实验一 计数器和序列检测器的设计

实验目的：

- 掌握简单时序逻辑电路的设计方法；
- 了解任意进制计数器的设计方法；
- 掌握有限状态机的实现原理和方法；
- 掌握序列检测的方法

实验原理：

计数器是一种常用的时序电路，它按照规定的方式改变内部各触发器的状态，以记录输入的时钟脉冲的个数。按照规定的计数顺序的不同，计数器可以分为加法计数器、减法计数器、可逆计数器和不同进制的计数器；按照工作方式的不同，又可以分为异步计数器和同步计数器。

有限状态机（Finite State Machine, FSM）是逻辑电路设计中经常要遇到的，在数字电路中，经常需要通过建立有限状态机的方式来进行时序数字逻辑的设计。在复杂数字系统设计中，有限状态机主要通过硬件描述语言实现，硬件描述语言能够清晰的描述状态转移过程和输入输出变量关系，使得时序逻辑设计大大简化，进而极大降低系统设计复杂度，提高系统模块化程度。有限状态机从本质上讲是由寄存器和组合逻辑构成的时序电路，各个状态之间的转移总是在时钟的触发下进行的。

实验内容：

（1）设计一个具有**异步复位**控制的 4bits 十进制同步加法计数器。

（2）在连续输入的串行数据流中检测特定序列“101011”，一旦检测到一个“101011”就输出一个宽度为 1 个时钟周期的高电平脉冲。例如，当输入为“00101011010111000101011 00”时，输出为“00000000100001000000000010”。两个 101011 序列可以重叠，如例子中的第二个输出 1。

（2.1） 用**有限状态机**设计序列检测器，检测序列“101011”。

（2.2） 用**移位寄存器**和组合逻辑实现序列检测器，检测序列“101011”。

实验要求：

（1）采用行为级设计方法设计；

（2）使用按键（KEY3-KEY5 任选）作为时钟输入。

（3）按绪论课要求完成实验报告，在报告中，应当写明设计方案（比如对有限状态机实现，请给出状态转移图），综合情况（使用 LUT、寄存器等资源情况，并进行分析，对有限状态机，应当说明工具是否识别出状态机，以及是如何编码的）。

（4）选做：探索按键抖动对实验结果的影响（KEY1 或 KEY2 作为时钟输入），使用 debounce.v 模块提供的防抖代码，对按键抖动进行消除。

推荐外部电路连接方法：

计数器：

器件	**管脚**	**功能**
KEY5	J22	异步复位信号
数码管	同 BCD7 演示实验	显示数字
KEY3	K22	外部时钟输入

状态机：

器件	**管脚**	**功能**
LED1	V2	数据输出
LED6-8	AA1、AB1、AB2	状态机编码
KEY5	J22	复位信号
KEY3	K22	外部时钟输入
GPIO8	M1	拨码开关、串行数据输入

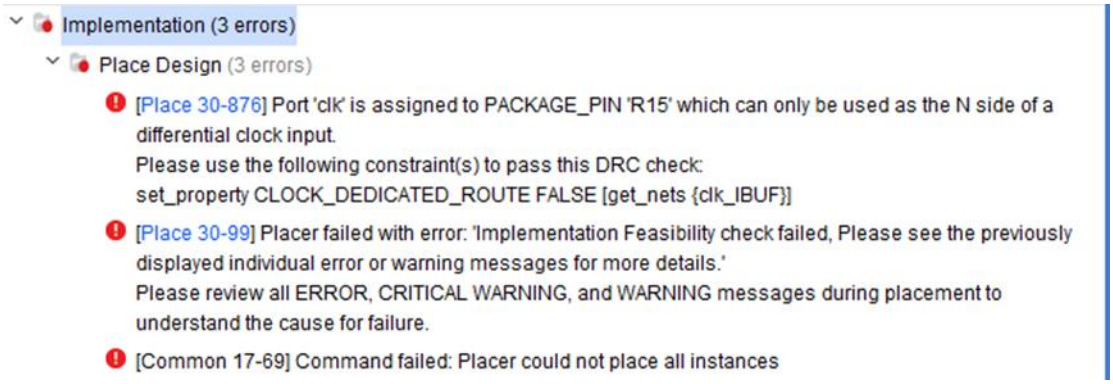
移位寄存器：

器件	**管脚**	**功能**
LED1	V2	数据输出
LED3-8	W2、Y1、Y2、AA1、AB1、AB2	移位寄存器数据
KEY5	J22	复位信号
KEY3	K22	外部时钟输入
GPIO8	M1	拨码开关、串行数据输入

实验提示：

1. 时钟管脚问题：

用按键连接引脚直接作为时钟输入，Vivado 会报出如下错误：



原因正如在这个错误消息中所说，FPGA 对时钟的引入管脚有特殊的限制，按键连的这个引脚不能够接入时钟分配网络。解决办法在消息中，其实已经给出：在约束文件中，加入

```
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets {clk_IBUF}];
```

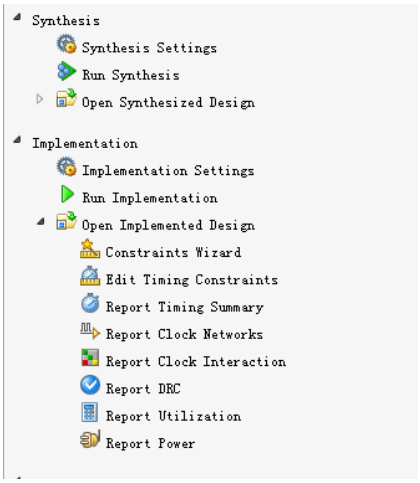
其中 clk_IBUF 就是 Verilog 模块中对应物理按键的端口名称。

2. 占用资源的查看：

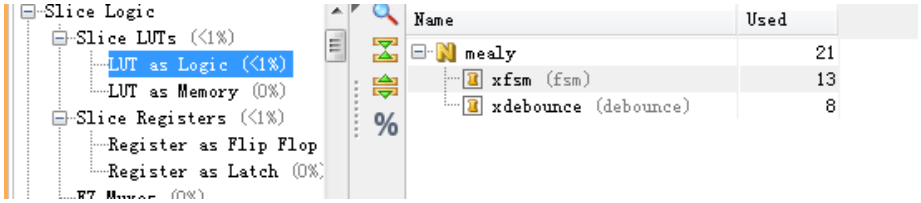
从实验一开始，我们在实验报告中，要求报告所设计的电路的资源占用情况。以下面代码为例，顶层由设计的 FSM（有限状态机）构成。

```
fsm xfsm(.clk(clk), .reset(reset), .data(data), .result(out), .cond(code));
```

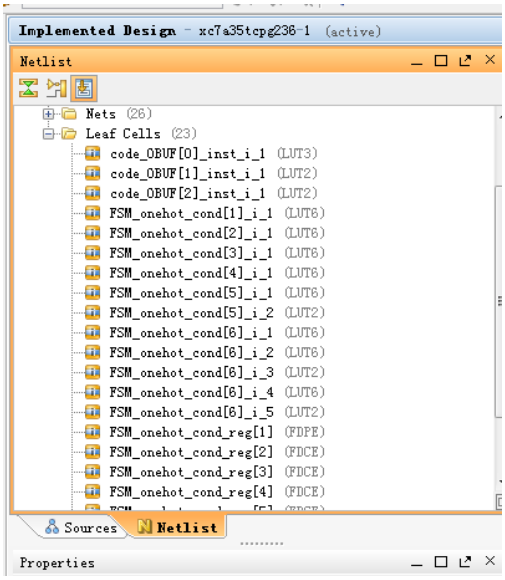
我们要分析我们设计的 FSM 的资源占用情况。在 Implementation 完成后，点击 Report Utilization 生成面积占用情况报告。



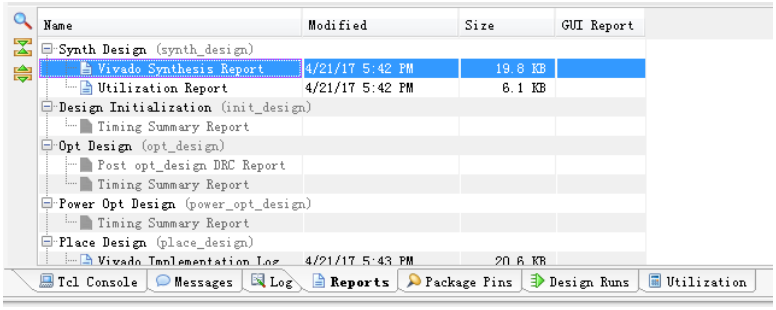
资源占用情况包括组合逻辑（在 FPGA 用查找表 LUT 实现）和寄存器。点开 Slice Logic 中的 Slice LUTs 可以看到 xfsm 占用了 13 个查找表，点开 Slice Registers 可以看到 xfsm 占用了 7 个寄存器。



占用的寄存器比预想的多。我们 Open Implemented Design 在 Netlist 中，看到寄存器的名字后面加了 onehot，猜想 Vivado 使用 one-hot 编码优化了状态机。



我们可以查阅 Vivado Synthesis Report，确认其编码方式。



其中确有如下内容:

INFO: [Synth 8-802] inferred FSM for state register 'cond_reg' in module 'fsm'

INFO: [Synth 8-3354] encoded FSM with state register 'cond_reg' using encoding 'one-hot' in module 'fsm'

3. 代码的风格:

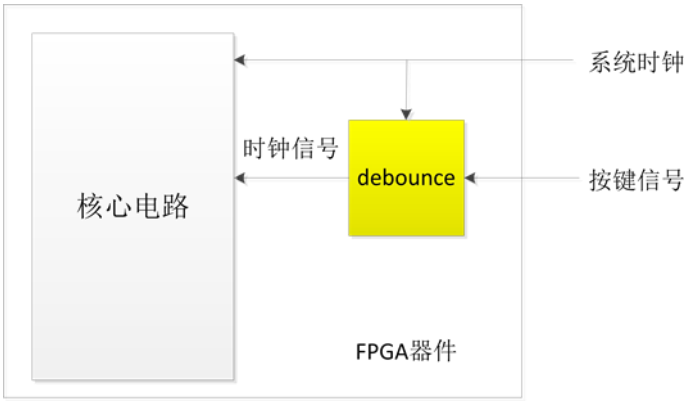
在实验过程中，有的同学已经遇到了语法正确，但是综合后烧写到板子上结果却不正确的情况。排除设计错误的前提下，其最大的可能性就是在 Verilog 的编写中，没有遵循可综合的 Verilog 代码风格。

代码风格一个很好的手册是 Xilinx 公司针对我们所用的 Vivado 工具给出的用户手册 UG901。这个文档我们已经上传到网络学堂的课程文件中，大家可以参考。移位寄存器的例子在 78 页，有限状态机的例子在 164 页。有限状态机用我们课件中的两段式也是可以的。

如果遇到问题，建议查看综合实现输出的 Message 中的 Warning 信息，有助于问题的定位与解决。

4. 按键抖动问题:

在实验中要求采用外部按键作为时钟输入, 通过按动按键产生有效的时钟边沿, 由于机械按键存在抖动现象, 表现在输入信号上会出现段时间内的震荡现象。所以，需要在设计内部首先对按键输入信号进行消抖动处理。



时钟按键可以使用防抖代码去除扰动，防抖代码可以使用 debounce.v。使用方法如下:

```
module instance_name(system_clk, clk_i, other_input output);
input system_clk, clk_i;
***
debounce xdebounce(.clk(system_clk),.key_i(clk_i),.key_o(clk_o));
***
endmodule
```

其中,system_clk 绑定到系统时钟 R4 上，clk_i 是用户自己的时钟，使用按键实现，使用防抖代码输出的 clk_o 作为最后的时钟使用。

WELOG 开发板 KEY1，KEY2 按键，快速按动时会存在一定的抖动现象，而这种现象随着按键的老化会更加明显。KEY3-KEY5 三个按键，在电路层面做了防抖处理，直接使用时，观察不到抖动现象。