

# **FreeARM7** 介绍文档

## **Free-arm**

# 目录

1. FreeARM7 微处理器介绍.....	3
1.1. 简介.....	3
1.2. 接口说明.....	4
1.2.1. 系统接口.....	5
1.2.2. 中断源.....	6
1.2.3. ROM 接口.....	9
1.2.4. 单口 RAM 接口.....	10
2. FreeARM7 整体架构.....	12
2.1. 各类指令的实现.....	13
2.1.1. 从 CPSR/SPSR 至寄存器组.....	13
2.1.2. 从寄存器组至 CPSR/SPSR.....	13
2.1.3. 从寄存器组至寄存器组.....	13
2.1.4. 寄存器组和 RAM 之间交换数据.....	14
附：FreeARM7 源代码（verilog）.....	16

# 1. FreeARM7 微处理器介绍

## 1.1. 简介

FreeARM 微处理器是一种兼容 ARM 架构的微处理器。ARM 架构是一个 32 位精简指令集 (RISC) 处理器架构，其广泛地使用在许多嵌入式系统设计。由于节能的特点，ARM 架构非常适用于行动通讯领域，符合其主要设计目标为低功耗的特性。FreeARM 微处理器作为完全兼容 ARM 架构的微处理器，它以 ARM 编译器编译的代码无缝植入为目标，期望达到同 ARM 公司微处理器 IP 核一样的低功耗效果。FreeARM 微处理器作为一个开源硬件项目，为用户提供免费的技术支援，提供高效、低功耗的 SoC 解决方案。使用者可以任意更改内核，以期达到使用者自定制的目标，所以它能更灵活、高效的服务于 SoC 设计。

FreeARM 微处理器的 ARM7 系列 (以下简称 FreeARM7) 是该类型微处理器中的第一款。它首发于 [www.socvista.com](http://www.socvista.com)，是由 free-arm 联合其他网友，基于 ARMv4 架构而开发的。它用可综合的 verilog 代码描述，接口简单、描述精炼，全部代码不超过 2000 行。它采用三级流水线和哈佛结构，全面兼容各种中断和指令 (除 THUMB 和协处理器指令)。经过评估发现，它在 Xilinx FPGA 和 SMIC 工艺库上都有上佳的实现结果。为方便使用者正确使用该款处理器的 IP 核，free-arm 编写这份文档，提供给使用者参考。在本节介绍后，附有整个 IP 核的全部代码。

		面积	关键路径
Xilinx	Spartan-3E 500	5400+(LUT)	26 ns
SMIC	0.35 um	27140(逻辑门)	40.91 ns
	0.18um	24432(逻辑门)	24.67 ns

注：DC 综合均在未加任何约束的情况；FPGA 综合结果类似；

更多更新请参阅：[www.socvista.com/bbs](http://www.socvista.com/bbs)。

## 1.2. 接口说明

FreeARM7 接口简单，共分为四类：一、系统接口，提供系统控制信号；二、中断源，提供 ARM 架构需要的五个中断信号；三、ROM 接口，同提供指令的 ROM 之间的接口；四、单口 RAM 接口，同单口 RAM 和外设之间数据交互的接口。

类别	名字	方向	位宽	描述
系统接口	clk	IN	1	时钟输入端口
	rst	IN	1	异步复位端口，高电平有效
	cpu_en	IN	1	同步使能端口，高电平有效
中断源	cpu_restart	IN	1	ARM 架构 reset 中断源，高有效
	fiq	IN	1	ARM 架构 FIQ(fast interrupt)中断源，高有效
	irq	IN	1	ARM 架构 IRQ(interrupt)中断源，高有效
	rom_abort	IN	1	ARM 架构 Prefetch Abort(instruction fetch memory abort)中断源，高有效
	ram_abort	IN	1	ARM 架构 Data Abort(data access memory abort)中断源，高有效
ROM 接口	rom_en	OUT	1	ROM 读使能，高有效
	rom_addr	OUT	32	ROM 地址总线
	rom_data	IN	32	ROM 读数据
单口 RAM 接口	ram_cen	OUT	1	RAM 使能信号，高有效
	ram_wen	OUT	1	RAM 写使能信号，高电平代表写，低电平代表读
	ram_addr	OUT	32	RAM 地址总线
	ram_rdata	IN	32	RAM 读数据总线
	ram_wdata	OUT	32	RAM 写数据总线
	ram_flag	OUT	4	RAM 各字节使能信号

## 1.2.1. 系统接口

- clk 信号

clk 信号为整个微处理器提供时钟。FreeARM7 工作在时钟的上升沿。使用者可以通过修改代码中寄存器的描述：`always @(posedge clk or posedge rst)`，使用自定义的时钟信号。

- rst 信号

异步复位信号。当 rst 为高电平时，FreeARM7 复位为初始态。它的复位方式为异步。使用者可以通过修改代码中寄存器的描述：`always @(posedge clk or posedge rst)`和 rst 的使用：`if (rst)`，改变为自定义的各种复位信号。

- cpu\_en 信号

同步使能信号。在时钟的上升沿，并且 cpu\_en 为低电平时，FreeARM7 的所有寄存器停止工作，但不改变保持的电平。使用者可以通过置位 cpu\_en 来让 FreeARM7 暂停工作。使用者可以使用它达到下列目的：

1. 当微处理器需要读 RAM 的某个数据，如果该数据在一个时钟内不能送达，可以置位 cpu\_en 为低电平，直至该数据准备完毕，方可置位 cpu\_en 为高电平。微处理器不会有任何影响。
2. 如果系统处于休眠状况，为了低功耗的目的，使用者可以置位 cpu\_en 为低电平，FreeARM7 停止工作，直至达到某个状态，使用者重新置位 cpu\_en 为高电平，FreeARM7 继续工作。
3. 如果需要读取文字池数据，但存放文字池的 ROM 只有一套接口，这时，可以置位 cpu\_en 为低电平，从 ROM 中读取文字池的内容，送入读数据端口，然后置位 cpu\_en 为高电平。

如果使用者不需要该信号，可以例化为 `1'b1`，使得 FreeARM7 处于永远工作的状态。

## 1.2.2. 中断源

根据 ARM 官方文档：ARM Architecture Reference Manual（序列号：DDI0100E），ARMv4 有七个中断：

Table 2-3 Exception processing modes

Exception type	Mode	Normal address	High vector address
Reset	Supervisor	0x00000000	0xFFFF0000
Undefined instructions	Undefined	0x00000004	0xFFFF0004
Software interrupt (SWI)	Supervisor	0x00000008	0xFFFF0008
Prefetch Abort (instruction fetch memory abort)	Abort	0x0000000C	0xFFFF000C
Data Abort (data access memory abort)	Abort	0x00000010	0xFFFF0010
IRQ (interrupt)	IRQ	0x00000018	0xFFFF0018
FIQ (fast interrupt)	FIQ	0x0000001C	0xFFFF001C

其中五个中断：Reset、Prefetch Abort、Data Abort、IRQ、FIQ 是需要外部中断源的。本类别的五个输入端口分别对应这五个中断。

- **cpu\_restart**：对应 Reset 中断

该中断源 reset 相当于一个系统同步复位。如果在时钟上升沿，cpu\_restart 为高电平，则进入 Reset 中断向量。FreeARM7 会执行下列操作：

- 置位 PC(rf) = 0x0000\_0000
- 置位模式标志寄存器：cpsr\_m = 5'b10011，使得系统进入管理模式
- 置位 IRQ 中断寄存器：cpsr\_i = 1'b1，关闭 IRQ 中断
- 置位 FIQ 中断寄存器：cpsr\_f = 1'b1，关闭 FIQ 中断

- **fiq**：对应 FIQ 中断

如果在时钟上升沿，fiq 输入端口出现高电平，并且 FIQ 中断没有关闭，则进入 FIQ 中断向量。FreeARM7 会执行下列操作：

- 置位 PC(rf) = 0x0000\_001c
- 置位模式标志寄存器：cpsr\_m = 5'b10001，使得系统进入 FIQ 模式

- 置位 IRQ 中断寄存器:  $\text{cpsr}_i = 1'b1$ , 关闭 IRQ 中断
- 置位 FIQ 中断寄存器:  $\text{cpsr}_f = 1'b1$ , 关闭 FIQ 中断
- 置位 FIQ 模式下的 LR:  $\text{re}_{\text{fiq}} = \text{rf} - 3'd4$ , 使得 LR 保存中断时的 PC
- 置位 FIQ 模式下的 SPSR:  $\text{spsr}_{\text{fiq}} = \text{cpsr}$ , 使得 SPSR 保存 CPSR

- **irq:** 对应 IRQ 中断

如果在时钟上升沿, irq 输入端口出现高电平, 并且 IRQ 中断没有关闭, 则进入 IRQ 中断向量。FreeARM7 会执行下列操作:

- 置位  $\text{PC}(\text{rf}) = 0x0000\_0018$
- 置位模式标志寄存器:  $\text{cpsr}_m = 5'b10010$ , 使得系统进入 IRQ 模式
- 置位 IRQ 中断寄存器:  $\text{cpsr}_i = 1'b1$ , 关闭 IRQ 中断
- 置位 IRQ 模式下的 LR:  $\text{re}_{\text{irq}} = \text{rf} - 3'd4$ , 使得 LR 保存中断时的 PC
- 置位 FIQ 模式下的 SPSR:  $\text{spsr}_{\text{irq}} = \text{cpsr}$ , 使得 SPSR 保存 CPSR

- **rom\_abort:** 对应 Prefetch abort 中断

如果在时钟上升沿, rom\_abort 出现高电平, 则系统进入 Prefetch abort 中断。有一种例外: 如果该指令的前一条指令是一条跳转指令, 并且满足条件执行, 则该中断无效。FreeARM7 会执行下列操作:

- 置位  $\text{PC}(\text{rf}) = 0x0000\_000c$
- 置位模式标志寄存器:  $\text{cpsr}_m = 5'b10111$ , 使得系统进入 ABT 模式
- 置位 IRQ 中断寄存器:  $\text{cpsr}_i = 1'b1$ , 关闭 IRQ 中断
- 置位 ABT 模式下的 LR:  $\text{re}_{\text{abt}} = \text{rf} - 3'd4$ , 使得 LR 保存中断时的 PC
- 置位 FIQ 模式下的 SPSR:  $\text{spsr}_{\text{abt}} = \text{cpsr}$ , 使得 SPSR 保存 CPSR

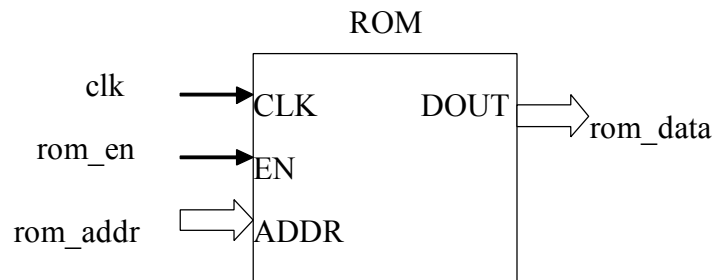
- **ram\_abort:** 对应 Data abort 中断

如果在读取 RAM 或写 RAM 时, 发生了意外, 可以置位该端口。如果在时钟上升沿, ram\_abort 为高电平, 则进入 Data abort 中断。FreeARM7 会执行下列操作:

- 置位  $PC(rf) = 0x0000\_0010$
- 置位模式标志寄存器:  $cpsr\_m = 5'b10111$ , 使得系统进入 ABT 模式
- 置位 IRQ 中断寄存器:  $cpsr\_i = 1'b1$ , 关闭 IRQ 中断
- 置位 ABT 模式下的 LR:  $re\_abt = rf - 3'd4$ , 使得 LR 保存中断时的 PC
- 置位 FIQ 模式下的 SPSR:  $spsr\_abt = cpsr$ , 使得 SPSR 保存 CPSR

如果使用者不需要使用某种接口, 连接该端口至  $1'b0$  即可。

### 1.2.3. ROM 接口



FreeARM7 认为生成的指令存放于一块单口同步 ROM 内。在时钟的上升沿, 如果 rom\_en 为高电平, 则在下一个时钟上升沿, rom\_data 等于 rom\_addr 对应的指令。

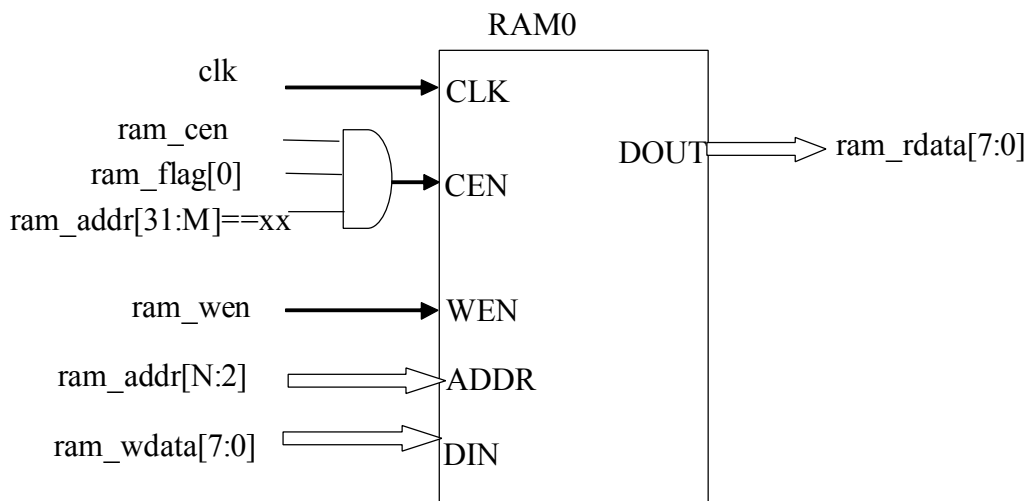
如果取指令出现异常, 可以通过 Prefetch Abort 中断处理, 也可以置位 cpu\_en 为低电平, 直至能正确取指令后, 重新置位 cpu\_en 为高电平。

## 1.2.4. 单口 RAM 接口

本类接口可以实现两类用途：一、驳接单口 RAM，使得 FreeARM7 能够正确的读写数据；二、驳接外设，使得 FreeARM7 能够正确操作外设。

### 1.2.4.1. 驳接单口 RAM

FreeARM7 建议 RAM 的存放以字节(byte)为单位。因此，建议用 4 块数据位宽为 8 的单口 RAM 存放 32bit 中的每一个 byte。为区分每一块单口 RAM，ram\_flag[3:0]可以用来标示每一块是否激活。



如上图所示，这是存放最低字节的 RAM。

它的 CEN 端连接： $ram\_cen \& ram\_flag[0] \& (ram\_addr[31:M]==xx)$ 。它的含义是：1，ram\_cen 标示是否读写 RAM；2，ram\_flag[0]标示是否访问本块字节 RAM；3，(ram\_addr[31:M]==xx)标示地址高位是否满足条件。如果三者满足，才能对该字节 RAM 进行读写操作。

ram\_wen 作为标示本次对 RAM 的操作是读操作还是写操作，直接连接入每一块字节 RAM 的 WEN。

字节 RAM 的 ADDR 连接 ram\_addr[N:2]。其中 N 随着分配 RAM 空间的大

小而变。

字节 RAM 的 DIN 端连接写数据。由于本字节 RAM 为最低字节，所以驳接：  
ram\_wdata[7:0]。另外三块分别对应 ram\_flag[1]、ram\_flag[2]、ram\_flag[3]，所以  
对应的 DIN 连接的是：ram\_wdata[15:8]、ram\_wdata[23:16]、ram\_wdata[31:24]。

同上，DOUT 的输出连接 ram\_rdata。

### 1.2.4.2. 驳接外设

外设同 FreeARM7 的连接方式同样是通过单口 RAM 来连接的。使用者在编写嵌入式软件时，应提前为每一个外设分配地址。例如地址：0xE000\_0001 是某外设状态。如果 ram\_cen = 1'b1, ram\_wen = 1'b0, ram\_addr = 32'hE000\_0000, ram\_flag=4'b0010。显然这代表读地址：0xE000\_0001。那么在下一个时钟时，ram\_rdata[15:8]就应该出现该外设状态，让 FreeARM7 来读入。

可以用一个寄存器 flag 保存 ram\_cen = 1'b1, ram\_wen = 1'b0, ram\_addr = 32'hE000\_0000, ram\_flag=4'b0010，这一组合的出现。则 ram\_rdata 可以用 verilog 这样表述：

```
if ( flag )
```

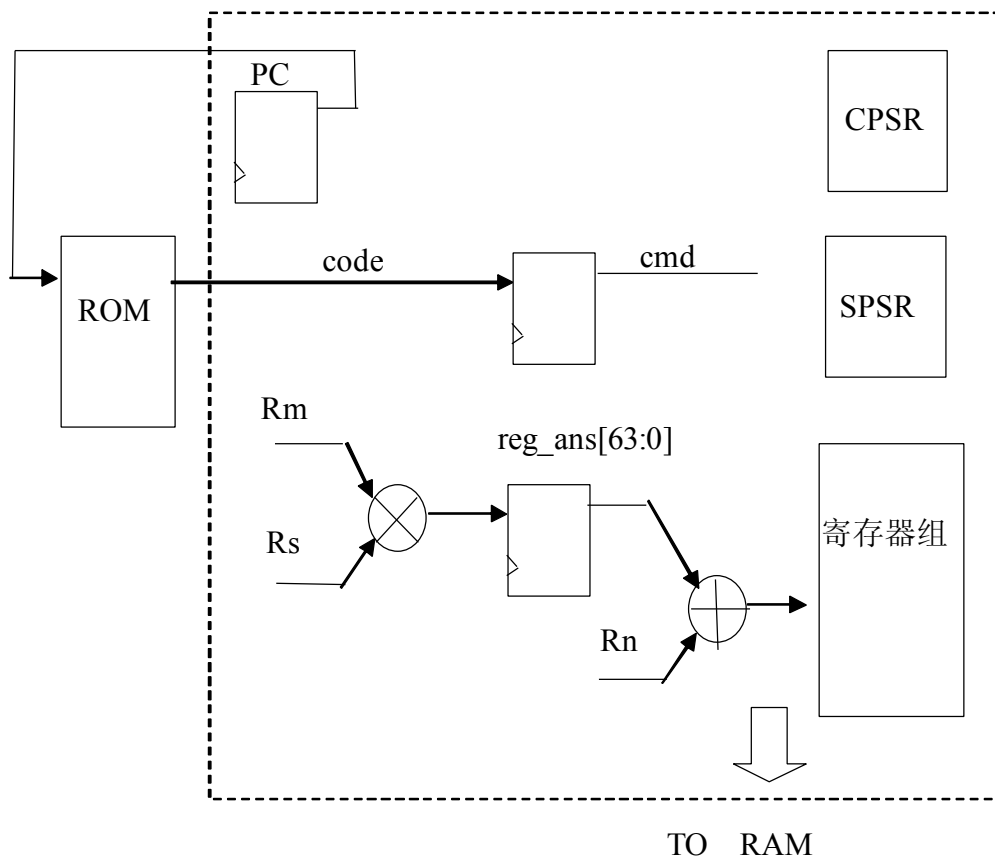
```
    ram_rdata = 外设状态;
```

```
else
```

```
    ram_rdata = DOUT FROM RAM;
```

如果嵌入式软件需要对外设置位，是通过写某个设定的地址实现的。由于外设和嵌入式软件已经约定地址，所以如果 ram\_cen = 1'b1, ram\_wen=1'b1, ram\_addr 等于约定的地址，则 ram\_wdata 可由外设取用，进行置位操作。

## 2. FreeARM7 整体架构



FreeARM7 采用三级流水线:

1. 第一级: PC 存放为指令的地址, 传递给 ROM, 取出该指令, 这是第一级。
2. 第二级: 指令取出, 是为 code。根据 code, 从寄存器组里得到 Rm 和 Rs。然后经过 32 位乘法器, 得到的 64bit 结果存放于 reg\_ans[63:0], code 也暂存于 cmd。这是第二级。
3. 第三级: 根据 cmd, 从 reg\_ans[63:0]里得到第二操作数, 从寄存器组里得到 Rn, 两者经过一个 32 位的加法器。得到的结果, 要么送入 CPSR, SPSR 或寄存器组, 或者作为读写 RAM 的地址。这是第三级。

可以看出, 关键路径的瓶颈在于 32 位的乘法器。

## 2.1. 各类指令的实现

下面分类讲述各类指令的实现。可以把目标和来源大致分为：CPSR/SPSR、寄存器组、RAM/ROM。CPSR/SPSR 指的两种状态寄存器 CPSR 和各种 SPSR；寄存器组指的是 R0~R15 共 15 类寄存器；RAM/ROM 指统一编制的 RAM 空间，通过单口 RAM 端口操作。

### 2.1.1. 从 CPSR/SPSR 至寄存器组

它包含一条指令：MRS。它表示把 CPSR/SPSR 写入相应的寄存器。由于它不含 Rm 和 Rs，输入是 CPSR/SPSR，所以 code 这一级不做任何操作。当 cmd 等于 MRS 时，把 CPSR/SPSR 送入相应的寄存器。

### 2.1.2. 从寄存器组至 CPSR/SPSR

它包含一条指令：MSR。它表示把某一个寄存器或立即数，送入 CPSR/SPSR。这里，可以赋值 Rm 为该寄存器或立即数；Rs=1'b1，那么 reg\_ans[63:0]存放为该数据。在 cmd 阶段，该数据不需要经过加法器，直接送入 CPSR/SPSR。

### 2.1.3. 从寄存器组至寄存器组

这一类又可分为三种：

- 跳转指令：B、BL、BX

这类指令是把跳转地址或偏移量+PC 送入 PC 内。如果 code 为 B/BL，则 Rm 等于偏移量，Rs=1，reg\_ans 保存为偏移量。如果 cmd 为 B/BL，Rn 等于 PC，加法器的结果送入 PC，则完成跳转功能。如果需要拷贝 PC 到 LR 寄存器，则同时完成。

如果 code 为 BX，则 Rm 为偏移地址，Rs = 1，reg\_ans 保存为偏移地址。如果 cmd 为 BX，则该偏移地址直接送入 PC。

- 数据处理指令：MOV/ADC 等

这一类指令需要将 Rm 进行移位操作，然后和 Rn 进行运算，结果要么保存入寄存器组，要么置位 CPSR。

FreeARM7 利用乘法器完成移位操作。例如逻辑左移位 2 位，则 Rm 保持不变，Rs 赋值为：2'b10，则相乘结果的 reg\_ans[31:0]即为逻辑左移位的结果。如果逻辑右移位，只需赋值 Rs 为：~rot\_num+1（其中 rot\_num 为移位数目），相乘结果的 reg\_ans[63:32]即为移位结果。

如果 cmd 为数据处理指令，则根据移位方式，选择 reg\_ans 为第二操作数，它和 Rn 进行运算，则得到运算结果。

- 乘法/乘加、长乘法/长乘加指令

这类指令的运算形式为： $Rm * Rs + Rn$ 。Rm 和 Rs、Rn 只要赋值为相应的值，则结果送入相应的寄存器内。

需要注意的是：长乘系列需要两个时钟完成。这是因为写入寄存器的通路只有一个。在前一个时钟内，reg\_ans[31:0]和相应的 Rn 相加，结果送入 Rn 对应的寄存器内，并置位 cpsr\_z；在后一个时钟内，reg\_ans[63:32]和相应的 Rn 相加，结果置位 cpsr\_n 和 cpsr\_z。

## 2.1.4. 寄存器组和 RAM 之间交换数据

- 从 RAM 内读出数据，送入寄存器组

这类指令包括 LDR, LDRB, LDRH, LDRSB, LDRSH。这类指令通常包括地址写回。

如果地址计算时需要移位，通过赋值 Rm, Rs 达到移位的目的。在 cmd 阶段，选择相应的第二操作数，和 Rn 进行加或减。结果要么进行地址回写，要么作为 RAM 端口的 ram\_addr。

置位读操作后，在下一个时钟，把 ram\_rdata 送入相应的寄存器。

- 从寄存器内选出数据，写入 RAM

这类指令包括：STR, STRB, STRH。这类指令通常包括地址回写。

它的地址计算方式同上。如果地址得到后，从寄存器组内选出需要写入的数据，送入 ram\_wdata 端口即可。

- 寄存器组和 RAM 进行数据交换

这类指令只包括 SWP/SWPB。

由于它需要读 RAM 和写 RAM 各一次，则指令在执行 SWP/SWPB 时，流水线需要延缓一个周期。在前一个时钟内，进行读操作；在下一个时钟内，这时，数据并没有真正写入，可以进行写操作。

- 单指令多数据操作

这类指令包括 LDM/STM。它是一条指令，指挥多个寄存器写入 RAM 内，或加载多个数据进入寄存器组。

由于读写 RAM 端口只有一套，所以如果遇见这类指令，流水线需要延缓。延缓长度根据指令要求的操作次数而定。

cmd[15:0]存放的是需要操作的寄存器的标志。如果该标示为 1，表示需要对该寄存器操作。从低到高开始，每次完成一个寄存器的操作，则置位相应的标志为 0，直到所有的寄存器处理完毕。

## 附：FreeARM7 源代码（verilog）

以下为 FreeARM7 源码，请复制到一个文本文件使用：

```
`timescale 1 ns/1 ns
`define DEL 2
module FreeARM7(
    clk,
    cpu_en,
    cpu_restart,
    fiq,
    irq,
    ram_abort,
    ram_rdata,
    rom_abort,
    rom_data,
    rst,

    ram_addr,
    ram_cen,
    ram_flag,
    ram_wdata,
    ram_wen,
    rom_addr,
    rom_en
);

input          clk;
input          cpu_en;
```

```

input          cpu_restart;
input          fiq;
input          irq;
input          ram_abort;
input [31:0]   ram_rdata;
input          rom_abort;
input [31:0]   rom_data;
input          rst;

```

```

output [31:0]  ram_addr;
output         ram_cen;
output [3:0]   ram_flag;
output [31:0]  ram_wdata;
output         ram_wen;
output [31:0]  rom_addr;
output         rom_en;

```

```

/*****/

```

```

//register definition area

```

```

/*****/

```

```

reg           add_flag;
reg           all_code;
reg [3:0]     cha_num;
reg           cha_vld;
reg [31:0]   cmd;
reg [31:0]   cmd_addr;
reg           cmd_flag;
reg           code_abort;

```

reg		code_flag;
reg	[31:0]	code_rm;
reg	[31:0]	code_rma;
reg	[4:0]	code_rot_num;
reg	[31:0]	code_rs;
reg	[2:0]	code_rs_flag;
reg	[31:0]	code_rsa;
reg		code_und;
reg		cond_satisfy;
reg		cpsr_c;
reg		cpsr_f;
reg		cpsr_i;
reg	[4:0]	cpsr_m;
reg		cpsr_n;
reg		cpsr_v;
reg		cpsr_z;
reg	[31:0]	dp_ans;
reg		extra_num;
reg		fiq_flag;
reg	[31:0]	go_data;
reg	[5:0]	go_fmt;
reg	[3:0]	go_num;
reg		go_vld;
reg		irq_flag;
reg		ldm_change;
reg	[3:0]	ldm_num;
reg	[3:0]	ldm_sel;
reg		ldm_usr;
reg		ldm_vld;
reg		multl_extra_num;

```
reg    [31:0]    r0;
reg    [31:0]    r1;
reg    [31:0]    r2;
reg    [31:0]    r3;
reg    [31:0]    r4;
reg    [31:0]    r5;
reg    [31:0]    r6;
reg    [31:0]    r7;
reg    [31:0]    r8_fiq;
reg    [31:0]    r8_usr;
reg    [31:0]    r9_fiq;
reg    [31:0]    r9_usr;
reg    [31:0]    ra_fiq;
reg    [31:0]    ra_usr;
reg    [3:0]     ram_flag;
reg    [31:0]    ram_wdata;
reg    [31:0]    rb_fiq;
reg    [31:0]    rb_usr;
reg    [31:0]    rc_fiq;
reg    [31:0]    rc_usr;
reg    [31:0]    rd;
reg    [31:0]    rd_abt;
reg    [31:0]    rd_fiq;
reg    [31:0]    rd_irq;
reg    [31:0]    rd_svc;
reg    [31:0]    rd_und;
reg    [31:0]    rd_usr;
reg    [31:0]    re;
reg    [31:0]    re_abt;
reg    [31:0]    re_fiq;
```

```

reg    [31:0]    re_irq;
reg    [31:0]    re_svc;
reg    [31:0]    re_und;
reg    [31:0]    re_usr;
reg    [63:0]    reg_ans;
reg    [31:0]    rf;
reg                    rm_msb;
reg    [31:0]    rn;
reg    [31:0]    rna;
reg    [31:0]    rnb;
reg                    rs_msb;
reg    [31:0]    sec_operand;
reg    [10:0]    spsr;
reg    [10:0]    spsr_abt;
reg    [10:0]    spsr_fiq;
reg    [10:0]    spsr_irq;
reg    [10:0]    spsr_svc;
reg    [10:0]    spsr_und;
reg    [3:0]    sum_m;
reg    [31:0]    to_data;
reg    [3:0]    to_num;

```

```

/*****/

```

```

/*****/

```

```

//wire definition area

```

```

/*****/

```

```

wire    [31:0]    and_ans;

```

```

wire [31:0] bic_ans;
wire bit_cy;
wire bit_ov;
wire cha_rf_vld;
wire cmd_is_b;
wire cmd_is_bx;
wire cmd_is_dp0;
wire cmd_is_dp1;
wire cmd_is_dp2;
wire cmd_is_ldm;
wire cmd_is_ldr0;
wire cmd_is_ldr1;
wire cmd_is_ldrh0;
wire cmd_is_ldrh1;
wire cmd_is_ldrsb0;
wire cmd_is_ldrsb1;
wire cmd_is_ldrsh0;
wire cmd_is_ldrsh1;
wire cmd_is_mrs;
wire cmd_is_msr0;
wire cmd_is_msr1;
wire cmd_is_mult;
wire cmd_is_multl;
wire cmd_is_multlx;
wire cmd_is_swi;
wire cmd_is_swp;
wire cmd_is_swpX;
wire cmd_ok;
wire [3:0] cmd_sum_m;
wire [31:0] code;

```

```

wire          code_is_b;
wire          code_is_bx;
wire          code_is_dp0;
wire          code_is_dp1;
wire          code_is_dp2;
wire          code_is_ldm;
wire          code_is_ldr0;
wire          code_is_ldr1;
wire          code_is_ldrh0;
wire          code_is_ldrh1;
wire          code_is_ldrsb0;
wire          code_is_ldrsb1;
wire          code_is_ldrsh0;
wire          code_is_ldrsh1;
wire          code_is_mrs;
wire          code_is_msr0;
wire          code_is_msr1;
wire          code_is_mult;
wire          code_is_multl;
wire          code_is_swi;
wire          code_is_swp;
wire [3:0]    code_rm_num;
wire          code_rm_vld;
wire [3:0]    code_rn_num;
wire          code_rn_vld;
wire [3:0]    code_rnhi_num;
wire          code_rnhi_vld;
wire [3:0]    code_rs_num;
wire          code_rs_vld;
wire [3:0]    code_sum_m;

```

```

wire [10:0] cpsr;
wire [1:0] cy_high_bits;
wire [31:0] eor_ans;
wire      fiq_en;
wire      go_rf_vld;
wire      high_bit;
wire      hold_en;
wire      int_all;
wire      irq_en;
wire [31:0] ldm_data;
wire      ldm_rf_vld;
wire [63:0] mult_ans;
wire [31:0] or_ans;
wire [31:0] r8;
wire [31:0] r9;
wire [31:0] ra;
wire [31:0] ram_addr;
wire      ram_cen;
wire      ram_wen;
wire [31:0] rb;
wire [31:0] rc;
wire [31:0] rf_b;
wire [31:0] rom_addr;
wire      rom_en;
wire [31:0] sum_middle;
wire [31:0] sum_m_rm;
wire      to_rf_vld;
wire      to_vld;
wire      wait_en;

```

```

/*****/

/*****/

//wire statement area
/*****/

assign and_ans =  rn & sec_operand;

assign bic_ans =  rn & ~sec_operand;

assign bit_cy =  cy_high_bits[1];

assign bit_ov =  bit_cy ^ sum_middle[31];

assign cha_rf_vld =  cha_vld & ( cha_num==4'hf );

assign cmd_is_b =  ( cmd[27:25]==3'b101 );

assign cmd_is_bx =  ( {cmd[27:23],cmd[20],cmd[7],cmd[4]}==8'b00010001 );

assign cmd_is_dp0 =  ( cmd[27:25]==3'b0 ) & ~cmd[4] & ( ( cmd[24:23]!=2'b10 ) |
cmd[20] );

assign cmd_is_dp1 =  ( cmd[27:25]==3'b0 ) & ~cmd[7] & cmd[4] &
( ( cmd[24:23]!=2'b10 ) | cmd[20] );

assign cmd_is_dp2 =  ( cmd[27:25]==3'b001 ) & ( ( cmd[24:23]!=2'b10 ) | cmd[20] );

```

```

assign cmd_is_ldm = ( cmd[27:25]==3'b100 );

assign cmd_is_ldr0 = ( cmd[27:25]==3'b010 );

assign cmd_is_ldr1 = ( cmd[27:25]==3'b011 );

assign cmd_is_ldrh0 = ( cmd[27:25]==3'b0 ) & ( cmd[7:4]==4'b1011 ) & ~cmd[22];

assign cmd_is_ldrh1 = ( cmd[27:25]==3'b0 ) & ( cmd[7:4]==4'b1011 ) & cmd[22];

assign cmd_is_ldrsb0 = ( cmd[27:25]==3'b0 ) & ( cmd[7:4]==4'b1101 ) & ~cmd[22];

assign cmd_is_ldrsb1 = ( cmd[27:25]==3'b0 ) & ( cmd[7:4]==4'b1101 ) & cmd[22];

assign cmd_is_ldrsh0 = ( cmd[27:25]==3'b0 ) & ( cmd[7:4]==4'b1111 ) & ~cmd[22];

assign cmd_is_ldrsh1 = ( cmd[27:25]==3'b0 ) & ( cmd[7:4]==4'b1111 ) & cmd[22];

assign cmd_is_mrs = ( {cmd[27:23],cmd[21:20],cmd[7],cmd[4]}==9'b000100000 );

assign cmd_is_msr0 = ( {cmd[27:23],cmd[21:20],cmd[7],cmd[4]}==9'b000101000 );

assign cmd_is_msr1 = ( cmd[27:25]==3'b001 ) & ( cmd[24:23]==2'b10 ) &
~cmd[20];

assign cmd_is_mult = ( cmd[27:25]==3'b0 ) & ( cmd[7:4]==4'b1001 ) &
( cmd[24:23]==2'b00 );

```

```
assign cmd_is_multl = (cmd[27:25]==3'b0 ) & ( cmd[7:4]==4'b1001 ) &
( cmd[24:23]==2'b01 );
```

```
assign cmd_is_multlx = ( cmd[27:24]==4'b1100 );
```

```
assign cmd_is_swi = ( cmd[27:25]==3'b111 );
```

```
assign cmd_is_swp = (cmd[27:25]==3'b0 ) & ( cmd[7:4]==4'b1001 ) &
( cmd[24:23]==2'b10 );
```

```
assign cmd_is_swpX = ( cmd[27:24]==4'b1101 );
```

```
assign cmd_ok = ~int_all & cmd_flag & cond_satisfy;
```

```
assign cmd_sum_m =
(cmd[0]+cmd[1]+cmd[2]+cmd[3]+cmd[4]+cmd[5]+cmd[6]+cmd[7]+cmd[8]+cmd[9]
+cmd[10]+cmd[11]+cmd[12]+cmd[13]+cmd[14]+cmd[15]);
```

```
assign code = rom_data;
```

```
assign code_is_b = ( code[27:25]==3'b101 );
```

```
assign code_is_bx = ( {code[27:23],code[20],code[7],code[4]}==8'b00010001 );
```

```
assign code_is_dp0 = ( code[27:25]==3'b0 ) & ~code[4] & ( ( code[24:23]!=2'b10 )
| code[20] );
```

```
assign code_is_dp1 = (code[27:25]==3'b0 ) & ~code[7] & code[4] &
( ( code[24:23]!=2'b10 ) | code[20] );
```

```
assign code_is_dp2 = ( code[27:25]==3'b001 ) & ( ( code[24:23]!=2'b10 ) | code[20]
);
```

```
assign code_is_ldm = ( code[27:25]==3'b100 );
```

```
assign code_is_ldr0 = ( code[27:25]==3'b010 );
```

```
assign code_is_ldr1 = ( code[27:25]==3'b011 );
```

```
assign code_is_ldrh0 = (code[27:25]==3'b0 ) & ( code[7:4]==4'b1011 ) &
~code[22];
```

```
assign code_is_ldrh1 = (code[27:25]==3'b0 ) & ( code[7:4]==4'b1011 ) & code[22];
```

```
assign code_is_ldrsb0 = (code[27:25]==3'b0 ) & ( code[7:4]==4'b1101 ) &
~code[22];
```

```
assign code_is_ldrsb1 = (code[27:25]==3'b0 ) & ( code[7:4]==4'b1101 ) & code[22];
```

```
assign code_is_ldrsh0 = (code[27:25]==3'b0 ) & ( code[7:4]==4'b1111 ) &
~code[22];
```

```
assign code_is_ldrsh1 = (code[27:25]==3'b0 ) & ( code[7:4]==4'b1111 ) & code[22];
```

```
assign code_is_mrs = ( {code[27:23],code[21:20],code[7],code[4]}==9'b000100000
);
```

```
assign code_is_msr0 =
({code[27:23],code[21:20],code[7],code[4]}==9'b000101000 );
```

```
assign code_is_msr1 = ( code[27:25]==3'b001 ) & ( code[24:23]==2'b10 ) &
~code[20];
```

```
assign code_is_mult = (code[27:25]==3'b0 ) & ( code[7:4]==4'b1001 ) &
( code[24:23]==2'b00 );
```

```
assign code_is_mult1 = (code[27:25]==3'b0 ) & ( code[7:4]==4'b1001 ) &
( code[24:23]==2'b01 );
```

```
assign code_is_swi = ( code[27:25]==3'b111 );
```

```
assign code_is_swp = (code[27:25]==3'b0 ) & ( code[7:4]==4'b1001 ) &
( code[24:23]==2'b10 );
```

```
assign code_rm_num = code[3:0];
```

```
assign code_rm_vld = code_flag &
( code_is_msr0|code_is_dp0|code_is_bx|code_is_dp1|code_is_mult|code_is_mult1|cod
e_is_swp|code_is_ldrh0|code_is_ldrsb0|code_is_ldrsh0|code_is_ldr1 );
```

```
assign code_rm_num = code[19:16];
```

```
assign code_rm_vld = code_flag &
( code_is_dp0|code_is_dp1|code_is_mult|code_is_swp|code_is_ldrh0|code_is_ldrh1|c
ode_is_ldrsb0|code_is_ldrsb1|code_is_ldrsh0|code_is_ldrsh1|code_is_dp2|code_is_ldr
0|code_is_ldr1|code_is_ldm );
```

```
assign code_rnhi_num = code[15:12];
```

```
assign code_rnhi_vld = code_flag &  
( code_is_mult|code_is_multl|((code_is_ldrh0|code_is_ldrh1|code_is_ldr0|code_is_ldr  
1)& ~code[20]) );
```

```
assign code_rs_num = code[11:8];
```

```
assign code_rs_vld = code_flag & ( code_is_dp1|code_is_mult|code_is_multl );
```

```
assign code_sum_m =  
(code[0]+code[1]+code[2]+code[3]+code[4]+code[5]+code[6]+code[7]+code[8]+cod  
e[9]+code[10]+code[11]+code[12]+code[13]+code[14]+code[15]);
```

```
assign cpsr = { cpsr_n,cpsr_z,cpsr_c,cpsr_v,cpsr_i,cpsr_f,cpsr_m};
```

```
assign cy_high_bits = add_flag ? ( rn[31] + sec_operand[31] + sum_middle[31] ) :  
( rn[31] - sec_operand[31] - sum_middle[31] );
```

```
assign eor_ans = rn ^ sec_operand;
```

```
assign fiq_en = fiq_flag & cmd_flag & ~cpsr_f;
```

```
assign go_rf_vld = go_vld & (go_num==4'hf);
```

```
assign high_bit = cy_high_bits[0];
```

```
assign hold_en = cmd_ok & ( cmd_is_swp | cmd_is_multl | ( cmd_is_ldm &  
(cmd_sum_m !=4'b0) ) );
```

```
assign int_all = cpu_restart|ram_abort|fiq_en|irq_en|( cmd_flag &
( code_abort|code_und|(cond_satisfy & cmd_is_swi)));
```

```
assign irq_en = irq_flag & cmd_flag & ~cpsr_i;
```

```
assign ldm_data = go_data;
```

```
assign ldm_rf_vld = (ldm_vld & ( ldm_num==4'hf ))((cmd_ok & cmd_is_ldm &
cmd[20])&(ldm_sel==4'hf));
```

```
assign mult_ans = code_rm * code_rs;
```

```
assign or_ans = rn | sec_operand;
```

```
assign r8 = (cpsr_m==5'b10001) ? r8_fiq : r8_usr;
```

```
assign r9 = (cpsr_m==5'b10001) ? r9_fiq : r9_usr;
```

```
assign ra = (cpsr_m==5'b10001) ? ra_fiq : ra_usr;
```

```
assign ram_addr = {cmd_addr[31:2],2'b0};
```

```
assign ram_cen = cpu_en & cmd_ok &
(cmd_is_ldrh0|cmd_is_ldrh1|cmd_is_ldrsb0|cmd_is_ldrsb1|cmd_is_ldrsh0|cmd_is_ldr
sh1|cmd_is_ldr0|cmd_is_ldr1|cmd_is_swp|cmd_is_swp)|cmd_is_ldm
&(cmd_sum_m!=4'b0));
```

```
assign ram_wen = cmd_is_swp ? 1'b0 : ~cmd[20];
```

```
assign rb = (cpsr_m==5'b10001) ? rb_fiq : rb_usr;
```

```
assign rc = (cpsr_m==5'b10001) ? rc_fiq : rc_usr;
```

```
assign rf_b = rf - 3'd4;
```

```
assign rom_addr = rf;
```

```
assign rom_en = cpu_en & ( ~(int_all | to_rf_vld | cha_rf_vld | go_rf_vld | wait_en |  
hold_en ) );
```

```
assign sum_middle = add_flag ? ( m[30:0] + sec_operand[30:0] + extra_num ) :  
( m[30:0] - sec_operand[30:0] - extra_num );
```

```
assign sum_rm_rm = {bit_cy,high_bit,sum_middle[30:0]};
```

```
assign to_rf_vld = cmd_ok & ( ( (cmd[15:12]==4'hf) &  
( (cmd_is_dp0|cmd_is_dp1|cmd_is_dp2) & (cmd[24:23]!=2'b10) ) ) | (cmd_is_b |  
cmd_is_bx) );
```

```
assign to_vld = cmd_ok &  
( cmd_is_mrs(((cmd_is_dp0|cmd_is_dp1|cmd_is_dp2)&(cmd[24:23]!=2'b10))|cmd_is  
_mult|cmd_is_multl|cmd_is_multlx(((cmd_is_ldrh0|cmd_is_ldrh1|cmd_is_ldrsb0|cmd  
_is_ldrsb1|cmd_is_ldrsh0|cmd_is_ldrsh1|cmd_is_ldr0|cmd_is_ldr1)&(cmd[21]  
~cmd[24]))|(cmd_is_ldm &(cmd_sum_m==4'b0)&cmd[21]) );
```

```
assign wait_en = (code_rm_vld & cha_vld  
&(cha_num==code_rm_num))|(code_rm_vld & to_vld &  
(to_num==code_rm_num))|(code_rm_vld & go_vld &(go_num==code_rm_num)) |  
(code_rs_vld & cha_vld & (cha_num==code_rs_num))|(code_rs_vld & to_vld &  
(to_num==code_rs_num))|(code_rs_vld & go_vld & (go_num==code_rs_num)) |
```

```
(code_rn_vld&cha_vld&(code_rn_num==cha_num)) |
(code_rnhi_vld&cha_vld&(code_rnhi_num==cha_num)) | (code_rm_vld & (ldm_vld
& ~hold_en) & (ldm_num==code_rm_num)) | (code_rs_vld & (ldm_vld & ~hold_en)
& (ldm_num==code_rs_num));
```

```
/**/
```

```
//register statement area
```

```
/**/
```

```
always @( * )
```

```
if ( cmd_is_mult|cmd_is_b|cmd_is_bx )
```

```
    add_flag = 1'b1;
```

```
else if ( cmd_is_multl|cmd_is_multlx )
```

```
    add_flag = cmd[22] ? ~(rm_msb^rs_msb) : 1'b1;
```

```
else if ( cmd_is_dp0|cmd_is_dp1|cmd_is_dp2 )
```

```
    add_flag =
```

```
(cmd[24:21]==4'b0100)|(cmd[24:21]==4'b0101)|(cmd[24:21]==4'b1011)|(cmd[24:21]
==4'b1101);
```

```
else
```

```
    add_flag = cmd[23];
```

```
always @( * )
```

```
if ( code[27:25]==3'b0 )
```

```
    if ( ~code[4] )
```

```
        if ( ( code[24:23]==2'b10 ) & ~code[20] )
```

```
            if ( ~code[21] )
```

```
                all_code = ( code[19:16]==4'hf ) & ( code[11:0] == 12'b0 );
```

```
            else
```

```
                all_code = ( code[18:17] == 2'b0 ) & ( code[15:12]==4'hf ) &
```

```
( code[11:4]==8'h0 );
```

```
        else
```

```

        all_code = ( code[24:23] != 2'b10 ) | code[20];
else if ( ~code[7] )
    if ( code[24:20] == 5'b10010 )
        all_code = ( code[19:4] == 16'hfff1 );
    else
        all_code = ( code[24:23] != 2'b10 ) | code[20];
else if ( code[6:5] == 2'b0 )
    if ( code[24:22] == 3'b0 )
        all_code = 1'b1;
    else if ( code[24:23] == 2'b01 )
        all_code = 1'b1;
    else if ( code[24:23] == 2'b10 )
        all_code = ( code[21:20] == 2'b0 ) & ( code[11:8] == 4'b0 );
    else
        all_code = 1'b0;
else if ( code[6:5] == 2'b01 )
    if ( ~code[22] )
        all_code = ( code[11:8] == 4'b0 );
    else
        all_code = 1'b1;
else //if ( ( code[6:5] == 2'b10 ) || ( code[6:5] == 2'b11 ) )
    if ( code[20] )
        if ( ~code[22] )
            all_code = ( code[11:8] == 4'b0 );
        else
            all_code = 1'b1;
    else
        all_code = 1'b0;
else if ( code[27:25] == 3'b001 )
    if ( ( code[24:23] == 2'b10 ) & ~code[20] )

```

```

        all_code = code[21] & ( code[18:17]==2'b0 ) & ( code[15:12]==4'hf );
    else
        all_code = ( code[24:23]!=2'b10 ) | code[20];
else if ( code[27:25]==3'b010 )
    all_code = 1'b1;
else if ( code[27:25]==3'b011 )
    all_code = ~code[4];
else if ( code[27:25]==3'b100 )
    all_code = 1'b1;
else if ( code[27:25]==3'b101 )
    all_code = 1'b1;
else if ( code[27:25]==3'b111 )
    all_code = code[24];
else
    all_code = 1'b0;

always @( * )
cha_num = cmd[15:12];

always @( * )
if ( cmd_ok )
    cha_vld =
( cmd_is_ldrh0|cmd_is_ldrh1|cmd_is_ldrsb0|cmd_is_ldrsb1|cmd_is_ldrsh0|cmd_is_ldrsh1|cmd_is_ldr0|cmd_is_ldr1|cmd_is_swp ) & cmd[20];
else
    cha_vld = 0;

always @( posedge clk or posedge rst )
if ( rst )
    cmd <= #`DEL 32'd0;

```

```

else if ( cpu_en )
    if ( ~hold_en )
        cmd <= #`DEL  code;
    else if ( cmd_is_swp ) begin
        cmd[27:25] <= #`DEL 3'b110;
        cmd[15:12] <= #`DEL cmd[3:0];
    end
    else if ( cmd_is_mult1 )
        cmd[27:25] <= #`DEL 3'b110;
    else if ( cmd_is_ldm ) begin
        cmd[0] <= #`DEL 1'b0;
        cmd[1] <= #`DEL cmd[0] ? cmd[1] : 1'b0;
        cmd[2] <= #`DEL ((cmd[1:0])) ? cmd[2] : 1'b0;
        cmd[3] <= #`DEL ((cmd[2:0])) ? cmd[3] : 1'b0;
        cmd[4] <= #`DEL ((cmd[3:0])) ? cmd[4] : 1'b0;
        cmd[5] <= #`DEL ((cmd[4:0])) ? cmd[5] : 1'b0;
        cmd[6] <= #`DEL ((cmd[5:0])) ? cmd[6] : 1'b0;
        cmd[7] <= #`DEL ((cmd[6:0])) ? cmd[7] : 1'b0;
        cmd[8] <= #`DEL ((cmd[7:0])) ? cmd[8] : 1'b0;
        cmd[9] <= #`DEL ((cmd[8:0])) ? cmd[9] : 1'b0;
        cmd[10] <= #`DEL ((cmd[9:0])) ? cmd[10] : 1'b0;
        cmd[11] <= #`DEL ((cmd[10:0])) ? cmd[11] : 1'b0;
        cmd[12] <= #`DEL ((cmd[11:0])) ? cmd[12] : 1'b0;
        cmd[13] <= #`DEL ((cmd[12:0])) ? cmd[13] : 1'b0;
        cmd[14] <= #`DEL ((cmd[13:0])) ? cmd[14] : 1'b0;
        cmd[15] <= #`DEL ((cmd[14:0])) ? cmd[15] : 1'b0;
    end
    else;
else;

```

```

always @( * )
if ( cmd_is_ldm )
    cmd_addr = sum_rn_m;
else if ( cmd_is_swp|cmd_is_swpX )
    cmd_addr = m;
else if ( cmd[24] )
    cmd_addr = sum_rn_m;
else
    cmd_addr = m;

always @( posedge clk or posedge rst )
if ( rst )
    cmd_flag <= #`DEL 1'd0;
else if ( cpu_en )
    if ( int_all )
        cmd_flag <= #`DEL 0;
    else if ( ~hold_en )
        if ( wait_en | to_rf_vld | cha_rf_vld | go_rf_vld )
            cmd_flag <= #`DEL 0;
        else
            cmd_flag <= #`DEL code_flag;
    else;
else;

always @( posedge clk or posedge rst )
if ( rst )
    code_abort <= #`DEL 1'd0;
else if ( cpu_en )
    if ( ~hold_en )
        code_abort <= #`DEL rom_abort;

```

```

    else;
else;

always @( posedge clk or posedge rst )
if ( rst )
    code_flag <= #'DEL 1'd0;
else if ( cpu_en )
    if ( int_all | to_rf_vld | cha_rf_vld | go_rf_vld | ldm_rf_vld )
        code_flag <= #'DEL 0;
    else
        code_flag <= #'DEL 1;
else;

always @( * )
if ( code_is_ldrh1|code_is_ldrsb1|code_is_ldrsh1 )
    code_rm = {code[11:7],code[3:0]};
else if ( code_is_b )
    code_rm = {{6{code[23]}} ,code[23:0],2'b0};
else if ( code_is_ldm )
    case( code[24:23] )
        2'd0 : code_rm = {(code_sum_m - 1'b1),2'b0};
        2'd1 : code_rm = 0;
        2'd2 : code_rm = {code_sum_m,2'b0};
        2'd3 : code_rm = 3'b100;
    endcase
else if ( code_is_ldr0 )
    code_rm = code[11:0];
else if ( code_is_msr1|code_is_dp2 )
    code_rm = code[7:0];
else if ( code_is_mult1 & code[22] & code_rma[31] )

```

```

        code_rm = ~code_rma + 1'b1;
else    if    (    (    (code[6:5]==2'b10)    &    code_rma[31]    )    &
(code_is_dp0|code_is_dp1|code_is_ldr1)    )
        code_rm = ~code_rma;
else
        code_rm = code_rma;

```

```

always @( * )

```

```

case ( code[3:0] )

```

```

4'h0 : code_rma = r0;

```

```

4'h1 : code_rma = r1;

```

```

4'h2 : code_rma = r2;

```

```

4'h3 : code_rma = r3;

```

```

4'h4 : code_rma = r4;

```

```

4'h5 : code_rma = r5;

```

```

4'h6 : code_rma = r6;

```

```

4'h7 : code_rma = r7;

```

```

4'h8 : code_rma = r8;

```

```

4'h9 : code_rma = r9;

```

```

4'ha : code_rma = ra;

```

```

4'hb : code_rma = rb;

```

```

4'hc : code_rma = rc;

```

```

4'hd : code_rma = rd;

```

```

4'he : code_rma = re;

```

```

4'hf : code_rma = code_is_dp1 ? (rf+4'b1000) : (rf+3'b100);

```

```

endcase

```

```

always @( * )

```

```

if ( code_is_dp0|code_is_ldr1 )

```

```

        code_rot_num = ( code[6:5] == 2'b00 ) ? code[11:7] : ( ~code[11:7]+1'b1 );

```

```

else if ( code_is_dp1 )
    code_rot_num = ( code[6:5] == 2'b00 ) ? code_rsa[4:0] : ( ~code_rsa[4:0]+1'b1
);
else if ( code_is_msrl|code_is_dp2 )
    code_rot_num = { (~code[11:8]+1'b1),1'b0 };
else
    code_rot_num = 5'b0;

always @ ( * )
if ( code_is_mult1 )
    if ( code[22] & code_rsa[31] )
        code_rs = ~code_rsa + 1'b1;
    else
        code_rs = code_rsa;
else if ( code_is_mult )
    code_rs = code_rsa;
else begin
    code_rs = 32'b0;
    code_rs[code_rot_num] = 1'b1;
end

always @ ( posedge clk or posedge rst )
if ( rst )
    code_rs_flag <= #`DEL 3'd0;
else if ( cpu_en )
    if ( ~hold_en )
        if ( code_is_dp1 )
            code_rs_flag <= #`DEL
{(code_rsa[7:0]>6'd32),(code_rsa[7:0]==6'd32),(code_rsa[7:0]==8'd0)};
        else

```

```

        code_rs_flag <= #'DEL 0;
    else;
else;

always @( * )
case ( code[11:8] )
4'h0 : code_rsa = r0;
4'h1 : code_rsa = r1;
4'h2 : code_rsa = r2;
4'h3 : code_rsa = r3;
4'h4 : code_rsa = r4;
4'h5 : code_rsa = r5;
4'h6 : code_rsa = r6;
4'h7 : code_rsa = r7;
4'h8 : code_rsa = r8;
4'h9 : code_rsa = r9;
4'ha : code_rsa = ra;
4'hb : code_rsa = rb;
4'hc : code_rsa = rc;
4'hd : code_rsa = rd;
4'he : code_rsa = re;
4'hf : code_rsa = code_is_dp1 ? (rf+4'b1000) : (rf+3'b100);
endcase

always @( posedge clk or posedge rst )
if ( rst )
    code_und <= #'DEL 1'd0;
else if ( cpu_en )
    if ( ~hold_en )
        code_und <= #'DEL ~all_code;

```

```

    else;

else;

always @( * )
case ( cmd[31:28] )
4'h0 : cond_satisfy = ( cpsr_z==1'b1 );
4'h1 : cond_satisfy = ( cpsr_z==1'b0 );
4'h2 : cond_satisfy = ( cpsr_c==1'b1 );
4'h3 : cond_satisfy = ( cpsr_c==1'b0 );
4'h4 : cond_satisfy = ( cpsr_n==1'b1 );
4'h5 : cond_satisfy = ( cpsr_n==1'b0 );
4'h6 : cond_satisfy = ( cpsr_v==1'b1 );
4'h7 : cond_satisfy = ( cpsr_v==1'b0 );
4'h8 : cond_satisfy = ( cpsr_c==1'b1 )&(cpsr_z==1'b0);
4'h9 : cond_satisfy = ( cpsr_c==1'b0)|(cpsr_z==1'b1);
4'ha : cond_satisfy = ( cpsr_n==cpsr_v);
4'hb : cond_satisfy = ( cpsr_n!=cpsr_v);
4'hc : cond_satisfy = ( cpsr_z==1'b0)&(cpsr_n==cpsr_v);
4'hd : cond_satisfy = ( cpsr_z==1'b1)|(cpsr_n!=cpsr_v);
4'he : cond_satisfy = 1'b1;
4'hf : cond_satisfy = 1'b0;
endcase

always @( posedge clk or posedge rst )
if ( rst )
    cpsr_c <= #`DEL 1'd0;
else if ( cpu_en )
    if ( cmd_ok )
        if ( cmd_is_msr0|cmd_is_msr1 )
            if ( ~cmd[22] & cmd[19] )

```

```

        cpsr_c <= #`DEL  reg_ans[29];
    else;
    else if ( cmd_is_dp0|cmd_is_dp1|cmd_is_dp2 )
        if ( cmd[20] )
            if ( cmd[15:12]==4'hf )
                cpsr_c <= #`DEL  spsr[8];
            else
                if
                ( (cmd[24:21]==4'b1011)|(cmd[24:21]==4'b0100)|(cmd[24:21]==4'b0101)|(cmd[24:21]==4'b0011)|(cmd[24:21]==4'b0111) )
                    cpsr_c <= #`DEL  bit_cy;
                else
                    if
                    ( (cmd[24:21]==4'b1010)|(cmd[24:21]==4'b0010)|(cmd[24:21]==4'b0110) )
                        cpsr_c <= #`DEL  ~bit_cy;
                else if ( cmd_is_dp1 & ~code_rs_flag[0] )
                    case ( cmd[6:5] )
                        2'b00 : cpsr_c <= #`DEL  code_rs_flag[2]? 1'b0 :
                            ( code_rs_flag[1]? reg_ans[0] : reg_ans[32] );
                        2'b01 : cpsr_c <= #`DEL  code_rs_flag[2]? 1'b0 :
                            ( code_rs_flag[1]? reg_ans[31] : reg_ans[31] );
                        2'b10 : cpsr_c <= #`DEL  code_rs_flag[2]? rm_msb :
                            ( code_rs_flag[1]? rm_msb : reg_ans[31] );
                        2'b11 : cpsr_c <= #`DEL  code_rs_flag[1]? cpsr_c :
                            reg_ans[31];
                    endcase
                else if ( cmd_is_dp2 )
                    cpsr_c <= #`DEL  reg_ans[31];
                else if ( cmd_is_dp0 )
                    case ( cmd[6:5] )
                        2'b00 : cpsr_c <= #`DEL  (cmd[11:7]==5'b0) ? cpsr_c :
                            reg_ans[32];

```

```

                2'b01 : cpsr_c <= #`DEL   reg_ans[31];
                2'b10  :  cpsr_c  <= #`DEL   (cmd[11:7]==5'b0)  ?
reg_ans[31]:reg_ans[31];
                2'b11  :  cpsr_c  <= #`DEL   (cmd[11:7]==5'b0)  ?
reg_ans[0]:reg_ans[31];
                endcase
                else;
                else;
                else if ( cmd_is_ldm & ( cmd_sum_m==4'b0 ) & ldm_change )
                    cpsr_c <= #`DEL   spsr[8];
                else;
                else;
else;

always @( posedge clk or posedge rst )
if ( rst )
    cpsr_f <= #`DEL 1'd0;
else if ( cpu_en )
    if ( cpu_restart | fiq_en )
        cpsr_f <= #`DEL 1;
    else if ( cmd_ok & ( cpsr_m != 5'b10000 ) )
        if ( cmd_is_msr0|cmd_is_msr1 )
            if ( ~cmd[22] & cmd[16] )
                cpsr_f <= #`DEL   reg_ans[6];
            else;
        else if ( cmd_is_dp0|cmd_is_dp1|cmd_is_dp2 )
            if ( cmd[20] )
                if ( cmd[15:12]==4'hf )
                    cpsr_f <= #`DEL   spsr[5];
                else;

```

```

        else;
    else if ( cmd_is_ldm & ( cmd_sum_m==4'b0 ) & ldm_change )
        cpsr_f <= #`DEL  spsr[5];
    else;
else;
else;

always @ ( posedge clk or posedge rst )
if ( rst )
    cpsr_i <= #`DEL 1'd0;
else if ( cpu_en )
    if ( int_all )
        cpsr_i <= #`DEL 1;
    else if ( cmd_ok & ( cpsr_m != 5'b10000 ) )
        if ( cmd_is_msr0|cmd_is_msr1 )
            if ( ~cmd[22] & cmd[16] )
                cpsr_i <= #`DEL  reg_ans[7];
            else;
        else if ( cmd_is_dp0|cmd_is_dp1|cmd_is_dp2 )
            if ( cmd[20] )
                if ( cmd[15:12]==4'hf )
                    cpsr_i <= #`DEL  spsr[6];
                else;
            else;
        else if ( cmd_is_ldm & ( cmd_sum_m==4'b0 ) & ldm_change )
            cpsr_i <= #`DEL  spsr[6];
        else;
    else;
else;

```

```

always @ ( posedge clk or posedge rst )
if ( rst )
    cpsr_m <= #`DEL 5'b10011;
else if ( cpu_en )
    if ( cpu_restart )
        cpsr_m <= #`DEL 5'b10011;
    else if ( fiq_en )
        cpsr_m <= #`DEL 5'b10001;
    else if ( ram_abort )
        cpsr_m <= #`DEL 5'b10111;
    else if ( irq_en )
        cpsr_m <= #`DEL 5'b10010;
    else if ( cmd_flag & code_abort )
        cpsr_m <= #`DEL 5'b10111;
    else if ( cmd_flag & code_und )
        cpsr_m <= #`DEL 5'b11011;
    else if ( cmd_flag & cond_satisfy & cmd_is_swi )
        cpsr_m <= #`DEL 5'b10011;
    else if ( cmd_ok & ( cpsr_m != 5'b10000 ) )
        if ( cmd_is_msr0|cmd_is_msr1 )
            if ( ~cmd[22] & cmd[16] )
                cpsr_m <= #`DEL reg_ans[4:0];
            else;
        else if ( cmd_is_dp0|cmd_is_dp1|cmd_is_dp2 )
            if ( cmd[20] )
                if ( cmd[15:12]==4'hf )
                    cpsr_m <= #`DEL spsr[4:0];
                else;
            else;
        else if ( cmd_is_ldm & ( cmd_sum_m==4'b0 ) & ldm_change )

```

```

        cpsr_m <= #`DEL  spsr[4:0];
    else;
else;
else;

always @( posedge clk or posedge rst )
if ( rst )
    cpsr_n <= #`DEL 1'd0;
else if ( cpu_en )
    if ( cmd_ok )
        if ( cmd_is_msr0|cmd_is_msr1 )
            if ( ~cmd[22] & cmd[19] )
                cpsr_n <= #`DEL  reg_ans[31];
            else;
        else if ( cmd_is_dp0|cmd_is_dp1|cmd_is_dp2 )
            if ( cmd[20] )
                if ( cmd[15:12]==4'hf )
                    cpsr_n <= #`DEL  spsr[10];
                else
                    cpsr_n <= #`DEL  dp_ans[31];
            else;
        else if ( cmd_is_mult|cmd_is_multlx )
            if ( cmd[20] )
                cpsr_n <= #`DEL  sum_m_m[31];
            else;
        else if ( cmd_is_ldm & ( cmd_sum_m==4'b0 ) & ldm_change )
            cpsr_n <= #`DEL  spsr[10];
        else;
    else;
else;

```

```

always @( posedge clk or posedge rst )
if ( rst )
    cpsr_v <= #`DEL 1'd0;
else if ( cpu_en )
    if ( cmd_ok )
        if ( cmd_is_msr0|cmd_is_msr1 )
            if ( ~cmd[22] & cmd[19] )
                cpsr_v <= #`DEL reg_ans[28];
            else;
        else if ( cmd_is_dp0|cmd_is_dp1|cmd_is_dp2 )
            if ( cmd[20] )
                if ( cmd[15:12]==4'hf )
                    cpsr_v <= #`DEL spsr[7];
                else
                    if
( (cmd[24:21]==4'd2)|(cmd[24:21]==4'd3)|(cmd[24:21]==4'd4)|(cmd[24:21]==4'd5)|(
cmd[24:21]==4'd6)|(cmd[24:21]==4'd7)|(cmd[24:21]==4'd10)|(cmd[24:21]==4'd11) )
                        cpsr_v <= #`DEL bit_ov;
                    else;
                else;
            else if ( cmd_is_ldm & ( cmd_sum_m==4'b0 ) & ldm_change )
                cpsr_v <= #`DEL spsr[7];
            else;
        else;
else;

always @( posedge clk or posedge rst )
if ( rst )
    cpsr_z <= #`DEL 1'd0;
else if ( cpu_en )

```

```

if ( cmd_ok )
    if ( cmd_is_msr0|cmd_is_msr1 )
        if ( ~cmd[22] & cmd[19] )
            cpsr_z <= #`DEL   reg_ans[30];
        else;
    else if ( cmd_is_dp0|cmd_is_dp1|cmd_is_dp2 )
        if ( cmd[20] )
            if ( cmd[15:12]==4'hf )
                cpsr_z <= #`DEL   spsr[9];
            else
                cpsr_z <= #`DEL   (dp_ans==32'b0);
            else;
    else if ( cmd_is_mult|cmd_is_multl )
        if ( cmd[20] )
            cpsr_z <= #`DEL   (sum_rm_rm==32'b0);
        else;
    else if ( cmd_is_multlx & cmd[20] )
        cpsr_z <= #`DEL   cpsr_z & (sum_rm_rm==32'b0);
    else if ( cmd_is_ldm & ( cmd_sum_m==4'b0 ) & ldm_change )
        cpsr_z <= #`DEL   spsr[9];
    else;
else;

always @ ( * )
case ( cmd[24:21] )
4'h0 : dp_ans =   and_ans;
4'h1 : dp_ans =  eor_ans;
4'h2 : dp_ans =  sum_rm_rm;
4'h3 : dp_ans =  ~sum_rm_rm;

```

```

4'h4 : dp_ans = sum_rn_rm;
4'h5 : dp_ans = sum_rn_rm;
4'h6 : dp_ans = sum_rn_rm;
4'h7 : dp_ans = ~sum_rn_rm;
4'h8 : dp_ans = and_ans;
4'h9 : dp_ans = eor_ans;
4'ha : dp_ans = sum_rn_rm;
4'hb : dp_ans = sum_rn_rm;
4'hc : dp_ans = or_ans;
4'hd : dp_ans = sum_rn_rm;
4'he : dp_ans = bic_ans;
4'hf : dp_ans = sum_rn_rm;
endcase

```

```

always @( * )

```

```

if ( cmd_is_mult | cmd_is_multl )

```

```

    extra_num = 1'b0;

```

```

else if ( cmd_is_dp0|cmd_is_dp1|cmd_is_dp2 )

```

```

    case ( cmd[24:21] )

```

```

4'b0010 : extra_num = 1'b0;

```

```

4'b0011 : extra_num = 1'b1;

```

```

4'b0100 : extra_num = 1'b0;

```

```

4'b0101 : extra_num = cpsr_c;

```

```

4'b0110 : extra_num = ~cpsr_c;

```

```

4'b0111 : extra_num = cpsr_c;

```

```

4'b1111 : extra_num = 1'b1;

```

```

    default: extra_num = 1'b0;

```

```

    endcase

```

```

else if ( cmd_is_multlx )

```

```

    extra_num = multl_extra_num;

```

```

else
    extra_num = 1'b0;

always @(posedge clk or posedge rst)
if (rst)
    fiq_flag <= #`DEL 1'b0;
else if (cpu_en)
    if (fiq)
        fiq_flag <= #`DEL 1'b1;
    else if (cmd_flag)
        fiq_flag <= #`DEL 1'b0;
    else;
else;

always @(*)
if (go_fmt[5])
    go_data = ram_rdata;
else if (go_fmt[4])
    if (go_fmt[1])
        go_data = {{16{go_fmt[2]&ram_rdata[31]}},ram_rdata[31:16]};
    else
        go_data = {{16{go_fmt[2]&ram_rdata[15]}},ram_rdata[15:0]};
else// if (cha_reg_fmt[3])
    case(go_fmt[1:0])
        2'b00 : go_data = { {24{go_fmt[2]&ram_rdata[7]}}, ram_rdata[7:0] };
        2'b01 : go_data = { {24{go_fmt[2]&ram_rdata[15]}}, ram_rdata[15:8] };
        2'b10 : go_data = { {24{go_fmt[2]&ram_rdata[23]}}, ram_rdata[23:16] };
        2'b11 : go_data = { {24{go_fmt[2]&ram_rdata[31]}}, ram_rdata[31:24] };
    endcase

```

```

always @(posedge clk or posedge rst)
if (rst)
    go_fmt <= #`DEL 6'd0;
else if (cpu_en)
    if (cmd_is_ldr0|cmd_is_ldr1|cmd_is_swp)
        go_fmt <= #`DEL cmd[22] ?{4'b0010,cmd_addr[1:0]}:
{4'b1000,cmd_addr[1:0]};
    else if (cmd_is_ldrh0|cmd_is_ldrh1)
        go_fmt <= #`DEL {4'b0100,cmd_addr[1:0]};
    else if (cmd_is_ldrsb0|cmd_is_ldrsb1)
        go_fmt <= #`DEL {4'b0011,cmd_addr[1:0]};
    else if (cmd_is_ldrsh0|cmd_is_ldrsh1)
        go_fmt <= #`DEL {4'b0101,cmd_addr[1:0]};
    else if (cmd_is_ldm)
        go_fmt <= #`DEL {4'b1000,cmd_addr[1:0]};
    else;
else;

```

```

always @(posedge clk or posedge rst)
if (rst)
    go_num <= #`DEL 4'd0;
else if (cpu_en)
    go_num <= #`DEL cha_num;
else;

```

```

always @(posedge clk or posedge rst)
if (rst)
    go_vld <= #`DEL 1'd0;
else if (cpu_en)
    go_vld <= #`DEL cha_vld;

```

```
else;
```

```
always @(posedge clk or posedge rst)
```

```
if (rst)
```

```
    irq_flag <= #`DEL 1'b0;
```

```
else if (cpu_en)
```

```
    if (irq)
```

```
        irq_flag <= #`DEL 1'b1;
```

```
    else if (cmd_flag)
```

```
        irq_flag <= #`DEL 1'b0;
```

```
    else;
```

```
else;
```

```
always @(posedge clk or posedge rst)
```

```
if (rst)
```

```
    ldm_change <= #`DEL 1'd0;
```

```
else if (cpu_en)
```

```
    if (~hold_en)
```

```
        ldm_change <= #`DEL code[22] & code[20] & code[15];
```

```
    else;
```

```
else;
```

```
always @(posedge clk or posedge rst)
```

```
if (rst)
```

```
    ldm_num <= #`DEL 4'd0;
```

```
else if (cpu_en)
```

```
    if (cmd_is_ldm)
```

```
        ldm_num <= #`DEL ldm_sel;
```

```
    else;
```

```
else;
```

```
always @( * )
if ( cmd[0] )
    ldm_sel = 4'h0;
else if ( cmd[1] )
    ldm_sel = 4'h1;
else if ( cmd[2] )
    ldm_sel = 4'h2;
else if ( cmd[3] )
    ldm_sel = 4'h3;
else if ( cmd[4] )
    ldm_sel = 4'h4;
else if ( cmd[5] )
    ldm_sel = 4'h5;
else if ( cmd[6] )
    ldm_sel = 4'h6;
else if ( cmd[7] )
    ldm_sel = 4'h7;
else if ( cmd[8] )
    ldm_sel = 4'h8;
else if ( cmd[9] )
    ldm_sel = 4'h9;
else if ( cmd[10] )
    ldm_sel = 4'ha;
else if ( cmd[11] )
    ldm_sel = 4'hb;
else if ( cmd[12] )
    ldm_sel = 4'hc;
else if ( cmd[13] )
    ldm_sel = 4'hd;
```

```

else if ( cmd[14] )
    ldm_sel = 4'he;
else if ( cmd[15] )
    ldm_sel = 4'hf;
else
    ldm_sel = 4'h0;

always @ ( posedge clk or posedge rst )
if ( rst )
    ldm_usr <= #`DEL 1'd0;
else if ( cpu_en )
    ldm_usr <= #`DEL  cmd_is_ldm & cmd[20] &  cmd[22] & ~cmd[15];
else;

always @ ( posedge clk or posedge rst )
if ( rst )
    ldm_vld <= #`DEL 1'd0;
else if ( cpu_en )
    ldm_vld <= #`DEL  cmd_ok & cmd_is_ldm & cmd[20] & (cmd_sum_m!=4'b0);
else;

always @ ( posedge clk or posedge rst )
if ( rst )
    multl_extra_num <= #`DEL 1'd0;
else if ( cpu_en )
    if ( cmd_ok & cmd_is_multl )
        multl_extra_num <= #`DEL  bit_cy;
    else
        multl_extra_num <= #`DEL  0;
else;

```

```

always @(posedge clk or posedge rst)
if (rst)
    r0 <= #`DEL 32'd0;
else if (cpu_en)
    if (ldm_vld & (ldm_num==4'h0))
        r0 <= #`DEL ldm_data;
    else if (go_vld & (go_num==4'h0))
        r0 <= #`DEL go_data;
    else if (cmd_ok & to_vld & (to_num== 4'h0))
        r0 <= #`DEL to_data;
    else;
else;

```

```

always @(posedge clk or posedge rst)
if (rst)
    r1 <= #`DEL 32'd0;
else if (cpu_en)
    if (ldm_vld & (ldm_num==4'h1))
        r1 <= #`DEL ldm_data;
    else if (go_vld & (go_num==4'h1))
        r1 <= #`DEL go_data;
    else if (cmd_ok & to_vld & (to_num== 4'h1))
        r1 <= #`DEL to_data;
    else;
else;

```

```

always @(posedge clk or posedge rst)
if (rst)
    r2 <= #`DEL 32'd0;

```

```

else if ( cpu_en )
    if ( ldm_vld & ( ldm_num==4'h2 ) )
        r2 <= #`DEL  ldm_data;
    else if ( go_vld & (go_num==4'h2 ) )
        r2 <= #`DEL  go_data;
    else if ( cmd_ok & to_vld & ( to_num== 4'h2 ) )
        r2 <= #`DEL  to_data;
    else;
else;

```

```

always @( posedge clk or posedge rst )

```

```

if ( rst )

```

```

    r3 <= #`DEL 32'd0;

```

```

else if ( cpu_en )

```

```

    if ( ldm_vld & ( ldm_num==4'h3 ) )

```

```

        r3 <= #`DEL  ldm_data;

```

```

    else if ( go_vld & (go_num==4'h3 ) )

```

```

        r3 <= #`DEL  go_data;

```

```

    else if ( cmd_ok & to_vld & ( to_num== 4'h3 ) )

```

```

        r3 <= #`DEL  to_data;

```

```

    else;

```

```

else;

```

```

always @( posedge clk or posedge rst )

```

```

if ( rst )

```

```

    r4 <= #`DEL 32'd0;

```

```

else if ( cpu_en )

```

```

    if ( ldm_vld & ( ldm_num==4'h4 ) )

```

```

        r4 <= #`DEL  ldm_data;

```

```

    else if ( go_vld & (go_num==4'h4 ) )

```

```

        r4 <= #`DEL go_data;
    else if ( cmd_ok & to_vld & ( to_num== 4'h4 ) )
        r4 <= #`DEL to_data;
    else;
else;

always @( posedge clk or posedge rst )
if ( rst )
    r5 <= #`DEL 32'd0;
else if ( cpu_en )
    if ( ldm_vld & ( ldm_num==4'h5 ) )
        r5 <= #`DEL ldm_data;
    else if ( go_vld & ( go_num==4'h5 ) )
        r5 <= #`DEL go_data;
    else if ( cmd_ok & to_vld & ( to_num== 4'h5 ) )
        r5 <= #`DEL to_data;
    else;
else;

always @( posedge clk or posedge rst )
if ( rst )
    r6 <= #`DEL 32'd0;
else if ( cpu_en )
    if ( ldm_vld & ( ldm_num==4'h6 ) )
        r6 <= #`DEL ldm_data;
    else if ( go_vld & ( go_num==4'h6 ) )
        r6 <= #`DEL go_data;
    else if ( cmd_ok & to_vld & ( to_num== 4'h6 ) )
        r6 <= #`DEL to_data;
    else;

```

else;

always @(posedge clk or posedge rst)

if (rst)

    r7 <= #`DEL 32'd0;

else if (cpu\_en)

    if (ldm\_vld & (ldm\_num==4'h7))

        r7 <= #`DEL ldm\_data;

    else if (go\_vld & (go\_num==4'h7))

        r7 <= #`DEL go\_data;

    else if (cmd\_ok & to\_vld & (to\_num==4'h7))

        r7 <= #`DEL to\_data;

    else;

else;

always @(posedge clk or posedge rst)

if (rst)

    r8\_fiq <= #`DEL 32'd0;

else if (cpu\_en)

    if (ldm\_vld & (ldm\_num==4'h8) & (~ldm\_usr & (cpsr\_m==5'b10001)))

        r8\_fiq <= #`DEL ldm\_data;

    else if (go\_vld & (go\_num==4'h8) & (cpsr\_m==5'b10001))

        r8\_fiq <= #`DEL go\_data;

    else if (cmd\_ok & to\_vld & (to\_num==4'h8) & (cpsr\_m==5'b10001))

        r8\_fiq <= #`DEL to\_data;

    else;

else;

always @(posedge clk or posedge rst)

if (rst)

```

    r8_usr <= #`DEL 32'd0;
else if ( cpu_en )
    if ( ldm_vld & ( ldm_num==4'h8 ) & ( ldm_usr | (cpsr_m!=5'b10001 ) ) )
        r8_usr <= #`DEL  ldm_data;
    else if ( go_vld & (go_num==4'h8 ) & (cpsr_m!=5'b10001 ) )
        r8_usr <= #`DEL  go_data;
    else if ( cmd_ok & to_vld & ( to_num== 4'h8 ) & (cpsr_m!=5'b10001 ) )
        r8_usr <= #`DEL  to_data;
    else;
else;

always @( posedge clk or posedge rst )
if ( rst )
    r9_fiq <= #`DEL 32'd0;
else if ( cpu_en )
    if ( ldm_vld & ( ldm_num==4'h9 ) & ( ~ldm_usr & (cpsr_m==5'b10001 ) ) )
        r9_fiq <= #`DEL  ldm_data;
    else if ( go_vld & (go_num==4'h9 ) & (cpsr_m==5'b10001 ) )
        r9_fiq <= #`DEL  go_data;
    else if ( cmd_ok & to_vld  & ( to_num== 4'h9 ) & (cpsr_m==5'b10001 ) )
        r9_fiq <= #`DEL  to_data;
    else;
else;

always @( posedge clk or posedge rst )
if ( rst )
    r9_usr <= #`DEL 32'd0;
else if ( cpu_en )
    if ( ldm_vld & ( ldm_num==4'h9 ) & ( ldm_usr | (cpsr_m!=5'b10001 ) ) )
        r9_usr <= #`DEL  ldm_data;

```

```

else if ( go_vld & (go_num==4'h9) & (cpsr_m!=5'b10001) )
    r9_usr <= #`DEL go_data;
else if ( cmd_ok & to_vld & ( to_num== 4'h9 ) & (cpsr_m!=5'b10001) )
    r9_usr <= #`DEL to_data;
else;

else;

always @( posedge clk or posedge rst )
if ( rst )
    ra_fiq <= #`DEL 32'd0;
else if ( cpu_en )
    if ( ldm_vld & ( ldm_num==4'ha ) & ( ~ldm_usr & (cpsr_m==5'b10001) ) )
        ra_fiq <= #`DEL ldm_data;
    else if ( go_vld & (go_num==4'ha) & (cpsr_m==5'b10001) )
        ra_fiq <= #`DEL go_data;
    else if ( cmd_ok & to_vld & ( to_num== 4'ha ) & (cpsr_m==5'b10001) )
        ra_fiq <= #`DEL to_data;
    else;

else;

always @( posedge clk or posedge rst )
if ( rst )
    ra_usr <= #`DEL 32'd0;
else if ( cpu_en )
    if ( ldm_vld & ( ldm_num==4'ha ) & ( ldm_usr | (cpsr_m!=5'b10001) ) )
        ra_usr <= #`DEL ldm_data;
    else if ( go_vld & (go_num==4'ha) & (cpsr_m!=5'b10001) )
        ra_usr <= #`DEL go_data;
    else if ( cmd_ok & to_vld & ( to_num== 4'ha ) & (cpsr_m!=5'b10001) )
        ra_usr <= #`DEL to_data;

```

```

    else;
else;

always @( * )
if ( cmd_is_ldr0|cmd_is_ldr1|cmd_is_swp|cmd_is_swpX )
    ram_flag =  cmd[22]? (1'b1<<cmd_addr[1:0]):4'b1111;
else if (cmd_is_ldrh0|cmd_is_ldrh1|cmd_is_ldrsh0|cmd_is_ldrsh1 )
    ram_flag =  cmd_addr[1] ? 4'b1100 : 4'b0011;
else if ( cmd_is_ldrsb0|cmd_is_ldrsb1 )
    ram_flag =  1'b1<<cmd_addr[1:0];
else
    ram_flag =  4'b1111;

```

```

always @( * )
if ( cmd_is_ldm )
    if ( cmd[0] )
        ram_wdata =  r0;
    else if ( cmd[1] )
        ram_wdata =  r1;
    else if ( cmd[2] )
        ram_wdata =  r2;
    else if ( cmd[3] )
        ram_wdata =  r3;
    else if ( cmd[4] )
        ram_wdata =  r4;
    else if ( cmd[5] )
        ram_wdata =  r5;
    else if ( cmd[6] )
        ram_wdata =  r6;
    else if ( cmd[7] )

```

```

        ram_wdata = r7;
else if ( cmd[8] )
        ram_wdata = cmd[22] ? r8_usr : r8;
else if ( cmd[9] )
        ram_wdata = cmd[22] ? r9_usr : r9;
else if ( cmd[10] )
        ram_wdata = cmd[22] ? ra_usr : ra;
else if ( cmd[11] )
        ram_wdata = cmd[22] ? rb_usr : rb;
else if ( cmd[12] )
        ram_wdata = cmd[22] ? rc_usr : rc;
else if ( cmd[13] )
        ram_wdata = cmd[22] ? rd_usr : rd;
else if ( cmd[14] )
        ram_wdata = cmd[22] ? re_usr : re;
else if ( cmd[15] )
        ram_wdata = rf;
else
        ram_wdata = 4'h0;
else if ( cmd_is_ldr0|cmd_is_ldr1|cmd_is_swp_x )
    if ( cmd[22] )
        case( cmd_addr[1:0] )
            2'b00 : ram_wdata = rna[7:0];
            2'b01 : ram_wdata = {rna[7:0],8'b0};
            2'b10 : ram_wdata = {rna[7:0],16'b0};
            2'b11 : ram_wdata = {rna[7:0],24'b0};
        endcase
    else
        ram_wdata = rna;
else if ( cmd_is_ldrh0|cmd_is_ldrh1 )

```

```

    if ( cmd_addr[1] )
        ram_wdata = {rma[15:0],16'b0};
    else
        ram_wdata = rma;
else
    ram_wdata = rma;

always @( posedge clk or posedge rst )
if ( rst )
    rb_fiq <= #`DEL 32'd0;
else if ( cpu_en )
    if ( ldm_vld & ( ldm_num==4'hb ) & ( ~ldm_usr & (cpsr_m==5'b10001 ) ) )
        rb_fiq <= #`DEL ldm_data;
    else if ( go_vld & (go_num==4'hb ) & (cpsr_m==5'b10001 ) )
        rb_fiq <= #`DEL go_data;
    else if ( cmd_ok & to_vld & ( to_num== 4'hb ) & (cpsr_m==5'b10001 ) )
        rb_fiq <= #`DEL to_data;
    else;
else;

always @( posedge clk or posedge rst )
if ( rst )
    rb_usr <= #`DEL 32'd0;
else if ( cpu_en )
    if ( ldm_vld & ( ldm_num==4'hb ) & ( ldm_usr | (cpsr_m!=5'b10001 ) ) )
        rb_usr <= #`DEL ldm_data;
    else if ( go_vld & (go_num==4'hb ) & (cpsr_m!=5'b10001 ) )
        rb_usr <= #`DEL go_data;
    else if ( cmd_ok & to_vld & ( to_num== 4'hb ) & (cpsr_m!=5'b10001 ) )
        rb_usr <= #`DEL to_data;

```

```

else;

else;

always @(posedge clk or posedge rst)
if (rst)
    rc_fiq <= #`DEL 32'd0;
else if (cpu_en)
    if (ldm_vld & (ldm_num==4'hc) & (~ldm_usr & (cpsr_m==5'b10001)))
        rc_fiq <= #`DEL ldm_data;
    else if (go_vld & (go_num==4'hc) & (cpsr_m==5'b10001))
        rc_fiq <= #`DEL go_data;
    else if (cmd_ok & to_vld & (to_num==4'hc) & (cpsr_m==5'b10001))
        rc_fiq <= #`DEL to_data;
    else;

else;

always @(posedge clk or posedge rst)
if (rst)
    rc_usr <= #`DEL 32'd0;
else if (cpu_en)
    if (ldm_vld & (ldm_num==4'hc) & (ldm_usr | (cpsr_m!=5'b10001)))
        rc_usr <= #`DEL ldm_data;
    else if (go_vld & (go_num==4'hc) & (cpsr_m!=5'b10001))
        rc_usr <= #`DEL go_data;
    else if (cmd_ok & to_vld & (to_num==4'hc) & (cpsr_m!=5'b10001))
        rc_usr <= #`DEL to_data;
    else;

else;

always @(*)

```

```

case ( cpsr_m )
5'b10001 : rd = rd_fiq;
5'b11011 : rd = rd_und;
5'b10010 : rd = rd_irq;
5'b10111 : rd = rd_abt;
5'b10011 : rd = rd_svc;
default  : rd = rd_usr;
endcase

always @ ( posedge clk or posedge rst )
if ( rst )
    rd_abt <= #`DEL 32'd0;
else if ( cpu_en )
    if ( ldm_vld & ( ldm_num==4'hd ) & ( ~ldm_usr & ( cpsr_m==5'b10111 ) ) )
        rd_abt <= #`DEL  ldm_data;
    else if ( go_vld & ( go_num==4'hd ) & ( cpsr_m==5'b10111 ) )
        rd_abt <= #`DEL  go_data;
    else if ( cmd_ok & to_vld  & ( to_num== 4'hd ) & ( cpsr_m==5'b10111 ) )
        rd_abt <= #`DEL  to_data;
    else;
else;

always @ ( posedge clk or posedge rst )
if ( rst )
    rd_fiq <= #`DEL 32'd0;
else if ( cpu_en )
    if ( ldm_vld & ( ldm_num==4'hd ) & ( ~ldm_usr & ( cpsr_m==5'b10001 ) ) )
        rd_fiq <= #`DEL  ldm_data;
    else if ( go_vld & ( go_num==4'hd ) & ( cpsr_m==5'b10001 ) )
        rd_fiq <= #`DEL  go_data;

```

```

else if ( cmd_ok & to_vld & ( to_num== 4'hd ) & ( cpsr_m==5'b10001 ) )
    rd_fiq <= #`DEL to_data;
else;

else;

always @( posedge clk or posedge rst )
if ( rst )
    rd_irq <= #`DEL 32'd0;
else if ( cpu_en )
    if ( ldm_vld & ( ldm_num==4'hd ) & ( ~ldm_usr & ( cpsr_m==5'b10010 ) ) )
        rd_irq <= #`DEL ldm_data;
    else if ( go_vld & ( go_num==4'hd ) & ( cpsr_m==5'b10010 ) )
        rd_irq <= #`DEL go_data;
    else if ( cmd_ok & to_vld & ( to_num== 4'hd ) & ( cpsr_m==5'b10010 ) )
        rd_irq <= #`DEL to_data;
    else;

else;

always @( posedge clk or posedge rst )
if ( rst )
    rd_svc <= #`DEL 32'd0;
else if ( cpu_en )
    if ( ldm_vld & ( ldm_num==4'hd ) & ( ~ldm_usr & ( cpsr_m==5'b10011 ) ) )
        rd_svc <= #`DEL ldm_data;
    else if ( go_vld & ( go_num==4'hd ) & ( cpsr_m==5'b10011 ) )
        rd_svc <= #`DEL go_data;
    else if ( cmd_ok & to_vld & ( to_num== 4'hd ) & ( cpsr_m==5'b10011 ) )
        rd_svc <= #`DEL to_data;
    else;

else;

```

```

always @(posedge clk or posedge rst)
if (rst)
    rd_und <= #`DEL 32'd0;
else if (cpu_en)
    if (ldm_vld & (ldm_num==4'hd) & (~ldm_usr & (cpsr_m==5'b11011)))
        rd_und <= #`DEL ldm_data;
    else if (go_vld & (go_num==4'hd) & (cpsr_m==5'b11011))
        rd_und <= #`DEL go_data;
    else if (cmd_ok & to_vld & (to_num== 4'hd) & (cpsr_m==5'b11011))
        rd_und <= #`DEL to_data;
    else;
else;

```

```

always @(posedge clk or posedge rst)
if (rst)
    rd_usr <= #`DEL 32'd0;
else if (cpu_en)
    if ( ldm_vld & ( ldm_num==4'hd ) & ( ldm_usr |
((cpsr_m==5'b10000)|(cpsr_m==5'b11111))) )
        rd_usr <= #`DEL ldm_data;
    else if ( go_vld & (go_num==4'hd) &
((cpsr_m==5'b10000)|(cpsr_m==5'b11111))) )
        rd_usr <= #`DEL go_data;
    else if ( cmd_ok & to_vld & ( to_num== 4'hd ) &
((cpsr_m==5'b10000)|(cpsr_m==5'b11111))) )
        rd_usr <= #`DEL to_data;
    else;
else;

```

```

always @( * )
case ( cpsr_m )
5'b10001 : re = re_fiq;
5'b11011 : re = re_und;
5'b10010 : re = re_irq;
5'b10111 : re = re_abt;
5'b10011 : re = re_svc;
default  : re = re_usr;
endcase

always @( posedge clk or posedge rst )
if ( rst )
    re_abt <= #`DEL 32'd0;
else if ( cpu_en )
    if ( ram_abort | ( ~fiq_en & ~irq_en & ( cmd_flag & code_abort ) ) )
        re_abt <= #`DEL  rf_b;
    else if ( ldm_vld & ( ldm_num==4'he ) & ( ~ldm_usr & ( cpsr_m==5'b10111 ) ) )
        re_abt <= #`DEL  ldm_data;
    else if ( go_vld & ( go_num==4'he ) & ( cpsr_m==5'b10111 ) )
        re_abt <= #`DEL  go_data;
    else if ( cmd_ok & cmd_is_b & cmd[24] & ( cpsr_m==5'b10111 ) )
        re_abt <= #`DEL  rf_b;
    else if ( cmd_ok & to_vld  & ( to_num== 4'he ) & ( cpsr_m==5'b10111 ) )
        re_abt <= #`DEL  to_data;
    else;
else;

always @( posedge clk or posedge rst )
if ( rst )
    re_fiq <= #`DEL 32'd0;

```

```

else if ( cpu_en )
    if ( fiq_en )
        if ( ram_abort )
            re_fiq <= #`DEL 32'h10;
        else
            re_fiq <= #`DEL rf_b;
    else if ( ldm_vld & ( ldm_num==4'he ) & ( ~ldm_usr & (cpsr_m==5'b10001) ) )
        re_fiq <= #`DEL ldm_data;
    else if ( go_vld & (go_num==4'he ) & (cpsr_m==5'b10001) )
        re_fiq <= #`DEL go_data;
    else if ( cmd_ok & cmd_is_b & cmd[24] & (cpsr_m==5'b10001) )
        re_fiq <= #`DEL rf_b;
    else if ( cmd_ok & to_vld & ( to_num== 4'he ) & (cpsr_m==5'b10001) )
        re_fiq <= #`DEL to_data;
    else;
else;

always @( posedge clk or posedge rst )
if ( rst )
    re_irq <= #`DEL 32'd0;
else if ( cpu_en )
    if ( ~ram_abort & ~fiq_en & irq_en )
        re_irq <= #`DEL rf_b;
    else if ( ldm_vld & ( ldm_num==4'he ) & ( ~ldm_usr & (cpsr_m==5'b10010) ) )
        re_irq <= #`DEL ldm_data;
    else if ( go_vld & (go_num==4'he ) & (cpsr_m==5'b10010) )
        re_irq <= #`DEL go_data;
    else if ( cmd_ok & cmd_is_b & cmd[24] & (cpsr_m==5'b10010) )
        re_irq <= #`DEL rf_b;
    else if ( cmd_ok & to_vld & ( to_num== 4'he ) & (cpsr_m==5'b10010) )

```

```

        re_irq <= #`DEL  to_data;
    else;
else;

always @( posedge clk or posedge rst )
if ( rst )
    re_svc <= #`DEL 32'd0;
else if ( cpu_en )
    if ( ~ram_abort & ~fiq_en & ~irq_en & ( cmd_flag & ~code_abort & ~code_und
& (cond_satisfy & cmd_is_swi) ) )
        re_svc <= #`DEL  rf_b;
    else if ( ldm_vld & ( ldm_num==4'he ) & ( ~ldm_usr & (cpsr_m==5'b10011) ) )
        re_svc <= #`DEL  ldm_data;
    else if ( go_vld & (go_num==4'he ) & (cpsr_m==5'b10011) )
        re_svc <= #`DEL  go_data;
    else if ( cmd_ok & cmd_is_b & cmd[24] & (cpsr_m==5'b10011) )
        re_svc <= #`DEL  rf_b;
    else if ( cmd_ok & to_vld  & ( to_num== 4'he ) & (cpsr_m==5'b10011) )
        re_svc <= #`DEL  to_data;
    else;
else;

always @( posedge clk or posedge rst )
if ( rst )
    re_und <= #`DEL 32'd0;
else if ( cpu_en )
    if ( ~ram_abort & ~fiq_en & ~irq_en & ( cmd_flag & ~code_abort & code_und )
)
        re_und <= #`DEL  rf_b;
    else if ( ldm_vld & ( ldm_num==4'he ) & ( ~ldm_usr & (cpsr_m==5'b11011) ) )

```

```

        re_und <= #`DEL ldm_data;
    else if ( go_vld & (go_num==4'he) & (cpsr_m==5'b11011) )
        re_und <= #`DEL go_data;
    else if ( cmd_ok & cmd_is_b & cmd[24] & (cpsr_m==5'b11011) )
        re_und <= #`DEL rf_b;
    else if ( cmd_ok & to_vld & ( to_num== 4'he ) & (cpsr_m==5'b11011) )
        re_und <= #`DEL to_data;
    else;
else;

always @( posedge clk or posedge rst )
if ( rst )
    re_usr <= #`DEL 32'd0;
else if ( cpu_en )
    if ( ldm_vld & ( ldm_num==4'he ) & ( ldm_usr |
((cpsr_m==5'b10000)|(cpsr_m==5'b11111)) ) )
        re_usr <= #`DEL ldm_data;
    else if ( go_vld & (go_num==4'he) &
((cpsr_m==5'b10000)|(cpsr_m==5'b11111)) )
        re_usr <= #`DEL go_data;
    else if ( cmd_ok & cmd_is_b & cmd[24] &
((cpsr_m==5'b10000)|(cpsr_m==5'b11111)) )
        re_usr <= #`DEL rf_b;
    else if ( cmd_ok & to_vld & ( to_num== 4'he ) &
((cpsr_m==5'b10000)|(cpsr_m==5'b11111)) )
        re_usr <= #`DEL to_data;
    else;
else;

always @( posedge clk or posedge rst )

```

```

if ( rst )
    reg_ans <= #`DEL 64'd0;
else if ( cpu_en )
    if ( ~hold_en )
        reg_ans <= #`DEL mult_ans;
    else if ( cmd_is_ldm )
        if ( cmd_sum_m==4'b1 )
            reg_ans[5:2] <= #`DEL sum_m;
        else if ( cmd[23] )
            reg_ans[5:2] <= #`DEL reg_ans[5:2] + 1'b1;
        else
            reg_ans[5:2] <= #`DEL reg_ans[5:2] - 1'b1;
    else;
else;

always @( posedge clk or posedge rst )
if ( rst )
    rf <= #`DEL 32'd0;
else if ( cpu_en )
    if ( cpu_restart )
        rf <= #`DEL 32'h0000_0000;
    else if ( fiq_en )
        rf <= #`DEL 32'h0000_001c;
    else if ( ram_abort )
        rf <= #`DEL 32'h0000_0010;
    else if ( irq_en )
        rf <= #`DEL 32'h0000_0018;
    else if ( cmd_flag & code_abort )
        rf <= #`DEL 32'h0000_000c;
    else if ( cmd_flag & code_und )

```

```

        rf <= #`DEL 32'h0000_0004;
    else if ( cmd_flag & cond_satisfy & cmd_is_swi )
        rf <= #`DEL 32'h0000_0008;
    else if ( ldm_vld & (ldm_num==4'hf) )
        rf <= #`DEL ldm_data;
    else if ( go_vld & (go_num==4'hf) )
        rf <= #`DEL go_data;
    else if ( cmd_ok & (cmd_is_dp0|cmd_is_dp1|cmd_is_dp2) &
( cmd[24:23]!=2'b10 ) & ( cmd[15:12]==4'hf ) )
        rf <= #`DEL dp_ans;
    else if ( cmd_ok & ( cmd_is_b | cmd_is_bx ) )
        rf <= #`DEL sum_rm_rm;
    else if ( ~hold_en & ~wait_en )
        rf <= #`DEL rf + 4;
    else;
else;

always @ ( posedge clk or posedge rst )
if ( rst )
    rm_msb <= #`DEL 1'd0;
else if ( cpu_en )
    if ( ~hold_en )
        rm_msb <= #`DEL code_rma[31];
    else;
else;

always @ ( * )
if
( cmd_is_bx|((cmd_is_dp0|cmd_is_dp1|cmd_is_dp2)&((cmd[24:21]==4'b1101)|(cmd[
24:21]==4'b1111))|(cmd_is_multlx & ~cmd[21]) )

```

```

    rn = 0;
else if ( cmd_is_mult|cmd_is_multl )
    if ( cmd[21] )
        rn = rna;
    else
        rn = 0;
else if ( cmd_is_b )
    rn = rf;
else
    rn = rnb;

always @( * )
case ( cmd[15:12] )
4'h0 : rna = r0;
4'h1 : rna = r1;
4'h2 : rna = r2;
4'h3 : rna = r3;
4'h4 : rna = r4;
4'h5 : rna = r5;
4'h6 : rna = r6;
4'h7 : rna = r7;
4'h8 : rna = r8;
4'h9 : rna = r9;
4'ha : rna = ra;
4'hb : rna = rb;
4'hc : rna = rc;
4'hd : rna = rd;
4'he : rna = re;
4'hf : rna = rf;
endcase

```

```

always @( * )
case ( cmd[19:16] )
4'h0 : rmb = r0;
4'h1 : rmb = r1;
4'h2 : rmb = r2;
4'h3 : rmb = r3;
4'h4 : rmb = r4;
4'h5 : rmb = r5;
4'h6 : rmb = r6;
4'h7 : rmb = r7;
4'h8 : rmb = r8;
4'h9 : rmb = r9;
4'ha : rmb = ra;
4'hb : rmb = rb;
4'hc : rmb = rc;
4'hd : rmb = rd;
4'he : rmb = re;
4'hf : rmb = rf;
endcase

```

```

always @( posedge clk or posedge rst )
if ( rst )
    rs_msb <= #'DEL 1'd0;
else if ( cpu_en )
    if ( ~hold_en )
        rs_msb <= #'DEL code_rsa[31];
    else;
else;

```

```

always @( * )
if ( cmd_is_dp0|cmd_is_ldr1 )
    case(cmd[6:5])
        2'b00 : sec_operand = reg_ans[31:0];
        2'b01 : sec_operand = (cmd[11:7]==5'b0)?32'b0 : reg_ans[63:32];
        2'b10 : sec_operand = (cmd[11:7]==5'b0){32{rm_msb}} : (rm_msb ?
~reg_ans[63:32] : reg_ans[63:32]);
        2'b11 : sec_operand = (cmd[11:7]==5'b0){cpsr,reg_ans[31:1]} :
(reg_ans[63:32]|reg_ans[31:0]);
    endcase
else if ( cmd_is_dp1 )
    case(cmd[6:5])
        2'b00 : sec_operand = ( code_rs_flag[2:1]!=2'b0 )? 32'b0: reg_ans[31:0];
        2'b01 : sec_operand = ( code_rs_flag[2:1]!=2'b0 )? 32'b0: ( code_rs_flag[0] ?
reg_ans[31:0] : reg_ans[63:32] );
        2'b10 : sec_operand = ( code_rs_flag[2:1]!=2'b0 )? {32{rm_msb}} :
( code_rs_flag[0] ? reg_ans[31:0] : ( rm_msb ? ~reg_ans[63:32]:reg_ans[63:32] ) );
        2'b11 : sec_operand = ( code_rs_flag[1]|code_rs_flag[0] ) ? reg_ans[31:0] :
( reg_ans[63:32]|reg_ans[31:0] );
    endcase
else if ( cmd_is_msr1|cmd_is_dp2 )
    sec_operand = reg_ans[63:32]|reg_ans[31:0];
else if ( cmd_is_multlx )
    sec_operand = reg_ans[63:32];
else
    sec_operand = reg_ans[31:0];

always @( * )
if ( cpsr_m == 5'b10011 )
    spsr = spsr_svc;

```

```

else if ( cpsr_m == 5'b10111 )
    spsr = spsr_abt;
else if ( cpsr_m == 5'b10010 )
    spsr = spsr_irq;
else if ( cpsr_m == 5'b10001 )
    spsr = spsr_fiq;
else if ( cpsr_m == 5'b11011 )
    spsr = spsr_und;
else
    spsr = cpsr;

always @( posedge clk or posedge rst )
if ( rst )
    spsr_abt <= #`DEL 11'd0;
else if ( cpu_en )
    if ( ram_abort | ( ~fiq_en & ~irq_en & ( cmd_flag & code_abort ) ) )
        spsr_abt <= #`DEL cpsr;
    else if ( cmd_ok & ( cpsr_m==5'b10111 ) & ( cmd_is_msr0|cmd_is_msr1 ) &
cmd[22] )
        spsr_abt <= #`DEL
{{cmd[15]?reg_ans[31:28]:spsr_abt[10:7]},{cmd[12]?{reg_ans[7:6],reg_ans[4:0]}:sp
sr_abt[6:0]}};
    else;
else;

always @( posedge clk or posedge rst )
if ( rst )
    spsr_fiq <= #`DEL 11'd0;
else if ( cpu_en )
    if ( fiq_en )

```

```

        if ( ram_abort )
            spsr_fiq <= #`DEL
{cpsr_n,cpsr_z,cpsr_c,cpsr_v,1'b1,cpsr_f,5'b10111};
        else
            spsr_fiq <= #`DEL  cpsr;
        else if ( cmd_ok & ( cpsr_m==5'b11011) & ( cmd_is_msr0|cmd_is_msr1 ) &
cmd[22] )
            spsr_fiq <= #`DEL
{{cmd[15]?reg_ans[31:28]:spsr_fiq[10:7]},{cmd[12]?{reg_ans[7:6],reg_ans[4:0]}:sp
sr_fiq[6:0]}};
        else;
else;

always @( posedge clk or posedge rst )
if ( rst )
    spsr_irq <= #`DEL 11'd0;
else if ( cpu_en )
    if ( ~ram_abort & ~fiq_en & irq_en )
        spsr_irq <= #`DEL  cpsr;
    else if ( cmd_ok & ( cpsr_m==5'b10010) & ( cmd_is_msr0|cmd_is_msr1 ) &
cmd[22] )
        spsr_irq <= #`DEL
{{cmd[15]?reg_ans[31:28]:spsr_irq[10:7]},{cmd[12]?{reg_ans[7:6],reg_ans[4:0]}:sp
sr_irq[6:0]}};
    else;
else;

always @( posedge clk or posedge rst )
if ( rst )
    spsr_svc <= #`DEL 11'd0;

```

```

else if ( cpu_en )
    if ( ~ram_abort & ~fiq_en & ~irq_en & ( cmd_flag & ~code_abort & ~code_und
& (cond_satisfy & cmd_is_swi) ) )
        spsr_svc <= #`DEL  cpsr;
    else if ( cmd_ok & ( cpsr_m==5'b10011) & ( cmd_is_msr0|cmd_is_msr1 ) &
cmd[22] )
        spsr_svc                                <=                                #`DEL
{{cmd[15]?reg_ans[31:28]:spsr_svc[10:7]},{cmd[12]?{reg_ans[7:6],reg_ans[4:0]}:sp
sr_svc[6:0]}};
    else;
else;

always @( posedge clk or posedge rst )
if ( rst )
    spsr_und <= #`DEL 11'd0;
else if ( cpu_en )
    if ( ~ram_abort & ~fiq_en & ~irq_en & ( cmd_flag & ~code_abort & code_und )
)
        spsr_und <= #`DEL  cpsr;
    else if ( cmd_ok & ( cpsr_m==5'b11011) & ( cmd_is_msr0|cmd_is_msr1 ) &
cmd[22] )
        spsr_und                                <=                                #`DEL
{{cmd[15]?reg_ans[31:28]:spsr_und[10:7]},{cmd[12]?{reg_ans[7:6],reg_ans[4:0]}:s
psr_und[6:0]}};
    else;
else;

always @( posedge clk or posedge rst )
if ( rst )
    sum_m <= #`DEL 4'd0;

```

```

else if ( cpu_en )
    if ( ~hold_en )
        sum_m <= #`DEL  code_sum_m;
    else;
else;

always @ ( * )
if ( cmd_is_mrs )
    to_data = cmd[22] ? {spsr[10:7],20'b0,spsr[6:5],1'b0,spsr[4:0]} :
    {cpsr[10:7],20'b0,cpsr[6:5],1'b0,cpsr[4:0]};
else if (cmd_is_dp0|cmd_is_dp1|cmd_is_dp2)
    to_data = dp_ans;
else
    to_data = sum_rn_rm;

always @ ( * )
if (cmd_is_mrs|(cmd_is_dp0|cmd_is_dp1|cmd_is_dp2)|cmd_is_multl)
    to_num = cmd[15:12];
else
    to_num = cmd[19:16];

/*****/

//function statement area

/*****/

endmodule

```