西安电子科技大学
XIDIAN UNIVERSITY

基于 FPGA 的数字系统设计

实验报告

实验名称：旋转门 Turnstile 状态机

任课教师：沈沛意老师

学号姓名：

提交日期：

一、实验介绍

A turnstile, used to control access to subways and amusement park rides, is a gate with three rotating arms at waist height, one across the entryway. Initially the arms are locked, barring the entry, preventing customers from passing through. Depositing a coin or token in a slot on the turnstile unlocks the arms, allowing them to rotate by one-third of a complete turn, allowing a single customer to push through. After the customer passes through, the arms are locked again until another coin is inserted

The turnstile has two states: Locked and Unlocked. There are two inputs that affect its state: putting a coin in the slot (coin) and pushing the arm (push). In the locked state, pushing on the arm has no effect; no matter how many times the input push is given it stays in the locked state. Putting a coin in, that is giving the machine a coin input, shifts the state from Locked to Unlocked. In the unlocked state, putting additional coins in has no effect; that is, giving additional coin inputs does not change the state. However, a customer pushing through the arms, giving a push input, shifts the state back to Locked

Current State	Input	Next State	Output
Locked	coin	Unlocked	Release turnstile so customer can push through
	push	Locked	None
Unlocked	coin	Unlocked	None
	push	Locked	When customer has pushed through lock turnstile

二、实验目标

学习状态机的 VHDL 语言描述方法;

完成对旋转门状态机的设计实现和测试。

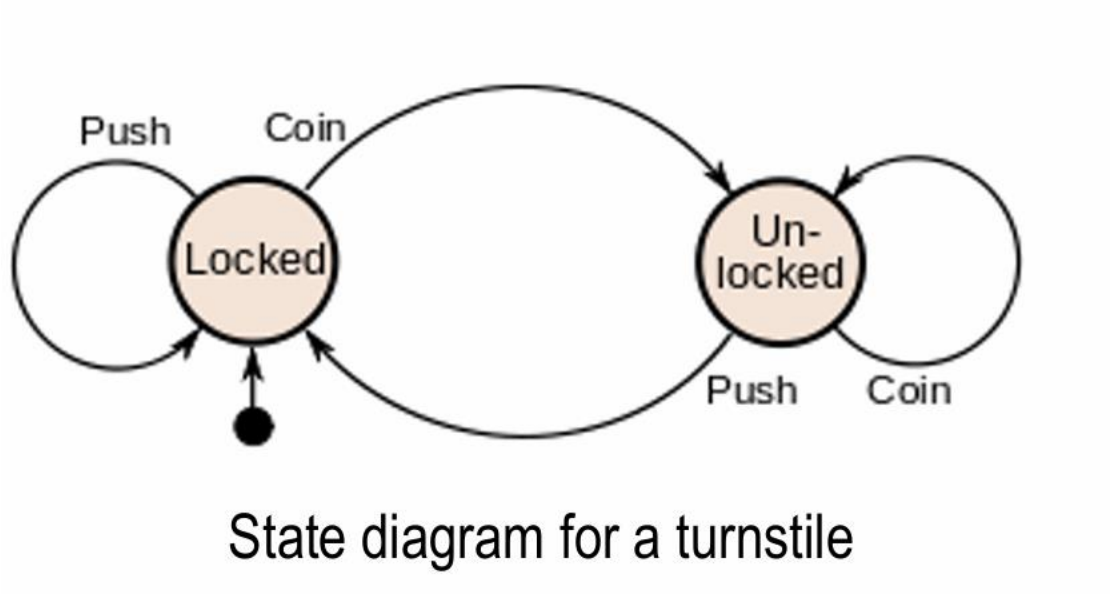
三、实验过程与步骤

本实验包含三个主要的部分：创建一个新的 ISE 工程;编写 Turnstile_FSM 的 VHDL 代码;仿真验证代码的功能正确性。

1. 启动 ISE 创建一个新的工程 LAB7

   过程不再赘述

2. 设计并实现 Turnstile_FSM



State diagram for a turnstile

根据旋转门的状态转移图，我们可以知道，在 LOCKED 状态下，只有 Coin 才能让状态变为 UNLOCKED;反之，在 UNLOCKED 状态下，只有 Push 才能让状态变为 LOCKED

所以在 VHDL Module 中需要实现上面描述的状态转移逻辑。此外，我们还需要对状态寄存器做同步复位和对是否开门的输出信号做输出逻辑。

其中，当 rst=1 时表复位，当前状态改为 LOCKED;

输出逻辑：仅在解锁后推动时开门

```
                if (current_state = UNLOCKED) and (push = '1') then
                    door_open <= '1';
                else
                door_open <= '0';
```

Turnstile_FSM.vhd:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Turnstile_FSM is
    Port (
        clk       : in  STD_LOGIC;   -- 时钟信号
        rst       : in  STD_LOGIC;   -- 同步复位
        coin      : in  STD_LOGIC;   -- 投币事件
        push      : in  STD_LOGIC;   -- 推动事件
        door_open : out STD_LOGIC    -- 开门信号
    );
end Turnstile_FSM;

architecture Behavioral of Turnstile_FSM is
     -- 定义状态枚举类型
    type state_type is (LOCKED, UNLOCKED);
   signal current_state, next_state : state_type;
begin

    -- 状态寄存器（同步复位）

   process(clk, rst)
   begin
     if rising_edge(clk) then
         if rst = '1' then
             current_state <= LOCKED;
         else
             current_state <= next_state;
         end if;
     end if;
  end process;

  -- 状态转移逻辑（组合逻辑）
    process(current_state, coin, push)
```

```
        begin
               case current_state is
                when LOCKED =>
                    if coin = '1' then
                        next_state <= UNLOCKED;
                    else
                        next_state <= LOCKED;
                    end if;

                when UNLOCKED =>
                    if push = '1' then
                        next_state <= LOCKED;
                    else
                        next_state <= UNLOCKED;
                    end if;

                when others =>
                    next_state <= LOCKED;
                end case;
        end process;

        -- 输出逻辑（仅在解锁后推动时开门）
        process(clk)
        begin
            if rising_edge(clk) then
                if (current_state = UNLOCKED) and (push = '1')
then
                    door_open <= '1';
                else
                    door_open <= '0';
                end if;
            end if;
        end process;

    end Behavioral;
```

3. 创建测试平台

在测试过程中，做四个测试用例:

测试用例 1: 锁定状态下投币 -> 解锁

测试用例 2: 解锁状态下推动 -> 开门并复位到锁定

测试用例 3: 锁定状态下无效推动（应保持锁定）

测试用例 4: 解锁状态下重复投币（应忽略）

TB_Turnstile_FSM.vhd:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity TB_Turnstile_FSM is
end TB_Turnstile_FSM;

architecture Behavioral of TB_Turnstile_FSM is
    -- 被测试模块的信号
    signal clk       : STD_LOGIC := '0';
    signal rst       : STD_LOGIC := '0';
    signal coin      : STD_LOGIC := '0';
    signal push      : STD_LOGIC := '0';
    signal door_open : STD_LOGIC;

    -- 时钟周期定义（50MHz）
    constant clk_period : time := 20 ns;
begin

    -- 实例化被测试模块
    UUT: entity work.Turnstile_FSM
        port map (
            clk       => clk,
            rst       => rst,
            coin      => coin,
            push      => push,
            door_open => door_open
        );

    -- 生成时钟
    clk_process: process
    begin
        clk <= '0';
        wait for clk_period/2;
        clk <= '1';
        wait for clk_period/2;
    end process;

    -- 测试激励
```

```
    stimulus: process
    begin
        -- 初始化复位
        rst <= '1';
        wait for clk_period * 2;
        rst <= '0';
        wait for clk_period;

        -- 测试用例 1: 锁定状态下投币 -> 解锁
        coin <= '1';
        wait for clk_period;
        coin <= '0';
        wait for clk_period;

        -- 测试用例 2: 解锁状态下推动 -> 开门并复位到锁定
        push <= '1';
        wait for clk_period;
        push <= '0';
        wait for clk_period;

        -- 测试用例 3: 锁定状态下无效推动（应保持锁定）
        push <= '1';
        wait for clk_period;
        push <= '0';
        wait for clk_period;

        -- 测试用例 4: 解锁状态下重复投币（应忽略）
        coin <= '1';
        wait for clk_period;
        coin <= '0';
        wait for clk_period;
        coin <= '1';
        wait for clk_period;
        coin <= '0';
        wait for clk_period;

        -- 结束仿真
        wait;
    end process;

end Behavioral;
```

四、实验结果及分析

功能仿真:

波形图满足四个用例的旋转门逻辑：

测试用例 1：锁定状态下投币 -> 解锁

测试用例 2：解锁状态下推动 -> 开门并复位到锁定

测试用例 3：锁定状态下无效推动（应保持锁定）

测试用例 4：解锁状态下重复投币（应忽略）