数字逻辑项目选题介绍（2020 年秋季学期）

南方科技大学计算机系
阮业淳，黎诗龙，朱元绍，于顺昌，张楠，张家澍，华正昌，王薇

说明:
Topic3(ALU),4(出租车计费器) 为特殊选题，仅供单人或双人组队选择。Topic1,2,5 可供双人或以上规模的小组选择。

Topic1. 智能空调

设计一个智能空调: 智能空调分为开机状态和关机状态，开机状态下拥有三种模式：送风模式，制冷模式、制热模式。智能空调在制冷和制热模式下可以随时设定温度，环境温度随之发生变化。智能空调使用充值卡来控制供电，在开机状态会消耗电量，用户可以进行电量充值。智能空调开机时能够显示空调设定温度、当前的环境温度、当前工作模式。进行充值卡充值时可以显示剩余可用电量和输入的充值电量。

1. 实现智能空调初始化。实现信息显示功能
   a) 环境温度、智能空调设定温度、剩余可用电量的初始值分别是 24.0℃、25.0℃、10.0 w·h
   b) 初始化指的是程序在开发板上刚开始运行时的数据初始化，或者按下 reset 键（需要自己实现）之后的所有数据初始化
2. 实现智能空调状态和模式控制功能
   a) 智能空调在开机状态下可以随时修改工作模式（可在送风、制冷、制热三种模式之间进行切换），开机时初始模式为送风模式
   b) 关机状态：不影响环境温度，没有电量消耗
   c) 开机状态：
      i. 送风模式：不影响环境温度，每 2 秒消耗 0.5w·h 电量
      ii. 制冷模式：如果环境温度高于智能空调设定温度，环境温度每 1 秒下降 0.1℃，每 1 秒消耗 1w·h 电量，否则，进入送风模式
      iii. 制热模式：如果环境温度低于智能空调设定温度，环境温度每 1 秒上升 0.1℃，每 1 秒消耗 1w·h 电量，否则，进入送风模式
   d) 关机状态时只显示当前的环境温度；开机状态时显示当前工作模式，当前的环境温度，设定温度
3. 实现智能空调温度设定功能
   a) 设定温度可用范围是 20~30℃
   b) 设定温度上升/下降的单元温度是 1℃，即每次只能上升/下降 1℃
   c) 智能空调在开机状态下可以随时设定温度
   d) 开机时的设定温度应为最近一次关机时的设定温度，第一次开机的设定温度为 1. 中提到的初始值 25.0 ℃
4. 实现电量充值功能
   a) 当剩余可用电量为 0 时，蜂鸣器响起警报并持续 3 秒，蜂鸣器停止警报后智能空调自动关机

b) 进入充值卡充值状态，显示剩余可用电量和输入的充值电量
c) 退出充值卡充值状态，显示进入充值卡充值状态之前的信息
d) 支持任意 w·h 的电量充值，最小精度为 0.1

5. 实现电量充值记录查询功能
   a) 进入充值记录查询状态时，显示可查询记录的条数 N，输入编号显示对应的充值记录
   b) 在充值过程中显示充值的电量，充值后的电量
6. 实现定时关机功能
   a) 在开机状态下，可以设定关机倒计时，当倒计时结束时，智能空调自动关机
   b) 建议使用一个 LED 灯标记当前是否处于倒计时状态

提示：温度、电量的显示可考虑使用七段数码管，工作模式的显示可考虑使用 LED 灯，充值电量的输入考虑使用小键盘。

Topic2. 简易售货机

设计一个简易的售货机，基本功能与校园内的售货机功能大致相同。支持选择货道，显示货道号、商品名称、商品剩余数量以及商品金额；支持计时功能，付款过程有时间限制；支持基本的付款以及找零功能，对不同的付款状态应该有不同的提示。管理员可以进行补货操作，通过选择货道和补货数量进行补货，可以查看商品的销售情况（销售总金额、各个商品的售出数量）。

1. 查询阶段。
   1） 通过小键盘或者拨码开关选择货道
      a) 滚动显示货道号、商品名称、商品剩余数量、商品金额
      b) 可切换货道查看信息
   2） 通过确认按钮进入付款阶段
      a) 如果商品剩余数量为 0，显示提示信息并回到查询阶段
2. 付款阶段。
   1） 付款阶段显示商品单价以及剩余付款时间
   2） 基本信息
      a) 商品单价：均为整数，且小于 20 元。（可自定）
      b) 付款支持的单张金额：1 元、2 元、5 元和 10 元。（可自定）
      c) 付款时间限制：30 秒。（可自定）
   3） 付款状态
      a) 未完成付款：如果在付款时间内，付款金额小于商品金额，则提示顾客继续付款。
      b) 超出付款时间：如果在付款时间内未完成付款，则显示退款信息。
      c) 付款成功：显示付款成功信息，如果付款金额大于商品单价，显示找零金额，相应的商品数量要进行调整。
      d) 各种状态可以发出提示（比如通过蜂鸣器发出不同的提示音）

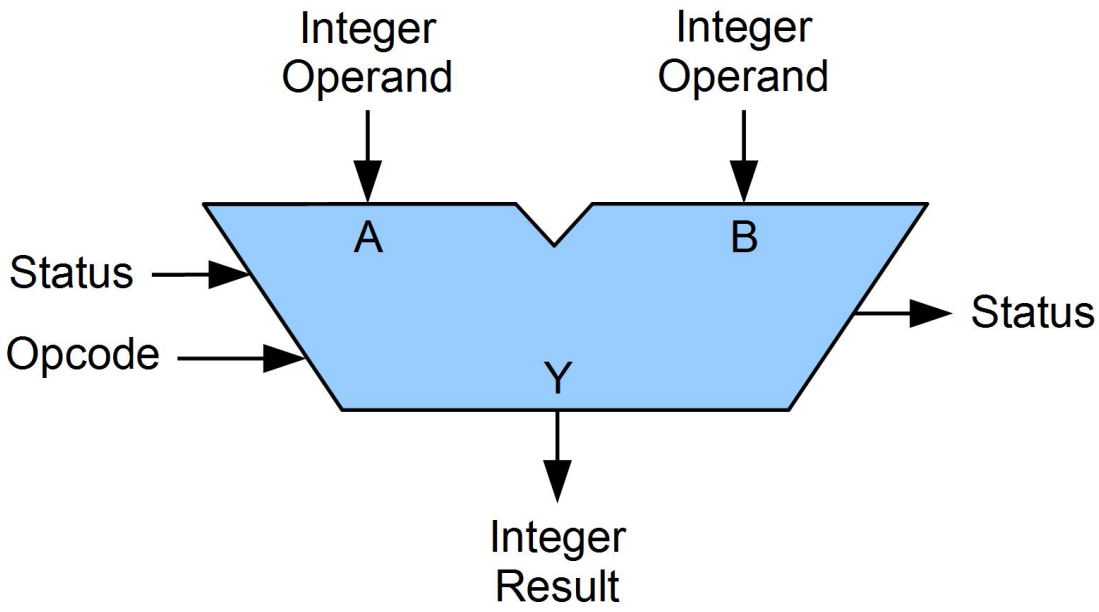
4） 剩余付款时间：在数码管上显示剩余付款时间，如果付款成功，则清零，超出付款时间则回到初始状态。
5） 可通过按钮返回查询阶段

3. 实现补货功能，由管理员进行操作。
   1） 初始状态所有货道商品数量均为 0，通过复位按钮实现初始化。
   2） 管理员选择货道进行补货
      a) 选择货道时显示当前货道商品名称和剩余数量
      b) 输入补货数量进行补货
      c) 补货完成后显示成功信息并显示补货后商品的剩余数量
   3） 查看商品的销售情况
      a) 查看各商品的售出数量
      b) 查看总销售金额
4. 实现其他功能
   1） 选商品时可自带 BGM
   2） 按动小键盘或拨动拨码开关可有不同的提示音。
   3） 支持一次购买多件同一商品

Topic3. 32 位 ALU 设计

说明：该任务不要上板测试，仅供单人或双人组队选择。（全部完成仅得总分 80%）
任务和要求：使用 Verilog 实现一个 32 位 ALU，要求根据输入的 opcode 能计算两个 32 位有符号整数之间的运算，并输出结果。其中乘法器和除法器的构建需要自行使用行为级建模构建，不可以使用 Verilog 的高级综合功能（即直接使用运算符）。

设计说明与提示：
算术逻辑单元（英语：Arithmetic logic unit，简称：ALU）是一种可对二进制整数执行算术运算或位运算的组合逻辑数字电路。ALU 的输入包括需要运算的数据（也称为运算数）和表明了运算操作类型的指令码。ALU 的输出是其执行运算的结果。在许多的设计中，ALU 还带有状态输入或输出，可将其之前操作或当前操作的信息在 ALU 和外部状态寄存器间传递。



一个基本的 ALU 具有三个并行的数据总线，包括两个输入操作数（A、B）和结果输出（Y）。每个数据总线都是一组传递了一个二进制整数的信号。通常，A、B 和 Y 的总线宽度（每个总线包含的信号数）是相同的。操作码通过并行总线输入，它向 ALU 传递选择的操作的信息，该操作选择码是对 ALU 要执行的算术或逻辑运算的枚举。ALU 的状态输出是各类单独的信号，补充表示了有关当前 ALU 操作结果的信息，例如溢出等。

需要支持的运算：

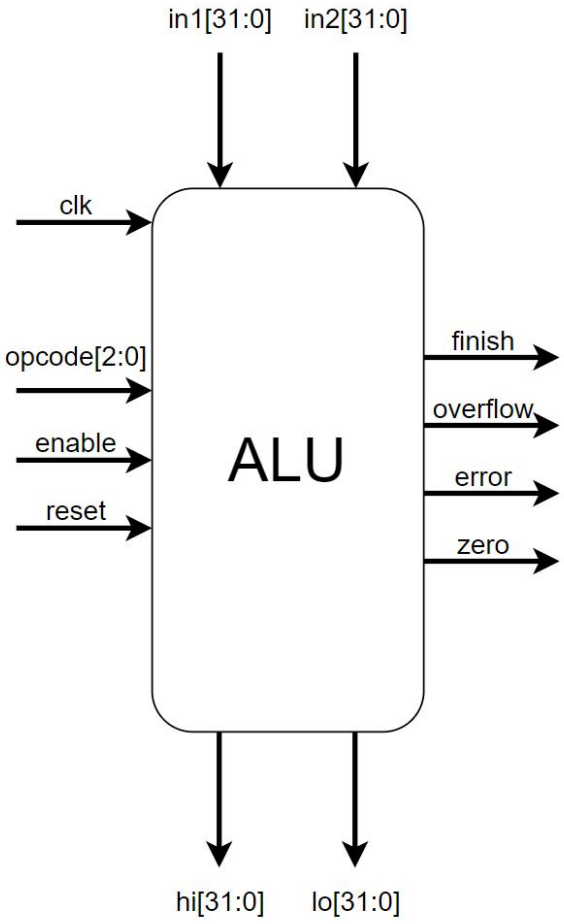
运算操作	缩写表示	opcode(binary)
加法	ADD	000
减法	SUB	001
乘法	MUL	010
除法	DIV	011
按位与	AND	100
按位异或	XOR	101
取反码	INV	110
左循环移位	SLL	111

输入输出端口（均为高电平有效）：

端口名称	方向	位宽	功能定义
in1	IN	32	第一个操作数
in2	IN	32	第二个操作数
clk	IN	1	时钟信号
opcode	IN	3	操作码
input	IN	1	输入有效标志
reset	IN	1	重置
hi	OUT	32	结果高 32 位
lo	OUT	32	结果低 32 位
finish	OUT	1	完成标志
overflow	OUT	1	溢出标志
error	OUT	1	错误标志
zero	OUT	1	输出为 0 标志

对输入输出端口的一些说明如下：

- hi, lo 两个端口分别存储结果的高 32 位与低 32 位。因为加法和减法不会产生 32 位以上的结果（若有溢出，overflow 位置 1，结果仍为 32 位），因此在加减法运算时只需将结果放在低 32 位输出即可。
- 如果加法或减法产生了溢出，将 overflow 位置 1。运算得到的结果可以截去高位后输出。
- 在乘法运算时，结果最高为 64 位，因此需要将结果的高 32 位和低 32 位分别放在 hi 和 lo 两个输出端口中。
- 除法运算时会产生余数和商，将商放在低 32 位，余数放在高 32 位输出。
- 除法运算时可能会遇到除数为 0 错误，此时将错误位置 1。
- 如果运算后高 32 位和低 32 位结果均为 0，将 zero 输出置 1，表示输出均为 0。
- 由于运算过程中会使用时序逻辑，因此在运算完成结果就绪后，finish 输出应当被置 1，表示结果运算已经完成，可以被读取。
- 在 input 信号上升沿到来时，ALU 应该读取输入的两个操作数和操作码，并开始运算。运算完成 finish 置 1 后，结果应当保持，直到下一个 input 信号上升沿到来。下一个 input 上升沿到来时，finish 信号应被置 0。
- 当 reset 信号被置 1 时，整个 ALU 应立即停止运算，并将所有输出端口置为初始状态。



设计提示：因为乘法器和除法器涉及到比较多的逐位运算，因此会在实际电路中产生很大的延迟。为了克服延迟带来的不确定性，防止结果错误，因此乘法器和除法器的设计应使用时序逻辑电路。建议使用多级流水线设计，在组合逻辑电路之间插入寄存器暂存中间数据。

Topic4. 出租车计价器

说明：该任务需要上板测试，仅供单人或双人组队选择。（全部完成仅得总分 80%）
任务和要求：设计一个出租车计价器实现出租车的计费功能，自行设计实现时间增加和路程增加功能，能够实现出租车计费，自行分配数码管，并利用数码管显示时间，路程和车费。
1. 出租车计费分为等待计费和行驶计费两部分（30 分，每部分 15 分）
 1）等待计费：等待时间两分钟内不计费，超过两分钟的每分钟 1 元计费（不足 1 分钟的按 1 分钟记）；
 2）行驶计费：行驶计费标准为起步价 5 元，超过 3km 后按每公里 2 元计费（不足 1 公里的按 1 公里记），超过 20 公里的部分，每公里加收 50%的行驶费用。
2. 利用数码管显示等待时间和行驶时间，显示时间按分钟记，每增加一分钟数码管显示时间+1。
3. 利用数码管显示当前车费和行驶里程，增加的车费和里程全部取整数。
4. （可选）利用 VGA 显示任务要求 2 和 3。（说明：该项功能可选，如果完成可在基础分之上加得附加分）

Topic5. 简易 VGA 显卡

任务和要求：

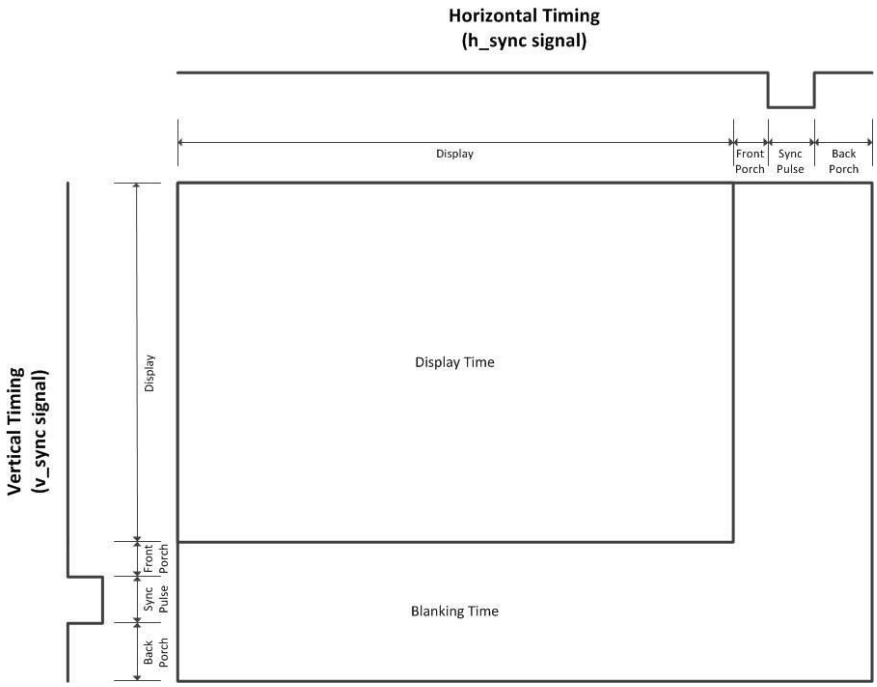
1. 使用开发板上的 FPGA 芯片和 VGA 输出接口，向显示器输出分辨率为 640*480 的彩色条带图像（黑、白、红、绿、蓝、青（蓝+绿）、粉（红+蓝）、黄（红+绿））。**（系统分中占比 20%）**
2. 通过简单的串口指令，实现绘图功能。要求可以在指定坐标绘制二维图形（至少包括点、线、矩形、圆，实现更多图形可算作 bonus），且可以控制颜色。**（系统分中占比 20%）**
3. 利用串口，使用提供的图像文件，将图像显示在屏幕上。**（系统分中占比 20%）**
4. （可选项）实现 VGA 不同分辨率之间的切换，用开发板上的按键控制分辨率切换，并在数码管上显示当前分辨率。同时，2、3 中的功能需要保持可用（可以不保留原有图像）**（系统分中占比 20%）**

设计说明与提示：

VGA 显示信号的工作原理：在老式的 CRT 显像管显示器中，为了显示一帧画面，显像管所发出的电子束需要在显示屏上进行逐行扫描，激发屏幕内不同颜色的荧光粉发光，从而形成不同明暗和颜色的画面。扫描从屏幕左上角开始，从左向右逐点扫描。每扫描完一行，电子束将回到下一行起始位置，并开始新一行的扫描。为了防止回到起始位置时造成路径的发光，在此期间电子束将被停止发射（称为消隐）。每行扫描

结束时，采用行同步信号进行同步。当全部行完成扫描时，形成一帧，并使用场同步信号进行同步，电子束回到屏幕左上角开始准备下一帧的扫描。在电子束回到起始点的过程中也需要进行消隐。

然而时至今日（2020），CRT 显示器在日常用途中早已被更先进的液晶显示器所淘汰，液晶显示器的原理决定了它不再需要按照逐行扫描的方式显示图像。然而 VGA 作为已被广泛应用的显示协议之一，出于兼容性的考虑而被保留了下来。因此，在 VGA 协议中一些 CRT 显示器时序上的特征被保留了下来。下面的图片显示了 VGA 的工作时序：



（https://www.digikey.com/eewiki/download/attachments/15925278/signal_timing_diagram.jpg?version=1&modificationDate=1368216804290&api=v2）

可以看到行时序和场时序都包含有四个部分：Display, front porch, sync pulse 和 back porch. 在 display 段内，显示器接受 VGA 端口发送来的数据信号，并将接收到的数据显示在屏幕上。Sync Pulse 就是我们提到过的行同步脉冲和场同步脉冲。Front porch 和 Back porch 是在同步脉冲前后的非显示区域，在老式的 CRT 显示器中用于控制图像的偏移。现代显示器都带有图像自动调节功能，因此我们只需要按照标准实现即可。Front porch、back porch、sync pulse 位于消隐间隔内，在此期间 RGB 信号无效，不显示数据。另外，在 VGA 的工业标准中，行同步，场同步都为负极性，即同步脉冲要求是负脉冲。

具体到不同分辨率和刷新率，不同部分所包含像素数各不相同。对于 640*480@60Hz，时序数据如下：

General timing

Screen refresh rate	60 Hz
Vertical refresh	31.46875 kHz
Pixel freq.	25.175 MHz

Horizontal timing (line)

Polarity of horizontal sync pulse is negative.

Scanline part	Pixels	Time [µs]
Visible area	640	25.422045680238
Front porch	16	0.63555114200596
Sync pulse	96	3.8133068520357
Back porch	48	1.9066534260179
Whole line	800	31.777557100298

Vertical timing (frame)

Polarity of vertical sync pulse is negative.

Frame part	Lines	Time [ms]
Visible area	480	15.253227408143
Front porch	10	0.31777557100298
Sync pulse	2	0.063555114200596
Back porch	33	1.0486593843098
Whole frame	525	16.683217477656

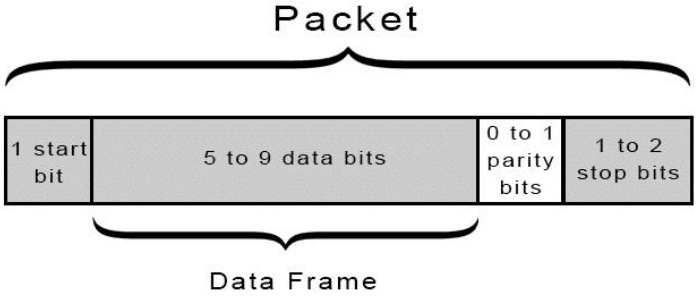
(http://tinyvga.com/vga-timing)

由于 25.175 MHz 和 25MHz 极为接近，因此在一般实践中，我们取像素频率为 25MHz。关于 Pixels 和时间的换算，我们可以用像素数除以像素频率来获得。例如，水平时序中的可见区域共有 640 像素，则所占时间为 600/25M=24us。

VGA 采用的是红绿蓝三原色模拟信号输出像素数据。我们可以将每个颜色分量最亮看作 1，最暗看作 0，利用板载 VGA 接口的电阻网络实现不同颜色值的像素输出。通过利用电阻分压网络，我们可以实现每个颜色分量 4bit 的色彩输出。关于电阻网络的具体用法，请参考 Minisys 或 EGO1 的使用手册。

总结：要实现基础的 VGA 视频输出，我们只需要向显示器提供两组信号：一组是时序信号（vsync 和 hsync），一组是数据信号（RGB 三个通道的模拟信号，通过电阻网络实现）。显示器会根据接口提供的时序自动识别分辨率信息，同时显卡发送的数据信号也需要遵循像素频率的规定。

USB-UART 串口通过指令绘图：UART 的全称是通用异步收发器，是实现设备之间低速数据通信的标准协议。“异步”指不需要额外的时钟线进行数据的同步传输，双方约定在同一个频率下收 发数据，此接口只需要两条信号线（RXD、TXD）就可以完成数据的相互通信， 接收和发送可以同时进行，也就是全双工。 收发的过程，在发送器空闲时间，数据线处于逻辑 1 状态，当提示有数据要传输时，首先使数据线的逻辑状态为低，之后是数据位、一位校验位（可选）、1-2 位停止位。校验一般是奇偶校验，停止位用于标示一帧的结束，接收过程亦类似， 当检测到数据线变低时，开始对数据线以约定的频率抽样，完成接收过程。UART 的数据帧格式如下：



通常建议使用 8 位数据位和 1 位停止位，无校验位。

波特率表示数据传输的速率，单位 bps，表示位每秒。比如 9600bps 就表示 1s 可以传输 9600bits 的数据。异步收发没有时钟控制数据的传输，就需要保证收发双方在波特率设置上的一致。确保接收数据的完整性。程序中通常使用 16 倍速率对 UART 通信时序进行采样，则 UART 通信所需的时钟频率就是 16*bps。常见的波特率有 9600、

115200 等。板载的 CP2102 支持波特率范围在 300-1Mbps 之间的任意波特率，我们可以利用开发板上的时钟分频后对串口进行采样并将其还原成我们所传输的数据。

绘图的具体要求如下：

1. 绘制点。要求在指定的坐标绘制一个像素点大小的点，颜色可以指定。
2. 绘制线。要求在指定的起始坐标和终止坐标之间绘制直线（直线可能会是倾斜直线），颜色可以指定、线宽可以指定。
3. 绘制矩形。要求在指定的坐标给定长和宽绘制矩形，矩形的填充颜色可以指定。
4. 绘制圆。要求在指定的坐标，以指定的半径绘制圆，圆的填充颜色可以指定。

另外，在绘制的过程中，显示新图形时屏幕上的已有图形应该被保留

串口指令的命令定义样例如下：（同学们也可以自行定义）

```
DRAWPOINT     x, y, color
DRAWLINE start, end, color, width
DRAWRECT x, y, width, height, color
DRAWCIRCLE    x, y, radius, color
```

其中，x, y 为坐标，start，end 为包含 x, y 的起始点和终止点坐标。Color 为颜色，因为本次实验中每个色彩分量的位数为 4 位，因此我们可以使用三位十六进制数表示色彩，例如红色可以表示为：F00。

例如：发送指令：

```
DRAWRECT 100 100 50 50 FFF
```

将在坐标 (100, 100) 处绘制一个长宽均为 50 像素的矩形，颜色为白色。

出于方便解析指令和压缩指令长度的考虑，也可以将上述指令用二进制编码。

在绘制图形的过程中，我们需要暂存已接收到的数据以便在屏幕上显示（显卡输出缓冲区）。这就需要用到开发板上的 BRAM 内存模块，并对 BRAM 模块进行读写。在串口数据到来时，需要实时将接收到的串口数据写入 BRAM，同时屏幕输出应持续。对 BRAM 的操作可以参考如下链接：http://xilinx.eetrend.com/d6-xilinx/blog/2018-10/13759.html

串口显示图片：我们将会提供样例图片（.bin 文件，其中存储着二进制的图片像素数据）以及一个用 Python 语言编写的图片转换脚本（img2bin.py）。提供的图片有两张：spectrum.png 和 rover.png，同学们可以选取任意一张。其中 png 文件为未转换之前的原图，demo.png 文件为去除透明度和降低色深后的效果图，bin 文件为转换后的图像数据文件，其中每个像素点的每个颜色分量使用四位二进制数表示，每两个像素点占用三个字节。如果实现了前一项，本项要求应该很容易实现：只需串口读取图片数据并更新进 BRAM 即可。（Hint：BRAM 其实就是显卡输出缓冲区）

实现不同分辨率之间的切换: 参考不同分辨率下的时序，显示器会根据对应的行时序和场时序自动识别分辨率和刷新率。但需要注意分辨率越大，所需要的时钟频率越高。板载时钟仅为 100MHz，因此可实现的分辨率存在上限。