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

entity TOP_LEVEL is
port(
           an    : out STD_LOGIC_VECTOR(3 downto 0); -- 4
anodos para seleccionar posicion DISPLAY
           seg : out STD_LOGIC_VECTOR(6 downto 0); -- 7
segmentos del DISPLAY
           clk : in STD_LOGIC; -- relojes
           led : out STD_LOGIC_VECTOR(15 downto 0);
           sw : in STD_LOGIC_VECTOR(0 downto 0);
           btnL, btnR : in STD_LOGIC;
           TDI_PLACA : out STD_LOGIC;
        TCK_PLACA : out STD_LOGIC;
        TMS_PLACA : out STD_LOGIC;
        TDO_PLACA : in STD_LOGIC
     );
end TOP_LEVEL;

architecture Behavioral of TOP_LEVEL is

-- Controlador TAP
    Component TAP_Controller
        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 Component;

-- DISPLAY 7 SEGMENTOS
    -- Controlador DISPLAY
        Component Display_7seg
            port(
                BCD : in STD_LOGIC_VECTOR(15 downto 0); --
valor a mostrar
                RELOJ : in STD_LOGIC; -- reloj
                SEGMENTOS : out STD_LOGIC_VECTOR(6 downto
0); -- 7seg
                ANODOS : out STD_LOGIC_VECTOR(3 downto 0); --
anodos 7seg
                ENABLE : in STD_LOGIC
            );
        End Component;
    -- REGISTRO VALORES DISPLAY
```

```
        Component Register_16bits
            port(
                DATA: in STD_LOGIC_VECTOR(15 downto 0);
                EN: in STD_LOGIC_VECTOR(3 downto 0);
                RELOJ: in STD_LOGIC;
                Q: out STD_LOGIC_VECTOR(15 downto 0)
            );
        End Component;

-- Contador máximo celdas entrada paralelas
    Component Contador_4bits
        port(
            EN: in std_logic;
            RELOJ: in std_logic;
            RESET: in std_logic;
            OUTPUT: out std_logic_vector(0 to 3)
        );
    End Component;

-- Contador input cells
    Component Contador_8bits is
        port(
            EN: in STD_LOGIC;
            RELOJ: in STD_LOGIC;
            RESET: in STD_LOGIC;
            OUTPUT: out STD_LOGIC_VECTOR (7 downto 0)
        );
    end Component;

-- Red y BSC de input
    Component I_CELLS_NETS is
        port (
           elemento : in std_logic_vector(4 downto 0);
           red : out std_logic_vector(7 downto 0);
           BSC : out std_logic_vector(7 downto 0)
        );
    End Component;

-- Red y BSC de output
    Component O_CELLS_NETS is
        port (
           elemento : in std_logic_vector(4 downto 0);
           red : out std_logic_vector(7 downto 0);
           BSC : out std_logic_vector(7 downto 0)
        );
    End Component;

-- BSC, CTRL, CTRLVALUE
        Component C_CELLS_NETS is
            port (
               elemento : in std_logic_vector(4 downto 0);
               BSC : out std_logic_vector(7 downto 0);
               CTRL : out std_logic_vector(7 downto 0);
               VALUE : out std_logic
            );
        End Component;
```

```
-- Vector W1W0
    Component Shift_Register_13bits
        port(
            RELOJ : in STD_LOGIC;
            EN : in STD_LOGIC;
            CARGAR : in STD_LOGIC;
            PARALLEL_IN : in STD_LOGIC_VECTOR (12 downto 0);
            PARALLEL_OUT : out STD_LOGIC_VECTOR (12 downto 0)
        );
    End Component;

-- Registros errores
    -- Registro JTAG, BS
        Component Register_1bit
            port(
                DATA: in STD_LOGIC;
                EN: in STD_LOGIC;
                RELOJ: in STD_LOGIC;
                Q: out STD_LOGIC
            );
        End Component;
    -- Registro STUCK-AT
        Component Register_19bits
            Port (
                RELOJ : in STD_LOGIC;
                EN : in STD_LOGIC;
                PARALLEL_IN : in STD_LOGIC_VECTOR (18 downto
0);
                PARALLEL_OUT : out STD_LOGIC_VECTOR (18 downto
0)
            );
        End Component;

-- SIGNALS
    -- Valores para el DISPLAY 7 segmentos
        -- Valores con los que se quiere actualizar el
registro
        signal REGISTRO_DISPLAY: STD_LOGIC_VECTOR(15 downto
0) := "0000000000000000";
        -- Habilita que el registro de valores del display se
actualice
        signal ACTUALIZAR_DISPLAY: STD_LOGIC_VECTOR(3 downto
0) := "0000";
        -- Pasa valores del registro al display
        signal VALOR_DISPLAY: STD_LOGIC_VECTOR(15 downto 0) :=
"0000000000000000";
        -- Habilita que el display se muestre
        signal MOSTRAR_DISPLAY: STD_LOGIC := '0';

    -- Valores para la máquina de estados que controla la
ejecución
        -- Estados del programa principal
        type state_type is (ST_Inicio, ST_T1, ST_JTAG, ST_T2,
ST_BS, ST_T3, ST_Test, ST_Resultados);
        -- Registro para los estados del programa principal
        signal state : state_type := ST_Inicio;
        -- Operación a realizar por el TAP
```

```
        signal TAP_INSTRUCTION : STD_LOGIC_VECTOR (2 downto
0) := "001";
        -- Ciclo del TAP realizado
        signal TAP_DONE : STD_LOGIC := '0';
        -- Control del TAP realizado
        signal CTRL_TAP_DONE : STD_LOGIC := '0';
        -- Valor del IR
        signal TAP_IR : STD_LOGIC_VECTOR (14 downto 0) :=
(others => '0');
        signal INSTRUCCION_IR : STD_LOGIC_VECTOR (14 downto
0);
        -- Valor del DR
        signal TAP_DR : STD_LOGIC_VECTOR (150 downto 0) :=
(others => '0');
        signal INSTRUCCION_DR : STD_LOGIC_VECTOR (150 downto
0);

        -- Control del test JTAG
        signal JTAG_EN : STD_LOGIC :='0';
        signal JTAG_DONE : STD_LOGIC :='0';
        signal JTAG_TEST : STD_LOGIC := '1'; -- Resultado del
Test JTAG (llegada al registro)
        signal JTAG_RESULTS : STD_LOGIC :='1'; -- Resultado
del Test JTAG (salida del registro)

        -- Control del test BS
        signal BS_EN : STD_LOGIC :='0';
        signal BS_DONE : STD_LOGIC :='0';
        signal BS_TEST : STD_LOGIC := '1'; -- Resultado del
Test BS (llegada al registro)
        signal BS_RESULTS : STD_LOGIC :='1'; -- Resultado del
Test BS (salida del registro)

        -- Control del test eléctrico
        signal ET_EN : STD_LOGIC :='0';
        signal ET_DONE : STD_LOGIC :='0';

        -- Control para mostrar resultados
        signal RSLTS_EN : STD_LOGIC :='0';

    -- Señales para el test eléctrico
        -- Stuck-at-zero
        signal SA0_ENABLE : STD_LOGIC :='0';
        signal CARGAR_VALORES_SA0 : STD_LOGIC_VECTOR (18
downto 0) := (others => '0');
        signal SA0_VALUES : STD_LOGIC_VECTOR (18 downto 0);
        signal SA0_UPDATE : STD_LOGIC_VECTOR (18 downto 0);

        -- Stuck-at-one
        signal SA1_ENABLE : STD_LOGIC :='0';
        signal CARGAR_VALORES_SA1 : STD_LOGIC_VECTOR (18
downto 0) := (others => '0');
        signal SA1_VALUES : STD_LOGIC_VECTOR (18 downto 0);
        signal SA1_UPDATE : STD_LOGIC_VECTOR (18 downto 0);

        -- Contador redes entrada paralelas
        signal PAR_I_CELLS_EN : STD_LOGIC :='0';
```

```
        signal PAR_I_CELLS_RST : STD_LOGIC :='0';
        signal PAR_I_CELLS : STD_LOGIC_VECTOR (3 downto 0);

        -- MÁXIMO ENTRADAS PARALELAS
        signal MAX_PAR_IN : STD_LOGIC_VECTOR (3 downto 0) :=
"0011";

        -- Test W1 o W0
        signal W1W0_NEXT : STD_LOGIC := '0';
        signal W1W0_EN : STD_LOGIC := '0';
        signal W1W0_RESULTS : STD_LOGIC;

        -- Vector test W1 W0
        signal W1W0_NETS_EN : STD_LOGIC := '0';
        signal W1W0_NETS_LOAD : STD_LOGIC := '0';
        signal W1W0_NETS_INPUT : STD_LOGIC_VECTOR (12 downto
0);
        signal W1W0_NETS_OUTPUT : STD_LOGIC_VECTOR (12 downto
0) := "0000000000001";

        -- Contador de redes
        signal NETS_EN : STD_LOGIC :='0';
        signal NETS_RST : STD_LOGIC :='0';
        signal NETS : STD_LOGIC_VECTOR (3 downto 0);

        -- Contador IN_CELLS
        signal IN_CELLS_EN : STD_LOGIC :='0';
        signal IN_CELLS_RST : STD_LOGIC :='0';
        signal IN_CELLS : STD_LOGIC_VECTOR (7 downto 0);

        signal TOTAL_I_CELLS : STD_LOGIC_VECTOR (7 downto 0) :
= "00010011";

        -- Contador C_CELLS
        signal C_CELLS_EN : STD_LOGIC :='0';
        signal C_CELLS_RST : STD_LOGIC :='0';
        signal C_CELLS : STD_LOGIC_VECTOR (7 downto 0);

        -- Contador OUT_CELLS
        signal OUT_CELLS_EN : STD_LOGIC :='0';
        signal OUT_CELLS_RST : STD_LOGIC :='0';
        signal OUT_CELLS : STD_LOGIC_VECTOR (7 downto 0);

        signal TOTAL_O_CELLS : STD_LOGIC_VECTOR (7 downto 0) :
= "00010011";

        -- Registro con input cells y redes
        signal NETS_I_CELLS : STD_LOGIC_VECTOR (4 downto 0);
        signal NETS_I_CELLS_NET : STD_LOGIC_VECTOR (7 downto
0);
        signal NETS_I_CELLS_BSC : STD_LOGIC_VECTOR (7 downto
0);

        -- Registro con output cells y redes
        signal NETS_O_CELLS : STD_LOGIC_VECTOR (4 downto 0);
        signal NETS_O_CELLS_NET : STD_LOGIC_VECTOR (7 downto
0);
```

```
        signal NETS_O_CELLS_BSC : STD_LOGIC_VECTOR (7 downto
0);

        -- Registro con CTRL cells y redes
        signal CTRL_CELLS : STD_LOGIC_VECTOR (4 downto 0);
        signal CTRL_CELLS_BSC : STD_LOGIC_VECTOR (7 downto 0);
        signal CTRL_CELLS_CTRL : STD_LOGIC_VECTOR (7 downto
0);
        signal CTRL_CELLS_VALUE : STD_LOGIC;

        signal TOTAL_C_CELLS : STD_LOGIC_VECTOR (7 downto 0) :
= "00010011";

        -- Contador de redes generadas
        signal NETS_GEN_EN : STD_LOGIC :='0';
        signal NETS_GEN_RST : STD_LOGIC :='0';
        signal NETS_GEN : STD_LOGIC_VECTOR (3 downto 0);

        -- Contador de redes repetidas
        signal NETS_REP_EN : STD_LOGIC :='0';
        signal NETS_REP_RST : STD_LOGIC :='0';
        signal NETS_REP : STD_LOGIC_VECTOR (3 downto 0);

        -- TOTAL REDES
        signal MAX_NETS : STD_LOGIC_VECTOR (3 downto 0) :=
"1101";

        -- Generación vector test
        signal VECTOR_TEST : STD_LOGIC_VECTOR (150 downto 0);
        signal VECTOR_GEN : STD_LOGIC := '0';

        -- Estados del generador de vectores de test
        type estados is (T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_
8);
        -- Registro para los estados del programa principal
        signal estado_Test : estados := T_1;

        signal INICIAR_TEST : STD_LOGIC := '0';
        signal TEST_DONE : STD_LOGIC := '0';

begin

-- Máquina de estados principal
    -- Generación del próximo estado
    process (clk)
    begin
        if rising_edge(clk) then
            case state is
                -- Inicio
                when ST_Inicio=>
                    --Esperamos reinicio del TAP para comenzar
el test
                    if TAP_DONE = '1' then
                        state <= ST_T1;
                    else
                        state <= ST_Inicio;
                    end if;
```

```
                -- Transicion 1 - Espera finalización Inicio
                when ST_T1=>
                    if TAP_DONE = '0' then
                        state <= ST_JTAG;
                    else
                        state <= ST_T1;
                    end if;
                -- Verificación JTAG
                when ST_JTAG=>
                    -- Hasta que no se realice el test no se
cambia de estado
                    if TAP_DONE = '1' then
                        state <= ST_T2;
                    else
                        state <= ST_JTAG;
                    end if;
                -- Transicion 2 - Espera finalización test
JTAG
                when ST_T2=>
                    if TAP_DONE = '0' and JTAG_EN = '1' then
                        -- Si hay fallo
                        if JTAG_RESULTS = '1' then
                            state <= ST_Resultados;
                        -- Si no hay fallo
                        else
                            state <= ST_BS;
                        end if;
                    else
                        state <= ST_T2;
                    end if;
                -- Verificación Boundary-Scan
                when ST_BS=>
                    -- Hasta que no se realice el test no se
cambia de estado
                    if TAP_DONE = '1' then
                        state <= ST_T3;
                    else
                        state <= ST_BS;
                    end if;
                -- Transicion 3 - Espera finalización test BS
                when ST_T3=>
                    if TAP_DONE = '0' and BS_EN = '1' then
                        -- Si hay fallo
                        if BS_RESULTS = '1' then
                            state <= ST_Resultados;
                        -- Si no hay fallo
                        else
                            state <= ST_Test;
                        end if;
                    else
                        state <= ST_T3;
                    end if;
                -- Realización Test eléctrico
                when ST_Test =>
                    -- Hasta que no se realice el test no se
cambia de estado
                    if TEST_DONE = '1' then
```

```
                        state <= ST_Resultados;
                    else
                        state <= ST_Test;
                    end if;
                -- Mostrar fallos
                when ST_Resultados=>
                    state <= ST_Resultados;
            end case;
        end if;
    end process;

     -- Generación de salidas acorde al estado actual
     process (state, JTAG_RESULTS, BS_RESULTS, TAP_IR, TAP_DR,
PAR_I_CELLS, MAX_PAR_IN, NETS, MAX_NETS, W1W0_RESULTS,
VECTOR_GEN, VECTOR_TEST, sw, btnL, btnR, SA0_VALUES, SA1
_VALUES)
     begin

         MOSTRAR_DISPLAY <= '1';
        ACTUALIZAR_DISPLAY <= "1111";
        JTAG_EN <= '0';
        JTAG_TEST <= '1';
        BS_EN <= '0';
        BS_TEST <= '1';
        TAP_INSTRUCTION <= "110";
        INSTRUCCION_IR <= (others => '0');
        INSTRUCCION_DR <= (others => '0');
        REGISTRO_DISPLAY <= "1111111111111111";

           case state is
            -- Inicio
            when ST_Inicio=>
                -- Mostrar en el display el estado 0000
                REGISTRO_DISPLAY <= "0000000000000000";
                -- Ejecutamos la instrucción de RESET al TAP
                TAP_INSTRUCTION <= "001";
            -- Transicion 1
            when ST_T1=>
                -- Mostramos en el display el estado TT11
                REGISTRO_DISPLAY <= "1011101100010001";
            -- Verificación JTAG
            when ST_JTAG=>
                -- Envío del código bypass y recepción de los
códigos INSTRUCTION_CAPTURE
                TAP_INSTRUCTION <= "010";
                INSTRUCCION_IR <= "000100001000010"; --
Cargamos el registro de instrucciones con 'sample/preload'
                -- Mostrar en el display el estado T001
                REGISTRO_DISPLAY <= "1011000000000001";
            -- Transicion 2
            when ST_T2=>
                -- Comprobamos si la cadena recibida es la
correcta y guardamos el resultado en el registro
                JTAG_EN <= '1';
                if TAP_IR = "000010000100001" then
                    JTAG_TEST <= '0'; -- OK
                else
```

```
                    JTAG_TEST <= '1'; -- NOK
                end if;
                -- Mostrar en el display el estado TT22
                REGISTRO_DISPLAY <= "1011101100100010";
            -- Verificación Boundary-Scan
            when ST_BS=>
                -- Deshabilitamos la actualización del
registro JTAG
                JTAG_EN <= '0';
                -- Envío del código EXTEST y recepción de los
códigos INSTRUCTION_CAPTURE
                TAP_INSTRUCTION <= "101";
                INSTRUCCION_DR <=
"10110011100011110000111111000001111111000000011111
11100000000111111111000000000111111111000000000011111111111100
00000000011111111111110000001";
                -- Mostrar en el display el estado T002
                REGISTRO_DISPLAY <= "1011000000000010";
            -- Transicion 3
            when ST_T3=>
                -- Comprobamos si la cadena recibida es la
correcta y guardamos el resultado en el registro
                BS_EN <= '1';
                if TAP_DR =
"10110011100011110000111111000001111111000000011111
11100000000111111111000000000111111111000000000011111111111100
00000000011111111111110000000" then
                    BS_TEST <= '0'; -- OK
                else
                    BS_TEST <= '1'; -- NOK
                end if;
                -- Mostrar en el display el estado TT33
                REGISTRO_DISPLAY <= "1011101100110011";
                -- Iniciamos el contador para la siguiente
parte del test
                PAR_I_CELLS_RST <= '1';
                NETS_RST <= '1';
                NETS_GEN_RST <= '1';
            -- Realización Test eléctrico
            when ST_Test =>
                -- Deshabilitamos la actualización del
registro BS
                JTAG_EN <= '0';
                PAR_I_CELLS_RST <= '0';
                NETS_RST <= '0';
                NETS_GEN_RST <= '0';
                INICIAR_TEST <= '0';
                if PAR_I_CELLS > MAX_PAR_IN then
                    -- Finaliza el test
                    TEST_DONE <= '1';
                else
                    -- Se han recorrido todas las redes para
este test
                    if NETS = MAX_NETS then
                        -- Se inicia test W1
                        if W1W0_RESULTS = '0' then
                            --Indicamos test W1
```

```
                            W1W0_EN <= '1';
                            W1W0_NEXT <= '1';
                            -- Reiniciamos desplazador de
redes
                            W1W0_NETS_LOAD <= '1';
                            W1W0_NETS_INPUT <=
"0000000000001";
                        else
                            -- Finaliza el test
                            TEST_DONE <= '1';
                        end if;
                    else
                        -- Se ejecuta el test
                        INICIAR_TEST <= '1';
                        if VECTOR_GEN = '1' then
                            W1W0_NETS_EN <= '0';
                            -- Envío del vector a comprobar y
recepción del previo
                            TAP_INSTRUCTION <= "100"; -- IDLE
                            INSTRUCCION_DR <= VECTOR_TEST;
                        end if;
                    end if;
                end if;
                -- MOSTRAR VALOR EN PANTALLA TEST
                REGISTRO_DISPLAY <= "1011000000000011";--T003
            -- Mostrar fallos
            when ST_Resultados=>
                TAP_INSTRUCTION <= "110"; -- IDLE
                -- Mostrar Resultados
                if JTAG_RESULTS = '1' then
                    -- Hay fallo en JTAG
                    REGISTRO_DISPLAY <= "1111101010111100";--
FJTA
                else
                    -- No hay fallo en JTAG
                    if BS_RESULTS = '1' then
                        -- Hay fallo en BS
                        REGISTRO_DISPLAY <=
"1111111110001110";--FFBS
                    else
                        -- No hay fallo en BS ni JTAG
                        if sw(0) = '0' then
                            -- Indica fallos SA0
                            REGISTRO_DISPLAY <=
"1111111011000000";--FSA0
                            if btnL = '1' then
                                led <= SA0_VALUES(18 downto
3);
                            elsif btnR = '1' then
                                led <= SA0_VALUES(15 downto
0);
                            end if;
                        else
                            -- Indica fallos SA1
                            REGISTRO_DISPLAY <=
"1111111011000001";--FSA1
                            if btnL = '1' then
```

```
                                led <= SA1_VALUES(18 downto
3);
                            elsif btnR = '1' then
                                led <= SA1_VALUES(15 downto
0);
                            end if;
                        end if;
                    end if;
                end if;
        end case;
      end process;


      -- Generación del próximo vector de test
    process (clk)
    begin
        if rising_edge(clk) then
            case estado_test is
                -- Inicio
                when T_1=>
                    -- Deshabilitamos todas las celdas de
control
                    VECTOR_TEST <= (others => '0');
                    -- Envío realizado, comprobamos respuesta
                    if TAP_DONE = '1' then
                        estado_test <= T_7;
                    end if;
                    if INICIAR_TEST = '1' then
                        estado_test <= T_2;
                        NETS_EN <= '1';
                        NETS_GEN_EN <= '1';
                        IN_CELLS_RST <= '1';
                    end if;
                when T_2 =>
                    NETS_EN <= '0';
                    NETS_GEN_EN <= '0';
                    IN_CELLS_RST <= '0';
                    NETS_GEN_EN <= '0';
                    if IN_CELLS < TOTAL_I_CELLS then
                        IN_CELLS_EN <= '1';
                        estado_test <= T_3;
                    else
                        -- ENVIAR VECTOR
                        VECTOR_GEN <= '1';
                        -- DESPLAZAR W1W0_NETS para analizar
la siguiente red
                        W1W0_NETS_EN <= '1';
                        estado_test <= T_1;
                    end if;
                when T_3 =>
                    NETS_I_CELLS <= IN_CELLS(4 downto 0);
                    NETS_REP_EN <= '0';
                    IN_CELLS_EN <= '0';
                    if NETS_I_CELLS_NET = IN_CELLS then
                        -- No activa la red correcta
                        estado_test <= T_2;
                    else
```

```
                        -- Esta celda activa la red correcta
                        NETS_REP_EN <= '1';
                        --Comprobamos si es la aparición
correcta
                        if NETS_REP = PAR_I_CELLS then
                            -- Aparición correcta, ejecutamos
W1W0
                            estado_test <= T_4;
                        else
                            -- Aparición incorrecta,
comprobamos si hay más celdas para la misma red
                            NETS_I_CELLS <= IN_CELLS(4 downto
0) + 1;
                            if NETS_I_CELLS_NET = NETS_GEN
then
                                -- Quedan celdas para la red
                                NETS_REP_EN <= '1';
                                IN_CELLS_EN <= '1';
                            else
                                -- No hay más celdas para esta
red
                                estado_test <= T_4;
                            end if;
                        end if;
                    end if;
                when T_4 =>
                    NETS_REP_EN <= '0';
                    NETS_I_CELLS <= IN_CELLS(4 downto 0);
                    -- Walking 0
                    if W1W0_RESULTS = '0' then

VECTOR_TEST(conv_integer(NETS_I_CELLS_BSC)) <= W1W0
_NETS_OUTPUT(conv_integer(NETS_GEN));
                    -- Walking 1
                    else

VECTOR_TEST(conv_integer(NETS_I_CELLS_BSC)) <= not W1W0
_NETS_OUTPUT(conv_integer(NETS_GEN));
                    end if;
                    C_CELLS_RST <= '1';
                    estado_test <= T_5;
                when T_5 =>
                    C_CELLS_RST <= '0';
                    C_CELLS_EN <= '0';
                    if C_CELLS < TOTAL_C_CELLS then
                        estado_test <= T_6;
                    else
                        -- No quedan CTRL CELLS, vamos a por
la siguiente input_cell
                        estado_test <= T_2;
                    end if;
                when T_6 =>
                    -- Comprobamos si es la CTRL CELL correcta
                    CTRL_CELLS <= C_CELLS(4 downto 0);
                    -- Sí es la CTRL CELL
                    if CTRL_CELLS_BSC = NETS_I_CELLS_BSC then
                        -- Se habilita la BSC desde la CTRL
```

```
cell

VECTOR_TEST(conv_integer(CTRL_CELLS_CTRL)) <=
CTRL_CELLS_VALUE;
                        NETS_GEN_EN <= '1';
                        -- CTRL CELL habilitada, vamos a por
la siguiente INPUT_CELL
                        estado_test <= T_2;
                    -- No es la CTRL CELL
                    else
                        C_CELLS_EN <= '1';
                        estado_test <= T_5;
                    end if;
                -- Comenzamos comprobación del vector
respuesta
                when T_7 =>
                    OUT_CELLS_RST <= '1';
                    estado_test <= T_8;
                when T_8 =>
                    OUT_CELLS_RST <= '0';
                    OUT_CELLS_EN <= '0';
                    if OUT_CELLS = TOTAL_O_CELLS then
                        -- Probamos el siguiente vector
                        estado_test <= T_1;
                    else
                        -- Analizamos la celda de salida
                        NETS_O_CELLS <= OUT_CELLS (4 downto
0);
                        if
TAP_DR(conv_integer(NETS_O_CELLS_BSC)) = '0' then
                            -- No hay fallo Stuck-at-1
                            SA1_ENABLE <= '1';
                            SA1_UPDATE <= SA0_VALUES;
                            SA1_UPDATE
(conv_integer(OUT_CELLS)) <= '0';
                            CARGAR_VALORES_SA1 <= SA1_UPDATE;
                        else
                            -- No hay fallo Stuck-at-0
                            SA0_ENABLE <= '1';
                            SA0_UPDATE <= SA0_VALUES;
                            SA0_UPDATE
(conv_integer(OUT_CELLS)) <= '0';
                            CARGAR_VALORES_SA0 <= SA0_UPDATE;
                        end if;
                        OUT_CELLS_EN <= '1';
                        estado_test <= T_8;
                    end if;
            end case;
        end if;
    end process;


-- Bloques VHDL incorporados al fichero

    -- Controlador TAP
        TAP_controlador: TAP_Controller PORT MAP (
            RELOJ => clk,
```

```
        INSTRUCCION => TAP_INSTRUCTION,
        DONE => TAP_DONE,
        TDI_PMOD => TDI_PLACA,
        TCK_PMOD => TCK_PLACA,
        TMS_PMOD => TMS_PLACA,
        TDO_PMOD => TDO_PLACA,
        IR_VALUES => TAP_IR,
        IR_INSTRUCTION => INSTRUCCION_IR,
        DR_VALUES => TAP_DR,
        DR_INSTRUCTION => INSTRUCCION_DR
    );

-- Registro JTAG
    JTAG_status: Register_1bit PORT MAP (
        DATA => JTAG_TEST,
        EN => JTAG_EN,
        RELOJ => clk,
        Q => JTAG_RESULTS
    );

-- Registro BS
    BS_status: Register_1bit PORT MAP (
        DATA => BS_TEST,
        EN => BS_EN,
        RELOJ => clk,
        Q => BS_RESULTS
    );

-- Display 7 seg
    display_controller: Display_7seg PORT MAP (
        BCD => VALOR_DISPLAY,
        RELOJ => clk,
        SEGMENTOS => seg,
        ANODOS => an,
        ENABLE => MOSTRAR_DISPLAY
    );

-- Registro 16 bits
    display_register: Register_16bits PORT MAP (
        DATA => REGISTRO_DISPLAY,
        RELOJ => clk,
        EN => ACTUALIZAR_DISPLAY,
        Q => VALOR_DISPLAY
    );

-- Contador máximo celdas entrada paralelas
    contador_redes_paralelas: Contador_4bits PORT MAP (
        EN => PAR_I_CELLS_EN,
        RELOJ => clk,
        RESET => PAR_I_CELLS_RST,
        OUTPUT => PAR_I_CELLS
    );

-- Contador redes
    contador_redes: Contador_4bits PORT MAP (
        EN => NETS_EN,
        RELOJ => clk,
```

```
        RESET => NETS_RST,
        OUTPUT => NETS
    );

-- Contador redes generadas
    contador_redes_generadas: Contador_4bits PORT MAP (
        EN => NETS_GEN_EN,
        RELOJ => clk,
        RESET => NETS_GEN_RST,
        OUTPUT => NETS_GEN
    );

-- Contador redes repetidas
    contador_redes_repetidas: Contador_4bits PORT MAP (
        EN => NETS_REP_EN,
        RELOJ => clk,
        RESET => NETS_REP_RST,
        OUTPUT => NETS_REP
    );

-- Registro W1W0
    W1W0: Register_1bit PORT MAP (
        DATA => W1W0_NEXT,
        EN => W1W0_EN,
        RELOJ => clk,
        Q => W1W0_RESULTS
    );

-- Vector W1W0
    W1W0_NETS: Shift_Register_13bits PORT MAP (
        RELOJ => clk,
        EN => W1W0_NETS_EN,
        CARGAR => W1W0_NETS_LOAD,
        PARALLEL_IN => W1W0_NETS_INPUT,
        PARALLEL_OUT => W1W0_NETS_OUTPUT
    );

-- Contador input cells
    contador_IN_cells: Contador_8bits PORT MAP (
        EN => IN_CELLS_EN,
        RELOJ => clk,
        RESET => IN_CELLS_RST,
        OUTPUT => IN_CELLS
    );

-- Contador output cells
    contador_OUT_cells: Contador_8bits PORT MAP (
        EN => OUT_CELLS_EN,
        RELOJ => clk,
        RESET => OUT_CELLS_RST,
        OUTPUT => OUT_CELLS
    );

-- Contador CTRL cells
    contador_C_cells: Contador_8bits PORT MAP (
        EN => C_CELLS_EN,
        RELOJ => clk,
```

```
            RESET => C_CELLS_RST,
            OUTPUT => C_CELLS
        );

    -- Registro de input cells
        registro_IN_CELLS: I_CELLS_NETS PORT MAP (
           elemento => NETS_I_CELLS,
           red => NETS_I_CELLS_NET,
           BSC => NETS_I_CELLS_BSC
        );

    -- Registro de outnput cells
        registro_OUT_CELLS: O_CELLS_NETS PORT MAP (
           elemento => NETS_O_CELLS,
           red => NETS_O_CELLS_NET,
           BSC => NETS_O_CELLS_BSC
        );

    -- Registro de CTRL cells
        registro_CTRL_CELLS: C_CELLS_NETS PORT MAP (
           elemento => CTRL_CELLS,
           BSC => CTRL_CELLS_BSC,
           CTRL => CTRL_CELLS_CTRL,
           VALUE => CTRL_CELLS_VALUE
        );

    -- Registro fallos STUCK-AT_ONE
        SA1_faults: Register_19bits PORT MAP (
            RELOJ => clk,
            EN => SA1_ENABLE,
            PARALLEL_IN => CARGAR_VALORES_SA1,
            PARALLEL_OUT => SA1_VALUES
        );

    -- Registro fallos STUCK-AT_ZERO
        SA0_faults: Register_19bits PORT MAP (
            RELOJ => clk,
            EN => SA0_ENABLE,
            PARALLEL_IN => CARGAR_VALORES_SA0,
            PARALLEL_OUT => SA0_VALUES
        );

end Behavioral;
```