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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if
instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity TAP_Controller is
    port(
           RELOJ : in STD_LOGIC; -- reloj a 100MHz
           INSTRUCCION : in STD_LOGIC_VECTOR (2 downto 0);
           DONE : out STD_LOGIC;
           TDI_PMOD : out STD_LOGIC;
           TCK_PMOD : out STD_LOGIC;
           TMS_PMOD : out STD_LOGIC;
           TDO_PMOD : in STD_LOGIC;
           IR_VALUES : out STD_LOGIC_VECTOR (14 downto 0);
           IR_INSTRUCTION : in STD_LOGIC_VECTOR (14 downto 0);
           DR_VALUES : out STD_LOGIC_VECTOR (150 downto 0);
        DR_INSTRUCTION : in STD_LOGIC_VECTOR (150 downto 0)
     );
end TAP_Controller;

architecture Behavioral of TAP_Controller is
-- TAP MOORE STATE MACHINE
    -- MAQUINA DE ESTADOS
        Component TAP_FSM
            port(
                TMS_I : in STD_LOGIC; -- TMS desde FPGA
                RELOJ_TAP : in STD_LOGIC; -- reloj 25 MHz
                TDO_I : out STD_LOGIC; -- TDO hacia la FPGA
                TDI_I_DR : in STD_LOGIC; -- TDI DR desde la
FPGA
                TDI_I_IR : in STD_LOGIC; -- TDI IR desde la
FPGA
                TMS_O : out STD_LOGIC;
                TCK_O : out STD_LOGIC;
                TDO_O : in STD_LOGIC;
                TDI_O : out STD_LOGIC
            );
        End Component;
    -- Registro de instruccion a realizar
        Component Register_3bits
            port(
                DATA: in STD_LOGIC_VECTOR (2 downto 0);
                RELOJ: in STD_LOGIC;
                Q: out STD_LOGIC_VECTOR (2 downto 0)
            );
        End Component;
```

```
-- Registros de datos pata el test
    -- Registro de instrucciones
        Component Shift_Register_15bits
            Port (
                RELOJ : in STD_LOGIC;
                EN : in STD_LOGIC;
                CARGAR : in STD_LOGIC;
                PARALLEL_IN : in STD_LOGIC_VECTOR (14 downto
0);
                DATA_IN : in STD_LOGIC;
                PARALLEL_OUT : out STD_LOGIC_VECTOR (14 downto
0);
                DATA_OUT : out STD_LOGIC
            );
        End Component;
    -- Registro de datos
        Component Shift_Register_151bits
            Port (
                RELOJ : in STD_LOGIC;
                EN : in STD_LOGIC;
                CARGAR : in STD_LOGIC;
                PARALLEL_IN : in STD_LOGIC_VECTOR (150 downto
0);
                DATA_IN : in STD_LOGIC;
                PARALLEL_OUT : out STD_LOGIC_VECTOR (150
downto 0);
                DATA_OUT : out STD_LOGIC
            );
        End Component;

-- Contador para envío de TMS
        Component Contador_9bits
            port(
                EN: in STD_LOGIC;
                RELOJ: in STD_LOGIC;
                RESET: in STD_LOGIC;
                OUTPUT: out STD_LOGIC_VECTOR (8 downto 0)
            );
        End Component;

-- SIGNALS
    signal INSTRUCCION_ACTUAL: STD_LOGIC_VECTOR (2 downto 0);
    -- Reloj adaptado a 25 MHz
        --signal PRESCALER: STD_LOGIC_VECTOR(25 downto 0) :=
"10111110101111000010000000";
        --signal PRESCALER_COUNTER: STD_LOGIC_VECTOR(25 downto
0) := (others => '0');
        signal PRESCALER: STD_LOGIC_VECTOR(21 downto 0) :=
"1111111111111111111111";
        signal PRESCALER_COUNTER: STD_LOGIC_VECTOR(21 downto
0) := (others => '0');
        --signal PRESCALER: STD_LOGIC_VECTOR(3 downto 0) :=
"1111";
        --signal PRESCALER_COUNTER: STD_LOGIC_VECTOR(3 downto
0) := "0000";
        signal TAP_clk : STD_LOGIC :='0';
    -- Propias del TAP
```

```
        signal TDI : STD_LOGIC;
        signal TDI_IR : STD_LOGIC;
        signal TDI_DR : STD_LOGIC;
        signal TDO : STD_LOGIC;
        signal TMS : STD_LOGIC := '0';
        -- Estados IR
            signal IR_ENABLE : STD_LOGIC := '0';
            signal CARGAR_IR : STD_LOGIC := '0';
            signal CARGAR_VALORES_IR : STD_LOGIC_VECTOR (14
downto 0) := (others => '0');
        -- Estados DR
            signal DR_ENABLE : STD_LOGIC := '0';
            signal CARGAR_DR : STD_LOGIC := '0';
            signal CARGAR_VALORES_DR : STD_LOGIC_VECTOR (150
downto 0) := (others => '0');
        -- TMS_COUNTER
            signal COUNTER_EN : STD_LOGIC := '0';
            signal COUNTER_RESET : STD_LOGIC := '0';
            signal COUNTER_OUT : STD_LOGIC_VECTOR (8 downto
0) := (others => '0');
begin

-- Generamos el reloj que usará el TAP (25 MHz, admitidos por
los 2 integrados)
    process (RELOJ)
    begin
        if rising_edge(RELOJ) then
                PRESCALER_COUNTER <= PRESCALER_COUNTER + 1;
                if(PRESCALER_COUNTER = PRESCALER) then
                    TAP_clk <= NOT(TAP_clk);
                    PRESCALER_COUNTER <= (others => '0');
                end if;
            end if;
    end process;

-- Seleccionamos el conjunto de instrucciones a realizar
    process (TAP_clk)
    begin
        if falling_edge(TAP_clk) then
            case INSTRUCCION_ACTUAL is
                -- RESET
                when "001" =>
                    -- Si es el primer ciclo iniciamos todos
                    if COUNTER_OUT = "000000000" then
                        COUNTER_EN <= '1';
                        COUNTER_RESET <= '0';
                        TMS <= '1';
                        DONE <= '0';
                    -- Reset a la máquina de estados
                    elsif COUNTER_OUT = "000000001" then
                        TMS <= '1';
                    -- Llevamos la máquina de estados a IDLE
                    elsif COUNTER_OUT = "000000110" then
                        TMS <= '0';
                        DONE <= '1';
                    elsif COUNTER_OUT = "000000111" then
                        COUNTER_EN <= '0';
```

```
                        COUNTER_RESET <= '1';
                        DONE <= '0';
                        TMS <= '0';
                    end if;

                -- SEND IR
                when "010" =>
                    -- Si es el primer ciclo iniciamos todo
                    if  COUNTER_OUT = "000000000" then
                        CARGAR_VALORES_IR <= IR_INSTRUCTION;
                        CARGAR_IR <= '1'; -- Habilitamos la
carga en paralelo
                        DONE <= '0';
                        COUNTER_EN <= '1';
                        COUNTER_RESET <= '0';
                        TMS <= '0';
                    -- Vamos a Select IR Scan
                    elsif COUNTER_OUT = "000000001" then
                        CARGAR_IR <= '0';
                        TMS <= '1';
                    -- Vamos a Shift IR Scan
                    elsif COUNTER_OUT = "000000011" then
                        TMS <= '0';
                    -- Primer ciclo de Shift IR Scan
                    elsif COUNTER_OUT = "000000101" then
                        TMS <= '0';
                        IR_ENABLE <= '1';
                    -- Último ciclo de Shift IR Scan, pasamos
a Exit1 IR
                    elsif COUNTER_OUT = "000010011" then
                        TMS <= '1';
                    -- Exit1 IR, deshabilitamos el registro de
intrucciones
                    elsif COUNTER_OUT = "000010100" then
                        IR_ENABLE <= '0';
                        TMS <= '1';
                    -- Llevamos la máquina de estados a IDLE
                    elsif COUNTER_OUT = "000010101" then
                        TMS <= '0';
                        DONE <= '1';
                    elsif COUNTER_OUT = "000010110" then
                        COUNTER_EN <= '0';
                        COUNTER_RESET <= '1';
                        DONE <= '0';
                        TMS <= '0';
                    end if;
                -- SEND DR
                when "100" =>
                    -- Si es el primer ciclo iniciamos todo
                    if  COUNTER_OUT = "000000000" then
                        CARGAR_VALORES_DR <= DR_INSTRUCTION;
                        CARGAR_DR <= '1'; -- Habilitamos la
carga en paralelo
                        DONE <= '0';
                        COUNTER_EN <= '1';
                        COUNTER_RESET <= '0';
                        TMS <= '0';
```

```
                    -- Vamos a Select DR Scan
                    elsif COUNTER_OUT = "000000001" then
                        CARGAR_DR <= '0';
                        TMS <= '1';
                    -- Vamos a Shift DR Scan
                    elsif COUNTER_OUT = "000000010" then
                        TMS <= '0';
                    -- Primer ciclo de Shift DR Scan
                    elsif COUNTER_OUT = "000000100" then
                        TMS <= '0';
                        DR_ENABLE <= '1';
                    -- Último ciclo de Shift DR Scan, pasamos
a Exit1 DR
                    elsif COUNTER_OUT = "010011010" then
                        TMS <= '1';
                    -- Exit1 DR, deshabilitamos el registro de
datos
                    elsif COUNTER_OUT = "010011011" then
                        DR_ENABLE <= '0';
                        TMS <= '1';
                    -- Llevamos la máquina de estados a IDLE
                    elsif COUNTER_OUT = "010011100" then
                        TMS <= '0';
                        DONE <= '1';
                    elsif COUNTER_OUT = "010011101" then
                        COUNTER_EN <= '0';
                        COUNTER_RESET <= '1';
                        DONE <= '0';
                        TMS <= '0';
                    end if;

                -- TEST DR
                when "101" =>
                    -- Si es el primer ciclo iniciamos todo
                    if  COUNTER_OUT = "000000000" then
                        CARGAR_VALORES_DR <= DR_INSTRUCTION;
                        CARGAR_DR <= '1'; -- Habilitamos la
carga en paralelo
                        DONE <= '0';
                        COUNTER_EN <= '1';
                        COUNTER_RESET <= '0';
                        TMS <= '0';
                    -- Vamos a Select DR Scan
                    elsif COUNTER_OUT = "000000001" then
                        CARGAR_DR <= '0';
                        TMS <= '1';
                    -- Vamos a Shift DR Scan
                    elsif COUNTER_OUT = "000000010" then
                        TMS <= '0';
                    -- Primer ciclo de Shift DR Scan
                    elsif COUNTER_OUT = "000000100" then
                        TMS <= '0';
                        DR_ENABLE <= '1';
                    -- Último ciclo de Shift DR Scan, pasamos
a Exit1 DR
                    elsif COUNTER_OUT = "100110001" then
                        TMS <= '1';
```

```
                    -- Exit1 DR, deshabilitamos el registro de
datos
                    elsif COUNTER_OUT = "100110010" then
                        DR_ENABLE <= '0';
                        TMS <= '1';
                    -- Llevamos la máquina de estados a IDLE
                    elsif COUNTER_OUT = "100110011" then
                        TMS <= '0';
                        DONE <= '1';
                    elsif COUNTER_OUT = "100110100" then
                        COUNTER_EN <= '0';
                        COUNTER_RESET <= '1';
                        DONE <= '0';
                        TMS <= '0';
                    end if;

                -- IDLE
                when others =>
                    TMS <= '0';
                    COUNTER_EN <= '0';
                    COUNTER_RESET <= '1';
                    DONE <= '0';
            end case;
        end if;
    end process;

-- Bloques VHDL incorporados al fichero
    -- TAP Moore FSM
        TAP_State_Machine: TAP_FSM PORT MAP (
            TMS_I => TMS, -- TMS desde FPGA
            RELOJ_TAP => TAP_clk, -- reloj 25 MHz
            TDO_I => TDO, -- TDO hacia la FPGA
            TDI_I_DR => TDI_DR, -- TDI desde la FPGA
            TDI_I_IR => TDI_IR, -- TDI_IR desde la FPGA
            TMS_O => TMS_PMOD, -- PIN P-MOD
            TCK_O => TCK_PMOD, -- PIN P-MOD
            TDO_O => TDO_PMOD, -- PIN P-MOD
            TDI_O => TDI_PMOD-- PIN P-MOD
        );

    -- Registro de instruccion a realizar
        Next_Instruction:  Register_3bits PORT MAP(
            DATA => INSTRUCCION,
            RELOJ => TAP_clk,
            Q => INSTRUCCION_ACTUAL
        );

    -- Registro de instrucciones
        IR_chain: Shift_Register_15bits PORT MAP (
            RELOJ => TAP_clk,
            EN => IR_ENABLE,
            CARGAR => CARGAR_IR,
            PARALLEL_IN => CARGAR_VALORES_IR,
            DATA_IN => TDO,
            PARALLEL_OUT => IR_VALUES,
            DATA_OUT => TDI_IR
        );
```

```
    -- Registro de datos
        DR_chain: Shift_Register_151bits PORT MAP (
            RELOJ => TAP_clk,
            EN => DR_ENABLE,
            CARGAR => CARGAR_DR,
            PARALLEL_IN => CARGAR_VALORES_DR,
            DATA_IN => TDO,
            PARALLEL_OUT => DR_VALUES,
            DATA_OUT => TDI_DR
        );

    -- Contador para envío de TMS
        TMS_counter: Contador_9bits PORT MAP (
            EN => COUNTER_EN,
            RELOJ => TAP_clk,
            RESET => COUNTER_RESET,
            OUTPUT => COUNTER_OUT
        );
end Behavioral;
```