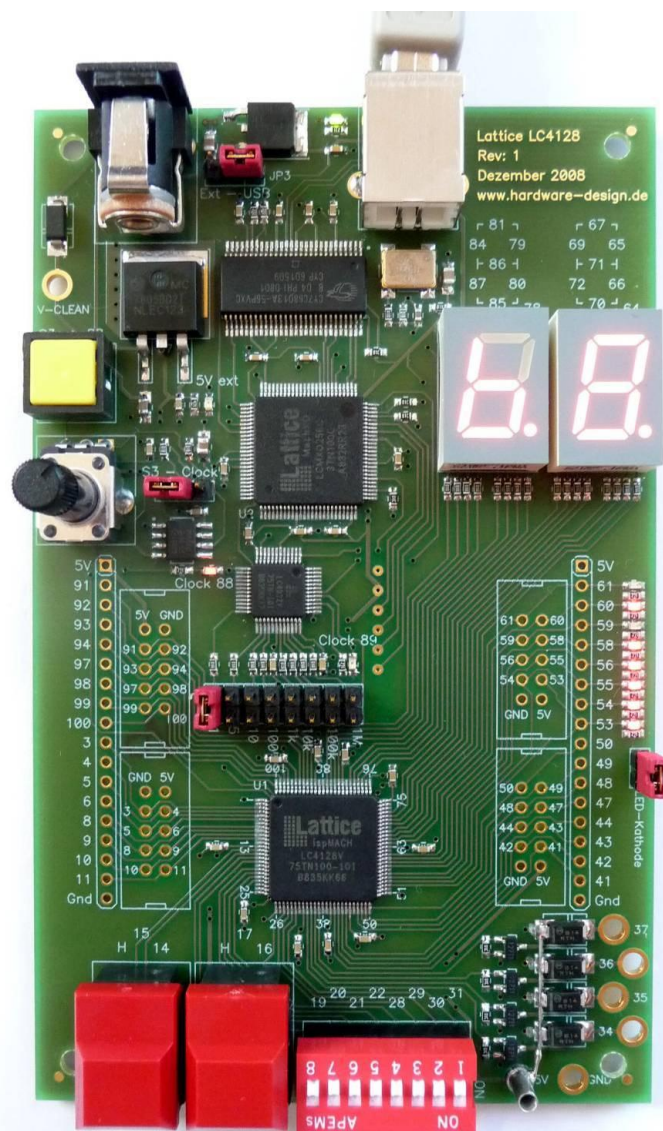


Formelsammlung zur Technischen Informatik

Programmierbare Logik

Lattice LC4128V



Karl Laber
Hohentwiel-Gewerbeschule Singen
Uhlandstr. 27
78224 Singen

Version 1.0
10.03.2020

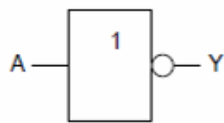
Inhaltsverzeichnis

| | |
|--|-------------|
| 1. Logische Grundgatter..... | S 2 |
| 2. Schaltwerke | S 3 |
| 3. ABEL-HDL | S 7 |
| 4. Software | S 9 |
| 4.1. ISPLeverProjectNavigator | S 9 |
| 4.2 LatticeDiamondProgrammer | S 12 |

1. Logische Grundgatter

Hinweis: Die jeweils erste Zeile entspricht der ABEL-HDL Syntax!

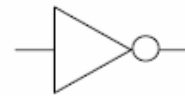
NOT (Negation)



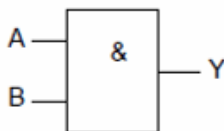
$Y = !A$
 $Y = \text{NOT } A$
 $Y = /A$
 $Y = \overline{A}$
 $Y = \neg A$

| A | Y |
|---|---|
| 0 | 1 |
| 1 | 0 |

auch zulässig:



AND (Konjunktion)

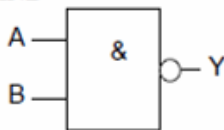


$Y = A \& B$
 $Y = A \text{ AND } B$
 $Y = A * B$
 $Y = A \wedge B$

| B | A | Y |
|---|---|---|
| X | 0 | 0 |
| 0 | X | 0 |
| 1 | 1 | 1 |



NAND

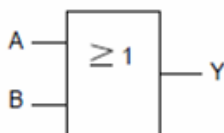


$Y = !(A \& B)$
 $Y = \text{NOT}(A \text{ AND } B)$
 $Y = /(A * B)$
 $Y = \overline{A \wedge B}$

| B | A | Y |
|---|---|---|
| 0 | X | 1 |
| X | 0 | 1 |
| 1 | 1 | 0 |



OR (Disjunktion)

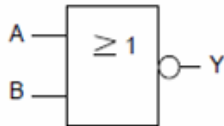


$Y = A \# B$
 $Y = A \text{ OR } B$
 $Y = A + B$ (nicht in ABEL)
 $Y = A \vee B$

| B | A | Y |
|---|---|---|
| 0 | 0 | 0 |
| X | 1 | 1 |
| 1 | X | 1 |



NOR

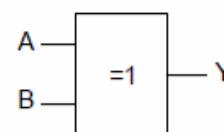


$Y = !(A \# B)$
 $Y = \text{NOT}(A \text{ OR } B)$
 $Y = /(A + B)$
 $Y = \overline{A \vee B}$

| B | A | Y |
|---|---|---|
| 0 | 0 | 1 |
| X | 1 | 0 |
| 1 | X | 0 |



XOR (Antivalenz)

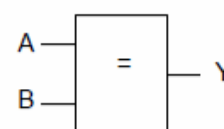


$Y = A \$ B$
 $Y = A \text{ XOR } B$
 $Y = A/B + /A/B$
 $Y = A \oplus B$

| B | A | Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

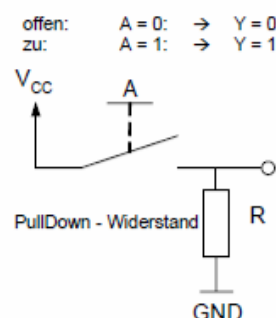
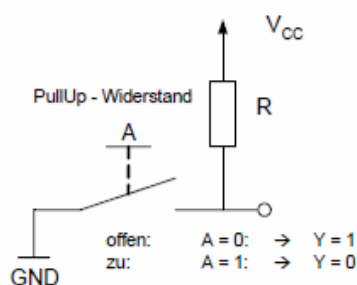


XNOR (Äquivalenz)

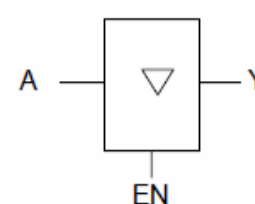


$Y = A !\$ B$
 $Y = A \text{ XNOR } B$
 $Y = AB + /A/B$
 $Y = \overline{A \oplus B}$

| B | A | Y |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |



Tristate



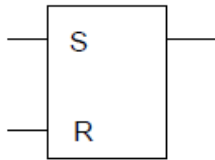
| EN | A | Y |
|----|---|---|
| 0 | 0 | Z |
| 0 | 1 | Z |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Z ≙ hochohmig

2. Schaltwerke

FlipFlop

RS-FlipFlop (statisch)



Zustandsfolgetabelle

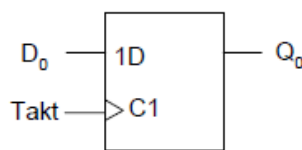
| S | R | Q^n | Q^{n+1} | |
|---|---|-------|-----------|----------------|
| 0 | 0 | 0 | 0 | speichern |
| 0 | 0 | 1 | 1 | |
| 0 | 1 | 0 | 0 | Reset |
| 0 | 1 | 1 | 0 | |
| 1 | 0 | 0 | 1 | Set |
| 1 | 0 | 1 | 1 | |
| 1 | 1 | ? | ? | nicht sinnvoll |

$$Q^{n+1} = (Q^n \& !R \# !R \& S)$$

$$Q^{n+1} = S \# !R \& Q^n \quad (\text{Minimalform})$$

in ABEL® steht kein zustandsgesteuertes SR-FF zur Verfügung

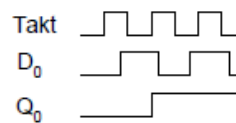
D-Flip-Flop (dynamisch)



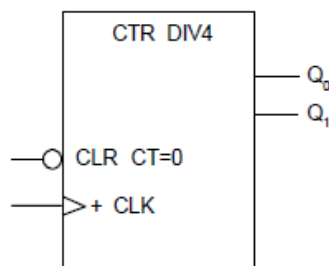
| C | D | Q^{n+1} |
|-----|---|-----------|
| 1 | X | Q^n |
| 0 | X | Q^n |
| pos | 0 | 0 |
| pos | 1 | 1 |
| neg | X | Q^n |

pos $\hat{=}$ positive Taktflanke
neg $\hat{=}$ negative Taktflanke

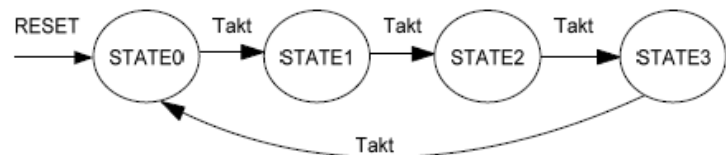
Impulssdiagramm



Zähler (Blockschaltbild)



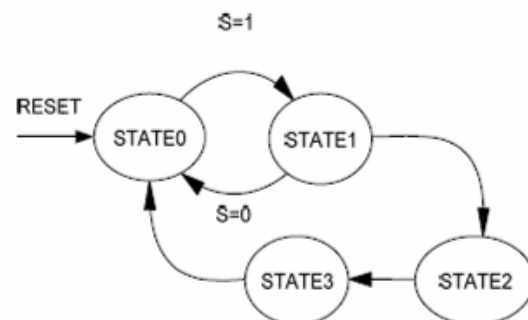
Zustandsdiagramm



- wenn es sich offensichtlich um ein getaktetes Schaltwerk handelt, kann die Angabe des Taktes als Übergangskriterium entfallen.
- Zusätzliche Übergangsbedingungen müssen angegeben werden

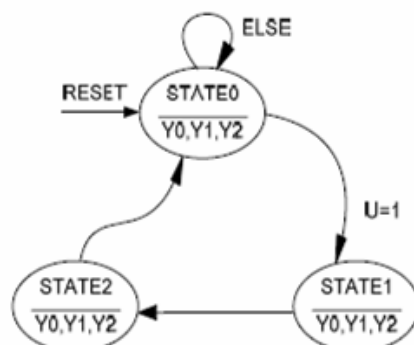
Zustandskodierung

| Zustand | Codierung $Q1, Q0$ |
|---------|-----------------------|
| STATE0 | 00 |
| STATE1 | 01 |
| STATE2 | 10 |
| STATE3 | 11 |



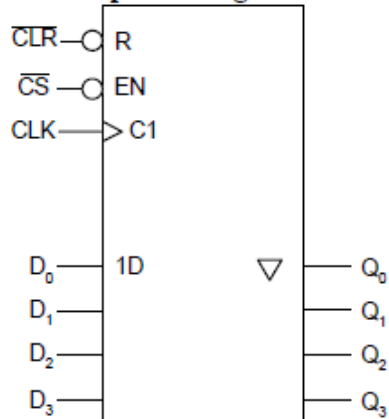
Codierte Zustandsübergangstabelle
(Codierte Zustandsfolgetabelle)

| | n | | n+1 | |
|---|----|----|-----|----|
| U | Q1 | Q0 | Q1 | Q0 |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| x | 0 | 1 | 1 | 0 |
| x | 1 | 0 | 0 | 0 |



- die Angabe von Ausgabewerten in einem Zustand werden durch einen (Unter-)Strich vom Zustandsnamen getrennt (die „else“-Angabe kann entfallen).

Speicherregister

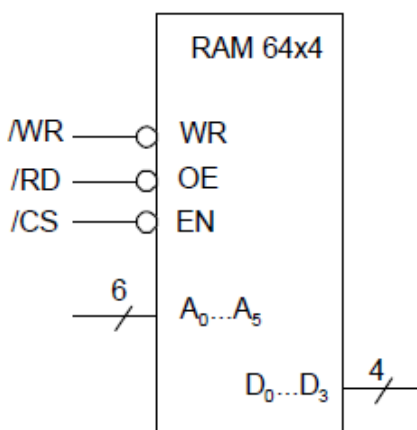


- 4-Bit Speicherregister (4 flankengesteuerte D-Flipflops)

- Parallele Ein- und Ausgabe

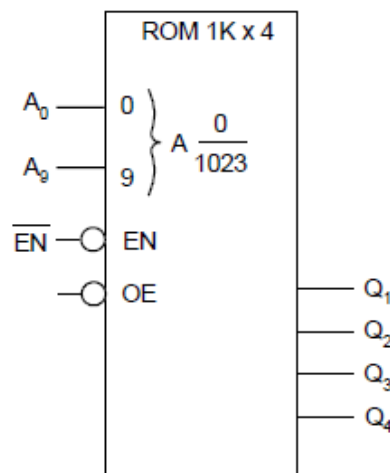
- Wenn der Baustein ausgewählt ist ($EN = 0$), werden mit der ansteigenden Flanke des Takt-Signals die an den Eingängen $D_0 \dots D_3$ anstehenden Daten übernommen.
- Mit einem 0-Signal am $\neg CLR$ -Eingang kann das Register gelöscht werden.
- EN ermöglicht, die Ausgänge in Tri-State zu schalten ($EN=1$) oder den Speicherinhalt auszulesen ($EN=0$)

RAM



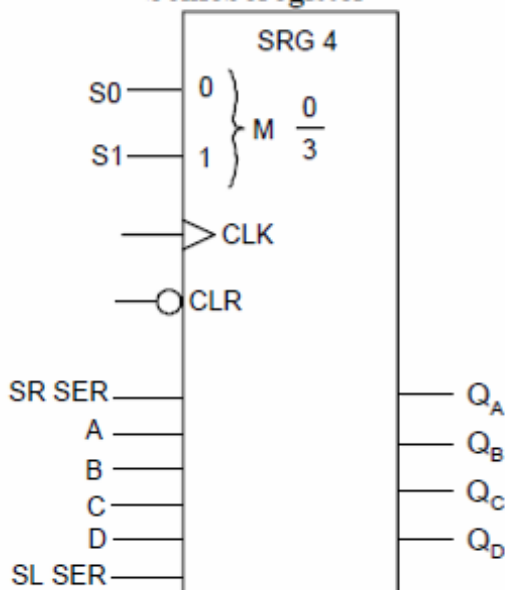
Schreib- Lesespeicher mit 64 x 4 Bit

ROM



- Read Only Memory
- mit den Adressen 0 ... 1023
- einer Wortbreite von 4 Bit
- und 1 Freigabeeingang

Schieberegister

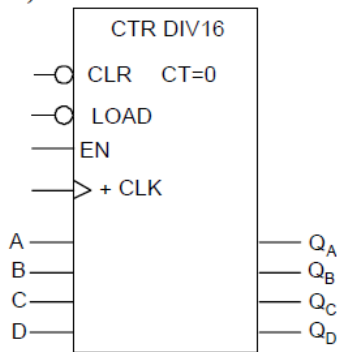


- 4-Bit Schieberegister
- Links- Rechtsbetrieb:

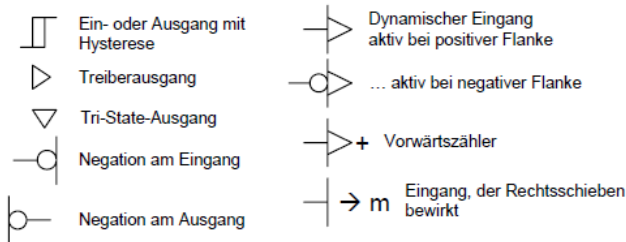
| Mode | S1 | S0 | Funktion |
|------|----|----|-------------------|
| 0 | 0 | 0 | - |
| 1 | 0 | 1 | rechts |
| 2 | 1 | 0 | links |
| 3 | 1 | 1 | parallele Eingabe |

- mit serieller Eingabe
- paralleler Ausgabe
- Vorwärtsschieben mit der positiven Taktflanke und der Möglichkeit, das gesamte Register zu löschen

Zähler (4-Bit)



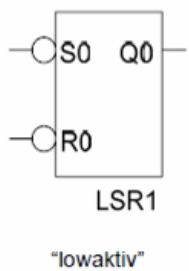
- $CTR \hat{=}$ Zähler
- $DIV\ 16 \hat{=}$ 16 verschiedene binäre Zustände
- Vorwärtszähler (+)
- $EN = 1$ und die positive Taktflanke führen zum nächsten Zählzustand
- Mit /LOAD kann ein Anfangszustand geladen werden



MUX / DX Multiplexer / Demultiplexer
 CTR m Zähler mit m Bits / Zykluslänge = 2^m
 CTR DIV m Zähler mit Zykluslänge m
 SRG m Schieberegister mit m Bits

Spezielle Schaltwerke der ISP-Logik

SR-FF (statisch)

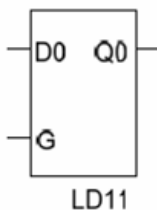


$Q.S0 = IS$
 $Q.R0 = IR$

| IS | IR | Q^{n+1} |
|----|----|-----------|
| 0 | 0 | Q^n |
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 1 | .* |

* nicht definiert

D-FF (statisch)

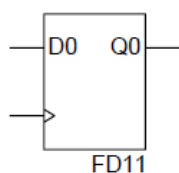


$Q.D = Dat$
 $Q.G = EN$

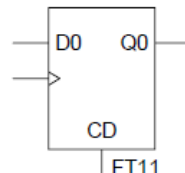
| EN | Dat | Q^{n+1} |
|----|-----|-----------|
| 0 | X | Q^n |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

D-Latch mit asynchronen Setz- und Rücksetzeingängen
 equations

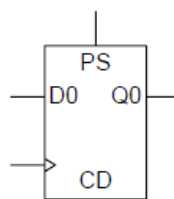
$Q.D = D_0$
 $Q.LH = G$
 $Q.ASET = S$
 $Q.ACLR = R$



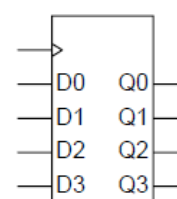
D-FF mit dynamischem Takteingang (pos. Flanke)



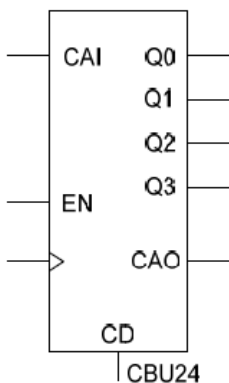
... zusätzlichem asynchronen Reset (ClearDevice)



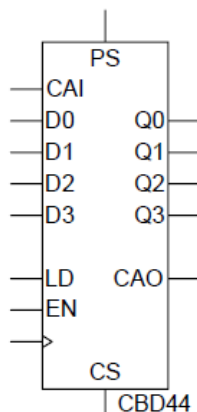
... zusätzlich synchronem Setzen (PreSet)



4 - D-FF (4-Bit-Register)



Vorwärtszähler mit CarryIn und CarryOut



CBD44

- Rückwärtszähler (CounterBinaryDown)
- Ein- und Ausgangsübertrag CAI / CAO (kaskadierbar)
- Enable (EN)
- Load (LD) - beliebiger Startwert möglich
- CS (synchrones Rücksetzen / clear)
- PS (synchrones Setzen)

| Input | | | | | | | Output | |
|-------|----|----|---|----|-----|------------|---------------|---------------|
| PS | CS | LD | D | EN | CAI | CLK | Q | CAO |
| 1 | x | x | x | x | x | \uparrow | 1 | 0 |
| 0 | 1 | x | x | x | x | \uparrow | 0 | ²⁾ |
| 0 | 0 | 1 | d | x | x | \uparrow | d | ³⁾ |
| 0 | 0 | 0 | x | 0 | x | x | Q | 0 |
| 0 | 0 | 0 | x | x | 0 | x | Q | 0 |
| 0 | 0 | 0 | x | 1 | 1 | \uparrow | ¹⁾ | ⁴⁾ |

¹⁾
²⁾
³⁾
⁴⁾

count down
 $CAO = CAI \& EN$
 $CAO = CAI \& EN \& ID0 \& ID1 \& ID2 \& ID3$
 $CAO = 1$ nach dem letzten Zählschritt

The *Header* Section

- **Module** (required)
- Interface (lower level, optional)
- Title

The *Declarations* Section

- Declarations Keyword
- Device Declaration
- Hierarchy Declarations
- **Signal** Declarations
- **Constant** Declarations
- Symbolic State Declarations
- Macro Declarations
- Library Declarations

The *Logic* Description Section

- Dot Extensions
- **Equations**
- **Truth Tables**
- **State Descriptions**
- Fuses Declarations
- XOR Factors

The Test Vectors Section

- Test Vectors
- Trace Statement

The *End* Statement

- Keyword: **end**

Beispiel

```

module szd07vr;

Takt          pin;
u,QC,QB,QA    pin istype 'com,dc';

equations
  QC.Clk = Takt;
  QB.Clk = Takt;
  QA.Clk = Takt;

truth table
  ([u,QC,QB,QA] :> [QC,QB,QA])

  [0, 0,0,0 ] :> [0,0,1];
  [0, 0,0,1 ] :> [0,1,0];
  [0, 0,1,0 ] :> [0,1,1];
  [0, 0,1,1 ] :> [1,0,0];
  [0, 1,0,0 ] :> [1,0,1];
  [0, 1,0,1 ] :> [1,1,0];
  [0, 1,1,0 ] :> [1,1,1];
  [0, 1,1,1 ] :> [0,0,0];

  [1, 0,0,0 ] :> [1,1,1];
  [1, 0,0,1 ] :> [0,0,0];
  [1, 0,1,0 ] :> [0,0,1];
  [1, 0,1,1 ] :> [0,1,0];
  [1, 1,0,0 ] :> [0,1,1];
  [1, 1,0,1 ] :> [1,0,0];
  [1, 1,1,0 ] :> [1,0,1];
  [1, 1,1,1 ] :> [1,1,0];

end

```

TRUTH_TABLE(*inputs* -> *outputs*) invalues -> outvalues ; Funktionstabelle
 or
TRUTH TABLE(*inputs* :> *reg outs*) invalues, reg values:> reg values ; Zustandsübergangstabelle

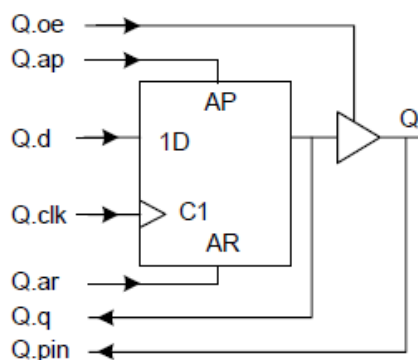
inputs Input signal names to the logic function.

outputs Output signal names from the logic function.

reg_outs Registered (clocked) output signal names.

- > Indicates the input to output function for combinational outputs.

:> Indicates the input to output function for registered outputs.



Detailed Dot Extensions

| D-Flipflop mit | Dot Ext. | Function |
|---|----------|--------------------------|
| • asynchronen Setz- und Rücksetzeingängen. | .AP | Asynchronous Preset |
| | .AR | Asynchronous Reset |
| | .CLK | Clock Input |
| • Der Möglichkeit, den Ausgang hochohmig zu schalten (TriState) | .D | D Flip-flop |
| | .LE | Latch Enable |
| | .LH | High Latch Enable |
| | .PIN | PIN Feedback |
| | .OE | Output Enable (TriState) |

3. ABEL-HDL

Beispiel: Schaltnetzentwurf

MODULE Druckbehaelter

TITLE 'Druckbehaelter'

//Name:

//Klasse:

//Datum:

//Aufgabe: Druckbehälter mit 3 Sensoren

DECLARATIONS

//Eingangssignale

S1..S3 PIN 29..31; //3 DIP-Schalter rechts

//Ausgangssignale

A1,W1 PIN 61,53 ISTYPE'Buffer,com'; //A=obere LED , W=untere LED

A2,W2 PIN 67,70 ISTYPE'Buffer,com';

TRUTH_TABLE //Wahrheitstabelle

([S3,S2,S1]->[A1,W1]);

[0,0,0]->[0,0];

[0,0,1]->[0,1];

[0,1,0]->[0,1];

[0,1,1]->[1,1];

[1,0,0]->[0,1];

[1,0,1]->[1,1];

[1,1,0]->[1,1];

[1,1,1]->[1,0];

EQUATIONS

A2= (S1&S2&!S3)#(S1&!S2&S3)#(!S1&S2&S3)#(S1&S2&S3);

W2=!((!S1 & !S2 & !S3)#(S1 & S2 & S3));

END

Beispiel: Schaltwerksentwurf**MODULE Effektbeleuchtung****TITLE 'Effektbeleuchtung'****//Name:****//Klasse:****//Datum:****//Aufgabe: Effektbeleuchtung Abi-Aufgabe****DECLARATIONS****//Eingangssignale****Takt PIN 89; //1Hz****S PIN 31; //S=0 Betriebsart 1, S=1 Betriebsart 2****//Zwischensignale****Q0..Q2 NODE ISTYPE'Buffer,reg';//Speicherausgänge SW****//Ausgangssignale****L0..L5 PIN 53..56,58,59 ISTYPE'Buffer,com';//SN bzw. Codierer****x=.X.; //don't care****EQUATIONS****Q0.clk=Takt;****Q1.clk=Takt;****Q2.clk=Takt;****TRUTH_TABLE //Zustandsfolgetabelle SW****([S,Q2,Q1,Q0] => [Q2,Q1,Q0]);****[1,x ,x ,x] => [0 ,0 ,0];****[0,0 ,0 ,0] => [****TRUTH_TABLE //Wahrheitstabelle Codierer SN****([Q2,Q1,Q0] -> [L5,L4,L3,L2,L1,L0]);****[0 , 0 , 0] -> [****END**

4. Software

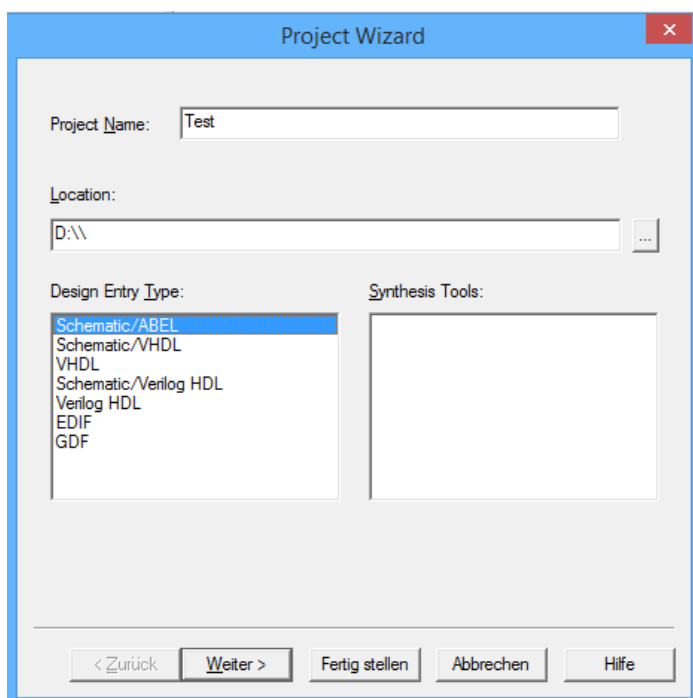
4.1 ISPLeverProjectNavigator



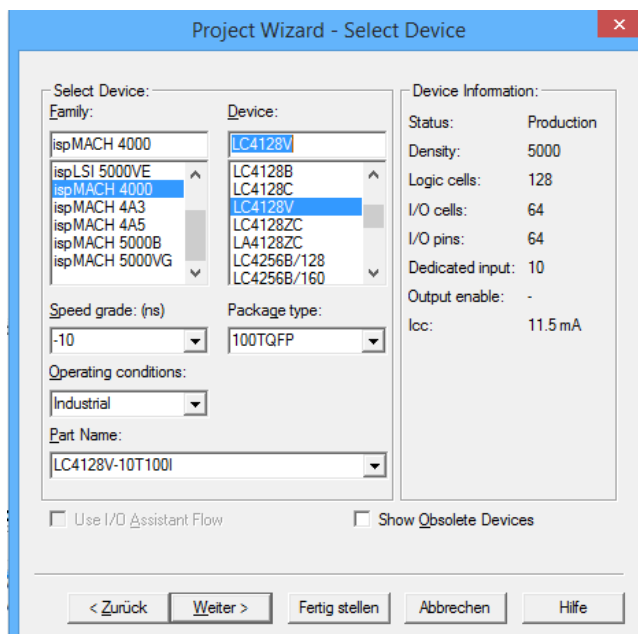
Software zum

- Anlegen eines Projektes
- Beschreibung der Schaltung u.a. in ABEL-HDL
- Erzeugung einer JEDEC-Files

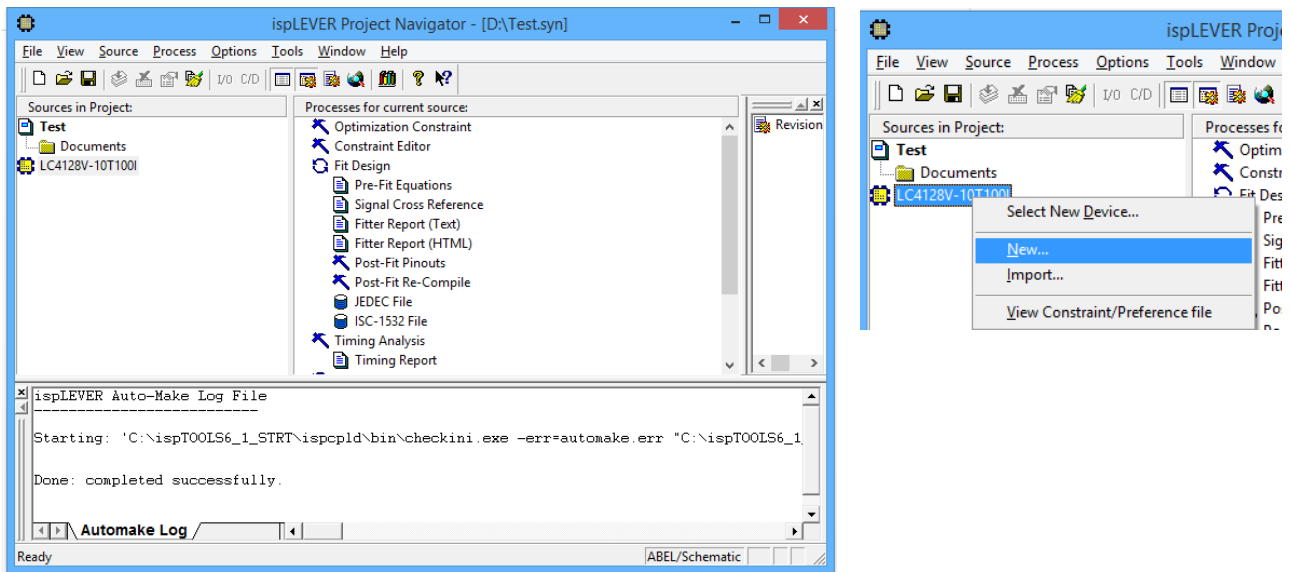
- a) File → New Project
- b) Projekt Name und Location angeben
- c) Schematic ABEL wählen, dann <Weiter>



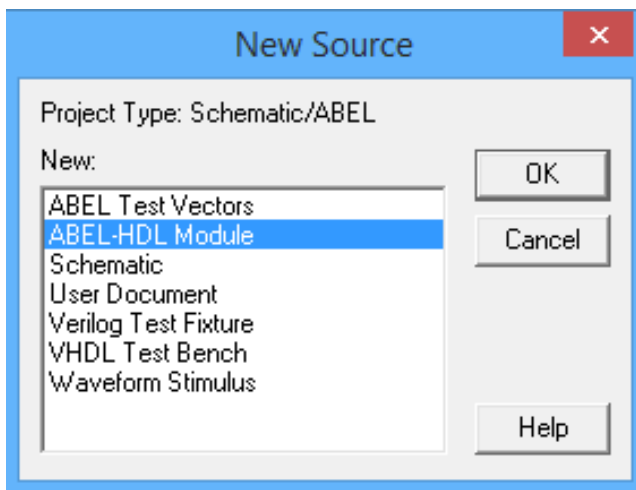
- d) Family: ispMach 4000 und Device: LC4128V selektieren, dann <Fertig stellen>



e) Mit der rechten Maustaste auf **LC4128V-10T100** klicken → **New...** wählen



f) **ABEL-HDL Module** wählen

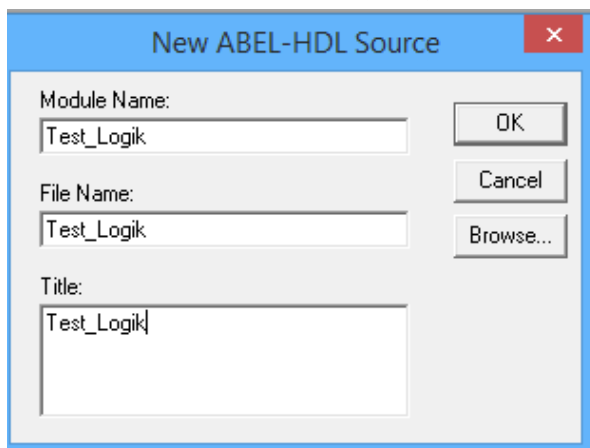


g) Dreimal die gleiche Bezeichnung eingeben

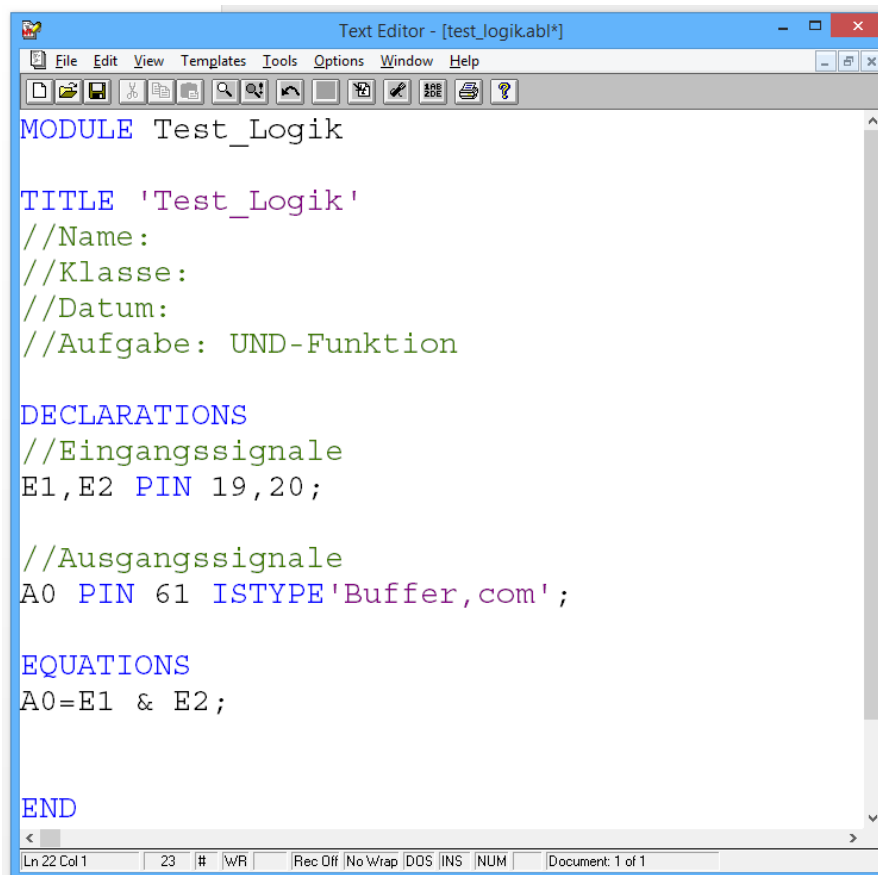
Beachte: Keine Umlaute (ä, ö, ü)

nicht mit einer Zahl beginnend

zusammenhängende Zeichenkette ohne Minus



- h) Eingabe der Aufgabenstellung im Editor (vgl. Kap. 3 ABEL-HDL), dann speichern



```

MODULE Test_Logik

TITLE 'Test_Logik'
//Name:
//Klasse:
//Datum:
//Aufgabe: UND-Funktion

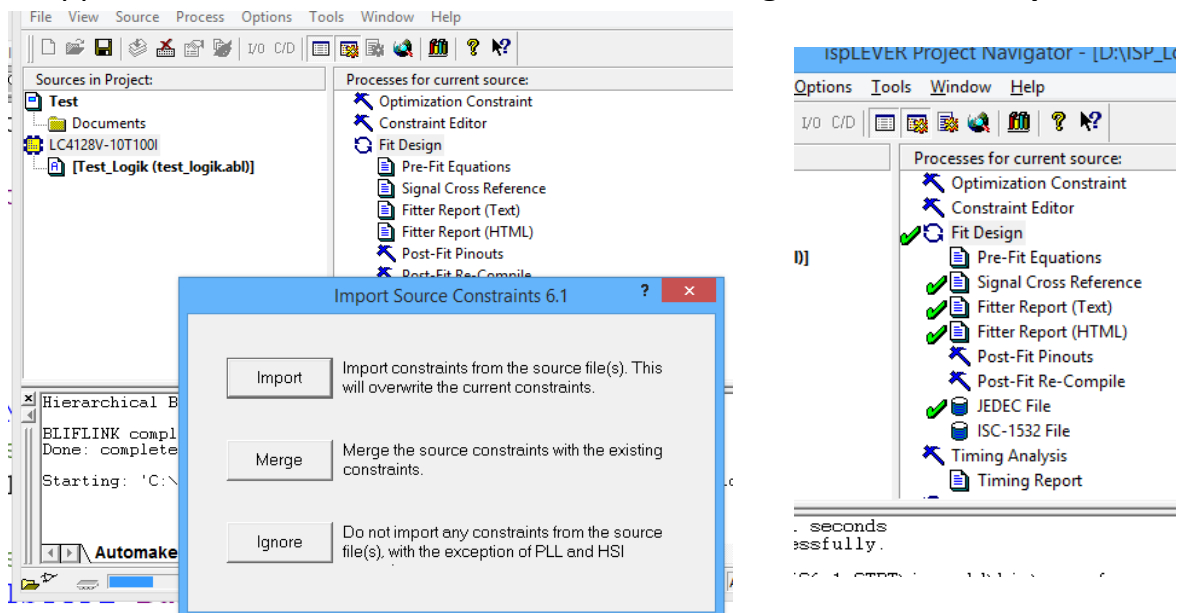
DECLARATIONS
//Eingangssignale
E1,E2 PIN 19,20;

//Ausgangssignale
A0 PIN 61 ISTYPE'Buffer,com';

EQUATIONS
A0=E1 & E2;

END
  
```

- i) Doppelklick mit der linken Maustaste auf **Fit Design** → danach **<Import>** wählen



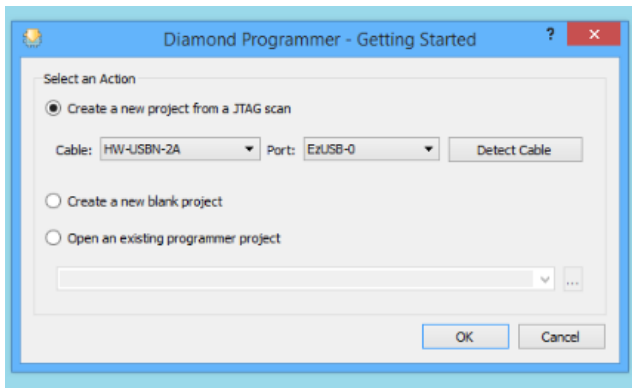
- j) Danach sollte bei dem Eintrag **JEDEC-File** ein grünes Häkchen erscheinen. Ansonsten entweder den/die Fehler beseitigen oder noch einmal auf den Eintrag JEDEC-File doppelklicken. In entsprechenden Pfad (Location) sollte nun eine Datei mit der Endung **.jed** vorliegen.

4.2 LatticeDiamondProgrammer

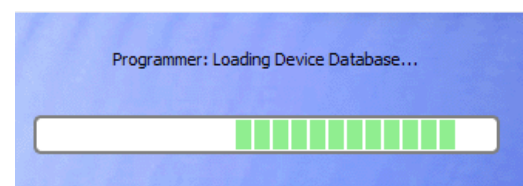


Software zum Downloaden der JEDEC-Datei
auf das Zielsystem (Hardware).

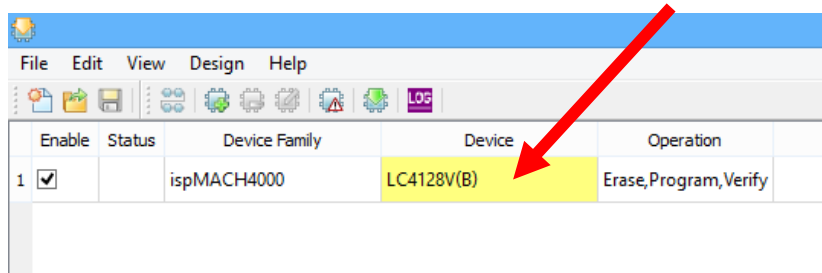
- a) Hardware mit der USB-Schnittstelle des PC verbinden, dann mit <OK> bestätigen



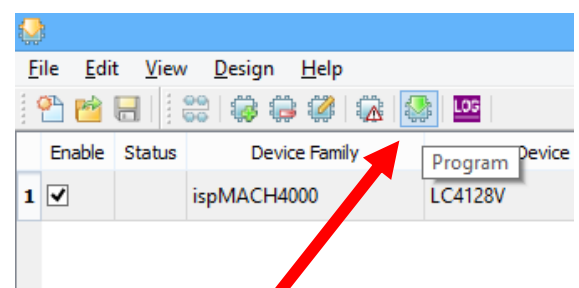
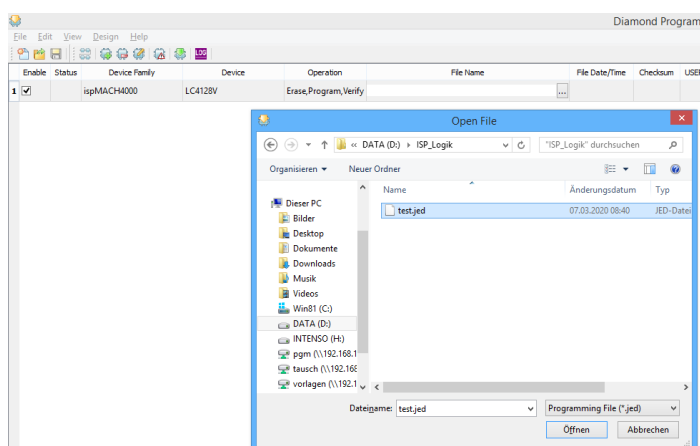
Scan-Prozess abwarten



- b) Korrekte **Device** einstellen → LC4128V ohne (B) !!!



- c) Bei **File Name...** in dem entsprechenden Verzeichnis die Jedec-Datei wählen.



- d) Mit der Aktivierung des Icons **Program** (grüner Pfeil nach unten) den Download Prozess starten. Danach sollte die Schaltung auf dem IC verifiziert sein.