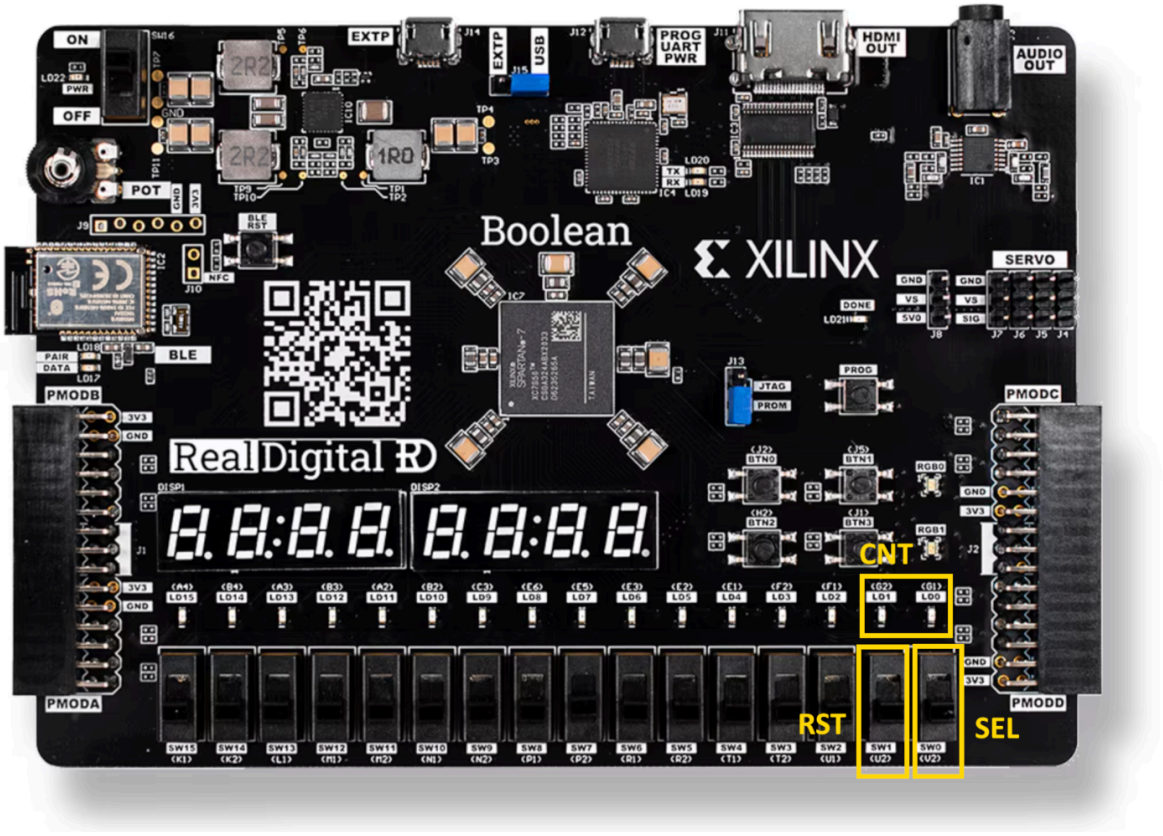
Лабораториска вежба 5
Имплементација на конечни автомати во FPGA чип

Напомена: Следната лабораториска вежба подразбира имплементација на VHDL дизајн на физички хардвер, конкретно FPGA компонента од типот AMD ‘Spartan 7’. За таа цел оваа лабораториска вежба ќе се работи во софтверот Xilinx Vivado. Упатствата за инсталација и употреба на истот се наоѓаат на е-курсеви во папката за лабораториски вежби.

1. Потребно е да се реализира два-битен синхрон бројач кој според избор на корисникот може да брои надолу или нагоре. Да се реализира хардверското решение на развојната плочка Boolean Board со влезни и излезни порти поставени според Табела 1 и визуелизирани на Слика 1. Да се испроба функционалноста на реализираниот бројач во двата режими и да се согледаат излезите од истиот. Одбројувањето да се одвива еднаш на секоја четвртина секунда.

Опис	FSM Пин	Boolean Board Пин	Тип на уред
Бит од бројач (излез)	CNT_1	G2	Зелена Лед Диода
Бит од бројач (излез)	CNT_0	G1	Зелена Лед Диода
Селекција (влез)	SEL	V2	Лизгачки прекинувач
Такт сигнал (влез)	CLK_IN	F14	Кристален осцилатор
Пин за рестарт (влез)	RST	U2	Лизгачки прекинувач

Табела 1. Поврзување на порти од бројач со пинови од Boolean Board



Слика 1. Порти од бараната компонента обележани на Boolean Board

Забелешка: Развојната плочка Boolean Board содржи интегриран такт генератор од 100MHz, па ако сакаме конечниот автомат да ја помнени својата состојба еднаш на секоја четвртина секунда, потребно е при влез во нова состојба да остане во истата 25000000 (дваесет и пет милиони или една четвртина од сто мега) такт циклуси. За таа цел ќе употребиме три помошни сигнали и тоа: еден бројач од нула до бројот на idle такт сигнали, едно знаменце кое ќе се сетира секој пат кога ќе се стигне границата на бројачот, и еден помошен такт сигнал кој ќе го употребиме за премин од една во друга состојба наместо интерниот такт генератор на системот.

Пример решение на задачата:

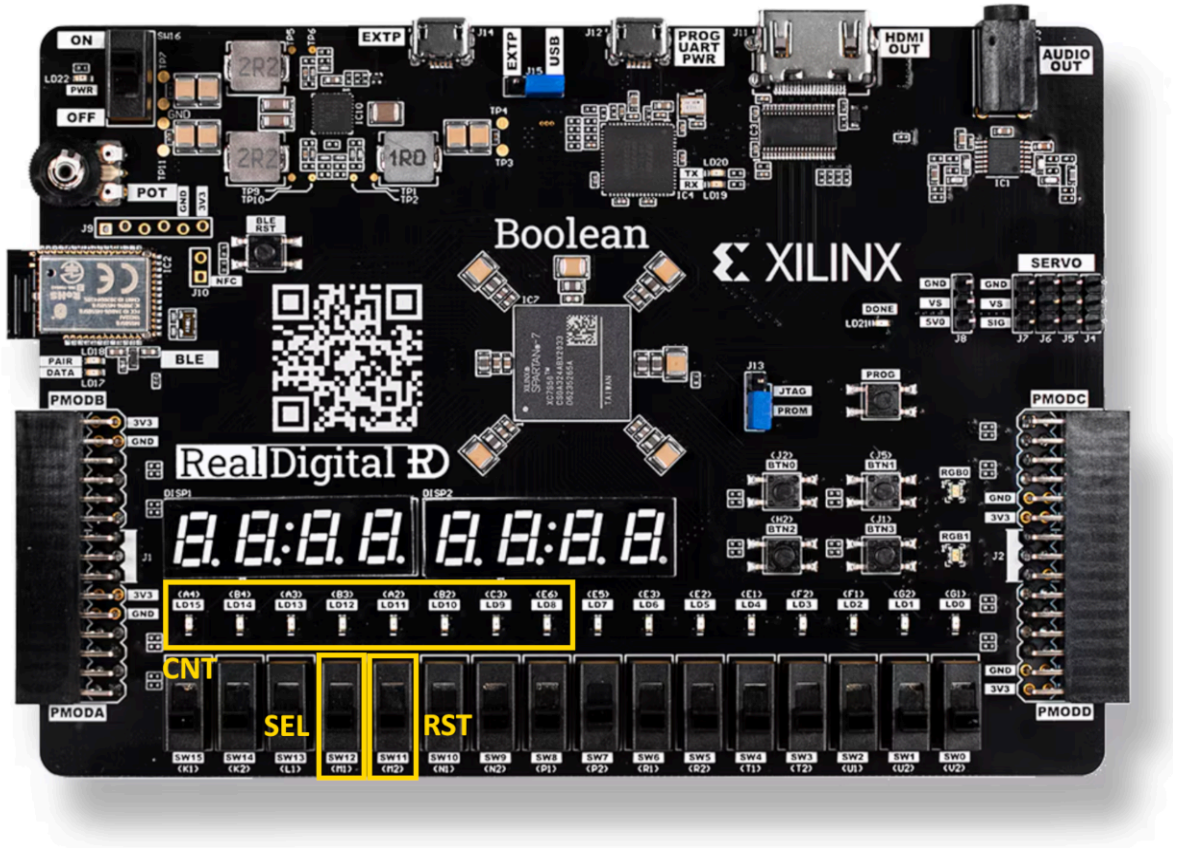
```
--zadacha 2-biten brojach (up/down) od auditoriski
--modificirana so dodaden delitel na frekvencija

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY COUNTER_2BIT_UPDOWN IS
PORT(
    CLK_IN,RST:IN STD_LOGIC; --promenet vlezen takt signal
    UP: IN STD_LOGIC;
    CNT: OUT STD_LOGIC_VECTOR(1 DOWNTO 0));
END ENTITY COUNTER_2BIT_UPDOWN;

ARCHITECTURE BEHAVIORAL OF COUNTER_2BIT_UPDOWN IS
TYPE STATE_TYPE IS (C0,C1,C2,C3);
SIGNAL CURRENT_STATE, NEXT_STATE: STATE_TYPE;

--vazhen dodatok pomoshni signali za delitel na frekvencija
signal temporal: STD_LOGIC;
signal counter : integer range 0 to 24999999 := 0;
signal CLK : STD_LOGIC;
--vazhen dodatok pomoshni signali za delitel na frekvencija

BEGIN

    --vazhen dodatok delitel na frekvencija
    frequency_divider: process (CLK_IN) begin
        if rising_edge(CLK_IN) then
            if (counter = 24999999) then
                temporal <= NOT(temporal);
                counter <= 0;
            else
                counter <= counter + 1;
            end if;
        end if;
    end process frequency_divider;
    CLK <= temporal;
    --vazhen dodatok delitel na frekvencija

     STATE_MEMORY: PROCESS(CLK,RST)
```

```
        BEGIN
            IF(RST='0')THEN
                CURRENT_STATE<=C0;
            ELSIF(RISING_EDGE(CLK))THEN
                CURRENT_STATE<=NEXT_STATE;
            END IF;
    END PROCESS STATE_MEMORY;

    NEXT_STATE_LOGIC:PROCESS(CURRENT_STATE,UP)
    BEGIN
        CASE(CURRENT_STATE) IS
            WHEN C0=>
                IF(UP='1') THEN
                    NEXT_STATE<=C1;
                ELSE
                    NEXT_STATE<=C3;
                END IF;
            WHEN C1=>
                IF(UP='1') THEN
                    NEXT_STATE<=C2;
                ELSE
                    NEXT_STATE<=C0;
                END IF;
            WHEN C2=>
                IF (UP='1') THEN
                    NEXT_STATE<=C3;
                ELSE
                    NEXT_STATE<=C1;
                END IF;
            WHEN C3=>
                IF(UP='1') THEN
                    NEXT_STATE<=C0;
                ELSE
                    NEXT_STATE<=C2;
                END IF;
            WHEN OTHERS=>NEXT_STATE<=C0;
        END CASE;
    END PROCESS NEXT_STATE_LOGIC;

    OUTPUT_LOGIC:PROCESS(CURRENT_STATE)
    BEGIN
        CASE(CURRENT_STATE)IS
            WHEN C0 => CNT<="00";
            WHEN C1=>CNT<="01";
            WHEN C2=>CNT<="10";
            WHEN C3=>CNT<="11";
            WHEN OTHERS=>CNT<="00";
        END CASE;
    END PROCESS OUTPUT_LOGIC;

END ARCHITECTURE;
```

2. Потребно е да се реализира VHDL дизајн на осум-битен синхрон бројач (конечен автомат) кој според избор на корисникот може да ги одбројува нагоре и надолу првите девет позитивни содржатели на бројот 28. Во овој опсег бараните броеви се 28, 56, 84, 112, 140, 168, 196, 224 и 252. Бројачот да се изработи како **конечен автомат** со претходно дефинирани **состојби**, а **не** како алгоритам кој ги пронаоѓа првите девет позитивни содржатели на бројот 28 во опсегот од 0 до 255. Да се реализира хардверското решение на развојната плочка Boolean Board со влезни и излезни порти поставени според Табела 2 и визуелизирани на Слика 2. Да се испроба функционалноста на реализираниот бројач во двата режими и да се согледаат излезите од истиот. Одбројувањето да се одвива еднаш на секоја половина секунда.

Опис	FSM Пин	Boolean Board Пин	Тип на уред
Бит од бројач (излез)	CNT_7	A4	Зелена Лед Диода
Бит од бројач (излез)	CNT_6	B4	Зелена Лед Диода
Бит од бројач (излез)	CNT_5	A3	Зелена Лед Диода
Бит од бројач (излез)	CNT_4	B3	Зелена Лед Диода
Бит од бројач (излез)	CNT_3	A2	Зелена Лед Диода
Бит од бројач (излез)	CNT_2	B2	Зелена Лед Диода
Бит од бројач (излез)	CNT_1	C3	Зелена Лед Диода
Бит од бројач (излез)	CNT_0	E6	Зелена Лед Диода
Селекција (влез)	SEL	M1	Лизгачки прекинувач
Такт сигнал (влез)	CLK	F14	Кристален осцилатор
Пин за рестарт (влез)	RST	M2	Лизгачки прекинувач

Табела 2. Поврзување на порти од бројач со пинови од Boolean Board



Слика 2. Порти од бараната компонента обележани на Boolean Board