT\_WIN\_LEN设定counter个数，时间跟clk\_mon有关

CHN\_INT.CNT\_INTERNAL\_START = 1开始统计



### 监控数据

Bus Monitor Debug 主要利用的 bus monitor 可以对总线行为进行监控的特点，如发现某个地址被异常修改，可用 bus monitor 将该地址空间监控起来，一旦发生写访问，立即触发中断，在中断中抓取现场数据。

#### 使用场景

Bus Monitor debug功能往往用在一些疑难杂症上，主要有以下场景时可以使用：

* 代码段被覆盖；
* 确定被异常覆盖的地址范围，确定master时；
* 确定被异常覆盖的地址范围，不确定master时；
* 寄存器被修改

对于每一个Bus Monitor而言，当其选定一个总线进行监测时，它具有以下功能：

* 监视总线上的地址
* 监视总线上的数据
* 条件触发ARM中断

#### 相关寄存器

设置：

CHN\_INT /CHN\_CFG /ADDR\_MIN /ADDR\_MAX /ADDR\_MASK /DATA\_MIN\_L32 /DATA\_MIN\_H32 /DATA\_MAX\_L32 /DATA\_MAX\_H32 /DATA\_MASK\_L32 /DATA\_MASK\_H32 /

结果：

MATCH\_ADDR /MATCH\_CMD /MATCH\_DATA\_L32 /MATCH\_DATA\_H32/ MATCH\_CMD

#### 局限性

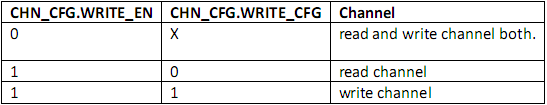
1. 不能一次监测全部的master，但至少可以排除是哪一个master引起的问题。也可以通过一次次地做实验，能够排查定位到master。
2. 通常情况下，DDR 都被映射成 cacheable 的，所以当 CPU 修改了 DDR 中的某个值之后，写 DDR 的行为不会被立即发射到总线上，而是在稍后的某个时刻执行，因此当 busmonitor 抓取到了特定的访问行为时，真正修改数据的现场可能已经不在了， 当然如果访问是由 DMA 或者其他 master 没有经过 cache 发起的，就不存在这个问题。
3. 并不是所有的寄存器都能被监控，只有所发命令的bus经过busmon才能被监控到。例如，对于master CA7，监控0x40000000~0x4fffffff这段地址是无效的，因为CA7对于这段地址内的访问，不会走ap\_main\_mtx这个matrix，而是走ap\_imc\_mtx，这个matrix上没有busmon模块；同样的，之前在tshark上想监控iram 0x00000000这段地址区域也是不行的，因为CA7走的也是ap\_imc\_mtx；但是对于其他master，如DMA/SDIO等，因为这些模块发起的访问必须要走ap\_main\_mtx，所以可以被监控到。（具体看芯片的架构图）

#### 使用方法

通过设置CNT\_INT.CNT\_EN=0来进入监控模式

**1、读写监控配置**

通过设置CHN\_CFG.WRITE\_EN和CHN\_CFG.WRITE\_CFG来控制监控的是读或/和写方式



**2、地址/数据配置**

通过配置ADDR\_MIN/ADDR\_MAX/ADDR\_MASK/DATA\_MIN\_L32/DATA\_MIN\_H32 /DATA\_MAX\_L32/DATA\_MAX\_H32/DATA\_MASK\_L32/DATA\_MASK\_H32这几个寄存器，可以控制监控的范围

**触发条件：**

DATA\_MIN ≤ data & (~DATA\_MASK) ≤ DATA\_MAX

ADDR\_MIN ≤ address & (~ADDR\_MASK) ≤ ADDR\_MAX

BUSMON\_AXI只要满足一条就触发。如果要监控某段地址的改写时，需要保证数据范围设定为非法，即data\_min > data\_max。

BUSMON\_AHB必要同时满足才触发。

**3、其他配置**

AXI/AHB总线有各自的信号特点，通过CHN\_CFG寄存器来配置监控条件

BUSMON\_AXI：LEN\_EN/ BURST\_EN/ SIZE\_EN/ ID\_EN/ STRB\_EN

BUSMON\_AHB：BURST\_EN/ MASTER\_EN/ SIZE\_EN

这个信号的含义，可以参考之前对总线的介绍

**4、举例**

监控AP写寄存器0x70000000~0x7fffffff区间的情况

设置：

lookat -s 0x0081C394 0x20e00000 // enable AHB busmon0

lookat -s 0x10000011 0x20e03000 // select CA7 port，这个需要特别注意，busmon0默认是DAP port

lookat -s 0x30000000 0x20f00000 // clear interrupt

lookat -s 0x00000003 0x20f00004 // write channel

lookat -s 0x70000000 0x20f00008 // min address

lookat -s 0x7fffffff 0x20f0000c // max address

lookat -s 0x00000000 0x20f00010 // address mask

lookat -s 0x00000000 0x20f00014 // min data low

lookat -s 0x00000000 0x20f00018 // min data high

lookat -s 0xffffffff 0x20f0001c // max data low

lookat -s 0x00000000 0x20f00020 // max data high

lookat -s 0x00000000 0x20f00024 // data mask low

lookat -s 0x00000000 0x20f00028 // data mask high

lookat -s 0x30000001 0x20f00000 // clear interrupt and start monitor

查看结果：

root@scx35l64\_ss\_sharklt8:/ # lookat -l4 0x20f00034

ADDRESS | VALUE

-----------+-----------

0x20f00034 | 0x71301000 // 被写的地址

0x20f00038 | 0x01800000 // 命令，根据定义，是write操作，位宽32bit

0x20f0003c | 0x00000200 // 被写入的值

0x20f00040 | 0x00000000

### 文件节点

如果通过寄存器来操作busmon，其实还是比较繁琐的，而且需要对busmon的原理特性要有一定的了解，为此，平台提供一个busmon的文件节点，通过交互式的操作，使使用变得更为简易。

* **代码**

kernel/drivers/platform/sprd/busmonitor.c

* **文件节点**

root@scx35l64\_ss\_sharklt8:/sys/devices/30040000.sprd\_bm # ll

-rw-r--r-- root root 4096 2012-01-01 09:00 ahb\_dbg

-rw-r--r-- root root 4096 2012-01-01 09:00 axi\_dbg

-rw-r--r-- root root 4096 2012-01-01 09:00 bandwidth

-rw-r--r-- root root 4096 2012-01-01 09:00 bus\_status

-rw-r--r-- root root 4096 2012-01-01 09:00 chn

-rw-r--r-- root root 4096 2012-01-01 09:00 continue

-rw-r--r-- root root 4096 2012-01-01 09:00 dfs\_off

-rw-r--r-- root root 4096 2012-01-01 09:00 disable

lrwxrwxrwx root root 2012-01-01 09:00 driver -> ../../bus/platform/drivers/sprd\_bm

-r--r--r-- root root 4096 2012-01-01 09:00 modalias

-rw-r--r-- root root 4096 2012-01-01 09:00 occur

-rw-r--r-- root root 4096 2012-01-01 09:00 panic

drwxr-xr-x root root 2012-01-01 08:58 power

-rw-r--r-- root root 4096 2012-01-01 09:00 stack

-rw-r--r-- root root 4096 2012-01-01 09:00 state

lrwxrwxrwx root root 2012-01-01 09:00 subsystem -> ../../bus/platform

-rw-r--r-- root root 4096 2012-01-01 08:58 uevent

* **ahb\_dbg**

设置BUSMON\_AHB的监控值

可读可写

**读：**

root@scx35l64\_ss\_sharklt8:/sys/devices/30040000.sprd\_bm # cat ahb\_dbg

0 0x70000000 0x7FFFFFFF 0x00000000 0xFFFFFFFF AP CPU

1 0x00000000 0x00000000 0x00000000 0x00000000 AP SDIO 0

2 0x00000000 0x00000000 0x00000000 0x00000000 AP NANDC

**写：**

格式：sscanf(buf, "%ld %ld %s %s %s %s", &channel, &chn\_sel, addr\_start, addr\_end, data\_min, data\_max);

示例：echo 0 0 0x30040000 0x3004005c 0x0 0xffffffff > ahb\_dbg

* **axi\_dbg**

设置BUSMON\_AXI的监控值

可读可写

**读：**

root@scx35l64\_ss\_sharklt8:/sys/devices/30040000.sprd\_bm # cat axi\_dbg

0 0x00000000 0x00000000 CPU

1 0x00000000 0x00000000 DISP

2 0x00000000 0x00000000 GPU

3 0x00000000 0x00000000 AP/ZIP

4 0x00000000 0x00000000 MM

5 0x8E900000 0x9ED73FFF CP0

6 0x8E900000 0x9ED73FFF CP0 DSP

7 0x8E900000 0x9ED73FFF CP1 LTEACC/HARQ

8 0x8E900000 0x9ED73FFF CP1 DSP

9 0x8E900000 0x9ED73FFF CP1

**写：**

格式：sscanf(buf, "%s %s %s %s",chn, start, end, mod);

示例：echo 0 0x89600000 0x90000000 rw > axi\_dbg

* **bandwidth**

进入带宽统计模式

只写

* **bus\_status**

AXI总线的状态

只读

* **chn**

列出所有channel

只读

1|root@scx35l64\_ss\_sharklt8:/sys/devices/30040000.sprd\_bm # cat chn

CPU

DISP

GPU

AP/ZIP

MM

CP0

CP0 DSP

CP1 LTEACC/HARQ

CP1 DSP

CP1

AP DAP

AP CPU

AP DMA READ

AP DMA WRITE

AP SDIO 0

AP SDIO 1

AP SDIO 2

AP EMMC

AP NANDC

AP OTG

AP HSIC

* **continue**

设置连续监测

可读可写

* **dfs\_off**

dfs统计开关

可读可写

* **disable**

disable/enable AXI BUSMON

只写

* **occur**

显示命中的情况

只读

root@scx35l64\_ss\_sharklt8:/sys/devices/30040000.sprd\_bm # cat occur

AP DAP

addr: 0x71301000

CMD: 0x1800000

msk\_data\_l: 0x200

msk\_data\_h: 0x0

id 0x0

* **panic**

命中时是否要panic

可读可写

* **stack**

命中时是否要打印调用栈

可读可写

* **state**

BUSMON当前所处的状态

只读

root@scx35l64\_ss\_sharklt8:/sys/devices/30040000.sprd\_bm # cat state

Bus Monitor in debug mode!

# 参考文献

1. [AMBA](http://wenku.baidu.com/view/d80cf604cc175527072208f1.html)
2. [AMBA协议规范](http://wenku.baidu.com/view/9915b841f12d2af90342e63e.html)
3. 《SC9838A Device Specification(external) - all.pdf》