

DesignExpert 8 Kurzeinführung

Jedes Projekt kann aus mehreren Dateien bestehen:

Im Projektnavigator (Abb. 2) werden alle zum Projekt gehörenden Dateien verwaltet. Eine Datei kann in mehrere Projekte einbezogen werden. Oft wird ein Projekt jedoch nur eine Quelldatei als Text oder Zeichnung enthalten.

Wichtig: Die zum Board überspielte Datei trägt den Namen des **Projektes** mit der Endung .jed (Jedec-Datei)

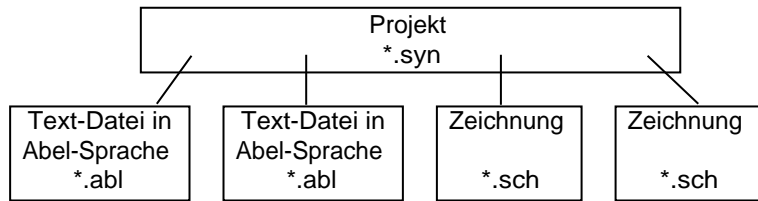


Abbildung 1

1. Projekt (*.syn) und Datei (*.abl) anlegen

File -> New Projekt, Projektnamen vergeben

Doppelklick, Projektname oder Beschreibung

Doppelklick, Bauteilname aus Liste suchen

Source -> New
Abel-HDL Module für Texteingabe in Abel-Sprache
Schematic für Schaltung zeichnen
Module Name und *File Name* gleich wählen
Titel: nicht eingetrag!!!

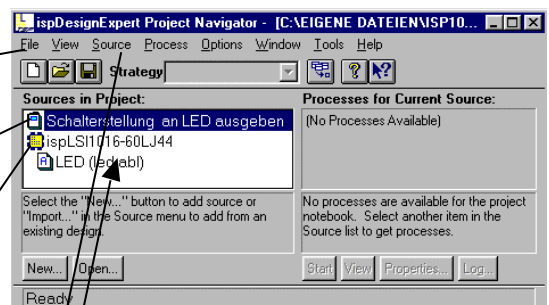


Abbildung 2

Hinweis zur Namensvergabe:
 kein Lehrzeichen, kein Umlaut,
 kein -, kein _ ,
 lange Dateinamen erlaubt

2. Eingabe in der Abel-Sprache

Aufruf des Editors aus dem Project Navigator :
 Doppelklick auf

Schlüsselwörter, die ganz bestimmte Funktionen haben, werden automatisch blau dargestellt und können groß oder klein geschrieben werden (Im Beispiel rechts groß geschrieben)

Zur besseren Übersicht sollte der Programmtext etwas eingerückt stehen. Bei Pin-Namen und Variablen-Namen wird zwischen Groß- und Kleinschreibung unterschieden! Schalter ≠ schalter, LED ≠ led.

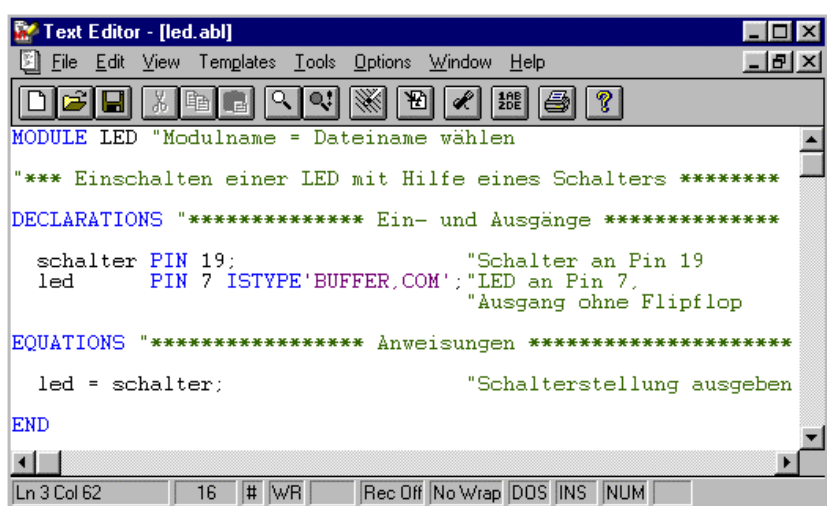


Abbildung 3

Kommentare beginnen mit " oder // und werden automatisch grün dargestellt

Alle diese Angaben sind Teil der ABEL-Hochsprache (HDL = High-Level Description Language).

3. Übersetzen (Compilieren)

Erstellte Abel-Datei anklicken, dann wird im rechten Fenster dargestellt, welche Aktionen mit dieser Datei durchgeführt werden können:

Zum Übersetzen Doppelklick
Bei Fehlern:
Fehlertext lesen
Datei wieder öffnen
(Doppelklick links auf Dateiname)

Wenn fehlerlos erscheint ein grüner Haken vor Compile Logic

Logik minimieren

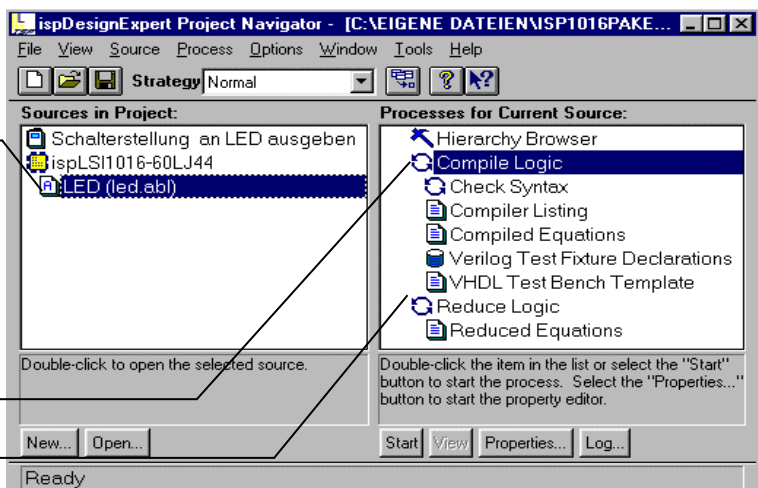


Abbildung 4

4. Logik auf den Baustein anpassen

auf den Bauteilnamen klicken rechts erscheinen die möglichen Aktionen:

Projekt aus einzelnen Dateien zusammensetzen.

Projekt auf den gewählten Baustein anpassen.

Programmieren des Bausteins (Herunterladen)

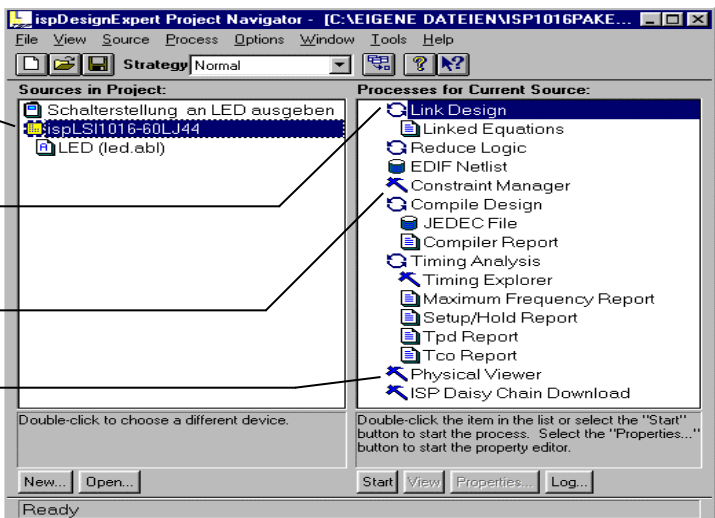


Abbildung 5

5. Baustein programmieren (Download)

Board anschließen und Verbindung zum PC herstellen. mit Scan-Button feststellen, welche Bausteine über das Downloadkabel erreichbar sind.

Diese Meldungen erscheinen

Jedec-Datei suchen, die den Baustein programmieren soll.

Run Operation:
Download ausführen.
(oder: Command -> Run Operation)

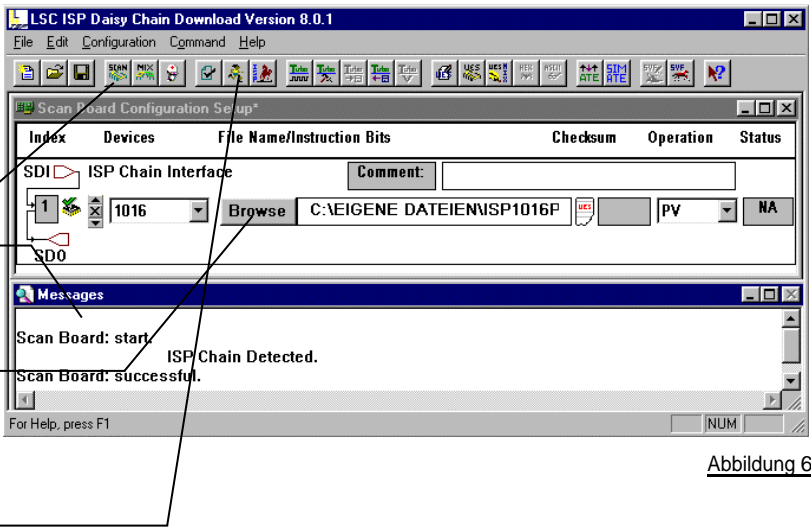


Abbildung 6

6. Funktionale Simulation (Beispiel: UND-Verknüpfung)

Zunächst sind unter dem Abel-Text sogenannte Testvektoren anzugeben:

```

MODULE UND "**** UND-Verknüpfung zweier Schalter ****"
DECLARATIONS "**** Angabe der Ein- und Ausgänge ****"
    ein2, ein1 PIN 19,21; "Eingänge: Schalter
    aus PIN 3 ISTYPE 'BUFFER.COM'; "Ausgang ohne Flipflop: LED
EQUATIONS "**** logische Verknüpfungen ****"
    aus = ein2 & ein1; "UND-Verknüpfung
TEST_VECTORS "**** Simulations-Vektoren ****"
    ([ein2, ein1] -> aus); "Festlegung der zu untersuchenden
                            "Eingänge -> Ausgänge
                            "zu simulierende Kombinationen folgen
                            "beide Eingänge 0 -> Ausgang don't care .x.
                            " usw.
    [ 0 , 0 ] -> .x ;
    [ 0 , 1 ] -> .x ;
    [ 1 , 0 ] -> .x ;
    [ 1 , 1 ] -> .x ;
END
    
```

Nach dem Abspeichern der Datei erkennt DesignExpert die Testvektoren und fügt im Projektnavigator die Möglichkeit der Simulation hinzu

Doppelklick: Testvektoren übersetzen

Doppelklick zur funktionalen Simulation ohne Beachtung der Gatterlaufzeiten.

Doppelklick zur Zeit-Simulation mit Beachtung der bausteinspezifischen Gatterlaufzeit

Simulation starten

Edit->Show um eventuell weitere Signale anzeigen zu lassen, die nicht automatisch dargestellt werden
Über Bus lassen sich mehrere Signale

Unter View finden sich Zoom-Möglichkeiten

Mit Hilfe der Timing-Simulation wurde eine Signal-Laufzeit vom Wechsel der Eingangssignale bis zur Änderung des Ausgangssignals von 20 ns gemessen.

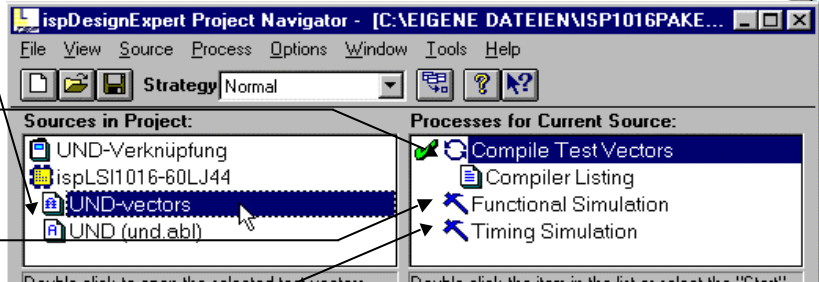


Abbildung 7

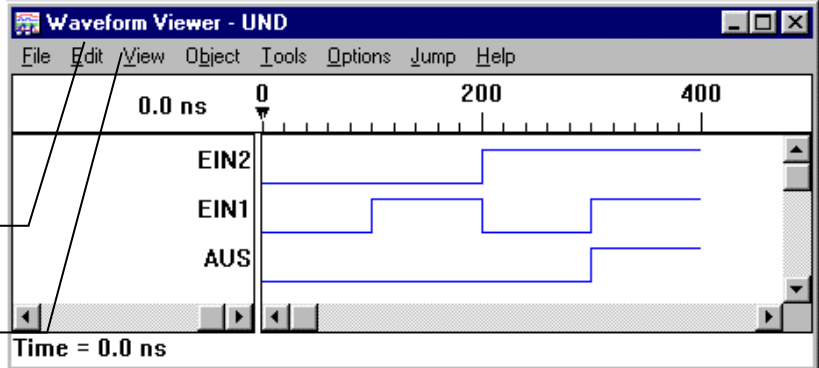
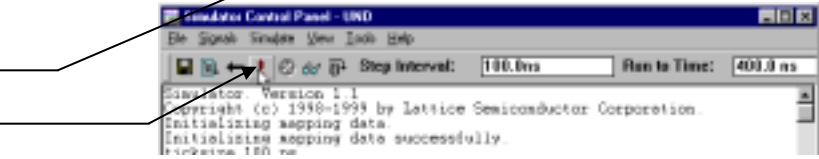
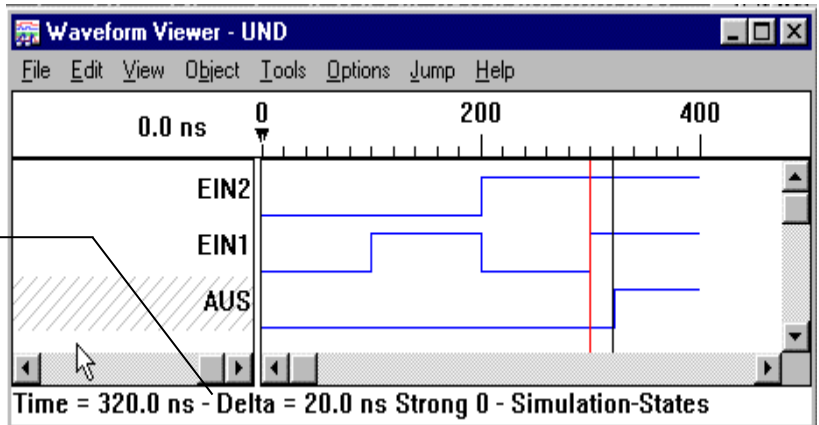


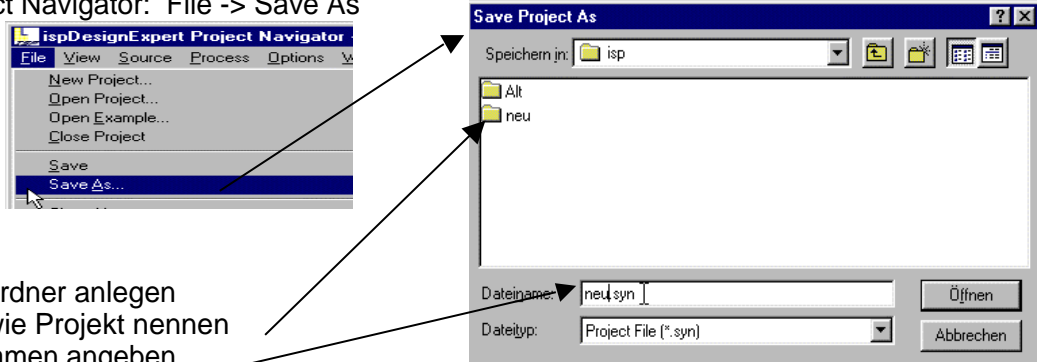
Abbildung 8



7. Vorhandenes Projekt unter neuem Namen speichern

Das Projekt „Alt“ wird unter dem Namen „Neu“ gespeichert.

- 1 Im Project Navigator: File -> Save As



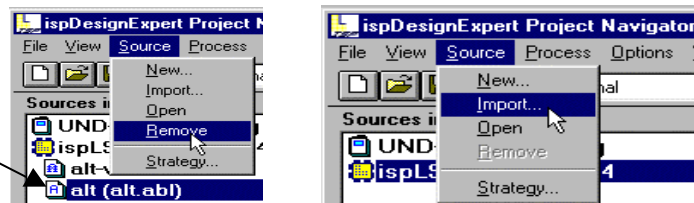
- 2 Neuen Ordner anlegen
Ordner wie Projekt nennen
Projektnamen angeben

Es folgen 2 Abfragen, die beide mit **Ja** beantwortet werden, daraufhin wird auch die „Abel-Datei“ *.abl mit ins neue Verzeichnis kopiert.

- 3 Abel-Datei umbenennen:
auf Name klicken

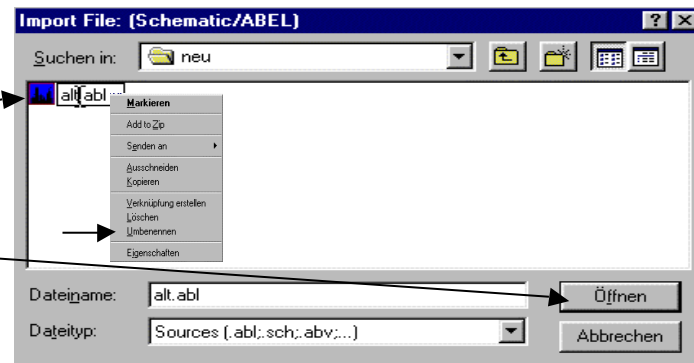
Source -> Remove

Source -> Import



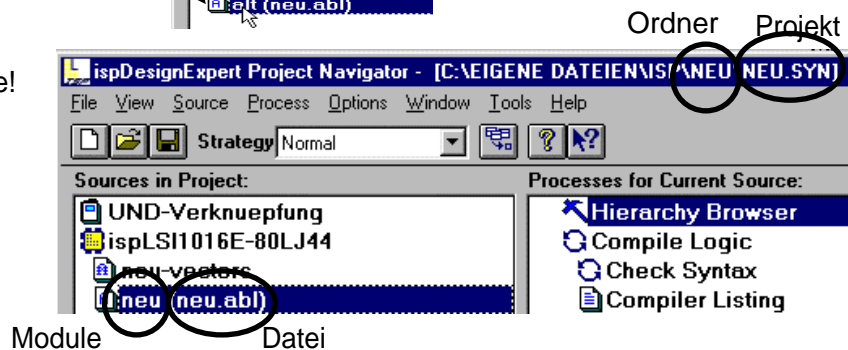
Im Öffnen (Import)-Fenster
Datei umbenennen nach
Klick mit rechter Maus auf
den Namen

umbenannte Datei öffnen



- 4 Nach Doppelklick auf
im Editor hinter Module auch
den neuen Namen eintragen
und Datei speichern.

Kontrolle: 4x der neue Name!



8. Grafische Eingabe mit selbst erzeugten Grafik-Symbolen (ab DesignExpert 8.1)

Ausgangspunkt: z.B. 3 Abel-Dateien (*.abl)
 (Pin-Nummern nicht nötig, stören aber auch nicht, DesignExpert nimmt später die Pins, die in der Zeichnung angegeben sind.)

```

MODULE bin7seg "Codeumschalter 8-4-2-1-Code -> 7-Segment-Code
DECLARATIONS "***** Ein- und Ausgänge *****
d8,d4,d2,d1 pin;
a,b,c,d,e,f,g pin ISTYPE 'BUFFER,COM'; "7-Segm-Anzeige
dua1zah1 = [d8,d4,d2,d1]; "Dualzah1
segment7 = [a,b,c,d,e,f,g]; "7-Segment-Anzeige
TRUTH_TABLE "***** Funktionstabelle 8-4-2-1-Code -> 7-Segment-Code *
(dualzah1) -> [a,b,c,d,e,f,g] "Eingänge -> Ausgänge
0 -> [1,1,1,1,1,1,0,1]; " LEDs high-aktiv!
1 -> [0,1,1,0,0,0,0,1]; " a
2 -> [1,1,0,1,1,0,1,1]; " ---
3 -> [1,1,1,1,0,0,1,1]; " f | g | b
4 -> [0,1,1,0,0,1,1,1]; " ---
5 -> [1,0,1,1,0,0,1,1]; " e | d | c
6 -> [1,0,1,1,1,1,1,1]; " ---
7 -> [1,1,1,0,0,0,0,1]; "
8 -> [1,1,1,1,1,1,1,1]; "
9 -> [1,1,1,1,0,1,1,1]; "
10 -> [1,1,1,0,1,1,1,1]; "A
11 -> [0,0,1,1,1,1,1,1]; "b
12 -> [0,0,0,1,1,1,0,1]; "c
13 -> [0,1,1,1,1,1,0,1]; "d
14 -> [1,0,0,1,1,1,1,1]; "E
15 -> [1,0,0,0,1,1,1,1]; "F
    
```

```

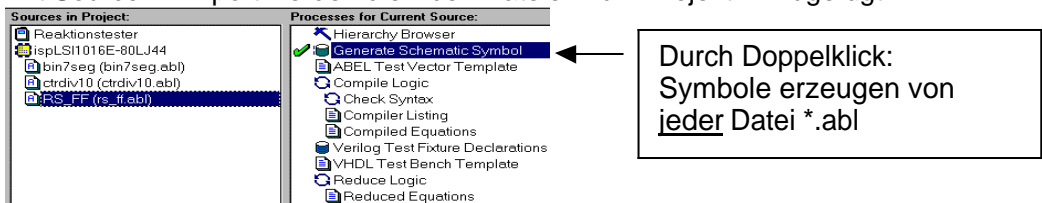
MODULE RS_FF "Highaktives RS-FlipFlop mit einem Ausgang
DECLARATIONS "***** Ein- und Ausgänge *****
S, R PIN ; "Eingänge RS-FlipFlop
Q PIN ISTYPE 'BUFFER,COM'; "FF-Ausgang Q
EQUATIONS
Q = IR & (S # 0); "rücksetzdominantes highaktives RS-FF
    
```

```

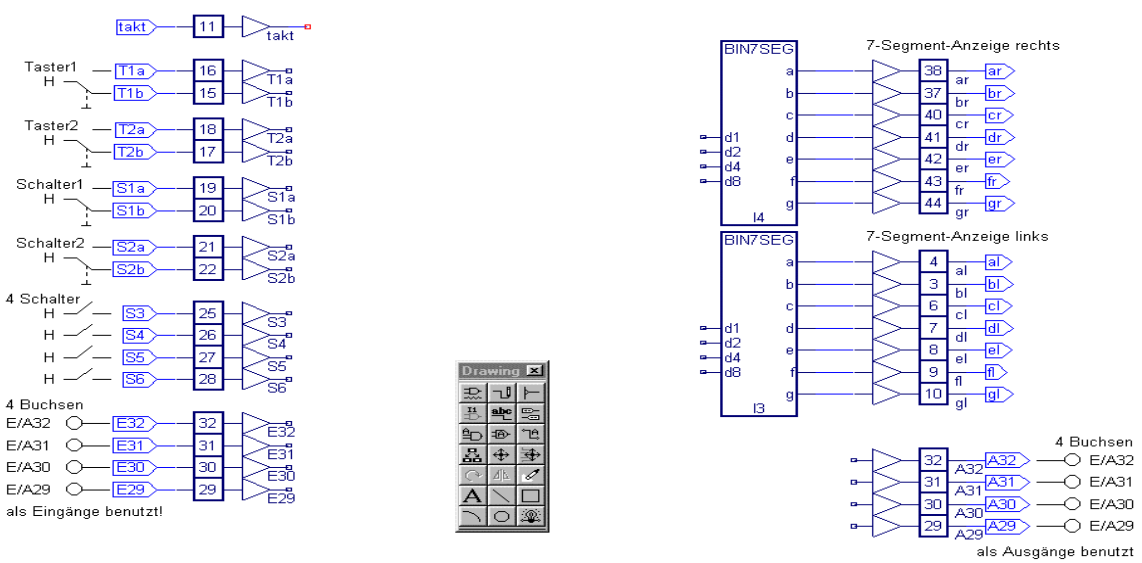
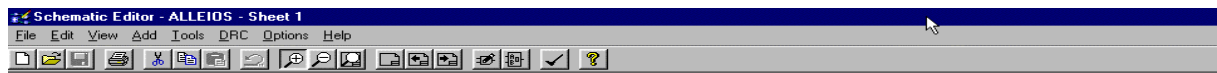
MODULE ctrdiv10 "Zähler 0 bis 9 mit Freigabe (= auch Übertragungsgang),
"Übertragungsausgang und asynchronem Reset
declarations
C,Freigabe,Reset pin; "Eingänge
01,02,04,08 pin istype 'buffer,reg'; "Ausgänge mit D-FF
CT9 pin istype 'buffer,com'; "Ausgänge ohne D-FF
Ctr = [08,04,02,01]; "Zähler aus 4 FFs
equations
when Freigabe then { when Ctr.q == 9 then Ctr.d = 0; "Rücksetzen bei nächsten Takt
else Ctr.d = Ctr.q +1; "Zählen
}
else Ctr.d = Ctr.q; "Stehenbleiben
Ctr.clk = C; "Takt für alle FlipFlops
when (Ctr.q == 9) & Freigabe then CT9 = 1 "Übertrag und Freigabe nächste Stelle
else CT9 = 0;
Ctr.ar = Reset; "asynchroner Reset
    
```

Im **Explorer**: Ordner für neues Projekt anlegen,
 gewünschte Dateien *.abl in den Ordner kopieren.
 Datei **alleIOs.sch** (alle Pins der Platine fertig gezeichnet, im Ordner alleIOs) in den Ordner kopieren und umbenennen, (so wie das Projekt heißt).

In **DesignExpert**: Projekt anlegen,
 mit Source -> Import werden die Abel- Dateien zum Projekt hinzugefügt

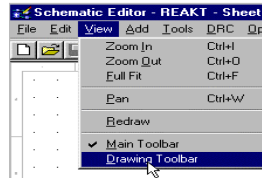


mit Source-> Import wird die Schematic-Datei (s.o.) hinzugefügt und durch Doppelklick geöffnet:

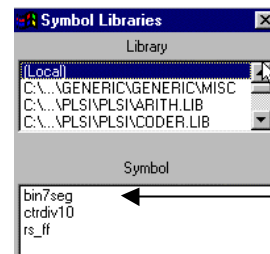
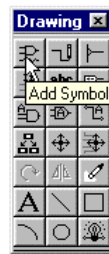


alle nicht benötigten Pins entfernen, sonst stürzt DesignExpert später ab!
 Entfernen: Edit->Cut oder Schere-Button, Fenster bei gedrückter Maus über zu entfernenden Teilen aufziehen oder Anklicken der Bauteile. (Rückgängig gibt's auch)

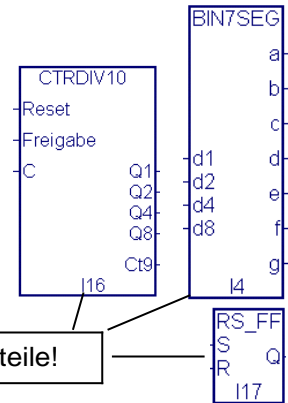
falls Drawing Toolbar nicht sichtbar:



Eigene Bauteile



holen:



Plazieren, eventuell verschieben fertig.



verbinden , speichern,



Im Projekt-Navigator wie gewohnt übersetzen und downloaden.

Die oben erzeugten Symbole sind in den Dateien *.sym im Projekt-Ordner gespeichert.

Eine eigene Bibliothek mit eigenen Bauteilen erzeugen sie wie folgt: (aus der Online-Hilfe)

Using Symbol Libraries in Your Design

Once you have created a symbol library, you can access it from the Schematic Editor. A library, in order to be accessible to the Schematic Editor, must be added to the search path of your project.

1. In the Project Navigator, choose Options _ All Schematic to open the INI Editor (Schematic Environment dialog). Then click the Symbol Paths tab.
2. Click Browse and go to the location of the library that you want to add. The path appears in the Path field.
3. To add the path to the symbol library search path, click Add. The path appears at the top of the search path list.
4. You can use the other buttons on his dialog to delete a path from the list, or to move a path up or down the search order.

Adding the Symbol to a Library

After you have saved the symbol, you can both leave it in the local design directory and add it to the schematic from this location, or you can add the symbol to a Symbol Library for use in other designs.

1. In the Project Navigator, choose Window _ Library Manager.
2. Choose File _ Open and open the library.
3. Choose Edit _ Add Symbol to open the Add Symbols to Library dialog.
4. Go to the location of the symbol that you want to add and select it. Then click Open. ispDesignEXPERT adds the symbol to the library.

9. Graphische Eingabe bei Synario und DesignExpert

Aufruf des Schematic-Editors vom Project Navigator: Source => New, Doppelklick auf Schematic Fenster Drawing: View->Drawing Toolbar

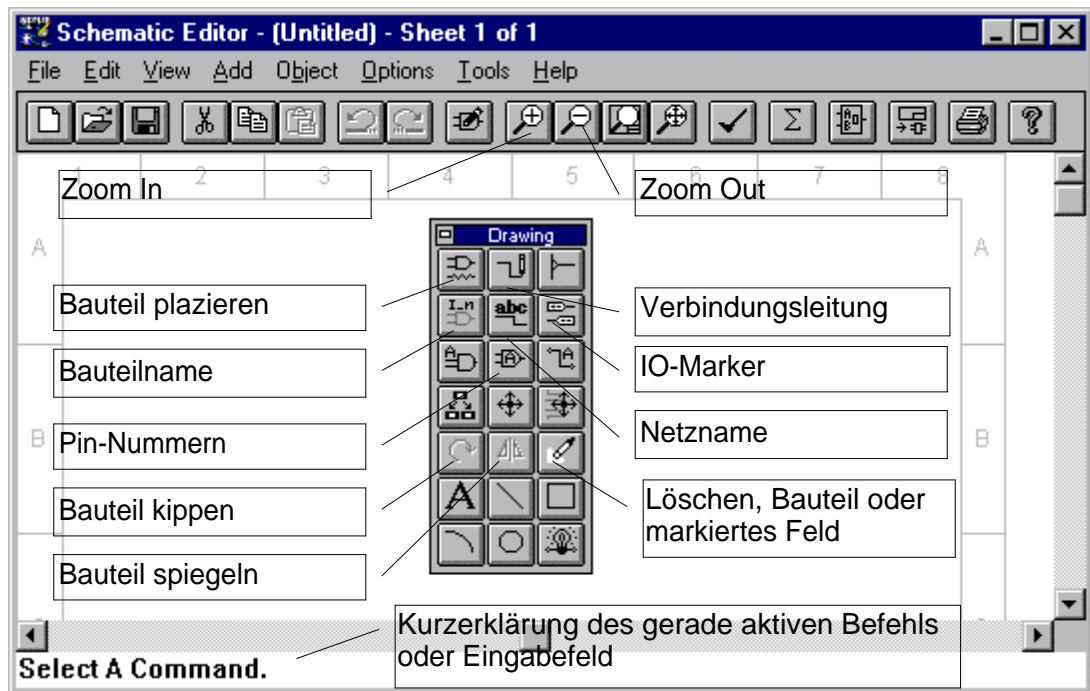


Abbildung 9

Vorgehensweise bei der graphischen Eingabe

Alle diese Schritte sind **in dieser Reihenfolge** nacheinander auszuführen.
Siehe auch Beispielschaltung Abbildung 11

1. Bauteile plazieren:

Add => Symbol, es öffnet sich nebenstehendes Fenster.
Bibliothek wählen, Bauteil wählen
Bauteile auf dem Blatt plazieren durch linken Mausklick.
Ende: rechter Mausklick.
Möglichst alle Bauteile an der richtigen Stelle plazieren, erst am Schluß verbinden.
Wichtig: Alle Verbindungen nach außen (IC-PINs) müssen mit IO-Pads versehen werden. (Bibliothek IOPADS)

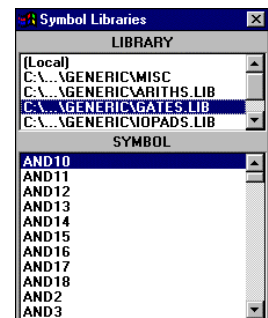


Abbildung 10

2. Verbindungsleitungen herstellen:

Add => Wire Klick Leitungsanfang, Leitung ziehen, Klick Leitungsende
Ende rechte Maustaste oder wenn Leitung am Bauteil endet.
Es dürfen keine offenen Leitungsenden entstehen.
(Ausnahme: Übergangsstellen zu Schaltungsteilen in anderen Dateien, die später mit einem IO-Marker versehen werden)
An die IOPADS sind von außen kurze Leitungsstücken zu hängen (weil sonst die spätere Beschriftung nicht richtig funktioniert.)

3. Bauteilnamen vergeben:

Add => Instance Name, in der unteren Fensterzeile erscheint der eingegebene Name
Name schreiben und mit der Maus auf das Bauteil plazieren.
Alle IOPADS müssen einen Namen bekommen andere Bauteile können einen erhalten.

4. Netznamen vergeben:

Add => Netname, in der unteren Fensterzeile erscheint der eingegebene Name.

Name schreiben und mit der Maus auf der Leitung plazieren.
 Alle Leitungsenden außerhalb der IOPADS müssen einen Namen erhalten, andere wichtige Leitungen können einen Namen bekommen.

5. I/O-Marker setzen:

Add => I/O Marker, Input oder Output wählen.

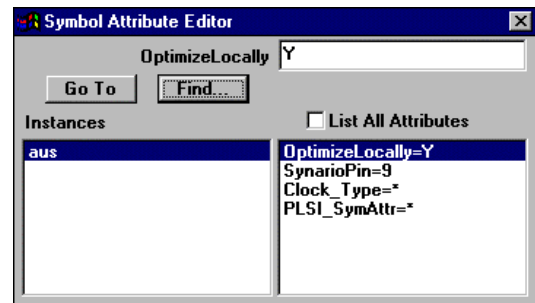
Mit der Maus an das Ende der Leitung, klick, und bei gedrückter Taste über die Beschriftung nach außen ziehen.

Es entsteht ein Pfeil  in dem der Name des Netzes steht.

6. PIN-Name vergeben:

Edit Symbol Attribute, es öffnet sich dieses Dialogfenster.

Auf das I/O-Pin in der Schaltung klicken, im linken Fenster erscheint der I/O-Pin-Name, rechts *SynarioPin* wählen, im Eingabefeld statt em Stern die gewünschte Pin-Nummer eingeben, nochmals auf das I/O-Pin klicken.



Nun könnte die Schaltung so aussehen:

Abbildung 12

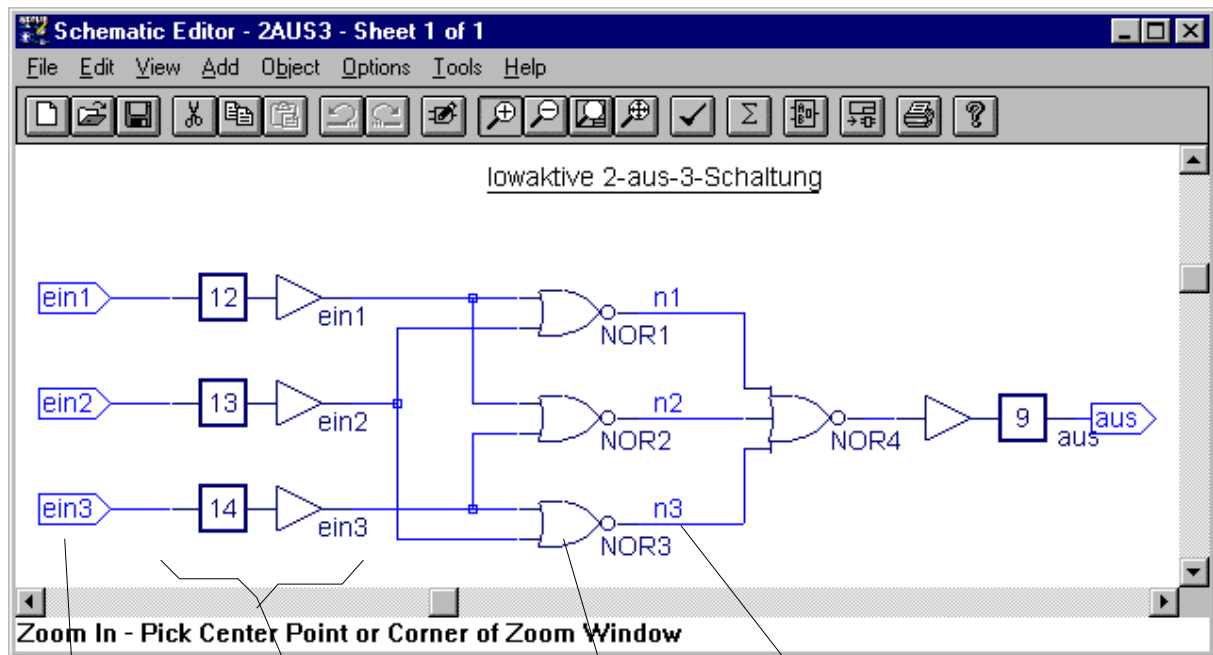
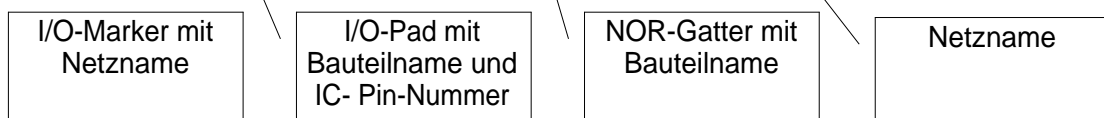


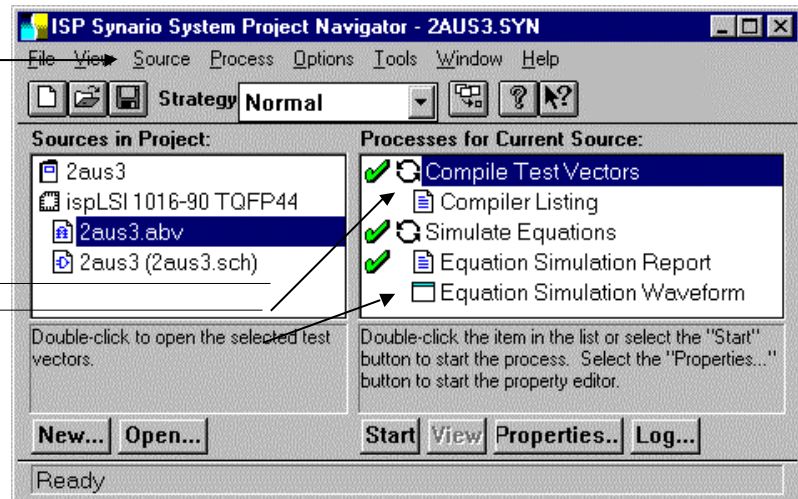
Abbildung 13



Editor verlassen, im Projekt-Navigator Schaltung compilieren. - Viel Glück und Ausdauer!

10. Funktionale Simulation bei grafischer Eingabe

- Schaltung zeichnen.
- *Source* => *New*
Abel Test Vectors wählen.
Dateinamen können gleich dem Dateinamen der Schaltung sein.
- Testvectors wie im Beispiel unten eingeben.
- Compilieren
- Ergebnis ansehen



Beispiel einer .abv Testvector-Datei für die in Abbildung 14 dargestellten Schaltung.

MODULE zweiaus3;

"Testvectors für die 2aus3-Schaltung"

DECLARATIONS "Namen und Pin-Nummern wie in der Zeichnung *****"

ein1, ein2, ein3 **PIN 12,13,14;** "Schalter und LED sind lowaktiv.
aus **PIN 9 ISTYPE 'COM';**

EQUATIONS

TEST_VECTORS "Testvectors für die Simulation. Diese können in dieser
"Test-Vector-Datei *.abv angegeben werden.

([ein1, ein2, ein3] -> aus); "3 Eingänge, 1 Ausgang.
@const i=0; "Festlegung der Zählvariablen
@repeat 16 { i -> .x.; "in { } folgendes 16 mal wiederholen
@const i = i+1; } "dabei Zählvariable um 1 erhöhen

END

