4. Logic Cell Array (LCA) Entwicklungen für einen GPS - Empfänger

Hans Fiesel

Fachhochschule Offenburg

Im Rahmen eines GPS-Projektes ist an der Fachhochschule Offenburg ein Konzept für einen experimentellen Navigationsempfänger entstanden. Hierfür wurde der digitale Teil entwickelt und aufgebaut. Für die Realisierung der Schaltung sollten benutzerprogrammierbare Gate Arrays von Xilinx (LCAs) verwendet werden, die sich schon bei einer anderen Arbeit an der Fachhochschule bewährt hatten.
Nachfolgend möchte ich dem Leser einen Überblick über das GPS-System und die Entwicklung der LCAs geben.

Einführung

Das GPS-System ist ein satellitengestütztes Navigationssystem, das ursprünglich für militärische Anwendungen entwickelt wurde.
Nach seiner Komplettierung werden 18 aktive und 3 Reservesatelliten um die Erde kreisen. Die Umlaufbahnen sind so gewählt, daß zu jedem Zeitpunkt und an jedem Ort der Erde mindestens 4 Satelliten empfangen werden können.
Zu einer dreidimensionalen Ortsbestimmung benötigt man die Entfernung dieses Ortes zu drei Satelliten und deren Positionen. Als Grundlage für die Entfernungsmessung werden die Laufzeiten der Satellitensignale herangezogen. Diese bestimmt man, indem der Zeitpunkt des Aussendens eines Signalereignisses mit dem Empfangszeitpunkt verglichen wird. Weil die Satellitenzeit mit der Zeit des Empfängers nicht genau übereinstimmmen wird, erhält man einen Meßfehler, der dem Produkt aus der Uhrzeitdifferenz und der Lichtgeschwindigkeit entspricht. Da es zu teuer wäre auch jeden Empfänger, wie schon die Satelliten, mit einer hochgenauen Atomuhr auszurüsten, mißt man zusätzlich die Entfernung zu einem 4. Satelliten. So erhält man ein System von 4 Gleichungen, aus denen man den dreidimensionalen Ort und die genaue Zeit bestimmen kann.

Das Satellitensignal

Alle Satelliten senden ihre Daten auf den gleichen beiden Frequenzen :

$$f_1 = 1575.42 \text{ MHz}$$
$$f_2 = 1227.60 \text{ MHz}$$

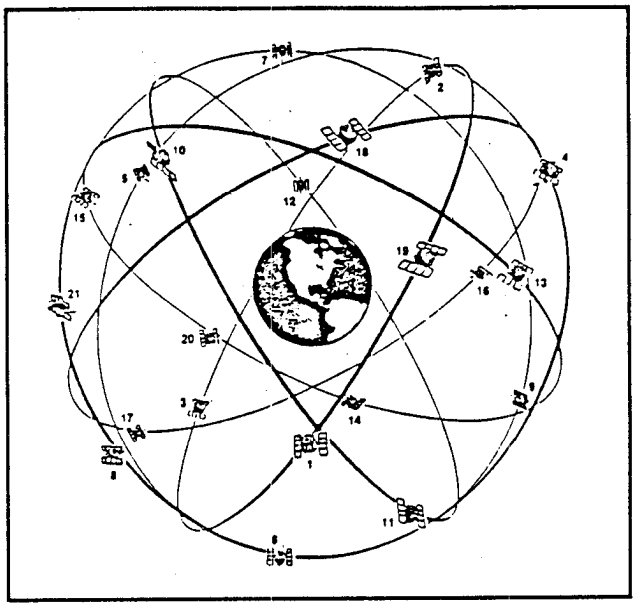

Bild 1 : Satellitenkonstellation :18 aktive, 3 Reservesatelliten; auf 6 verschiedenen Umlaufbahnen ; 20200 km über der Erde; 55° Inklination.

Zur Trennung der einzelnen Satellitensignale wird das Codemultiplex- (Spread Spectrum-) Verfahren angewandt. Die Phase des HF-Trägers wird hierbei nicht nur durch den digitalen Datenstrom moduliert, sondern auch durch eine digitale Codefolge, deren Bitfrequenz sehr viel höher ist. Dabei wird die Bandbreite des Nutzsignals auf die Bandbreite des Codes gespreizt. Das Empfangssignal muß mit dem gleichen, synchronen Code korreliert werden, um die Daten wieder auf die ursprüngliche Bandbreite zurückzusetzen.
Das GPS-System verwendet hierfür 2 verschiedene Codes; zum einen den C/A- (Coarse Aquisition) Code, der jedem Benutzer zur Verfügung steht und zum anderen den P- (Precise) Code, welcher militärischen Anwendungen vorbehalten bleibt.
Beide Codes werden mit den Daten durch eine Modulo-2-Addition (EXOR) verknüpft. Die Sendesignalerzeugung zeigt Bild 2.

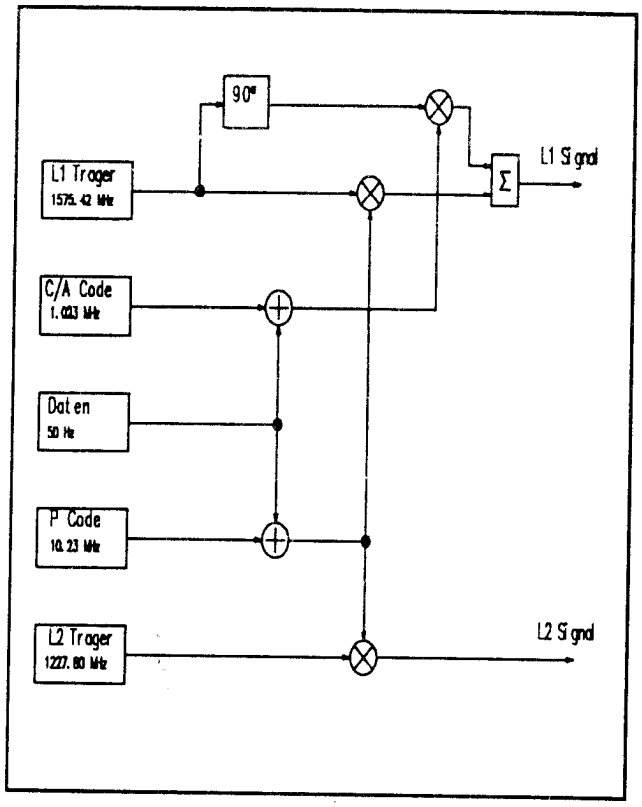

Bild 2 : Sendesignalerzeugung

Verwendete Codes

C/A Code

Der C/A-Code ist ein 1023 Bit Gold-Code, der aus der Modulo-2-Addition zweier 1023 Bit PRN Codes entsteht:

$$C/A_i(T) = G1(t) \ (+) \ G2(t + iT)$$

Die Folgen G1(t) und G2(t) werden von jeweils einem linear rückgekoppelten 10-Bit Schieberegister generiert.Die Generatorpolynome lauten:

$$G1 = X^{10} + X^3 + 1$$
$$G2 = X^{10} + X^9 + X^8 + X^6 + X^3 + X^2 + 1$$

Die Schieberegister werden mit 1,023 MHz getaktet und mit einem Synchronisierungssignal auf "alle 1" gesetzt (siehe Bild 3).
Zur Selektion benötigt jeder Satellit einen eigenen C/A-Code. Die verschiedenen C/A-Codes werden erzeugt, indem man die G2-Folge um eine ganzzahlige Anzahl von C/A-Takten verzögert. Dabei macht man sich die "Shift and Add" -Eigenschaft solcher Folgen zunutze, die besagt, daß die gleiche, verzögerte Codefolge entsteht, wenn man eine beliebige Anzahl von Parallelabgriffen des Schieberegisters miteinander EXOR-verknüpft.

Tabelle 1 zeigt die Zuordnung der Parallelabgriffe zu den entsprechenden Verzögerungen und den Codenummern.

Satelliten-nummer	Signal-nummer	G2-Tab-Selektion	G2-Code Verzögerung
1	1	2 (+) 6	5
2	2	3 (+) 7	6
3	3	4 (+) 8	7
4	4	5 (+) 9	8
5	5	1 (+) 9	17
6	6	2 (+) 10	18
7	7	1 (+) 8	139
8	8	2 (+) 9	140
9	9	3 (+) 10	141
10	10	2 (+) 3	251
11	11	3 (+) 4	252
12	12	5 (+) 6	254
13	13	6 (+) 7	255
14	14	7 (+) 8	256
15	15	8 (+) 9	257
16	16	9 (+) 10	258
17	17	1 (+) 4	469
18	18	2 (+) 5	470
19	19	3 (+) 6	471
20	20	4 (+) 7	472
21	21	5 (+) 8	473
22	22	6 (+) 9	474
23	23	1 (+) 3	509
24	24	4 (+) 6	512
25	25	5 (+) 7	513
26	26	6 (+) 8	514
27	27	7 (+) 9	515
28	28	8 (+) 10	516
29	29	1 (+) 6	859
30	30	2 (+) 7	860
31	31	3 (+) 8	861
32	32	4 (+) 9	862

Tabelle 1 : Wertetabelle der verschiedenen C/A-Codes

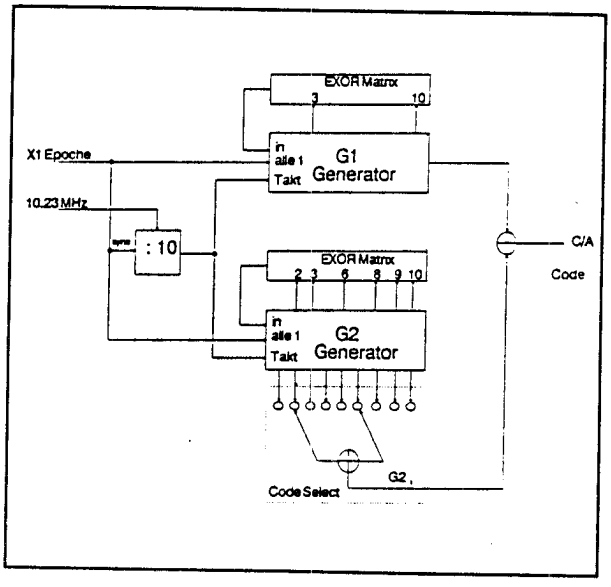

Bild 3 : C/A-Codeerzeugung

Codesynchronisation

Das bei der Übertragung der Satellitensignale angewandte Codemultiplexverfahren verlangt, daß das Empfangssignal mit einem intern erzeugten Code korreliert wird, welcher dem Signal-Code entspricht und mit diesem in Phase ist. Das empfangene Signal wird also mit einem intern erzeugten Code multipliziert. Bei dieser Verknüpfung ergibt sich ein Signalanteil (Korrelationspegel) entsprechend der Verschiebung beider Codes zueinander. Den Korrelationspegel in Abhängigkeit von der Phasendifferenz der Codes gibt die Autokorrelationsfunktion an. Die angenäherte Autokorrelationskennlinie des C/A-Codes zeigt Bild 4 und 5.

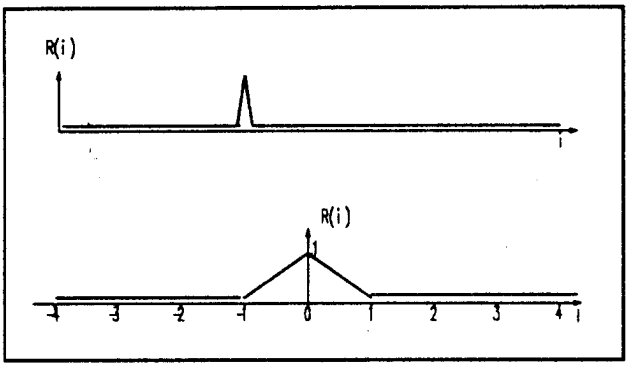

Bild 4 und 5 : Über die gesamte Codelänge ergibt sich nur bei der Phasendifferenz null eine Korrelationsspitze.

Zur Codesynchronisation wird der Empfängercode um +/- eine halbe Chiplänge verschobenen (early/late). Die jeweils entstehenden Korrelationssignale besitzen einen Anteil, welcher in einem engen Bereich der Codeverschiebung proportional ist (vgl. Autokorrelationskennlinie). Sind diese Anteile beider Signale gleich groß und größer als null, so ist der lokale Code mit dem Satellitencode in Phase.
In einem ersten Durchgang wird die Empfängercodephase in groben Schritten solange variiert bis ein Korrelatiossignal einen von Null verschiedenen Anteil besitzt (Aquisition). Danach wird die Differenz zwischen beiden Signalen auf null geregelt. Dafür wird bei dem Empfängerkonzept der FH Offenburg eine sogenannte Tau-Dither-Loop verwendet.

Tau Dither Loop

Durch die zyklische Verschiebung des Codes (Dither) wird sich, sofern die Codephasendifferenz des Empfangscodes zu dem intern generierten Code kleiner gleich einer Taktlänge ist, ein der Autokorrelationskennlinie entsprechender Korrelationspegel einstellen (siehe Bild 7). Dieser Pegel wird nach dem Pegeldetektor PD entsprechend der Codephase mit +/- 1 bzw 0 bewertet. Ist der Mittelwert des Ausgangssignals dieser Bewertung gleich 0, dann sind die Pegel der vor- und der nacheilenden Codephase gleich groß und somit ist der Code in Phase.

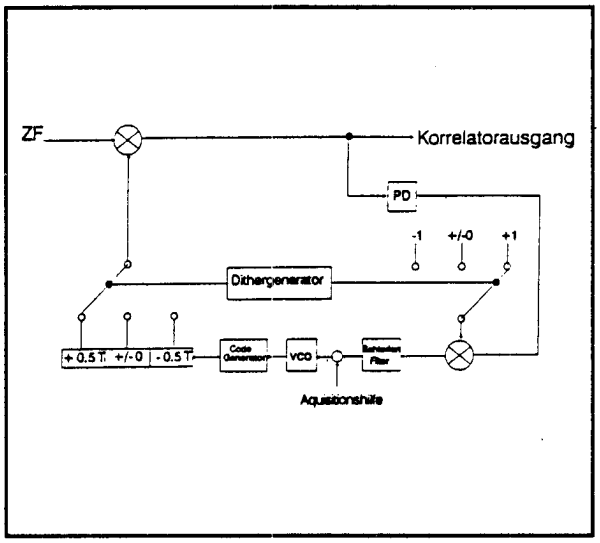

Bild 6 : Prinzipschaltbild Tau-Dither-Loop

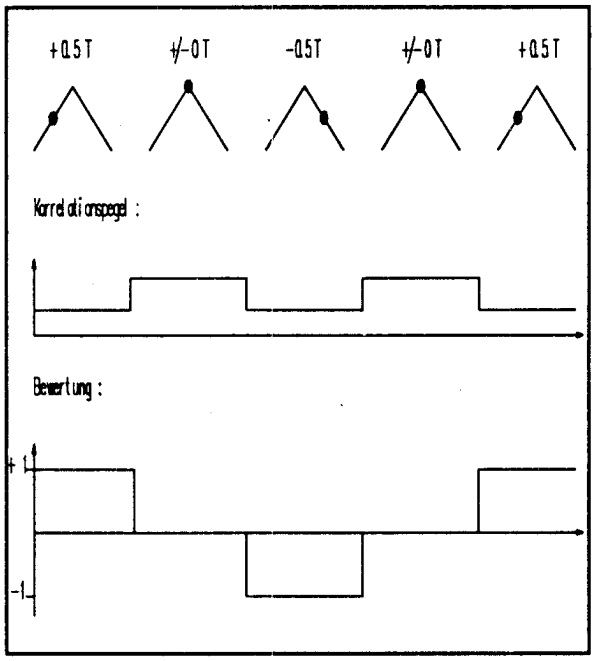

Bild 7 : Signale bei der Tau Dither Loop

Das gesamte Empfängerkonzept

Das Empfangssignal wird zuerst bei der Antenne verstärkt, um die Leitungsdämpfung zu kompensieren. Nach der Umsetzung in die Zwischenfrequenz von 40,92 MHz wird das Signal mit dem Code korreliert und durch den VCXO auf eine 2. Zwischenfrequenz von 5,115 MHz umgesetzt. Eine Trägernachführungsschleife regelt dabei durch den spannungsgesteuerten Quarzoszillator die Dopplerverschiebung des Signals aus.

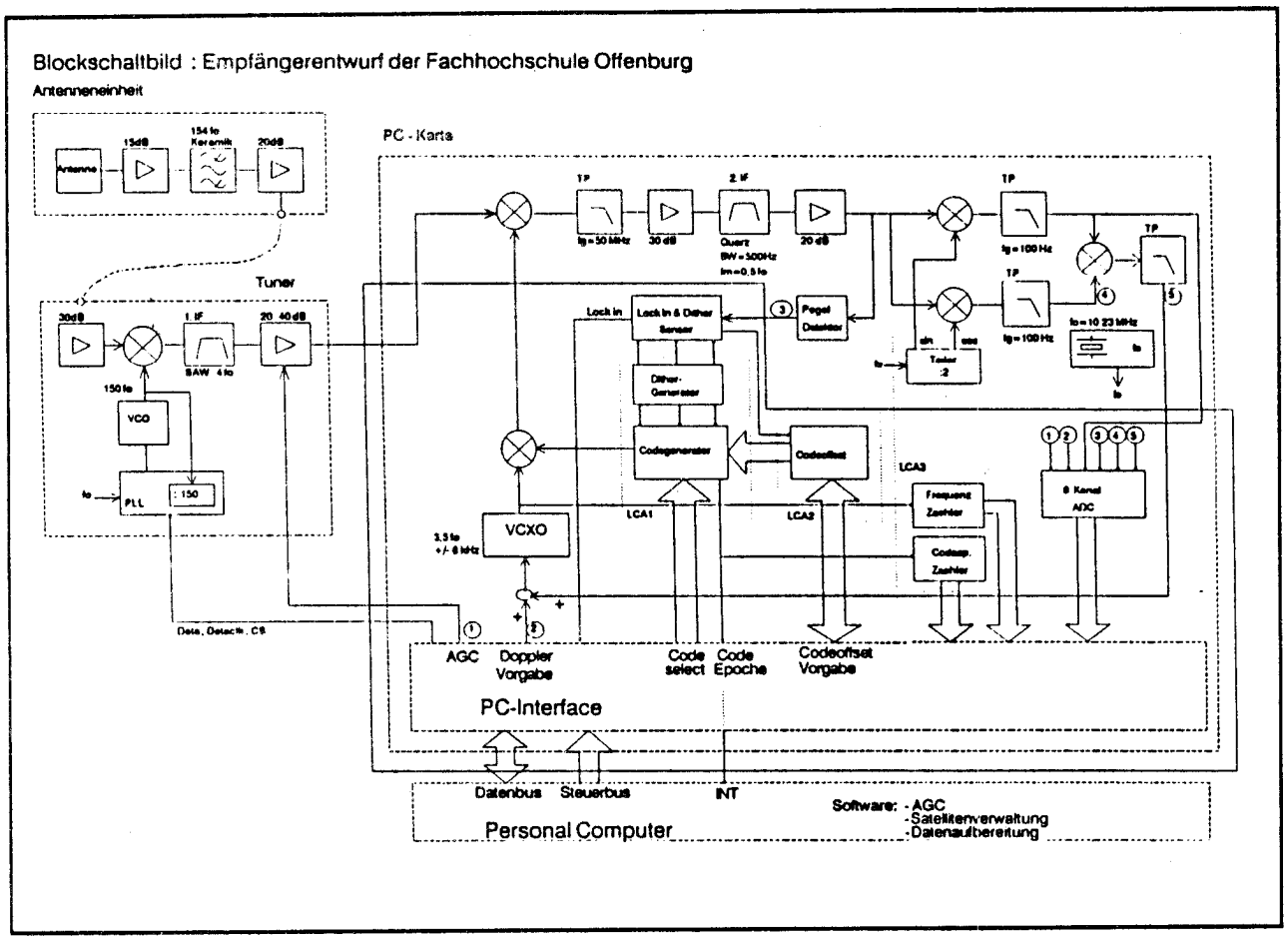

Bild 8 : Blockschaltbild des Empfängerkonzepts

Die Codekorrelation erfolgt im Zwischenfrequenzbereich. Dabei ist die C/A-Code-Nummer vom PC aus einstellbar. Die Codephase kann einerseits auch durch den PC vorgewählt und anderseits durch die Tau-Dither-Loop variiert werden.
Des weiteren besitzt der Entwurf einen Frequenzzähler zur Bestimmung der Dopplerverschiebung und einen Codeepochezähler, der als Zeitmaßstab verwendet werden soll.
Der Digitalteil dieses Empfängers ist mit Ausnahme des PC-Interfaces mit Hilfe von benutzerprogrammierbaren Gate Arrays verwirklicht worden. Insgesamt drei Bausteine enthalten den Codegenerator, einen Dithergenerator, einen 14-Bit Auf-/Abwärtszähler mit mehreren Steuerungsautomaten, welche für die Codeverschiebung verantwortlich sind, einen 16-Bit Frequenzzähler und einen 10-Bit Codeepochenzähler.

Benutzerprogrammierbare Gate Arrays: Logic-Cell-Arrays (LCAs) von Xilinx

Xilinx bietet seit 1985 als erster Hersteller benutzerprogrammierbare Gate Arrays an, die es ermöglichen komplexe Logikentwürfe schnell und leicht modifizierbar in preisgünstigen Bausteinen unterzubringen. Es stehen mehrere LCA-Familien zur Auswahl, die sich in der Anzahl der internen Verbindungsmöglichkeiten und integrierbarer Logik unterscheiden. Die größten Bausteine besitzen eine Kapazität von 9000 Gattern. Bei allen Serien kann man zwischen drei Geschwindigkeitsstufen wählen : 50 MHz, 70 MHz und 100 MHz maximale Flip-Flop Taktrate. Diese Angaben beziehen sich aber auf nur eine FF-Durchgangs-

zeit. Daher werden die tatsächlichen Taktfrequenzen zwischen 5 und 20% dieser Werte liegen.

Architektur der LCAs

LCAs bestehen aus drei verschiedenen Elementen:
- Logik-Blöcke
- Ein/Ausgabe-Blöcke
- Verbindungen

Eine zentrale Matrix aus einheitlichen Logik-Blöcken (Configurable Logic Blocks) wird von Ein/Ausgabe-Blöcken umgeben; dazwischen liegt ein Netz aus verschiedenartigen Verbindungsmöglichkeiten (siehe Bild 9). Die vom Anwender programmmierbare Logikschaltung wird durch die CLBs erzeugt, wobei die E/A-Blöcke die Schnittstelle zur Bausteinperipherie darstellen.

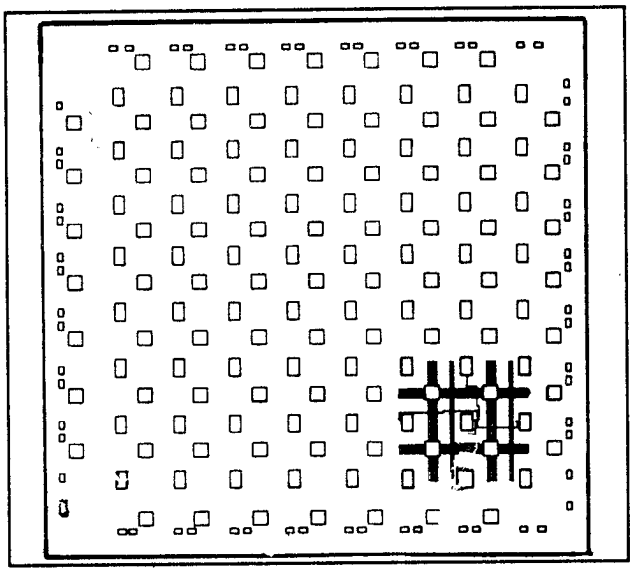

Bild 9 : Innerer Aufbau eines LCA

Alle LCA-Elemente und Funktionen werden durch ein Konfigurationsprogramm bestimmt, wodurch erst die eigentliche Anwenderschaltung entsteht. Das Programm wird während des Betriebes in einem On-Chip-Memory gespeichert und kann auf unterschiedliche Weise automatisch vom Baustein selbst aus einem externen PROM, oder gesteuert durch die Peripherie, geladen werden.
Zur Erzeugung dieses Programms bietet Xilinx ein komfortables Entwicklungssystem an.

Das LCA-Entwicklungs-System von Xilinx

Das System besteht aus einem Programmpaket mit einer komfortablen Benutzer-oberfläche - dem Xilinx Design

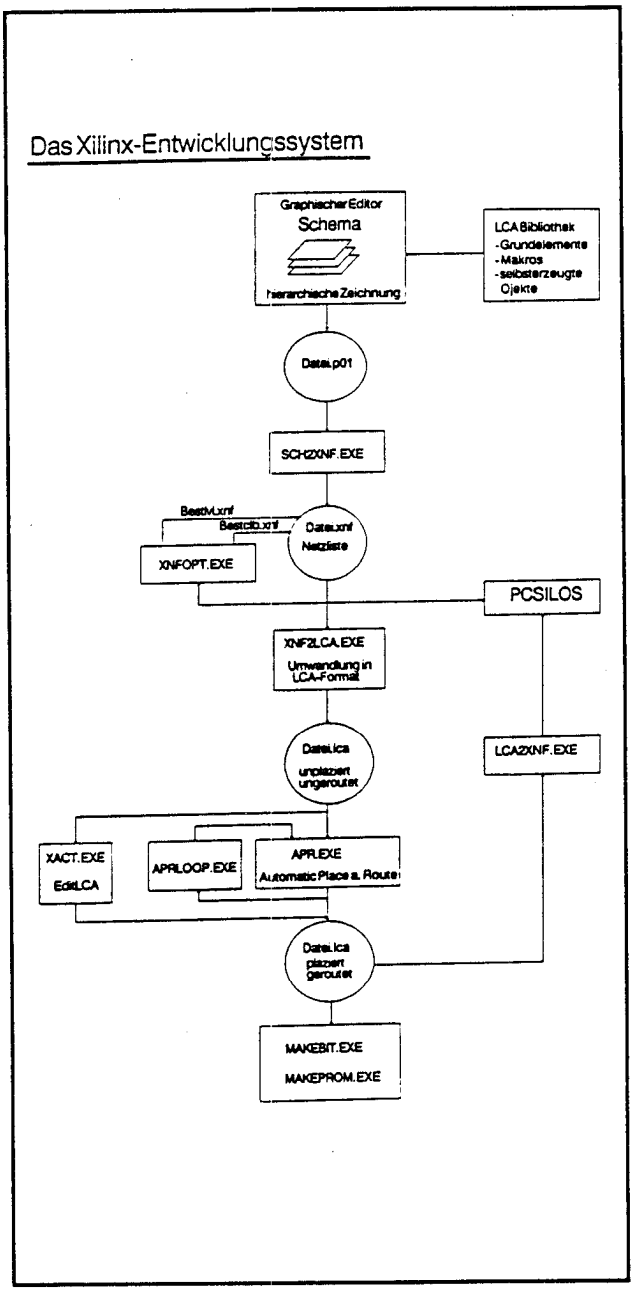

Bild 10 : Entwicklungssystem

Manager. Aus diesem lassen sich über Pull-Up-Menüs alle im Entwicklungsdurchlauf benötigten Hilfsprogramme mit den entsprechenden Optionen und Dateizusätzen aufrufen. Nachfolgend sind diejenigen Entwicklungswerkzeuge des Systems beschrieben, die ich selbst im Rahmen meiner Diplomarbeit benutzt habe. Das Softwarepaket besitzt im Bereich der Designeingabe einige weitere Optionen.
Erster Schritt der Designerzeugung ist die Erstellung eines Schaltplans mit Hilfe eines graphischen Editors ; beim System der Fachhochschule Offenburg ist dies Schema II

plus, aber das Xilinx-Programmpaket besitzt auch Schnittstellen zu anderen Editoren. Beim Zeichnen dieses Plans dürfen nur Symbole einer LCA-Bibliothek verwendet werden. Diese enthält sowohl Grundelemente wie zum Beispiel : Gatter , Puffer oder Flip-Flops als auch Makros (Zähler, Dekoder, Register...). Weiterhin besteht die Möglichkeit eigene Symbole zu kreieren und deren "Innenleben" in anderen untergeordneten Zeichnungen zu definieren. So entsteht ein hierarchischer Entwurf aus selbsterzeugten Symbolen und den dazugehörigen Zeichnungen, die in einer ersten Datei mit der Erweiterung .p01 abgelegt werden und von einem Konvertierungsprogramm (SCH2XNF.EXE) in eine Xilinx-Netzliste umgewandelt werden.
Diese Netzliste kann einerseits zur Simulation durch ein weiteres Hilfsprogramm in PC-Silos-Format umgewandelt werden und andererseits durch ein Optimierungsprogramm auf Geschwindigkeit bzw. Blockaufwand optimiert werden.
War die Simulation erfolgreich, kann die Netzliste in das LCA-Format übersetzt werden. Dabei werden die im Schaltbild verwendeten Symbole in LCA-Blöcke umgewandelt. Diese Blöcke müssen dann im LCA plaziert und dort elektrisch verbunden werden. Dazu steht neben einem LCA-Graghik-Editor, ein Programm zur Verfügung, welches diese Aufgabe übernimmt : APR (AUTOMATIC PLACE AND ROUTE).
Das Ergebnis dieses Programms hängt wesentlich von einer am Bearbeitungsanfang gewählten Zufallszahl ab, durch die die Anfangsplazierung bestimmt wird. Startet man APR mehrmals mit dem gleichen Entwurf, so wird man am Ende trotzdem unterschiedlich gute Designs erhalten. APRLOOP bietet daher die Möglichkeit automatisch den gleichen Entwurf mehrmals von APR plazieren und routen zu lassen, um sich hinterher das beste Ergebnis heraussuchen zu können. Wenn das Verbinden der Blöcke erfolgreich war, steht am Ende dieses Durchgangs eine Datei mit der Erweiterung .lca , welche durch zwei weitere Hilfsprogramme in ein PROM programmiert werden kann. Diese Datei kann ebenfalls in PC-Silos-Format umgewandelt werden, um in die Simulation auch die Signallaufzeiten im LCA miteinberechnen zu lassen.

Bild 10 zeigt den gesamten Entwicklungsablauf auf einen Blick.

Erfahrungen mit LCAs und deren Entwicklungssystem

Bei meiner Diplomarbeit habe ich innerhalb dreier Monate zwei LCAs mit je 2000 Gattern und eines mit 3000 Gattern entwickelt und hardwaremäßig realisiert. Dabei hat die Einarbeitung in das System und das Austesten der Effektivität der einzelnen Optionen den größten Teil dieser Zeit in Anspruch genommen.
Alle drei Designs wurden von APR automatisch plaziert und geroutet. Dabei konnte ich allerdings einige Schwachpunkte erkennen:

1. Netze mit einer großen Anzahl von Anschlußpins führen zu hohen Laufzeitverzögerungen von 100ns und mehr. Auch wenn man diese Netze als "Critical" definiert, ergeben sich keine wesentlichen Verbesserungen.

2. Das Verhältnis von Logik-Blöcken und Verbindungsleitungen ist bei dem kleineren LCA (XC3020) günstig gewählt. Bei größeren Bausteinen, die zwangsläufig mehr Logik aufnehmen müssen, findet man aber die gleichen Verdrahtungsmöglichkeiten vor. Auf ein größeres LCA umzusteigen, um das Routen eines Designs zu erleichtern, hat deshalb keinen entscheidenden Vorteil gebracht. Aus diesem Grund wird auch das Routen eines Entwurfs, der fast sämtliche Logik-Blöcke des jeweiligen LCAs ausnützt, mit zunehmender Bausteingröße immer schwieriger.

3. Gibt man die Anschlußbelegung vor, erhält man wesentlich größere Durchlaufzeiten.

Mittlerweile hat Xilinx eine verbesserte und erweiterte Version des Autorouter-Pakets (ADI 3.0) ausgeliefert. Den schon vom Hersteller beschriebenen enormen Fortschritt gegenüber der älteren Version kann ich bestätigen. Ich habe diese neuen Programme auf ein Design angewendet, welches von dem alten System nur in 5% aller Versuche erfolgreich geroutet werden konnte. ADI 3.0 konnte diesen Wert auf 50% steigern. Dabei ergaben sich bei den größeren Netzen bis zu 5mal kürzere Verzögerungszeiten.
Mit dem Optimierungsprogramm XNFOPT habe ich keine positiven Erfahrungen gemacht. Die optimierten Entwürfe benötigten prinzipiell mehr Blöcke. Auch einen Geschwindigkeitsvorteil dieser Designs war nicht zu erkennen, da diese wesentlich mehr Verbindungsleitungen benötigten und dadurch die Signalverzögerungen anstiegen.
Die Schnittstelle zu PC-Silos liefert eine komfortable Möglichkeit, den Entwurf mit allen Durchlaufzeiten zu simulieren. Die Simulation hat sich als sehr exakt erwiesen, so daß die Umsetzung des Entwurfs in die Hardware nach erfolgreicher Simulation völlig unkritisch ist.

Literaturverzeichnis

[1] Kromer, D., Landis, J., Military Agency for Standardization, Navstar Global Positioning System (GPS), System Characteristics 1987

[2] Mäusl, R., Digitale Modulationsverfahren Hüthig Verlag 2.Auflage 1988

[3] Meinke Gundlach, Taschenbuch der Hochfrequenztechnik Band 3 Springer Verlag 4.Auflage

1986

[4] Navstar GPS Joint Program Office, Los Angeles Air Force Station, US Air Force Space Division/ CWNI, Introduction To Navstar GPS User Equipment 1988

[5] Spilker, J. J., Global Positioning System : Signal Structure and Performance Characteristics, Stanford Telecommunications Inc., 1979

[6] Xilinx, The Programmable Gate Array Data Book 1989