Lösung 5 Clock Instantiation

Jan Grapengeter

May 2, 2019

1 Einlesen eines Frequenzsignals

1.:
VHDL_main:

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

entity main is
    Port ( sw : in STD_LOGIC_vector(15 downto 0);
           uhr : 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";
Signal teiler : std_logic_vector (32 downto 0);

begin

process(uhr,teiler)
begin
if(rising_edge(uhr)) then
teiler <= teiler+1;
end if;
end process;

process(zustand,sw,uhr,teiler)
begin
if(rising_edge(uhr)) then
case zustand is
when "00" =>
if (sw(0)='1') then
zustand(0)<='1';
end if;
when "01" =>
if (sw(1)='1') then
```

```
zustand(1)<='1';
end if;
when "11" =>
if (sw(2)='1') then
zustand(0)<='0';
end if;
when "10" =>
if (sw(3)='1') then
zustand(1)<='0';
end if;
end case;
end if;
end process;

process(zustand)
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';
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';
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';
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';
end case;
end process;
end Behavioral;
```

Contraints:

```
set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 } [get_ports  sw[0]];
set_property -dict { PACKAGE_PIN V16   IOSTANDARD LVCMOS33 } [get_ports  sw[1]];
set_property -dict { PACKAGE_PIN w16   IOSTANDARD LVCMOS33 } [get_ports  sw[2]];
set_property -dict { PACKAGE_PIN w17   IOSTANDARD LVCMOS33 } [get_ports  sw[3]];
set_property -dict { PACKAGE_PIN w15   IOSTANDARD LVCMOS33 } [get_ports  sw[4]];
set_property -dict { PACKAGE_PIN v15   IOSTANDARD LVCMOS33 } [get_ports  sw[5]];
set_property -dict { PACKAGE_PIN w14   IOSTANDARD LVCMOS33 } [get_ports  sw[6]];
set_property -dict { PACKAGE_PIN w13   IOSTANDARD LVCMOS33 } [get_ports  sw[7]];
set_property -dict { PACKAGE_PIN v2   IOSTANDARD LVCMOS33 } [get_ports  sw[8]];
set_property -dict { PACKAGE_PIN t3   IOSTANDARD LVCMOS33 } [get_ports  sw[9]];
set_property -dict { PACKAGE_PIN t2   IOSTANDARD LVCMOS33 } [get_ports  sw[10]];
set_property -dict { PACKAGE_PIN r3   IOSTANDARD LVCMOS33 } [get_ports  sw[11]];
set_property -dict { PACKAGE_PIN w2   IOSTANDARD LVCMOS33 } [get_ports  sw[12]];
set_property -dict { PACKAGE_PIN u1   IOSTANDARD LVCMOS33 } [get_ports  sw[13]];
set_property -dict { PACKAGE_PIN t1   IOSTANDARD LVCMOS33 } [get_ports  sw[14]];
set_property -dict { PACKAGE_PIN r2   IOSTANDARD LVCMOS33 } [get_ports  sw[15]];

set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS33 } [get_ports  ld[0]];
set_property -dict { PACKAGE_PIN e19   IOSTANDARD LVCMOS33 } [get_ports  ld[1]];
set_property -dict { PACKAGE_PIN u19   IOSTANDARD LVCMOS33 } [get_ports  ld[2]];
set_property -dict { PACKAGE_PIN v19   IOSTANDARD LVCMOS33 } [get_ports  ld[3]];
set_property -dict { PACKAGE_PIN w18   IOSTANDARD LVCMOS33 } [get_ports  ld[4]];
set_property -dict { PACKAGE_PIN u15   IOSTANDARD LVCMOS33 } [get_ports  ld[5]];
set_property -dict { PACKAGE_PIN u14   IOSTANDARD LVCMOS33 } [get_ports  ld[6]];
set_property -dict { PACKAGE_PIN v14   IOSTANDARD LVCMOS33 } [get_ports  ld[7]];
set_property -dict { PACKAGE_PIN v13   IOSTANDARD LVCMOS33 } [get_ports  ld[8]];
set_property -dict { PACKAGE_PIN v3   IOSTANDARD LVCMOS33 } [get_ports  ld[9]];
set_property -dict { PACKAGE_PIN w3   IOSTANDARD LVCMOS33 } [get_ports  ld[10]];
set_property -dict { PACKAGE_PIN u3   IOSTANDARD LVCMOS33 } [get_ports  ld[11]];
set_property -dict { PACKAGE_PIN p3   IOSTANDARD LVCMOS33 } [get_ports  ld[12]];
set_property -dict { PACKAGE_PIN n3   IOSTANDARD LVCMOS33 } [get_ports  ld[13]];
set_property -dict { PACKAGE_PIN p1   IOSTANDARD LVCMOS33 } [get_ports  ld[14]];
set_property -dict { PACKAGE_PIN l1   IOSTANDARD LVCMOS33 } [get_ports  ld[15]];

set_property -dict { PACKAGE_PIN w5   IOSTANDARD LVCMOS33 } [get_ports  uhr];
```

2 Frequenzteiler

1.:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_unsigned.all;
use IEEE.numeric_std.all;

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

architecture Behavioral of main is

Signal teiler : std_logic_vector (32 downto 0);

begin

process(uhr,teiler)
begin
if(rising_edge(uhr)) then
teiler<=teiler+1;
end if;

ld(14)<= teiler(26);

end process;
end Behavioral;
```

Exakte Frequenz ist 100MHz/2n̂ (n=26,27)
2.:
VHDL_main:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_unsigned.all;
use IEEE.numeric_std.all;

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

architecture Behavioral of main is

Signal teiler : std_logic_vector (26 downto 0):="10111110101111000010000000";
```

```
signal a : std_Logic;

begin

process(uhr,teiler)
begin
if(rising_edge(uhr)) then
teiler<=teiler-1;
if(teiler="00000000000000000000000000000") then
teiler<="10111110101111000010000000";
a<=not a;
end if;
end if;
end process;

ld(0)<=a;

end Behavioral;
```

3.:
VHDL_main:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_unsigned.all;
use IEEE.numeric_std.all;

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

architecture Behavioral of main is

Signal teiler : std_logic_vector (1 downto 0) :="11";

begin

process(uhr,teiler)
begin
if(rising_edge(uhr)) then
teiler<=teiler-1;
if(teiler="00") then
ld(0)<='1';
else
ld(0)<='0';
end if;
end if;

end process;
end Behavioral;
```

3 Synchrones und Asynchrones Design

1.:

Synchron vs Asynchron		
L	Synchron	Asynchron
Vorteile	Einsparung von FFs, kleinere Chips möglich	Reset wird schneller ausgeführt
Nachteile	Reset kann verzögert sein. Mögliche Metastabilität bei Assertierung des Reset-Signals	Mögliche Metastabilität bei Deassertierung des Reset-Signals

Table 1: Synchron vs Asynchron

[?]
2.:
VHDL_main:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_unsigned.all;
use IEEE.numeric_std.all;

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

architecture Behavioral of main is

signal reset1,reset2 : std_logic :='0';
signal temp1,temp2 : std_logic_vector(7 downto 0) :="00000000";

begin

reset1<=sw(0);
reset2<=sw(1);

process(uhr)
begin

if(rising_edge(uhr)) then
if(reset1='1') then
temp1<="00000000";
else
temp1<=temp1+1;
end if;
end if;

if(reset2='1') then
```

```
temp2<="00000000";
elsif(rising_edge(uhr)) then
temp2<=temp2+1;
end if;

end process;
ld(7 downto 0)<=temp1;
ld(15 downto 8)<=temp2;
end Behavioral;
```

3.:
VHDL_main:

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

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

architecture Behavioral of main is

Signal zustand : std_logic_vector (2 downto 0):="000";
Signal teiler : std_logic_vector (32 downto 0);

begin

process(zustand,sw,uhr,teiler)
begin
if(rising_edge(uhr)) then
if(btnd='1') then
zustand<="000";
else
case zustand is
when "000" =>
if (sw="000001") then
zustand<="001";
end if;

when "001" =>
if (sw="000010") then
zustand<="010";
end if;

when "010" =>
if (sw="000100") then
zustand<="011";
```

```
end if;

when "011" =>
if (sw="001000") then
zustand<="100";
end if;

when "100" =>
if (sw="010000") then
zustand<="101";
end if;

when "101" =>
if (sw="100000") then
zustand<="000";
end if;
when others=>
zustand<="000";
end case;
end if;
end if;
end process;

process(zustand)
begin
case zustand is
when "000" =>
ld(0)<='1';
ld(1)<='1';
ld(2)<='0';
ld(3)<='0';
ld(4)<='0';
ld(5)<='0';
ld(6)<='0';
ld(7)<='0';
ld(8)<='0';
ld(9)<='0';
ld(10)<='0';
ld(11)<='0';
when "001" =>
ld(0)<='0';
ld(1)<='0';
ld(2)<='1';
ld(3)<='1';
ld(4)<='0';
ld(5)<='0';
ld(6)<='0';
ld(7)<='0';
ld(8)<='0';
ld(9)<='0';
ld(10)<='0';
ld(11)<='0';
```

```
when "010" =>
ld(0)<='0';
ld(1)<='0';
ld(2)<='0';
ld(3)<='0';
ld(4)<='1';
ld(5)<='1';
ld(6)<='0';
ld(7)<='0';
ld(8)<='0';
ld(9)<='0';
ld(10)<='0';
ld(11)<='0';
when "011" =>
ld(0)<='0';
ld(1)<='0';
ld(2)<='0';
ld(3)<='0';
ld(4)<='0';
ld(5)<='0';
ld(6)<='1';
ld(7)<='1';
ld(8)<='0';
ld(9)<='0';
ld(10)<='0';
ld(11)<='0';
when "100" =>
ld(0)<='0';
ld(1)<='0';
ld(2)<='0';
ld(3)<='0';
ld(4)<='0';
ld(5)<='0';
ld(6)<='0';
ld(7)<='0';
ld(8)<='1';
ld(9)<='1';
ld(10)<='0';
ld(11)<='0';
when "101" =>
ld(0)<='0';
ld(1)<='0';
ld(2)<='0';
ld(3)<='0';
ld(4)<='0';
ld(5)<='0';
ld(6)<='0';
ld(7)<='0';
ld(8)<='0';
ld(9)<='0';
ld(10)<='1';
ld(11)<='1';
```

```
when others=>
ld(0)<='1';
ld(1)<='0';
ld(2)<='1';
ld(3)<='0';
ld(4)<='1';
ld(5)<='0';
ld(6)<='1';
ld(7)<='0';
ld(8)<='1';
ld(9)<='0';
ld(10)<='1';
ld(11)<='0';
end case;
end process;
end Behavioral;
```

Contraints:

```
set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 } [get_ports  sw[0]];
set_property -dict { PACKAGE_PIN V16   IOSTANDARD LVCMOS33 } [get_ports  sw[1]];
set_property -dict { PACKAGE_PIN w16   IOSTANDARD LVCMOS33 } [get_ports  sw[2]];
set_property -dict { PACKAGE_PIN w17   IOSTANDARD LVCMOS33 } [get_ports  sw[3]];
set_property -dict { PACKAGE_PIN w15   IOSTANDARD LVCMOS33 } [get_ports  sw[4]];
set_property -dict { PACKAGE_PIN v15   IOSTANDARD LVCMOS33 } [get_ports  sw[5]];
set_property -dict { PACKAGE_PIN w14   IOSTANDARD LVCMOS33 } [get_ports  sw[6]];
set_property -dict { PACKAGE_PIN w13   IOSTANDARD LVCMOS33 } [get_ports  sw[7]];
set_property -dict { PACKAGE_PIN v2   IOSTANDARD LVCMOS33 } [get_ports  sw[8]];
set_property -dict { PACKAGE_PIN t3   IOSTANDARD LVCMOS33 } [get_ports  sw[9]];
set_property -dict { PACKAGE_PIN t2   IOSTANDARD LVCMOS33 } [get_ports  sw[10]];
set_property -dict { PACKAGE_PIN r3   IOSTANDARD LVCMOS33 } [get_ports  sw[11]];
set_property -dict { PACKAGE_PIN w2   IOSTANDARD LVCMOS33 } [get_ports  sw[12]];
set_property -dict { PACKAGE_PIN u1   IOSTANDARD LVCMOS33 } [get_ports  sw[13]];
set_property -dict { PACKAGE_PIN t1   IOSTANDARD LVCMOS33 } [get_ports  sw[14]];
set_property -dict { PACKAGE_PIN r2   IOSTANDARD LVCMOS33 } [get_ports  sw[15]];

set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS33 } [get_ports  ld[0]];
set_property -dict { PACKAGE_PIN e19   IOSTANDARD LVCMOS33 } [get_ports  ld[1]];
set_property -dict { PACKAGE_PIN u19   IOSTANDARD LVCMOS33 } [get_ports  ld[2]];
set_property -dict { PACKAGE_PIN v19   IOSTANDARD LVCMOS33 } [get_ports  ld[3]];
set_property -dict { PACKAGE_PIN w18   IOSTANDARD LVCMOS33 } [get_ports  ld[4]];
set_property -dict { PACKAGE_PIN u15   IOSTANDARD LVCMOS33 } [get_ports  ld[5]];
set_property -dict { PACKAGE_PIN u14   IOSTANDARD LVCMOS33 } [get_ports  ld[6]];
set_property -dict { PACKAGE_PIN v14   IOSTANDARD LVCMOS33 } [get_ports  ld[7]];
set_property -dict { PACKAGE_PIN v13   IOSTANDARD LVCMOS33 } [get_ports  ld[8]];
set_property -dict { PACKAGE_PIN v3   IOSTANDARD LVCMOS33 } [get_ports  ld[9]];
set_property -dict { PACKAGE_PIN w3   IOSTANDARD LVCMOS33 } [get_ports  ld[10]];
set_property -dict { PACKAGE_PIN u3   IOSTANDARD LVCMOS33 } [get_ports  ld[11]];
set_property -dict { PACKAGE_PIN p3   IOSTANDARD LVCMOS33 } [get_ports  ld[12]];
set_property -dict { PACKAGE_PIN n3   IOSTANDARD LVCMOS33 } [get_ports  ld[13]];
set_property -dict { PACKAGE_PIN p1   IOSTANDARD LVCMOS33 } [get_ports  ld[14]];
set_property -dict { PACKAGE_PIN l1   IOSTANDARD LVCMOS33 } [get_ports  ld[15]];
```

```
set_property -dict { PACKAGE_PIN w5   IOSTANDARD LVCMOS33 } [get_ports  uhr];
set_property -dict { PACKAGE_PIN u17   IOSTANDARD LVCMOS33 } [get_ports  btnd];
```

4.:
VHDL_main:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_unsigned.all;
use IEEE.numeric_std.all;

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

architecture Behavioral of main is

Signal teiler : std_logic_vector (23 downto 0):="100110001001011010000000";
signal start,stop,reset : std_Logic :='0';
signal stopuhr : std_logic_vector(15 downto 0):="0000000000000000";

begin

process(uhr,teiler,stopuhr,start,stop)
begin
if(rising_edge(uhr)) then
if(reset='1') then
reset<='0';
teiler<="100110001001011010000000";
stopuhr<="0000000000000000";
start<='0';
stop<='0';
else
if(start='1' and stop='0') then
if(teiler="000000000000000000000000") then
teiler<="100110001001011010000000";
stopuhr<=stopuhr+1;
else
teiler<=teiler-1;
end if;
end if;
end if;
end if;

if(rising_edge(uhr)) then
if(sw(0)='1') then
start<='1';
end if;
end if;
```

```
if(rising_edge(uhr)) then
if(sw(1)='1') then
stop<='1';
end if;
end if;

if(rising_edge(uhr)) then
if(sw(2)='1') then
reset<='1';
end if;
end if;

end process;

ld<=stopuhr;

end Behavioral;
```

Maximale Auflösung bei einer 100MHz Uhr: 10^-8. Durch Oversampling oder das zusätzliche Abfragen von fallenden Taktflanken, lässt sich die Auflösung jedoch erhöhen. Die maximale Auflösung entpricht dann der Gatterlaufzeit der D-FF.
Das Verwenden eines Ringoszillators mit einer Frequenz>100MHz erhöht ebenfalls die Auflösung. Kompliziertere Methoden wie Vernier-Delay-Lines sollten noch nicht versucht werden, mit diesen ist es jedoch möglich, die Zeitauflösung in den Pikosenkundenbereich zu drücken, also deutlich unterhalb der D-FF-Gatterlaufzeiten.
Maximale Zeit, die gemessen werden kann: ((2^17)-1)*0.1s

4 Gatterlaufzeiten

1.:
VHDL_main:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_unsigned.all;
use IEEE.numeric_std.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 Inverter is
    Port ( ein : in STD_LOGIC;
           aus : out STD_LOGIC);
end component;

Signal a,b : std_logic_vector (4 downto 0);
```

```
signal teiler : std_logic_vector (31 downto 0);

begin

GEN_INV:
for i in 0 to 4 generate
inve0:
if (i=0) generate
inv1 : Inverter port map
(ein=>b(4),aus=>b(i));
end generate;
inve1:
if(i/=0) generate
inv1 : Inverter port map
(ein=>b(i-1),aus=>b(i));
end generate;
end generate;

process(b,teiler)
begin

if(rising_edge(b(0))) then
teiler<=teiler+1;
end if;
ld(12)<=teiler(29);

end process;

end Behavioral;
```

Constraints:

```
set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 } [get_ports  sw[0]];
set_property -dict { PACKAGE_PIN V16   IOSTANDARD LVCMOS33 } [get_ports  sw[1]];
set_property -dict { PACKAGE_PIN w16   IOSTANDARD LVCMOS33 } [get_ports  sw[2]];
set_property -dict { PACKAGE_PIN w17   IOSTANDARD LVCMOS33 } [get_ports  sw[3]];
set_property -dict { PACKAGE_PIN w15   IOSTANDARD LVCMOS33 } [get_ports  sw[4]];
set_property -dict { PACKAGE_PIN v15   IOSTANDARD LVCMOS33 } [get_ports  sw[5]];
set_property -dict { PACKAGE_PIN w14   IOSTANDARD LVCMOS33 } [get_ports  sw[6]];
set_property -dict { PACKAGE_PIN w13   IOSTANDARD LVCMOS33 } [get_ports  sw[7]];
set_property -dict { PACKAGE_PIN v2   IOSTANDARD LVCMOS33 } [get_ports  sw[8]];
set_property -dict { PACKAGE_PIN t3   IOSTANDARD LVCMOS33 } [get_ports  sw[9]];
set_property -dict { PACKAGE_PIN t2   IOSTANDARD LVCMOS33 } [get_ports  sw[10]];
set_property -dict { PACKAGE_PIN r3   IOSTANDARD LVCMOS33 } [get_ports  sw[11]];
set_property -dict { PACKAGE_PIN w2   IOSTANDARD LVCMOS33 } [get_ports  sw[12]];
set_property -dict { PACKAGE_PIN u1   IOSTANDARD LVCMOS33 } [get_ports  sw[13]];
set_property -dict { PACKAGE_PIN t1   IOSTANDARD LVCMOS33 } [get_ports  sw[14]];
set_property -dict { PACKAGE_PIN r2   IOSTANDARD LVCMOS33 } [get_ports  sw[15]];

set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS33 } [get_ports  ld[0]];
set_property -dict { PACKAGE_PIN e19   IOSTANDARD LVCMOS33 } [get_ports  ld[1]];
set_property -dict { PACKAGE_PIN u19   IOSTANDARD LVCMOS33 } [get_ports  ld[2]];
```

```
set_property -dict { PACKAGE_PIN v19   IOSTANDARD LVCMOS33 } [get_ports  ld[3]];
set_property -dict { PACKAGE_PIN w18   IOSTANDARD LVCMOS33 } [get_ports  ld[4]];
set_property -dict { PACKAGE_PIN u15   IOSTANDARD LVCMOS33 } [get_ports  ld[5]];
set_property -dict { PACKAGE_PIN u14   IOSTANDARD LVCMOS33 } [get_ports  ld[6]];
set_property -dict { PACKAGE_PIN v14   IOSTANDARD LVCMOS33 } [get_ports  ld[7]];
set_property -dict { PACKAGE_PIN v13   IOSTANDARD LVCMOS33 } [get_ports  ld[8]];
set_property -dict { PACKAGE_PIN v3   IOSTANDARD LVCMOS33 } [get_ports  ld[9]];
set_property -dict { PACKAGE_PIN w3   IOSTANDARD LVCMOS33 } [get_ports  ld[10]];
set_property -dict { PACKAGE_PIN u3   IOSTANDARD LVCMOS33 } [get_ports  ld[11]];
set_property -dict { PACKAGE_PIN p3   IOSTANDARD LVCMOS33 } [get_ports  ld[12]];
set_property -dict { PACKAGE_PIN n3   IOSTANDARD LVCMOS33 } [get_ports  ld[13]];
set_property -dict { PACKAGE_PIN p1   IOSTANDARD LVCMOS33 } [get_ports  ld[14]];
set_property -dict { PACKAGE_PIN l1   IOSTANDARD LVCMOS33 } [get_ports  ld[15]];

set_property ALLOW_COMBINATORIAL_LOOPS true [get_nets -of_objects [get_cells ld_OBUF[0]_inst_i_1]]

set_property SEVERITY {Warning}  [get_drc_checks LUTLP-1]

set_property SEVERITY {Warning} [get_drc_checks NSTD-1]
```

Inverter:

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

entity Inverter is
    Port ( ein : in STD_LOGIC;
           aus : out STD_LOGIC);
end Inverter;

architecture Behavioral of Inverter is

begin

aus <= not ein;

end Behavioral;
```

Die Zeit, die die LED leuchtet x2, ist die Ausgangstaktperiode des Frequenzteilers. Es können natürlich auch mehrere Taktperioden gemessen werden und dann die durchschnittliche Zeit berechnet werden. Die Ausgangsfrequenz des Ringoszillators ergibt sich dann, indem man die Frequenz der blinkenden LED mit dem Teilerwert multipliziert. Aus der so berechneten Frequenz lässt sich die Periodendauer des Ringoszillators berechnen. Diese Zeit entspricht der Zeit, die das Signal braucht, um einmal durch alle Gatter zu laufen. Teilt man diese Zeit dann durch die Anzahl der verwendeten Gatter, erhält man grob die Durchlaufzeit eines einzelnen Gatters, wenn man die Verzögerungszeiten zwischen den Gattern vernachlässigt.
Ergebnis ungefähr: 1ns

5 Vivado Clock Manager

VHDL_main:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_unsigned.all;
use IEEE.numeric_std.all;

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

architecture Behavioral of main is

signal temp : std_logic_vector(15 downto 0) :="1111111111111111";
signal teiler : std_logic_vector(28 downto 0) :="10011000100101101000000000000";
signal schnelluhr,resets,locks : std_logic :='0';

component clk_wiz_0 is
port
(clk_out1 : out std_logic;
reset : in std_logic;
locked : out std_logic;
clk_in1 : in std_logic);
end component;

begin

UHR1: clk_wiz_0 port map(clk_out1=>schnelluhr,reset=>resets,locked=>locks,clk_in1=>uhr);

process(schnelluhr)
begin
if(rising_edge(schnelluhr)) then
if(teiler="00000000000000000000000000000") then
teiler<="10011000100101101000000000000";
temp<=temp+1;
else
teiler<=teiler-1;
end if;
end if;
end process;

ld<=temp;
end Behavioral;
```