Atmel FP*SLIC*

Seminararbeit von
Wolfgang Becker
Matr. Nr.: 010340

erstellt im
Sommersemester 2003
an der
Fachhochschule Karlsruhe
betreut von
Prof. Dr. Albrecht Ditzinger

Atmel FPSLIC, ein System on a Chip: Überblick über die auf dem Chip vorhandene Hardware und Betrachtung des Entwicklungsablauf bei Verwendung des Atmel System Designer Pakets.

Inhaltsverzeichnis

1 Atmel

Die im kalifornischen San Jose ansässige Halbleiterfirma Atmel wurde 1984 gegründet. Seither ist das Unternehmen zu einem führenden Anbieter in seinem Bereich aufgestiegen. Zur Produktpalette gehören nichtflüchtige Speicher ebenso wie ASICS, RAM oder Mikrocontroller. Mit Fertigungsstätten in Nordamerika und Europa sowie Weltweit verteilten Filialen entwickelt, produziert und vertreibt Atmel seine Produkte selbst.

2 Das FPSLIC

Für embedded Devices wird das Co-Design immer wichtiger. Das bedeutet: Hard- und Software soll gleichzeitig entwickelt werden können. Bereits vor fertigstellen eines Prototyps soll das neue System am Simulator getestet werden und Programme dafür geschrieben werden.

Für ein „System on a Chip“ werden mehrere Komponenten, die für einen Rechner nötig sind werden auf einem Chip zusammengefasst. Dadurch benötigt das Gesamtsystem weniger Platz. Die Herstellung wird günstiger da nur ein Gehäuse gebraucht wird. Bei der Schaltungsbestückung muß nur ein Bauteil verbaut werden. Die Kommunikation zwischen den Komponenten geht schneller.

Die AT94 FPSLIC Familie von Atmel zielt auf diesen Markt. FPSLIC steht für „**F**ield **P**rogrammable **S**ystem **L**evel **I**ntegrated **C**ircuits. „Field Programmable“ weil auf dem Chip ein FPGA, also ein Field-Programmable Gate Array“ enthalten ist. Es ist „System Level“ weil der FPSLIC einen Mikrocontroller, SRAM und Peripherie enthält. Alle Komponenten befinden sich auf einem Chip als integrierte Schaltkreis (Integrated Circuit).

3 Die Komponenten

Abbildung 1 zeigt den schematischen Aufbau eines AT94K.

3.1 Der AVR Kern

Der verwendete Mikrocontrollerkern ist ein AVR 8-Bit RISC. Er wird mit einer Taktrate von 25 MHz betrieben. AVR ist ein Standard Mikrocontroller, der ebenfalls zu Atmels Produktpalette gehört. Als Einzelbauteil ist dieser Mikrocontroller weit verbreitet und wird gerne z.B. im Automobilbereich verwendet.

Die embedded Variante erreicht etwa 1Mips/MHz. Da die meisten Maschinenbefehle innerhalb eines Taktes ausgeführt werden erreicht der AVR auf dem FPSLIC mit seinen 25MHz bis zu 25 MIPS, im Normalbetrieb etwa 20 MIPS. (Million Instructions Per Second)

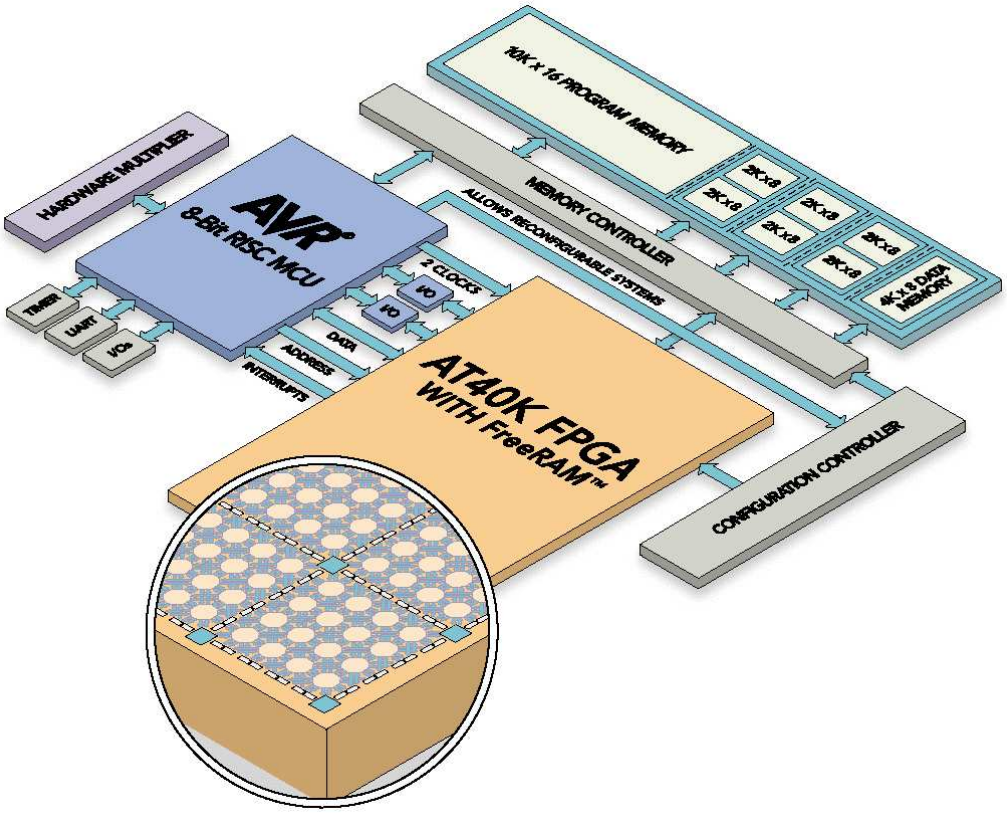

Abbildung 1: Schema AT94K

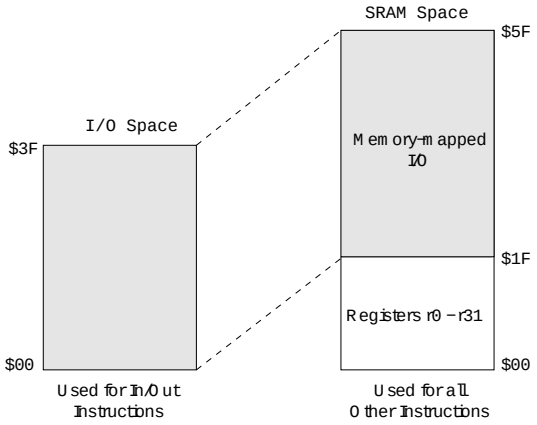

Abbildung 2: Memory-mapped I/O

Der AVR-Kern ist ein Niedrigenergie CMOS 8-Bit Mikrocontroller der auf der AVR RISC Architektur basiert. Diese erweiterte RISC Architektur verfügt über 32 8Bit Register. All diese Register sind direkt mit der ALU (Arithmetic Logic Unit) verbunden so daß mit einer einzigen Anweisung auf zwei unabhängige Register zugegriffen werden kann. Diese Anweisung wird innerhalb eines Taktes ausgeführt. Diese Architektur erzielt einen bis zu 10 Mal höheren Befehlsdurchsatz als ein CISC (Complex Instruction Set Computing, Sammelbezeichnung für nicht RISC R=reduced) Mikrocontroller.

AVR verwendet die Harvard Architektur: Programm- und Datenspeicher sind getrennt[1] . Der Prozessor hat getrennte Busse für Instruktions- und Datenzugriff. Dadurch ist Parallelverarbeitung möglich. Während eine Anweisung ausgeführt wird kann bereits die Nächste gelesen werden. Dadurch kann in jedem Taktzyklus eine Anweisung ausgeführt werden. Da das bei bedingten Sprüngen nicht funktioniert ist 1 MIPS/MHz die maximal mögliche Leistung, realistisch wird etwa 0.8 MIPS/MHz erreicht

Die Register werden in den Speicher abgebildet. Jedes Register hat eine Adresse im Datenbereich. Genauso ist der I/O Bereich „Memory Mapped“ wie in Abbildung 2 dargestellt. Die I/O Ports sind mit der Peripherie verbunden.

3.2 Ram

Das FPSLIC verfügt je nach Version über bis zu 36KBytes SRAM zur Programm- und Datenspeicherung. 4K sind auf jeden Fall für Daten, 20K (4K beim AT94K05AL) sind fix als Programmspeicher zugeordnet. Der restliche Speicher (12K) kann in Blöcken zu 4K frei verteilt werden. Der einzige Unterschied zwischen den verschiedenen Mitgliedern der AT94K Familie was den Speicher angeht ist also die die Größe des fest zugeordneten Programmspeicher.

[1] Im Gegensatz zur von Neumann Architektur ist selbstmodifi zierender Code nicht m¨oglich.

Das verwendete SRAM (Static RAM) benötigt keinen refresh, Daten werden solange gehalten wie genügend Strom vorhanden ist. SRAM ist schneller und braucht weniger Strom als ein DRAM.

Sowohl der AVR Kern als auch das FPGA können auf Programm- und Datenspeicher zugreifen.

3.3 FPGA

Das FPGA ist technisch ebenfalls ein Standardprodukt. Im Gegensatz zur CPU ist das FPGA mit 100MHz viermal so schnell getaktet wie die CPU und eignet sich besonders für aufwendige Berechnungen als schneller Coprozessor oder DSP. Der Hautpgeschwinndigkeitsgewinn liegt weniger in der Taktrate, als in der Tatsache, daß die FPGA-Logik in Hardware besteht und nicht in auszuführendem Code.

Der Aufbau des FPGAs entspricht der Architektur des AT40K. Durch verwenden dieser bereits bekannten Architektur kann bestehendes Know How weiterverwendet werden. Auch die zur Programmierung notwendige Software kann übernommen werden.

Jede Zelle im FPGA steht mit allen acht Nachbarn in Verbindung. Dadurch ist es möglich, komplexe Funktionen ohne Buszugriff auszuführen.

Das FPGA enthält SRAM Blöcke direkt zwischen den Zellen. An den Ecken jedes 4×4 Blocks aus Zellen befinden sich SRAM Speicher. Die 16 Byte Speicher sind 32×4 organisiert. Dieses FreeRam genannte System bietet den Vorteil, daß Ram da verfügbar ist, wo es gebraucht wird. Durch den kurzen Weg zum Ram sind weniger Verbindungen innerhalb des FPGA nötig.

3.4 Peripherie auf dem Chip

Auf dem Chip ist an Peripherie ein Hardware Multiplexer, zwei 8 Bit Timer, zwei UARTs, eine serielle Schnittstelle und ein 16 Bit Timer in Hardware vorhanden. Des weiteren gibt es einen Watchdog Timer und eine JTAG Schnittstelle.

Der Multiplexer ist ein häufig benötigtes Bauteil, der falls er nicht in Hardware vorhanden wäre oft im FPGA aufgebaut werden müßte.

Die Timer können entweder intern getaktet werden, oder als Zähler mit externem Eingangspin verwendet werden.

Das serielles Zweidrahtinterface ist eine bidirektionale Standardschnittstelle. An sie können mehrere Geräte angeschlossen werden. Die CPU kommuniziert mit diesem Zweidrahtbus über spezielle I/O Register.

Mit den UARTs (universal asynchronous receiver-transmitter) ist asynchrone serielle Kommunikation möglich. Sie werden zur Steuerung serieller Schnittstellen verwendet.

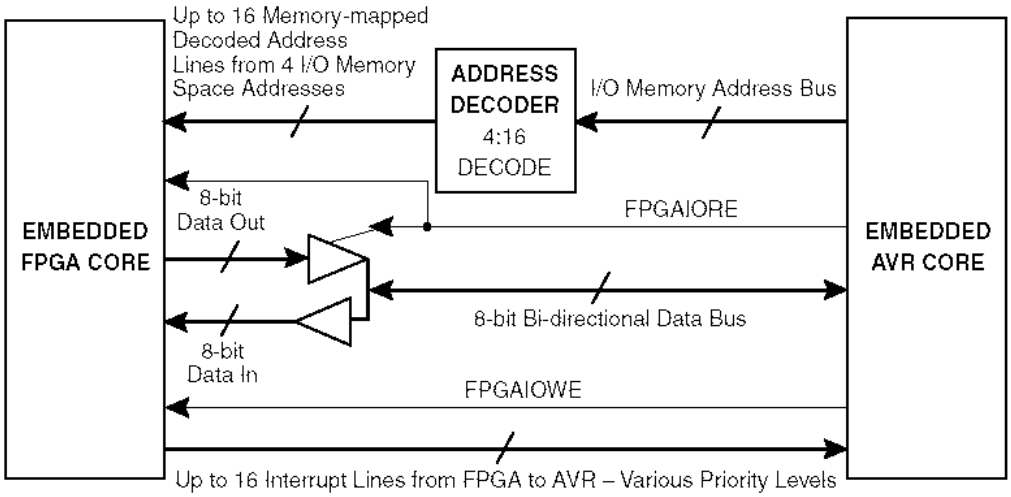

Abbildung 3: Mögliche Verbindungen zwischen AVR und FPGA.

Die JTAG Schnittstelle ist eine genormte Schnittstelle zum Debuggen von integrierten Schaltkreisen. Sie wird auf vier Pins nach außen geführt. Mehr zu JTAG im Abschnitt 4.5. Diese Schnittstelle ist nicht auf allen FPSLIC vorhanden.

3.5 Verbindung AVR/FPGA

Für eine effiziente Zusammenarbeit zwischen dem AVR-Kern und dem FPGA ist die Übertragung von Daten zwischen diesen beiden Komponenten entscheidend. Um dies zu gewährleisten können beide auf das SRAM zugreifen.

Abbildung 3 zeigt die Verbindungen, die mit dem „Interface Connections“ Dialog (siehe 4.3) im System Designer programmiert werden.

Zum einen ist das eine Datenbusverbindung. Das AVR Datenbusinterface ist direkt mit dem Bus des FPGA verbunden, so daß das FPGA als großes I/O Device behandelt wird.

Mittels zu 16 „decoded address lines“ können direkt AVR Speicherzugriffe direkt auf Schnittstellen des FPGA gemappt werden.

Das FPGA hat die Möglichkeit Interrupts zum AVR zu schicken. Dabei sind bis zu 16 verschiedene Interrupts möglich.

3.6 Configuration Controller

Beim Booten des Chips ist der Configuration Controller dafür zuständig, die Einstellungen des FPGA, die Speicherkonfiguration und den Speicherinhalt aus dem seriellen EEPROM zu lesen und das System entsprechen einstellen.

Zur Laufzeit kann der Configuration Controller auf Befehl des AVR hin das FPGA umprogrammieren. Auch die Speicheraufteilung ist zur Laufzeit änderbar.

3.7 EEPROM

Das FPSLIC ist flüchtig. Sowohl der Inhalt des SRAM als auch die Konfiguration des FPGA gehen beim Ausschalten verloren.

Nach dem Einschalten oder einem Reste muß sich das FPGA seine Daten aus einem EEPROM holen. In der Standardausführung hat das FPSLIC kein EEPROM on board so daß zum Booten des Systems ein externer Programmspeicher benötigt wird.

Hierfür gibt es das FPSLIC Support Device. Atmel hat zu jedem AT94Kxx ein passendes ATFSxx im Programm. Dieses serielle EEPROM verwendet ein spezielles Format zur FPGA Konfiguration. Es benötigt ein Clock-Signal sowie eine bidirektionale Signalleitung. Des weiteren muß Clock Enable gesetzt sein, um den Baustein auszuwählen.

Da das FPGA nur sequentiell lesen muß können, wenn nicht alles Daten in ein einzelnes Bauteil passen, diese seriellen EEPROM kaskadiert werden. Das FPGA muß nicht wissen, wie groß das angeschlossenen EEPROM ist, nachdem eines ausgelesen wurde schaltet es selbstständig seinen Data-Output ab und aktiviert den nächsten Nachbarn in der Kaskade.

Der Reseteingang des Hilfsbausteins wird mit dem FPSLIC-Reset verbunden, damit beide Bausteine gleichzeitig initialisiert werden. Ansonsten konnte das EEPROM asynchron zum FPGA werden.

In der Secure-Version des FPSLIC ist ein Konfiguarations-EEPROM auf dem Chip integriert.

3.8 Die verschiedene Mitglieder der FPSLIC Familie

Die verschiedenen Mitglieder der AT94k FPSLIC Familie unterscheiden sich durch die Anzahl der FPGA Gates. Es stehen zwischen 5K und 40K zur Verfügung. Auch die SRAM Größe ist, wie in Abschnitte 3.2 beschrieben, Modellabhängig. Zur Zeit gibt es drei verschiedene Speicherkonfigurationen: 5K FPGA Gates und 16KB SRAM, 10K FPGA Gates und 32KB SRAM sowie 40K FPGA Gates mit 32KB SRAM.

Des weiteren hat die „Secure“ Variante ein EEPROM auf dem Chip. Diese AT94S gibt es in den drei Speichergrößen. Die Secure Variante hat einen geschützten Modus, in dem nur eine komplettes löschen der Programmierung, aber kein auslesen von Daten möglich ist. Um das Chip umzuprogrammieren muß es im Sicherheitsmodus zunächst vollständig gelöscht werden

Es gibt eine Ausführung des FPSLIC in einem 84-Pin PLCC Gehäuse, die Varianten mit höheren Pinzahlen (100, 144, 208, und 240) verwenden flat-pack SMT Gehäuse. Für die Secure Variante wir ein 256-Pin CABGA verwendet. Dann gib es noch die ganz große Bauform: 352-Pin in einem BGA Gehäuse. Auf der Platine im Starterkit ist die 208-Pin SMT Version

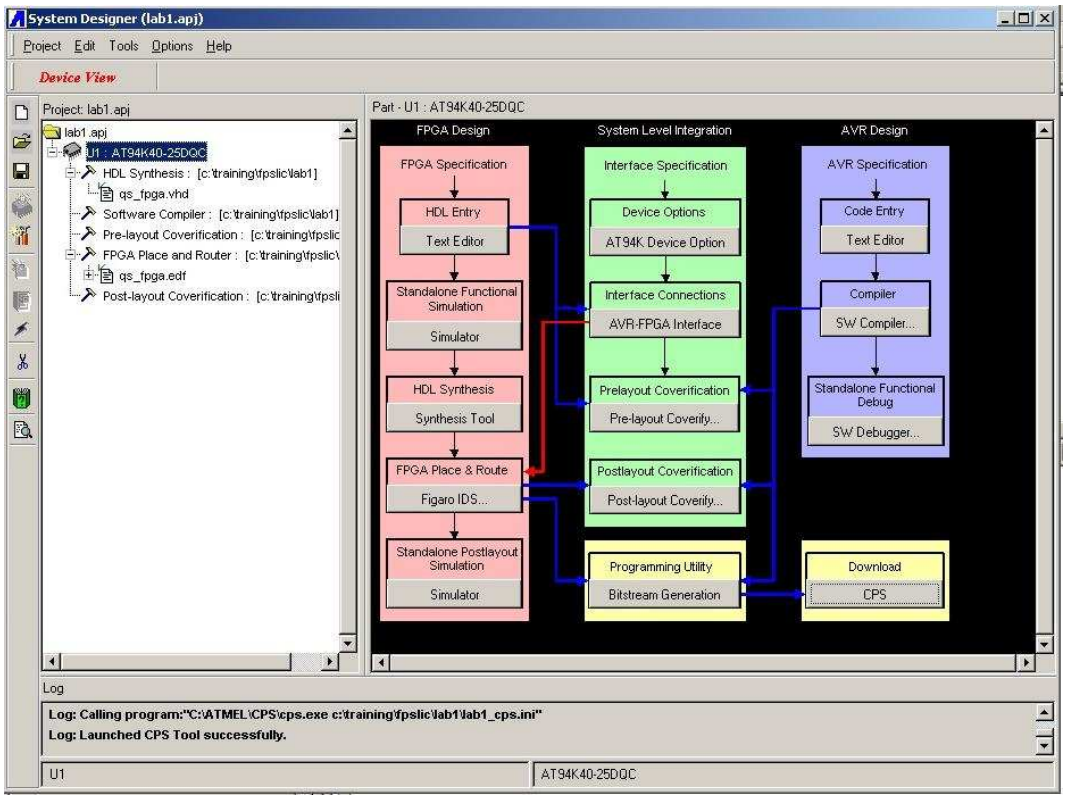

Abbildung 4: Der Atmel System Designer unterstützt den gesamten Entwicklungsablauf.

aufgelötet. Die meisten FPSLIC Versionen sind in mehreren Gehäusetypen erhältlich. Alle Varianten im gleichen Gehäusetyp sind Pinkompatibel.

Zusätzlich gibt es das FPLIC noch mit oder ohne JTAG ICE Support. Mehr zum „In Circuit Emulator“ im Abschnitt 4.5.

4 Entwicklung mit dem Atmel System Designer

Der System Designer ist ein von Atmel entwickeltes Werkzeug zur Programmentwicklung auf für das FPSLIC. Es eignet sich für alle AT94, für die K Serie genau so wie für die S Serie. Diese unter dem Betriebssystem Windows (ab 95) laufende Entwicklungsumgebung ist im Starterkit enthalten. Der Entwicklungsprozess läuft in drei teilweise parallelisierbaren Pfaden ab. Für die beiden Komponenten AVR und FPGA gibt es jeweils einen Entwicklungspfad, diese werden durch den „System Level Integration Pfad“ verbunden. Abbildung 4 zeigt den Entwicklungsablauf, wie er im System Designer dargestellt wird.

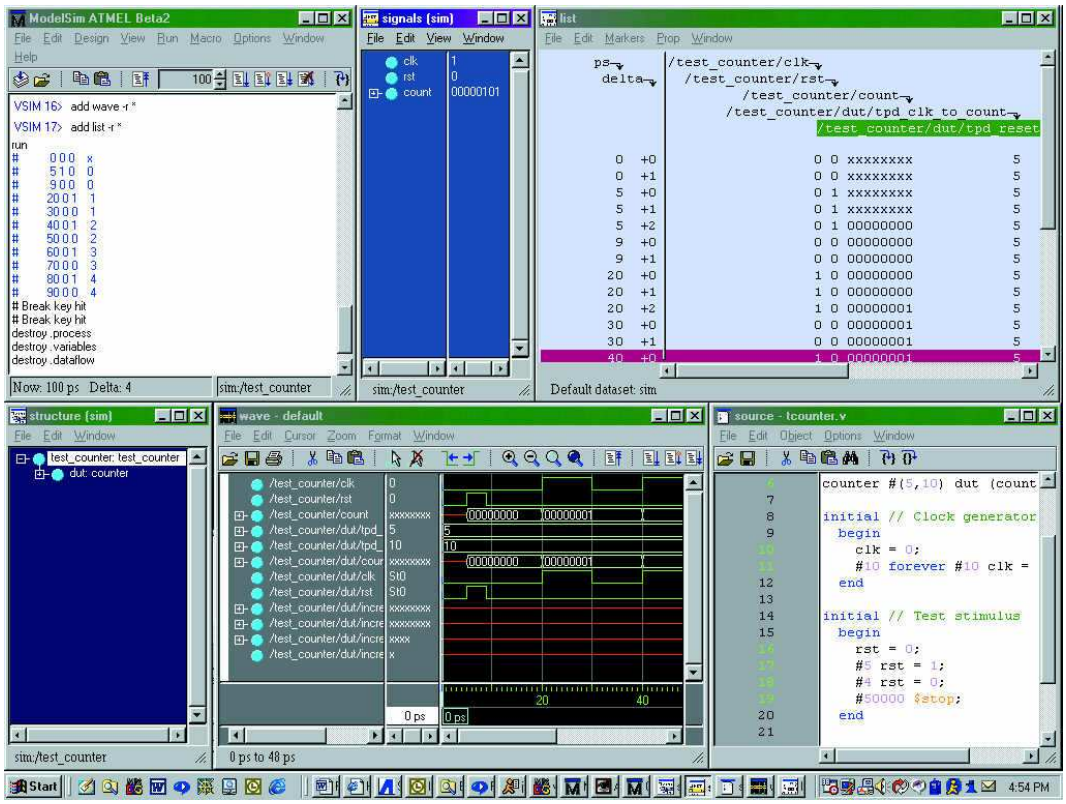

Abbildung 5: ModelSim erlaubt den Test des FPGA-Entwurfts am PC.

4.1 Ablauf des FPGA Entwurfs: der FPGA Design Path

Zu Beginn des FPGA Design Pfads im System Designer wird eine HDL (Hardware Description Language) Beschreibung des FPGA erwartet. Sowohl Verilog als auch VHDL ist möglich. Man kann diese Datei entweder von Hand erstellen, oder mit einem grafischen Codegenerator wie dem im Paket enthaltenen „HDL Planner“ erzeugen.

Sobald die Beschreibung des FPGA fertig ist geht es weiter zum nächsten Schritt: der funktionalen Simulation. Hier kommt ModelSim von Mentor Graphics zum Einsatz. (Abbildung 5) Für die Simulation wird eine „Testbench“ benötigt. Das ist eine vom Entwickler zu erstellende Datei, die dazu verwendet wird, Eingaben für den zu testenden Hardware-Entwurf zu erzeugen. Die Simulation ist immer nur so gut, wie die verwendeten Testdaten.

Dieser Schritt ist optional und kann auch dann durchgeführt werden, wenn die FPGA Entwicklung unabhängig von der AVR-Entwicklung erfolgt. Hier wird nur das FPGA getestet.

Bisher besteht eine allgemeingültige Beschreibung der des logischen Aufbaus der Schaltung. Um diese jetzt auf die in der Hardware vorhandenen Gates auf dem FPGA anzupassen bedarf es der HDL Synthese. In diesem Schritt wird das Design analysiert und auf die Zielhardware angepasst. Das Programm „LeonardoSpectrum“ (Abbildung 6) ist für das erstellen der

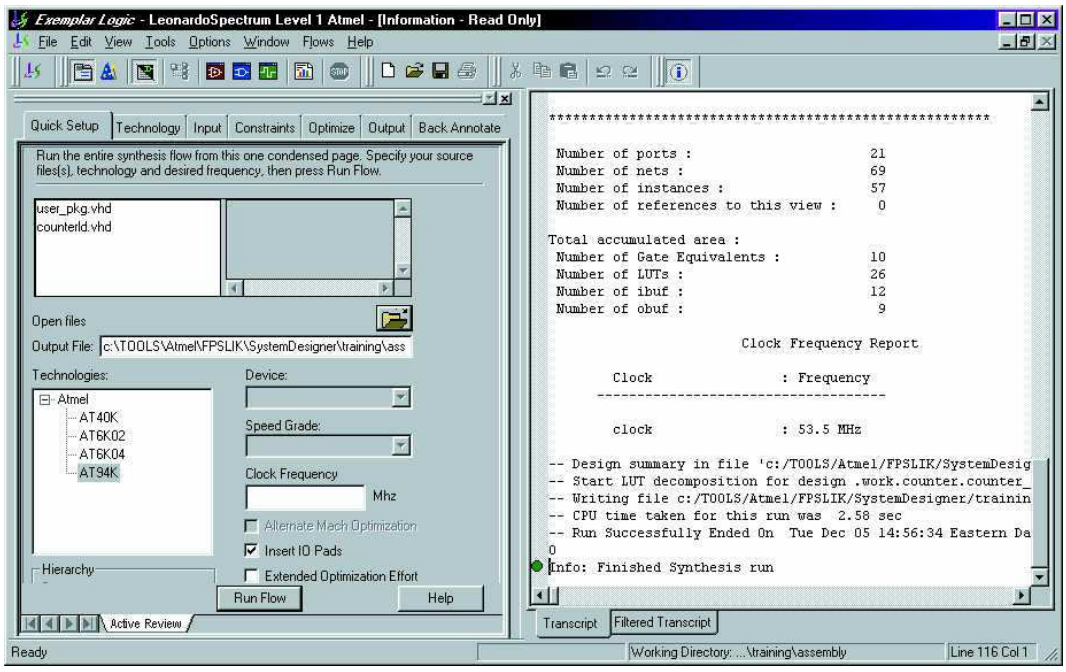

Abbildung 6: Leonardo Spectrum passt den FPGA Entwurf für die Zielhardware an.

Netzlistendatei zuständig.

Diese Netzliste wird im nächsten Schritt, „FPGA Place & Route“ in das Programm „Figaro“ importiert. Figaro ist für die Platzierung der Schaltung in den einzelnen Gates und den Entwurf der Verbindung verantwortlich. Mittels der Netzliste bestimmt Figaro die für das Design benötigten Zellen. Platzierung und Verbindung ist ähnlich wie beim Entwurf von Leiterplatinen. Zunächst werden Zellen unabhängig von den Verbindungen ausgewählt. Dann werden sie verschoben, bis die benötigten Verbindungen möglich sind. Je nach Komplexität des Designs und gewünschter Optimierung ist dieser Vorgang sehr rechenintensiv. Der Prozess läuft automatisch ab, es ist aber möglich das Ergebnis von Hand zu optimieren. Das Ergebnis wird als Schaltplan wie in Abbildung 7 angezeigt. Figaro erzeugt die zur Programmierung der Hardware nötige Bitstream Datei.

Besonders wenn von Hand Verbesserungen vorgenommen wurden bietet sich der optionale letzte Schritt im FPGA-Entwurfsablauf an: die Postlayout Simulation. Hiermit kann überprüft werden, ob sich das fertige Design immer noch so verhält wie es soll.

4.2 Programmierung des AVR: der AVR Design Path

Der erste Schritt, die Codeeingabe ruft einen Texteditor auf. Dieser kann verwendet werden, um Quellcode einzugeben. Laut „System Designer User Guide“ werden die Sprachen C/C++, Assembler und Pascal unterstützt.

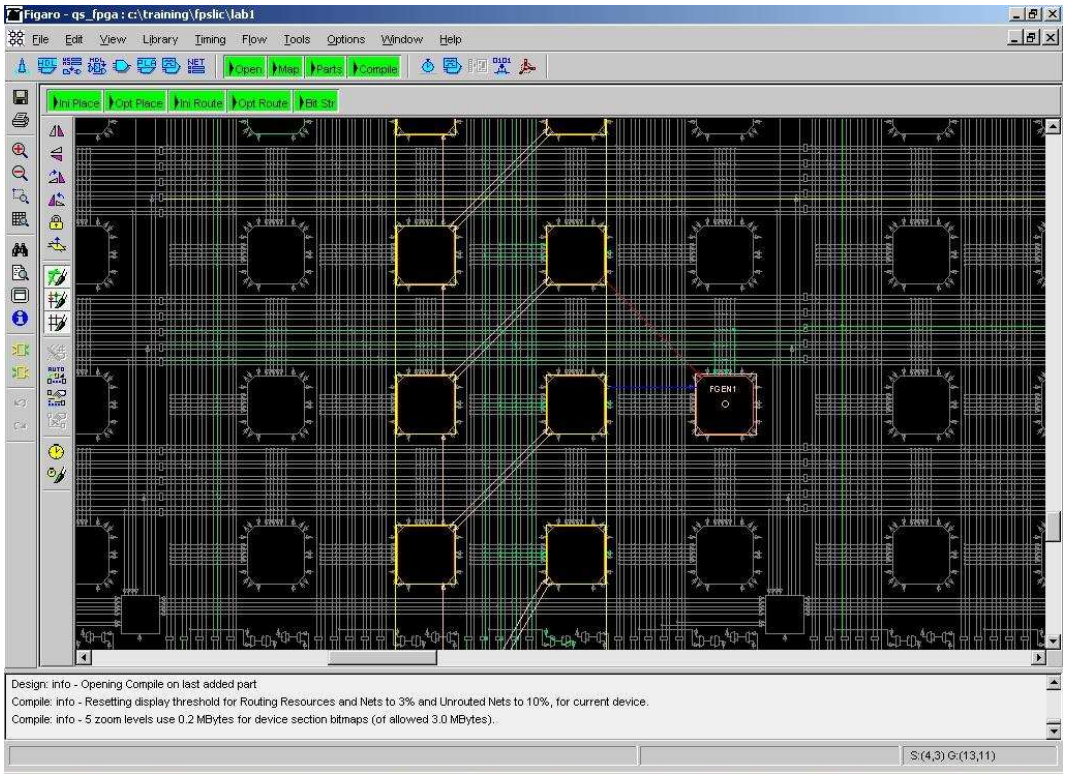

Abbildung 7: Figaro: FPGA Place & Rout FPGA Place & Route

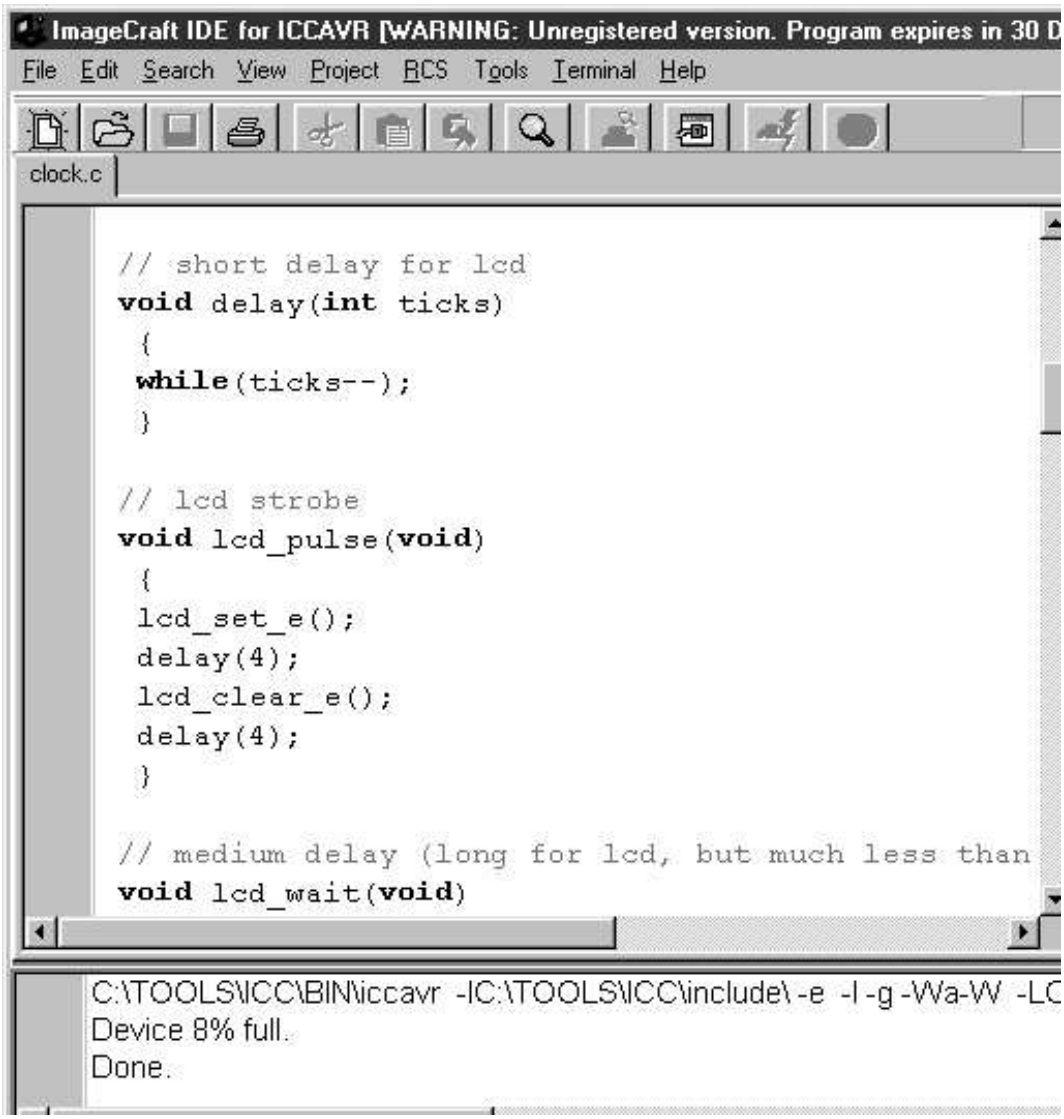

Abbildung 8: ImageCraft C Compiler

Mit dem im Starterkit enthaltenen Wavrasm Assembler wird Assemblerquellcode im zweiten Schritt compiliert. Für C Quellen ist im Starterkit eine 30 Tage Demoversion des ImageCraft C Compilers enthalten, die in Abbildung 8 zu sehen ist. Im System Designer kann eingestellt werden, welcher Compiler verwendet werden soll. Je nach gewünschter Sprache muß ein passender Compiler von einem Dritthersteller erworben werden.

Wie aus dem Designflow in Abbildung 4 hervorgeht wird die Compilerausgabe sowohl für die Co-Verifikation im System Level Integartions Pfad verwendet als auch zur Programmierung des AVR Kerns auf dem Chip.

Die letzte Funktion, die der AVR Design Pfad anbietet ist der funktionale Debugger. Die Fehlersuche im AVR Code kann unabhängig vom Entwicklungsstand des FPGA begonnen werden.

Hier endet der AVR Design Pfad. Die Programmierung ist abgeschlossen und das Programm kann auf die Hardware überspielt werden.

4.3 Zusammenarbeit auf dem Chip: System Level Integration Path

Bis jetzt haben wir die Entwicklung für AVR und FPGA getrennt betrachtet. Um aber wirklich einen Vorteil vom System on a Chip zu haben müssen

Zuerst sind gerätespezifische Einstellungen vorzunehmen. Normalerweise findet dieser Schritt bereits vor Beginn der übrigen Entwicklung statt. Es muß entschieden werden, wieviel Speicher für Code und wieviel für Daten zur Verfügung steht. Es muß festgelegt werden, mit welchem Takt das FPGA betrieben wird.

Darüber hinaus muß sich der Entwickler klar werden, welche Teile seines Entwurfs in Hardware auf dem FPGA realisiert werden und was als Programm auf dem AVR. Gerade wenn als Team entwickelt wird ist eine genaue Festlegung der Zuständigkeiten nötig.

Sobald die HDL Beschreibung des FPGA fertig ist ist können die physikalischen Verbindungen zwischen dem FPGA und dem AVR festgelegt werden. Den Dialog zum Herstellen dieser „Interface Connections“ zeigt Abbildung 9. Das linke Feld zeigt die möglichen Anschlüsse am AVR-Kern, rechts stehen die des FPGA. Um ein Paar zu verbinden werden einfach beide mit der Maus ausgewählt und der „Connect“ Button gedrückt. Auf Wunsch kann automatisch eine Vorlage für eine entsprechende „Testbench“ generiert werden.

Nachdem der AVR-Code fertig ist kann der erste Durchlauf der Co-Verifikation beginnen. Die Co-Verifikation erlaubt es, den fertigen AVR-Code mittels einer Simulation auszuführen. Hierzu wird wieder ModleSim eingesetzt. Das erste Mal berühren sich hier die Entwicklungspfade für FPGA und AVR.

Das FPGA Entwicklerteam kann mit der HDL Synthese weitermachen während die Co-Verifikation läuft. Durch die Parallelisierung dieser Pfade kann Entwicklungszeit eingespart werden. Zumindest solange, bis bei der Co-Verifikation Fehler entdeckt werden.

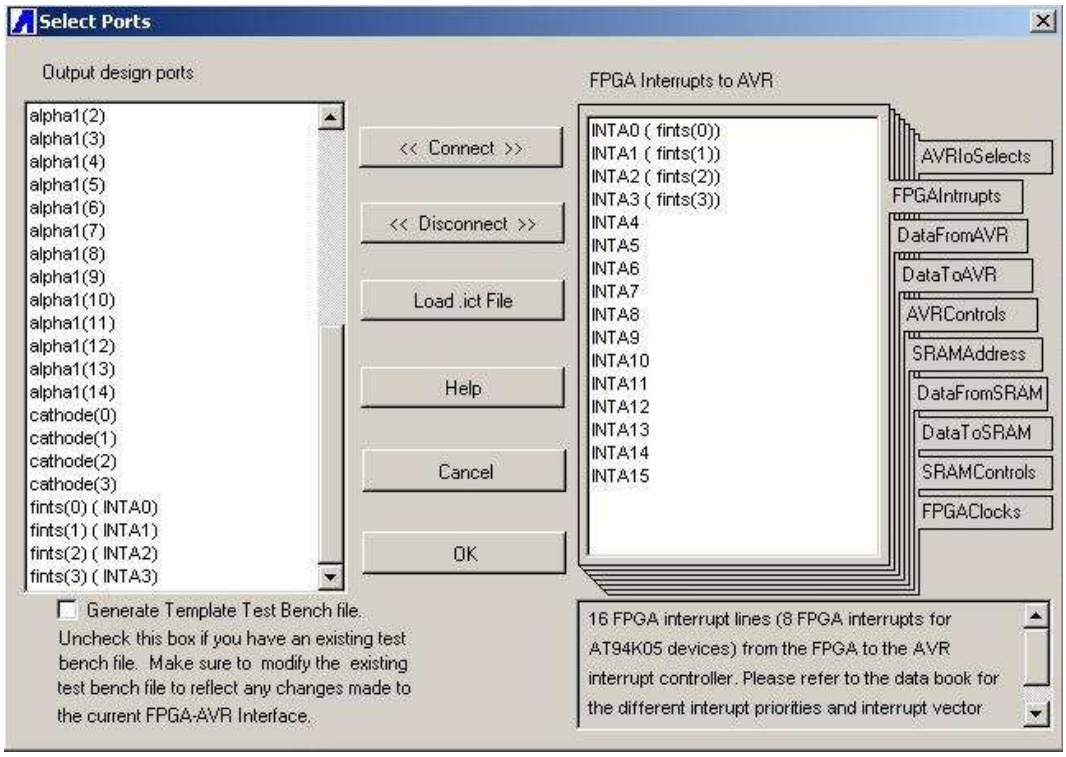

Abbildung 9: Interface Connections: Verbinden von FPGA und AVR ohne löten per Mausklick.

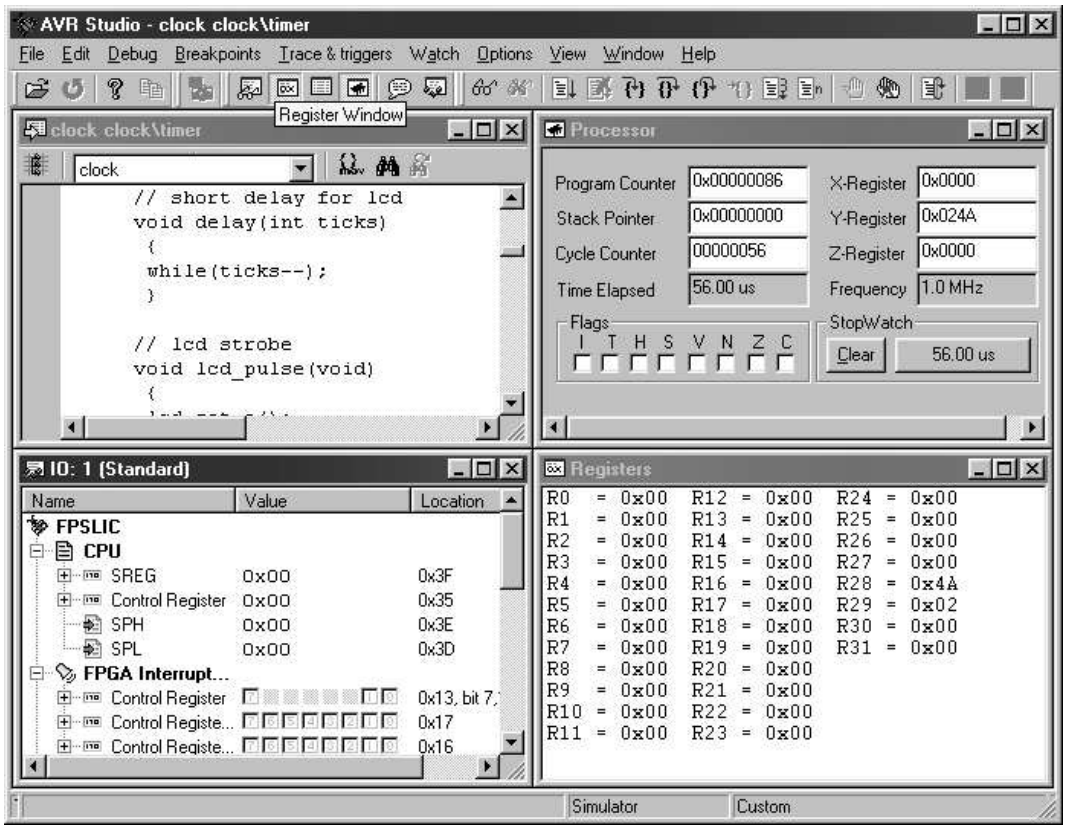

Abbildung 10: Der System Designer kann gleichzeitig das FPGA und den AVR Kern simulieren.

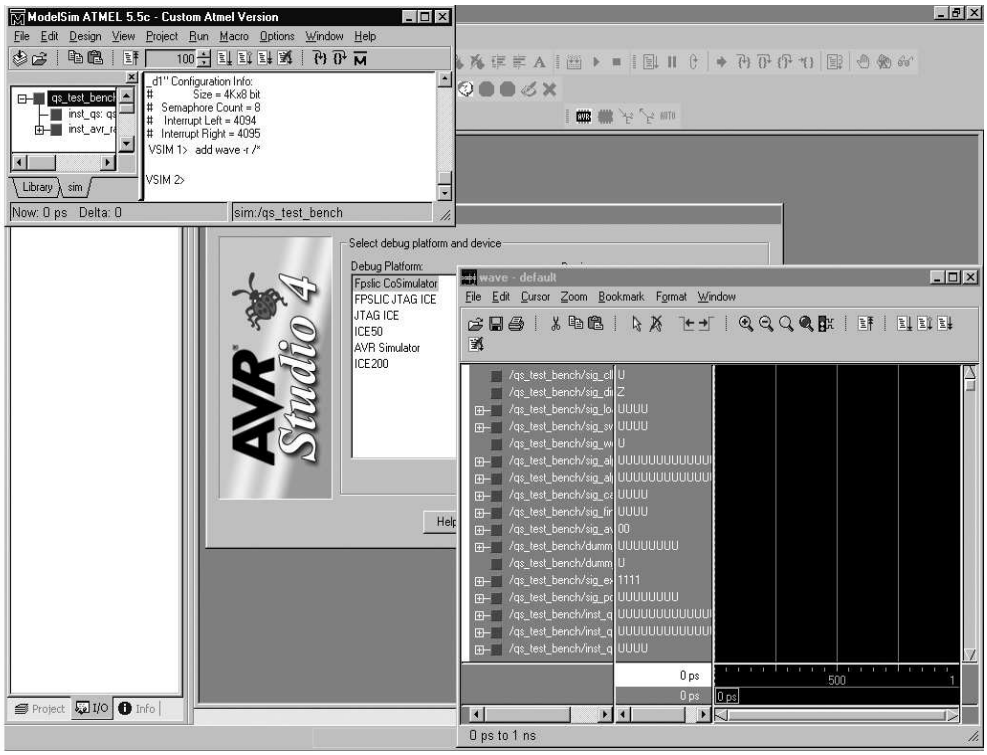

Abbildung 11: Detailansicht der Co-Simulation

Sobald das FPGA Entwicklerteam mit Synthese, Layout und Routing fertig ist beginnt der zweite Durchlauf der Co-Verifikation. In diesem zweiten Durchlauf wird eine Simulation unter der Berücksichtigung der Timinginformationen vom FPGA Plazierungs- und Routingprozess durchgeführt. Die Abbildungen 10 und 11 zeigt die gleichzeitige Simulation von AVR und FPGA.

Gewöhnlich wird jetzt auch ein Prototyp der Platine des Geräts, in dem das System on a Chip eingesetzt werden soll, entworfen. Es ist auch möglich, das FPSLIC Starter Kit als frühen Prototyp zu verwenden.

4.4 Übertragung der Programmierung auf das Chip

Eine Bitstreamdatei zur Übertragung auf das Chip kann, wie in Abbildung 4 gezeigt, aus dem Ergebnis der „FPGA Place & Route“ Vorgangs und dem Compilierten AVR Code erzeugt werden. Es ist jedoch auch Möglich, nur eine dieser beiden Komponenten zu verwenden. Dies kann zu Testzwecken sinnvoll sein. Normalerweise nimmt man aber Beide.

Die fertige Bitstreamdatei wird nun mittels dem „Configurator Programming System“ (CPS) auf des serielle EEPROM übertragen. Da das FPLIC selbst nur flüchtigen Speicher enthält liest es bei jedem Neustart seine Konfiguration und Programmierung aus diesem Baustein ein.

4.5 In System Debuging

Da beim FPSLIC die Programmierung des FPGA des AVR gleichzeitig erfolgt, ergibt sich bei der Fehlersuche die Schwierigkeit: Ist es ein Software- oder ein Hardwareproblem? Zur Fehlersuche gibt im Rahmen des Entwicklungsprozess mehrere Debuggingmöglichkeiten. Wie beschrieben lassen sich FPGA und AVR jeweils einzeln in eine Simulation testen. Auch die gemeinsame Simulation beider Komponenten ist möglich.

Trotz all dieser Möglichkeiten der Simulation ist es oft erforderlich, direkt im echten System nach Fehlern zu suchen. Hierzu dient der JTAG ICE. Der Joint Test Access Port In-Circuit Emulator. Die JTAG-Schnittstelle ist in IEEE 1149.1 genormt. Der ICE ist ein Werkzeug, mit dem ein On Chip Debuging für alle AVR 8-bit Mikrokontroller und alle FPSLIC mit JTAG Unterstützung ermöglicht wird.

Bei der Emulation wird das Verhalten des Chips per Software simuliert. Beim On-Chip Debuging dagegen werden die zu testenden Anweisungen vom Zielsystem ausgeführt.

Das System Designer Software Paket unterstützt diese Art der Fehlersuche. Einige Sonderfälle sind jedoch zu beachten. Die Software kann Schrittweise ausgeführt werden, jedoch laufen I/O Operationen weiter, auch wenn ein Breakpoint erreicht wurde. Der Watchdog-Timer muß für diese Form des Debuggens abgeschaltet werden, da er sonst das Gerät zurücksetzt.

5 Fazit

Das FPSLIC kombiniert zwei Produkte,FPGA und AVR, die jedes für sich weit verbreitet sind. Beide brauchen spezielle Entwicklerwerkgzeuge und fundiertes Fachwissen zum effizienten Einsatz. Alle nötigen Tools um das FPGA und AVR zu einem ganzen zusammenzufassen sind im System Designer enthalten. Da hier viele Produkten von Drittherstellern integriert wurden sind einige Lizenzen nötig. Im Gegensatz zu anderen Hardwarekomponenten werden Entwicklertools nicht als Zugabe bereitgestellt sondern müssen extra gekauft werden. Das Starterkit einhält eine Kurzlizenz des System Designers, nach diesen sechs Monaten sind für jedes weitere halbe Jahr etwa 500 Dollar fällig. Dazu kommen noch die Kosten für einen Compiler. Die Laufzeitbeschränkung der Software ist ein häufig geäußerter Kritikpunkt.

Falls bisher schon Entwicklungssoftware für die Einzelkomponenten vorhanden ist kann diese in den System Designer integriert werden. Gegenüber den Einzelkomponenten bietet das FPSLIC eine höhere Geschwindigkeit bei geringerem Stromverbrauch und kleinerer Bauform. Atmel empfiehlt den Einsatz des FPSLIC Aufgrund dessen für den Einsatz in mobilen Geräten.

Das FPSLIC ist eine interessante Kombination von Bauteilen, mit der Secure-Variante ist auch ein gewisser Schutz des Programms vor unbefugtem auslesen gewährleistet.

Durch den schnelleren Entwicklungszyklus ermöglicht das FPSLIC eine schnellere Fertigstellung des Produktes. Ob sich das seit 2000 auf dem Markt befindliche System durchsetzen kann, oder ob es ein Nischenprodukt bleibt muß sich noch zeigen.

A Abkürzungen

ALU	Arithmetic Logic Unit
ASIC	Application Specific Integrated Circuit
AVR	Bezeichnung für die MCU. Bedeutet laut Atmel garnichts. (nach den Entwicklern „Alf (Egil Bogen) and Vegard (Wollan) 's Risc processor“)
BGA	Ball Grid Array
CABGA	Chip Array BGA
CISC	Complex Instruction Set Computing (Sammelbezeichnung für nicht RISC CPUs)
CMOS	Complementary Metal Oxide Semiconductor
CPU	Central Processing Unit
DRAM	Dynamic RAM
DSP	Digitaler Signal Prozessor
FPGA	Field Programmable Gate Array
FPSLIC	Field Programmable System Level IC
HDL	Hardware Description Language
IC	Integrated Circuit / Integrierter Schaltkreis
JTAG	Joint Test Action Group
MIPS	Million Instructions Per Second
MCU	Micro Control Unit
PLCC	Plastic Leaded Chip Carrier
RAM	Random Access Memory
RISC	Reduced Instruction Set Computing
SMT	Surface Mount Technology
SRAM	Static Random Access Memory
UART	Universal Asynchronous Receiver-Transmitter
VHDL	VHSIC HDL
VHSIC	Very High Speed Integrated Circuits

B Links

Die meisten Informationen sind nur in englischer Sprache verfügbar. Einzelne Berichte sind aber (auch) auf deutsch vorhanden.

ATMEL Homepage. Die wichtigste Quelle für Datenblätter und Beschreibungen zum FPSLIC.
http://www.atmel.com/

Atmel Ships FPSLIC For Portables: Ein Product Review
http://www.chipcenter.com/pld/products_600-699/prod601.htm

Ausführlicher Artikel über das FPSLIC von Jeff Bachiochi
http://www.circuitcellar.com/pastissues/articles/jeff128/text.htm

Electronic Component News Artikel über das Starterkit
http://www.e-insite.net/ecnmag/index.asp?layout=article&articleid=CA255249

Eine weitere Vorstellung des FPSLIC mit schöner Abbildung des Dies.
http://www.chipcenter.com/circuitcellar/april01/c0401su3.htm

Das Yahoo Groups Diskussionsforum zum FPSLIC
http://groups.yahoo.com/group/fpslic/

Kurzinformation zum AVR vom Farba Research Support
http://www.farbaresearch.com/atmelavr.htm

Dontronics - ATMEL AVR Devices and 8051 Family.
http://www.dontronics.com/atmel.html

Ein deutschsprachiger Artikel über die Zusammenarbeit von Atmel und Mentor Graphics
http://www.mentor.com/germany/infobox/presse/2002/0402oem.htm

Bericht in den Embedded News über den kommerziellen Einsatz des FPSLIC duch die Firma WaveBolt.
http://www.embeddedstar.com/press/content/2002/7/embedded4574.html

Eine deutschsprachige Vorstellung des Secure Variante des FPSLIC der CODICO GmbH.
http://www.codico.com/de/impulse022002/impulse-313.html

Ebenfalls von CODICO. ein allgemeiner FPSLIC-Artikel auf deutsch
http://www.codico.com/de/impulse032000/impulse-A01.asp

Die Avrfreaks stellen den einen Gnu-Compiler für AVR bereit.
http://www.avrfreaks.com/AVRGCC/index.php

Artikel von Greg Ratzel über System on a Chip
http://www.commsdesign.com/story/OEG20020906S0077

Ein Artikel über die Hardware des FPSLIC von Chih-Chieh Han
http://www.ee.ucla.edu/ simonhan/ee202a/hw1.htm

Diese Seminararbeit in elektronischer Form, sowie die Folien zum Vortrag gibt es unter http://www.fh-karlsruhe.de/~bewo0012/seminar/ im Internet.