有限状态机

简介

有限状态机(FSM)是许多数字系统中用来控制系统和数据流路径行为的时序电路。FSM的实例包括控制单元和定序仪。 本实验介绍了两种类型的FSM（Mealy和Moore）的概念，以及开发此类状态机的建模方式。 请参阅Vivado教程，了解如何使用Vivado工具创建项目和验证数字电路。

目标

在本次实验中，你将会学到:

- 对 Mealy FSMs 建模
- 对 Moore FSMs 建模

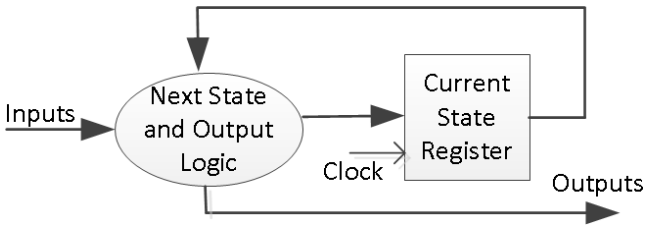
Mealy FSM Part 1

有限状态机（FSM）或称简单状态机用于设计计算机程序和时序逻辑电路。它被设想为抽象机器，可以处于有限数量的用户定义状态之一。机器一次只能处于一种状态; 它在任何给定时间所处的状态称为当前状态。 当由触发事件或条件启动时，它可以从一种状态改变为另一种状态; 这称为过渡。特定FSM由其状态列表和每个转换的触发条件定义。

在现代社会中的许多设备中可以观察到状态机的踪影，这些设备根据发生的事件序列执行预定的动作序列。 简单的例子是自动售货机，当存放硬币的金额达到商品价格时分配产品；电梯在把乘客送达楼上后才会下降；交通灯按一定的时间改变信号来控制车流；以及需要输入一串正确的数字才能打开的组合锁。

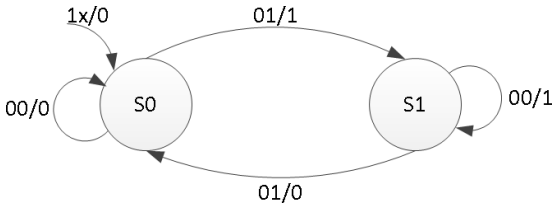
状态机使用两种基本类型建模--Mealy和Moore。 在Mealy机器中，输出取决于当前状态和当前输入。在Moore机器中，输出仅取决于当前状态。

Mealy型状态机的一般模型由组合过程电路和状态寄存器组成，组合过程电路生成输出和下一个状态，状态寄存器保存当前状态，如下图所示。状态寄存器通常建模为D触发器。状态寄存器必须对时钟边缘敏感。其他块可以使用always过程块或always过程块和dataflow建模语句的混合来建模；always过程块必须对所有输入敏感，并且必须为每个分支定义所有输出，以便将其建模为组合块。两段式Mealy机器可以表示为



下面是奇偶校验校验机的状态图和相关模型:

© copyright 2015 Xilinx

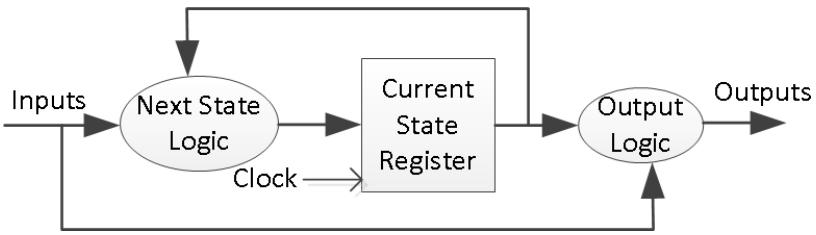



```
module mealy_2processes(input clk, input reset, input x, output reg parity);
    reg state, nextstate;
    parameter S0=0, S1=1;

always @(posedge clk or posedge reset)    // always block to update state
if (reset)
     state <= S0;
else
    state <= nextstate;

always @(state or x) // always block to compute both output & nextstate
begin
    parity = 1'b0;
    case(state)
       S0: if(x)
           begin
               parity = 1; nextstate = S1;
           end
           else
                nextstate = S0;
       S1: if(x)
                nextstate = S0;
           else
           begin
                parity = 1; nextstate = S1;
           end
       default:
           nextstate = S0;
    endcase
end
endmodule
```

三段式Mealy机器的图示及其建模如下：



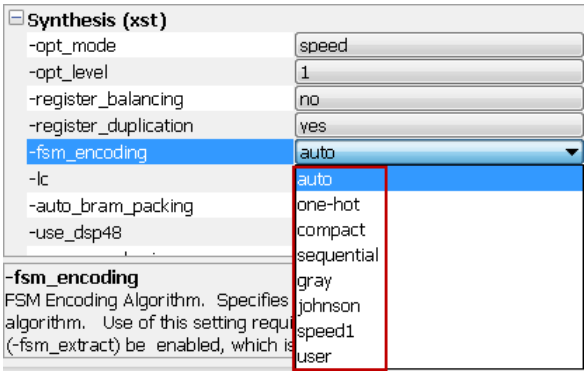

```
module mealy_3processes(input clk, input reset, input x, output reg parity);
    reg state, nextstate;
    parameter S0=0, S1=1;

  always @(posedge clk or posedge reset)  // always block to update state
  if (reset)
       state <= S0;
  else
      state <= nextstate;
```


www.xilinx.com/university
xup@xilinx.com
© copyright 2015 Xilinx

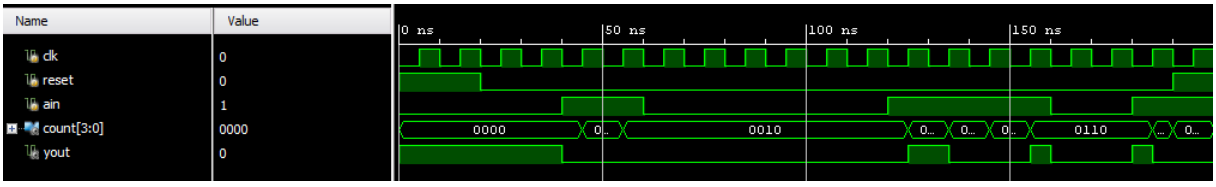
```
always @(state or x)  // always block to compute output
begin
   parity = 1'b0;
   case(state)
      S0: if(x)
            parity = 1;
      S1: if(!x)
            parity = 1;
   endcase
end
always @(state or x)  // always block to compute nextstate
begin
   nextstate = S0;
   case(state)
      S0: if(x)
            nextstate = S1;
      S1: if(!x)
            nextstate = S1;
      endcase
end
endmodule
```

状态分配可以使用独热码（one – hot code），二进制编码，格雷码以及其他编码。通常，综合工具将确定状态分配的编码，但用户也可以通过更改综合属性来强制特定编码，如下所示。状态分配编码将对状态寄存器中使用的位数产生影响：独热编码使用最多的位数，但解码非常快，二进制编码使用最少的位数，但解码较长。



1-1. 使用三段式Mealy状态机的实现一个序列检测器。

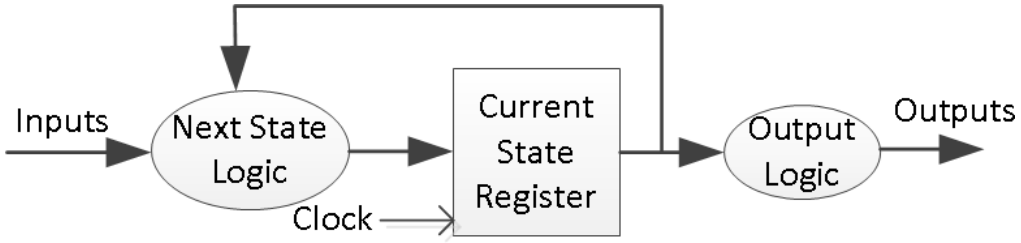
Mealy状态机有一个输入（ain）和一个输出（yout）。 当且仅当接收到的1的总数可被3整除时，输出为1（提示：0也算被3整除，但是，在复位周期中不把计数器归为0，复位信号过后把计数器归0——参考模拟波形时间= 200。设计一个testbench并通过behavioral simulation验证模型。 使用SW15作为时钟输入，SW0作为输入，BTNU按钮作为电路的复位输入， LED7：LED4作为“1”的个数的输出，LED0作为yout的输出。 完成设计流程，生成比特流，并将其下载到Basys3或Nexys4 DDR板。验证功能。



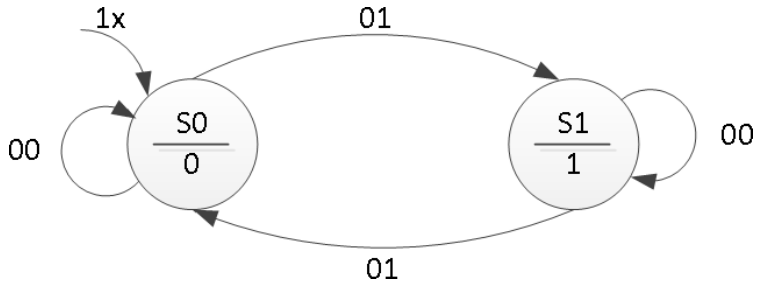
© copyright 2015 Xilinx

Moore FSM Part 2

Moore型有限状态机的一般模型如下所示。 其输出由状态寄存器块生成。 使用当前输入和当前状态确定下一状态。 这里的状态寄存器也使用D触发器建模。 通常，Moore机器使用三个块来描述，其中一个块必须是顺序的，另外两个块可以使用always块或always和dataflow建模结构的组合来建模。



以下是使用Moore型有限状态机实现的奇偶校验器的状态图。与之关联模型如下所示。




```
module moore_3processes(input clk, input reset, input x, output reg parity);
    reg state, nextstate;
    parameter S0=0, S1=1;

  always @(posedge clk or posedge reset)  // always block to update state
  if (reset)
       state <= S0;
  else
      state <= nextstate;

  always @(state)              // always block to compute output
  begin
      case(state)
         S0: parity = 0;
         S1: parity = 1;
      endcase
  end
  always @(state or x)         // always block to compute nextstate
  begin
      nextstate = S0;
      case(state)
         S0: if(x)
                nextstate = S1;
         S1: if(!x)
                nextstate = S1;
      endcase
  end
  endmodule
```


© copyright 2015 Xilinx

在本例中，输出块很简单，可以使用dataflow建模构造进行建模。 可以使用以下代码代替always块。 您还需要将输出类型从reg更改为wire。

```
assign parity = (state==S0) ? 1'b0: 1'b1;
```

2-1. 使用使用三段式Moore状态机实现一个序列检测器。

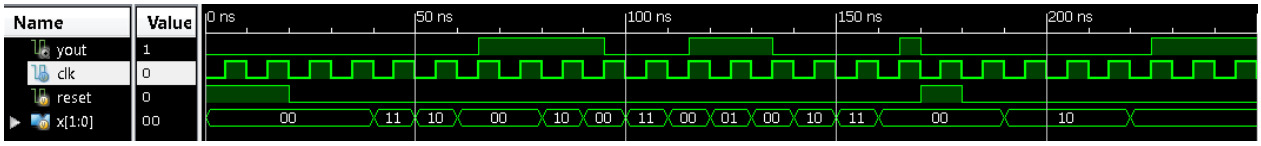
Moore状态机有两个输入（ain [1：0]）和一个输出（yout）。 除非出现以下输入序列之一，否则输出将从0开始并保持为常量值：

(i) 输入序列ain [1：0] = 01,00使输出变为0

(ii) 输入序列ain [1：0] = 11,00使输出变为1

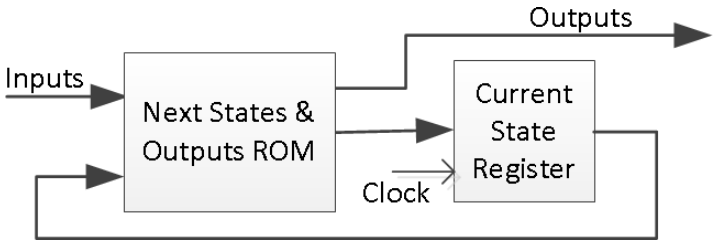
(iii) 输入序列ain [1：0] = 10,00使输出切换。

设计一个testbench（类似于下面显示的波形）并通过behavioral simulation验证模型。 使用SW15作为时钟输入，SW1-SW0作为ain [1：0]输入，BTNU按钮作为电路的复位输入，LED0作为yout输出。 完成设计流程，生成比特流，并将其下载到Basys3或Nexys4 DDR板。 验证功能。



使用ROM的Mealy FSM Part 3

Mealy有限状态机也可以使用ROM存储器实现，如下所示。 ROM存储器保存下一个状态并输出内容。外部输入和当前状态构成ROM的地址输入。 ROM通常使用LUT而不是BlockRAM来实现，因为LUT通过设计中较少数量的状态提供了更好的利用率。

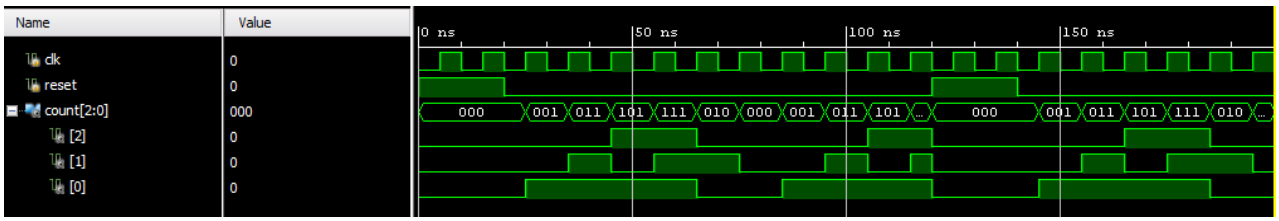


3-1. 使用ROM设计一个特定的计数计数器（下面列出的计数序列）来开发一个Mealy状态机。

设计一个testbench并通过behavioral simulation验证模型。 使用SW15作为时钟输入，BTNU按钮作为电路的复位输入，LED2：LED0作为计数器的计数输出。 完成设计流程，生成比特流，并将其下载到Basys3或Nexys4 DDR板。 验证功能。

计数序列是: 000, 001, 011, 101, 111, 010 (repeat) 000, ...

© copyright 2015 Xilinx



总结

在这个实验中，您学习了Mealy和Moore状态机建模方法。还设计并实现了序列检测器、序列生成器和编码转换器，实践了两段和三段式的状态机。

© copyright 2015 Xilinx