# Reconfigurable Embedded Systems: Übung #3

Abgabe am Montag, 03. November 2014

**Günther Schindler, Sven Dorkenwald, Kilian Bender**

# Inhaltsverzeichnis

## LED-Schieberegister

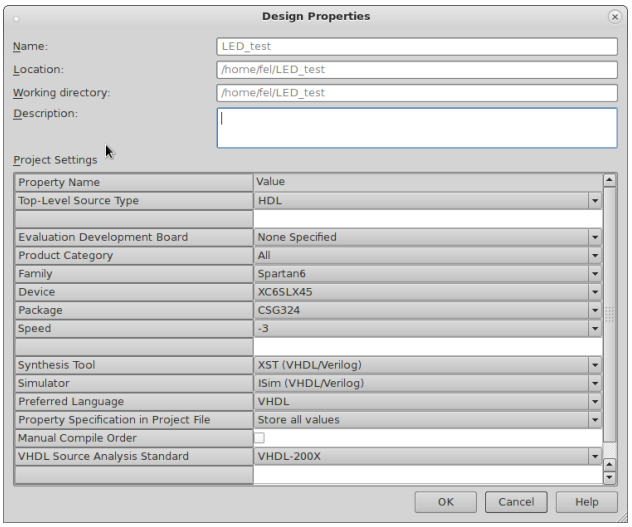
In der dritten Übung soll der in Übung zwei entwickelte LED-Schieberegister für das Atlys Entwicklungsboard implementiert werden. Dieser LED-Schieberegister basiert auf folgendem VHDL-Code.

```
entity led is
  port( clk_deb, clk_in, reset, din: in std_logic;
        LED_REG: out std_logic_vector(3 downto 0));
end led;

architecture Behavioral of led is

signal INT_REG: std_logic_vector(3 downto 0) := (others => '0');
type states is (S0, S1, S2, S3);
signal state: states := S0;
signal clk: std_logic;

begin
  process(clk, reset)
  begin
    if reset = '1' then
      INT_REG <= (others => '0');
      LED_REG <= (others => '0');
      state <= S0;
    elsif rising_edge(clk) then
      case state is
      when S0 =>
        LED_REG <= INT_REG;
        INT_REG <=  INT_REG(2 downto 0) & din;
        state <= S1;
      when S1 =>
        INT_REG <= INT_REG(2 downto 0) & din;
        state <= S2;
      when S2 =>
        INT_REG <= INT_REG(2 downto 0) & din;
        state <= S3;
      when S3 =>
        INT_REG <= INT_REG(2 downto 0) & din;
        state <= S0;
      end case;
    end if;
  end process;
end Behavioral;
```

## Design Properties

Der erste Schritt einer Implementierung für das Atlys-Board ist die Konfiguration der Desing Properties in der Xilinx ISE. Aus dem Atlys Reference Manual entnimmt man, dass es sich um einen Xilinx Spartan-6 LX-45 FPGA handelt. Folglich muss die Family auf Spartan6 und das Device auf XC65LX45 eingestellt

werden. Außerdem kann man dem Reference Manual entnehmen, dass das 324-pin BGA Package verwendet wird. Dies wird in der Xilinx ISE durch Auswahl von CSG324 als Package vollzogen.




## Debounce

Als Debounce wird das Entprellen des elektromechanischen Schalters verstanden. Dieser Schalter ruft bei Betätigung (statt des sofortigen elektrischen Kontakts) kurzzeitig ein mehrfaches Schließen und Öffnen des Kontakts hervor. Der Grund dieses Prellens sind Federungen an Bauteilen der Schaltmechanik.
Um diesem physikalischen Effekt entgegenzuwirken wird eine Möglichkeit benötigt, Schalterbetätigungen mit gewisser Frequenz zu filtern. Dies wird über Verwendung des vorgegebenen Debounce VHDL-Codes vorgenommen. Der VHDL-Code muss in das Projekt eingebunden werden. Die Schalter Prellen mit etwa 1 kHz. Danach wird der Debounce bezüglich des LED-Codes wie folgt eingebunden.

```
...
architecture Behavioral of led is
...
COMPONENT debounce
  PORT(
    clk : IN std_logic;
    rst : IN std_logic;
    din : IN std_logic;
    dout : OUT std_logic
    );
...
begin
  ...
  Inst_debounce: debounce PORT MAP(
    clk => clk_deb,
    rst => reset,
    din => clk_in,
    dout => clk
    );
  ...
  process(clk, reset)
```

```
  begin
  ...
  end process;
end Behavioral;
```

## Xilinx-.ucf-File

In den Xilinx-.ucf-File (User Constraint File) wird die Verbindung von logischen Netzen (in- und out-Ports im top level entity) mit physikalischen Pins hergestellt. So gibt es speziell für das Atlys Board das AtlysGeneral.ucf File, welches alle verwendbaren Pins enthält. Zusätzlich können auch Timing-Bedingungen abgebildet werden. So wird der auf dem Atlys Board enthaltene Takt als Referenz Takt für die Debounce-Schaltung verwendet.

```
# clock pin for Atlys rev C board
 NET "clk_deb"   LOC = "L15";

# onBoard Leds
 NET "LED_REG<0>" LOC = "U18";
 NET "LED_REG<1>" LOC = "M14";
 NET "LED_REG<2>" LOC = "N14";
 NET "LED_REG<3>" LOC = "L14";

# onBoard PUSH BUTTONS
 NET "reset" LOC = "P4";
 NET "clk_in" LOC = "F6";

# onBoard SWITCHES
 NET "din" LOC = "A10";
```

## Testbench

Mit der Testbench wollen wir die korrekte Einbundung und Funktion des Debounce testen.
Über die Konstante clk_deb_period setzen wir die Periodendauer auf 10ns und erhalten über den Prozess clk_deb_process einen freilaufenden Takt von 100MHz. Dieser Takt simuliert den Referenztakt auf dem Atlys-Board.

```
...
-- Clock period definitions
constant clk_deb_period : time := 10 ns;
...
BEGIN
  ...
  -- Clock process definitions
  clk_deb_process :process
  begin
    clk_deb <= '0';
    wait for clk_deb_period/2;
    clk_deb <= '1';
    wait for clk_deb_period/2;
```

```
  end process;
...
END;
```

Die Prozedur pulse wird aus der tb_debounce verwendet um verschiedene Pulsweiten zu simulieren. Mit der Prozedur werden die zu entprellenden Signale simuliert.

```
ARCHITECTURE behavior OF tb_led IS
  ...
  procedure pulse (delay: in natural; signal o: out std_logic) is
  begin
    o <= '0';
    wait for delay * 1000 * clk_deb_period;
    o <= '1';
    wait for delay * 1000 * clk_deb_period;
    o <= '0';
  end procedure;
...
BEGIN
  ...
END;
```

Über die doppelte for-Schleife werden je 10 Impulse der Weite 1.0101kHz, 1.000kHz, 0.990kHz und 0.980kHz. Es ist zu erwarten, dass der Debounce die beiden größeren Frequenzen filtert.

```
ARCHITECTURE behavior OF tb_led IS
  ...
BEGIN
  ...
  din <= '1';
  l1: for i in 0 to 3 loop
    l2: for j in 0 to 10 loop
      pulse((99 + i), clk_in);
    end loop;
  end loop;
  ...
END;
```

Wie erwartet werden die beiden größeren Frequenzen (bis 40ms) gefiltert. Danach werden die positiven Taktflanken von din gewertet. Nach 45ms ist das Ende des 4ten gezählten Zyklus von din erreicht und der Register wird auf die LEDs geschalten.