1/4” HD CMOS Image Sensor

GC1004

Application Notes
V1.1

2014-04-15

GalaxyCore Inc.

目　　录

1. 简介

此文档为应用系统设计者了解 GC1004 提供关键说明，针对 GC1004 芯片在寄存器设定及系统应用上给予指导，方便系统设计及调试工程师快速建立应用方案和调试效果。如果需要更详细的寄存器定义，请参考 GC1004 DataSheet 文档。

GC1004 是格科微电子有限公司研发的最新 HD CMOS 图像传感器芯片。它采用了公司最新的 pixel 工艺和图像处理技术，可在监控、行车记录仪、视频等领域为客户提供高性价比的解决方案。

GC1004 的 I2C 总线读写地址为 0x78/0x79，芯片判断通过 0xf0/0xf1 只读寄存器来实现，如果读出值为 0x10/0x04，则为 GC1004。

GC1004 的功能框图：

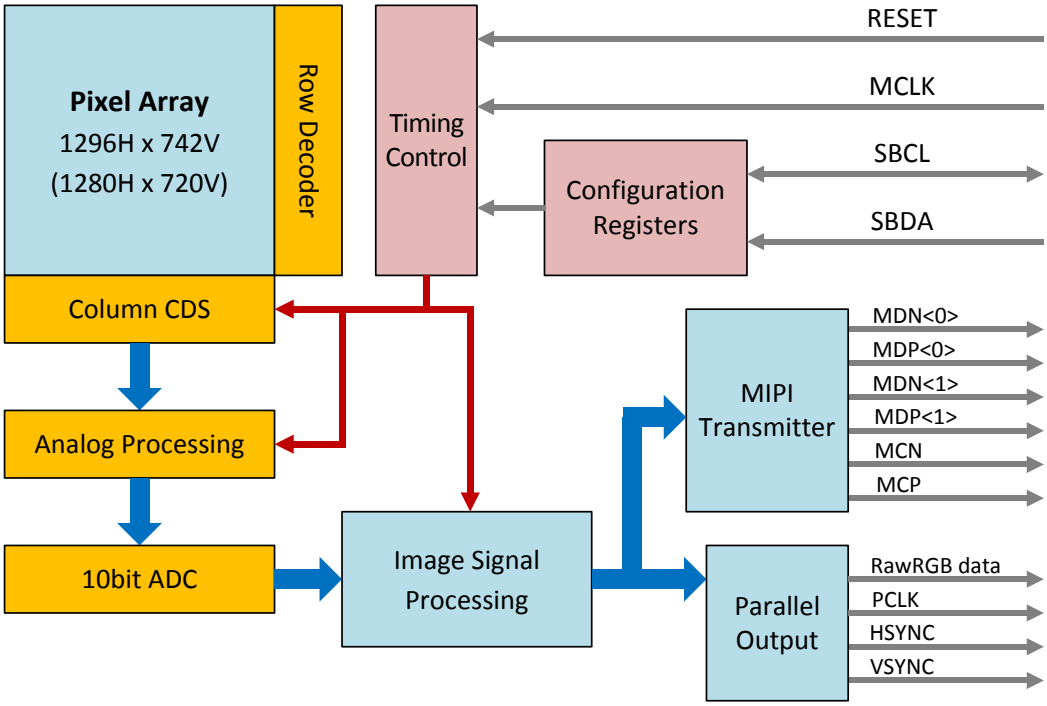


图 1-1 功能框图

2. Pixel Array 说明

GC1004 的像素阵列大小为 1296 列、742 行，除此之外还有 16 行 dark row 可用。

GC1004 的像素阵列上覆盖着彩色滤光片(Color Filter)，并且彩色滤光片以 BG/GR 的方式每行交错排列。

像素的输出按照逐行读出的方式进行，读出的顺序见下图：

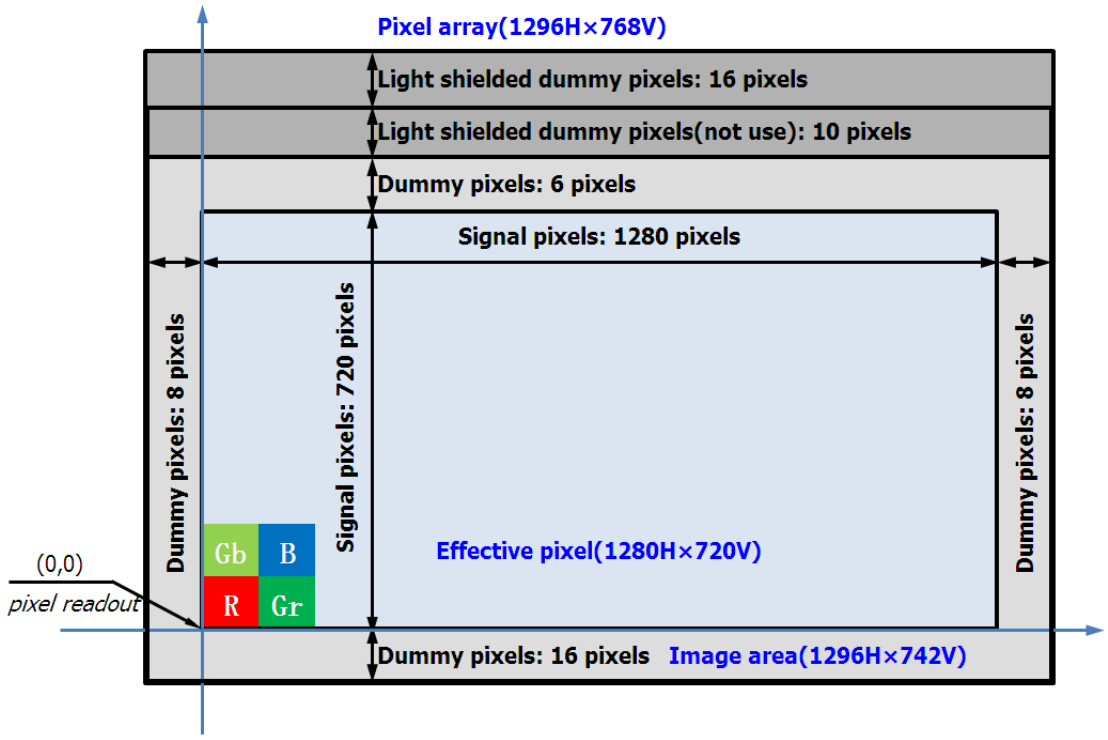


图 2-1 像素阵列图

3. 系统应用

3.1 外围连接

3.1.1 DVP 接口

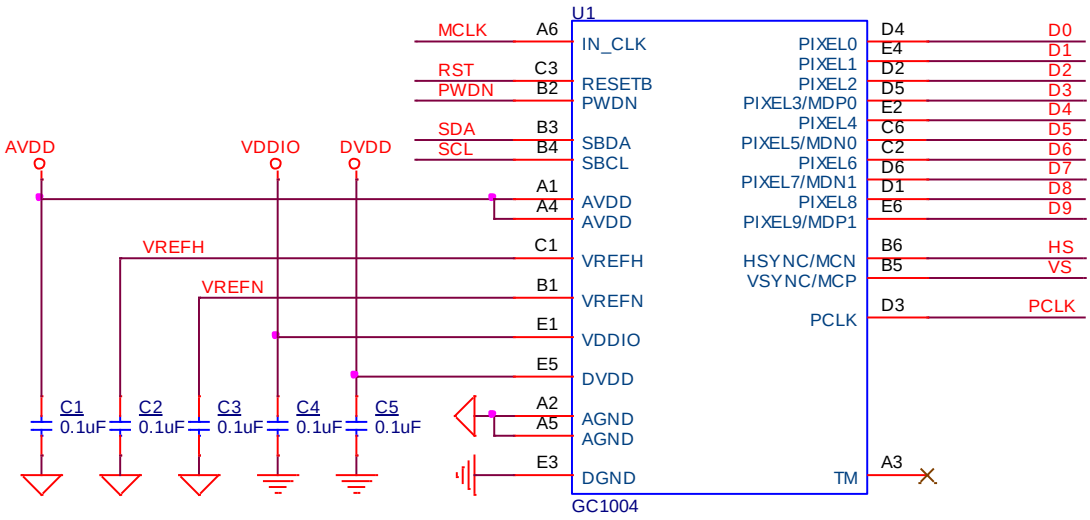


图 3-1 DVP 接口外围电路图

3.1.2 MIPI 接口(1_lane)

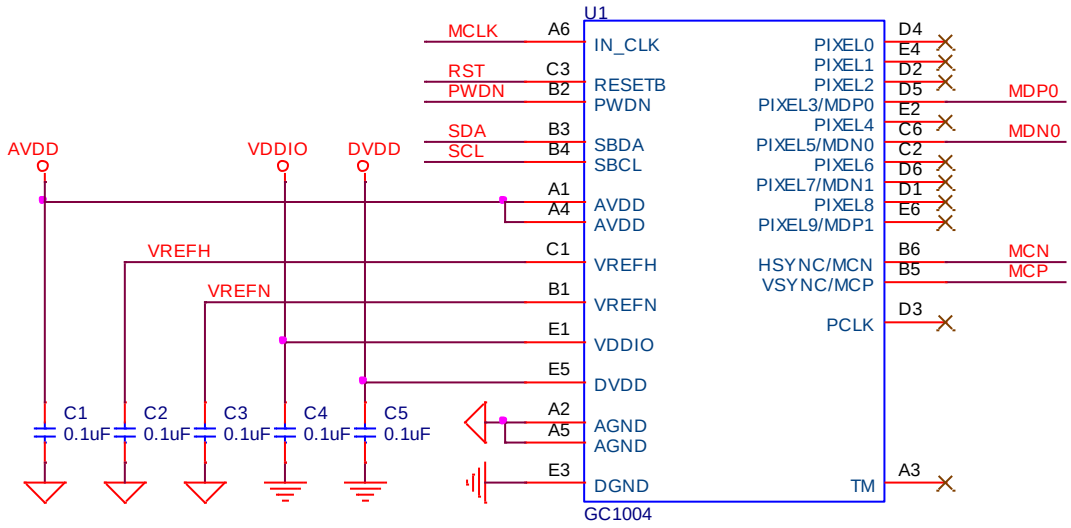


图 3-2 MIPI 接口(1_lane)外围电路图

3.1.3 MIPI 接口(2_lane)

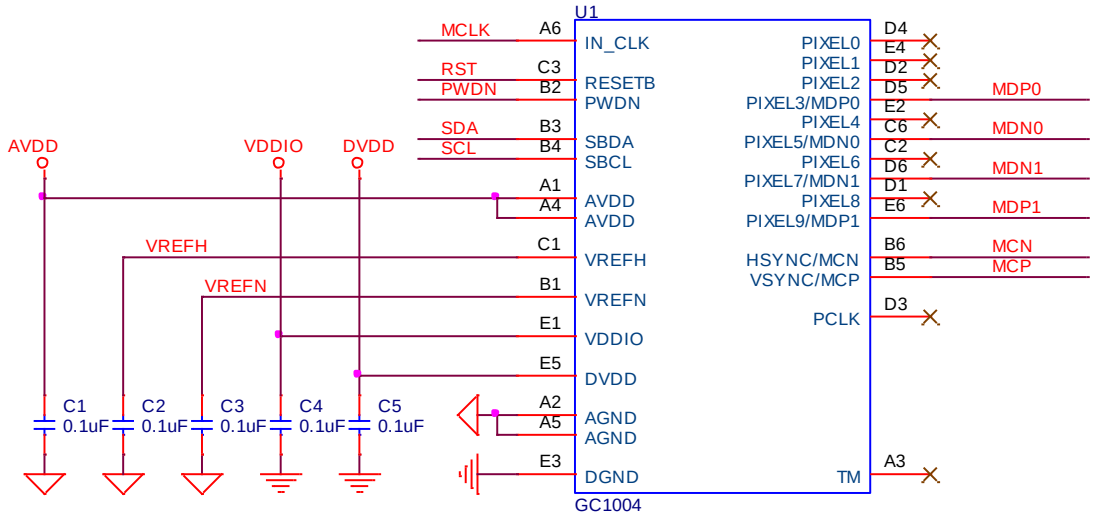
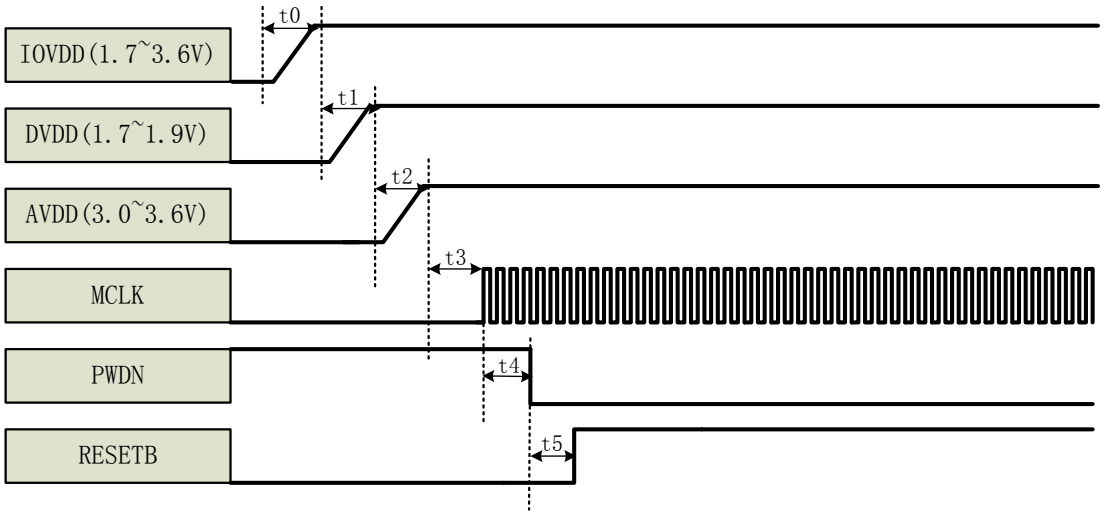


图 3-3 MIPI 接口(2_lane)外围电路图

3.2 应用时序

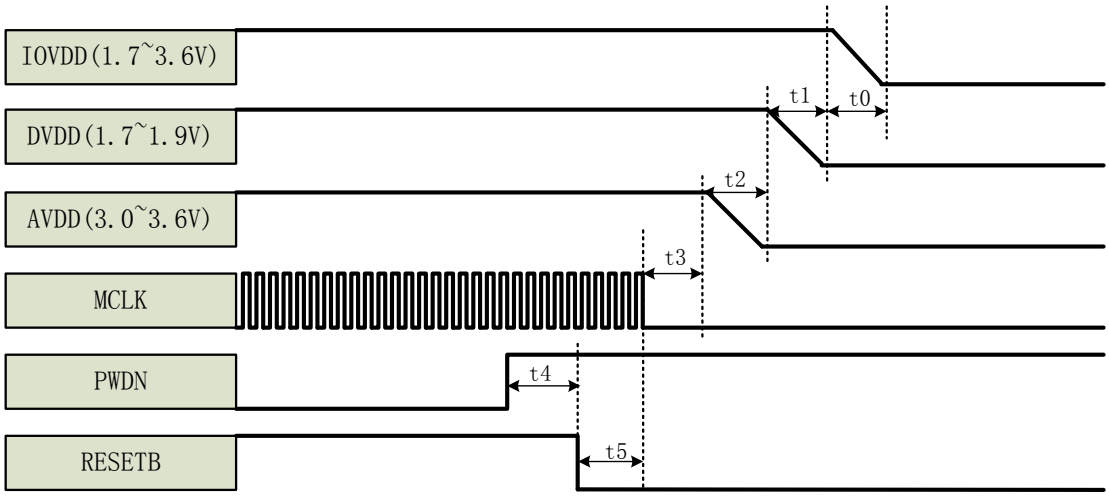
1) PWDN 与 RESET 均是异步设计，生效时不需要 MCLK pin 有时钟提供。
2) PWDN 高有效，Low –> 正常工作，High –> 省电模式；
   RESET 低有效，Low –> reset 芯片，High –> 正常工作。
3) 主时钟必须在 sensor two-wire serial interface 读写前提供。
4) 建议的应用时序如下图所示：



Parameter	Description	Min.	Max.	Unit
t0	IOVDD rising time	100		us
t1	From IOVDD to DVDD	50		us

t2	From DVDD to AVDD	50		us
t3	From AVDD to MCLK applied	>0		us
t4	From MCLK applied to Sensor enable	>0		us
t5	From PWDN pull low to RESET pull high	>0		us

图 3-4 上电时序说明



Parameter	Description	Min.	Max.	Unit
t0	From DVDD to IOVDD falling time	>0		us
t1	From AVDD to DVDD falling time	>0		us
t2	AVDD falling time	>0		us
t3	From MCLK disable to sensor AVDD power down	>0		us
t4	From sensor disable sensor RESET pull low	>0		us
t5	From sensor RESET pull low to MCLK disable	>0		us

图 3-5 下电时序说明

- 以上为推荐的上电/下电时序。
- 如果在应用上有特殊需求的，请与我们联系以作确认。

3.3 芯片控制

3.3.1 芯片复位

芯片复位请将 RESET pin 接入低电平。

3.3.2 Standby 模式控制

使芯片置于 Standby 模式有两种方式：

1) PWDN pin 接入高电平。此方式会降低功耗，输出 pin 高阻，寄存器值保持不变。此时寄存器将无法读写。

   要恢复 normal 工作模式，只需将 PWDN pin 接入低电平即可。

2) 将寄存器 0xfc[0]置 1，0xf2 写为 0x00。此方式同样降低功耗，输出 pin 高阻，寄存器值保持不变。此时寄存器是可以读写的。

   要恢复 normal 工作模式，需要将 0xfc[0]置 0，0xf2 写为 0x0f。

3.3.3 输出使能控制

GC1004 可以通过写寄存器来控制几组输出 pin 的输出使能。

Function	Register	Description
VSync output enable	0xf2[1]	控制 VSYNC pin 输出 0 –> VSYNC pin 高阻 1 –> VSYNC pin 正常输出
HSync output enable	0xf2[0]	控制 HSYNC pin 输出 0 –> HSYNC pin 高阻 1 –> HSYNC pin 正常输出
PCLK output enable	0xf2[2]	控制 PCLK pin 输出 0 –> PCLK pin 高阻 1 –> PCLK pin 正常输出
Pixel data[7:0] output enable	0xf2[3]	控制 data pin 输出 0 –> data pin 高阻 1 –> data pin 正常输出

表 3-1 输出使能控制

3.3.4 输出 Pin 驱动能力

GC1004 可以通过写寄存器来控制输出 pin 的驱动能力。

Function	Register	Description
PCLK PIN 驱动能力	P0:0x24 [1:0]	控制 PCLK pin 输出驱动能力。 00 –> 8mA 01 –> 12mA 10 –> 16mA

		11 –> 20mA
Data_high PIN 驱动能力	P0:0x24 [3:2]	控制 data_high pin 输出驱动能力。 00 –> 8mA 01 –> 12mA 10 –> 16mA 11 –> 20mA
SYNC PIN 驱动能力	P0:0x24 [5:4]	控制 HSYNC/VSYNC pin 输出驱动能力。 00 –> 4mA 01 –> 8mA 10 –> 12mA 11 –> 16mA
Data_low PIN 驱动能力	P0:0x24 [7:6]	控制 data_low pin 输出驱动能力。 00 –> 8mA 01 –> 12mA 10 –> 16mA 11 –> 20mA

表 3-2 输出驱动能力控制

4. 芯片功能方面配置

4.1 Pixel Array 控制

GC1004 采用逐行扫描的方式将阵列产生的信号依次输入到模拟信号处理模块中。最开始的行为 0 行。在默认寄存器设置下，Sensor 的阵列数据输出顺序为从下到上，从左到右。

GC1004 可通过寄存器控制扫描顺序，实现镜像/垂直翻转。

原始图像	镜像翻转	垂直翻转	镜像&垂直翻转
			

图 4-1 图像翻转控制

Function	Register Address	Register Value
正常图像	P0:0x17[1:0]	00
镜像翻转	P0:0x17[1:0]	01
垂直翻转	P0:0x17[1:0]	10
镜像&垂直翻转	P0:0x17[1:0]	11

表 4-1 镜像/垂直翻转控制

4.2 时钟预分频

外部 MCLK 时钟输入后，通过 clock divider 模块对 MCLK 进行分频，芯片内部工作频率基于分频后的频率。GC1004 最大分频比为 1/8 分频。

Function	Register	Description
MCLK 内部分频比	0xfa[7:4]	此值+1 为实际的分频率，如 7 表示 8 分频。
分频后占空比	0xfa[3:0]	分频后高电平的个数。如果是 8 分频，0xfa 设为 0x77，表示 8 分频后的波形占空比为 H:L = 7:1

表 4-2 内部分频设置 1

一般分频的推荐设置如下表：

Function	Register	Description
内部分频设置	0xfa	0x00 –> 1/1 MCLK 0x11 –> 1/2 MCLK 0x21 –> 1/3 MCLK 0x32 –> 1/4 MCLK 0x42 –> 1/5 MCLK 0x53 –> 1/6 MCLK 0x63 –> 1/7 MCLK 0x74 –> 1/8 MCLK

表 4-3 内部分频设置 2

4.3 输出时序说明及同步信号控制

4.3.1 Parallel 输出时序说明

假设帧同步信号 Vsync 低有效，行同步 Hsync 为高有效，而输出格式 Raw RGB 图像的话，Vsync 和 Hsync 的关系如下：

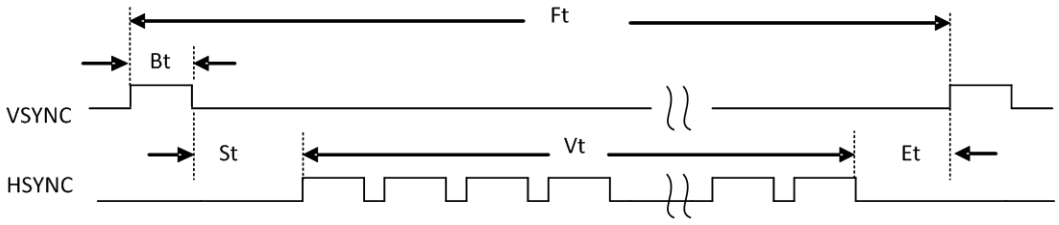


图 4-2 DVP 输出时序图

Ft = VB + Vt +16 (dark line) (单位均为 row_time，row_time 在“5.2”中描述)

VB+16= Bt + St + Et，VB 一般称为 Vblank/Dummy line，由寄存器 P0:0x07 和 P0:0x08 设置。

- Ft –> Frame time，一个帧周期的时间。
- Bt –> Blank time，Vsync 无效时间。
- St –> Start time，帧头与第一行有效数据开始之间的时间，由 P0:0x13 寄存器来设定。
- Et –> End time，最后一行有效数据与帧尾之间的时间，由 P0:0x14 寄存器来设定。

◆ Vt –> 有效行的时间。Vt = win_height，由寄存器 P0:0x0d 和 P0:0x0e 所设定。

当 exp_time(曝光时间)<= win_height+VB+16 时，Bt=VB+16-St-Et。帧率由 window_height+VB+16 控制。

当 exp_time > win_height+VB+16 时，Bt=exp_time-win_height-St-Et。帧率由 exp_time 决定。

4.3.2 同步信号极性控制

VSYNC 为场同步信号，HSYNC 为行同步信号，PCLK 为输出 data 的同步时钟。GC1004 可以通过寄存器来控制这三个信号的极性。默认配置下，PCLK 下降沿出数据，建议后端 DSP 用 PCLK 的上升沿采集数据。

Function	Register	Description
VSYNC 极性控制	P0:0x8c[0]	0 –> 低有效。 1 –> 高有效。 表示 VSYNC 为高时 sensor 输出有效数据。
HSYNC 极性控制	P0:0x8c[1]	0 –> 低有效。 1 –> 高有效。 表示 HSYNC 为高时 sensor 输出有效数据。
PCLK 极性控制	P0:0x8c[2]	0 –> 下降沿出 data，default 值。 1 –> 上升沿出 data。

表 4-4 同步信号极性控制

4.3.3 MIPI 输出时序说明

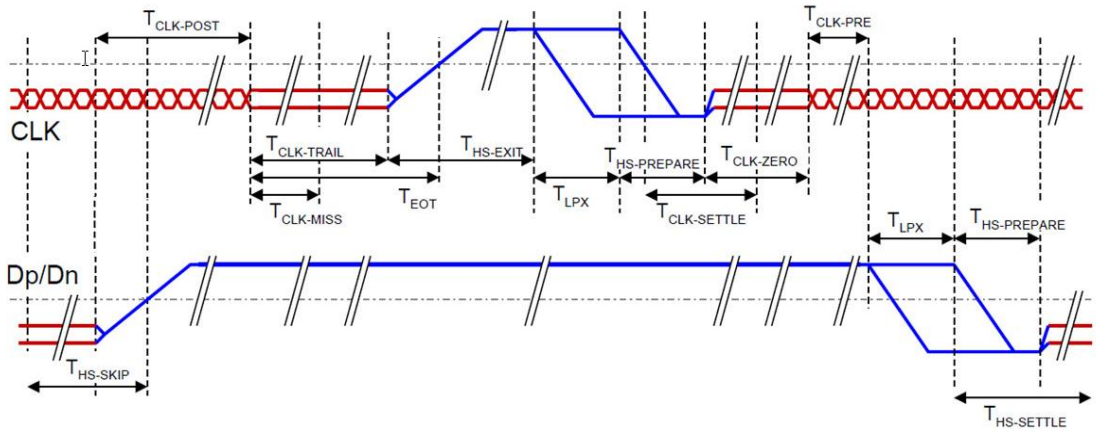


图 4-3 Clock lane low-power

- Clock 在数据高速传输时和 data lane 模式切换时必须是有效的高速时钟。
- 只有当 data lane 处于 low-power 状态下 clock lane 才能进入 low-power 状态。
- 理论上，低功耗状态的 data lane 和高速的 clock lane 无相关性。

T_{CLK_PRE} 由寄存器 P3: 0x24 所设定。

$T_{CLK\_HS\_PRE}$ 由寄存器 P3: 0x22 所设定。

T_{CLK_POST} 由寄存器 P3: 0x25 所设定。

T_{CLK_ZERO} 由寄存器 P3: 0x23 所设定。

T_{CLK_TRAIL} 由寄存器 P3: 0x26 所设定。

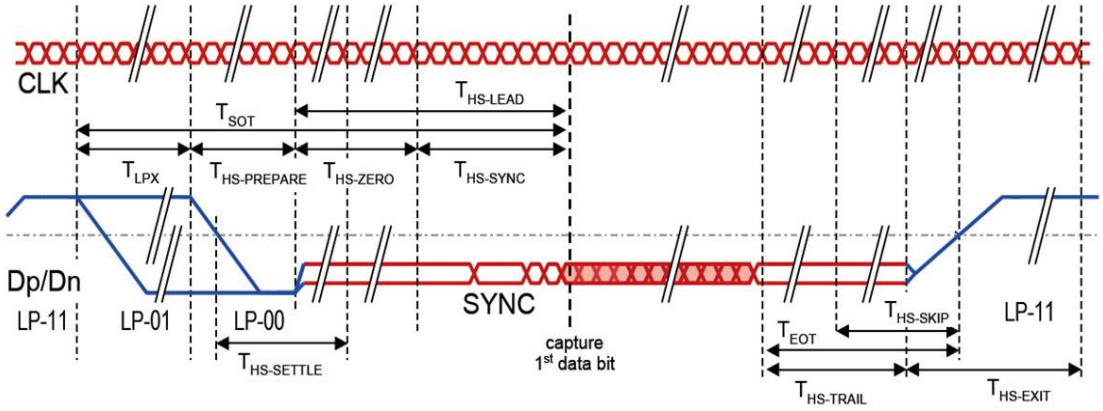


图 4-4 data brust

- Data lane 在传送时，clock 必须已经在高速运行状态并用以对 data lane 进行采样（只在 data lane 高速时采样）。
- 要求有清晰定义的头和尾序列，以甄别出真实传送的 bit 信息。
- 接收时尾序列可以在物理层直接丢弃。
- 在 mipi 线上状态变化时，如不满足基本时序参数要求可予忽略。

T_{LPX} 由寄存器 P3:0x21 所设定。

$T_{HS\_PREPARE}$ 由寄存器 P3: 0x29 所设定。

T_{HS_ZERO} 由寄存器 P3: 0x2a 所设定。

T_{HS_TRAIL} 由寄存器 P3: 0x2b 所设定。

T_{HS_EXIT} 由寄存器 P3: 0x27 所设定。

4.4 图像窗口设置

GC1004 可以截取任意尺寸的窗口输出，且有两种模式实现窗口输出，这两种模式输出小窗口时，视角均会变小。

1) Windowing 模式

- 此模式输出小尺寸时，会加快帧率。
- 使用此模式时，anti-flicker 的设置需要重新计算。
- 如果想实现高速小尺寸输出，建议用此模式。

Windowing 挖窗口时，用 column start 和 row start 来分别确定要挖窗口起始的 X/Y 坐标，用 window width-16 和 window height-22(请注意寄存器设置要比实际输出多)来确定所需要窗口的宽度和高度。

2) Crop window 模式

- 此模式输出小尺寸时，帧率不变。
- anti-flicker 的配置不需要重新计算。
- Windowing(P0:0x0d~P0:0x10)寄存器按 1280x720 的配置。
- 要使用 crop window 模式，需要将 P0:0x90[0]置 1。

用 Crop window 模式挖窗口时，用 Out window x0 和 Out window y0 来分别确定要挖窗口起始的 X/Y 坐标，用 Out window width 和 Out window height 来确定所需要窗口的宽度和高度。

5. 芯片应用方面配置

5.1 R/Gr/Gb/B 的阵列顺序

原始图像输出时 R/Gr/Gb/B 排列如“图 5-1 像素阵列图”所示，当图像进行了 mirror/flip 翻转后，此 BG/GR 阵列会随之改变，具体如下图。

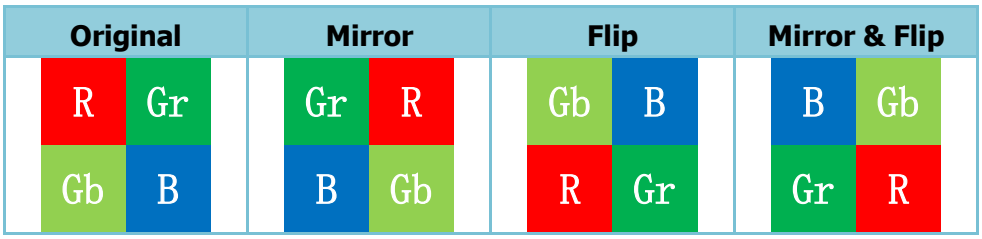


图 5-1 R/Gr/Gb/B 阵列

5.2 Row_time 的计算

一行时间**(row_time)**的计算方法：

row_time = (Hb + Sh_delay + win_width/2 + 4)/HPCLK。

- Hb –> 为 HBlank 或 dummy pixel，由 P0:0x05 和 P0:0x06 设定。
- Sh_delay –> 由寄存器 P0:0x11, P0:0x12 设定。
- win_width –> P0:0x0f 和 P0:0x10 所设定，比实际需要的输出尺寸要大 16，所以输出 HD 时，需要设置窗口宽度为 1296。
- HPCLK –> half PCLK。

例如，当 Hb=177H，Sh_delay=0CH，win_width=510H，PCLK=48M；那么 row_time=43us。

5.3 Gain 的使用

GC1004 的 Gain 分为 Analog Gain 和 Digital Gain 两部分。当需要增加 Gain 时，优先使用 Analog Gain。通过寄存器 P0:0xb6 控制 Analog Gain 的增长；通过寄存器 P0:0xb0、0xb1、0xb2 控制 Digital Gain（Global Gain 和 Pre-gain）的增长。总增益为 Analog Gain×Global Gain×Pre-gain。

- P0: 0xb6 –> Analog gain，0x00 为 1 倍，最大为 0x08。各档对应的 Analog Gain 如下表所示：

P0: 0xb6	Analog Gain
0x00	1.0x
0x01	1.4x
0x02	1.8x
0x03	2.6x
0x04	3.4x
0x05	4.7x
0x06	6.84x
0x07	9.4x
0x08	13.2x

- P0: 0xb0 –> Global gain，0x40 代表 1 倍，最大为 0xff 即 4 倍。
- P0: 0xb1[3:0]–> Pre-gain[9:6]，为 Pre-gain 的整数部分，0x01 代表 1 倍，最大为 0x0f 即 16 倍。
- P0: 0xb2[7:2] –> Pre-gain[5:0]，为 Pre-gain 的小数部分，每增加 1 代表 Gain 增加 1/64。

```
//Set Gain
#define ANALOG_GAIN_1 64   // 1.00x
#define ANALOG_GAIN_2 90   // 1.4x
#define ANALOG_GAIN_3 118   // 1.8x
#define ANALOG_GAIN_4 163   // 2.56x
#define ANALOG_GAIN_5 218   // 3.40x
#define ANALOG_GAIN_6 304   // 4.7x
#define ANALOG_GAIN_7 438   // 6.84x
#define ANALOG_GAIN_8 602   // 9.4x
#define ANALOG_GAIN_9 851   // 13.2x


kal_uint16 GC1004_SetGain(kal_uint16 iGain)
{
    kal_uint16 iReg,temp;
    if(GC1004_Lock)
        return;

#ifdef GC1004_DRIVER_TRACE
    SENSORDB("GC1004_SetGain iGain = %d \n",iGain);
```

```
#endif
    GC1004_write_cmos_sensor(0xb1, 0x01);
    GC1004_write_cmos_sensor(0xb2, 0x00);

    iReg = iGain;
    if(iReg < 0x40)
        iReg = 0x40;
    else if((ANALOG_GAIN_1<= iReg)&&(iReg < ANALOG_GAIN_2))
    {
        //analog gain
        GC1004_write_cmos_sensor(0xb6,   0x00);//
        temp = iReg;
        GC1004_write_cmos_sensor(0xb1, temp>>6);
        GC1004_write_cmos_sensor(0xb2, (temp<<2)&0xfc);
        SENSORDB("GC1004 analogic gain 1x , GC1004 add pregain = %d\n",temp);
    }
    else if((ANALOG_GAIN_2<= iReg)&&(iReg < ANALOG_GAIN_3))
    {
        GC1004_write_cmos_sensor(0xb6,   0x01);//
        temp = 64*iReg/ANALOG_GAIN_2;
        GC1004_write_cmos_sensor(0xb1, temp>>6);
        GC1004_write_cmos_sensor(0xb2, (temp<<2)&0xfc);
        SENSORDB("GC1004 analogic gain 1.4x , GC1004 add pregain = %d\n",temp);
    }
    else if((ANALOG_GAIN_3<= iReg)&&(iReg < ANALOG_GAIN_4))
    {
        GC1004_write_cmos_sensor(0xb6,   0x02);//
        temp = 64*iReg/ANALOG_GAIN_3;
        GC1004_write_cmos_sensor(0xb1, temp>>6);
        GC1004_write_cmos_sensor(0xb2, (temp<<2)&0xfc);
        SENSORDB("GC1004 analogic gain 1.8x , GC1004 add pregain = %d\n",temp);
    }
    else if((ANALOG_GAIN_4<= iReg)&&(iReg < ANALOG_GAIN_5))
    {
        GC1004_write_cmos_sensor(0xb6,   0x03);
        temp = 64*iReg/ANALOG_GAIN_4;
```

```
        GC1004_write_cmos_sensor(0xb1, temp>>6);
        GC1004_write_cmos_sensor(0xb2, (temp<<2)&0xfc);
        SENSORDB("GC1004 analogic gain 2.56x , GC1004 add pregain = %d\n",temp);
    }
    else if((ANALOG_GAIN_5<= iReg)&&(iReg < ANALOG_GAIN_6))
    {
        GC1004_write_cmos_sensor(0xb6,   0x04);
        temp = 64*iReg/ANALOG_GAIN_5;
        GC1004_write_cmos_sensor(0xb1, temp>>6);
        GC1004_write_cmos_sensor(0xb2, (temp<<2)&0xfc);
        SENSORDB("GC1004 analogic gain 3.4x , GC1004 add pregain = %d\n",temp);
    }
    else if((ANALOG_GAIN_6<= iReg)&&(iReg < ANALOG_GAIN_7))
    {
        GC1004_write_cmos_sensor(0xb6,   0x05);//
        temp = 64*iReg/ANALOG_GAIN_6;
        GC1004_write_cmos_sensor(0xb1, temp>>6);
        GC1004_write_cmos_sensor(0xb2, (temp<<2)&0xfc);
        SENSORDB("GC1004 analogic gain 4.7x , GC1004 add pregain = %d\n",temp);
    }
    else if((ANALOG_GAIN_7<= iReg)&&(iReg < ANALOG_GAIN_8))
    {
        GC1004_write_cmos_sensor(0xb6,   0x06);//
        temp = 64*iReg/ANALOG_GAIN_7;
        GC1004_write_cmos_sensor(0xb1, temp>>6);
        GC1004_write_cmos_sensor(0xb2, (temp<<2)&0xfc);
        SENSORDB("GC1004 analogic gain 6.84x, GC1004 add pregain = %d\n",temp);
    }
    else if((ANALOG_GAIN_8<= iReg)&&(iReg < ANALOG_GAIN_9))
    {
        GC1004_write_cmos_sensor(0xb6,   0x07);
        temp = 64*iReg/ANALOG_GAIN_8;
        GC1004_write_cmos_sensor(0xb1, temp>>6);
        GC1004_write_cmos_sensor(0xb2, (temp<<2)&0xfc);
        SENSORDB("GC1004 analogic gain 9.4x,GC1004 add pregain = %d\n",temp);
    }
```

```
        else if((ANALOG_GAIN_9<= iReg)&&(iReg < ANALOG_GAIN_10))
        {
            GC1004_write_cmos_sensor(0xb6, 0x08);
            temp = 64*iReg/ANALOG_GAIN_9;
            GC1004_write_cmos_sensor(0xb1, temp>>6);
            GC1004_write_cmos_sensor(0xb2, (temp<<2)&0xfc);
            SENSORDB("GC1004 analogic gain 13.2x ,GC1004 add pregain = %d\n",temp);
        }
    }
```

5.4 Shutter 的使用

通过寄存器 P0:0x03、0x04 设置 shutter。

- P0: 0x03 –> exp_time[12:8]。
- P0: 0x04 –> exp_time[7:0]。

```
//Set shutter
void GC1004_Write_Shutter(kal_uint16 iShutter)
{
    if(iShutter < 1) iShutter = 1;
    if(iShutter > 8191) iShutter = 8191;//2^13
    //Update Shutter
    GC1004_write_cmos_sensor(0x04, (iShutter) & 0xFF);
    GC1004_write_cmos_sensor(0x03, (iShutter >> 8) & 0x1F);
}
```

5.5 帧率的配置方法

5.5.1 固定帧率的实现

根据 4.3.1 中所述：

当 exp_time(曝 光 时 间)<=win_height+VB+16 时 ， 帧 率 由 window_height+VB+16 控制，

即帧率=1/(window_height +VB+16(dark 行))*row_time;

当 exp_time>win_height+VB+16 时，帧率由 exp_time 决定，即帧率 =1/(exp_time*row_time).

根据以上的条件，要固定帧率，则必须满足 exp_time<=win_height+VB+16。设置 win_height 和 VB 的值，从而决定了 exp_time 的最大值。只要在平台上控制 exp_time 始终小于这个最大值，那么帧率就能被固定。

例如：当相关寄存器设置如下，win_height=2e6H，VB=10H，HB=177H，Sh_delay=0CH，win_width=510H，HPCLK=24M。

那么，exp_time 的最大值为 2b5H。

帧率=1/[(win_height+VB+16)*row_time]=1/[(742+16+16)*43]=30fps，即会固定在 30fps。

5.5.2 亮暗帧率切换的实现

以亮处固定 50fps、暗处固定 25fps 举例说明。

根据施密特触发器原理，设置两个阈值 LV_h 和 LV_l，代码里通过 Luma Value（LV）与 LV_h 和 LV_l 的关系，来判断当前的帧率。

1）LV>=LV_h，判断当前为亮处，芯片工作在 50fps；

2）LV<=LV_l，判断当前为暗处，芯片工作在 25fps；

3）LV_l<LV<LV_h，芯片维持在上一帧的帧率。