1 Entwicklungsablauf mit Quartus II

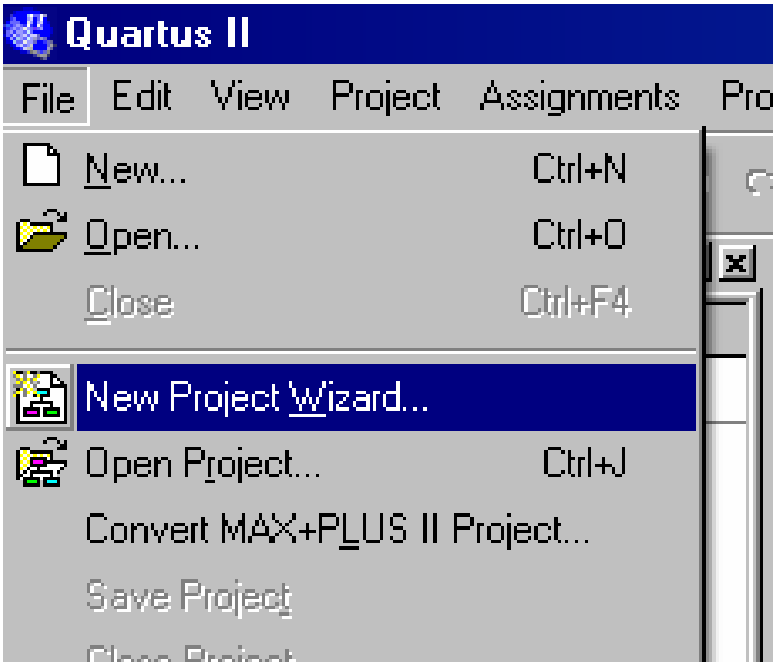
Quartus ist eine integrierte Entwicklungsumgebung für

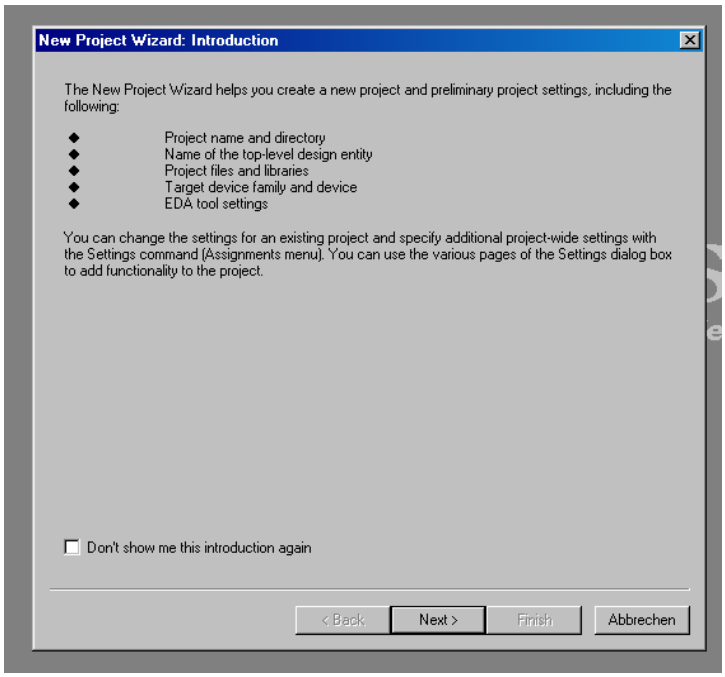
- den funktionellen Entwurf digitaler Systeme
  - in einer Hardwarebeschreibungssprache
  - in Form von graphischen Funktionsplänen
  - in Mischformen
- die Simulation der entworfenen Systeme
- die Implementierung der Systeme in programmierbare Logikschaltkreise
- die Timinganalyse in Abhängigkeit von der ausgewählten Schaltkreisfamilie

1.1 Erstellen eines neuen Projektes

Jeder Entwurf beginnt mit der Erstellung eines neuen Projektes

1. Schritt
File →New Project Wizard



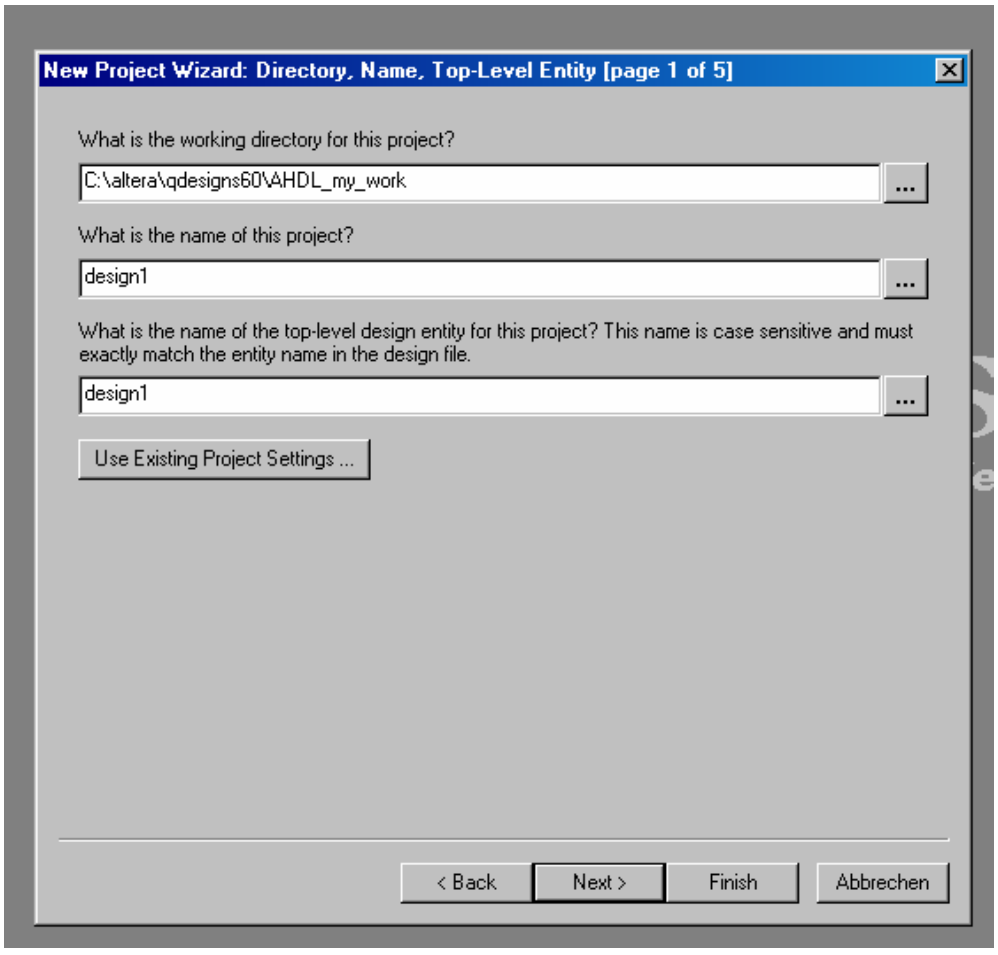


Ordner für das Projekt wählen

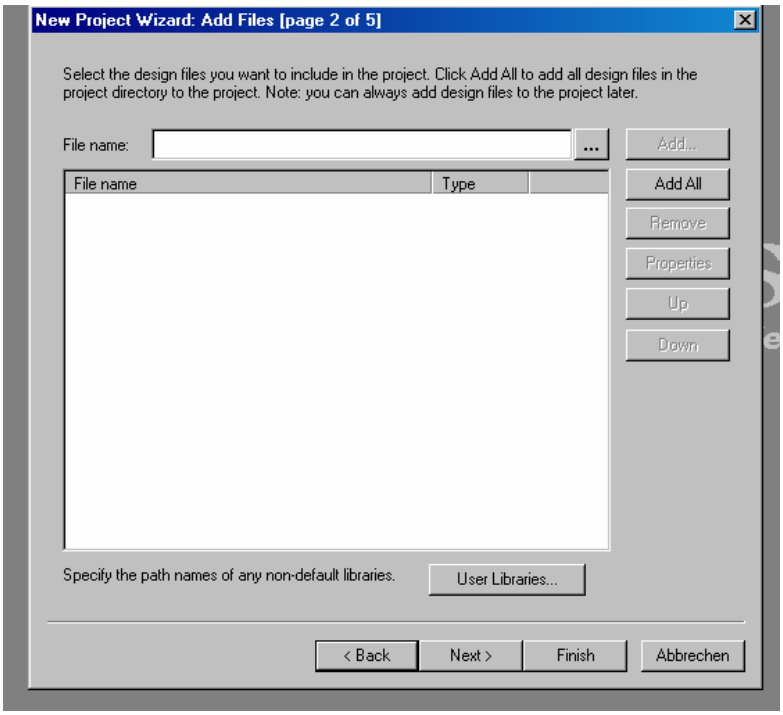
Name des Projektes festlegen

Name für das Design angeben

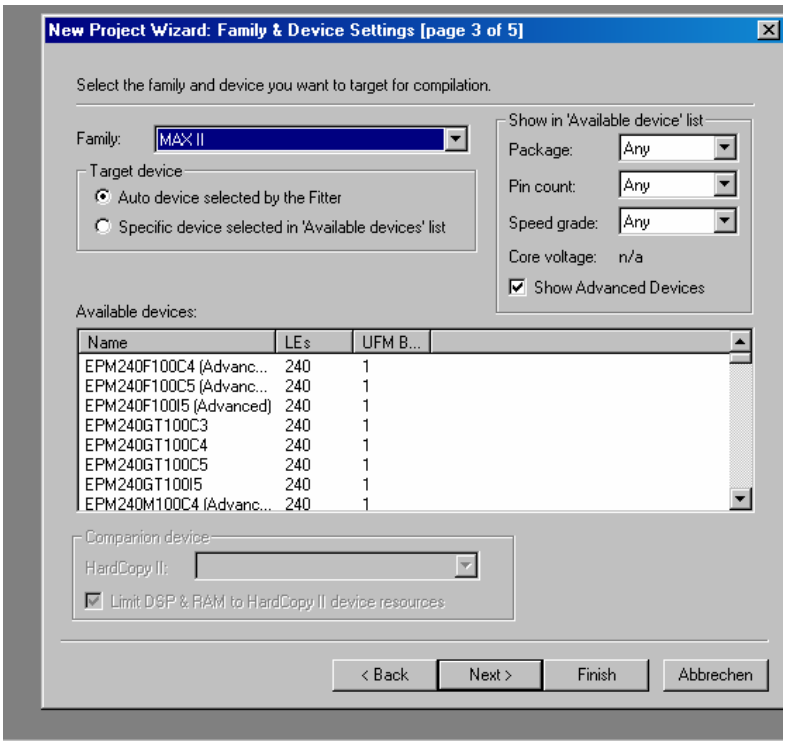
Name des Projektes und Name des Designs müssen gleich sein



Beim ersten Entwurf keine Files hinzufügen → Next

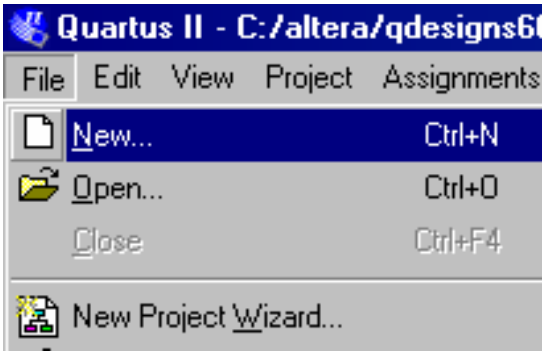


Schaltkreisfamilie wählen

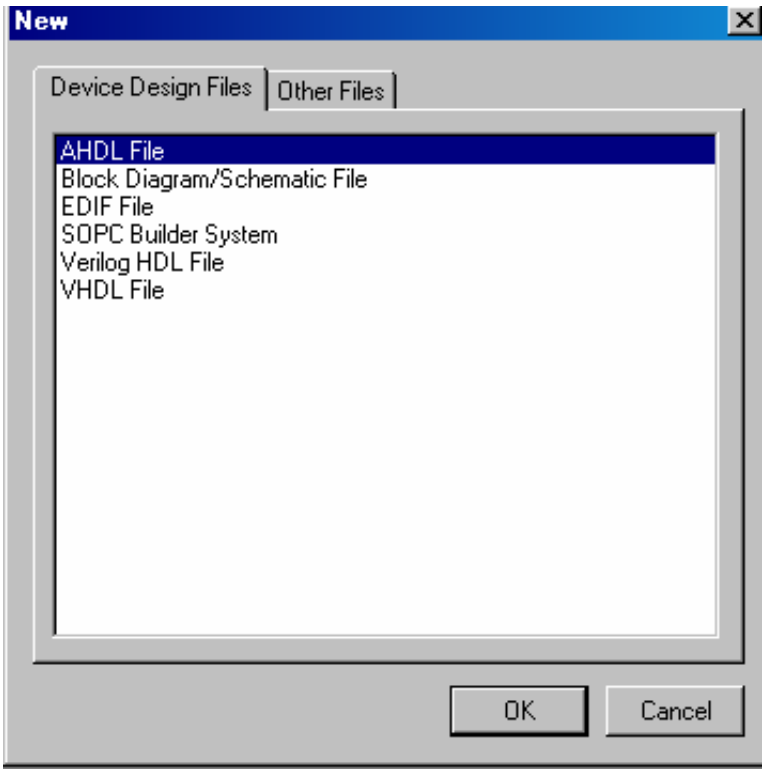


→ Next → Finish

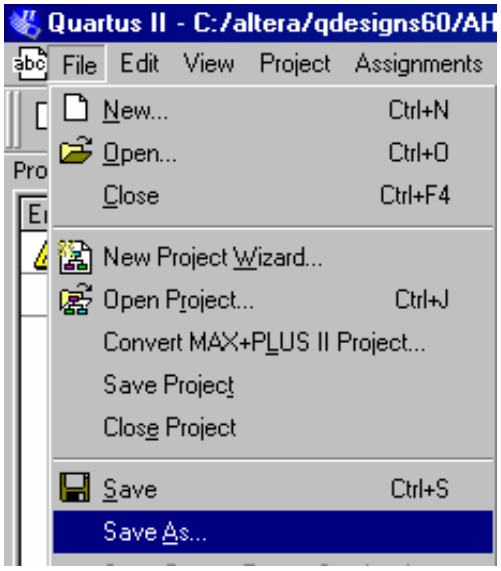
1.2 File für einen neuen Entwurf im Projekt erstellen



Filetyp für den Entwurf auswählen
AHDL File wählen

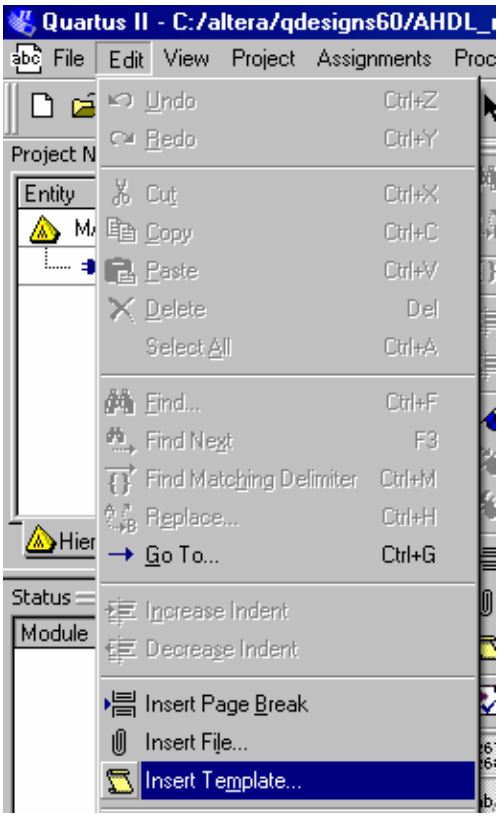


File mit *Save as* speichern der Name des Files muss der Design Name sein
Der Feilname hat die Erweiterung .tdf

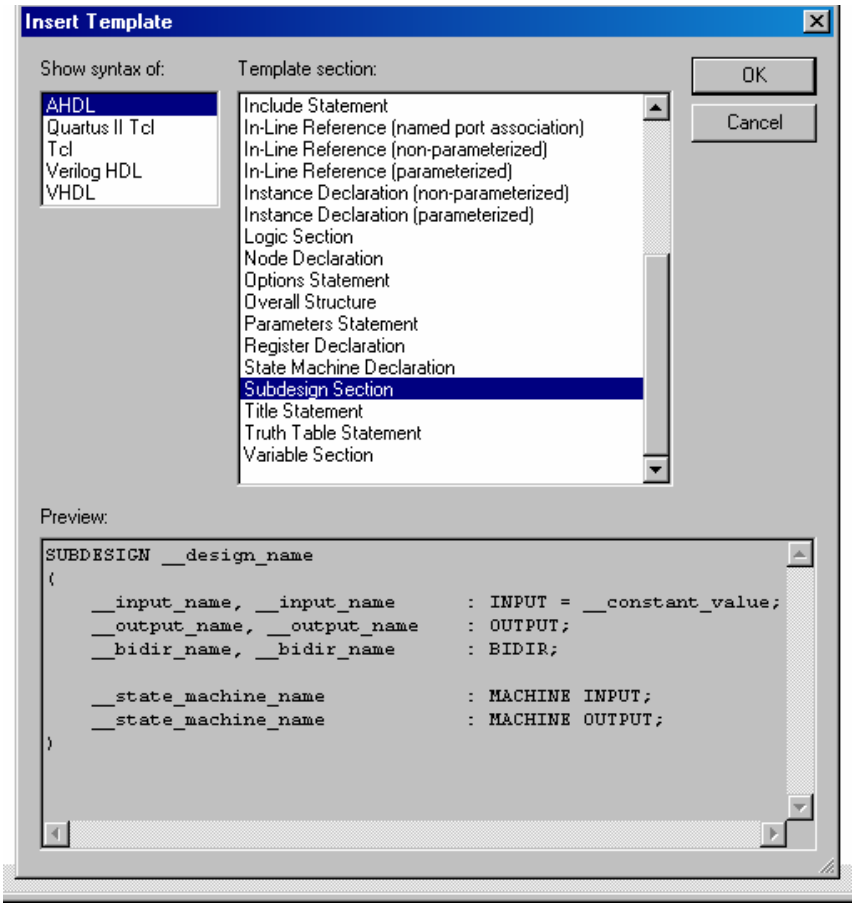


1.3 Ersten Entwurf mit der Hardwarebeschreibungssprache AHDL erstellen

Verwendung der Templates



AHDL Textfile beginnt mit SUBDESIGN
Ein und Ausgänge der Schaltung werden festgelegt



Template

SUBDESIGN __design_name
(
__input_name, __input_name : INPUT = __constant_value;
__output_name, __output_name : OUTPUT;
__bidir_name, __bidir_name : BIDIR;

__state_machine_name : MACHINE INPUT;
__state_machine_name : MACHINE OUTPUT;
)

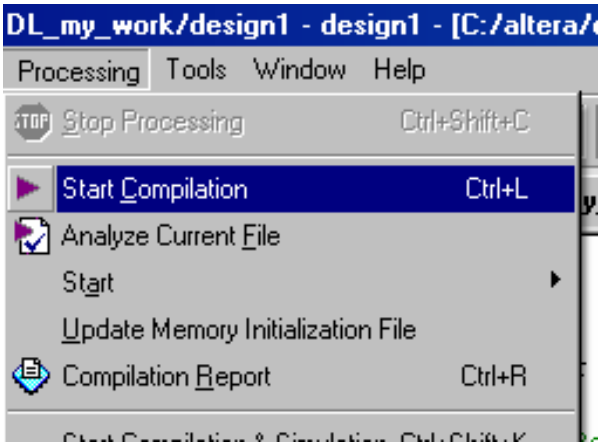
Editieren des Templates

__design_name muss dem Filenamen des Textfiles entsprechen

Beschreibung der Funktion im Logikbereich

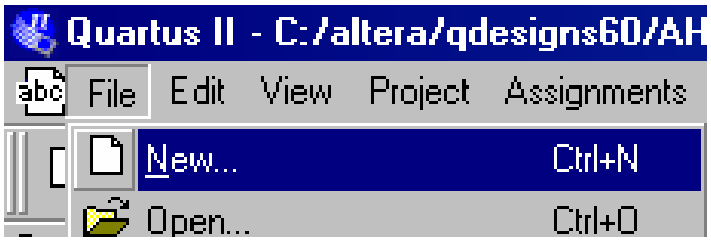
```
SUBDESIGN design1
(
        xa,xb,xc            : INPUT;
        yund,yoder,yxor     : OUTPUT;
)
% Logik Bereich - Beschreibung der Funktion %
BEGIN
        yund = xa & xb;     % UND mit 2 Eingängen %
        yoder= xa # xc;     % ODER mit 2 Eingängen %
        % Logikgleichung für EXCLUSIVEODER %
        yxor = xa & !xb # !xa & xb;
END;
```

Übersetzen des Textfiles

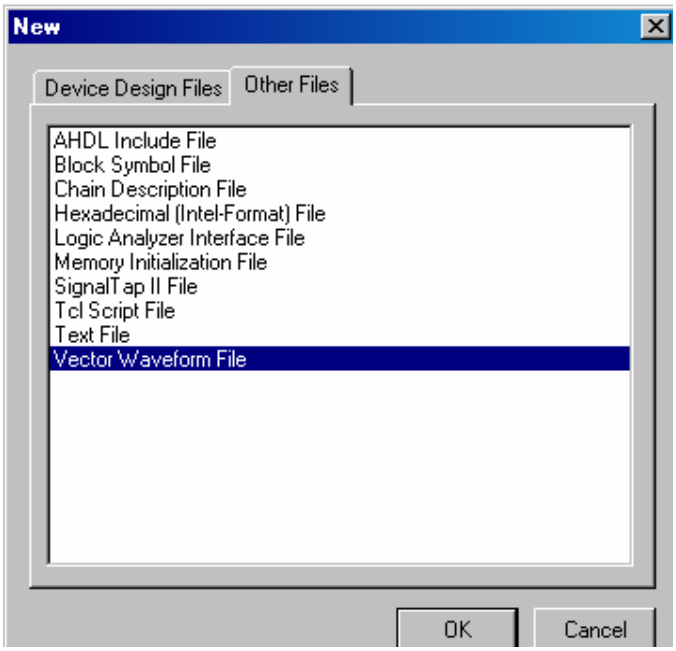


1.4 *Simulation des Entwurfes*

Waveform File erstellen



Other Files → Vector Waveform File



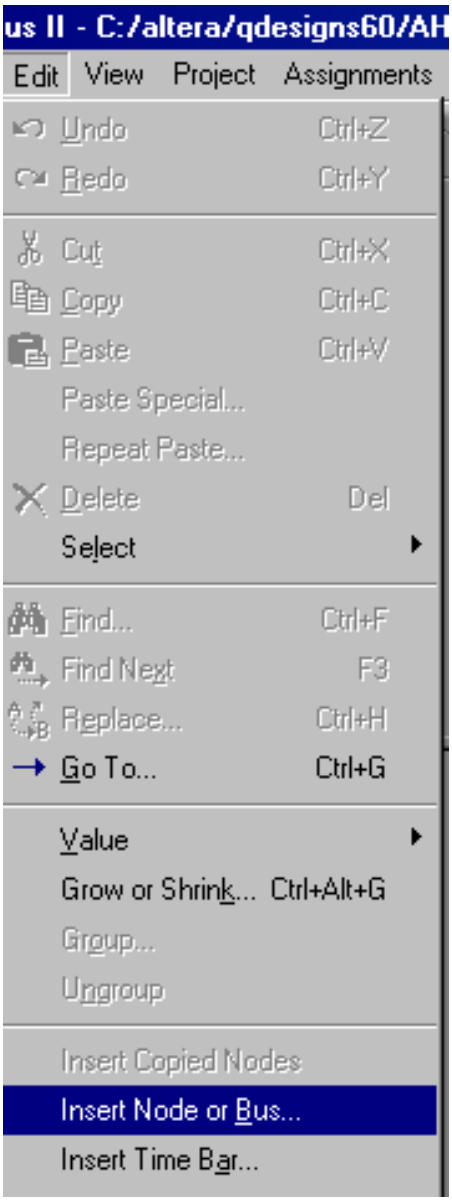
File Speichern

File → Save as (Filename muss mit Designname übereinstimmen)

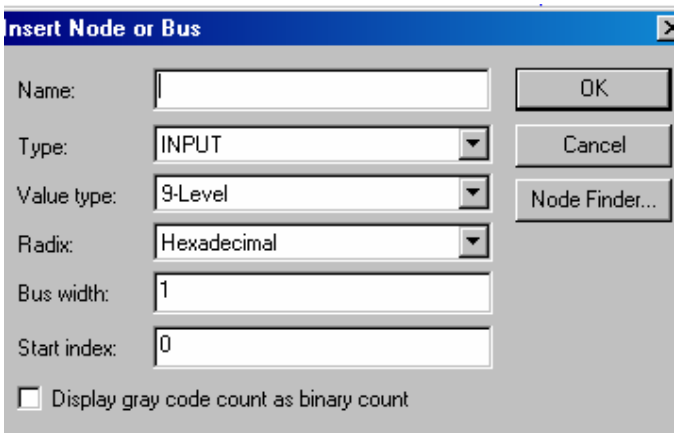
Der Feilname hat die Erweiterung .vwf

Signale hinzufügen

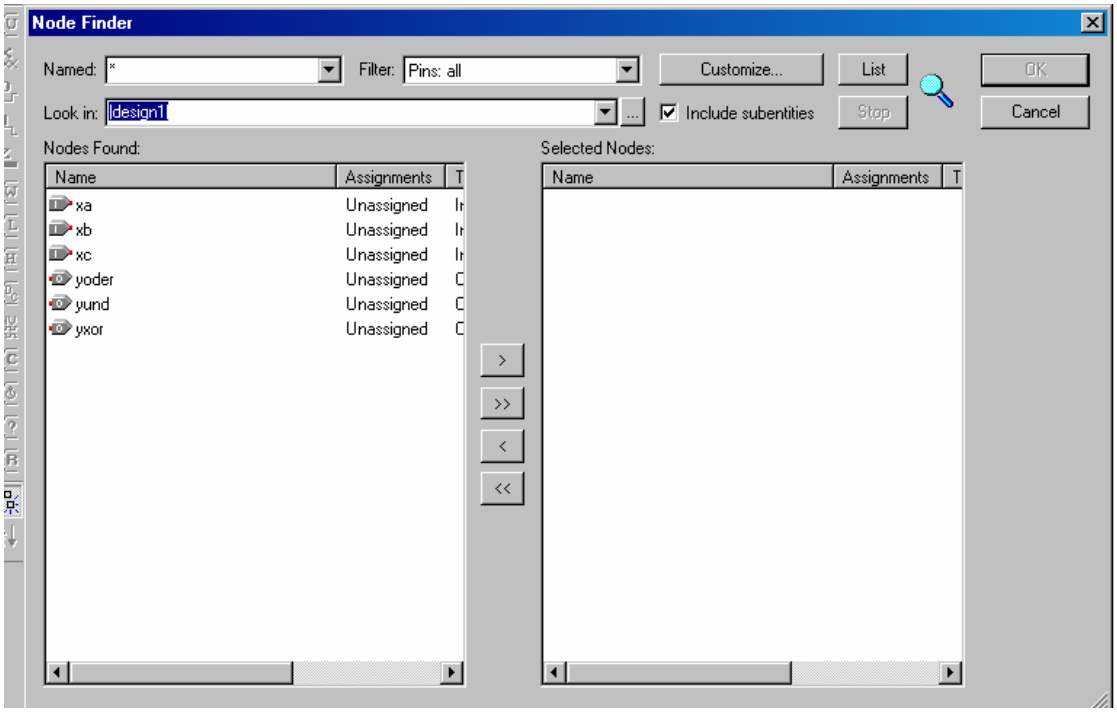
Waveform File ist im aktiven Fenster



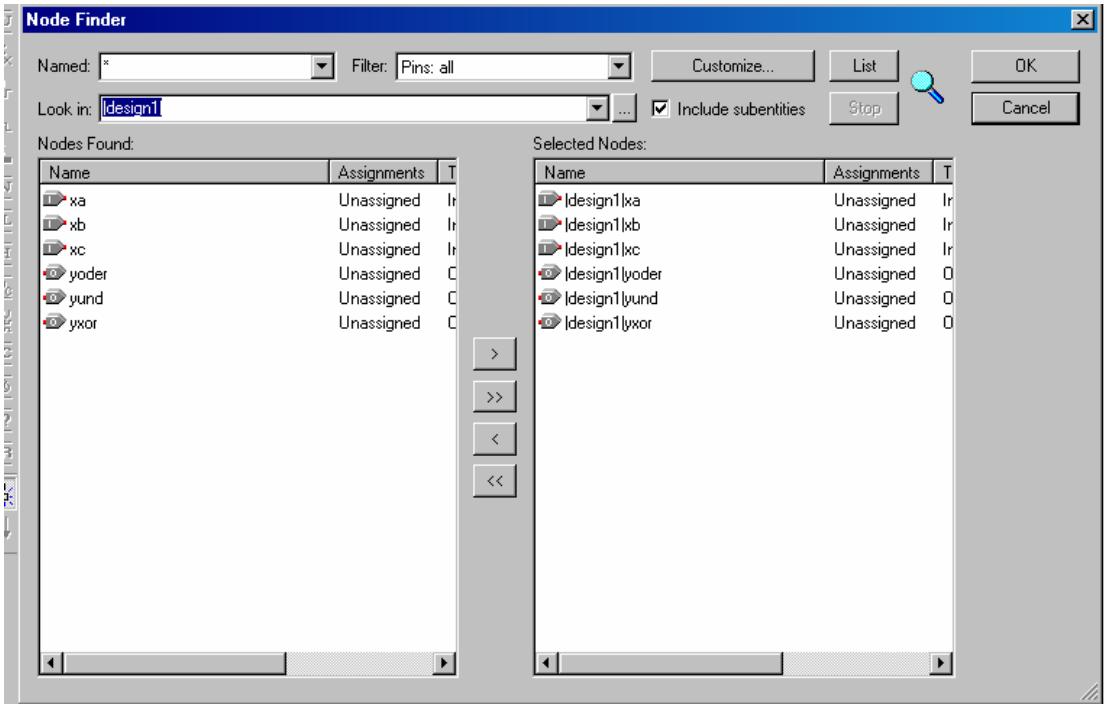
Note Finder



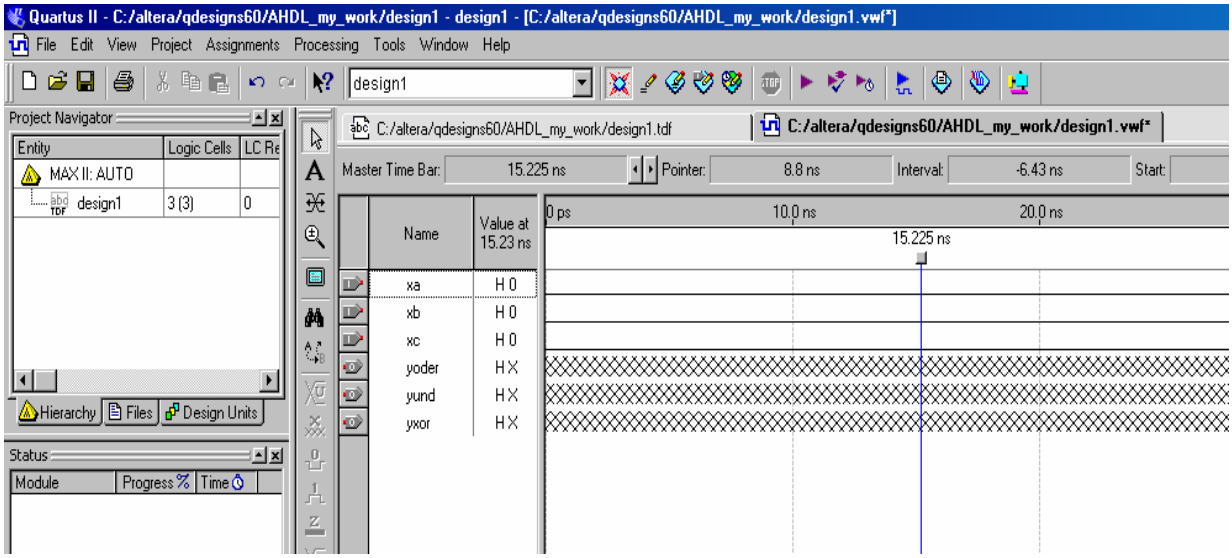
List



Auswahl der Signale

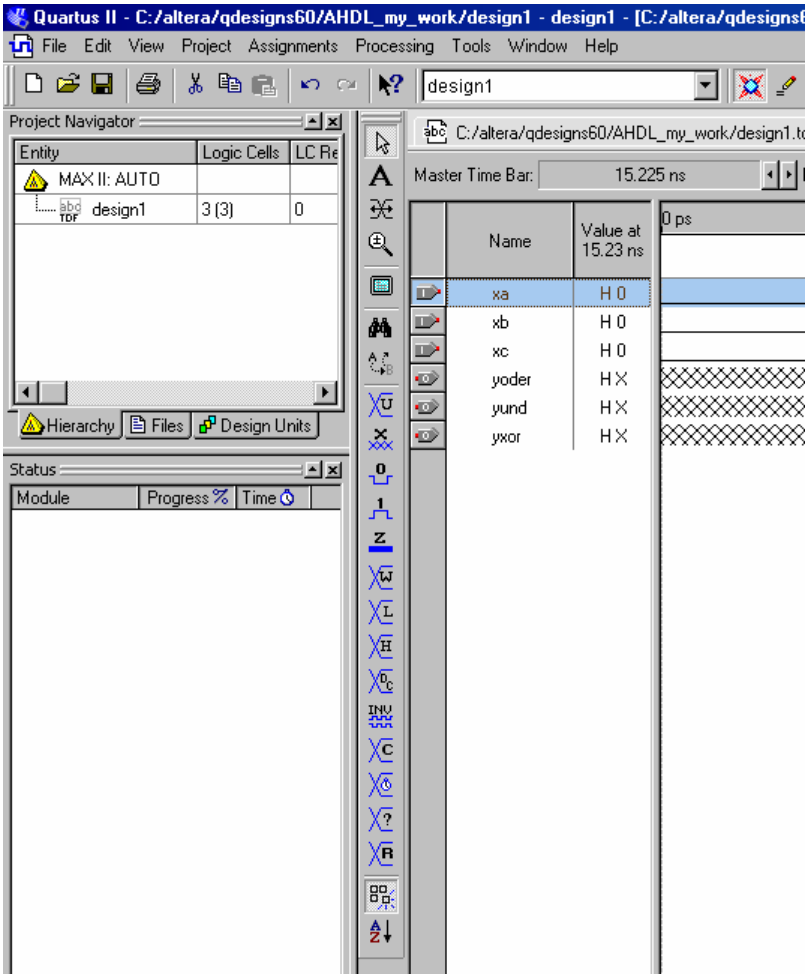


Ausgewählte Signale im Waveform File

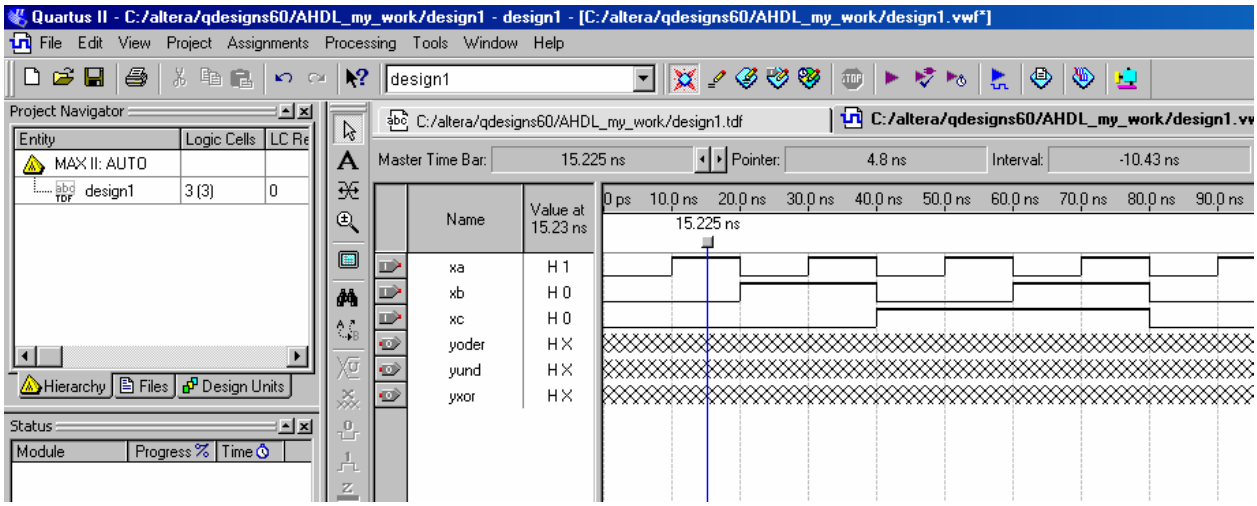


Belegung der Eingangssignale festlegen

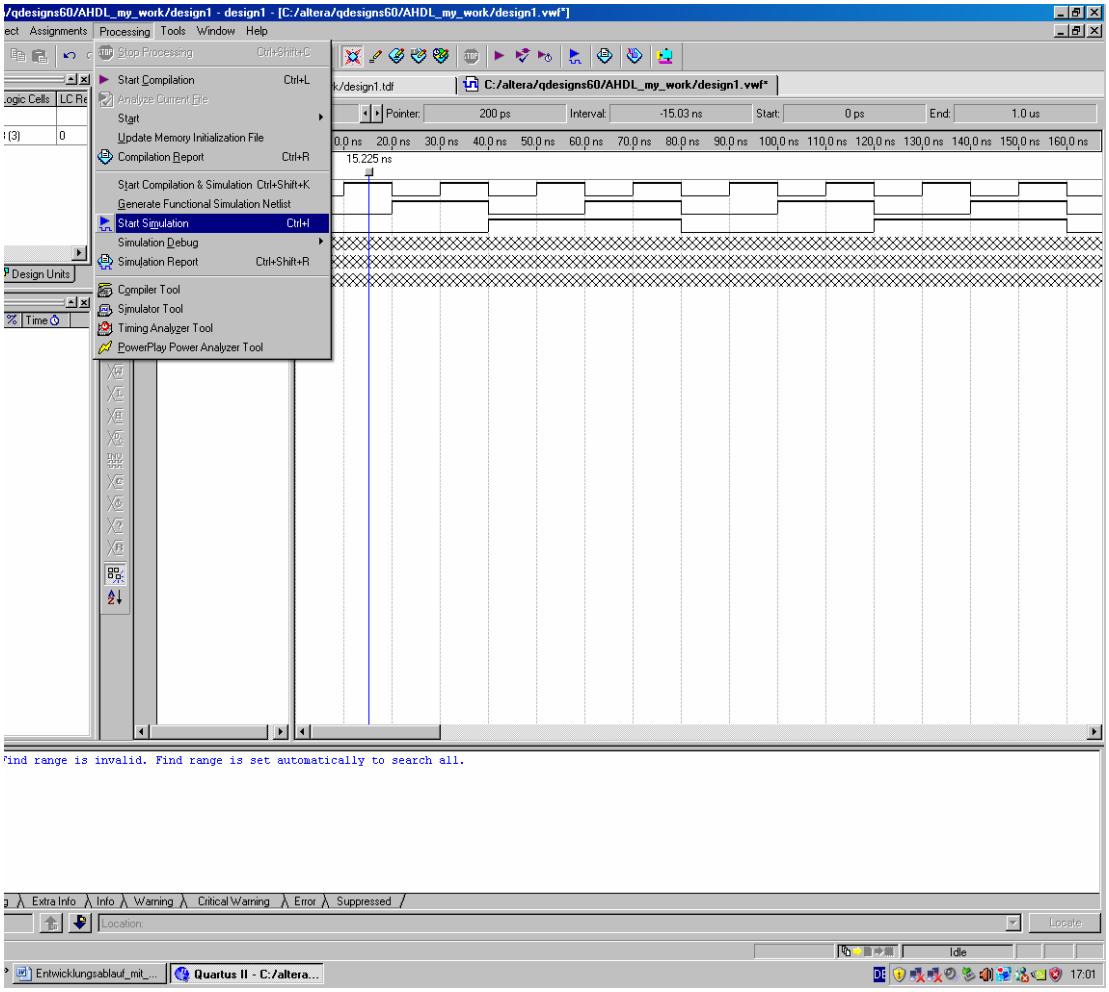
Signal bzw. Bereich eines Signals wählen **Einstellmöglichkeit auswählen**



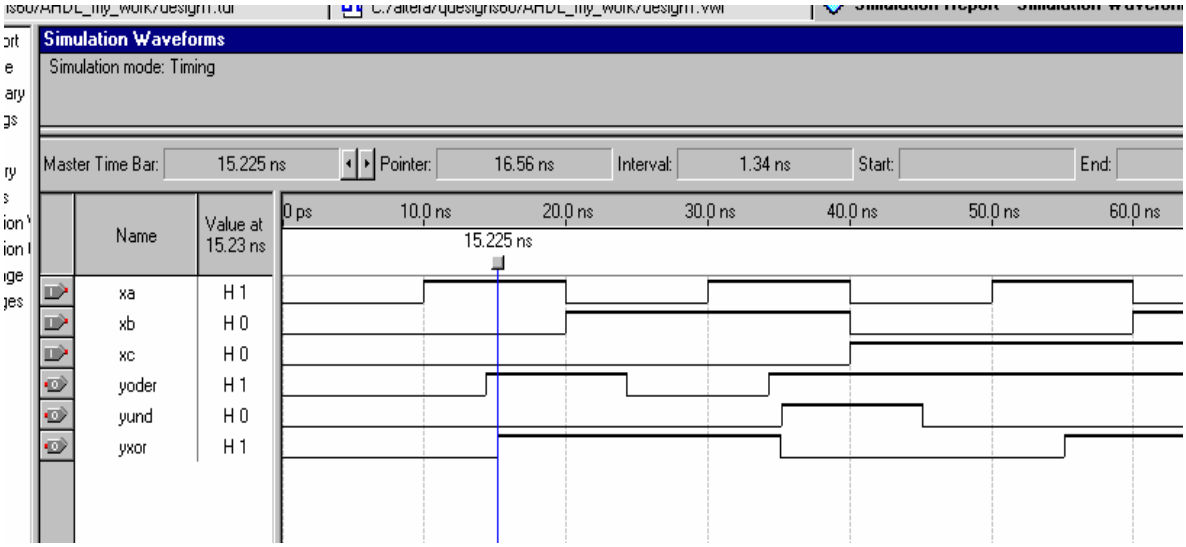
Eingangssignalbelegung für die Simulation definiert



Simulation starten

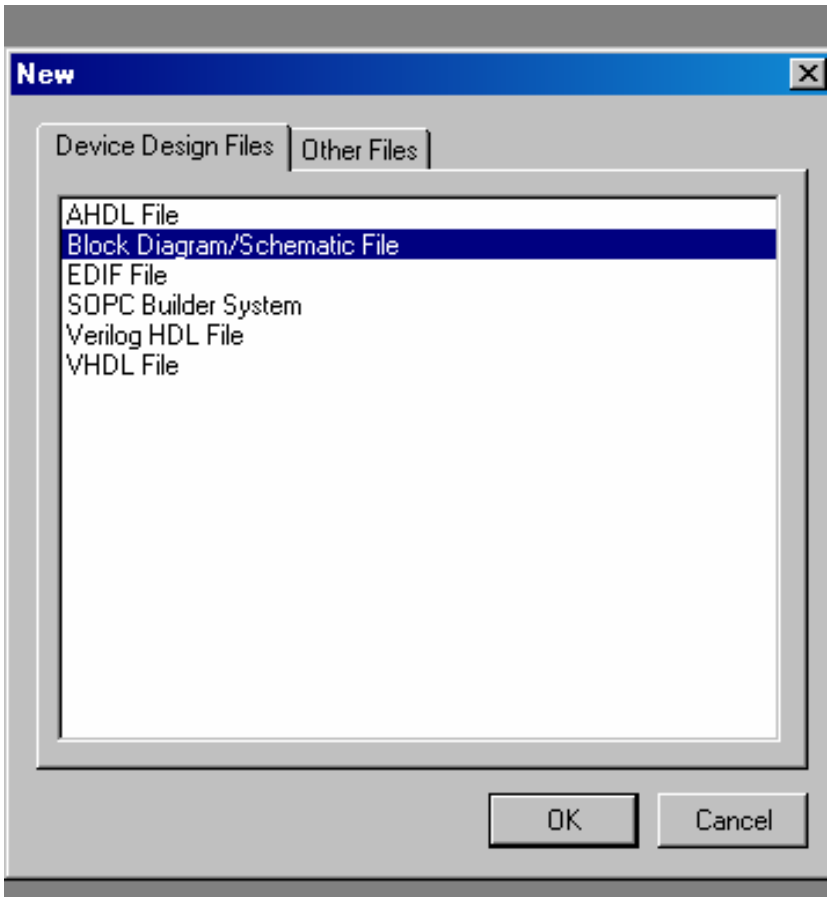


Simulationsergebnis – Ausgangsbelegung berechnet

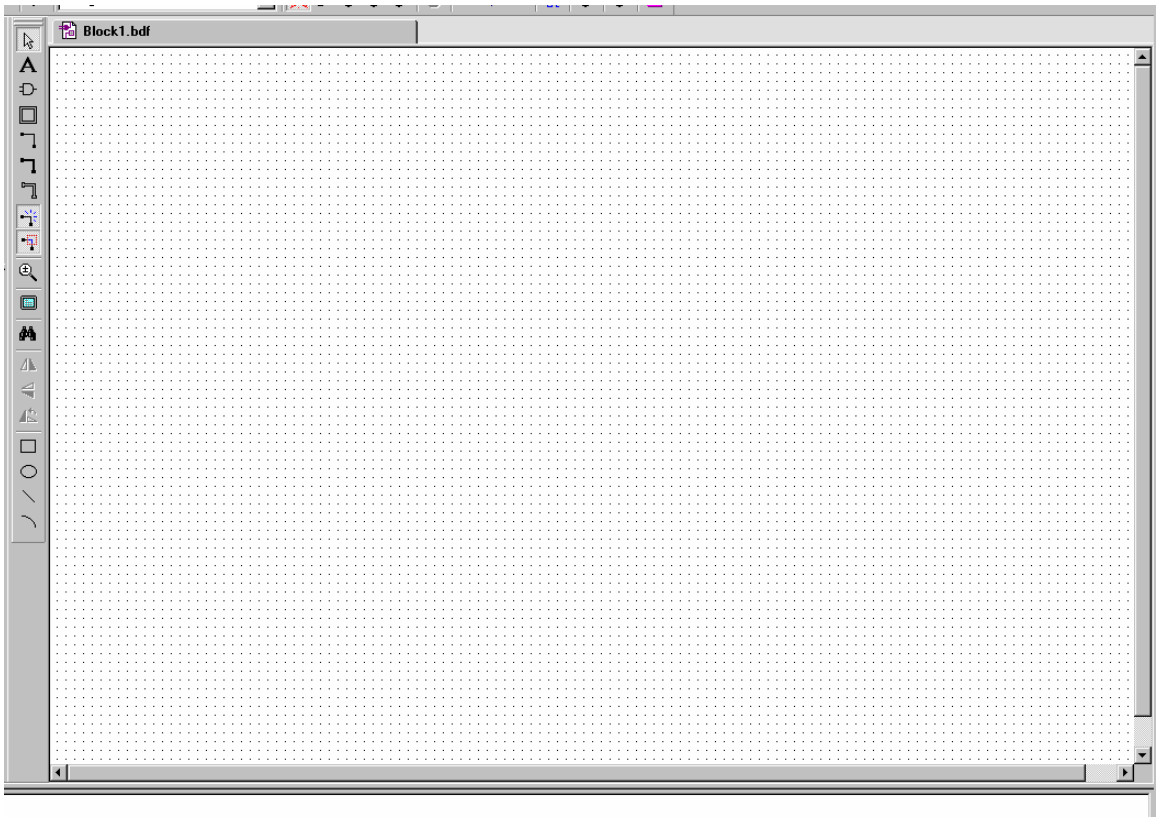


2 Entwurf mit einem Block Diagram / Schematic File

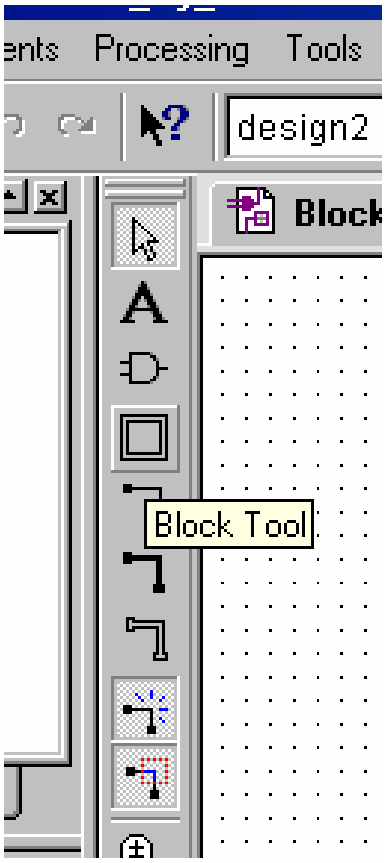
Neues Projekt wie unter 1. beschrieben erstellen
File → New



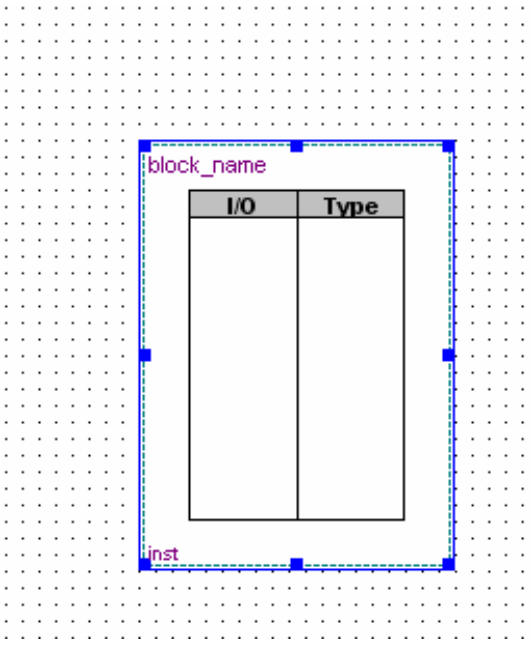
Ergebnis ist ein Fenster zur graphischen Eingabe



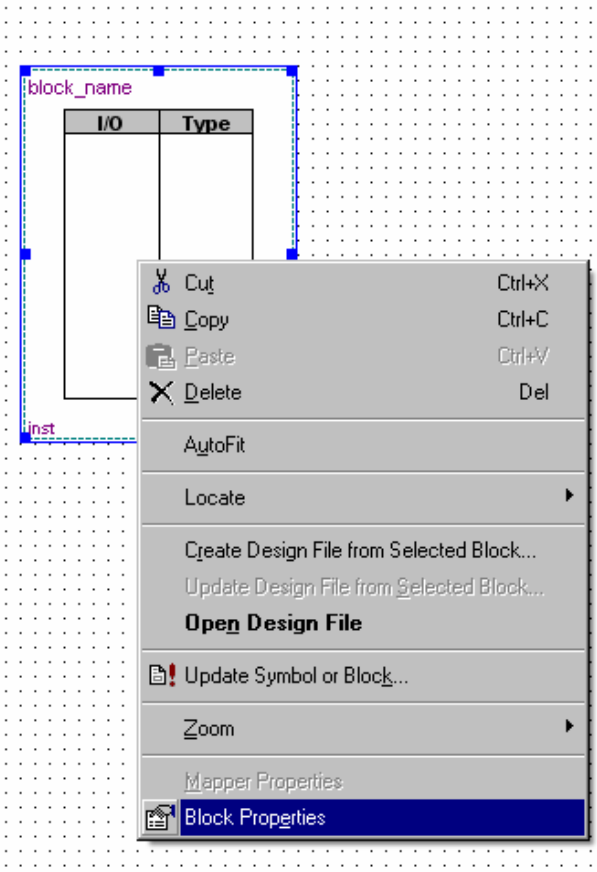
Block Tool auswählen



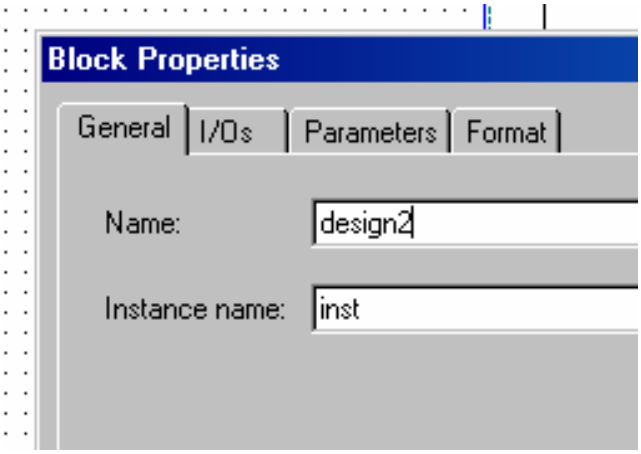
Maus im Zeichenfeld positionieren
Linke Maustaste drücken und Rechteck aufziehen
Ergebnis:



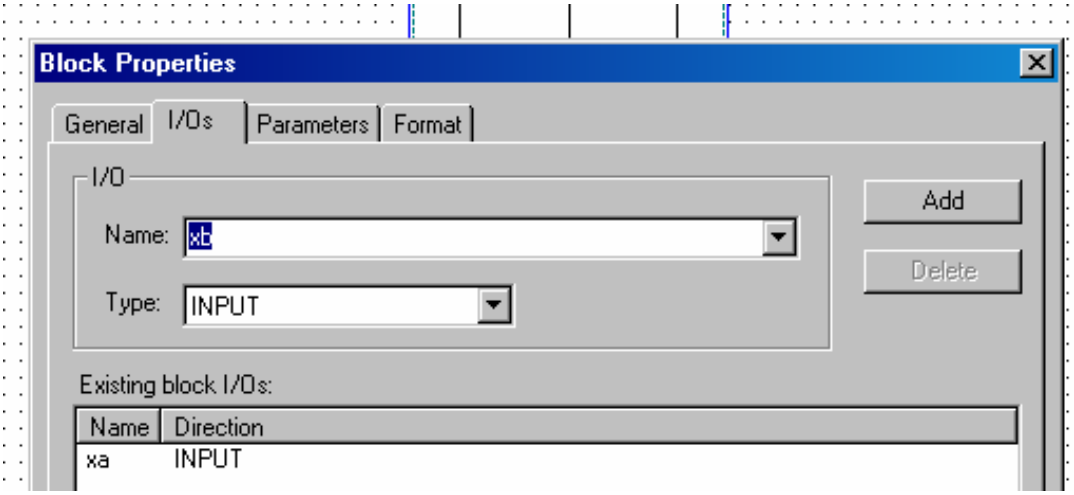
Im Block rechte Maustaste Drücken und Block Properties wählen



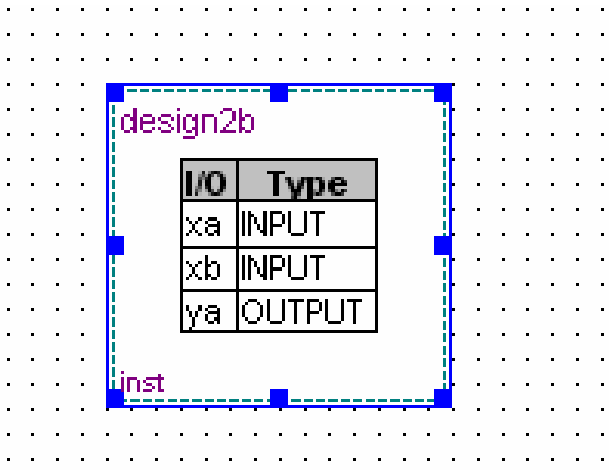
Block Namen festlegen



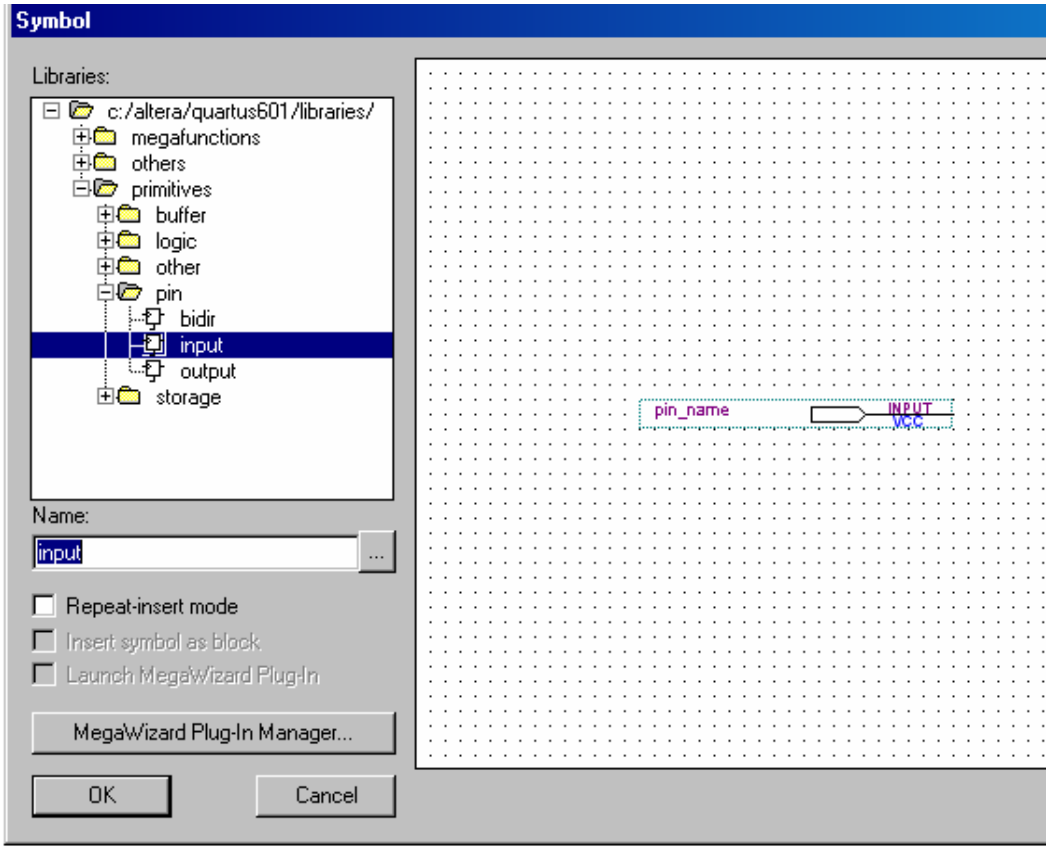
Ein- Ausgänge des Blocks festlegen



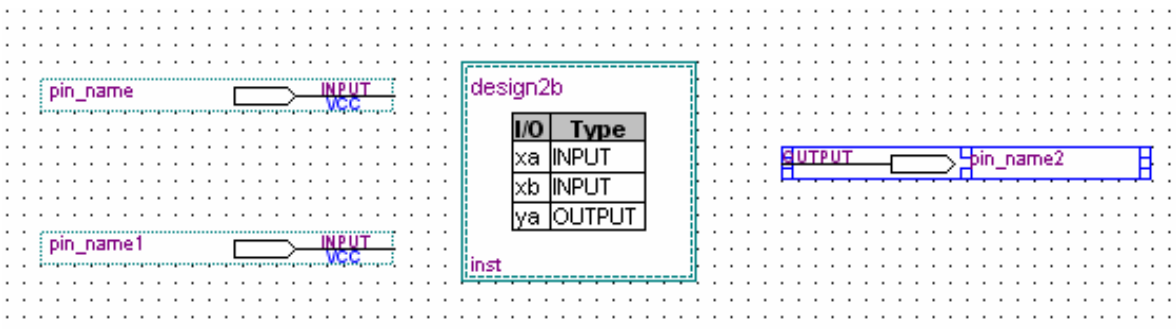
Block mit Ein_ und Ausgängen definiert



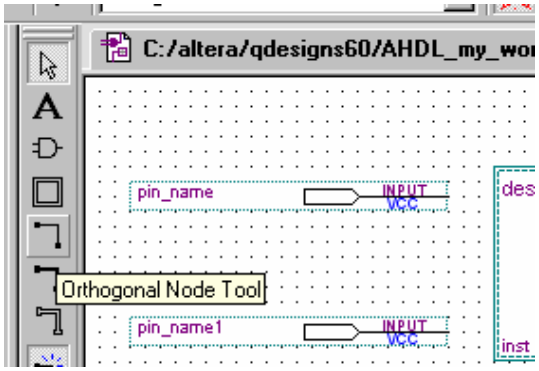
Funktionselemente aus einer Bibliothek platzieren
Im Zeichenfeld Doppelklick mit der linken Maustaste
Ergebnis: öffnen der Bibliothek

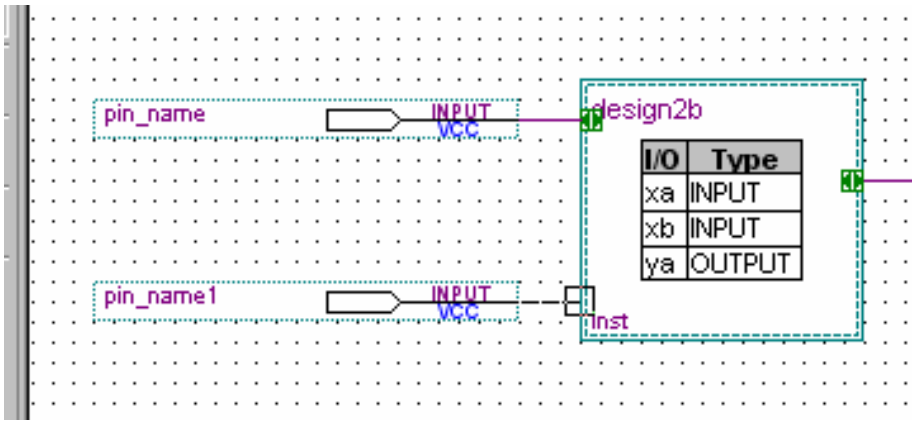


Ein- und Ausgänge platzieren

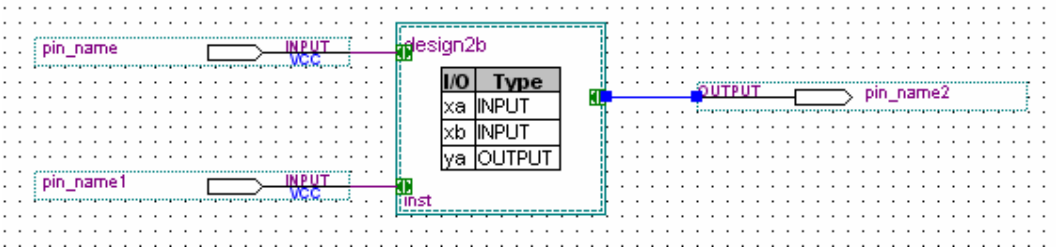


Ein-Ausgänge mit Block verbinden

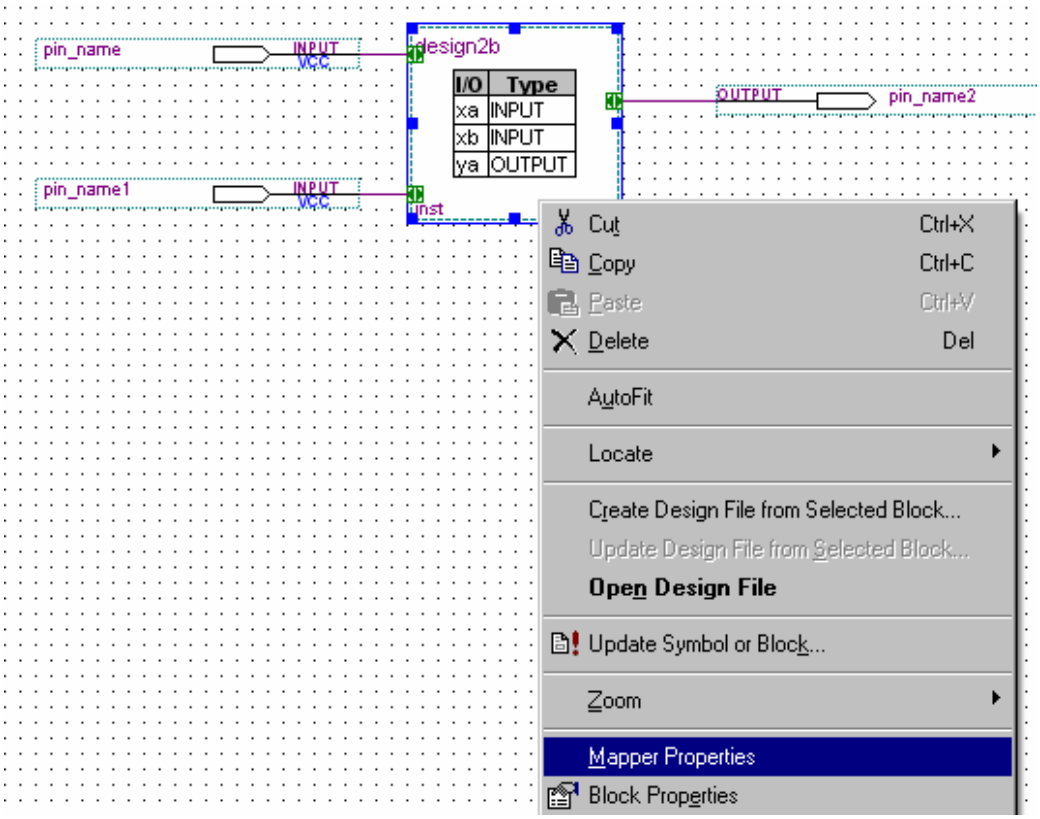




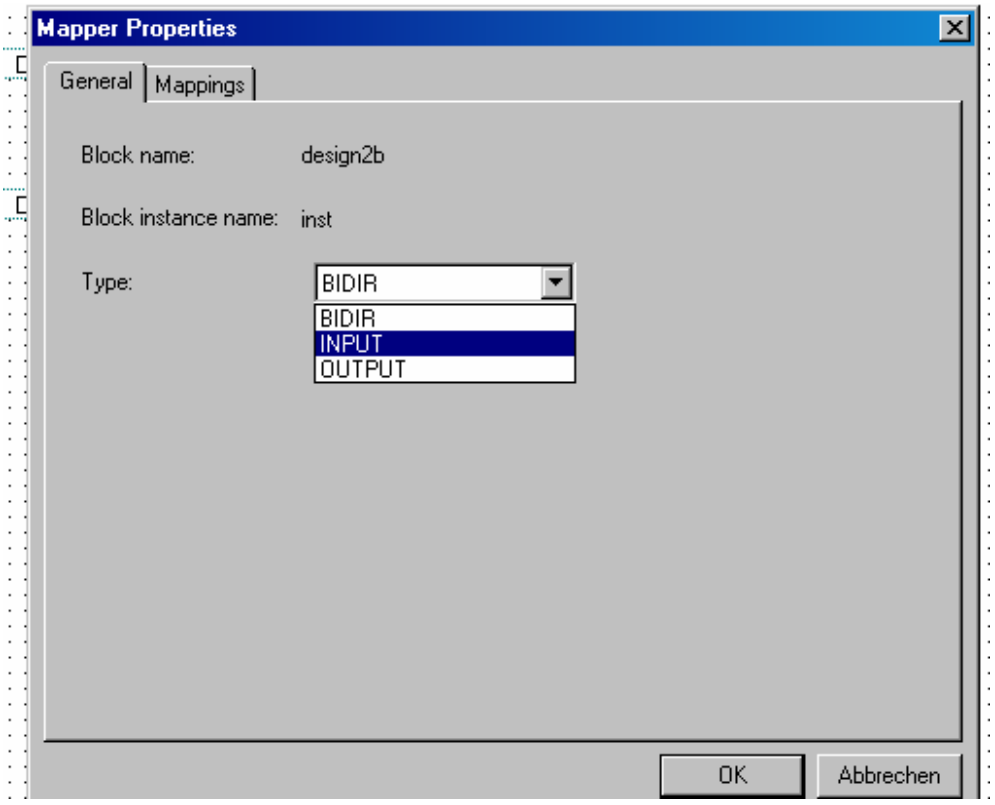
Ein_Ausgänge mit dem Block verbunden



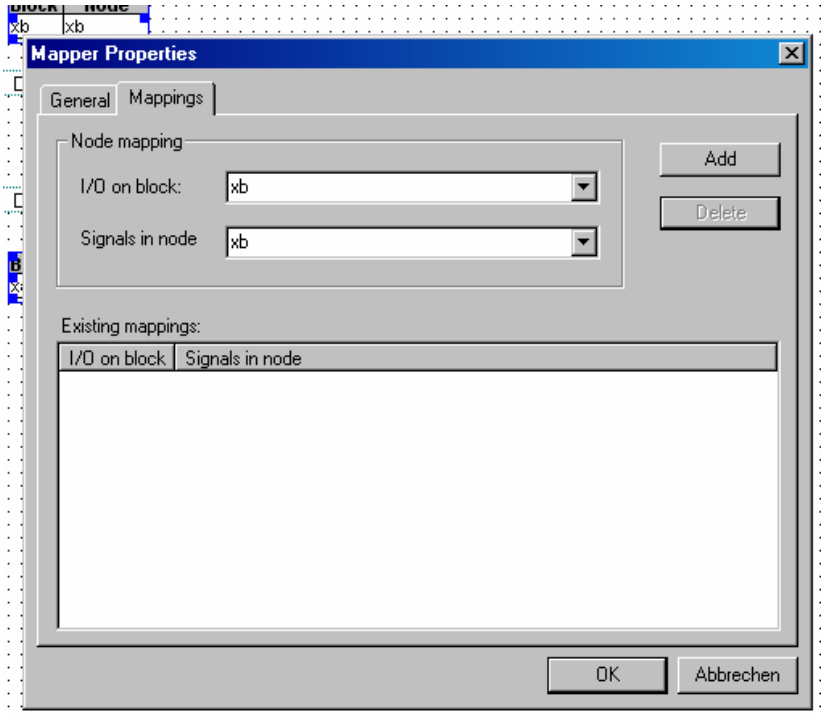
Signalnamen und Signaltyp festlegen
Mapper Properties wählen mit rechter Maustaste im Block



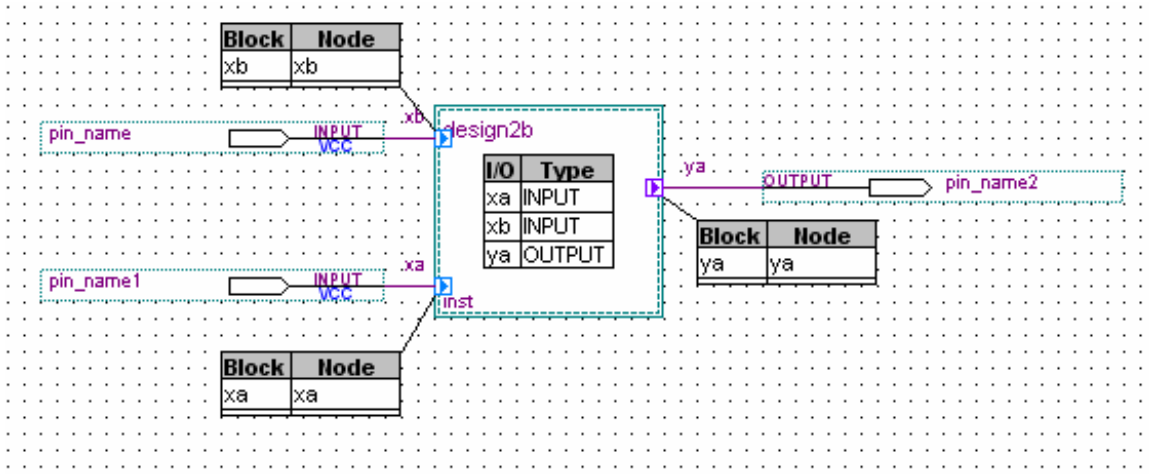
Signaltyp wählen



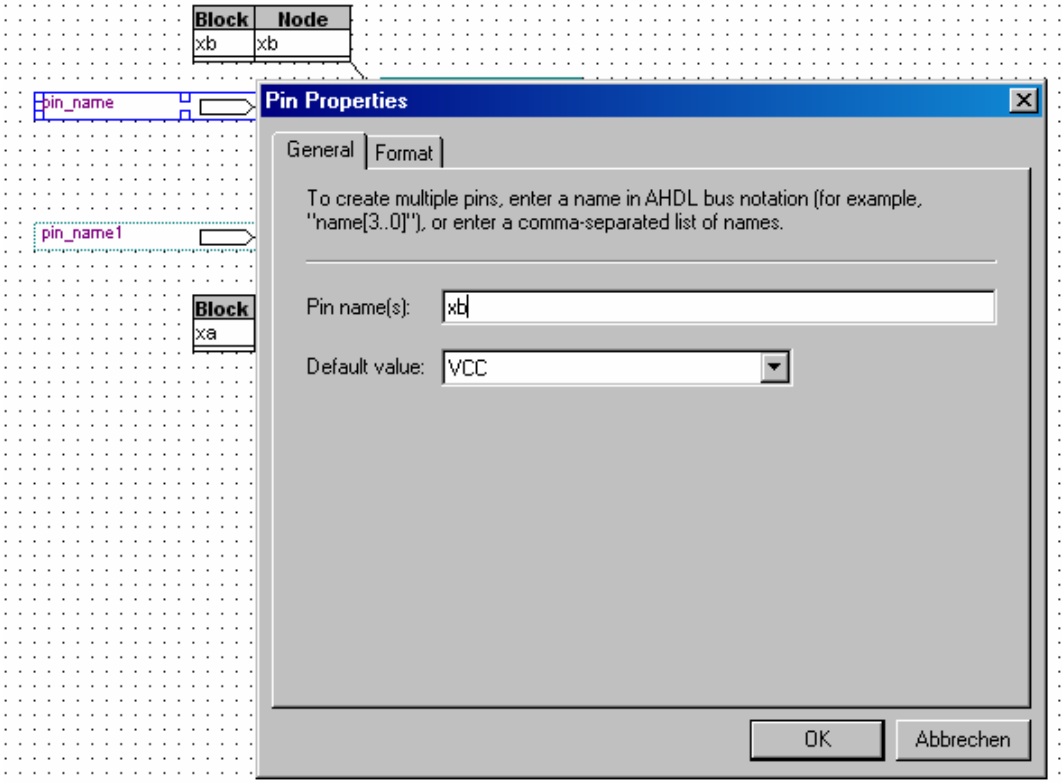
Signalnamen festlegen



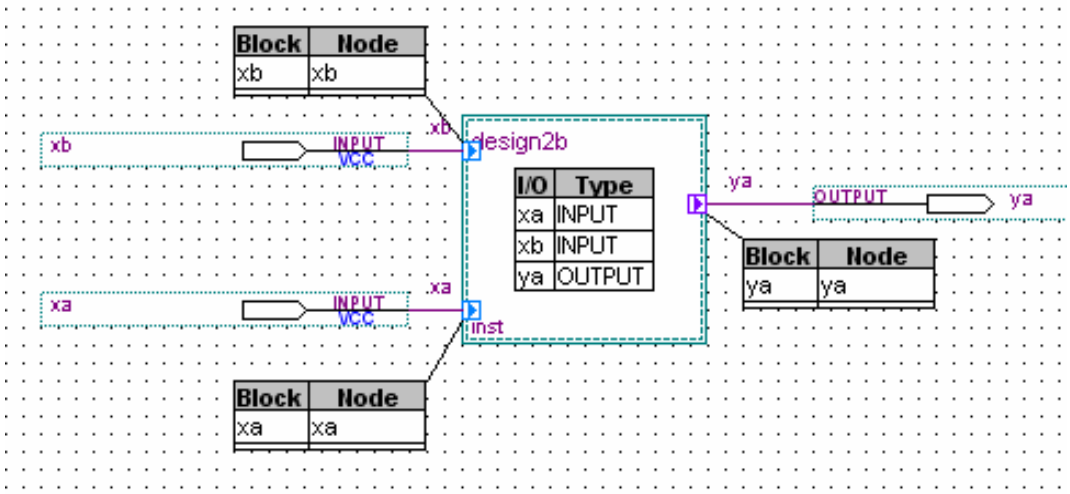
Signalnamen festgelegt



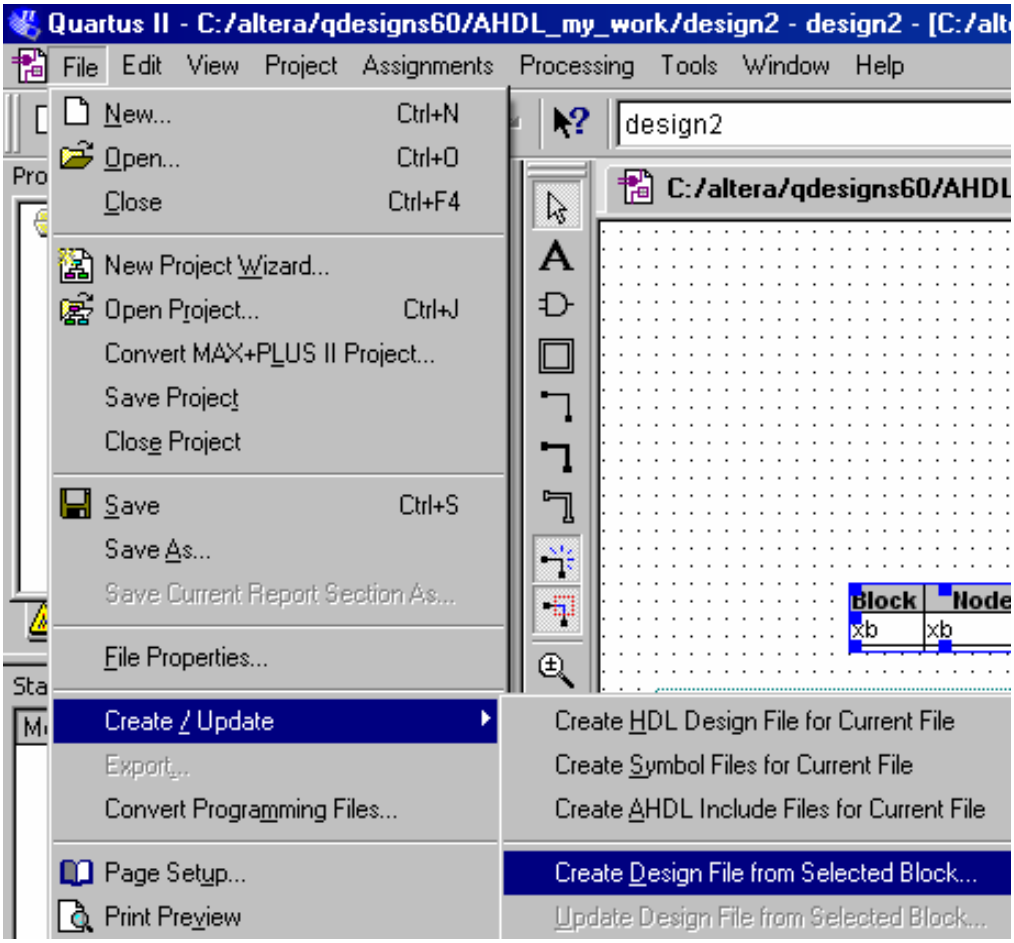
Pin-Namen eingeben
Doppelklick auf Pin



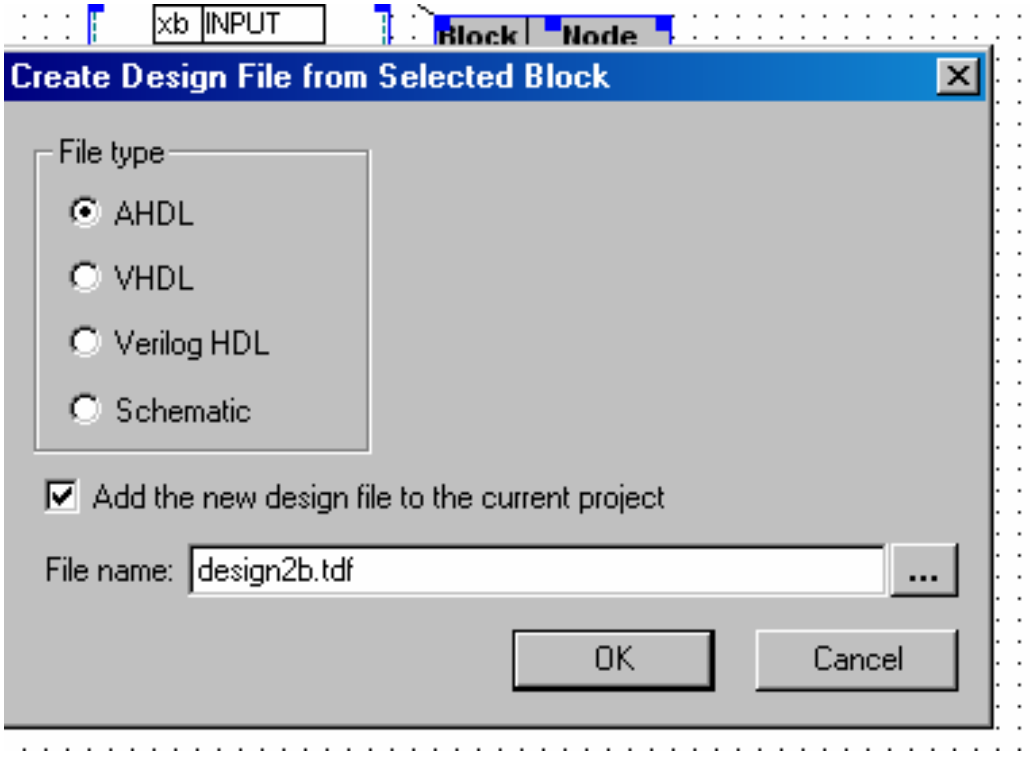
Blockdesign fertig gestellt



Block auswählen mit linker Maustaste im Block
Textdesignfile erstellen



Hardwarebeschreibungssprache auswählen



Generierter AHDL File

```
-- Generated by Quartus II Version 6.0 (Build Build 178 04/27/2006)
-- Created on Mon Feb 05 17:37:38 2007

--  Title Statement (optional)
TITLE "__your_title";

--  Include Statement (optional)
INCLUDE "__include_filename.inc";

--  Parameters Statement (optional)

-- {{ALTERA_PARAMETERS_BEGIN}} DO NOT REMOVE THIS LINE!
-- {{ALTERA_PARAMETERS_END}} DO NOT REMOVE THIS LINE!


--  Subdesign Section

SUBDESIGN design2b
(
        -- {{ALTERA_IO_BEGIN}} DO NOT REMOVE THIS LINE!
        xa : INPUT;
        xb : INPUT;
        ya : OUTPUT;
        -- {{ALTERA_IO_END}} DO NOT REMOVE THIS LINE!
)
```

Logikbereich muss noch hinzugefügt werden

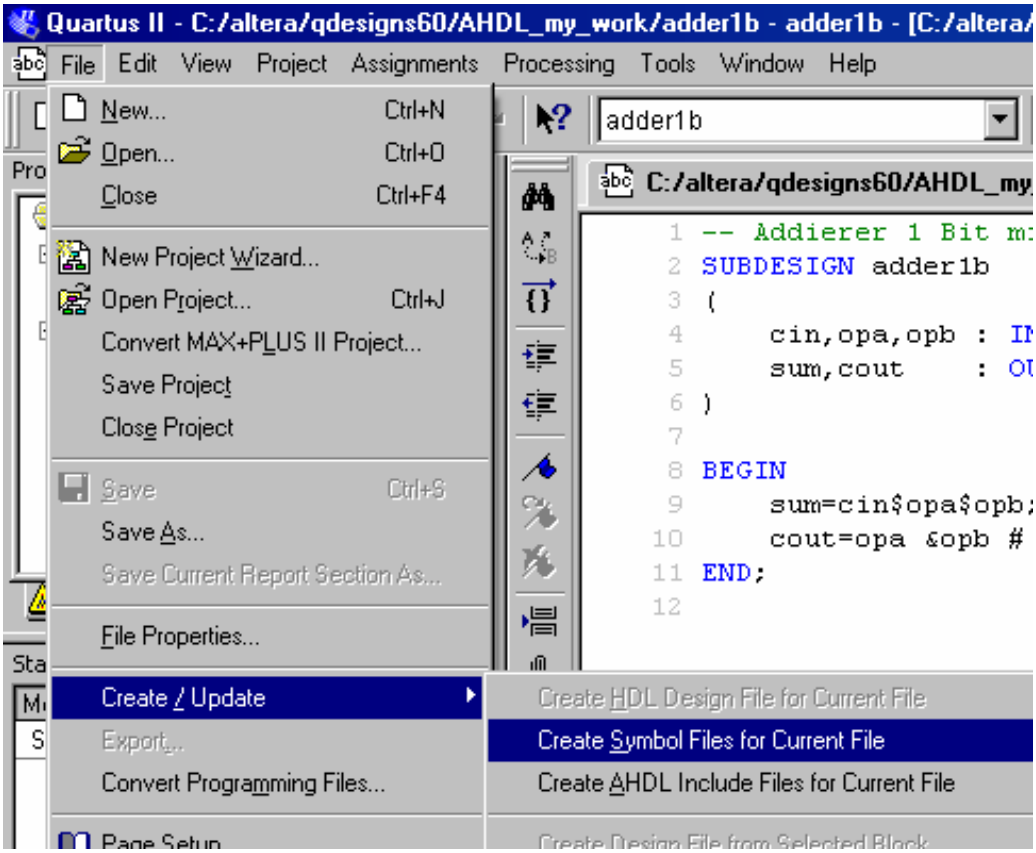
3. Symbol aus einem Textdesign erstellen

Wir erstellen in einem neuen Projekt einen 1 Bit Adder

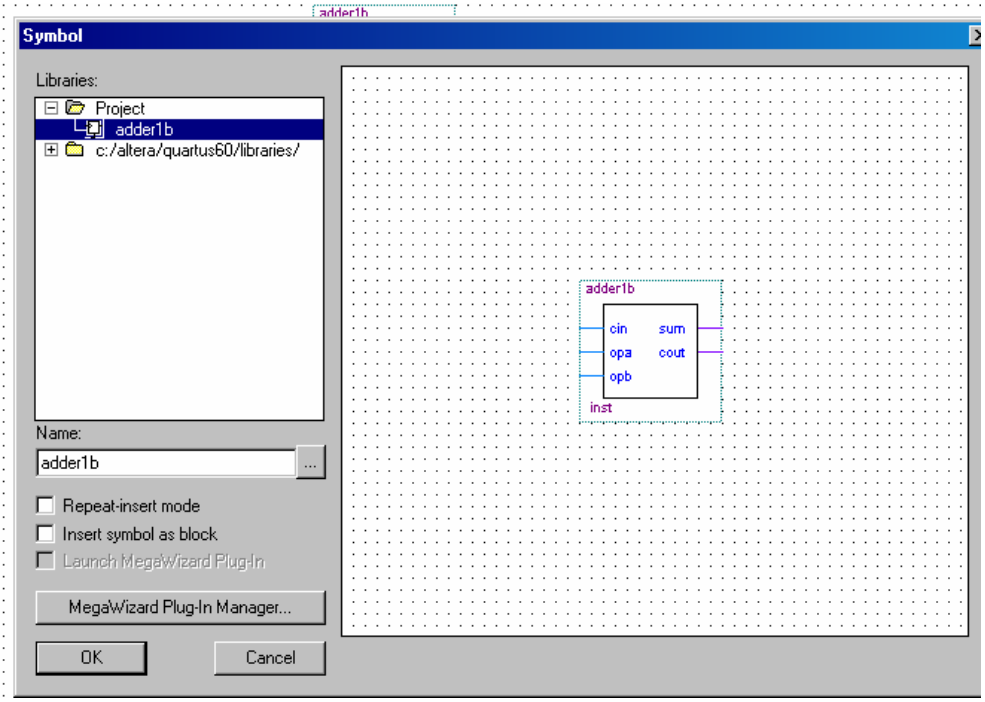
```
-- Addierer 1 Bit mit Logikgleichungen
SUBDESIGN adder1b
(
        cin,opa,opb   : INPUT;
        sum,cout      : OUTPUT;
)

BEGIN
        sum=cin$opa$opb;
        cout=opa &opb # opa & cin # opb & cin;
END;
```

Erstellen eines Symbols



Anwendung des Symbols in einem neuen Projekt in einem Grafik – Design
Erstellen eines 4 Bit Addierers mit Hilfe des 1 Bit Addierers
Symbol des ! Bit Addierers aus der Projekt Bibliothek holen



Symbole für den 4 Bit Addierer und für Inputs und Outputs platzieren und verbinden