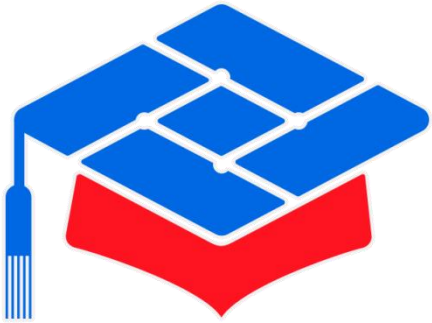

第八届

全国大学生集成电路创新创业大赛

报告类型*：系统设计报告

参赛杯赛*：安谋科技杯

作品名称*：基于 ARM Cortex-M0 的视频处理系统

队伍编号*：CICC4023

团队名称*：爱抚屁寄诶

1. 系统设计总体框图

本系统的主要功能部件包括 ARM Cortex-M0 内核、AHB 总线、CMSIS-DAP 调试接口、SD 卡控制模块、ISP 视频处理模块、HDMI 显示模块等，具体系统框架如图 1 所示。该系统利用 Cortex_M0 检测按键状态，并根据按键状态改变特殊功能寄存器（SFR）的值，进而实现控制 sd_ctrl 模块对 sd 卡的读写操作。sd_ctrl 将 SD 卡的数据发送至 fifo，并且根据 ISP 的请求将数据发送给 ISP 进行数据处理，ISP 将处理后的数据发送至 sdram，hdmi 模块将 sdram 的数据显示至 pc 端。图 2 是时钟域和带宽示意图。

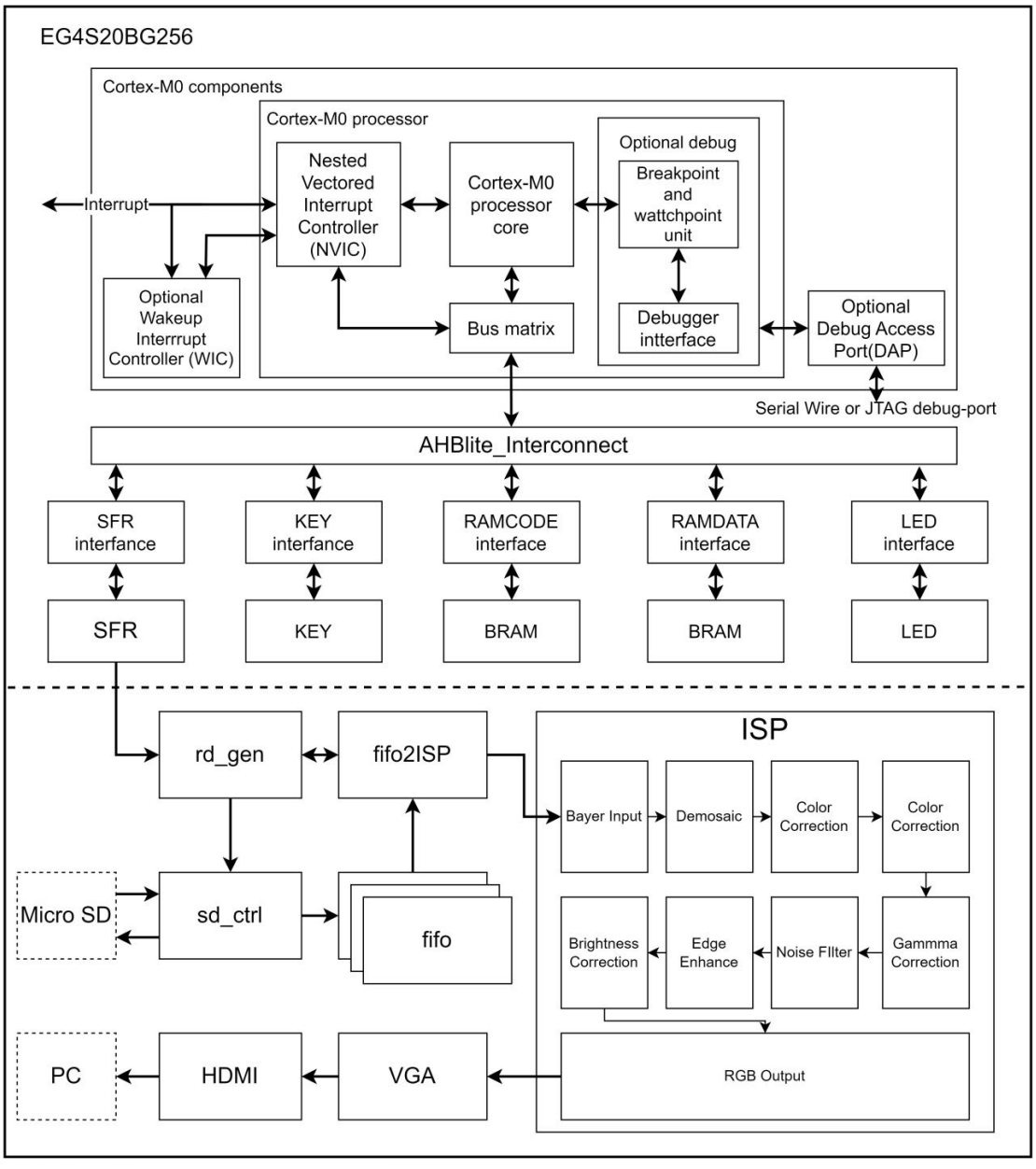

图 1 系统总体框架图

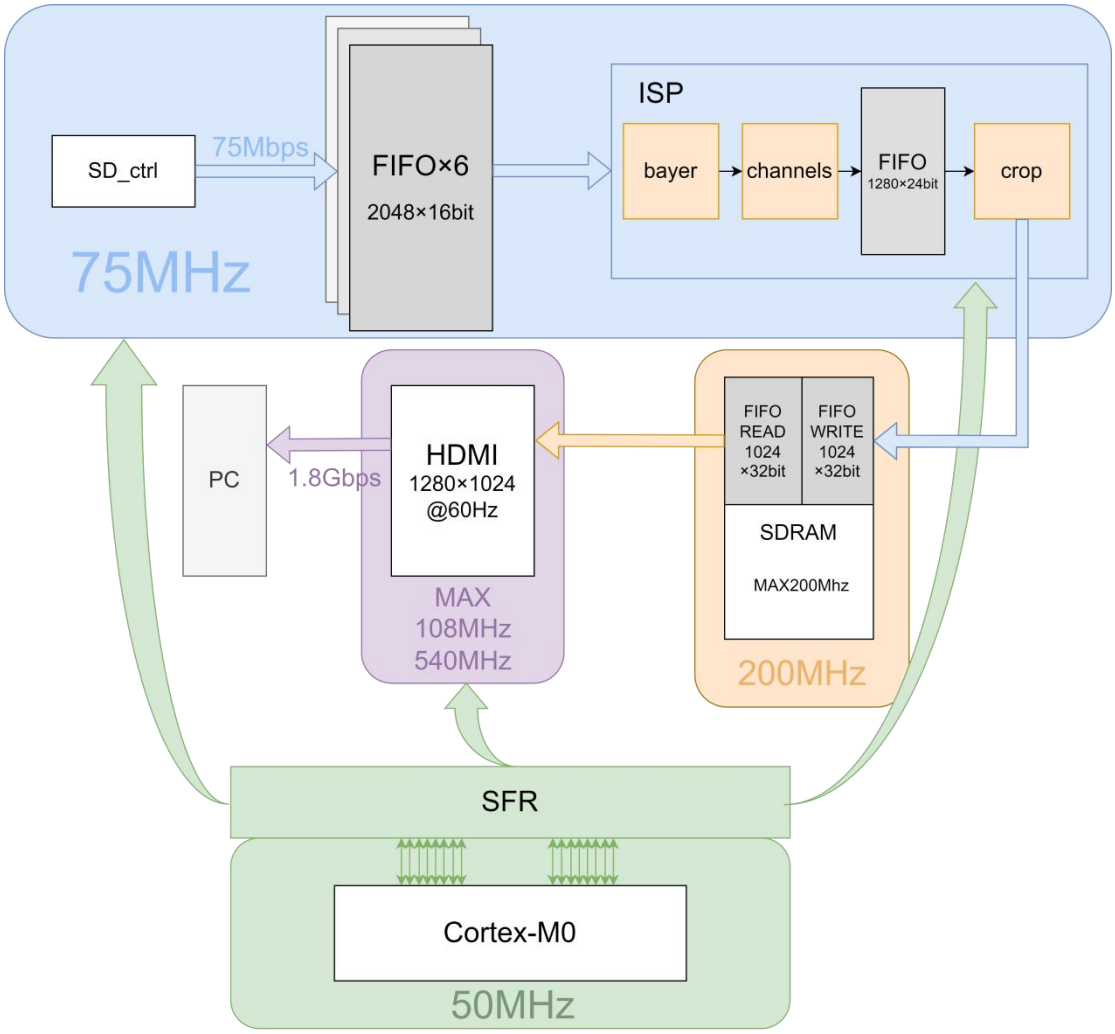

图 2 系统时钟域和带宽情况

2. Cortex-M0 处理器的集成情况

2.1. Cortex_M0 处理器

Cortex-M0 处理器基于冯·诺伊曼架构（单总线接口），使用 32 位的精简指令集（RISC）。总共支持 56 个基本指令。由于读写存储器的指令相互独立，而且算术或逻辑操作的指令使用寄存器，Cortex-M0 处理器可以被归到加载-存储（load-store）的结构中。

处理器核心包括寄存器组、算术逻辑单元（ALU）、数据总线和控制逻辑。其中，嵌套向量中断控制器（NVIC）可以处理最多 32 个中断请求和一个不可屏蔽中断（NMI）输入。

调试子系统包括多个功能模块，以处理调试控制、程序断点和数据监视点（data watchpoint）。当调试事件发生时，处理器内核会被置于暂停状态，这时开发人员可以检查当前的处理器状态。JTAG（联合测试行动组）和 SWD（串行线调试）提供了通向总线系统和调试功能的入口。

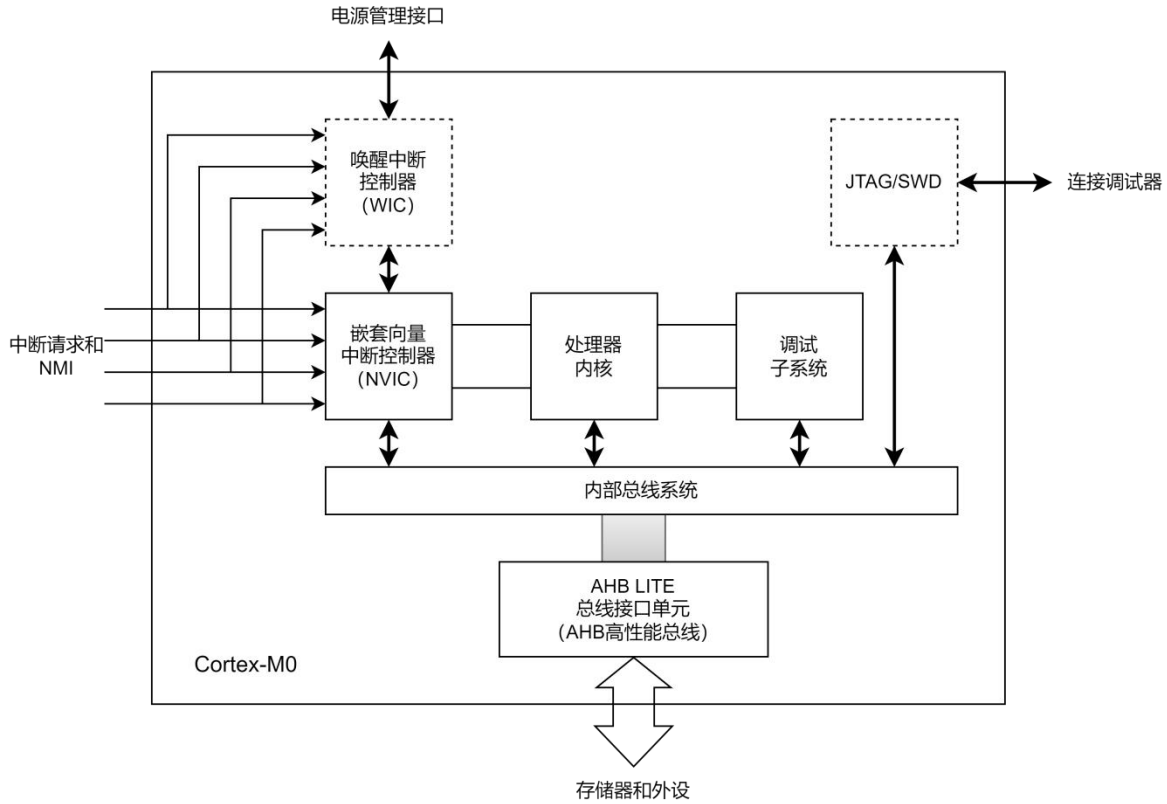

图 3 Cortex-M0 处理器基本结构

Cortex-M0 处理器由于门数量低，高效率的架构，低功耗特性和逻辑单元提升使得该处理器拥有低功耗的特性。并且由于 Cortex-M 处理器的架构和指令集近似，可以较为容易的实现软件的移植和平台的切换。

2.2. AHB 总线

AHB:AdvancedHigh-performanceBus(先进高性能总线)可以微控制器(cpu)、高带宽的片上 RAM.、高带宽的外部存储器接口、DMA 总线 master、各种拥有 AHB 接口的控制器等等连接起来构成个独立的完整的 SOC 系统，不仅如此，还可以通过 AHB-APB 桥来连接 APB 总线系统。

AHB 总线主要由四部分组成：主设备 Master，从设备 Slave，仲裁器 Arbiter

和译码器 Decoder。

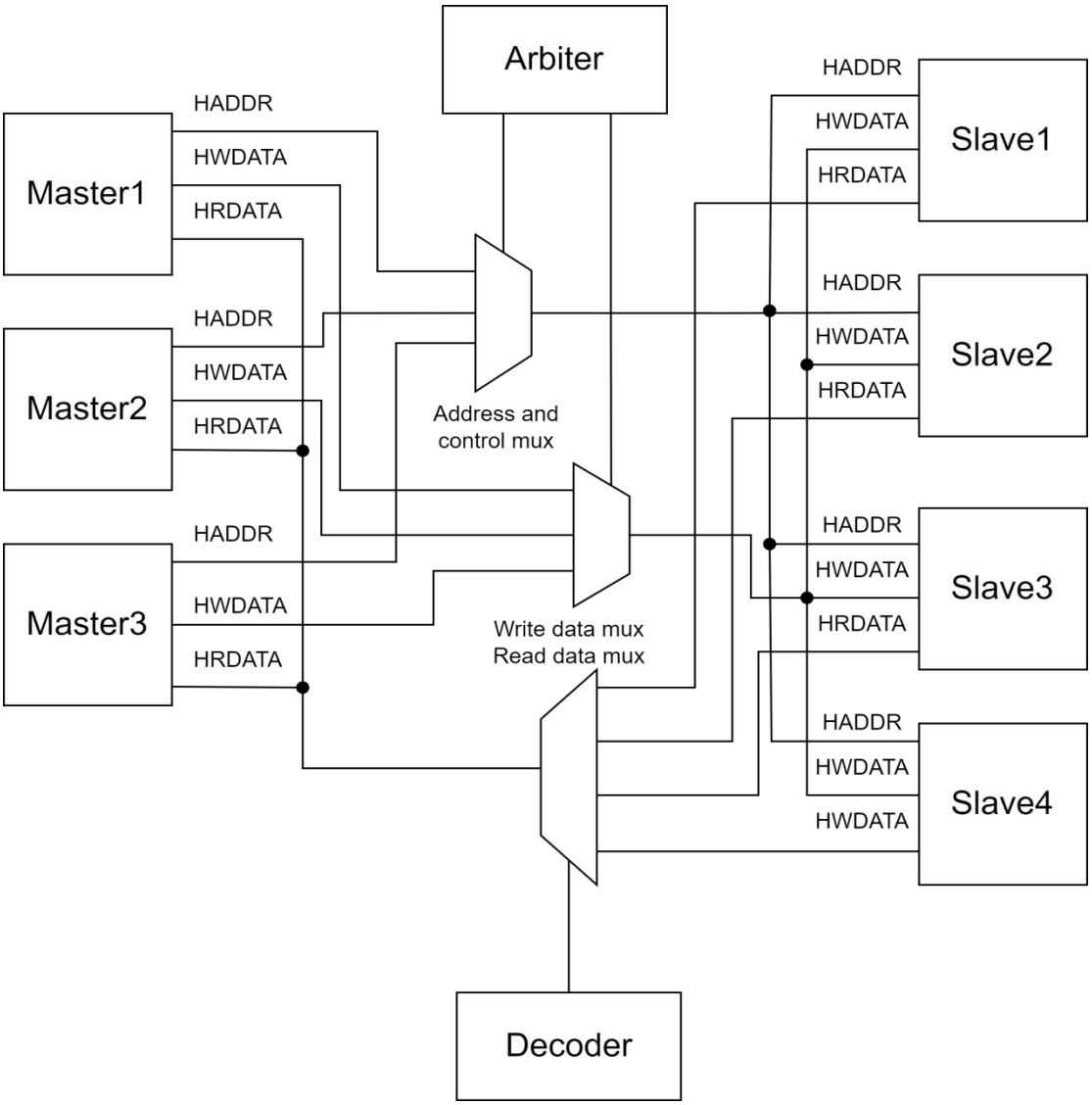

图 4　AHB 总线主要结构

有需要占用总线的 Master 向 arbiter 发出请求，arbiter 授权给指定的 master。获得总线授权的 master 开始 AHB 传输，首先发出地址和控制信号，提供地址信息、传输方向、带宽和 burst 类型。总线统一规划 slave 的地址，译码器根据地址和控制信号确定哪个 slave 与 master 进行数据通信。数据传输通过数据总线完成。为避免出现三态总线，AHB 将读写总线分开，写数据总线用于从 master 到 slave 的数据传输，读数据总线用于从 slave 到 master 的数据传输。每笔传输包括一个地址和控制周期，一个或多个数据周期。

2.3.Cortex-M0 的架构和外设

本系统将Cortex-M0作为唯一的master,利用FPGA片内寄存器资源实现RAM（DATARAM）和ROM（CODERAM），使用AHB总线连接CPU，RAM和ROM，并引出接口控制器连接外设（slave）以此来构成一个独立完整的片上SOC系统。

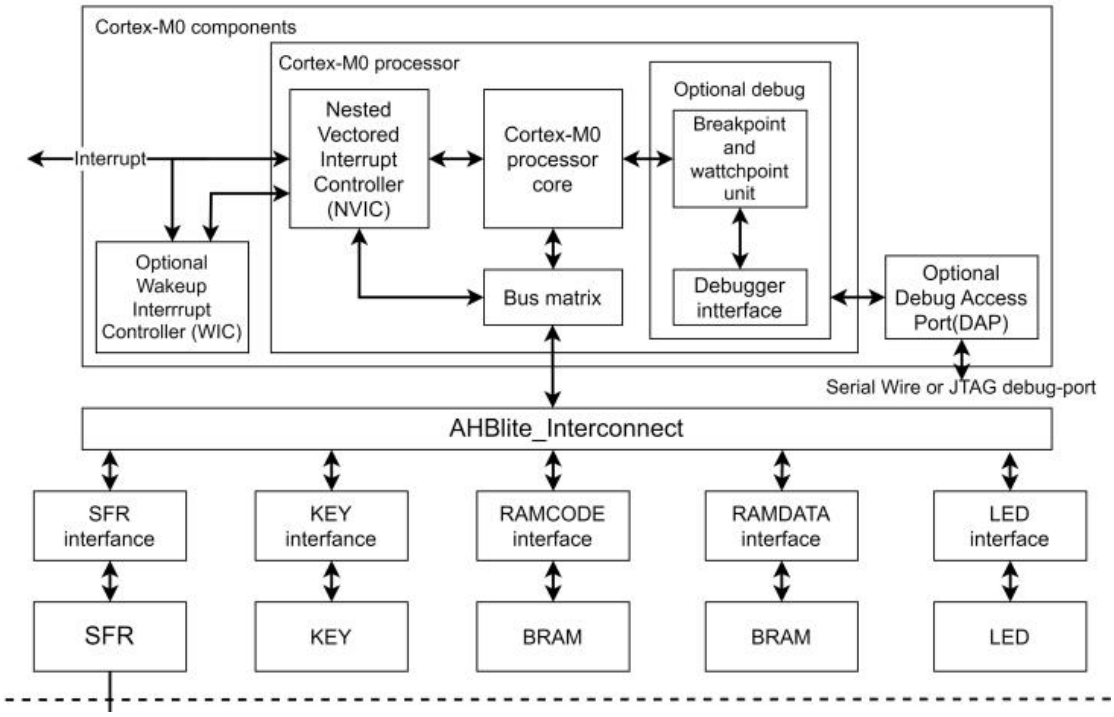

图 5 Cortex-M0的架构及外设

本系统依靠FPGA片上资源实现一块64KB（2^14*32）的RAM和一块64KB的ROM（可以通过修改参数进行更改），通过地址译码器Decoder在AHB总线上引出控制器接口，添加LED和KEY两个外设，实现通过按键改变LED的显示效果证实系统的运行情况，可以通过按键修改特殊功能寄存器的值控制屏幕显示哪个文件，并通过LED进行提示。也可以通过修改SFR的值改变显示视频的分辨率，所使用的视频处理算法。

3. 软硬件功能划分

3.1. 硬件功能

主要实现基于Cortex_M0的片上SOC系统的搭建，SD卡读写，FIFO控制数据收发，ISP视频数据处理，同步动态随机存储器SDRAM，VGA显示模块和HDMI显示模块。

利用SDIO协议读取SD卡中的原始数据，并且将原始数据送入FIFO中，ISP

模块请求 FIFO 中的数据，对原始数据进行数据处理，处理完成后将数据发送给 SDRAM，HDMI 模块将 SDRAM 中的数据显示在屏幕上。

3.2. 软件功能

通过检测按键改变 LED 的运行状态证实系统运行状况，通过按键或代码改变 SFR 的值，控制 SD 卡模块读取不同的原始数据，进而将不同视频显示在 PC。通过 KEY 改变 ISP 的运行状态，将同一视频不同 ISP 算法处理过后的视频显示在 PC 端，并进行对比。也可以通过修改 SFR 的值改变显示视频的分辨率。

4. ISP 硬件算法介绍及架构

4.1. ISP 硬件算法介绍

5.1.1 Bayer 输入

Bayer 输入，即 ISP 输入。输入采用 3*16 读取图像三行的 Bayer 数据，并分别添加一个数据有效信号和数据请求信号，来保证数据传输的正确性。读取完 3*3 的数据后，保存进内部缓存中，方便后面进行色彩转换。

5.1.2 色彩转换

为了保证速度，色彩转换采用线型插值算法，通过计算 3*3 范围内各颜色的平均值，来获得中心像素点的颜色数据，由此输出三通道 RGB 的色彩。同时针对不同的 Bayer，进行算法的适配，保证其色彩转换的通用性。后续完成后，再考虑使用更加复杂准确的 CFA 插值算法，提高颜色的准确性。

图 6 Bayer 转换示意图

$$\mathrm{Color} = \frac{1}{\mathrm{n}}\sum_{1}^{3}\sum_{1}^{3}\mathrm{Color}_{\mathrm{i,j}}$$

Color 表示 RGB 三色中的一色，n 表示在 3*3 内该颜色所占的格子，位于中心的颜色直接取中心点的值，不用求平均值。

值得注意的是，由于插值算法是计算 3*3 范围内的中心像素点的颜色数据，因此无法保持原视频流的分辨率。

5.1.3 色深转换

由于 HDMI 视频输出以及包括速度在内的各方面考虑，在没有进行深度的算法优化的情况下，EG4S20 不具备直接输出 12bit 色深的能力。因此在初期，我们对于图像的色深通过算法将色深调整到 8bit，再将其输出，使其能够正常输出视频流。

5.1.4 图像裁剪

由于视频是 1936*1088 的视频流，但由于 ISP 处理后，为了满足 VESA 视频流的标准要求。通过输入偏移和分辨率，裁切成为符合 VESA 视频流的标准要求。

5.1.5 色彩矫正

由于 bayer 输入的图像可能会有较大的色差，因此我们需要对于输入的图像进行色彩矫正。通过设置三个个矫正系数与原来的 RGB 数值对应，两者相乘并与色深的最大值作比较，如果超出最大值就输出色深的最大值，小于就输出校正后的 RGB 数值。其核心公式如下：

$$\begin{cases} R' = R_{cor} * R \\ G' = G_{cor} * G \\ B' = B_{cor} * B \end{cases}$$

$$\{R'',G'',B''\}=\{\max\{R',R_{max}\},\max\{G',G_{max}\},\max\{B',B_{max}\}\}$$

其中R_{cor}，G_{cor}，B_{cor}为矫正系数，R_{max}，G_{max}，B_{max}为相应原始数据最大值。R''，G''，B''为最后的输出值。

5.1.6 伽马矫正

伽马校正是一种非线性操作，用于对线性亮度或 RGB 值进行编码和解码，以匹配显示设备的非线性特性。伽马校正有助于将数据映射到感知上更均匀的域，

从而优化有限信号范围（例如每个 RGB 分量中有限的位数）的感知性能。在最简单的情况下，伽马校正由以下公式定义：

$$V_{out} = AV_{in}^{\gamma}$$

A 是输入的增益系数，γ 小于 1 称为γ衰减，γ 大于 1 称为γ增幅。

5.1.7 对比度增强

视觉感知中的对比度是同时或连续看到的两个或多个视场部分的外观差异（因此：亮度对比度、明度对比度、颜色对比度、同时对比度、连续对比度等）。对比度定义为系统能够产生的最亮颜色（白色）的亮度与最暗颜色（黑色）的亮度的比率。高对比度比率是任何显示器所期望的方面。它与动态范围类似。

在此利用分段线性变换的方法，分段线性变换的公式可以表示为：

$$g(i,j) = T[f(i,j)]$$

其中：

g(i,j) 表示输出图像在坐标 (i,j) 处的像素值

f(i,j) 表示输入图像在坐标 (i,j) 处的像素值

T 表示分段线性变换函数

T 函数可以用多个线性函数段来定义，每个线性段对应一个特定的灰度值范围。假设我们定义了 k 个线性段，每个线性段的起始灰度值为 r_i，终止灰度值为 r_{i+1} (i = 0, 1, ..., k-1)，则第 i 个线性段的表达式为：

$$g = a_i * f + b_i \quad (r_i <= f < r_{i+1})$$

其中：

a_i 表示第 i 个线性段的斜率，它决定了灰度值的拉伸或压缩程度。

b_i 表示第 i 个线性段的截距，它决定了灰度值的偏移量

由于视频为黑夜和白天，我们要让暗部区域进行拉伸，亮部区域进行压缩，可以用以下两个线性段来定义 T 函数：

线性段 1: $r_0 = 0$, $r_1 = 128$, $a_1 = 2$, $b_1 = 0$

线性段 2: $r_1 = 128$, $r_2 = 255$, $a_2 = 0.5$, $b_2 = 128$

这意味着：当图像灰度值在 0 到 128 之间时，使用斜率为 2 的线性段进行拉伸。当图像灰度值在 128 到 255 之间时，使用斜率为 0.5 的线性段进行压缩。通过这种方式，可以根据需要调整每个线性段的斜率和截距，从而实现不

同的对比度增强效果。

5.1.8 边缘增强

边缘增强算法是一类图像处理技术，旨在通过增强图像边缘的对比度来提高图像的清晰度和锐利度。它通常用于改善图像的视觉效果，尤其是在图像细节较弱或模糊的情况下。边缘增强算法通过识别图像中边缘区域，并在边缘区域周围增加对比度来实现增强效果。其基本原理可以概括为以下步骤：

边缘检测： 首先使用边缘检测算子（如 Sobel 算子、Canny 算子等）识别图像中的边缘区域。

边缘增强： 然后，在边缘区域周围增加对比度，从而增强边缘的视觉效果。常见的增强方法包括：

锐化滤波： 使用高通滤波器来增强图像的细节和边缘，例如拉普拉斯算子、高斯-拉普拉斯算子等。

非锐化掩模 (Unsharp Masking)： 通过将原始图像与一个模糊版本进行比较，并将差异叠加到原始图像上，来增强图像的边缘。

高频增强： 将图像分解为低频和高频部分，增强高频部分来提高图像锐度。

直方图均衡化： 通过调整图像的灰度级分布，增强图像的对比度，从而突出边缘。

5.1.9 饱和度矫正

饱和度矫正算法是一种图像处理技术，用于调整图像的饱和度，使其更符合视觉上的舒适度或更接近真实世界中的颜色。饱和度是指颜色的纯度或强度，饱和度高的颜色鲜艳明亮，饱和度低的颜色则偏灰暗。饱和度矫正算法的目标是调整图像中不同颜色区域的饱和度，使其更符合视觉要求，例如：增强图像的鲜艳度、降低图像的色彩过度以及修正图像的颜色偏差等。

在此利用分段线性饱和度调整方法，可以针对不同的饱和度区域进行不同的调整，假设我们将饱和度范围划分为 N 个区间，每个区间对应一个比例因子 k_i，则分段线性饱和度调整的公式可以表示为：

$$S' = S * k_i \quad (S_i <= S < S_{i+1})$$

其中：

S_i 和 S_{i+1} 表示第 i 个区间的起始和终止饱和度值。

k_i 表示第 i 个区间的比例因子。

5.1.10 白平衡矫正

白平衡矫正是指在不同光照条件下，将图像的色彩还原为真实色彩，消除光源颜色带来的影响。例如，在日光下拍摄的照片，由于日光偏暖色，照片中白色物体可能会呈现黄色，而如果在灯光下拍摄照片，由于灯光偏冷色，照片中白色物体可能会呈现蓝色。白平衡矫正就是通过调整图像的色彩平衡，消除这些光源色温的影响，使图像呈现出更真实的色彩。

白平衡矫正的基本原理是首先需要在图像中识别一个白色点，可以是白色物体，也可以是白色背景。然后根据白色点的颜色信息，调整图像的红色、绿色和蓝色通道的比例，使白色点呈现为真正的白色。最后将调整后的颜色平衡应用于整个图像，使所有其他颜色的还原更接近真实颜色。

在此我们使用完美反射算法，该算法假设图像中存在一个完美反射的白色点，该点的 R、G、B 值相等且为最大值。因此，可以根据这个假设来计算一个比例因子 r，用于调整每个通道的值。在计算的过程中，对 RGB 之和的直方图进行倒叙遍历，找到使白点像素个数超过总像素个数比例的阈值，T；遍历原始图像，计算 RGB 之和大于 T 的像素，各个通道取平均，得到$\overline{R}$，$\overline{G}$，$\overline{B}$；并且得到每隔通道的最大值R_{MAX}，G_{MAX}，B_{MAX}

利用其核心公式：

$$\begin{cases} R^{'} = R * R_{MAX}/\overline{R} \\ G^{'} = G * G_{MAX}/\overline{G} \\ B^{'} = B * B_{MAX}/\overline{B} \end{cases}$$

就可以得到处理后的 RGB 数据。

4.2. ISP 架构介绍

原始视频数据经过色彩，色深转换，图像裁剪，色彩矫正，伽马矫正，边缘增强，饱和度和白平衡矫正后原始数据变成经过处理的 RGB 数据并通过 HDMI 显示。

5. 目前进度

5.1. Arm Cortex-M0

在安路 EG4S20BG256 上成功搭载 Arm Cortex-M0 处理器，利用 FPGA 板上寄存器资源实现了 64KB 的 ROM 和 RAM，支持使用外部在线开发调试的 JTAG 调试接口，并且搭载了连接芯片外部引脚的 GPIO 等外设，目前已经实现流水灯等功能。可以使用 keil 工具编写并生成的文件，实现在 SoC 运行的软件程序的在线实时调试。

5.2. ISP 硬件加速算法

目前该模块主要完成了 raw 格式向 rgb 格式的转换，色彩和色深的转换，以及通过修改相应参数实现对视频的裁切，改变视频的分辨率，满足当前状况下 hdmi 的显示要求。已经完成的 ISP 模块如下。

5.2.1 Bayer 输入

Bayer 输入，即 ISP 输入。输入采用 3*16 读取图像三行的 Bayer 数据，并分别添加一个数据有效信号和数据请求信号，来保证数据传输的正确性。读取完 3*3 的数据后，保存进内部缓存中，方便后面进行色彩转换。

5.2.2 色彩转换

为了保证速度，色彩转换采用线型插值算法，通过计算 3*3 范围内各颜色的平均值，来获得中心像素点的颜色数据，由此输出三通道 RGB 的色彩。同时针对不同的 Bayer，进行算法的适配，保证其色彩转换的通用性。后续完成后，再考虑使用更加复杂准确的 CFA 插值算法，提高颜色的准确性。

5.2.3 色深转换

由于 HDMI 视频输出以及包括速度在内的各方面考虑，在没有进行深度的算法优化的情况下，EG4S20 不具备直接输出 12bit 色深的能力。因此在初期，我们对于图像的色深通过算法将色深调整到 8bit，再将其输出，使其能够正常输出视频流。

5.2.4 图像裁剪

由于视频是 1936*1088 的视频流，但由于 ISP 处理后，为了满足 VESA 视频流的标准要求。通过输入偏移和分辨率，裁切成为符合 VESA 视频流的标准要求。

5.2.5 显示效果

基于目前搭载的 ISP 算法，利用原始视频数据对 ISP 算法进行仿真，根据仿真后的结果可以显示经过 ISP 算法处理后的图片，下图是 night 中原始数据经过处理后的结果：

图 7 只做格式转换的图片

图 8 ISP 仿真后的图片

在 FPGA 上搭载 ISP 算法后，只做格式转换，和利用 ISP 处理后的图像如下图所示：

图 9 只做格式转化的图片

图 10 经过 ISP 算法处理后的图片

5.3. 视频显示

5.3.1 SD 卡读写

Micro SD 卡，原名 Trans-flash Card（TF 卡），SD 卡从物理结构看包括 5 个部分，分别为存储单元、存储单元接口、电源检测、卡及接口控制器和接口驱

动器。

常规的 SD 卡共有 9 个引脚接口，其中包括 3 根电源线、1 根时钟线、1 根命令线和 4 根数据线。工作模式有两种：SDIO 模式和 SPI 模式。在 SPI 模式下，SD 卡共使用到 CS(DAT[3])、CLK、MISO(DAT[0])、MOSI(CMD)四根信号线；SPI 总线与多张 SD 卡连接时，除 CS 片选信号线不可共用外，其他信号均可公用。

在 SDIO 模式下，SD 卡共使用到 CLK、CMD、DAT[3:0]六根信号线；SDIO 总线与多张 SD 卡连接时，可以共用 CLK 时钟信号线，对于 CMD、DAT[3:0]信号线，每张 SD 卡都要独立连接。

与 SD 卡的通信是基于命令和数据传输的。通讯由一个起始位(“0”)开始，由一个停止位(“1”)终止。SD 通信一般是主机发送一个命令(Command)，从设备在接收到命令后作出响应(Response)，如有需要会有数据(Data)传输参与。SD 数据是以块(Block)形式传输的，SDHC 卡数据块长度一般为 512 字节（SD 卡一扇区为 512 字节），数据可以从主机到卡，也可以是从卡到主机。读写操作都是由主机发起的，主机发送不同的命令表示读或写， SD 卡接收到命令后先针对命令返回响应。在读操作中， SD 卡返回一个数据块，数据块中包含 CRC 校验码；在写操作中，主机接收到命令响应后需要先发送一个标志（TOKEN）然后紧跟一个要写入的数据块，卡接收完数据块后会返回一个数据响应及忙碌标志，当 SD 卡把接收到的数据写入到内部存储单元完成后，会停止发送忙碌标志，主机确认 SD 卡空闲后，可以发送下一个命令。读写时序如下：

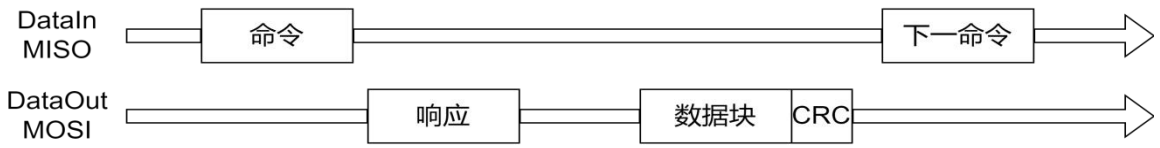

图 11 单块读操作

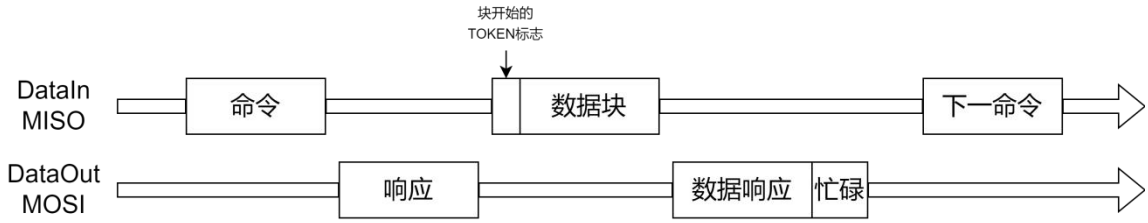

图 12 单块写操作

SD 卡读写系统由初始化模块，SD 卡读模块和 SD 卡写模块构成，目前实测工作频率最高可达 75MHz，由于 SPI 模式下数据传输位宽为一，理论传输速率有 75Mbps，因为 SPI 单线传输速度过慢的问题，后续会使用 SDIO 协议使用四线进行 SD 卡的读写，以提高 SD 卡的读写速度。

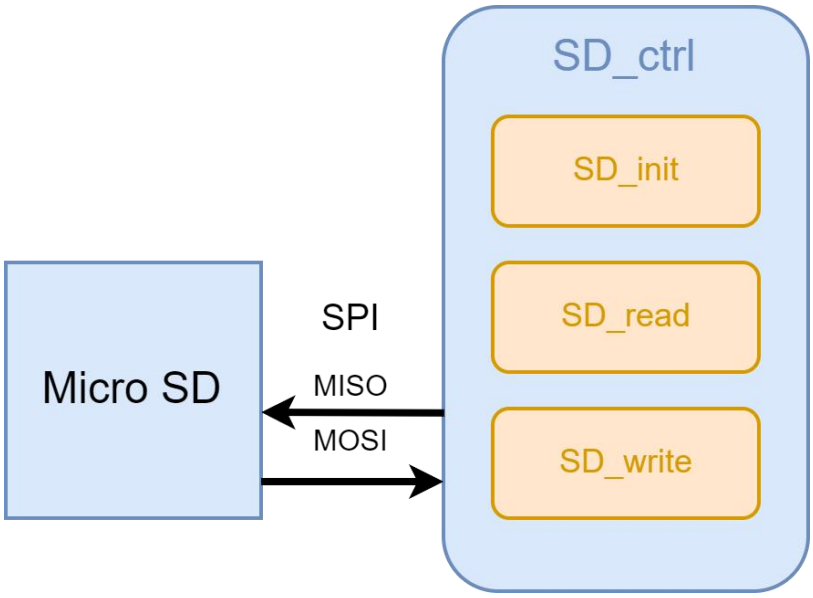

图 13 SPI 模式 SD 卡读写系统

5.3.2 SDRAM 与 FIFO

sdram 系统通过调用 2Mx32bits 的 sdram 硬件资源实现视频帧的存储。ISP 图像处理后的 rgb888 像素点 24bit 数据将补足高 8 位存入位宽 32bit 的 sdram 中，写入地址[HDMI_SDRAM_MIN_ADDR ， HDMI_SDRAM_MIN_ADDR+HPIXELxVPIXEL]共一帧数据。实测工作频率最高达 200MHz。

sdram 系统由 fifo 数据读写暂存系统，多方读写仲裁系统，sdram 控制器系统两大部分构成。其中 fifo 数据读写暂存系统可根据需要例化多个模块以满足多方同时读或写的需求，多方读写仲裁系统负责处理多方同时向 sdram 发生读请求或写请求的 sdram 资源调度，sdram 控制器系统则负责相应读写请求，解析虚拟地址，以及指令传输，并直接连接 sdram 硬件。

fifo 读写暂存模块提供一定范围内的 sdram 地址的数据暂存，内部调用一读一写两个异步软核 fifo，外界只需接入读写请求和数据线，fifo 将自动按顺序将数据存入指定地址范围内，当满足突发长度后会自动全页写入或读出，非常适合图片、视频存入和读出等批量固定寻址范围的操作，且读 fifo/写 fifo/读 sdram/写 sdram 可分别使用不同的时钟域，实现 HDMI 与 SDRAM 的跨时钟域数据

传输。

多方读写仲裁模块是为了处理当出现多方同时希望读写 sdram 时的调度问题，以达到分线器的效果。时序仿真下可在两个时钟周期内完成 sdram 调度。目前实测可在双读双写下正常工作，理论上支持扩展三方甚至更多方的接口，当未来需要加入更多模块读写 sdram 时可轻松接入系统中。

sdram 控制器又可分为 sdram_init，sdram_read，sdram_write，sdram_arbit 四大模块，分辨负责初始化，读/写，以及仲裁模块，实现 sdram 充电和初始化，自动刷新，突发读写等操作，并向外提供虚拟地址，方便外部模块调用资源。

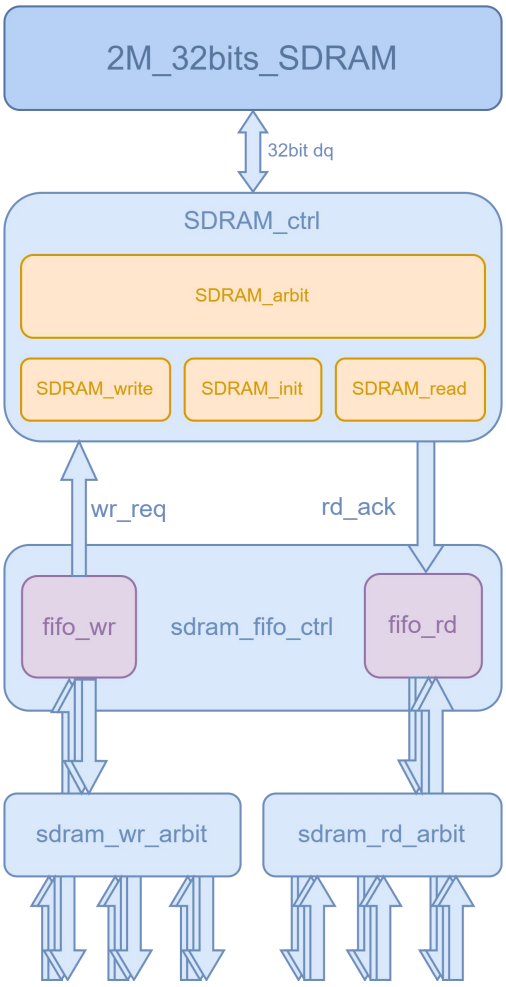

图 14 SDRAM 控制器结构

sdram 系统设计思路为层层简化：将复杂的 sdram 指令控制、上电、自刷新操作等等拆解为一个个状态机切换；全权交给 sdram_ctrl 模块处理后，sdram 对外界的表现如同 BRAM 一样，再无需考虑指令操作，外界只需连接地址线，请求/响应线和数据线即可与 sdram 建立通讯。

fifo 读写暂存模块将视频帧的读写所需的计数操作统一管理，外界只需提供时钟线、数据线和请求线，系统便可跨时钟域处理请求，自动寻址并存入；多方读写仲裁模块则使外界模块调用 sdram 时无需考虑其是否处于被占用状态，只需等待仲裁模块发回响应即可。最终，sdram 的操作变得简单，便捷，快速，智能，既可跨时钟域操作，也可任意突发长度读写。

5.3.3 HDMI 显示

高清晰度多媒体接口 HDMI（High-definition multimedia interface）是消费产品常用接口，也已经成为行业的连接标准。HDMI 是一个可在单一接口结合无压缩高清视频，多声道环绕音频，和智能命令数据的数字接口。HDMI 连接高清设备极为简单，同时对最新的产品功能提供一致和高性能效的传输。

TMDS(Time Minimized Differential Signal)最小化传输差分信号传输技术，是一种利用 2 个引脚间电压差来传送信号的技术。传输数据的数值(“0″或者″1”)由两脚间电压正负极性和大小决定。采用 2 根线来传输信号，一根线上传输原来的信号，另一根线上传输与原来信号相反的信号。这样接收端就可以通过让一根线上的信号减去另一根线上的信号的方式来屏蔽电磁干扰，从而得到正确的信号。HDMI 利用 TMDS 进行信息传输，可以提高传输的稳定性。

我们利用 VGA 时序根据 HDMI 的协议独立设计了 HDMI 模块。

HDMI 输出系统由三部分组成：fifo 数据暂存系统，VESA 时序生成系统，TMDS 差分输出系统。整体采用参数化，流水线设计，可以适应多 640x480@60Hz, 1280x1024@60Hz, 1920x1080@60Hz 等多种分辨率输出模式。

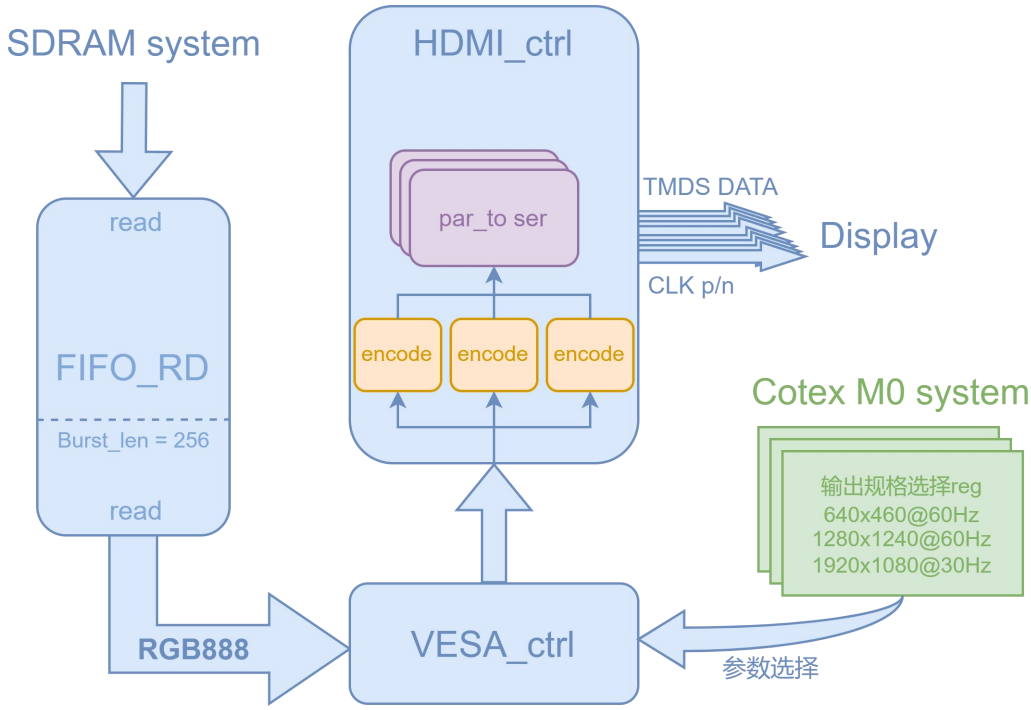

图 15 HDMI 输出系统

fifo 数据暂存系统负责向 sdram 读请求数据和数据暂存，一次请求可读取 256x32bits 数据量，理论带宽可达 1.8Gbits/s。

VESA 时序生成系统由 VGA 时序模块改装而成，内置行场定义参数可适应主流标准和非标准 VESA 时序，支持 RGB565、RGB555、RGB16、RGB24、RGB32 等多种输出格式。

TMDS 差分输出系统中的数据线使用时钟双边沿触发传输数据，传输效率更高，差分信号由芯片内置硬件 oddr 实现，实测最高时钟频率可达 540MHz。

HDMI 输出系统可与 Cortex M0 子系统通过寄存器建立通讯，Cortex M0 可通过更改寄存器值改变 HDMI 输出规格，如 1280x1024@60Hz rgb888，640x480@60Hz rgb888 等。未来预计由 Cortex M0 控制 HDMI 接口的 SCL 和 SDA 线，与显示器建立 I2C 通讯，实时获取显示器参数信息，针对性更改 HDMI 输出规格，实时监测显示器连接情况，在线缆拔出后实现向 FPGA 发送待机指令，降低功耗。

5.4. 其他

我们设计了 SD 卡，SDRAM 的仿真模型，可以利用仿真模型对系统进行行为级仿真。可以使用 TD 生成的文件在 modelsim 上进行时序仿真。

6. 未来展望

6.1. ISP 硬件加速算法

今后会逐步实现还未完成的 ISP 算法，也将不断的尝试新的 ISP 算法，争取在充分利用 FPGA 资源的情况下实现更好的视频处理效果。

6.2. SD 卡控制系统

由于 SPI 模式下单线传输速率较低，今后会根据 SDIO 协议对 SD 卡读写模块进行更新，根据 SDIO 协议实现 4 线读写，提高带宽，50M 时钟频率下理论传输速率达 200Mbps。未来也会和 HDMI 模块自设板卡。

6.3. Arm Cortex-M0

设置更多的特殊功能寄存器，使得 CPU 可以更好的控制硬件电路，建立与 HDMI 控制系统的连接，使得 Cortex M0 可通过更改寄存器值改变 HDMI 输出规格，如 1280x1024@60Hz rgb888，640x480@60Hz rgb888 等。未来预计由 Cortex M0 控制 HDMI 接口的 SCL 和 SDA 线，与显示器建立 I2C 通讯，实时获取显示器参数信息，针对性更改 HDMI 输出规格，实时监测显示器连接情况，在线缆拔出后实现向 FPGA 发送待机指令，降低功耗。

未来也会在 Cortex-M0 系统中挂载 UART 外设，通过 UART 传输数据，实现对 SFR 的更改，提高系统的灵活性。

6.4. HDMI 输出系统

未来会自行设计板卡，集成 HDMI 模块，VGA 模块和 SD 卡模块，实现双路 HDMI 输出，实现更高时钟频率下稳定运行，提高 HDMI 传输效率。同时建立与 Cortex-M0 系统的联系，实现 Cortex-M0 控制 HDMI 系统显示效果。