**“电子设计自动化(EDA)”实验报告(四)**

姓名：刘宗鑫 学号：201710413046 日期：2020.5.29

**一、实验名称：**用VHDL设计分频器,移位寄存器,状态机

**二、实验内容和实验目的**

实验内容：

1. 学习使用VDHL设计时序基本电路，分频器,移位寄存器,状态机，仿真验证，查看RTL 。

实验目的：学习用VHDL设计计数器，学会做完整的仿真。

**三、实现方法(或设计思想)及实验结果**

**实验一**

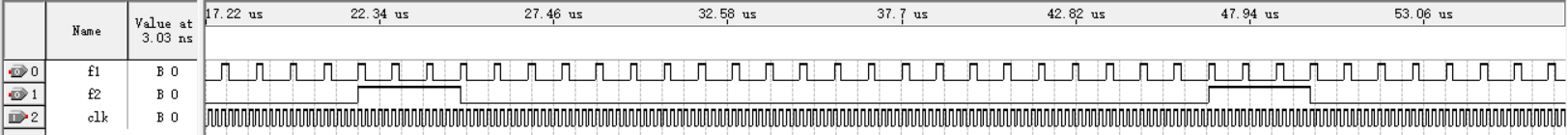
代码：

|  |  |
| --- | --- |
| Counter5 | library ieee;  use ieee.std\_logic\_1164.all;  use ieee.std\_logic\_unsigned.all;  entity counter5 is      port(          clk: in std\_logic;          cout: out std\_logic      );  end counter5;  architecture arc of counter5 is      signal cnt: integer range 0 to 4;  begin      process( clk )      begin          if clk'event and clk = '1' then              if cnt = 4 then                  cnt <= 0;              else                  cnt <= cnt + 1;              end if;          end if;      end process;      cout <= '1' when cnt = 4 else '0';  end arc; |
| Counter25 | library ieee;  use ieee.std\_logic\_1164.all;  use ieee.std\_logic\_unsigned.all;  entity counter25 is      port(          clk: in std\_logic;          cout: out std\_logic      );  end counter25;  architecture arc of counter25 is      signal cnt: integer range 0 to 24;  begin      process( clk )      begin          if clk'event and clk = '1' then              if cnt = 24 then                  cnt <= 0;              else                  cnt <= cnt + 1;              end if;          end if;      end process;      cout <= '1' when cnt > 21 else '0';  end arc; |
| Fdiv | library ieee;  use ieee.std\_logic\_1164.all;  use ieee.std\_logic\_unsigned.all;  entity fdiv is      port(          clk: in std\_logic;          f1, f2: out std\_logic      );  end fdiv;  architecture arc of fdiv is      component counter5          port(              clk: in std\_logic;              cout: out std\_logic          );      end component;      component counter25          port(              clk: in std\_logic;              cout: out std\_logic          );      end component;      signal div\_25\_en: std\_logic;  begin      freq\_div\_5: counter5 port map( clk, div\_25\_en  );      freq\_div\_25: counter25 port map( div\_25\_en, f2 );      f1 <= div\_25\_en;  end arc; |

说明：

* 为了构造分频器首先构造两个计数器
  + 模25计数器
  + 模5计数器
* 模5计数器以原始时钟为时钟，每计数满5个输出一个波形。
* 模25计数器以模5计数器的输出为时钟，每计数到22个的时候开始输出持续三个当前时钟的波形。

实验结果：



f1不是方波，其周期 T1= 5Tclk = 4Tclk(低电平) + 1Tclk(高电平) ; 其中Tclk是指一个clk时钟的时间长度。

每25个f1脉冲，就输出一个f2脉冲。f2不是方波，其周期 T2= 25T1 = 22T1(低电平) + 3T1(高电平);

**实验二**

代码：

|  |
| --- |
| library ieee;  use ieee.std\_logic\_1164.all;  use ieee.std\_logic\_unsigned.all;  entity sequencer is      port(          clk: in std\_logic;          cout: out std\_logic      );  end sequencer;  architecture arc1 of sequencer is      signal Q: std\_logic\_vector( 4 downto 1 ) := "0000";  begin      process(clk)          variable tmp : std\_logic\_vector( 4 downto 1 );      begin          if clk'event and clk = '1' then              tmp := Q;              for i in 2 to 4 loop                  Q(i) <= tmp(i-1);              end loop;              Q(1) <= (tmp(4) xor tmp(3)) or ( not (tmp(1) or tmp(2) or tmp(3) or tmp(4) ) );          end if;      end process;      cout <= Q(4);  end arc1;  architecture arc2 of sequencer is      signal Q: std\_logic\_vector( 4 downto 1 );  begin      process(clk)      begin          if clk'event and clk = '1' then              if Q = "0000" then                   Q <= "0001";              elsif Q = "0001" then                   Q <= "0010";              elsif Q = "0010" then                   Q <= "0100";              elsif Q = "0011" then                   Q <= "0110";              elsif Q = "0100" then                   Q <= "1001";              elsif Q = "0101" then                   Q <= "1011";              elsif Q = "0110" then                   Q <= "1101";              elsif Q = "0111" then                   Q <= "1111";              elsif Q = "1000" then                   Q <= "0001";              elsif Q = "1001" then                   Q <= "0011";              elsif Q = "1010" then                   Q <= "0101";              elsif Q = "1011" then                   Q <= "0111";              elsif Q = "1100" then                   Q <= "1000";              elsif Q = "1101" then                   Q <= "1010";              elsif Q = "1110" then                   Q <= "1100";              elsif Q = "1111" then                   Q <= "1110";              end if;          end if;      end process;      cout <= Q(4);  end arc2;  architecture arc3 of sequencer is      signal Q: std\_logic\_vector( 4 downto 1 ) := "0000";  begin      process(clk)          variable tmp : std\_logic\_vector( 4 downto 1 );      begin          if clk'event and clk = '1' then              tmp := Q;              Q(2) <= tmp(1);              Q(3) <= tmp(2);              Q(4) <= tmp(3);              Q(1) <= (tmp(4) xor tmp(3)) or ( not (tmp(1) or tmp(2) or tmp(3) or tmp(4) ) );          end if;      end process;      cout <= Q(4);  end arc3;  configuration cfg of sequencer is      for arc3      end for;  end cfg; |

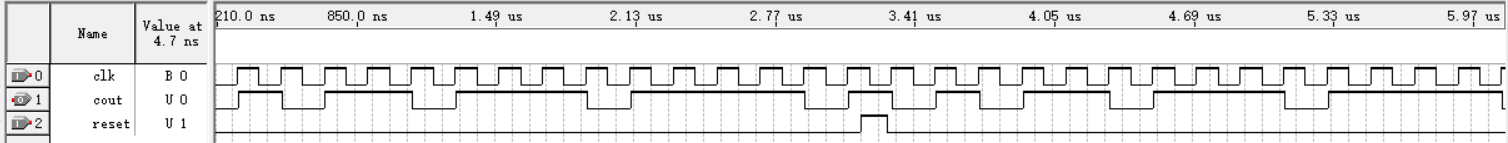
**说明：**

实验二使用了三种方法。使用configuration可以选择使用那种architecture进行仿真。

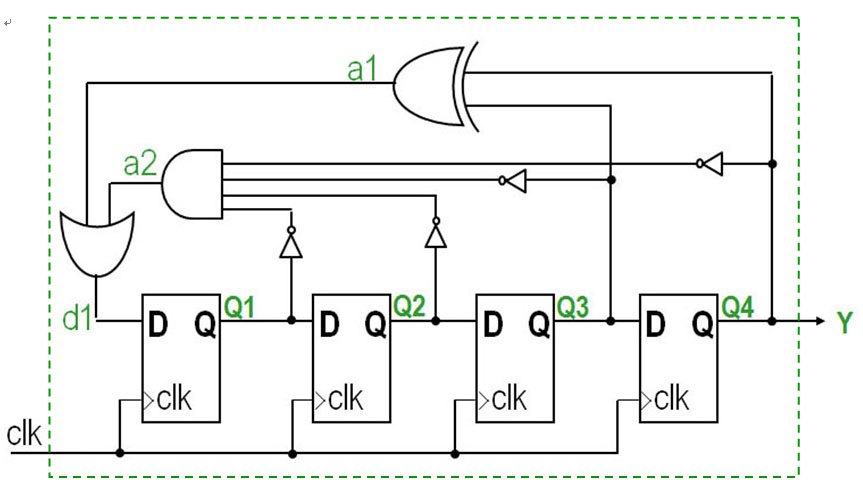
方法说明：

* 方法一：for loop语句。
  + 使用循环，使输入Q循环左移，对最低位特殊处理。
* 方法二：if
  + 计算出状态转移图，使用if枚举出每种情况。
* 方法三：if
  + 手动的模拟移位，一个一个的进行赋值。

实验结果：



即为下图寄存器组可以输出的波形：

**实验三**

代码：

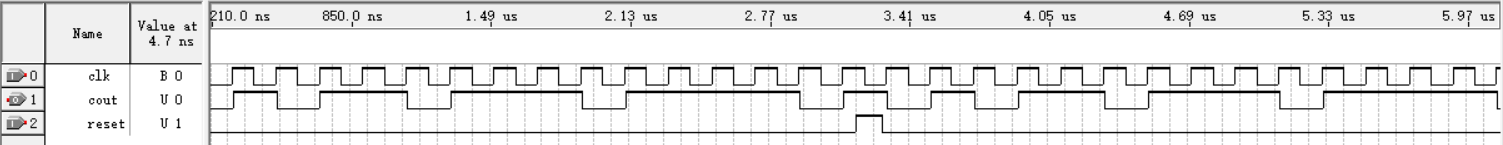
|  |
| --- |
| library ieee;  use ieee.std\_logic\_1164.all;  use ieee.std\_logic\_unsigned.all;  entity generator is      port(          clk, reset: in std\_logic;          cout: out std\_logic      );  end generator;  architecture arc1 of generator is      type state\_type is ( s1, s2, s3, s4, s5 );      signal state : state\_type;      signal cnt, num : integer;  begin      process( clk, reset )      begin          if reset = '1' then              cnt <= -1;              num <= 1;              state <= s1;          elsif clk'event and clk = '1' then              cnt <= cnt + 1;              if cnt = num then                  cnt <= 0;                  case state is                      when s1 =>                          num <= 2;                          state <= s2;                      when s2 =>                          num <= 3;                          state <= s3;                      when s3 =>                          num <= 4;                          state <= s4;                      when s4 =>                          num <= 5;                          state <= s5;                      when s5 =>                          num <= 1;                          state <= s1;                  end case;              end if;          end if;      end process;      cout <= '0' when cnt = 0 else '1';  end arc1;  architecture arc2 of generator is      type state\_type is(s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13,s14,s15,s16,s17,s18,s19);      signal state : state\_type;  begin      process(clk, reset)      begin          if reset='1' then              state<= s0;          elsif clk'event and clk='1' then             case state is               when s0  => state <= s1;  cout <= '0';               when s1  => state <= s2;  cout <= '1';               when s2  => state <= s3;  cout <= '0';               when s3  => state <= s4;  cout <= '1';               when s4  => state <= s5;  cout <= '1';               when s5  => state <= s6;  cout <= '0';               when s6  => state <= s7;  cout <= '1';               when s7  => state <= s8;  cout <= '1';               when s8  => state <= s9;  cout <= '1';               when s9  => state <= s10; cout <= '0';               when s10 => state <= s11; cout <= '1';               when s11 => state <= s12; cout <= '1';               when s12 => state <= s13; cout <= '1';               when s13 => state <= s14; cout <= '1';               when s14 => state <= s15; cout <= '0';               when s15 => state <= s16; cout <= '1';               when s16 => state <= s17; cout <= '1';               when s17 => state <= s18; cout <= '1';               when s18 => state <= s19; cout <= '1';               when s19 => state <= s0; cout <= '1';               when others => state <=s0; cout <= '0';            end case;         end if;      end process;  end arc2;  configuration cfg of generator is      for arc2      end for;  end cfg; |

说明：

使用两种方法

* 方法1：
  + 使用五个状态和一个计数变量。当计数变量达到规定值的时候，进入下一个状态，五个状态为一个循环。
* 方法2：
  + 使用二十个状态，每个状态有一个对应的输出，二十个为一个循环。

实验结果：



循环发生序列01011011101111011111。可以看到本来下个时钟应该为5个1但是此时reset按下，于是下一个时钟沿到来的时候，重新从第一个数字开始输出。

**四、总结**

* Configuration可以在一个文件中保存多个文件并方便切换
* 循环语句大大缩短了编写代码所需要的时间，同时可以省略人工计算，更不易出错。
* 状态转移时序关系较为复杂，编写代码时一定要想清楚再写。

**五、思考题**

查看编译系统根据代码实现的逻辑电路示意图:

如下：

