Grundlagen der Digitaltechnik
Laborübung 2
VHDL und FPGAs

Prof. G. Kemnitz, Dr. C. Giesemann

Institut für Informatik, Technische Universität Clausthal

13. April 2022

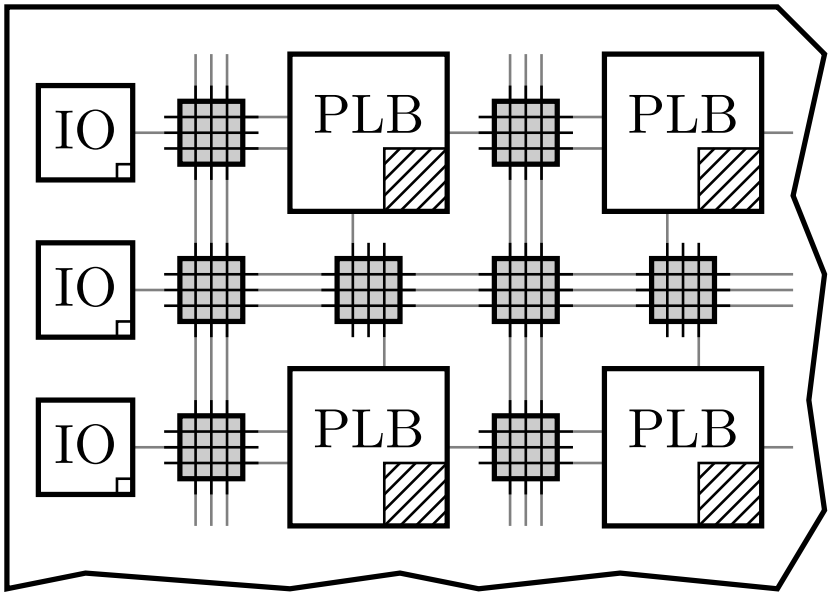
Hardware-Programmierung

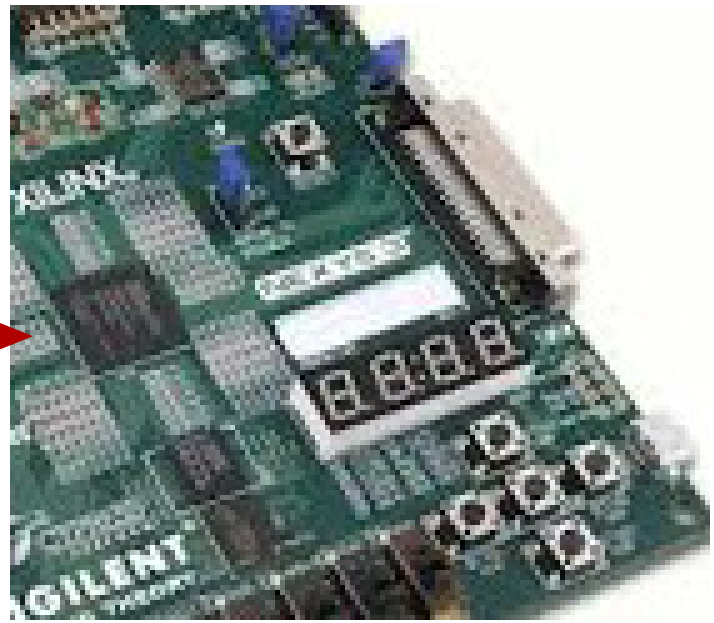
Heutiger Stand der Technik für den Entwurf digitaler Schaltungen ist die Beschreibung der Zielfunktion in einer Hardware-Beschreibungssprache. Simulation auf dem Rechner. Prototyp-Fertigung mit programmierbaren Logikschaltkreisen (FPGA **F**ield **P**rogrammable **G**ate **A**rray). FPGAs bestehen aus

- programmierbaren Logikblöcken,
- programmierbaren EA-Schaltungen,
- einem programmierbaren Verbindungsnetzwerk und
- optional weiteren konfigurierbaren Schaltungsblöcken, z.B. Blockspeichern, Multiplizierern, Taktversorgung und Prozessorkernen.

Programmierung ähnlich wie bei Mikrorechnern über eine serielle Programmierschnittstelle (JTAG, ISP,...). Programmierdauer wenige Minuten.

Prototyp-Plattform für die Laborübungen

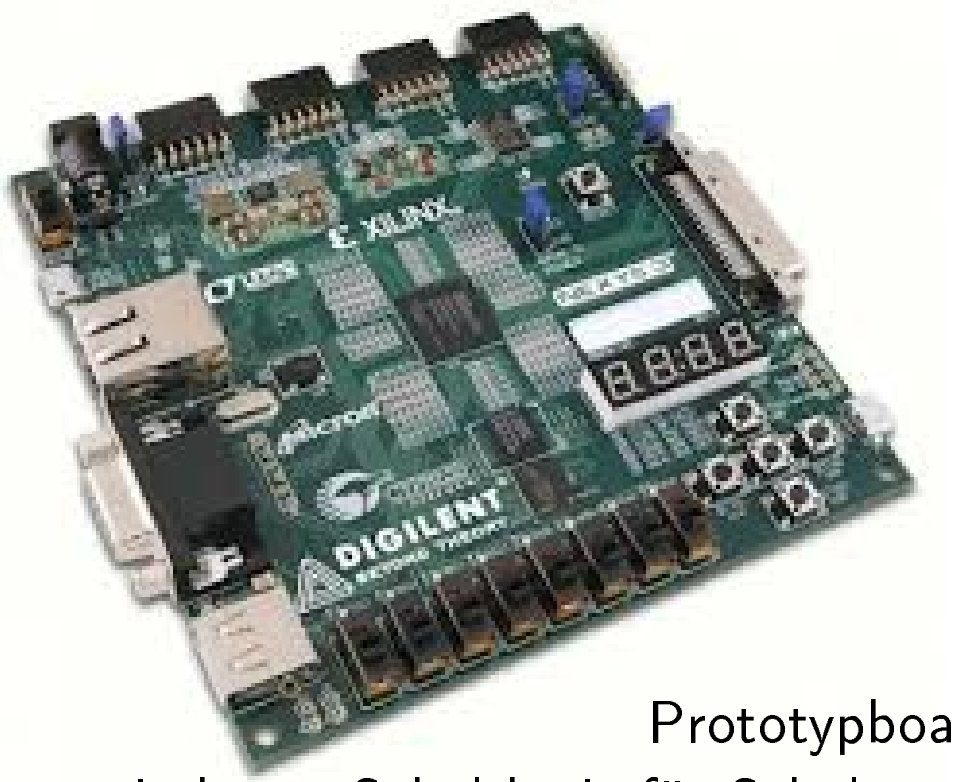




programmierbares Verbindungsnetzwerk

IO programmierbare Eingabe-Ausgabe-Schaltung

PLB programmierbarer Logikblock



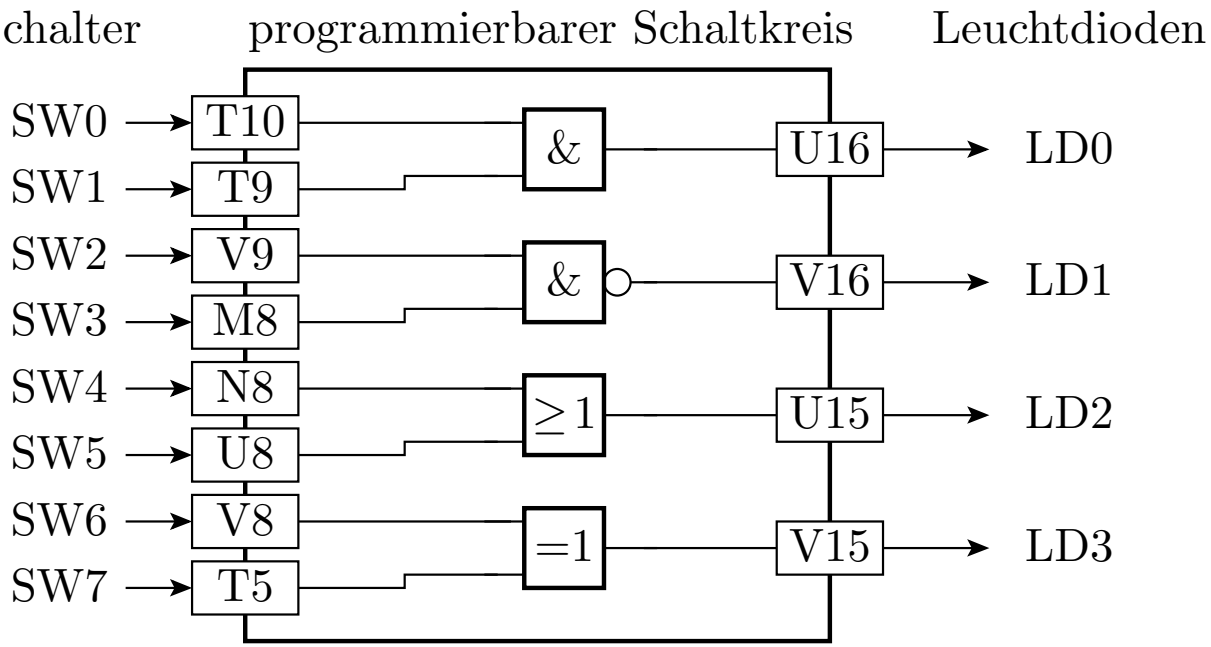
Das verwendete Prototypboard hat

- einen programmierbaren Schaltkreis für Schaltungen, die mit normalen Schaltkreisen bis zu 10^6 Gatter umfassen können,
- 100MHz Quarztakt, 8 Leuchtdioden, 4 7-Segment-Anzeigen,
- 8 Schalter, 5 Taster, 1MB SRAM,
- Anschlüsse für USB-Tastatur / -Mouse, USB-UART und VGA.

Ausreichend für alles, was im Studium vermittelbar ist.

Einfache Gatterschaltung

Einprogrammieren von Logikfunktionen



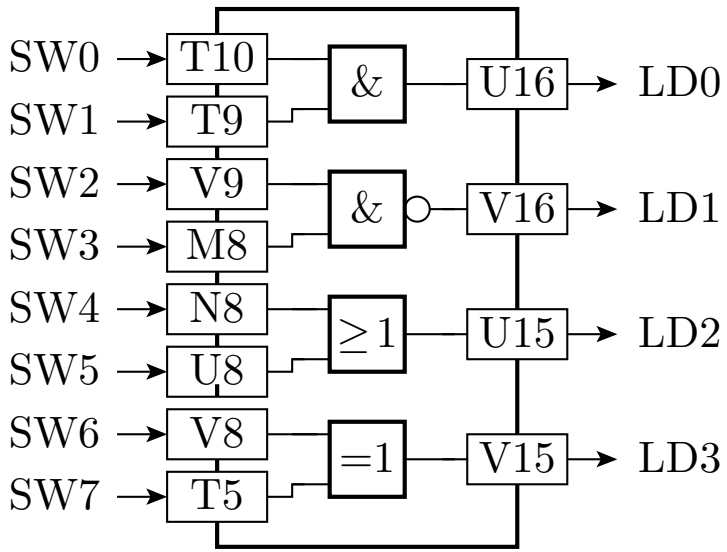
Im ersten Beispiel sollen vier Gatter so einprogrammiert werden, dass die Eingänge von Schaltern steuer- und die Ausgänge mit LEDs beobachtbar sind. Die Kästchen mit »T10« etc. sind die Bezeichner der Schaltkreisanschlüsse, an denen die Schalter und LEDs auf der Baugruppe angeschlossen sind.

Eine VHDL-Beschreibung besteht aus Schnittstellenbeschreibung:

```
entity Logikrechner is
 port(SW0, SW1, SW2, SW3, SW4, SW5, SW6, SW7:
                                  in std_logic;
      LD0, LD1, LD2, LD3: out std_logic);
end Logikrechner;
```

und einer Beschreibung der Realisierung

```
architecture Behavioral of Logikrechner is
begin
 LD0 <= SW0 and  SW1;
 LD1 <= SW2 nand SW3;
 LD2 <= SW4 or   SW5;
 LD3 <= SW6 xor  SW7;
end Behavioral;
```



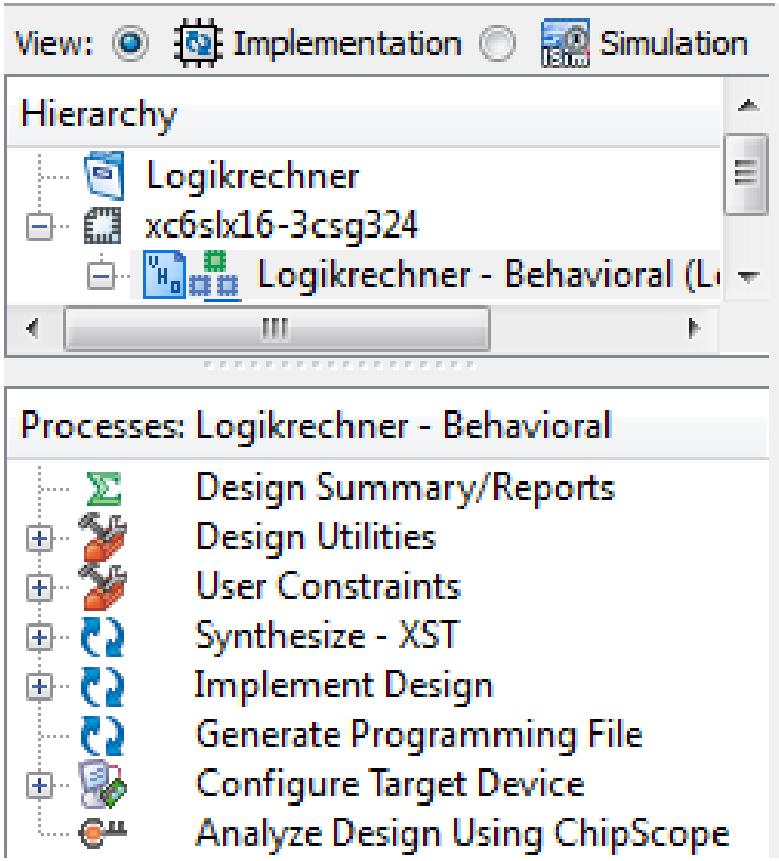
Die Schnittstelle definiert die Anschlüsse (Bezeichner, Richtung, Datentyp. Die Realisierung beschreibt die Schaltung.

Entwurfsvorbereitung

- Zip-Datei EDS_GU2.zip von der Web-Seite laden und entpacken.
- Mit dem Web-Browser diesen Foliensatz öffnen.
- »Xilinx ISE« öffnen:

  `Start ▷ All Programs ▷ Xilinx Design Tools ▷ ISE Design Suite 14.6 ▷ ISE Design Tools ▷ 64-bit-Project Navigator`
- Fenster »What's New ...« und »Tip of the Day« schließen.
- (Project Navigator) File ▷ Open Project ▷ <entpacktes Archiv> ▷ Logikrechner ▷ Logikrechner.xise auswählen.
- (Hierarchie-Fenster) .../Logikrechner.vhd auswählen, mit Doppelklick öffnen. Schaltungsbeschreibung kontrollieren.

Editier- und Werkzeugansicht

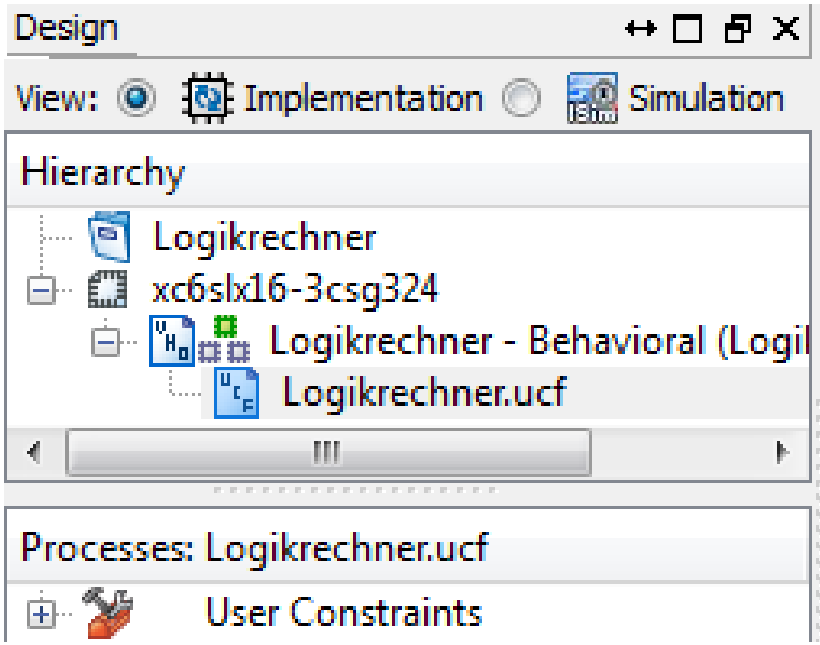



```
4  ----------------------------------
5  library IEEE;
6  use IEEE.STD_LOGIC_1164.ALL;
7
8  entity Logikrechner is
9   Port ( SW0, SW1, SW2, SW3, SW4, SW5,
10         SW6, SW7 : in  STD_LOGIC;
11   LD0, LD1, LD2, LD3 : out  STD_LOGIC);
12 end Logikrechner;
13
14 architecture Behav: of Logikrechner is
15 begin
16  LD0 <= SW0  and SW1;
17  LD1 <= SW2 nand SW3;
18  LD2 <= SW4   or SW5;
19  LD3 <= SW6  xor SW7;
20 end Behavioral;
```

Fenster unten links zeigt die für das Entwurfsobjekt »Logikrechner« verfügbaren Bearbeitungswerkzeuge.

Die Constraints-Datei

Doppelklick auf »Logikrechner.ucf«:



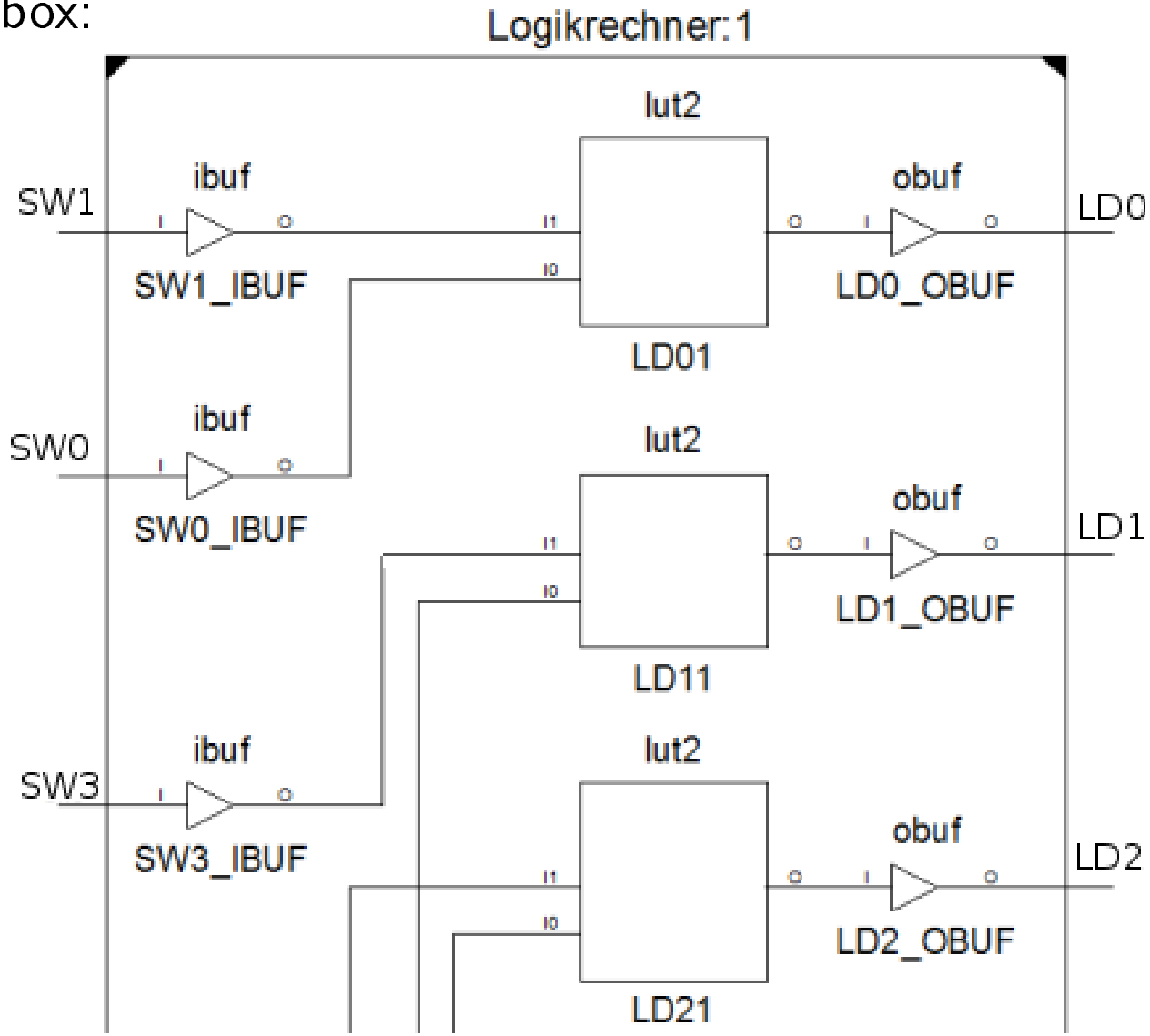

```
 1 NET "SW0" LOC = T10;
 2 NET "SW1" LOC = T9;
 3 NET "SW2" LOC = V9;
 4 NET "SW3" LOC = M8;
 5 NET "SW4" LOC = N8;
 6 NET "SW5" LOC = U8;
 7 NET "SW6" LOC = V8;
 8 NET "SW7" LOC = T5;
 9 NET "LD0" LOC = U16;
10 NET "LD1" LOC = V16;
11 NET "LD2" LOC = U15;
12 NET "LD3" LOC = V15;
```

Die Constraints-Datei enthält alle zusätzlichen Informationen zur Vorgabe der Zielfunktion, die nicht in der VHDL-Datei stehen: die Pin-Namen der Schaltungsanschlüssen (s.o.), Taktfrequenz, ...

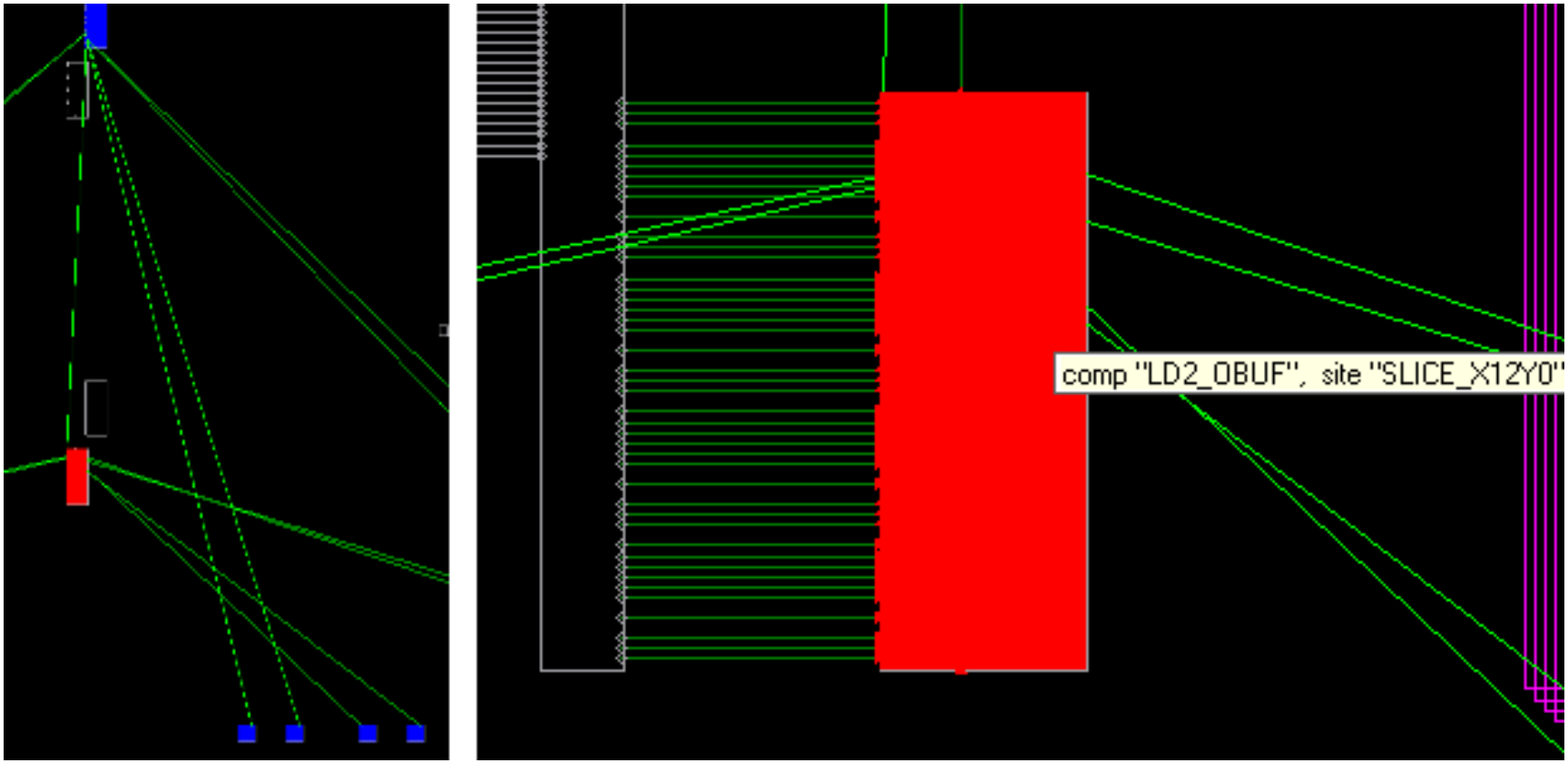
Übersetzen, Programmieren und Ausprobieren

- (Hierarchy-Fenster) .../Logikrechner.vhd auswählen. (Processes) mit Doppelklick auf »Generate Programming File« (Synthese, Platzierung, Verdrahtung, ...).
- »Configure Target Device ...« aufklappen
- (Processes) Doppelklick auf »Manage Configuration Project (iMPACT)«
- (iMPACT) Doppelklick auf »Boundary Scan«
- (Boundary Scan) Rechtsklick auf (Right click ...) ▷ »Initialize Chain«. »Auto Assign ...« Yes.
- Für »xc6slx16« »logikrechner.bit« im Verzeichnis »Logikrechner« auswählen; »Open«.
- »Attache SPI...« »No«. Nächstes Fenster »OK«.
- Rechtsklick auf den Chip »xc6slx16« ▷ »Program«.
- Ausprobieren: Schaltereinstellungen vornehmen und LED-Anzeige kontrollieren.

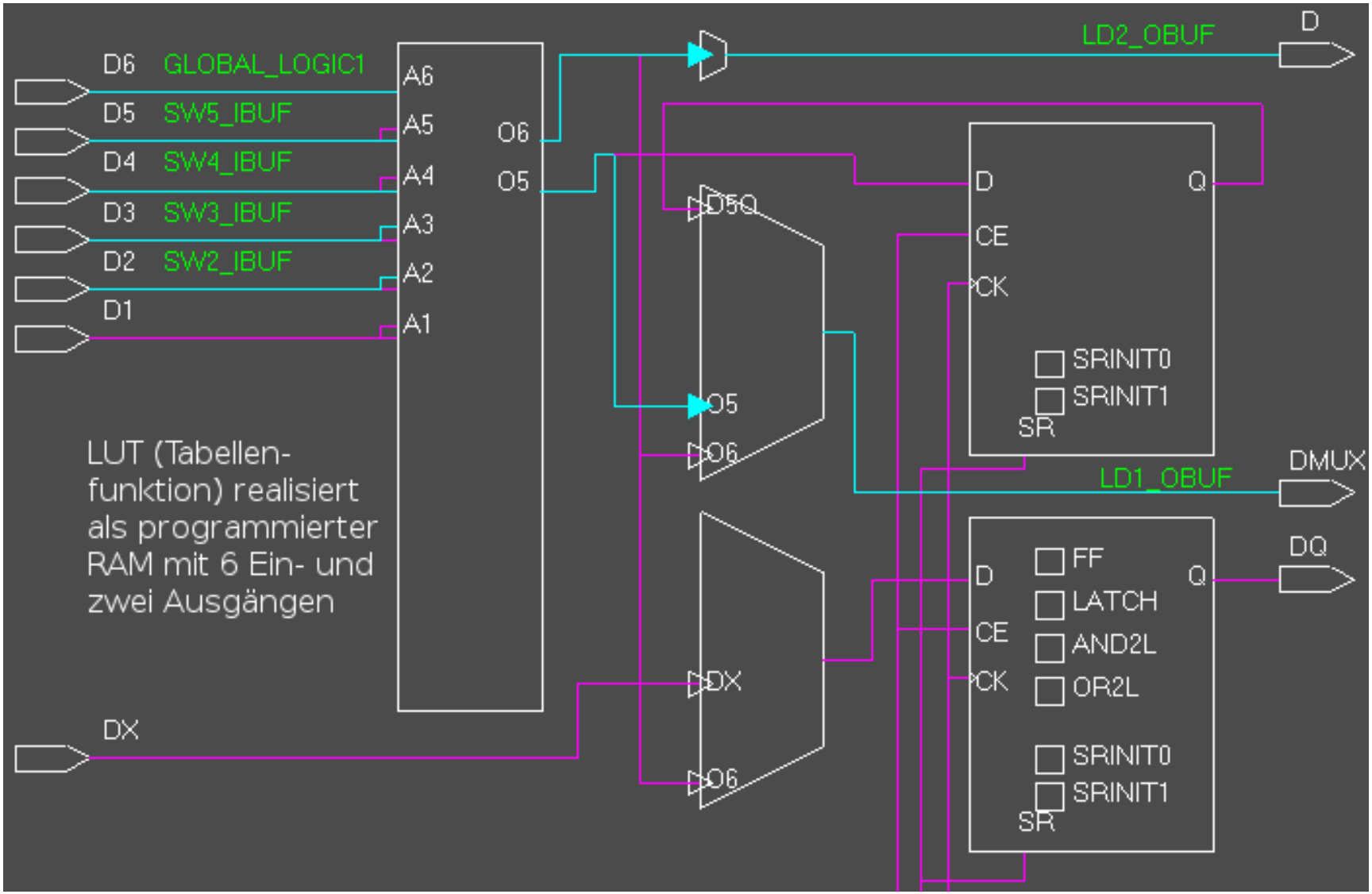
Das Syntheseergebnis: »Synthesize - XST ▷ View Technology Schematic«; Start with a schematic of a top-level block ▷ OK ▷ Doppelklick Blackbox:



Das Ergebnis der Platzierung und Verdrahtung: »Implement Design ▷ Map ▷ Manually Place & Route (FPGA Editor)«.



Schaltung in dem rot hervorgehobenen Slice:



Simulation

Vorbemerkungen zur Simulation[1]

Die Simulation benötigt ein Testobjekt. Das sei hier die nachfolgende Zählfunktion:

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

entity Increment4Bit is
    Port ( x : in  std_logic_vector (3 downto 0);
           y : out std_logic_vector (3 downto 0));
end Increment4Bit;

architecture Behavioral of Increment4Bit is
begin
    y <= std_logic_vector(unsigned(x) + 1);
end Behavioral;
```

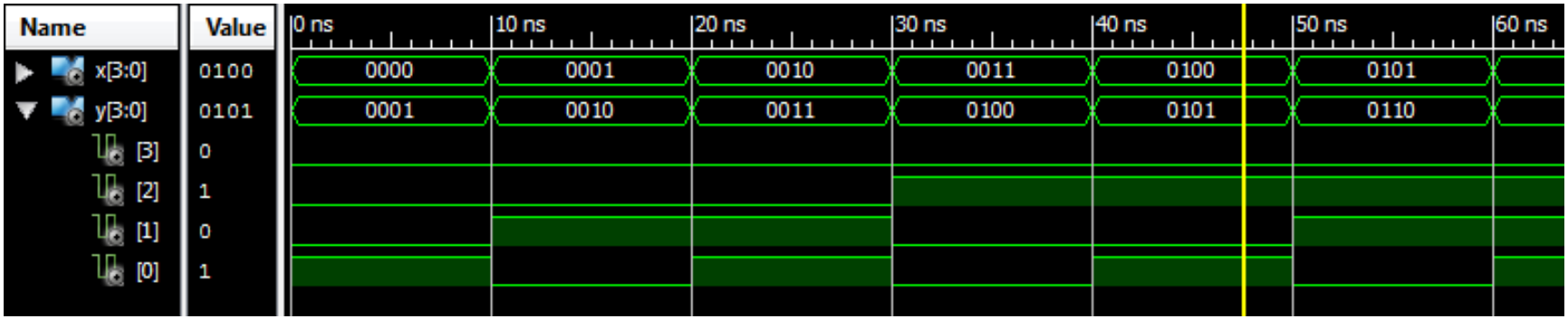
Zusätzlich wird ein Testrahmen zur Erzeugung der Eingabesignale benötigt. Das ist eine Entwurfseinheit mit dem Testobjekt als Teilschaltung und einem Prozess zur Erzeugung der Eingaben.

[1] Die schrittweise Beschreibung, was zu tun ist, folgt ab Folie 19.

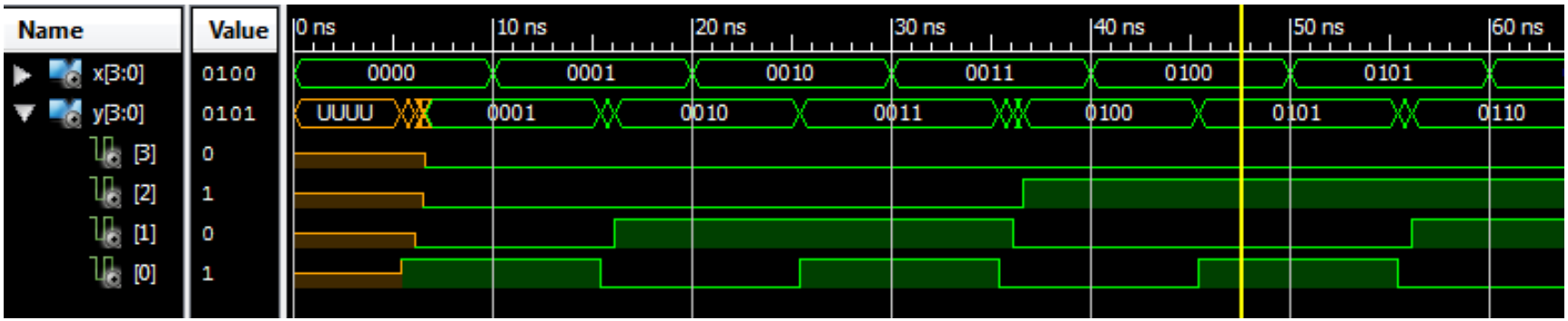
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY IncTB IS
END IncTB;

ARCHITECTURE behavior OF IncTB IS
   SIGNAL x : std_logic_vector(3 downto 0) := (others => '0');
   SIGNAL y : std_logic_vector(3 downto 0);
BEGIN
   -- Instantiate the Unit Under Test (UUT)
   uut: entity work.Increment4Bit PORT MAP (
         x => x,
         y => y
        );
   -- Stimulus process
   stim_proc: process
   begin
      for i in 1 to 20 loop
      wait for 10 ns;
      x <= std_logic_vector(unsigned(x) + 1);
      end loop;
      wait;
   end process;
END;
```

Simulationsarten

- Simulation des Verhaltens ohne Verzögerungen:



- Simulation der synthetisierten und verdrahteten Schaltung (mit Verzögerungen)



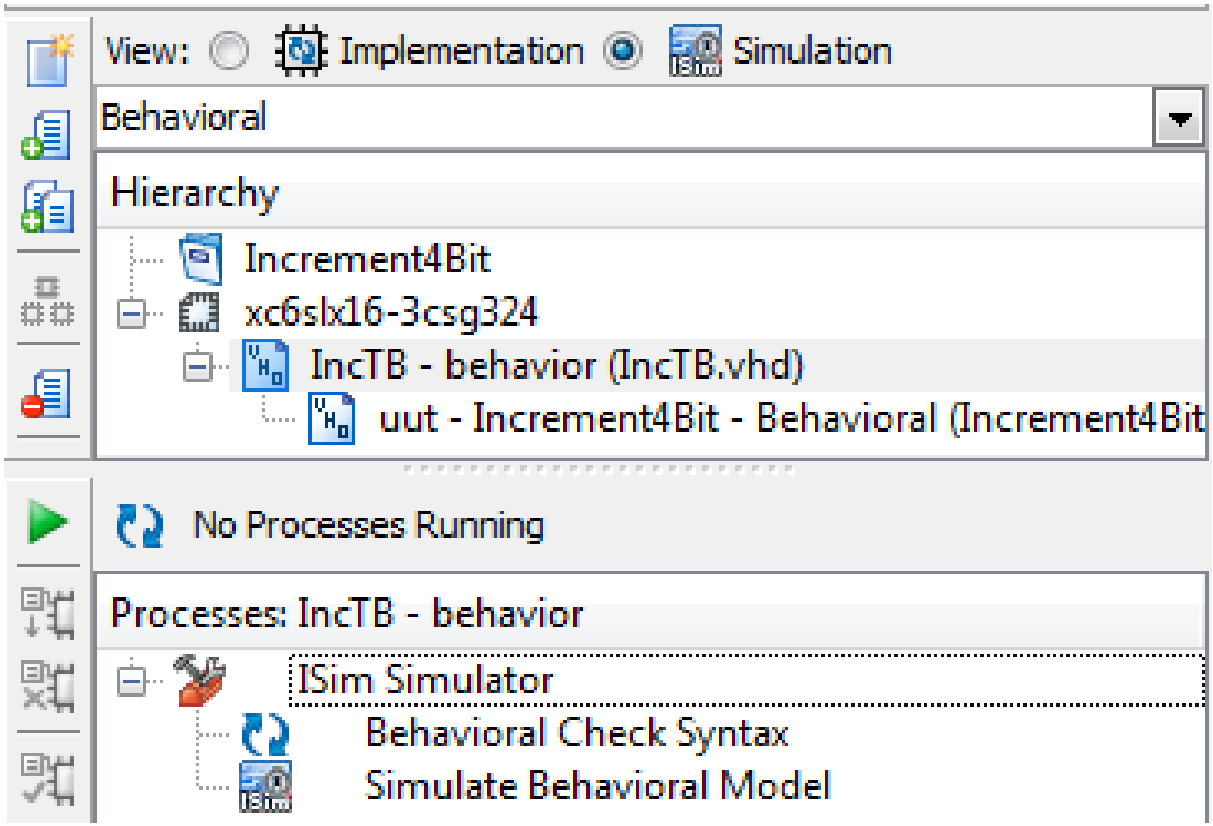
Arbeitsschritte

Anlegen eines neuen Projekts:

- (ISE) »File ▷ New Project«.
- Name »Increment4Bit« ▷ Next .
- Einstellungen bleiben wie im Bild gezeigt ▷ Next ▷ Finish:

Property Name	Value
Product Category	All
Family	Spartan6
Device	XC6SLX16
Package	CSG324
Speed	-3
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	VHDL

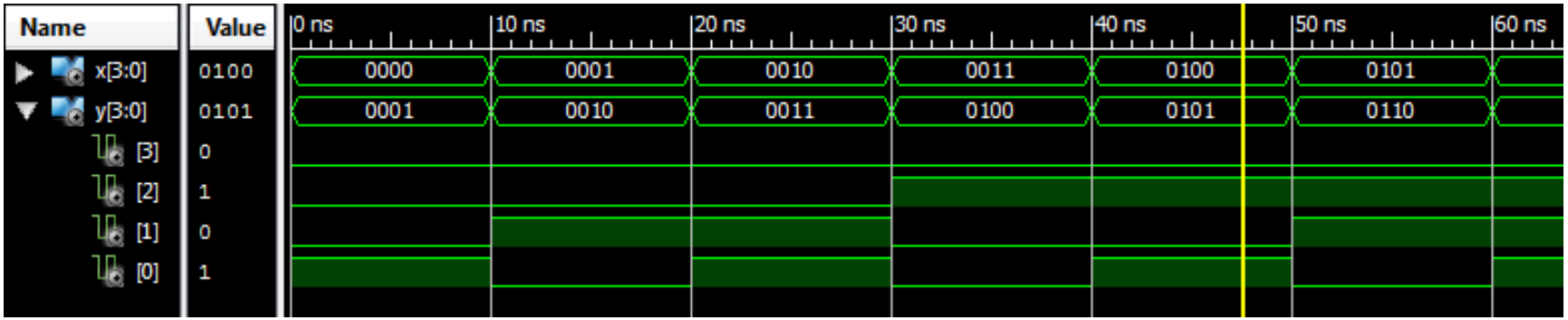
Simulation vorbereiten



Mit »Project ▷ New Source« ein »VHDL Module« »Increment4Bit.vhd« und eine »VHDL Test Bench« »IncTB.vhd« hinzufügen und die Beschreibungen von den Folien zuvor eintippen. Von Implementation auf Simulation umstellen. Nach der Eingabe für beide Dateien »Behavioral Check Syntax« und Fehlerbeseitigung.

Simulation ohne Verzögerung

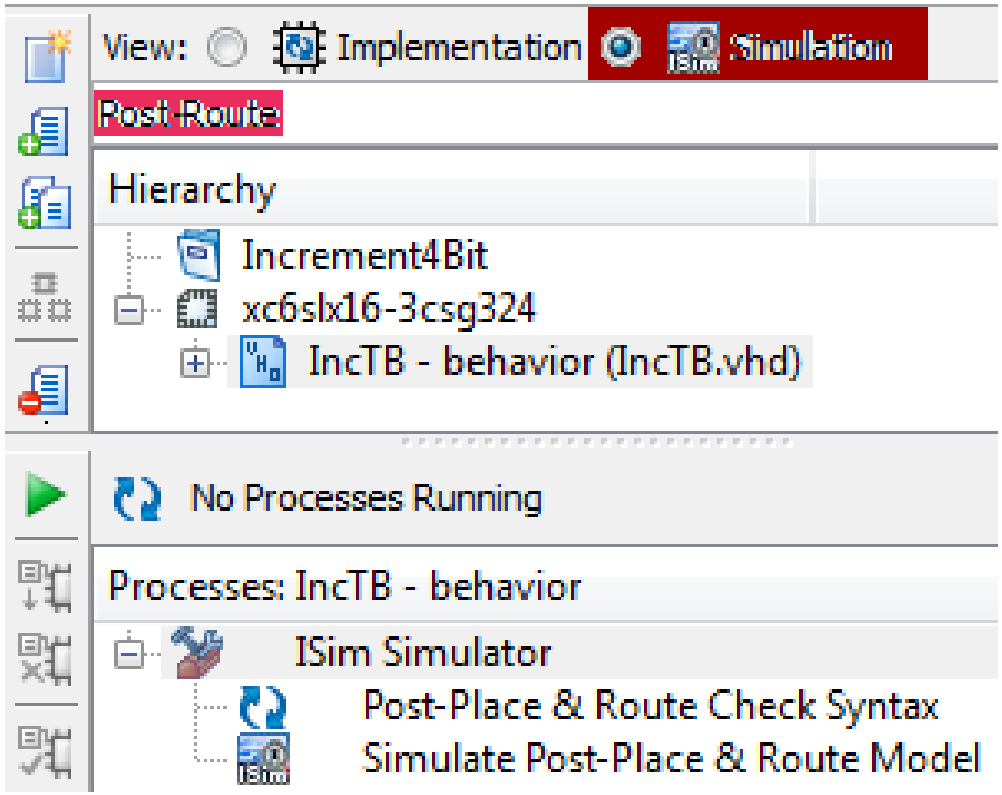
- »Simulate Behavioral Model«
- Darstellungsbereich mit dem Zoom richtig einstellen.



- Simulator schließen, bevor eine neue Simulation aus »ise« heraus gestartet wird.

Post-Place & Route Simulation

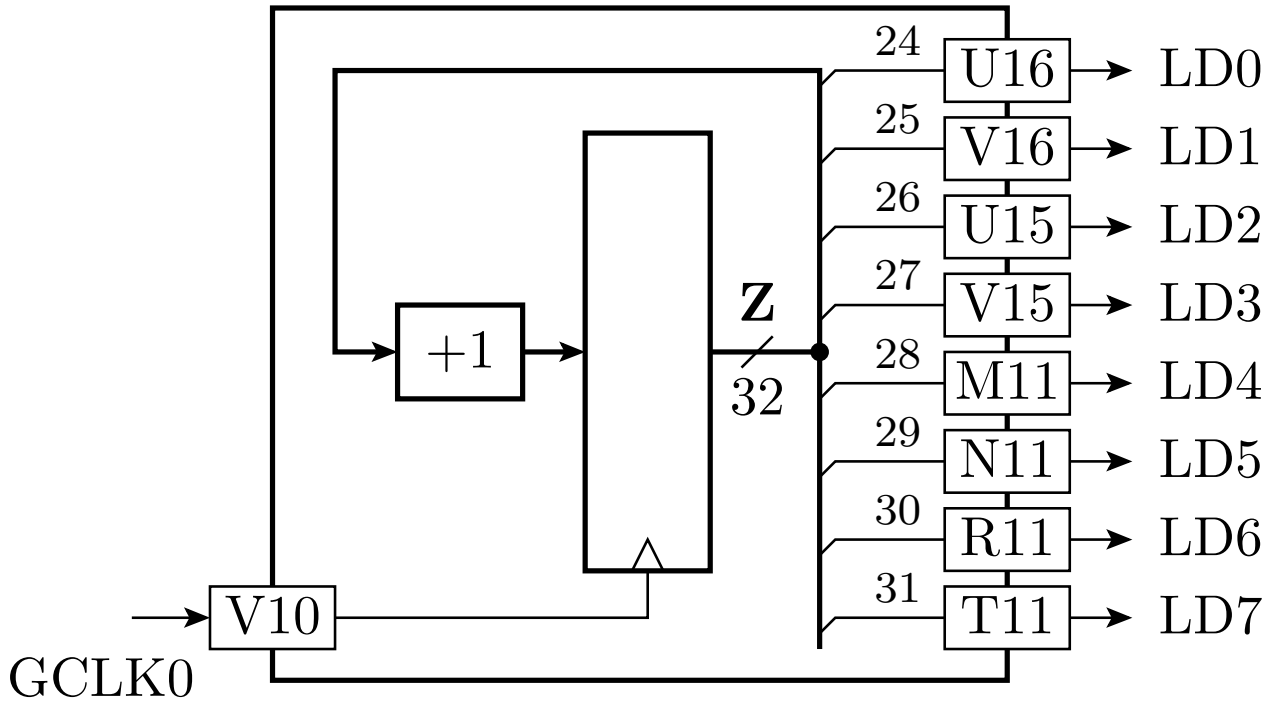
- Auf »Implementation« umstellen und »Implement Design« ausführen.
- Auf »Simulation« zurückstellen, Simulationsart auf »Post Route« umstellen.
- ... Syntax Check.
- »Simulate Post-Place & Route Model«.



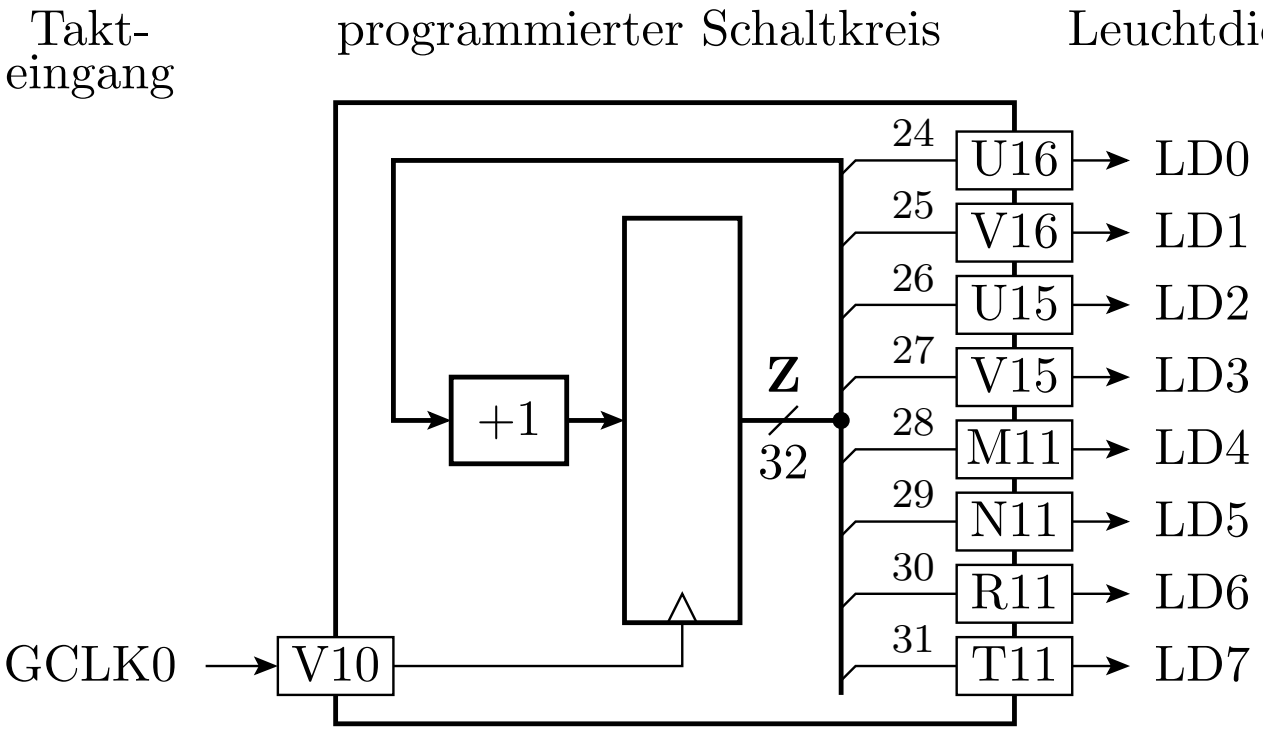
Taktteiler

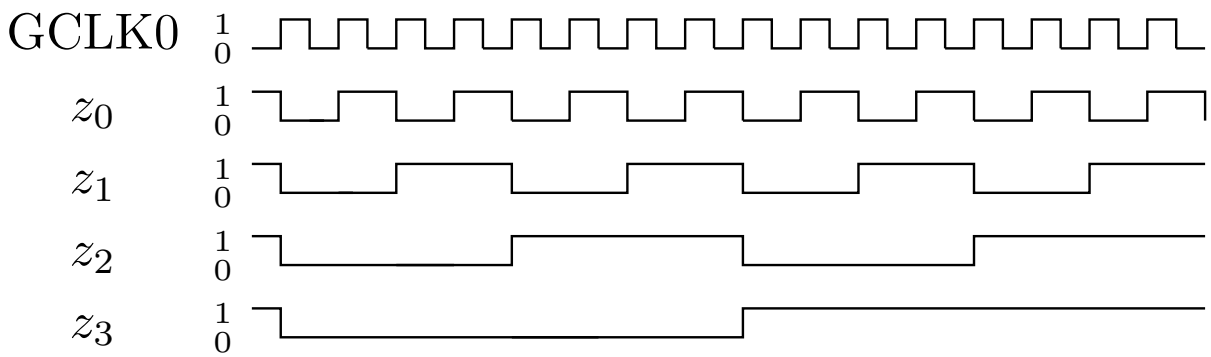
Taktteiler

- Teilen des 100MHz-Eingabetakts durch 2^{32}
- Ausgabe der höchstwertigen Zählerstellen auf Leuchtdioden

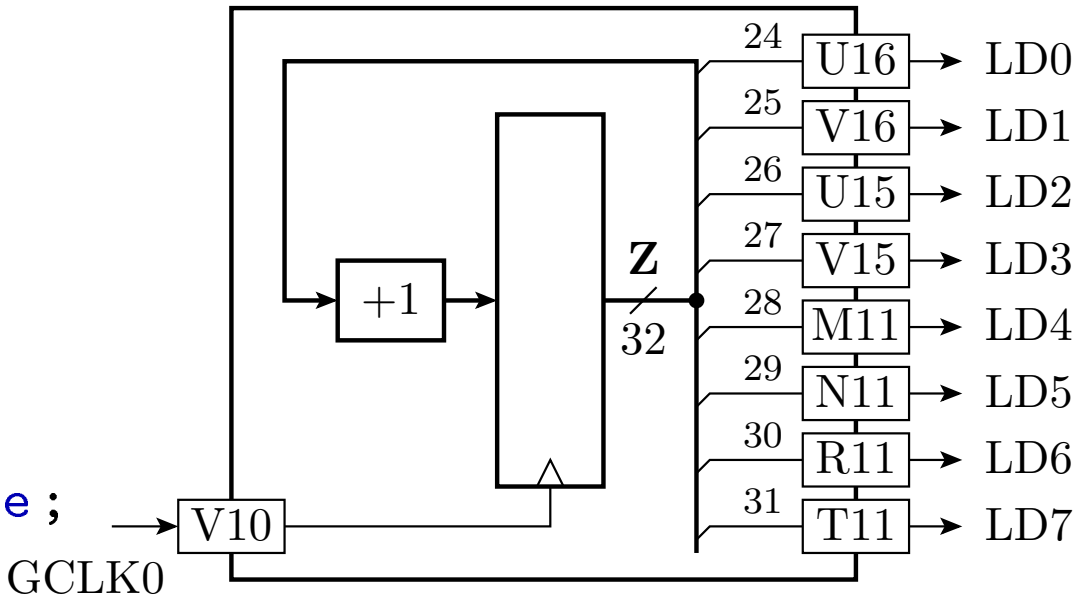


- $100/2^{24} \approx 6\,\text{Hz}$; $100/2^{25} \approx 3\,\text{Hz}$; ...


Takt-
eingang
programmierter Schaltkreis
Leuchtdioden
GCLK0
V10
+1
Z
32
24
25
26
27
28
29
30
31
U16
V16
U15
V15
M11
N11
R11
T11
LD0
LD1
LD2
LD3
LD4
LD5
LD6
LD7


GCLK0
z_0
z_1
z_2
z_3
1
0

```
entity Taktteiler is
 port(GCLK0: in std_logic;
      LD0, LD1, ..., LD7: out std_Logic);
end entity;
architecture test of Taktteiler is
 signal z: std_logic_vector(31 downto 0);
begin
 process(GCLK0)
 begin
  if rising_edge(GCLK0) then
   z <= z + 1;
  end if;
 end process;
 LD0 <= z(24);
 LD1 <= z(25);
 ...
 LD7 <= z(31);
end architecture;
```



Erzeugung der Konfigurationsdatei

- (Projektnavigator) File ▷ Open Project ▷ Ebene zurück ▷ Taktteiler ▷ Taktteiler.xise.
- (Hierarchy-Fenster) .../Taktteiler.vhd auswählen. Mit Doppelklick öffnen (Schaltungsbeschreibung).
- (Hierarchy-Fenster) Taktteiler-Behavioral aufklappen, .../Taktteiler.ucf auswählen. Mit Doppelklick öffnen (Constraints).
- (Hierarchy-Fenster) .../Taktteiler.vhd auswählen (Processes). Mit Doppelklick auf »Generate Programming File« (Abarbeitung: Synthese, Platzierung, Verdrahtung bis Konfigurationsdaten erzeugen).

Programmieren und Ausprobieren

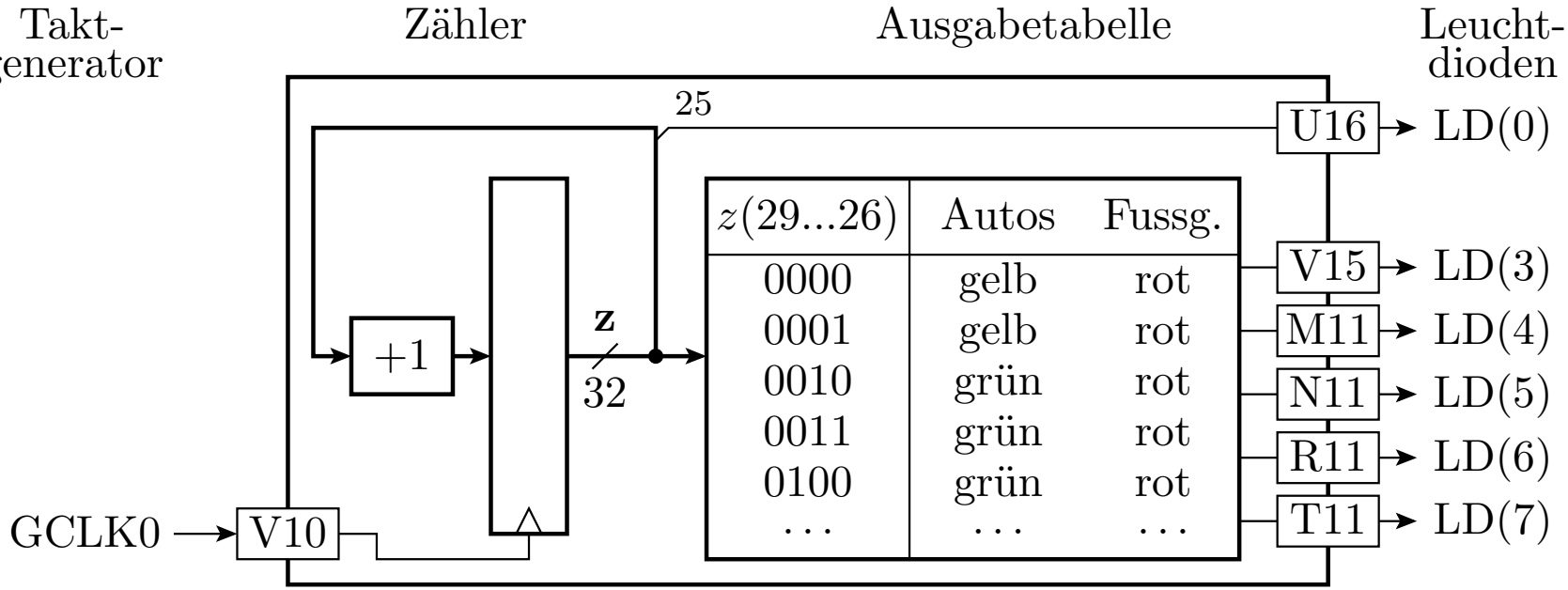
Wenn »Impact« noch geöffnet ist[2]:

- Rechtsklick auf den Chip »xc6slx16«. »Assign New Configuration File«.
- »taktteiler.bit« im Verzeichnis »Taktteiler« auswählen. »Open«. Attach SPI ... ▷ No.
- Rechtsklick auf den Chip »xc6slx16« ▷ »Program«.
- Ausprobieren.
- Schaltung ändern, z.B. höherfrequente Takte (niederwertigere) Zählerbits ausgeben oder Anschlusszuordnung in der ucf-Datei ändern.

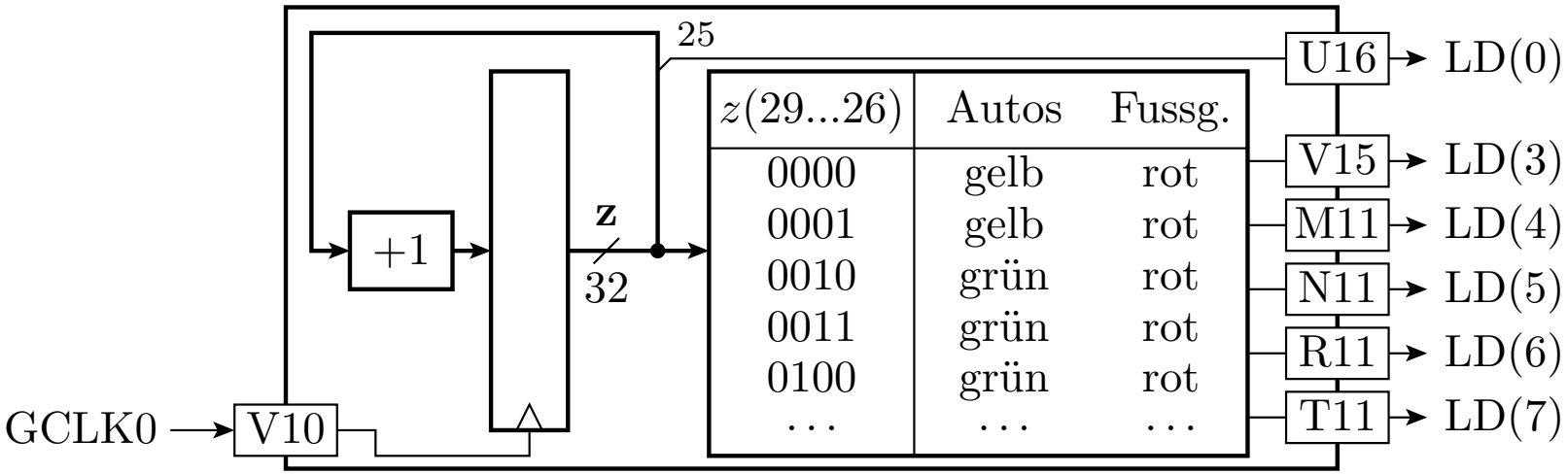
[2] »Impact« öffnen siehe Folie 11.

Ampelsteuerung

Ampelsteuerung




```
process(GCLK0)
begin
 if rising_edge(GCLK0) then
  z <= z + '1';              -- Zähler
  LD(0) <= z(25);              -- Taktausgabe
```




```
  case z(29 downto 26) is
   when "0000"|"0001"
    => LD(7 downto 3) <= b"010_10"; --A:gelb, F:rot
   when "0010"|"0011"
    => LD(7 downto 3) <= b"001_10"; --A:grün, F:rot
    -- ab hier selbst weiterentwickeln
   when others
    => LD(7 downto 3) <= b"100_10"; --A:rot, F:rot
  end case;
 end if;
end process;
```

Erzeugung der Konfigurationsdatei

- (Project Navigator) File ▷ Open Project ▷ Ebene zurück ▷ Ampel ▷ Ampel.xise.
- (Hierarchy-Fenster) .../Ampel.vhd auswählen. Mit Doppelklick öffnen (Schaltungsbeschreibung).
- (Hierarchy-Fenster) Ampel-Behavioral aufklappen, .../Ampel.ucf auswählen. Mit Doppelklick öffnen (Constraints).
- (Hierarchy-Fenster) .../Ampel.vhd auswählen (Processes). Mit Doppelklick auf »Generate Programming File« (Abarbeitung: Synthese, Platzierung, Verdrahtung bis Konfigurationsdaten erzeugen).

Programmieren und Ausprobieren

- Rechtsklick auf den Chip »xc6slx16«. »Assign New Configuration File«.
- »ampel.bit« im Verzeichnis »Ampel« auswählen. »Open«; Attach SPI ... ▷ No.
- Rechtsklick auf den Chip »xc6slx16« ▷ »Program«.
- Ausprobieren.
- Schaltung ändern, z.B. höherfrequente Takte (niederwertigere) Zählerbits ausgeben oder Anschlusszuordnung in der ucf-Datei ändern.

Ausgang	A:grün	A:gelb	A:rot	F:grün	F:rot
Anschluss	J7	H3	G1	L7	J6