**Protokollierung zu Versuch 5**

**Gruppe 12**

Yazan Al-Kabbani (1648827)

Osama Hammoud (1536285)

Ahmet Taner Kahraman (1660926)

**Protokollierung Aufgabe 1**

VollAddierer - fullAdder.vhdl

*----------------------------------------------------------------------------*

    library IEEE;

    use IEEE.STD\_LOGIC\_1164.ALL;

    entity fullAdder is

        Port ( a : in STD\_LOGIC;

            b : in STD\_LOGIC;

            c\_in : in STD\_LOGIC;

            s : out STD\_LOGIC;

            c\_out : out STD\_LOGIC );

    end fullAdder;

    architecture Behavioral of fullAdder is

    begin

        s <= ( a XOR b ) XOR c\_in after 1ns;

        c\_out <= ( ( a XOR b ) AND c\_in ) OR ( a AND b) after 1ns;

    end Behavioral;

VollAddierer - xdcFullAdder.xdc

*----------------------------------------------------------------------------*

set\_property -dict {PACKAGE\_PIN G15 IOSTANDARD LVCMOS33} [get\_ports {a}]

set\_property -dict {PACKAGE\_PIN P15 IOSTANDARD LVCMOS33} [get\_ports {b}]

set\_property -dict {PACKAGE\_PIN W13 IOSTANDARD LVCMOS33} [get\_ports {c\_in}]

set\_property -dict {PACKAGE\_PIN M14 IOSTANDARD LVCMOS33} [get\_ports {s}]

set\_property -dict {PACKAGE\_PIN M15 IOSTANDARD LVCMOS33} [get\_ports {c\_out}]

VollAddierer - Testbench.vhdl

*------------------------------------------------------------------------*

    library IEEE;

    use IEEE.STD\_LOGIC\_1164.ALL;

    entity Testbench is

*--  Port ( );*

    end Testbench;

    architecture Behavioral of Testbench is

        component fullAdder

        port(a,b,c\_in: in std\_logic;

            s,c\_out: out std\_logic );

        end component;

        signal a, b, c\_in: std\_logic := '0';

        signal s, c\_out: std\_logic;

    begin

        uut: fullAdder port map (

            a => a, b => b, c\_in => c\_in, s => s, c\_out => c\_out

        );

        stim\_proc: process

            begin

                a <= '0'; b <= '0'; c\_in <= '0';

                wait for 100ns;

                a <= '0'; b <= '0'; c\_in <= '1';

                wait for 100ns;

                a <= '0'; b <= '1'; c\_in <= '0';

                wait for 100ns;

                a <= '0'; b <= '1'; c\_in <= '1';

                wait for 100ns;

                a <= '1'; b <= '0'; c\_in <= '0';

                wait for 100ns;

                a <= '1'; b <= '0'; c\_in <= '1';

                wait for 100ns;

                a <= '1'; b <= '1'; c\_in <= '0';

                wait for 100ns;

                a <= '1'; b <= '1'; c\_in <= '1';

                wait for 100ns;

            end process;

        response\_control: process

        begin

        REPORT " simulation :" SEVERITY note ;

        WAIT FOR 5 ns;

        ASSERT s ='0' AND c\_out = '0'  REPORT " right result " SEVERITY note;

        ASSERT NOT (s ='0' AND c\_out = '0') REPORT " wrong result " SEVERITY error;

        WAIT FOR 100ns;

       ASSERT s ='1' AND c\_out = '0'  REPORT " right result " SEVERITY note;

       ASSERT s ='0' OR c\_out = '1' REPORT " wrong result " SEVERITY error;

       WAIT FOR 100ns;

       ASSERT s ='1' AND c\_out = '0'  REPORT " right result " SEVERITY note;

       ASSERT NOT (s ='1' AND c\_out = '0') REPORT " wrong result " SEVERITY error;

       WAIT FOR 100ns;

      ASSERT s ='0' AND c\_out = '1'  REPORT " right result " SEVERITY note;

       ASSERT NOT (s ='0' AND c\_out = '1') REPORT " wrong result " SEVERITY error;

       WAIT FOR 100ns;

       ASSERT s ='1' AND c\_out = '0'  REPORT " right result " SEVERITY note;

       ASSERT NOT (s ='0' AND c\_out = '0') REPORT " wrong result " SEVERITY error ;

       WAIT FOR 100ns;

      ASSERT s ='0' AND c\_out = '1'  REPORT " right result " SEVERITY note;

       ASSERT NOT (s ='0' AND c\_out = '1') REPORT " wrong result " SEVERITY error;

       WAIT FOR 100ns;

       ASSERT s ='0' AND c\_out = '1'  REPORT " right result " SEVERITY note;

       ASSERT NOT (s ='0' AND c\_out = '1') REPORT " wrong result " SEVERITY error;

       WAIT FOR 100ns;

       ASSERT s ='1' AND c\_out = '1'  REPORT " right result " SEVERITY note;

       ASSERT NOT (s ='1' AND c\_out = '1') REPORT " wrong result " SEVERITY error;

       wait;

      end process;

    end Behavioral;

   Blinkmodul - led.vhdl

*------------------------------------------------------------------------*

    library IEEE;

    use IEEE.STD\_LOGIC\_1164.ALL;

    entity led is

        Port ( a : in STD\_LOGIC;

            b : in STD\_LOGIC;

            clk : in STD\_LOGIC;

            l1 : out STD\_LOGIC;

            l2 : out STD\_LOGIC;

            l3 : out STD\_LOGIC);

    end led;

    architecture Behavioral of led is

*-- 8ns \* 62.500.000 = 0,5s*

    constant period\_blink : integer := 62500000 ;

    signal led\_blink : std\_logic := '0';

    signal count1Hz : natural range 0 to period\_blink := 0;

    begin

        process(clk)

        begin

            if (a = '1') and (b = '1') then *-- an*

                l1 <= '1';

                l2 <= '1';

                l3 <= '1';

            elsif (a = '1') and (b = '0') then *-- blinken*

                l1 <= led\_blink;

                l2 <= led\_blink;

                l3 <= led\_blink;

            elsif (a = '0') and (b = '1') then *-- 50% leuchten (50% Duty Cycle)*

                l1 <= clk;

                l2 <= clk;

                l3 <= clk;

            else *-- aus*

                l1 <= '0';

                l2 <= '0';

                l3 <= '0';

            end if;

        end process;

  process(clk)

        begin

            if rising\_edge(clk) then

                if count1Hz = (period\_blink - 1) then

                    led\_blink <= (not led\_blink);

                    count1Hz <= 0;

                else

                    count1Hz <= (count1Hz + 1);

                end if;

            end if;

        end process;

    end Behavioral;

Blinkmodul – led\_TB.vhdl

*------------------------------------------------------------------------*

    library IEEE;

    use IEEE.STD\_LOGIC\_1164.ALL;

    entity led\_TB is

*--  Port ( );*

    end led\_TB;

    architecture Behavioral of led\_TB is

    COMPONENT led

    PORT(   a : in STD\_LOGIC;

            b : in STD\_LOGIC;

            clk : in STD\_LOGIC;

            l1 : out STD\_LOGIC;

            l2 : out STD\_LOGIC;

            l3 : out STD\_LOGIC );

    END COMPONENT;

    signal a : std\_logic := '0';

    signal b : std\_logic := '0';

    signal clk : std\_logic := '0';

    signal l1 : std\_logic;

    signal l2 : std\_logic;

    signal l3 : std\_logic;

    constant clk\_period : time :=  8 ns;

    begin

        uut: led PORT MAP (

        a => a,

        b => b,

        clk => clk,

        l1 => l1,

        l2 => l2,

        l3 => l3

        );

        clk\_process :process

        begin

            clk <= '0';

            wait for clk\_period/2;

            clk <= '1';

            wait for clk\_period/2;

        end process;

        stim\_proc: process

        begin

            wait for 2000000000ns;

            a <= '1';

            b <= '1';

            wait for 2000000000ns;

            a <= '1';

            b <= '0';

            wait for 2000000000ns;

            a <= '0';

            b <= '1';

            wait for 2000000000ns;

            a <= '0';

            b <= '0';

            wait;

        end process;

    end Behavioral;

Blinkmodul – xdc\_led.xdc

*----------------------------------------------------------------------------*

set\_property -dict { PACKAGE\_PIN G15 IOSTANDARD LVCMOS33 } [ get\_ports a]

set\_property -dict { PACKAGE\_PIN P15 IOSTANDARD LVCMOS33 } [ get\_ports b]

set\_property -dict { PACKAGE\_PIN K17 IOSTANDARD LVCMOS33 } [ get\_ports clk]

set\_property -dict { PACKAGE\_PIN M15 IOSTANDARD LVCMOS33 } [ get\_ports l1]

set\_property -dict { PACKAGE\_PIN G14 IOSTANDARD LVCMOS33 } [ get\_ports l2]

set\_property -dict { PACKAGE\_PIN D18 IOSTANDARD LVCMOS33 } [ get\_ports l3]

**Protokollierung Aufgabe 2**

1) Entwurf: Code in VHDL

2) Simulation mit Testbench (optional)

3) Design Implementierung

Translate -> Code in Schaltfunktion übersetzen

Map -> Übertragung auf vorhandene Hardware

(welche Transistoren werden wie verknüpft, ..)

Place & Route -> Interfaces (In – und Outputs, Anschlüsse) integrieren

4) Programming File (Bit-File) generieren

5) In Impac die generierte Bit-File auswählen und den FPGA programmieren

(FPGA muss über JTAG-Kabel mit PC verbunden sind)

**Vorbereitung 5 – die korrigierten Notizen**

**Aufgabe 13**

xxxx