Русенски университет „Ангел Кънчев“ Факултет „Електротехника, електроника и автоматика“
Дисциплина „Технология на проектирането “

Курсова задача

име: Георги Живков Чапъров
фак. номер: 223025
група: 26 А
курс 2

Проверил: ……

/гл. ас. д-р Николай Костадинов/

2024.21.05

1. Задание

Вариант 8

ТЕХНОЛОГИЯ НА ПРОЕКТИРАНЕТО
Курсова работа – Етап 2
Задание

Да се проектира блок за управление на асансьор, движещ се между два етажа.
Блокът да се реализира като краен автомат с памет със следните сигнали:
Входове:
- **Up** – бутон "Старт нагоре" (бутон BTNU);
- **Dn** – бутон "Старт надолу" (бутон BTND);
- **SensorDn** - вход от датчик за долно положение - етаж 1 (микропревключвател SW0);
- **SensorUp** - вход от датчик за горно положение - етаж 2 (микропревключвател SW1);

Изходи:
- **MoveUp** – управление на движение нагоре (светодиод LD0);
- **MoveDn** – управление на движение надолу (светодиод LD1);

Състоянието на автомата да се извежда чрез седем-сегментния индикатор на макета.
Указание:
Могат да се предвидят следните състояния на автомата:
- **S0** - положение етаж 1 (спрял);
- **S1** - движение нагоре;
- **S2** - положение етаж 2 (спрял);
- **S3** - движение надолу;

Блокът да се реализира като автомат на Mealy.

2. VHDL Описание

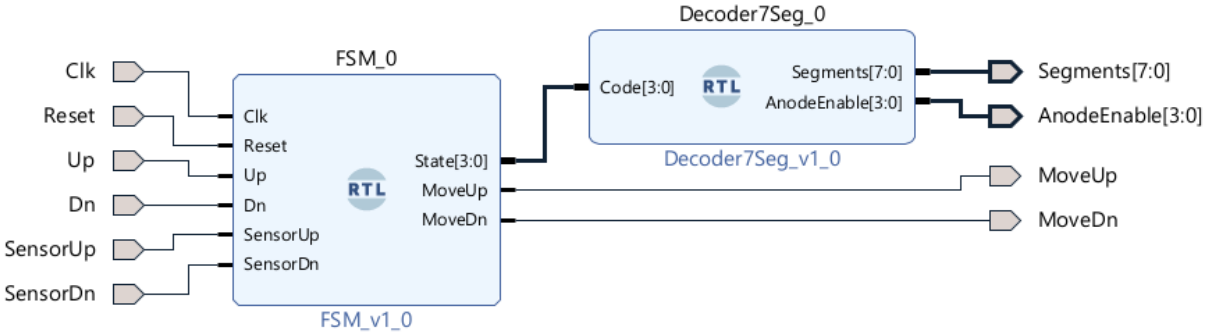
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity FSM is
    port(
        Clk       : in  STD_LOGIC;
        Reset     : in  STD_LOGIC;  -- Вход за рестартиране.
        Up        : in  STD_LOGIC;  -- Вход за отиване нагоре.
        Dn        : in  STD_LOGIC;  -- Вход за отиване надолу.
        SensorUp  : in  STD_LOGIC;  -- Вход оказващ че текущата позиция на асансьора е първи етаж.
        SensorDn  : in  STD_LOGIC;  -- Вход оказващ че текущата позиция на асансьора е нулев етаж.
        State     : out STD_LOGIC_VECTOR(3 downto 0); -- Изход към декодера.
        MoveUp    : out STD_LOGIC;  -- Изход оказващ че асансьора трябва да отиде нагоре.
        MoveDn    : out STD_LOGIC   -- Изход оказващ че асансьора трябва да отиде надолу.
    );
end FSM;

architecture Behavioral of FSM is
type STATE_TYPE is (S0, S1, S2, S3, S4); -- Дефиниране на видовете състояния.
signal CurrentState, NextState : STATE_TYPE;  -- Дефиниране на променливи служещи за оказване на текущото и следващото състояние.

begin
MEM:
    -- Блок памет на състоянието
    process (Reset, Clk)
    begin
        if (Reset = '1') then -- при сегнал за рестартирване
            CurrentState <= S0; -- Задаваме текущото състояние да е началното състояние.
        elsif (falling_edge(Clk)) then -- при сигнал от клока
            CurrentState <= NextState;  -- задаваме текущото състояние да е равно на следващото състояние.
        end if ;
    end process MEM;

NEXT_STATE_LOGIC:
    -- Логика за определяне на следващото състояние
    process (CurrentState, Up, Dn, SensorUp, SensorDn)
    begin
        NextState <= CurrentState; -- Задаваме следаващото състояниеда е равно на текущото
        MoveUp <=  '0'; -- Задаваме изхода за отиване нагоре да е "изключен" (Оказваме на асансьора че не трябва да ходи нагоре).
        MoveDn <=  '0'; -- Задаваме изхода за отиване надолу да е "изключен" (Оказваме на асансьора че не трябва да ходи надолу).
        case CurrentState is
            when S0=>                -- Когато сме в първо състояние.
                if (Up = '1' and SensorDn = '1') then   -- Ако имаме сигнал за отиване нагоре и сме на нулев етаж.
                    NextState <= S1; -- Задаваме следаващото състояниеда е равно на второ състояние.
                    MoveUp <=  '1';  -- Задаваме изхода за отиване нагоре да е "включен" (Оказваме на асансьора че трябва да ходи нагоре).
                    MoveDn <=  '0';  -- Задаваме изхода за отиване надолу да е "изключен" (Оказваме на асансьора че не трябва да ходи надолу).
                end if;
            when S1=>                -- Когато сме в второ състояние.
                if (SensorUp = '1') then -- Ако Сме на първи етаж.
                    NextState <= S2; -- Задаваме следаващото състояниеда е равно на трето състояние.
                    MoveUp <=  '0';  -- Задаваме изхода за отиване нагоре да е "изключен" (Оказваме на асансьора че не трябва да ходи нагоре).
                    MoveDn <=  '0';  -- Задаваме изхода за отиване надолу да е "изключен" (Оказваме на асансьора че не трябва да ходи надолу).
                end if;
            when S2=>                -- Когато сме в трето състояние.
                if (Dn = '1' and SensorUp = '1') then   -- Ако имаме сигнал за отиване надолу и сме на първи етаж.
                    NextState <= S3; -- Задаваме следаващото състояниеда е равно на четвърто състояние.
                    MoveUp <=  '0';  -- Задаваме изхода за отиване нагоре да е "изключен" (Оказваме на асансьора че не трябва да ходи нагоре).
                    MoveDn <=  '1';  -- Задаваме изхода за отиване надолу да е "включен" (Оказваме на асансьора че трябва да ходи надолу).
                end if;
            when S3 =>               -- Когато сме в четвърто състояние.
                if (SensorDn = '1') then  -- Ако Сме на нулев етаж.
                    NextState <= S0; -- Задаваме следаващото състояниеда е равно на първо състояние.
                    MoveUp <=  '0';  -- Задаваме изхода за отиване нагоре да е "изключен" (Оказваме на асансьора че не трябва да ходи нагоре).
                    MoveDn <=  '0';  -- Задаваме изхода за отиване надолу да е "изключен" (Оказваме на асансьора че не трябва да ходи надолу).
                end if;
            when others=>            -- Ако се появи не предвидено състояние
                NextState <= S0;     -- Задаваме следаващото състояниеда е равно на първо състояние.
        end case;

    end process NEXT_STATE_LOGIC;
    -- Извеждане на състоянието
    with CurrentState select
        State <= "0000" when S0,
                 "0001" when S1,
                 "0010" when S2,
                 "0011" when S3,
                 "0100" when S4,
                 "1111" when others;
end Behavioral;
```

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

entity Decoder7Seg is
    Port( Code : in std_logic_vector (3 downto 0);
          Segments : out std_logic_vector (7 downto 0);
          AnodeEnable : out std_logic_vector (3 downto 0));
end Decoder7Seg;

architecture Behavioral of Decoder7Seg is
begin
    process(Code)
    begin
        case Code is
            when "0000" => Segments <= "11000000"; -- "0"
            when "0001" => Segments <= "11111001"; -- "1"
            when "0010" => Segments <= "10100100"; -- "2"
            when "0011" => Segments <= "10110000"; -- "3"
            when "0100" => Segments <= "10011001"; -- "4"
            when "0101" => Segments <= "10010010"; -- "5"
            when "0110" => Segments <= "10000010"; -- "6"
            when "0111" => Segments <= "11111000"; -- "7"
            when "1000" => Segments <= "10000000"; -- "8"
            when "1001" => Segments <= "10010000"; -- "9"
            when "1110" => Segments <= "10100011"; -- "o"
            when "1111" => Segments <= "11110111"; -- "_"
            when others => Segments <= "11111111"; -- " "
        end case;
    end process;

    AnodeEnable <= "1110";
end Behavioral;
```

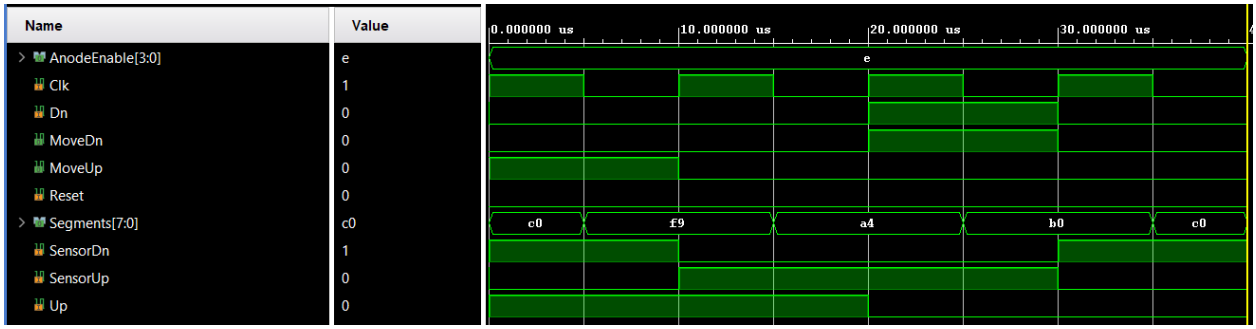
3. Constraints

```
set_property PACKAGE_PIN W19 [get_ports Clk]
set_property IOSTANDARD LVCMOS33 [get_ports Clk]

set_property PACKAGE_PIN U16 [get_ports MoveUp]
set_property IOSTANDARD LVCMOS33 [get_ports MoveUp]

set_property PACKAGE_PIN E19 [get_ports MoveDn]
set_property IOSTANDARD LVCMOS33 [get_ports MoveDn]

set_property PACKAGE_PIN U18 [get_ports Reset]
set_property IOSTANDARD LVCMOS33 [get_ports Reset]

set_property PACKAGE_PIN T18 [get_ports Up]
set_property IOSTANDARD LVCMOS33 [get_ports Up]

set_property PACKAGE_PIN U17 [get_ports Dn]
set_property IOSTANDARD LVCMOS33 [get_ports Dn]

set_property PACKAGE_PIN V17 [get_ports SensorUp]
set_property IOSTANDARD LVCMOS33 [get_ports SensorUp]

set_property PACKAGE_PIN V16 [get_ports SensorDn]
set_property IOSTANDARD LVCMOS33 [get_ports SensorDn]

set_property PACKAGE_PIN W4 [get_ports {AnodeEnable[3]}]
set_property PACKAGE_PIN V4 [get_ports {AnodeEnable[2]}]
set_property PACKAGE_PIN U4 [get_ports {AnodeEnable[1]}]
set_property PACKAGE_PIN U2 [get_ports {AnodeEnable[0]}]
set_property PACKAGE_PIN V7 [get_ports {Segments[7]}]
set_property PACKAGE_PIN U7 [get_ports {Segments[6]}]
set_property PACKAGE_PIN V5 [get_ports {Segments[5]}]
set_property PACKAGE_PIN U5 [get_ports {Segments[4]}]
set_property PACKAGE_PIN V8 [get_ports {Segments[3]}]
set_property PACKAGE_PIN U8 [get_ports {Segments[2]}]
set_property PACKAGE_PIN W6 [get_ports {Segments[1]}]
set_property PACKAGE_PIN W7 [get_ports {Segments[0]}]

set_property IOSTANDARD LVCMOS33 [get_ports {AnodeEnable[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AnodeEnable[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AnodeEnable[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AnodeEnable[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Segments[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Segments[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Segments[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Segments[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Segments[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Segments[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Segments[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Segments[0]}]
```

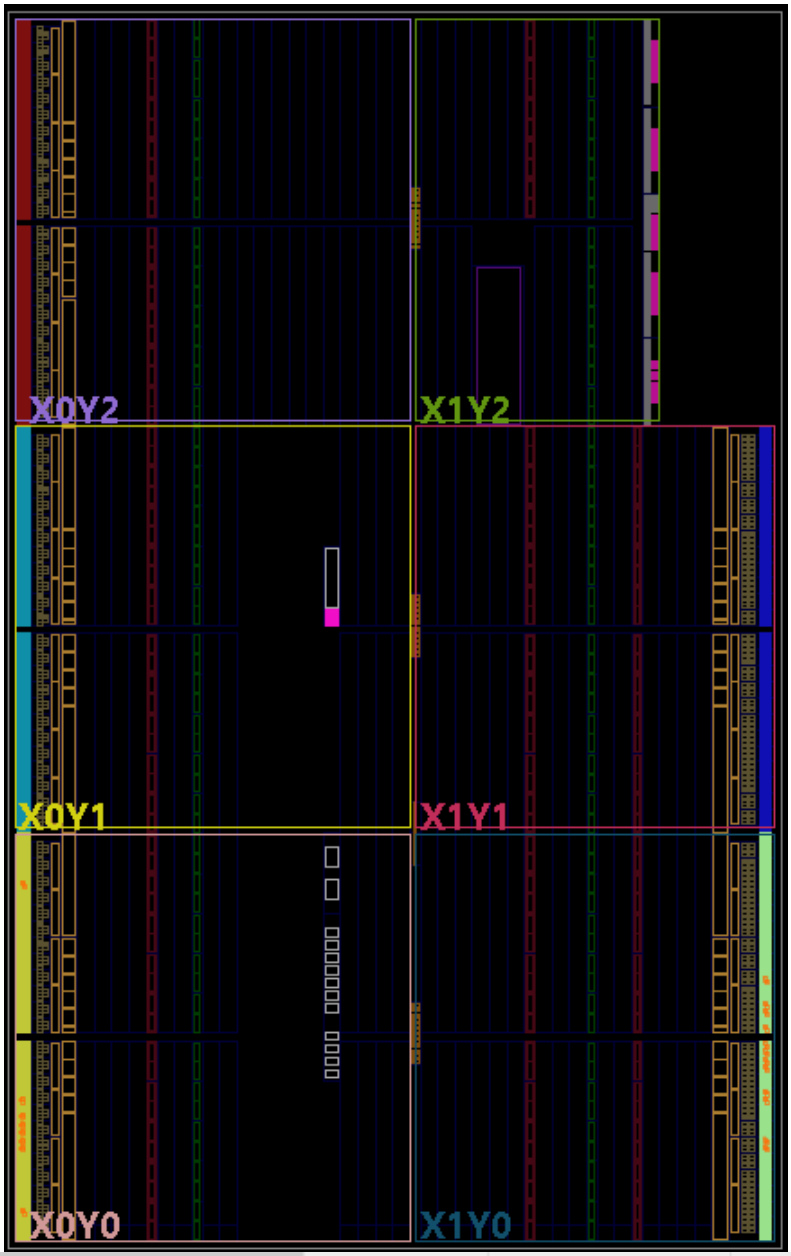
4. Wrapper


Clk
Reset
Up
Dn
SensorUp
SensorDn
FSM_0
Clk
Reset
Up
Dn
SensorUp
SensorDn
RTL
State[3:0]
MoveUp
MoveDn
FSM_v1_0
Decoder7Seg_0
Code[3:0]
RTL
Segments[7:0]
AnodeEnable[3:0]
Decoder7Seg_v1_0
Segments[7:0]
AnodeEnable[3:0]
MoveUp
MoveDn

5. VHDL тестов набор и времедиаграма



6. Отчет от реализацията



Name	Slice LUTs (20800)	Slice Registers (41600)	Bonded IOB (106)
N Wrapper_wrapper	0.03%	<0.01%	18.87%
Wrapper_i (Wrapper)	0.03%	<0.01%	0.00%