Entwurf eines Logic-Cell Arrays (LCA) für eine Lichtwellenleiterübertragungsstrecke von einer CCD - Kamera zu einem Personal-Computer

Bernd Reinke
Fachhochschule Offenburg

Einführung

Die Elektronikindustrie bietet für die Realisierung digitaler Logik eine Vielzahl integrierter Bausteine an, die ein Höchstmaß an Zuverlässigkeit als auch Integrationsdichte ermöglichen.

Je nach Integrationsdichte unterscheidet man hierbei zwischen Standardlogik (TTL,CMOS,DTL...), programmierbarer Logik (PLA, GAL...), Gate Arrays und ASIC-Bausteinen. Mit steigender Integrationsdichte werden Systemeigenschaften verbessert, wie Leistungsverbrauch, Platzbedarf, und Zuverlässigkeit.

Jedoch steht ihr auch ein stark erhöhter Kosten- und Entwicklungsaufwand gegenüber, der den Einsatz hochintegrierter Bausteine in Einzelfertigung bzw. Kleinserien verhindert.

Xilinx bietet nun mit seiner LCA-Produktreihe (logic cell array) eine Alternative zu bestehender hochintegrierbarer Logik an, mit der es möglich sein soll, Vorteile der genannten Einzelproduktgruppen zu übernehmen, und deren Nachteile zu beseitigen.

Im Rahmen einer Diplomarbeit wurde ein solcher LCA-Baustein (XC3020) eingesetzt. Anhand der gegebenen konkreten Anwendung konnte hierbei untersucht werden, wie schnell sich ein solcher Baustein in bestehende Hardware eingliedern läßt, und welche Integrationsdichte er ermöglicht.

Im Folgenden sollen nun als Schwerpunkte das Einsatzgebiet, die Entwicklung und die Simulation des LCA bei vorliegender Aufgabenstellung aufgezeigt werden.

An dieser Stelle möchte ich mich für die Unterstützung von Herrn Prof. Dr. Dirk Jansen bedanken, ohne den die Bearbeitung einer solch umfangreichen Aufgabe nicht möglich gewesen wäre.

Aufgabenstellung

Im Bereich industrieller Meßtechnik besteht ein Bedarf nach einer Linear-CCD-Kamera, die abgesetzt von einem Auswerterechner betrieben werden soll.

Um den hohen Anforderungen an Übertragungssicherheit und Datenrate gerecht zu werden, ist eine digitale Lichtwellenleiter-Übertragungsstrecke zu entwickeln.

Zusätzlich sollen beim Aufbau der Kameraelektronik hochintegrierte programmierbare Logikbausteine eingesetzt werden. Dies ist erforderlich, un den Platzbedarf und den Leistungsverbrauch zu reduzieren.

Die ausgesendeten Daten sollen in einer Empfangselektronik (PC-Einsteckplatine) weiterverarbeitet werden.

Kameraelektronik:	Empfangselektronik:
SMD-Bauteile	programmierbare Logik
programmierbare Logik	konventionelle Bauteile
(LCA)	(kleine PC-Einsteckkarte)

Bild 1: Komponenten der Übertragungsstrecke

Wie leicht zu verstehen ist, ist der Aufbau der Empfangselektronik vom Platzbedarf unkritisch. Die PC-Einsteckplatine bietet genügend Raum für konventionelle Elektronikbauteile.

Um daher das hier angesprochene Thema nicht unnötig auszuweiten, wird im Folgenden nur die Kamera-

bzw. Senderelektronik behandelt.

Anforderungen an die Kameraelektronik

Zur Aufbereitung und Übertragung der Bild-Daten müssen folgende elektronische Vorkehrungen getroffen werden:

* Analoge Aufbereitung der Sensordaten

* Umwandlung der Daten in ein digitales Signal (gefordert sind acht Bit; entsprechend 256 Graustufen)

* Erzeugung eines Rahmens für die Datenübertragung
  - BOS (begin of string)
  - EOS (end of string)

* Umwandlung der digitalen Signale in einen übertragungsfähigen Code.

* Erzeugung eines CRC (cyclic redundancy check) zur Erkennung von Datenübertragungsfehlern
  - BCS (block check sequence)

* Übertragung einer vorgegebenen Bitfolge zwischen den Datenübertragungen
  - IDLE (Ruhe- bzw. Belichtungszustand)

* NRZ-Code (modifiziert durch einen 4Bit/5Bit-Code)

Aufbau der Kameraelektronik

Datenstrecke: Analogteil

Zunächst ist die Datenstrecke vom CCD-Sensor bis zur LWL-Sendediode zu betrachten:

der CCD-Sensor besteht aus einer Anordnung von 3684 lichtempfindlichen Photodioden (angeordnet in einer Zeile), die bei Lichteinfall auf den Sensor ein elektrisches Potential erzeugen. Diese Potentiale können nach einer beliebig einstellbaren Belichtungszeit seriell ausgelesen werden (CCD-Prinzip). Die Ausgangs- bzw. Bildsignale liegen dabei als Differenzsignale vor, wobei die Intensität eines belichteten Bildpunktes (Pixel) durch die Größe des Differenzsignals gegeben ist.

Zur Verarbeitung dieser Differenzsignale wird eine einfache Differenzverstärkerschaltung mittels Video-Operationsverstärker eingesetzt.

Für die spannungsmäßige Anpassung dieses Signals an die Eingangsstufe eines AD-Wandlers wird eine passive Klemmung des Signals durchgeführt.

Datenstrecke: AD-Wandler

Die anschließende Digitalisierung des analogen Signals erfolgt durch einen AD-Wandler, der nach dem Parallelverfahren arbeitet (Flash-Typ). Der Vorteil gegenüber anderen Wandlerverfahren liegt hier be-

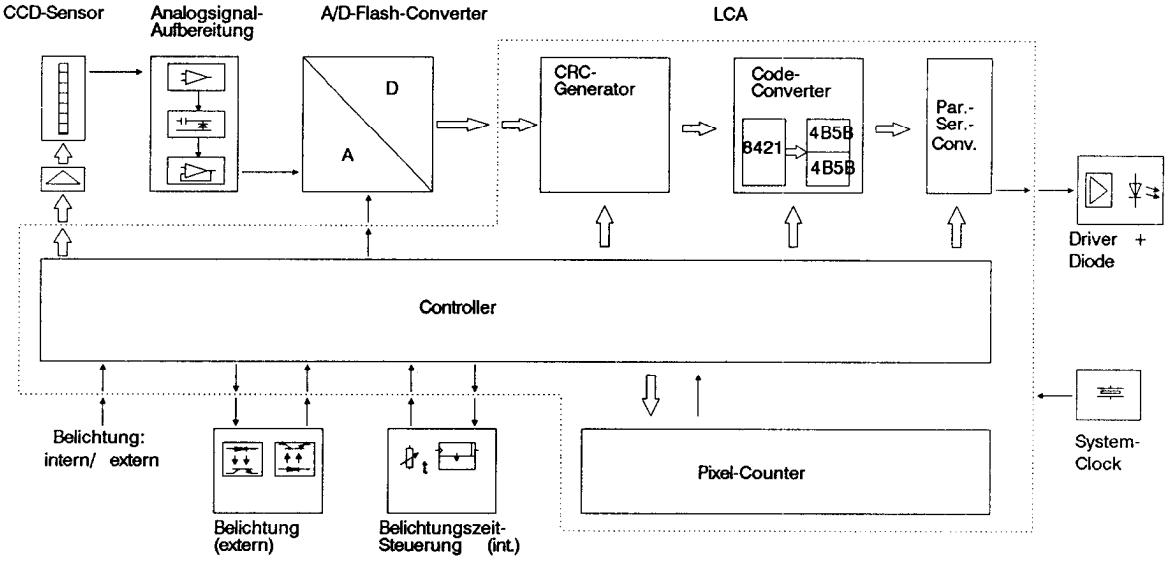

Bild 2: Blockschaltbild Kamera

sonders in der Wandelgeschwindigkeit (Bei unserer Anwendung erfolgt die Abtastung des Analogsignal mit ca. 1MHz).

Ab hier steht nun jeder Bildpunkt (Pixel) als digitales acht Bit breites Datenwort zur Verfügung.

Für die serielle Übertragung über Lichtwellenleiter muß noch eine digitale Aufbereitung erfolgen.

Datenstrecke: 4B/5B-Code

Bei Datenübertragungen mittels Aderleitungen ist es sehr einfach, den Sendetakt der zu übertragenden Daten anhand einer zweiten Leitung mitzuliefern. Dies ist bei einer LWL-Verbindung nicht möglich. Das Taktsignal muß also aus der übertragenen Bitsequenz zurückgewonnen werden (Die unterschiedlichen Methoden der Taktrückgewinnung sollen hier nicht angesprochen werden. Hierfür sei auf einschlägige Literatur verwiesen).

Bei der hier vorliegenden Anordnung wird folgendes Verfahren eingesetzt:

zum Verständnis stelle man sich einen im Empfänger angeordneten Schwinkreis hoher Güte vor, der durch einen Impuls angeregt wird. Bevor die Amplitude der Schwingung einen vorgegebenen Wert unterschreitet, soll ein neuer Impuls den Schwingkreis wieder anregen. Wenn man nun diese Impulsfolge durch den Datenstrom erzeugt (z.B. bei jedem Übergang von logisch null nach eins und umgekehrt) so muß man nur gewährleisten können, daß dieser Impuls wieder nach einer maximalen vorgegebenen Zeit auftritt. Wenn zusätzlich die Mittenfrequenz des Schwingkreises der des Sendetaktes entspricht, so kann nun aus dem Oscillatorsignal des Schwingkreises die Phase und die Frequenz des Sendetaktes zurückgewonnen werden. Damit läßt sich dann das empfangene Signal zeitrichtig abtasten.

8421	4b5b
0	00101
1	00110
2	01001
3	01010
4	01011
5	01100
6	01101
7	01110
8	10001
9	10010
a	10011
b	10100
c	10101
d	10110
e	11001
f	11010
flag	11000
sync	10101
nsync	01010

Bild 3: 4B/5B-Kode

Für dieses Verfahren ist es notwendig, den 8421-Code des AD-Wandlers in einen Code überzuführen, der die obengenannte Forderung erfüllt:

dabei werden die acht Bit des AD-Wandlers in zwei Nibble unterteilt, und jeweils in entsprechende fünf Bit gewandelt. Der Code ist hierbei so angelegt, daß beim Aneinanderreihen von mehreren fünf-Bit-Sequenzen maximal vier Nullen bzw. Einsen in Folge auftreten. Es muß im Empfänger also gewährleistet sein, daß der Schwingkreis seine Amplitude über maximal vier Impulspausen beibehält, bevor er wieder angeregt wird.

Eine weitere Eigenschaft dieses Codes besteht darin, daß in einem Datenstrom ein ausgeglichenes Verhältnis zwischen logisch null bzw. eins besteht (statistisch ca. 50%).

Datenstrecke: CRC-Generator

Bevor jedoch die Codewandlung erfolgt, wird jedes Datenbyte durch einen CRC-Generator hindurchgeschleust. Er erzeugt aus den Daten einer Bildzeile (3684 Bildpunkte) eine acht-Bit-Sequenz, die später zur Fehlererkennung herangezogen werden kann. Prinzipiell findet in ihm eine polynome Teilung der acht-Bit-Folge mittels eines einstellbaren Generatorpolynoms statt. Dabei entspricht die Datenfolge dem Dividend; das Generatorpolynom ist der Divisor. Durch die Teilung entsteht ein Quotient mit Rest, wobei nur der Rest (block check sequence) für die Fehlererkennung herangezogen wird. Der Quotient wird vernachlässigt.

Datenstrecke: Shift-Register

Die Anwendung des NRZ-Codes zur Datenübertragung erlaubt minimalen Schaltungsaufwand. So müssen die parallel anliegenden Daten nach der Code-Wandlung nur noch über ein Shift-Register (parallel ein-seriell aus) in ein serielles Format umgewandelt werden.

Controller

Kontinuierliche und damit synchrone Übertragung ergibt sich dann, wenn nach jedem vollständig ausgelesenen Datenwort das nächste Datenwort zur Übertragung bereitsteht.
Die Steuerung, die diese Aufgabe übernimmt, läßt sich mittels Zustandsdiagramm beschreiben:

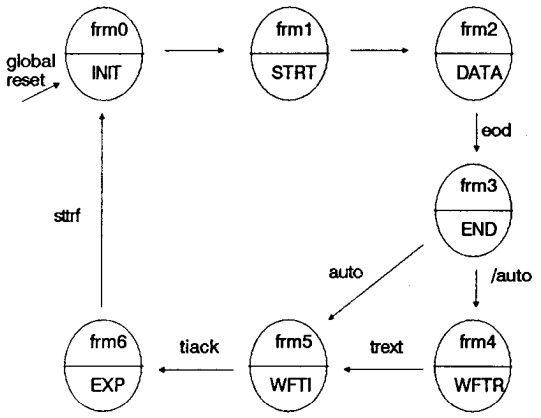

Bild 4: Zustandsdiagramm der Kamerasteuerung

Während des Zustands frm0 (frame) befindet sich die Kamera in der Initialisierungsphase (INIT). Das bedeutet, daß die Datenstrecke auf die eigentliche Datenübertragung vorbereitet wird. Beim Übergang nach frm1 wird eine Startsequenz (STRT) gesendet, die dem Empfänger den Beginn einer Datenzeile meldet. Nun kann die Datenübertragung der einzelnen Bilddaten beginnen (DATA). Sobald die Datenübertragung beendet ist, geht der Controller in den Endezustand (END) über, und gibt damit die block check sequence (BCS) frei. Das Ende der Datenübertragung bildet die Endesequenz.

Je nach Einstellung kann nun zwischen automatischer Belichtung oder extern gesteuerter Belichtung gewählt werden. Der Zustand frm5 (wait for timer) ist nur ein Übergangszustand, der dann verlassen wird, wenn das angeschlossene Monoflop (Belichtungzeitgeber) in den instabilen Zustand übergegangen ist. Frm6 ist der Zustand des Controllers während der Belichtungsphase (EXPosure). Er wird dann verlassen, wenn das Monoflop (Belichtungszeitgeber) in den stabilen Zustand übergegangen ist. Ab nun beginnt wieder die Übertragung einer belichteten Bildzeile.

Es ist noch zu erwähnen, daß sich die LWL-Strecke während den Zuständen frm0,4,5,6 in einem Quasi-Ruhezustand befindet, bei dem IDLE-Sequenzen gesendet werden.

Datenverwaltung/Belichtungssteuerung

Der im Blockschaltbild angedeutete Pixelcounter beschreibt die Datenmenge einer Bildzeile (hier 3684 Bildpunkte). Er besteht schaltungstechnisch aus einem synchronen mehrstufigen Zähler.

Die Belichtungszeitsteuerung übernimmt, wie schon beim Controller erwähnt ein Monoflop. Im angeregten Zustand des Monoflops findet Belichtung statt, im stabilen Zustand Übertragung.

Nachdem die Funktion der Kamera anhand des Block-

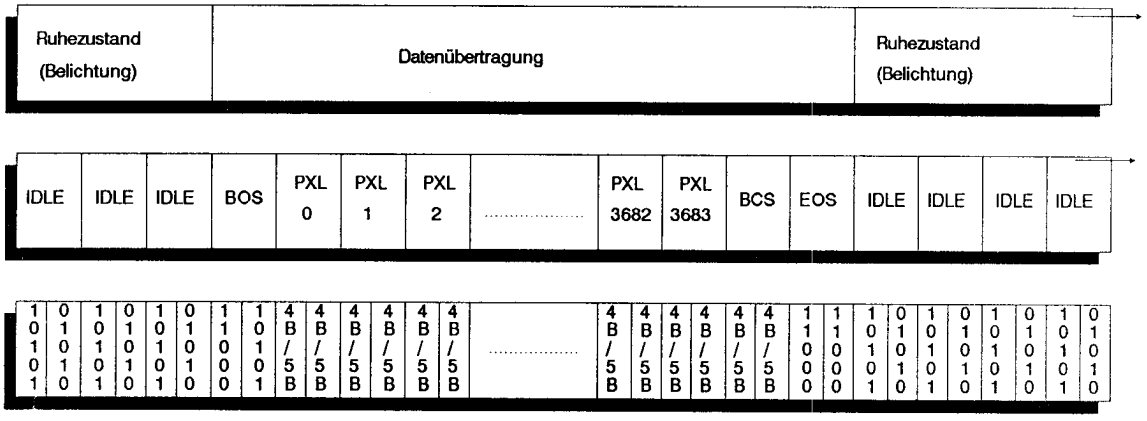

Bild 5: Datenübertragungsformat

schaltbilds voll erfaßt ist, kann nun zur schaltungstechnischen Umsetzung übergegangen werden.

Schaltungstechnische Realisierung

Hier stellt sich zu allererst die Frage, welche Baugruppen sich ohne Schwierigkeiten integrieren lassen.
Leider ist es bis zum heutigen Zeitpunkt noch nicht möglich, mit überschaubarem Entwicklungsaufwand analoge und digitale Elektronik in einem Chip zu integrieren. Dies gilt besonders in unserer Anwendung, bei der nur kleine Stückzahlen benötigt werden. Es ist deshalb sinnvoll, den Analogteil in SMD-Technik aufzubauen, zumal diese Bauteile in genügender Auswahl auf dem Elektronikmarkt vorhanden sind.

Auch die Treiber für den CCD-Sensor bzw. für die LWL-Diode müssen getrennt von der restlichen digitalen Elektronik aufgebaut werden. Dies ergibt sich aus leistungsmäßigen Gesichtspunkten.

Das Monoflop für die Belichtungszeitsteuerung ist als Universalbaustein (SMD) erhältlich. Es wäre auch nicht im Sinne einer späteren Erweiterung, die Belichtungszeitsteuerung in den Digitalteil einzugliedern.

Somit verbleibt die reine digitale Elektronik (CRC-Generator, Code-Wandler, Shift-Register, Pixelcounter, Controller und Systemtaktgenerator), die sich in einem entsprechenden Baustein integrieren läßt.

Wie zu Anfang erwähnt, wird hierfür ein LCA-Baustein vom Typ XC3020 (Xilinx) eingesetzt.

Entwicklung des LCA

Die Produktpalette der LCA-Bausteine von Xilinx umfaßt derzeit sieben Größenordnungen, deren maximales Integrationsvolumen bei 1200-9000 Gatter (asynchron) bzw. 122-928 Flip-Flops und Latches (synchron) liegt. Die Anzahl der Ein- und Ausgänge pro Chip beträgt zwischen 58 und 144.
Dabei ist jeder Chip in einzelne Logikblöcke (Makrozellen) unterteilt (64..320), die kombinatorische als auch sequentielle Logik ermöglichen.

Weiterhin befindet sich ein Taktgenerator auf dem Chip, der eine Beschaltung durch einen handelsüblichen Quarz ermöglicht.
Kompatibel sind die Bausteine zu handelsüblicher TTL und CMOS Logik.

Lieferbar sind die Bausteine in drei Gehäuseformen:
-PLCC (plastic leadless chip carrier)
-PGA (pin grid array)
-Quad Flat Package

Wesentlich beim Einsatz dieser Bausteine ist, daß sie beim Anlegen der Versorgungsspannung von einem externen Speicher geladen werden müssen!

Dies kann durch ein einfaches Rom, oder auch durch ein Mikroprozessorsystem erfolgen, das die Initialisierungsphase des LCA übernimmt.

Auf den ersten Blick bietet diese Produktfamilie eine ungeahnte Vielzahl an Einsatzmöglichkeiten bzw. Integrationsvolumen.
Durch die angegebene maximale Toggle-Frequenz von bis zu 100MHz erscheinen die Bausteine auch von der Geschwindigkeit für unseren Anwendungsfall geeignet (Die in unserem Anwendungsfall auftretende Frequenz beträgt ca. 10,7MHz).
Dabei muß berücksichtigt werden, daß bei Anwendung kombinatorischer Logik, bei der Logikblöcke (im LCA) kaskadiert werden müssen mit zusätzlichen Laufzeiten innerhalb des Bausteins zu rechnen ist.
Trotzdem erscheint die Geschwindigkeitsreserve des Bausteins ausreichend zu sein.

Abschätzung des benötigten LCA-Typs

Eine erste Abschätzung über die erforderliche Größe des Bausteins erfolgt durch die Anzahl der benötigten Ein-bzw. Ausgänge:

	Ein/Ausgänge	Anzahl
* 8 Bit Daten	E	8
* Systemtakt	E	2
* Umschaltung Belichtung intern/extern	E	1
* Start Datenübertragung	E	1
* Steuerung CCD-Sensor	A	3
* Steuerung AD-Wandler	A	1
* Serieller Datenausgang	A	1
* Belichtungssteuerung intern/extern	A	2
* Hilfsausgänge		
Triggermöglichkeiten	A	2
Erweiterungen	E/A	3
Ein-bzw. Ausgänge		24

Die zweite Abschätzung erfolgt durch die Anzahl maximal möglicher Zustände bzw. benötigter Flip-Flops:

	Flip-Flops (ca.)
* CRC-Generator	16
* Par-Ser-Converter	10
* Controller	8
* Pixelcounter	12
Flip-Flops	46

Anhand dieser beiden Abschätzungen läßt sich schon eine Vorentscheidung treffen, welcher Baustein von seiner Größenordnung her geeignet ist.

Bezüglich der benötigten Ein-bzw. Ausgänge könnte man den kleinsten Baustein der Familie einsetzen (XC2064).

Er bietet 58 frei wählbare Anschlußpins. Jedoch besitzt er nur 64 Makrozellen, mit jeweils vier frei wählbaren Eingängen. Berücksichtigt man, daß mehr als nur vier Eingänge pro Block benötigt werden, so kann die logische Funktion einer Ausgangsbedingung nur durch Kaskadieren von Logik-

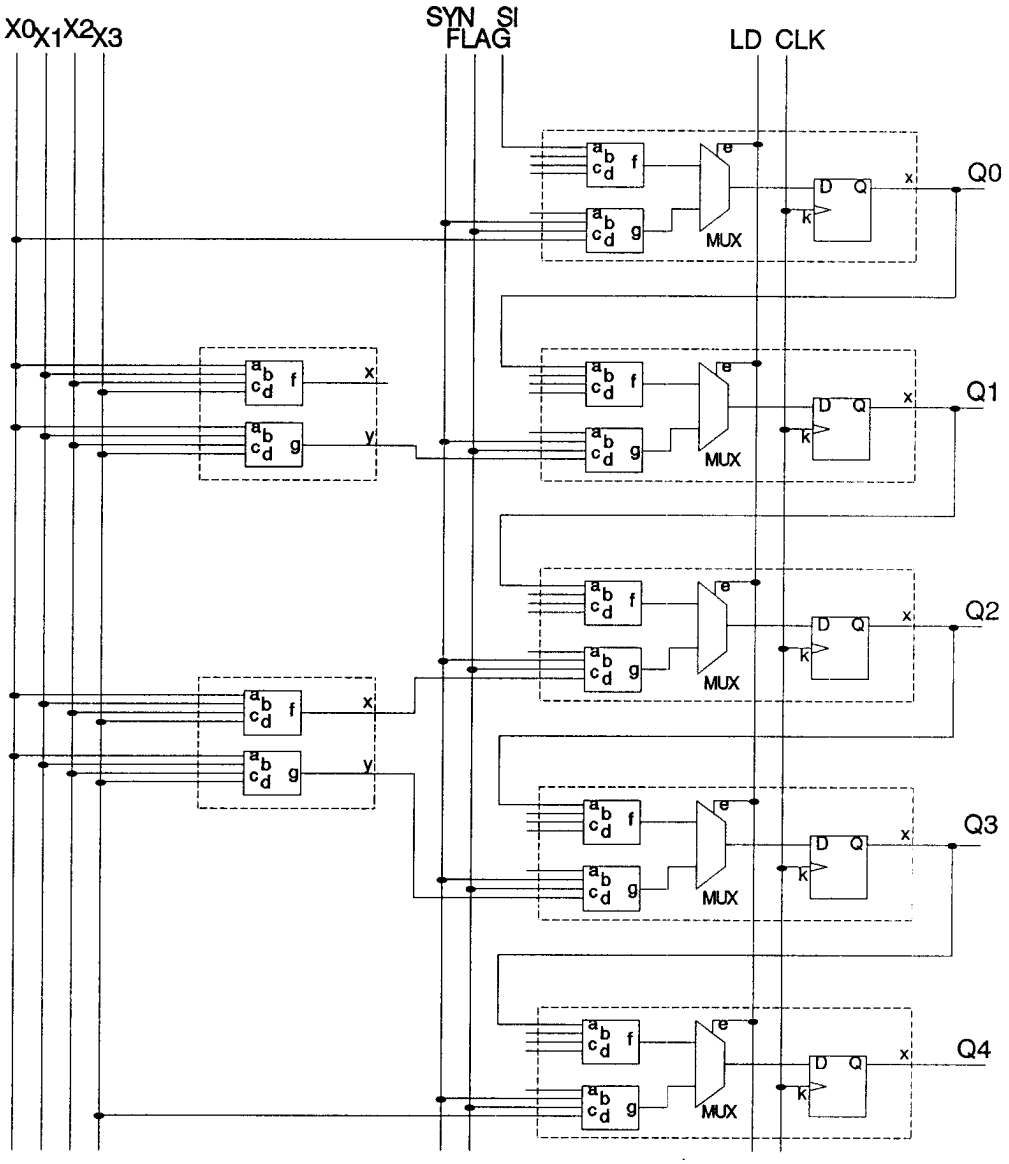

Bild 6: Makrozellenstruktur 4B/5B-Encoder + Shift-Register

blöcken erreicht werden. Damit stößt man schnell an die obere Leistungsgrenze dieses Bausteins.

Es erscheint damit sinnvoller, auf den nächst größeren Chip überzugehen. Dies ist der XC3020. Er bietet um den Faktor 1,66 mehr verfügbare Logik als der XC2064. Von den mechanischen Ab-messungen sind beide Typen gleich.

Die dritte Abschätzung erfolgt durch die benötigte Kombinatorik:

diese Abschätzung ist jedoch selbst bei Kenntnis der zu integrierenden Schaltung nur bedingt möglich. Beim Einsatz der LCA-Bau-steine wird diese dadurch erleichtert, indem man die gesamte Kombinatorik auf Einzelblöcke zurückführt, die der logischen Geometrie der Makrozellen innerhalb der LCA's entsprechen. Dadurch erkennt man recht schnell zusätzlich benötigte Makrozellen. Anschaulich ist dies am Beispiel des 4b/5b-Code-Converters mit anschließender Parallel-Seriell-Wandlung aufgezeigt (Bild6):

für den Aufbau des Converters werden neun Eingangssignale und fünf Ausgangssignale benötigt. Dies sind vier Datenleitungen (high oder low nibble des AD-Wandlers), vier Steuersignale und ein Taktsignal. Jede Makrozelle sollte also mit maximal neun Eingängen beschaltet werden können.
Die Anzahl der Eingänge ist jedoch durch den Aufbau der Makrozelle beschränkt.
Da wir aus Abschätzung eins und zwei den Bausteintyp schon festgelegt haben, läßt sich die Logik auf die Geometrie der zur Verfügung stehenden Makrozellen zurückführen. Diese Vorarbeit reduziert den späteren Aufwand bei der Eingabe der einzelnen Blöcke um einen wesentlichen Faktor.
Beim 4b/5b-Codewandler werden also, wie aus dem Schaubild ersichtlich, sieben Blöcke benötigt.

Bei der Umsetzung der eigenen Schaltung in die Blockstruktur des entsprechenden LCA ist auch zu berücksichtigen, daß nicht alle boolschen Kombinationen der verschiedenen physikalischen Eingänge der Makrozelle erlaubt sind. Aus diesem Grund ist alles, was über die grobe Blockgliederung hinausgeht unnötig.

Nachdem die Logik auf diese Art neu erfaßt ist, läßt sich eine konkrete Aussage darüber treffen, wieviele Blöcke benötigt werden, und wie stark der LCA etwa ausgelastet sein wird.

Beurteilung des Entwurfs

1) Wie schnell soll der LCA arbeiten

Mehr als 30% der angegebenen Toggle-Frequenz sollte hierbei nicht überschritten werden. Ausgenommen sind hierbei natürlich Schaltungen, die keinerlei Kaskadierung der einzelnen Makrozellen untereinander benötigen.

Ein weiterer Faktor ist die Blockausnutzung:
je weniger Blöcke genutzt werden, desto mehr Verbindungsmöglichkeiten stehen den anderen Blöcken untereinander zur Verfügung. Damit lassen sich Verbindungslängen auf ein Mindestmaß reduzieren.

In diesem Zusammenhang seien die Typen von Verbindungen erwähnt, die die LCA-Bausteine bereitstellen:

Verbindung / Bewertung

- **Direct interconnect**
  direkte Verbindung zwischen Blöcken
  keine Laufzeiten
  nur sehr beschränkt einsetzbar
  nur zwischen benachbarten Blöcken

- **General purpose interconnect**
  Laufzeiten
  vom Autorouter primär verwendet
  schlecht abschätzbarer Laufzeitfaktor
  Optimierung notwendig

- **Long Lines**
  geringe Laufzeiten
  Busstruktur möglich
  wired and, wired or möglich

Die Laufzeiten bei den Verbindungen entstehen dadurch, daß Einzelverbindungen mittels Transistoren verknüpft werden. Damit erhöht jede Verknüpfung zweier Einzelverbindungen die Laufzeit auf der entsprechenden Leitung.

Ein weiterer wesentlicher Geschwindigkeitsverlust entsteht durch die schon erwähnte, und in Bild 5 veranschaulichte Kaskadierung von mehreren Makrozellen. Dabei addieren sich die Einzeldurchlaufzeiten der jeweiligen Makrozellen. Hinzu kommt natürlich wieder die Laufzeit auf jeder Leitung.

2) Welcher Entwicklungsaufwand ist erforderlich

Grundsätzlich steht beim Eingeben der Schaltung in den LCA ein Autorouter zur Verfügung. Dieser verwendet jedoch hauptsächlich die General-Purpose-Verbindungen.

Damit entstehen Laufzeiten, die bei einer hohen Blockausnutzung nicht mehr abschätzbar sind. Es ist also bei zeitkritischen Anwendungen eine Optimierung von Hand durchzuführen. Diese wird umso zeitaufwendiger, je höher der Blocknutzungsgrad steigt.
Daher ist es auch sehr wichtig, die Anordnung der eigenen Logik innerhalb des LCA günstig zu wählen. Gesichtspunkte hierfür sind Anzahl benötigter Ein- bzw. Ausgänge innerhalb der zu integrierenden Logik.

Als Beispiel sei der Controller genannt. Er wird in unserer Anwendung in der physikalischen Mitte des LCA angeordnet. Dadurch kann er die zu steuernden anderen Logikbausteine auf kürzestem Weg erreichen.

Wie man an vorgenannter Problematik erkennt, steigt der Entwicklungsaufwand mit steigender Blockausnutzung beträchtlich an. Weiterhin sinkt dadurch die maximal mögliche Verarbeitungsgeschwindigkeit des LCA.
Es ist also insgesamt ein Kompromiß zwischen Integrationsdichte und Entwicklungsaufwand einzu-

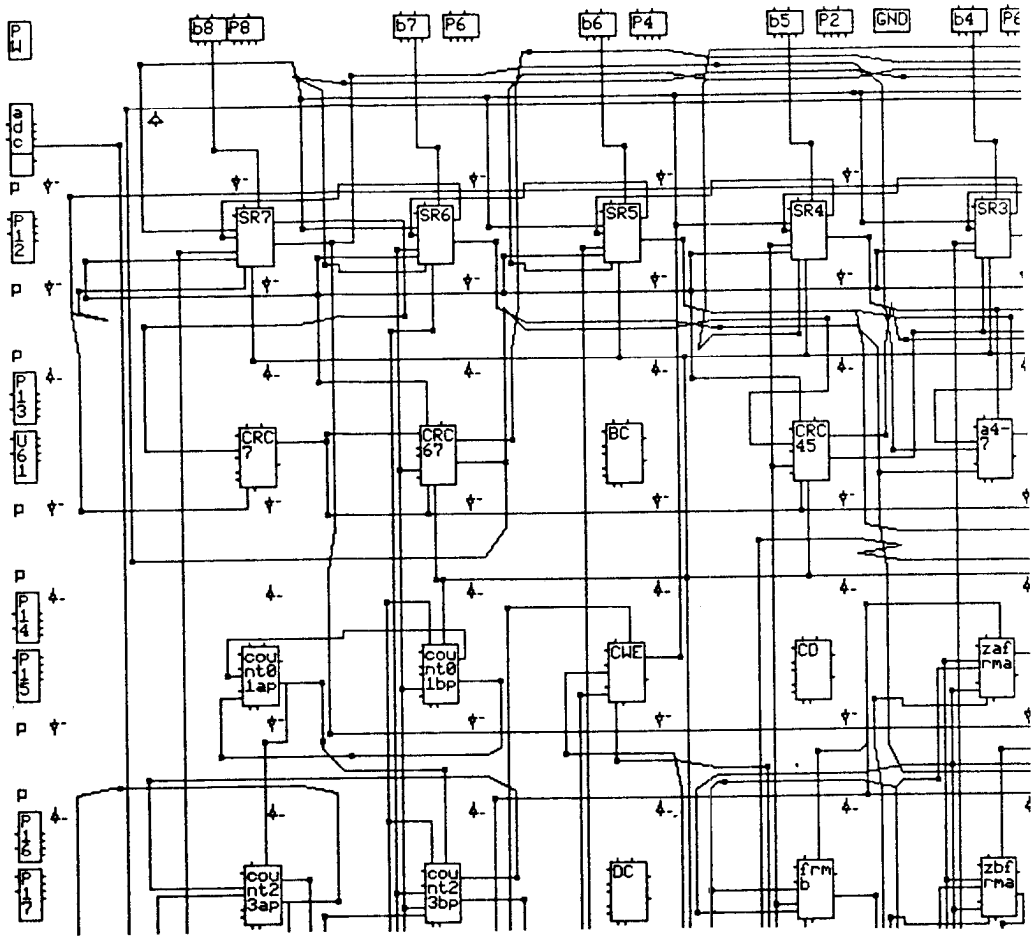

Bild 7: Ausschnitt aus der Struktur des LCA3020 (Xilinx)

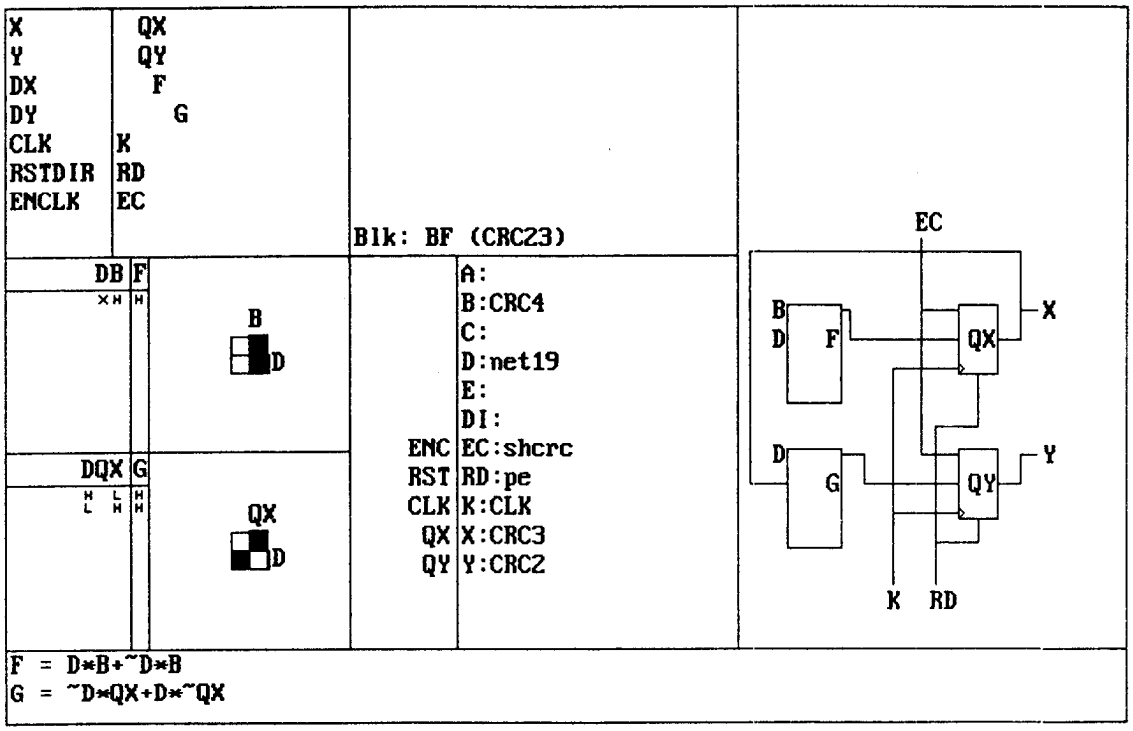

Bild 8: Programmierung einer Makrozelle (Beispiel)

gehen.

In unserer Anwendung wurde zum Verständnis der Programmierung der LCA-Architektur weitgehend auf den Autorouter verzichtet.
Dadurch wurde folgende Integrationsdichte möglich:

	Ausnutzung (XC3020)
Makrozellen:	ca. 90%
Flip-Flops:	ca. 35%
Ein-Ausgänge:	ca. 20%
Gatterausnutzung:	ca. 50%

Die Entwicklung des LCA erfolgte dabei anhand des dargestellten Schaubildes (Bild 9). Diese Vorgehensweise stellt natürlich nur einen Ausschnitt der Möglichkeiten dar, die beim Entwickeln eines LCA vorhanden sind.

Prinzipiell ist der Entwickler aber in den meisten Fällen gezwungen, in die Design-Ebene des LCA (LCA Design File) zu gehen, um Optimierungen bzw. Platzierung von Logik durchzuführen.

Beim Entwickeln des LCA werden außer der Design-Oberfläche noch mehrere Entwicklungstools bereitgestellt. Hierbei stellt die Simulation des LCA das wohl wichtigste Hilfsmittel dar, den Baustein auf seine Funktion zu prüfen.

Bei unserer Entwicklungsumgebung wurde das Simulationspaket SILOS eingesetzt.

Simulation/Test des LCA

Zur Simulation des LCA wird das LCA-Design-File auf MS-DOS-Ebene mittels zweier Konvertierungsprogramme umgesetzt. Dabei entsteht ein XNF-File (Xilinx Netlist File) das wiederum in eine Simulationsnetzliste für SILOS umgewandelt wird.

Es entstehen hierbei zwei Dateien. Eine enthält die komplette Beschreibung der Logik im LCA in boolscher Form (Extension .SIM). Sie ist soweit untergliedert, daß es möglich ist, auch Signale innerhalb einer Makrozelle zu testen.

Die andere Datei stellt eine Oberfläche dar, bei der man Ein-und Ausgänge des LCA beliebig logisch

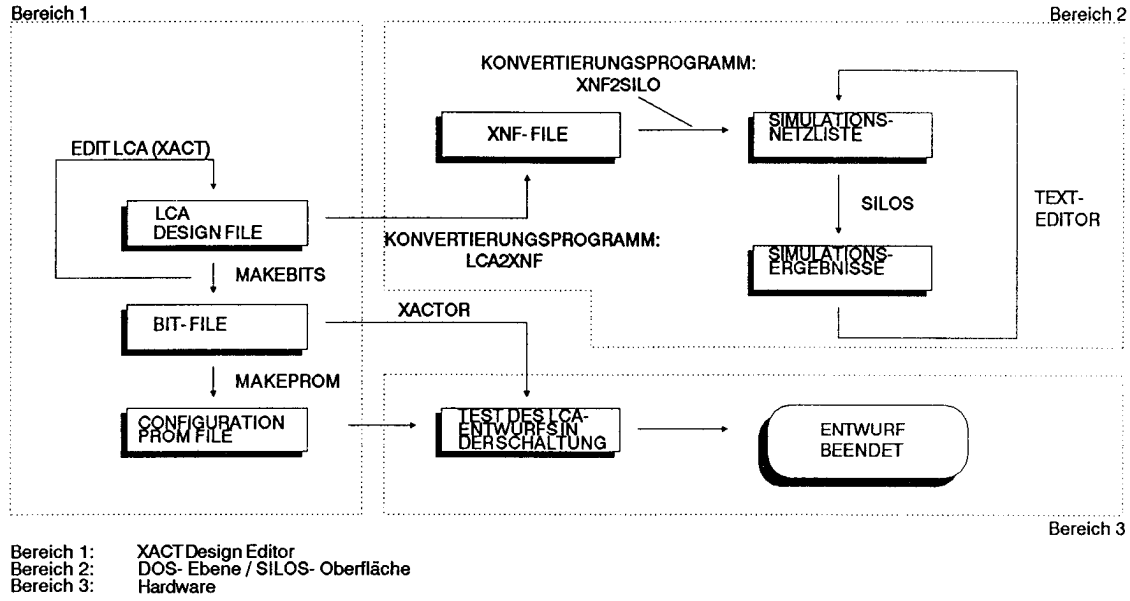

Bild 9: Schritte zur LCA Entwicklung

beschalten, und damit testen kann (Extension .DAT).

Es können nun beliebige logische Funktionen an die Eingänge des LCA gelegt werden (Mittels Editor; nicht interaktiv in der SILOS-Oberfläche).
Der Test des LCA erfolgt dann anhand numerischer bzw. grafischer Ausgabemöglichkeiten, wobei die Grafik sehr einfach gehalten ist. Sie bietet jedoch die Möglichkeit, logische Fehler sowie auch Spikes bzw. Glitches schnell zu erkennen.

Sollte ein Fehler innerhalb des LCA-Designs auftreten, so kann man in der Simulationsnetzliste mittels Editor Änderungen durchführen. Dies kann so lange erfolgen, bis der Entwurf einwandfrei ist.
Entsprechende Änderungen müssen dann noch im LCA-Design-File durchgeführt werden.

Xilinx verspricht mit der Simulation des LCA eine Worst-Case-Betrachtung. Das bedeutet also für den Anwender, daß Toleranzen der einzelnen echten logischen Signale innerhalb simulierter logischer Signale liegen.

Dies hat sich beim Einsatz des LCA in unserer Schaltung bestätigt. Damit ist die Entwicklung des LCA mit der erfolgreichen Simulation beendet.

Es ist nun möglich, den Chip direkt in seiner Hardwareumgebung zu testen. Dabei kann man den LCA vom PC mittels Download-Cable verbinden.
Ist die Funktion auch in diesem Fall einwandfrei, so kann das LCA-File in einem ROM (oder entsprechender Speichter) untergebracht werden.

Zusammenfassung

Insgesamt ist der Entwurf eines LCA auch bis in die Design-Oberfläche leicht verständlich, und bietet nach angemessener Einarbeitungszeit die Möglichkeit, schnell komplexe digitale Strukturen zu integrieren (Dies wird zusätzlich durch Makrofunktionen unterstützt).

Dabei ist der Entwickler losgelöst von jeglichen physikalischen Problemen, da bei den LCA's bereits vorgefertigte Makrozellen bestehen.

Inwieweit sich der LCA gegenüber äußeren Einflüssen verhält, muß sich noch im Laufe der Zeit herausstellen. Wenn man den Angaben von Xilinx vertraut, so sollen die RAM-Zellen des LCA besonders störsicher aufgebaut sein, und damit gegenüber äußeren Einflüssen resistent sein.

Literaturverzeichnis:

- The Programmable Gate Array Data Book
Xilinx

- Digital Communications with Fiber Optics and
Satellite Applications
Harold B. Killen
Prentice-Hall International Editions
ISBN 0-13-213091-2