实验报告

实验题目：序列检测器　　　　姓名：牟真伟　　　　学号：PB20051061

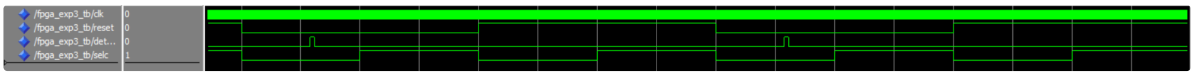
实验内容

设计两个序列产生器，分别产生一个含有待检测序列（）的序列和一个不含待检测序列的序列，通过一个2选1选择器选择其中一个序列送入序列检测器，序列检测器检测到待检测序列时则发出一个脉冲，系统时钟信号由实验箱上50Mhz经分频得到。

设计分析

采用模块电路的方法，将序列检测器，序列产生器，选择器，分频器分别实现其实体，在用元件例化的方式将各个元件联系起来。对于序列检测器和序列产生器，可以采用状态机的设计方法，分清系统各个状态和状态之间的转移规律，选择器采用行为描述法，分频器通过设置计数器，在时钟上升沿计数，计数器达到一定数值时反转输出信号，实现时钟分频。

仿真结果记录

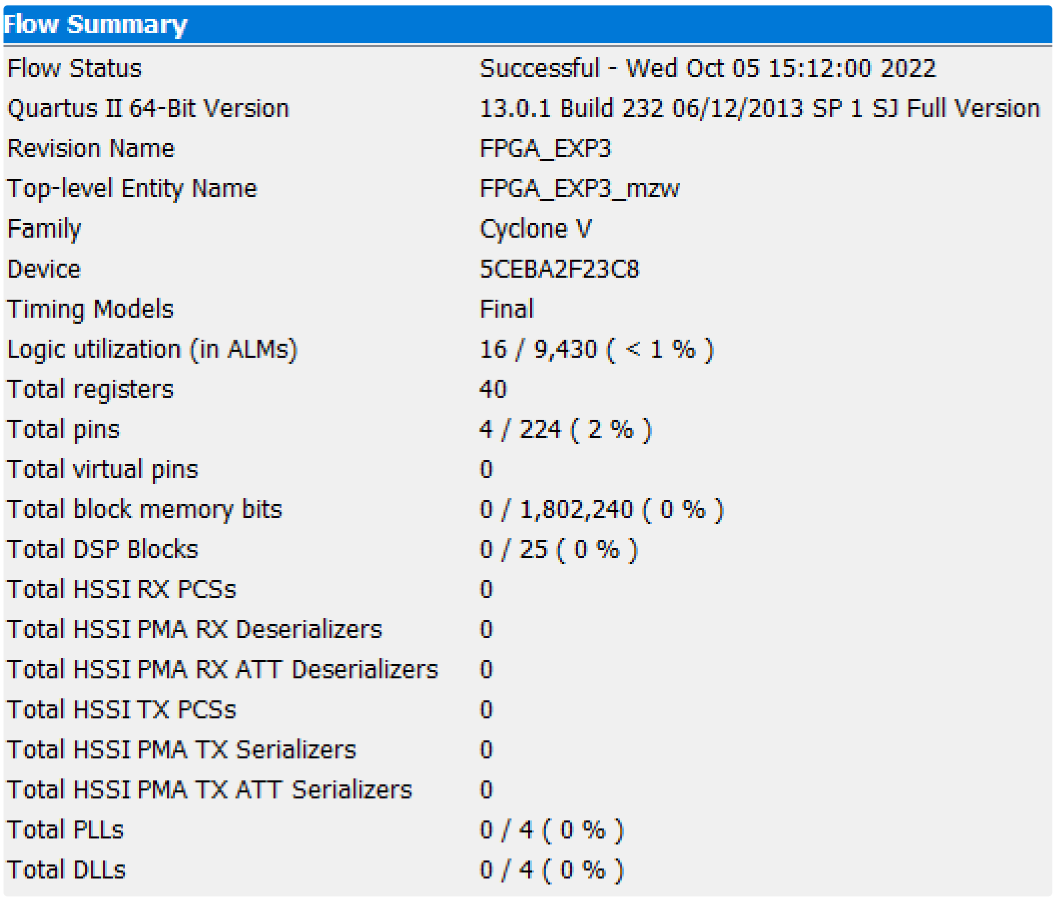


输入为时钟clk信号，选择selc信号，复位reset信号，检测器输出detector_out信号。

仿真的时钟信号采用4分频，reset信号高电平有效，当reset信号有效时，detector_out为低，序列产生器和序列检测器状态复位，当reset信号无效时，序列检测器工作，selc为低电平时，选择器将序列1送入序列检测器，由于序列1含有待检测序列，此时输出detector_out含有脉冲，selc为高电平时，选择器将序列2送入序列检测器，由于序列2不含待检测序列，此时输出detector_out不含脉冲。

中间结果记录

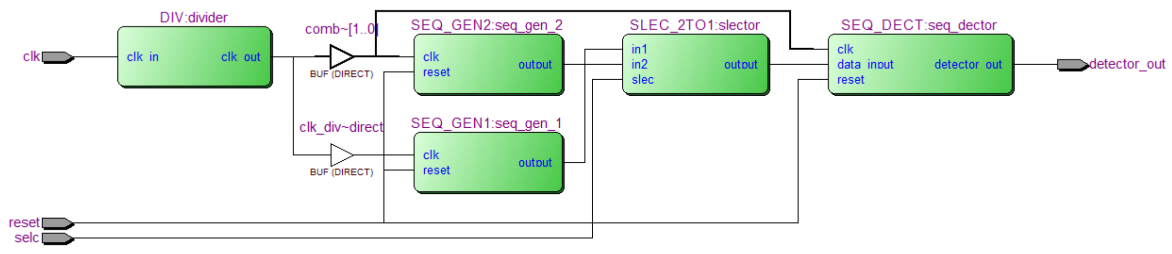
全编译后资源占用情况：



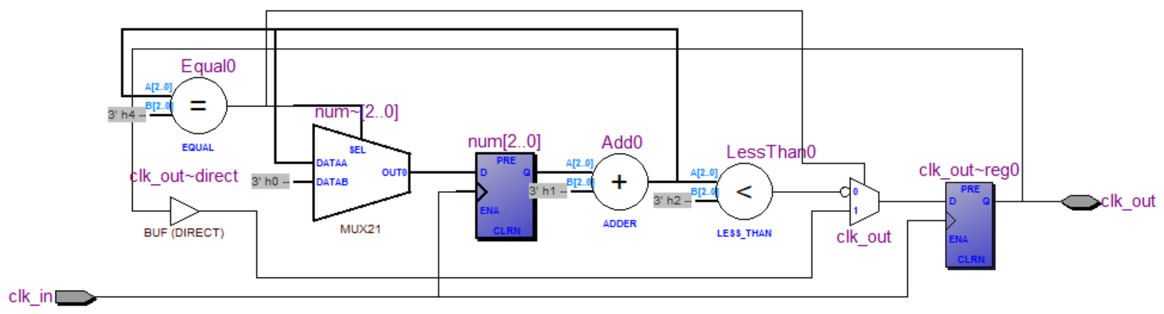
Flow Summary

Flow Status	Successful - Wed Oct 05 15:12:00 2022
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Full Version
Revision Name	FPGA_EXP3
Top-level Entity Name	FPGA_EXP3_mzw
Family	Cyclone V
Device	5CEBA2F23C8
Timing Models	Final
Logic utilization (in ALMs)	16 / 9,430 (< 1 %)
Total registers	40
Total pins	4 / 224 (2 %)
Total virtual pins	0
Total block memory bits	0 / 1,802,240 (0 %)
Total DSP Blocks	0 / 25 (0 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI PMA RX ATT Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total HSSI PMA TX ATT Serializers	0
Total PLLs	0 / 4 (0 %)
Total DLLs	0 / 4 (0 %)

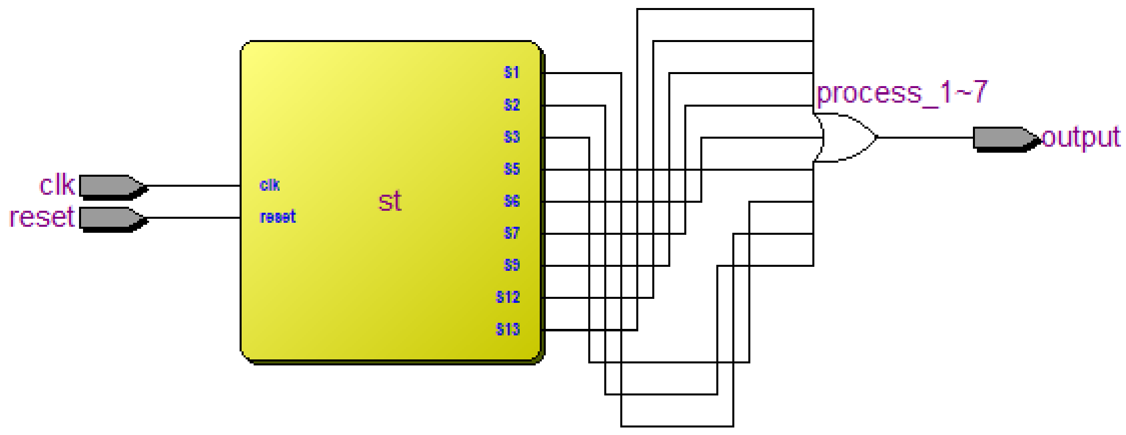
电路总RTL结构图：



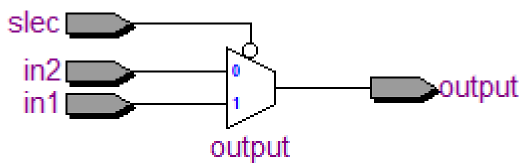
分频器RTL结构图：



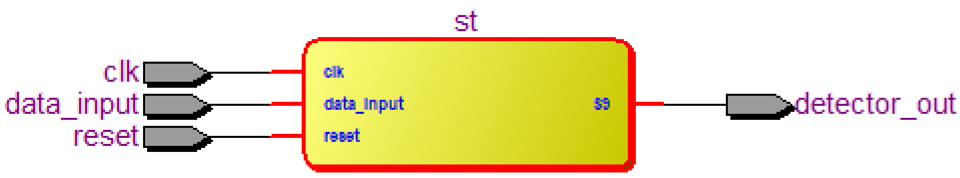
序列产生器RTL结构图：



选择器RTL结构图：

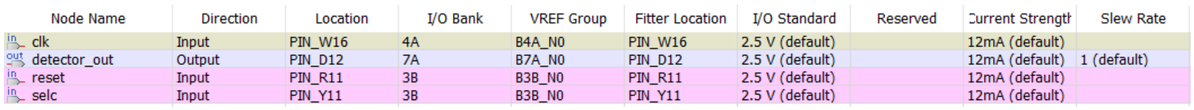


序列检测器RTL结构图：



FPGA验证结果记录

实验管脚约束情况如下：



Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strengt	Slew Rate
clk	Input	PIN_W16	4A	B4A_N0	PIN_W16	2.5 V (default)		12mA (default)	
detector_out	Output	PIN_D12	7A	B7A_N0	PIN_D12	2.5 V (default)		12mA (default)	1 (default)
reset	Input	PIN_R11	3B	B3B_N0	PIN_R11	2.5 V (default)		12mA (default)	
selc	Input	PIN_Y11	3B	B3B_N0	PIN_Y11	2.5 V (default)		12mA (default)	

clk采用实验箱上50Mhz的时钟信号，为能看清实验现象，采取10Mhz分频。detector_out为LED灯D0，当检测到序列时，detector_out输出一脉冲，LED灯点亮闪烁时钟周期。reset为拨码快关DIP1，selc为拨码快关DIP0。

实验现象如下：

reset为1时，无论selc为1还是0，输出led灯均不点亮。

reset为0，selc为0时，输出led灯周期闪烁。

reset为0，selc为1时，输出led灯不点亮。

实验总结

本次实验通过序列检测器和序列产生器的实现，学习了使用状态机这种电路设计方法，通过分频器和选择器，复习了常用电路模块的设计。本次实验通过设计电路模块，再用元件例化的方式将各个元件连接起来。本次实验所需电路模块较多，最好对每个模块都编写一个测试仿真文件，及时测试。单独对某个模块进行测试时，要将该实体设置为顶层实体，否则仿真时测试文件不能将引脚绑定到该实体上，导致输出一直为未定状态。

VHDL源代码

```vhdl
-- FPGA_EXP3 顶层实体 @PB20051061牟真伟

library ieee;
use ieee.std_logic_1164.all;

entity FPGA_EXP3_mzw is
    port(
        clk :IN std_logic;
        reset :IN std_logic;
        selc :IN std_logic;
        detector_out :OUT std_logic
    );
end FPGA_EXP3_mzw;

architecture arch_FPGA_EXP3 of FPGA_EXP3_mzw is
    component SLEC_2TO1 port(
            in1 :IN std_logic;
            in2 :IN std_logic;
            slec :IN std_logic;
            output :OUT std_logic
        );
    end component;

    component SEQ_GEN1 port(
            clk :IN std_logic;
            reset :IN std_logic;
            output :OUT std_logic
        );
    end component;

    component SEQ_GEN2 port(
            clk :IN std_logic;
            reset :IN std_logic;
            output :OUT std_logic
    );
    end component;

    component SEQ_DECT port(
        clk :IN std_logic;
        reset :IN std_logic;
        data_input :IN std_logic;
        detector_out :OUT std_logic
    );
    end component;

    component DIV port(
        clk_in :IN std_logic;
        clk_out :INOUT std_logic
    );
    end component;

    signal sgn1 :std_logic;
    signal sgn2 :std_logic;
    signal sgn :std_logic;
    signal clk_div :std_logic;
begin

    divider: DIV port map(clk, clk_div);
    seq_gen_1: SEQ_GEN1 port map(clk_div, reset, sgn1);
    seq_gen_2: SEQ_GEN2 port map(clk_div, reset, sgn2);
    seq_dector: SEQ_DECT port map(clk_div, reset, sgn, detector_out);
    slector: SLEC_2TO1 port map(sgn1,sgn2,selc,sgn);
end arch_FPGA_EXP3;

-- FPGA_EXP3 顶层实体test bench @PB20051061牟真伟

library ieee;
use ieee.std_logic_1164.all;

entity FPGA_EXP3_tb is
end FPGA_EXP3_tb;

architecture arch_FPGA_EXP3_tb of FPGA_EXP3_tb is
    component FPGA_EXP3_mzw port(
        clk :IN std_logic;
        reset :IN std_logic;
        selc :IN std_logic;
        detector_out :OUT std_logic
    );
    end component;
    signal clk :std_logic;
    signal reset :std_logic;
    signal detector_out :std_logic;
    signal selc :std_logic;

begin
    exp3: FPGA_EXP3_mzw port map(clk,reset,selc,detector_out);
    process
    begin
    selc <= '0';
    wait for 2000 ns;
    selc <= '1';
    wait for 2000 ns;
    end process;

    process
    begin
    reset <= '0';
    wait for 4000 ns;
    reset <= '1';
    wait for 4000 ns;
    end process;

    process
    begin
            clk <= '0';
            wait for 10 ns;
            clk <= '1';
            wait for 10 ns;
    end process;

end arch_FPGA_EXP3_tb;

 -- FPGA_EXP3 分频器 @PB20051061牟真伟

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity DIV  is
    port(
        clk_in :IN std_logic;
        clk_out :INOUT std_logic
    );
end div;

architecture arch_DIV of DIV is
begin
    process(clk_in)
            variable num: integer range 0 to 4;  --仿真4分频
            --variable num: integer range 0 to 10000000; --实际10M
分频
    begin
         if (clk_in'event and clk_in='1') then
                        num := num + 1;
              if  (num = 4)  then -- 仿真4分频
             --if  (num = 10000000)  then--实际10M分频
                        num := 0;
              elsif  (num < 2)  then -- 仿真4分频
```

```vhdl
              -- elsif  (num < 5000000)  then --实际10M分频
                        clk_out <= '0';
               else
                        clk_out <= '1';
               end if;
          end if;
      end process;
end arch_DIV;

-- FPGA_EXP3 2to1选择器 @PB20051061牟真伟

library ieee;
use ieee.std_logic_1164.all;

entity SLEC_2TO1 is
    port(
        in1 :IN std_logic;
        in2 :IN std_logic;
        slec :IN std_logic;
        output :OUT std_logic
    );
end SLEC_2TO1;

architecture arch_SLEC_2TO1 of SLEC_2TO1 is
begin
    output <= in1 when slec='0' else in2;
end arch_SLEC_2TO1;

-- FPGA_EXP3 序列检测器 @PB20051061牟真伟

library ieee;
use ieee.std_logic_1164.all;

entity SEQ_DECT is
    port(
        clk :IN std_logic;
        reset :IN std_logic;
        data_input :IN std_logic;
        detector_out :OUT std_logic
    );
end SEQ_DECT;

architecture arch_SEQ_DECT of SEQ_DECT is
    type states is (S0,S1,S2,S3,S4,S5,S6,S7,S8,S9);
    signal st :states;
begin
    detector_out <= '1' when st=S9 else '0';
    -- 状态寄存器和次态逻辑
    process(clk,reset)
    begin
```

```
if(reset = '1') then
    st <= S0;
elsif(clk'event and clk='1') then
    case st is
        when s0 => if(data_input = '1') then
                            st <= S1;
                        else
                            st <= S0;
                        end if;
        when s1 => if(data_input = '1') then
                            st <= S2;
                        else
                            st <= S0;
                        end if;
        when s2 => if(data_input = '1') then
                            st <= S3;
                        else
                            st <= S0;
                        end if;
        when s3 => if(data_input = '1') then
                            st <= S3;
                        else
                            st <= S4;
                        end if;
        when s4 => if(data_input = '1') then
                            st <= S5;
                        else
                            st <= S0;
                        end if;
        when s5 => if(data_input = '1') then
                            st <= S2;
                        else
                            st <= S6;
                        end if;
        when s6 => if(data_input = '1') then
                            st <= S1;
                        else
                            st <= S7;
                        end if;
        when s7 => if(data_input = '1') then
                            st <= S8;
                        else
                            st <= S0;
                        end if;
        when s8 => if(data_input = '1') then
                            st <= S9;
                        else
                            st <= S0;
                        end if;
        when s9 => if(data_input = '1') then
```

```vhdl
                                    st <= S1;
                                else
                                    st <= S0;
                                end if;
            end case;
        end if;
    end process;

    -- 输出逻辑
end arch_SEQ_DECT;

-- FPGA_EXP3 序列产生器1 @PB20051061牟真伟

library ieee;
use ieee.std_logic_1164.all;

entity SEQ_GEN1 is
    port(
        clk :IN std_logic;
        reset :IN std_logic;
        output :OUT std_logic
    );
end SEQ_GEN1;

architecture arch_SEQ_GEN1 of SEQ_GEN1 is
    type states is (S0,S1,S2,S3,S4,S5,S6,S7,S8,S9,S10,S11,S12,S13,S14);
    signal st :states;
begin
    -- 状态寄存器和次态逻辑
    process(clk,reset)
    begin
        if(reset = '1') then
            st <= S14;
        elsif(clk'event and clk='1') then
            case st is
                when s0 => st <= S1;
                when s1 => st <= S2;
                when s2 => st <= S3;
                when s3 => st <= S4;
                when s4 => st <= S5;
                when s5 => st <= S6;
                when s6 => st <= S7;
                when s7 => st <= S8;
                when s8 => st <= S9;
                when s9 => st <= S10;
                when s10 => st <= S11;
                when s11 => st <= S12;
                when s12 => st <= S13;
                when s13 => st <= S14;
                when s14 => st <= S0;
            end case;
        end if;
    end process;

    -- 输出逻辑
    process(st)
    begin
        if(st=S1 or st=S2 or st=S3 or st=S5 or st=S6 or st=S7 or st=S9 or
st=S12 or st=S13) then
            output <= '1';
        else
            output <= '0';
        end if;
    end process;
end arch_SEQ_GEN1;

-- FPGA_EXP3 序列产生器2 @PB20051061牟真伟

library ieee;
use ieee.std_logic_1164.all;

entity SEQ_GEN2 is
    port(
        clk :IN std_logic;
        reset :IN std_logic;
        output :OUT std_logic
    );
end SEQ_GEN2;

architecture arch_SEQ_GEN2 of SEQ_GEN2 is
    type states is (S0,S1,S2,S3,S4,S5,S6,S7,S8);
    signal st :states;
begin
    -- 状态寄存器和次态逻辑
    process(clk,reset)
    begin
        if(reset = '1') then
            st <= S7;
        elsif(clk'event and clk='1') then
            case st is
                when s0 => st <= S1;
                when s1 => st <= S2;
                when s2 => st <= S3;
                when s3 => st <= S4;
                when s4 => st <= S5;
                when s5 => st <= S6;
                when s6 => st <= S7;
                when s7 => st <= S8;
                when s8 => st <= S0;
            end case;
        end if;
    end process;

    -- 输出逻辑
    process(st)
    begin
        if(st=S0 or st=S1 or st=S5 or st=S6) then
            output <= '1';
        else
            output <= '0';
        end if;
    end process;
end arch_SEQ_GEN2;
```