Rotate V1 Interface Hardware Description

Beken Corporation
41，1387 Zhangdong Rd, Shanghai 201203, China
PHONE: (86)21 5108 6811
FAX: (86)21 6087 1089

Revision History

Rev.	Date	Remark
0.1	20/Oct/2022	Leasung，Initial version

Contents

1. 模块简介

旋转模块（Rotate module）主要功能是将储存在 memory 的 yuv422 图像，旋转成 rgb565 格式的图像，完成后储存在另一块 memory 中。

模块共有三种工作模式：1.顺时针旋转模式；2.逆时针旋转模式；3.不旋转只进行格式转换。

旋转过程中将整张图像分解为多张小图像（Rblock），按 Rblock 从左至右，从上之下全部旋转完成。

2. 模块结构

Rotate Module 内部主要由 8 个部分组成。其结构如下图所示。

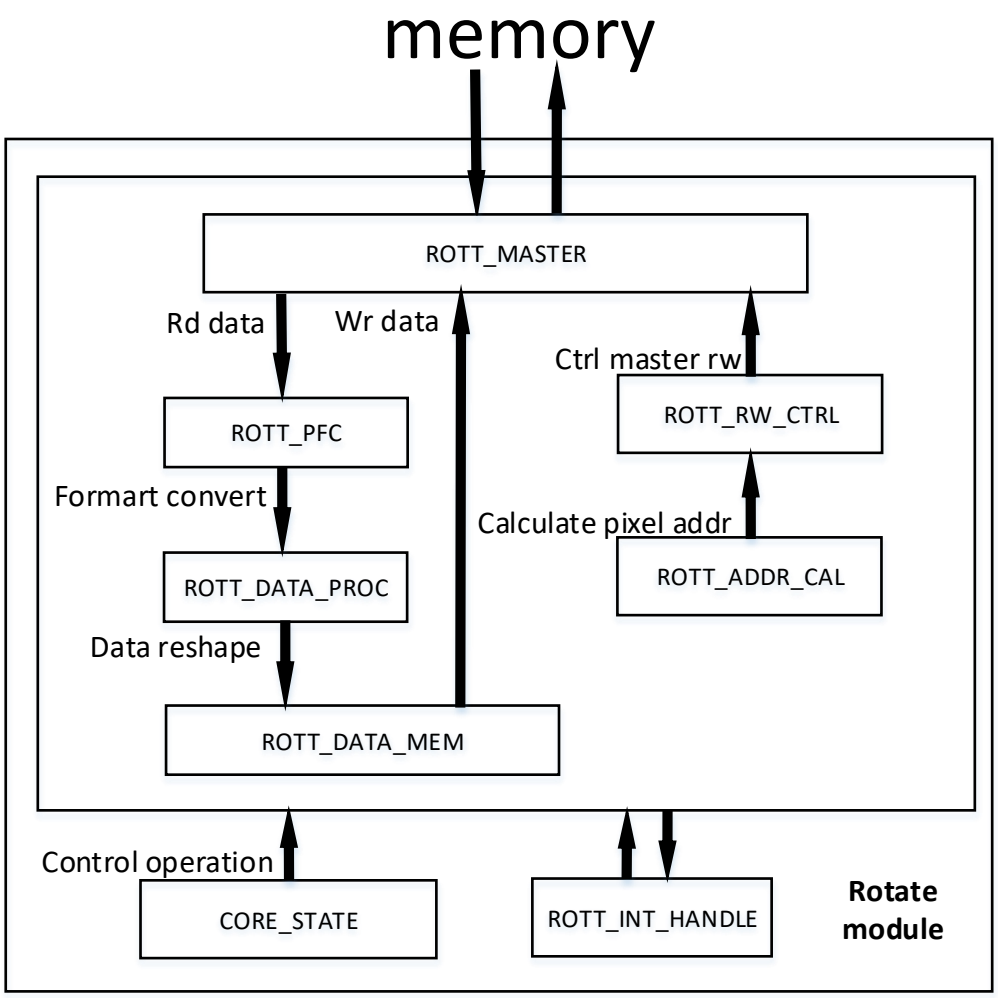


图 1.Rotate Module 结构图

Rotate Module 由 ROTT_MASTER, ROTT_PFC, ROTT_DATA_PROC, ROTT_DATA_MEM, ROTT_RW_CTRL, ROTT_ADDR_CAL, CORE_STATE 和 ROTT_INT_HANDLE8 个模块组成。每个模块执行具体的功能，模块对应的功能为：

1.ROTT_MASTER：与总线上存储区的交互模块。通过模块读写存储区的数据。将存储区的数据搬运至模块或将旋转后的数据写入存储区。

2.ROTT_PFC：格式转换模块，将写入的 yuv422 数据转换为 rgb565 格式数据。

3.ROTT_DATA_PROC：数据整形模块，将按行写入的 rgb565 数据，按位宽进行数据重组，分配好每个 pixel 数据对应 memory 的位置。整形完成后写入下一级的 memory 中。

4.ROTT_DATA_MEM：内部的数据存储 memory。将整形后重新排列的数据写入该地址。大小为 32bit*2400depth。所以每个 Rblock 最大为 4800 个 pixel。

5.ROTT_RW_CTRL： 对模块 master 的控制模块。产生相应信号，控制 rott_master 的读写。

6.ROTT_ADDR_CAL：产生模块 master 的读写地址。在每次读写传输时计算 master 需要读写的地址。

7.CORE_STATE：整个 ROTT_MODULE 的控制模块，控制整个模块的运行。

8.ROTT_INT_HANDLE：ROTT MODULE 的中断处理模块，产生和处理相应的中断。

3. 运行流程

Rotate Module 进行旋转操作时，并非是直接对整幅图像图像直接进行旋转，而是将图像分解为多个低像素的 Rblock（大小由用户控制）。

对每一个 Rblock 进行旋转，写入内部的 memory 中，旋转完成后直接将 Rblock 写入外部的存储区。

遵循从左至右，从上之下的顺序，对每一个 Rblock 旋转写入，直到整个图像完成。

实际运行流程如下所示：

在 Rotate Module 使能运行后。CORE_STATE 模块分配两个 cycle，计算出当前 Rblock 的读与写地址（在一整幅图的相对位置）。

之后 ROTT_RW_CTRL 模块控制 ROTT_MASTER 模块去存储区读取 Rblock 的单行的 yuv 数据，ROTT_PFC 将数据转换为按 pixel 的 rgb565 数据。

ROTT_DATA_PROC 将 ROTT_PFC 模块的 pixel 数据拆分，并对位置重新计算，按旋转后的位置按地址写入内部的 ROTT_DATA_MEM 中。

写入完成后，CORE_STATE 开始控制模块对该 Rblock 的下一单行进行旋转。直到 Rblock 的所有行旋转完成，CORE_STATE 控制 ROTT_RW_CTRL 模块，将 ROTT_DATA_MEM 内的 Rloblock 数据读出，通过 ROTT_MASTER 写入外部的存储区中。当前 Rblock 旋转完成。

当前 Rblock 旋转完成后，CORE_STATE 控制整个模块进行下一 Rblock 的旋转，直到检测到所有 Rblock 完成，产生相应的中断。退出旋转功能。

4. 系统参数

4.1. 寄存器列表

共 12 个寄存器，如下：

	映射基址	0x480c0000	
序号	**寄存器名**	**寄存器地址[bit]**	**控制信号名**
1	device_id	0x0[31:0]	device_id
2	version_id	0x1[31:0]	version_id
3	module_contol	0x2[1]	clk_gate
		0x2[0]	soft_reset
4	global_status	0x3[31:0]	global_Status
5	rotate_ctrl	0x4[31:6]	reserved
		0x4[6]	rotate_cfg_err_int_ena
		0x4[5]	rotate_wtmk_int_ena
		0x4[4]	rotate_done_int_ena
		0x4[1]	rotate_bps
		0x4[0]	rotate_ena
6	rotate_fmt	0x5[6:4]	yuv_fmt
		0x5[3]	rotate_anticlock
		0x5[2]	pfc_o_hword_reve
		0x5[1]	pfc_i_byte_reve
		0x5[0]	pfc_i_hword_reve
7	block_resolu	0x6[22:12]	blk_clum_pixel
		0x6[10:0]	blk_line_pixel
8	rd_addr	0x7[31:0]	rotate_rd_base_addr
9	wr_addr	0x8[31:0]	rotate_wr_base_addr
10	picture_resolu	0x9[22:12]	pic_clum_pixel
		0x9[10:0]	pic_line_pixel
11	block_count	0xa[15: 0]	rotate_blk_count
		0xa[26:16]	wtmk_clum_pixel
12	int_status	0xb[0]	rotate_finish
		0xb[1]	rotate_wtmk
		0xb[2]	rotate_cfg_err

表 1.寄存器列表

4.2. 寄存器描述

寄存器描述，如没有注出，则默认高电平有效。

1.reg0, device_id.

reg0[31:0]: device_id. ROTT 的 ascii 码。缺省值 0x524f5454。

2.reg1, version_id.

reg1[31:0]: version_id. 模块版本号 v1.1，缺省值 0x00010001。

3.reg2, module_ctrl.

Reg2[0]: soft_rest. 模块软复位。缺省值 1。需要使用时先置 0，等待一段时间再置 1。

Reg2[1]: clk_gate. 模块 bus 时钟门控单元，缺省值为 0。

4.reg4, rotate_ctrl.

Reg4[0]: roate_ena. 模块使能位。置 1 模块功能开启。

Reg4[1]: rotate_bps. Bypass 模块的旋转功能，只进行格式转换功能。

Reg4[4]: rotate_done_int_ena. 旋转完成中断使能。该中断指示整张图像旋转完成。

Reg4[5]: rotate_wtmk_int_ena. 旋转的水位中断使能。该中断指示旋转过程到达用户设置的水位位置。

Reg4[6]: rotate_cfg_err_int_ena.旋转设置错误中断使能。该中断指示用户的配置出现错误。

5.reg5, rotate_fmt.

Reg5[0]: pfc_i_hword_reve. 进入 pfc 的数据按半字翻转。

Reg5[1]: pfc_i_byte_reve. 进入 pfc 的数据按 byte 翻转。

Reg5[2]: pfc_o_hword_reve. 输出 pfc 的数据按半字翻转。

Reg5[3]: rotate_anticlock. 旋转进行逆时针旋转。默认顺时针旋转。

Reg5[6:4]: yuv_fmt. 输入的数据的格式。000：default，rgb565；001：YUYV；010: UYVY；011: YYUV；100: UVYY；101: VUYY。

6.reg6, block_resolu,

Reg6[10:0]: Blk_line_pixel. Rblock 的行像素个数。

Reg6 [22:12]: Blk_clum_pixel. Rblock 的列像素个数。

7.reg7/8, r/w addr.

Reg7[31:0]: rotate_rd_base_addr. 源图像的存放地址。

Reg8[31:0]: rotate_wr_base_addr. 旋转后目标地址。

8.reg9, pic_resolu.

Reg9[10:0]: pic_clum_pixel. 整幅图像的行 pixel 个数。

Reg9[22:12]: pic_line_pixel. 整幅图像的列 pixel 个数。

9.rega, block_count.

Rega[15:0]: rotate_blk_count. Rblock 的总共个数。需要用户自己算出来给到硬件。

Rega[26:16]: wtmk_clum_pixel. 用户对水位中断的配置。到多少 Rblock 时，产生提醒。

10.regb, int_status.

Regb[0]: rotate_finish_int. 旋转完成中断。指示整幅图像完成旋转。

Regb[0]: rotate_wtmk_int. 水位中断，当旋转进程到某一 Rblock 时产生中断。

Regb[0]: rotate_cfg_err_int. 配置错误中断。在初始化时配置错误会直接产生，并关掉旋转使能信号。能够产生此中断的错误原因：在使能旋转开关后，1.Rblock 的行列像素为 0。 2.整幅图像的行列像素为 0。 3.水位为 0 但水位中断使能。 4. rotate_blk_count 为 0，Rblock 个数异常。 5.Rblock 的行列像素大于整幅图像的行列像素。 6.旋转逆时针和旋转 bypass 同时开启。

4.3. 软件编写

旋转模块软件编写简单，举例进行描述， 如将 480*320 大小的图像进行顺时针旋转。

1.首先计算 Rblock 大小和 Rblock 的个数。内部的存储 memory 为 32bit*2400，所以最大单个 Rblock 为 2400*2 = 4800 个 piexel。 可以使用 60x80 的 Rblock，这样共 32 个 Rblock ((480/60）*(320/80))。这样 320p 的图像将分为 32 个 60x80p 的图像分块被传输。

2.配置 Rotate module 的时钟，该时钟与 bus clk 同源，更改该时钟更改 bus_clk。

3.选择要使用的中断，并对中断进行使能。

4.选择模式，顺时针旋转（默认），逆时针旋转， 不旋转只转换。操作 rotate_anticlock 和 rotate_bps 寄存器。

5.翻转选择。

6.分辨率配置。分别配置整幅图像分辨率，Rblock 分辨率，以及 Rblock 的个数。 rotate_blk_count， pic_clum_pixel， pic_line_pixel， pic_clum_pixel， pic_line_pixel。

7.源原始图像和目标图像的地址设置。rotate_rd_base_addr 和 rotate_wr_base_addr。

8.如果要在旋转过程中进行操作，可以对 wtmk 进行设置。

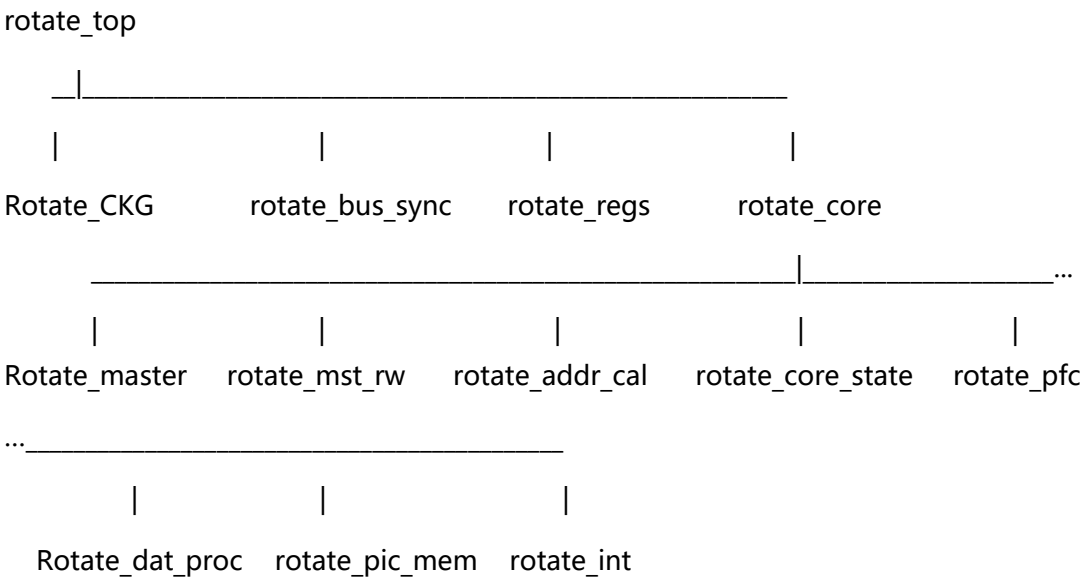
9.设置旋转使能。roate_ena。

10.旋转完成，进入中断。旋转使能会在完成后自动关闭（与中断无关系）。

11.需再次使用，按需配置。

5. 数字概述（for 数字组）

5.1. 代码结构



rotate_top：模块 top level。

Rotate_bus_sync：模块 apb bus rw 模块。

Rotate_regs：模块 cfg reg 的分配。

Rotate_core： 旋转 core，code 都在里面。只给出了 cfg reg 和 ahb master bus 的接口。

Rotate_master：读写数据的 ahb master 接口，burst 传输，有 1k 边界和 64 max words 的检测。

Rotate_mst_rw：ahb master 的控制信号，req，rw，baseaddr 等进行控制。

Rotate_addr_cal：ahb master 控制信号 base addr 的计算。

Rotate_core_state：主状态机对 master 读写时机控制以及地址计算进行控制。

Rotate_pfc：输入数据转换成 rgbdata，可以 bypass 掉，亦可以按 byte 翻转。

Rotate_dat_proc：数据处理模块，横数据转换成两列的竖数据，也要产生写入的地址。

Rotate_pic_mem：内部缓存 block pic 的 mem。

Rotate_int：模块中断的操作，根据输入产生 pulse，并能 clear 掉。

5.2. 状态机流程

状态机为 rotate_core_state。共 9 个状态。

1.state_idle：idle 状态。

2.state_rd_mult：每次 block 旋转前，计算每个 block 的读基址。

3.state_wr_mult：每次 block 旋转前，计算每个 block 的写入的基址。

4.state_rw_idle：rw 地址计算完后，进入 block 的 idle 状态。Single block 最后一行没读完前，会跳入 line_rd 状态。当最后一行读完后，如果 bypass 旋转，进入 line_dwr；正常旋转，进入 line_wr 状态。

5.state_line_rd： 读取原始图像状态，每次读取一条 line。等待 master 读完，跳入 rw_dile。

6.state_line_wr：将内部 memory 写完的图像在该状态按行写入外部储存器。每次进入该状态写入一行，等待 master 写入完成，跳入 line_jg。

7.state_line_dwr：将内部的 memory 写完的图像（bypass 旋转）在该状态按行写入外部储存器。每次进入该状态写入一行，等待 master 写入完成，跳入 line_jg。

8.state_line_jg：判断该 block 是否旋转完成，每次旋转数据按行写完成都会跳入该状态，当最后一行写入完成，跳入 pic_jg；否则跳入 line_wr 或 line_dwr 继续下一行写入。

9.state_pic_jg：判断整幅图像的 block 是否都旋转完成。完成跳入 idle。没完成跳入 cal_rd，开始计算下一个 block 的地址。

状态机运行状态图，如图 2 所示：

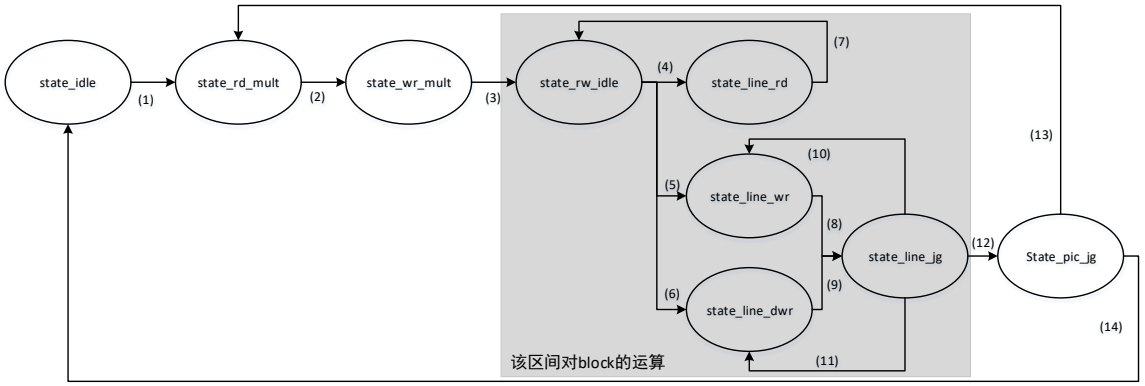


图 2.状态机状态图

状态机的单次图片旋转的运行顺序如下：

（1）初始状态 state_idle。Rotate_ena 打开后，进入 state_rd_mult。

（2）state_rd_mult 状态。持续一个 cycle，计算完成 block rd 地址，进入 state_wr_mult。

（3）state_wr_mult 状态。持续一个 cycle，计算完成 block wr 地址，进入 state_rw_idle。

（4）state_rw_idle 状态。如果 block 的所有 line 没有读取完，则会一直进入 state_line_rd。

（5）state_rw_idle 状态。如果 block 的所有 line 读取完成，若没有 bypass rotate，进入 state_line_wr。

（6）state_rw_idle 状态。如果 block 的所有 line 读取完成，若使能 bypass rotate，进入 state_line_dwr。

（7）state_line_rd 状态。如果 bus_master 读完一条 line，进入 state_rw_idle。

（8）state_line_wr 状态。如果 bus_master 写完一条 line。进入 state_line_jg。

（9）state_line_dwr 状态。如果 bus_master 写完一条 line。进入 state_line_jg。

（10）state_line_jg 状态。如果 bus master 没有写完最后一条 line，且未 bypass rotate，跳回 state_line_wr。

（11）state_line_jg 状态。如果 bus master 没有写完最后一条 line，且 bypass rotate，跳回 state_line_dwr。

（12）state_line_jg 状态。如果 bus master 写完最后一条 line，则进入 state_pic_jg。

（13）state_pic_jg 状态。如果当前 block 不是整幅图片最后一个 block，则调回 state_rd_mult，计算下一 block 的地址。

（14）state_pic_jg 状态。如果当前 block 是整幅图片最后一个 block，旋转完成，进入 state idle。

5.3. 代码补充