计算机系统综合设计实验平台

实验手册

西安电子科技大学　　计算机科学与技术学院

计算机网络与信息安全国家级实验教学示范中心

张剑贤　刘锦辉　吴文华　陈勉　编写

2021 年 6 月

目　录

计算机系统综合设计实验平台简介

计算机系统综合设计实验平台是基于 Kintex7 可编程逻辑器件构建的一套计算机专业课程实验平台。该平台采用 Kintex7 系列芯片为核心，片内具有丰富的可编程逻辑资源，可针对数字逻辑、计算机组成原理、处理器系统设计、嵌入式系统设计等课程设计相应课程实验。平台提供有丰富的开关按键、LED 灯等外设接口，还具有千兆以太网络接口、音频、视频输入输出接口。同时平台还具有丰富的扩展 IO，便于进行功能扩展。

- ◆ 实验平台系统主芯片 Kintex7 XC7K160T-2FBG676I
- ◆ 实验平台开发软件：Xilinx Vivado 集成开发软件
- ◆ 实验平台接口外设
  - ✓ 1024MB DDR3 存储器
  - ✓ 256MB Flash 配置存储器
  - ✓ 256MB Flash 用户存储器
  - ✓ 10/100/1000M 以太网
  - ✓ USB-JTAG 接口
  - ✓ JTAG 2X7 接口
  - ✓ USB-UART 接口
  - ✓ USB-HID 接口
  - ✓ Micro-SD 接口
  - ✓ OLED 128x64 液晶屏
  - ✓ 4 个独立按键
  - ✓ 4x4 矩阵按键
  - ✓ 32 个独立开关
  - ✓ 32 个独立 LED 灯
  - ✓ 16 个数码管
  - ✓ 1 个蜂鸣器
  - ✓ 音频输入输出接口
  - ✓ VGA 视频输出接口
  - ✓ HDMI 输入接口
  - ✓ HDMI 显示接口
  - ✓ PMODx3 扩展接口

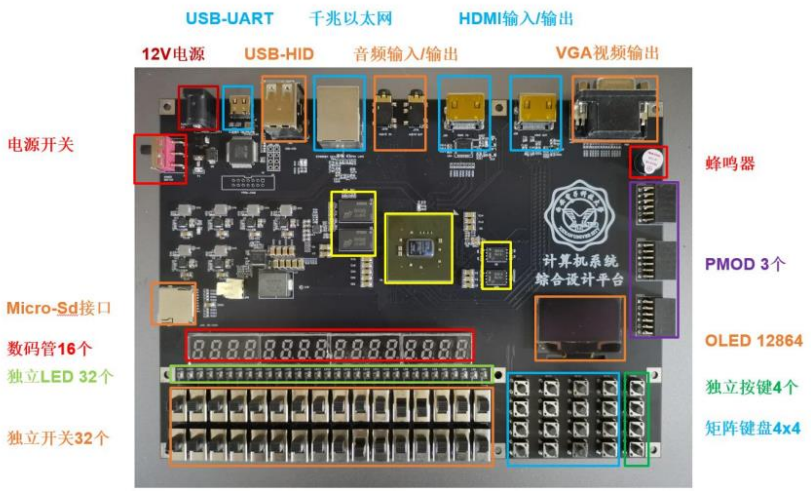


表 1 硬件接口资源引脚说明

元器件名称	元器件编号/信号	FPGA 引脚
LED 灯 LD0~LD31	LED_0	G16
	LED_1	H16
	LED_2	D16
	LED_3	D15
	LED_4	C18
	LED_5	C17
	LED_6	B19
	LED_7	C19
	LED_8	A17
	LED_9	B17
	LED_10	K20
	LED_11	J20
	LED_12	J18
	LED_13	J19
	LED_14	L19
	LED_15	L20
	LED_16	K16
8 段数码管 U17~U20	Seg_DIG1	B20
	Seg_DIG2	C22
	Seg_DIG3	D21
	Seg_DIG4	C24
	Seg_DIG5	B26
	Seg_DIG6	B25
	Seg_DIG7	L23
	Seg_DIG8	K21
	Seg_DIG9	H22
	Seg_DIG10	J24
	Seg_DIG11	J25
	Seg_DIG12	A20
	Seg_DIG13	C26
	Seg_DIG14	D26
	Seg_DIG15	A24
	Seg_DIG16	A23
	Seg_a	J21
	Seg_b	H24
	Seg_c	H23
	Seg_d	G21
	Seg_e	H21
	Seg_f	H26

	Seg_g	J26
	Seg_dp	E26
32 个拨动开关 SW0~SW31	SW_0	C9
	SW_1	B9
	SW_2	G11
	SW_3	F10
	SW_4	D10
	SW_5	E11
	SW_6	D11
	SW_7	A14
	SW_8	B10
	SW_9	A10
	SW_10	B15
	SW_11	A15
	SW_12	A13
	SW_13	A12
	SW_14	D8
	SW_15	D9
	SW_16	F8
	SW_17	F9
	SW_18	H11
	SW_19	H12
	SW_20	G14
	SW_21	J10
	SW_22	H14
	SW_23	J11
	SW_24	H13
	SW_25	J13
	SW_26	G9
	SW_27	G10
	SW_28	H8
	SW_29	H9
	SW_30	J14
	SW_31	J8
4*4 矩阵键盘 KEY0~KEYF	BTN_C0	AA12
	BTN_C1	AA13
	BTN_C2	AC12
	BTN_C3	AB12
	BTN_R0	AE10
	BTN_R1	AD10
	BTN_R2	AE12

	BTN_R3	AD12
UART 串行收发 U27	FPGA_TXD_K7	E18
	FPGA_RXD_K7	D18
蜂鸣器	BEEP_EN_K7	F25
独立按键 BNT0~BNT3	BTN0	AF12
	BTN1	Y13
	BTN2	AD13
	BTN3	AC13
VGA 接口 J43	VGA_HS	M21
	VGA_VS	M22
	VGA_B1	U17
	VGA_B2	T17
	VGA_B3	R18
	VGA_B4	P18
	VGA_R1	N16
	VGA_R2	U16
	VGA_R3	N26
	VGA_R4	M26
	VGA_G1	P19
	VGA_G2	P20
	VGA_G3	R25
	VGA_G4	P25
音频接口 U22	AUD_MCLK	R22
	AUD_SCL	N24
	AUD_SDA	P24
	AUD_LRCLK（GPIO3）	P26
	AUD_BCLK（GPIO2）	R26
	AUD_ADC_SDATA（GPIO1）	K26
	AUD_DAC_SDATA（GPIO0）	K25
	AUD_ADR0	M25
	AUD_ADR1	L25
HDMI 输出接口 J39	HDMI_TX0_P	T24
	HDMI_TX1_P	T20
	HDMI_TX2_P	T22
	HDMI_TX_CLK_P	N21
	HDMI_TX_CEC	D25
	HDMI_TX_HPD	E25
	HDMI_TX_SDA	R17
	HDMI_TX_SCL	R16
HDMI 输入接口 J38	HDMI_RX0_P	U19
	HDMI_RX1_P	T18

	HDMI_RX2_P	P16
	HDMI_RX_CLK_P	R21
	HDMI_RX_CEC	B21
	HDMI_RX_HPD	C21
	HDMI_RX_SDA	M19
	HDMI_RX_SCL	N18
千兆以太网络接口 U16	ETH_RXD0	F14
	ETH_RXD1	F13
	ETH_RXD2	G12
	ETH_RXD3	F12
	ETH_RXCK	C12
	ETH_RXCTL	C11
	ETH_TXD0	E13
	ETH_TXD1	E12
	ETH_TXD2	C14
	ETH_TXD3	C13
	ETH_MDC	B12
	ETH_MDIO	B11
	ETH_RST_B	B14
PMOD 接口 PMODA~PMODC	JA1_P	G17
	JA1_N	F18
	JA2_P	G19
	JA2_N	F20
	JA3_P	F19
	JA3_N	E20
	JA4_P	H19
	JA4_N	G20
	JB1_P	F17
	JB1_N	E17
	JB2_P	D19
	JB2_N	D20
	JB3_P	E15
	JB3_N	E16
	JB4_P	H17
	JB4_N	H18
	JC1_P	G24
	JC1_N	F24
	JC2_P	E21
	JC2_N	E22
	JC3_P	L22
	JC3_N	K22

	JC4_P	K23
	JC4_N	J23
用户 QSPI FLASH 接口 U25	QSPI_DQ0	E23
	QSPI_DQ1	F22
	QSPI_DQ2	D24
	QSPI_DQ3	D23
	QSPI_CS	F23
	QSPI_SCK	G22
Micro SD 卡接口 J44	SD_CD	R23
	SD_D0	L24
	SD_D1	M24
	SD_D2	M20
	SD_D3	N19
	SD_CLK	P23
	SD_CMD	N23
	SD_CD	R23
时钟引脚	100M/100Mhz	E10
	200M/200Mhz	AA10

实验 1 3-8 译码器设计

1. 实验名称：3-8 译码器设计

2. 实验目的

1）掌握 Vivado 开发工具的使用，掌握 FPGA 开发的基本步骤；

2）掌握组合逻辑电路设计的一般方法；

3）掌握程序下载方法，了解 XDC 文件的格式；

4）初步了解开发板资源，掌握开发板使用方法。重点了解拨动开关和 LED 显示灯的使用方法。

3.实验内容

3.1 用硬件描述语言实现 3-8 译码器模块

译码器电路如图 1.1 所示。其功能如表 1.1 所示。试用硬件描述语言实现该译码器，并在开发板上进行检验。

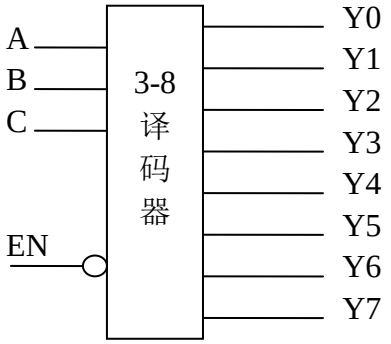


图 1.1 3-8 译码器

表 1.1 译码器功能表

EN	A	B	C	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
1	X	X	X	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	1	0	0
0	0	1	1	0	0	0	0	1	0	0	0
0	1	0	0	0	0	0	1	0	0	0	0
0	1	0	1	0	0	1	0	0	0	0	0
0	1	1	0	0	1	0	0	0	0	0	0
0	1	1	1	1	0	0	0	0	0	0	0

3.2 将程序下载到 FPGA 并进行检验

- 资源使用要求：
  - 用拨动开关 SW3~SW1 作为输入 A，B，C；
  - 拨动开关 SW0 控制 EN；
  - 8 个 LED 灯表示 8 个输出。
- 检验方法：

当 SW0 处于 ON（EN=1）位置时，所有 LED 灯灭；

当 SW0 处于 OFF（EN=0），反映当前输入的译码输出在 LED 灯上显示，即当输入为 000（拨动开关 SW3-SW1 处于 OFF 状态），LED0 亮，其它灯灭，等等。

4.实验步骤

1）启动 Vivado，新建工程文件，编写 3-8 译码器的硬件描述语言模块；

2）新建 XDC 文件，输入引脚约束；

3）完成综合、实现，生成下载文件；

4）连接开发板 USB 下载线，开启开发板电源；

5）下载到 FPGA；

6）拨动开关，验证结果是否正确。

实验 2 多路选择器设计

1．实验名称：多路选择器设计

2．实验目的

1）掌握 Vivado 开发工具的使用，掌握 FPGA 开发的基本步骤；

2）掌握 4 选 1 多路选择器电路设计的一般办法；

3）掌握程序下载的办法；

4）初步了解开发板资源，掌握开发板的使用方法，重点掌握按键，开关，LED 的使用方法。

3.实验内容

（1）用硬件描述语言实现 4 选 1 多路选择器

4 选 1 多路选择器顶层模块电路如图 2.1 所示。

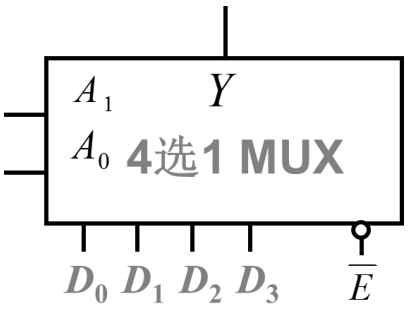


图 2.1 8 位串行全加器顶层模块

其中 E 为使能信号， D0~D3：数据输入，A0/A1 选择信号，使用板上开关（SW0~SW15）；Y：运算结果输出，使用 LED 显示运算结果。

（2）资源使用要求：

用开关作为控制输入，用 LED 显示输出结果。

（3）实验步骤

1）启动 Vivado,新建工程文件；

2）编写 4 选 1 多路选择器 MUX;

3）编写完选择器模块之后，在顶层文件上实现映射；

4）新建 XDC 文件，输入引脚约束；

5）完成综合，实现，生成下载文件；

6）连接开发板 USB 下载线，开启开发板电源；

7）下载 FPGA;

8）输入数据，验证结果。

实验 3 移位寄存器设计

1．实验名称：移位寄存器器设计

2．实验目的

1）掌握 Vivado 开发工具的使用，掌握 FPGA 开发的基本步骤；

2）掌握双向移位寄存器电路设计的一般办法；

3）掌握程序下载的办法；

4）初步了解开发板资源，掌握开发板的使用方法，重点掌握按键，开关，LED 的使用方法。

3.实验内容

（1）用硬件描述语言实现双向移位寄存器

双向移位寄存器顶层模块电路如图 3.1 所示。

图 3.1　8 位串行全加器顶层模块

其中 clk 为移位寄存器时钟，上升沿有效；din 为串行数据输入信号；clr 为异步清零端，dir 为方向选择，低电平左移，高电平右移；dout[7..0]表示 8 位的并行数据输出。

（2）将程序下载到 FPGA 并进行检验

资源使用要求：用开关（SW0~SW2）作为控制输入；用 LED（LD0~LD7）显示运算结果。

（3）实验步骤

1）启动 Vivado,新建工程文件；

2）编写双向移位寄存器；

3）编写完选择器模块之后，在顶层文件上实现映射；

4）新建 XDC 文件，输入引脚约束；

5）完成综合，实现，生成下载文件；

6）连接开发板 USB 下载线，开启开发板电源；

7）下载 FPGA;

8）输入数据，验证结果。

实验 4 可逆计数器设计

1.实验名称：可逆计数器设计

2.实验目的

1）进一步熟练 Vivado 开发工具，巩固 FPGA 开发的基本步骤，掌握功能仿真方法；

2）掌握时序逻辑电路设计的一般方法，掌握时钟分频程序设计方法；

3）理解硬件描述语言的层次结构设计；

4）巩固程序下载方法；

5）了解开发板时钟资源，以及时钟分频方法。

3.实验内容

3.1 实现如下基本的可逆计数器

可逆计数器电路图及功能表如图 4.1 和表 4.1 所示。

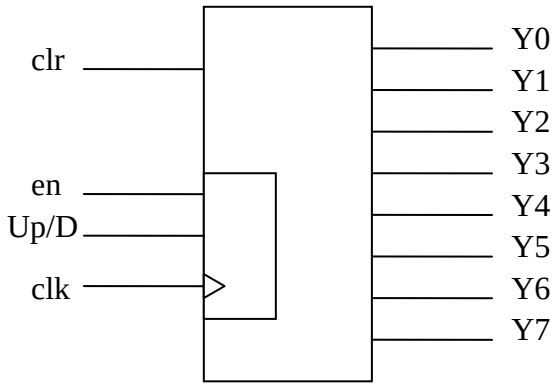


图 4.1 可逆计数器原理图

表 4.1 可逆计数器功能表

clr	en	Up/Dn	clk	Y7 ~ Y0
1	X	X	X	00000000
0	0	X	X	停止计数
0	1	1	↑	计数器+1 操作
0	1	0	↑	计数器-1 操作

3.2 资源使用要求及实现方法

1）用 LED0~LED7 作为计数器输出显示，LED7 为高位，LED0 为低位；

2）SW0 为计数方向 up/dn 控制；

3）SW1 为计数允许 EN 控制端；

4）BTN_EAST 为 clr 按钮；

5）计数时钟频率为 1Hz，通过对 50Mhz 系统时钟分频得到，分频电路独立编写一个模块，如图 4.1 所示；

6）扩展：可以对按键次数进行计数（按键为 BTN_SOUTH），即通过 SW2 选择计数源。2 选 1 电路如图 3-3 所示。

7）将图 4.1~图 4.3 三个模块连接起来，构成一个完整计数器，如图 4.4 所示。

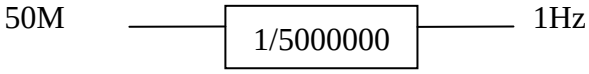


图 4.2　1/5000000 分频器

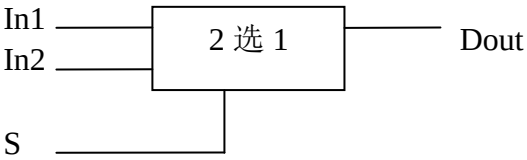


图 4.3 2 选 1 电路

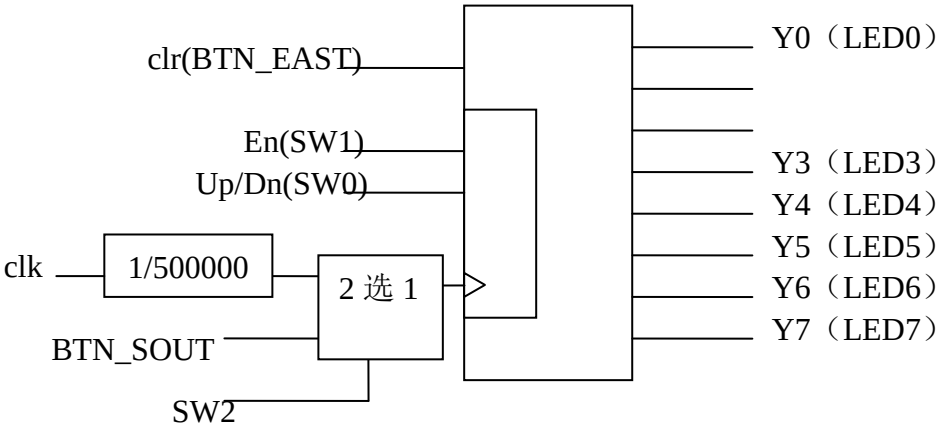


图 4.4 完整的可逆计数器

4.实验步骤

1）建立工程文件，顶层文件为原理图；

2）编写分频模块程序，要求输入为 50MHz 系统时钟，输出为 1Hz 计数时钟；

3）编写 2 选 1 模块，输入为按键、1Hz 时钟和开关 SW2，输出为计数源；

4）编写计数器模块主程序模块；

5）在原理图中，将各个模块连接，使用 pinhead 分配引脚资源；

6）对程序进行功能仿真；

7）下载程序，进行验证。

实验 5 分频器设计

1．实验名称：50%占空比奇数分频器设计

2．实验目的

1）掌握 Vivado 开发工具的使用，掌握 FPGA 开发的基本步骤；

2）掌握奇数分频器电路设计的一般办法；

3）掌握程序下载的办法；

4）初步了解开发板资源，掌握开发板的使用方法，重点掌握按键，开关，LED 的使用方法。

3.实验内容

（1）用硬件描述语言实现 50%占空比奇数分频器

50%占空比奇数分频器顶层仿真结果图如图 5.1 所示。

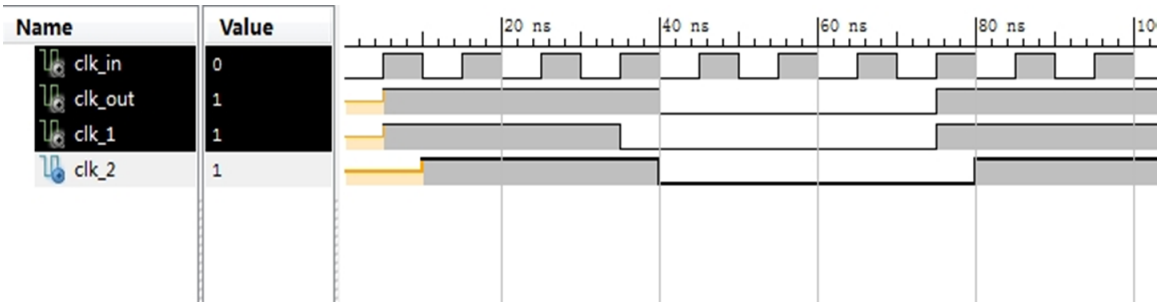


图 5.1　50%占空比的 7 分频电路仿真结果

其中 clk_in 为时钟输入信号，clk_out 为 7 分频信号输出。

（2）将程序下载到 FPGA 并进行检验

资源使用要求：　用 Modelsim 工具进行模块仿真验证。

（3）实验步骤

1）启动 Vivado,新建工程文件；

2）编写 50%占空比奇数分频器。

3）编写完加法器模块之后，在顶层文件上实现映射；

4）新建 XDC 文件，输入引脚约束；

5）完成综合，实现，生成下载文件；

6）编写测试文件，在仿真工具下验证结果。

实验 6 跑马灯控制设计

1.实验名称：跑马灯电路设计

2.实验目的

1）进一步熟练 Vivado 开发工具，巩固 FPGA 开发的基本步骤，掌握功能仿真方法；

2）掌握时序逻辑电路设计的一般方法，掌握时钟分频程序设计方法；

3）理解硬件描述语言的层次结构设计，掌握多个进程设计方法；

4）巩固程序下载方法；

5）了解开发板时钟资源，以及时钟分频方法。

3.实验内容

3.1 实现如下基本的跑马灯电路

跑马灯电路图如图 6.1 所示。

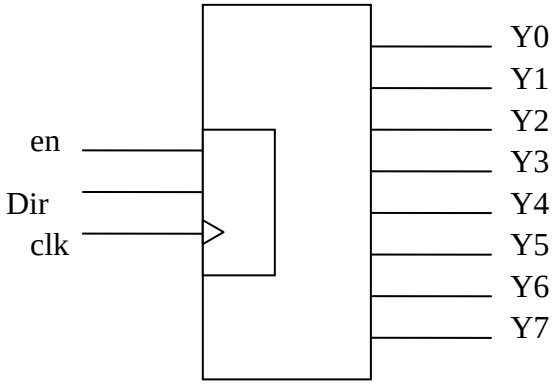


图 6.1 跑马灯原理图

功能叙述：初始情况下 Y0=‘1’，其它为‘0’。然后，在 en 为高电平的情况下，在时钟信号 clk 的下降沿进行移位。当 dir=‘1’时，每来一个时钟信号，循环左移一位，当 dir=‘0’时，每来一个时钟，循环右移一位。

另外，移位控制时钟可以选择为按键，即每按键一次相当于一个时钟信号，系统可以在按键和系统分频时钟之间进行选择。

要求：

1）Clk 信号的周期为 1s，利用分频电路对 50Mhz 系统时钟分频得到。

2）在一个结构体中，写多个进程：时钟分频进程，移位进程，二选一进程。

3.2 资源使用要求

1）用 LED0~LED7 作为跑马灯输出显示，LED7 为高位，LED0 为低位；

2）SW0 为循环方向控制；

3）SW1 为工作允许 EN 控制端；

4）计数时钟频率为 1Hz，通过对 50Mhz 系统时钟分频得到；

6）扩展：可以由按键来控制循环（按键为 BTN_SOUTH），即每按一次 BTN_SOUTH, 则完成一次移位。通过 SW2 选择移位控制时钟沿。

完整的原理图如图 6.2 所示。

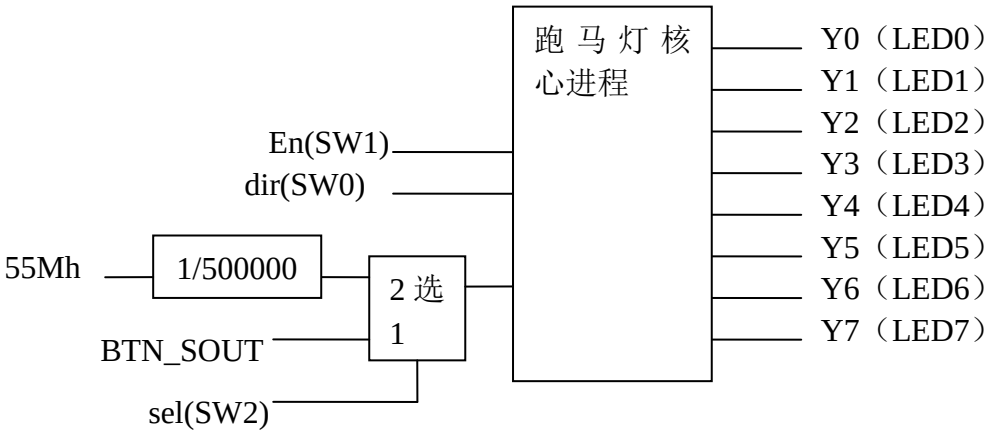


图 6.2 完整的跑马灯电路

4.实验步骤

1）建立工程文件，顶层文件为 HDL；

2）编写顶层程序。程序包括多个进程；

3）编写分频进程程序，要求输入为 50MHz 时钟，输出为 1Hz 计数时钟；

4）编写 2 选 1 进程，输入为按键、1Hz 时钟和开关 SW2，输出为跑马灯移位控制源；

5）编写移位进程；

6）对程序进行功能仿真；

7）下载程序，进行验证。

实验 7 有限状态机设计

1．实验名称：有限状态机设计

2．实验目的

1）掌握 Vivado 开发工具的使用，掌握 FPGA 开发的基本步骤；

2）掌握有限状态机电路设计的一般办法；

3）掌握程序下载的办法；

4）初步了解开发板资源，掌握开发板的使用方法，重点掌握按键，开关， LED 的使用方法。

3.实验内容

（1）用硬件描述语言实现有限状态机

根据有限状态机的仿真结果进行有限状态机电路设计，如图 7.1 所示。

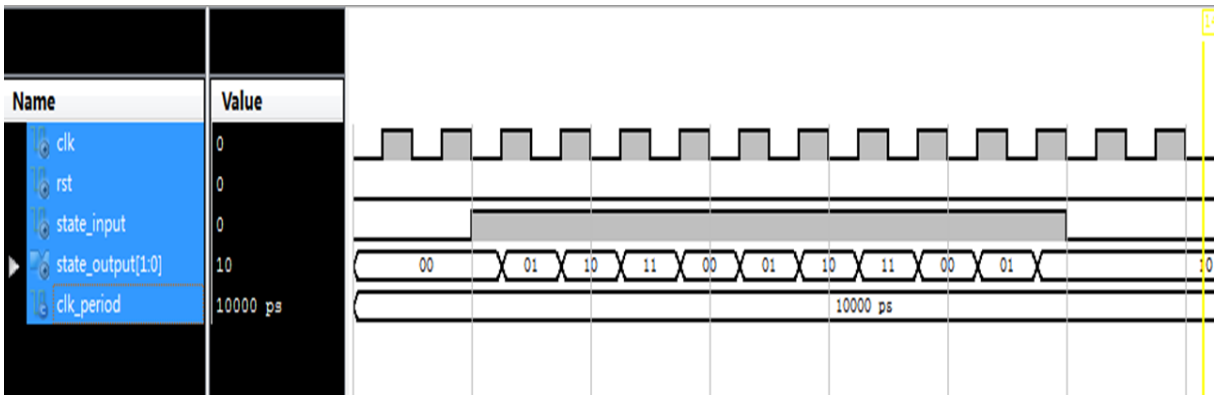


图 7.1 有限状态机仿真波形

其中 rst，state_input：数据输入，使用板上开关（SW0~SW15）；

State_output：运算结果输出，使用 LED 显示运算结果。

（2）将程序下载到 FPGA 并进行检验

资源使用要求：用开关（SW0~SW15）输入状态控制信号；用 LED（LD8~LD15）显示状态输出结果。

（3）实验步骤

1）启动 Vivado,新建工程文件；

2）编写有限状态机电路程序。

3）编写完状态机电路模块之后，在顶层文件上实现映射；

4）新建 XDC 文件，输入引脚约束；

5）完成综合，实现，生成下载文件；

6）连接开发板 USB 下载线，开启开发板电源；

7）下载 FPGA;

8）输入数据，验证结果。

实验 8 按键消抖电路设计

1.实验名称：按键抖动消除及验证电路设计

2.实验目的

1）进一步熟练 Vivado 开发工具，巩固 FPGA 开发的基本步骤，掌握功能仿真方法；

2）了解按键抖动的原因，抖动消除方法

3）掌握状态机的设计方法；

4）掌握消除抖动的状态机设计方法

5）巩固程序下载方法。

3.实验内容

3.1 原理简介

按键动作发生时,按键的输出会出现不稳定的逻辑'0'和逻辑'1'的跳变。该信号直接输入到计数器之类电路,会发生计数错误。如图 8.1 所示。去抖的目的是根据抖动信号产生如图 8.2 所示的信号。

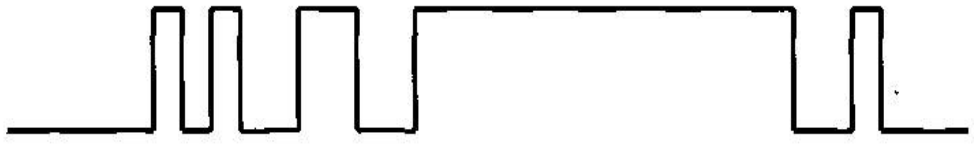


图 8.1 信号抖动(pushbtn)

图 8.2 去抖后的信号(key_out)

3.2 提示：状态机设计思路

设置六个状态：S0，S1，S2，S3，S4，S5。

电路启动时，进入复位状态 S0，在 S0 状态下，去抖输出信号 key_out 为'0'，时钟信号 divclk（频率为 2.5Mhz，通过 50Mhz 系统时钟 20 分频得到）以一定频率采样按键输入信号 pushbtn，如果采样到 pushbtn = '0'，则停留在 S0 状态，并继续采样，一旦采样到 pushbtn='1'，则转入 S1 延时状态，进行消抖延时（延时可以用计数器来完成，比如 50 个 divclk 周期），当延时结束时，Delay_end='1'，则转入 S2 状态，在此状态下，时钟信号 divclk 以一定频率采样按键输入 pushbtn，如果 pushbtn = '0'则转入 S0，否则转入 S3；状态 S3，S4 的转换过程和条件与 S2 相同，在状态 S4 下，如果 pushbtn='1'，则转入 S5 状态，当进入 S5 时，表示经过 S2，S3，S4 三个连续状态检测按键输入 pushbtn 都为'1'，则认为按键处于稳定状态，在 S5 输出按键确认信号 key_out ='1'。

同时在状态 S5 下，时钟信号 divclk 检测按键输入 pushbtn，如果 pushbtn='1'，表示按键未释放，继续停留在 S5，检测输入信号，如果检测到 pushbtn='0'，表示按键已经释放，输出 key_out= '0', 返回到状态 S0，等待下一次按键操作。

3.3 完成验证电路设计

设计一个按键计数器，通过选择开关，对未去抖的信号和去抖后的信号分别进行计数。验证设计的正确性。完整的原理图如图 8.3 所示。消抖电路的采样时钟最好通过 50Mhz 进行分频后产生。

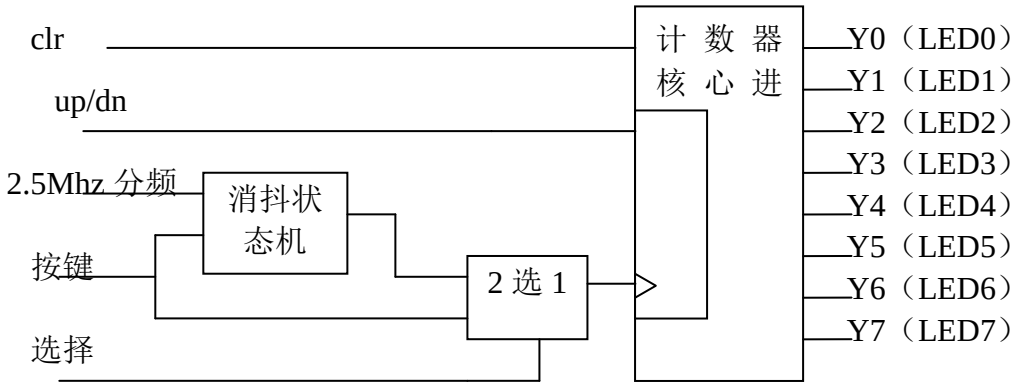


图 8.3 完整电路

3.4 资源使用要求

1）用 LED0~LED7 作为计数输出。

2）BTN_SOUTH 作为待计数的按键。

3）对去抖信号还是直接按键计数选择用 SW1 开关控制。即当 SW1 处于‘1’状态，对去抖后的信号计数，否则直接对按键 btn_south 计数。

4）SW0 用来控制加 1 计数还是减 1 计数，当 up/dn = '1'时，加 1 计数。

5）BTN_EAST 作为计数器清 0 控制，当按下时，对计数器清 0。

注：状态机使用对系统时钟进行分频后的时钟信号。

4.实验步骤

1）画出电路的状态转换图；

2）编写完整的硬件描述语言程序；

3）下载程序，进行验证。

实验 9 乒乓球游戏设计

1.实验名称：乒乓球游戏设计

2.实验目的

1）进一步熟练 Vivado 开发工具，巩固 FPGA 开发的基本步骤，掌握功能仿真方法；

2）巩固状态机的设计方法；

3）巩固按键消抖电路设计方法；

4）掌握多进程程序设计方法；

5）巩固程序下载方法；

6）了解开发板时钟资源，以及时钟分频方法。

3.实验内容

3.1 原理简介

两人乒乓游戏机用 8 个发光二极管代表乒乓球台，中间两个发光二极管作为乒乓球网，用点亮的发光二极管按照一定的方向移动来表示球的运动。在游戏机的两侧各设置发球和击球开关，甲乙双方按乒乓球比赛规则来操作开关。当甲方按动发球开关时，靠近甲方的第一个发光二极管亮，然后发光二极管由甲方向乙方依次点亮，代表乒乓球的移动，当球过网后，按照设计者规定的球位乙方就可以击球。若乙方提前击球或者未击到球，则甲方得分。然后重新发球比赛，直到一方达到 21 分为止，记分清 0，重新开始新一局比赛。

3.2 提示：状态机设计思路

设置七个状态："等待发球状态"、"第一盏灯亮状态"、"第八盏灯亮状态"、"球向乙移动状态" 、"球向甲移动状态"、"允许甲击球状态" 、"允许乙击球状态"。开始的时候处于"等待发球状态"，若甲发球则状态转移到"第一盏灯亮状态"，若乙发球则转移到"第八盏灯亮状态"。以甲发球为例：若发球后乙没有提前击球（规定球移动到对方第一个发光二极管时允许击球），那么状态机从"第一盏灯亮状态"转移到"球向乙移动状态"，若此时乙依然没有提前击球，状态就转移到"允许乙击球状态"，在此状态下，如果乙击球了，那么状态就转移到"球向甲移动状态"。在"第一盏灯亮状态"、"球向乙移动状态"中，如果乙击球了，就算提前击球，这样甲得分，状态转移到"等待发球状态"等待发球。"球向甲移动状态"之后的过程和前面的过程只不过是甲乙角色的调换而已，状态转移规则一样。状态转换图如图 9.1 所示。

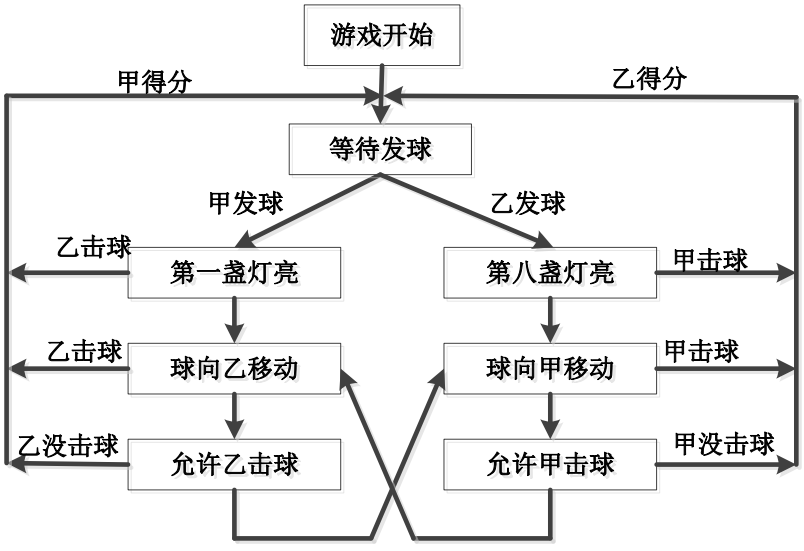


图 9.1 乒乓球游戏状态转换图

3.3 实体设计提示

输入端口：

1）考虑一个异步置位端口 reset，用于在系统不正常时回到初始常态；

2）两个发球输入端 serve1 和 serve2，逻辑‘1’分别标识甲方和乙方发球；

3）两个击球输入端 hit1 和 hit2，逻辑‘1’分别标识甲方和乙方击球；

4）一个开始游戏按钮 startbtn，处于逻辑‘1’标识可以游戏；

5）时钟输入端口 clk

输出端口：8 个二极管，标识乒乓球台。

3.4 资源使用要求

1）用 LED0~LED7 作为乒乓球台，其中 LED3，LED4 作为球网，总是点亮；

2）BTN_EAST 作为开始游戏按钮，按下一次，重新开始游戏；

3）BTN_SOUTH，BTN_EAST 分别作为甲乙发球按钮；

4）SW0 作为甲击球开关，SW3 作为乙击球开关。击球的动作为 ON-OFF。

4.实验步骤

1）画出游戏的状态转换图；

2）建立程序工程；

3）编写按键去抖进程

4）编写状态机进程；

5）下载程序，进行验证。

实验 10 8 位串行全加器设计

1．实验名称：8 位串行全加器设计

2．实验目的

1）掌握 Vivado 开发工具的使用，掌握 FPGA 开发的基本步骤；

2）掌握 8 位串行全加器电路设计的一般办法；

3）掌握程序下载的办法；

4）初步了解开发板资源，掌握开发板的使用方法，重点掌握按键，开关，LED 的使用方法。

3.实验内容

（1）用硬件描述语言实现 8 位串行全加器

8 位串行全加器顶层模块电路如图 10.1 所示。

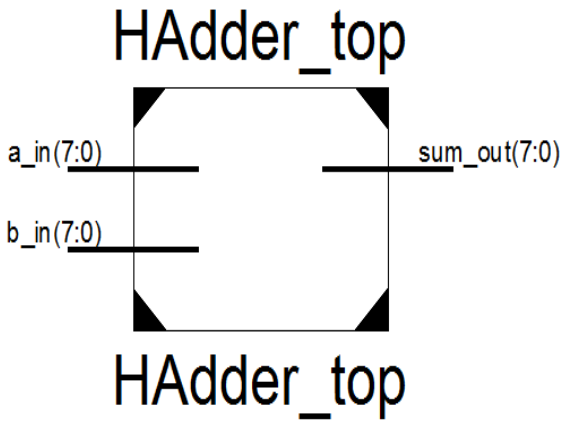


图 10.1　8 位串行全加器顶层模块

其中 a_in，b_in：数据输入，使用板上开关（SW0~SW15）；

sum_out：运算结果输出，使用 LED 显示运算结果。

（2）将程序下载到 FPGA 并进行检验

资源使用要求：用开关（SW0~SW15）输入加数，被加数；用 LED（LD8~LD15）显示运算结果。

（3）实验步骤

1）启动 Vivado,新建工程文件；

2）编写 8 位串行全加器模块 Hadder。

3）编写完加法器模块之后，在顶层文件上实现映射；

4）新建 XDC 文件，输入引脚约束；

5）完成综合，实现，生成下载文件；

6）连接开发板 USB 下载线，开启开发板电源；

7）下载 FPGA;

8）输入数据，验证结果。

实验 11 8 位并行全加器设计

1.实验名称：8 位并行全加器设计

2.实验目的

1）掌握 Vivado 开发工具的使用，掌握 FPGA 开发的基本步骤；

2）掌握 8 位并行全加器电路设计的一般办法；

3）掌握程序下载的办法；

4）初步了解开发板资源，掌握开发板的使用方法，重点掌握按键，开关，LED 的使用方法。

3.实验内容

（1）用硬件描述语言实现 8 位并行全加器

8 位并行全加器的顶层模块如图 11.1 所示。

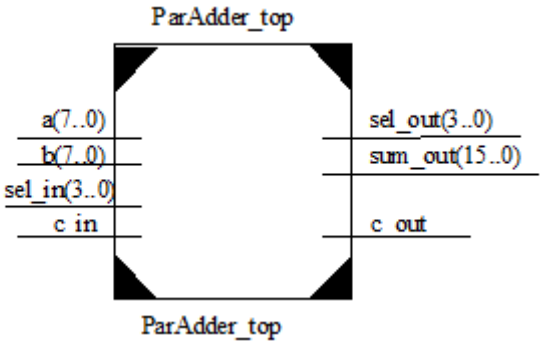


图 11.1　8 位全加器顶层模块

a_in，b_in:输入 8 位加数和被加数；sel_in:数码管片选端；

c_in，c_out:进位输入，进位输出；sum_out:运算结果的输出。

（2）将程序下载到 FPGA 并进行检验

资源使用要求：用开关（SW0~SW7）输入加数，被加数，（SW8~SW11）控制用哪个数码管显示数据，s12 用于进位输入；用 D8 显示结果进位。

（3）实验步骤

1）启动 Vivado,新建工程文件；

2）编写 8 位并行全加器模块 ParAdder, 要求 8 位全加器采用并行进位算法。

3）编写完加法器模块之后，在顶层文件上实现映射；

4）新建 XDC 文件，输入引脚约束；

5）完成综合，实现，生成下载文件；

6）连接开发板 USB 下载线，开启开发板电源；

7）下载 FPGA;

8）输入数据，验证结果。

实验 12 数码管显示程序设计

1．实验名称：数码管显示程序设计

2.实验目的

1）掌握 Vivado 开发工具的使用，掌握 FPGA 开发的基本步骤；

2）掌握数码管扫描显示的基本原理和设计方法；

3）掌握 Vivado 仿真调试的基本方法。

3.实验内容

（1）用硬件描述语言实现 16 个七段数码管扫描显示模块

七段数码管扫描显示模块电路如图 12.1 到图 12.3 所示，主要包括顶层的测试模块，数码管扫描显示模块，以及输入数据向七段数码管进行译码的模块。使用硬件描述语言实现由扩展板上的拨码开关输入显示数值，在 16 个数码管的扫描显示功能，并使用 Vivado 仿真工具进行检验。

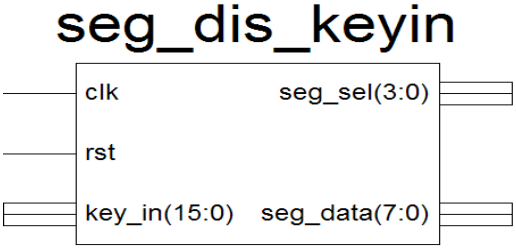


图 12.1 数码管扫描显示顶层测试模块

该模块是一个顶层的测试模块，其中，clk 为系统的时钟输入，rst 信号为复位信号，key_in(15:0)为十六个拨码开关的输入值，seg_sel(3:0)为数码管选择编码输出信号，seg_data(7:0)为数码管显示数据输出。

图 12. 2 数码管扫描显示模块

该模块实现数码管扫描显示功能，扩展板上的 16 个数码管四个一组，分为 A，B，C，D 四组。模块管脚说明如下：clk 为系统的时钟输入，rst 信号为复位信号，data_in_A(15:0)为 A 组四个数码管的输入显示值，其中，data_in_A(3:0)对应于第一个数码管的输入显示值；data_in_A(7:4)对应第二个数码管的输入显示值；data_in_A(11:8）对应第三个数码管的输入显示值；data_in_A(15:12)对应第四个数码管的输入显示值，其它各组与 A 组分配一致；data_in_B(15:0)为 B 组四个数码管的输入显示值；data_in_C(15:0)为 C 组四个数码管的输入显示值；data_in_D(15:0)为 D 组四个数码管的输入显示值；seg_sel(3:0)为数码管选择编码输出信号，seg_data(7:0)为数码管显示数据输出。

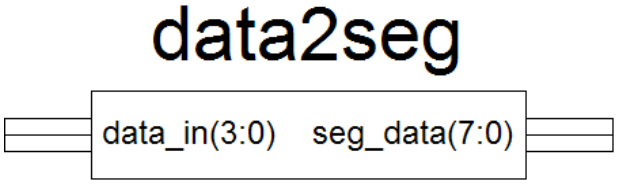


图 12.3 数据译码模块模块

该模块实现将输入的 4 为二进制数转换为数码管显示的数据；其中，data_in(3:0)为输入值，seg_data(7:0)为数码管编码的输出值。

（2）原理简介

key_in(15:0)为四组数码管的输入显示值。

Clk，rst 为时钟和复位信号的输入。时钟主要用于状态机的设计。采用异步复位信号进行数据输入。

seg_sel(3:0)为数码管选择编码输出信号；

seg_data(7:0)为数码管显示数据输出。

（3）数码管扫描显示状态机设计思路

状态机的描述采用三段式方法设计，使用三个进程分别描述状态存储逻辑，次态生成逻辑和输出及控制信号产生逻辑三部分。每次开始运算前，需要对电路进行复位。扫描显示状态机进入初始状态 st0，对应于第一个数码管，输出的选择信号 seg_sel(3:0)=0000，seg_data(7:0)输出对应于输入值进行编码后的数值；之后在每一个刷新信号 fre_flag 为高电平时状态发生跳转，并在相应的状态下进行输出，如数码管扫描电路的状态转移图所示。

为了对数码管扫描显示功能进行验证，需要从拨码开关输入各种显示的数值，根据输入的不同二进制数值，能够在数码管上正确显示。

4.实验步骤

1）启动 Vivado,新建工程文件；

2）编写数码管扫描显示模块；

3）新建 XDC 文件，输入引脚约束；

4）完成综合，实现，生成下载文件；

5）连接开发板 USB 下载线，开启开发板电源；

6）下载 FPGA;

7) 验证结果。

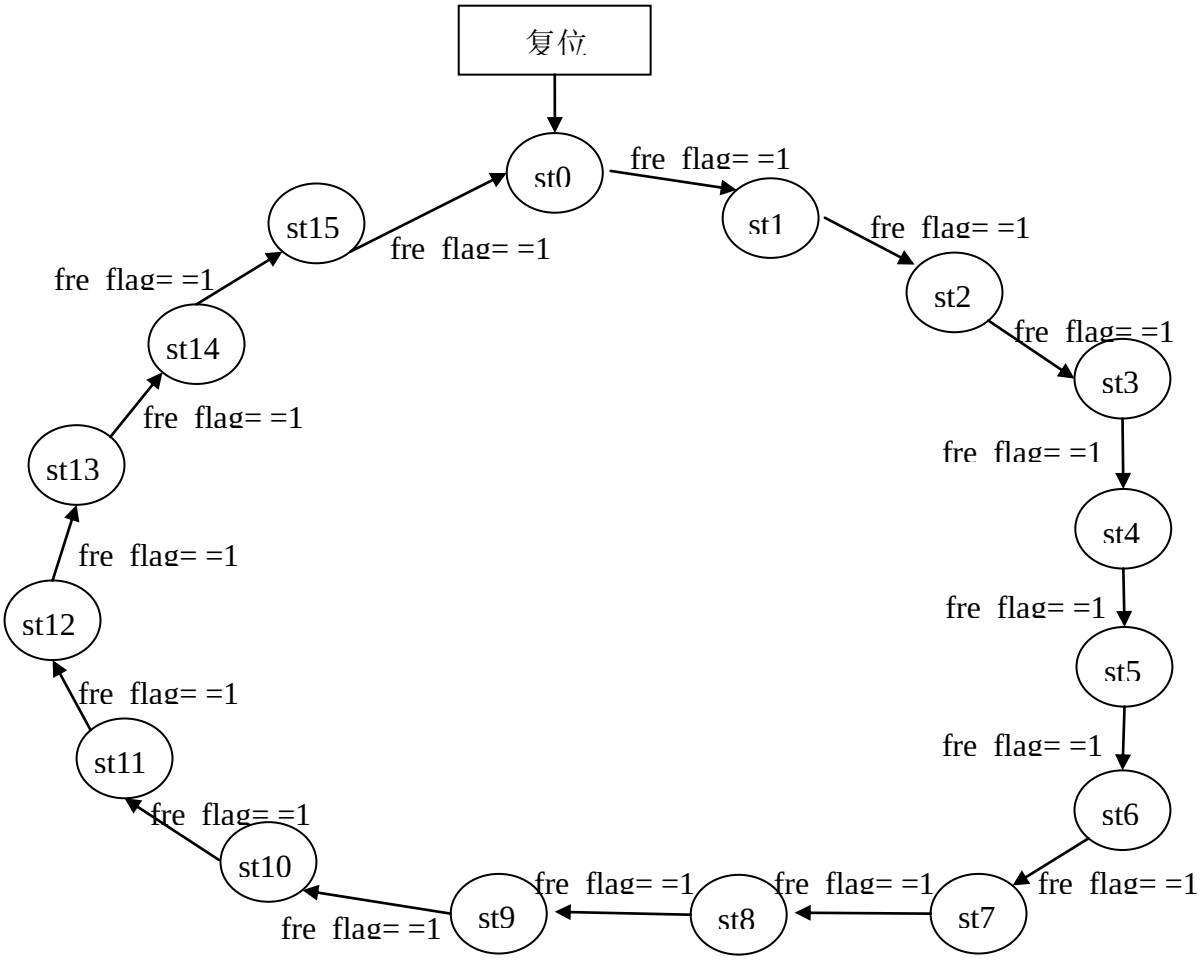


图 12.4 扫描显示状态转移图

实验 13 原码一位乘

1. 实验名称：8 位乘法器设计——原码一位乘

2.实验目的

1）掌握 ISE 开发工具的使用，掌握 FPGA 开发的基本步骤；

2）理解原码一位乘法器的设计思想，用硬件描述语言实现原码一位乘法的算法；

3）掌握程序下载的办法；

4）掌握开发板的使用方法，重点掌握按键，开关，LED 的使用方法。

3.实验内容

（1）用硬件描述语言实现原码一位乘法器

根据原码一位乘的流程图，利用硬件描述语言描述原码乘法器的逻辑实现程序。输入为二进制的原码形式（8 位，含 1 位符号位），输出为乘积的原码形式（16 位，含 1 位符号位）。

原码一位乘法的法则是：

①乘积的符号为被乘数的符号位与乘数的符号位相异或；

②乘积的绝对值为被乘数的绝对值与乘数的绝对值之积。即

[X]原×[Y]原＝(X0 ⊕Y0)(|X|×|Y|)

原码码一位乘法器的顶层模块如图 13.1 所示。

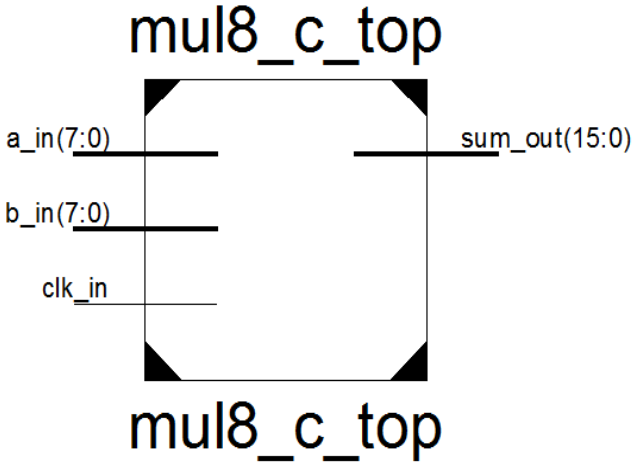


图 13.1 原码一位乘法器顶层模块

a_in，b_in 是数据输入端，sum_out 是运算结果输出端，clk_in 是时钟控制信号。

（2）将程序下载到 FPGA 并进行检验

资源使用要求：用开关(SW0~SW8）输入乘数，被乘数；用 LD0~LD15 显示运算结果。

（3）实验步骤

1）启动 ISE,新建工程文件；

2）编写 8 位原码一位乘模块。

3）编写完加法器模块之后，在顶层文件上实现映射；

4）新建 XDC 文件，输入引脚约束；

5）完成综合，实现，生成下载文件；

6）连接开发板 USB 下载线，开启开发板电源；

7）下载 FPGA;

8）输入数据，验证结果。

实验 14 原码二位乘

1.实验名称：8 位乘法器设计——原码二位乘

2.实验目的

1）掌握 Vivado 开发工具的使用，掌握 FPGA 开发的基本步骤；

2）理解原码两位乘法器的设计思想，用硬件描述语言实现原码两位乘法的算法；

3）掌握程序下载的办法；

4）掌握开发板的使用方法，重点掌握按键，开关，LED 的使用方法。

3.实验内容

（1）用硬件描述语言实现原码两位乘法器

根据原码两位乘的流程图，利用硬件描述语言描述原码两位乘法器的逻辑实现程序。输入为二进制的原码形式（8 位，含 1 位符号位），输出为乘积的原码形式（16 位，含 1 位符号位）。实现思路采用加法和移位的方法。由于两个 8 位二进制原码的乘积应为 15 位（1 位符号位，14 位数值位），为使乘积凑够 16 位，将乘积的末位恒置为 0。

原码两位乘法器的顶层模块如图 14.1 所示。

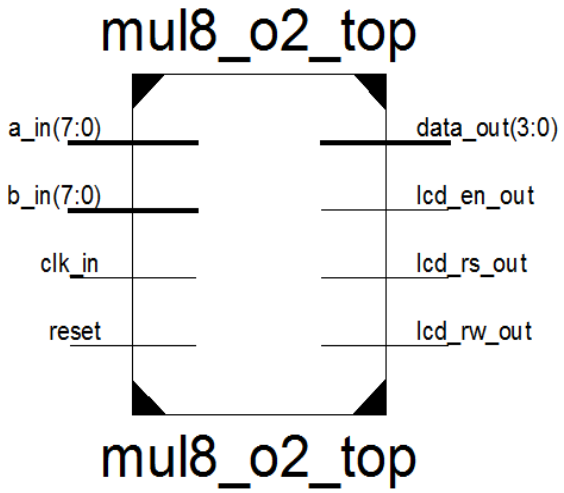


图 14.1 原码两位乘法器顶层模块

a_in，b_in 是数据输入端；reset 信号有效时使能输入数据；lcd_en_out，lcd_rs_out，lcd_rw_out，data_out 是 lcd 的控制信号；clk_in 是时钟输入信号。

（2）将程序下载到 FPGA 并进行检验

资源使用要求：用拨动开关（SW0~SW15）输入乘数，被乘数；用按键开关 BTN0 作为数据输入使能端；用 LED0~LED15 显示运算结果。

（3）实验步骤

1）启动 Vivado,新建工程文件；

2）编写 8 位原码两位乘模块

3）编写完加法器和 LED 显示模块之后，在顶层文件上实现映射；

4）新建 XDC 文件，输入引脚约束；

5）完成综合，实现，生成下载文件；

6）连接开发板 USB 下载线，开启开发板电源；

7）下载 FPGA;

8）输入数据，验证结果。

实验 15 8 位布斯乘法器设计

1.实验名称：补码一位乘

2.实验目的

1）掌握 Vivado 开发工具的使用，掌握 FPGA 开发的基本步骤；

2）理解补码一位乘法器的设计思想，用硬件描述语言实现补码一位乘法的算法；

3）掌握程序下载的办法；

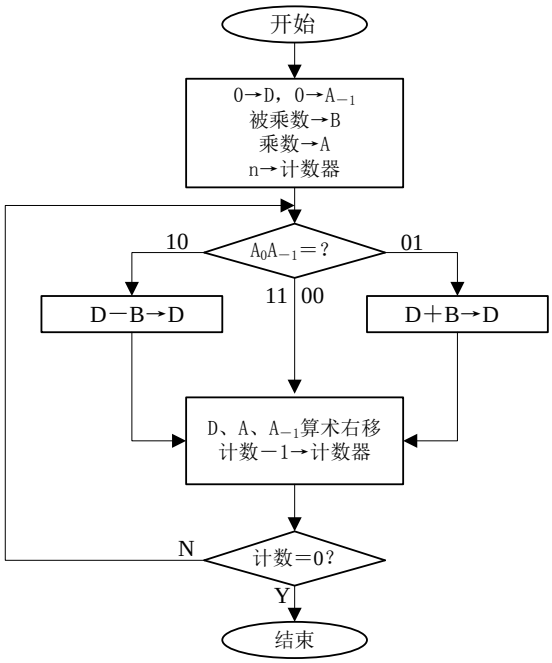
4）掌握开发板的使用方法，重点掌握按键，开关，LED 的使用方法。

3．实验内容

（1）用硬件描述语言实现补码一位乘法器

根据补码一位乘的流程图，利用硬件描述语言描述补码乘法器的逻辑实现程序。输入为二进制的补码形式（8 位，含 1 位符号位），输出为乘积的补码形式（16 位，含 1 位符号位）。可以看到，补码乘法器中符号同数值位一样参与运算，无需单独处理。实现思路采用加法和移位的方法。由于两个 8 位二进制补码的乘积应为 15 位（1 位符号位，14 位数值位），为使乘积凑够 16 位，将乘积的末位恒置为 0。

补码一位乘法器的顶层模块如图 15.2 所示。



布斯补码一位乘法流程图

图 15.1 布斯补码一位乘流程图

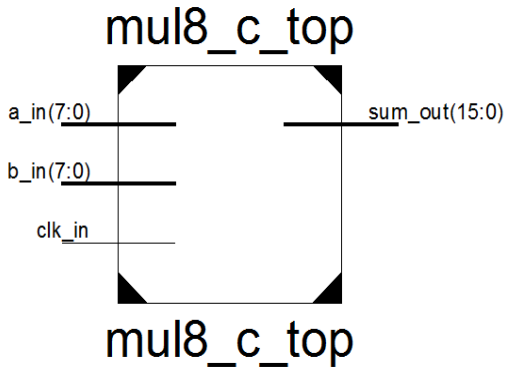


图 15.2 补码一位乘法器顶层模块

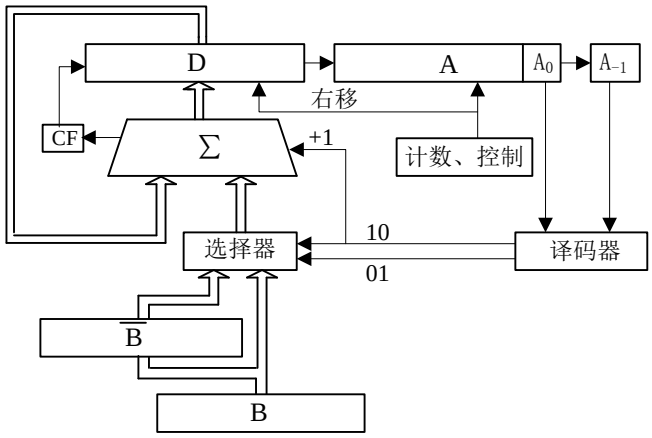


图 15.3 布斯补码一位乘原理图

（2）将程序下载到 FPGA 并进行检验

资源使用要求：用拨动开关(SW0~SW8）输入乘数，被乘数；用数码管显示运算结果。

（3）实验步骤

1）启动 Vivado,新建工程文件；

2）编写 8 位补码一位乘模块。

3）编写完加法器模块之后，在顶层文件上实现映射；

4）新建 XDC 文件，输入引脚约束；

5）完成综合，实现，生成下载文件；

6）连接开发板 USB 下载线，开启开发板电源；

7）下载 FPGA;

8）输入数据，验证结果。

实验 16 阵列乘法器设计

1. 实验名称：阵列乘法器

2.实验目的

1）掌握 Vivado 开发工具的使用，掌握 FPGA 开发的基本步骤；

2）理解阵列乘法器的设计思想，用硬件描述语言实现阵列乘法的算法；

3）掌握程序下载的办法；

4）掌握开发板的使用方法，重点掌握按键，开关，LCD,LED 的使用方法。

3.实验内容：

（1）用硬件描述语言实现阵列乘法器

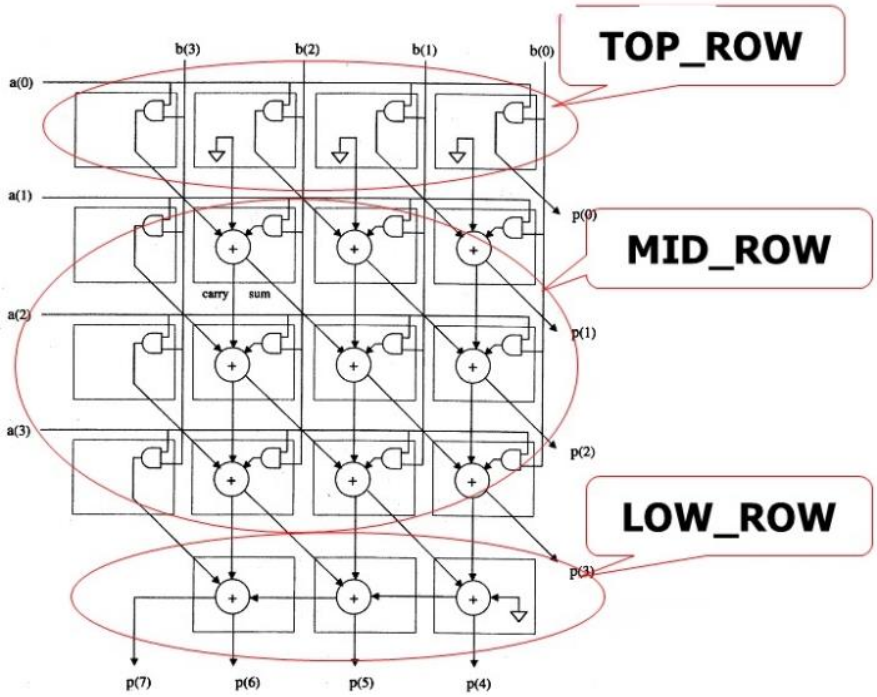


图 16.1 阵列乘法器原理图

根据阵列乘的原理图，利用硬件描述语言描述阵列乘法器的逻辑实现程序。输入为二进制的原码形式（8 位，含 1 位符号位），输出为乘积的原码形式（16 位，含 1 位符号位）。可以看到，阵列乘法器中符号同数值位一样参与运算，无需单独处理。实现思路采用多次加法和移位的方法。

基本乘加单元如图 16.2 所示：

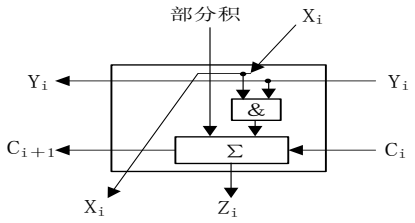


图 16.2 基本乘加单元

阵列乘法器的顶层模块如图 16.3 所示。

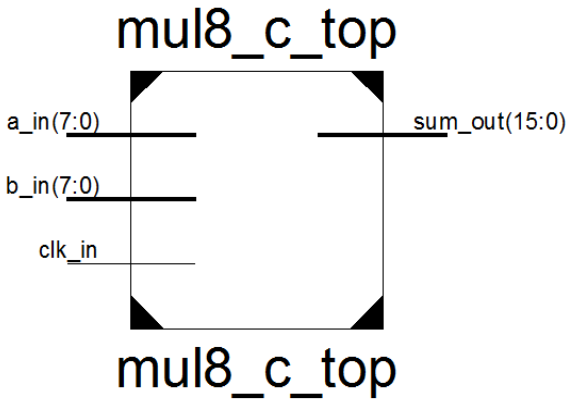


图 16.3 阵列乘法器顶层模块

a_in，b_in 是数据输入端，sum_out 是运算结果输出端，clk_in 是时钟控制信号。

（2）将程序下载到 FPGA 并进行检验

资源使用要求：用拨动开关(SW0~SW8）输入乘数，被乘数；用数码管显示运算结果。

（3）实验步骤：

1）启动 Vivado,新建工程文件；

2）编写 8 位阵列乘模块。

3）编写完加法器模块之后，在顶层文件上实现映射；

4）新建 XDC 文件，输入引脚约束；

5）完成综合，实现，生成下载文件；

6）连接开发板 USB 下载线，开启开发板电源；

7）下载 FPGA;

8）输入数据，验证结果。

实验 17 加减交替除法器

1.实验名称：加减交替除法器

2.实验目的：

1）掌握 Vivado 开发工具的使用，掌握 FPGA 开发的基本步骤；

2）理解加减交替除法器的设计思想，用硬件描述语言实现原码加减交替除法的算法；

掌握程序下载的办法；

4）掌握开发板的使用方法，重点掌握按键，开关，LED 的使用方法。

3.实验内容：

3.1 用硬件描述语言实现加减交替除法器

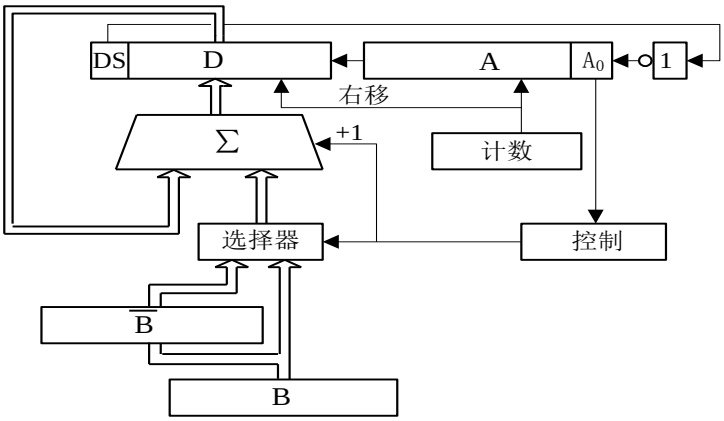


图 17.1 加减交替除法器原理图

根据加减交替除法的原理图，利用硬件描述语言描述加减交替除法器的逻辑实现程序。输入为二进制的原码形式（8 位，含 1 位符号位），输出为乘积的原码形式（16 位，含 1 位符号位）。可以看到，加减交替除法器中符号同数值位一样参与运算，无需单独处理。实现思路采用多次加法、减法和移位的方法。

原码加减交替除法器的运算法则：

1）除法运算前，应满足条件：X*<Y*,且 Y*≠0,否则，按溢出或非法除数处理；

2）符号位不参与运算，单独处理：qf= xf ⊕ yf ;

3）部分余数采用单符号位或双符号位；

4）每步部分余数运算规则：

①若余数 R≥0, 则商上 1，左移一次，减除数；

②若余数 R＜0, 则商上 0，左移一次，加除数。

3.2 将程序下载到 FPGA 并进行检验

资源使用要求：用拨动开关(SW0~SW7)输入除数，被除数；用数码管显示运算结果。

4.实验步骤：

1）启动 Vivado,新建工程文件；

2）编写加减交替除法模块。

3）编写完加减交替除法模块之后，在顶层文件上实现映射；

4）新建 XDC 文件，输入引脚约束；

5）完成综合，实现，生成下载文件；

6）连接开发板 USB 下载线，开启开发板电源；

7）下载 FPGA;

8）输入数据，验证结果。

实验 18 ROM 存储器的设计

1.实验名称：ROM 存储器的设计

2.实验目的：

1）掌握 Vivado 开发工具的使用，掌握 FPGA 开发的基本步骤；

2）理解 ROM 存储器的设计思想，用硬件描述语言实现 ROM 存储器的算法；

3）掌握程序下载的办法；

4）掌握开发板的使用方法，重点掌握按键，开关，数码管的使用方法。

3.实验内容：

3.1 用硬件描述语言实现 ROM 存储器

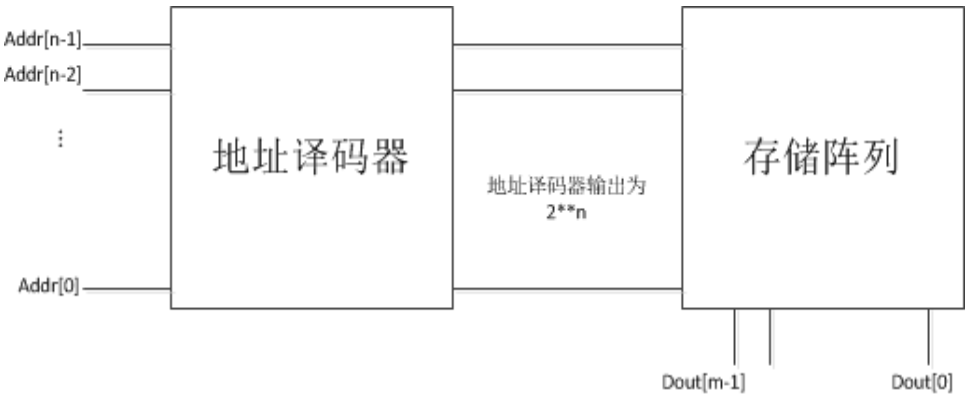


图 18.1 只读存储器 ROM 的电路结构

如上只读存储器 ROM 的电路结构：

1)存储矩阵：由许多存储单元排列而成，而且每个存储单元都被编号，称为地址。

2)地址译码器：它将输入的地址变量译成相应的地址控制信号，该控制信号可将某存储单元从存储矩阵中选出来，并将存储在该单元的信息送至输出缓冲器。

3)输出缓冲器：即输出驱动器，主要实现输出的三态控制。

设计思想：采用二进制译码器的设计方式，将每个输入组态对应的输出与一组存储数据对应起来。

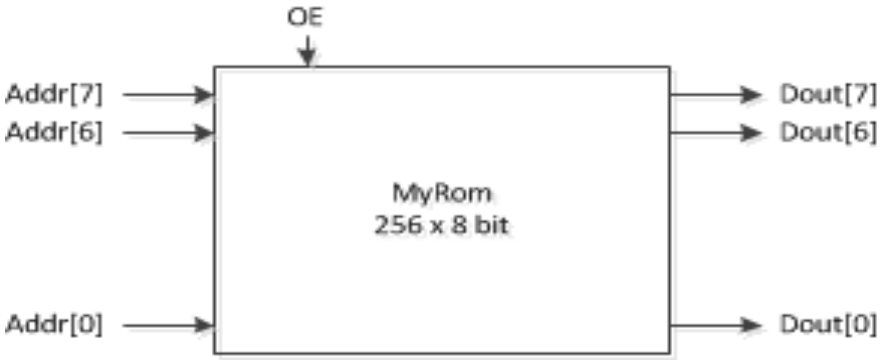


图 18.2 容量为 256*8bit 的 ROM

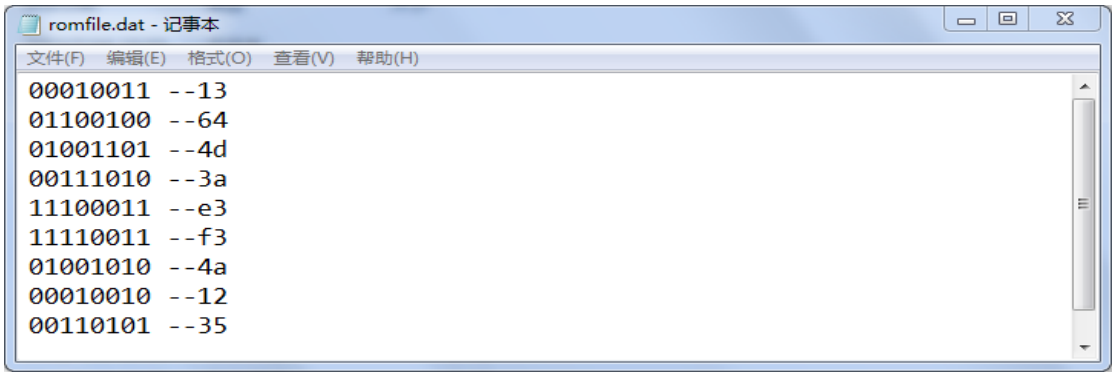


图 18.3　romfile.dat 的数据

从图 18.2 可以看出，端口 Addr 为地址输入端，OE 为输出允许，低有效，Dout 为数据输出，当地址从 00 变化到 09 时，从 MyRom 中读出的数据 Dout 与图 18.3 即 romfile.dat 的数据完全一致，可知所设计的只读存储器的功能满足设计要求。

3.2 将程序下载到 FPGA 并进行检验

资源使用要求：用拨动开关(SW0~SW7)输入地址数据；用数码管显示地址对应数据。

4.实验步骤：

1）启动 Vivado，新建工程文件；

2）编写 rom 模块。

3）添加 romfile.dat 的数据

4）编写完 rom 模块之后，在顶层文件上实现映射；

5）新建 XDC 文件，输入引脚约束；

6）完成综合，实现，生成下载文件；

7）连接开发板 USB 下载线，开启开发板电源；

8）下载 FPGA;

9）输入数据，查看显示结果。

实验 19 先进先出 FIFO 的设计

1.实验名称：FIFO 的设计

2.实验目的:

1)掌握 Vivado 开发工具的使用，掌握 FPGA 开发的基本步骤；

2)理解 FIFO 的设计思想，用硬件描述语言实现 FIFO 的算法；

3)掌握程序下载的办法；

4)掌握开发板的使用方法，重点掌握按键，开关，LCD,LED 的使用方法。

3.实验内容:

3.1 用硬件描述语言实现 FIFO 存储器

FIFO 的原理结构框图如图 19.1 所示，FIFO 由 5 个模块构成，它们是双口 RA M、写地址计数器、读地址计数器、数据满标志、数据空标志。

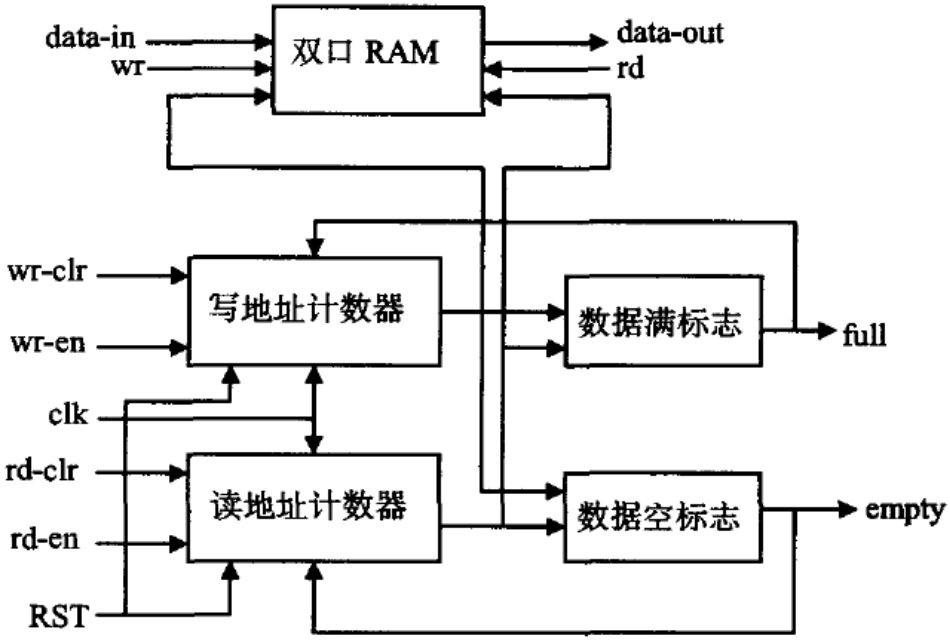


图 19.1 FIFO 的原理结构框图

存入数据按顺序排放，存储器全满时给出信号并拒绝继续存入，全空时也给出信号并拒绝读出；读出时按先进先出原则；存储数据读出就从存储器中消失。

先进先出（First In First Out，FIFO）与普通存储器的区别是没有外部读写地址线，其数据地址由内部读写指针自动加减 1 完成。

FIFO 通常利用双口 RAM 和读写地址产生模块来实现其功能。

不能像普通存储器那样可以由地址线决定读取或写入某个指定的地址。

写地址产生模块一般还根据读地址和写地址来产生 FIFO 的满标志；读地址产生模块一般根据读地址和写地址来产生 FIFO 的空标志。为了实现正确的读写和避免 FIFO 的上溢或下溢，通常还应给出与读时钟和写时钟同步的 FIFO 的空标志(empty)和满标志(ful1)，以禁止读写操作。写地址产生模块通常根据写时钟和写有效信号来产生递增的写地址，而读地址产生模块则根据读时钟和读有效信号来产生递增的读地址。

规则：

- FIFO 为空，不可从 FIFO 读数据，但可写；
- FIFO 为满，不可向 FIFO 写数据，但可读；
- 非空非满时，FIFO 可读、可写。
- FIFO 的读写受同一时钟控制；
- FIFO 的大小为 N。

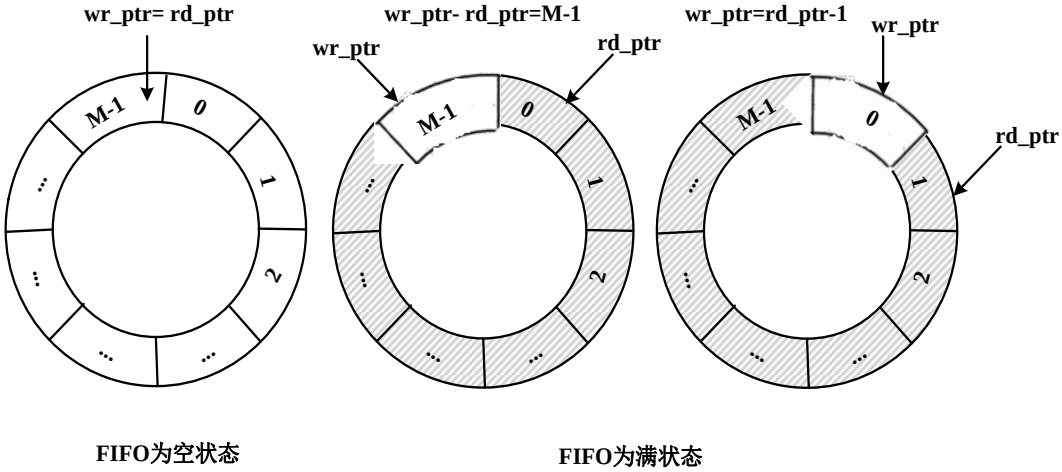


图 19.2 读指针和写指针与 FIFO 状态的关系图

- 当 wr_ptr=rd_ptr 时，FIFO 数据为空；
- 当 wr_ptr-rd_ptr=M-l 或 rd_ptr-wr_ptr=l 时，FIFO 数据为满；
- 当 wr_ptr>=rd_ptr 时，wr_ptr-rd_ptr 为 FIFO 内数据个数；
- 当 wr_ptr<=rd_ptr 时，M-(rd_ptr-wr_ptr)为 FIFO 内数据个数。

3.2 将程序下载到 FPGA 并进行检验

资源使用要求:用拨动开关(SW0~SW7)输入一组数据;用 LED0~LED15 按先进先出顺序显示数据输出。

4.实验步骤:

1）启动 Vivado,新建工程文件；

2）编写 FIFO 模块。

3）编写 rom 模块，用来存储顺序输入的数据

4）编写完 FIFO 和 rom 模块之后，在顶层文件上实现映射；

5）新建 XDC 文件，输入引脚约束；

6）完成综合，实现，生成下载文件；

7）连接开发板 USB 下载线，开启开发板电源；

8）下载 FPGA;

9）顺序输入数据，并顺序输出数据，对比验证结果。

实验 20 时钟模块的设计

1.实验名称：CPU 时钟模块的设计

2.实验目的：

1）掌握 Vivado 开发工具的使用，掌握 FPGA 开发的基本步骤；

2）理解时钟模块的设计思想，用硬件描述语言实现时钟模块的算法；

3）掌握程序下载的办法；

4）掌握开发板的使用方法，重点掌握按键，开关，的使用方法。

3.实验内容：

3.1 用硬件描述语言实现时钟模块

- 一个节拍包含 2 个时钟周期
- 时钟上升沿或下降沿触发

根据系统时钟信号进行分频或取反，来得到反向时钟信号和 2 分频时钟信号以及 2 分频反向时钟信号。并最终得到节拍信号。

波形仿真图如下：

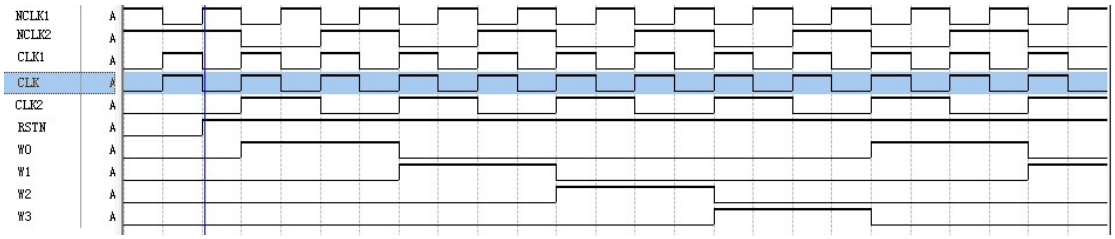


图 20.1 波形仿真图

4.实验步骤：

1）启动 Vivado,新建工程文件；

2）编写时钟模块。

3）编写完时钟模块之后，在顶层文件上实现映射；

4) 输入测试数据，并仿真输出，对比验证结果。

实验 21 PC 程序计数器设计

1.实验名称：PC 程序计数器的设计

2.实验目的：

1）掌握 Vivado 开发工具的使用，掌握 FPGA 开发的基本步骤；

2）理解 PC 模块的设计思想，用硬件描述语言实现 PC 模块的算法；

3）掌握程序下载的办法；

4）掌握开发板的使用方法，重点掌握按键，开关，数码管的使用方法。

3.实验内容：

3.1 用硬件描述语言实现 PC

◆ **PC 功能分析**

加 1 功能、更新地址功能、PC 数值送到数据总线

◆ **PC 功能实现**

1）全局异步复位功能

◆ ADDR<=“000000000000”;

◆ 数据总线高阻态;

2）加 1 功能

◆ clk_PC 上升沿有效;

◆ M_PC 高电平有效，PC+1→ADDR;

3）地址更新功能

4）clk_PC 上升沿有效，nLD_PC 低电平有效，新的 PC→ADDR；

5）PC 数值送到数据总线，nPCH 和 nPCL 低电平有效，注意分两次输出到总线上，先高 8 位后低 8 位。

3.2 将程序下载到 FPGA 并进行检验

资源使用要求： 用拨动开关(SW0~SW7)输入一个 PC 值；用数码管显示 PC 值和 PC 值加 1 以后的值。

4.实验步骤：

1）启动 Vivado,新建工程文件；

2）编写 PC 模块。

3）编写完 PC 模块之后，在顶层文件上实现映射；

4）新建 XDC 文件，输入引脚约束；

5）完成综合，实现，生成下载文件；

6）连接开发板 USB 下载线，开启开发板电源；

7）下载 FPGA;

8）输入数据，并输出数据，对比验证结果。

实验 22 程序存储器 ROM 设计

1.实验名称：程序存储器 ROM 的设计

2.实验目的：

1）掌握 Vivado 开发工具的使用，掌握 FPGA 开发的基本步骤；

2）理解程序存储器 ROM 存储器的设计思想，用硬件描述语言实现 ROM 存储器的算法；

3）掌握程序下载的办法；

4）掌握开发板的使用方法，重点掌握按键，开关，数码管的使用方法。

3.实验内容：

3.1 用硬件描述语言实现程序存储器 ROM 存储器

程序存储器 ROM 的电路结构参考实验 18 的电路结构图，但是要注意本实验与实验 18 通用 ROM 的区别，根据 SOC 设计要求，采用以下的端口定义进行设计 4KB 的 ROM 存储器。

```
entity module_rom is
  port(
      clk_ROM :in std_logic;                          --ROM时钟信号
      M_ROM   :in std_logic;                          --ROM片选信号
      ROM_EN  :in std_logic;                          --ROM使能信号
      addr    :in std_logic_vector(11 downto 0);--ROM地址信号
      data    :inout std_logic_vector(7 downto 0) --数据总线
     );
end module_rom;
```

图 22.2 容量为 4096*8bit 的 ROM 的端口定义

端口 Addr 为地址输入端，OE 为输出允许，低有效，Dout 为数据输出，当地址从 00 变化到 09 时，从 MyRom 中读出的数据 Dout 与图 18.3 即 romfile.dat 的数据完全一致，可知所设计的只读存储器的功能满足设计要求。

3.2 将程序下载到 FPGA 并进行检验

资源使用要求： 用拨动开关(SW0~SW7)输入地址数据；用数码管显示地址对应数据。

4.实验步骤：

1）启动 Vivado,新建工程文件；
2）编写 rom 模块。
3）添加 romfile.dat 的数据
4）编写完 rom 模块之后，在顶层文件上实现映射；
5）新建 XDC 文件，输入引脚约束；
6）完成综合，实现，生成下载文件；
7）连接开发板 USB 下载线，开启开发板电源；
8）下载 FPGA;
9）输入数据，查看显示结果。

实验 23 指令存储器 IR 设计

1.实验名称：指令存储器 IR 的设计

2 实验目的：

1）掌握 Vivado 开发工具的使用，掌握 FPGA 开发的基本步骤；
2）理解指令存储器 IR 的设计思想，用硬件描述语言实现指令存储器 IR 的的算法；
3）掌握程序下载的办法；
4）掌握开发板的使用方法，重点掌握按键，开关，数码管的使用方法。

3.实验内容：

3.1 用硬件描述语言实现指令寄存器 IR 存储器

- **IR 功能分析**
  - 传送指令编码到微控制器
  - 生成 PC 的新地址
  - 生成 RAM 的读写地址
- **IR 功能实现**
  - 传送指令编码到微控制器

    clk_IR 上升沿有效，LD_IR1 高电平有效， data→IR。
  - 寄存器地址操作

    Data[0]→ RS；Data[1]→ RD；
  - 生成 PC 的新地址

clk_IR 上升沿有效，LD_IR2 高电平有效，data[3..0]→PC[11..8];

clk_IR 上升沿有效，LD_IR3 高电平有效，data[7..0]→PC[7..0]。

- 生成 RAM 的读写地址

clk_IR 上升沿有效，LD_IR3 高电平有效 data[7..0]→PC[7..0]; nARen 低电平有效，PC[6..0]→AR[6..0]。

3.2 将程序下载到 FPGA 并进行检验

资源使用要求：用拨动开关(SW0~SW7)输入地址数据；用数码管显示地址对应数据。

4.实验步骤：

1）启动 Vivado,新建工程文件；

2）编写 IR 模块。

3）编写完 IR 模块之后，在顶层文件上实现映射；

4）新建 XDC 文件，输入引脚约束；

5）完成综合，实现，生成下载文件；

6）连接开发板 USB 下载线，开启开发板电源；

7）下载 FPGA;

8）输入数据，验证结果。

实验 24 寄存器 RN 设计

1.实验名称：寄存器 RN 的设计

2.实验目的:

1）掌握 Vivado 开发工具的使用，掌握 FPGA 开发的基本步骤；

2）理解寄存器 RN 的设计思想，用硬件描述语言实现寄存器 RN 的的算法；

3）掌握程序下载的办法；

4）掌握开发板的使用方法，重点掌握按键，开关，数码管的使用方法。

3.实验内容：

3.1 用硬件描述语言实现寄存器 RN 存储器

RN 功能分析：

- 数据锁存功能
- 读写功能
  - ✓ 读寄存器操作

clk_RN 上升沿有效，Ri_EN 低电平有效，读信号 RDRi 高电平有效，选择 RS 寄存器，输出 data[7..0]。

✓ 写寄存器操作

clk_RN 上升沿有效，Ri_EN 低电平有效，写信号 WRRi 高电平有效，选择 RD 寄存器，data[7..0]→ RD 。

3.2 将程序下载到 FPGA 并进行检验

资源使用要求:用拨动开关(SW0~SW7)输入地址数据;用 LED0~LED15 显示地址对应数据。

4.实验步骤：

1）启动 Vivado,新建工程文件；
2）编写 RN 模块。
3）编写完 RN 模块之后，在顶层文件上实现映射；
4）新建 XDC 文件，输入引脚约束；
5）完成综合，实现，生成下载文件；
6）连接开发板 USB 下载线，开启开发板电源；
7）下载 FPGA;
8）输入数据，验证结果。

实验 25 ALU 算术逻辑单元设计

1.实验名称：ALU 算术逻辑单元设计

2.实验目的：

1）掌握 ALU 功能及工作原理；
2）掌握 ALU 与其他模块信号之间的关联性。

3.实验内容：

3.1 ALU 功能与结构

算术逻辑单元（ALU）:执行各种算术和逻辑运算。

- 算术运算操作 :加、减、乘、除
- 逻辑运算操作:与、或、非、异或
- ALU 输入:操作数以及来自控制单元的控制命令
- ALU 输出:运算结果，以及状态信息

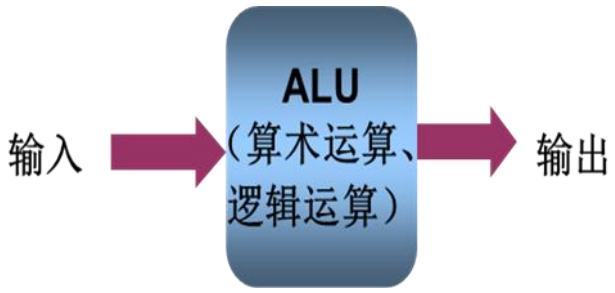


图 25.1　ALU 功能与结构

3.2 基于器件的 8 位 ALU 设计

采用硬件描述语言语言设计 74373 及 74181 模块，并通过相应的接口将两个模块连接在一起形成 8 位 ALU 模块。

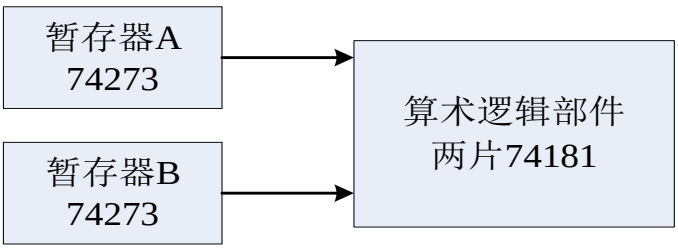


图 25.2　ALU 实现结构

3.3 采用硬件描述语言语言设计 ALU 模块

具体端口定义如下：

```
entity module_ALU is
        port(
                        clk_ALU         :in std_logic;      --ALU时钟信号
                        nreset          :in std_logic;      --全局复位信号
                        M_A,M_B         :in std_logic;      --暂存器控制信号
                        M_F             :in std_logic;      --程序状态字控制信号
                        nALU_EN         :in std_logic;      --ALU运算结果输出使能
                        nPSW_EN         :in std_logic;      --PSW输出使能
                        C0              :in std_logic;      --进位输入
                        S:in std_logic_vector(4 downto 0);  --运算类型和操作选择
                        F_in:in std_logic_vector(1 downto 0); --移位功能选择
                        data:inout std_logic_vector(7 downto 0)    --数据总线
                        AC              :out std_logic;     --半进位标志
                        CY              :out std_logic;     --进位标志
                        ZN              :out std_logic;     --零标志
                        OV              :out std_logic      --溢出标志
                );
end module_ALU;
```

图 25.3　ALU 端口定义

其中 clk_ALU=nclk2，在组建 CPU 模块时使用。

4.实验步骤:

1）启动 Vivado,新建工程文件；

2）编写 74273 模块。

3）编写 74181 模块。

4）编写完 74273 和 74181 模块之后，在顶层文件上实现映射；

5）新建 XDC 文件，输入引脚约束；

6）完成综合，实现，生成下载文件；

7）连接开发板 USB 下载线，开启开发板电源；

8）下载 FPGA;

9）根据 ALU 功能，输入相应运算数据，对比验证结果。

实验 26 数据存储器 RAM 设计

1．实验名称：数据存储器 RAM 设计

2.实验目的：

1）掌握 RAM 功能及工作原理；

2）掌握 RAM 与其他模块信号之间的关联性。

3.实验内容：

3.1 采用硬件描述语言语言设计 RAM 模块

具体端口定义如下：

```
entity module_ram is
  port(
        clk_RAM        :in std_logic;              --RAM时钟信号
        n_reset        :in std_logic;              --RAM选择信号
        RAM_CS         :in std_logic;              --RAM片选信号
        nRAM_EN         :in std_logic;              --RAM输出使能信号
        wr_nRD         :in std_logic;              -- 读写信号
        AR             :in std_logic_vector(6 downto 0); -- RAM地址信号
        data:inout std_logic_vector(7 downto 0)    --数据总线

     );
end module_ram;
```

图 26.1　数据存储器 RAM 端口定义

高电平写操作有效，低电平读有效。其中 clk_RAM=nclk1 & W1，在组建 CPU 模块时使用。

3.2 功能实现

1）读数据操作：clk_RAM 上升沿有效，RAM_CS 高电平，wr_nRD 低电平，nRAM_EN 低电平，[AR] → data 。

2）写数据操作：clk_RAM 上升沿有效，RAM_CS 高电平，wr_nRD 高电平有效，data→[AR]。

4.实验步骤:

1）启动 Vivado,新建工程文件；
2）编写读数据模块和写数据模块；
3）在顶层文件上实现映射；
4）新建 XDC 文件，输入引脚约束；
5）完成综合，实现，生成下载文件；
6）连接开发板 USB 下载线，开启开发板电源；
7）下载 FPGA;
8）根据要求输入验证数据，对比数据输出结果。

实验 27 堆栈指针 SP 设计

1．实验名称：堆栈指针 SP 设计

2.实验目的：

1）掌握 SP 功能及工作原理；
2）掌握 SP 与其他模块信号之间的关联性。

3.实验内容：

3.1 采用硬件描述语言语言设计 SP 模块

具体端口定义如下：

```
entity module_sp is
  port(
       clk_SP       :in std_logic;--SP时钟信号
       nreset       :in std_logic;--复位信号
       SP_CS        :in std_logic;--SP选择信号
       SP_UP        :in std_logic;--SP+1控制
       SP_DN        :in std_logic;--SP-1控制信号
       nSP_EN        :in std_logic;--SP输出使能
       AR           :in std_logic_vector(6 downto 0);--SP指向的RAM地址
       data         :inout std_logic_vector(7 downto 0)--数据总线
     );
end module_sp;
```

图 27.1　SP 端口定义

其中 clk_SP=nclk2，在组建 CPU 模块时使用。

3.2 功能实现

1）数据存储功能：clk_SP 上升沿有效，SP_CS 高电平，，nSP_EN 高电平，data→SP。
2）加 1 功能：clk_SP 上升沿有效，SP_CS 高电平，SP_UP 高电平，nSP_EN 低电平有效，

SP+1→SP,SP→AR。

3）减 1 功能：clk_SP 上升沿有效，SP_CS 高电平，SP_DN 高电平，nSP_EN 低电平有效，SP-1→SP,SP→AR。

4.实验步骤:

1）启动 Vivado,新建工程文件；

2）编写 SP 模块。

3）编写完 SP 模块之后，在顶层文件上实现映射；

4）新建 XDC 文件，输入引脚约束；

5）完成综合，实现，生成下载文件；

6）连接开发板 USB 下载线，开启开发板电源；

7）下载 FPGA;

8）根据要求输入验证数据，对比数据输出结果。

实验 28 IO 端口设计

1．实验名称：IO 端口设计

2.实验目的：

1）掌握 IO 功能及工作原理；

2）掌握 IO 与其他模块信号之间的关联性。

3.实验内容：

3.1 采用硬件描述语言语言设计 IO 模块

具体端口定义如下：

```
entity module_PO is
  port(
      clk_PO          :in std_logic;      --PO时钟信号
      nreset          :in std_logic;      --复位信号
      PO_CS           :in std_logic;      --PO选择信号
      PO_IEN          :in std_logic;      --PO输入使能信号
      PO_OEN          :in std_logic;      --PO输出使能信号
      PO_IN           :in std_logic_vector(7 downto 0);     --PO输入信号
      PO_OUT          :out std_logic_vector(7 downto 0);    --PO输出信号
      data            :inout std_logic_vector(7 downto 0)   --数据总线
     );
end module_PO;
```

图 28.1 IO 模块端口定义

其中 clk_P0=nclk2，在组建 CPU 模块时使用。

3.2 功能实现

1）输入锁存：

clk_PO 上升沿有效，P0_CS 高电平，P0_IEN 低电平，P0_IN→ 暂存器，RIEN 低电平，暂存器 → 数据总线（data）。

2）输出锁存：

clk_PO 上升沿有效，P0_CS 高电平，P0_OEN 低电平，数据总线（data）→ 暂存器，ROEN 低电平，暂存器 →P0_OUT。

4.实验步骤:

1）启动 Vivado,新建工程文件；

2）编写 P0 端口模块。

3）编写完 P0 端口模块之后，在顶层文件上实现映射；

4）新建 XDC 文件，输入引脚约束；

5）完成综合，实现，生成下载文件；

6）连接开发板 USB 下载线，开启开发板电源；

7）下载 FPGA;

8）根据要求输入验证数据，对比数据输出结果。

实验 29 微控制器设计

1．实验名称：微控制器设计

2.实验目的：

1）掌握微控制器功能及工作原理；

2）掌握微控制器与其他模块信号之间的关联性。

3.实验内容：

3.1 微程序控制器基本原理

1)将指令分解为基本的微命令序列，把操作控制信号编制成微指令，存放到控制存储器(CM)。

2)运行时，从控存中取出微指令，产生指令运行所需的操作控制信号。

3.2 微程序控制器基本结构

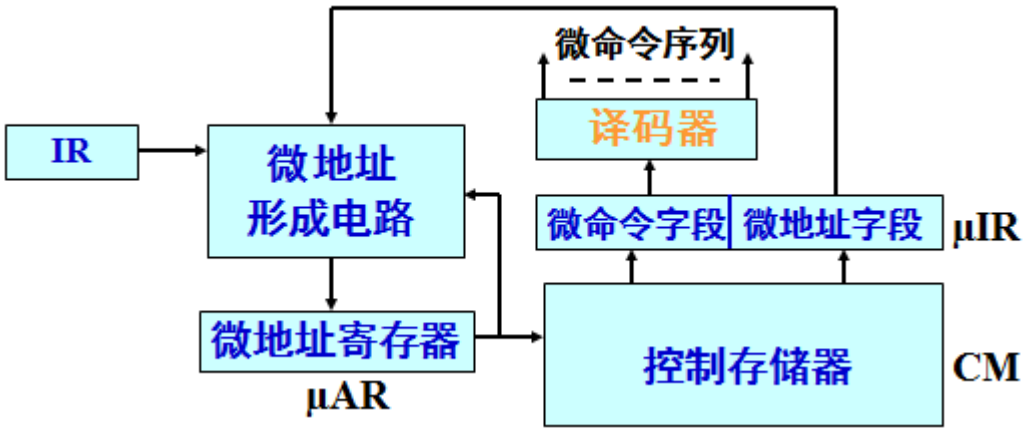


图 29.1 微程序控制器基本结构

- ✓ 控制存储器 CM --存放微程序
- ✓ 微指令寄存器 μIR --存放现行微指令
- ✓ 微地址形成电路--提供下一条微地址
- ✓ 微地址寄存器 μAR--存放现在微地址

3.3 控制信号汇总

- ◆ PC 模块（4 条）
  - LD_PC:in std_logic; --装载新地址
  - M_PC:in std_logic; --PC 加 1 控制信号
  - nPCH,nPCL::in std_logic; --PC 输出总线控制信号
- ◆ ROM 模块（2 条）
  - M_ROM :in std_logic; --ROM 片选信号
  - ROM_EN :in std_logic; --ROM 使能信号
- ◆ IR 模块（4 条）
  - LD_IR1,LD_IR2,LD_IR3 :in std_logic; --IR 指令存储控制信号
  - nARen :in std_logic; --IR 中 RAM 地址控制信号
- ◆ RN 模块（4 条）
  - Ri_CS :in std_logic; --RN 选择信号
  - Ri_EN :in std_logic; --RN 寄存器使能
  - RDRi,WRRi :in std_logic; --RN 读写信号
- ◆ ALU 模块（13 条）
  - M_A,M_B :in std_logic; --暂存器控制信号
  - M_F :in std_logic; --程序状态字控制信号
  - nALU_EN :in std_logic; --ALU 运算结果输出使能

- nPSW_EN :in std_logic; --PSW 输出使能
- C0 :in std_logic; --进位输入
- S:in std_logic_vector(4 downto 0); --运算类型和操作选择
- F_in:in std_logic_vector(1 downto 0); --移位功能选择

◆ RAM 模块（3 条）

- RAM_CS :in std_logic; --RAM 片选信号
- nRAM_EN :in std_logic; --RAM 输出使能信号
- wr_nRD :in std_logic; -- 读写信号

控制信号设计

- 39 条控制信号（39 位编码）
- 27 条指令（5 位编码）→8 位微地址

3.4 采用硬件描述语言语言设计微控制器模块

具体端口定义如下：

```
entity micro_controller is
       port(
              clk_MC          :IN std_logic;          --微程序控制器时钟信号
              nreset          :in std_logic;          --复位信号
              IR              :IN std_logic_vector(7 downto 2);--IR操作码信息
              M_uA            :IN std_logic;          --微地址控制信号
              CMROM_CS        :IN std_logic;          --控制存储器选通信号
              CM:out std_logic_vector(47 downto 0)     --控制信号输出
               );
end micro_controller;
```

其中 clk_MC=clk2 & W0，在组建 CPU 模块时使用。

4.实验步骤:

1）启动 Vivado,新建工程文件；

2）编写控制存储器 CM、微指令寄存器 μIR、微地址形成电路、微地址寄存器 μAR 等功能模块；

3）编写完各个模块之后，在顶层文件上实现映射；

4）新建 XDC 文件，输入引脚约束；

5）完成综合，实现，生成下载文件；

6）连接开发板 USB 下载线，开启开发板电源；

7）下载 FPGA;

8）根据要求输入验证数据，对比数据输出结果。

实验 30　单周期 CPU 设计

1. CPU 结构要求

根据下面 CPU 结构进行单周期 CPU 设计。所设计的单周期 CPU 支持 16 条指令。

2. 实验目的

熟悉 CPU 基本工作原理及执行顺序，设计完成单周期 CPU 功能。

3. 实验内容及要求

1) CPU 基本组成单元连接

2）每条指令能够执行

3）设计一段可执行的程序，并转换为相应的指令码，能够正确执行，并将相应的结果显示在数码管上。

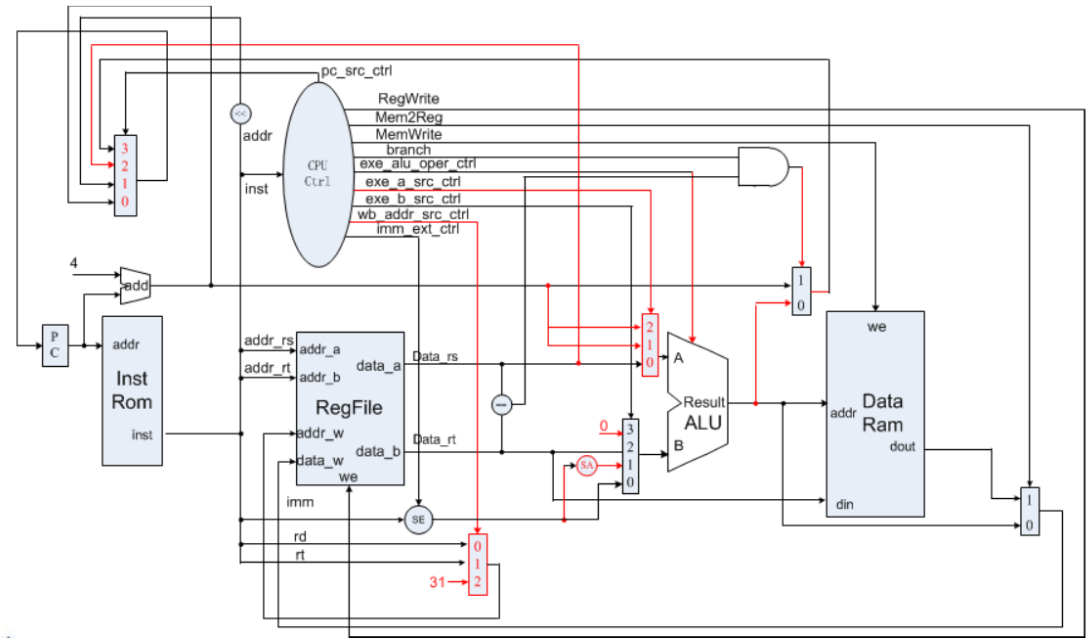


图 30.1　单周期 CPU 结构

4.实验步骤:

1）启动 Vivado,新建工程文件；

2）根据 CPU 结构原理图，连接 CPU 内部各个功能模块；

3）在顶层文件上实现映射；

4）新建 XDC 文件，输入引脚约束；

5）完成综合，实现，生成下载文件；

6）连接开发板 USB 下载线，开启开发板电源；

7）下载 FPGA;

8）编写一段指令程序，运行在所设计 CPU 中，验证相应功能的正确性。

实验 31 多周期 CPU 设计

1. CPU 结构要求

根据下面 CPU 结构进行多周期 CPU 设计。

2. 实验目的

熟悉 CPU 基本工作原理及执行顺序，设计完成多周期 CPU 功能。

3. 实验内容及要求

1) CPU 基本组成单元连接

2）每条指令能够执行

3）设计一段可执行的程序，并转换为相应的指令码，能够正确执行，并将相应的结果显示在数码管上。

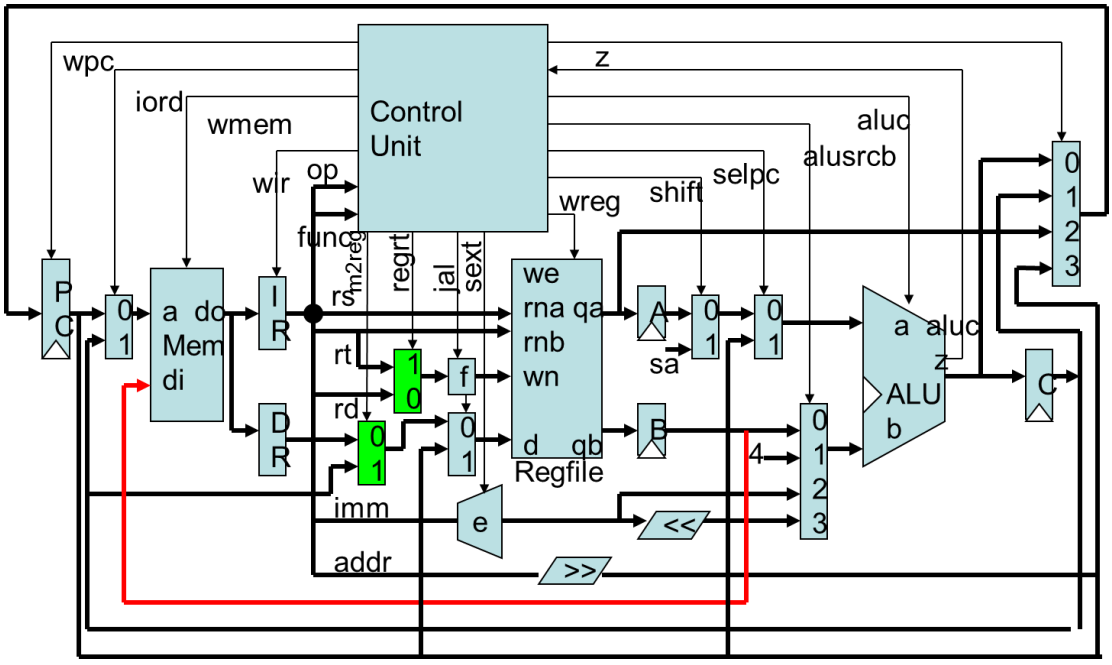


图 31.1 多周期 CPU 结构

4.实验步骤:

1）启动 Vivado,新建工程文件；

2）根据 CPU 结构原理图，连接 CPU 内部各个功能模块；

3）在顶层文件上实现映射；

4）新建 XDC 文件，输入引脚约束；

5）完成综合，实现，生成下载文件；

6）连接开发板 USB 下载线，开启开发板电源；

7）下载 FPGA;

8）编写一段指令程序，运行在所设计 CPU 中，验证相应功能的正确性。

实验 32 8 位 SOC 综合设计

1. SOC 结构要求

根据以上所设计的功能模块连接成完整的 8 位 SOC 结构。

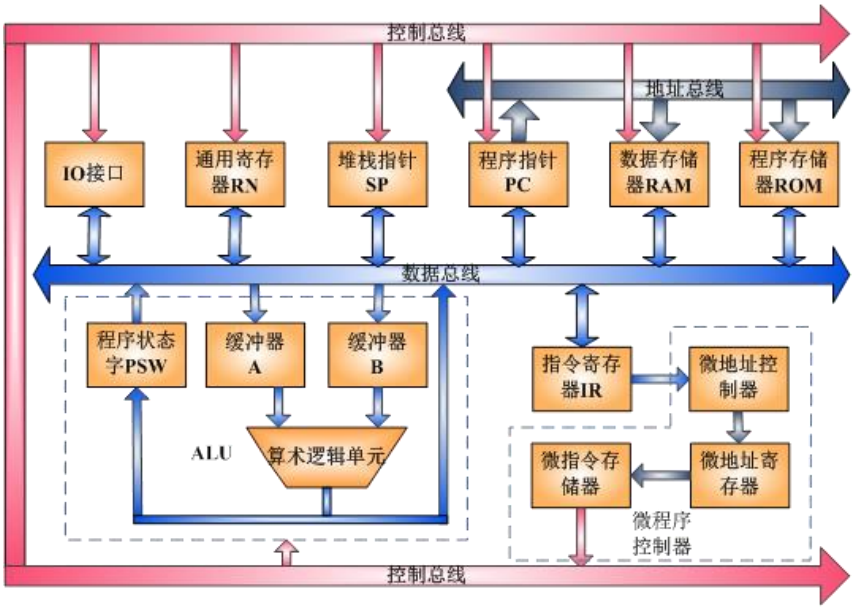


图 32.1 SOC 结构图

2. 实验目的

熟悉 CPU 基本工作原理及执行顺序，设计完成 8 位 SOC 功能。

3. 实验内容及要求

1) SOC 基本组成单元连接

2）每条指令能够执行

3）设计一段可执行的程序，并转换为相应的指令码，能够正确执行，并将相应的结果显示在数码管上。

4.实验步骤

1）启动 ISE,新建工程文件；

2）根据 SOC 结构原理图，连接 SOC 内部各个功能模块；

3）在顶层文件上实现映射；

4）新建 XDC 文件，输入引脚约束；

5）完成综合，实现，生成下载文件；

6）连接开发板 USB 下载线，开启开发板电源；

7）下载 FPGA;

8）编写一段流水灯的指令程序，运行在所设计 SOC 中，验证相应功能的正确性。