SAD Architecture Specification

腾讯蓬莱实验室

1 Architecture Overview

SAD(Signed Absolute Differences)实现 16x16 的 8bit 矩阵对应有符号数值之差的绝对值求和，最大支持连续 16 个计算。

公式表示如下：

$$sad(u,v) = Sum\{|Left(u,v) - Right(u,v)|\}$$

通过APB配置并启动SAD计算，通过类AXI接口写入计算内容，计算完成后当cal_ready拉高时将结果输出。

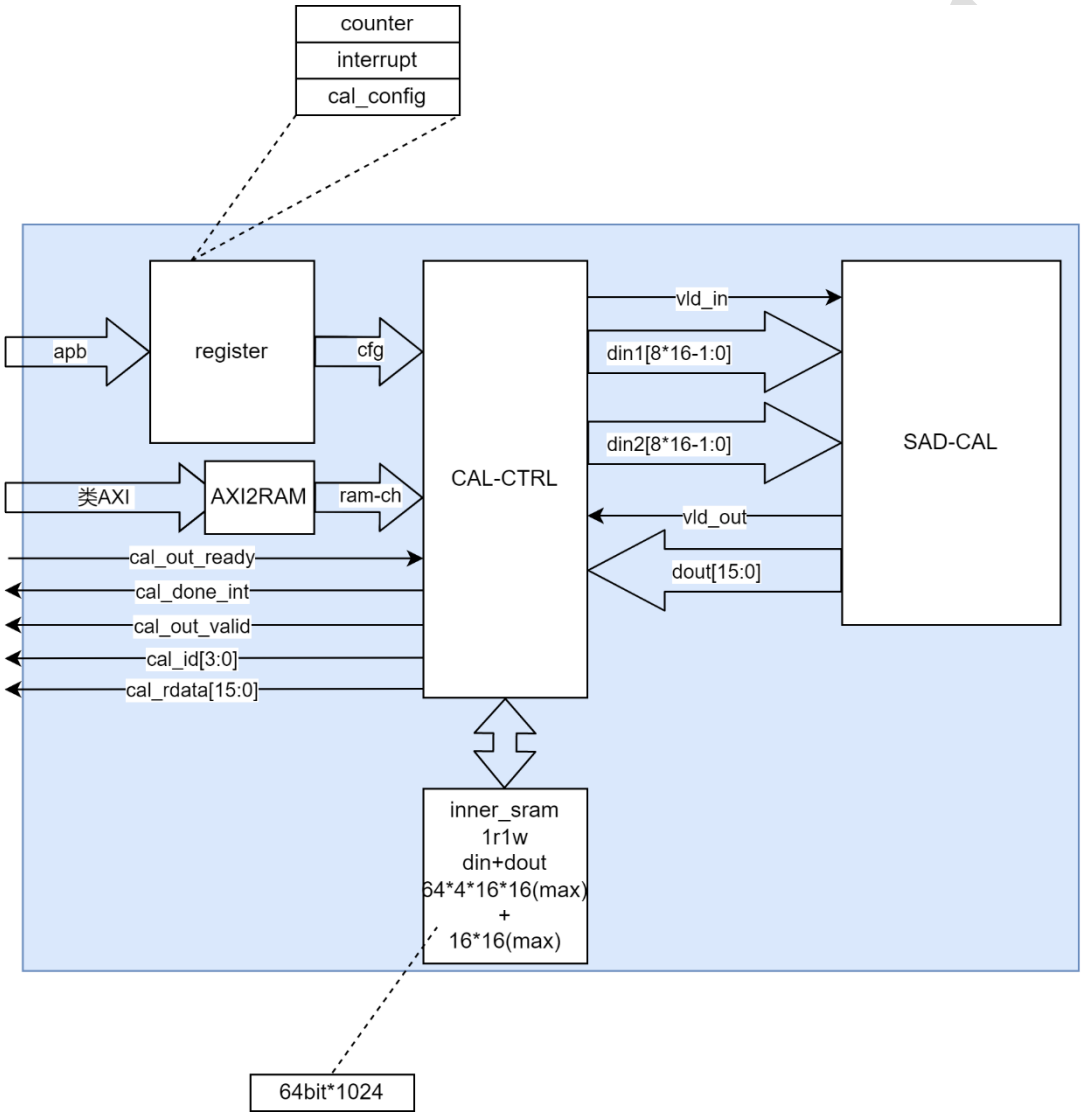

图1-1 VT100 Block Diagram

2 Information flow

2.1 Config information flow

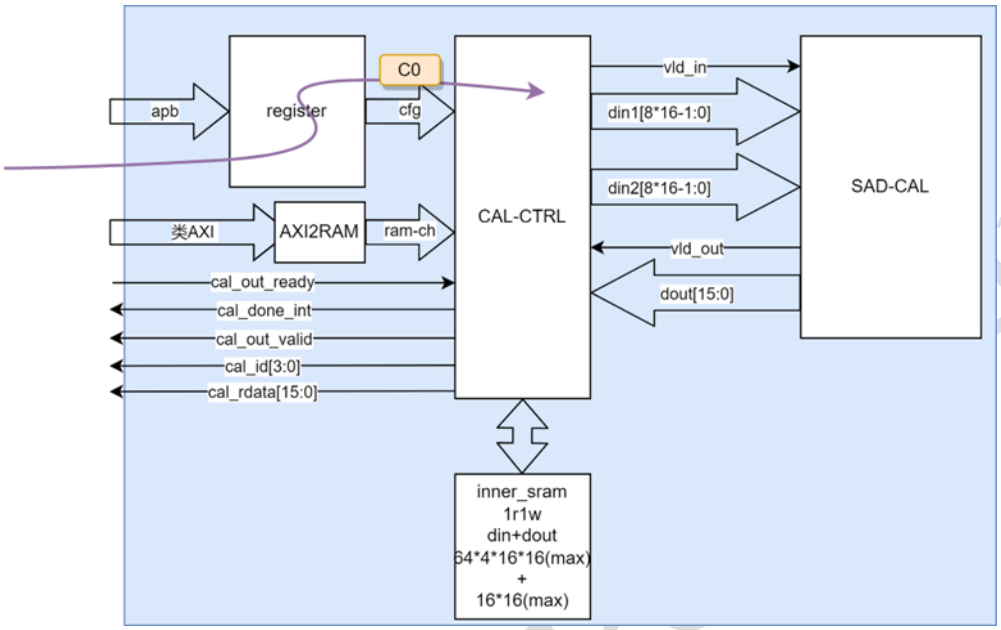

图 2- Config information flow

2.2 Data information flow

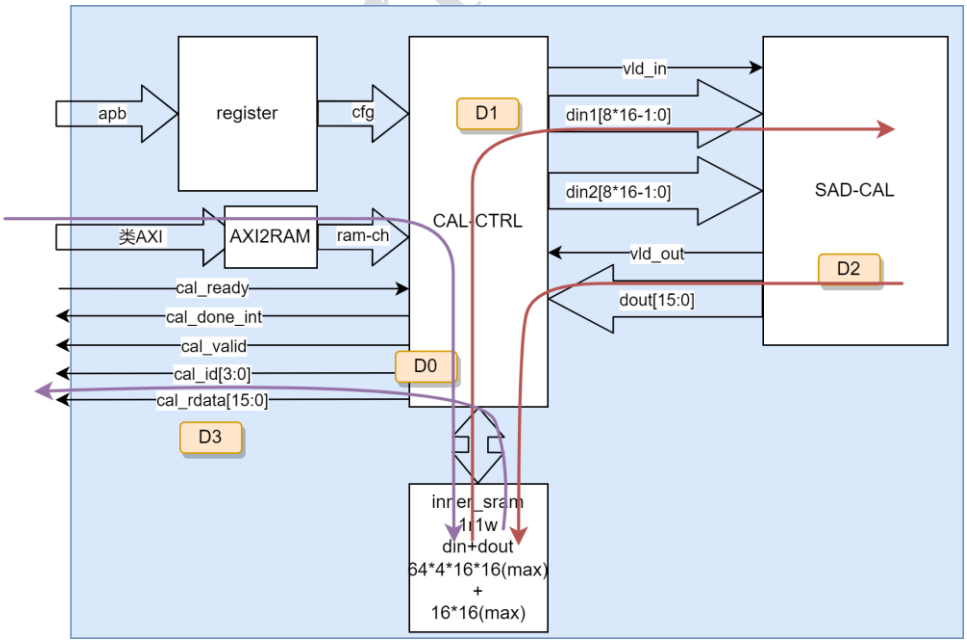

图2-2 Data information flow

3 data structure

Inner_sram的数据结构如下表:

表3-1 数据结构的汇总表单

axi addr	data structure	data size（bit）	description	inner addr
0x0000	g0_din1_l[0]	64	第一组第一行计算的被减数低64bit输入	0x000
0x0008	g0_din1_h[0]	64	第一组第一行计算的被减数高64bit输入	0x001
0x0010	g0_din1_l[1]	64	第一组第二行计算的被减数低64bit输入	0x002
0x0018	g0_din1_h[1]	64	第一组第二行计算的被减数高64bit输入	0x003
…	…			
0x00E0	g0_din1_l[14]	64	第一组第十五行计算的被减数低64bit输入	0x01c
0x00E8	g0_din1_h[14]	64	第一组第十五行计算的被减数高64bit输入	0x01d
0x00F0	g0_din1_l[15]	64	第一组第十六行计算的被减数低64bit输入	0x01e
0x00F8	g0_din1_h[15]	64	第一组第十六行计算的被减数高64bit输入	0x01f
…	…			
0x0100	g1_din1_l[0]	64	第二组第一行计算的被减数低64bit输入	0x020
0x0108	g1_din1_h[0]	64	第二组第一行计算的被减数高64bit输入	0x021
0x0110	g1_din1_l[1]	64	第二组第二行计算的被减数低64bit输入	0x022
0x0118	g1_din1_h[1]	64	第二组第二行计算的被减数高64bit输入	0x023
…	…			
0x01E0	g1_din1_l[14]	64	第二组第十五行计算的被减数低64bit输入	0x03c
0x01E8	g1_din1_h[14]	64	第二组第十五行计算的被减数高64bit输入	0x03d
0x01F0	g1_din1_l[15]	64	第二组第十六行计算的被减数低64bit输入	0x03e
0x01F8	g1_din1_h[15]	64	第二组第十六行计算的被减数高64bit输入	0x03f
…	…			

0x0F00	g15_din1_l[0]	64	第十六组第一行计算的被减数低64bit输入	0x1e0
0x0F08	g15_din1_h[0]	64	第十六组第一行计算的被减数高64bit输入	0x1e1
0x0F10	g15_din1_l[1]	64	第十六组第二行计算的被减数低64bit输入	0x1e2
0x0F18	g15_din1_h[1]	64	第十六组第二行计算的被减数高64bit输入	0x1e3
…	…			
0x0FE0	g15_din1_l[14]	64	第十六组第十五行计算的被减数低64bit输入	0x1fc
0x0FE8	g15_din1_h[14]	64	第十六组第十五行计算的被减数高64bit输入	0x1fd
0x0FF0	g15_din1_l[15]	64	第十六组第十六行计算的被减数低64bit输入	0x1fe
0x0FF8	g15_din1_h[15]	64	第十六组第十六行计算的被减数高64bit输入	0x1ff
…	…			
0x1000	g0_din2_l[0]	64	第一组第一行计算的减数低64bit输入	0x200
0x1008	g0_din2_h[0]	64	第一组第一行计算的减数高64bit输入	0x201
0x1010	g0_din2_l[1]	64	第一组第二行计算的减数低64bit输入	0x202
0x1018	g0_din2_h[1]	64	第一组第二行计算的减数高64bit输入	0x203
…	…			
0x10E0	g0_din2_l[14]	64	第一组第十五行计算的减数低64bit输入	0x21c
0x10E8	g0_din2_h[14]	64	第一组第十五行计算的减数高64bit输入	0x21d
0x10F0	g0_din2_l[15]	64	第一组第十六行计算的减数低64bit输入	0x21e
0x10F8	g0_din2_h[15]	64	第一组第十六行计算的减数高64bit输入	0x21f
…	…			
0x1100	g1_din2_l[0]	64	第二组第一行计算的减数低64bit输入	0x220
0x1108	g1_din2_h[0]	64	第二组第一行计算的减数高64bit输入	0x221
0x1110	g1_din2_l[1]	64	第二组第二行计算的减数低64bit输入	0x222
0x1118	g1_din2_h[1]	64	第二组第二行计算的减数高64bit输入	0x223

…	…			
0x11E0	g1_din2_l[14]	64	第二组第十五行计算的减数低64bit输入	0x23c
0x11E8	g1_din2_h[14]	64	第二组第十五行计算的减数高64bit输入	0x23d
0x11F0	g1_din2_l[15]	64	第二组第十六行计算的减数低64bit输入	0x23e
0x11F8	g1_din2_h[15]	64	第二组第十六行计算的减数高64bit输入	0x23f
…	…			
0x1F00	g15_din2_l[0]	64	第十六组第一行计算的减数低64bit输入	0x3e0
0x1F08	g15_din2_h[0]	64	第十六组第一行计算的减数高64bit输入	0x3e1
0x1F10	g15_din2_l[1]	64	第十六组第二行计算的减数低64bit输入	0x3e2
0x1F18	g15_din2_h[1]	64	第十六组第二行计算的减数高64bit输入	0x3e3
…	…			
0x1FE0	g15_din2_l[14]	64	第十六组第十五行计算的减数低64bit输入	0x3fc
0x1FE8	g15_din2_h[14]	64	第十六组第十五行计算的减数高64bit输入	0x3fd
0x1FF0	g15_din2_l[15]	64	第十六组第十六行计算的减数低64bit输入	0x3fe
0x1FF8	g15_din2_h[15]	64	第十六组第十六行计算的减数高64bit输入	0x3ff
…	…			
0x2000	{g3_dout,…,g0_dout}	64	第4/3/2/1组计算结果	0x400
…	…			
0x2018	{g15_dout,…,g12_dout}	64	第16/15/14/13组计算结果	0x403
0x2020	reserved			

4 Interface

4.1 Extern Interface

4.1.1 类AXI

- 时序参考axi协议
- 只保留AXI3中的AW、W、B通道，不支持AR、R通道（AR/R valid内部tie0）。若有debug和AXI协议理解的需求，可将DUT中AR和R通道中的握手信号打开（ready和vaild按逻辑接）
- 为了简化实现，仅要求实现single操作、数据长度为64bit、突发类型为FIXED、ready和vaild握手，无需支持busrt、outstanding、窄传输、非对齐、乱序访问等操作

4.1.2 APB

参考APB3。

4.1.3 其他顶层接口信号

表4-1 其他顶层接口信号

序号	接口名称	接口位宽	方向	接口描述
top<->cal_ctrl				
1	cal_ready	1	input	数据发出信号，高有效
2	cal_int_out	1	output	计算完成中断输出，脉冲信号
3	cal_valid	1	output	表示该对应组数的计算结果是否有效，高有效
4	cal_id	4	output	标识对应组数
5	cal_rdata	16	output	将计算结果按组输出

信号时序如下图所示，当cal_int_out信号拉高后，cal_radta会在下一拍发送数据，cal_id与cal_radta成对同时发出，只有cal_ready和cal_valid握手成功时的cal_id和cal_rdata才有效。

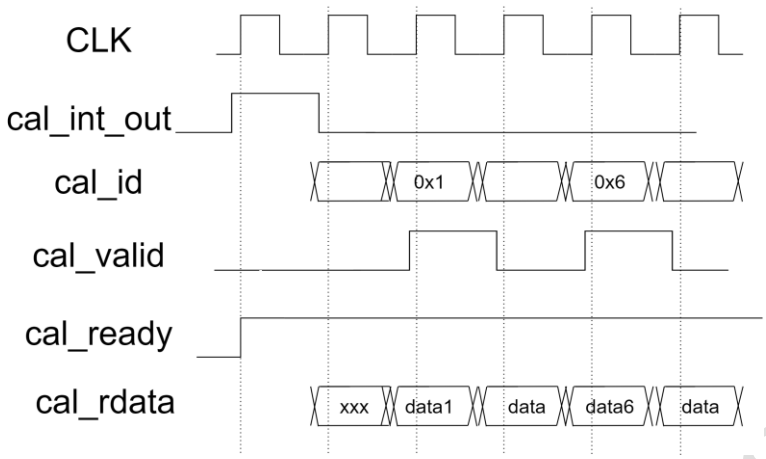

图4-1 其他顶层信号时序示意图

4 Internal Interface

表4-2 内部模块接口信号

序号	接口名称	接口位宽	方向	接口描述
register<->cal_ctrl				
1	cal_start	1	input	通过寄存器配置使能SAD计算开始脉冲，高有效，1cycle有效脉冲
2	cal_num	4	input	寄存器配置进来的当前一次SAD计算多少(cal_num+1)个SAD数据，最大16个数，再cal_start为1时有效
3	cal_busy_state	1	output	cal_ctrl模块处于忙状态标记，高有效，可用过寄存器往外读出对应状态
4	int_set_en	1	input	软件强制配置中断输出使能，为1时，中断输出为软件配置值int_set_value，为0时，中断输出为电路输出值
5	int_set_value	1	input	软件强制配置中断输出值，在int_set_en为1时有效
6	int_en	1	input	中断使能，为1时中断有效，为0时中断保持常0
7	int_state_rd	1	input	中断状态读取使能，用于中断状态输出清零，1：当前拍读取中断状态，0：当前拍无中断状态读取动作
8	int_state	1	output	中断状态输出
9	int_mask	1	input	中断输出mask，为1时，中断输出保持常0，为0时中断输出为经过中断使能后的中断值
10	total_cnt_flag	1	input	统计计数器是饱和还是非饱和计数器选择标志，1：饱和计数器，0：非饱和计数器

11	total_cnt_rd	1	input	统计计数器读取使能，用于统计计清零
12	total_cnt	16	output	统计计数器输出
13	cal_ready	1	input	计算结果输出使能,高有效
cal_ctrl<->sad_cal				
1	sad_vld_in	1	output	使能sad_cal计算使能信号，高有效
2	sad_din1	8*16	output	sad_cal计算第一个矩阵数据，在sad_vld_in为高时有效
3	sad_din2	8*16	output	sad_cal计算第二个矩阵数据，在sad_vld_in为高时有效
4	sad_vld_out	1	input	sad_cal计算结果有效使能，高有效
5	sad_dout	16	input	sad_cal计算结果（有符号数），在sad_vld_out为高时有效
cal_ctrl<->inner_sram（rf_1r1w_wrapper：64*66）				
1	inner_sram_we_n	1	output	写inner_sram的使能信号，低有效
2	inner_sram_waddr	11	output	写inner_sram的地址信息，在inner_sram_we_n为低时有效
3	inner_sram_wdata	64	output	写inner_sram的数据内容，在inner_sram_we_n为低时有效
4	inner_sram_rd_n	1	output	读inner_sram的使能信号，低有效
5	inner_sram_raddr	11	output	读inner_sram的地址信息，在inner_sram_rd_n为低时有效
6	inner_sram_rdata	64	input	读inner_sram的数据内容，在inner_sram_rd_n为低时有效
cal_ctrl<->axi2ram				
1	axi2ram_cs_n	1	input	类axi写ram片选使能，低有效
2	axi2ram_we_n	1	input	类axi写ram读写控制标值，0：写，在axi_ram_cs_n为低时有效
3	axi2ram_addr	11	input	类axi写ram地址信息，一共有64+4个64位宽的地址，在axi_ram_cs_n为低时有效
4	axi2ram_wdata	64	input	类axi写ram的数据内容，在axi_ram_cs_n为低并且axi_ram_we_n为低时有效

5 Sub module architecture

5.1 Cal_ctrl

模块实现以下两个基本功能：

- 写sram：接收axi2ram模块发送过来的写sram命令，将对应数据存入sram_din中
- 执行sad计算：接受register发送过来的start命令，从sram_din中读取对应数据后拼接成sad_cal模块所需输入，送入sad_cal进行计算后将得到的计算结果存入sram_dout中

5.1.1 类AXI写sram

根据axi2ram模块送过来的写信息，写sram对应位置即可。在cal_ctrl非IDLE状态时，写sram不成功。即要求master在模块busy时不进行sram的数据写动作。

5.1.2 SAD计算

在接收到register送过来的cal_start脉冲后，电路跳出IDLE状态，读取inner_sram中的计算系数，送到sad_cal模块进行计算，计算结束后将计算结果存入inner_sram中的相应位置。

电路状态跳转如下：

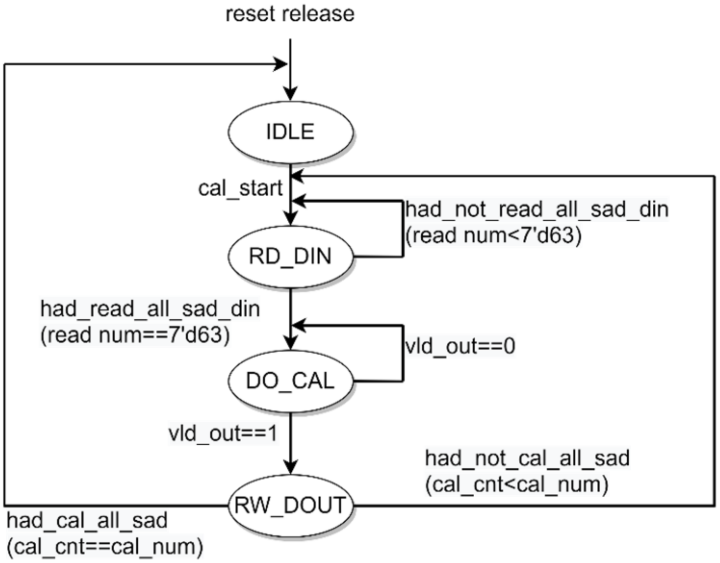

图5-1 SAD计算状态跳转图

图中：

- IDLE：电路处于空闲状态，此时cal_state状态寄存器标志为空闲
- RD_DIN：根据cal_cnt读取4个地址的inner_sram内容，拼接成sad输入系数

送到cal_sad模块进行计算

- DO_CAL：发送计算系数之后，等待计算结果
- RW_DOUT：接收到计算结果之后，根据此时的cal_cnt将结果回写如inner_sram对应位置，而后判断cal_cnt是否已经计算到cal_num个数，是的话则跳回IDLE并且cal_cnt清理，否的话则跳至RD_DIN计算下一个数据并且cal_cnt+1

5.1.3 中断处理电路

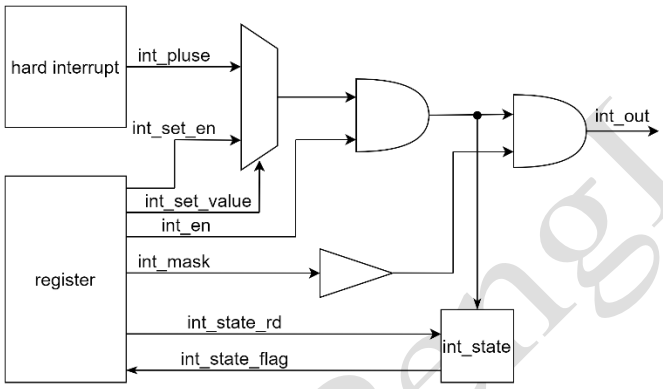

图5-2 中断处理电路图

5.2 Register

5.2.1 Register描述

地址	寄存器名称	field		RW/RO	描述
0x0	cal_start	31:1	reserved	RO	reserved
		0	start_en	WO	计算开始配置使能，写1启动计算，写0无效，写1后电路下一拍自动清零，回读结果一定是0x0
0x4	cal_num	31:4	reserved	RO	reserved
		3:0	quantity	RW	此次计算多少个SAD结果配置，个数为配置值加1
0x8	cal_state	31:1	reserved	RO	reserved
		0	is_busy	RO	当前模块是否处于计算忙状态，为1表示当前模块忙，为0表示当前模块空闲
0xC	int_set	31:2	reserved	RO	reserved

		1	set_value	RW	软件强制输出中断高低状态值
		0	set_enable	RW	软件强制输出中断使能，1表示当前中断状态强制为set_vale值，0为当前中断输出为电路正常状态
0x10	int_enable	31:1	reserved	RO	reserved
		0	enable	RW	中断输出使能，为1表示当前电路中断输出，为0时表示当前电路中断不输出(即使是int_set寄存器配置也不输出)
0x14	int_status	31:1	reserved	RO	reserved
		0	status	RC	中断状态只读寄存器，将中断脉冲转为电平存储到寄存器内，读取后寄存器自动清零。
0x18	int_mask	31:1	reserved	RO	reserved
		0	mask	RW	中断输出mask使能，为1表示电路输出中断被mask为常0输出，为0表示电路中断正常输出
0x1C	total_cnt_flag	31:1	reserved	RO	reserved
		0	flag	RW	当前total_cnt计数器是否为饱和计数器，为1表示饱和计数器，为0表示非饱和计数器
0x20	total_cnt	31:16	reserved	RO	reserved
		15:0	counter	RC	每接收一个sad_done，计数器加1，根据total_cnt_flag选择计数最大后是否饱和，读清
0x24	cal_ready	31:1	reserved	RO	reserved
		0	ready	RW	计算数据输出使能，为1时输出，为0时不输出

6 Clock and reset

模块时钟关系表如下：

表6-1 模块时钟关系表

Module	Clock
register	clk_core（1500Mhz）
cal_ctrl	clk_core（1500Mhz）
sad_cal	clk_core（1500Mhz）
axi2ram	clk_core（1500Mhz）
inner_sram	clk_core（1500Mhz）

Apb和类axi接口时钟的频率分别为800Mhz和1500Mhz。