

VHDL Synthese mit WebPACK

B **U** **L** **M** **E**  Höhere Technische
Bundes-Lehr- und
Versuchsanstalt
BULME Graz – Götting

V1.0

F. Wolf

Graz, Jänner 2002

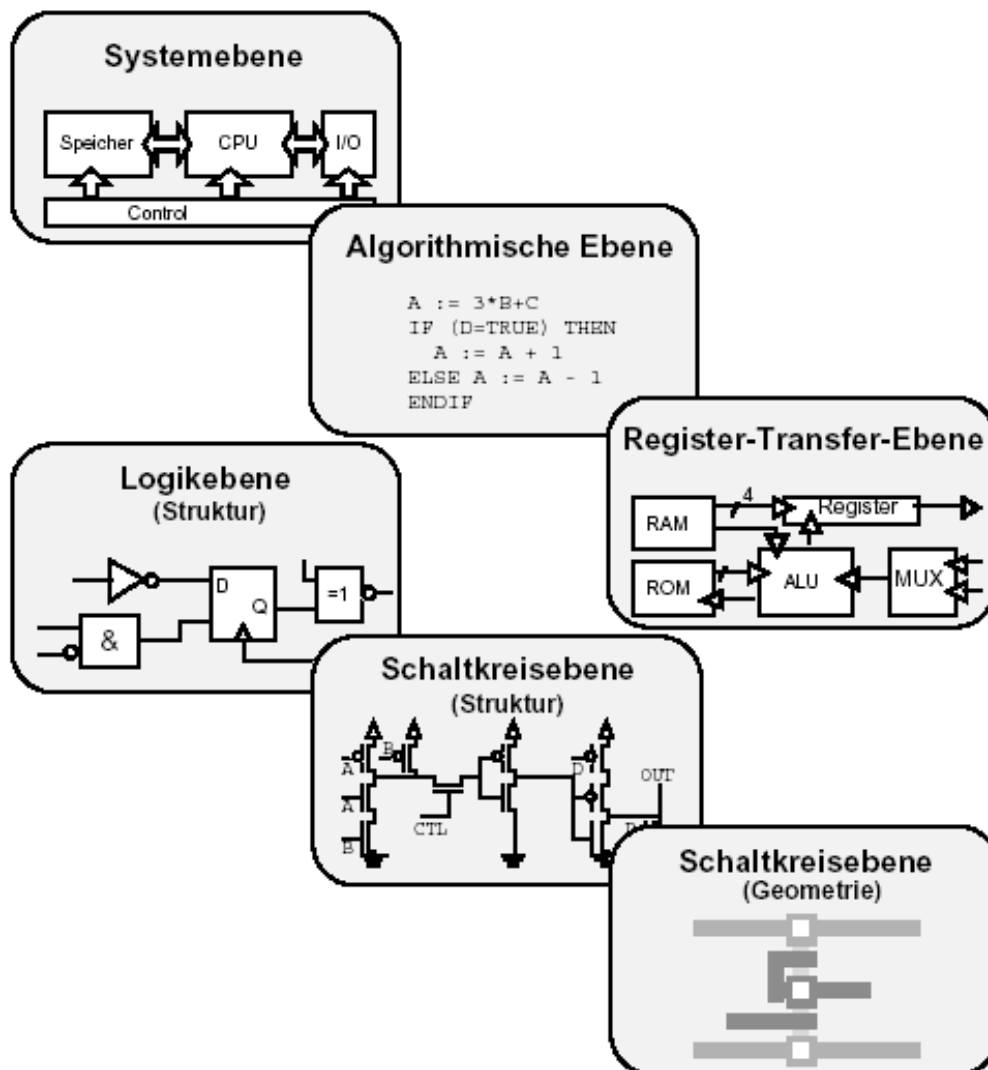
Inhaltsverzeichnis

1	<i>Einleitung</i>	3
2	<i>Abstraktionsebenen</i>	4
2.1	Systemebene	5
2.1.1	Abstraktionsebenen	5
2.2	Beschreibungs-Domänen	6
2.3	Was ist RTL Synthese ?	7
2.3.1	Place&Route	8
2.3.2	Boundary Scan	8
2.4	Darstellung der Entwicklungsschritte mit WebPACK	8
3	<i>Hochstarten von WebPACK 3.2 WP 3.x</i>	9
3.1	Anlegen eines neuen Projektes	9
3.2	Was ist ein Fitter ?	11
3.3	Simulation mit ModelSim	11
3.4	Programmierung der Hardware	12

VHDL Synthese mit WebPACK

1 Einleitung

Unter Synthese versteht man allgemein den Übergang von der formalen Beschreibung eines Verhaltens zu einer dieses Verhalten realisierenden Struktur. Abhängig vom Abstraktionsgrad der Beschreibung, die als Eingabe für die Synthese dient, spricht man von Logiksynthese, Register-Transfer-Synthese, Algorithmischer Synthese und Systemsynthese. Während die Systemsynthese gegenwärtig noch vom Entwickler von Hand durchzuführen ist, stehen für die übrigen Synthesearten bereits Programme zur Verfügung. Die meisten beschränken sich dabei jedoch auf die Register-Transfer-Ebene oder Logikebene.



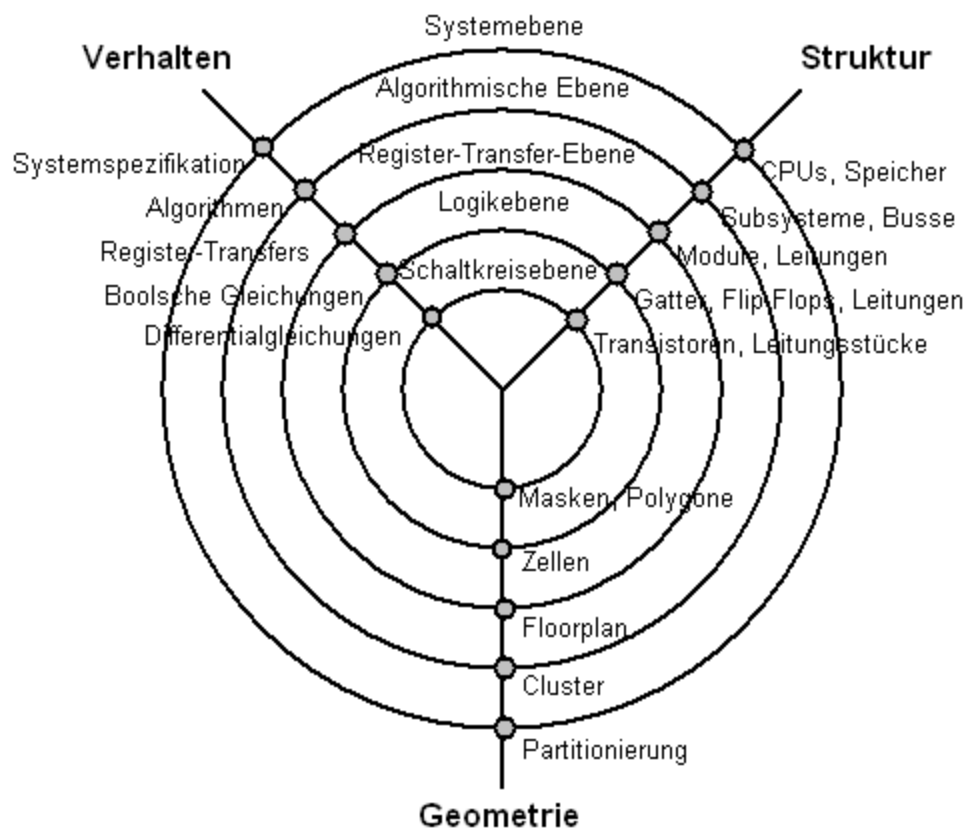
2 Abstraktionsebenen

Die hierarchische Strukturierung und dabei die Abstraktionsebenen bei der Auswahl der Design-Werkzeuge und dann auch beim Entwurf ein sehr wichtiger Aspekt. Bei der VLSI-Schaltungsentwurf werden die Abstraktionsebenen oft mit drei unterschiedlichen Domänen in sog. Y-Diagramm dargestellt.

Diese Domänen bezeichnet man als:

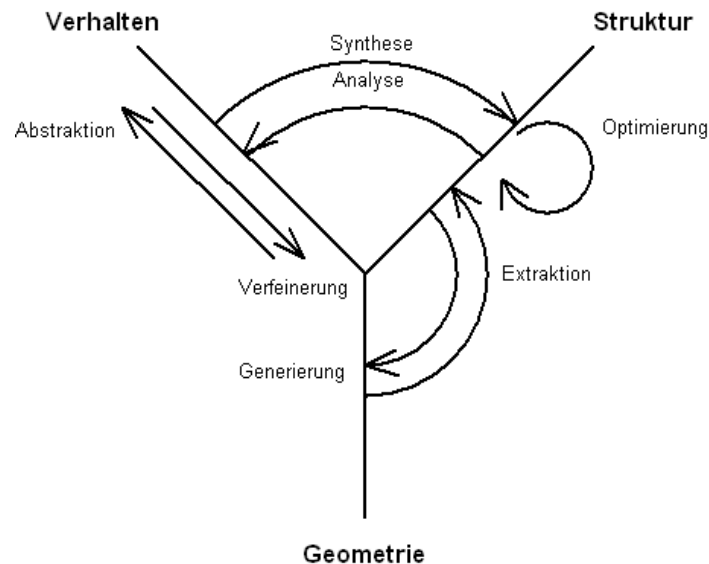
- Verhalten (Behavioral Domain)
- Struktur (Structural Domain)
- Geometrie (Geometrical Domain)

Zur Darstellung der Abstraktionsebenen und der Beschreibungs-Domänen wird im Bereich des VLSI-Entwurfs das Y-Diagramm (siehe nächste Seite) von Gajski und Kuhn verwendet.



Die verschiedenen Abstraktionsebenen beschreiben die unterschiedlichen Tiefen der Konkretisierung unabhängig von den Sichtweisen. Innerhalb eines Top-Down-Entwurfs, der von der Aufgabenbeschreibung ausgehend eine tiefere Beschreibung und Definition des zu bearbeitenden Projekts durchläuft, dies mit Transformationen ggf. zwischen den jeweiligen Sichtweisen, werden die einzelnen Ebenen manuell oder automatisch durchlaufen.

Die einzelnen Ebenen sowie die drei Sichtweisen werden im Regelfall keineswegs komplett durchlaufen, da dies nicht unbedingt notwendig ist. Andererseits ist es auch nicht möglich, aus einer einzigen Beschreibung auf hoher Abstraktionsebene auf automatische Weise eine elektronische Schaltung zu erzeugen. In der untenstehende Abbildung zeigt einen möglichen Weg sowie die Übergänge zwischen den einzelnen Ästen.



2.1 Systemebene

Beschreibt die grundsätzlichen Eigenschaften eines elektronischen Systems unter Verwendung typischer Blöcke. Als solche Blöcke dienen beispielsweise Interfaceeinheiten, Prozessoren usw. Diese Blöcke werden durch ihre Funktionalität (im Beispiel einer CPU durch ihren Typ oder durch ihren Befehlssatz), ihre Protokolle (z.B. bei Interfaces allgemeiner Art) oder durch stochastische Prozesse beschrieben. Die Beschreibungen auf Systemebene werden häufig in natürlicher Sprache oder durch Blockschaltbilder gegeben.

2.1.1 Abstraktionsebenen

Algorithmenebene (Systemverhalten, Verhalten von Ein-/Ausgänge...) : das System bzw. seine Blöcke werden durch nebenläufige (d.h. parallel zueinander ausführbare) Algorithmen beschrieben. Typische Beschreibungselemente hierfür sind Funktionen, Prozesse und Kontrollstrukturen. Insbesondere die Verhaltenssicht wird dem Begriff der Algorithmen gerecht, da hier die Beschreibungen durch algorithmische Darstellungen mit Variablen und darauf wirkenden Operatoren dem (softwaregeprägten) Begriff sehr nahe kommen. Die Algorithmen der anderen Sichtweisen entsprechen den Darstellungen funktionaler Blöcke mit Kommunikation durch Signale zwischen diesen Blöcken. Zeitliche Zuordnungen durch Takt- und Resetsignale fehlen an dieser Stelle noch ebenso wie konkrete Hardwarebezogenheit.

Register-Transfer-Ebene (Logikmodule wie Register, Addierer, Multiplexer..): stellt eine der wichtigsten Ebenen innerhalb der Abstraktionen dar, da Synthesetools ab dieser Ebene arbeiten können, was bedeutet, dass bei korrekter Definition der RTL eine automatische Synthese des kompletten Systems (oder der Teilblöcke) erfolgen kann.

Logikebene (*Schaltnetze und Schaltwerke ...*): die Beschreibungen der Logikebene stellen die elektronische Schaltung durch logische Verknüpfungen auf Basis der Booleschen Algebra und deren zeitliche Verzögerungen z.B. durch Laufzeiten innerhalb der Gatter dar. Der Verlauf der Ausgangssignale wird durch die der Eingangssignale und der logischen Verknüpfung bestimmt, wobei den Signalen nur diskrete Werte wie 'low', 'high' oder 'undefined' zugeordnet werden.

Um eine vollständige Simulation auch im wertediskreten Bereich zu ermöglichen, können dabei wesentlich mehr Werte als einer zweiwertigen Logik (eventuell mit der Erweiterung eines 'dritten Zustands', der einem hohen Ausgangswiderstand äquivalent ist) entsprechend zugeordnet werden, solange diese einer endlichen Menge entstammen und auf die Werte der Booleschen Algebra mit gegebenen Randbedingungen abbildbar sind.

Schaltkreisebene (*Transistoren, Widerstände, Kapazitäten...*): die Form der Netzliste, die Elemente der elektronischen Schaltung verbindet, bleibt innerhalb der strukturalen Sicht beim Übergang von der Logikebene in die Schaltkreisebene erhalten. Während jedoch in der vorigen Abstraktionsebene die Elemente aus Gattern mit entsprechender Logikfunktion bestanden, werden nunmehr Transistoren und adäquate Bestandteile wie Widerstände und Kapazitäten der elektronischen Schaltung miteinander verbunden. Die geometrische Sicht verfeinert dies in Richtung geometrischen Abschnitte und Dotierungen auf dem Chip. Damit gehört die Schaltkreisebene nicht mehr zum Entwurfsablauf bei der Realisierung eines FPGA und soll daher nicht näher betrachtet werden. Lediglich sind hier einige elektrischen Parameter zur Bedeutung, wie z.B. Signalpegel, Fan-In und Fan-Out usw.

Wie Sie sehen, hat jede Abstraktionsebene eine bestimmte Funktion. Die oberen Entwurfsebenen dienen eher zur Darstellung einer Gesamtübersicht. Die unteren Ebenen werden für die detaillierte Darstellung benutzt. So eignen sich System- und Algorithmenebenen neben der Spezifikation auch für die Dokumentation. Die Register-Transfer-Ebene bietet die Grundlagen für eine funktionelle Simulation und die Logikebene für eine Timing-Simulation.

2.2 Beschreibungs-Domänen

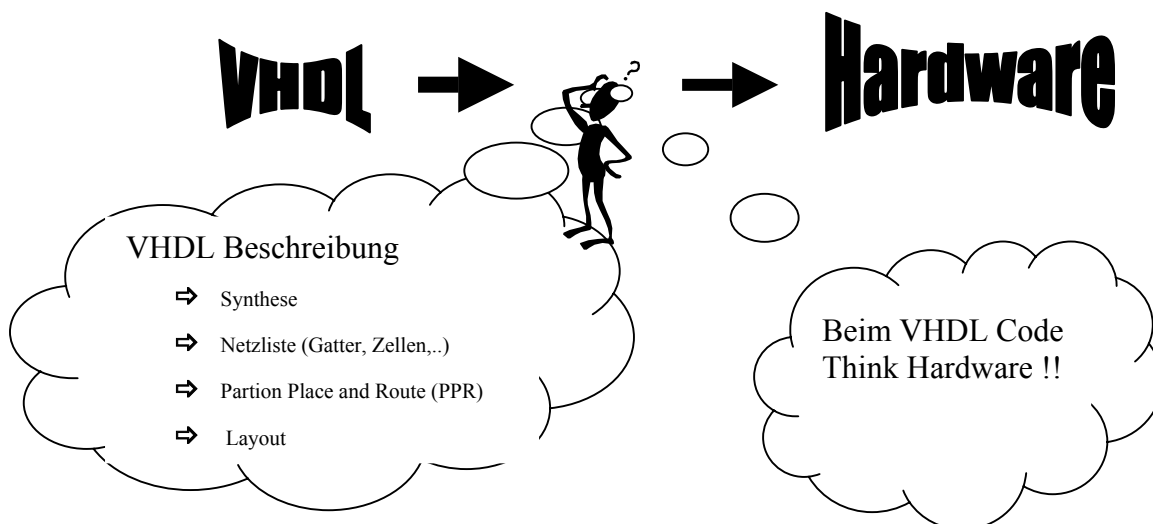
Verhaltensdomäne: hier steht die Funktion der Schaltung im Vordergrund. Es wird beschrieben, was die Schaltung tun soll und nicht, wie sie aufgebaut ist. Wie schon erwähnt, auf Systemebene dient die Verhaltensdomäne zur Spezifikation einer Schaltung, also der Anforderungen an Funktion und Schnittstellen. Als Beschreibungsmittel können in den oberen drei Ebenen (Systemebene, Algorithmenebene, Register-Transfer-Ebene) Hardwarebeschreibungssprachen wie VHDL oder Verilog HDL eingesetzt werden.

Strukturdomäne: in dieser Domäne wird der Aufbau des Entwurfsobjekt als Zusammenschaltung einfacher Komponenten dargestellt. Auf Logikebene sind es Gatter und Flipflops. In der RT-Ebene werden Logikmodule wie Register, Addierer, Multiplexer etc. In den höheren Abstraktionsebenen werden Funktionsblöcke verschaltet, die jeweils für ein Teilsystem oder komplexe Funktionseinheiten wie Prozessoren oder Speicher stehen. Auf diese Art und Weise haben wir hier eine hierarchische Strukturierung des Schaltungsentwurfes.

Geometriedomäne: hier werden die Layout-Eigenschaften festgehalten. Da diese Eigenschaften bei der FPGA durch entsprechende Konfigurierung festgelegt werden, wird diese Domäne eher als eine Konfigurationsdomäne angesehen.

2.3 Was ist RTL Synthese ?

RTL bedeutet „Register Transfer Ebene“ und ist die computergestützte automatische Umsetzung eines mit VHDL beschriebenen Modells eines Design in eine optimierten technologiespezifische Beschreibung auf Gatterebene. Dies bedeutet, Register-Transfer-Synthese implementiert die Daten- und Kontrollwege in Form von Logikfunktionen, Registern, Flipflops und Verbindungsleitungen. Alle Stufen vor der Synthese erfordern einen „kreativen Prozess“, RTL läst nur ganz spezifische auf die Zielhardware zugeschnittene Funktionen zu. Synthesesoftware ist die zentrale Verbindung beim Top-Down-Entwurf zwischen VHDL und Zieltechnologie. Das Synthesetool gibt vor, welche Konstrukte mit welchen Einschränkungen für das Design verwendet werden dürfen. Mittels Constrains (Rahmenbedingungen) kann der Entwickler das Ergebnis der Synthese beeinflussen (z.B. Fläche und Timing). Synthese erzeugt eine funktionelle Abbildung des VHDL-Modells.



Vorteile der Synthese:

- Kurze Entwicklungszeiten
- Beherrschung der Komplexität
- Automatische Optimierung
- Technologieunabhängige Entwurfseingabe, dadurch mehr Portabilität.

Nachteile der Synthese:

- Ergebnis abhängig vom Beschreibungsstil
- Effizienz abhängig von der Synthese-Werkzeug
- Kontrolle über die resultierende Hardware verringert
- Manuelle FPGA-Entwurf (oft) schneller und kleiner

2.3.1 Place&Route

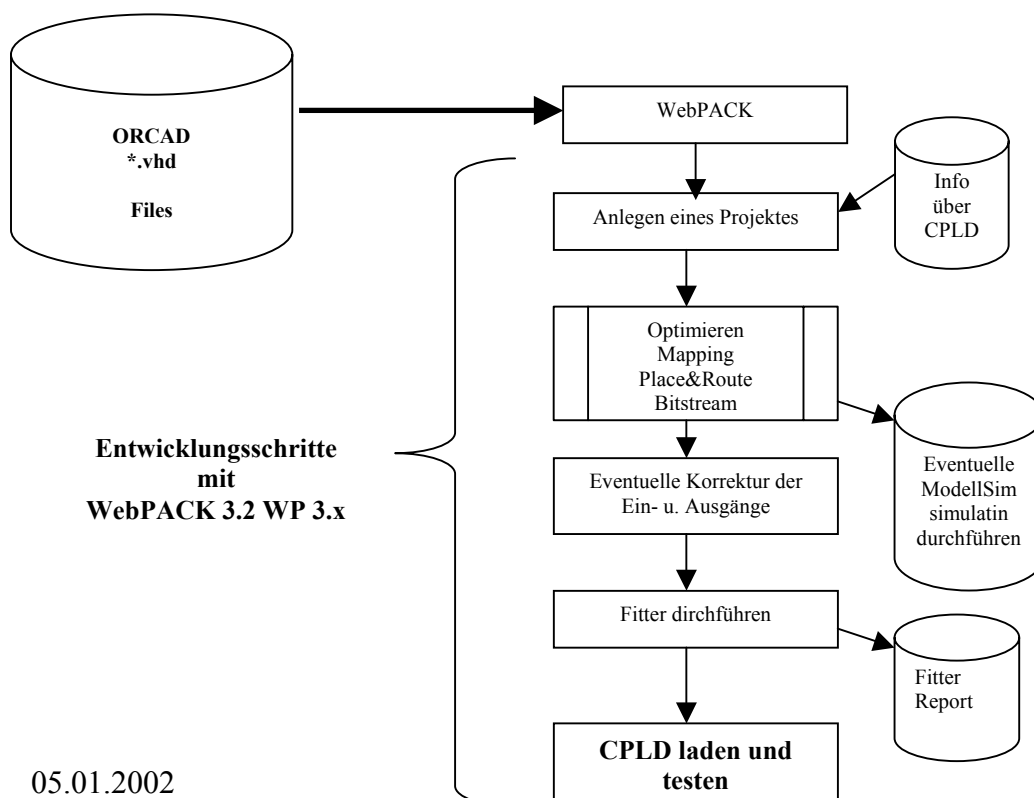
In der Designphase Synthese werden die einzelnen Logikfunktionen auf die FPGA-Ressourcen abgebildet. Ähnlich wie bei einem Leiterkartendesign fehlt noch die Entscheidung des Entwicklers darüber, wo die einzelnen Funktionen zu platzieren sind und wie sie miteinander verdrahtet werden sollen. Diese wird mit dem Place&Route Werkzeug meistens automatisch nach Vorgaben des Entwicklers realisiert. Diese Vorgaben können bereits in der Definitionsphase als Annotation eingegeben werden.

Auch nachträgliche Änderungen zur weiteren Optimierung eines automatisch durchgeführten Place&Route sind möglich. In diesem Fall arbeitet man mit autointeraktiven Werkzeugen, wie z.B. Floorplanning beim ASIC-Design.

2.3.2 Boundary Scan

Der JTAG Boundary Scan Test (JTAG-BST) ist eine wichtige Standardmethode zur Überprüfung von Bauteilen und insbesondere deren Verdrahtung. Die zu testenden Knotenpunkte einer Schaltung, die normalerweise über die Pinanschlüsse nicht zugänglich sind, werden an eine Abtast-Zelle, genannt Scan-Zelle, gelegt. Im Prinzip handelt es sich um ein im FPGA integriertes Schieberegister, das serielle Einstellungen und Beobachtung aller Bauelementanschlüsse ermöglicht. Der Boundary Scan Test ("Knotenpunkt-Abtastung") wurde von der Joint Test Action Group (JTAG), einem Zusammenschluss von Elektronikfirmen, entwickelt, definiert und als IEEE1149.1 Standard genormt. Die Schnittstelle für die Datenübertragung bei dem JTAG-Test, heißt TAP (Test Access Port). Mit nur 4 Anschlüssen kann diese Schnittstelle mit relativ niedrigem Aufwand auf einem FPGA realisiert werden.

2.4 Darstellung der Entwicklungsschritte mit WebPACK

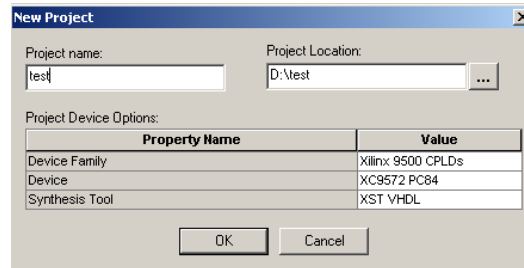


3 Hochstarten von WebPACK 3.2 WP 3.x

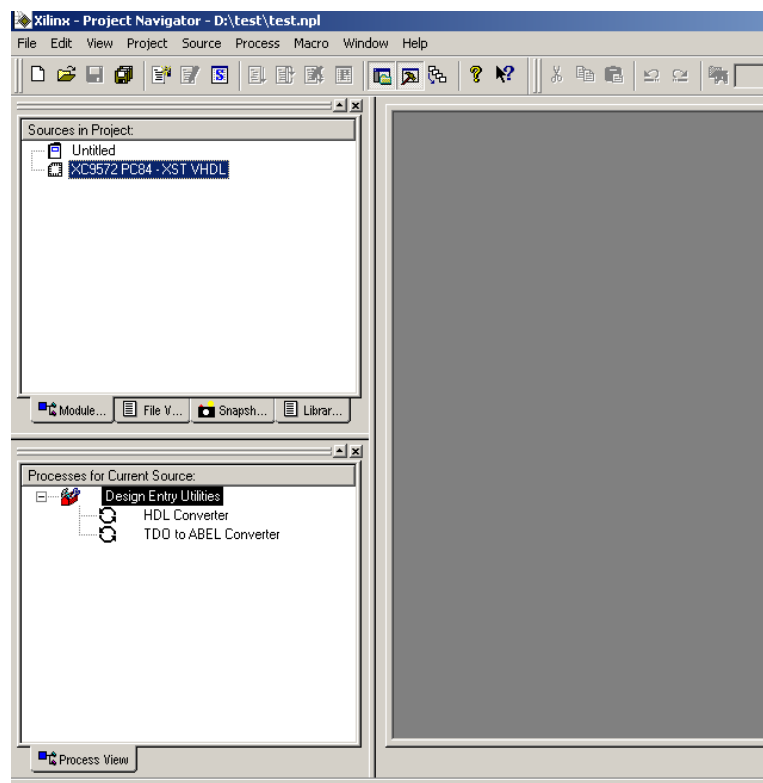
3.1 Anlegen eines neuen Projektes

WebPACK wird über das Windowsstartmenü gestartet, danach wird ein neues Projekt angelegt. Hier wird auch schon die Bausteinserie von Xilinx ausgewählt.

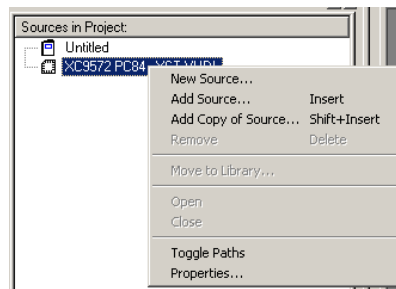
FILE ⇒ New Project



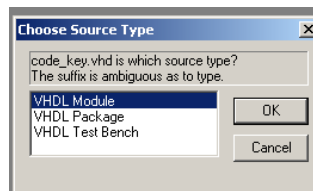
jetzt erscheint diese Arbeitsfläche



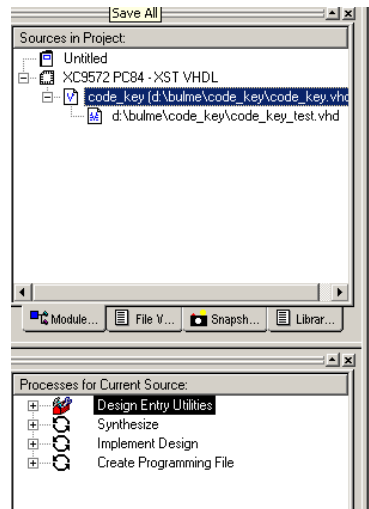
☞ Rechte Maustaste auf XC9572 PC84-XST VHDL ⇒ Add Source.



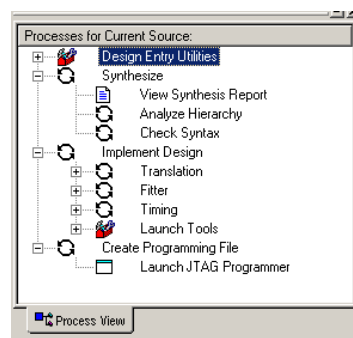
Nun die Files die zuvor in ORCAD simuliert wurden dazufügen. Hier wird noch abgefragt, ob es sich um VHDL Module (VHDL Modell) und VHDL Testbench angeben



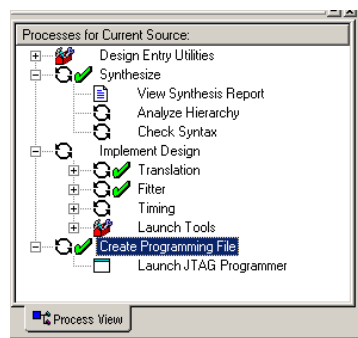
Jetzt wählt man das Modell aus, und das Fenster „Processes for Current Source“ verändert sich.



Man geht nun in die Untermenüs rein



Der Einfachste Weg ist nun ein Doppelklick auf „Create Programming File“ und die Synthese wird nun Schritt für Schritt durchgeführt. Ist alles erfolgreich verlaufen, erscheinen „grüne“ Häkchen. Die Synthese ist nun abgeschlossen.



Jetzt könnte man sich noch den Fitter-Report ansehen. Dieser enthält hier auch ein Portmapping.

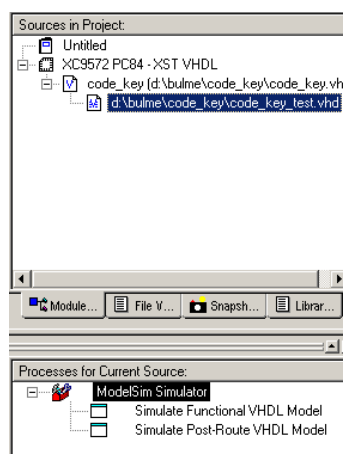
3.2 Was ist ein Fitter ?

Dies ist eine Software, die das entworfene Design auf die Zieltechnologie optimiert. Die Hardwarebeschreibung (z.B.: Wahrheitsgleichungen oder Boolesche Gleichungen) werden durch den Fitter in Programmierdaten, hier für ein CPLD (Complex Programmable Logic Device) umgesetzt. Vom Fitter wird dann die Binärdatei erzeugt, die ins Bauteil geladen wird (z.B.: JEDEC-Datei). Der Fitter ist also auf die jeweilige Hardware abgestimmt. Er wird teilweise auch als Compiler bezeichnet.

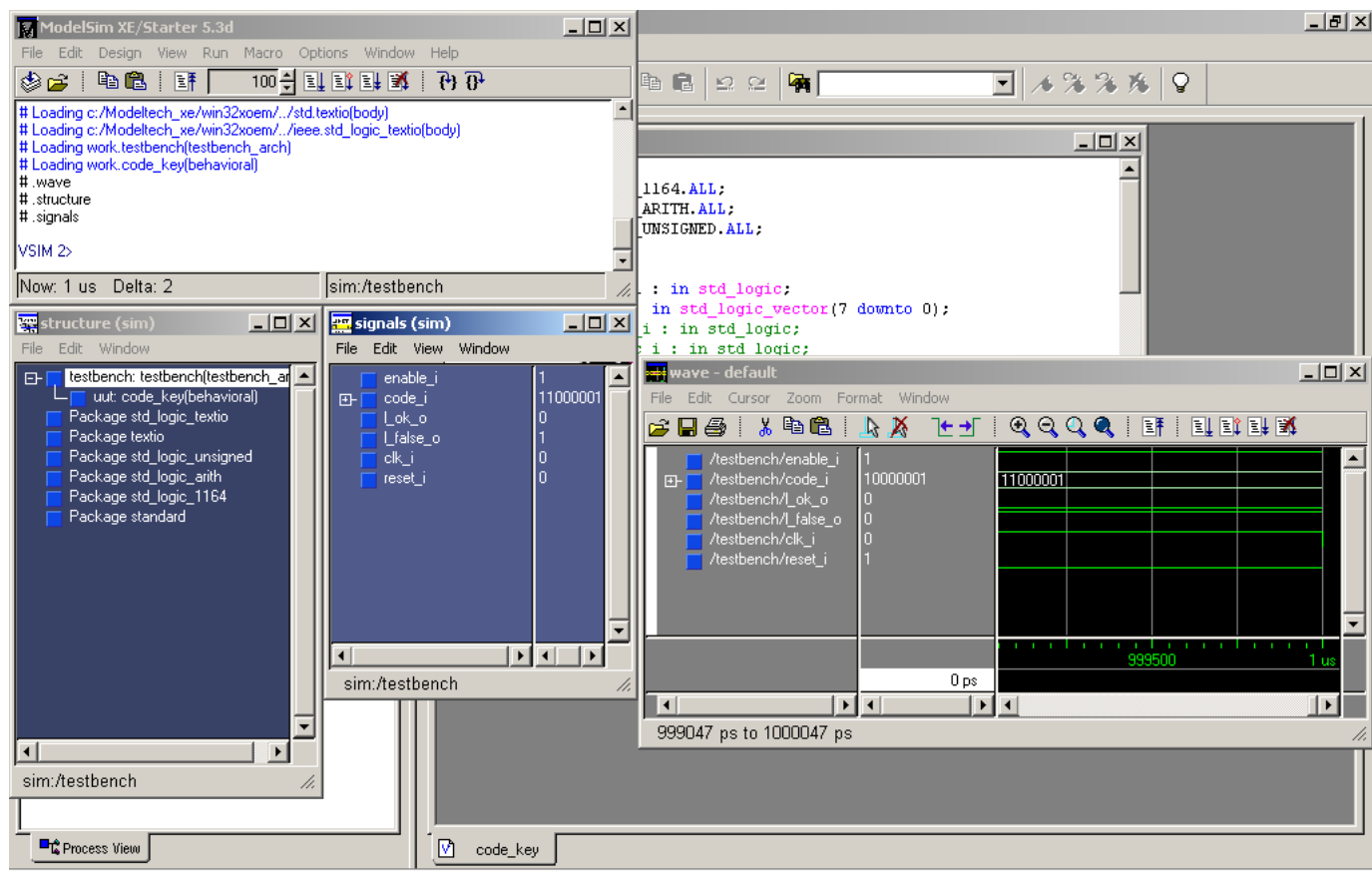
3.3 Simulation mit ModelSim

Setzt die Installation und die Organisation des licens-Files von Xilinx über das Internet voraus.

Man wählt die Testbench aus, und klickt danach im „Processes for Current Source“ Fenster auf „Simulate Functional VHDL Model“

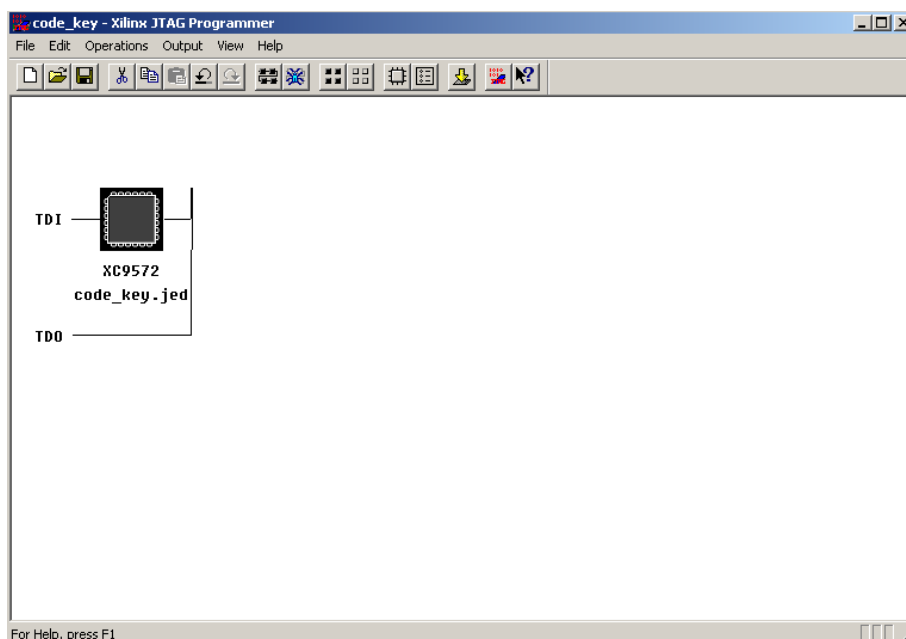


Es startet nun der Simulator ModelSim.



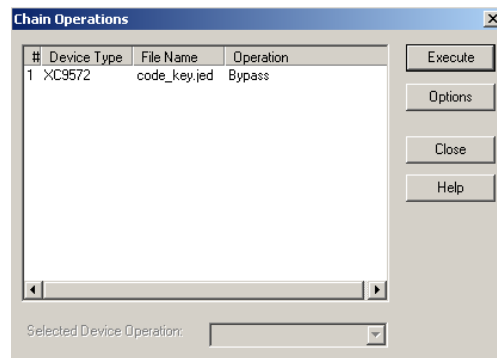
3.4 Programmierung der Hardware

Durch Doppelklick auf „Launch JTAG Programmer“ erscheint nun eine neue Oberfläche.

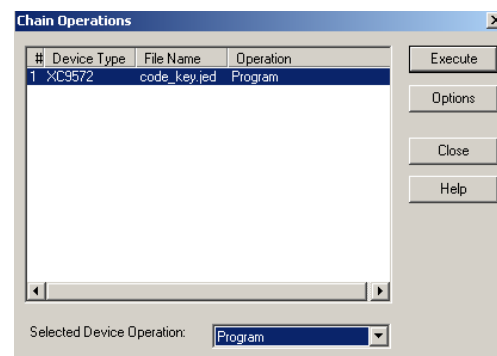


Jetzt werden der Programmer spezifiziert:

Operation ⇒ Chain Operation (oder Gelber Pfeil drücken)



Hier wird nun die Zeile angeklickt. Und bei „Select Device Operation“ Programm auswählen



Jetzt kann die Programmierung der Hardware mit „Execute“ durchgeführt werden. War alles erfolgreich erscheint folgender Kommentar. Jetzt kann der Funktionstest an der Hardware durchgeführt werden.

