实验二：序列检测器

一. 实验内容

构造一个完整的结构，包括输入端口：clk, reset, sel，和输出端口detector_out。结构内部两个序列产生器以产生两个不同的序列，可以通过输出端口sel进行选择；一个序列检测器用来检测特定的序列，每检测到一次对应序列，detector_out输出一次信号。clk为输入时钟。

二. 设计分析

编程思路：将完整结构拆分为序列发生器1/2、序列检测器、分频器、选择器几个单元模块分别实现。

序列检测器：采用Mealy型有限状态机模型，输入序列作为条件，转移到特定状态代表检测到完整序列，输出信号。

序列产生器：采用Moore型有限状态机模型，产生对应序列。

三. VHDL源代码

顶层实体：

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

entity FPGA_EXP3_dxh is
    port(
        clk: in std_logic;
        reset: in std_logic;
        sel: in std_logic;
        seq_output: out std_logic;
        detector_out: out std_logic);
end FPGA_EXP3_dxh;

architecture arch_FPGA_EXP3 of FPGA_EXP3_dxh is
    component seq_gen1
        port (
        clk: in std_logic;
        reset: in std_logic;
        seq1: out std_logic);
    end component;

    component seq_gen2
        port (
        clk: in std_logic;
        reset: in std_logic;
        seq2: out std_logic);
    end component;
```

```
    component selector
        port(
        sel: in std_logic;
        seq_1: in std_logic;
        seq_2: in std_logic;
        seq: out std_logic);
    end component;

    component seq_detector
        port (
            clk: in std_logic;
            reset: in std_logic;
            datainput: in std_logic;
            detector_out: out std_logic);
    end component;

    component FD
        port(clk: in std_logic;
        newclk: out std_logic
    );
    end component;
    signal seq1, seq2, seq, newclk: std_logic;
begin
    u1: seq_gen1 port map(newclk, reset, seq1);
    u2: seq_gen2 port map(newclk, reset, seq2);
    u3: selector port map(sel, seq1, seq2, seq);
    u4: seq_detector port map(newclk, reset, seq, detector_out);
    u5: FD port map (clk, newclk);
    seq_output <= seq;
end arch_FPGA_EXP3;
```

序列检测器：

检测输入序列，如果出现待检测序列“111010011”，输出一个周期的高电平。如果未检测到序列“111010011”，则输出一直为低电平。

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

entity seq_detector is
    port (
        clk: in std_logic;
        reset: in std_logic;
        datainput: in std_logic;
        detector_out: out std_logic);
end seq_detector;

architecture arch_seq_detector of seq_detector is
    type states is (s0, s1, s2, s3, s4, s5, s6, s7, s8, s9);
    signal st: states := s0;
begin
    process(clk, reset)
    begin
```

```
        if(reset='0') then
            st <= s0;
        elsif(clk'event and clk='1') then
            case st is
                when s0 =>
                    if(datainput='1') then
                        st <= s1;
                    else st <= s0;
                    end if;
                when s1 =>
                    if(datainput='1') then
                        st <= s2;
                    else st <=s0;
                    end if;
                when s2 =>
                    if(datainput='1') then
                        st <= s3;
                    else st <=s0;
                    end if;
                when s3 =>
                    if(datainput='0') then
                        st <= s4;
                    else st <=s3;
                    end if;
                when s4 =>
                    if(datainput='1') then
                        st <= s5;
                    else st <=s0;
                    end if;
                when s5 =>
                    if(datainput='0') then
                        st <= s6;
                    else st <=s2;
                    end if;
                when s6 =>
                    if(datainput='0') then
                        st <= s7;
                    else st <=s1;
                    end if;
                when s7 =>
                    if(datainput='1') then
                        st <= s8;
                    else st <=s0;
                    end if;
                when s8 =>
                    if(datainput='1') then
                        st <= s9;
                    else st <=s0;
                    end if;
                when s9 =>
                    if(datainput='0') then
                        st <= s1;
                    else st <=s0;
                    end if;
            end case;
```

```
        end if;
    end process;
    with st select
    detector_out <=
        '1' when s9,
        '0' when others;
end arch_seq_detector;
```

序列产生器1:

产生顺序为“011101110100110”的周期性序列（包含待检测序列“111010011”）

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

entity seq_gen1 is
    port (
        clk: in std_logic;
        reset: in std_logic;
        seq1: out std_logic);
end seq_gen1;

architecture arch_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
    process(clk, reset)
    begin
        if(reset='0') then
            st <= s0;
        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;
    seq1 <=
        '0' when st=s0 else
        '1' when st=s1 else
        '1' when st=s2 else
        '1' when st=s3 else
        '0' when st=s4 else
        '1' when st=s5 else
        '1' when st=s6 else
        '1' when st=s7 else
        '0' when st=s8 else
        '1' when st=s9 else
        '0' when st=s10 else
        '0' when st=s11 else
        '1' when st=s12 else
        '1' when st=s13 else
        '0';
end arch_gen1;
```

序列产生器2:

产生顺序为“0101”的周期性序列（不包含待检测序列“111010011”）

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

entity seq_gen2 is
    port (
        clk: in std_logic;
        reset: in std_logic;
        seq2: out std_logic);
end seq_gen2;

architecture arch_gen2 of seq_gen2 is
    type states is (S0, S1, S2, S3, S4, S5, S6, S7, S8, S9, S10, S11, S12,
S13, S14);
    signal st: states;
begin
    process(clk, reset)
    begin
        if(reset='0') then
            st <= s0;
        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;
    seq2 <=
        '0' when st=s0 else
        '1' when st=s1 else
        '0' when st=s2 else
        '1' when st=s3 else
        '0' when st=s4 else
        '1' when st=s5 else
        '0' when st=s6 else
        '1' when st=s7 else
        '0' when st=s8 else
        '1' when st=s9 else
        '0' when st=s10 else
        '1' when st=s11 else
        '0' when st=s12 else
        '1' when st=s13 else
        '0';
end arch_gen2;
```

分频器：

实现2500000被分频，将试验箱上的50MHz主频分频为20Hz

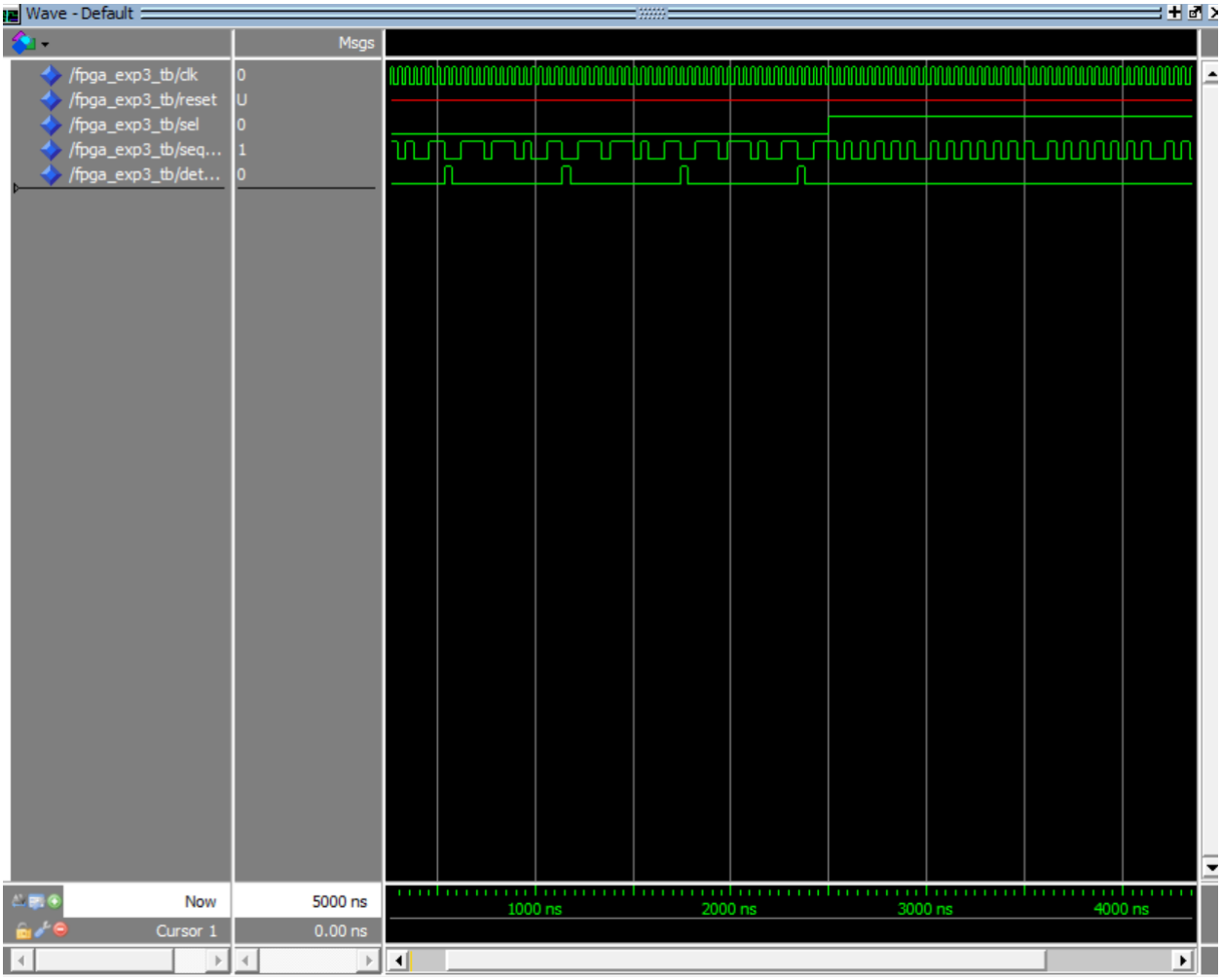
```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity FD is
    port(clk: in std_logic;
    newclk: out std_logic
    );
end FD;

architecture arch_FD of FD is
    signal num:integer range 0 to 2499999;
begin
    process(clk)
    begin
        if(clk'event and clk = '1') then
            if (num = 2499999) then
                num <= 0;
            elsif (num < 1250000) then
                newclk <= '0';
                num <= num + 1;
            else
                newclk <= '1';
                num <= num + 1;
            end if;
        end if;
    end process;
end arch_FD;
```

序列选择器:

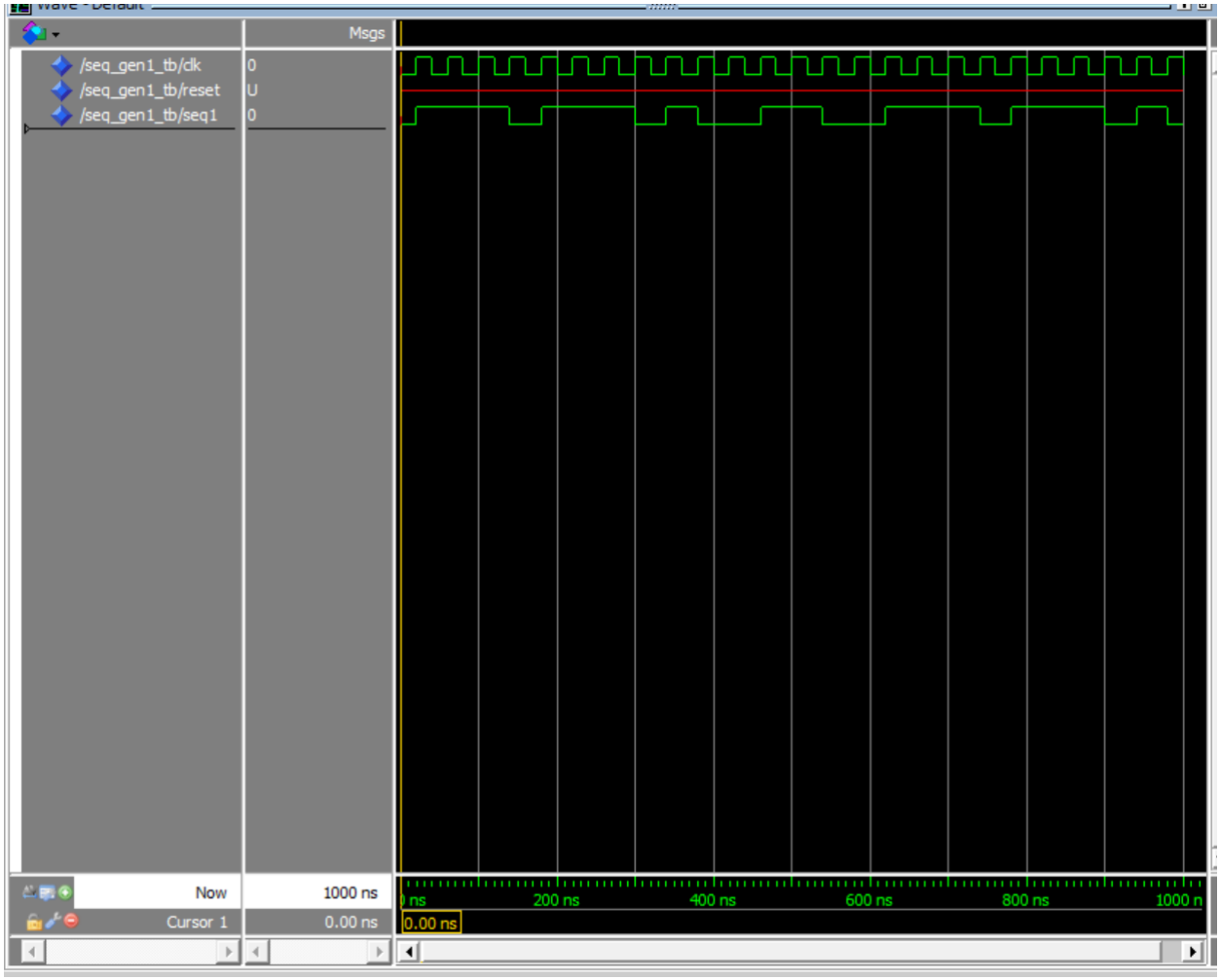
在两种序列中选择一个输入序列检测器

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

entity selector is
    port(
        sel: in std_logic;
        seq_1: in std_logic;
        seq_2: in std_logic;
        seq: out std_logic);
end selector;

architecture arch_select of selector is
begin
    seq <=
        seq_1 when sel='0' else
        seq_2;
end arch_select;
```

四. 仿真结果



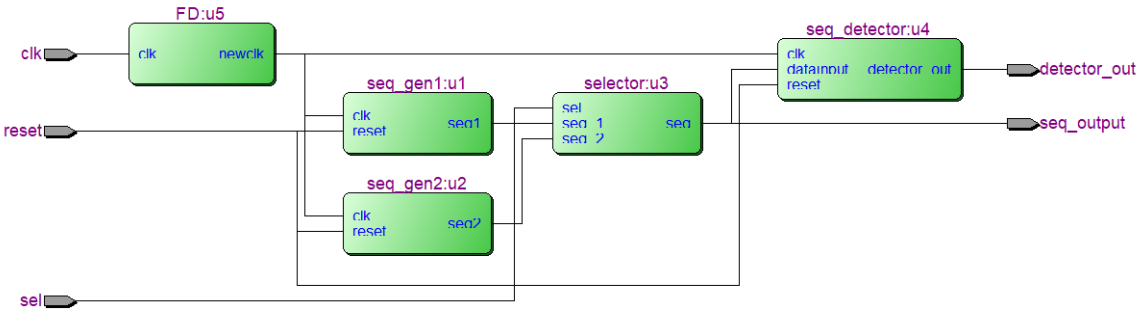
顶层实体仿真(仿真时未加分频器)



序列产生器1的仿真结果

五. 中间结果

1.RTL电路图



2.资源占用信息

Flow Summary	
Flow Status	Successful - Fri Oct 14 16:36:55 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
Family	Cyclone V
Device	5CEFA2F23C8
Timing Models	Final
Logic utilization (in ALMs)	33 / 9,430 (< 1 %)
Total registers	63
Total pins	5 / 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 %)

六. 硬件验证

管脚锁定情况:

端口	管脚
clk	PIN_W14
detector_out	PIN_D12
reset	PIN_Y11
sel	PIN_R11
seq_output	PIN_G12

LED灯D1显示了当前的序列,LED亮代表1,灭代表0.D0显示输出信号,每检测到一次完整的待检测序列D0亮一次,然后熄灭,周期性重复.通过拨码开关DIP1可以选择当前序列,DIP0可以置位,使输出一直为0.

七. 实验总结

在仿真过程中,我尝试创建多个test_banch分别对各个部件分别进行仿真以验证其功能.但部分元件在仿真中并不能输出真实的波形.请教老师后得知原因:即使在一个项目中创建多个test_banch,但是它们总是默认从顶层实体出发,这会造成对个别原件的仿真不能正常进行.

解决办法:

对需要仿真验证的各个元件分别创建一个工程目录,组织在顶层项目的文件夹下,则可以对各个部分分别仿真.并在顶层项目结构中添加各个部件的程序设计文件,可以进行整体仿真.