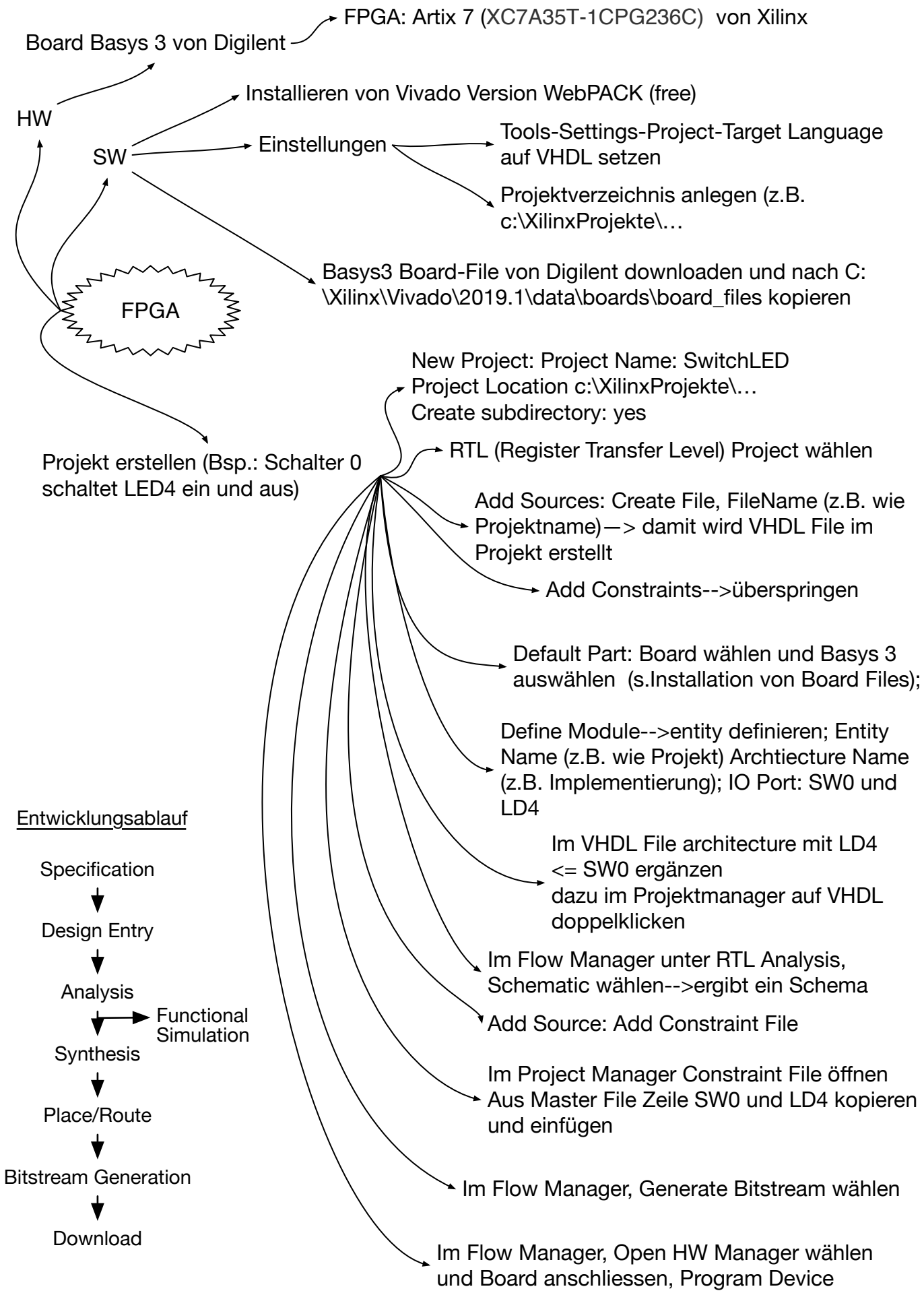
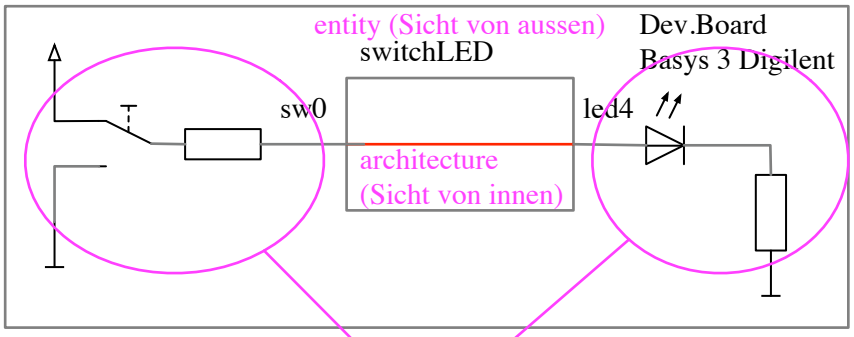

Board Basys 3 von Digilent
FPGA: Artix 7 (XC7A35T-1CPG236C) von Xilinx
HW
SW
Installieren von Vivado Version WebPACK (free)
Einstellungen
Tools-Settings-Project-Target Language auf VHDL setzen
Projektverzeichnis anlegen (z.B. c:\XilinxProjekte\…
Basys3 Board-File von Digilent downloaden und nach C:\Xilinx\Vivado\2019.1\data\boards\board_files kopieren
FPGA
Projekt erstellen (Bsp.: Schalter 0 schaltet LED4 ein und aus)
New Project: Project Name: SwitchLED
Project Location c:\XilinxProjekte\…
Create subdirectory: yes
RTL (Register Transfer Level) Project wählen
Add Sources: Create File, FileName (z.B. wie Projektname)—> damit wird VHDL File im Projekt erstellt
Add Constraints-->überspringen
Default Part: Board wählen und Basys 3 auswählen (s.Installation von Board Files);
Define Module-->entity definieren; Entity Name (z.B. wie Projekt) Archtiecture Name (z.B. Implementierung); IO Port: SW0 und LD4
Im VHDL File architecture mit LD4 <= SW0 ergänzen
dazu im Projektmanager auf VHDL doppelklicken
Im Flow Manager unter RTL Analysis, Schematic wählen-->ergibt ein Schema
Add Source: Add Constraint File
Im Project Manager Constraint File öffnen
Aus Master File Zeile SW0 und LD4 kopieren und einfügen
Im Flow Manager, Generate Bitstream wählen
Im Flow Manager, Open HW Manager wählen und Board anschliessen, Program Device
Entwicklungsablauf
Specification
Design Entry
Analysis
Functional Simulation
Synthesis
Place/Route
Bitstream Generation
Download

1. Specification	New Project	Create Project Name and Location RTL Project (Register Transfer Level) Add Sources (VHDL und Constraints) Create File (select VHDL) Next Add Constraints File (from Digilent Basys3BoardMasterXDCFile.xdc) Default Part —> Board (Refresh to download latest Driver) Basys 3 Finish Define Module I/O Names and directions (in/out)
2. Design entry in Vivado (VHDL)	VHDL	Project Manager Sources switchLED.vhd —> entity kontrollieren —> architecture definieren
	Constraints	Project Manager Sources constr —> Basys3 File öffnen —>Kommentierung # bei entsprechenden Zeilen entfernen und Namen gemäss VHDL anpassen; restliche Zeilen löschen —>Source File Properties Type XDC wählen
3. Simulation		Run Simulation—> Result in Timediagram
4. RTL Analysis		Ergibt Blockschema
4. Synthesis		Ergibt Implementierung auf Chip und detailliertes Schema
5. Implementation		Routing
6. Generate Bitstream		Hex-File wird erzeugt
7. Download		Open HW-Manager—>Program Device —> Pfad zum bit-File angeben

Beispiel switchLED



switchLED.xlc

Constraints (Begrenzungen) definieren die HW auf dem Board

```
1 set_property PACKAGE_PIN V17 [get_ports {sw0}]
2 set_property IOSTANDARD LVCMOS33 [get_ports {sw0}]
3 set_property PACKAGE_PIN W18 [get_ports {led4}]
4 set_property IOSTANDARD LVCMOS33 [get_ports {led4}]
```

switchLED.vhd

Libraries

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

entity Sicht von aussen

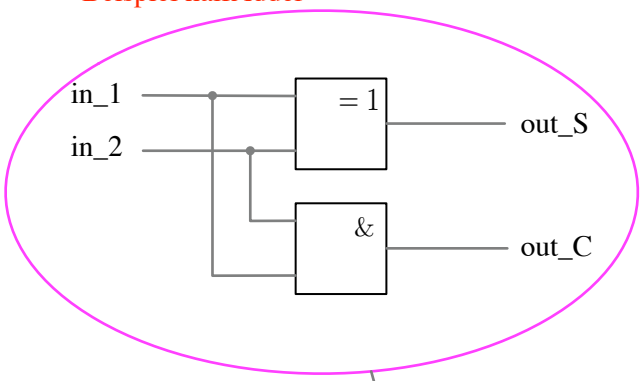
```
4 entity switchLED is
5     Port ( sw0: in STD_LOGIC;
6            led4: out STD_LOGIC);
7 end switchLED;
```

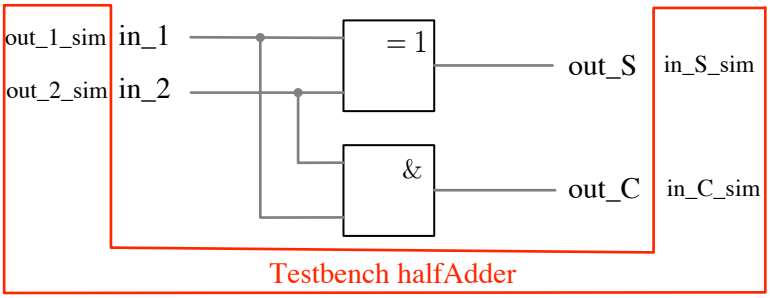
architecture Sicht von innen

```
1 architecture Behavioral of switchLED is

3 begin
4 led4 <= sw0;
5 end Behavioral;
```

Beispiel halfAdder






```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity halfAdder is
port
(
in_1 : in std_logic;
in_2 : in std_logic;
out_S : out std_logic;
out_C : out std_logic
);
end halfAdder;

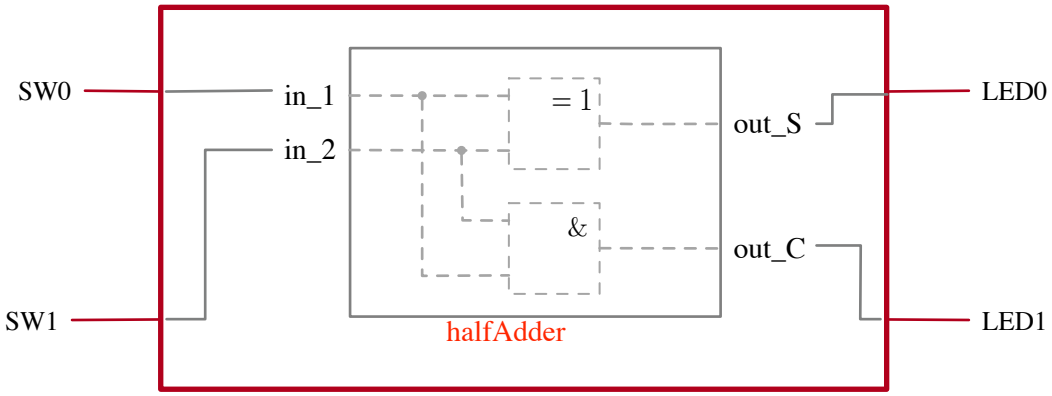
architecture Implementierung of halfAdder is
begin
out_S <= in_1 xor in_2;
out_C <= in_1 and in_2;
end Implementierung;
```

```vhdl
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity halfAdderTB is
5  --leer
6  end halfAdderTB;
7
8  architecture Simulation of halfAdderTB is
9  component halfAdder
10         port
11         (
12         in_1 : in std_logic;
13         in_2 : in std_logic;
14         out_S : out std_logic;
15         out_C : out std_logic
16         );
17 end component;
18
19 signal out_1_sim : std_logic;
20 signal out_2_sim : std_logic;
21 signal in_S_sim : std_logic;
22 signal in_C_sim : std_logic;
23
24 begin
25         dut : halfAdder
26         port map
27                 (
28         in_1 => out_1_sim,
29         in_2 => out_2_sim,
30         out_S => in_S_sim,
31         out_C => in_C_sim
32                 );
33
34                 process
35                 begin
36                 out_1_sim <= '0';
37                 out_2_sim <= '0';
38
39                 wait for 50 ns;
40                 out_1_sim <= '0';
41                 out_2_sim <= '1';
42
43                 wait for 50 ns;
44                 out_1_sim <= '1';
45                 out_2_sim <= '0';
46
47                 wait for 50 ns;
48                 out_1_sim <= '1';
49                 out_2_sim <= '1';
50
51                 wait;
52                 end process;
53 end Simulation;
```

Zum Testen der Schaltung auf der HW
Constraintfile als Source erstellen

```
set_property PACKAGE_PIN V17 [get_ports {in_1}]
set_property IOSTANDARD LVCMOS33 [get_ports
   {in_1}]
set_property PACKAGE_PIN V16 [get_ports {in_2}]
set_property IOSTANDARD LVCMOS33 [get_ports
   {in_2}]
set_property PACKAGE_PIN U16 [get_ports {out_s}]
set_property IOSTANDARD LVCMOS33 [get_ports
   {out_s}]
set_property PACKAGE_PIN E19 [get_ports {out_c}]
set_property IOSTANDARD LVCMOS33 [get_ports
   {out_c}]
```

Beispiel halfAdderWrap



Neues Projekt halfAdderWrap erstellen (Sources: halfAdderWrap und contraintsWrap)

VHDL Code für Schaltung (rot) erstellen. Mit Component halfAdder inkludieren.

Add Source: Datei halfAdder.vhd aus halfAdder-Projekt wählen.

Achtung: File-Hierarchie beachten. Top muss roter Rahmen sein und nicht halfAdder.vhd (Rechtsklick dann Top wählen)

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

entity halfAdderWrap is
    Port ( SW0 : in STD_LOGIC;
           SW1 : in STD_LOGIC;
           LED0 : out STD_LOGIC;
           LED1 : out STD_LOGIC);
end halfAdderWrap;

architecture Implementierung of halfAdderWrap is

component halfAdder is
    Port ( in_1 : in STD_LOGIC;
           in_2 : in STD_LOGIC;
           out_s : out STD_LOGIC;
           out_c : out STD_LOGIC);
end component;

begin
halfAdderWrap: halfAdder
port map(
in_1=>SW0,
in_2=>SW1,
out_S=>LED0,
out_C=>LED1
);
end Implementierung;
```

```
library ieee;
use ieee.std_logic_1164.all;

entity halfAdder is
port
(
in_1 : in std_logic;
in_2 : in std_logic;
out_S : out std_logic;
out_C : out std_logic
);
end halfAdder;

architecture Implementierung of halfAdder is
begin
out_S <= in_1 xor in_2;
out_C <= in_1 and in_2;
end Implementierung;
```

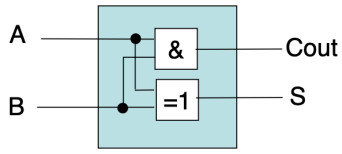
Constraints:

```
set_property PACKAGE_PIN V17 [get_ports {SW0}]
set_property IOSTANDARD LVCMOS33 [get_ports {SW0}]
set_property PACKAGE_PIN V16 [get_ports {SW1}]
set_property IOSTANDARD LVCMOS33 [get_ports {SW1}]
set_property PACKAGE_PIN U16 [get_ports {LED0}]
set_property IOSTANDARD LVCMOS33 [get_ports
   {LED0}]
set_property PACKAGE_PIN E19 [get_ports {LED1}]
set_property IOSTANDARD LVCMOS33 [get_ports
   {LED1}]
```

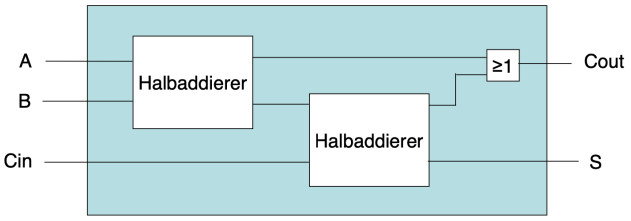
```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity HALBADDIERER is
  port ( A : in std_logic;
         B : in std_logic;
         S : out std_logic;
         Cout : out std_logic);
end HALBADDIERER;

architecture Behavioral of HALBADDIERER is
begin
  S<=(not A and B) or (A and not B);
  Cout<=(A and B);
end Behavioral;
```



Hierarchie: Ein Volladdierer




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

entity VOLLADDIERER is
  port ( A : in std_logic;
         B : in std_logic;
         Cin : in std_logic;
         S : out std_logic;
         Cout : out std_logic);
end VOLLADDIERER;

architecture Structural of VOLLADDIERER is
component HALBADDIERER
  port( A: in std_logic;
        B: in std_logic;
        S: out std_logic;
        Cout : out std_logic);
end component;

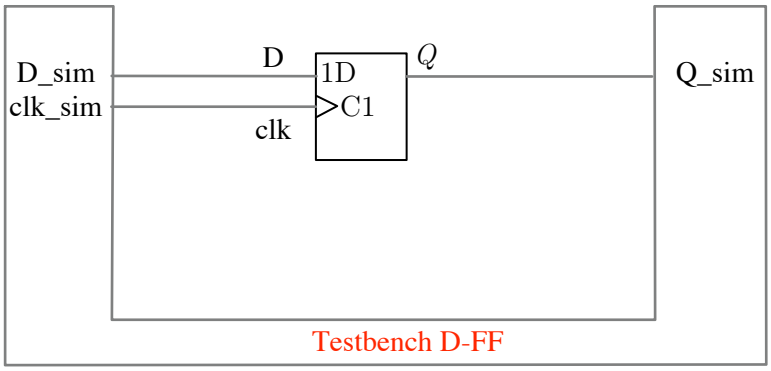
signal S1HA : std_logic;
signal C1HA : std_logic;
signal S2HA : std_logic;
signal C2HA : std_logic;

begin
  HA1: HALBADDIERER PORT MAP(
                        A => A,
                        B => B,
                        S => S1HA,
                        Cout => C1HA
);

HA2: HALBADDIERER PORT MAP(
                        A => S1HA,
                        B => Cin,
                        S => S2HA,
                        Cout => C2HA
);

Cout <= (C1HA or C2HA);
S <= S2HA;
end;
```

Beispiel D-FF




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

entity D_FF is
    Port (
            clk : in STD_LOGIC;
            D : in STD_LOGIC;
            Q : out STD_LOGIC
            );

end D_FF;

architecture Implementierung of D_FF is
begin

     process (clk)
     begin

       if rising_edge(clk) then
       Q <= D;
       end if;

end process;


end Implementierung;
```

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

entity D_FF_TB is
end D_FF_TB;

architecture Implementierung of D_FF_TB is
component D_FF
     port
     (
     clk : in STD_LOGIC;
     D : in STD_LOGIC;
     Q : out STD_LOGIC
     );
end component;

signal clk_sim : STD_LOGIC;
signal D_sim : STD_LOGIC;
signal Q_sim : STD_LOGIC;

constant clk_per : time := 400ns;

begin
     dut : D_FF
     port map
     (
     clk => clk_sim,
     D => D_sim,
     Q => Q_sim
     );

process
begin

     D_sim <= '0';
     clk_sim <= '0';

     wait for clk_per/2;
     clk_sim <= '1';

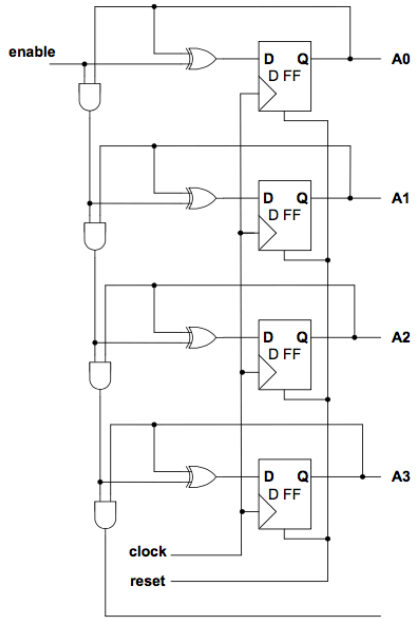
     wait for clk_per/2;
     clk_sim <= '0';
     D_sim <= '1';

     wait for clk_per/2;
     clk_sim <= '1';

     wait for clk_per/2;
     clk_sim <= '0';

     wait;
   end process;
end Implementierung;
```

Beispiel counter mit D-FF



https://stackoverflow.com/questions/53382974/2-bit-up-4-bit-counter-with-d-flip-flops-vhdl

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY dff IS
    PORT
    (
       clk : IN  STD_LOGIC;
       rst : IN  STD_LOGIC;
       din : IN  STD_LOGIC;
      qout : OUT STD_LOGIC
    );
END dff;

ARCHITECTURE behavioral OF dff IS

BEGIN

    PROCESS (clk, rst)
    BEGIN
        IF (rst = '0') THEN
            qout<='0';
        ELSIF(clk = '1' AND clk'EVENT) THEN
            qout <= din;
        END IF;
    END PROCESS;

END behavioral;
```

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY upcount4 IS
   PORT
   (
      clk    : IN  STD_LOGIC;
      rst : IN  STD_LOGIC;
      en     : IN  STD_LOGIC;
    qout  : OUT STD_LOGIC_VECTOR(3 DOWNTO 0)
   );
END upcount4;

ARCHITECTURE rt_level OF upcount4 IS
   COMPONENT dff IS
      PORT
      (
         clk    : IN  STD_LOGIC;
         rst : IN  STD_LOGIC;
         din : IN  STD_LOGIC;
         qout   : OUT STD_LOGIC
      );
   END COMPONENT;

   SIGNAL dffs_out : STD_LOGIC_VECTOR(3 DOWNTO 0);
   SIGNAL dffs_in : STD_LOGIC_VECTOR(3 DOWNTO 0);
   SIGNAL ands_out : STD_LOGIC_VECTOR(2 DOWNTO 0);

BEGIN
   ands_out(0) <= en AND dffs_out(0);
   ands_out(1) <= ands_out(0) AND dffs_out(1);
   ands_out(2) <= ands_out(1) AND dffs_out(2);
   dffs_in(0) <= en XOR dffs_out(0);
   dffs_in(1) <= ands_out(0) XOR dffs_out(1);
   dffs_in(2) <= ands_out(1) XOR dffs_out(2);
   dffs_in(3) <= ands_out(2) XOR dffs_out(3);
   qout <= dffs_out;

   dff_0:dff
   PORT MAP
   (
      clk  => clk,
      rst  => rst,
      din  => dffs_in(0),
      qout => dffs_out(0)
   );
   dff_1:dff
   PORT MAP
   (
      clk => clk,
      rst => rst,
      din => dffs_in(1),
      qout=> dffs_out(1)
   );
   dff_2:dff
   PORT MAP
   (
      clk => clk,
      rst => rst,
      din => dffs_in(2),
      qout=> dffs_out(2)
   );
   dff_3:dff
   PORT MAP
   (
      clk => clk,
      rst => rst,
      din => dffs_in(3),
      qout=> dffs_out(3)
   );
END rt_level;
```

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY counter_tb IS
END counter_tb;

ARCHITECTURE behavior OF counter_tb IS

    COMPONENT upcount4
    PORT(
         clk : IN  std_logic;
         rst : IN  std_logic;
         en : IN  std_logic;
         qout : OUT  std_logic_vector(3 downto 0)
        );
    END COMPONENT;

   signal clk : std_logic := '0';
   signal rst : std_logic := '0';
   signal en : std_logic := '0';

   signal qout : std_logic_vector(3 downto 0);

BEGIN

   uut: upcount4 PORT MAP(
        clk => clk,
         rst => rst,
         en => en,
         qout => qout);

    clk <= NOT clk AFTER 5 NS;

    rst <= '0',
           '1' AFTER 30 NS,
           '0' AFTER 260 NS,
           '1' AFTER 290 NS;

    en  <= '0',
           '1' AFTER 40 NS,
           '0' AFTER 215 NS,
           '1' AFTER 295 NS;

END;
```

Beispiel Counter

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;

5  entity binCntr is
6    port (
7          clk : in std_logic;
8          counter : out std_logic_vector (1 downto 0)
9          );
10 end binCntr;

12 architecture impl of binCntr is
13   signal cntr : unsigned (1 downto 0) := "00";
14   begin
15     process (clk)
16         begin
17           if rising_edge (clk) then
18               cntr <= cntr + 1;
19           end if;
20    end process;
21    counter <= std_logic_vector (cntr);
22 end impl;
```

```
1  library ieee;
2  use ieee.std_logic_1164.all;

4  entity binCntrTB is
5  end binCntrTB;

7  architecture sim of binCntrTB is
8    component binCntr is
9    port
10       (
11       clk : in std_logic;
12       counter : out std_logic_vector(1 downto 0)
13       );
14   end component;

16 signal clk_sim : std_logic;
17 signal counter_sim : std_logic_vector (1 downto 0);

19 constant clk_period : time := 1ms;

21 begin
22 dut : binCntr

24 port map
25     (
26     clk => clk_sim,
27     counter => counter_sim
28     );

30 clk_gen : process
31 begin
32   clk_sim <= '1';
33   wait for clk_period/2;
34   clk_sim <= '0';
35   wait for clk_period/2;
36 end process clk_gen;

38 sim_proc : process
39 begin
40   wait for 10ms;
41   wait;
42 end process sim_proc;

44 end sim;
```

Simulationssettings auf 10ms stellen.
Rechtsklick auf Simulation, dann
Simulation settings…

Beispiel Blink

Für eine Blinkfrequenz von 1 Hz muss der Counter vom vorherigen Beispiel um n-Bit erweitert werden.

$$f_{LED} = \frac{f_{clk}}{2^n}$$

$$n = log_2 \frac{f_{clk}}{f_{LED}}$$

$$n = log_2 \frac{100MHz}{1Hz} = 26.5$$

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity blink is
5   port
6    (
7    CLK100MHZ : in std_logic;
8    led_blink: out std_logic
9    );
10 end blink;
11
12 architecture impl of blink is
13 component binCntr is
14   port
15   (
16    clk: in std_logic;
17    counter_out :
18       out std_logic_vector(26 downto 0)
19   );
20 end component;
21
22 begin
23   myCntr: binCntr
24   port map
25   (
26     clk => CLK100MHZ,
27     counter (26) => led_blink,
28     counter (25 downto 0) => open
29   );
30 end impl;
```

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity binCntr is
6    port (
7          clk : in std_logic;
8          counter : out std_logic_vector (26 downto 0)
9          );
10 end binCntr;
11
12 architecture impl of binCntr is
13   signal cntr : unsigned (26 downto 0)
14                 := (others => '0');
15   begin
16     process (clk)
17         begin
18           if rising_edge (clk) then
19               cntr <= cntr + 1;
20           end if;
21    end process;
22    counter <= std_logic_vector (cntr);
23 end impl;
```

```
1  # Clock signal
2  set_property -dict { PACKAGE_PIN W5   IOSTANDARD LVCMOS33 } [get_ports
       {CLK100MHZ}]
3  create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get
       ports {CLK100MHZ}]
4
5  # LEDs
6  set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS33 } [get_ports
       {led_blink}]
```