Chip-Design – warum nicht asynchron?

Zahlreiche Vorteile bieten sich – Details zur NCL-Technologie

Synchrone Chip-Designs werfen oft Probleme auf: Störungen durch hohe Spannungsabfälle bei Schalt-Stromspitzen im IC sind hier nur ein Stichwort. Warum sollte man also manches nicht asynchron realisieren? – Die NCL-Technologie (Null Convention Logic) gibt hier gleich auch eine praktische Antwort.

Von Jürgen Pintaske

Beim Chip-Design werden aus der reinen Logik-Betrachtungsweise einer Boole'schen Algebra heraus weder die Laufzeiten in den Gattern noch die Signalverzögerungen in den Verbindungen des ICs berücksichtigt. Eventuelle Probleme müssen aber vorab über das Design, bei der Simulation oder später nach der Synthese in die Gatter oder nach der Platzierung auf dem Chip berücksichtigt werden. Bis zu einer gewissen Komplexität lässt sich dies recht gut planen, auch die Erfahrung hilft viel, aber bei der weiter steigenden Gatteranzahl in den heutigen Chips wird dies immer unübersichtlicher. Als weiteres Problem kommen bei komplexen SoCs (System-on-a-Chip-Designs) der Einsatz von bereits existierenden Funktionen oder von Intellectual Property (IP) von Drittanbietern als zugekaufte Funktionen hinzu, wobei die innere Funktionalität der IP wenig bekannt oder schwer nachzuvollziehen ist.

Problem der „worst case"-Designs: maximale Geschwindigkeit nicht möglich

Fast alle Designs werden im Augenblick in synchroner Technik entwickelt (*Bild 1a*). Es ergeben sich beim Einsatz der modernsten Halbleitertechnologien mit ihren kleinen Strukturbreiten auf dem Chip jedoch dabei einige Probleme, die es zu beherrschen gilt:

- Stromspitzen: Wenn z.B. alle vorhandenen Register zur gleichen Zeit (synchron) schalten, bilden sich erhebliche Stromspitzen in den Versorgungsleitungen. Die in diesen Leitungen dann entstehenden Spannungsabfälle können so groß sein, dass es zu Fehlfunktionen in der Schaltung kommt. Und dies kann zum Teil erst nach der Fertigung der ersten Muster-ICs evaluiert werden.
- Der Taktbaum auf dem Chip wird immer größer, länger und verzweigter (*Bild 1b*). Um die Taktflanken dabei „steil" zu halten, sind zusätzliche Taktpuffer auf dem IC notwendig, die die entsprechenden Kapazitäten der Verbindungsleitungen umladen müssen. Ein solcher Taktbaum ist immer aktiv und neben dem Reset-Pfad vermutlich der geometrisch längste Verbindungspfad auf dem Chip – und es ist der Verbindungspfad mit der maximalen Taktrate, selbst wenn der Chip nichts verarbeitet und „nur" auf neue Daten wartet.
- Jedes Design muss auf den schlimmsten Fall, den „worst case", abgestimmt werden: und zwar bezüglich niedrigst möglicher Betriebsspannung, höchster Temperatur und bezüglich der Fertigungsschwankungen des jeweiligen Halbleiterprozesses.

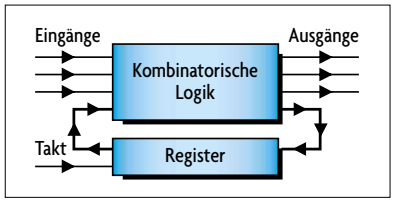


Bild 1a. Eine „State Machine" (FSM, Finite State Machine) arbeitet zeitdiskret, der Signalstatus wird in Speicherzellen gehalten.

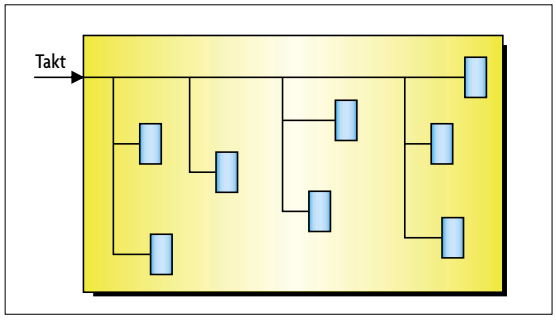


Bild 1b. Der lange Taktverteilungs-Baum auf modernen Chips verlangt meist eine Reduzierung der eigentlich maximal möglichen Geschwindigkeit.

In all diesen Fällen ist das Einbauen von „Sicherheitsbereichen" bei allen Parametern notwendig, die den Chip aber letztlich daran hindern, Daten mit maximal möglicher Geschwindigkeit zu bearbeiten und dann sofort in eine stromsparende Wartephase zu schalten. Auf einen 100-m-Lauf bezogen würde dies bedeuten, dass der Trainer die maximale und minimale Laufzeit

bestimmt und nicht der Läufer, auch wenn dieser schneller laufen könnte: Der Läufer bekommt die Laufzeit vorgegeben, und zwar einen Wert, der nicht an die physikalisch vielleicht mögliche Grenze heranreicht, um eben „Sicherheit“ einzubauen.

Die Auswege gibt es

Welche Auswege beim Chip-Design gibt es nun für diese Problematik? – Die Lösung liegt in der stärkeren Einbeziehung asynchroner Entwicklungstechniken. Tritt man sozusagen einmal von den Problemen der komplett synchronen Impementierungen zurück, sieht man, dass es verschiedene andere Techniken gibt, die zum Teil auch schon eingesetzt werden, wenn es sein muss (*Tabelle 1*). Und damit wird auch das weit verbreitete Missverständnis klar: Synchron ist gar kein Gegensatz zu asynchron, denn die Implementierung als komplett synchrones Design ist eigentlich ein Spezialfall der asynchronen Logik. Umgekehrt muss jedoch anerkannt werden, dass der schnelle Anstieg der Komplexität auf dem Chip nur durch die Beschränkung auf synchrone Designs möglich war.

Aspekt	Erklärung
Daten-Information steckt im Pegel-Potential	0 und 1 wie in normaler Logik. Der Übergang von Null auf Eins kann Probleme machen, je nach Anstieg der Flanke. Der Übergang stellt einen dritten logischen Pegel („unknown“) dar.
Daten-Information steckt in der Signalflanke	Erst nach dem Flankenende ist das Signal akzeptiert. Damit spielen Spannungspegel und Flankensteilheit keine Rolle mehr – außer durch die Verzögerung.
Daten bündeln	Eine Gruppe von Daten wird zusammen mit dem Taktsignal übertragen, um in etwa die gleiche Laufzeit zu erreichen.
Doppelte Signalführung	Low und High werden auf separaten Drähten als aktive Signale geführt. Nur das jeweilige High-Signal ist relevant.
Geschwindigkeitsunabhängig	Das System arbeitet unabhängig von der Geschwindigkeit der Gatter (die ist unterschiedlich wegen Fertigungs-Streuung, Betriebsspannungs- oder Temperaturänderungen). Die Empfindlichkeit gegenüber Verzögerungen in den Verbindungen bleibt.
Verzögerungsunempfindlich	Hier ist eine vollkommene Unabhängigkeit von allen Verzögerungen erreicht.

Tabelle 1. Aspekte bei der Implementierung asynchroner Systeme

Es geht also in der nächsten Zukunft darum, die Grenzen und Möglichkeiten der Tools zu erweitern. Eine Erweiterung der Boole’schen Algebra auf der Basis der NCL (Null Convention Logic) von Theseus ist hier ein wichtiger Baustein: Es wird weiterhin in den bekannten Hochsprachen wie VHDL gearbeitet; die Erweiterungen auf der Basis der NCL-Logik bearbeitet – vereinfacht gesagt – der Compiler.

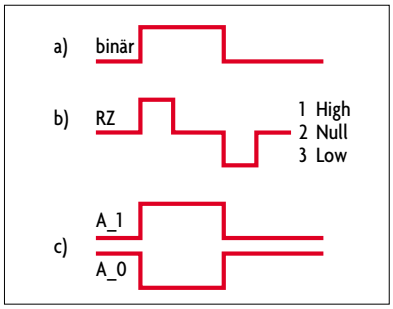


Bild 2. Bei der RZ-Codierung (Return-to-Zero, Kurvenzug b) ergibt die positive Flanke des Originalsignals (Kurve a) einen positiven Puls, die negative Flanke einen negativen Puls; Signal c zeigt die Dual-Rail-Codierung auf die zwei verschiedenen Datenleitungen.

Die Pluspunkte von NCL in Kürze

Aber wie funktionieren diese NCL-spezifischen Erweiterungen der herkömmlichen Logik? – Im Folgenden dazu in einer stark vereinfachten Weise die wichtigsten Details, wobei es weniger um Genauigkeit geht als um das Grundprinzip.

Als erstes gilt es, die Abhängigkeit des Systems vom Takt zu reduzieren oder ganz auszuschalten. Ohne Takt geht es zwar nicht, aber man kann ganz radikal dazu übergehen, ihn sozusagen in jedes einzelne Signal einzubetten. In der Signalübertragung ist dies nichts Neues. RZ (Return-to-Zero) – *Bild 2*, hier Kurvenverlauf b – zeigt eine Möglichkeit der Implementierung: Die positive Flanke des Originalsignals ergibt bei RZ einen positiven Puls, die negative Flanke einen negativen Puls, ansonsten keine Aktivität. Takt und Pegel können auf der Empfangsseite wiederhergestellt werden. Überträgt man diese Gedanken auf die Logik auf dem Chip, spielen Laufzeiten auf den Verbindungen keine Rolle mehr, und auch Temperatur- oder Betriebsspannungsänderungen sowie Fertigungsstreuungen haben keinen Einfluss auf die Funktion mehr, sondern lediglich auf die Durchsatzgeschwindigkeit. Bei „Aktivität“ eines Eingangssignales läuft dieses Signal so schnell wie möglich durch die ganze Schaltung bis zum Ausgang. Gibt es dann keine Aktivität mehr, also auch nicht über interne Rückführungen von Ausgangssignalen, wartet die Schaltung auf die nächste Aktivität. Es fließt dann nur der Ruhestrom.

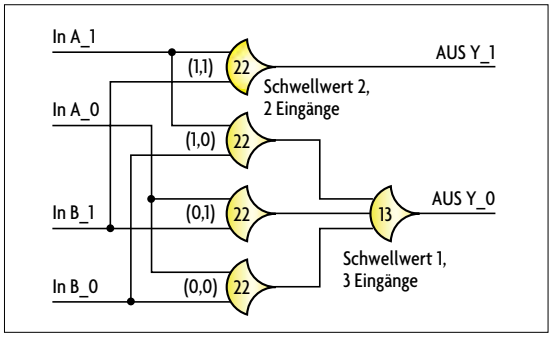


Bild 3. Logisches Verhalten eines NCL-Und-Gliedes.

NCL-Und	B TRUE	B FALSE	B Null
A TRUE	TRUE	FALSE	Previous
A FALSE	FALSE	FALSE	Previous
A Null	Previous	Previous	Null

NCL-Oder	B TRUE	B FALSE	B Null
A TRUE	TRUE	TRUE	Previous
A FALSE	TRUE	FALSE	Previous
A Null	Previous	Previous	Null

NCL-Nicht	Y
A TRUE	FALSE
A FALSE	TRUE
A Null	Null

Tabelle 2. Die logischen Grundfunktionen „Und“, „Oder“ und „Nicht“ in NCL realisiert

Man kann sich abstrakt jedes Signal wie einen Läufer auf der Leitung vorstellen, der von der Quelle bis zum nächsten Entscheidungspunkt, einem Gatter, läuft. Leider ergibt sich damit aber eine dreiwertige Logik, die eigentlich auch drei Signalpegel braucht (die es bei kommerziellen digitalen Halbleitertechnologien nicht gibt). Bei asynchroner Logik mit eingebettetem Takt muss damit eine andere Implementierung gewählt werden: Es wird nicht

Der NCL08GP32: ein 8-bit-Mikrocontroller mit NCL-Core

Der NCL08-Prozessor (*Bild A* und *Bild B*) ist instruktionskompatibel zum S08 von Motorola und damit auch rückwärtskompatibel mit den Prozessorfamilien 6805 und 6808. Der Core in NCL besteht aus dem NCL08, dem Speicherinterface, der Interrupteinheit und dem Businterface. Dieser Core wird mit den Controllerfunktionen, mit 32 K Flash, 4 K SRAM und dem lokalen Bus-Access-Modul zum „Supercore“ erweitert (Bild A). Dieser Block ist die Basis für weitere kundenspezifische Implemetierungen. Zusätzlich auf dem Chip: Tastatur-Interface, zwei 2-Kanal-Timer, ein UART und SPI-Interface sowie 33 bit programmierbare birektionale I/O-Kanäle. Das Gehäuse ist ein 44-Pin-QFP. Als wichtigste Unterschiede im Vergleich zu einer synchronen Implementierung ergeben sich 40 % geringere Leistungsaufnahme und 11 dB weniger Störungen auf den Betriebsspannungsleitungen. Neben den Chips bietet Theseus auch komplette Designkits und Evaluationboards an, die auch die entsprechenden Softwaretools enthalten.

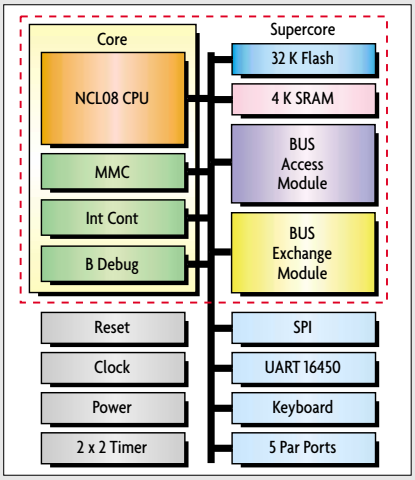


Bild A. Die Blockschaltung des Prozessors NCL08GP32.

Bild B. Chip-Layout des Prozessors.

mehr nur mit den Signalen 0 und 1 gearbeitet, sondern mit „Aktivitäten“. Der Pegel Null ist damit keine Aktivität mehr, und die bekannten Werte Low und High werden als „TRUE“ und „FALSE“ definiert; der inaktive Zustand wird als „Null“ bezeichnet.

Um diese drei Zustände zu beschreiben, benötigt man entweder drei Pegel auf der Leitung (z.B. 0 V, +2,5 V, +5 V), oder man verteilt die drei Zustände auf zwei Leitungen (Dual Rail, *Bild 2c*). Chiptechnologien sind für digitale Signale, also zwei Zustände optimiert, damit ist es einfacher, NCL als 2NCL zu implementieren, als über das 3-wertige 3NCL. Jedes Signal wird in einen separaten TRUE- und einen FALSE-Strang aufgeteilt; gibt es keine Aktivität, sind beide Leitungen Low, das ist der Null-Zustand. Beide Leitungen zur gleichen Zeit aktiv ist damit der nicht erlaubte vierte Zustand.

Eine Beschränkung der Implementierung auf zwei Leitungen pro Signal gibt es nicht. Eine interessante Lösung

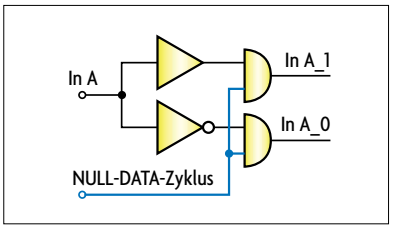


Bild 4. Umsetzer synchron nach asynchron: Aufspaltung des Signals in zwei Pfade und zusätzliche Und-Gatter zur Generierung des Null-Zustandes.

ergibt sich bei vier Leitungen. Im Gegensatz zu zwei Leitungen ergibt sich weniger Busaktivität, also auch weniger „dynamische Stromaufnahme“ (die ja Probleme bereiten kann).

Mit einer Dual-Rail-Implementierung ergibt sich natürlich ein anderes Verhalten der Grundgatter (*Tabelle 2*). Am besten lässt sich das Verhalten aber an einem Beispiel verstehen: ein Und-Gatter in Dual-Rail (*Bild 3*). Es ist nicht optimiert, damit man sich den Signalverlauf und die Zustände einfach vorstellen kann.

Zuerst muss das Eingangssignal auf zwei Leitungen aufgeteilt werden (*Bild 4*). Um auch den Null-Zustand zu implementieren, müssen beide Signale über Und-Gatter geführt werden. Da auch die beiden Dual-Rail-Ausgänge erzeugt werden müssen, gilt es beim Und-Glied auch zwei Signalwege zu implementieren: TRUE und FALSE.

Für TRUE (logisch 11) kann z.B. das Muller-C-Element (*Bild 5*) eingesetzt werden, eine asynchrone Und-Verknüpfung zweier Signale (kein asynchrones Und-Gatter in NCL!). Beide Eingänge Low ergibt am Ausgang Low. Beide Eingänge High ergibt am Ausgang High. Der Unterschied zur herkömmlichen Logik ergibt sich durch die Rückführung: Erst wenn beide Signale angekommen sind, wird geschaltet (Laufzeitunabhängigkeit von der Signalankunft). Nach der Änderung eines Signals auf Low bleibt der Ausgang High noch erhalten, bis auch der zweite Eingang Low ist; es ergibt sich eine Schwellwertfunktion, etwa vergleichbar dem Schmitt-Trigger bei analogen Schaltungen.

Etwas komplexer wird es, den Ausgang beim Und-Gatter für FALSE zu erzeugen: Es gilt, drei FALSE-Möglichkeiten abzudecken (Tabelle 2). FALSE bedeutet 00 oder 10 oder 01, also drei Muller-C-Elemente, die den jeweiligen Zustand anzeigen, und ein 1-aus-3-Schwellwertgatter, das das endgültige Ausgangssignal erzeugt. Wegen der dreiwertigen Logik reicht hier ein Inverter am Ausgang nicht aus.

Die Oder-Funktion lässt sich in analoger Weise nach dem DeMorgan-Theorem aufbauen: A OR B = NOT (NOT A AND NOT B). Der Inverter ist übrigens in NCL extrem preiswert: Es werden einfach die beiden Rails vertauscht.

Nach den Grundgattern fehlt noch das Speicherelement. Bei NCL wird hier mit Anfrage und Quittierung gearbeitet (*Bild 6*). Ist das System in Ruhephase, dann ist sozusagen der Eingang offen, und es können Dual-Rail-Signale ankommen. Auch hier passt das Muller-C-Element: Ein Eingang wird als Signaleingang eingesetzt, ein Eingang als Enable. Am Ausgang wird zusätzlich der Empfang von Daten beobachtet und an den vorhergehenden Block gemeldet.

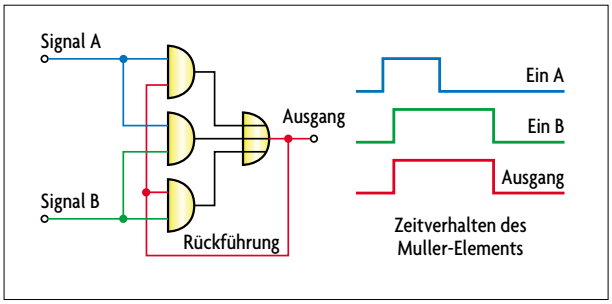
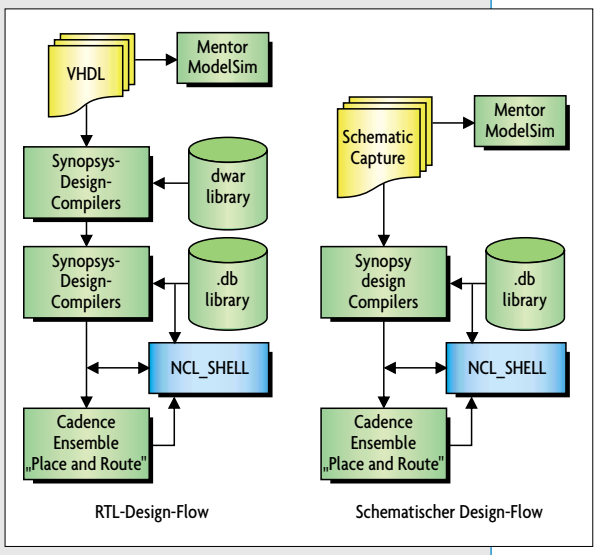


Bild 5. Das Muller-C-Element ist eine asynchrone Und-Verknüpfung zweier Signale: Nach der Änderung eines Signals von High auf Low (Signal A) bleibt der Ausgang High noch erhalten, bis auch der zweite Eingang (Signal B) wieder auf Low ist. So ergibt sich eine Schwellwertfunktion, etwa vergleichbar dem Schmitt-Trigger bei analogen Schaltungen.

Der NCL-Design-Flow auf der Basis kommerzieller Tools

Beim NCL-Design-Flow (*Bild*) wurde besonders darauf geachtet, dass keine speziellen und proprietären Werkzeuge eingesetzt werden müssen, sondern dass der Entwicklungsablauf bei dieser Technologie dem bekannten Ablauf so ähnlich wie möglich ist. Ausgegangen wird von einer Schaltungsbeschreibung in VHDL. Die Simulation erfolgt mit Mentor Modelsim, die Synthese mit dem Synopsys Designcompiler und die Platzierung mit Cadence Ensemble.



Design-Flow, basierend auf Standard-Werkzeugen.

Modul	synchron	NCL	Verhältnis
AddrConv	41	62	1,5
X2VHD	395	826	2,1
Decoder	1010	1804	1,8
32x16FIFO	1691	1123	0,7

Tabelle 3. Vergleich der Transistor-Anzahl bei synchronem und NCL-Design, entweder als synchrone Implementierung oder als Redesign in NCL.

Ein wichtiger Punkt von NCL muss jetzt noch erwähnt werden: Eigentlich ist ja immer eines der beiden Dual-Rail-Signale aktiv. Wie kommt man dann in die Null-Phase, um die Schwellwertgatter wieder in die Ausgangsphase zu setzen? – Dies wird sequentiell realisiert. Das gesamte System schaltet dauernd zwischen einer NULL-Phase und einer DATA-Phase. Auch hier passt das Beispiel der 100-m-Läufer. NULL bedeutet: Läufer gehen auf die Plätze (Bahn frei), DATA bedeutet: Sie treffen nacheinander am Ziel ein, danach wieder dieselben beiden Phasen mit anderen Läufern.

Doch einige wesentliche Pluspunkte

Was bringt nun NCL letztlich? – Wie man sieht, wird der Taktbaum auf dem Chip eingespart, und allein dadurch ergibt sich eine ganze Reihe von Vorteilen. Umgekehrt wird aber für die Implementierung von NCL durch die komplexeren Gatter zusätzliche Chipfläche verbraucht. Es sind wohl bei einer Entscheidung auch andere Aspekte zu berücksichtigen: extrem geringe Stromaufnahme, einfachere Integration von bestehenden synchronen IP-Blöcken, weniger Störstrahlung – besonders wichtig bei Smartcards wegen des Schutzes und bei den Mixed-Signal-Designs wegen der automatisch besseren Leistung der Analogteile durch weniger Störimpulse.

Funktion	Manuell	Synthese	% Synthese gegenüber manuell
MUX	40	40	100
AND4	66	68	103
Test7	140	126	90
CLIPPER	339	212	63
Set_Cnt	238	208	87
AND16	352	342	99
SHIFT	506	248	56
CASE	594	482	81
Synch-State	1008	814	81
Bit_Cnt	1059	794	75

Tabelle 4. Vergleich der Transistorzahlen bei manuellem Design und Synthese mit dem Design-Compiler, basierend auf der LSI-LCB500k-Bibliothek

Vorteile beim Einsatz von NCL in verschiedenen Anwendungen

- Smartcard: Erhöhte Sicherheit gegen Entschlüsseln der Daten. Smartcards enthalten kartenspezifische Daten im Speicher, die als Zugriffsdaten benötigt werden. Um diese Daten zu entschlüsseln, wird versucht, durch „Abhören" des Stromverbrauches und der Störstrahlung auf die jeweilige Aktivität und die Daten zu schließen. Bei synchronem Design des IC ist dies zwar ein komplexer Vorgang, aber möglich, da zum synchronen Schaltzeitpunkt im IC je nach Funktion bestimmte Strom- und Störspektren entstehen, von denen auf die Daten geschlossen werden kann. Beim Einsatz von NCL gibt es keinen Takt, und damit ergibt sich beim Strom ein fast reines Zufallsrauschen und damit auch eine viel geringere Störstrahlung, was zu höherer Sicherheit führt.
Ein Beispiel ist die Implementierung des DES-Algorithmus beim NCL-Lizenznehmer Infineon. Umzusetzen in NCL sind dabei etwa 5000 Zeilen VHDL: 64 bit Datenblocklänge, 56 bit Schlüssel, Implementierung in einem 0,13-µm-CMOS-Prozess. Der Chip ist in der Evaluierungsphase.
- Sensoren mit geringer Stromaufnahme: Minimierung der Stromaufnahme ist hier das Ziel, z.B. für den Einsatz in Hörgeräten. Durch die asynchrone Logik wird zum einen der Stromverbrauch im Vergleich zu existierenden Designs weiter reduziert; durch die fehlenden Schaltspitzen ergibt sich zusätzlich eine erhöhte Leistung der vier analogen Blöcke Mikrofonverstärker, A/D, D/A und Hörertreiber.
- ICs mit analogen und digitalen Funktionen: Funktionsverbesserungen ergeben sich durch das Quasi-Zufallsrauschen bei der NCL-Implementierung: Generell stören die digitalen Teile auf dem IC die Analogblöcke, da Schaltspitzen über die Betriebsspannungsleitungen Einfluss auf die Analogteile nehmen können. Bei gegebenem Layout verbessert sich bei der Umsetzung des Digitalteils in NCL automatisch der Störabstand. Beim Vergleich eines Mikroprozessors in NCL waren die Spitzen bei der Störstrahlung um 11 dB kleiner im Vergleich zur synchronen Implementierung.
- Empfindlichkeitsverbesserungen bei Wireless-ICs: Hier ergeben sich weniger Störungen durch die digitale On-Chip-Logik. NCL verbessert indirekt das Verhalten bei den Verstärkern, in den Mischstufen und bei den VCOs. Eine Applikation wäre zum Beispiel die drahtlose Übertragung per Bluetooth vom Handy zur Hör-Sprech-Einheit: Jede Verringerung der Stromaufnahme ermöglicht kleinere und leichtere Batterien und ein einfacheres Design der analogen HF-Bausteine.
- Interpolierende D/A-Wandler (IDACs): Diese Produkte werden in hohen Stückzahlen in Basisstationen für mobile Telefone eingesetzt. Die wichtigsten Blöcke sind FIR-Filter und damit Multiplizierer. Auch wenn die Daten in synchroner Weise ankommen, ergeben sich über die asynchrone Weiterverarbeitung reduzierte Störpegel.

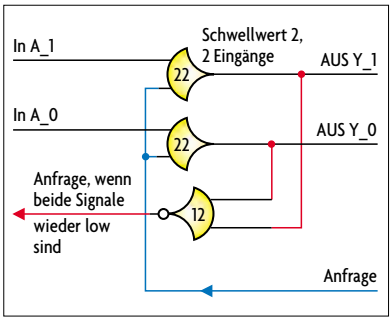


Bild 6. Ein NCL-Register mit Abfrageleitungen für neue Daten über die Rückführung.

Auch eine teilweise Implementierung eines SoC in NCL bringt Vorteile an kritischen Stellen oder als Verbindungsebene zwischen synchronen Blöcken. Wichtig ist, dass die Frage „Synchron wie bisher oder asynchron mit NCL?“ eigentlich nicht richtig ist; besser ist wohl die Überlegung, in welchen Applikationen der Einsatz der NCL-Technologie beim ersten und bei weiteren Designs mit denselben Blöcken so viele Vorteile bringt bei der Implementierung, bei den Einsparungen an Designzeit oder bei der Reduzierung der Re-Designs bis zum fertigungsreifen Produkt, dass eine Entscheidung für den Einsatz wohl mehr eine politische als eine kommerzielle Frage ist. In diesem Zusammenhang zeigen die beiden *Tabellen 3* und *4* abschließend noch die verschiedenen Transistor-Anzahlen bei den unterschiedlichen Implementierungsformen – auch hier ergeben sich interessante Werte. *ha*

Informationsquellen zur NCL-Technologie

[1] www.Theseus.com
[2] Synthese: www.synopsys.com
[3] Platzierung: www.cadence.com
[4] NCL-Prozessor: www.theseus.com/FramesProducts.htm
[5] Simulierte Beispiele: www.exemark.com/ASYNCH

Dipl.-Ing. Jürgen Pintaske

studierte Nachrichtentechnik in Aachen und arbeitete dann als Entwicklungsingenieur bei Horstmann in Heiligenhaus im Bereich Analog- und Prozessorsysteme. Daran schlossen sich fünf Jahre Applikationsingenieur bei RCA/Harris an. Es folgten Marketingmanager-Funktionen bei Hitachi und Tekelec sowie Vertrieb bei Hughes Microelectronics und Mixed Mode in München. Seit 1998 in Exeter/England lebend, unterstützt er nach seiner Funktion als Director Germany bei Marketbroad jetzt bei ExMark amerikanische und europäische Kunden in PR, Marketing und Sales.

E-Mail: exmarkjpi@aol.com