Lab2-Excerises5 斐波那契数列

高政 11912015

1、简述

1.1、 实验目的

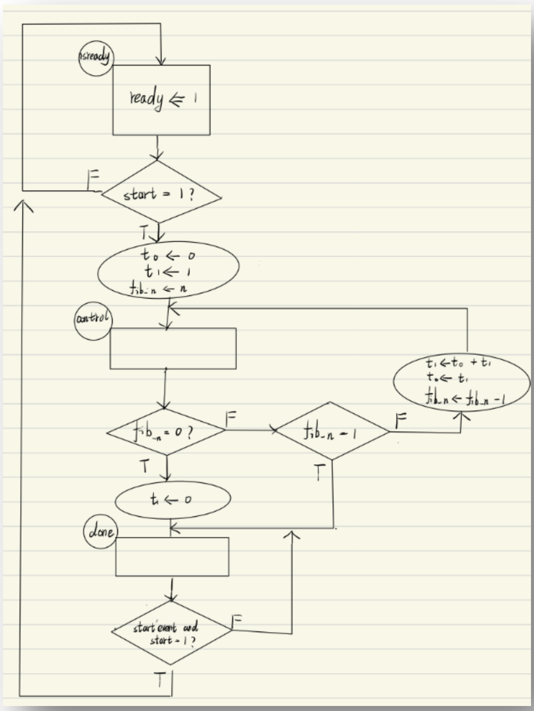
通过本实验了解带数据路径的有限状态机（FSMD）的基本设计流程，学习 VHDL 语言中的 case 语句的使用。

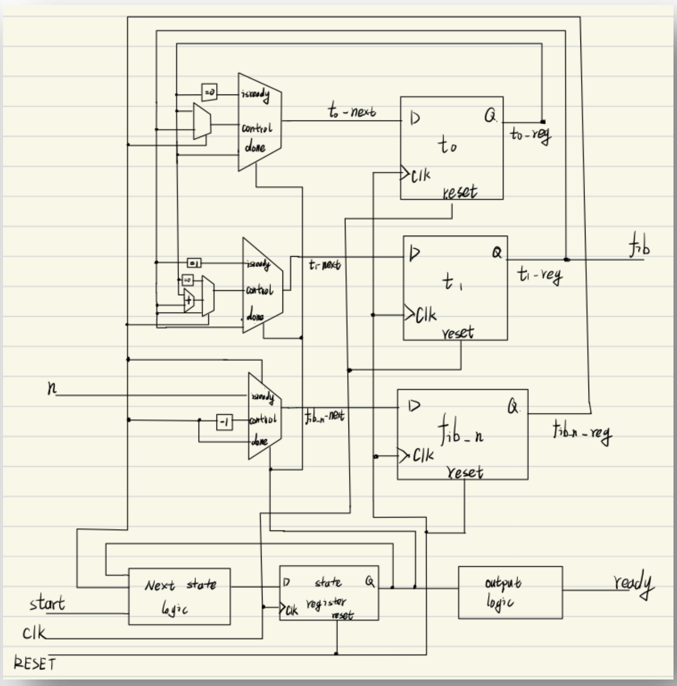
1.2、 斐波那契数列

$$fib(n)=\begin{cases}0 & \text{if } n=0\\ 1 & \text{if } n=1\\ fib(n-1)+fib(n-2) & \text{if } n>1\end{cases}$$

斐波那契数列数列的求解公式如上所示，本次实验中输入 n 是一个六比特的二进制信号，输出则是 43 比特的信号。

2、实验准备





斐波那契数列的 ASM 框图如左上图，其相应电路如右上图。

伪代码如下：

```
Fib(n):{
    if (n = 0) then
        fib = 0;
    else if (n = 1) then
        fib = 1;
    else {
         fib = Fib(n - 1) + Fib(n - 2);
    }
}
```

3、 VHDL 代码与 test bench 代码

斐波那契数列 VHDL 代码实现：

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;

entity fsmd is
    Port ( CLK : in STD_LOGIC;
           RESET : in STD_LOGIC;
           start : in std_logic;
           n : in STD_LOGIC_VECTOR (5 downto 0);
           ready : out std_logic;
           fib : out STD_LOGIC_VECTOR (42 downto 0));
end fsmd;

architecture Behavioral of fsmd is
type state_type is(isready, contro, done);
signal state, state_next : state_type;
signal t0_next, t1_next, t0_reg, t1_reg : std_logic_vector(42 downto 0);
signal fib_n_next, fib_n_reg : std_logic_vector(5 downto 0);
signal do : std_logic;
begin
clk_pro: process(CLK, RESET)is
begin
    if RESET = '1' then
        state <= isready;
        t0_reg <= (others => '0');
        t1_reg <= (others => '0');
        fib_n_reg <= n;
    elsif (CLK'event and clk = '1')then
        state <= state_next;
        t0_reg <= t0_next;
        t1_reg <= t1_next;
        fib_n_reg <= fib_n_next;
    end if;
end process clk_pro;
```

```
-- combinational circuit
process(state, state_next, t0_reg, t0_next, t1_reg, t1_next, fib_n_reg, fib_n_next) is
begin
        state_next <= state;
        t0_next <= t0_reg;
        t1_next <=t1_reg;
        fib_n_next <= fib_n_reg;
        ready <= '0';
        case state is
        when isready =>
                if start = '1' then
                        t0_next <= (others => '0');
                        t1_next <= (0 => '1', others => '0');
                        state_next <= contro;
                else
                        state_next <= isready;
                end if;
                ready <= '1';
        when contro =>
                if fib_n_reg = 0 then
                        t1_next <= (others => '0');
                        state_next <= done;
                elsif fib_n_reg = 1 then
                        state_next <= done;
                else
                        t0_next <= t1_reg;
                        t1_next <= t0_reg + t1_reg;
                        fib_n_next <= fib_n_reg - 1;
                        state_next <= contro;
                end if;
        when done =>
                state_next <= isready;
        end case;
end process;
fib <= t1_reg;
end Behavioral;
```

斐波那契数列 testbench 代码如下：

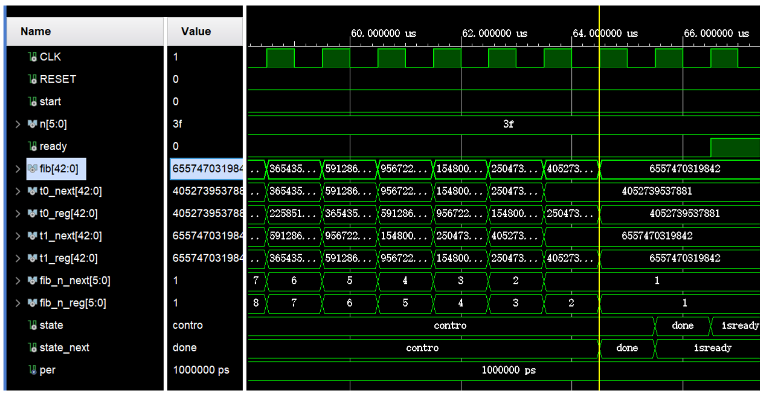
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity tb_fsmd is
end tb_fsmd;

architecture Behavioral of tb_fsmd is
```

```
	component fsmd is
	Port ( CLK : in STD_LOGIC;
		RESET : in STD_LOGIC;
		start : in std_logic;
		n : in STD_LOGIC_VECTOR (5 downto 0);
		ready : out std_logic;
		fib : out STD_LOGIC_VECTOR (42 downto 0));
	end component;

	--input
	signal CLK : std_logic := '0';
	signal RESET : std_logic := '0';
	signal start : std_logic := '0';
	signal n : std_logic_vector(5 downto 0) := "000000";

	--output
	signal ready : std_logic := '0';
	signal fib : std_logic_vector(42 downto 0) := (others => '0');

	constant per : time := 1 us;

begin
uut: fsmd
port map(
CLK => CLK,
RESET => RESET,
start => start,
n => n,
ready => ready,
fib => fib
);

clk_pro: process
begin
	CLK <= '0';
	wait for per/2;
	CLK <= '1';
	wait for per/2;
end process;

reset_pro: process
begin
	RESET <= '1';
	for i in 1 to 2 loop
```

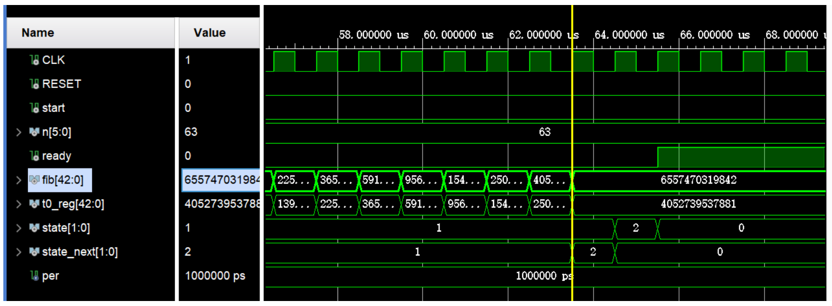
```
        wait until CLK = '1';
        end loop;
        RESET <= '0';
        wait;
end process;

in_pro: process
begin
        start <= '1';
        n <= "111111";
        wait for 10*per;
                start <= '0';
        wait;
end process;
end Behavioral;
```

4、 实验结果与分析

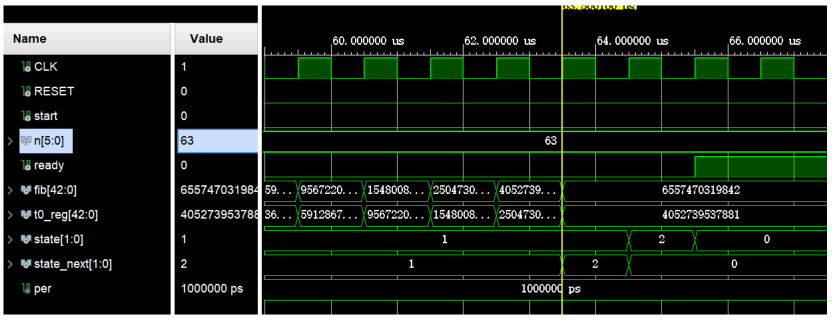
a) 行为仿真



b) 综合后功能仿真

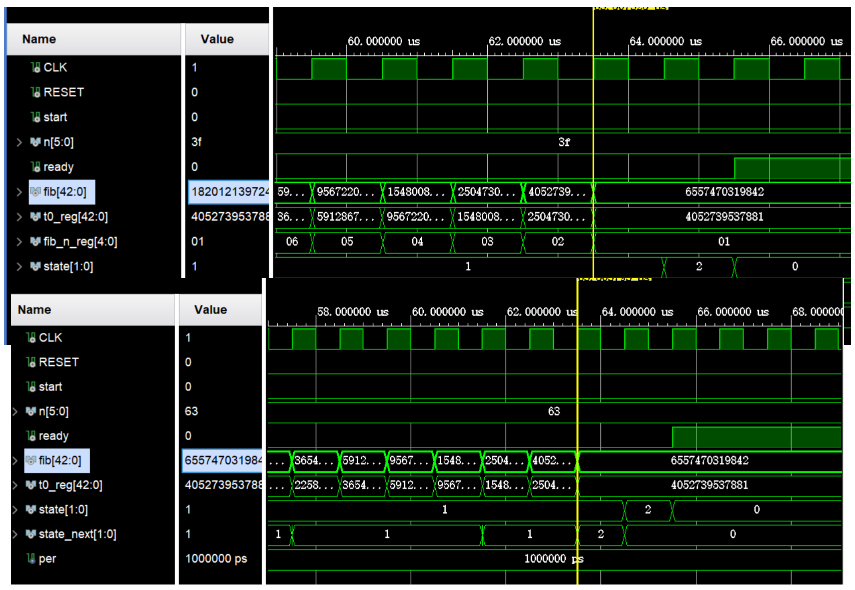


c) 综合后时序仿真



d) 布线后功能仿真

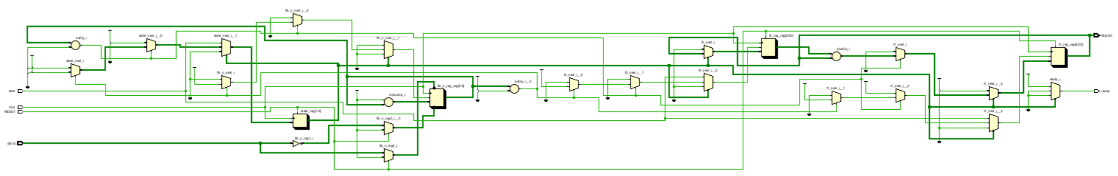
e) 布线后时序仿真



Name	Value
CLK	1
RESET	0
start	0
n[5:0]	3f
ready	0
fib[42:0]	182012139724
t0_reg[42:0]	405273953788
fib_n_reg[4:0]	01
state[1:0]	1

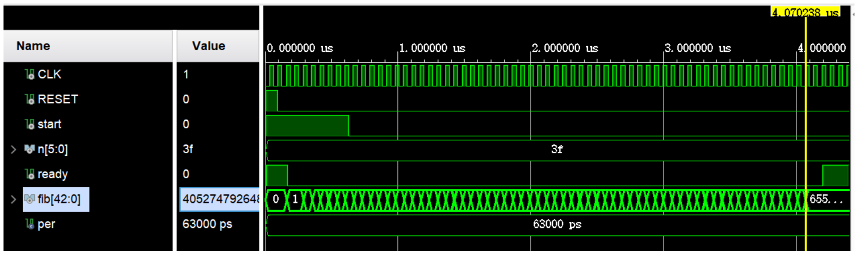
Name	Value
CLK	1
RESET	0
start	0
n[5:0]	63
ready	0
fib[42:0]	655747031984
t0_reg[42:0]	405273953788
state[1:0]	1
state_next[1:0]	1
per	1000000 ps

f) RTL 原理图



由以上结果可以看出，代码基本实现预期目标，能够完成计算斐波那契数列的效果。

g) 性能估计



Name	Value
CLK	1
RESET	0
start	0
n[5:0]	3f
ready	0
fib[42:0]	405274792648
per	63000 ps

经测试，本程序能正常工作的最短时钟周期为 63ns。

5、 实验结论

通过本次实验我了解了 FSMD 的基本设计流程和 VHDL 语句中的 case 语句的使用。