Entwurfsmethodik bei FPGA's

Proseminar 2003

Michael Rupp

Inhalt

1. Einleitung

Der allgemeine Entwurf von digitalen integrierten Schaltungen hat sich im Lauf der Zeit wesentlich verändert. Wie in Abbildung 1 ersichtlich, wurden immer andere Verfahren notwendig, um der Komplexität und der gestiegenen Gatteranzahl Herr zu werden, und somit eine kürzere Entwicklungszeit zu erreichen. Konnte man früher noch komplette Schaltkreise in der Transistorebene entwerfen, so ist das heute wegen der gestiegenen Komplexität der Schaltungen nicht mehr möglich. Durch die gestiegene Anzahl von Gattern stieg auch die Wahrscheinlichkeit von Fehlern im Entwurf. Analog hierzu sank die Effizienz bei einem manuellen Entwurf. Bei einer geringen Anzahl von Gattern konnte man sich auch noch den Aufbau eines Prototypen zu Testzwecken, erlauben, was bei neueren Bauelementen zu teuer und unwirtschaftlich wäre.

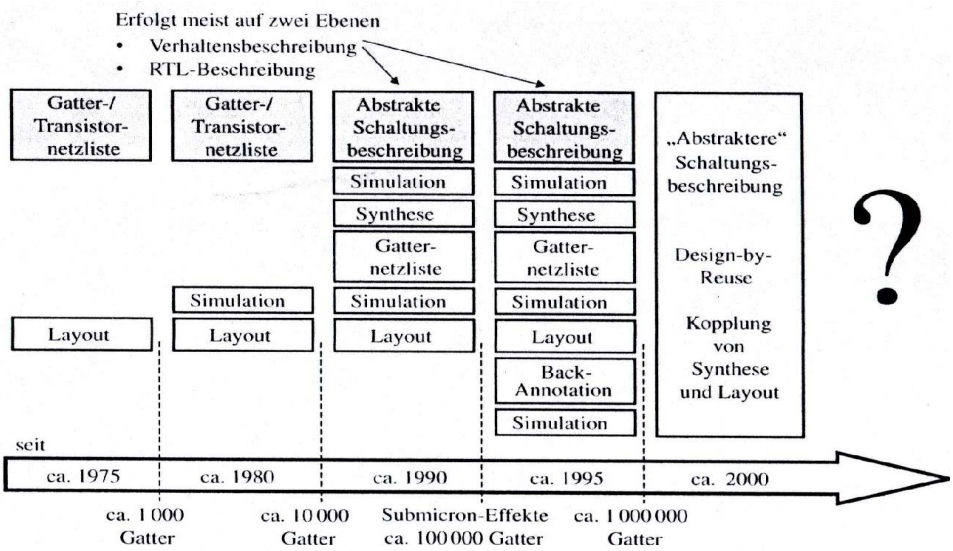

Abbildung 1:Bestandteile des Entwurfsprozesses digitaler integrierter Schaltungen [1]

Um hier Abhilfe zu schaffen, wurde der Entwurfsprozess durch automatisierte Werkzeuge (EDA = Electronic Design Automation) unterstützt. Diese Werkzeuge lassen es zu, dass der Entwurfsprozess in verschiedene abstrakte Ebenen aufgeteilt wird, was für deutlich mehr Übersichtlichkeit sorgt. Neben diesem äusserst nützlichen Feature, können die Werkzeuge auch einen Entwurf simulieren, d.h. der Bau eines teuren Prototypen entfällt. Durch die Simulation der Schaltung können somit Fehler frühzeitig erkannt werden, was zu einer Kostenersparnis führt.
Die Schritte des Entwurfs lassen sich wie folgt aufteilen:

- Entwurfserfassung
- Entwurfsablauf
- Entwurfskontrolle

2. Anforderungen an den Entwurf

Ein Entwurf einer FPGA-Schaltung stellt im wesentlichen folgende Anforderungen:

- Es ist eine möglichst kurze Entwicklungszeit nötig, um die Produkte möglichst schnell auf den „Markt“ zu bringen (engl: *Time to Market*).
- Es ist eine Einarbeitung in neue Bausteine nötig, um diese auch effizient ausnützen zu können.
- Ggf. sollte ein Entwurf unabhängig von der Zielarchitektur sein, falls man ihn auf verschiedenen FPGA's einsetzten will.
- Eine Selbstverständlichkeit sollte auch die Fehlerfreiheit des Produktes sein, sowie die einfache Testbarkeit um Fehler zuverlässig zu entdecken.
- Eine vollständige und aktuelle Dokumentation erleichtert den Überblick über das Projekt.

Um die aufgeführten Kriterien zu erfüllen, sollte der Schaltungsentwurf hierarchisch strukturiert werden. Die Beschreibung der Funktion des Entwurfs, sollte nach Möglichkeit auf einer ziemlich hohen Abstraktionsebene stattfinden. Hierbei lohnt sich der Einsatz von EDA-Werkzeugen, was auch schon zu einer wahrscheinlichen Verminderung möglicher Fehler beiträgt. Desweiteren sollte man auf die einfache Testbarkeit des Entwurfs achten, was eine spätere Simulation und Verifikation vereinfacht. Genauso wichtig ist die Beachtung einiger grundlegender Entwurfsregeln: synchrones Design, keine Torschaltungen bei Taktsignalen, usw.

3. Abstraktionsebenen eines Entwurfes

Abbildung 2 zeigt hierbei die typische Aufteilung in die verschiedenen (Abstraktions-) domänen und ebenen.

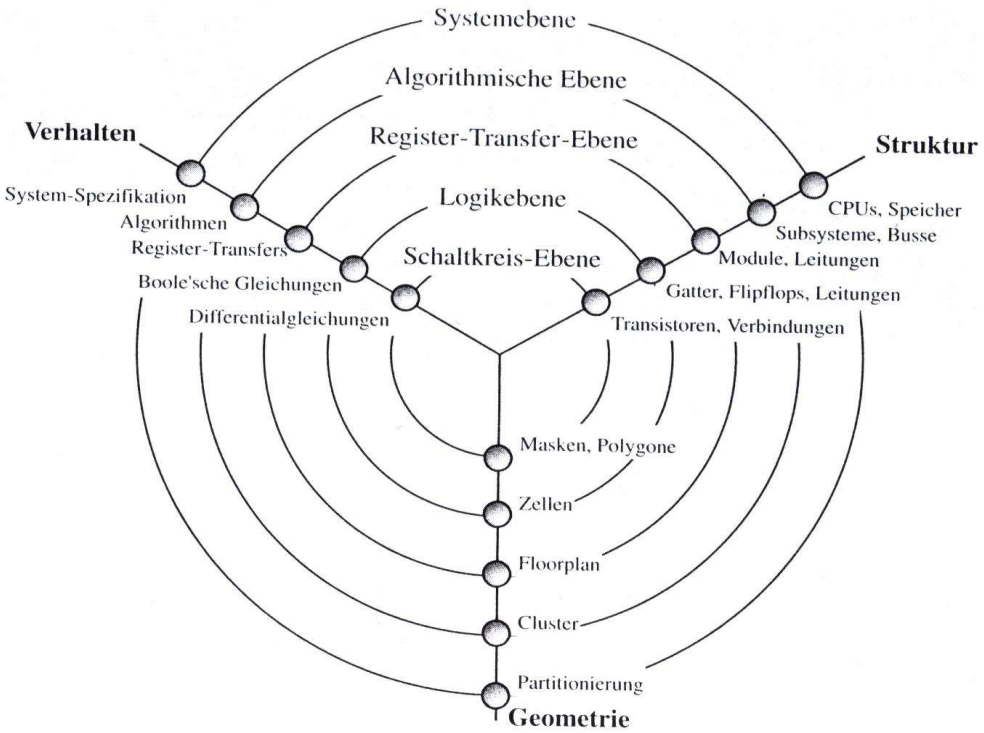

Abbildung 2: Entwurfsdomänen und Abstraktionsebenen im Y-Diagramm [2]

Die einzelnen Ebenen entsprechen den verschiedenen Radien, wobei der äusserste Radius die höchste Abstraktionsebene darstellt. Die einzelnen Äste nennt man (Beschreibungs-)Domänen. Sie spiegeln die drei Sichtweisen des Entwurfs wieder.
In der Verhaltensdomäne beschreibt man im allgemeinen, was man vom System erwartet; also was es machen soll oder nicht, wie es auf was für Eingaben reagieren soll, usw. Hier kommen auch in den oberen 3 Ebenen Hardwarebeschreibungssprachen wie VHDL zum Einsatz.
In der Strukturdomäne wird der Aufbau des Systems mit einfachen Komponenten beschrieben. In den unteren Ebenen sind dies z.B. FlipFlops, weiter oben dann einfache Funktionen wie z:b Addierer oder Register. Noch weiter oben sind dies dann schon komplette Teilsysteme wie z.B. Speicher.
In der Geometriedomäne wird im Normalfall die Anordnung der einzelnen Bauteile spezifiziert, wobei beim FPGA die Geometriedomäne eher als Konfigurationsdomäne angesehen werden muss.
Die einzelnen Ebenen werden wie folgt beschrieben:

- In der Systemebene wird beschrieben, was die Schaltung zu tun hat (Verhalten), welche Bauelemente sie enthällt (Struktur) und die physikalische Realisierung (Geometrie).
- Die Algorithemenebene spiegelt die Art der Realisierung durch gewünschte Funktionen wieder.
- In der Register-Transferebene wird das Verhalten anhand von Transferoperationen der einzelnen Daten zwischen den Registern beschrieben.
- In der Logikebene wird das Verhalten einzelner Logikgatter beschrieben.
- Die Schaltkreisebene stellt den Entwurf der Schaltung mit Elektronischen Grundbausteinen (z.B. Transistoren, Widerstände, ...) dar. Da auf dieser Ebene nicht mehr zeitdiskret gearbeitet wird, d.h. die Signale nicht mehr synchron sondern mit geg. Signallaufzeiten vorliegen, wird sie beim Entwurf eines FPGA's meistens nicht mehr berücksichtigt (der Vollständigkeit halber hier jedoch aufgeführt).

Durch die verschiedenen Domänen und vor allem durch die verschiedenen Ebenen wird nun eine größere Übersichtlichkeit des Entwurfs gewährleistet. Bei verschiedenen Entwurfsproblemen, muss man sich mit diesen jeweils nur auf der dementsprechenden Ebene auseinandersetzen.

4. Entwurfsablauf

Natürlich sollte das wichtigste Kriterium bei einem Entwurf sein, dass die fertige Schaltung nach der Fertigstellung die gewünschten Eigenschaften vorweist. Daneben gibt es aber auch noch andere Bedingungen, auf die man achten sollte. Das wären z.B. Entwicklungskosten , Materialkosten, Fertigungsaufwand, Wartbarkeit und Nachbauschutz.
In Abbildung 3 kann man die typischen Schritte erkennen wie der Entwurf abläuft. Die Synthese bezeichnet hierbei die automatische Transformation von Verhaltensbeschreibungen in Strukturbeschreibungen. Für jede Abstraktionsebene gibt es eine eigene Synthese:

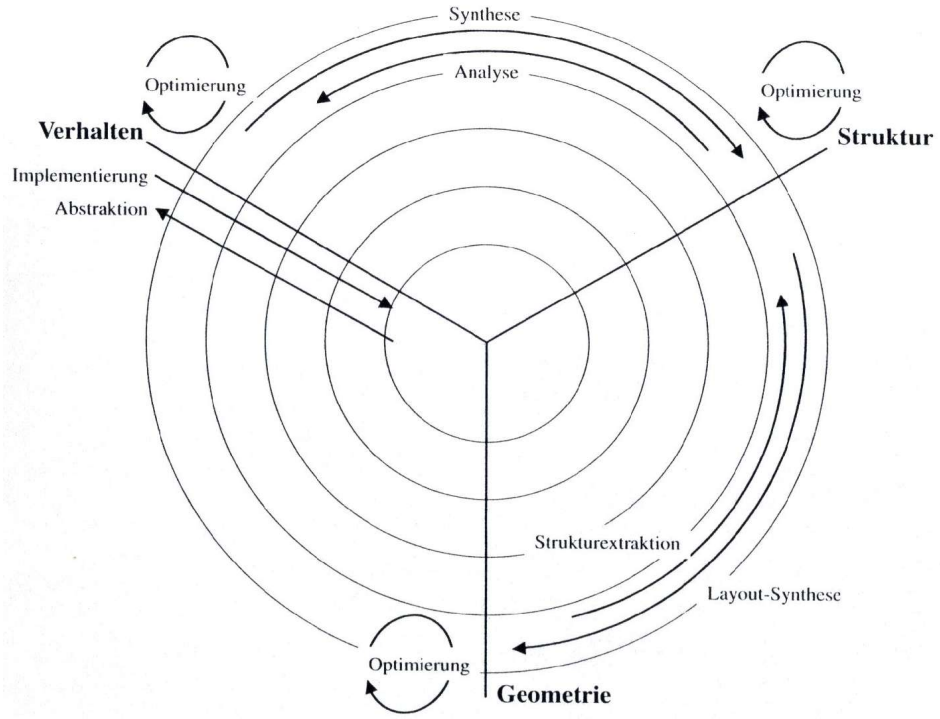

Abbildung 3: Übergänge im Y-Diagramm [2]

- System-Synthese: Sie dient im Allgemeinen zur Partitionierung eines Systems. D.h. sie zerlegt das System in einzelne Teilsysteme , und definiert die Aufgaben der einzelnen Teilsysteme.
- Algorithmische Synthese: Sie dient zur Auswahl der Art und der Anzahl der verwendetetn Komponenten. Ausserdem ordnet sie die durchzuführenden Operationen in einen Zeitplan ein, genauso wie die Zuweisung der einzelnen Operationen auf die verfügbaren Bauteile. Sie erzeugt einen eine Darstellung auf der RT-Ebene mit einem Datenpfad und einem Kontrollpfad. Wobei der Datenpfad alle Elemente zur Speicherung, Manipulation und Weiterleitung von Daten enthällt. Der Kontrollpfad hingegen erzeugt die Signale, die zur Steuerung des Datenpfads erforderlich sind.
- Register-Transfer-Synthese: Hierbei werden die oben gewonnenen Daten- und Kontrollpfade in boolesche Funktionen, FlipFlops, Register und Verbindungsleitungen umgesetzt. Ebenso findet hier die Codierung von Zuständen und die Ansteuerung der verschiedenen FlipFlops statt.
- Logik-Synthese: Hierbei werden die in der RT-Synthese gewonnen Ergebnisse auf logische Grundgatter abgebildet.
- (Schaltungssynthese: Diese Art der Synthese erübrigt sich bei einem FPGA da sie eigentlich die Technologieabbildung durchführt. Dies ist aber beim FPGA nicht erforderlich/ möglich, da die Zielarchitektur ja schon früher ausgewählt wurde.)

Der Entwurf der Schaltungen wird nach dem Prinzip „Teile und Herrsche“ (divide und conquer) durchgeführt, wobei die Schaltung in einzelne Module zerlegt wird. Hierbei wird nach dem Top-Down-Prinzip vorgegangen, d.h. man arbeitet sich von der obersten Abstraktionsebene nach unten durch.

Es gibt verschiedene Arten von Entwurfsabläufen, wobei sie sich nur in der Systembeschreibung unterscheiden. Da wäre z.B. der Entwurf mittels Schaltplaneingabe. Hierbei wird der Schaltplan mit einem grafischen Editor erstellt. Daraus wird dann eine Netzliste generiert, welche den Aufbau des Schaltplans in Textform beschreibt. Neben der direkten Eingabe des Schaltplanes, besteht auch noch

die Möglichkeit Wahrheitstabellen, boolesche Gleichungen, KV-Diagramme und FSMs (Finite State Maschines) direkt einzugeben. Bei oben beschriebener Methode, stösst man insbesondere bei komplexeren Aufbauten ziemlich schnell an die Grenzen. Auch können sich hier schnell kleinere Fehler einschleichen..
Eine andere Art ist die schon erwähnte Beschreibung über eine Hardwarebeschreibungssprache. Ein Vorteil dieser Systembeschreibung ist, neben der Bewältigung komplexer Aufgaben, sicherlich die Möglichkeit, eine unmittelbare Simulation durchzuführen. Nach dieser Simulation kann mittels Synthese die Beschreibung in die Logikebene transformiert werden (vgl. Abbildung 4). Hiermit wird eine sog. Netzliste erzeugt. Diese beschreibt die Verdrahtung der Schaltung in reiner Textform. Sie sagt aber noch nichts über die spätere Verteilung der einzelnen Komponenten im Chip aus. Ist die Netzliste generiert, kommt es zum sog. Place & Route. Hierunter versteht man die Anordnung der einzelnen Schaltungselemente im Baustein und deren Verdrahtung. Dies geschieht meist automatisch. Hier gibt es den Annealing-Algorithmus, welcher für möglichst kurze Verdrahtungslängen sorgt, und das Timing Driven Routing, wo darauf geachtet wird, dass die Vorgaben für kritische Signallaufzeiten beachtet werden. Schließlich, nach Abschluss der verschiedenen Simulationen (siehe Kap. 5) , wird ein sogenannter Bitstream (neben der erweiterten Netzliste mit Komponentenanordnung) erzeugt. Mit diesem verschiedenen Konfigurationsdaten wird das FPGA dann konfiguriert.

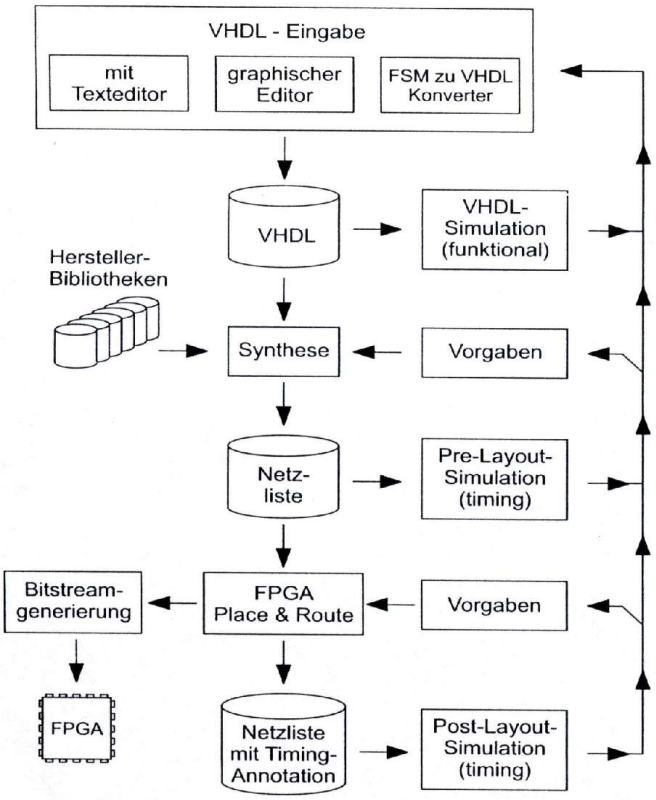

Abbildung 4: Ablauf des FPGA-Entwurfs mit VHDL [2]

Es gibt auch noch Mischwerkzeuge, welche meist ziemlich teuer sind, die beide Eingabemöglichkeiten vereinen, oder sogar die eine auf die andere Möglichkeit transformieren.
Eine gern genutzte Möglichkeit beim Entwurf ist auch die Verwendung von bereits fertigen Baugruppen (Design by Reuse). Hierbei werden einfach fertige, funktionstüchtige Module, die sog. IP-Cores (Intellectual Property), herangezogen und in die Schaltung integriert. Dieses führt auch zur Verminderung von Fehlern, da die IP-Cores im Regelfall schon einwandfrei funktionieren, und somit nur noch auf die „Verdrahtung“ geachtet werden muss.

5. Simulation und Verifikation

Mit einer Simulation wird versucht, in einem Entwurf Entwurfsfehler festzustellen. Neben der VHDL-Simulation gibt es noch die sog. Pre-Layout-Simulation und die Post-Layout-Simulation. Bei der Pre-Layout-Simulation, wird die Funktion der Schaltung ohne fertiges Layout getestet. d.h. nur die Funktion, ohne gegebene Signallaufzeiten. Bei der Post-Layout-Simulation, liegt ja schon die Netzliste mit der Anordnung der verschiedenen Komponenten vor. Somit kann die komplette Schaltung auch bzgl. der verschiedenen Signallaufzeiten getestet werden. Beides geschieht mit einer einfachen Methode, und zwar durch die Eingabe von Testvektoren (= Stimuli), werden die verschiedenen Ausgaben, mittels Funktionaler Verifikation, auf den verschiedenen Abstraktionsebenen überprüft. So können in einer Schaltung Fehler nachgewiesen werden.

Bei der Simulation einer Schaltung mit einem VHDL Simulator kann eine sogenannte Testbench (engl. für Prüfstand) erzeugt werden. Dabei simuliert diese eine reale Umgebung für den zu testenden Entwurf (DUT = Device under Test). Diese überprüft dann, mit dem Stimuli als Eingang, die Ausgabe, wie in Abbildung 5 ersichtlich. Bei einem auftretenden Fehler, wird dann eine Fehlermeldung ausgegeben oder die Simulation abgebrochen.

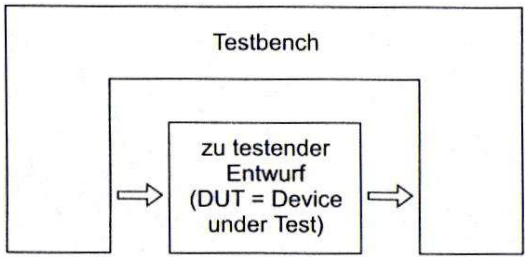

Abbildung 5: Simulation einer Testbench [2]

Eine Simulation garantiert aber keine hundertprozentige Fehlerfreiheit, da das System nicht auf alle möglichen Eventualitäten getestet werden kann. Da die Funktionale Verifikation immer mehr Rechenzeit in Anspruch nimmt (die Anzahl der benötigten Stimuli steigt expotentiell mit der Anzahl der Gatter), wird auch noch die Formale Verifikation verwendet.

Bei der Formalen Verifikation wird schon beim Entwurf die Schaltung auf Fehler kontrolliert. Sie kann als eine automatische Durchführung mathematischer Beweise angesehen werden. Das Resultat entspricht der funktionallen Simulation bei Eingabe aller Testvektoren. Mittels der formalen Verifikation kann man die Anzahl der benötigten Testvektoren bei der funktionalen Verifikation potenziell signifikant verringern.

Wie wichtig eine frühe Fehlererkennung und -beseitigung ist, verdeutlicht das in Abbildung 6 dargestellte Diagramm, die so genannte Zehnerregel. Deshalb entfallen auch mindestens 60 % des Aufwandes bei der Entwurfsentwicklung auf Verifikation und Fehlersuche (Debugging).

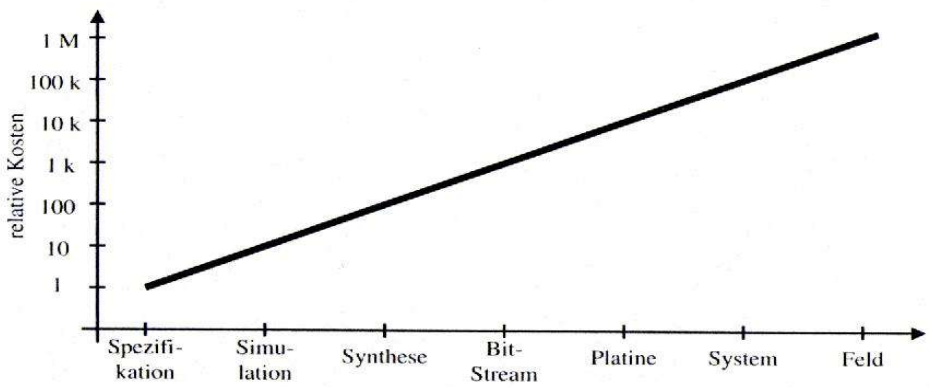

Abbildung 6: Zehner Regel: Kosten der Fehlerbeseitigung [2]

6. Verschiedene Werkzeuge

Für den Entwurf von FPGA's gibt es eine Vielzahl von verschiedenen Entwurfswerkzeugen. Diese CAE-Werkzeuge (Computer Aided Engineering) entstanden in den 70er Jahren, nachdem die Komplexität und Größe der Schaltungen ständig zu nahm. Diese wurden jedoch zuerst bei anderen Bausteinen eingesetzt.
Viele Hersteller von FPGA-Bausteinen bieten hauseigene Entwicklungswerkzeuge an. Diese sind dann meist kostengünstiger, da sie in Kombination mit den Bausteinen vertrieben werden. Die bekanntesten Werkzeuge wären hierbei:

- ISE 5 von Xilinx
- Libero von Actel
- ispLEVER Lattice
- bei Altera meistens abhängig vom Baustein
- ...

Neben den herstellereigenen Tools gibt es auch noch herstellerfremde Tools welche speziell zum Entwurf von el. Bausteinen geeignet sind. Diese zeichnen sich meist durch größere Kompatibilität mit mehreren Bausteinen aus. Die drei größten Hersteller in dieser Sparte wären:

- Cadence (http://www.cadence.com)
- Synopsys (http://www.synopsys.com)
- Mentor (http://www.mentor.com/)

Viele Hersteller bieten nicht nur komplette Plattformen an, sondern auch einzelne Werkzeuge. Man kann somit verschiedene Werkzeuge, von verschiedenen Herstellern, miteinander kombinieren, um eine individuelle Entwicklungsumgebung zu erhalten. Die Unterteilung der Werkzeuge gestaltet sich meistens wie folgt:

- Design
- (IP Integration)
- Simulation
- Debug und Analyse
- Synthese

Die meisten Werkzeuge wurden nicht speziell für den FPGA-Entwurf entwickelt, sondern allgemein für den Entwurf von el. Bausteinen. Es gibt auch Werkzeuge die aus völlig anderen Bereichen stammen, wie z.B. MATLAB. Dieses stellt vor allem ein Analyse-Tool dar, kann aber in Kombination mit Simulink auch zum Entwurf verwendet werden. Mit Simulink und MATLAB kann man dann problemlos die Modellierung, Analyse, Simulation, usw. eines Systems realisieren. Simulink bietet dem Benutzer z.B. auch Bibliotheken an, die vordefinierte Funktionen / Anweisungen/Bauteile/etc. Enthalten, genauso wie z.B. das automatische „Verdrahten“ (also die Leitungsführung). (vgl. auch www.mathworks.com).
Man kann die Hardwarebeschreibungssprachen auch noch zu den Werkzeugen zählen. Ganz vorne ist hierbei VHDL. Neben VHDL sind auch noch andere Hardewarebeschreibungssprachen erhältlich, wie VERILOG, SystemC, ... Eine Sprache, die immer mehr im Kommen ist, ist SystemC. Hierbei handelt es sich um eine freie C/C++-basierte Modellierungsplattform (eigentlich eine

Klassenbibliothek für C++, die neue Datentypen integriert), die es ermöglicht, Systemkomponenten unterschiedlichster Abstraktionsebenen, Entwurfsdomänen und Applikationsbereiche zu vereinigen. Dies geschieht innerhalb eines einheitlichen Systemmodells. Hierbei wird auch eine frühzeitige Simulation des Entwurfs ermöglicht (vgl. auch www.SystemC.org).

7. Entwurfswerkzeug ISE 5 von Xilinx

Laut Xilinx, soll ISE 5 (ISE 5.1i) die erste komplette Plattform für das SystemDesign sein. Neben den Virtex-II Pro Bausteinen erlaubt die Software auch den Einsatz an anderen Xilinx Bausteinen. ISE 5.1i verspricht weiterhin die jeweilige Bausteinperformance um über 20%, im Vergleich zu konkurrierenden Produkten, zu steigern. Es ist lauffähig unter den neueren Windowsversionen, Solaris und Linux.

Es enthällt verschiedene Editoren, Navigatoren, grafische Tools zum Design, Simulatoren, ein Tool zur formalen Verifikation, ... Hier einige Beispiele:

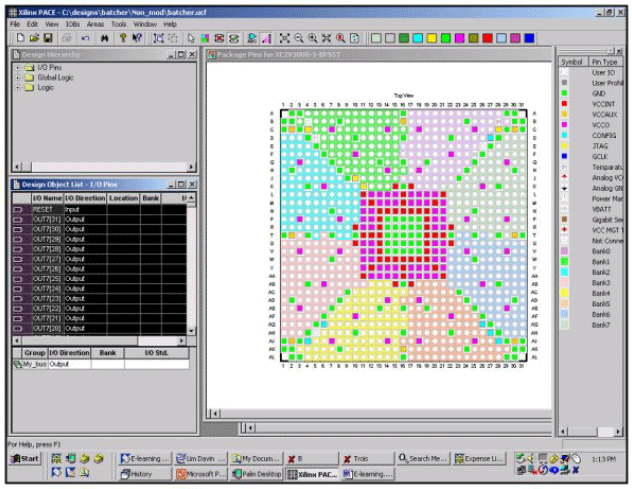

Abbildung 7: ISE5 Entwicklungswerkzeug von Xilinx (PACE)

- HDL-Editor welcher alle Eigenschaften moderner Editoren besitzt, so wie kontextsensitiv, Farbcodierungen, ... Er unterstützt VHDL, VERILOG und ABEL.
- Floorplaner: er erlaubt es grafisch, mittels Drag&Drop, die Anordnung der verschiedenen Logikblöcke zu optimieren.
- Architecture Wizarrds erlauben es die Bausteinfunktionen von Virtex-II und Virtex-II Pro optimal auszunutzen. (z.B. Digital Clock Manager)
- FPGA Editor Probe erlaubt es einem in Echtzeit das Verhalten des Bausteines an bestimmten Signalen zu überprüfen. Ausserdem kann man mit ihm immer genau feststellen, was für Schritte als nächstes anfallen.
- PACE(= Pinout und Area Constraints Editor), den Abbildung 7 zeigt. Der Editor ermöglicht dem Anwender die Durchführung des Floorplannings für I/Os und Logik. Hierbei können auch die IO-Management-Funktionen angewendet werden, welche eine grafisches Management der Pins ermöglichen.
- Chip Scope (Pro), ein Analyse- und Debugtool auf Logikebene.

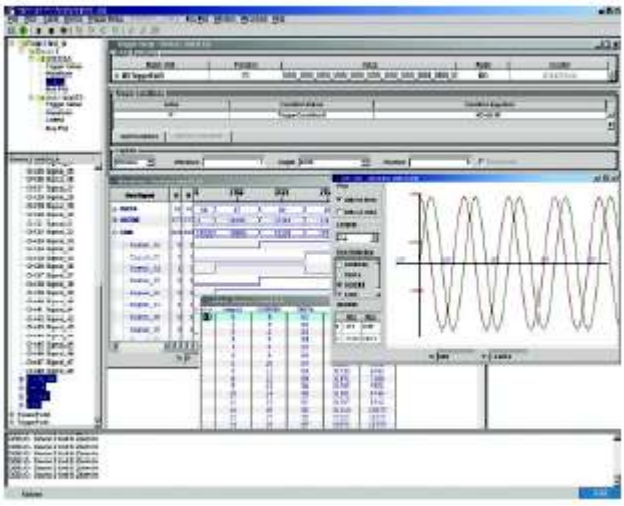

Abbildung 5: Chipscope Analysetool von Xilinx

Desweiteren benutzen die Tools der ISE 5 Plattform folgende Optionen:

- Incremental Design stellt sicher, dass bei der Änderung des Designs nur betroffene Gebiete re-implementiert werden.
- Macro-Builder erlaubt es frühere Designs, oder Teile davon, in späteren Entwürfen einzubauen (Reuse).
- Modular Design ermöglicht die Realisierung des Divide-and-Conquer-Prinzips, indem es ein komplexes Design in mehrere kleine Teilmodule zerlegen lässt.

8. Quellenangabe

- [1] A. Sikora. Programmierbare Logikbauelemente – Architekturen und Anwendungen. Carl Hanser Verlag

- [2] M. Wannenmacher. Das FPGA-Kochbuch (mit CD-ROM). Thomas Publishing, 1998.

- Webpages:
  - www.Xilinx.com
  - www.mathworks.com
  - www.SystemC.org
  - www-ti.informatik.uni-tuebingen.de
  - www.hpe.fzk.de