

《半定制器件原理与应用》课程设计



设计报告书

题目名称: PS2 人机输入显示系统

所在学院: XXXXXXXXXX 学院

学生姓名: XXXXXXXXXXXXXXXX

指导老师: XXXXXXXXXX

(2019年01月)

目录

一、 课程设计的目的与任务	错误!未定义书签。
二、 课程设计题目	错误!未定义书签。
1、 指定题目:	错误!未定义书签。
2、 自选题目:	错误!未定义书签。
三、 课程设计的内容与要求	错误!未定义书签。
1、 设计内容.....	3
2、 设计要求.....	4
四、 实验仪器设备	4
五、 设计方案	4
1、 PS2 解码	4
2、 设计思路.....	6
3、 模块设计.....	6
4、 各模块分析.....	7
(1) PS2 时钟检测模块	7
(2) PS2 解码模块	9
(3) PS2 组合模块	11
(4) 控制 LED 模块.....	13
(5) PS2 总的组合模块	15
六、 综合与仿真	16
1、 综合.....	16
2、 仿真.....	17
(1) 电平检测模块仿真.....	17
(2) LED 灯控制模块仿真.....	18
(3) PS2_module 总模块仿真	18
七、 硬件下载	21
八、 心得体会	23
九、 参考文献	23

一、课程设计概述

1.1 课程设计的目的与任务

- (1) 熟练掌握 EDA 工具软件 QuartusII 的使用；
- (2) 熟练用 Verilog HDL 硬件描述语言描述数字电路；
- (3) 学会使用 Verilog HDL 进行大规模集成电路设计；
- (4) 学会用 CPLD\FPGA 使用系统硬件验证电路设计的正确性；
- (5) 初步掌握 EDA 技术并具备一定的可编程逻辑芯片的开发能力；

1.2 课程设计题目

人机输入显示系统：PS/2 键盘输入， LCD 显示

1.3 设计功能要求

- (1) 实现 PS/2 键盘的输入
- (2) 将键盘输入的内容显示在液晶显示器上；
- (3) 键盘的功能实现自定，例如是否考虑功能键，是否使用连击等；
- (4) 显示格式自定，例如是否使用多行，是否使用大小写等。

1.4 设计实现提示

方法是将 PS/2 模块与 LCD 模块连接

- (1) 问题是数据如何传递？
- (2) 利用 LCD 数据存储器？
- (3) 使用 RAM 宏模块？

1.5 课程设计的内容与要求

1.5.1 设计内容

- (1) 系统功能的分析；
- (2) 实现系统功能的实际方案；
- (3) 编写各功能模块的 VHDL 语言程序；
- (4) 对各功能模块进行编译、综合、仿真、分析；
- (5) 顶层文件设计
- (6) 对整个系统进行编译、综合、仿真、分析；
- (7) 在 CPLD\FPGA 实验开发系统试验箱上进行硬件验证；

(8) 写课程设计报告；

1.5.1 设计要求

- (1) 按所布置的题目要求，每一位学生独立完成全过程；
- (2) 分模块层次化设计；
- (3) 各功能模块的底层文件必须用 VHDL 语言设计，顶层文件可用 VHDL 语言设计，也可以用原理图设计。

二、实验仪器设备

- (1) PC 机；
- (2) QuartusII 软件；
- (3) 黑金 FPGA 实验开发系统，芯片为 Cyclone II 的 EP2C5Q208C8；

三、设计方案

1、PS2 解码

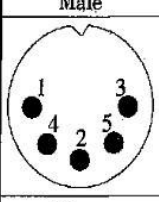
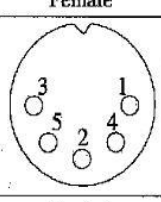
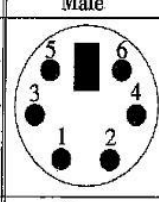
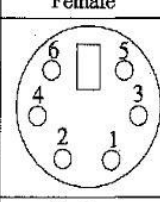
Male	Female	5-pin DIN	Male	Female	6-pin Mini-DIN
		1-时钟			1-数据
		2-数据			2-保留
		3-保留			3-电源地
		4-电源地			4-电源+5V
Plug	Socket	5-电源+5V	Plug	Socket	5-时钟
					6-保留

图 1 PS/2 接口连接器定义

图 1 为 PS2 的接口图。我使用的右边的 PS2 接口，即 1 脚为数据脚，5 脚为时钟脚，同时我编写的 VHDL 代码只对 1 脚和 5 脚操作。

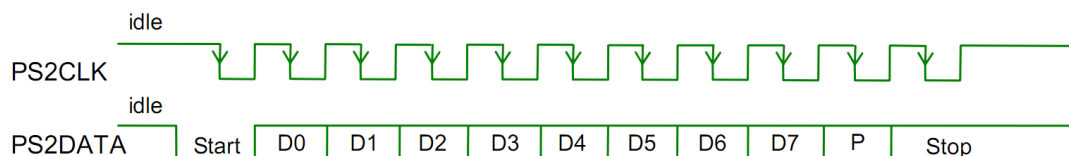


图 2 PS2 协议时序图

图 2 为 PS2 协议时序图。由图可以解读出，PS2 协议对数据的读取时“CLK 的下降沿”有效，而数据的放置时在“CLK 的上升沿”。PS2 频率比较慢，大概为 10KHz。

第 N 位	属性
0	开始位
1~8	数据位
9	校验位
10	结束位

表 1 PS2 数据说明

PS2 的一帧数据时 11 位。对 PS2 进行解码，我们需要得到的是 1~8 位的数据位。其他的位，可以使用取巧的方法编写。

键盘的编码有“通码（Make）”和“断码（Break）”之分。通码相当于某个按键按下了，断码相当于某个按键释放了。假设，我们按下了“Z”键不放，大约每秒有 10 个 X “1A”的通码（10KHz），而当我们释放“Z”键，就会输出断码 X “F0”和 X “1A”。同时，键盘编码一次只能有一个输出，即多个按键同时按下时，只有一个有效。

下表为第二套 PC 键盘扫描码。

键 名	通 码	断 码	-	键 名	通 码	断 码	-	键 名	通 码	断 码
A	1C	F0, 1C		9	46	F0, 46		[54	F0, 54
B	32	F0, 32		`	0E	F0, 0E		INSERT	E0, 70	E0, F0, 70
C	21	F0, 21		-	4E	F0, 4E		HOME	E0, 6C	E0, F0, 6C
D	23	F0, 23		=	55	F0, 55		PG UP	E0, 7D	E0, F0, 7D
E	24	F0, 24		\	5D	F0, 5D		DELETE	E0, 71	E0, F0, 71
F	2B	F0, 2B		BKSP	66	F0, 66		END	E0, 69	E0, F0, 69
G	34	F0, 34		SPACE	29	F0, 29		PG DN	E0, 7A	E0, F0, 7A
H	33	F0, 33		TAB	0D	F0, 0D		U ARROW	E0, 75	E0, F0, 75
I	43	F0, 43		CAPS	58	F0, 58		L ARROW	E0, 6B	E0, F0, 6B
J	3B	F0, 3B		L SHFT	12	F0, 12		D ARROW	E0, 72	E0, F0, 72
K	42	F0, 42		L CTRL	14	F0, 14		R ARROW	E0, 74	E0, F0, 74
L	4B	F0, 4B		L GUI	E0, 1F	E0, F0, 1F		NUM	77	F0, 77
M	3A	F0, 3A		L ALT	11	F0, 11		KP /	E0, 4A	E0, F0, 4A
N	31	F0, 31		R SHFT	59	F0, 59		KP *	7C	F0, 7C
O	44	F0, 44		R CTRL	E0, 14	E0, F0, 14		KP -	7B	F0, 7B
P	4D	F0, 4D		R GUI	E0, 27	E0, F0, 27		KP +	79	F0, 79
Q	15	F0, 15		R ALT	E0, 11	E0, F0, 11		KP EN	E0, 5A	E0, F0, 5A
R	2D	F0, 2D		APPS	E0, 2F	E0, F0, 2F		KP .	71	F0, 71

S	1B	F0, 1B		ENTER	5A	F0, 5A		KP 0	70	F0, 70
T	2C	F0, 2C		ESC	76	F0, 76		KP 1	69	F0, 69
U	3C	F0, 3C		F1	05	F0, 05		KP 2	72	F0, 72
V	2A	F0, 2A		F2	06	F0, 06		KP 3	7A	F0, 7A
W	1D	F0, 1D		F3	04	F0, 04		KP 4	6B	F0, 6B
X	22	F0, 22		F4	0C	F0, 0C		KP 5	73	F0, 73
Y	35	F0, 35		F5	03	F0, 03		KP 6	74	F0, 74
Z	1A	F0, 1A		F6	0B	F0, 0B		KP 7	6C	F0, 6C
0	45	F0, 45		F7	83	F0, 83		KP 8	75	F0, 75
1	16	F0, 16		F8	0A	F0, 0A		KP 9	7D	F0, 7D
2	1E	F0, 1E		F9	01	F0, 01]	5B	F0, 5B
3	26	F0, 26		F10	09	F0, 09		;	4C	F0, 4C
4	25	F0, 25		F11	78	F0, 78		'	52	F0, 52
5	2E	F0, 2E		F12	07	F0, 07		,	41	F0, 41
6	36	F0, 36		PRNT SCRN	E0, 12, E0, 7C	E0, F0, 7C, E0, F0, 12		.	49	F0, 49
7	3D	F0, 3D		SCROLL	7E	F0, 7E		/	4A	F0, 4A
8	3E	F0, 3E		PAUSE	E1, 14, 77, E1, F0, 14, F0, 77	-NONE-				

表 2 PC 键盘第二套扫描码

2、设计思路

- (1) PS2 时钟的检测；
- (2) PS2 数据的接受并提取需要的 8 位数据；
- (3) 对 PS2 提取的 8 位数据进行解码，确定按键；
- (4) 通过 LED 灯显示按键的解码的结果；
- (5) 设置多个按键，多种 LED 显示方式；

对于 PS2 键盘扫描程序，我的设计思路是一个模块一个功能，这样能清晰分辨模块，同时易于修改代码。代码条理清晰，便于解读。而对于多个模块则使用层次化的形式来编写，顶层文件并不包含功能的设定，只包含各个子功能模块。

3、模块设计

PS2 键盘扫描分为：电平检测，PS2 解码，PS2 组合，LED 控制和总 PS 组合六个模块。下面为各个模块的简易模块图。

(1) PS2 时钟检测模块:

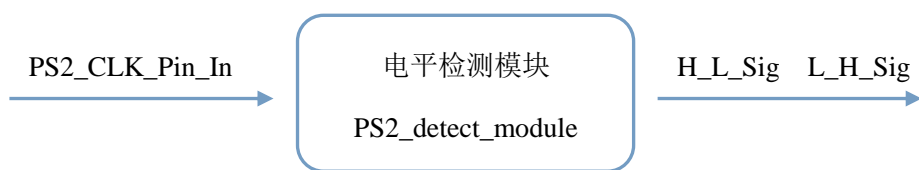


图 3 电平检测模块图

(2) PS2 解码模块:

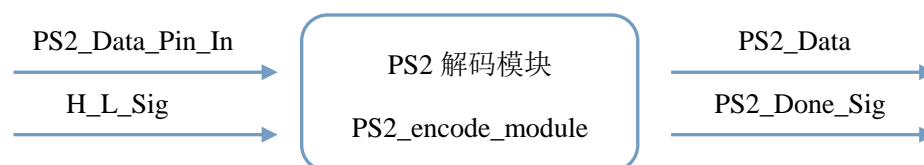


图 4 PS2 解码模块图

(3) PS2 组合模块:

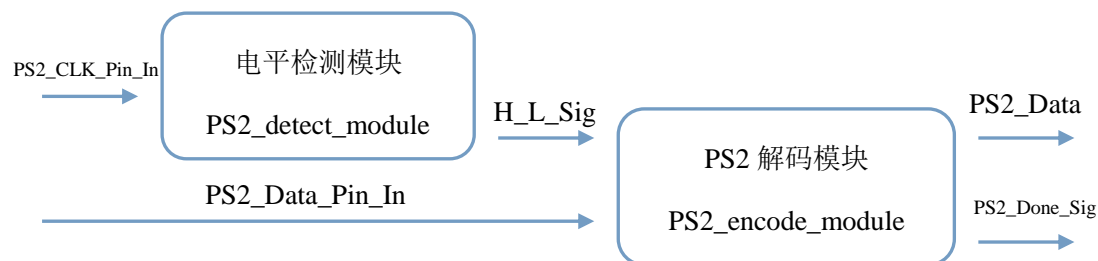


图 5 PS2 组合模块图

(4) 控制 LED 模块:

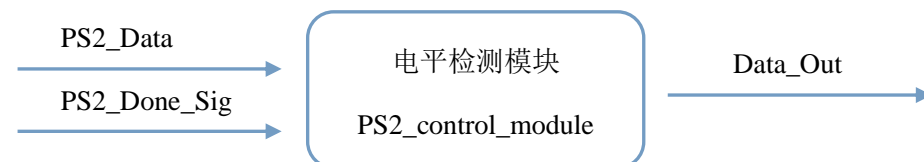


图 6 LED 控制模块图

(5) PS2 总的组合模块:

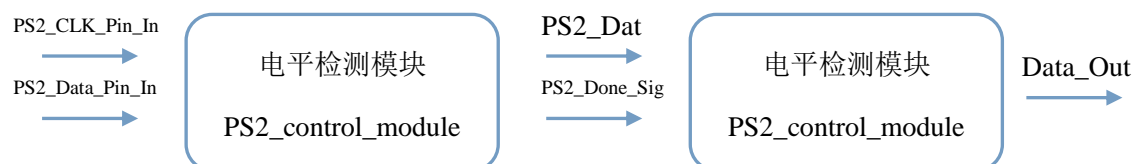


图 7 PS2 模块图

4、各模块分析

(1) PS2 时钟检测模块

PS2 电平检测模块主要的作用是检测 PS2 接口键盘的时钟信号，因为 PS2

的协议规定数据是在时钟的下降沿读取的。所以电平检测模块要检测 PS2 时钟的下降沿，有下降沿来临时，要做相应的数据读取动作。下面是代码的分析。

```

LIBRARY IEEE;                                --库

USE IEEE.STD_LOGIC_1164.ALL;

USE IEEE.STD_LOGIC_UNSIGNED.ALL;

-----

ENTITY PS2_detect_module IS                  --实体声明

    PORT(

        CLK,RSTn      : IN STD_LOGIC;

        PS2_CLK_Pin_In  : IN STD_LOGIC;

        H_L_Sig        : OUT STD_LOGIC;      --电平由高变低，输出一个信号

        L_H_Sig        : OUT STD_LOGIC      --电平由低变高，输出一个信号

    );

END ENTITY PS2_detect_module;

-----

ARCHITECTURE PS2_detect OF PS2_detect_module IS    --结构体声明

    SIGNAL H_L_F1 : STD_LOGIC := '1';              --声明 4 个信号，用于电平输入的变化

    SIGNAL H_L_F2 : STD_LOGIC := '1';              --4 个信号都赋了初值

    SIGNAL L_H_F1 : STD_LOGIC := '0';

    SIGNAL L_H_F2 : STD_LOGIC := '0';

    BEGIN

        PROCESS(CLK,RSTn)

            BEGIN

                IF (CLK'event AND CLK='1') THEN    --同步进行

                    IF (RSTn='0') THEN            --同步复位动作

                        H_L_F1 <= '1';H_L_F2 <= '1';L_H_F1 <= '0';L_H_F2 <= '0';

                    ELSE

                        H_L_F1 <= PS2_CLK_Pin_In;H_L_F2 <= H_L_F1;

                        L_H_F1 <= PS2_CLK_Pin_In;L_H_F2 <= L_H_F1;

                    END IF;

                END IF;
            END IF;
        END PROCESS;
    END ARCHITECTURE PS2_detect;

```



```
END IF;

END PROCESS;

H_L_Sig <= H_L_F2 AND (NOT H_L_F1);    --输出信号

L_H_Sig <= L_H_F1 AND (NOT L_H_F2);

END ARCHITECTURE PS2_detect;           --结构体结束
```

在结构体中声明了 4 个信号，用于电平的检测 F2 信号是接着 F1 信号的，如果 F1 信号变化了，F2 信号还不会立即变化，F2 还会保持 F1 的前一个状态，以两者的逻辑关系，可以判断输入的是上升沿还是下降沿。结果如表格 3。

时间	H_L_F1	H_L_F2	H_L_Sig <= H_L_F2 AND (NOT H_L_F1)
Initial	1	1	0
T1	0	1	1
T2	0	0	0
时间	L_H_F1	L_H_F2	L_H_Sig <= L_H_F1 AND (NOT L_H_F2);
Initial	0	0	0
T1	1	0	1
T2	1	1	0

表 3 电平检测变化表

(2) PS2 解码模块

```
LIBRARY IEEE;                                --库

USE IEEE.STD_LOGIC_1164.ALL;

USE IEEE.STD_LOGIC_UNSIGNED.ALL;

-----

ENTITY PS2_decode_module IS                    --实体声明

    PORT(

        CLK,RSTn          : IN STD_LOGIC;

        H_L_Sig           : IN STD_LOGIC;

        PS2_Data_Pin_In   : IN STD_LOGIC;

        PS2_Done_Sig      : OUT STD_LOGIC;

        PS2_Data          : OUT STD_LOGIC_VECTOR(7 DOWNT0 0)
```

```

);

END ENTITY PS2_decode_module;

-----

ARCHITECTURE PS2_decode OF PS2_decode_module IS

    SIGNAL Done : STD_LOGIC := '0';                                --声明一个完成信号

    SIGNAL i      : STD_LOGIC_VECTOR(4 DOWNTO 0) := "00001";      --声明步骤 i

    SIGNAL Data : STD_LOGIC_VECTOR(7 DOWNTO 0) := X"32";

BEGIN

    PROCESS(CLK,RSTn,i)

    BEGIN

        IF (CLK'event AND CLK='1') THEN

            IF (RSTn='0') THEN

                i <= "00001";Done <= '0';Data <= X"00";

            ELSE

                CASE i IS

                    WHEN "00000" => i <= "00001";

                    WHEN "00001" => IF (H_L_Sig='1') THEN i <= "00010";Data(0) <= PS2_Data_Pin_In;END IF;

                    WHEN "00010" => IF (H_L_Sig='1') THEN i <= "00011";Data(1) <= PS2_Data_Pin_In;END IF;

                    WHEN "00011" => IF (H_L_Sig='1') THEN i <= "00100";Data(2) <= PS2_Data_Pin_In;END IF;

                    WHEN "00100" => IF (H_L_Sig='1') THEN i <= "00101";Data(3) <= PS2_Data_Pin_In;END IF;

                    WHEN "00101" => IF (H_L_Sig='1') THEN i <= "00110";Data(4) <= PS2_Data_Pin_In;END IF;

                    WHEN "00110" => IF (H_L_Sig='1') THEN i <= "00111";Data(5) <= PS2_Data_Pin_In;END IF;

                    WHEN "00111" => IF (H_L_Sig='1') THEN i <= "01000";Data(6) <= PS2_Data_Pin_In;END IF;

                    WHEN "01000" => IF (H_L_Sig='1') THEN i <= "01001";Data(7) <= PS2_Data_Pin_In;END IF;

                    WHEN "01001" => IF (H_L_Sig='1') THEN i <= "01010";END IF;

                    WHEN "01010" => IF (H_L_Sig='1') THEN i <= "01011";END IF;

                    WHEN "01011" => IF (Data=X"F0") THEN i <= "01100";ELSE i <= "10110";END IF;

                    WHEN "01100" => IF (H_L_Sig='1') THEN i <= "01101";END IF;

                    WHEN "01101" => IF (H_L_Sig='1') THEN i <= "01110";END IF;

                    WHEN "01110" => IF (H_L_Sig='1') THEN i <= "01111";END IF;

```

```

    WHEN "01111" => IF (H_L_Sig='1') THEN i <= "10000";END IF;

    WHEN "10000" => IF (H_L_Sig='1') THEN i <= "10001";END IF;

    WHEN "10001" => IF (H_L_Sig='1') THEN i <= "10010";END IF;

    WHEN "10010" => IF (H_L_Sig='1') THEN i <= "10011";END IF;

    WHEN "10011" => IF (H_L_Sig='1') THEN i <= "10100";END IF;

    WHEN "10100" => IF (H_L_Sig='1') THEN i <= "10101";END IF;

    WHEN "10101" => IF (H_L_Sig='1') THEN i <= "10110";END IF;

    WHEN "10110" => IF (H_L_Sig='1') THEN i <= "10111";Done <= '1';END IF;

    WHEN "10111" => IF (H_L_Sig='1') THEN i <= "00001";Done <= '0';END IF;

    WHEN OTHERS => i <= "00001";

        END CASE;

    END IF;

END IF;

END PROCESS;

PS2_Data <= Data;

PS2_Done_Sig <= Done;

END ARCHITECTURE PS2_decode;

```

这个模块我有点偷懒，只对键盘输入的 8 位有效数据进行了提取，其他位基本是忽略了，第一位开始位忽略了，然后是读取 8 位有效数据，第 9 步和第 10 步跳过了检测位和结束位，然后是判断。如果是 0XF0，则证明是断码，断码的话后面的直接跳过，如果不是 0XF0，则证明是有效的数据，立即跳到步骤 22，向顶层的模块回馈一个完成信号，并将有效数据输出。

（3）PS2 组合模块

```

LIBRARY IEEE;

USE IEEE.STD_LOGIC_1164.ALL;

USE IEEE.STD_LOGIC_UNSIGNED.ALL;

```

```

ENTITY PS2 IS

```

```

    PORT(

        CLK,RSTn        : IN STD_LOGIC;

```

```
        PS2_Data_Pin_In : IN STD_LOGIC;

        PS2_CLK_Pin_In  : IN STD_LOGIC;

        PS2_Done_Sig    : BUFFER STD_LOGIC;

        PS2_Data        : OUT STD_LOGIC_VECTOR(7 DOWNT0 0)

    );

END ENTITY PS2;

-----

ARCHITECTURE PS2_behave OF PS2 IS

    COMPONENT PS2_detect_module

        PORT(

            CLK,RSTn      : IN STD_LOGIC;

            PS2_CLK_Pin_In : IN STD_LOGIC;

            H_L_Sig       : OUT STD_LOGIC;

            L_H_Sig       : OUT STD_LOGIC

        );

    END COMPONENT;

    COMPONENT PS2_decode_module

        PORT(

            CLK,RSTn      : IN STD_LOGIC;

            H_L_Sig       : IN STD_LOGIC;

            PS2_Data_Pin_In : IN STD_LOGIC;

            PS2_Done_Sig   : OUT STD_LOGIC;

            PS2_Data       : OUT STD_LOGIC_VECTOR(7 DOWNT0 0)

        );

    END COMPONENT;

--    COMPONENT PS2_code_module

--        PORT(

--            CLK,RSTn      : IN STD_LOGIC;

--            L_H_Sig       : IN STD_LOGIC;
```

```

--          PS2_Done_Sig      : IN STD_LOGIC;

--          PS2_Data_Pin_Out : OUT STD_LOGIC

--      );

--  END COMPONENT;

  SIGNAL H_L : STD_LOGIC;

--  SIGNAL L_H : STD_LOGIC;

--  SIGNAL PS2_Data_Pin_Out : STD_LOGIC;

BEGIN

  U1:PS2_detect_module PORT MAP(CLK,RSTn,PS2_CLK_Pin_In,H_L,L_H);

  U2:PS2_decode_module PORT MAP(CLK,RSTn,H_L,PS2_Data_Pin_In,PS2_Done_Sig,PS2_Data);

--  U2:PS2_decode_module PORT MAP(CLK,RSTn,H_L,PS2_Data_Pin_Out,PS2_Done_Sig,PS2_Data);

--  U3:PS2_code_module   PORT MAP(CLK,RSTn,L_H,PS2_Done_Sig,PS2_Data_Pin_Out);

END ARCHITECTURE PS2_behave;

```

这是一个组合例化的模块，是对 PS2 时钟电平检测和 PS2 解码的一个简单模块。这一个模块初步实现了 PS2 的解码功能。上面的是代码的方法实现例化功能，同时也可以使用原理图的方式来实现例化，下面为原理图例化的图。

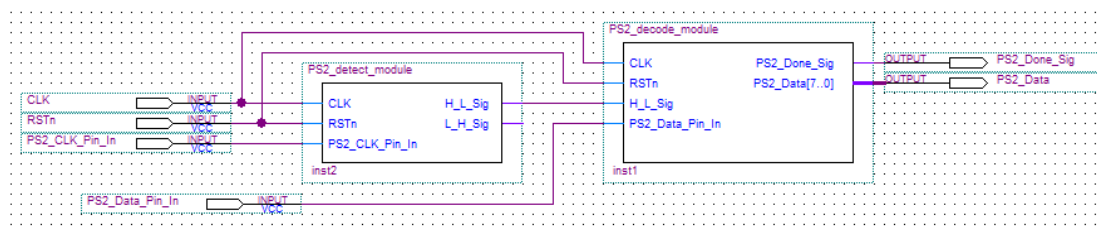


图 8 PS2 例化原理图

(4) 控制 LED 模块

```

LIBRARY IEEE;

USE IEEE.STD_LOGIC_1164.ALL;

USE IEEE.STD_LOGIC_UNSIGNED.ALL;

```

```

ENTITY PS2_contorl_module IS

```

```

  PORT(

    CLK,RSTn      : IN STD_LOGIC;

    PS2_Done_Sig   : IN STD_LOGIC;

```

```
        PS2_Data      : IN STD_LOGIC_VECTOR(7 DOWNTO 0);

        Data_Out      : OUT STD_LOGIC_VECTOR(3 DOWNTO 0)

    );

END ENTITY PS2_contorl_module;

-----

ARCHITECTURE PS2_contorl OF PS2_contorl_module IS

    SIGNAL Data : STD_LOGIC_VECTOR(3 DOWNTO 0);

    BEGIN

        PROCESS(CLK,RSTn)

        BEGIN

            IF (CLK'event AND CLK='1') THEN

                IF (RSTn='0') THEN

                    Data <= "0001";

                ELSIF(PS2_Done_Sig='1') THEN

                    CASE PS2_Data IS

                        WHEN X"22" => Data <= (Data(2 DOWNTO 0)&Data(3));

                        WHEN X"1A" => Data <= (Data(0)&Data(3 DOWNTO 1));

                        WHEN X"14" => Data <= (Data(0)&Data(1)&Data(2)&Data(3));

                        WHEN X"21" => Data <= "1111";WHEN X"2A" => Data <= "0000";

                        WHEN X"5A" => Data <= "0001";WHEN X"32" => Data <= "0001";

                        WHEN X"31" => Data <= "0011";WHEN X"3A" => Data <= "0111";

                        WHEN OTHERS => Data <= Data;

                    END CASE;

                END IF;

            END IF;

        END PROCESS;

        Data_Out <= Data;

    END ARCHITECTURE PS2_contorl;
```

LED 控制模块主要的作用是用于显示结果。在 PS2 键盘扫描后，得到的 8

位有效结果，使用 4 盏 LED 灯作为检查结果的输出，使用不用的 LED 闪亮方式来表示不同的按键按下了。本程序只做了 11 个按键，分别是“Z”，“X”，“C”，“V”，“B”，“N”，“M”，“Entet”和“Ctrl”。“Z”按下后，LED 向左移一个单位，“X”是向右移一个单位，“Ctrl”是 LED 灯互换，“B”是点亮一盏 LED，“N”是点亮两盏 LED，“M”是点亮三盏 LED，“Entet”是复原 LED 灯的情况。

(5) PS2 总的组合模块

```
LIBRARY IEEE;
```

```
USE IEEE.STD_LOGIC_1164.ALL;
```

```
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
```

```
-----  
ENTITY PS2_module IS
```

```
    PORT(
```

```
        CLK,RSTn      : IN STD_LOGIC;
```

```
        PS2_Data_Pin_In  : IN STD_LOGIC;
```

```
        PS2_CLK_Pin_In   : IN STD_LOGIC;
```

```
        Data_Out        : OUT STD_LOGIC_VECTOR(3 DOWNTO 0)
```

```
    );
```

```
END ENTITY PS2_module;
```

```
-----  
ARCHITECTURE PS2 OF PS2_module IS
```

```
    COMPONENT PS2
```

```
    PORT(
```

```
        CLK,RSTn      : IN STD_LOGIC;
```

```
        PS2_Data_Pin_In  : IN STD_LOGIC;
```

```
        PS2_CLK_Pin_In   : IN STD_LOGIC;
```

```
        PS2_Done_Sig     : OUT STD_LOGIC;
```

```
        PS2_Data        : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
```

```
    );
```

```
END COMPONENT;
```

```
COMPONENT PS2_contorl_module
```

```

PORT(

    CLK,RSTn      : IN STD_LOGIC;

    PS2_Done_Sig   : IN STD_LOGIC;

    PS2_Data       : IN STD_LOGIC_VECTOR(7 DOWNTO 0);

    Data_Out       : OUT STD_LOGIC_VECTOR(3 DOWNTO 0)

);

END COMPONENT;

SIGNAL Done_Sig   : STD_LOGIC;

SIGNAL Data       : STD_LOGIC_VECTOR(7 DOWNTO 0);

BEGIN

    U1:PS2          PORT MAP(CLK,RSTn,PS2_Data_Pin_In,PS2_CLK_Pin_In,Done_Sig,Data);
--
    U1:PS2          PORT MAP(CLK,RSTn,PS2_CLK_Pin_In,Done_Sig,Data);

    U2:PS2_contorl_module  PORT MAP(CLK,RSTn,Done_Sig,Data,Data_Out);

END ARCHITECTURE PS2;

```

这是一个组合例化的模块，是对 PS2 功能模块和 LED 控制的一个简单组合的模块。这一个模块是 PS2 键盘扫描的最顶层文件，只做例化作用，不包含其他功能代码。上面的是代码的方法实现例化功能，同时也可以使用原理图的方式来实现例化，下面为原理图例化的图。

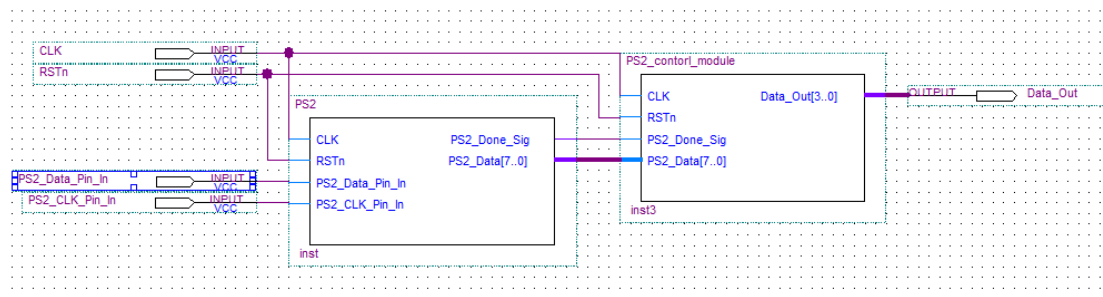


图 9 PS2_module 总例化原理图

四、综合与仿真

1、综合

对编写好的源程序进行综合，同时生成 RTL 电路图，RTL 电路图如下。

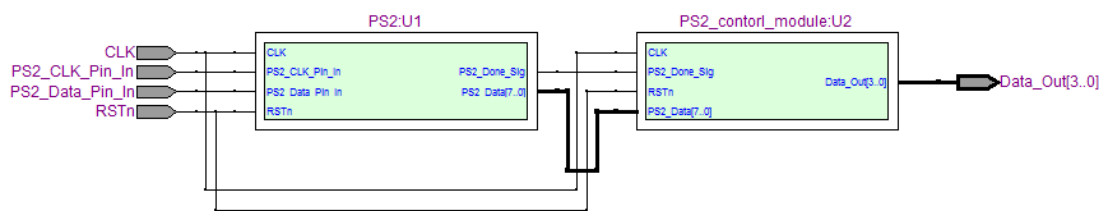


图 10 PS2_module 模块 RTL 图

2、仿真

在本次设计中,仿真工作只对电平检测模块,LED 灯控制模块和 PS2_module 总模块进行了仿真。对于前两个模块,采用的仿真方法是使用 Quartus II 自带的波形仿真软件进行仿真,而对于 PS2_module 总模块的仿真则使用了编写代码的方式进行仿真。因为 PS2_module 总模块对键盘按键的波形设置很麻烦,而且仿真效果不好,所以直接使用代码的方式产生一个键盘按键,同时观察波形来检测。

(1) 电平检测模块仿真

电平检测模块原理图如下:

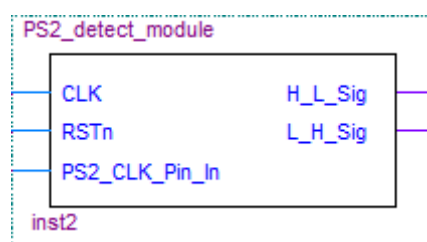


图 11 电平检测原理图

仿真波形如下:

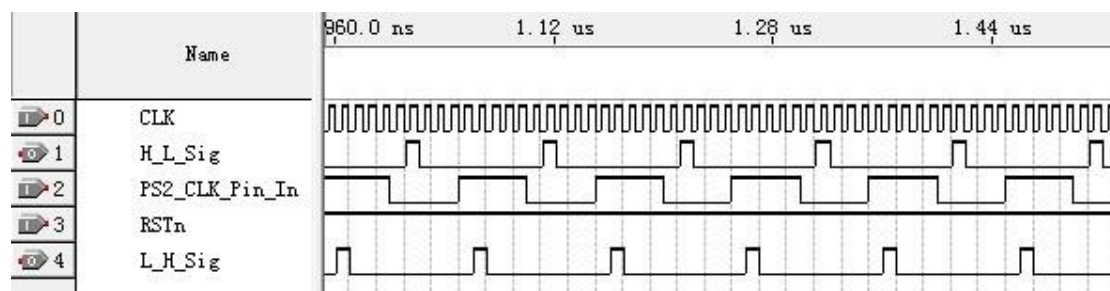


图 12 电平检测模块仿真波形

系统时钟给了 10ns, PS2_CLK_Pin_In 给了 100ns, RSTn 复位一直为高电平,可以观察波形得到,在 PS2_CLK_Pin_In 时钟的上升沿发生后, L_H_Sig 发出了一个高脉冲的信号,大概相隔两个系统时钟;在 PS2_CLK_Pin_In 时钟的下降沿发生后, H_L_Sig 也发出了一个高脉冲,也是相隔两系统时钟,而高脉冲大概是两个系统时钟左右。

(2) LED 灯控制模块仿真

LED 灯控制模块原理图如下：

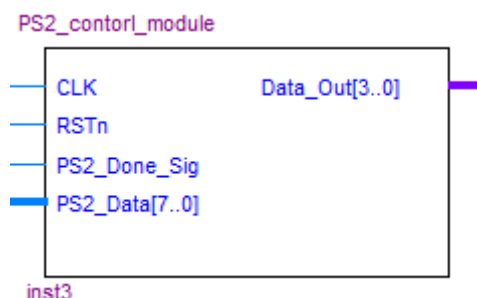


图 13 LED 灯控制模块原理图

仿真波形如下：

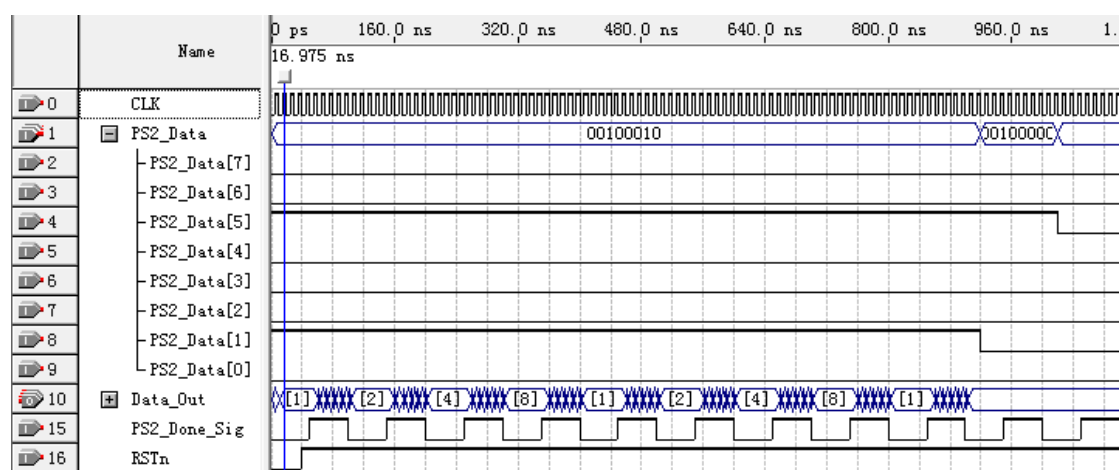


图 14 LED 灯控制模块仿真波形

系统时钟给了 10ns，RSTn 复位在一开始的时候给了 ‘0’，进行了一次复位，PS2_Done_Sig 完成信号则是 100ns 一次，PS2_Data 给的是 X “22” (即 X 被按下)。复位后 Data_Out 输出的是“0001”，第一盏 LED 灯点亮，然后在 PS2_Done_Sig 信号后，Data_Out 输出变化了，变为了“0010”，LED 第二盏给点亮了，LED 灯移位了。因为一直是 X 被按下，所以信号不断移位，LED 分别为 1，2，4，8。在最后那里，由于 PS2_Data 给的是 X “20”，并没有在控制那设置这个按键，所以 LED 没反应，一直保持在 2 (即第二盏灯点亮)。

(3) PS2_module 总模块仿真

由于 PS2 总仿真的时候需要键盘的通码输入，但使用波形的改变来作为通码输入的方式太麻烦了，而且很容易出错，得不到想要的波形。所以为了克服这个问题，我使用了另外一种方法来仿真，直接写键盘编码来给整个模块，然后开输出波形就可以观察。

PS2_module 总模块仿真点简单方框图如下：

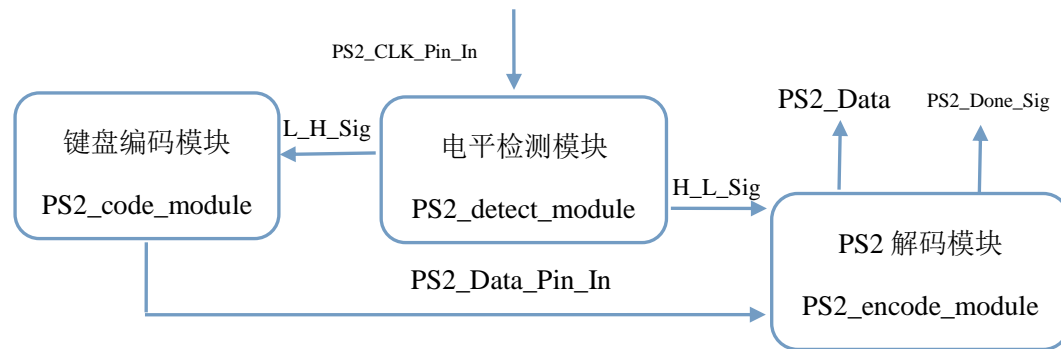


图 15 PS2_module 总模块方框图

PS2_code_module 键盘编码模块代码如下：

LIBRARY IEEE;

USE IEEE.STD_LOGIC_1164.ALL;

USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY PS2_code_module IS

PORT(

CLK,RSTn : IN STD_LOGIC;

L_H_Sig : IN STD_LOGIC;

PS2_Done_Sig : IN STD_LOGIC;

PS2_Data_Pin_Out : OUT STD_LOGIC

);

END ENTITY PS2_code_module;

ARCHITECTURE PS2_code OF PS2_code_module IS

CONSTANT n_0:STD_LOGIC_VECTOR:=X"22";CONSTANT n_1:STD_LOGIC_VECTOR:=X"1A";

CONSTANT n_2:STD_LOGIC_VECTOR:=X"14";CONSTANT n_3:STD_LOGIC_VECTOR:=X"21";

CONSTANT n_4:STD_LOGIC_VECTOR:=X"2A";CONSTANT n_n:STD_LOGIC_VECTOR:=X"F0";

SIGNAL data : STD_LOGIC :='0';

SIGNAL i : STD_LOGIC_VECTOR(4 DOWNT0 0) := "00000";

BEGIN

PROCESS(CLK,RSTn,i)

BEGIN

IF (CLK'event AND CLK='1') THEN

IF (RSTn='0') THEN

i <= "00000";

data <= '0';

ELSE

CASE i IS

WHEN "00000" => i <= "00001";

WHEN "00001" => IF (L_H_Sig='1') THEN i <= "00010";data <= n_1(0);END IF;

WHEN "00010" => IF (L_H_Sig='1') THEN i <= "00011";data <= n_1(1);END IF;

WHEN "00011" => IF (L_H_Sig='1') THEN i <= "00100";data <= n_1(2);END IF;

WHEN "00100" => IF (L_H_Sig='1') THEN i <= "00101";data <= n_1(3);END IF;

WHEN "00101" => IF (L_H_Sig='1') THEN i <= "00110";data <= n_1(4);END IF;

WHEN "00110" => IF (L_H_Sig='1') THEN i <= "00111";data <= n_1(5);END IF;

WHEN "00111" => IF (L_H_Sig='1') THEN i <= "01000";data <= n_1(6);END IF;

WHEN "01000" => IF (L_H_Sig='1') THEN i <= "01001";data <= n_1(7);END IF;

WHEN "01001" => IF (L_H_Sig='1') THEN i <= "01010";END IF;

WHEN "01010" => IF (L_H_Sig='1') THEN i <= "01011";END IF;

WHEN "01011" => IF (PS2_Done_Sig='1') THEN i <= "01100";END IF;

WHEN "01100" => IF (L_H_Sig='1') THEN i <= "01101";data <= n_n(0);END IF;

WHEN "01101" => IF (L_H_Sig='1') THEN i <= "01110";data <= n_n(1);END IF;

WHEN "01110" => IF (L_H_Sig='1') THEN i <= "01111";data <= n_n(2);END IF;

WHEN "01111" => IF (L_H_Sig='1') THEN i <= "10000";data <= n_n(3);END IF;

WHEN "10000" => IF (L_H_Sig='1') THEN i <= "10001";data <= n_n(4);END IF;

WHEN "10001" => IF (L_H_Sig='1') THEN i <= "10010";data <= n_n(5);END IF;

WHEN "10010" => IF (L_H_Sig='1') THEN i <= "10011";data <= n_n(6);END IF;

WHEN "10011" => IF (L_H_Sig='1') THEN i <= "10100";data <= n_n(7);END IF;

WHEN "10100" => IF (L_H_Sig='1') THEN i <= "10101";END IF;

WHEN "10101" => IF (L_H_Sig='1') THEN i <= "10110";END IF;

WHEN "10110" => IF (L_H_Sig='1') THEN i <= "10111";END IF;

```

        WHEN "10111" => IF (L_H_Sig='1') THEN i <= "00000";END IF;

        WHEN OTHERS    => i <= "00000";

    END CASE;

END IF;

END IF;

END PROCESS;

PS2_Data_Pin_Out <= data;

END ARCHITECTURE PS2_code;

```

代码实现的功能就是每一个 PS2_CLK_Pin_In 的上升沿来临的时候，设置好数据给 S2_CLK_Pin_In 的下降沿时后面模块的读取，时间要与 PS2_decode_module 模块同步。

仿真出来的波形图如下：

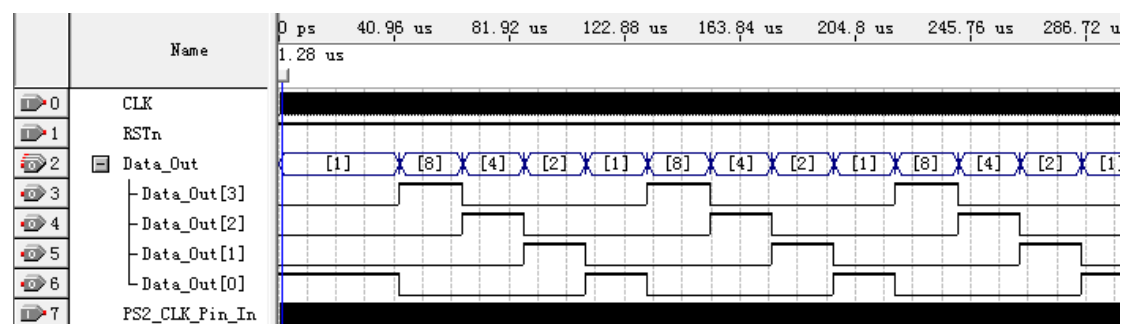


图 16 输入为“Z”的波形图

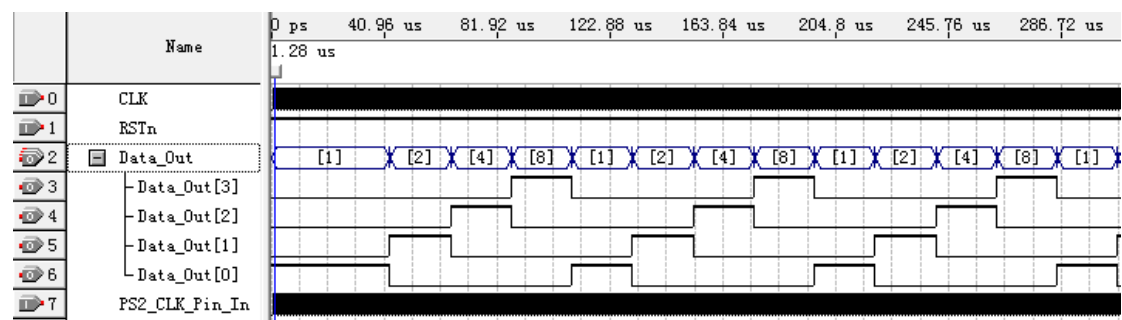


图 17 输入为“X”的波形图

由图 16 和图 17 可以看出整个模块的仿真结果。“X”和“Z”按键的作用是一个向左移，一个向右移，在上图中能很清晰地看出他们的变化。

五、硬件下载

先设置好芯片类型，因为我是使用自己的 FPGA 开发板，所以我用的芯片是 Cyclone II 的 EP2C5Q208C8，如下图。

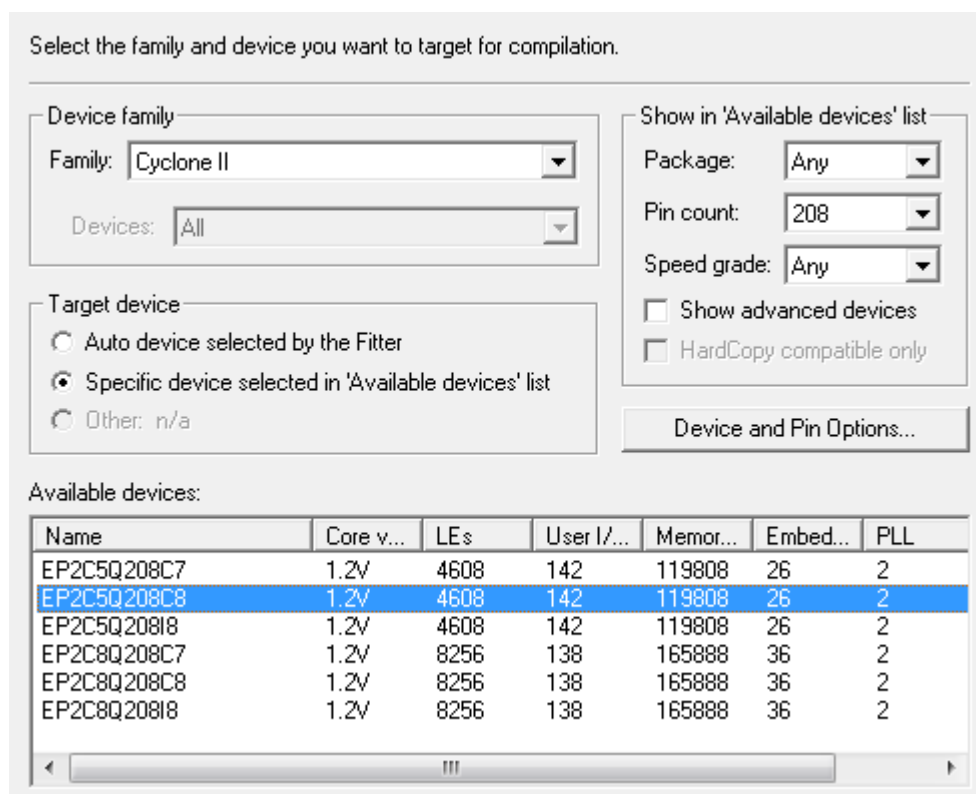


图 18 设置芯片型号

设置芯片后，接着的是设置不用到引脚位，将不用的引脚设置为三态输入，以防芯片烧坏，设置方式如下图：

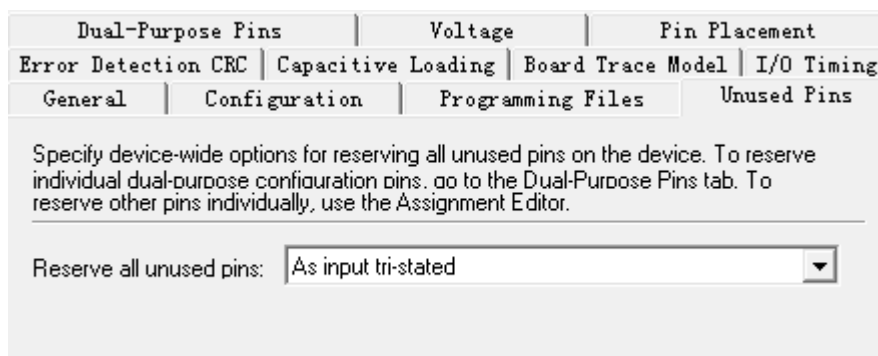


图 19 设置没有用到的引脚

成功编译后，是引脚分配，将需要用到的端口分配到 FPGA 的引脚上关联起来，引脚分配如下图：

	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved
1	CLK	Input	PIN_24	1	B1_N0	3.3-V LVTTTL (default)	
2	Data_Out[3]	Output	PIN_206	2	B2_N1	3.3-V LVTTTL (default)	
3	Data_Out[2]	Output	PIN_205	2	B2_N1	3.3-V LVTTTL (default)	
4	Data_Out[1]	Output	PIN_203	2	B2_N1	3.3-V LVTTTL (default)	
5	Data_Out[0]	Output	PIN_201	2	B2_N1	3.3-V LVTTTL (default)	
6	PS2_CLK_Pin_In	Input	PIN_192	2	B2_N1	3.3-V LVTTTL (default)	
7	RSTn	Input	PIN_27	1	B1_N1	3.3-V LVTTTL (default)	
8	PS2_Data_Pin_In	Unknown	PIN_191	2	B2_N1	3.3-V LVTTTL (default)	
9	<<new node>>						

图 20 引脚分配图

最后的操作是硬件下载，在 PROGRAMMER 中进行，要设置好 USB 仿真口，

FPGA 下载方式是 JTAG，下载图如下：

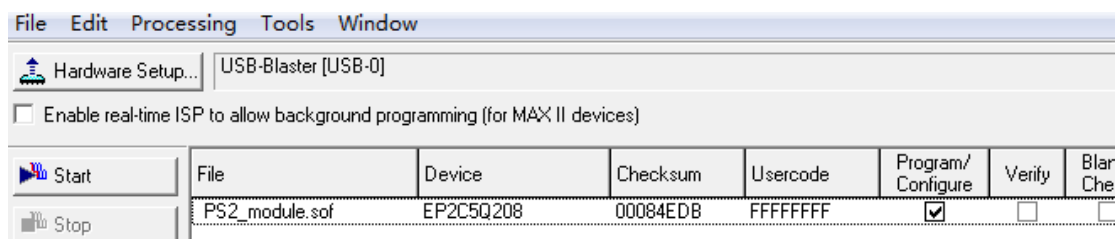


图 21 文件下载图

最后是在讲 PC 键盘接在 FPGA 开发板上，按下按键来试调观察结果。

六、心得体会

七、参考文献