VHDL-Seminar

Thema: Libero

„Design Creation/Verification“

WS 04/05

Andreas Schibilla (ii4900)

1. Inhaltsverzeichnis

2. Einführung und Übersicht

2.1 Worum geht es bei „Design Creation / Verification“

Wird ein neues Hardware-Projekt durchgeführt, so beginnt man nach der Vorbereitungsphase zunächst mit der Verhaltens-Beschreibung der gewünschten Funktionen:

Design Creation:
Dies kann in Actel Libero zum einen durch das Schreiben von HDL-Code (VHDL oder Verilog) geschehen (→Text Based Entry). Eine andere Möglichkeit bietet das mitgelieferte Tool View Draw AE, welches es gestattet mit Hilfe einer grafischen Oberfläche die Funktionalität durch „zusammenklicken“ abzubilden. (→Schematic Based Entry).

Ist das Design bzw. eine Teilfunktion fertig beschrieben, so muss die richtige Funktionsweise der Einheit überprüft werden. Dies geschieht mit Hilfe von Testbenches, die bestimmte Signalzustände erzeugen, so dass das Ergebnis verifiziert werden kann.

Design Verification:
In Actel Libero können selbständig geschriebene VHDL-Testbenches zur Simulation eingesetzt werden. Mit Hilfe der mitgelieferten und etwas eingeschränkten „Lite-Version“ von SynaptiCAD WaveFormer können gezielte Testabläufe über eine Waveform in einer grafischen Oberfläche eingegeben und anschließend als VHDL-Testbench exportiert werden. Die Testphase und Fehlersuche kann dadurch wesentlich vereinfacht und verkürzt werden. Liegt eine Testbench für die gewünschte Funktion vor, so kann damit die Verhaltens-Simulation des Designs in ModelSim AE gestartet werden.

2.2 Zugehöriger Design Flow in Libero

Der Design Flow im Bereich Design Creation/Verification besteht aus folgenden Phasen:

1. HDL Design Eingabe

- Eingabe des Designs in einem Texteditor (HDL-Editor) oder mit Hilfe der grafischen Oberfläche in View Draw AE (Schematics).
- Erstellen und Einbinden (Instanziierung) von ACTgen Macros

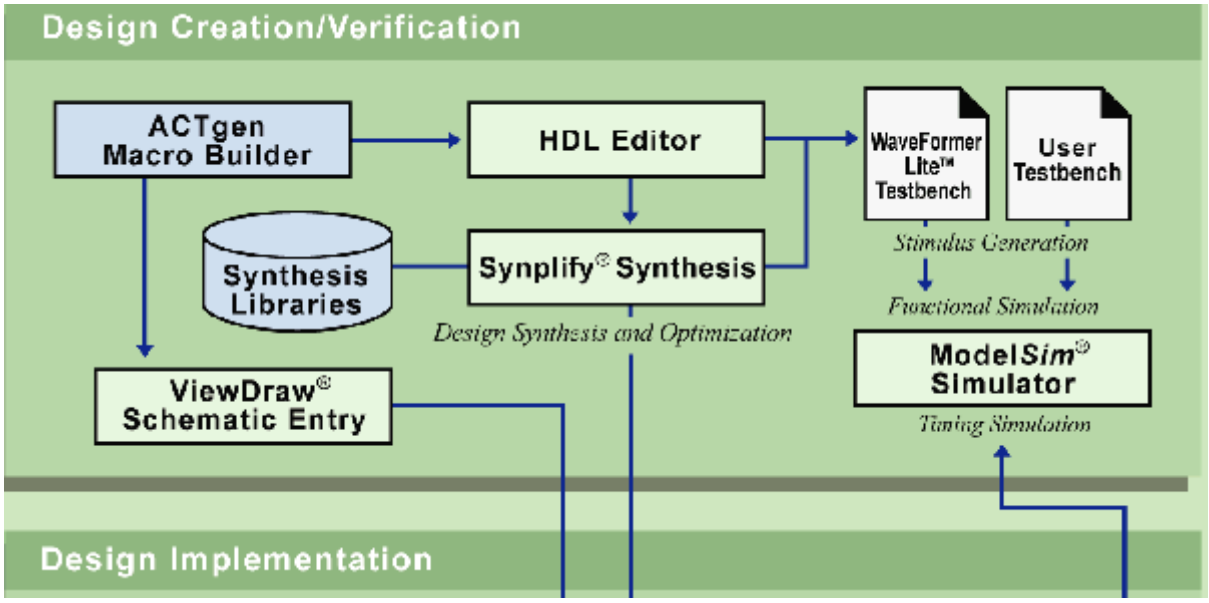
2. Funktionale Simulation

- Erstellen einer Standard HDL Testbench mit typischen nicht synthetisierbaren Elementen (non RTL) wie z.B. `„wait for“` oder `„sig <= value after time“`, bzw. Generation über Waveforms mit dem Zusatzprogramm „WaveFormer Lite“
- Durchführen einer funktionalen Simulation vor der Synthese zur Verifikation der Funktionalität des HDL-Codes → Stimulus Simulation mit dem Zusatzprogramm „ModelSim“
- Auswertung der Simulationsergebnisse: Fehlersuche, Verbesserungen...

3. Synthese/EDIF Generation

- Nachdem das Design erstellt und die korrekte Funktionsweise verifiziert wurde, folgt die Synthese durch Aufruf eines Zusatzprogramms.
- Das Synthestool transformiert die Verhaltensbeschreibung (behavioral HDL) in eine „Gate-Level“ Netzliste und optimiert das Design für die Zielplattform.
- Reine Schematic Designs werden von dem Libero-Tool automatisch in eine Netzliste umgewandelt

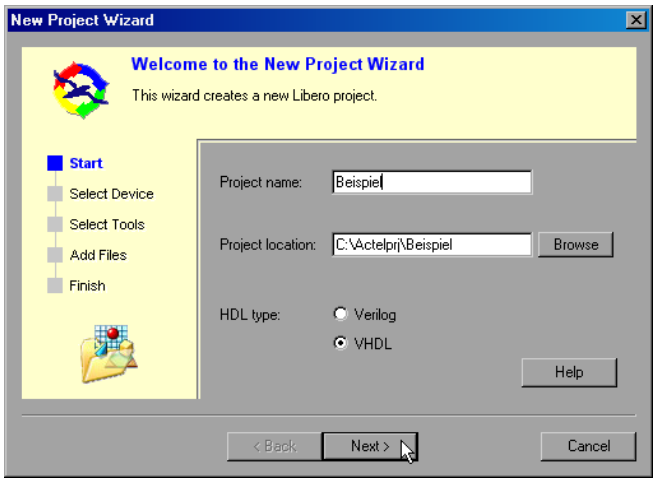
Die folgende Abbildung veranschaulicht den Design Flow in Libero noch einmal:



Die Libero-IDE unterstützt den Design-Flow und ermöglicht das Schnelle Aufrufen der verschiedenen Tools. Welche Tools dabei während der „Design Creation/Verification“-Phase zum Einsatz kommen, zeigt die untenstehende Tabelle:

Function	Tool	Company
Project Manager, HDL Editor	Libero IDE	Actel
Schematic Capture	ViewDraw AE	Mentor Graphics
Synthesis	Synplify AE	Synplicity
Simulation	Model*Sim* AE	Mentor Graphics
Testbench Creation	WaveFormer Lite AE	SynaptiCAD
Physical Synthesis	PALACE AE	Magma Design Automation

2.2 Anlegen eines neuen Projekts



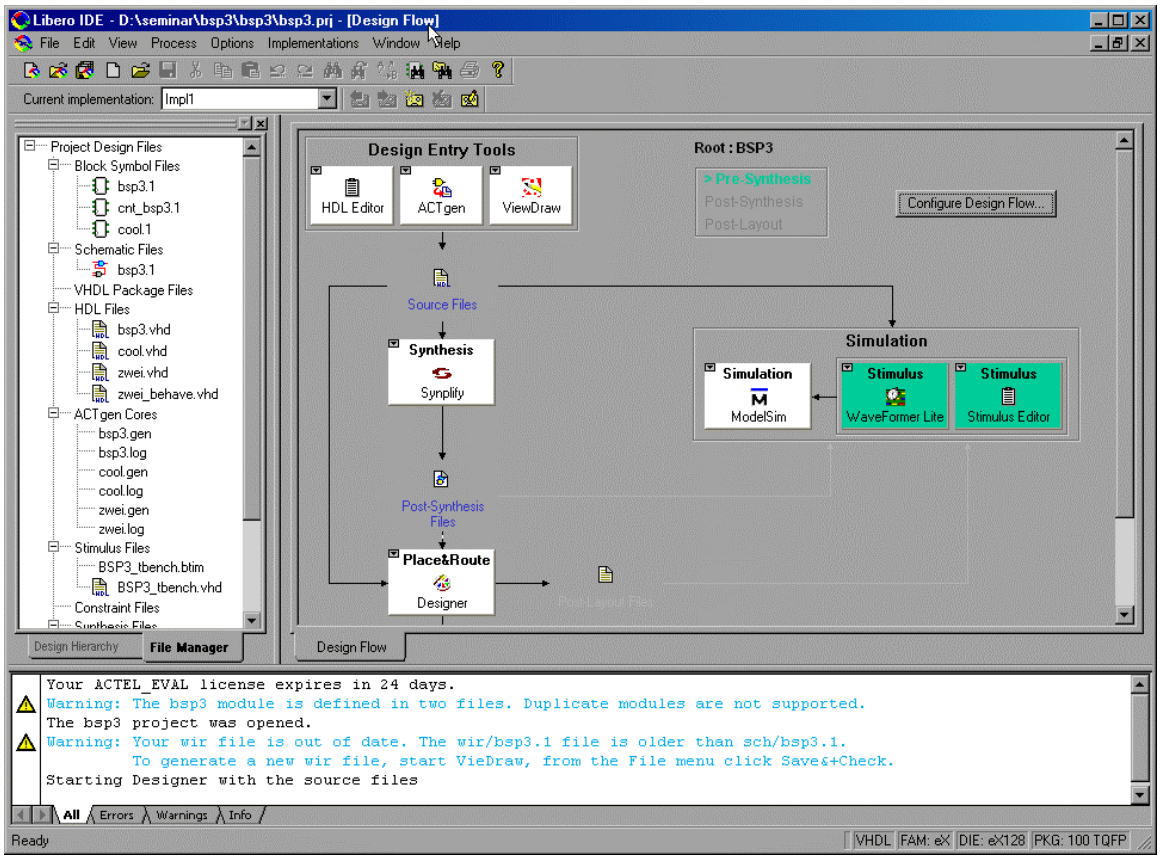
Nach dem Start der Actel Libero Software ist die Arbeitsfläche zunächst leer. Über den Menüpunkt **„File→New Project“** wird der **„New Project Wizard“** aufgerufen, mit dem ein neues Projekt schrittweise konfiguriert und erzeugt wird:

Im Verlauf der einzelnen Schritte müssen der Projektname und Typ, der Speicherort, der Zielbaustein, die zur Verwendung kommenden Fremdtools und optional vorhandene Dateien angegeben werden.

Ein Klick auf **„Finish“** erzeugt das Projekt, die Design Eingabe kann beginnen!

2.3 Die Oberfläche (IDE) in Libero

Die Libero Oberfläche besteht aus drei Fensterbereichen, die im Hauptprogramm eingebettet sind. Über eine zusätzliche Toolbar sind wichtige Grundfunktionen wie Laden & Speichern von Dateien schnell zu erreichen, ohne den Umweg über das Menü:



Im linken Bereich befindet sich die **„Design Hierarchy“** zusammen mit dem **„File Manager“**, der alle Projektdateien in einer Baumstruktur anzeigt. Über einen Tab-Reiter kann zwischen den beiden Ansichten gewechselt werden.
Im rechten Bereich ist der Design Flow angezeigt, der Buttons für alle entsprechenden Funktionen wie z.B. das Anlegen einer neuen HDL-Datei oder den Aufruf des Simulators bereitstellt. Dies ist ebenfalls der Bereich zum Editieren von HDL-Dateien. Auch hier kann über den unteren Tab-Reiter gewechselt werden.
Das untere Fenster ist für Log- und Statusmeldungen reserviert, die allesamt oder gefiltert (nur Errors, nur Warnings oder nur Infos) ausgegeben werden können.

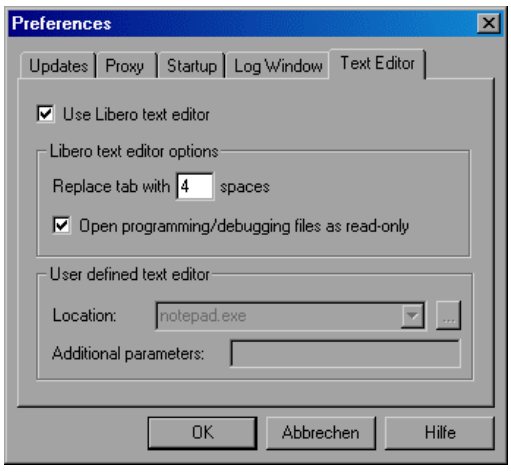
Außerdem werden unterschiedliche Aktionen durch den Aufruf von Fremdprogrammen ermöglicht, wie beispielsweise Schematic-Design. Hierzu wird das jeweilige Programm aus der IDE heraus gestartet.

3. Text Based Entry

3.1 Einleitung

Für die textuelle Eingabe neuer Designs bzw. das Bearbeiten vorhandener Designs wird ein entsprechender Texteditor benötigt. Die Libero-Oberfläche besitzt einen eingebauten Editor, der die Hardwarebeschreibungssprachen VHDL und Verilog unterstützt und den Code für beide Sprachen farblich mit Syntaxhighlighting darstellen kann. Alternativ kann ein beliebiger externer Editor (wie z.B. SciTE, UltraEdit oder Notepad) angegeben werden.
Sobald ein neues Projekt erstellt ist, kann mit dem Anlegen und Schreiben eigener Entwürfe begonnen werden. Es empfiehlt sich jedoch zunächst die wichtigsten Einstellungen festzulegen.

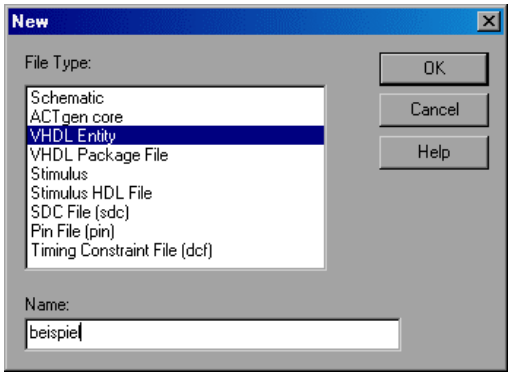
3.2 Editor-Auswahl und Optionen



Über den Menüpunkt **„Files→Preferences...“** wird der Einstellungs-Dialog angezeigt. Nach Klick auf das Auswahlfeld **„Text Editor“** kann entweder der in Libero integrierte Editor aktiviert werden **„→Use Libero Text Editor“**, oder man legt einen externen Editor fest **„→User defined text editor“** (dazu muss der Haken in der Checkbox **„Use Libero text editor“** entfernt werden)

3.3 Erstellen, Öffnen und Importieren von HDL-Dateien

3.3.1 Neue HDL-Dateien erstellen:



Nachdem das Projekt erzeugt und die wichtigsten Optionen eingestellt sind, kann eine neue HDL-Datei erstellt werden. Dazu müssen folgende Schritte durchgeführt werden:

1. Menüpunkt **„File→New“** anwählen (oder entspr. Button im Design Flow)
2. Auf **„VHDL Entity“** klicken und Dateinamen im unteren Feld **„Name“** eingeben und mit **„OK“** bestätigen (die Dateiendung kann weggelassen werden, sie wird autom. angehängt). Der HDL-Editor öffnet sich, das Design kann bearbeitet und über den Menüpunkt **„File→Save“** gespeichert werden. Die neue Datei wurde dem Projekt hinzugefügt und erscheint im Libero *„File-Manager“*.

3.3.2 HDL-Dateien öffnen oder importieren

Dateien öffnen:
Über den Menüpunkt **„File→Open...“** kann eine HDL-Quelldatei geöffnet und anschließend im Editor bearbeitet werden. Im Öffnen-Dialog sollte dazu der **Dateityp „HDL File (*.vhd, *.vhdl)“** ausgewählt werden. Die auf diesem Wege geöffneten Dateien werden nicht zum aktuellen Projekt hinzugefügt, sondern werden lediglich editiert!

Dateien importieren:
Soll eine Datei in das Projekt aufgenommen werden, so ist dies über den Menüpunkt **„File→Import Files...“** möglich. Es wird eine Kopie der ausgewählten Datei im Projektverzeichnis angelegt und in die Projekt-Hierarchie aufgenommen.

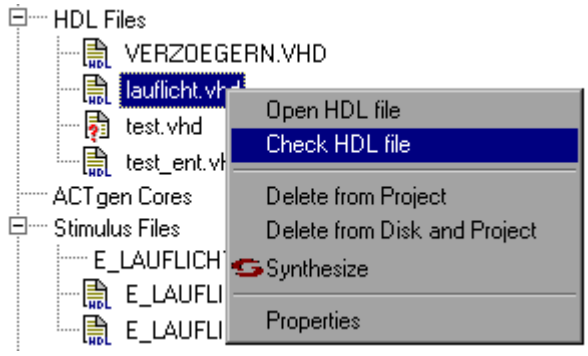
3.4 Besonderheiten des integrierten HDL-Editors

Mehrere geöffnete Dateien:
Der Libero HDL-Editor unterstützt mehrerer geöffnete Dateien, die jeweils über einen Tab-Reiter zur aktuellen Bearbeitung ausgewählt werden können.

Standard Editier-Funktionen:
Die normalen Windows Editier-Funktionen wie Copy/Paste usw. sind über Tastaturshortcuts oder das Kontextmenü der rechten Maustaste möglich.

Ein/Auskommentieren:
Sollen eine oder mehrere Zeilen im HDL-Code ein- bzw. auskommentiert werden, so geschieht dies am einfachsten mit Hilfe der Kommentar-Funktion **(„Comment out“** bzw. **„Uncomment out“**), die entweder über das Edit-Menü oder das Kontextmenü der rechten Maustaste aufgerufen werden kann.

3.5 Syntax-Checker



Nach dem Schreiben einer HDL-Datei ist es sinnvoll, den HDL-Code auf richtige Syntax zu überprüfen. Actel Libero bietet hierfür folgende Funktion an:

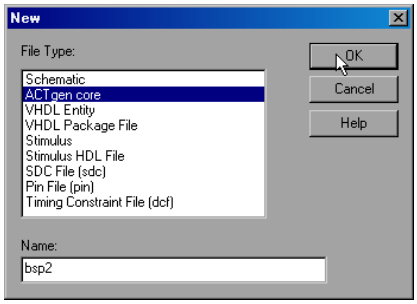
1. Wichtig: Alle Änderungen an der geöffneten HDL-Datei müssen gespeichert werden (Strg+S).
2. Im Libero **„File Manager“** mit der rechten Maustaste auf die zu prüfende Datei klicken und **„Check HDL File“** auswählen.

Die Datei wird überprüft und evtl. vorhandene syntaktische Fehler werden im unteren Log-Fenster angezeigt.

4. ActGen Core Builder – Arbeiten mit Macros

4.1 Einleitung

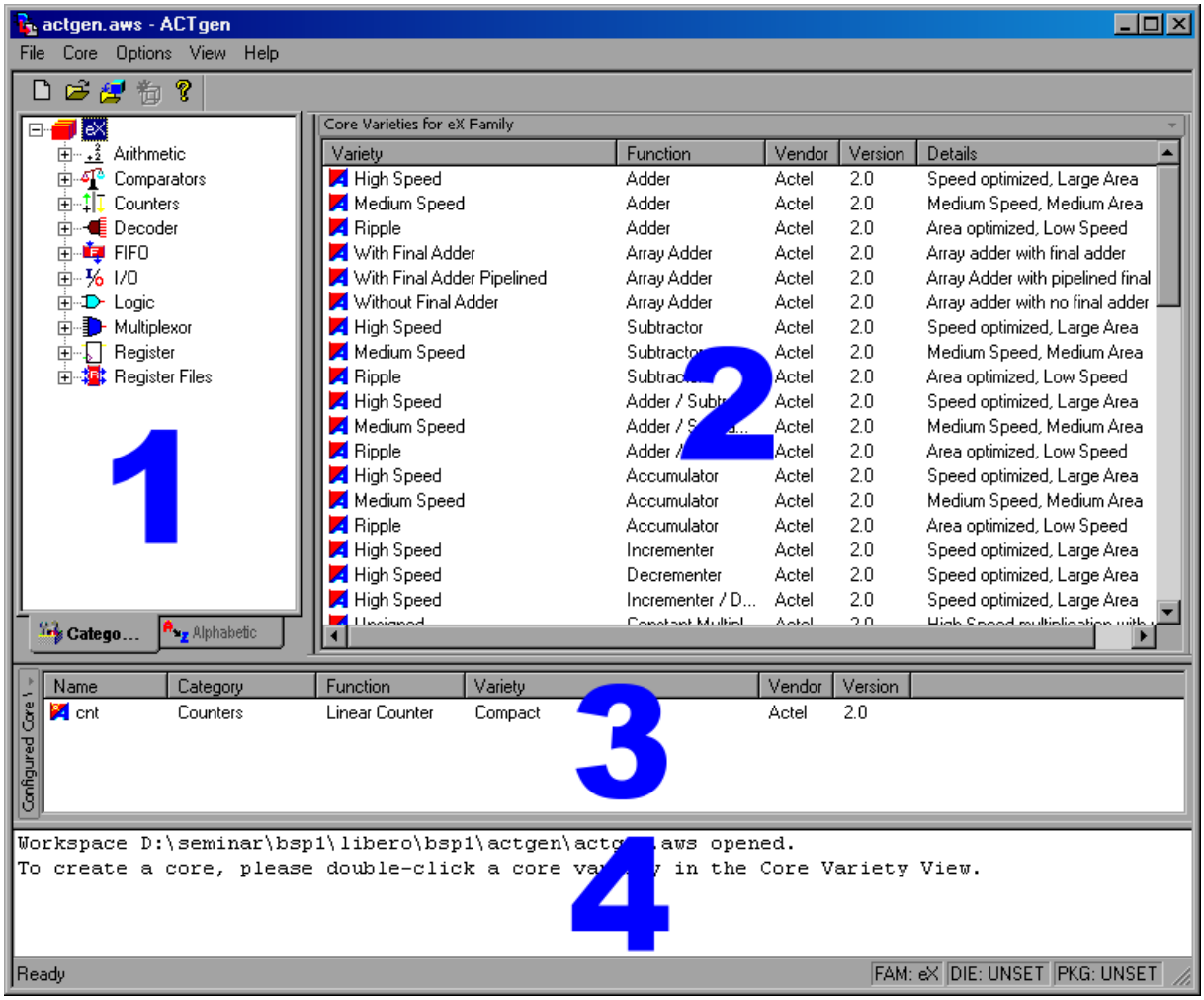
Actel Libero bietet mit dem ACTgen Core Builder die Möglichkeit, viele häufig benutzte Funktionen wie z.B. Zähler, Schieberegister, Addierer oder Multiplexer in das eigene Design zu übernehmen. Die gewünschte Funktion kann in einer grafischen Oberfläche ausgewählt, parametrisiert und als Netzliste (EDIF, VHDL, Verilog) oder Verhaltensmodell (zu Simulationszwecken) erzeugt werden. Die erzeugten Macros (Cores) können sowohl als Symbol im Schematic-Design, als auch im rein textuellen Design eingebunden werden.



Ein neues Makro kann über den Menüpunkt **„File→New“** oder einfacher mit dem **„ACTgen Button“** im Design Flow erzeugt werden. Nach der Eingabe eines Namens und klick auf **„OK“**, wird der Core Builder als neues Programm gestartet. Dort werden die Macros mit den gewünschten Optionen erzeugt und automatisch mit in das Projekt aufgenommen.

4.2 Aufbau der Programmoberfläche

Wie der folgende Screenshot demonstriert, ist die Oberfläche in vier Bereiche aufgeteilt:



(1) Core Catalog:
In diesem Fenster werden alle zur aktuell ausgewählten Hardware-Zielplattform zur Verfügung stehenden Macros (Cores) angezeigt (entweder nach Kategorien oder alphabetisch sortiert). In der Baumansicht kann das gewünschte Element ausgewählt werden.

(2) Variety View Fenster:
Ist ein Element im Catalog selektiert, dann sind in diesem Fenster die möglichen Variationen passend zur Auswahl aufgelistet (wenn z.B. im Catalog „*Counters*“ ausgewählt wurde, stehen hier „*Sklansky, Fast Brent-Kung, Ripple...*“). Per Doppelklick auf das gewünschte Element erscheint ein Dialog zur Parametrisierung und Erzeugung des Cores.

(3) Configured Core View Fenster:
In diesem Fenster (Workspace) werden alle bereits konfigurierten Cores angezeigt und können per Rechtsklick verändert oder entfernt werden.

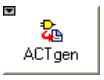
(4) Log Fenster:
Das Log-Fenster zeigt Informationen zu den konfigurierten Cores an

4.3 Beispiel: Wie binde ich einen Zähler in mein HDL-Design ein

Das folgende Beispiel demonstriert das Generieren und Einbinden von ACTgen Cores in einen VHDL-Entwurf für den Baustein *„eX256-128CS“*.
Es soll dazu ein Zähler entworfen werden, der von 0 bis 7 (3 Bit) zählt und anschließend automatisch einen Reset durchführt und wieder von vorne beginnt. Der Zähler wird dabei nicht vom Programmierer durch entsprechenden VHDL-Code beschrieben, sondern mit Hilfe eines Macros generiert, den gewünschten Bedürfnissen angepasst und schließlich in das Projekt in VHDL integriert.

1. Schritt:
Anlegen der VHDL-Datei, in der das Macro verwendet werden soll (darauf wird hier nicht näher eingegangen, das vorige Kapitel beschreibt, wie ein VHDL-Design erstellt werden kann).

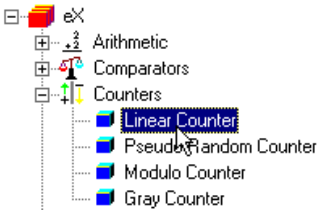
2. Schritt:

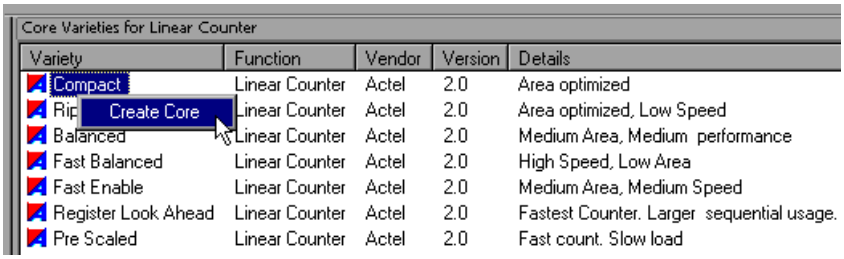


Einen neuen Core für den Zähler erstellen: Dazu wird im Hauptfenster von Libero im Design Flow der ACTgen-Button (links zu sehen) angeklickt.

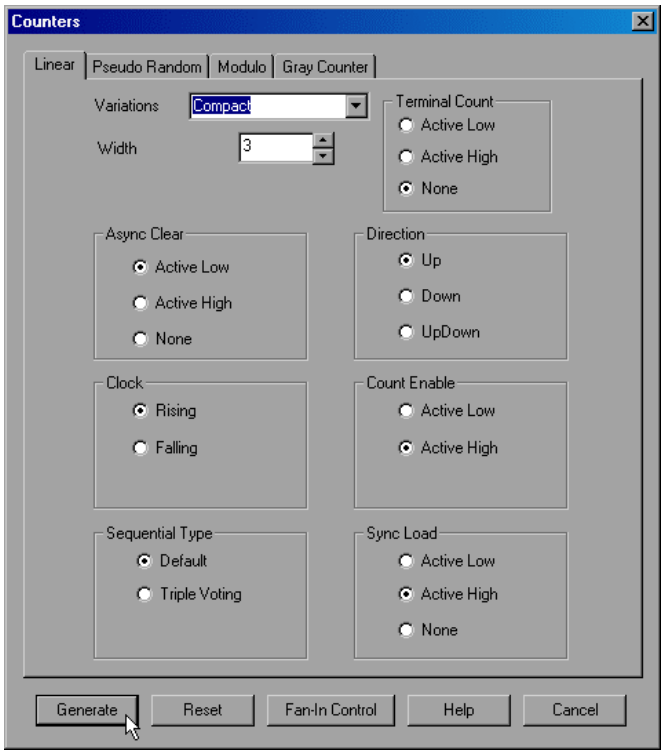
Im sich darauf öffnenden Dialog wird **„ACTgen Core“** ausgewählt und ein Name (z.B. *„my_cnt“*) angegeben. Nach Bestätigung wird der ACTgen Core Builder geladen und erscheint am Monitor.

3. Schritt:
Wie auf dem Screenshot zu sehen, muss jetzt im linken Fenster ***„Core Catalog“*** die gewünschte Funktion angewählt werden.
Für unseren Zähler wählen wir **„Counters→Linear Counter“**. Im rechten Fenster werden daraufhin die unterstützen Variationen aufgelistet. In der **„Details“**-Spalte sind die Unterschiede erläutert:

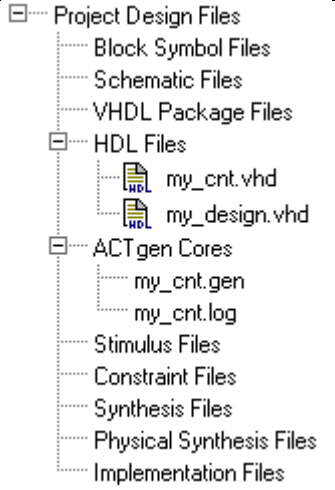




4. Schritt:
Hat man sich für eine der angebotenen Variationen entschieden (für dieses Beispiel wurde *„Compact“* ausgesucht), wird diese selektiert. Dann kann mit Hilfe des Kontextmenüs der rechten Maustaste **„Create Core“** gewählt werden, woraufhin ein neuer Dialog mit den Parametern erscheint.



Die Parameter für den neuen Zähler können bequem über die Steuerelemente im Dialogfenster eingestellt werden. Über den **„Help“-Button** erfährt man nähere Informationen zum ausgewählten Core und die Bedeutung seiner Parameter. Da unser Counter von 0 bis 7 zählen soll, muss mit drei Bit gearbeitet werden, was unter **„Width“** angegeben werden kann.
Sind alle Einstellungen gemacht, wird über den **„Generate“-Button** das Erzeugen des Macros gestartet, woraufhin erneut ein Dialog auftaucht, indem der Dateiname und Typ des neuen Cores sowie das Netzlistenformat (VHDL, Verilog, EDIF) angegeben werden muss. Für das Beispiel wird „VHDL“ gewählt, ein Verhaltensmodell („Behavioral Model“) wird nicht benötigt. Als Verzeichnis sollte stets das aktuelle Projektverzeichnis beibehalten werden, damit das neue Macro im Workspace des ACTgen Core Builders auftaucht und bearbeitet werden kann.
Nach dem Speichern wurden folgende Dateien angelegt (der Dialog schließt sich leider nicht von selbst, sondern muss über **„Cancel“** von Hand geschlossen werden):



Dateien:

my_cnt.vhd:
Die VHDL-Beschreibung des Cores. Enthält die Entity „my_cnt“ (und zugehörige Architektur) des Zählers:

```
entity my_cnt is
 port( Data : in std_logic_vector(2 downto 0);
       Enable, Sload, Aclr, Clock : in std_logic;
       Q : out std_logic_vector(2 downto 0)) ;
end my_cnt;
```

my_cnt.gen:
Hierin werden die Macro-Parameter gespeichert, die mit dem ACTgen Core Builder geändert werden können.

my_cnt.log:
Diese Log-Datei enthält alle Details zu den ausgewählten Parametern der erstellten Cores in reinem Textformat.

5. Schritt:
Die Funktion des erstellten Macros kann wie in VHDL üblich durch Port-Mapping erreicht werden. Dazu muss eine Komponente der Counter-Entity erstellt und instanziiert werden, wie der folgende Codeausschnitt demonstriert:

Code-Ausschnitt zum Einbinden des Cores:

```
entity E_MY_DESIGN is
    port ( CLK:         in  std_logic;
           RST:         in  std_logic;
           STOP:        in  std_logic;
           LED:         out std_logic_vector(3 downto 0)
    );
end E_MY_DESIGN;

architecture A_BSP of E_MY_DESIGN is
type   t_zustand is (zLED1, zLED2, zLED3, zLED4);
...
component my_cnt is
    port( Data                      : in std_logic_vector(2 downto 0);
          Enable, Sload,  Aclr, Clock : in std_logic;
          Q                         : out std_logic_vector(2 downto   0)) ;
end component;
...
signal INT_ZUSTAND   : t_zustand;
signal tq : std_logic_vector(2 downto   0);
begin
...
 my_cnt_inst : my_cnt PORT MAP ( "000", '1', '0', '1', CLK, tq );
...
end A_BSP;
```

5. Schematic Based Entry

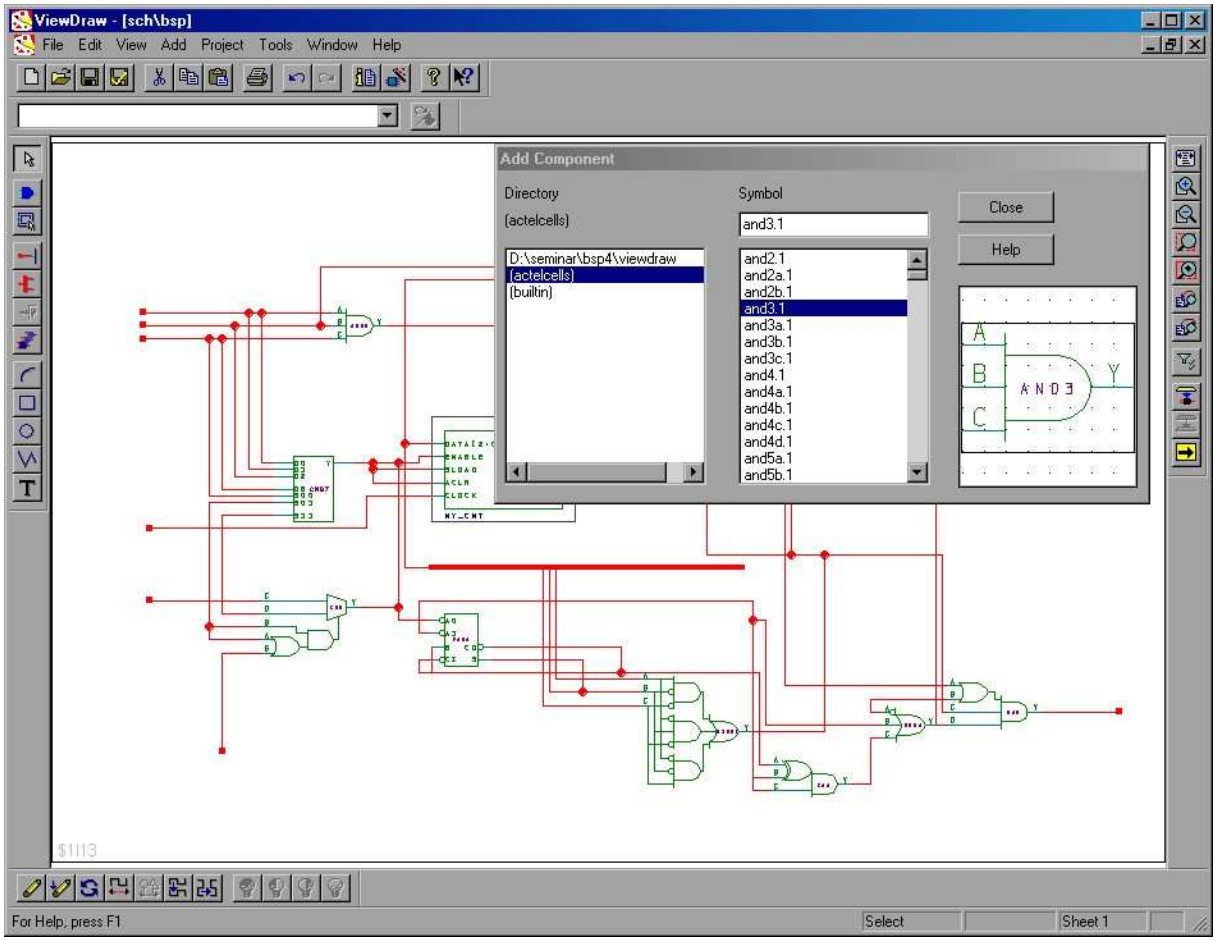
5.1 Einleitung

Actel Libero bietet dem Entwickler neben der textuellen VHDL-Entwurfsebene die Möglichkeit ein vollständiges Design in einer grafischen Oberfläche zu erstellen. So kann mittels eines grafischen Editors (Schematic-Editor) ein Schaltplan entworfen werden, der die Strukturbeschreibung eines Entwurfs auf der Logikebene darstellt. Actel liefert dazu das angepasste Fremdtool **„View Draw AE“** von Mentor Graphics mit, welches aus der Libero IDE heraus gestartet werden kann und sich so in den Design Flow integriert. In View Draw AE wird das Design dann mit Hilfe von Menüs, Dialogfenstern, Toolbars und Buttons mit der Maus formuliert. Da sich die beiden Entwurfsebenen (Text-based und Schematic-based) auch kombinieren lassen (Mixed-mode), kann der Design Flow beschleunigt und optimiert werden.

Wie wird in View Draw AE gearbeitet?

Im Schematic-Editor „**View Draw AE“** arbeitet der Entwickler mit unterschiedlichen Blöcken, die durch grafische Symbole dargestellt werden und in einer Hierarchie auf verschiedenen Ebenen (Levels) stehen. Auf der untersten Ebene stehen die Grundelemente (Primitive) wie z.B. Logikgatter AND, OR ... Werden mehrere Blöcke in einem Schaltplan zusammengefasst, entsteht ein neuer Block (composite block), der wiederum in anderen Designs eingesetzt werden kann. Die Funktionalität ergibt sich durch das Zusammenschalten und Verknüpfen der eingesetzten Symbole auf einem oder mehreren Arbeitsblättern (sheets).

5.2 Die Oberfläche von ViewDraw AE



Wie auf dem Screenshot zu erkennen besteht das View Draw AE Fenster im wesentlichen aus dem weißen Arbeitsblatt (in der Mitte des Fensters), auf dem der Schaltplan zusammengebaut wird. Um die Arbeitsfläche herum sind verschiedene Toolbar Buttons, die den schnellen Zugriff auf häufig genutzte Funktionen oder Zeichenkomponenten erlauben.

Die obere Toolbar bietet wichtige Standard-Aktionen, wie Laden, Speichern oder Copy & Paste.

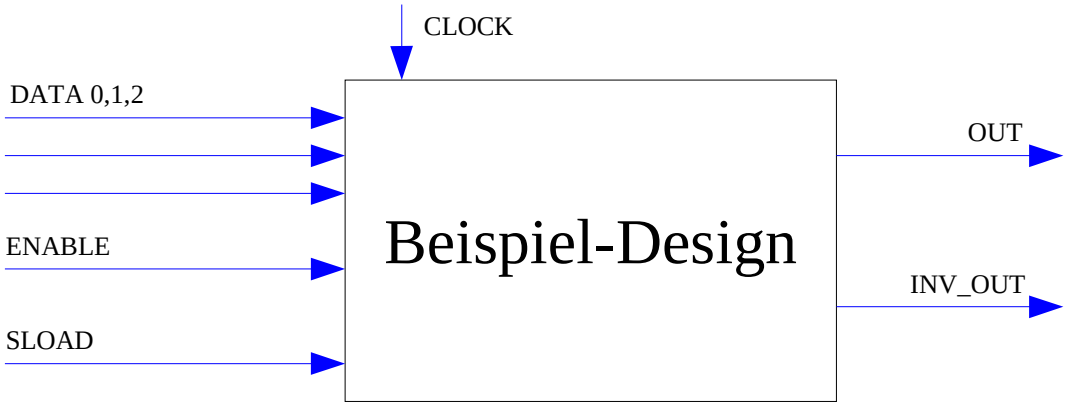
Die linke Toolbar enthält die wichtigsten Elemente um neue Komponenten auszuwählen oder Leitungen, Pins und Busse zu zeichnen. So wird nach Klick auf den **„Add Component-Button "** das entsprechende Dialogfenster angezeigt (wie im Screenshot zu sehen). Dort kann aus verschiedenen Verzeichnissen eine Komponente ausgewählt und mit der Maus per Drag & Drop auf das Arbeitsblatt gezogen werden. Nach Klick auf den **„Net-Button "** können die einzelnen Symbole dann durch Zeichnen von Leitungen miteinander verknüpft werden. Die unteren Buttons enthalten Zeichenfunktionen (z.B. für Rechtecke, Kreise, Linien..) um Schematic-Symbole anzupassen.

Die rechte Toolbar enthält unterschiedliche Zoomfunktionen. Relativ wichtig ist der unterste Button dieser Toolbar, der einen Wechsel zwischen mehreren Arbeitsblättern ermöglicht.

Die untere Toolbar beinhaltet Funktionen zum Manipulieren einzelner Objekte. Damit ist es möglich Blöcke zu löschen, zu drehen, zu skalieren, zu strecken oder zu spiegeln.

5.3 Ein Beispielentwurf - Arbeiten mit ViewDraw

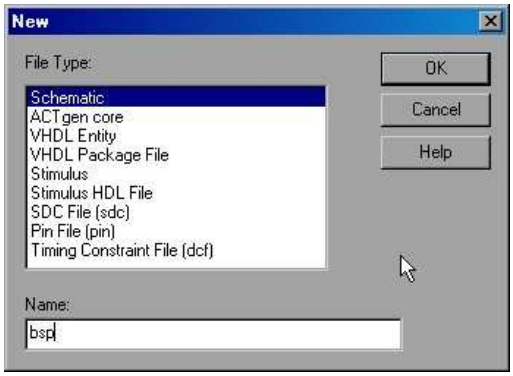
Anhand des folgenden Beispiels soll das Erstellen eines Schematic-Designs mit View Draw AE demonstriert werden. Der Beispielentwurf soll mit Hilfe eines 3-Bit-Zählers ein Ausgangssignal erzeugen, welches vom Zählerstand abhängig ist und nur dann High-Pegel besitzt, wenn eines der drei Bits logisch 1 ist, oder alle drei Bits gesetzt sind.
Dazu wird eine simple Logikauswertung eines ACTgen-3-Bit-Zählers mit folgender Struktur aufgebaut:



Alle *Eingangswerte* beziehen sich auf den internen 3-Bit-Zähler. Wird an **DATA 0,1,2** eine binäre Zahl angelegt, kann der Counter mittels High-Pegel am **SLOAD**-Eingang initialisiert werden. Damit der Zähler arbeitet, muss außerdem das **ENABLE**-Signal auf **„High“** gesetzt sein.
Am *Ausgang* ist lediglich ein logisches Signal **„OUT“** hinausgeführt, **„INV_OUT“** ist das gleiche Signale, jedoch invertiert. **„OUT“** soll immer dann auf High-Pegel sein, wenn exakt eines der drei Zähler-Ausgangs-Bits 1 ist oder wenn alle Bits gesetzt sind.

1. Schritt: Eine neue Schematic Datei erzeugen

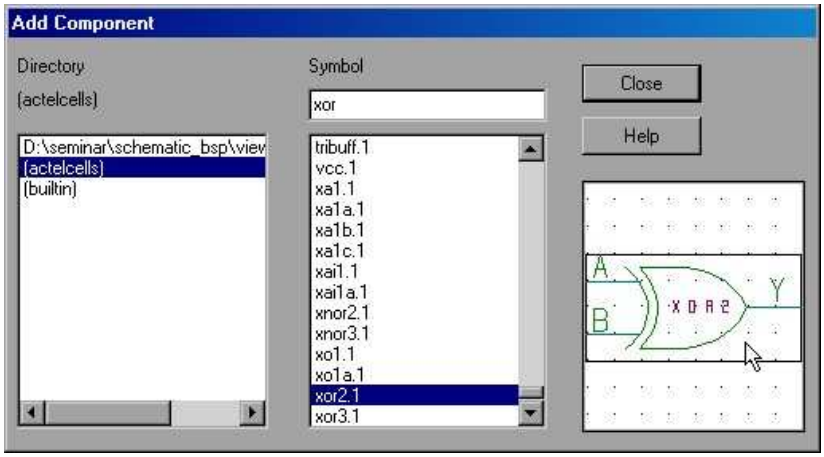
Ist ein neues Libero Projekt erstellt wurden, so kann dort im Design Flow der **„View Draw-Button“** (links zu sehen) angeklickt oder alternativ der Menüpunkt **„File →New“** angewählt werden.



Im sich öffnenden Dialogfenster **„New“** wird nun der File Type **„Schematic“** selektiert und anschließend ein Name (im Beispiel „bsp“) für das neue Schematic Objekt angegeben. Nach Bestätigung mit **„OK“** wird das Programm View Draw AE in einem eigenständigen Fenster gestartet und die Eingabe kann beginnen. Über den Menüpunkt **„Project→Settings“** können vorab diverse Einstellungen (wie z.B. im Tab-Reiter „Color Palette“ die Auswahl eines anderen Farbschemas) vorgenommen werden.

2. Schritt: Komponenten hinzufügen

Nun kann die Eingabe und Verknüpfung der Symbole beginnen. Über den Button **„Add Component-Button**  **“** der linken Toolbar oder über die **„c“-Taste** wird zunächst das Dialogfenster geöffnet, in dem die einzelnen Komponenten aufgelistet sind:



In der linken Liste (Directory) befinden sich die zur Verfügung stehenden Komponentensammlungen. Das oberste Verzeichnis (D:\seminar\schematic_bsp\...) bezieht sich auf Symbole des aktuellen Projekts. Der Eintrag **(actelcells)** enthält die Standard-Komponenten von Actel, aus der für dieses Beispiel ein XOR-Symbol **„xor2.1“** (2 Eingänge, 1 Ausgang) mit zwei Eingängen ausgewählt wurde (wie im Vorschaufenster zu erkennen). Um dieses Symbol nun im Design verwenden zu können, muss es über den Symbolnamen oder dem Vorschaubild mit der Maus in die Arbeitsfläche gezogen werden (Drag & Drop). Das Dialogfenster kann während des Designvorgangs geöffnet bleiben, so muss es nicht für jedes neue Symbol wieder geöffnet werden.

Da der Zähler in unserem Beispiel drei Ausgangsbits haben soll, wird noch ein zweites XOR-Glied benötigt. Dieses kann entweder wieder aus dem **„Add Component“**-Dialogfenster gezogen werden, eine andere Möglichkeit ist jedoch, das bereits auf dem Arbeitsblatt vorhandene Symbol zu kopieren:

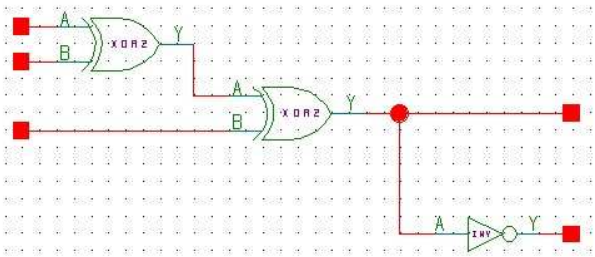
3. Schritt: Komponenten kopieren

Um das erste XOR-Glied zu kopieren wird es zunächst per Linksklick selektiert. Dann wird die STRG-Taste gedrückt gehalten und dabei eine Kopie des markierten Objekts zur gewünschten Stelle auf dem Arbeitsblatt geschoben. Alternativ kann ein Objekt nachdem es selektiert wurde mit STRG+C (bzw. STRG+X) kopiert (bzw. ausgeschnitten) werden und an einer anderen Stelle mit STRG+V und linkem Mausklick erneut eingefügt werden.

Für den negierten Ausgang „INV_OUT“ unseres Beispiels wird außer den beiden XOR-Gliedern noch ein Inverter für den invertierenden Ausgang benötigt. Dieser hat den intuitiven Namen **„inv.1“**.
(Statt den beiden XOR-Gliedern könnte natürlich auch ein XOR-Glied mit drei Eingängen verwendet werden: „xor3.1“).

4. Schritt: Komponenten miteinander verbinden

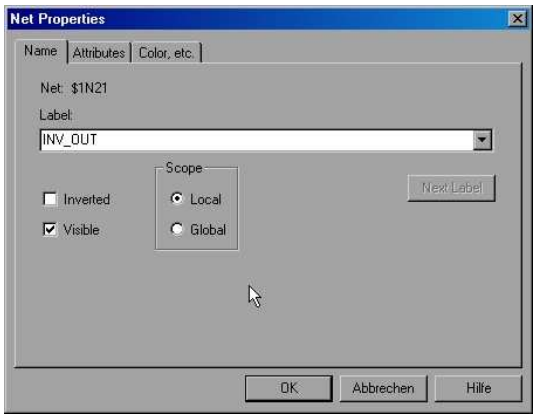
Nachdem die Symbole auf der Arbeitsfläche abgelegt wurden, können sie nun miteinander verbunden werden. Dazu wird zunächst der **„Net-Button**  **“** aktiviert. Anschließend werden die Leitungen mit der Maus gezeichnet, dabei wird jeweils an den Symbol-Anschlüssen begonnen:



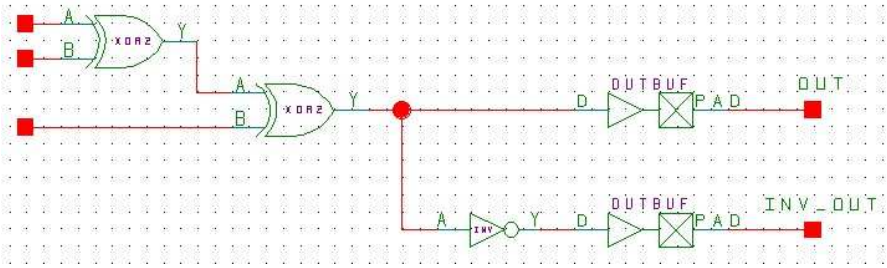
Tipp: Während des Zeichnens einer Leitung kann durch klicken der rechten Maustaste ein Stützpunkt eingefügt werden! Dadurch sind auch komplizierte Bahnen realisierbar.

5. Schritt: Ein- und Ausgänge (I/O) definieren

Damit die Ein- und Ausgänge eines Designs (in und out einer Entity) für die Beschreibung und Simulation gültig sind, müssen diese im Schematic-Editor mit Hilfe von **„inbuf.1“** und **„outbuf.1“** Symbolen eingezeichnet werden. Dies geschieht nach demselben Prinzip wie das Einfügen der XOR-Komponenten. Das Symbol wird auf die Arbeitsfläche gezogen und mit den entsprechenden Signalen verbunden.



Vor- bzw. hinter dem Ein-/Ausgangs-Buffer wird noch eine kleine Leitung gezeichnet. Mit Doppelklick auf diese Leitung öffnet sich der **„Net Properties“** Dialog, in dem unter **„Label“** ein Name für den Ein- bzw. Ausgang eingegeben werden kann, in unserem Design sind das zunächst nur die Ausgänge „OUT“ und „INV_OUT“ (die Eingänge werden später mit dem Counter hinzugefügt). Der Name wird nun über der Leitung angezeigt und kann mit der Maus zur besseren Übersicht verschobene werden:

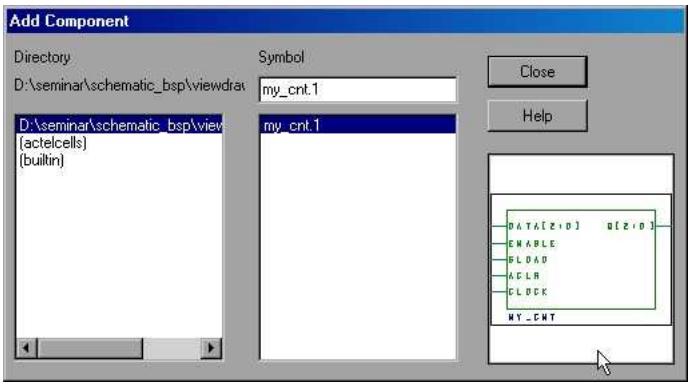


6. Schritt: Kommentare und Grafikelemente hinzufügen

Damit das Design übersichtlicher und aussagekräftiger wird, ist es evtl. nützlich Kommentare oder grafische Symbole (z.B. Gruppierungskästen...) in den Schaltplan einzuzeichnen. View Draw AE bietet mit der linken Toolbar dazu die Möglichkeit. Die Zeichenelemente haben keinerlei Einfluss auf den eigentlichen Schaltplan.

7. Schritt: Erzeugen und Einbinden eines Zählers (ACTgen Core)

Nun soll der Counter in den Schematic-Schaltplan integriert werden. Dazu wird zunächst (wie im vorigen Kapitel beschrieben) mit Hilfe des ACTgen Core Builders ein Macro für den Zähler erzeugt (Counters→Linear→Compact→3-Bit-Breite) und dem Libero Projekt hinzugefügt, so dass er in der Libero Design Hierarchie auftaucht. Dort wird das ACTgen Modul (hier: **my_cnt (my_cnt.vhd)**) mit der rechten Maustaste angeklickt. Im erscheinenden Kontextmenü wird **„Create Symbol“** gewählt – ein neues Schematic-Symbol für den Zähler wird erzeugt und im File Manager unter **„Block Symbol files“** angezeigt.

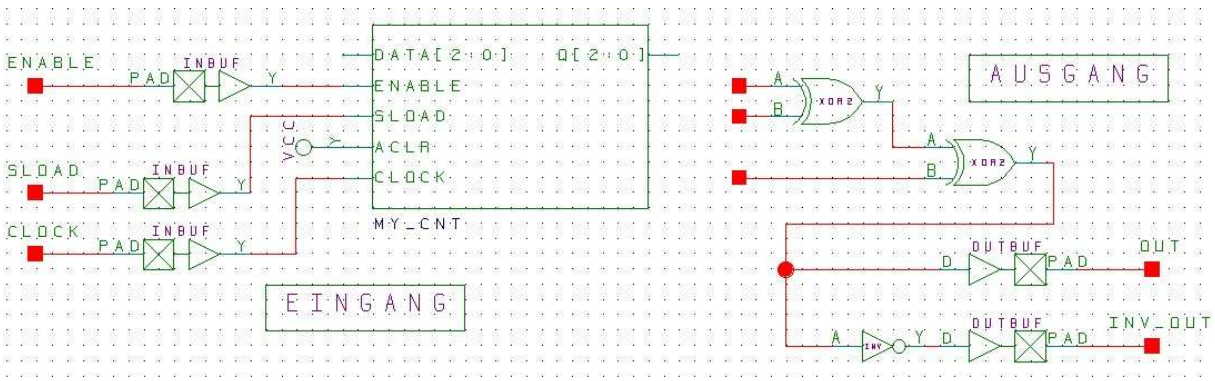


Im nächsten Schritt wird der Schaltplan wieder in View Draw AE geöffnet. Wie zuvor wird über den **„Add Component-Button“** das **„Add Component Dialogfenster“** angezeigt. Diesmal wird als Directory jedoch nicht „(actelcells)“ sondern der aktuelle Projektpfad selektiert (hier: „D:\seminar\schematic_bsp\...“).

Der Counter erscheint als Symbol und kann auf das Arbeitsblatt gezogen werden. Anschließend werden die einfachen Eingänge mit der restlichen Schaltung verbunden (DATA und Q werden erst später verdrahtet, da es sich hierbei nicht um einzelne Leitungen handelt. Diese Anschlüsse sollen später über einen BUS zusammengeschlossen werden).

Festen Low oder High-Pegel anschließen (VCC, GND):
Der ACLR-Eingang des Counters wird in unserem Design nicht benötigt und soll immer logisch 1 sein (1 wegen Active Low), da kein „Clearen“ erfolgen darf. Dazu wird er mit dem Symbol **„vcc.1“** verbunden (logisch 0 wäre entsprechend **„gnd.1“**.

Die erweiterte Schaltung sieht mit entsprenden Eingängen und Verschaltung wie folgt aus:

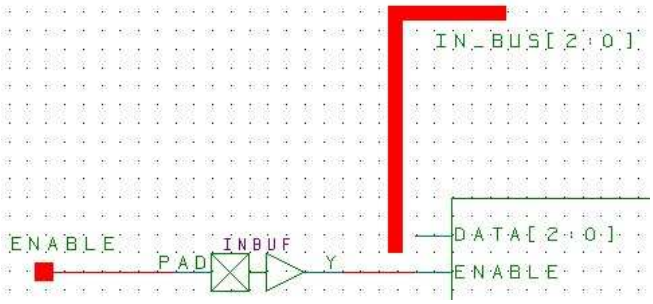


8. Schritt: Objekte manipulieren

View Draw AE bietet verschiedene Möglichkeiten die Symbole im Schematic-Editor zu manipulieren. Dazu ist am unteren Fensterrand eine eigene Toolbar. Insbesondere die Drehen- und Spiegelfunktionen erlauben mehr Flexibilität bei der Schaltplaneingabe und übersichtlichere Ergebnisse.

9. Schritt: Erstellen eines Busses

Zum Schluss müssen die beiden fehlenden Ports „DATA“ und „Q“ angeschlossen werden. Da es sich um einen 3-Bit Zähler handelt sind diese beiden Ports vom VHDL-Typen „std_logic_vector(2 downto 0)“ und sollen jeweils über einen BUS zusammengefasst werden.

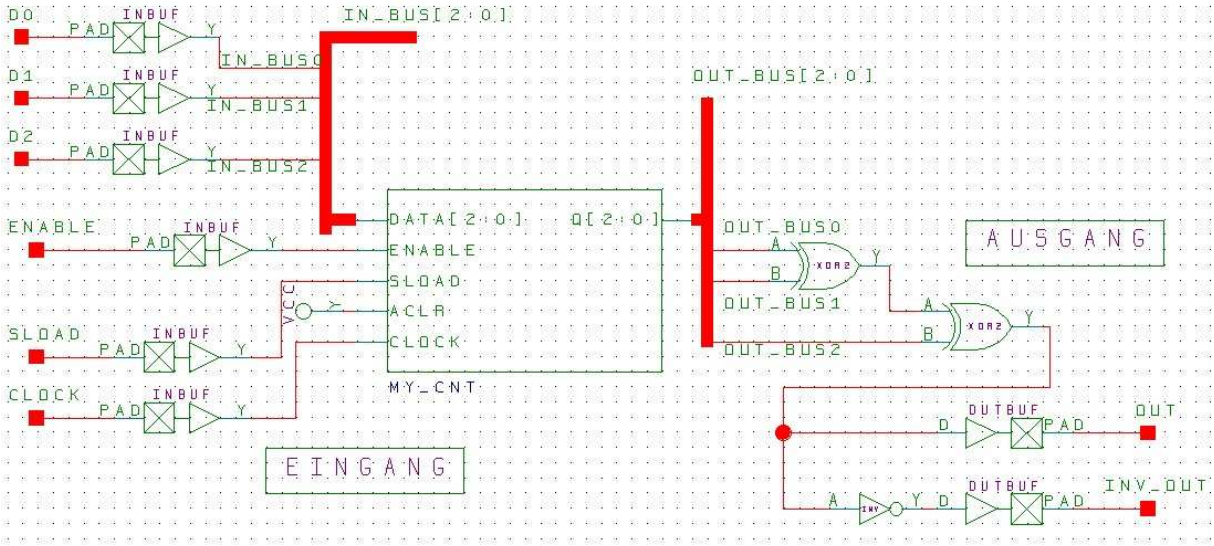


Zu Beginn werden die beiden BUSSE frei in den Schaltplan eingezeichnet, ohne das andere Signale berührt werden. Nach Aktivieren des **„BUS-Buttons** **“** kann das Zeichnen beginnen. Ist der BUS eingezeichnet, muss er per Doppelklick mit einem Namen versehen werden, der auch die Bitbreite (→std_logic_vector) angibt.

In unserem Beispiel wählen wir den Namen **„IN_BUS[2:0]“** bzw. **„OUT_BUS[2:0]“** und geben diesen jeweils im **„Label-Feld“** des **„Net Properties“-Dialogs** ein.

Als nächstes werden die Ein- und Ausgänge an den Bus angeschlossen. Dies geschieht über die bekannte Netzverbindung (mittels „Net-Button“). Jeder Anschluss muss dann einen BUS-spezifischen Namen bekommen, der angibt welches Bit gemeint ist. Der Name muss mit dem des BUSSES übereinstimmen, jedoch statt der Klammer wird die Zahl für das gewünschte Bit angegeben (z.B: **„IN_BUS0“, „IN_BUS1“**..).

Das betrachtete Beispieldesign bekommt nach Ergänzen der fehlenden Eingangssignale und Anschließen der BUS-Komponenten folgendes Aussehen:



10. Schritt: Speichern und Testen der Schaltung

Selbstverständlich muss die fertige Schaltung abgespeichert werden (sicherer ist es auch zwischendurch mehrmals zu speichern), dies geschieht entweder über den Speichern-Button bzw. über den Menüpunkt **„File→Save“** oder über die **„Save+Check Funktion“**. Das hat den Vorteil, dass die Schaltung zusätzlich überprüft wird (ob alle Anschlüsse korrekt verbunden sind) und evtl. vorliegende Fehler ausgegeben werden.

5.4 Besondere Funktionen in View Draw AE

5.4.1 Mehrere Schematic-Arbeitsblätter verwenden (multi-page)

Da der Platz auf einem Schematic-Arbeitsblatt nur begrenzt ist, bietet View Draw AE dem Entwickler die Möglichkeit an, mehrere Arbeitsblätter (sheets) zu kombinieren (multi-page).

Ein neues Blatt kann über den **„Goto Page-Button "** angelegt werden, indem zu einem neuen, noch nicht vorhandenen Blatt gewechselt wird. Ein leeres Blatt wird aufgemacht und ein neuer Schematic-Entwurf kann begonnen werden. Über denselben Button wird zwischen den einzelnen Arbeitsblättern hin- und her gewechselt.

5.4.2 Was sind Fubes?

View Draw AE bietet dem Entwickler mit Hilfe von Fubes die Möglichkeit, vorläufig eine „schwarze Kiste" in das Design zu integrieren. Dies ist ein Block, der zwar schon fest im Design verdrahtet ist, jedoch noch keinerlei Funktionalität abdeckt. Fubes können als „Under Construction"-Symbole interpretiert werden. Sie werden in der weiteren Entwurfsphase irgendwann durch „reale" Module ersetzt, stehen aber während des Entwicklungsverlaufs bereits an der richtigen Stelle, können mit beliebigen Anschlüssen oder Bussen versehen werden (die Pins werden autom. aktualisiert!) . Zur Übersicht können die Pins mit Namen versehen werden.
Fubes können mit Hilfe des **„Fub-Button "** auf der Arbeitsfläche platziert werden.

5.4.3 Neue Symbole erzeugen und einbinden

Der Schematic-Editor View Draw AE ermöglicht es dem Entwickler, eigene Symbole zu erzeugen, die dann in späteren Entwürfen eingesetzt werden können.
Grundsätzlich können vier verschiedene Arten von Symbolen erzeugt werden – jeweils für unterschiedliche Anwendungszwecke:

1. Module Symbol (repräsentieren Basisfunktionen für das Design)
2. Zusammensetzungs Symbol (composite symbol, besitzen eine untergeordnete Schaltung, Schematic)
3. Pin Symbol (repräsentieren einen Port oder Interface in einem Schematic-Design)
4. Kommentar Symbol (annotate symbol, grafische Symbole für Dokumentationszwecke, ohne schaltungsspezifische Eigenschaften)

Das Generieren dieser Symbole erfolgt mit Hilfe des **„Symbol Wizard "** Assistenten, der über den entsprechenden Button der oberen Toolbar aufgerufen werden kann.

6. Stimulus / Testbench - WaveFormer Lite

6.1 Einleitung

Zum Paket von Actel Libero wird das Fremdtool WaveFormer Lite AE von SynaptiCAD mitgeliefert. Dieses Programm ermöglicht das grafische Eingeben von individuellen Signalverläufen, aus denen sich schnell und einfach eine HDL Testbench exportieren lässt. Diese kann direkt mit dem Simulator ModelSim ausgewertet oder vorher noch per Hand angepasst werden (im VHDL Code). Trotz der eingeschränkten „Lite“ Version, in der nicht alle Funktionen enthalten sind, ist das Tool auf den Libero Design Flow abgestimmt. So werden die Signalinformationen automatisch aus dem HDL-Design gelesen und stehen bereits beim Programmaufruf zur Verfügung.

6.1.1 Programmstart aus Libero

WaveFormer Lite kann innerhalb eines Libero Projekts direkt aufgerufen werden, um eine Testbench zu erzeugen. Dazu muss im Design Flow der **„Stimulus-Button“** (links abgebildet) per Linksklick angewählt werden. Daraufhin startet das Tool und lädt automatisch die Signalinformation zum aktuell festgelegten Hauptdesign (welche Design-Einheit das ist, kann in der **„Design Hierarchy“** von Libero per Rechtsklick über den Menüeintrag **„Set As Root“** eingestellt werden; falls es nur eine Design-Einheit gibt, entfällt die Option).

6.1.2 Aufrufoptionen festlegen

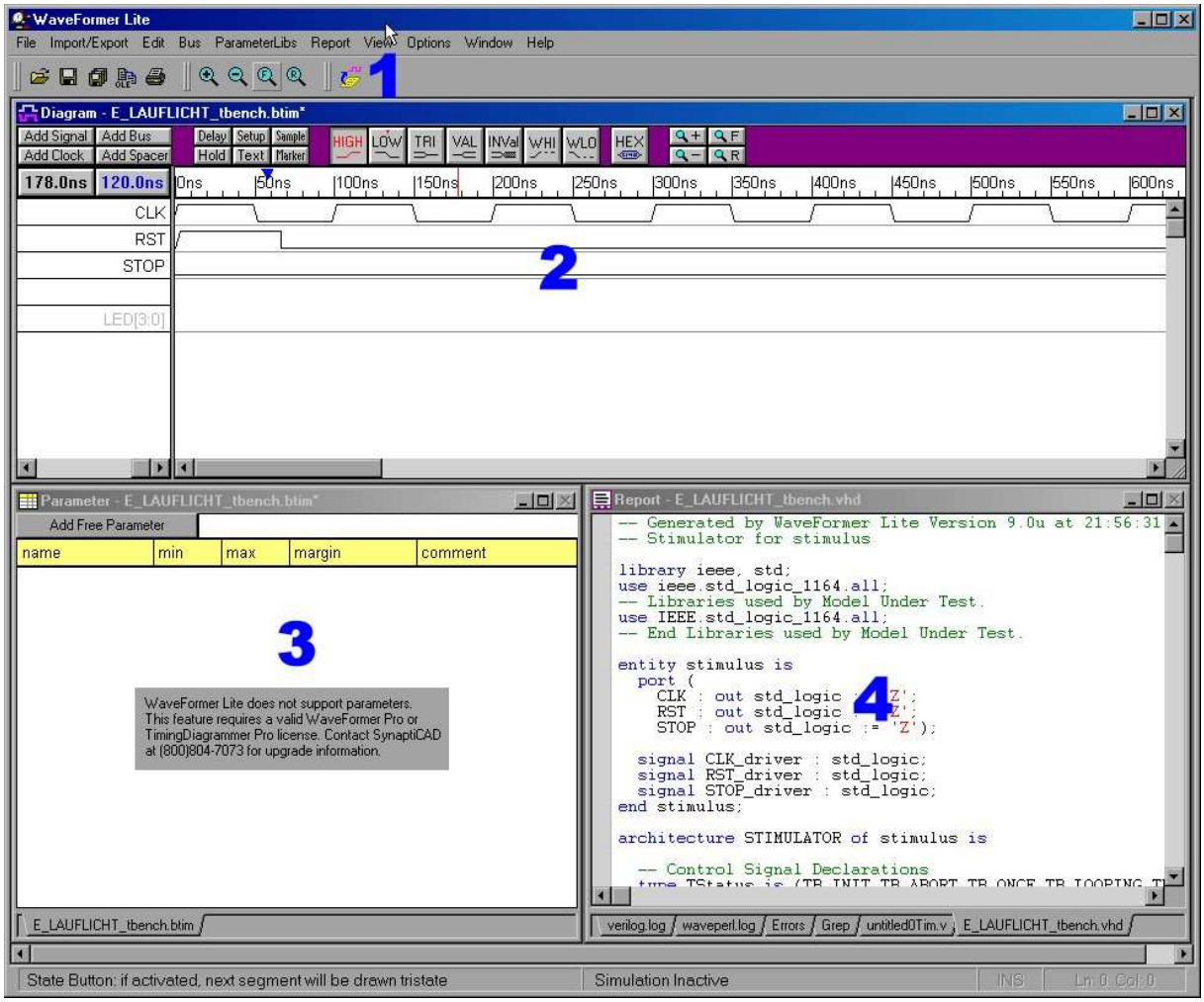
Wird der **„Stimulus-Button“** per Rechtsklick angewählt, erscheint ein Kontextmenü mit mehreren Optionen zur Stimulus-Erzeugung. Über den Eintrag **„Select Stimulus“** kann ein Dialogfenster geöffnet werden, in dem die unterschiedlichen, vorhandenen Stimulus-Dateien für das Projekt aktiviert werden können. Außerdem kann eine neue Stimulus-Datei erzeugt oder geöffnet werden.

6.2 Die grafische Oberfläche von WaveFormer Lite

WaveFormer Lite wird stets in einem eigenen Fenster geöffnet. Der Aufbau der Oberfläche wird im Screenshot auf der nächsten Seite demonstriert.
Das Programm enthält unter der Menüleiste eine Button Toolbar mit den wichtigsten Standard-Funktionen wie Laden und Speichern sowie Zoom-Optionen für die Anzeige der Signalverläufe (**Nr. 1** im Screenshot).
Im Client Bereich können drei verschiedene Fenster frei angezeigt und verschoben werden, so dass sowohl alle drei Fenster sichtbar sein können, als auch nur ein Fenster im Vordergrund stehen kann:

- **Nr. 2** – Diagramm Fenster
- **Nr. 3** – Parameter Fenster
- **Nr. 4** – Report Fenster

Die folgende Abbildung zeigt die Programmoberfläche:



***Nr. 2** – Diagramm Fenster:*
Dieses Fenster ist das Herzstück von WaveFormer Lite. In der Mitte werden die Signalverläufe der Signale angezeigt bzw. editiert, links daneben stehen die zugehörigen Signalnamen. Darüber sind Buttons mit wichtigen Funktionen (z.B: für die Eingabe der Signalpegel / Wechsel)

***Nr. 3** – Parameter Fenster:*
Dieses Fenster kann ignoriert bzw minimiert werden, da Parameter nur in der PRO-Version von WaveFormer freigeschaltet sind.

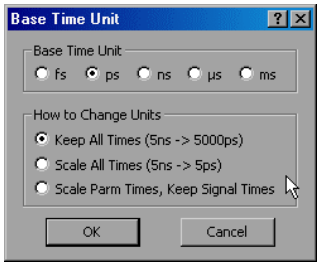
***Nr. 4** – Report Fenster:*
Hier werden verschiedene Informationen und Hinweise angezeigt. Nach dem Exportieren einer Testbench kann hier der erzeugte VHDL-Code editiert werden. Da dies jedoch auch in Libero möglich ist, spielt auch dieses Fenster eine eher untergeordnete Rolle.

6.3 Wichtige Optionen

Bevor mit der Erstellung und dem Editieren der Waveforms (Signalverläufe) für die individuellen Design-Tests begonnen wird, sollten einige Optionen angepasst werden, da die Standard-Vorgaben nicht für jedes Projekt übernommen werden können.

6.3.1 Die Base Time Unit setzen

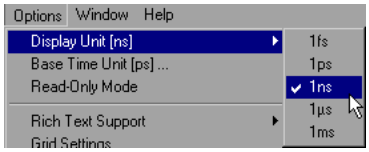
Die Base Time bestimmt die kleinste Zeiteinheit, die alle übrigen Zeiteinheiten im gesamten Programm beeinflusst, da diese auf ein Vielfaches der Base Time skaliert werden. Aus diesem Grunde sollte sie deutlich kleiner sein, als die angezeigte Zeit bzw. die typischen Arbeitszeitintervalle. Arbeitet eine Schaltung (bzgl. Propagation Delay / Taktzeiten) im Nanosekundenbereich, ist es im allgemeinen sinnvoll die Base Time eine Einheit kleiner zu stellen (also Picosekunden). Einerseits ist die Genauigkeit so ausreichend, anderseits wird eine gute Performance bei der internen Bearbeitung gewährleistet.



Die Base Time kann in einem Dialog (siehe linke Abbildung) eingestellt werden, der über den Menüpunkt **„Options→Base Time Unit“** geöffnet wird. In der Rubrik **„How to Change Units“** kann zusätzlich angegeben werden, wie die bereits existierenden Zeitwerte an die neue Base Time Unit angepasst werden sollen. Ist das Timing Diagramm noch leer, hat dies keinen Effekt und kann ignoriert werden.

6.3.2 Display Time Unit

Über den Menüpunkt **„Options→Display Unit“** kann eine Zeiteinheit ausgewählt werden, auf die alle sichtbaren Zeiteinheiten in den Arbeitsfenstern bzw. in verschiedenen Eingabefeldern vordefiniert werden.



6.4 Arbeiten mit WaveFormer Lite

6.4.1 Signale hinzufügen und entfernen

Wird WaveFormer Lite direkt aus dem Design Flow von Libero heraus aufgerufen, sind bereits beim Start alle Signalinformationen der aktuellen HDL-Beschreibung aufgelistet und können mit Hilfe der Maus selektiert werden.
Soll ein markiertes Signal (oder mehrere markierte Signale) entfernt werden, ist dies durch Drücken der **„ENTF-Taste“** möglich. Das Signal verschwindet aus der Liste und wird nicht in die Testbench aufgenommen bzw. bleibt uninitialisiert, wenn es sich um ein Eingangssignal handelt! Über folgendene Wege können Signale hinzugefügt werden:

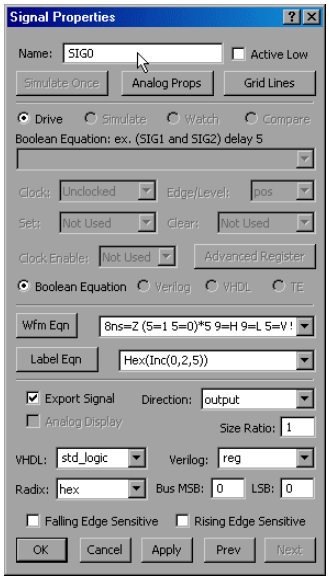
1. Durch Anklicken des **„Extract MUT ports Buttons**  **“** ist es möglich, wieder alle Signale aus dem Design zu laden (Zustand wie nach dem Programmstart). Diese Funktion ist sehr hilfreich, wenn versehentlich ein Signal entfernt wurde!

2. Über den **„Add Signal Button** Add Signal **“** kann ein neues Signal hinzugefügt werden, welches automatisch mit einem Namen (z.B. „SIG0“) versehen und der Liste ergänzt wird. Der neue Name besteht aus einem festen Anteil (=Prefix; z.B: „SIG“) und einer angehängten Nummer. Das verwendete Prefix (der Standardname) kann per Rechtsklick auf den **„Add Signal Button** Add Signal **“** definiert werden, die Nummerierung erfolgt automatisch:

Nachdem das neue Signal hinzugefügt wurde, sollten die Eigenschaften festgelegt werden.

6.4.2 Signaleigenschaften festlegen

Per Doppelklick auf ein selektiertes Signal wird ein neues Dialogfenster (siehe Screenshot) geöffnet, in dem die Signaleigenschaften festgelegt werden können.

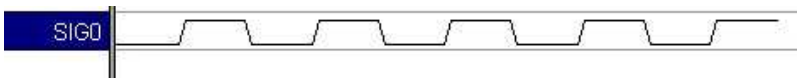


Die wichtigsten Optionen:

- Im oberen Feld **„Name“** kann der Signalname angepasst werden (doppelte Namen sind in der Testbench nicht erlaubt, auch wenn im WaveFormer Lite beim Anlegen keine Fehlermeldung ausgegeben wird!)
- Die Checkbox **„Export Signal“** bestimmt, ob das Signal bei der späteren Exportierung zur Testbench (HDL-Code) einbezogen wird oder nicht.
- Über das Auswahlmenü **„Direction“** kann bestimmt werden, ob das Signal input oder output (oder inout...) repräsentieren soll.
- Im Auswahlfeld **„Radix“** kann die Zahlenbasis (z.B: binär, dezimal, hex oder ASCII) für die Darstellung im Diagramm ausgewählt werden.

Ein interessantes Feature ist die Waveform-Erzeugung über den **„Wfm Eqn Button“**. Im Eingabefeld rechts daneben muss zunächst ein Ausdruck eingegeben werden, der den Signalverlauf beschreibt. Durch Anklicken des Buttons wird dann das durch den Ausdruck beschriebene Verhalten in eine Waveform umgesetzt und an der aktuellen Stelle des selektierten Signals eingefügt! Die Syntax besteht aus einer Liste von Zeit/Wert-Paaren, die durch Leerzeichen getrennt werden (z.B. „8ns=1“ für 8 Nanosekunden logisch 1 erzeugen).

Beispiel für ein Taktverhalten für 50ns mit 10ns als Periode:




```
5ns=0 5ns=1 5ns=0 5ns=1 5ns=0 5ns=1 5ns=0 5ns=1 5ns=0 5ns=1
```

Für die Wertangaben sind nicht nur 0 und 1 möglich, sondern auch: Z, V, H, L, X (std_logic).

Außerdem ist es zulässig, einen geklammerten Ausdruck zu multiplizieren. Das Taktverhalten lässt sich also auch bequemer über eine Periode beschreiben, die 5x wiederholt wird:

```
(5ns=0 5ns=1)*5
```

Dieser kürzere Ausdruck erzeugt genau die gleiche Waveform. Weitere Informationen zur Sytnax können in der Online-Hilfe nachgeschlagen werden.

In ähnlicher Weise können die Signale im Diagramm mit der Funktion **„Label Eqn“** ausdrucksgesteuert beschriftet werden.
Handelt es sich bei dem Signal um ein Clock Signal sind diese beiden Funktionen nicht verfügbar. Stattdessen können mit Hilfe des **„Clock Properties Buttons“** die Takteinstellungen verändert werden.

6.4.3 Ein Taktsignal (Clock) festlegen

Um ein Taktsignal zu erzeugen gibt es zwei Möglichkeiten:

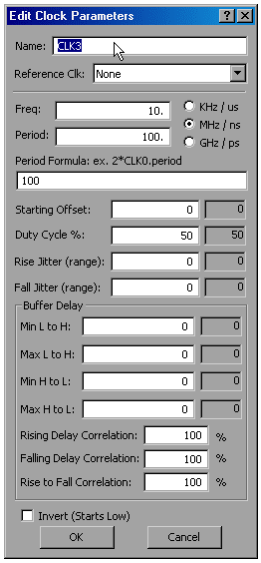
1. Ein vorhandenes Signal kann in ein Taktsignal umgewandelt werden. Dazu muss das gewünschte Signal zunächst per Linksklick selektiert werden. Dann kann per Rechtsklick auf

den Signalnamen ein Kontextmenü geöffnet werden. Über den Menüpunkt **„Signal(s)** <-> **Clock(s)“** wird das Signal in einen Takt umwandelt. Auf die gleiche Weise kann andersherum ein Takt in ein Signal konvertiert werden.

2. Ein neues Taktsignal wird durch den **„Add Clock Button** 

“ hinzugefügt und bekommt (wie ein „normales“ Signal) automatisch einen Namen zugewiesen (z.B. „CLK0“). Auch hier kann das Namen-Prefix per Rechtsklick umgestellt werden. Als nächstes müssen die Takteinstellungen vorgenommen werden.

6.4.4 Die Takteinstellungen festlegen

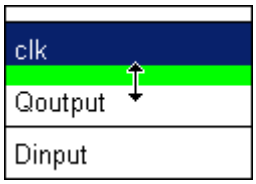


Im **„Edit Clock Parameters Fenster“** können die Takteinstellungen angepasst werden (das Fenster kann über einen Doppelklick auf einen Clock-Signalnamen und dann über den Button **„Clock Properties“** geöffnet werden). Die wichtigsten Optionen sind:

- Im oberen Eingabefeld **„Name“** kann der Clock-Name geändert werden
- In der **Clock Rate Section** kann die Clock-Rate entweder als Frequenz im Feld **„Freq“** angegeben werden oder die Periode wird unter **„Period“** direkt eingetragen. Alternativ kann die Periode auch unter **„Period Formula“** durch einen von anderen Größen abhängigen Ausdruck definiert werden (eine Angabe ist ausreichend; Sollten im Zusammenhang mit der Clock-Frequenz Rundungsfehler auftreten muss die **„Base Time Unit“** angepasst werden → Kap. 6.3.1)

- In der **Clock Properties Section** kann ein **„Offset“** angegeben werden, der den Verlauf um die Offset-Zeit verzögert. Zum Zeitpunkt Null ist ein Takt normalerweise „High“, wenn kein Offset existiert. (Dieses Verhalten kann über den **„Invert (Starts Low Button“** umgedreht werden!)
- **„Duty cycle“** legt prozentual fest, wie lange das Taktsignal während einer Periode auf „High“ bleibt. Die Steilheit der Flanken kann dadurch beeinflusst werden. Eine andere Methode hierfür bietet die **„jitter“** Funktion, sowohl für die positive als auch für die negative Flanke.

6.4.4 Signale kopieren und verschieben



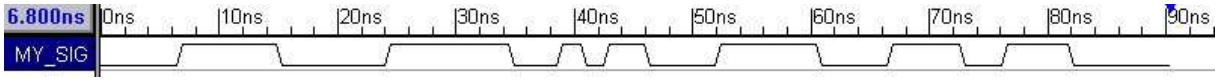
Für die Übersichtlichkeit und Strukturierung lassen sich ein oder mehrere Signale innerhalb der Liste von WaveFormer Lite mit der Maus per Drag & Drop verschieben. Dazu muss das gewünschte Signal oberhalb oder unterhalb des Namens angefasst und entsprechend nach oben oder unten gezogen werden.

Außerdem können Signale mitsamt ihres Verlaufs kopiert, ausgeschnitten und wieder eingefügt werden. Wie in Windows üblich geschieht das mit den Shortcuts STRG+C, STRG+X und STRG+V oder alternativ über das **„Edit-Menü“**.

6.4.5 Signalverlauf zeichnen

Dieses Kapitel beschreibt das Zeichnen und Manipulieren der Waveforms. Die benötigten Funktionen sind über die unten abgebildeten Toolbar Buttons zu erreichen:

Das Zeichnen eines Verlaufs (z.B. wie unten abgebildet) in WaveFormer Lite basiert auf dem Prinzip, dass nur die Pegelwechsel gesetzt werden brauchen, um den gesamten Verlauf eindeutig zu definieren.



Aus diesem Grund arbeitet das Tool mit alternierenden Funktions-Buttons. Wie im oberen Screenshot zu erkennen ist der **„Low-Button“** aktivert (rote Schrift). Über dem **„High-Button“** ist außerdem ein kleines, rotes **„T“**, welches andeutet, dass dies der Zustand ist, der als nächstes folgen wird. Soll nun die oben gezeigte Waveform erstellt werden, so wird zu Beginn durch einfaches Linksklicken in der Signalzeile an der Stelle des ersten Pegelwechsels (≈ bei 6ns) die erste (pos.) Flanke definiert. Die Toolbar Buttons wechseln automatisch: Der **„High-Button“** wird aktiviert (rote Schrift) und der **„Low-Button“** bekommt das kleine, rote **„T“**. Es kann jetzt also direkt weiter nach rechts zur nächsten (neg.) Flanke (≈ bei 15ns) gegangen werden. Ein Linksklick setzt die Flanke und die Toolbar-Buttons „toggeln“ erneut. Dies kann für den weiteren Verlauf wiederholt werden.

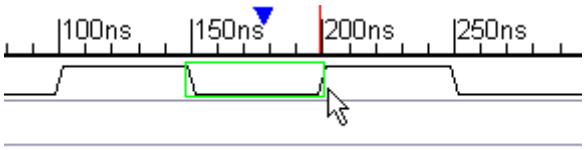
Um den Verlauf im richtigen Maßstab bearbeiten zu können, kann mit Hilfe der rechten vier **„Zoom-Buttons“** in die Waveform hinein- oder hinausgezoomt werden.

Neben den klassischen Pegeln können mit Hilfe der übrigen Buttons auch andere Zustände gezeichnet werden. Die Bedeutung ist wie folgt:

- HIGH, LOW
- TRIstate
- VALid, INValid
- WHI weak high, WLO weak low

Es ist auch möglich, den „Toggle-Zustand“ frei zu wählen. Dazu muss zuerst der gewünschte „Toggle-Zustand“ angeklickt werden (er wird rot / aktiviert), anschließend kann der Partner-Zustand per Linksklick ausgewählt werden. Beim Zeichnen der Pegelwechsel wird jetzt automatisch zwischen den beiden Zuständen gewechselt (Soll kein Wechseln erfolgen muss der gewünschte Zustand zweimal angeklickt werden).
Beim Erstellen der Signalverläufe mit der Maus, werden die per Klick eingezeichneten Pegelwechsel automatisch auf ein Gitter ausgerichtet. Der Gitter-Abstand lässt sich über den Menüpunkt **„Options→Grid Settings→Signal Edge Grid“** ändern.

Die gezeichnete Waveform wird vom Programm automatisch in einzelnen Segmente eingeteilt, jeweils in der Breite bis zum nächsten Pegel- bzw. Zustandswechsel. Das folgende Bild zeigt ein markiertes Segment (grüner Kasten):

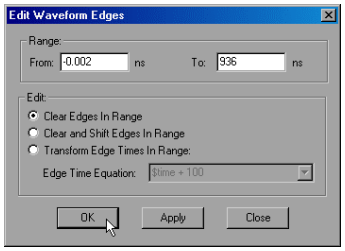


Jedes Segment kann selektiert und editiert werden. Dazu muss es zunächst per Mausklick aktiviert werden (grüner Kasten), dann kann über die Toolbar Buttons ein neuer Zustand ausgewählt werden. Mit Hilfe des Cursors können die Kanten außerdem nach links/rechts verschoben werden.

Ein markiertes Segment kann mit der „**ENTF-Taste“** gelöscht werden.

Innerhalb eines großen Segments kann ein kleines Segment erzeugt werden, indem zuerst das große Segment ausgewählt wird. Anschließend wird in das Segement geklickt und der Cursor mit gedrückter Maustaste nach links oder rechts bewegt. Am Ziel kann die Maustaste losgelassen werden.

Arbeitet ein Design neben den sieben Standard-Zuständen auch mit virtuellen Zuständen (wie z.B: Integer, ASCII oder Aufzählungstypen), können diese über den **„Hex-Button“** eingegeben werden.



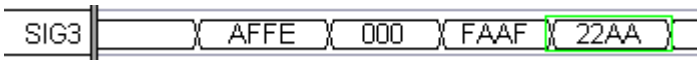
Eine wichtige Option, die leider etwas „versteckt“ ist, befindet sich im Menü unter **„Edit→Edit Waveform Edges“**. Damit ist es unter anderem möglich alle Kanten eines Signals zu ändern.
Ebenfalls sehr nützlich ist die Funktion **„Edit→Right Click Delete Mode“**. Sie ermöglicht schnelles Löschen einzelner Segmenten per Rechtsklick.

6.4.6 Einen BUS hinzufügen

WaveFormer Lite unterstützt drei Arten von Bussen: **„Group Buses“**, **„Virtual Buses“** und **„Simulated Buses“**.

1. Virtueller Bus:

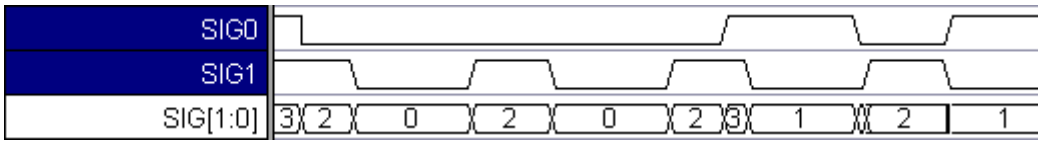
Virtuelle Busse werden durch normale Signale repräsentiert und können beliebige Informationen enthalten. Den Zustand eines virtuellen Busses kann man entweder über den **„HEX-Button“** definieren oder mit dem „**VAL-Button“**. Dazu bietet es sich an, den **„VAL-Button“** zweimal anzuklicken, damit dieser den Zustand beibehält (er ist aktiviert, rot und mit kleinem „T“ versehen). Nachdem die einzelnen Segemente gezeichnet wurden, müssen sie selektiert werden. Nach einem Doppelklick auf das Segment kann dann der Zustand im **„Edit Bus State Dialog“** eingegeben werden. Der untere Screenshot zeigt das Ergebnis:



Virtuelle Busse werden bei der Testbench-Exportierung unterstützt und in entsprechenden VHDL-Code umgesetzt.

2. Group Bus:

Mehrere Signale können zu einem „Group-Bus“ zusammengesetzt werden. Der Bus-Zustand ist dann direkt mit den einzelnen Zuständen der untergeordneten Signalen gekoppelt. Diese müssen somit nicht mehr einzeln gesetzt werden, sondern erhalten ihren Zustand aus dem Wert des Busses. Der folgende Screenshot zeigt einen „Group-Bus“ mit den beiden zugehörigen Signalen „SIG0“ und „SIG1“:



Folgende Schritte verbinden mehrer Signale zu einem Bus:

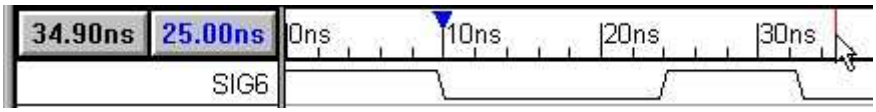
1. Selektieren der einzelnen Signale (hier „SIG0“ und „SIG1“) durch Klicken auf den Signalnamen. Die Reihenfolge bestimmt das jeweilige Bit im Bus. Das Signal, welches zuerst selektiert wurde wird LSB, das letzte wird MSB.
2. Sind alle Signale markiert, wird der **„Add Bus Button“** gedrückt und der **„Choose Bus Type Dialog“** öffnet sich.
3. Der Radio-Button **„Group Bus“** wird ausgewählt und nach Bestätigung mit **„OK“** erscheint der neue Bus im Diagramm.

3. Simulierte Busse:

Ein simulierter Bus wird lediglich simuliert und besitzt keine untergeordneten Signale, die geändert werden dürfen. Er wird durch boolsche Gleichungen beschrieben, kann jedoch nur mit der PRO-Version von WaveFormer benutzt werden.

6.4.7 Time Button und Delta Button

Die beiden im Screenshot abgebildeten Buttons über den Signalnamen zeigen zwei verschiedene Zeiten (34.90ns / 25.00ns):

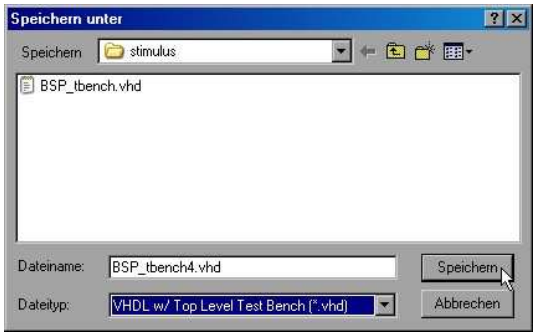


Der linke **„Time Button“** mit der schwarzen Schrift zeigt die aktuelle Position des Mauscursors im Zeitdiagramm an. Der rechte **„Delta Button“** mit der blauen Schrift bildet die Differenz zwischen der Position des Mauscursors und dem blauen **„Delta Mark“** (das kleine Dreieck über der Zeitachse). Jedes Mal wenn die linke Maustaste im Timing-Diagramm losgelassen wird, wird das **„Delta Mark“**-Zeichen automatisch auf diese Position gesetzt. Dadurch wird es sehr einfach, Abstände zwischen verschiedenen Punkten einer Waveform zu messen.

Scrolling:
Die beiden Buttons können außerdem benutzt werden, um schnell in langen Timing-Diagramm zu scrollen. Wird der linke Button angeklickt, kann ein absoluter Zeitwert eingegeben werden, zu dem gesprungen wird. Wird eine Zeitangabe über den rechten Button angegeben, wird relativ vom aktuellen Bildausschnitt um den angegebenen Wert gescrollt.

6.5 Import/Export - Testbench-Erzeugung

Die wichtigste Funktion nach Erstellen alle Signalverläufe ist das Exportieren des Timing-Diagramms zu einer VHDL-Testbench, die anschließend für die Simulation verwendet werden kann. Dies geschieht über den Menüpunkt **„Import/Export → Export Timing Diagrams As“**.



Im abgebildeten Dialogfenster muss ein Name für die neue Testbench angegeben werden. Als Dateityp muss für die Verwendung in Actel Libero unbedingt **„VHDL w/ Top Level Test Bench (*.vhd)“** ausgewählt sein!
Nachdem die neue Testbench erzeugt wurde, wird sie im Report-Fenster von WaveFormer Lite angezeigt und kann überprüft bzw. auf VHDL-Ebene angepasst werden. Anschließend kann zurück in die Libero Oberfläche gewechselt werden, auch hier erscheint die neue Testbench im **„Libero File Manager“**. Die Funktionale Simulation kann beginnen (→ siehe Kap. 6.7)!

6.6 Ausblick auf die PRO-Version

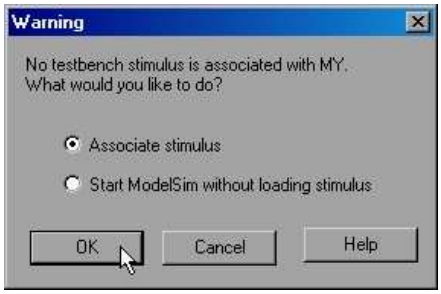
Die folgende Tabelle gibt eine Übersicht über Features und Funktionen der PRO-Version von WaveFormer, die in der Lite Version nicht oder nur eingeschränkt zur Verfügung stehen:

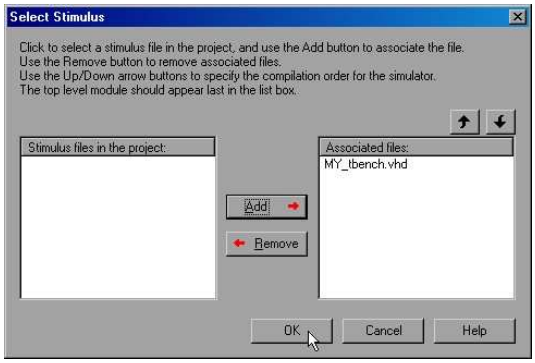
Ein Interaktiver Simulator ermöglicht das Beschreiben von Signalverläufen über boolsche Gleichungen
Es stehen mehr Möglichkeiten zur Beschreibung von Propagation Time bzw. Setup Time etc. zur Verfügung
Es können sämtliche Werte im Timing-Diagramm durch Parameter ausgedrückt werden.
Makros erlauben flexible Substitutionen für Signalnamen und Werte
Vielseitige Dokumentations-Funktionen kommen dazu, mit den die WaveForms übersichtlicher gestaltet und erweitert werden können (z.B. Marker & Text-Funktionen)
Es können diverse Bilder / Grafiken in vielen Formaten generiert werden
Besser Import-Möglichkeiten von WaveForms anderer Hersteller wie z.B. Altera Quartus
Komplette Scripting-Funktionalität in der Sprache Perl
...

6.7 Funktionale Simulation

6.7.1 Aufrufen von ModelSim aus Libero

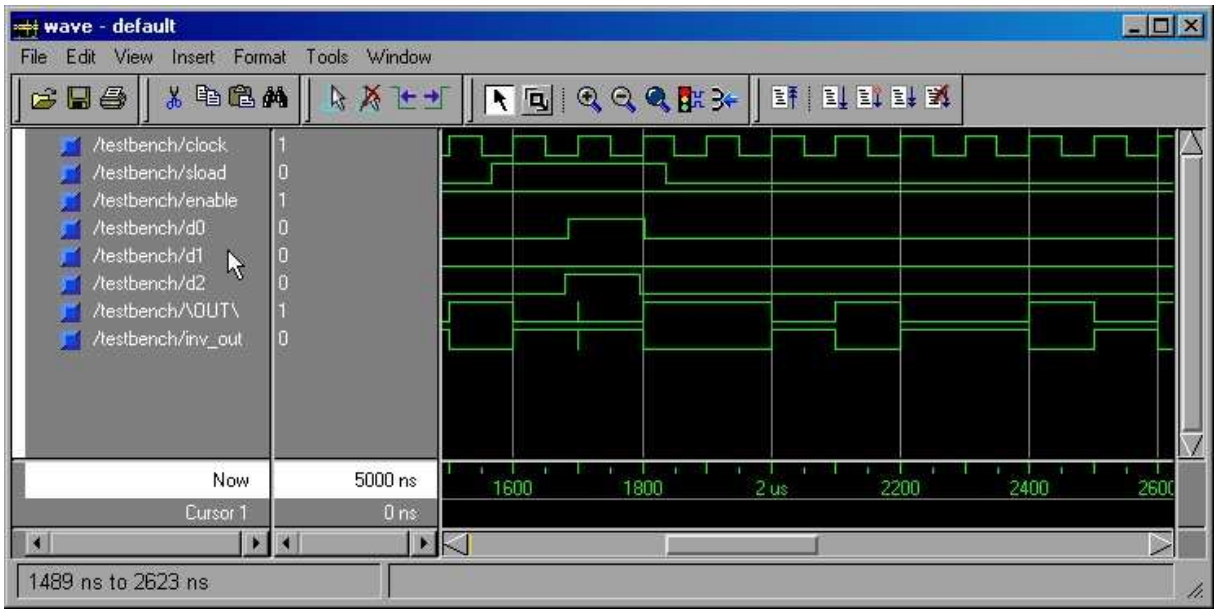
Nachdem die VHDL-Testbench aus WaveFormer Lite exportiert wurde und in Libero zur Verfügung steht, kann die funktionale Simulation durchgeführt werden. Dazu wird im Libero Design Flow auf den **„Simulations-Button“** (links zu sehen) geklickt. Im darauf folgenden Dialogfenster **„Warning“** wird gefragt, ob dem Simulator eine Testbench zugewiesen werden soll. Wir wählen **„Associate stimulus“** und bestätigen die Auswahl mit **„OK“**.





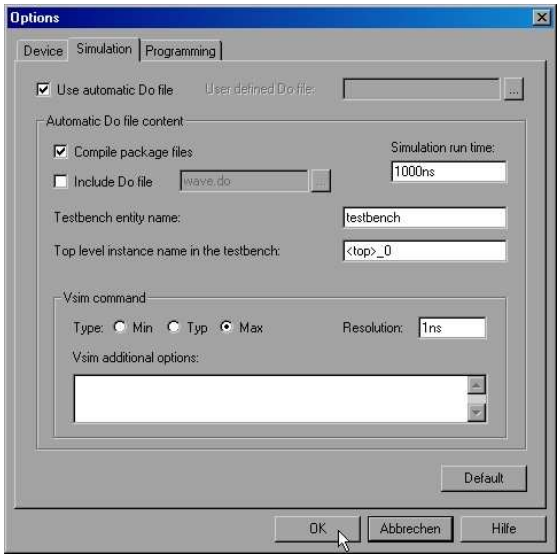
Im Dialogfenster **„Select Stimulus“** muss nun die Testbench ausgewählt werden und über den **„Add Button“** zugewiesen werden. Nach Klick auf **„OK“** wird der Simulator **„ModelSim“** in einem neuen Fenster geöffnet und veranschaulicht in einem Timing-Diagramm das Simulationsergebnis, welches ausgewertet und überprüft werden sollte.

Der untenstehende Screenshot demonstriert das Simulationsergebnis zum Schematic-Beispiel-Design aus Kapitel 5.3:



6.7.2 Simulations - Optionen

Wird im Libero Design Flow mit der rechten Maustaste auf den **„Simulations-Button“** geklickt, so erscheint ein neues Kontextmenü. Über den Menüpunkt **„Options“** öffnet sich ein Dialogfenster in dem wichtige Einstellungen für den Simulationsablauf vorgenommen werden können. So kann im Feld **„Simulation run time“** die Zeitspanne / Simulationsdauer eingetragen werden. Häufig ist der Standard-Wert von 1000ns nicht ausreichend. Wurde die Testbench angepasst, hat sich evtl. auch der Name der Haupt-Entity verändert und kann im Eingabefeld **„Testbench entity name“** korrigiert werden.



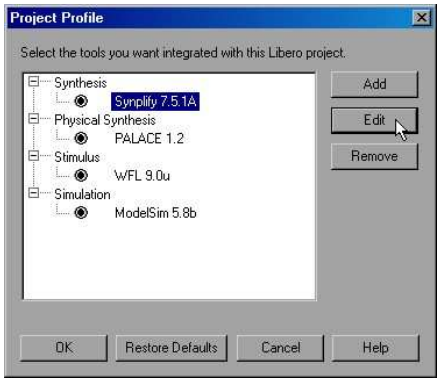
7. Synthese

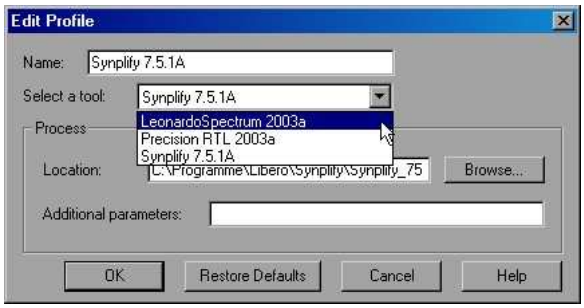
7.1 Einleitung

Ist ein HDL-Entwurf aufgestellt und die Funktionalität mit Hilfe einer Testbench erfolgreich simuliert wurden, kann das Design synthetisiert werden. Die Oberfläche von Actel Libero unterstützt dazu folgende drei Synthese-Tools anderer Hersteller, die an den Libero Design Flow angepasst sind:

- **Synplify** von Synplicity
- **LeonardoSpectrum** von Mentor Graphics
- **Precision RTL** von Mentor Graphics

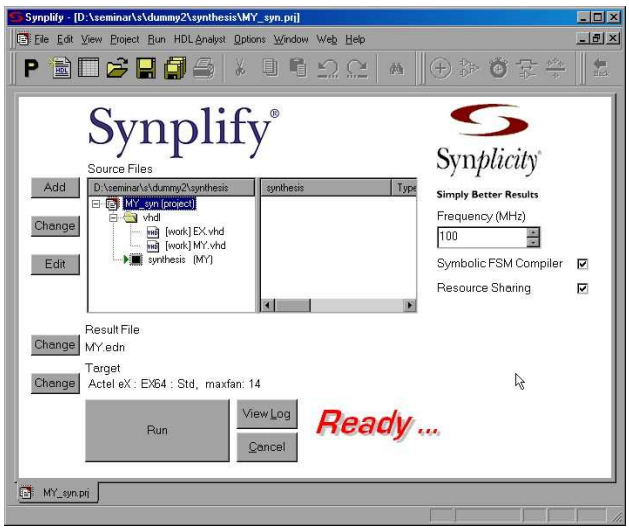
Mit Hilfe eines Rechtsklick auf den **„Synthesis-Button“** im Libero Desgin Flow und der Auswahl des Menüpunktes **„Profile“** kann im anschließenden Dialogfenster das gewünschte Synthese-Tool ausgewählt werden.





Da im Standard Design-Package von Actel Libero lediglich Synplify AE von Synplicity integriert ist, bezieht sich dieses Kapitel auf dieses Tool. Mit Linksklick auf den **„Synthesis-Button“** wird das Programm aus Libero heraus in einem neuem Fenster geöffnet.

7.2 Die Oberfläche von Synplify



Die linke Abbildung zeigt die Oberfläche von Synplify. Da das Programm aus Libero heraus aufgerufen wurde, sind bereits alle Dateien des Projekts ausgewählt und in einer Baumstruktur unter **„Source Files“** zu sehen. Auch die Zielplattform (FPGA) wurde bereits aus Libero übernommen und ist unter **„Target“** wiederzufinden (im Beispiel: „Actel eX“).
Das Syntheseergebnis ist eine Netzliste im EDIF-Format. Auch hier ist der Name voreingestellt und unter der Beschreibung **„Result File“** zu finden.

7.3 Der Synthesevorgang

Der eigentliche Synthesvorgang wird über den **„Run-Button“** eingeleitet. Sämtliche Statusinformation sowie evtl. auftretende Warnung oder Fehler werden in einer Log-Datei festgehalten und können per Klick angezeigt werden. In der Mitte des Fensters erscheinen die generierten Dateien (Gate / RTL Netzlistem, Log-Dateien...).
Ist die Synthese abgeschlossen, kann das Ergebnis betrachtet und analysiert werden. Da es sich jedoch um eine eingeschränkte „Lite-Version“ handelt, sind die Möglichkeiten sehr begrenzt und werden hier nicht weiter erläutert.

Über die Toolbar Buttons **„RTL View“** und **„Technology View“** kann eine Übersicht über das Synthese-Ergebnis betrachtet werden. In einer Baumstruktur sind verschiedene Elemente wie verwendete Ports, Netze, Instanzen jeweils mit Fan-Out untergebracht. Die grafische Zuordnung der Elemente im Schaltplan kann in der eingeschränkten Version leider nicht geöffnet werden.

8. Anmerkungen zu den Tools

8.1 Fehler in exportierter Testbench beheben

Es kann manchmal vorkommen das die Export-Testbench-Funktion von WaveFormer Lite VHDL-Code erzeugt, der folgenden Fehler im Simulator ModelSim hervorruft.

```
...
# ** Error: C:/bsp/stimulus/BSP_tbench.vhd(146): near "1.": Number must terminate with a digit.
...
```

Der Fehler entsteht, weil WaveFormer Lite hin und wieder abschließende Nullen hinter dem Komma weglässt, wie z.B. in der unteren Zeile:

```
wait for 1.e-003 ns;
```

Korrigiert man die Zeile wie folgt, funktioniert die Simulation wieder:

```
wait for 1.0e-003 ns;
```

8.2 Fehlerquelle: Testbench-Export-Format

Sollte ModelSim im Log-Fenster einmal mit folgender Fehlermeldung abbrechen:

```
# ** Error: (vsim-3170) Could not find '../simulation/postsynth.testbench'.
# Error loading design
# Error: Error loading design
```

Dann mit Rechtsklick auf das ModelSim Icon im Desgin Flow von Libero klicken und **„Options“** wählen und den Testbench Entity Name überprüfen!

8.3 Fehlerquelle: ModelSim zeigt keine Output-Signale an

Wenn ModelSim die Output-Signale nicht anzeigt, wurde vermutlich die Testbench falsch exportiert, es wurde als Dateiformat beim Export fälschlicherweise **„VHDL (*.vhd)“** und nicht **„VHDL w/ Top Level Testbench (*.vhd)“** angegeben!

8.4 Fehlerquelle: ViewDraw start nicht aus Libero heraus

Je nach Version, Lizenz und Installation kann es vorkommen, dass ViewDraw AE nicht aus Libero heraus gestartet werden kann. Klickt man im Design Flow auf den **„ViewDraw-Button“** und gibt ein neuen Schematic-Namen ein, erscheint lediglich im Log-Fenster eine Fehlermeldung: **„Unable to start ViewDraw AE...“**. Dieses Problem kann folgendermaßen behoben werden:

1. **Schritt:** Das aktuelle Projekt speichern und Libero beenden
2. **Schritt:** In das Libero-Installationsverzeichnis wechseln und dort den Unterordner ViewDraw auswählen (z.B. „C:\Programme\Libero\ViewDraw“)
3. **Schritt:** Im Ordner ViewDraw befindet sich das Programm **„configurator.exe“**, welches nun aufgerufen werden kann. Das Programm setzt die internen Windows Pfad-Variablen neu und beendet sich anschließend automatisch.
4. **Schritt:** Jetzt kann Libero erneut gestartet werden. ViewDraw lässt sich wie gewünscht aus dem Design Flow heraus aufrufen.