Digitale Systeme

Prof. Dr.-Ing. Irenäus Schoppa

HTWG Konstanz

Begleitende Literatur

- Pong P. Chu: *FPGA Prototyping by VHDL Examples*, John Wiley & Sons, 1. Auflage, 2008.
- F. Kesel: *FPGA Hardware-Entwurf: Schaltungs- und System-Design mit VHDL und C/C++,* 4. Auflage, Oldenbourg, 2018.
- J. Reichardt und B. Schwarz: *VHDL-Synthese: Entwurf digitaler Schaltungen und Systeme*, Oldenbourg, 7. Auflage, 2015.
- K. Skahill: *VHDL for Programmable Logic*, Addison-Wesley, 1994.
- P. J. Ashenden: *The Designer's Guide to VHDL*, Morgan Kaufmann Publishers, 2008.
- Actel: *Actel HDL Coding – Style Guide*, Actel Corporation, 2009. (hdlcode.pdf)
- Microsemi: Benutzerhandbücher, Datenblätter, Applikationsberichte, Microsemi Corporation, 2008..2021

- Digilent Pmod™ Interface Specification

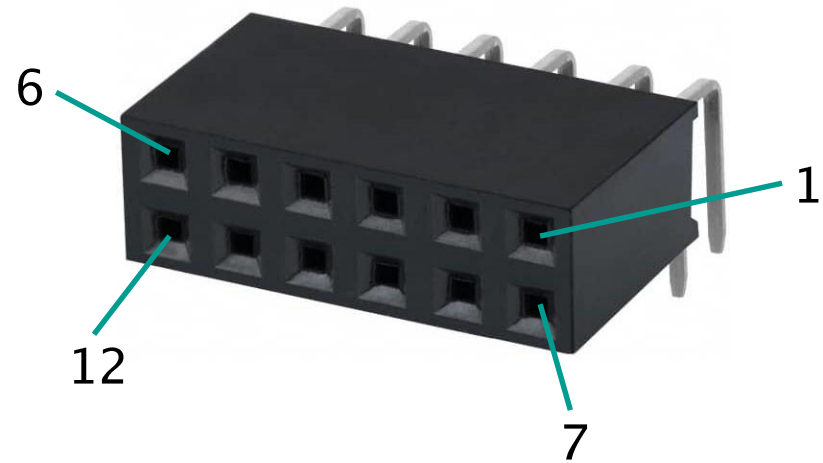
Pmod Type 1 (GPIO)

Pin	Signal	Direction
1	IO1	In/Out
2	IO2	In/Out
3	IO3	In/Out
4	IO4	In/Out
5	GND	
6	VCC	(+3,3V)

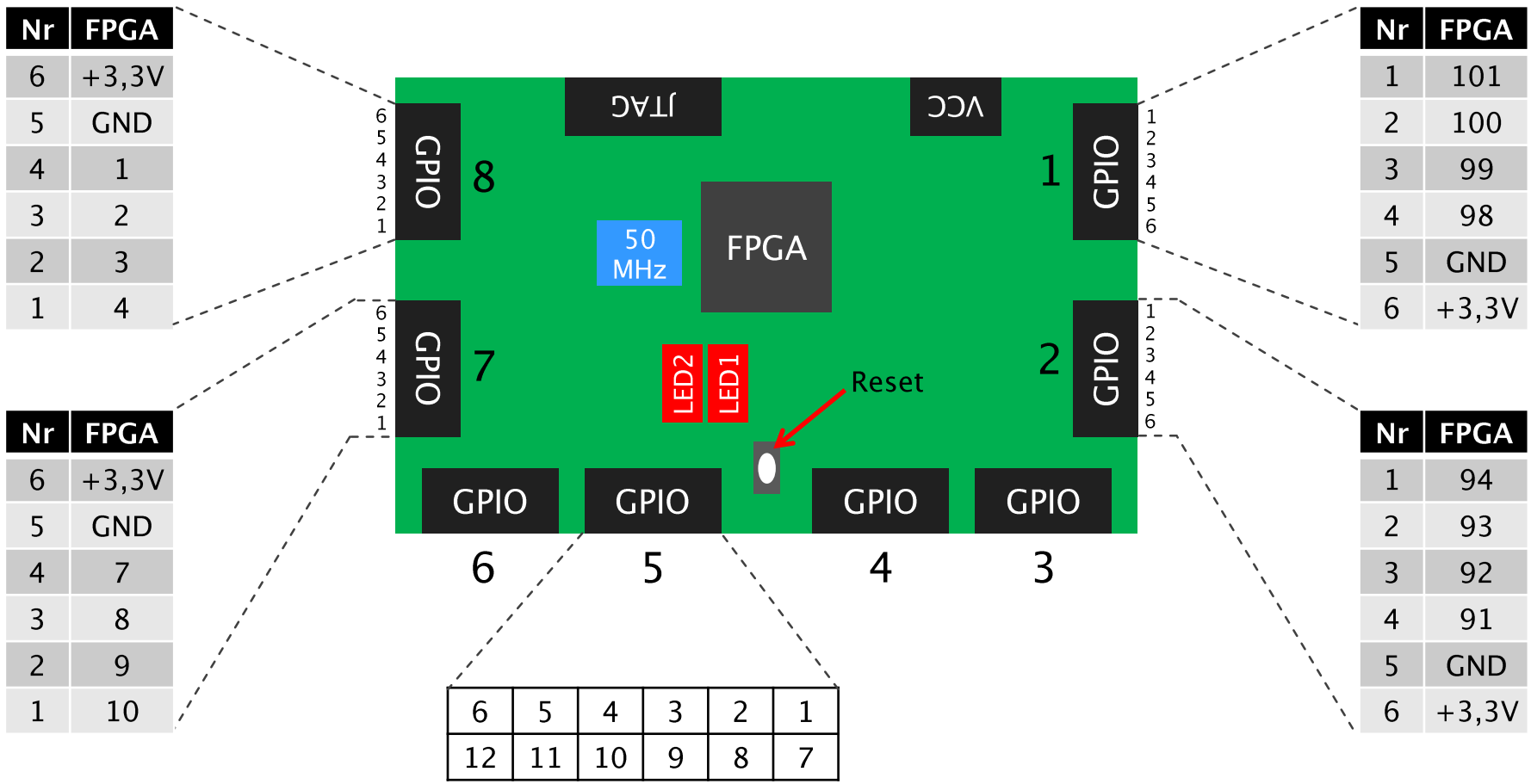
Type 1A (Expanded GPIO)

Pin	Signal	Direction
1	IO1	In/Out
2	IO2	In/Out
3	IO3	In/Out
4	IO4	In/Out
5	GND	
6	VCC	(+3,3V)

Pin	Signal	Direction
7	IO5	In/Out
8	IO6	In/Out
9	IO7	In/Out
10	IO8	In/Out
11	GND	
12	VCC	(+3,3V)



PMOD: 4 x Pmod Type 1 (GPIO)



4 x Type 1A (Expanded GPIO)

PMOD: 4 x Type 1A (Expanded GPIO)

6

Nr	FPGA	Nr	FPGA
1	94	7	94
2	93	8	93
3	92	9	92
4	91	10	91
5	GND	11	GND
6	+3,3V	12	+3,3V

5

Nr	FPGA	Nr	FPGA
1	46	7	47
2	42	8	43
3	40	9	41
4	38	10	39
5	GND	11	GND
6	+3,3V	12	+3,3V

4

Nr	FPGA	Nr	FPGA
1	66	7	61
2	62	8	63
3	60	9	60
4	56	10	57
5	GND	11	GND
6	+3,3V	12	+3,3V

3

Nr	FPGA	Nr	FPGA
1	89	7	90
2	87	8	88
3	85	9	86
4	81	10	82
5	GND	11	GND
6	+3,3V	12	+3,3V

Signal	FPGA
LED1	51
LED2	50
CLK	24
RST	53

Beispiel: „Hallo, world!” in Hardware -> „I’m alive”

- „Eine neue Programmiersprache lernt man nur, wenn man in ihr Programme schreibt.” (Zitat aus dem Standardwerk von Kernighan und Ritchie „Programmieren in C“)

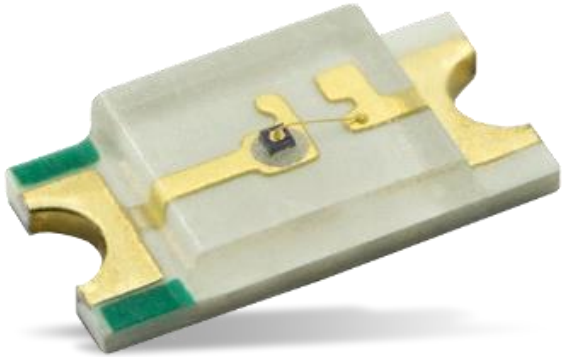
```
#include <stdio.h>

void main(void) {
   printf(”Hello, world!\n”);
}
```

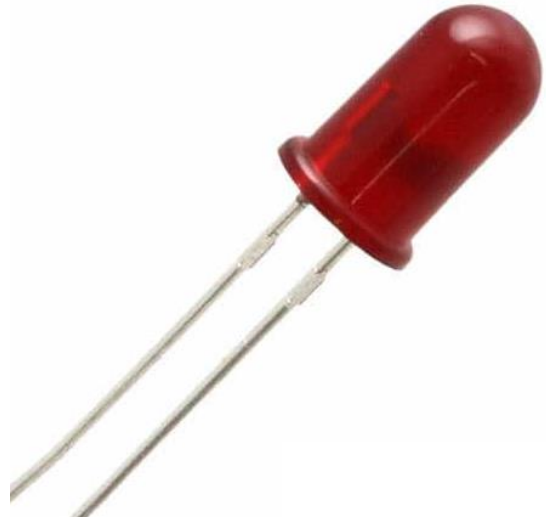
- In der Hardware ist eine solche Textausgabe kaum realisierbar.
- Wir wollen aber erreichen, dass ein digitales System sich der Umgebung irgendwie bekannt gibt, dass „es lebt”.
- Wir ersetzen die Ausgabe „Hallo, world! ” durch eine visuelle Anzeige „I’m alive”.

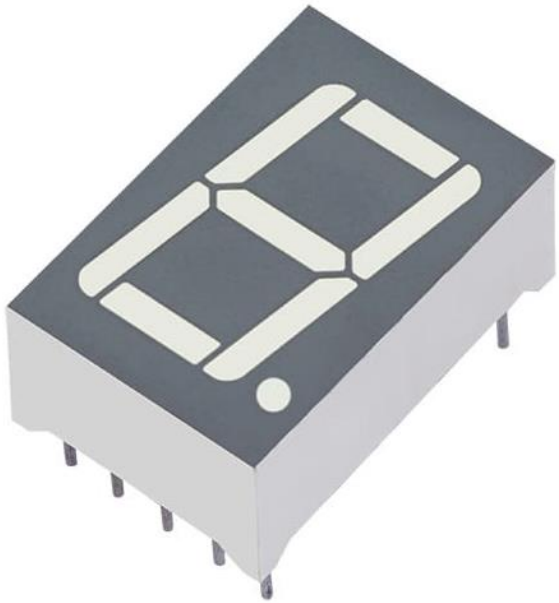
Beispiel: „I'm alive"

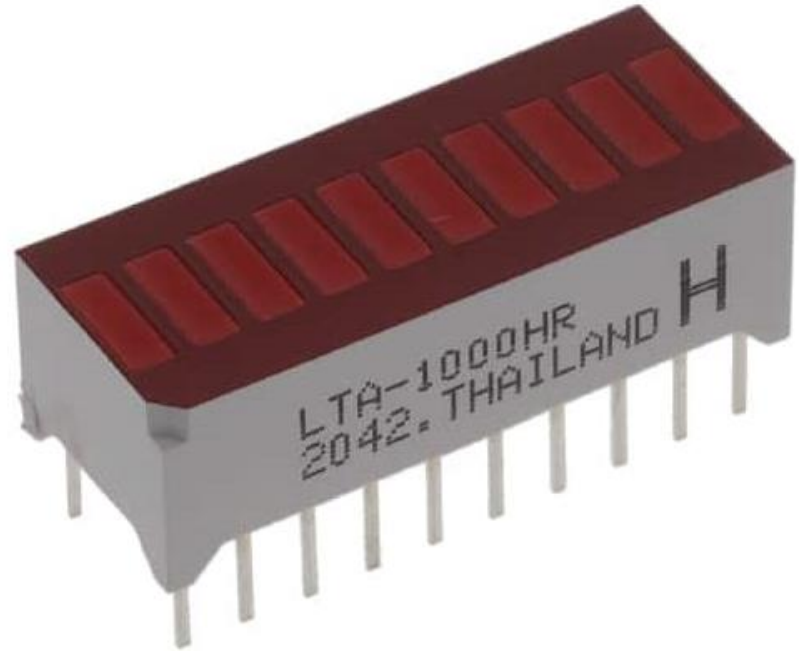
- Ein digitales System soll mit dem Blinken einer Leuchtdiode (LED) zeigen, dass es lebt.
- Es ist immer gut, eine oder mehrere (grüne, rote) LEDs in einem digitalen System zu haben. Auf diese Weise lassen sich verschiedene interne Zustände der Schaltung mit unterschiedlichen Blinkmustern anzeigen.
- Was braucht man, um eine LED blinken zu lassen?
  - Man muss wissen,
    - wo sich die LED physikalisch befindet, also an welchem Pin sie angeschlossen ist,
    - ob sie mit LOW oder mit HIGH angesteuert wird,
    - mit welcher Geschwindigkeit (bzw. mit welchem Blinkmuser) sie arbeiten soll.

Verschiedene Bauformen von LEDs



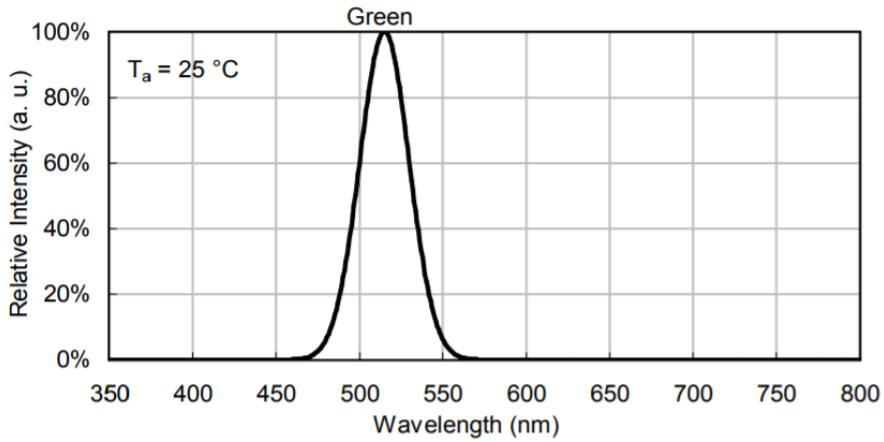


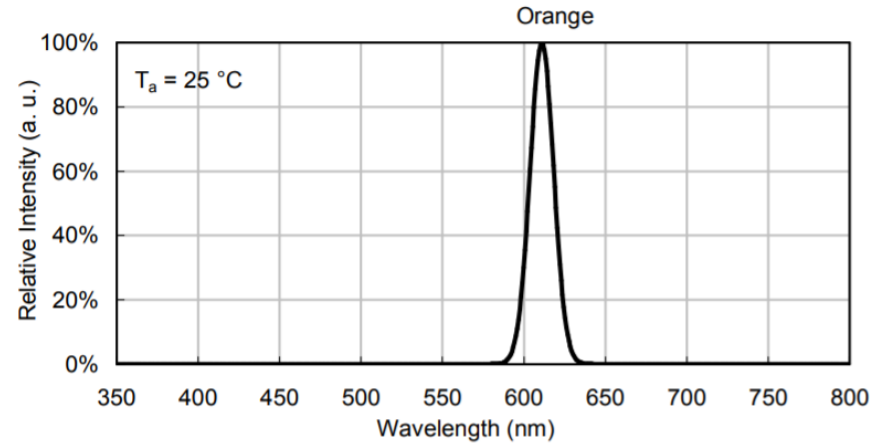




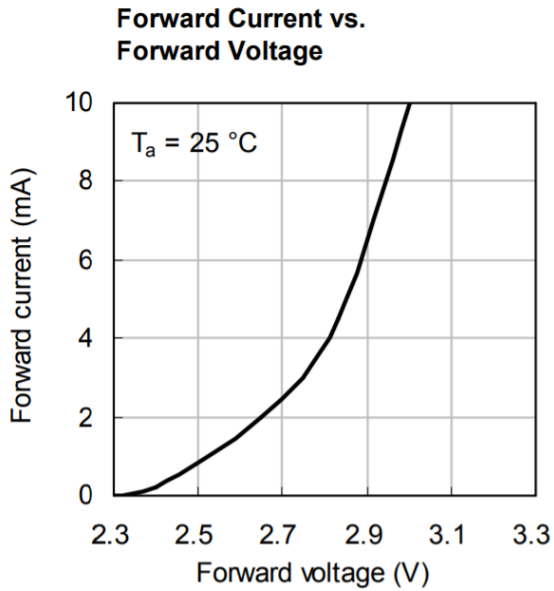
Technische Parameter einer LED

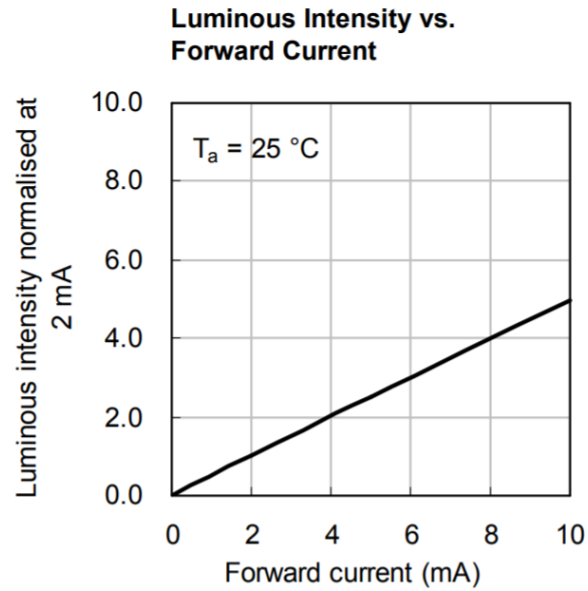
- Farbe (Wellenlänge)





- Helligkeit





Technische Parameter einer LED

Kingbright

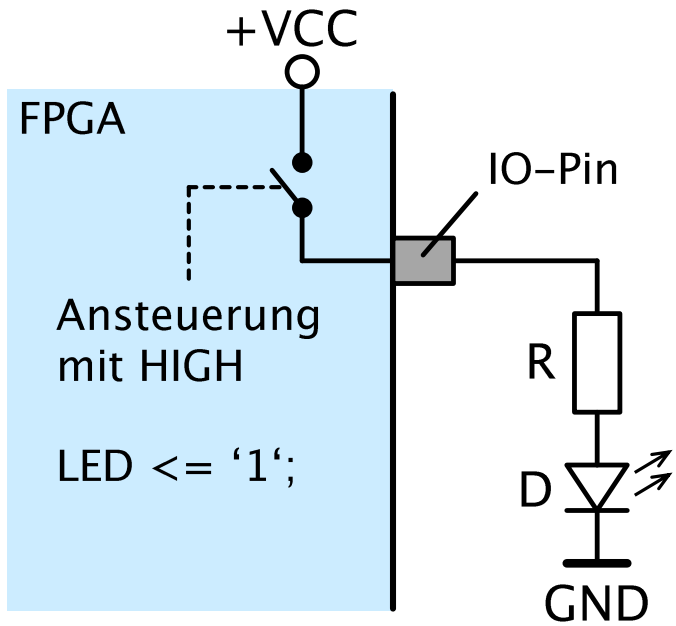
APT1608LZGCK

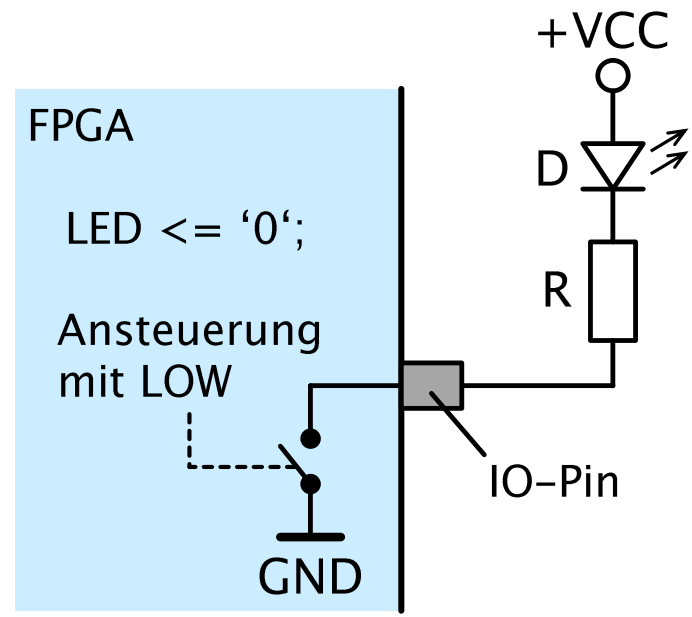
ELECTRICAL / OPTICAL CHARACTERISTICS at T_A=25°C

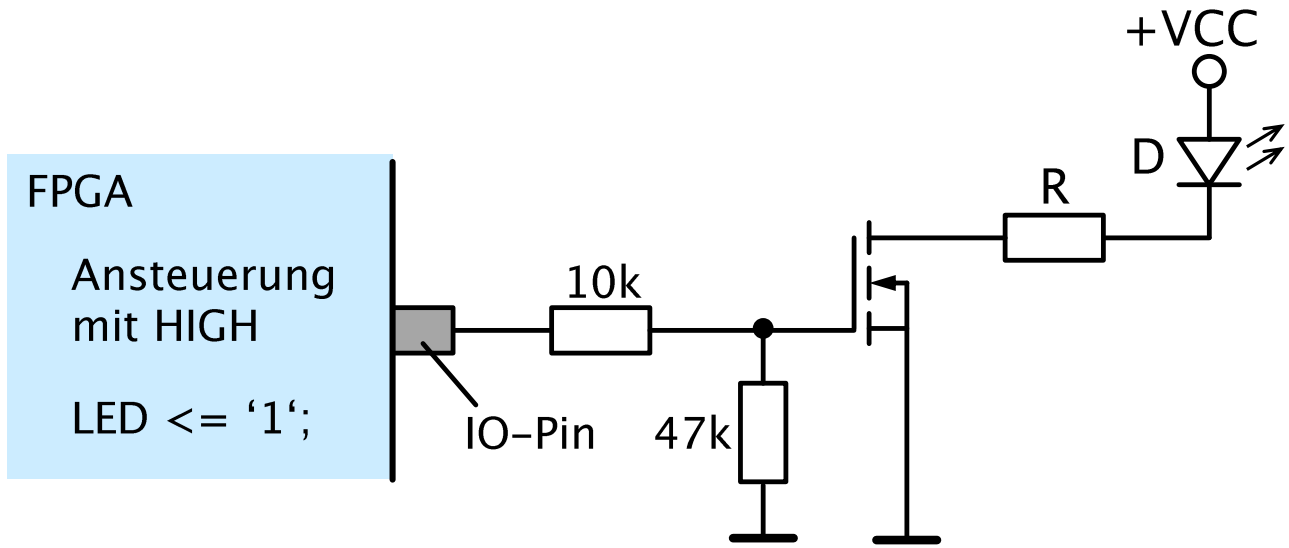
Parameter	Symbol	Emitting Color	Value			Unit
			Min.	Typ.	Max.	
Wavelength at Peak Emission I_F = 2mA	λ_{peak}	Green	-	515	-	nm
Dominant Wavelength I_F = 2mA	λ_{dom} [1]	Green	-	525	-	nm
Spectral Bandwidth at 50% Φ REL MAX I_F = 2mA	Δλ	Green	-	35	-	nm
Capacitance	C	Green	-	45	-	pF
Forward Voltage I_F = 2mA	V_F [2]	Green	2.2	2.65	3.0	V
Reverse Current (V_R = 5V)	I_R	Green	-	-	50	μA
Temperature Coefficient of λ_{peak} I_F = 2mA, -10°C ≤ T ≤ 85°C	$TC_{\lambda peak}$	Green	-	0.05	-	nm/°C
Temperature Coefficient of λ_{dom} I_F = 2mA, -10°C ≤ T ≤ 85°C	$TC_{\lambda dom}$	Green	-	0.03	-	nm/°C
Temperature Coefficient of V_F I_F = 2mA, -10°C ≤ T ≤ 85°C	TC_V	Green	-	-2.9	-	mV/°C

Notes:
1. The dominant wavelength (λd) above is the setup value of the sorting machine. (Tolerance λd : ±1nm.)
2. Forward voltage: ±0.1V.
3. Wavelength value is traceable to CIE127-2007 standards.
4. Excess driving current and / or operating temperature higher than recommended conditions may result in severe light degradation or premature failure.

Ansteuerung einer LED







Ansteuerung einer LED

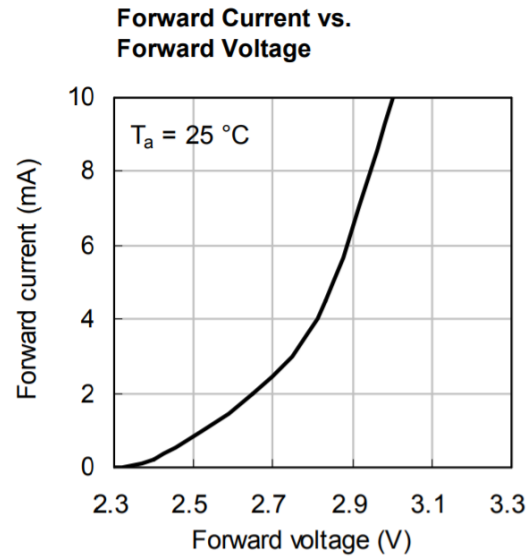
- Berechnung des Vorwiderstandes R

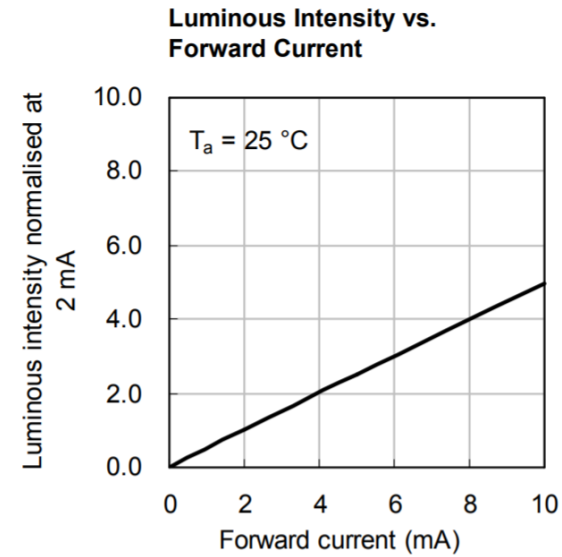
$$R = \frac{V_{HIGH} - V_F - V_{LOW}}{I_F}$$

- Parameter

  $V_{HIGH} = +3{,}3V$, $V_{LOW} = 0V$,
  $I_F = 2mA$, $V_F = 2{,}65V$

$$R = \frac{3{,}3V - 2{,}65V - 0V}{2mA} = 325\Omega$$





  - Beim Einsatz der LED auf der Platine als Indikator ist die Helligkeit der LED bei diesem Vorwiderstand viel zu hoch => der Anwender wird geblendet. Die abgestrahlte Lichtmenge kann man auf ein angenehmes Niveau reduzieren, indem man einen Vorwiderstand mit einem größeren Nominalwert z.B. 5,1 kΩ einsetzt.

Blinken einer LED

- Notation in einer Pseudosprache

eine LED soll mit der Frequenz 1 Hz blinken

```
wiederhole folgende Anweisungen:
  schalte die LED ein;
  warte 500 Millisekunden;
  schalte die LED aus;
  warte 500 Millisekunden;
```

- Notation als ein simulierbares aber nicht synthesefähiges VHDL-Programm.

Beispiel 1

```
-- eine LED soll mit der Frequenz 1 Hz blinken

ENTITY blink1 IS
  PORT(LED: OUT bit);
END blink1;

ARCHITECTURE verhalten OF blink1 IS
BEGIN
  PROCESS BEGIN
    LOOP
      LED <= '1';
      WAIT FOR 500 ms;
      LED <= '0';
      WAIT FOR 500 ms;
    END LOOP;
  END PROCESS;
END verhalten;
```

Blinken einer LED


```
-- eine LED soll mit der Frequenz 1 Hz blinken
```

Kommentarzeile

```
ENTITY blink1 IS
  PORT(LED: OUT BIT);
END blink1;
```

Eine Entwurfseinheit (ENTITY) namens *blink1* mit einer Schnittstellendeklaration (PORT). Die Schnittstelle beinhaltet ein Ausgangssignal (OUT) namens *LED* vom Typ BIT.

```
ARCHITECTURE verhalten OF blink1 IS
BEGIN
  PROCESS BEGIN
    LOOP
      LED <= '1';
      WAIT FOR 500 ms;
      LED <= '0';
      WAIT FOR 500 ms;
    END LOOP;
  END PROCESS;
END verhalten;
```

Eine Beschreibung (ARCHITECTURE) namens *verhalten*, die zur Entwurfseinheit *blink1* gehört, beinhaltet einen parallelen Anweisungsbereich (BEGIN), in dem sich ein Prozess (PROCESS – END PROCESS) mit seinem sequentiellen Anweisungsbereich befindet. Im Prozess ist eine Endlosschleife (LOOP – END LOOP) enthalten. In deren Anweisungsbereich vier Anweisungen stehen: zwei elementare Signalzuweisungen (<=) sowie zwei Warteanweisungen mit einer Verzögerungsangabe von 500 Millisekunden (WAIT FOR 500 ms).

Blinken einer LED

- Die Beschreibung einer Schaltung in VHDL kann auf unterschiedliche, aber funktional äquivalente Arten erfolgen. Allerdings bringt das bei einer Schaltungssynthese aus einer VHDL-Beschreibung nicht immer die gleichen Ergebnissen hervor.
- Aus der Sicht der Synthese enthält VHDL Sprachkonstrukte, die überhaupt nicht (FILE, ASSERT, AFTER) oder nur bedingt (RECORD, REGISTER, WHILE) in Schaltungsstrukturen umgesetzt werden können.

- Die Schaltungssynthese aus einer VHDL-Beschreibung ist nur unter Einhaltung bestimmter Beschreibungsregeln und eines bestimmten Beschreibungsstils möglich.

Blinken einer LED


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

IEEE-Bibliothek (LIBRARY) mit der neunwertigen Standardlogik *std_logic_1164* (Man sollte sie immer benutzen.)

Beispiel 2

```
ENTITY blink IS
   PORT(rst: IN  std_logic;  -- low active
        clk: IN  std_logic;  -- rising edge active
        led: OUT std_logic); -- high active
END blink;
```

Schnittstellendeklaration (PORT) mit zwei Eingangssignalen (*rst* und *clk*) und einem Ausgangssignal (*led*). Eingangssignale werden mit IN und Ausgangssignale mit OUT deklariert.

```
ARCHITECTURE verhalten OF blink IS

   CONSTANT RSTDEF: std_logic := '0';
   CONSTANT SYSFRQ: real      := 100.0e6;
   CONSTANT LEDFRQ: real      := 1.0;
   CONSTANT CNTMAX: natural   := natural(SYSFRQ/LEDFRQ);

   SIGNAL cnt: integer RANGE 0 TO CNTMAX-1;

BEGIN

  -- Anweisungsbereich mit einem Prozess

END verhalten;
```

Definitions- und Deklarationsbereich einer Architektur.
Hier sind vier Konstanten definiert sowie ein lokales Signal namens *cnt* deklariert. Das Signal *cnt* hat den Datentyp *integer*, allerdings mit einem eingeschränkten Wertebereich (RANGE 0 TO CNTMAX-1).

Blinken einer LED


```
ARCHITECTURE verhalten OF blink IS

-- Definitions- und Deklarationsbereich

BEGIN

  p1: PROCESS (rst, clk) IS
  BEGIN

    IF rst=RSTDEF THEN
      cnt <= 0;
      led <= '0';
    ELSIF rising_edge(clk) THEN
      cnt <= (cnt + 1) MOD CNTMAX;
      IF cnt=CNTMAX/2-1 THEN
        led <= '1';
      ELSIF cnt=CNTMAX-1 THEN
        led <= '0';
      END IF;
    END IF;

  END PROCESS;

END verhalten;
```

Im Anweisungsbereich der Architektur steht ein „getakteter“ also synchroner Prozess namens *p1*, der über seine Sensitivitätsliste (*rst*, *clk*) durch Werteänderungen an den beiden Signalen getriggert wird. Im Prozess steht eine IF-ELSIF-END IF-Anweisung, die eine Priorisierung der beiden Signale bewirkt.
Im IF-Zweig werden ein Zähler namens *cnt* und die LED auf Anfangswerte (zurück-)gesetzt. Die Bedingung (rst=RSTDEF) in dem IF-Zweig gibt an, wann das Zurücksetzen passieren soll. Im ELSIF-Zweig werden Anweisungen ausgeführt, wenn die vorherige Bedingung nicht erfüllt ist. Die Bedingung im ELSIF-Zweig (rising_edge(clk)) gibt an, dass die Ausführung der Anweisungen nur dann stattfinden soll, wenn eine steigende Flanke an *clk* festgestellt wird.
Danach kommen das Inkrementieren des Zählern mit einem Modulo-Operator, sowie zwei Vergleichs-operationen mit den dazugehörigen Signalzuweisungen.

Basisbeschreibungen synchroner Prozesse


```
-- synchroner Prozess mit einem
-- asynchronen Rücksetzverhalten

PROCESS (rst, clk) IS
-- lokale Definitionen und Deklarationen
BEGIN

  IF rst=RSTDEF THEN
    -- Reset-Anweisungen
  ELSIF rising_edge(clk) THEN
    -- Anweisungen
  END IF;

END PROCESS;
```

```
-- synchroner Prozess mit einem
-- synchronen Rücksetzverhalten

PROCESS (clk) IS
-- lokale Definitionen und Deklarationen
BEGIN

  IF rising_edge(clk) THEN
    IF rst=RSTDEF THEN
      -- Reset-Anweisungen
    ELSE
      -- Anweisungen
    END IF;
  END IF;

END PROCESS;
```

Basisbeschreibungen synchroner Prozesse

- Getaktete, digitale Systeme benötigen zwingend zwei Signale zum Arbeiten:
  - ein Rücksetzsignal (reset, rst), mit dem interne Speicherelemente (Flipflops oder Register) auf vorgegebene Anfangswerte gesetzt werden,
    (eine Initialisierung bei der Deklaration hat nur in der Simulation eine Wirkung, sie ist nicht synthesefähig)
  - ein Taktsignal (clock, clk), mit dem Schaltvorgänge ausgeführt werden, wobei die Schaltvorgänge auf die steigende Taktflanke (rising_edge(clk)) oder auf die fallende Taktflanke (falling_edge(clk)) reagieren sollen.

- Der Takt in einem digitalen System ist eine „heilige“ Größe und darf mit anderen Signalen auf keinen Fall verknüpft werden.

Blinken einer LED

- Vermeidung der Modulo-Operation

Beispiel 3

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

ENTITY blink IS
  PORT(rst:  IN  std_logic;
       clk:  IN  std_logic;
       led:  OUT std_logic);
END blink;


ARCHITECTURE verhalten OF blink IS

  CONSTANT RSTDEF: std_logic := '0';
  CONSTANT SYSFRQ: real      := 100.0e6;
  CONSTANT LEDFRQ: real      := 1.0;
  CONSTANT CNTMAX: natural   := natural(SYSFRQ/LEDFRQ)/2;

  SIGNAL cnt: integer RANGE 0 TO CNTMAX-1;
  SIGNAL dff: std_logic;
```

```
BEGIN

  led <= dff;

  p1: PROCESS (rst, clk) IS
  BEGIN
    IF rst=RSTDEF THEN
      cnt <= 0;
      dff <= '0';
    ELSIF rising_edge(clk) THEN
      IF cnt=CNTMAX-1 THEN
        dff <= NOT dff;
        cnt <= 0;
      ELSE
        cnt <= cnt + 1;
      END IF;
    END IF;
  END PROCESS;

END verhalten;
```

Blinken einer LED

- Aufteilung des Zählers in zwei kürzere Zähler

Beispiel 4

```
ARCHITECTURE verhalten OF blink IS

  CONSTANT RSTDEF:  std_logic := '0';
  CONSTANT CNTMAX1: natural   :=  5000;
  CONSTANT CNTMAX2: natural   := 10000;

  SIGNAL dff1: std_logic;
  SIGNAL dff2: std_logic;
  SIGNAL cnt1: integer RANGE 0 TO CNTMAX1-1;
  SIGNAL cnt2: integer RANGE 0 TO CNTMAX2-1;

BEGIN

  led <= dff2;

  p1: PROCESS  . . .

  p2: PROCESS  . . .

END verhalten;
```

```
p1: PROCESS (rst, clk) IS
BEGIN
  IF rst=RSTDEF THEN
    dff1 <= '0';
    cnt1 <= 0;
  ELSIF rising_edge(clk) THEN

    dff1 <= '0';
    IF cnt1=CNTMAX1-1 THEN
      dff1 <= '1';
      cnt1 <= 0;
    ELSE
      cnt1 <= cnt1 + 1;
    END IF;
  END IF;

END PROCESS;
```

```
p2: PROCESS (rst, clk) IS
BEGIN
  IF rst=RSTDEF THEN
    dff2 <= '0';
    cnt2 <= 0;
  ELSIF rising_edge(clk) THEN
    IF dff1='1' THEN
      IF cnt2=CNTMAX2-1 THEN
        dff2 <= not dff2;
        cnt2 <= 0;
      ELSE
        cnt2 <= cnt2 + 1;
      END IF;
    END IF;
  END IF;
END PROCESS;
```

Blinken einer LED

- Ein herkömmlicher Modulo-N-Zähler arbeitet auf der Grundlage eines Inkrementierers ((i+1) mod N).
  - Ein Modulo-N-Zähler erzeugt eine Folge von Zwischenwerten, die nummerisch zwischen 0 und N-1 aufsteigend liegen (z.B. Modulo-8-Zähler: 0, 1, 2, 3, 4, 5, 6, 7).
  - Die Differenz zwischen zwei benachbarten Werten beträgt immer 1.
  - Der Zählvorgang wird beim Erreichen des Endwertes (N-1) beendet und ggf. mit einem Initialisierungswert neu gestartet.
- Kommt es bei einem Zählvorgang nicht auf die Reihenfolge der Zwischenwerte an, so kann ein Zähler auch so realisiert werden, dass er bis zu einem vorgegebenen Endwert zählt, die Zwischenwerte aber völlig egal sind, solange sie eindeutig sind.
- In diesem Fall kann ein Zähler als linear rückgekoppeltes Schiebe-register (LFSR, linear feedback shift register) mit XOR-Operationen realisiert werden.

Blinken einer LED

- Linear rückgekoppelte Schieberegister kommen zum Einsatz als
  - Pseudozufallszahlengeneratoren
  - Rauschgeneratoren in der digitalen Signalverarbeitung
  - Prüfsummengeneratoren bei der Fehlererkennung in Kommunikations-protokollen oder Kompressionsverfahren, (Zyklische Redundanzprüfung, CRC, Cyclic Redundancy Check)
- Ein linear rückgekoppeltes Schieberegister besteht aus einem N-Bit-Schieberegister und aus einem sog. Generatorpolynom.
  - N-Bit-Schieberegister werden in der Digitaltechnik mit bitseriell verketteten Speicherelementen, vorzugsweise mit synchronen D-Flipflops, realisiert.
  - An manchen Stellen zwischen den Speicherelementen befinden sich Abzweigungen für eine Rückkopplungsfunktion.
  - Rückkopplungsfunktionen basieren auf Generatorpolynomen und bestimmen sowohl die Anzahl als auch die Position dieser Abzweigungen im Schieberegister.

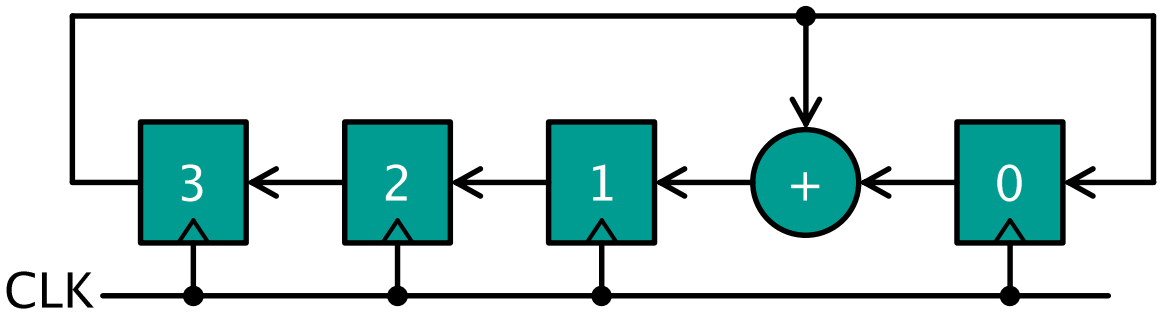
Blinken einer LED

- Als Generatorpolynome kommen häufig primitive Polynome n-ten Grades zum Einsatz.
- Solche Polynome zeichnen sich dadurch aus, dass sie eine maximale Periodenlänge von 2^n-1 haben. Nach Ablauf der Periode wiederholt sich die Sequenz der generierten Werte.
- Listen/Tabellen mit primitiven Polynomen findet man im Internet
  - Quelle: https://www.partow.net/programming/polynomials/index.html

```
x^2 + x^1 + 1
x^3 + x^1 + 1
x^4 + x^1 + 1
x^5 + x^2 + 1
x^5 + x^4 + x^2 + x^1 + 1
x^5 + x^4 + x^3 + x^2 + 1
x^6 + x^1 + 1
x^6 + x^5 + x^2 + x^1 + 1
x^6 + x^5 + x^3 + x^2 + 1
. . .
```

Blinken einer LED

- Realisierung des Zählers als Schieberegister mit XOR



G(x) = x^4 + x^1 + 1

G(x) als Bitvektor 10011

```
SIGNAL reg: std_logic_vector(3 DOWNTO 0);

PROCESS (rst, clk) IS
BEGIN
  IF rst=RSTDEF THEN
    reg <= (OTHERS => '1');
  ELSIF rising_edge(clk) THEN
    reg(3) <= reg(2);
    reg(2) <= reg(1);
    reg(1) <= reg(0) XOR reg(3);
    reg(0) <= reg(3);
  END IF;
END PROCESS;
```

Schritt	–	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	1
Wert (Hex)	F	D	9	1	2	4	8	3	6	C	B	5	A	7	E	F	D

Blinken einer LED

- Realisierung des Zählers als Schieberegister mit XOR

Beispiel 5

```
ARCHITECTURE verhalten OF blink IS

  CONSTANT RSTDEF: std_logic := '0';
  CONSTANT RES: std_logic_vector :=
      "01111110101011001111010110";

  SIGNAL reg: std_logic_vector(25 DOWNTO 0);
  SIGNAL dff: std_logic;

BEGIN

  led <= dff;

  p1: PROCESS . . .

END verhalten;
```

```
p1: PROCESS (rst, clk) IS
BEGIN
  IF rst=RSTDEF THEN
    dff <= '0';
    reg <= (OTHERS => '1');
  ELSIF rising_edge(clk) THEN
    IF reg=RES THEN
      dff <= NOT dff;
      reg <= (OTHERS => '1');
    ELSE
      -- Polynom: x^26 + x^6 + x^2 + x^1 + 1
      reg(25 DOWNTO 7) <= reg(24 DOWNTO 6);
      reg(6) <= reg(5) XOR reg(25);
      reg(5 DOWNTO 3) <= reg(4 DOWNTO 2);
      reg(2) <= reg(1) XOR reg(25);
      reg(1) <= reg(0) XOR reg(25);
      reg(0) <= reg(25);
    END IF;
  END IF;
END PROCESS;
```

Blinken einer LED

- Zieltechnologie
  - ProASIC3: A3P060-VQ100
  - Core Cells: 1536
  - ext. Quarz: 100.0 MHz

- Syntheseresultate

Ressourcen	Beispiel2	Beispiel3	Beispiel4	Beispiel5
Core Cells	3358 (218.6%)	130 (8.4%)	123 (8.0%)	55 (3.6%)
Clock Frq.	4.9 MHz	106.2 MHz	118.8 MHz	122.1 MHz