Modulare Grafische Programmierung (MGP) von FPGAs

Dr.-Ing. Jörg Pospiech
AVT GmbH Ilmenau
Am Hammergrund 1
D-98693 Ilmenau
Tel.: +49 (0) 3677 64790
Fax.: +49 (0) 3677 647999
Mail: j.pospiech@avt-ilmenau.de
Web: www.avt-ilmenau.de

1 Einführung

FPGAs (Field Programmable Gate Arrays = frei programmierbare Schaltkreise) zählen zu den leistungsfähigsten digitalen Schaltkreisen der Rechentechnik. Sie werden insbesondere in embedded Systemen als universelle programmierbare Lösung eingesetzt.

Die wichtigsten Vorteile von FPGAs sind:

- sehr schnell für rechenintensive Anwendungen durch parallele Berechnungen
- kleine Abmessungen, geringer Energiebedarf und geringe thermische Beanspruchungen
- flexibel durch beliebig oft programmierbare Logikblöcke
- sehr viele frei definierbare Ein- und Ausgänge (mehr als 1100) mit vielen programmierbaren Standards
- keine bzw. wenige Redesigns der Leiterplatten
- schnelle Entwicklung von Designs mit komfortablen Software-Werkzeugen verkürzen „time-to-market“
- Integration vieler zusätzlicher Komponenten in den FPGA durch Hardware-Programmierung, z.B. CPU, I/O Steuerung, RAM Steuerung, Filter, ...
- Einbindung von Soft-IP-Cores als Expertenwissen in die Anwendung
- ständige Verbesserung des Preis/Leistungsverhältnisses durch die Weiterentwicklungen neuer FPGAs

Diese Vorteile machen ein „System-on-Reprogrammable-Chip“ (SoRC) mit geringeren Gesamtkosten für das Design möglich. Leiterplattenlayouts können, speziell beim Einsatz von Softcontrollern (ladbare Controller bzw. CPUs) im FPGA, vereinfacht werden.

Typische FPGA-Applikationen sind Lösungen mit Echtzeitanforderungen, System-on-Chip Design, Videobearbeitung und umfangreiche Kommunikationslösungen. Die Anwendungsmöglichkeiten nehmen bei ständig steigender Leistungsfähigkeit der FPGAs zu.

2 Programmierung von FPGAs

FPGAs entwickeln ihre volle Leistungsfähigkeit durch eine effektive Programmierung ihrer Hardware. Bei früheren Programmiertechniken wurden Netzlisten der FPGA-Systemgatter generiert. Die Netzlisten werden mit Hilfe des vom FPGA-Hersteller gelieferten Implementierungswerkzeugs in Programmierdateien für den FPGA (bit files) umgerechnet. Bei größeren FPGAs entstehen jedoch sehr große und unübersichtliche Netzlisten. Die Handhabung und Fehlersuche gestalten sich dadurch sehr schwierig, so dass sinnvollerweise andere Programmierwerkzeuge eingesetzt werden sollten.

Hardware-Beschreibungssprachen (Hardware Description Languages, HDL) sind kommandozeilenorientierte Programmier- bzw. Verifikationswerkzeuge aktueller Designbeschreibungen. Beispiele dieser Beschreibungssprachen sind VHDL oder Verilog. Synthesewerkzeuge sind in der Lage, Netzlisten aus den HDL Dateien als Basis für die Implementierung zu erzeugen. Damit ist auch eine Programmierung von komplexen Funktionen, sowie von hochkomplexen FPGAs möglich. Dieser Ablauf der Programmierung des Designs ist im Bild 1a dargestellt.

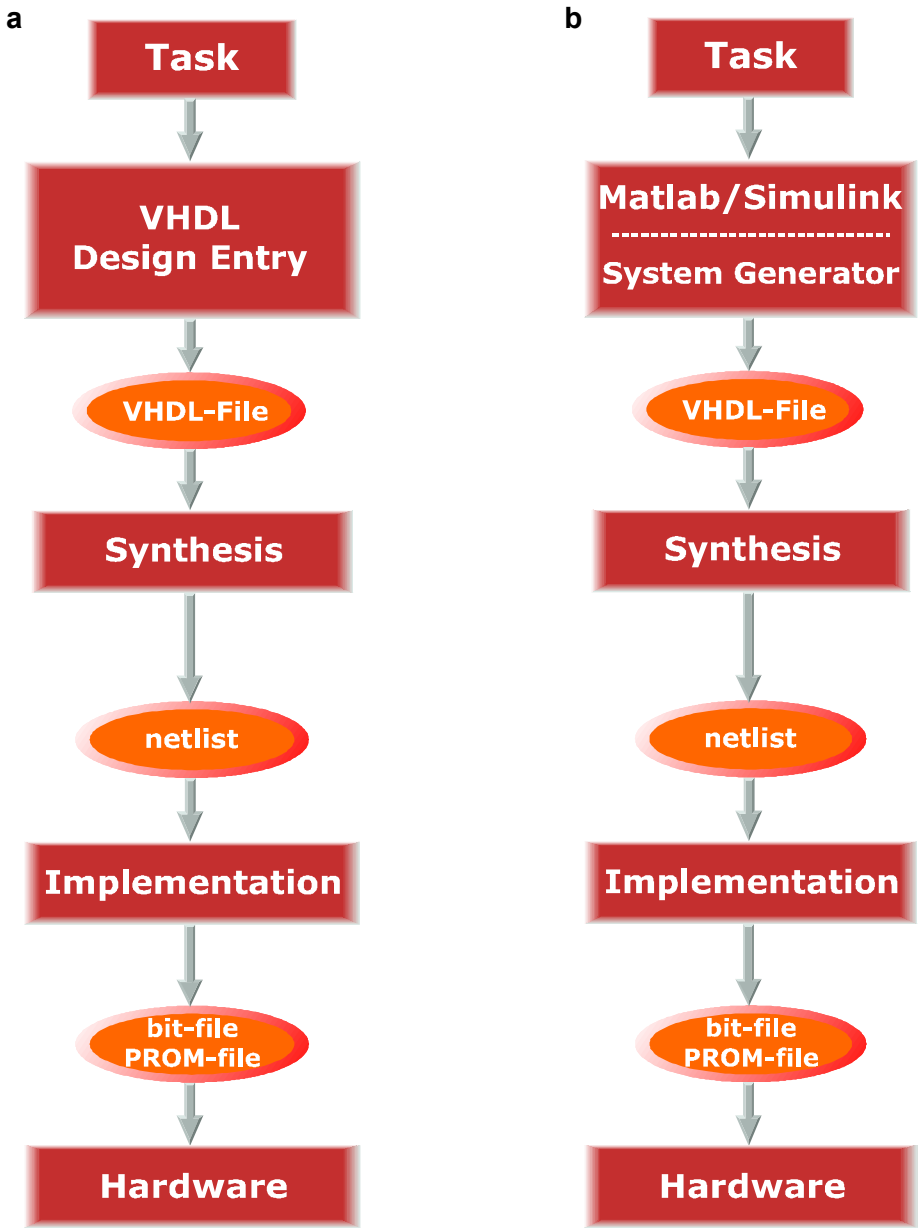

Bild 1: Aktueller Designablauf (design flow) für die FPGA Programmierung

Der Nachteil dieser Applikationsprogrammierung ist die notwendige sehr gute Kenntnis der Hardware-Beschreibungssprache. Applikationsingenieure benötigen dazu eine lange Einarbeitungsphase, verbunden mit hohen Personal- und Schulungskosten.

Neuere Möglichkeiten erlauben die grafische Programmierung auf einer hohen Abstraktionsebene. Das bekannteste Beispiel ist die Kooperation von „The Mathworks Company“ und XILINX mit der Programmierplattform Matlab/Simulink und XILINX System Generator für

DSP (Digital Signal Processing). Mit diesen Programmierwerkzeugen kann der Applikationsingenieur die Anwendung mit FPGA-kompatiblen Logikblöcken programmieren und simulieren. Die Ergebnisdateien des XILINX System Generators sind kombinierte HDL und Netzlisten-Dateien für die Berechnung im Implementierungswerkzeug (Bild 1b).

Diese Designbeschreibung garantiert einen guten Überblick über die Programmierung der Anwendung, beinhaltet die grafische Programmdokumentation, kann Testbenches erzeugen und ermöglicht die bit- und zeitgenaue Simulation (Bild 2).

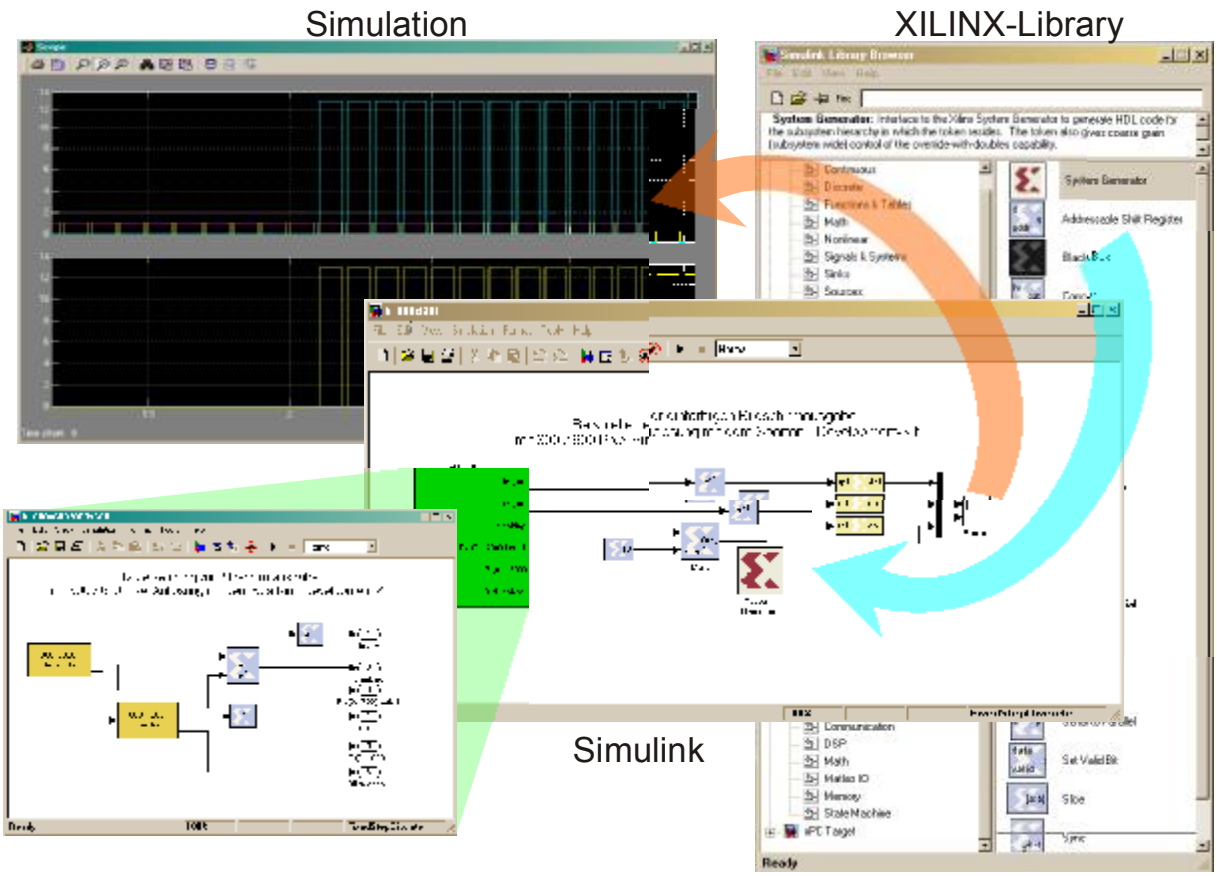

Bild 2: Designbeschreibung mit Matlab/Simulink für die Programmierung und Simulation

Die Nachteile dieser Designbeschreibung sind:

- notwendige Kenntnis der HDL Syntax, der kommandozeilenorientierten Matlab Scriptsprache und der grafischen Simulink-Programmierung (lange Einarbeitungszeit verbunden mit hohen Personalkosten)
- hohe Softwarekosten für die Nutzung von Matlab/Simulink, zusätzlichen Toolboxen, zusätzlichen Blocksets, XILINX System Generator und Implementierungswerkzeug

Von kleineren Firmen und Universitäten hört man oft die Argumente, daß die Leistungsfähigkeit, die Kosten, die Einstiegszeit bzw. Einarbeitungszeit in diese Technik nicht einschätzbar sind. Insbesondere die hohe aktuelle Arbeitsbelastung ist meist Anlaß, die technische Entwicklung bzw. Neuausrichtung nicht mitzugehen. Weiterhin wird von hohen Kosten für die Planung und Entwicklung der FPGA Hardware ausgegangen. Diese Nachteile stellen hohe Schwellen für die Nutzung von FPGAs dar.

Das Problem der Hardwareentwicklung kann inzwischen mit kleinen, leistungsfähigen und umfangreich ausgestatteten (RAM, Flash, Video, serielle und Ethernet-Kommunikation, ...)

Development Kits gelöst werden. Dabei ist es bei einigen Anbietern möglich, die Hardwarebestückung der Development Kits für finale Anwendungen auf die notwendigen Komponenten zu reduzieren. Auf diese Weise können kleine bis mittlere Serien getesteter Platinen ohne eigene Hardwareentwicklung durch die Firmen aufgebaut werden. Damit ist eine kostengünstige und schnelle Markteinführung gegeben.

Für die schnelle Entwicklung von Applikationen ist ein einfaches Konzept für die schnelle Programmierung als nächster Schritt notwendig. Beste Konzepte sind nahtlos in die Implementierungswerkzeuge der FPGA Hersteller integriert.

3 Ein neues Konzept der FPGA-Programmierung

Die Applikationsentwicklung sollte auf einer einfachen Programmierplattform des Development Kits durchgeführt werden, die leicht erlernbar ist. Der beste Weg dafür ist eine grafische Programmierung mit einfachen, aber auch komplexen funktionalen Modulen, die auch für kleinere Firmen und Universitäten finanzierbar und leicht zu erlernen ist.

Ein neues Konzept wird mit der Modularen Grafischen Programmierung (MGP) mit XILINX ISE (Integrated Software Environment) beschritten. Der FPGA Marktführer XILINX bietet sein Implementierungsprogramm ISE auch als Web-Pack kostenfrei an. Restriktionen bestehen dabei in nicht vorhandenen Programmteilen und in der Anzahl der maximal programmierbaren Systemgatter je nach FPGA-Familie. Für kleinere bis mittlere FPGAs ist die vollständige Implementierung möglich. Ein Programmteil der ISE ist das grafische Schaltplaneingabe-Programm ECS/Schematic Editor, in dem Schaltpläne, Zähler, Gatter- und Logikfunktionen programmiert werden können.

Die Erweiterung Modulare Grafische Programmierung (MGP) nutzt diese grafische Programmierumgebung als Designeingabe und erweitert die Funktionen um effiziente und komplexe Module (z.B. Filter, Bildverarbeitungsalgorithmen, Kompressionsblöcke, Prozessoren...). Dadurch wird ECS/Schematic zu einem einfach zu benutzenden und umfangreichen Werkzeug für die FPGA-Applikationsentwicklung, speziell durch die direkte Integration in die Implementierungsumgebung.

Wesentlichen Vorteile dieser Programmiertechnik sind:

- sehr kurze Einarbeitungszeit
- hohe Übersichtlichkeit, gute Dokumentation und schnelle Ergebnisse durch grafische FPGA-Programmierung
- Erweiterung der MGP-Funktionen mit ECS/Schematics Grundelementen, zusätzlichen grafischen Modulen, IP-Cores und HDL Code (eigene Erweiterungen möglich)
- kostenfreies Implementierungswerkzeug

Komponenten der Modularen Grafischen Programmierung (MGP) sind:

- ECS/Schematic Rahmenprogramm entsprechend der Hardwareumgebung auf der FPGA-Platine
- MGP Toolbox mit Modulen und Dokumentation der Modulfunktionen
- Software-Dokumentation des Prinzips der Modularen Grafischen Programmierung (MGP), des Implementierungswerkzeugs und des ECS/Schematic-Programms

Der Applikationsingenieur muß lediglich die Aufgabe und Idee für die Realisierung der Anwendung besitzen, danach kann er mit MGP den FPGA-Designprozeß starten (Bild 3).
.

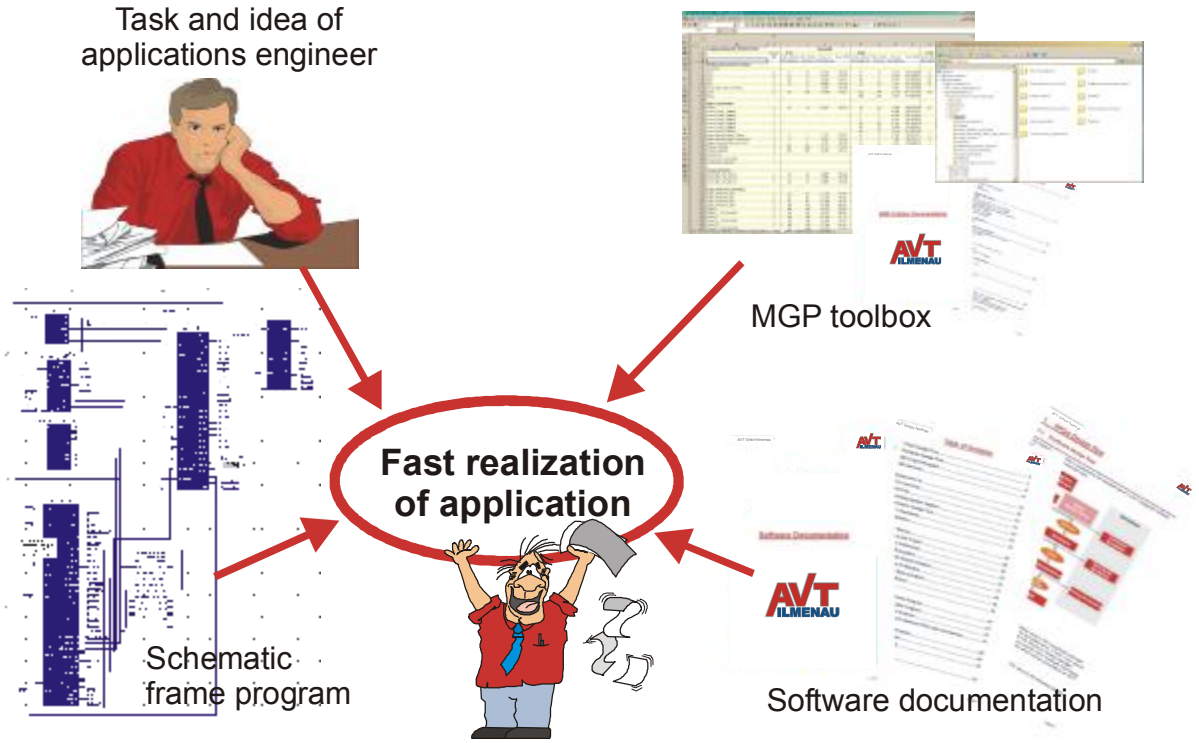

Bild 3: Prinzip und Teile der Modularen Grafischen Programmierung (MGP)

Modulare Grafische Programmierung (MGP) besteht aus sechs folgenden Schritten:

1. Schritt: Berechnung der Design Ressourcen

   Dieser Schritt besteht aus der Auswahl der notwendigen Design-Funktionen bzw. der Module für die Gesamtapplikation. Die Bitbreite und Anzahl der notwendigen Module der Anwendung werden dabei in eine Excel-Tabelle eingegeben (Bild 4).

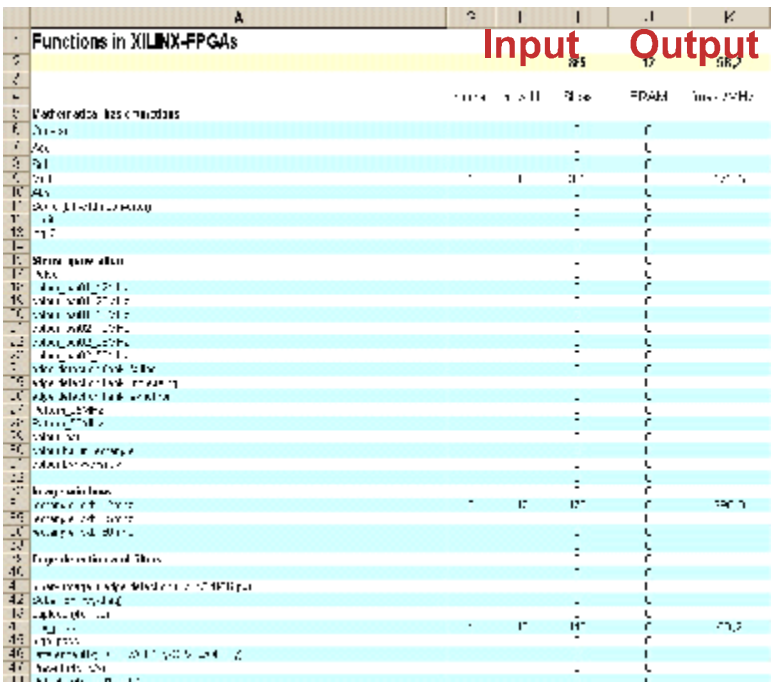

Bild 4: Berechnung der Design Ressourcen als 1. Schritt der MGP

Ausgabegrößen der Tabelle sind die Summe des Platzverbrauches (Slices) und des benötigten Speicherplatzes (BlockRAM). Diese Informationen geben die Kontrolle, ob das Design im gewählten FPGA realisiert werden kann. Werden mehr Slices oder BlockRAM verbraucht, ist ein kleineres Design oder ein größerer FPGA notwendig. Diese Entscheidung kann im ersten Schritt getroffen werden, noch bevor ein großer Teil der Arbeit geleistet wurde.

2. Schritt: Kopieren der notwendigen Moduldateien

   Die im ersten Schritt ausgewählten Dateien der Module werden aus den Toolbox-Verzeichnissen in das Arbeitsverzeichnis mit dem Rahmenprogramm für die Anwendung kopiert (Bild 5). Diese Moduldateien bestehen aus Netzlisten und grafischen Symboldateien.

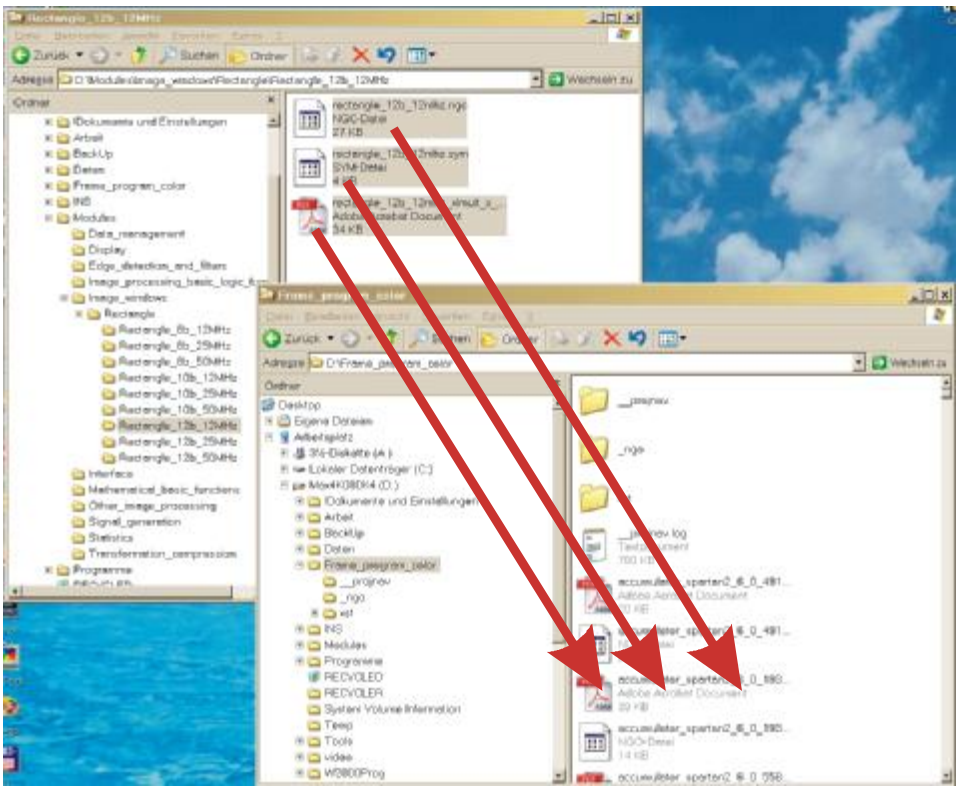

Bild 5: Kopieren der notwendigen Moduldateien als 2. Schritt der MGP

3. Schritt: Start des Rahmenprogramms im Implementierungswerkzeug XILINX ISE

   Der Project Navigator der XILINX ISE zeigt Teile des Designs und die Schritte der Implementierung (Bild 6). Die Designbeschreibung beginnt mit dem Öffnen des Rahmenprogramms in ECS/Schematic. Das Rahmenprogramm besteht aus grafischen Modulen zur Steuerung der FPGA Umgebung auf der Platine, z.B. clock_dll, pixel counter, VGA output, SRAM control, I²C control,...

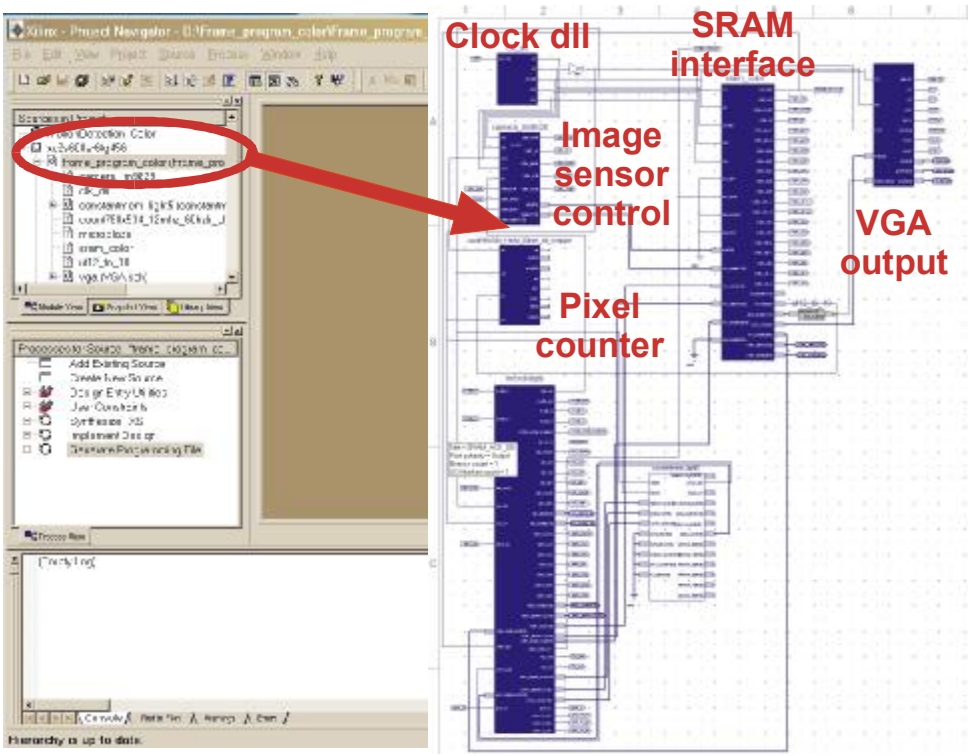

Bild 6: Start des Rahmenprogramms in der XILINX ISE als 3. Schritt der MGP

4. Schritt: Zusammensetzen der Modulsymbole zur Applikation

Die Symbole der Modulfunktionen befinden sich im Arbeitsverzeichnis. Nach der Modulauswahl können sie in die Schaltung des Rahmenprogramms eingebaut werden (Bild 7).

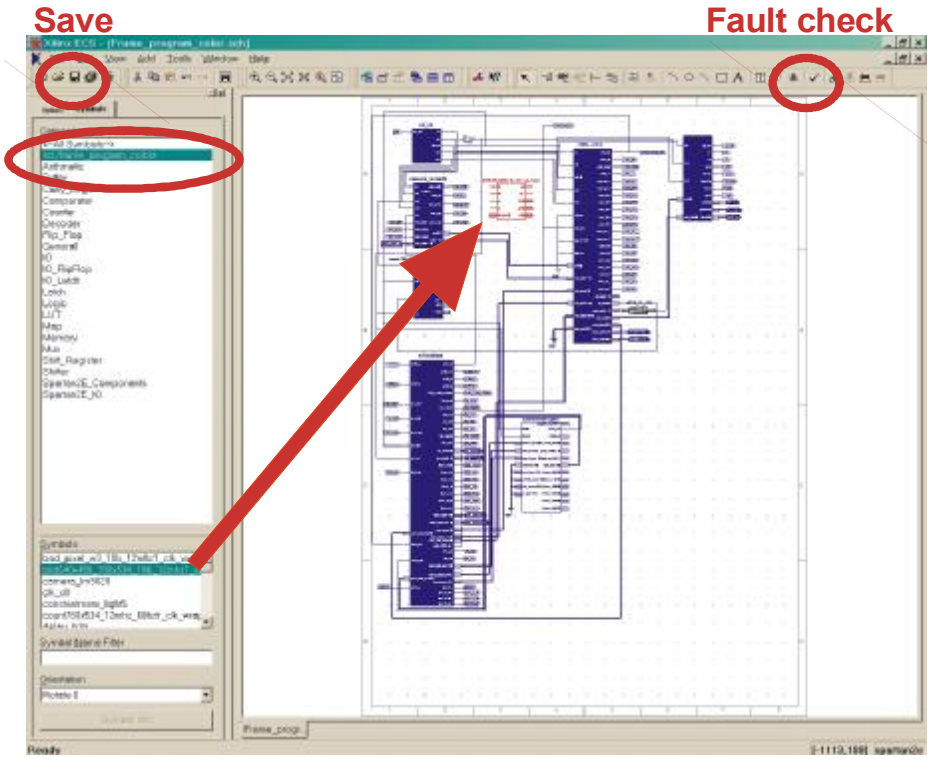

Bild 7: Einbau der Modulsymbole in die Applikationsschaltung als 4. Schritt der MGP

Die Datenverbindungen zwischen den Modulen des Rahmenprogramms können getrennt und mit den eingefügten Modulen wieder verbunden werden. Ein Beispiel ist die Nutzung von Bildverarbeitungsfiltern (bad pixel elimination, median, high pass, low pass, edge detection,...). So ist es ebenfalls möglich, mehrere Filter in Reihe oder Parallel in eine Videodatenleitung zu schalten.

Nach der Fertigstellung der Designbeschreibung wird ein Fehlercheck durch das ECS/Schematic empfohlen. Schritt 4 wird mit dem Speichern der Designbeschreibung abgeschlossen.

5. Schritt: Generierung der Programmierdatei

   Die automatische Berechnung der Programmierdatei wird durch Doppelklicken des letzten Implementierungsschrittes (“Generate programming file”) im Program Navigator gestartet (Bild 8). Alle Implementierungsschritte und die Erzeugung von Reportdateien erfolgen automatisch.

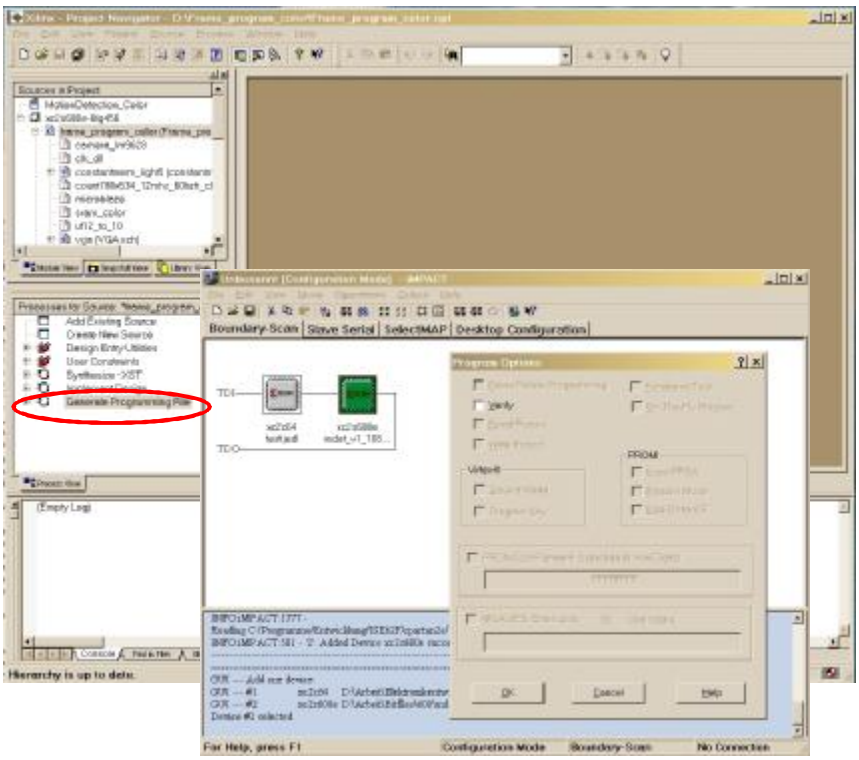

Bild 8: Generierung der Programmierdatei und Laden in den FPGA sind die Schritte 5 und 6 der MGP

6. Schritt: Programmierung des FPGAs

   XILINX ISE beinhaltet den Programmteil IMPACT, um die generierte Programmierdatei in den FPGA zu laden (Bild 8). Mit einem JTAG Programmiergerät wird die JTAG-Kette analysiert und der FPGA kann mit der Datei für die Anwendung programmiert werden.

Die MGP Toolbox besteht aus mehr als 800 Modulen, die in verschiedene Kategorien, wie mathematische Basisfunktionen, Signalgenerierung, Kantenerkennung und Filter usw. sortiert sind. Dazu gehören auch sehr komplexe Funktionen wie JPEG-Komprimierung, sowie die Nutzung des XILINX MicroBlaze 32 bit RISC Prozessors im FPGA mit Hard- und Softwaremodulen.

4 Zusammenfassung

Die Modulare Grafische Programmierung (MGP) wurde für einen einfachen und schnellen Einstieg in die FPGA -Programmierung geschaffen, die in jede Richtung erweitert werden kann. Nach dem Beginn der Arbeit mit FPGAs hilft MGP das vorhandene Wissen über die FPGA-Programmierung systematisch zu erweitern. Schnelle Ergebnisse und kostensparende Designprogrammierung stehen dabei im Vordergrund. MGP ist einfach durch weitere Module oder eigene Funktionen erweiterbar, es ist schnell erlernbar und kostengünstig.

MGP ist jedoch nicht nur für den Einstieg in die FPGA-Programmierung gedacht. Verschiedene Programmiermethoden können miteinander kombiniert werden: ECS/Schematic Module, VHDL Module, von XILINX System Generator erzeugte Module und IP Cores. Diese Möglichkeiten erschließen sich durch die nahtlose Integration in das Implementierungswerkzeug ISE.

Mit mehr als 800 Modulen, inklusive komplexer Funktionen, kann MGP eines der bedeutendsten Programmiertools für FPGAs sein. MGP ist ebenfalls für System-on-Chip Designs auf FPGAs mit Hard- und Softwarebibliotheken mit MicroBlaze Prozessor geeignet. Dabei wird die Arbeit an neueren Versionen und weiteren Funktionen ständig fortgesetzt.

Unterstützt wird dieser Einstieg durch modulare Seminare. Mit einem neuen Konzept werden ideale Bedingungen für dynamisches kosteneffektives Lernen und Einarbeiten geboten. Es stehen mehrere thematische Richtungen zur Auswahl, wobei die Applikationserstellung im Vordergrund steht.

Dr.-Ing. J. Pospiech studierte und promovierte 1991 bis 2001 an der Technischen Universität Ilmenau und arbeitet seit 2001 in der CE-SYS GmbH Ilmenau. Seit 2004 ist er Geschäftsführer der AVT GmbH.