Lösung 4 Process Structure und State Machines

Jan Grapengeter

May 2, 2019

1 Process Structure

VHDL_main:

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

entity main is
    Port ( sw : in STD_LOGIC_vector(15 downto 0);
           ld :out STD_LOGIC_vector(15 downto 0));
end main;

architecture Behavioral of main is
begin
process(sw)
begin

ld(0)<=sw(0) and sw(1);

end process;

end Behavioral;
```

2 If Statement

1.:
VHDL_main:

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

entity main is
    Port ( sw : in STD_LOGIC_vector(15 downto 0);
           ld :out STD_LOGIC_vector(15 downto 0));
end main;

architecture Behavioral of main is

begin
```

```
process(sw)
begin

if(sw(1 downto 0)="11" ) then
ld(0)<='1';
else
ld(0)<='0';
end if;

end process;

end Behavioral;
```

2., 3.:
VHDL_main:

```
library IEEE;
use IEEE.STD_LOGIC_1164.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 main is
    Port ( sw : in STD_LOGIC_vector(15 downto 0);
           ld :out STD_LOGIC_vector(15 downto 0));
end main;

architecture Behavioral of main is

begin

process(sw)
begin

if(sw(1 downto 0)="11" or sw(1 downto 0)="10") then
ld(0)<='1';
else
ld(0)<='0';
end if;

if(sw(3 downto 2)="11") then
ld(2)<='1';
elsif(sw(3 downto 2)="10")then
ld(2)<='1';
else
```

```
ld(2)<='0';
end if;

end process;

end Behavioral;
```

4.: Funktional gleicher Code kann unterschiedliche Hardware instanzieren. Dies kann Auswirkungen auf die Funktion der Schaltung haben. Dies macht sich insbesondere bei der Statemachine Aufgabe in dieser Übung bemerkbar.

3 Case When Statement

1.:
VHDl_main:

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

entity main is
    Port ( sw : in STD_LOGIC_vector(15 downto 0);
           ld :out STD_LOGIC_vector(15 downto 0));
end main;

architecture Behavioral of main is

begin

process(sw)
begin

case sw(1 downto 0) is
when "11" =>
ld(0)<='1';
when "10" =>
ld(0)<='1';
when others =>
ld(0)<='0';
end case;

end process;

end Behavioral;
```

3.:
VHDL_main:

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

entity main is
```

```
    Port ( sw : in STD_LOGIC_vector(15 downto 0);
           ld :out STD_LOGIC_vector(15 downto 0));
end main;

architecture Behavioral of main is

begin

process(sw)
begin

case sw(1 downto 0) is
when "10" =>
ld(0)<='1';
when "01" =>
ld(0)<='1';
when others =>
ld(0)<='0';
end case;

if(sw(3 downto 2)="10" or sw(3 downto 2)="01") then
ld(2)<='1';
else
ld(2)<='0';
end if;

if(sw(5 downto 4)="10") then
ld(4)<='1';
elsif(sw(5 downto 4)="01") then
ld(4)<='1';
else
ld(4)<='0';
end if;

ld(6)<=sw(6) xor sw(7);

end process;

end Behavioral;
```

4 State Machines

1.:
VHDL_main:

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

entity main is
    Port ( sw : in STD_LOGIC_vector(15 downto 0);
           btnd : in std_logic;
           ld :out STD_LOGIC_vector(15 downto 0));
end main;

architecture Behavioral of main is

Signal zustand : std_logic_vector (1 downto 0) := "00";

begin

process(zustand,sw,btnd)
begin

case zustand is
when "00" =>
ld(0)<='1';
ld(1)<='1';
ld(2)<='0';
ld(3)<='0';
ld(4)<='0';
ld(5)<='0';
ld(7)<='0';
ld(10)<='0';
ld(11)<='0';
ld(12)<='0';
case sw is
when "0001" =>
zustand(0)<='1';
when others =>

end case;

when "01" =>
ld(0)<='0';
ld(1)<='0';
ld(2)<='1';
ld(3)<='1';
ld(4)<='1';
ld(5)<='0';
ld(7)<='0';
ld(10)<='0';
ld(11)<='0';
ld(12)<='0';
case sw is
when "0010" =>
zustand(1)<='1';
when others =>

end case;

when "11" =>
ld(0)<='0';
ld(1)<='0';
ld(2)<='0';
ld(3)<='0';
ld(4)<='0';
ld(5)<='1';
ld(7)<='1';
ld(10)<='0';
ld(11)<='0';
ld(12)<='0';
case sw is
when "0100" =>
zustand(0)<='0';
when others =>

end case;

when "10" =>
ld(0)<='0';
ld(1)<='0';
ld(2)<='0';
ld(3)<='0';
ld(4)<='0';
ld(5)<='0';
ld(7)<='0';
ld(10)<='1';
ld(11)<='1';
ld(12)<='1';
case sw is
when "1000" =>
zustand(1)<='0';
when others =>

end case;
end case;
end process;
end Behavioral;
```

2.:
Die Umschaltung zwischen den einzelnen Zuständen funktioniert nur schlecht. Die Schaltung ist im Zustandswechsel nicht eindeutig festgelegt. Dies lässt sich in der nächsten Übung mit den clock-Statements lösen.
3.:
VHDL_main:

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

entity main is
    Port ( sw : in STD_LOGIC_vector(15 downto 0);
           ld :out STD_LOGIC_vector(15 downto 0));
end main;

architecture Behavioral of main is
```

```
component volladdierer is
port(A1 : in STD_LOGIC;
     A2 : in STD_LOGIC;
     cin : in std_logic;
     Ubertrag : out STD_LOGIC;
     Summe : out STD_LOGIC);
end component;

Signal a,b,cin,u,s : std_logic_vector (3 downto 0);

begin

a(0)<=sw(1);
a(1)<=sw(2);
a(2)<=sw(3);
a(3)<=sw(4);
b(0)<=sw(5);
b(1)<=sw(6);
b(2)<=sw(7);
b(3)<=sw(8);

GENs_voll:
for i in 0 to 3 generate
voll1:
if (i=0) generate
full0 : volladdierer port map
(A1=>a(i),A2=>b(i),cin=>sw(0),Ubertrag=>u(i),Summe=>s(i));
end generate;
voll2:
if(i/=0) generate
full1 : volladdierer port map
(A1=>a(i),A2=>b(i),cin=>u(i-1),Ubertrag=>u(i),Summe=>s(i));
end generate;
end generate;

ld(1)<=s(0);
ld(2)<=s(1);
ld(3)<=s(2);
ld(4)<=s(3);
ld(5)<=u(3);

end Behavioral;
```