FPGA 开发平台
用户手册

目 录

一、 简介

在这里，对这款 FPGA 开发平台进行简单的功能介绍。

此款开发板使用的是 ALTERA 公司的 Cyclone IV 系列 FPGA，型号为 EP4CE10F1717N， 256 个引脚的 FBGA 封装。此款 FPGA 的资源如下图所示：

Resources	EP4CE6	EP4CE10	EP4CE15	EP4CE22	EP4CE30	EP4CE40	EP4CE55	EP4CE75	EP4CE115
Logic elements (LEs)	6,272	10,320	15,408	22,320	28,848	39,600	55,856	75,408	114,480
Embedded memory (Kbits)	270	414	504	594	594	1,134	2,340	2,745	3,888
Embedded 18 × 18 multipliers	15	23	56	66	66	116	154	200	266
General-purpose PLLs	2	2	4	4	4	4	4	4	4
Global Clock Networks	10	10	20	20	20	20	20	20	20
User I/O Banks	8	8	8	8	8	8	8	8	8
Maximum user I/O (1)	179	179	343	153	532	532	374	426	528

其中，主要的参数，

参数	数值
逻辑单元 Logic elements(LEs)	10320
内存 Embedded memory(Kbits)	414
乘法器 Embedded 18x18multipliers	23
全局锁相环 PLLs	2
时钟单元 Global Clock Networks	10
最大可用 IO 数量	179
内核电压	1.15V-1.25V(推荐 1.2V);
工作温度	-40-85℃

图为整个系统的结构示意图 1.1 所示：

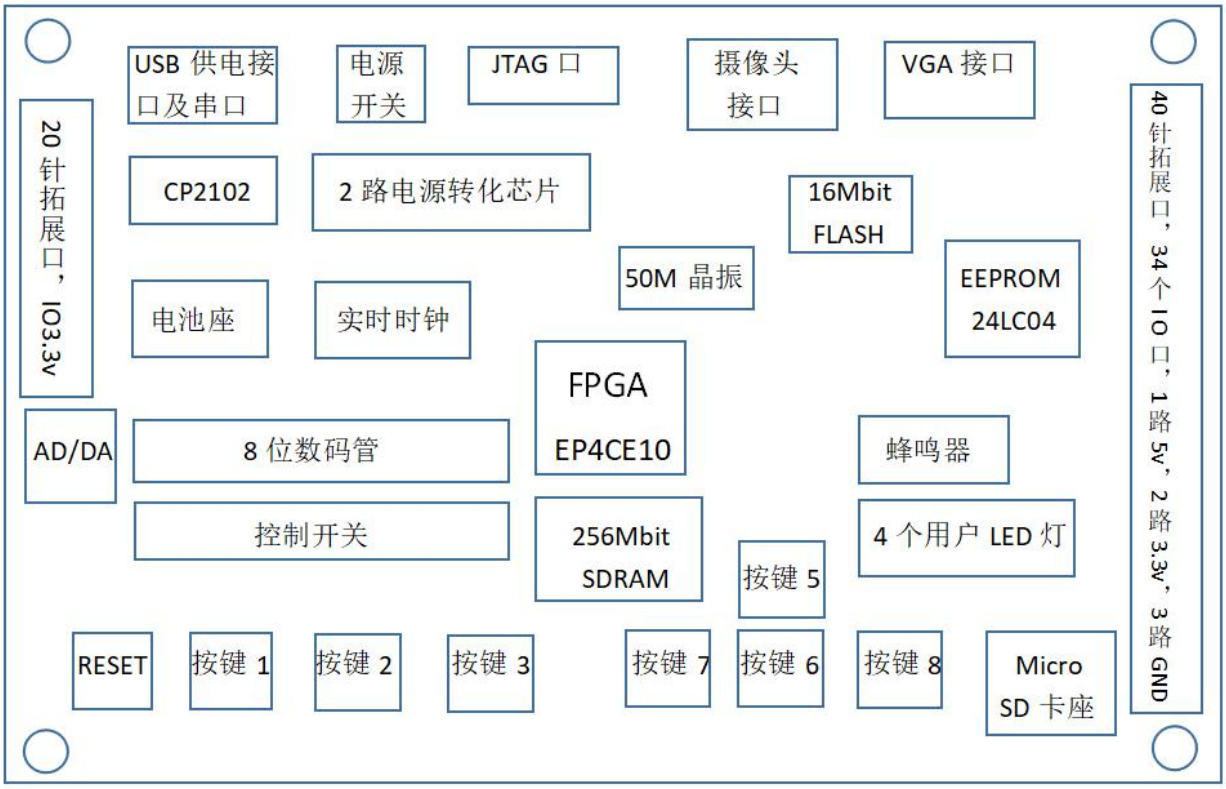


图 1.1 开发板系统结构图

通过这个示意图，我们可以看到，我们这个开发平台所能实现的功能。

- USB 接口供电，同时实现 USB 转串口功能；
- 一片大容量的 256Mbit SDRAM,可作为数据的缓存;
- 一片 16Mbit 的 SPI FLASH, 可用作 FPGA 配置文件和用户数据的存储;
- 一个摄像头接口，可以接 30 万的 OV7670 摄像头或者 500 万的 OV5640 摄像头；
- 一路 VGA 接口，VGA 接口为 16bit，可以显示 65536 种颜色，可以显示彩色图片等信息。
- 一片的 RTC 实时时钟，配有电池座，电池的型号为 CR1220。
- 一片 IIC 接口的 EEPROM 24LC04;
- 4 个红色 LED，可实现流水灯功能;
- 4 个按键，一个复位按键，3 个用户按键；
- 板载 50M 的有源晶振，给开发板提供稳定的时钟源；
- 1 路 40 针的AX 扩展口（2.54mm 间距），其中 34 个 IO 口，1 路 5V 电源， 2 路 3.3V 电源，3 路 GND。可接扩展模块，例如 4.3 寸 TFT 模块和 AD/DA 模块等扩展模块。
- 预留了 JTAG 口，可对 FPGA 进行调试和程序固化。
- 1 路 Micro SD 卡座，支持 SPI 模式。
- 1 个 8 位数码管，可以 8 位数字的动态显示。

二、 电源

开发板通过 USB 供电，用 MINI USB 线将开发板跟电脑的 USB 连接，按键电源开关，既可以给开发板供电。开发板上的电源设计示意图如下:

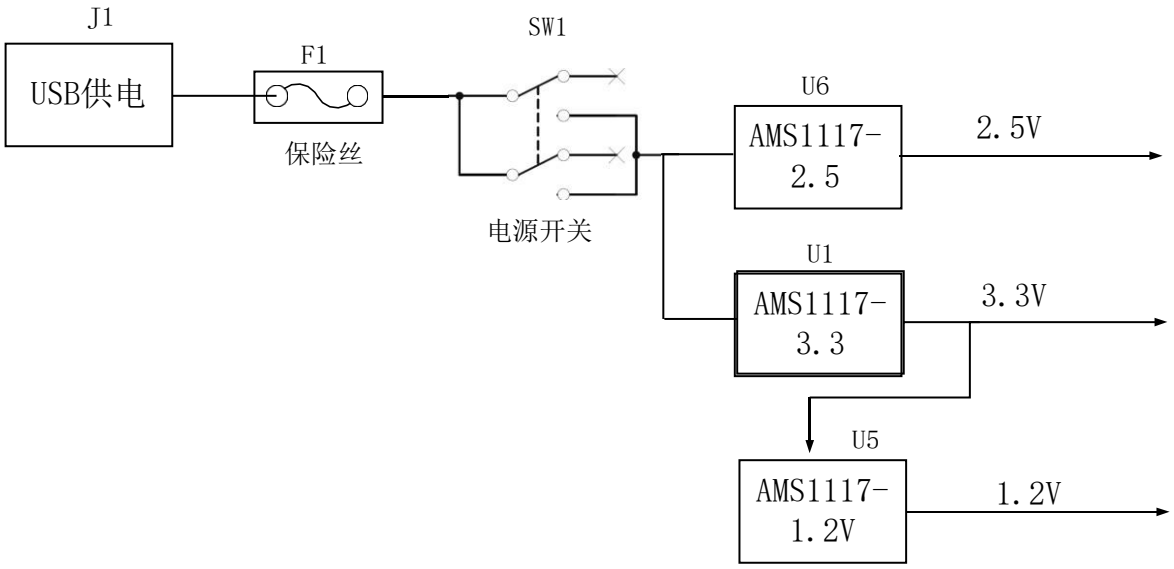


图 2.1 原理图中电源接口部分

开发板用 USB 供电, 通过 3 路 LDO 电源芯片分别产生+3.3V，+2.5V，+1.2V 三路电源，满足 FPGA 的 BANK 电压和内核电压。

在 PCB 板上我们预留了各个电源的测试点， 以便用户确认板上的电压。

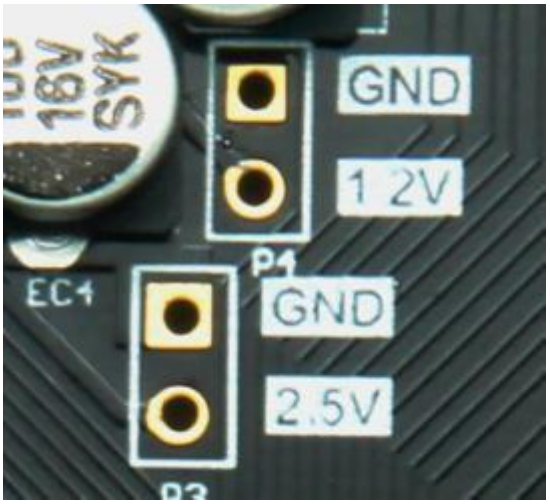


图 2.2 实物图中的电源测试点

三、 FPGA

我们所使用的 FPGA 型号为 EP4CE10F17C8，属于 ALTERA 公司Cyclone IV 的产品。此型号为 BGA 封装，256 个引脚。再次说明一下 FPGA 引脚的定义。我们使用 BGA 封装的芯片以后，引脚名称变为由**字母+数字**的形式，比如 E3，G3 等等，因此我们在看原理图的时候，看到的**字母+数字**这种形式的，就是代表了 FPGA 的引脚。说完这个，我们来看与 FPGA 有关系的各个部分的功能。图 3.1 为开发板所用的 FPGA 芯片实物图。

图 3.1 FPGA 芯片实物

1) JTAG 接口

首先我们来说 FPGA 的配置和调试接口：JTAG 接口。JTAG 接口的作用是将编译好的程序(.sof)下载到 FPGA 中或把 FLASH 配置程序(.jic)下载到 SPI FLASH，sof 文件下载到 FPGA 后，掉电以后就会丢失，需要上电重新下载才可以。这时我们可以通过 Quartus 软件把 sof 文件转换成 jic 文件，通过 JTAG 下载到 jic 文件到开发板的 FLASH 以后，掉电以后就不会丢失，重新上电后 FPGA 会读取 FLASH 中的 jic 配置文件并运行。

图 3.2 就是 JTAG 口的原理图部分，其中涉及到 TCK,TDO,TMS,TDI 这四个信号。这四个

信号直接由 FPGA 引脚引出，每个信号在开发板上做了二级管的过压保护电路。

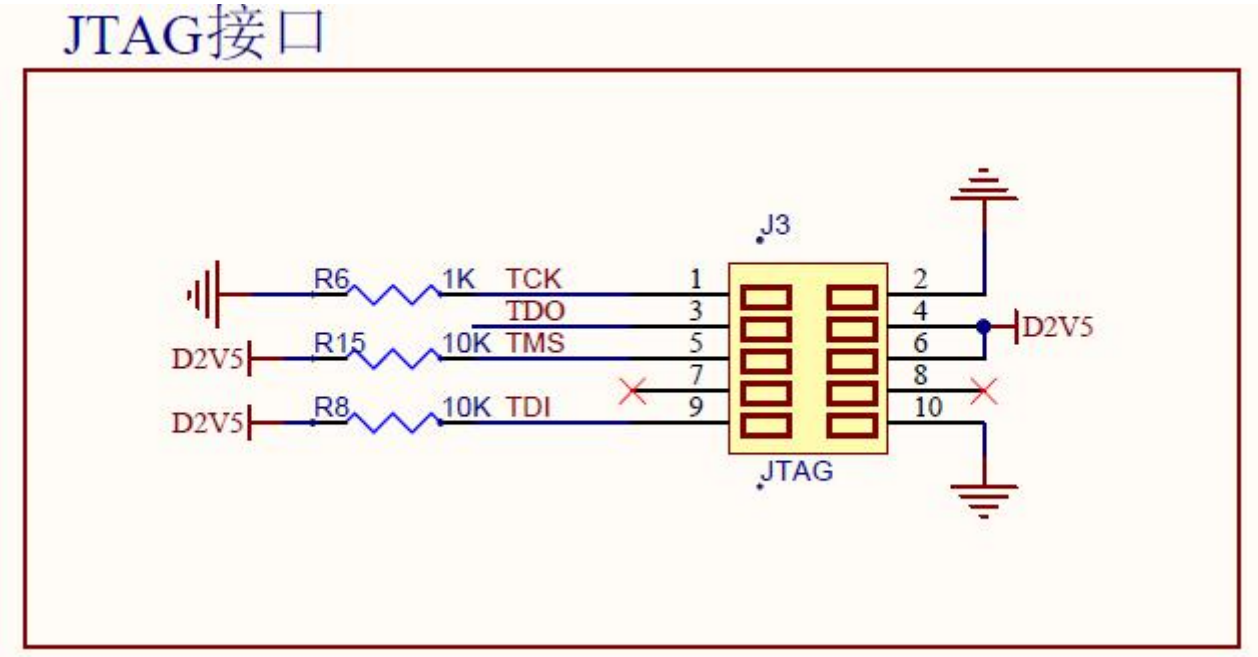


图 3.2 原理图中 JTAG 接口部分

JTAG 接口采用 10 针的 2.54mm 标准的连接器，图 3.3 为 JTAG 接口在开发板上的实物图

图 3.3 JTAG 接口实物图

2) FPGA 电源和GND 引脚

接下来，我们说一下 FPGA 的电源引脚部分，其中包括每一个 bank 的电源引脚,内核电压引脚，模拟电压和锁相环供电引脚，VCCINT 为 FPGA 内核供电引脚，接 1.2V； VCCIO 是 FPGA 的每个 BANK 的供电电压，其中 VCCIO0 是 FPGA 的 BANK0 的供电引脚，同理，VCCIO1~ VCCIO3 分别是 FPGA 的 BANK~BANK3 的供电引脚，在开发板中，VCCIO 都接了 3.3V 电压，也就是说，这款开发板 FPGA 引脚均为 3.3V 输入和输出。VCCA 为 FPGA 模拟供电引脚，接 2.5V， VCCD_PLL 为 FPGA 的锁相环供电引脚，也接 2.5V, FPGA 芯片的电

源连接图如图 3.4 所示。

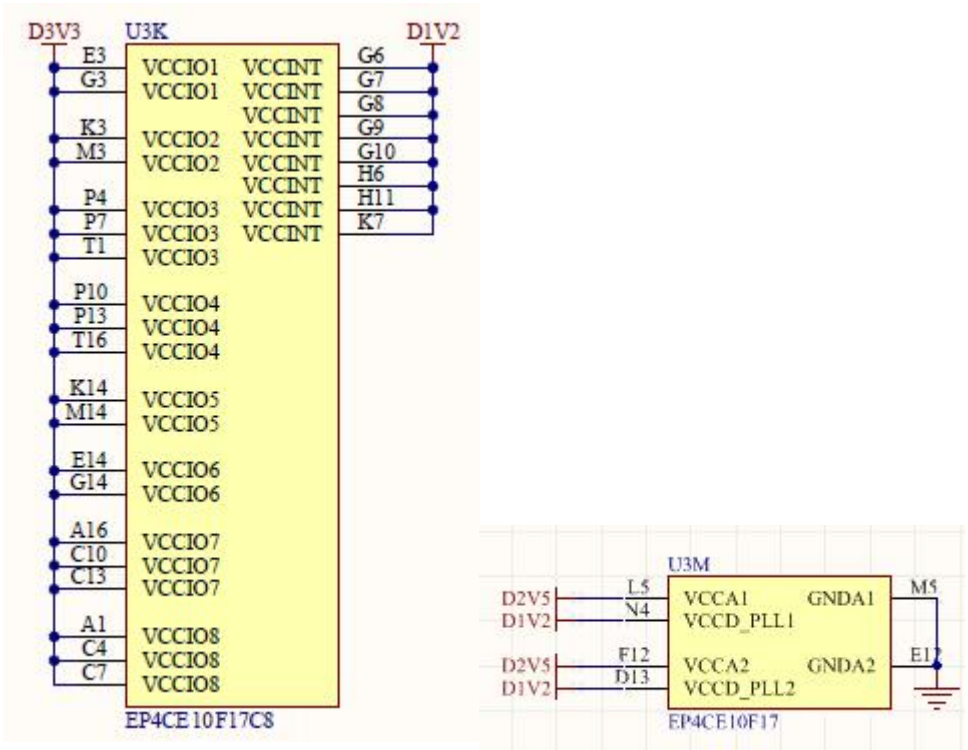


图 3.4 FPGA 电源引脚

另外，FPGA 还有很多引脚需要连接 GND，保证 FPGA 内部有一个平稳的参考地。FPGA 连接的 GND，如图 3.5 所示。

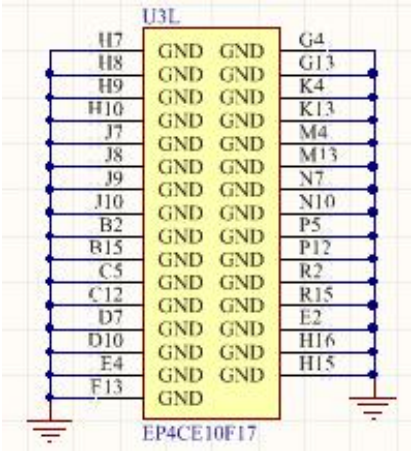
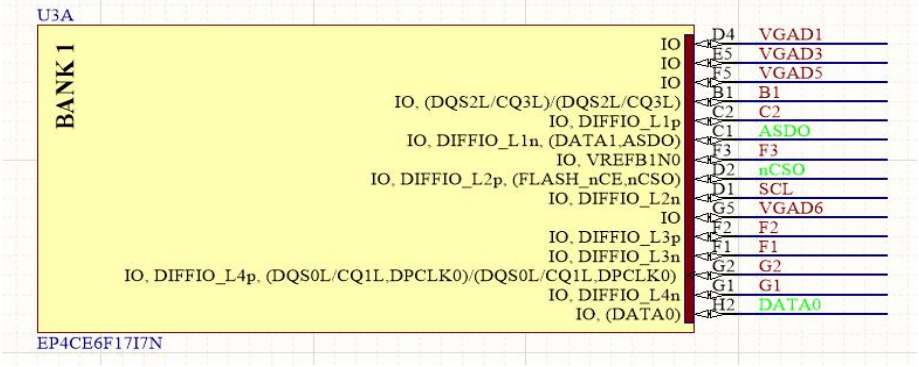
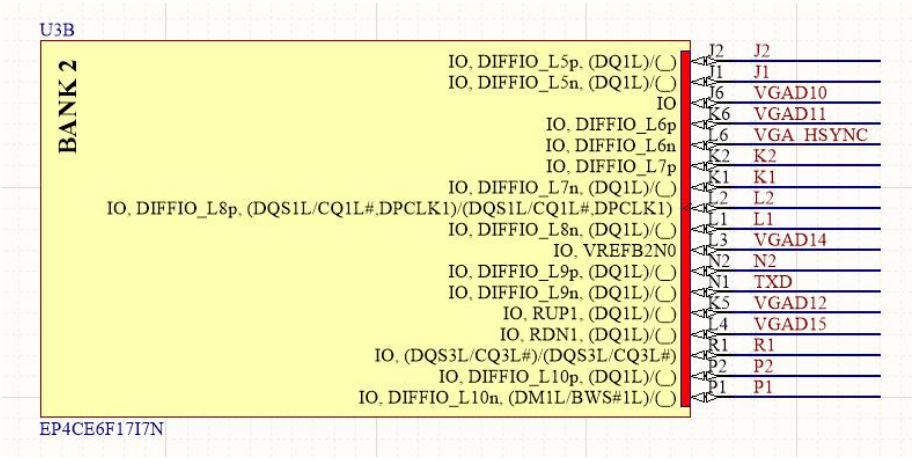
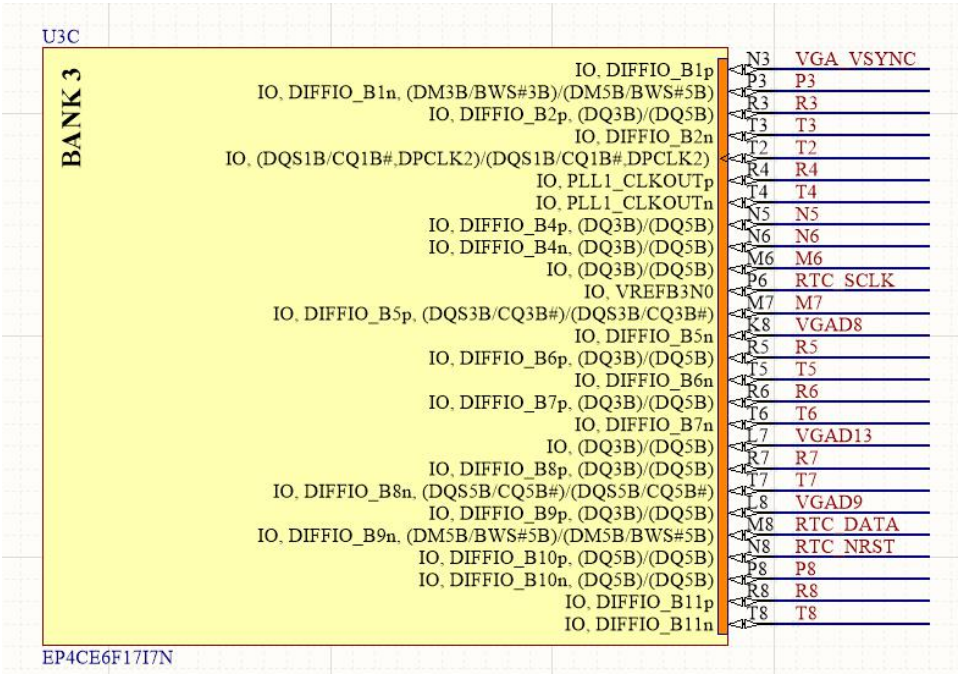


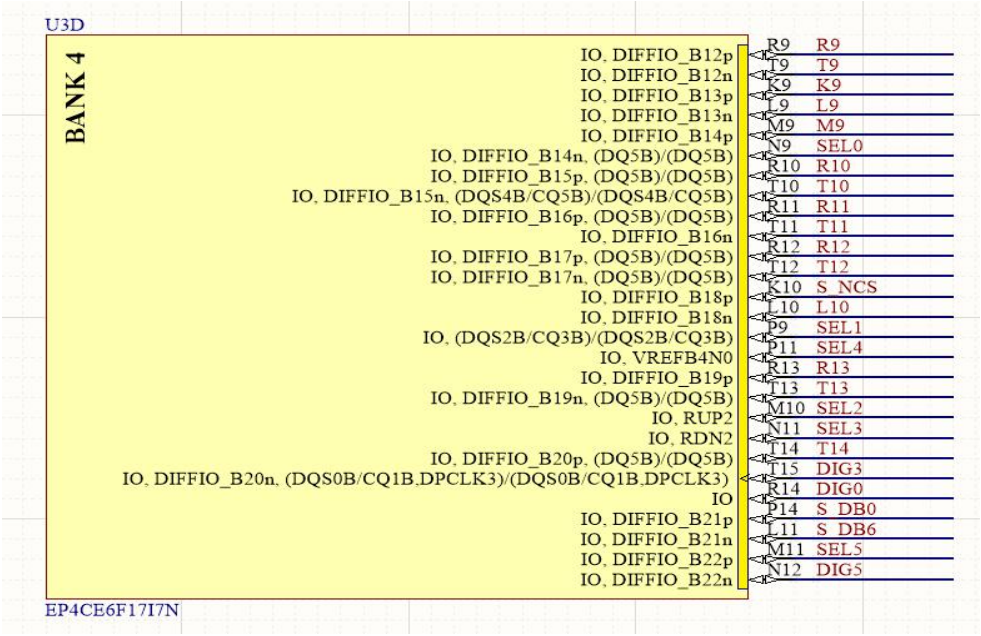
图 3.5 FPGA GND 引脚

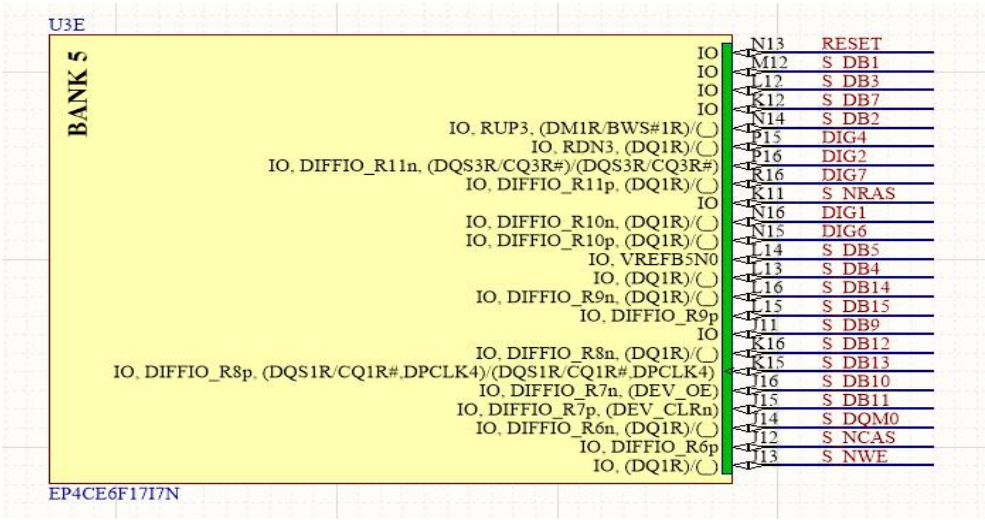
FPGA 各引脚分配：

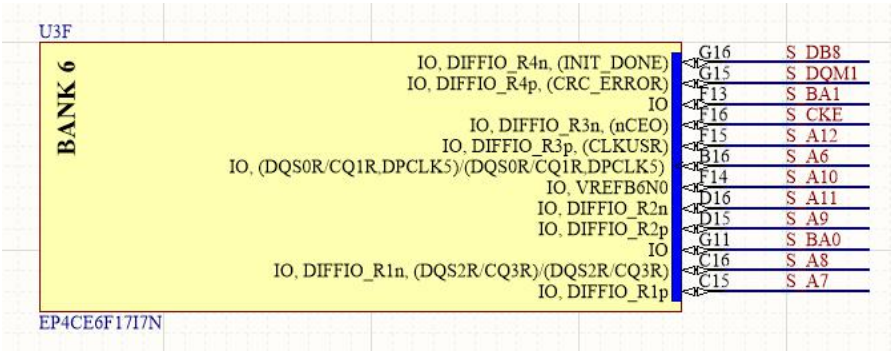


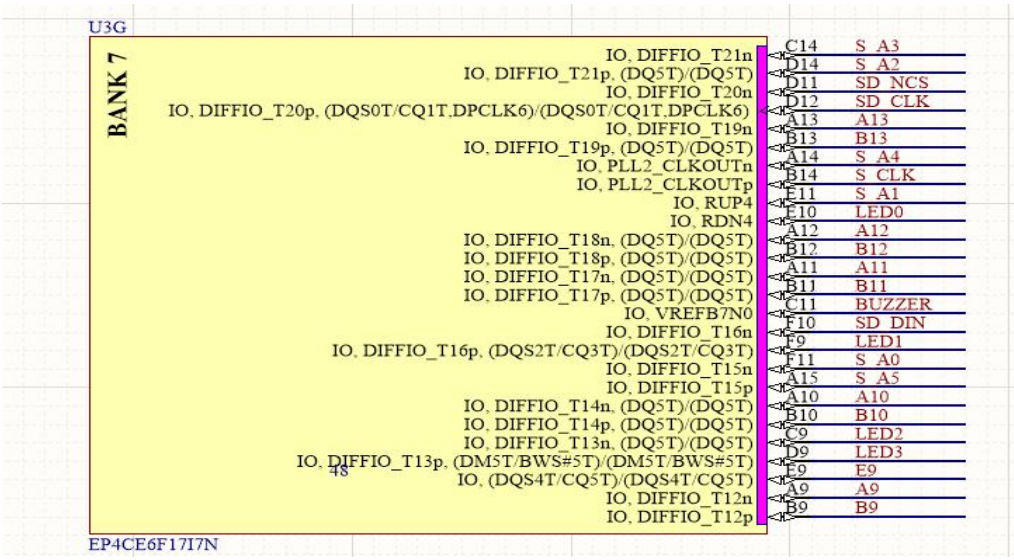

U3B
BANK 2
IO, DIFFIO_L5p, (DQ1L)/(_)
IO, DIFFIO_L5n, (DQ1L)/(_)
IO
IO, DIFFIO_L6p
IO, DIFFIO_L6n
IO, DIFFIO_L7p
IO, DIFFIO_L7n, (DQ1L)/(_)
IO, DIFFIO_L8p, (DQS1L/CQ1L#,DPCLK1)/(DQS1L/CQ1L#,DPCLK1)
IO, DIFFIO_L8n, (DQ1L)/(_)
IO, VREFB2N0
IO, DIFFIO_L9p, (DQ1L)/(_)
IO, DIFFIO_L9n, (DQ1L)/(_)
IO, RUP1, (DQ1L)/(_)
IO, RDN1, (DQ1L)/(_)
IO, (DQS3L/CQ3L#)/(DQS3L/CQ3L#)
IO, DIFFIO_L10p, (DQ1L)/(_)
IO, DIFFIO_L10n, (DM1L/BWS#1L)/(_)
J2 J2
J1 J1
J6 VGAD10
K6 VGAD11
L6 VGA_HSYNC
K2 K2
K1 K1
L2 L2
L1 L1
L3 VGAD14
N2 N2
N1 TXD
K5 VGAD12
L4 VGAD15
R1 R1
P2 P2
P1 P1
EP4CE6F17I7N


U3C
BANK 3
IO, DIFFIO_B1p
IO, DIFFIO_B1n, (DM3B/BWS#3B)/(DM5B/BWS#5B)
IO, DIFFIO_B2p, (DQ3B)/(DQ5B)
IO, DIFFIO_B2n
IO, (DQS1B/CQ1B#,DPCLK2)/(DQS1B/CQ1B#,DPCLK2)
IO, PLL1_CLKOUTp
IO, PLL1_CLKOUTn
IO, DIFFIO_B4p, (DQ3B)/(DQ5B)
IO, DIFFIO_B4n, (DQ3B)/(DQ5B)
IO, (DQ3B)/(DQ5B)
IO, VREFB3N0
IO, DIFFIO_B5p, (DQS3B/CQ3B#)/(DQS3B/CQ3B#)
IO, DIFFIO_B5n
IO, DIFFIO_B6p, (DQ3B)/(DQ5B)
IO, DIFFIO_B6n
IO, DIFFIO_B7p, (DQ3B)/(DQ5B)
IO, DIFFIO_B7n
IO, (DQ3B)/(DQ5B)
IO, DIFFIO_B8p, (DQ3B)/(DQ5B)
IO, DIFFIO_B8n, (DQS5B/CQ5B#)/(DQS5B/CQ5B#)
IO, DIFFIO_B9p, (DQ3B)/(DQ5B)
IO, DIFFIO_B9n, (DM5B/BWS#5B)/(DM5B/BWS#5B)
IO, DIFFIO_B10p, (DQ5B)/(DQ5B)
IO, DIFFIO_B10n, (DQ5B)/(DQ5B)
IO, DIFFIO_B11p
IO, DIFFIO_B11n
N3 VGA_VSYNC
P3 P3
R3 R3
T3 T3
T2 T2
R4 R4
T4 T4
N5 N5
N6 N6
M6 M6
P6 RTC_SCLK
M7 M7
K8 VGAD8
R5 R5
T5 T5
R6 R6
T6 T6
L7 VGAD13
R7 R7
T7 T7
L8 VGAD9
M8 RTC_DATA
N8 RTC_NRST
P8 P8
R8 R8
T8 T8
EP4CE6F17I7N


U3D
BANK 4
IO, DIFFIO_B12p
IO, DIFFIO_B12n
IO, DIFFIO_B13p
IO, DIFFIO_B13n
IO, DIFFIO_B14p
IO, DIFFIO_B14n, (DQ5B)/(DQ5B)
IO, DIFFIO_B15p, (DQ5B)/(DQ5B)
IO, DIFFIO_B15n, (DQS4B/CQ5B)/(DQS4B/CQ5B)
IO, DIFFIO_B16p, (DQ5B)/(DQ5B)
IO, DIFFIO_B16n
IO, DIFFIO_B17p, (DQ5B)/(DQ5B)
IO, DIFFIO_B17n, (DQ5B)/(DQ5B)
IO, DIFFIO_B18p
IO, DIFFIO_B18n
IO, (DQS2B/CQ3B)/(DQS2B/CQ3B)
IO, VREFB4N0
IO, DIFFIO_B19p
IO, DIFFIO_B19n, (DQ5B)/(DQ5B)
IO, RUP2
IO, RDN2
IO, DIFFIO_B20p, (DQ5B)/(DQ5B)
IO, DIFFIO_B20n, (DQS0B/CQ1B,DPCLK3)/(DQS0B/CQ1B,DPCLK3)
IO
IO, DIFFIO_B21p
IO, DIFFIO_B21n
IO, DIFFIO_B22p
IO, DIFFIO_B22n
R9 R9
T9 T9
K9 K9
L9 L9
M9 M9
N9 SEL0
R10 R10
T10 T10
R11 R11
T11 T11
R12 R12
T12 T12
K10 S_NCS
L10 L10
P9 SEL1
P11 SEL4
R13 R13
T13 T13
M10 SEL2
N11 SEL3
T14 T14
T15 DIG3
R14 DIG0
P14 S_DB0
L11 S_DB6
M11 SEL5
N12 DIG5
EP4CE6F17I7N


U3E
BANK 5
IO
IO
IO
IO
IO, RUP3, (DM1R/BWS#1R)/(_)
IO, RDN3, (DQ1R)/(_)
IO, DIFFIO_R11n, (DQS3R/CQ3R#)/(DQS3R/CQ3R#)
IO, DIFFIO_R11p, (DQ1R)/(_)
IO
IO, DIFFIO_R10n, (DQ1R)/(_)
IO, DIFFIO_R10p, (DQ1R)/(_)
IO, VREFB5N0
IO, (DQ1R)/(_)
IO, DIFFIO_R9n, (DQ1R)/(_)
IO, DIFFIO_R9p
IO
IO, DIFFIO_R8n, (DQ1R)/(_)
IO, DIFFIO_R8p, (DQS1R/CQ1R#,DPCLK4)/(DQS1R/CQ1R#,DPCLK4)
IO, DIFFIO_R7n, (DEV_OE)
IO, DIFFIO_R7p, (DEV_CLRn)
IO, DIFFIO_R6n, (DQ1R)/(_)
IO, DIFFIO_R6p
IO, (DQ1R)/(_)
N13 RESET
M12 S_DB1
L12 S_DB3
K12 S_DB7
N14 S_DB2
P15 DIG4
P16 DIG2
R16 DIG7
K11 S_NRAS
N16 DIG1
N15 DIG6
L14 S_DB5
L13 S_DB4
L16 S_DB14
L15 S_DB15
J11 S_DB9
K16 S_DB12
K15 S_DB13
J16 S_DB10
J15 S_DB11
J14 S_DQM0
J12 S_NCAS
J13 S_NWE
EP4CE6F17I7N


U3F
BANK 6
IO, DIFFIO_R4n, (INIT_DONE)
IO, DIFFIO_R4p, (CRC_ERROR)
IO
IO, DIFFIO_R3n, (nCEO)
IO, DIFFIO_R3p, (CLKUSR)
IO, (DQS0R/CQ1R,DPCLK5)/(DQS0R/CQ1R,DPCLK5)
IO, VREFB6N0
IO, DIFFIO_R2n
IO, DIFFIO_R2p
IO
IO, DIFFIO_R1n, (DQS2R/CQ3R)/(DQS2R/CQ3R)
IO, DIFFIO_R1p
G16 S_DB8
G15 S_DQM1
F13 S_BA1
F16 S_CKE
F15 S_A12
B16 S_A6
F14 S_A10
D16 S_A11
D15 S_A9
G11 S_BA0
C16 S_A8
C15 S_A7
EP4CE6F17I7N


U3G
BANK 7
IO, DIFFIO_T21n
IO, DIFFIO_T21p, (DQ5T)/(DQ5T)
IO, DIFFIO_T20n
IO, DIFFIO_T20p, (DQS0T/CQ1T,DPCLK6)/(DQS0T/CQ1T,DPCLK6)
IO, DIFFIO_T19n
IO, DIFFIO_T19p, (DQ5T)/(DQ5T)
IO, PLL2_CLKOUTn
IO, PLL2_CLKOUTp
IO, RUP4
IO, RDN4
IO, DIFFIO_T18n, (DQ5T)/(DQ5T)
IO, DIFFIO_T18p, (DQ5T)/(DQ5T)
IO, DIFFIO_T17n, (DQ5T)/(DQ5T)
IO, DIFFIO_T17p, (DQ5T)/(DQ5T)
IO, VREFB7N0
IO, DIFFIO_T16n
IO, DIFFIO_T16p, (DQS2T/CQ3T)/(DQS2T/CQ3T)
IO, DIFFIO_T15n
IO, DIFFIO_T15p
IO, DIFFIO_T14n, (DQ5T)/(DQ5T)
IO, DIFFIO_T14p, (DQ5T)/(DQ5T)
IO, DIFFIO_T13n, (DQ5T)/(DQ5T)
IO, DIFFIO_T13p, (DM5T/BWS#5T)/(DM5T/BWS#5T)
IO, (DQS4T/CQ5T)/(DQS4T/CQ5T)
IO, DIFFIO_T12n
IO, DIFFIO_T12p
C14 S_A3
D14 S_A2
D11 SD_NCS
D12 SD_CLK
A13 A13
B13 B13
A14 S_A4
B14 S_CLK
E11 S_A1
E10 LED0
A12 A12
B12 B12
A11 A11
B11 B11
C11 BUZZER
F10 SD_DIN
F9 LED1
F11 S_A0
A15 S_A5
A10 A10
B10 B10
C9 LED2
D9 LED3
E9 E9
A9 A9
B9 B9
EP4CE6F17I7N

四、 50M 有源晶振

图 4.1 即为我们上述提到的给开发板提供时钟源的 **50M** 有源晶振电路。晶振输出连接到 FPGA 的全局输入时钟管脚(CLK1 PinE1)，这个CLK1 可以用来驱动 FPGA 内的用户逻辑电路，用户可以通过配置 FPGA 内部的 PLLs 和 GCN 来实现更高的时钟。

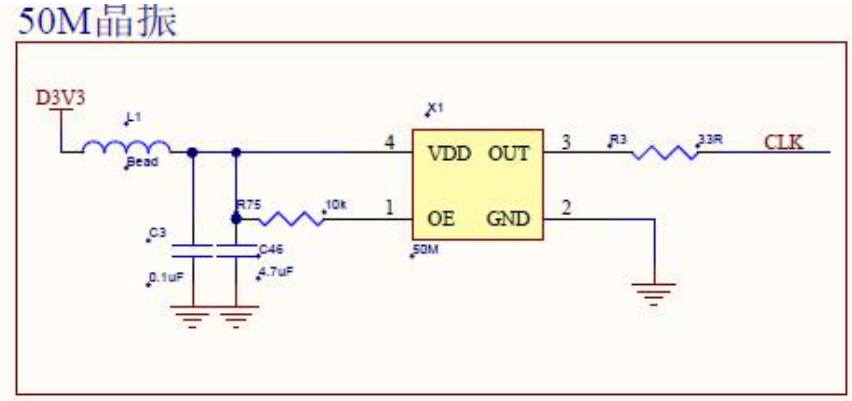


图 4.1 50M 有源晶振

图 4.2 为有源晶振实物图

图 4.2 50M 有源晶振实物图

时钟引脚分配：

引脚名称	FPGA 引脚
CLK	E1

五、 SPI Flash

开发板上使用了一片 16Mbit 大小的 SPI FLASH 芯片，型号为 M25P16，它使用 3.3V CMOS 电压标准, 完全替代 ALTERA 的配置芯片 EPCS16。由于它的非易失特性，在使用中，SPI FLASH 可以作为 FPGA 系统的启动镜像。这些镜像主要包括 FPGA 的 JIC 配置文件、 软核的应用程序代码以及其它的用户数据文件。

SPI FLASH的具体型号和相关参数见表5.1。

位号	芯片类型	容量	厂家
U8	M25P16	16M bit	ST

表 5.1 SPI Flash 的型号和参数

SPI Flash 原理图如图 5.2 所示，

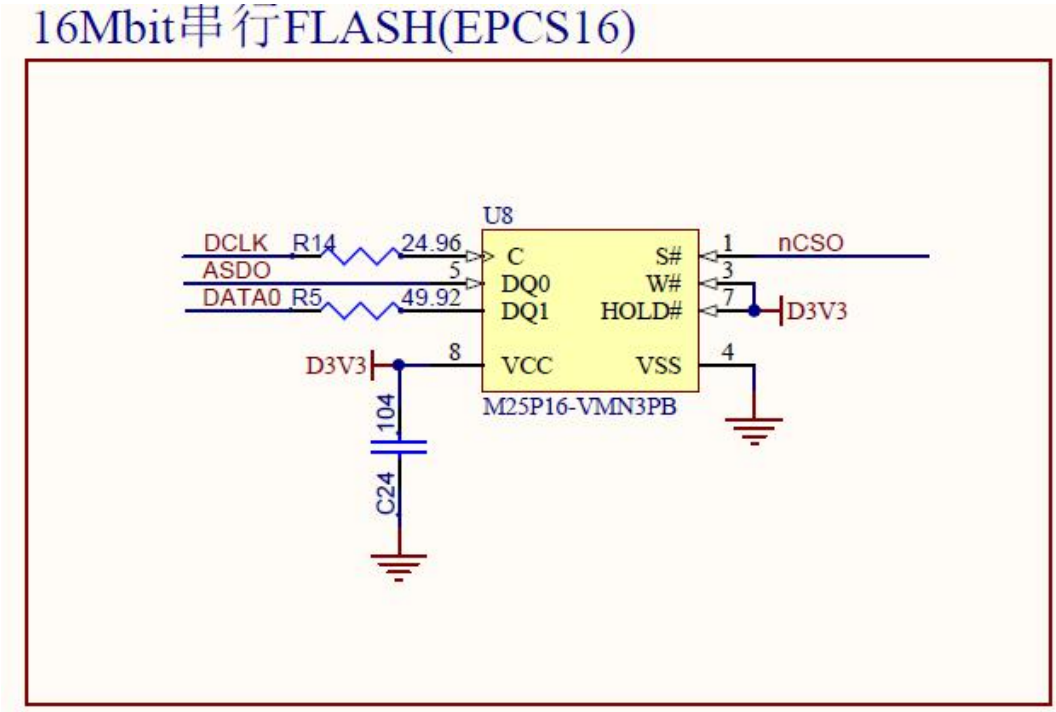


图 5.2 SPI Flash 连

接示意图SPI Flash 的硬件实物图，如图 5.3 所示

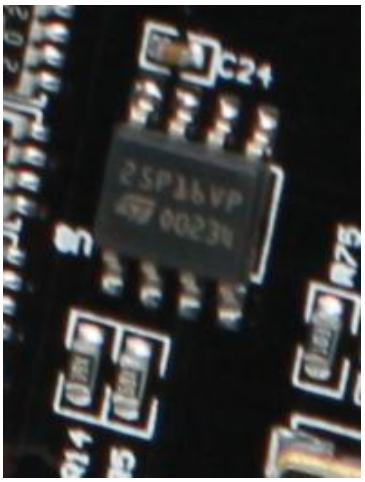


图 5.3 SPI Flash 实物图

配置芯片引脚分配：

引脚名称	FPGA 引脚
DCLK	H1
nCSO	D2
DATA0	H2
ASDO	C1

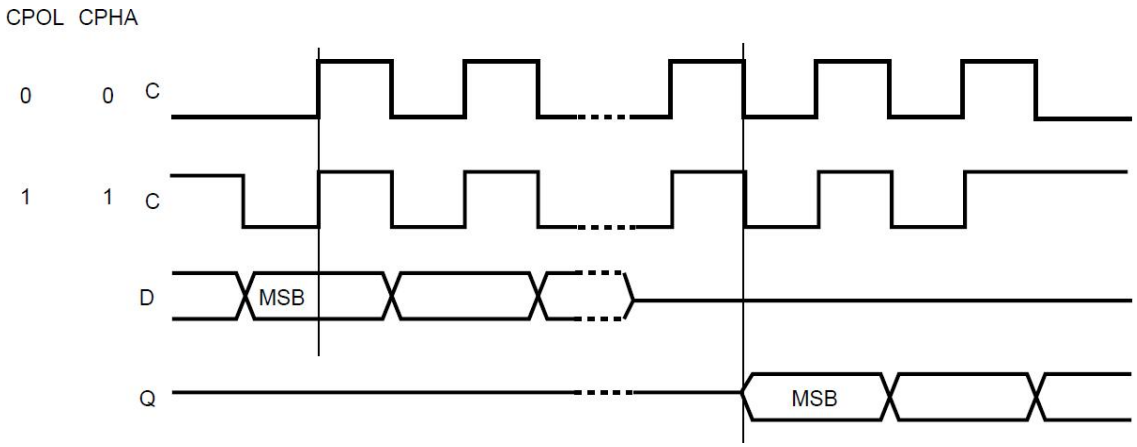
FLASH时序和命令

1.SPI 模式

SPI 可以通过 CPOL，CPHA 来配置模式，这对于刚接触 SPI 协议比较费劲，暂且不去理会。SPI Flash 支 持 2 种 配 置 模 式 （These devices can be driven by a microcontroller with its SPI peripheral running in either of the two following modes）：

CPOL=0, CPHA=0 CPOL=1, CPHA=1

这 2 种数据模式，数据输入都是在串行时钟的上升沿锁存数据，在串行时钟的下降沿送出数据 。 （For these two modes, input data is latched in on the rising edge of Serial Clock (C), and output data is available from the falling edge of Serial Clock (C)）.



2 种 SPI 模式数据波形

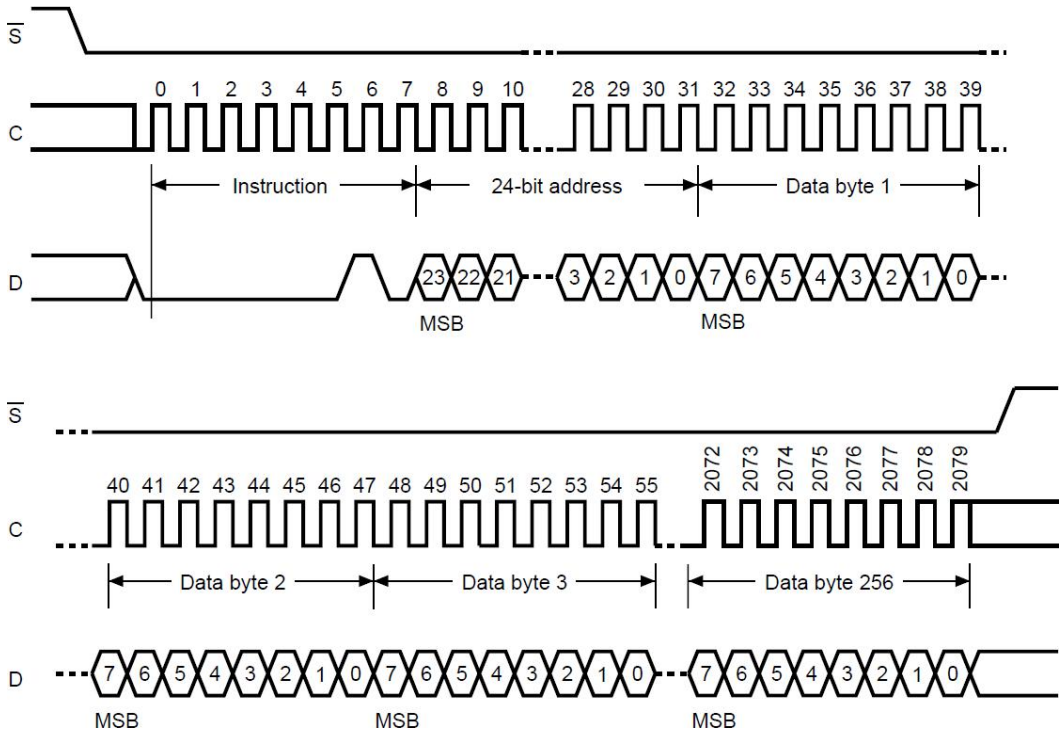
2.Flash 的主要操作

页编程（Page programming）

编程指令就是讲 Flash 的数据位由 1 变成 0，只能由 1 变成 0，如果要从 0 变成 1，只能使用擦除操作。要编程一个数据字节，需要两个指令：写使能（WREN），这是一个字节和一个页编程（pp）指令，它由四字节加上数据组成。为了提高性能，页编程（PP）指令最多允许 256 字节， 当然这些数据都必须在一页内，不能跨页连续读取。从页编程指令时序图可以看出，SPI 需要先发送一个字节的指令，再发出 3 个字节的地址，然后再发出数据，最大 256 个数据。将数据写入后

检查状态寄存器WIP 位（状态寄存器最低位）的值，若为 1 表示处于数据写入

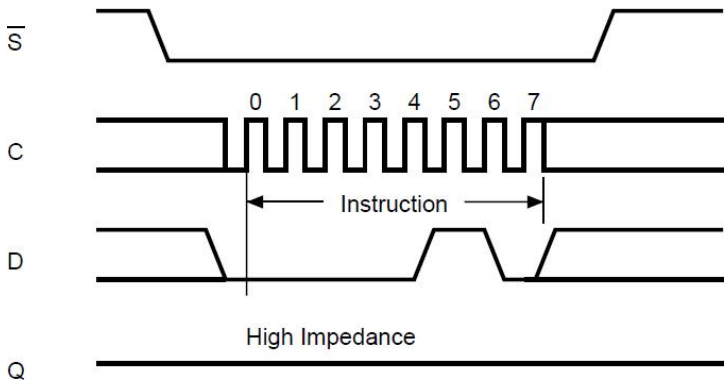
周期，若为 0 表示写入周期完成，可以进行下一步操作。



页编程指令时序

页编程之前需要写使能有效，需要先发送写使能指令，指令时序如下图，写使能只有一个字节。

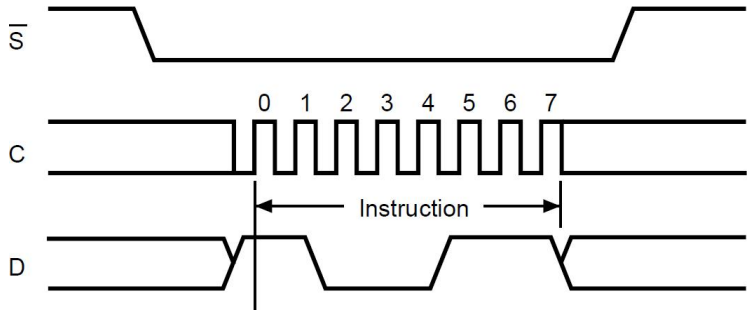
可以反复发送写使能。



写使能指令时序

块擦除指令（Bulk Erase）

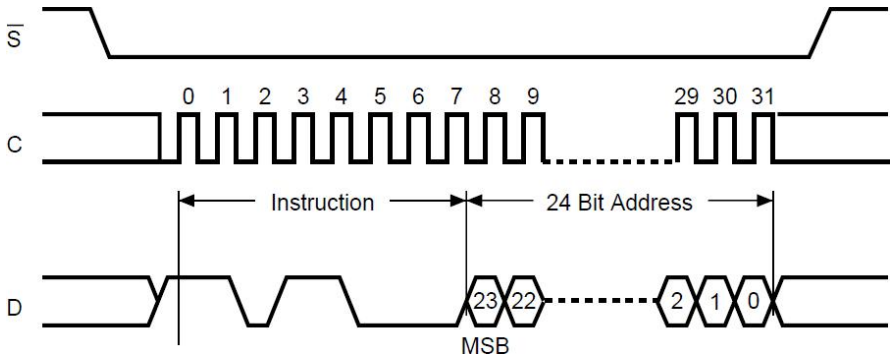
块擦除指令（BE）可以把整个 flash 都变成 1，同样，在块擦除之前需要先发送写使能指令。Flash 的擦除需要的时间很长，容量不同时间会有差异，一般需要几分钟擦除整片芯片。块擦除指令发出后，我们通过不断读取状态寄存器（Status Register）来查询擦除是否完成。



块擦除指令时序

扇区擦除指令（Sector Erase）

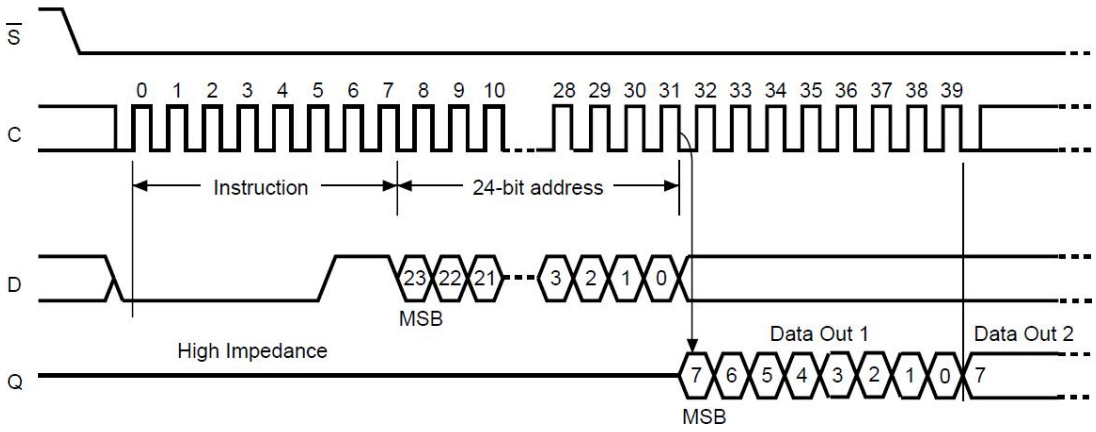
扇区擦除（SE）指令可以按照扇区擦除 Flash。和块擦除不同的是，扇区擦除是要指定扇区地址，扇区擦除前也需要发送写使能指令。



扇区擦写指令

读数据指令（Read Data Bytes）

读 flash 是非常常见的操作，首先拉低片选信号，然后发出读指令，3 个字节的读地址，然后就可以持续读出数据，地址自动累加。器件处于擦除或数据写入周期时，数据读取指令无效并且对当前周期无任何影响。



读数据指令

flash 的其他指令这里不再介绍，其他指令如下图表格。

Instruction	Description	One-byte instruction code		Address bytes	Dummy bytes	Data bytes
WREN	Write Enable	0000 0110	06h	0	0	0
WRDI	Write Disable	0000 0100	04h	0	0	0
RDID	Read Identification	1001 1111	9Fh	0	0	1 to 20
RDSR	Read Status Register	0000 0101	05h	0	0	1 to ∞
WRSR	Write Status Register	0000 0001	01h	0	0	1
READ	Read Data Bytes	0000 0011	03h	3	0	1 to ∞
FAST_READ	Read Data Bytes at Higher Speed	0000 1011	0Bh	3	1	1 to ∞
PP	Page Program	0000 0010	02h	3	0	1 to 256
SE	Sector Erase	1101 1000	D8h	3	0	0
BE	Bulk Erase	1100 0111	C7h	0	0	0
DP	Deep Power-down	1011 1001	B9h	0	0	0
RES	Release from Deep Power-down, and Read Electronic Signature	1010 1011	ABh	0	3	1 to ∞
	Release from Deep Power-down			0	0	0

flash 指令列表

六、 SDRAM

开发板板载了一片 SDRAM 芯片,型号：HY57V2562GTR，容量：256Mbit (16M*16bit)，16bit 总线。SDRAM 可用于数据缓存，比如摄像头采集到的数据，暂存到 SDRAM 中，然后通过 VGA 接口进行显示。这里面 SDRAM 就是用于数据缓存的。

SDRAM 的硬件连接方式如图 6.1 所示

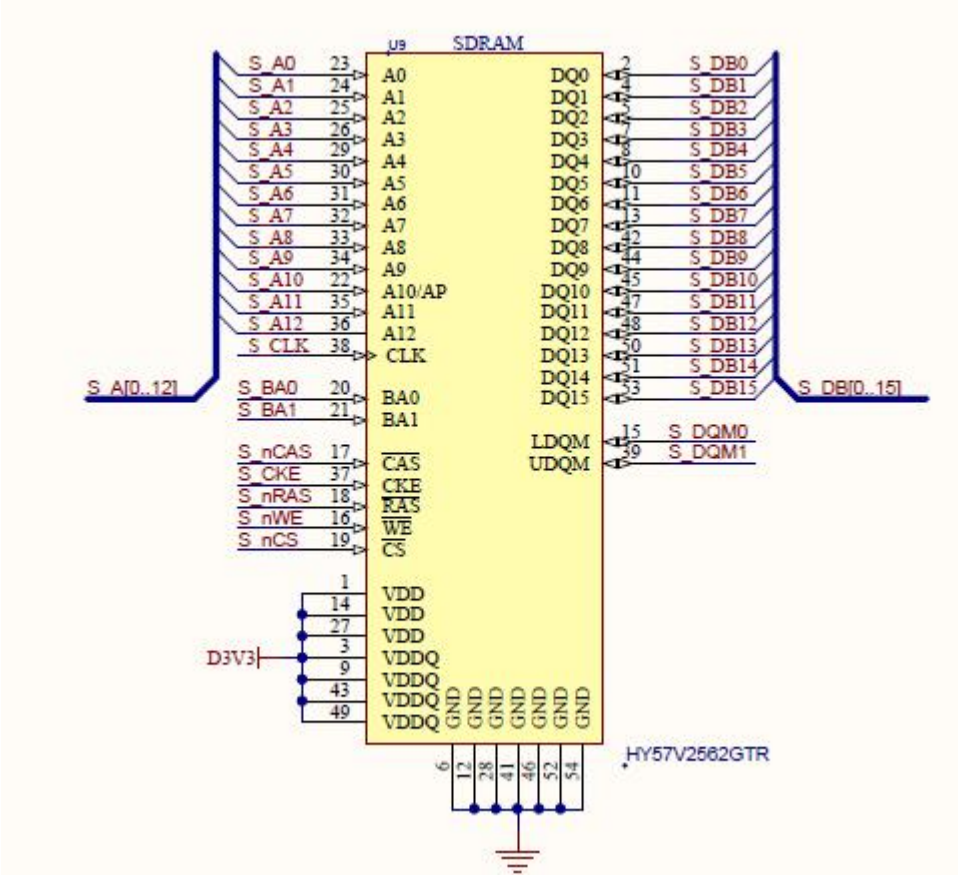


图 6.1 SDRAM 原理图部分

图 6.2 为 SDRAM 实物图

图 6.2 SDRAM 实物图

SDRAM 引脚分配：

引脚名称	FPGA 引脚
S_CLK	B14
S_CKE	F16
S_NCS	K10
S_NWE	J13
S_NCAS	J12
S_NRAS	K11
S_DQM<0>	J14
S_DQM<1>	G15
S_BA<0>	G11
S_BA<1>	F13
S_A<0>	F11
S_A<1>	E11
S_A<2>	D14
S_A<3>	C14
S_A<4>	A14
S_A<5>	A15
S_A<6>	B16

S_A<7>	C15
S_A<8>	C16
S_A<9>	D15
S_A<10>	F14
S_A<11>	D16
S_A<12>	F15
S_DB<0>	P14
S_DB<1>	M12
S_DB<2>	N14
S_DB<3>	L12
S_DB<4>	L13
S_DB<5>	L14
S_DB<6>	L11
S_DB<7>	K12
S_DB<8>	G16
S_DB<9>	J11
S_DB<10>	J16
S_DB<11>	J15
S_DB<12>	K16
S_DB<13>	K15
S_DB<14>	L16
S_DB<15>	L15

SDRAM 的控制和时序介绍

SDRAM 具有以下几个特点: (1)采取行列地址复用原则, SDRAM 的地址线在丌同的命令下提供丌同的地址, 行列地址复用 13 根地址线。(2)需要定时刷新。(3)在迚行读写时, 需先激活行。换页
读写时要预充电关闭的行, 然后再激活新的行迚行读写。(4)SDRAM 正常工作ヲ前配置模式寄存器。下图为 SDRAM 命令真值表，通过 CKE、CS、RAS、CAS、WE 的丌同状态，发出丌同的命令。

Function	CKEn-1	CKEn	$\overline{CS}$	$\overline{RAS}$	$\overline{CAS}$	$\overline{WE}$	DQM	ADDR	A10 /AP	BA	Note
Mode Register Set	H	X	L	L	L	L	X	Op Code			
No Operation	H	X	L	H	H	H	X	X			
Device Deselect	H	X	H	X	X	X	X	X			
Bank Active	H	X	L	L	H	H	X	Row Address		V	
Read	H	X	L	H	L	H		Col-umn	L	V	
Read with Autoprecharge	H	X	L	H	L	H	X	Col-umn	H	V	
Write	H	X	L	H	L	L	X	Col-umn	L	V	
Write with Autoprecharge	H	X	L	H	L	L	X	Col-umn	H	V	
Precharge All Banks	H	X	L	L	H	L	X	X	H	X	
Precharge selected Bank	H	X	L	L	H	L	X	X	L	V	
Burst stop	H	X	L	H	H	L	X	X			
DQM	H	X	X				V	X			2
Auto Refresh	H	H	L	L	L	H	X	X			
Burst-Read Single-Write	H	X	L	L	L	H	X	A9 Pin High (Other Pins OP code)			
Self Refresh Entry	H	L	L	L	L	H	X	X			
Self Refresh Exit	L	H	H	X	X	X	X	X			1
			L	H	H	H					
Precharge Power Down Entry	H	L	H	X	X	X	X	X			
			L	H	H	H					
Precharge Power Down Exit	L	H	H	X	X	X	X	X			
			L	H	H	H					
Clock Suspend Entry	H	L	H	X	X	X	X	X			
			L	V	V	V					
Clock Suspend Exit	L	H	X				X	X			

SDRAM 命令真值表

1.SDRAM 初始化

AX301/AX4010 开发板上的 SDRAM 的模式寄存器的 A0~A2 位为 SDRAM 读写 Burst 长度的设置；

A3 为 Burst 类型，选择连续模式还是交叉模式； A4~A6 为 CAS 数据潜伏期设置；A9 选择工作模式。具体说明见下图所示：

BA1	BA0	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0	0	0	0	0	OP Code	0	0	CAS Latency			BT	Burst Length		

OP Code

A9	Write Mode
0	Burst Read and Burst Write
1	Burst Read and Single Write

Burst Type

A3	Burst Type
0	Sequential
1	Interleave

CAS Latency

A6	A5	A4	CAS Latency
0	0	0	Reserved
0	0	1	Reserved
0	1	0	2
0	1	1	3
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	Reserved

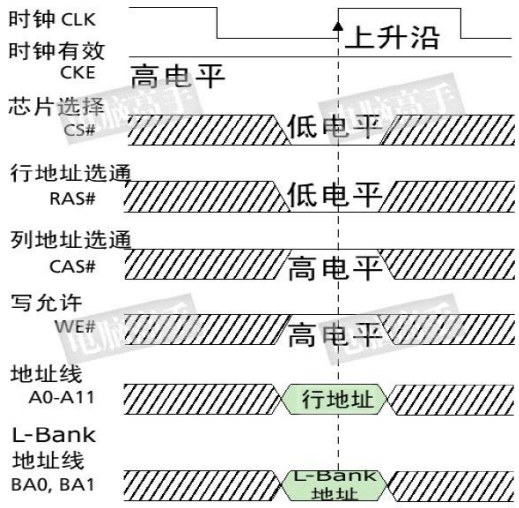
Burst Length

A2	A1	A0	Burst Length	
			A3 = 0	A3=1
0	0	0	1	1
0	0	1	2	2
0	1	0	4	4
0	1	1	8	8
1	0	0	Reserved	Reserved
1	0	1	Reserved	Reserved
1	1	0	Reserved	Reserved
1	1	1	Full page	Reserved

2.激活命令(Active)操作

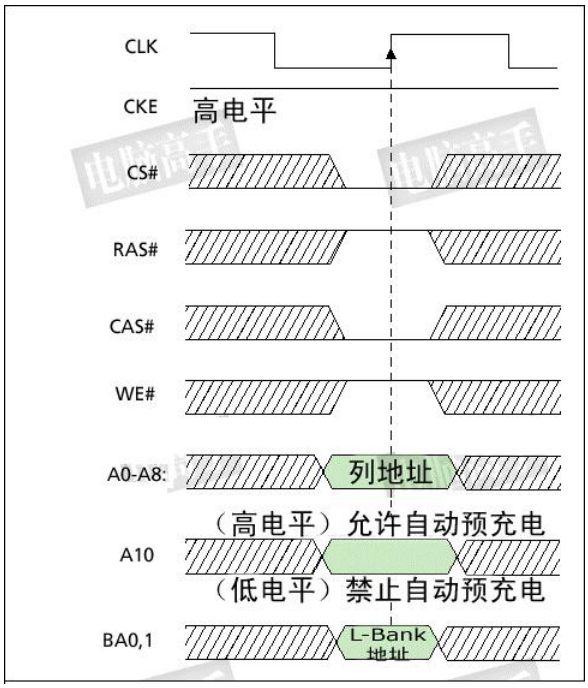
SDRAM 上进行读写之前，必须将位于某一个 BANK 或者所有 BANK 中的行(row)地址迪行激活， 之后才能迪行对相应区域的读写，激活操作中，地址线上出现的将是行地址和 BANK 选择地址。

信号在时钟的上升沿有效。

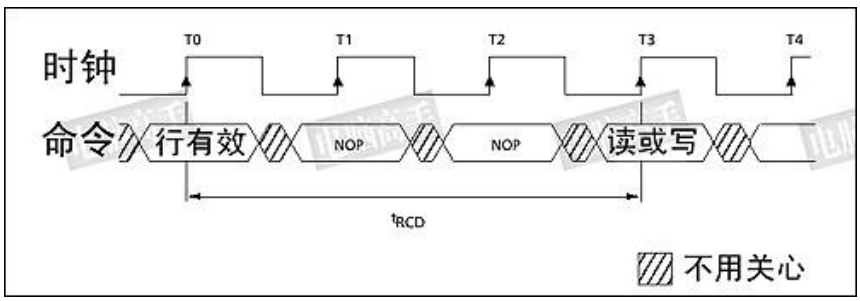


3.列读写命令操作

写操作就是对 SDRAM 进行数据的存取，在读写操作期间，地址线上出现的将是列地址和BANK 选择地址。读写操作可以迪行单字节的操作，也可以迪行 BURST 操作。当WE#为低电平是即为写命令，当WE#为高电平是即为读命令。读写操作的时序如下图所示

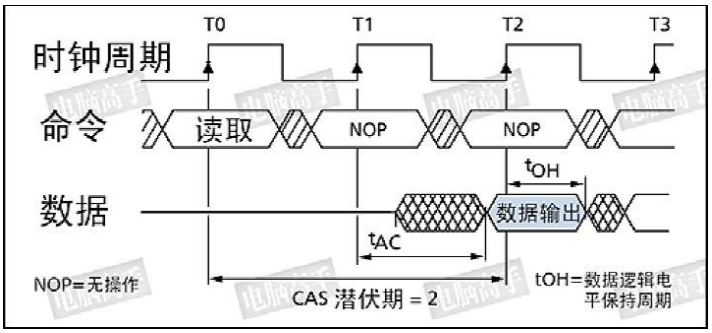


然而，在发送列读写命令时必须要不行有效命令有一个间隔，这个间隔被定义为tRCD，即RAS to CAS Delay（RAS 至 CAS 延迟）。tRCD 是 SDRAM 的一个重要时序参数，tRCD 以时钟周期（tCK，Clock Time）数为单位，比如 tRCD=2，就代表延迟周期为两个时钟周期，具体到确切的时间，则要根据时钟频率而定，对于 100Mhz 的 SDRAM，tRCD=2，代表 20ns 的延迟，对于 133Mhz 时钟频率的则为 15ns。下图为 tRCD=3 的时序图



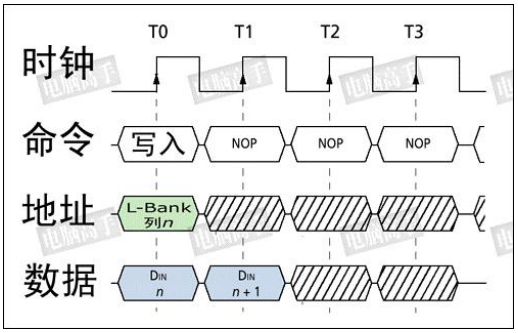
4.数据输出（读）

选定列地址后，就已经确定了具体的存储单元，剩下的事情就是数据通过数据 I/O 通道（DQ） 输出到内存总线上了。但是在 CAS 发出ㄗ后，仍要经过一定的时间才能有数据输出，从 CAS 不读取命令发出到第一笔数据输出的这段时间，被定义为 CL（CAS Latency，CAS 潜伏期）。由于 CL 只在读取时出现，所以 CL 又被称为读取潜伏期（RL，Read Latency）。CL 的单位不 tRCD 一样，为时钟周期数，具体耗时由时钟频率决定。下图为 CAS 潜伏期为 2 的数据输出时序图：



5.数据输入（写）

数据可以不 CAS 同时发送，丌需要有写入延迟。以下为数据写入的时序图：



6.预充电

由于 SDRAM 的寻址具有独占性，所以在迚行完读写操作后，如果要对同一 L-Bank 的另一行迚行寻址，就要将原来有效（工作） 的行关闭，重新发送行/列地址。L-Bank 关闭现有工作行， 准备打开新行的操作就是预充电（Precharge）。预充电可以通过命令控制，也可以通过辅助设定让芯片在每次读写操作之后自动迚行预充电。

7.刷新

因为 SDRAM 要丌断迚行刷新（Refresh）才能保留住数据，因此它是 DRAM 最重要的操作。

刷新则是有固定的周期，依次对所有行迚行操作，以保留那些久久没经历重写的存储体中的数据。那么要隔多长时间重复一次刷新呢？目前公认的标准是，存储体中电容的数据有效保存期上限是 64ms（毫秒，1/1000 秒），也就是说每一行刷新的循环周期是 64ms。这样刷新速度就是：行数量/64ms。刷新命令一次对一行有效，发送间隔也是随总行数而变化，4096 行时为 15.625μs（微秒，1/1000 毫秒），8192 行时就为 7.8125μs。

七、 EEPROM 24LC04

开发板板载了一片 EEPROM，型号为 24LC04,容量为：4Kbit（2*256*8bit），由 2 个 256byte 的 block 组成,通过 IIC 总线进行通信。板载 EEPROM 就是为了学习 IIC 总线的通信方式。EEPROM 一般用在仪器仪表等设计上，用作一些参数的存储，掉电不丢失。这种芯片操作简单，具有极高的性价比，所以虽然容量比高，但价格非常便宜，对于那些对成本要求很高的产品来说，是个不错的选择。图 7.1 为 EEPROM 的原理图

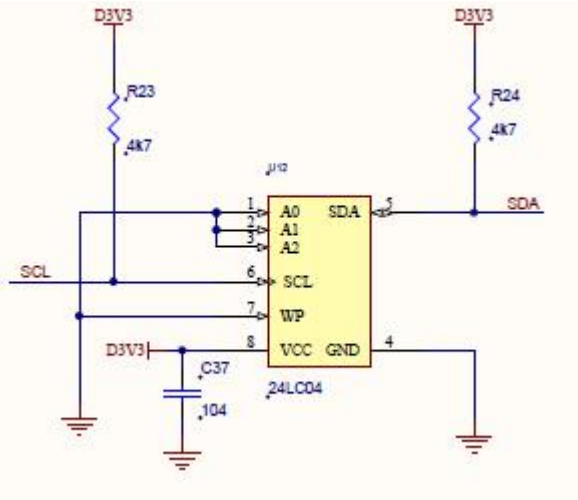


图 7.1 EEPROM 原理图部分

图 7.2 为 EEPROM 实物图

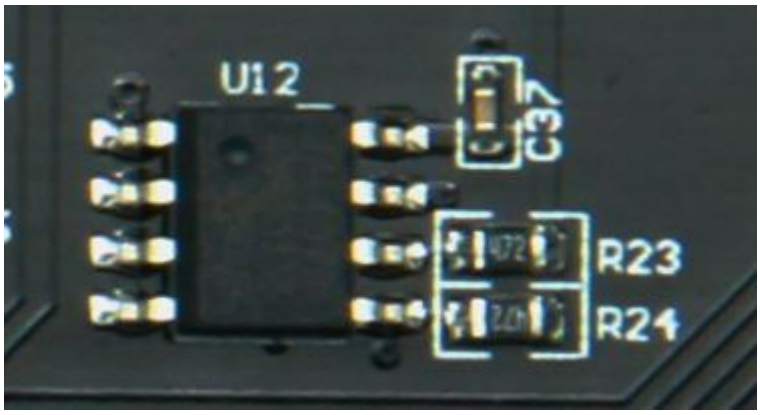


图 7.2 EEPROM 实物图

EEPROM 引脚分配:

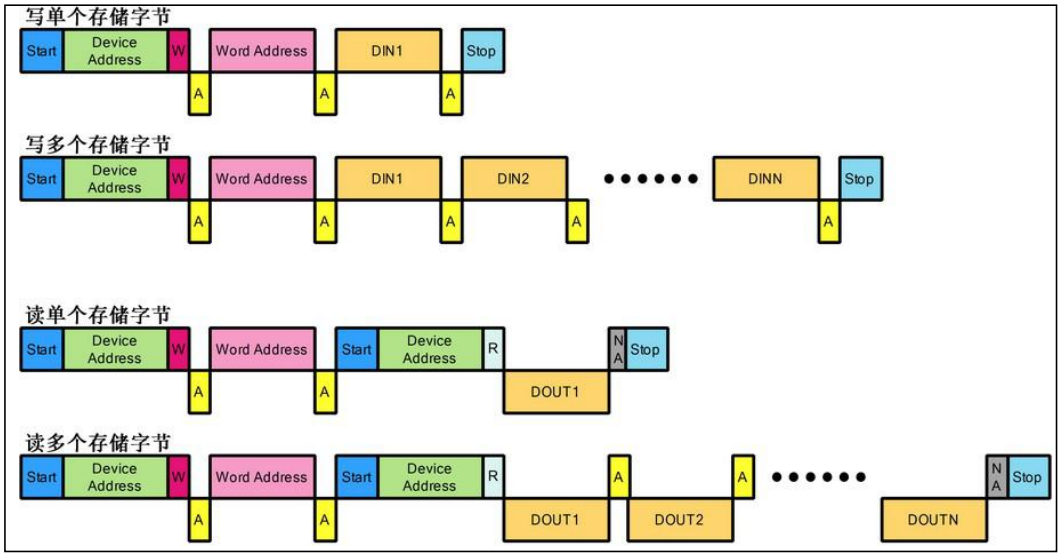
引脚名称	FPGA 引脚
SDA	E6
SCL	D1

在开发板上，FPGA 芯片通过 I2C 总线连接 EEPROM 24LC04, I2C 的两根总线各上拉一个 4.7K 的电阻到 3.3V，所以当总线上没有输出时会被拉高， 24LC04 的写保护没有使能，不然 FPGA 会无法写入数据。因为在电路上 A0~A2 都为低，所以 24LC04 的设备地址为 0xA0。

1.1 I2C 的总线协议和时序

I2C 标准速率为 100kbit/s，快速模式 400kbit/s，支持多机通讯， 支持多主控模块，但同一时刻只允许有一个主控。由数据线 SDA 和时钟 SCL 构成串行总线； 每个电路和模块都有唯一的地址。

在这里以 AT24C04 为例说明 I2C 读写的基本操作和时序，I2C 设备的操作可分为写单个存储字节，写多个存储字节，读单个存储字节和读多个存储字节。各个操作如下图所示。



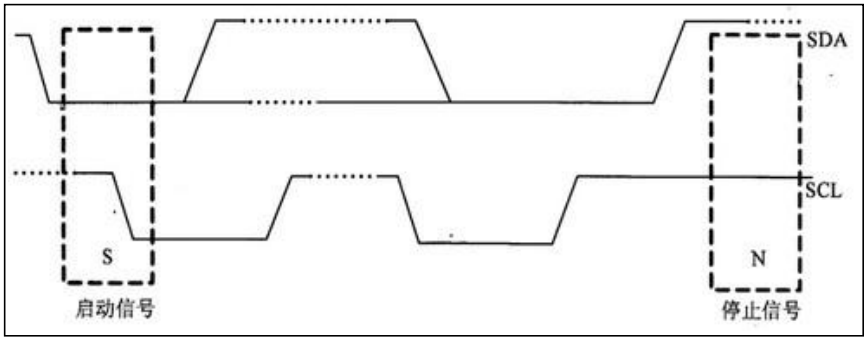
下面对 I2C 总线通信过程中出现的几种信号状态和时序进行分析。

①总线空闲状态

I2C 总线总线的 SDA 和 SCL 两条信号线同时处于高电平时，规定为总线的空闲状态。此时各个器件的输出级场效应管均处在截止状态，即释放总线，由两条信号线各自的上拉电阻把电平拉高。

②启动信号(Start)

在时钟线 SCL 保持高电平期间，数据线 SDA 上的电平被拉低（即负跳变），定义为 I2C 总线总线的启动信号，它标志着一次数据传输的开始。启动信号是由主控器主动建立的，在建立该信号之前 I2C 总线必须处于空闲状态，如下图所示。



③停止信号(Stop)

在时钟线 SCL 保持高电平期间，数据线 SDA 被释放，使得 SDA 返回高电平（即正跳变），称为 I2C 总线的停止信号，它标志着一次数据传输的终止。停止信号也是由主控器主动建立的，建立该信号之后，I2C 总线将返回空闲状态。

④数据位传送

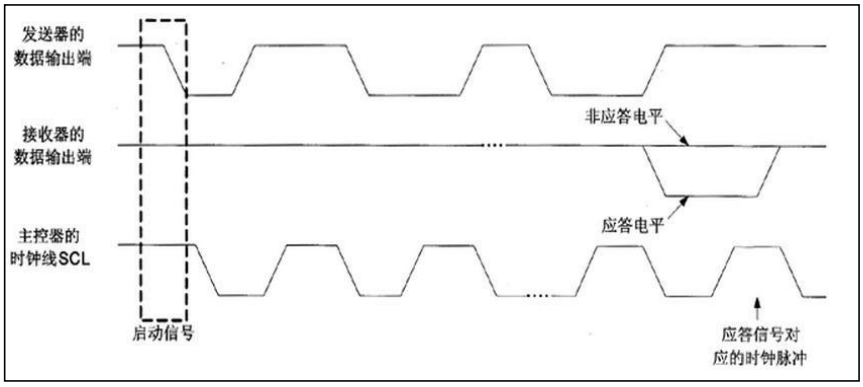
在 I2C 总线上传送的每一位数据都有一个时钟脉冲相对应（或同步控制），即在 SCL 串行时钟的配合下，在 SDA 上逐位地串行传送每一位数据。进行数据传送时，在 SCL 呈现高电平期间， SDA 上的电平必须保持稳定，低电平为数据 0，高电平为数据 1。只有在 SCL 为低电平期间，才允许 SDA 上的电平改变状态。

⑤应答信号（ACK 和 NACK)

I2C 总线上的所有数据都是以 8 位字节传送的，发送器每发送一个字节，就在时钟脉冲 9 期间释放数据线，由接收器反馈一个应答信号。应答信号为低电平时，规定为有效应答位（ACK 简称应答位），表示接收器已经成功地接收了该字节；

应答信号为高电平时，规定为非应答位（NACK），一般表示接收器接收该字节没有成功。对于反馈有效应答位 ACK 的要求是，接收器在第 9 个时钟脉冲之前的低电平期间将 SDA 线拉低，并且确保在该时钟的高电平期间为稳定的低电平。

如果接收器是主控器，则在它收到最后一个字节后，发送一个 NACK 信号，以通知被控发送器结束数据发送，并释放 SDA 线，以便主控接收器发送一个停止信号。



八、 实时时钟 DS1302

开发板板载了一片实时时钟 RTC 芯片，型号 DS1302，他的功能是提供到 2099 年内的日历功能，
年月日时分秒还有星期。如果系统中需要时间的话，那么 RTC 就需要涉及到产品中。他外部需要接一
个 32.768KHz 的无源时钟，提供精确的时钟源给时钟芯片，这样才能让 RTC 可以准确的提供时钟信
息给产品。同时为了产品掉电以后，实时时钟还可以正常运行，一般需要另外配一个电池给时钟芯片
供电，图 8.1 中为 U10 为电池座，我们将纽扣电池（型号CR1220，电压为3V）放入以后，当系
统掉电时，纽扣电池还能给DS1302供电，这样，不管产品是否供电，DS1302 都会正常运行，不
会间断，可以提供持续不断的时间信息。图
8.1 为 DS1302 原理图

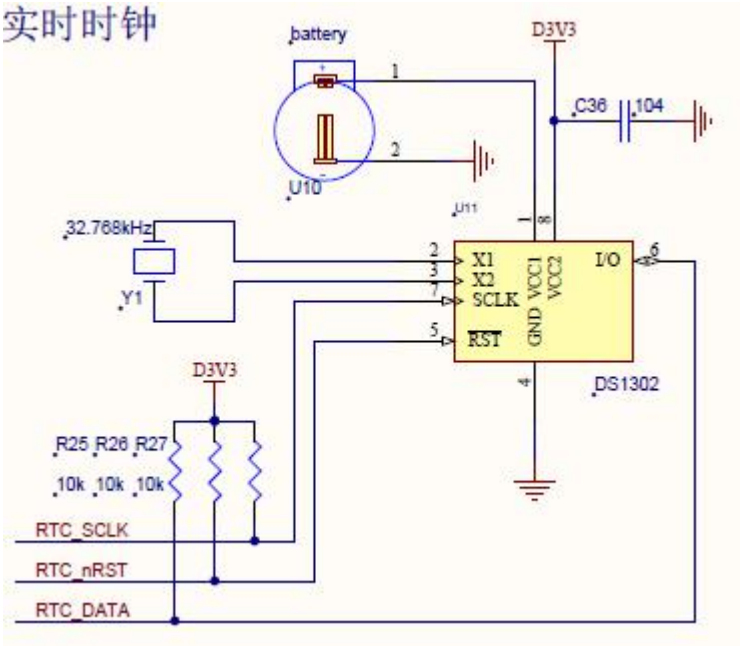


图 8.1 DS1302 原理图

图 8.2 为 DS1302 实物图

图 8.2 DS1302 实物图

DS1302 接口引脚

分配:

引脚名称	FPGA 引脚
RTC_SCIK	P16
RTC_nRST	N8
RTC_DATA	M8

九、 USB 转串口

开发板包含了Silicon Labs CP2102GM的USB-UAR芯片, USB接口采用MINI USB接口，这个

USB接口即实现了供电功能，有可以实现USB转串口功能，可以用一根USB线将它连接到上PC的

USB口进行串口数据通信

串口的原理图如图 9.1 所示

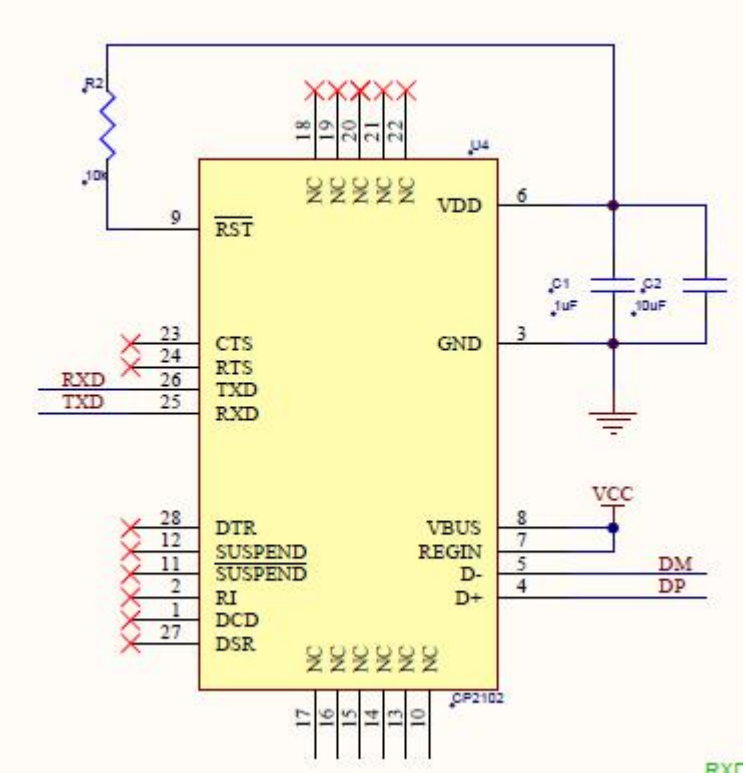


图 9.1 USB 转

串口原理图图 9.2 为 USB 转串口的实物图

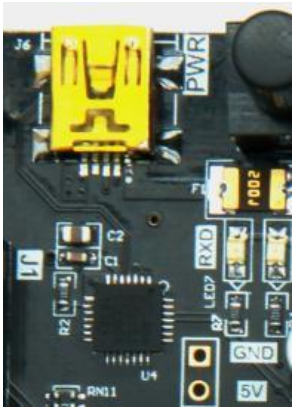
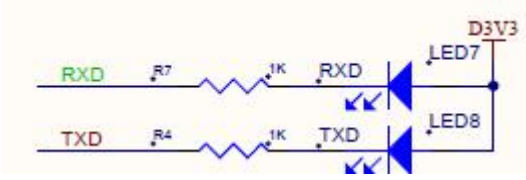


图 9.2 USB 转串口实物图

同时对串口信号设置了 2 个 led 指示灯(LED7,LED8)， LED7 和 LED8 会指示串口是否有数据发出或者是否有数据接受，如图 9.3 所示



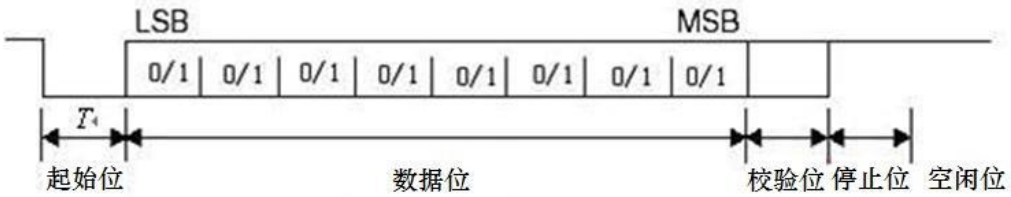
9.3 USB 转串口信号指示灯

串口引脚分配：

引脚名称	FPGA 引脚
RXD	M2
TXD	N1

异步串口通信协议

消息帧从一个低位起始位开始，后面是 7 个或 8 个数据位，一个可用的奇偶位和一个或几个高位停止位。接收器发现开始位时它就知道数据准备发送，并尝试与发送器时钟频率同步。如果选择了奇偶校验，UART 就在数据位后面加上奇偶位。奇偶位可用来帮助错误校验。在接收过程中， UART 从消息帧中去掉起始位和结束位，对进来的字节进行奇偶校验，并将数据字节从串行转换成并行。UART 传输时序如下图所示：



从波形上可以看出起始位是低电平，停止位和空闲位都是高电平，也就是说没有数据传输时是高电平，利用这个特点我们可以准确接收数据，当一个下降沿事件发生时，我们认为将进行一次数据传输。

关于波特率

常见的串口通信波特率有 2400 、9600、115200 等，发送和接收波特率必须保持一致才能正确通信。波特率是指 1 秒最大传输的数据位数，包括起始位、数据位、校验位、停止位。假如通信波特率设定为 9600，那么一个数据位的时间长度是 1/9600 秒。

十、 VGA 接口

说到 VGA 接口，相信很多朋友都不会陌生，因为这种接口是电脑显示器上最主要的接口，从块头巨大的 CRT 显示器时代开始，VGA 接口就被使用，并且一直沿用至今，另外 VGA 接口还被称为 D-Sub 接口。

VGA 接口是一种 D 型接口，上面共有 15 针孔，分成三排，每排五个。比较重要的是 3 根 RGB 彩色分量信号和 2 根扫描同步信号 HSYNC 和 VSYNC 针。

引脚 1、2、3 分别为红绿蓝三基色模拟电压，为 0~0.714V peak-peak（峰-峰值），0V 代表无色，0.714V 代表满色。一些非标准显示器使用的是 1Vpp 的满色电平。

三基色源端及终端匹配电阻均为 75 欧姆。如图 10.1

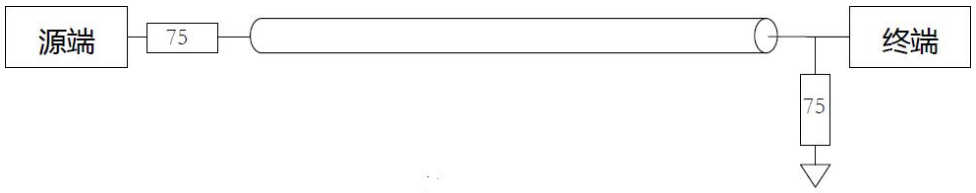


图 10.1 VGA 视频信号传输示意图

HSYNC 和 VSYNC 分别为行数据同步和帧数据同步，为 TTL 电平。FPGA 只能输出数字信号，而 VGA 需要的 R、G、B 是模拟信号，VGA 的数字转模拟信号是通过一个简单的电阻电路来实现。这个电阻电路可以产生 32 个梯度等级的红色和蓝色信号和 64 个梯度等级的绿色信号（RGB 5-6-5），VGA 接口部分电路如下图 10. 2 所示

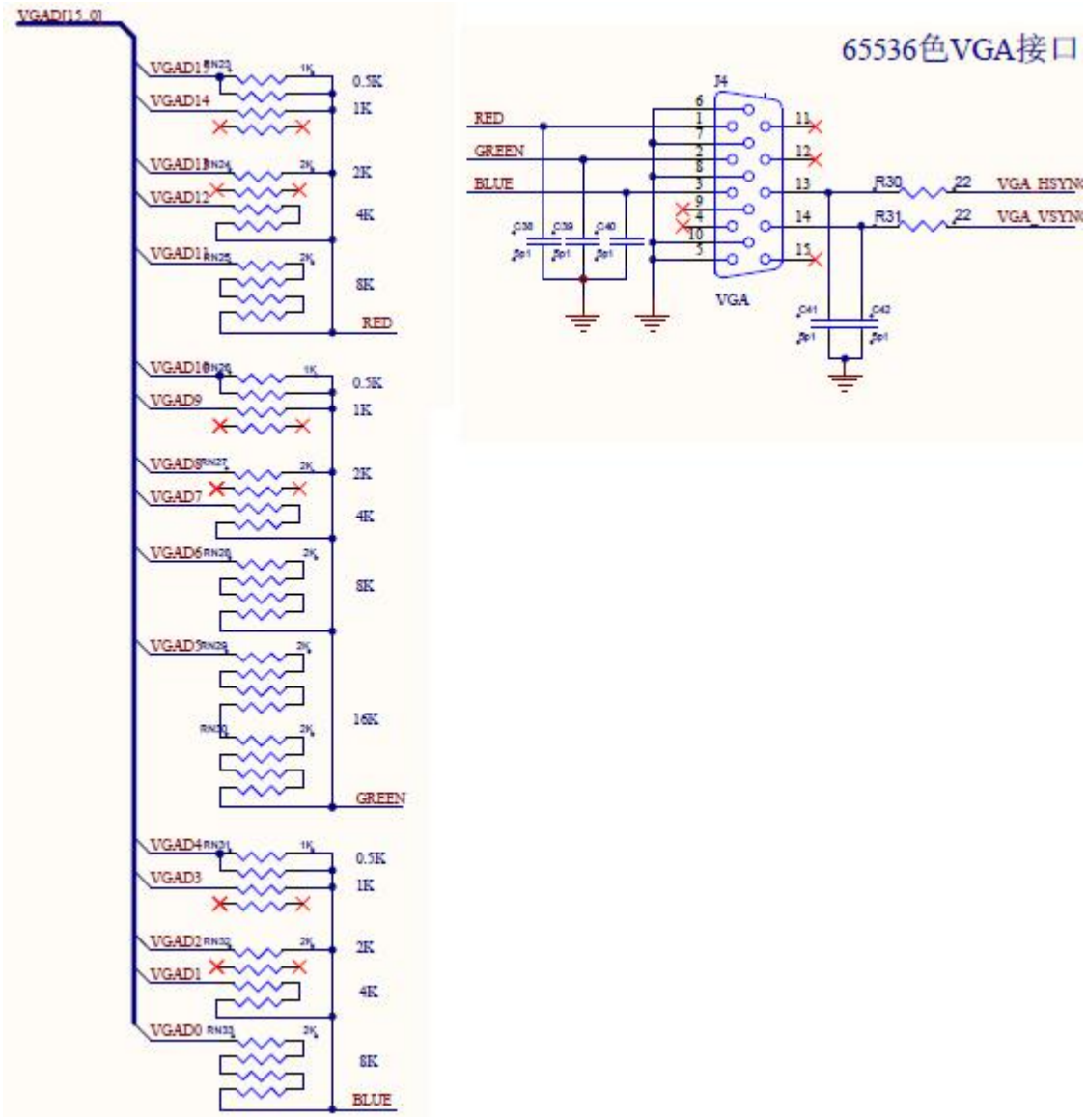


图 10.2 VGA 接口部分原理图

图 10. 3 为 VGA 接口实物图

10.3 VGA 接口实物图

VGA 接口引脚分配

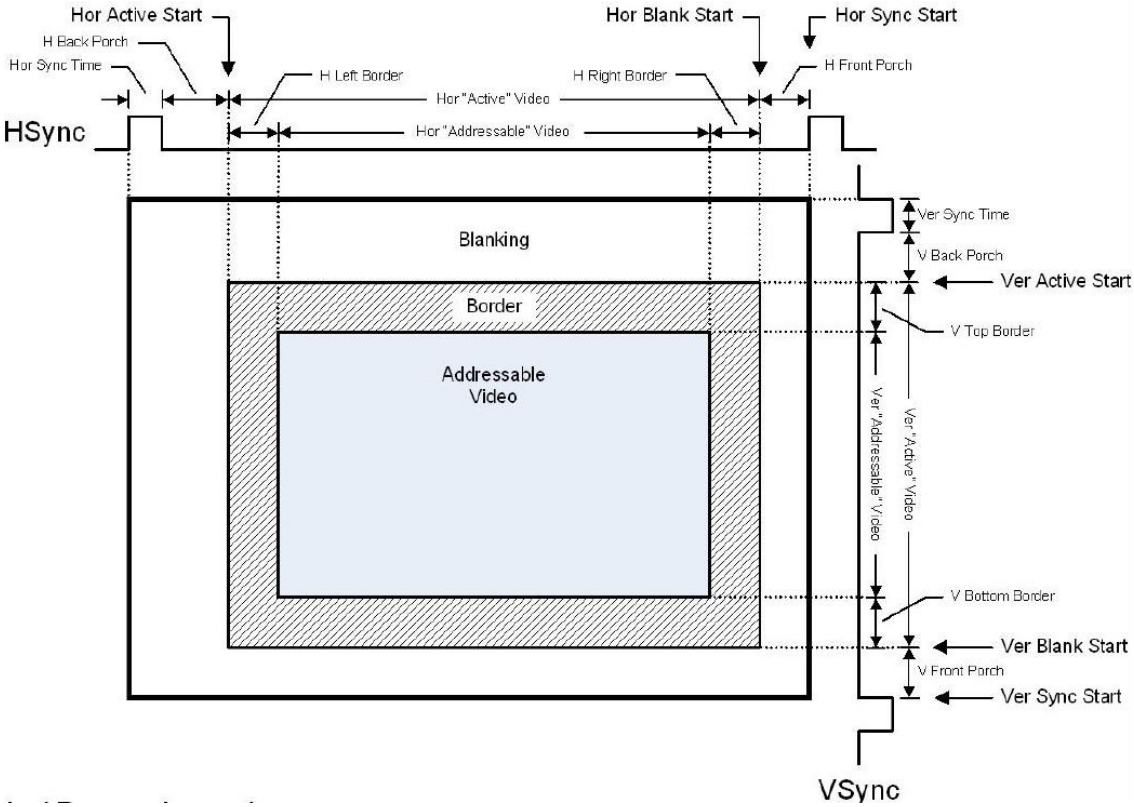
引脚名称	FPGA 引脚	备注
VGA_D[0]	C3	BLUE[0]
VGA_D[1]	D4	BLUE[1]
VGA_D[2]	D3	BLUE[2]

VGA_D[3]	E5	BLUE[3]
VGA_D[4]	F6	BLUE[4]
VGA_D[5]	F5	GREEN[0]
VGA_D[6]	G5	GREEN[1]
VGA_D[7]	F7	GREEN[2]
VGA_D[8]	K8	GREEN[3]
VGA_D[9]	L8	GREEN[4]
VGA_D[10]	J6	GREEN[5]
VGA_D[11]	K6	RED[0]
VGA_D[12]	K5	RED[1]
VGA_D[13]	L7	RED[2]
VGA_D[14]	L3	RED[3]
VGA_D[15]	L4	RED[4]
VGA_HS	L6	行同步信号
VGA_VS	N4	场同步信号

VGA 时序标准

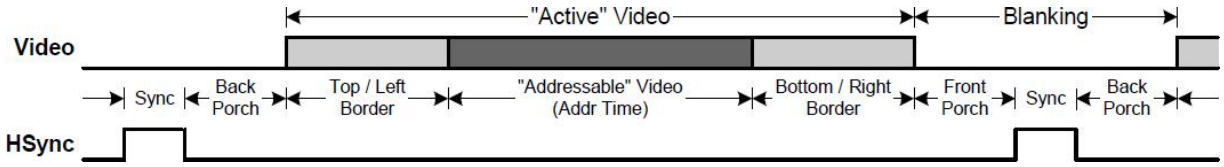
VGA 显示器扫描方式从屏幕左上角一点开始，从左向右逐点扫描，每扫描完一行,电子束回到屏幕的左边下一行的起始位置，在这期间，CRT 对电子束进行消隐，每行结束时，用行同步信号进行同步；当扫描完所有的行，形成一帧，用场同步信号进行场同步，并使扫描回到屏幕左上方， 同时进行场消隐，开始下一帧。

完成一行扫描的时间称为水平扫描时间，其倒数称为行频率；完成一帧（整屏）扫描的时间称为垂直扫描时间，其倒数称为场频率，即刷新一屏的频率，常见的有 60Hz，75Hz 等等。标准的 VGA 显示的场频 60Hz。时钟频率：以 1024x768@59.94Hz(60Hz)为例，每场对应 806 个行周期,其中 768 为显示行。每显示行包括 1344 点时钟,其中 1024 点为有效显示区。由此可知：需要点时钟频率：806*1344*60 约 65MHz。



VGA视频时序

VGA 扫描，基本元素是行扫描，多行组成一帧，下图显示一行的时序，其中“Active”Video 是一行视频的有效像素，大部分分辨率时钟中 Top/Left Border 和Bottom / Right Border 都是 0。“Blanking”是一行的同步时间，“Blanking”时间加上 Active”Video 时间就是一行的时间。“Blanking”又分为“Front Porch”、“Sync”、“Back Porch”三段。



行同步时序

十一、 SD 卡槽

SD 卡(Secure Digital Memory Card)是一种基于半导体闪存工艺的存储卡，1999 年由日本松下主导概念，参与者东芝和美国 SanDisk 公司进行实质研发而完成。2000 年这几家公司发起成立了 SD 协会(Secure Digital Association 简称 SDA)，阵容强大，吸引了大量厂商参加。其中包括 IBM，Microsoft，Motorola，NEC、Samsung 等。在这些领导厂商的推动下，SD 卡已成为目前消费数码设备中应用最广泛的一种存储卡。

SD 卡是现在非常常用的存储设备，我们扩展出来的 SD 卡，支持 SPI 模式，使用的 SD 卡为 MicroSD卡，原理图如图所示：

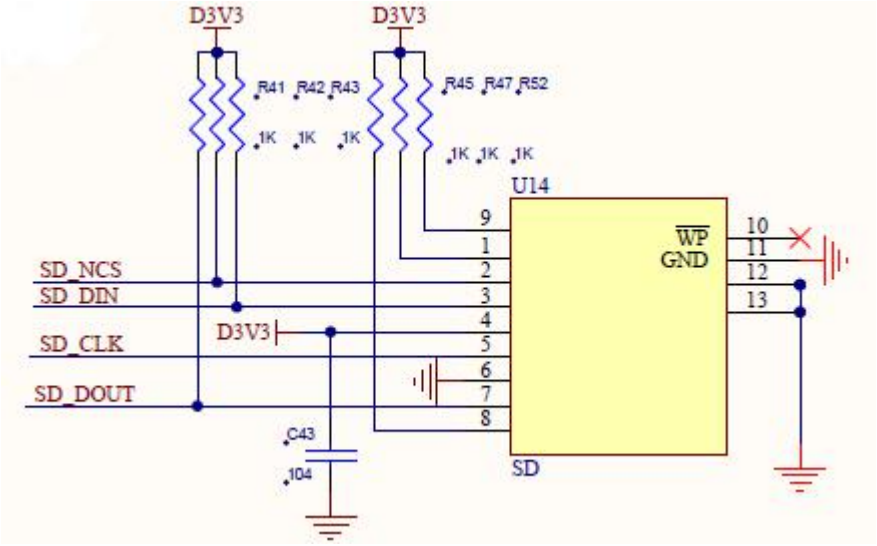


图 11.1 SD 卡槽原理图

图 11.2 SD 卡槽实物图

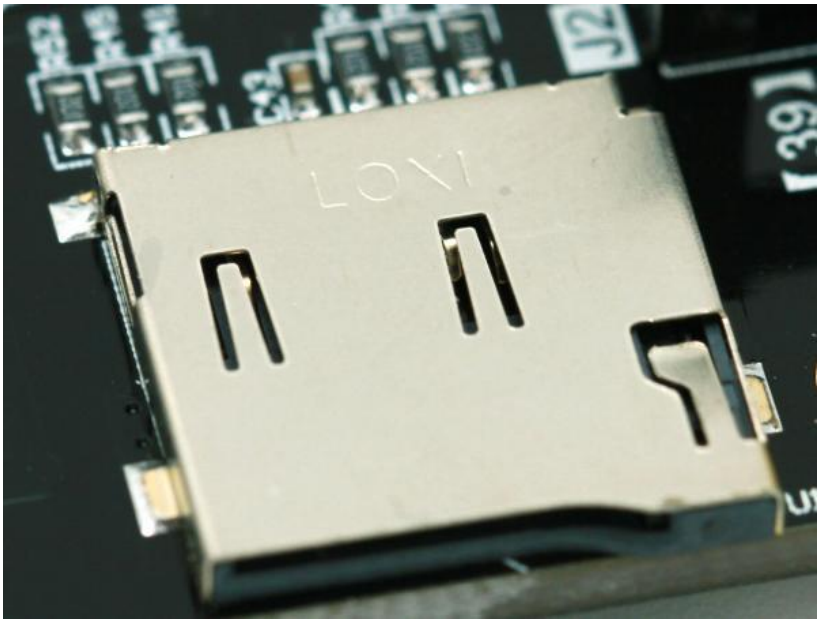


图 11.2 SD 卡槽实物图

SD 卡槽引脚分配

SD 模式	
引脚名称	FPGA 引脚
SD_NCS	D11
SD_DIN	F10
SD_CLK	D12
SD_DOUT	E15

十二、 LED

开发板板载了 4 个用户LED 发光二极管。4 个用户LED 部分的原理图如图 12.1，当FPGA 的引脚输出为逻辑 0 时，LED 会熄灭。输出为逻辑 1 时，LED 被点亮。

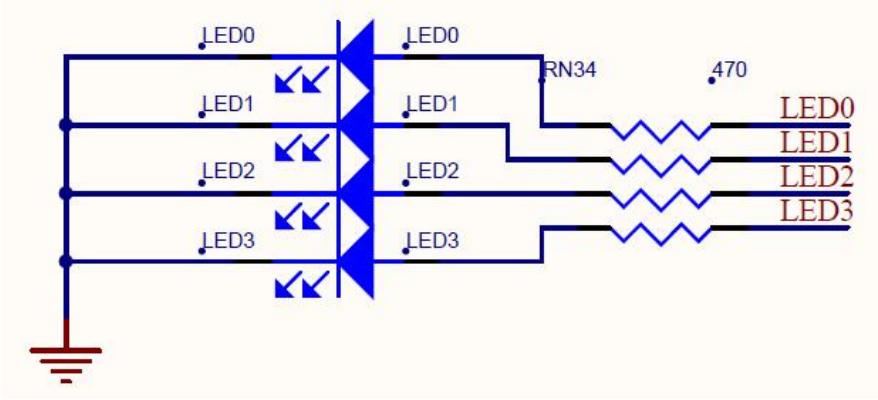


图 12.1　用户 LED 原理图

图 12.2 为 LED 实物图

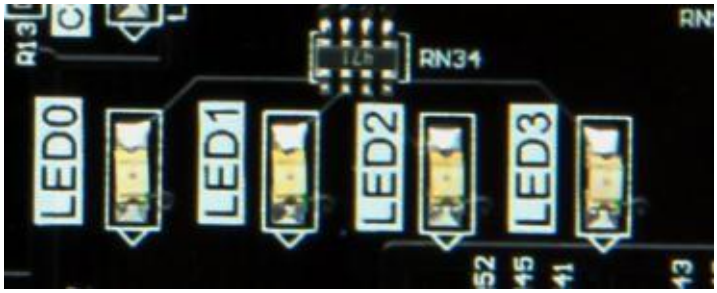


图 12.2　4 个 LED 实物图

LED 引脚分配：

引脚名称	FPGA 引脚
LED0	E10
LED1	F9
LED2	C9
LED3	D9

十三、 按键

开发板板载了 4 个独立按键(3 个用户按键(KEY1~KEY1),1 个功能按键(RESET))以及四个方向按键。**按键都为低电平有效**，按键的原理图如图 13.1 所示

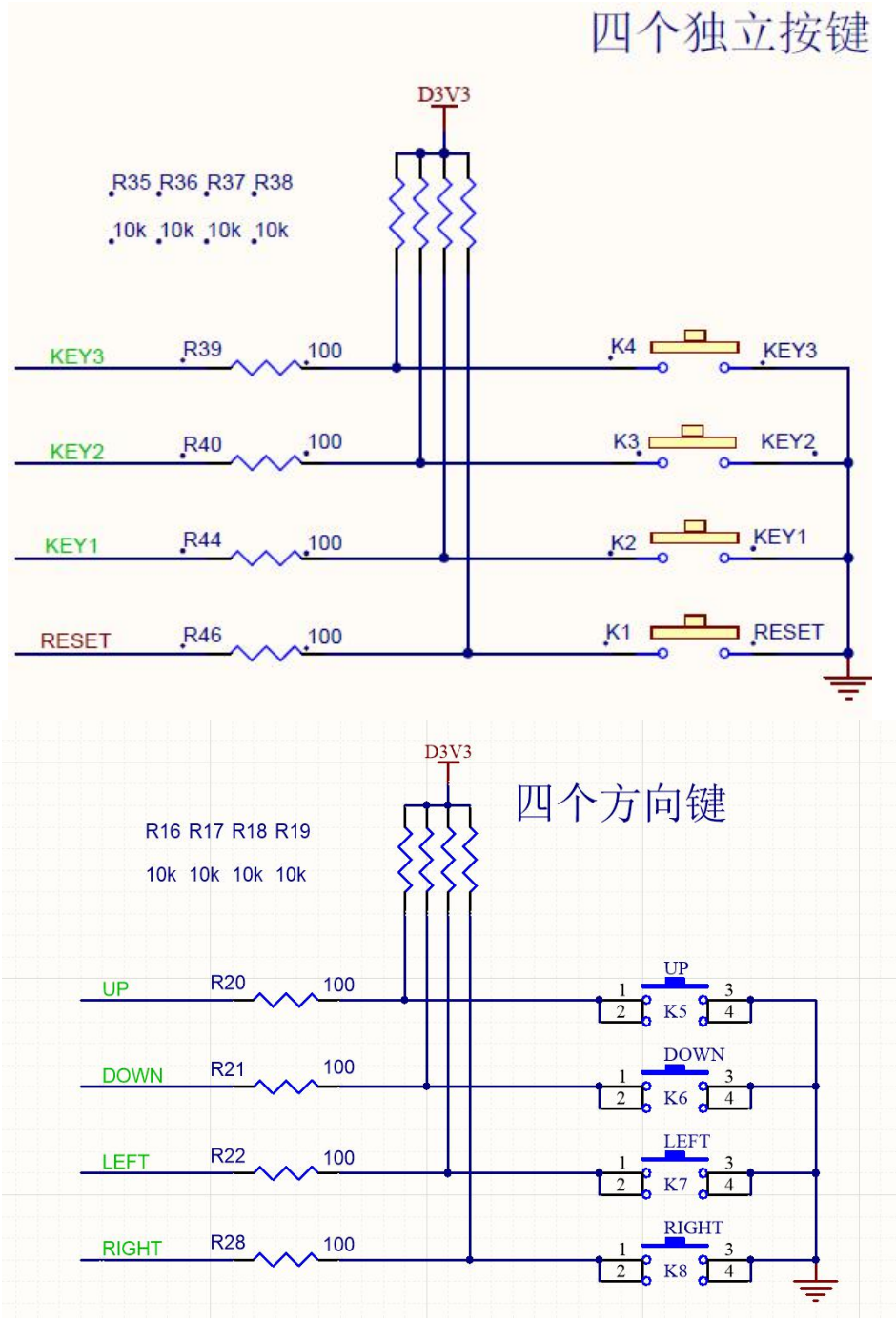


图 13.1

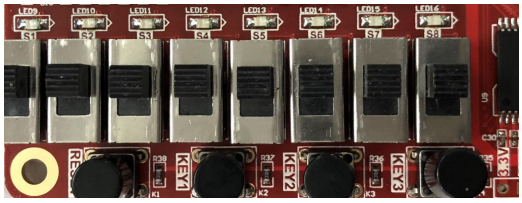


图 13.2.1 4 个独立按键实物图

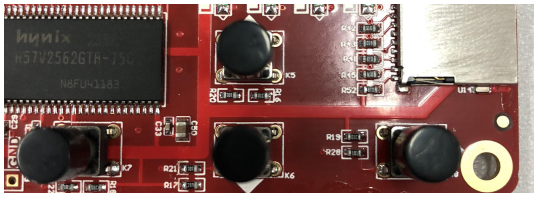


图 13.2.2 4 个方向按键实物图

按键引脚分配：

按键名称	FPGA 引脚	按键标号
RESET	N13	RESET

KEY1	M15	KEY 1
KEY2	M16	KEY 2
KEY3	E16	KEY 3
KEY5	T13	KEY5
KEY6	T12	KEY6
KEY7	R11	KEY7
KEY8	R12	KEY8

十四、 摄像头接口

开发板包含了一个 18 针的 CMOS 摄像头接口，可以连接 OV7670 摄像头模块，可以实现视频采集功能，采集以后，可以通过 TFT 液晶屏或者 VGA 接口连接显示器进行显示。OV7670，30W 像素，输出分辨率为 640*480。关于摄像头选择，用户可以根据自己实际需要进行选购。

CMOS 摄像头接口原理图如图 14.1 所示

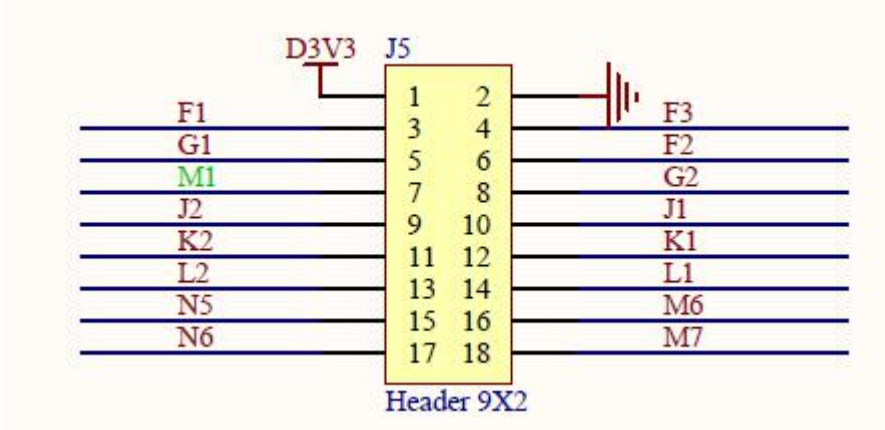


图 14.1 摄像头接口原理图

实物图如图 14.2 所示（**摄像头模块为选配件**）

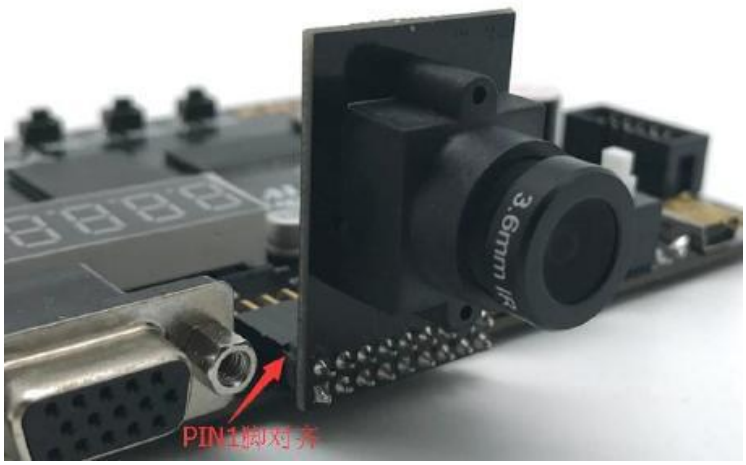


图 14.2 摄像头接口实物图

摄像头接口引脚分配：

引脚号	FPGA 引脚	OV7670 摄像头模块
PIN1	+3.3V	+3.3V
PIN2	GND	GND
PIN3	F1	CMOS_SCL
PIN4	F3	CMOS_SDA
PIN5	G1	CMOS_VSYNC
PIN6	F2	CMOS_HREF
PIN7	M1	CMOS_PCLK
PIN8	G2	CMOS_XCLK
PIN9	J2	CMOS_D7
PIN10	J1	CMOS_D6
PIN11	K2	CMOS_D5
PIN12	K1	CMOS_D4
PIN13	L2	CMOS_D3
PIN14	L1	CMOS_D2
PIN15	N5	CMOS_D1
PIN16	M6	CMOS_D0
PIN17	N6	-
PIN18	M7	-

十五、 数码管

数码管是很常见的一种显示设备，一般分为七段数码管和八段数码管，两者区别就在于八段数码管比七段数码管多了一个“点”。我们采用的数码管为 8位一体的八段数码管，数码管的段结构图 15.1 所示

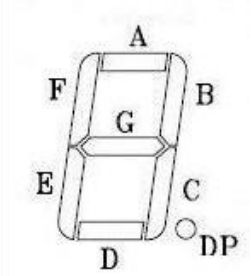


图 15.1 数码管的段结构

我们使用的是共阳极数码管，当某一字段对应的引脚为低电平时，相应字段就点亮，当某一字段对应的引脚为高电平时，相应字段就不亮。

说完上面的原理图，我们来看我们开发板上的设计。

8位一体数码管是属于动态显示，由于人的视觉暂留现象及发光二极管的余辉效应，尽管实际上各位数码管并非同时点亮，但只要扫描的速度足够快，给人的印象就是一组稳定的显示数据，不会有闪烁感。

8位一体数码管的相同的段都接在了一起，一共是 8 个引脚，然后加上 8个控制信号引脚，一共是 16个引脚，如图 15.2 所示，其中 DIG[0..7]是对应数码管的 A,B,C,D,E,F,G,H(即点DP)；SEL[0..7]是8个数码管的8个控制引脚，也是低电平有效，当控制引脚为低电平时，对应的数码管有了供电电压，这样数码管才能点亮，否则无论数码管的段如何变化，也不能点亮对应的数码管。

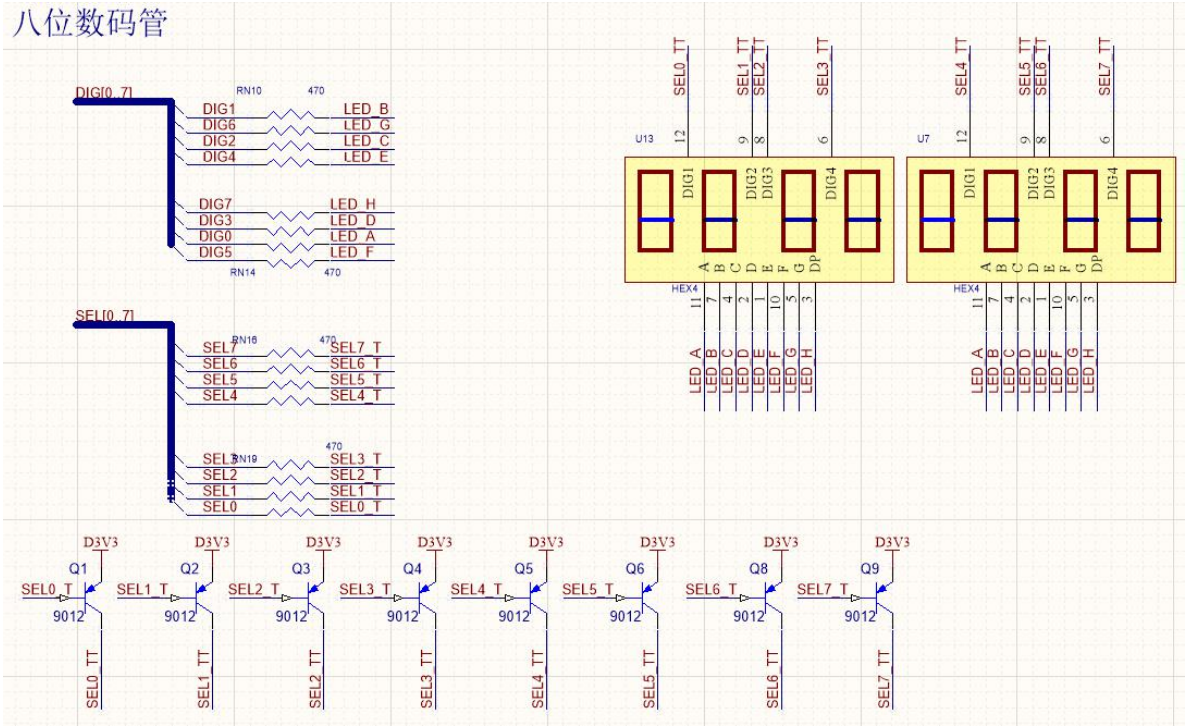


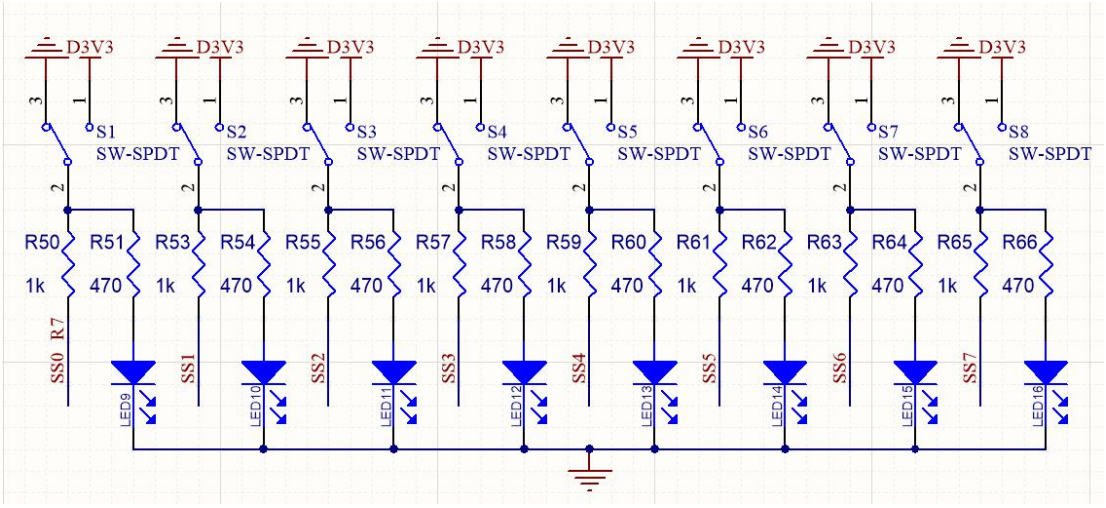
图 15.2 数码管原理图

数码管接口引脚分配：

引脚号	FPGA 引脚	备注
DIG0	R14	对应段A
DIG1	N16	对应段B
DIG2	P16	对应段C
DIG3	T15	对应段D
DIG4	P15	对应段E
DIG5	N12	对应段F
DIG6	N15	对应段G
DIG7	R16	对应点OP
SEL0	T14	从左第一个数码管

SEL1	R13	从左第二个数码管
SEL2	M11	从左第三个数码管
SEL3	P11	从左第四个数码管
SEL4	N11	从左第五个数码管
SEL5	M10	从左第六个数码管
SEL6	P9	从左第七个数码管
SEL7	NP	从左第八个数码管

同时配合使用的8个拨动开关：



引脚分配：

SW0	T11	从左第一个开关
SW1	R10	从左第二个开关
SW2	T10	从左第三个开关
SW3	R9	从左第四个开关
SW4	T9	从左第五个开关
SW5	R8	从左第六个开关
SW6	T8	从左第七个开关
SW7	R7	从左第八个开关

十六、 蜂鸣器

蜂鸣器不多解释了，我们在设计的时候，通过一个三极管进行控制，当低电平时，三极管导通，蜂鸣器响；当高电平，三极管截止，蜂鸣器不响；**为了方便起见，我们在蜂鸣器跟 FPGA 之间加入了一个跳帽（CB1）,如果讨厌蜂鸣器响，可以把跳帽去掉即可**。原理图如图 16.1

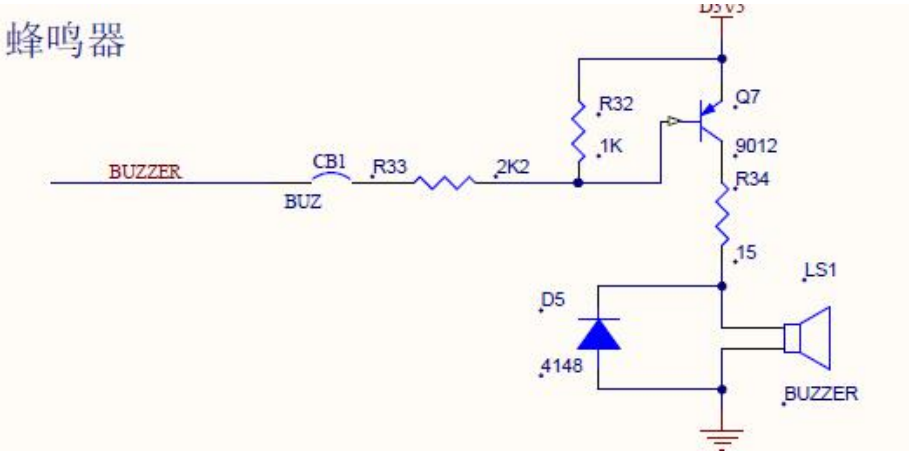


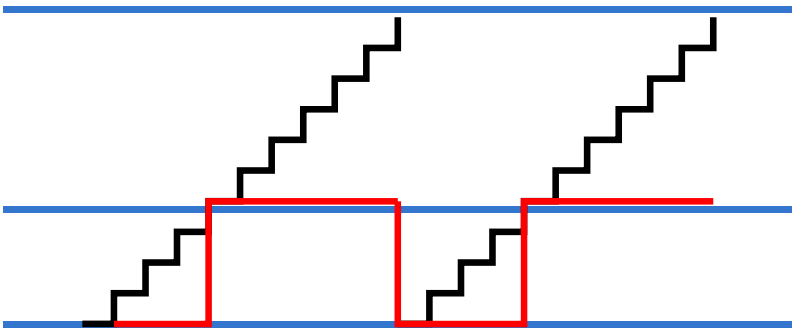
图 16.1 蜂鸣器原理图

图 16.2 为蜂鸣器实物图

蜂鸣器引脚分配：

引脚名称	FPGA 引脚
BUZZER	C11

如下图所示，用一个 N 比特的计数器，最大值可以表示为 2 的 N 次方，最小值 0，计数器以“period”为步进值累加，加到最大值后会溢出，进入下一个累加周期。当计数器值大于“duty” 时，脉冲输出高，否则输出低，这样就可以完成图中红色线所示的脉冲占空比可调的脉冲输出， 同时“period”可以调节脉冲频率。



PWM 脉宽调制示意图

不同的脉冲占空比的方波输出后加在蜂鸣器上，蜂鸣器发出声音大小不同。

十七、 扩展口

开发板预留 2 个扩展口，扩展口分别有 20、40 个信号，其中，5V 电源 1 路，3.3V 电源 2 路，地3 路，IO 口 34 路。这些 IO 口都是独立的 IO 口，没有跟其他设备复用。IO 口连接到 FPGA 引脚上，电平为 3.3V。**切勿直接跟 5V 设备直接连接，以免烧坏 FPGA。如果要接 5V 设备， 需要接电平转换芯片。**

在扩展口和 FPGA 连接之间串联了 33 欧姆的排阻，用于保护 FPGA 以免外界电压或电流过高造成损坏，扩展口 J1,J2 的电路如图 17.1, 17.2 所示

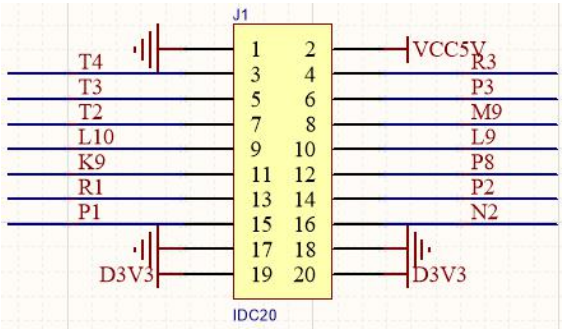


图17.1 J1拓展口原理图

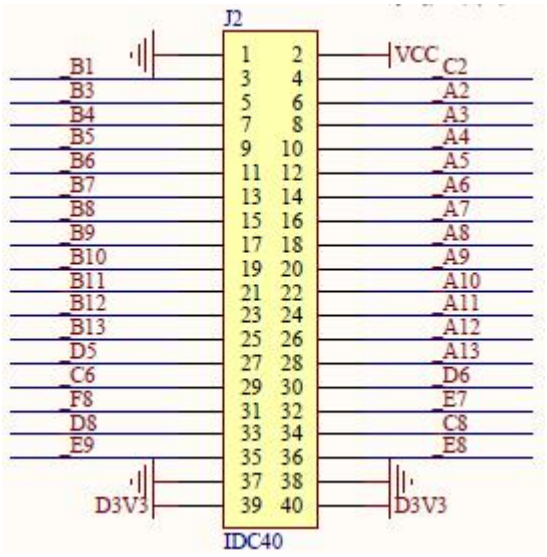


图 17.2 J2 扩展口原理图

图 17.3 为 J1,J2 扩展口实物图，扩展口的 Pin1，Pin2 和 Pin39，Pin40 已经在板上标示出。

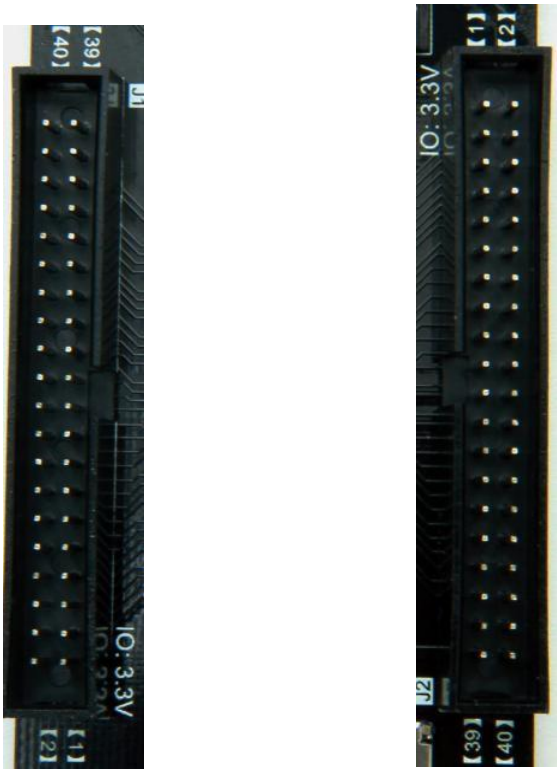


图 17.3 J1, J2 扩展口实物图

此扩展口在连接我们的扩展模块的时候，方向如图 17.4 所示，1,2 脚在接口的上方（注意 PCB 上的标识)。