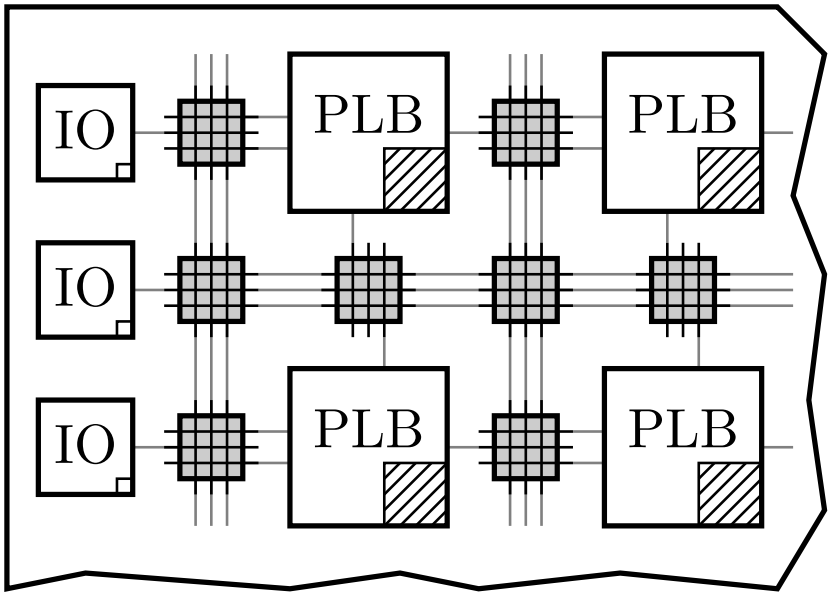
Schülerseminar
Programmieren einer Ampelsteuerung

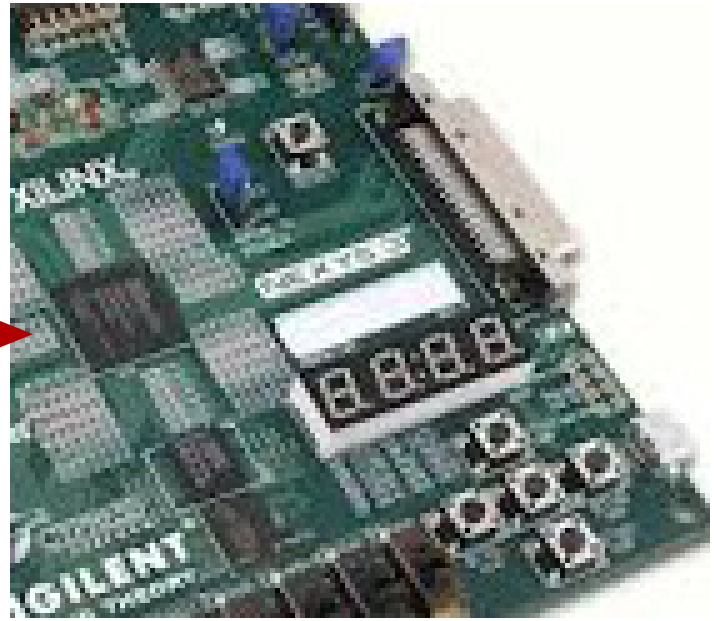
Prof. G. Kemnitz

Institut für Informatik

23. April 2016

Hardware-Programmierung





programmierbares Verbindungsnetzwerk

IO programmierbare Eingabe-Ausgabe-Schaltung

PLB programmierbarer Logikblock

- Man kann heute einen kompletten Rechner in wenigen Minuten in einen programmierbaren Schaltkreis auf einer Baugruppe laden.
- In den Schaltkreis passt eine Schaltung mit Millionen von Schaltelementen.
- Die Schalter, Leuchtdioden etc. auf der Baugruppe dienen zum Testen der Schaltungen

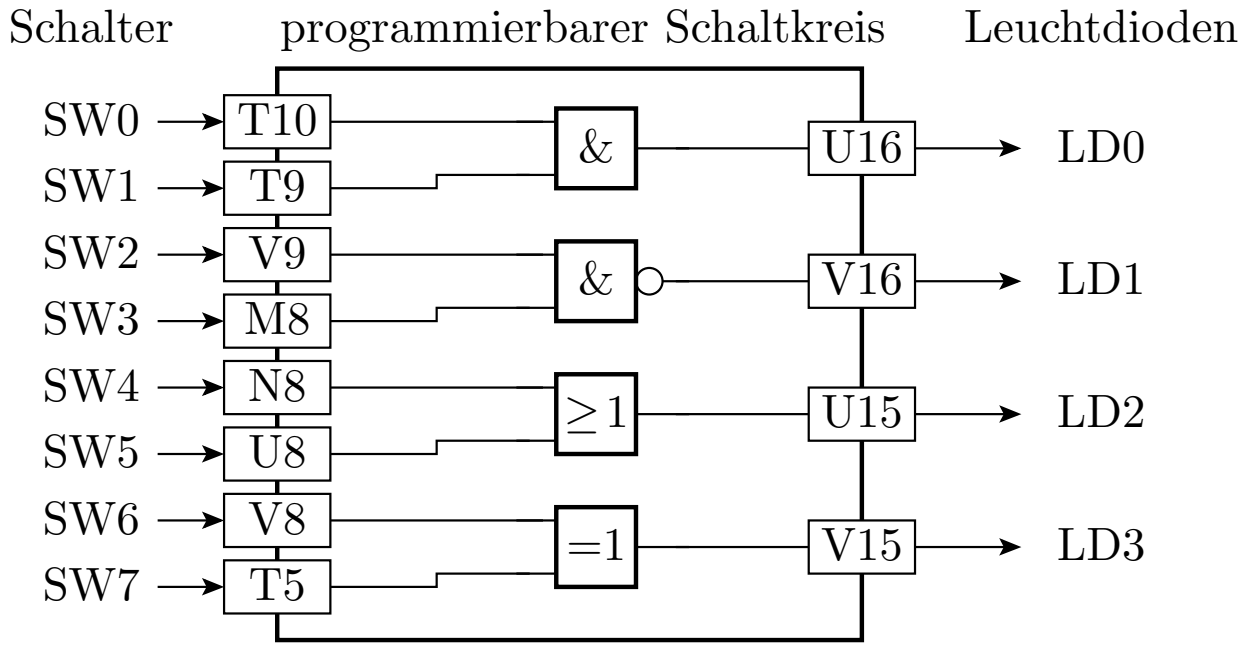
Einloggen und Foliensatz öffnen

- Unter Windows anmelden.
- Unter

  `Documents` ▷ `Schuelersem` ▷ `Schuelerseminar.pdf`

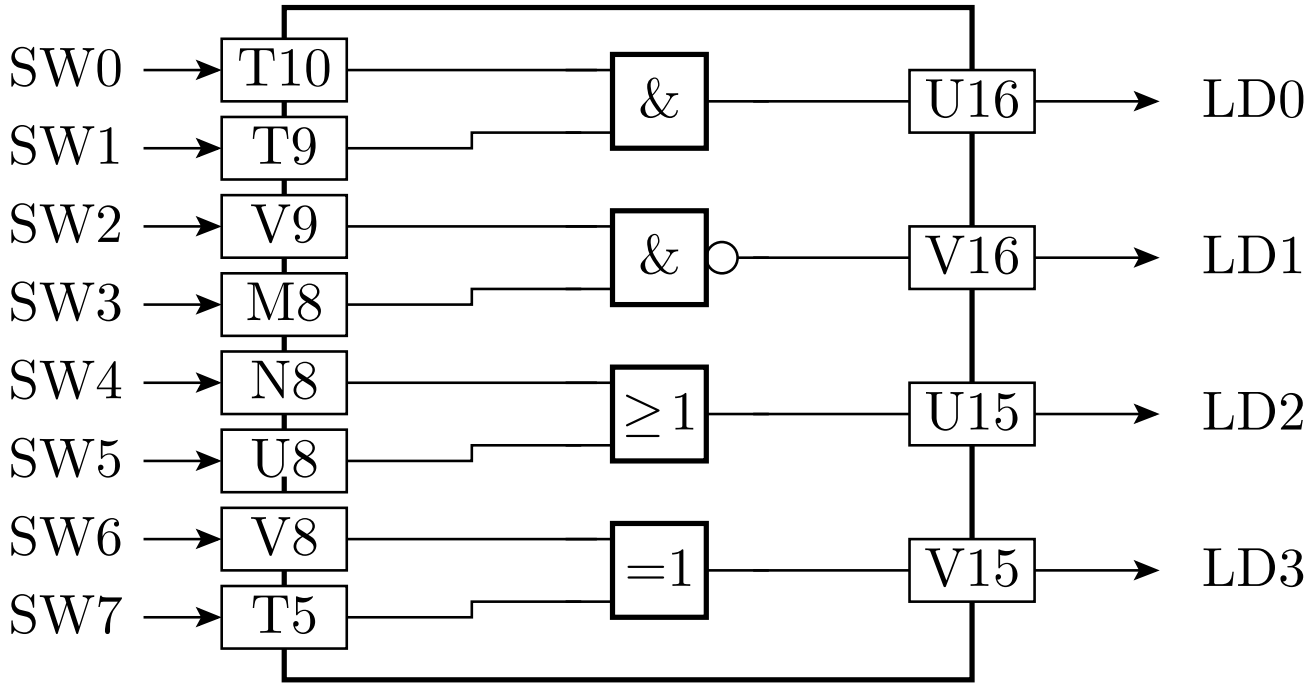
  öffnen.
- PDF-Viewer auf den rechten Bildschirm schieben.

Experiment 1: Logikrechner



- Schnittstellenbeschreibung

```
entity Logikrechner is
 port(SW0, SW1, ..., SW7: in STD_LOGIC;
      LD0, LD1, LD2, LD3: out STD_LOGIC);
end Logikrechner;
```



- Beschreibung der Funktion

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

Erzeugung der Konfigurationsdatei

- »Xilinx ISE« öffnen:
  `Start ▷ All Programs ▷ Xilinx Design Tools ▷ ISE Design Suite 14.6 ▷ ISE Design Tools ▷ 64-bit-Project Navigator`
- Fenster »Tip of the Day« schließen.
- (Project Navigator) File ▷ Open Project ▷ Documents ▷ Schuelersem ▷ Logikrechner ▷ Logikrechner.xise.
- (Hierarchy-Fenster) .../Logikrechner.vhd auswählen. Mit Doppelklick öffnen (Schaltungsbeschreibung).
- (Hierarchy-Fenster) Logikrechner-Behavioral aufklappen, .../Logikrechner.ucf auswählen. Mit Doppelklick öffnen.
- (Hierarchy-Fenster) .../Logikrechner.vhd auswählen. (Processes) mit Doppelklick auf »Generate Programming File« (Synthese, Platzierung, Verdrahtung, ...).

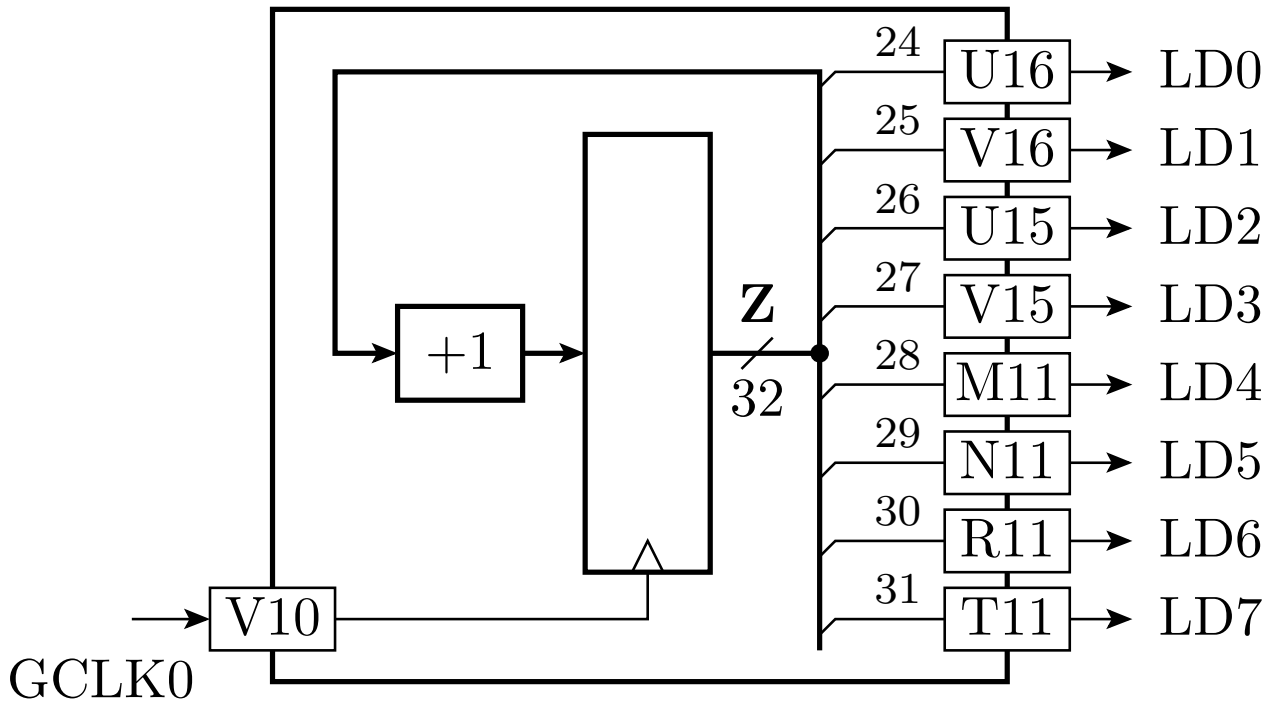
Programmieren und Ausprobieren

- »Configure Target Device ...« aufklappen.
- (Processes) Doppelklick auf »Manage Configuration Project (iMPACT)«.
- (iMPACT) Doppelklick auf »Boundary Scan«.
- (Boundary Scan) Rechtsklick auf »Initialize Chain«.
- »Auto Assign ...« »Yes«.
- für »xc6slxlx16« »logikrechner.bit« im Verzeichnis »Logikrechner« auswählen. »Open«.
- »Attache SPI...« »No«. Nächstes Fenster »OK«.
- Rechtsklick auf den Chip »xc6slxlx16« ▷ »Program«.
- Ausprobieren.
- Schaltung ändern, z.B.:

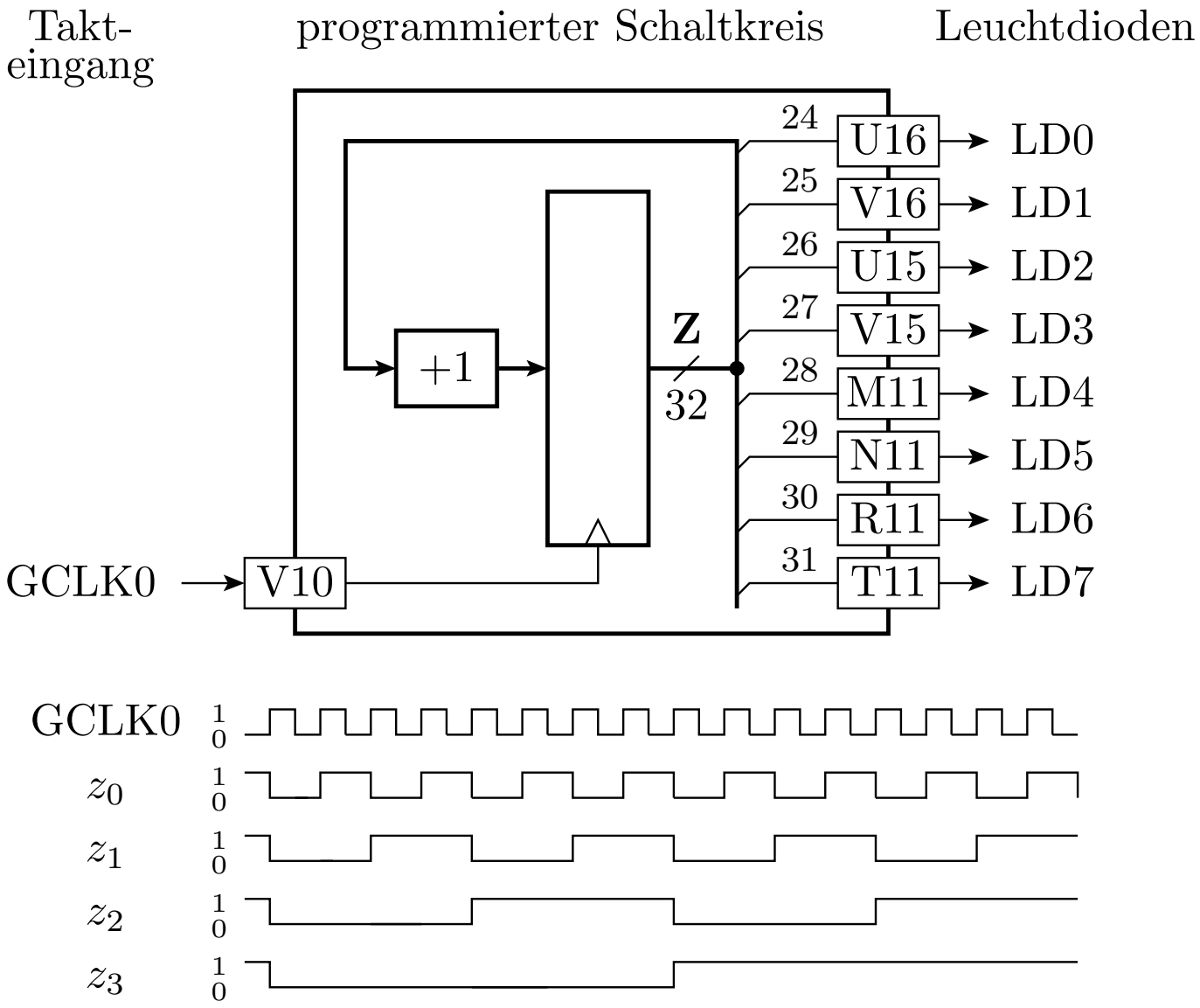
```
LD0 <= (SW0 and not SW1) or (not SW7 and SW6);
```

Experiment 2: Taktteiler

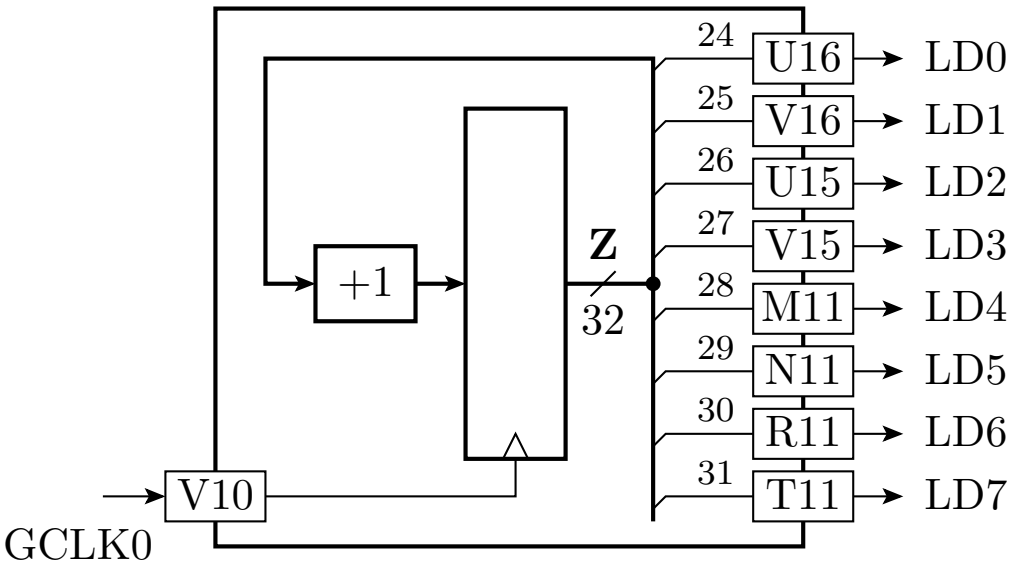
- Teilen des 100MHz-Eingabetakts durch 2^{32}
- Ausgabe der höchstwertigen Zählerstellen auf Leuchtdioden



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


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 Taktteiler;
architecture Behavioral of Taktteiler is
 signal z: STD_LOGIC_VECTOR(32 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 Behavioral;
```



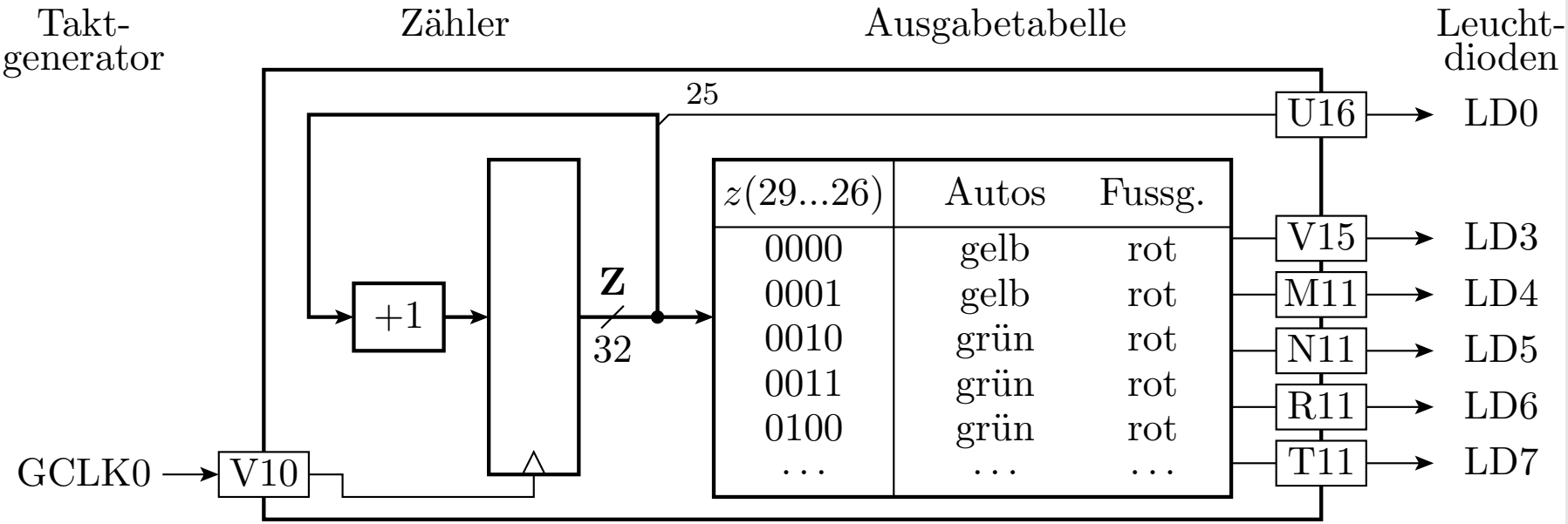
Erzeugung der Konfigurationsdatei

- (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.
- (Hierarchy-Fenster) .../Taktteiler.vhd auswählen; (Processes) mit Doppelklick auf »Generate Programming File« (Synthese, Platzierung, Verdrahtung, Konfigurationsdaten erzeugen)

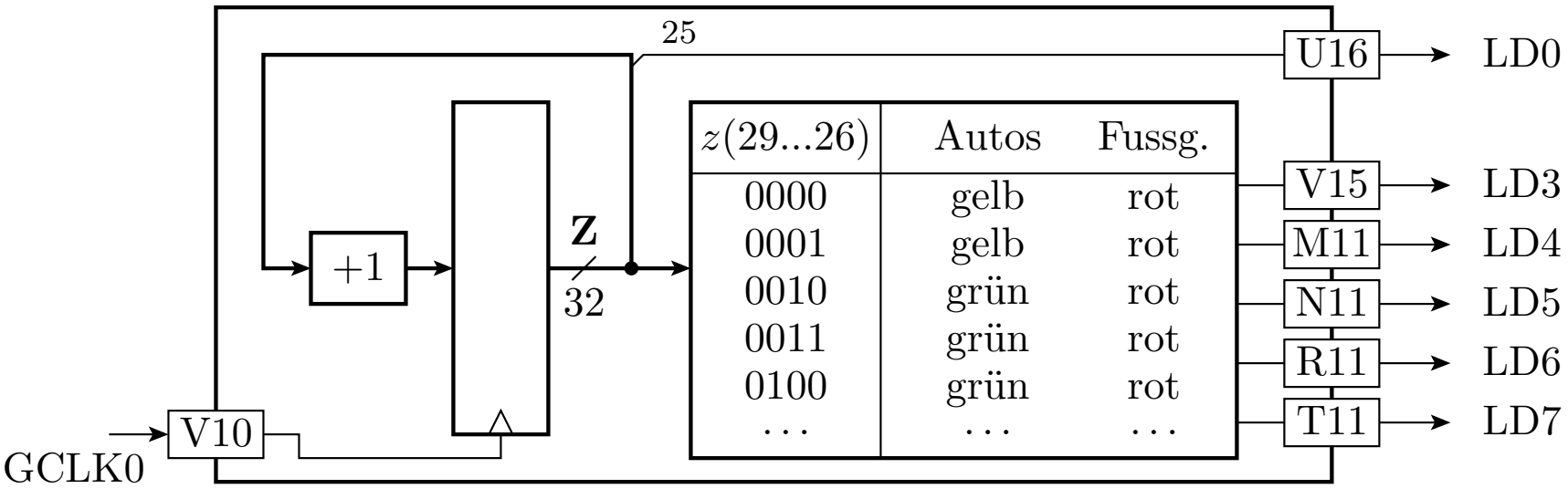
Programmieren und Ausprobieren

- Rechtsklick auf den Chip »xc6slx16«; »Assign New Configuration File«.
- »taktteiler.bit« im Verzeichnis »Taktteiler« auswählen; »Open«.
- Rechtsklick auf den Chip »xc6slx16« ▷ »Program«.
- Ausprobieren.
- Schaltung ändern, z.B. höherfrequente Takte (niederwertigere) Zählerbits ausgeben oder Anschlusszuordnung in der ucf-Datei ändern.

Experiment 3: Ampelsteuerung




```
process(GCLK0)
begin
 if RISING_EDGE(GCLK0) then
  z <= z + '1';          -- Zaehler
  LD0 <= 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.
- (Hierarchy-Fenster) .../Ampel.vhd auswählen; (Processes) mit Doppelklick auf »Generate Programming File« (Synthese, Platzierung, Verdrahtung, Konfigurationsdaten erzeugen).

Programmieren und Ausprobieren

- Rechtsklick auf den Chip »xc6slx16«; »Assign New Configuration File«
- »ampel.bit« im Verzeichnis »Ampel« auswählen; »Open«
- 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