第四章 基本时序逻辑电路建模

Classification of sequential logic

In terms of register actions

- Synchronous: The register actions are triggered by clock changes only
- Asynchronous: Asynchronous sequential logic is not synchronized by a clock signal; the circuit output can chage directly in response to input changes

In terms of output signal

- Mealy: the output depends on both memory state and the input
- Moore: The output depends on memory state only

4.1 锁存器Latch

Latch和FF的区别：Latch的敏感参数表中包含输入参数

Latch输入信号D的变化会影响输出Q，在时钟enable的时间内Q随D的变化而变化；FF仅在时钟边沿才会判断改变输出Q的值。二者波形有很大不同。

4.1.1 RS latch

真值表见课本P67

RTL图要记住！！

```
library ieee;
use ieee.std_logic_1164.all;

entity rs_latch is
    port(R,S: in std_logic;
    Q,Qbar: out std_logic);
end entity rs_latch;

architecture behav of rs_latch is
begin
    process(R,S)
        variable rs: std_logic_vector(1 downto 0);
    begin
        rs:=R&S;
        case rs is
            when "00"=>
                Q<=1;Qbar<=1;
            when "01"=>
                Q<=0;Qbar<=1;
            when "10"=>
                Q<=1;Qbar<=0;
            when others=>
                null;
        end case;
    end process;
end behav;
```

4.1.2 D Latch

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;

ENTITY D_latch IS
    PORT(
        D,Enable: IN std_logic;
        Q: OUT std_logic
    );
END D_latch;

ARCHITECTURE behav OF D_latch IS
BEGIN
    PROCESS (D, Enable)
    BEGIN
        IF (Enable = '1') THEN Q<=D;
        end IF;
    END PROCESS             -- described by incomplete if statement
END behav;
```

4.2 触发器Flip-Flop

4.2.1 D F.F

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;

ENTITY D_FF IS

    PORT(D, CLK: IN std_logic;
        Q: OUT std_logic;)
END D_FF;

ARCHITECTURE behav OF D_FF IS
BEGIN
    PROCESS(CLK)
    BEGIN-------------------------多种表述方式
        IF (CLK'event AND CLK = '1') THEN Q<=D;
        -- or
        IF RISING_EDGE(CLK) THEN Q<=D;  --can be used only if CLK is std_log
        ----------------------------
        END IF;
    END PROCESS;
END behav;
```

最常用的边沿检测表达方式

```
-- 注意这两个语句后面不能接ELSE分支
IF CLK'event and CLK = '1' THEN -- 上升沿
IF CLK'event and CLK = '0' THEN -- 下降沿
```

4.2.2 带有$\overline{Q}$输出的D触发器

将signal放在process中时，有几个signal变量，仿真时就会产生几个中间节点，导致设计结果不符合要求

着重参考课本*P73-74*页例*4-5*

- 使用signal作为中间变量时，注意Q和Qbar的赋值语句要放在Process结构的外面，否则将产生三个signal变量，导致仿真延迟不符合要求；若放在Process外部，则后面两个赋值语句与Process为并行关系
- 使用variable做中间变量时要注意variable 为局部量

4.2.3 JK触发器

真值表

J	K	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
0	1	0	1
1	0	1	0
1	1	$\overline{Q}$	Q

```
Case R&S is
    When “00”=> Q0<=Q0;
    When “01”=> Q0<=’0’;
    When “10”=> Q0<=’1’;
    When “11”=> Q0<=not Q0;
    When others =>null;
End case;
```

4.2.4 T触发器

P77 应用：计数器，期末答疑ppt43

4.2.5 SR锁存器

```
Case R&S is
    When “00”=> Q1<=Q1;
    When “01”=> Q1<=’1’;
    When “10”=> Q1<=’0’;
    When “11”=> Q1<=’X’;
    When others =>null;
End case;
```

4.3 寄存器Register

Multibit Register

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;

ENTITY four_bit_reg IS
    GENERIC(n:NATURAL:=4);
    PORT(D: IN std_logic_vector(n-1 DOWNTO 0);
    clk,rst: IN std_logic;
    Q: OUT std_logic_vector(n-1 DOWNTO 0));
END ENTITY four_bit_reg;

ARCHITECTURE behav OF four_bit_reg IS
BEGIN
    PROCESS(clk,rst)
        IF rst ='0' THEN
            Q <= (OTHERS =>'0');
        ELSIF clk'event AND clk='1' THEN
            Q <= D;
        END IF;
    END PROCESS;
END behav;
```

Shift Register

- SISO: Serial input -serial output
- SIPO: Serial input –parallel output
- PISO: Parallel input –serial output

SIPO shift register

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;

ENTITY sipo IS
    GENERIC(n:NATURAL:=8);
    PORT(clk,a:IN std_logic;
    out: OUT std_logic_vector(n-1 DOWNTO 0));
END ENTITY sipo;

ARCHITECTURE behav OF sipo IS
BEGIN
    PROCESS(clk)
        VARIABLE reg:=std_logic_vector(n-1 DOWNTO 0);
    BEGIN
        IF clk'EVENT AND clk='1' THEN
            reg:=reg(n-1 DOWNTO 0)&a;
        END IF;
        out<=reg;
    END PROCESS;
END behav;
```

4.4 计数器Counter

用T触发器构成计数器

- 可能会产生毛刺 复习2pptP66
- 注意代码和rtl图 课本p81

用普通加法做计数器

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.std_logic_unsigned.all;

ENTITY counter IS
    GENERIC(n:NATURAL:=8);
    PORT(clk,rst:IN std_logic;
    cnt: OUT std_logic_vector(n-1 DOWNTO 0));
END ENTITY counter;

ARCHITECTURE behav OF counter IS
BEGIN
    PROCESS(clk)
        VARIABLE cnt_m:=std_logic_vector(n-1 DOWNTO 0);
    BEGIN
        IF rst='0' THEN
            cnt:=(OTHERS=>'0');
        ELSIF clk'EVENT AND clk='1' THEN
            cnt=cnt+1;
        END IF;
        cnt<=cnt_m;
    END PROCESS;
END behav;
```

4.5 乘法器Multiplier

```
-- 三位乘法器，用于进行两个三位二进制数的乘法运算
-- 输出为6位二进制数
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.std_logic_unsigned.all;

ENTITY mul3 IS
    PORT(a,b:IN std_logic_vector(2 DOWNTO 0);
    c: OUT std_logic_vector(5 DOWNTO 0));
END ENTITY counter;

ARCHITECTURE behav OF mul3 IS
SIGNAL temp1: std_logic_vector (2 DOWNTO 0);
SIGNAL temp2: std_logic_vector (3 DOWNTO 0);
SIGNAL temp3: std_logic_vector (4 DOWNTO 0);
BEGIN
    temp1<=a WHEN a(0)='1' ELSE temp1<="000";
    temp2<=a WHEN a(1)='1' ELSE temp2<="0000";
    temp3<=a WHEN a(2)='1' ELSE temp3<="00000";
    c<=temp1+temp2+('0'&temp3);
END behav;
```